COPENHAGEN BUSINESS SCHOOL

M.Sc. Finance and Investments

Master's Thesis

Option Pricing Using Artificial Neural Networks

No. of Pages: 80

May 15, 2019

Author

Mads Højer Petersen (92279)

Supervisor

Gyuri Venter

Abstract

This thesis examines an artificial neural network option pricing model with an emphasis on its ability to capture the volatility surface and the effect of supply and demand of options on option prices. I compare the empirical pricing errors of the artificial neural network model to the Black-Scholes-Merton model and the Practitioner Black-Scholes model on S&P 500 and Dow Jones Industrial Average index options for the period 2010 to 2017. I find that the lowest pricing errors occur when the VIX (VXD) volatility index is used as volatility parameter in the artificial neural network model. Further, I find that the artificial neural model prevails both of the comparison models across option moneyness and time-to-maturity implying that the ANN model captures the volatility surface better than the comparison models. Finally, I find that the model's pricing error is reduced further by including the proportional bid-ask spread as a proxy for the liquidity premium resulting from an imbalance between supply and demand, and the slope of the yield curve as a proxy the perceived recession risk by the market. I attribute the improvement of the artificial neural network model to its ability to learn the dynamics of option markets without imposing restrictive assumptions like those of the Black-Scholes-Merton model. However, I recognize that the artificial neural network model suffers from opaqueness which may render the model impractical to some users.

Table of Contents

Chapter 1

Introduction

1.1 Background & Motivation

The first recorded option contract dates back to the 4th-century BC (Poitras, 2008). Aristotle describes how the philosopher Thales entered into contracts with owners of olive presses that gave him the right, but not the obligation, to use olive presses during harvest time. While the concept of option contracts is old, it took more than 2,300 years before a well-regarded method for valuing these contracts was discovered. In 1973, Black and Scholes (1973) in collaboration with Merton (1973) proposed the Black-Scholes equation, a partial differential equation, that they claimed all derivative securities had to satisfy, as well as closed-form solutions to this equation for European call and put options.

The most prominent innovation of Black, Scholes, and Merton was not the actual option pricing formula, but the notion of dynamic hedging/replication of options. This notion gave birth to the framework that is used to price all derivative securities today. While the Black-Scholes-Merton model was an important breakthrough in option pricing and without doubt changed the financial markets forever, the model suffers from restrictive and unrealistic assumptions that researchers have tried to alleviate through a range of extensions such as

using deterministic volatility functions (Dumas, Fleming, & Whaley, 1998) as opposed to the constant volatility assumption of the Black-Scholes-Merton model.

Another less widespread method of option pricing involves the use of machine learning. The first such attempt was by Hutchinson, Lo, and Poggio (1994), who explored the validity of machine learning in option pricing by testing four different machine learning algorithms on simulated option prices as well as observed option prices on the S&P 500 index. Their results were promising from an academic perspective, but neither the technology, the data availability, nor the machine learning research was sufficiently developed for their model to be accurate enough to be of practical use. However, with the exponential growth in computing power and data availability, and the significantly increased attention machine learning has received in the past twenty years, it is time to reassess option pricing using machine learning techniques.

The primary proposed advantage of machine learning algorithms over traditional option pricing methods is their non-parametric nature. Instead of relying on simplifying assumptions about the underlying asset dynamics, machine learning algorithms are assumption-free. The algorithms work by identifying patterns and relationships in data, and they use this information to predict option prices. In this way, machine learning algorithms have the potential to identify complex relationships in data without any restrictive assumptions.

Machine learning is already applied to a myriad of problems within all areas of scientific research including financial theory. I find the potential for knowledge creation through a combination of scientific fields particularly interesting. Machine learning is a relatively new field of study, and the possible uses are far from exploited yet. I hope to be able to contribute to our understanding of how machine learning can be used in a financial context with this thesis.

1.2 Problem Formulation

This thesis aims to answer the following research question:

Can Artificial Neural Networks Price Exchange-Traded Options Better Than Traditional Methods?

In the search for an answer to this question, I will further study the following questions:

1. Which is the optimal volatility forecasting model for the artificial neural network option pricing model?
2. Are artificial neural networks able to capture the well-known volatility surface?
3. Can additional variables in the artificial neural network option pricing model capture the effect of supply and demand on option pricing?

1.3 Limitations

There are endless possible specifications of artificial neural networks and even more additional variables that have the potential to influence the accuracy of the artificial neural network models. Even though more optimal specifications of the artificial neural network model likely exist, the scope of this thesis is limited to the multilayer perceptron artificial neural networks. Also, the additional variables examined are limited to four variables that attempt to capture the effect of supply and demand of options on their price as well as option liquidity.

The scope of this thesis is limited to the pricing of European call index options on the S&P 500 and the Dow Jones Industrial Average during the period 2010 to 2017. I will focus the analysis on the S&P 500 index options whenever it is redundant and information overload to discuss the results of both the S&P 500 and the Dow Jones Industrial Average index options. In addition, I assume that the put-call parity holds meaning the price of put options can easily

be determined based on the price of call options of the same strike price and time-to-maturity. I explain the put-call parity in more detail in section 2.1.

The comparison models are limited to the Black-Scholes-Merton model and the Practitioner Black-Scholes model. While a myriad of extensions to the Black-Scholes-Merton model exist, I have chosen to focus on the performance of the artificial neural network model in comparison to only these two well-known and popular option pricing models to keep the analysis focused. I define these models as the traditional option pricing models mentioned in the Research Question in section 1.2.

1.4 Literature Review

In this section, I present a brief overview of the most noticeable existing literature on the topic of option pricing using artificial neural networks in order to motivate the purpose and addition of this thesis in relation to existing literature.

The earliest attempt at pricing options using non-parametric machine learning algorithms is by Hutchinson et al. (1994). They investigate if any of four non-parametric machine learning methods, including an artificial neural network, can price and hedge S&P 500 futures index options from 1987 to 1991. They describe their results utilizing the R^2 from a linear regression of the predicted option price against the observed market price. The artificial neural network predictions result in an out-of-sample R^2 of 95.53 percent, thus establishing artificial neural networks as a feasible option pricing method.

Despite having more than twenty years of history, research in the cross-field between machine learning and option pricing is still relevant today due primarily to advances in computational power, increases in data availability, accessibility of machine learning tools and progress in machine learning research.

Amilon (2003) uses a similar approach to Hutchinson et al. (1994) but has the advantage of nine years of developments in computing power, increases in data availability and further

progress in machine learning research. Concretely, Amilon (2003) prices European call options on the Swedish stock market index using a multilayer perceptron artificial neural network during the period 1997 to 1999. The results are however measured differently from Hutchinson et al. (1994) making comparison difficult. Amilon (2003) presents an out-of-sample mean squared error of 5.57 for the bid price and 6.40 for the ask price.

Note, that the unit of the mean squared error is comparable across currencies, because the price of the underlying asset and the call price are both divided by the strike price resulting in the mean squared error being normalized to the same scale despite the papers pricing options in different currencies. See section 4.2 for a discussion of this normalization method.

Park, Kim, and Lee (2014) compares a range of parametric and non-parametric option pricing models on approximately 21,000 KOSPI 200 index call options from 2001 to 2010. They find significant outperformance of non-parametric methods over the Black-Scholes-Merton model. One of these models is again the multilayer perceptron artificial neural network. For this model, Park et al. (2014) present out-of-sample mean squared errors between 2.47 and 2.87 for in-the-money options, between 0.82 and 1.30 for at-the-money options, and between 0.48 and 1.39 for out-of-the-money options, thus beating the model of Amilon (2003). The study from Park et al. (2014) was released eleven years after the one of Amilon (2003), and it is therefore not surprising that the model of Park et al. (2014) prevails the one of Amilon (2003) again due to the beforementioned reasons.

Whereas the studies of Hutchinson et al. (1994), Amilon (2003) and Park et al. (2014) use similar methodologies, other researchers have tried to identify improvements to basic multilayer perceptron artificial neural network. One of these is Wang (2009), who investigate whether the integration of different volatility forecast models into the artificial neural network improves the results. The data used by Wang (2009) consists of approximately 21,000 call options on the Taiwanese stock market index from 2005 to 2006. The volatility forecasting models used are the GARCH(1,1), the GJR-GARCH, and the Grey-GJR-GARCH. Wang finds that the Grey-GJR-GARCH volatility forecast results in the best predictions with out-of-sample mean

absolute percentage errors of 12% for in-the-money options, 359% for at-the-money options and 1833% for out-of-the-money options.

The final study of interest concerning this thesis is Gradojevic, Gençay, and Kukolj (2009). Gradojevic et al. (2009) discovered a weakness in artificial neural network's ability to price out-of-the-money options and suggested to use a Modular Neural Network trained separately on nine modules consisting of different option moneyness and time-to-maturity. They use S&P 500 index call options from 1987 to 1994. The out-of-sample mean absolute percentage error of the modular neural network is presented year-by-year and is between 3.01% and 27.67%, while the out-of-sample mean squared error is between 0.02 and 3.18.

This brief overview of the existing literature illustrates many similarities between methodologies with some differentiating features. This thesis takes a similar approach to the existing literature, but some important distinctions apply. Firstly, the size of the S&P500 index data is larger with approximately 64,000 call options and a total of roughly 2.4 million option prices, while the Dow Jones Industrial Average has approximately 9,000 call options with nearly 430,000 option prices. Secondly, forecasted realized volatility and the VIX/VXD implied volatility indices are investigated as volatility forecasting models. However, the most significant novelty of this thesis is the integration of additional variables in order to capture the effect of supply and demand on option prices.

The papers mentioned in this section are not exhaustive, but the literature on this topic is rather homogeneous with only small differences in configurations of artificial neural networks as well as different time periods and underlying assets. The reader is referred to the following additional literature on the topic: Lajbcygier and Connor (1997), Gençay and Qi (2001), Yang and Lee (2011), and Liang, Zhang, Xiao, and Chen (2009).

1.5 Structure

To comprehensively answer the research question, I have structured the thesis into five chapters followed by a conclusion. I will briefly provide an overview of each chapter next.

The primary aim of Chapter 2 is to derive the Black-Scholes-Merton model and discuss the restrictive assumptions underlying this model. In addition to this, I derive the Practitioner Black-Scholes model as a second comparison model to the artificial neural network model, and I introduce the notion of supply and demand in empirical option pricing.

In Chapter 3, I introduce the theory of the artificial neural network used in this thesis and discuss the primary properties of artificial neural networks that are beneficial in relation to option pricing. In addition, I introduce the bias-variance tradeoff as a framework for testing the generalizability of the model.

In Chapter 4, I describe the methodology of this thesis, concretely the data, the volatility forecasting models, the evaluation criteria, the optimization of hyperparameters of the artificial neural network model, the Practitioner Black-Scholes model, and the additional option pricing variables.

I describe the empirical results of the artificial neural network model and the comparison models in Chapter 5. I examine the ability ofartificial neural networks to capture the volatility surface as well as the effects of supply and demand on option pricing. I conclude the chapter with a discussion of the results.

Chapter 2

Option Pricing Theory

The primary aim of this chapter is to derive the Black-Scholes equation and its analytical solutions to European options as well as the Practitioner Black-Scholes model since I use these models as benchmarks for comparison to the artificial neural network model. Derivations and formulas in this chapter are based on Hull (2018) unless stated otherwise in the text.

The chapter is structured in the following manner: Initially, I introduce basic characteristics of options and the hedging argument in discrete-time. Next, I introduce relevant stochastic processes and derive Itô's lemma from Itô's calculus. I then use Itô's lemma to derive the Black-Scholes partial differential equation, and I present analytical solutions to the equation for European call and put options. Finally, I discuss the assumptions underlying the Black-Scholes-Merton model, and I introduce the notion of supply and demand affecting on option pricing.

2.1 Introduction to Options

A derivative security (such as an option) is a financial instrument whose value deriveswo from another asset called the underlying asset. Options are bilateral contracts that give the holder the right, but not the obligation, to buy or sell the underlying asset at a specified price and

date. Options can have different exercise styles, whereas this thesis is limited to European options that can only be exercised at maturity of the option.

European options can take two forms: call options and put options. Call (put) options give the owner the right to buy (sell) a number of shares at a predetermined price, called the strike price, at maturity of the option contract. If the price of the underlying asset at option maturity is above the strike price, the call option is considered in-the-money (ITM) and the option holder can buy the underlying asset at a discount and receive a positive payoff. Vice versa, the option is said to be out-of-the-money (OTM) if the underlying asset price at option maturity is below the strike price. In this case, the option holder would not exercise the option, since this would lead to a negative payoff. Instead, the option contract would expire worthlessly, and the option holder would lose the premium paid for the option. The opposite is true for put options that benefit from a price decline in the underlying asset.

The payoff functions of European call and put options can be written as

$$c(S,T) = \max(S_T - K, 0) \tag{2.1a}$$

$$p(S,T) = \max(K - S_T, 0) \tag{2.1b}$$

where S_T is the price of the underlying asset at maturity T and K is the strike price.

While the focus of this thesis is on call options only, the results can be extended to include put options using the so-called put-call parity. Concretely, the put-call parity describes the relationship between put and call options of the same strike price and time-to-maturity. The put-call parity states that

$$p + Se^{-qT} = c + Ke^{-rT} \tag{2.2}$$

where q is the dividend yield, and r is the risk-free rate.

It is apparent from (2.2) that the price of a put option is a function of a call option, the strike price and time-to-maturity of both options, the dividend yield on the underlying asset and the risk-free interest rate: all variables given exogenously.

The payoffs of both call and put options can never become negative since the option holder would choose not to exercise the option in such cases. With a strictly positive expected payoff, there must be a price to gain access to the option payoff function. Estimating this price using the cross-section of financial theory and machine learning is the focus of this thesis.

In 1973, Fischer Black and Myron Scholes published the seminal paper 'The Pricing of Options and Corporate Liabilities'. The option pricing model proposed by Black and Scholes (1973) revolutionized the financial industry by building a common framework for option pricing. They introduced the notion of dynamic replication and hedging of options and the resulting no-arbitrage option pricing argument. During the same year, the Chicago Board Options Exchange (Cboe) was founded as a marketplace for trading listed options as an alternative to over-the-counter options. These two events can be largely accredited as the catalysts of the enormous derivative markets today.

2.2 Hedging Argument

The theoretical price of an option is determined based on the principle of absence of arbitrage, namely that arbitrage opportunities are assumed to be non-existent. The argument is that arbitrage opportunities would attract arbitrageurs, who would place positions to capture the riskless profit, causing the market price to converge towards the arbitrage-free price. In this section, I show how to hedge a derivative using positions in the underlying stock. When this hedge is perfect, the price of hedging has to be equal to the price of the derivative, or an arbitrageur could capture the spread without taking any risk by dynamically hedging the option.

Assume the price of an asset, S_0, can be either S_0u or S_0d one period from today, while a derivative on the asset, f_0, can take on the values, f_u or f_d. Here, u and d denote two constants that represent the possible changes to S_0 one period ahead. The aim is to determine the position in the underlying that perfectly hedges the short sale of the derivative.

At time $t = 0$, sell the derivative and go long Δ of the underlying asset. This results in a cash flow at time $t = 0$ of $f_0 - \Delta S_0$. The value of this portfolio can then take on one of two possible values at time $t = 1$: $-f_u + \Delta S_0 u$ or $-f_d + \Delta S_0 d$. To achieve a perfect hedge, the value of Δ must be determined such that the two possible cash flows one period ahead are identical

$$-f_u + \Delta S_0 u = -f_d + \Delta S_0 d \tag{2.3a}$$

$$\Delta = \frac{f_u - f_d}{S_0 (u - d)} \tag{2.3b}$$

The return on the portfolio from time t to $t + \Delta t$ must then be equal to the risk-free rate, since all risk is eliminated

$$(-f_0 + \Delta S_0) e^{r\Delta t} = (-f_u + \Delta S_0 u) = (-f_d + \Delta S_0 d) \tag{2.4a}$$

$$f_0 = (p f_u + (1 - p) f_d)\, e^{-r\Delta t}, \tag{2.4b}$$

$$\text{where p} \equiv \frac{e^{r\Delta t} - d}{u - d} \tag{2.4c}$$

In (2.4b) and (2.4c) , p denotes the risk-neutral probability of the u-scenario being realized. In the risk-neutral world, the expected return on the underlying asset equals the return on a risk-free investment by construction. This result allows for arbitrage-free pricing of options independent on the real-world expected return on the underlying asset.

This can be generalized to a multi-step binomial lattice using the binomial probability model. Defining n as the number of steps in the binomial lattice, p as the risk-neutral probability, and $f(\cdot)$ as the payoff function of the derivative security, the price of a European derivative is

$$f_0 = \left[\sum_{j=0}^{n} \frac{n!}{(n-j)!j!} p^j (1-p)^{n-j} f(S_0 u^j d^{n-j}) \right] e^{-rT} \tag{2.5}$$

By adding more steps to the discrete time binomial lattice while letting the step size, Δt, approach zero, the binomial lattice converges towards the lognormal distribution and the option price converges towards the Black-Scholes-Merton price.

2.3 Stochastic Processes

In order to understand the connection between the binomial lattice model of section 2.2 and the Black-Scholes-Merton option pricing formula, stochastic processes are briefly introduced. Stochastic processes take an essential part in the derivation of the partial differential equation identified and solved by Black, Scholes, and Merton.

A stochastic process is defined as the uncertain behavior of variables across time, where time can be measured both discretely and continuously. They are used in numerous scientific fields to model variables that behave in a stochastic or unpredictable manner, including physics, biology, and engineering. They also find applications in the financial markets, since they are often considered unpredictable and random. This section aims to derive the stochastic process of a stock as assumed by Black and Scholes in their model: the geometric brownian motion.

Wiener Process. Let z be a variable that follows a Wiener process. Two properties defines Wiener processes: The first property is that instantaneous changes in z are normally distributed with a mean of 0 and a variance of $\mathrm{d}t$, i.e., $\mathrm{d}z \sim N(0, \mathrm{d}t)$. The second property is that $\mathrm{d}z$ is independent across any two different intervals of time. This implies that Wiener processes satisfy the Markov property, namely that the value of the process in the next time step, whether discrete or continuous, only depends on the current level of the process, i.e., it is independent of past values of the process

$$E\left[z_{t+1} \middle|\ z_1, z_2, \ldots, z_t\right] = E\left[z_{t+1} \middle|\ z_t\right] \tag{2.6}$$

Wiener processes are also termed Brownian Motion, which stems from their use as a model for the random motion of particles in fluids as discovered by botanist Brown (2019).

Generalized Wiener Process. Let x be a variable that follows a generalized Wiener process defined as $\mathrm{d}x = a\ \mathrm{d}t + b\ \mathrm{d}z$ where a and b are constants corresponding to the drift coefficient determining the trend of the process and the diffusion coefficient determining the variation around the trend, respectively. Changes in the generalized Wiener process are normally distributed with a mean of $a\ \mathrm{d}t$ and a variance of $b^2\ \mathrm{d}t$, i.e. $\mathrm{d}x \sim N(a\ \mathrm{d}t,\ \ b^2\ \mathrm{d}t)$.

Itô Process. Constant drift and diffusion coefficients are unrealistic properties of stocks, since investors generally are concerned with the expected return and variance in percentage of the stock price. To address is issue, the Itô process is introduced. Let x be a variable that follows an Itô process defined as

$$\mathrm{d}x = a(x,t)\ \mathrm{d}t + b(x,t)\ \mathrm{d}z \tag{2.7}$$

where the drift and diffusion coefficients are functions of the process x and time t. Define x as the stock price, S, and define μ as the constant expected percentage return on S and σ as the constant standard deviation of S. The process of S is then

$$\mathrm{d}S = \mu S\ \mathrm{d}t + \sigma S\ \mathrm{d}z \tag{2.8}$$

while the percentage change in S follows the process

$$\frac{\mathrm{d}S}{S} = \mu\ dt + \sigma\ \mathrm{d}z \tag{2.9}$$

where $\mathrm{d}S/S \sim N(\mu\ \mathrm{d}t, \sigma^2\ \mathrm{d}t)$. This variation of the Itô process is referred to as geometric brownian motion.

2.4 Itô's Lemma

One way to prove the Black-Scholes partial differential equation relies on Itô's lemma: a stochastic calculus proposition used to identify differentials of time-dependent stochastic processes Itô (1944). Defining G as a twice-differentiable function of the Itô process x and of time t, a Taylor series expansion can be used to approximate the change in the function G

$$\mathrm{d}G = \frac{\delta G}{\delta x}\ \mathrm{d}x + \frac{\delta G}{\delta t}\ \mathrm{d}t + \frac{1}{2}\frac{\delta^2 G}{\delta x^2}\ \mathrm{d}x^2 \tag{2.10}$$

In (2.10), terms of time increments, $\mathrm{d}t$, are retained up to the first order of the Taylor series expansion, while terms of the Itô process increments, $\mathrm{d}x$, are retained up to the second order. Substituting (2.7) into (2.10) while omitting arguments, so $a(x,t)$ and $b(x,t)$ becomes a and b, yields

$$\mathrm{d}G = \frac{\delta G}{\delta x}(a\ dt + b\ \mathrm{d}z) + \frac{\delta G}{\delta t}\ \mathrm{d}t + \frac{1}{2}\frac{\delta^2 G}{\delta x^2}(a^2\ \mathrm{d}t^2 + b^2\ \mathrm{d}z^2 + 2ab\ \mathrm{d}t\ \mathrm{d}z) \tag{2.11}$$

To simply (2.11), the following properties from Itô calculus are used:

$$\mathrm{d}z^2 = \mathrm{d}t \tag{2.12a}$$

$$\mathrm{d}t^2 = 0 \tag{2.12b}$$

$$\mathrm{d}z \times \mathrm{d}t = 0 \tag{2.12c}$$

Proving the first property requires rigorous mathematics that are out of the scope of this thesis, but the last two properties can be understood (not proven) easily. Concretely, $\mathrm{d}t^2$ and $\mathrm{d}z \times \mathrm{d}t$ must approach zero faster than $\mathrm{d}t$ asymptotically, since these values are smaller than $\mathrm{d}t$. For this reason, they can be considered asymptotically equal to zero. These properties simplify (2.11) into

$$\mathrm{d}G = \frac{\delta G}{\delta x}\ (a\ \mathrm{d}t + b\ \mathrm{d}z) + \frac{\delta G}{\delta t}\ \mathrm{d}t + \frac{1}{2}\frac{\delta^2 G}{\delta x^2}\ (b^2\ \mathrm{d}t) \tag{2.13}$$

which can be rearranged to bring forth the well-known representation of Itô's lemma

$$\mathrm{d}G = \left(\frac{\delta G}{\delta x}\ a + \frac{\delta G}{\delta t} + \frac{1}{2}\frac{\delta^2 G}{\delta x^2}\ b^2\right)\mathrm{d}t + \frac{\delta G}{\delta x}\ b\ \mathrm{d}z \tag{2.14}$$

Letting $x = S$, and consequently replacing the drift and diffusion coefficients a and b by the drift and diffusion coefficients of S as given by (2.8), the instantaneous change in a derivative on S is

$$\mathrm{d}f = \left(\frac{\delta f}{\delta S}\ \mu S + \frac{\delta f}{\delta t} + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\ \sigma^2 S^2\right)\mathrm{d}t + \frac{\delta f}{\delta S}\ \sigma S \mathrm{d}z \tag{2.15}$$

2.5 Black-Scholes Partial Differential Equation

In conjunction, Itô's lemma and the hedging argument under assumptions of absence of arbitrage provide a way to derive the Black-Scholes partial differential equation (PDE). Analogous to section 2.2, the derivation of the Black-Scholes PDE builds on the principle of constructing a portfolio consisting of a short position in the derivative security and a long position equal to $\delta f/\delta S$ in the underlying asset. This portfolio achieves the goal of eliminating the risk during time $\mathrm{d}t$ that stems from changes in the Wiener process, $\mathrm{d}z$. By the absence of arbitrage argument, this portfolio must have a return equal to the risk-free rate and its price must be equal to the price of the derivative.

Assume the underlying asset, S, follows the process given in (2.8) and that the price of a derivative on S is known at time t. Define $\Pi(S,t)$ as the value of a portfolio at time t of a short position in the derivative and a long position in S equal to $\delta f(S,t)/\delta S$

$$\Pi(S,t) = -f(S,t) + \frac{\delta f(S,t)}{\delta S}\ S \tag{2.16}$$

For notional purposes, the arguments of $f(S,t)$ and $\Pi(S,t)$ are omitted. Letting $\delta f/\delta S$ be fixed during the period t to $t+\mathrm{d}t$, the instantaneous change in the value of the portfolio is

$$\mathrm{d}\Pi = -\mathrm{d}f + \frac{\delta f}{\delta S}\ \mathrm{d}S \tag{2.17}$$

The instantaneous change in the value of the derivative, $\mathrm{d}f$, is given by Itô's lemma in (2.15). By substituting (2.15) and (2.8) into (2.17)

$$\mathrm{d}\Pi = \left(-\frac{\delta f}{\delta S}\ \mu S - \frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\ \sigma^2 S^2\right)\mathrm{d}t - \frac{\delta f}{\delta S}\ \sigma S\ \mathrm{d}z + \frac{\delta f}{\delta S}(\mu S\ \mathrm{d}t + \sigma S\ \mathrm{d}z) \tag{2.18a}$$

$$\mathrm{d}\Pi = \left(-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\ \sigma^2 S^2\right)\mathrm{d}t \tag{2.18b}$$

it becomes clear that the Wiener process is eliminated and that changes in the value of the portfolio is a deterministic function of time. This implies that only the passing of time has an effect on the portfolio value, and this effect is perfectly predictable under the assumptions

of Black-Scholes (1973). Assuming continuous trading in S and that S is perfectly divisible, it follows that this portfolio is risk-free from time t to $t + \mathrm{d}t$. According to the absence of arbitrage argument, the return on this portfolio must be equal to the risk-free rate

$$\mathrm{d}\Pi = r\Pi \, \mathrm{d}t \tag{2.19}$$

Substituting (2.16) and (2.18) into (2.19)

$$\left(-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)\mathrm{d}t = r\left(-f + \frac{\delta f}{\delta S}S\right)\mathrm{d}t \tag{2.20}$$

and dividing (2.20) by $\mathrm{d}t$, the Black-Scholes PDE materializes

$$\frac{\delta f}{\delta t} + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2 + \frac{\delta f}{\delta S}rS - rf = 0 \tag{2.21}$$

The partial differential equation in (2.21), referred to as the Black-Scholes equation, is the breakthrough that changed the financial markets by providing a framework for pricing and hedging of derivative securities. Concretely, the price of any derivatives that is a function of only S and t must satisfy the Black-Scholes partial differential equation.

2.6 Black-Scholes-Merton Option Pricing Formula

The majority of derivative securities do not have an analytical solution to the Black-Scholes equation and must, therefore, be priced using numerical methods, most prominently using binomial lattice, Monte Carlo and finite difference methods. However, Black and Scholes (1973) succeeded in identifying analytical solutions to the Black-Scholes equation for European call and put options.

In order to solve the Black-Scholes equation, it is necessary to identify boundary conditions for a particular derivative. The boundary conditions specify which derivative is being priced

by using properties of the derivative. For a European call option, the boundary conditions are

$$f(T, S) = \max(S - K, 0) \tag{2.22a}$$

$$f(t, 0) = 0 \tag{2.22b}$$

$$\lim_{S \to \infty} f(t, S) = S - Ke^{-r(T-t)} \tag{2.22c}$$

and for a European put option, the boundary conditions are

$$f(T, S) = \max(K - S, 0) \tag{2.23a}$$

$$f(t, 0) = Ke^{-r(T-t)} \tag{2.23b}$$

$$\lim_{S \to \infty} f(t, S) = 0 \tag{2.23c}$$

The first condition of both call and put options defines the values of the options at maturity T as their respective payoff functions. The second condition defines the value of the call option as zero if the price of S at any time $t \leq T$ is 0, since S is unable to recover given it follows a geometric brownian motion. On the other hand, the put option value is the present value of K if $S = 0$, since the payoff at maturity is K with certainty. The third condition defines the value of a call option in the limit as $S \to \infty$ as S less the present value of the strike price, since the probability that the option is exercised at maturity approaches 100%. On the other hand, the value of a put option is 0 as $S \to \infty$, since the probability that the option will expire worthlessly approaches 100%.

Black and Scholes (1973) solved the Black-Scholes equation for European call and put options using the boundary conditions in (2.22) and (2.23). They found the price of a European call option on a dividend-paying underlying asset to be

$$c(S_0, r, q, \sigma, K, T) = S_0 e^{-qT} N(d_1) - Ke^{-rT} N(d_2) \tag{2.24}$$

and the price of a European put option on a dividend-paying underlying asset to be

$$p(S_0, r, q, \sigma, K, T) = Ke^{-rT} N(-d_2) - S_0 e^{-qT} N(-d_1) \tag{2.25}$$

where

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(r - q + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T} \tag{2.26}$$

and

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \mathrm{d}x \tag{2.27}$$

is the cumulative distribution function of a standard normal distribution.

The function $N(d_1)$ can be interpreted as the position in the underlying asset needed for continuous hedging, commonly referred to as the delta. The function $N(d_2)$ can be interpreted as the risk-neutral probability that a call option expires in-the-money. In addition to providing the fair price of a European option, the Black-Scholes-Merton model (henceforth the BSM model) provides dealers with a methodology of how to hedge the risk associated with issuing and holding options. In particular, the only source of risk under the assumptions of the BSM model comes from changes in the price of the underlying asset, and this risk can be eliminated through continuous hedging using positions in the underlying asset equal to $N(d_1)$, or delta.

2.7 Assumptions of the Black-Scholes-Merton Model

The BSM model is, however, only as good as the assumptions underlying the model and these assumptions are generally considered unrealistic. As I discuss in section 3.4, one of the advantageous properties of artificial neural networks is that they are assumption-free and therefore have the potential to compete against the BSM model. In this section, I will briefly discuss the assumptions of the BSM model.

The primary assumptions underlying the BSM model are:

1. The underlying asset follows a geometric brownian motion with constant drift and diffusion coefficients.

2. The risk-free interest rate is constant and identical across all maturities.

3. Securities trade continuously and are perfectly divisible.

4. There are no transaction costs or taxes.

5. Short selling of securities with full use of proceeds is permitted.

6. There are no riskless arbitrage opportunities.

The first two assumptions are not strictly necessary in order to solve the Black-Scholes equation. For instance, several extensions to the BSM model attempt to circumvent the assumption of constant volatility such as the Practitioner Black-Scholes model (Christoffersen & Jacobs, 2004) that assumes a deterministic volatility function and the Heston model (Heston, 1993) that assumes stochastic volatility. This is discussed further in section 2.9, where I derive the Practitioner Black-Scholes model as a benchmark for comparison to the artificial neural network model.

The third assumption states that hedging using the delta, $N(d_1)$, can occur continuously at fractions of a share such that the replicating portfolio is always risk-free, which is impossible in the real world. Instead, frequent discrete hedging is necessary, but it leads to high transaction costs in the real world which breaks the fourth assumption. The fifth assumption is unrealistic due to margin requirements typically applying to investors wanting to sell short, hence restricting the use of proceeds from short selling. I will not go in details about the truthfulness of the sixth assumption since there are several schools of thought regarding the degree of market efficiency and these opposing views are out of the scope of this thesis.

2.8 Implied Volatility

The expensiveness of options is often measured in terms of implied volatility. The implied volatility of an option is defined as the volatility that makes the BSM price equal to the observed market price

$$c_{obs}(K,T) = C_{BSM}(K,T,\sigma_{IV}) \tag{2.28}$$

where $c_{obs}(K,T)$ is the observed market price of an option as a function of K, the strike price, and T, the time-to-maturity, and $C_{BSM}(K,T,\sigma_{IV})$ is the BSM price as a function of K, T and σ_{IV}, the implied volatility.

Under the assumptions of the BSM model, the implied volatility would be constant. However, when plotting implied volatility against strike price for index options, the fitted line resembles a smirk, referred to as the volatility smirk (Hull, 2018). This implies that volatility is a function of strike price and that OTM puts and ITM calls are more expensive than ITM puts and OTM calls.

Moreover, evidence shows that implied volatility is a function of time-to-maturity, referred to as the volatility term structure (Hull, 2018). Concretely, implied volatility is typically increasing with maturity when short-term volatilities are lower than average since volatility is expected to increase in the future. The opposite case is often seen when short-term volatility is higher than average.

These empirical observations are often combined into a so-called volatility surface: a surface plot of implied volatility against moneyness on the x-axis and time-to-maturity on the z-axis. The existence of the volatility surface is one of the major breakpoints of the BSM model, that assumes constant volatility.

2.9 Practitioner Black-Scholes Model

The Practitioner Black-Scholes (PBS) model is a popular extension to the original BSM model. It circumvents the undesirable assumption of constant volatility in the underlying asset by modeling implied volatility as a quadratic function of the strike price and the time-to-maturity (Rouah & Vainberg, 2007). It was developed by Dumas et al. (1998) under the name 'ad-hoc model', but it is better known as the Practitioner Black-Scholes Model as Christoffersen and Jacobs (2004) later termed it.

Dumas et al. (1998) use deterministic volatility functions (DVF) to model implied volatility, and then use the predicted implied volatility as input to the BSM model. They consider four specifications of the DVF, but I will focus on the one they found to be performing best since I only use the model as a benchmark model. This DVF is defined as

$$\sigma_{IV} = \max\left(a_0 + a_1 K + a_2 K^2 + a_3 T + a_4 T^2 + a_5 KT, 0.01\right) \tag{2.29}$$

where the model parameters, a_i, are the result of an ordinary least squares (OLS) regression and the fitted implied volatility is restricted to a minimum value of 0.01 in order to avoid negative predicted volatility. The DVF is a continuous function that can be interpreted as an estimated volatility surface from which the BSM volatility parameter can be extracted. Dumas et al. (1998) show that the implied volatility surface primarily depends on K, K^2, T, and KT, while T^2 provides little improvement to the model. From this, they conclude that the curvature in the volatility smile is determined primarily by the strike price, whereas implied volatility is primarily linearly related to time-to-maturity.

I include the PBS model as a benchmark model to compare with the artificial neural network model I develop in this thesis. I do not include the Heston stochastic volatility model (Heston, 1993) mentioned in section 2.7, since Christoffersen and Jacobs (2004) find that the PBS model is better able to fit option prices and due to the much simpler model specifications of the PBS model.

2.10 Supply-and-Demand-Based Option Pricing

The BSM model is a theoretical model that gives the fair price of European options under the assumptions presented in section 2.7. Here, fair price refers to the no-arbitrage price that was proven using the hedging argument. However, Bates (2003) argue that the model cannot fully capture or explain the empirical properties of exchange-traded option prices. He bases his proposition on market makers' imperfect ability to hedge option positions due to nonperfect capital markets unlike the assumption of the BSM model. Instead, he argues that inventory

risk, or demand pressure, should be considered when pricing derivative securities (Bates, 2003). Amihud, Mendelson, and Pedersen (2006) define inventory risk as the compensation market makers require from making market when demand is larger than market supply. Inventory risk is thus a compensation to market makers when the demand exceeds the market makers willingness to make market. Amihud et al. (2006) further defines inventory risk as an exogenous liquidity premium and argue that market makers increase bid-ask spreads of assets due to regulatory capital constraints and in order to reduce risk exposure. In this way, market makers can limit the demand to a level where they are comfortable with the associated risk by increasing the bid-ask spread, thus introducing a liquidity premium.

According to Bollen and Whaley (2004), inventory risk allows market prices to diverge from theoretical option prices, and it creates a no-arbitrage band within which the market price can fluctuate without arbitrage opportunities. Garleanu, Pedersen, and Poteshman (2008) explain this divergence of market prices from theoretical prices by varying demand and supply. The varying demand arises due to option-users having various reasons for trading options, while the varying supply arises due to varying willingness of market makers to make market by taking the opposite position. Bollen and Whaley (2004) find that particularly index put options are affected by buying pressure from institutional investors, who use the put options as portfolio insurance. This is in accordance with Garleanu et al. (2008) and Constantinides and Lian (2015), who find evidence that market makers are net sellers of index put options. Since the demand for put options typically exceed the market supply, market makers need to provide liquidity by acting as counterparties and hedging the related risks. Garleanu et al. (2008) argue that this supply and demand imbalance can explain approximately one-third of index option expensiveness.

The literature presented suggest that option prices fluctuate within a no-arbitrage band, partly as a result of imbalances between demand and supply and the inventory risk associated with demand pressure to market makers. In Chapter 5, I examine the ability of artificial neural networks to incorporate this effect using four variables that I hypothesize proxy the market's demand for insurance in the form of put options and the liquidity premium.

Chapter 3

Artificial Neural Networks

The focus of this chapter is to introduce the machine learning theory underlying the methodology of this thesis. The chapter is structured in the following manner: Initially, I introduce the scientific field of machine learning with a focus on supervised learning and the bias-variance tradeoff, followed by an introduction to the artificial neural network called multilayer perceptron and its underlying mathematics. Particular emphasis will be given to the structure of multilayer perceptrons and the machine learning algorithm used to train them.

3.1 Introduction to Machine Learning

The primary purpose of machine learning (ML) algorithms is to draw conclusions based on data, known as statistical inference. In simple terms, machine learning algorithms are statistical models that learn from data without being given explicit instructions. As the algorithms are given more data to learn from, they are better able to provide insights into the underlying probability distribution of the data and to provide accurate predictions.

The ML algorithm examined in this thesis is of the supervised learning type. In supervised learning, a loss function is minimized by adjusting the model parameters through optimization

Alpaydin (2010). A well-known ML algorithm that satisfies this definition is linear regression. Linear regression identifies linear relationships between variables by minimizing the mean squared error between the estimated model predictions and the actual data by adjusting the model parameters using ordinary least squares. Even more advanced supervised learning algorithms follow this same structure but the model specifications, loss functions, and optimization methods are different. Specifically, supervised learning is defined as a machine learning method, where the objective is to learn the mapping from an input x to an output y (Alpaydin, 2010). This implies that the output is known in advance such that the ML algorithm can learn the relationship between the input and the output and infer a function that can be used to predict new observations.

Supervised learning is further subcategorized into classification and regression. Classification problems are concerned with predicting to which category an input belongs. In other words, the classification supervised learning algorithms map input to categorical outputs, such as whether an email is spam or not or which dog breed is shown in a picture. On the other hand, regression problems are concerned with mapping inputs to a real-valued output, such as the amount of rainfall in a month or the price of a financial derivative as is the focus of this thesis. Essentially, the aim of this thesis is to identify a function that connects some input variables to market prices of call options.

3.2 Bias-Variance Tradeoff

In analyzing the performance of machine learning algorithms, the bias-variance tradeoff is very important. The bias-variance tradeoff is essentially a tradeoff between underfitting and overfitting a dataset. In particular, the bias-variance tradeoff is a measure of the generalization error of a machine learning model seen from the perspective of three sources of errors: bias, variance and irreducible error. Bias is defined as the prediction error and can be measured as the mean squared error or a similar function. Variance is defined as the variability of model predictions and it can be measured using k-fold cross-validation as explained in section 4.3.

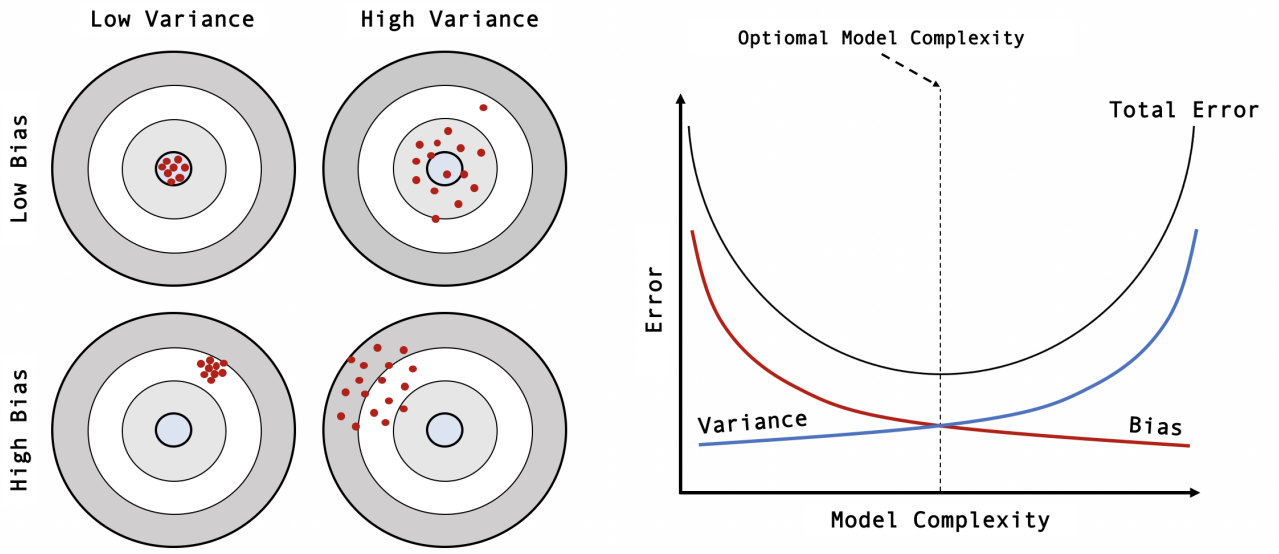

FIGURE 3.1: Illustrations of the bias-variance tradeoff showcasing the importance of identifying the optimal tradeoff between bias and variance.

Finally, the irreducible error is the noise contained within data that cannot be reduced by building a better model.

The purpose of utilizing the bias-variance framework in model building is to find the right balance between underfitting and overfitting data. The perfect scenario involves a minimization of both the bias and variance such that only the irreducible error remains. In cases where the bias is high, the model is underfitting the data and could benefit from an increase in model complexity. On the other hand, when the model starts to explain noise in the data, the variance is high, and the model will generalize poorly to new observations. A good tradeoff between bias and variance occurs when the model complexity is such that the model predicts accurately and consistently. The considerations when building low bias and low variance ANNs are thus to find the optimal complexity of the model with respect to the inputs used, the number of hidden layers, the number of perceptrons, etc.

3.3 Introduction to Artificial Neural Networks

The human brain is perhaps the greatest known learning machine with purportedly 100 billion interconnected neurons that process information in a very complex but efficient manner (Herculano-Houzel, 2009). The human brain's impressive learning capabilities inspired Mcculloch and Pitts (1943) to develop an Artificial Neural Network (ANN): a mathematical representation of neurons in the human brain.

The neurons of the human brain can be viewed as computational units that process information. Neurons receive signals through so-called dendrites, which are 'wires' that connect neurons. After the neurons have 'computed' the input signals, they are passed on to other neurons through another set of 'wires' known as axons. In this way, neurons of the human brain are structured in interconnected networks consisting of billions of connections (Herculano-Houzel, 2009).

However, the work of Mcculloch and Pitts (1943) was merely a structural and mathematical representation of how the human brain processes information, known as forward propagation in machine learning terminology. A technique for training the ANN to learn from data was not developed until 43 years later by Rumelhart, Hinton, and Williams (1986).

ANNs can take numerous forms depending on the objective at hand. For instance, a certain type of ANN is often used for object recognition in pictures and videos, while another is preferred for text and speech recognition. Since this thesis is dealing with the approximation of a continuous function, the most appropriate method is the 'multilayer perceptron' (MLP) and I will focus only on this type of ANN.

In MLPs, the neurons of the human brain are modelled by perceptrons that work in a similar way to biological neurons in the brain. Perceptrons perform computations on inputs received from data or from the outputs of other perceptrons and then channels the results to yet other perceptrons. Similar to the neurons in the human brain, perceptrons are structured in interconnected networks consisting of one to multiple layers, called hidden layers, which explains why the type of ANN is referred to as a multilayer perceptron.

Each perceptron contains an activation function that processes the information it receives in an often nonlinear manner and channels the results of this computation to the connected perceptrons in the next layer. This process is repeated stepwise for each layer in the MLP until the final layer, where the output is channeled into a single perceptron that forms the final prediction. This process of producing predictions based on propagating inputs through the MLP is denoted forward propagation.

Each connection between perceptrons has an associated weight that determines how much information is passed on to each of the perceptrons in the next layer. These weights are obtained through training of the MLP, where the objective is to identify the weights that result in the most accurate predictions. This is achieved using the gradient descent algorithm described in section 3.7.

Similar to the constant in linear regression allowing for vertical shifts of the fitted line along the y-axis, MLPs have bias units. The bias units are set equal to one, while the associated weight is optimized through the gradient descent algorithm. This allows for the activation functions connected to the bias unit to shift vertically in the same way that the constant in linear regression allows for the fitted line to change its intercept through a vertical shift. The result of this is that the ANN can fit data better similarly to how the constant of linear regression allows for a better fit to data.

I have included a generalized representation of an MLP in Figure 3.2, where $a_i^{(j)}$ denotes the value of perceptron i in layer j. The number of hidden layers as well as perceptrons per hidden layer is discretionary, but they are often chosen using the grid search method described in section 4.3. Each arrow in Figure 3.2 correspond to a weight, $\Theta_{i,k}^{(j)}$, that connects $a_k^{(j)}$ to $a_i^{(j+1)}$ where subscripts k and i are related to perceptron k and i in layer j and $j+1$, respectively. The weights are only illustrated for the last layer for simplicity, but a weight is similarly associated with each other arrow connecting perceptrons.

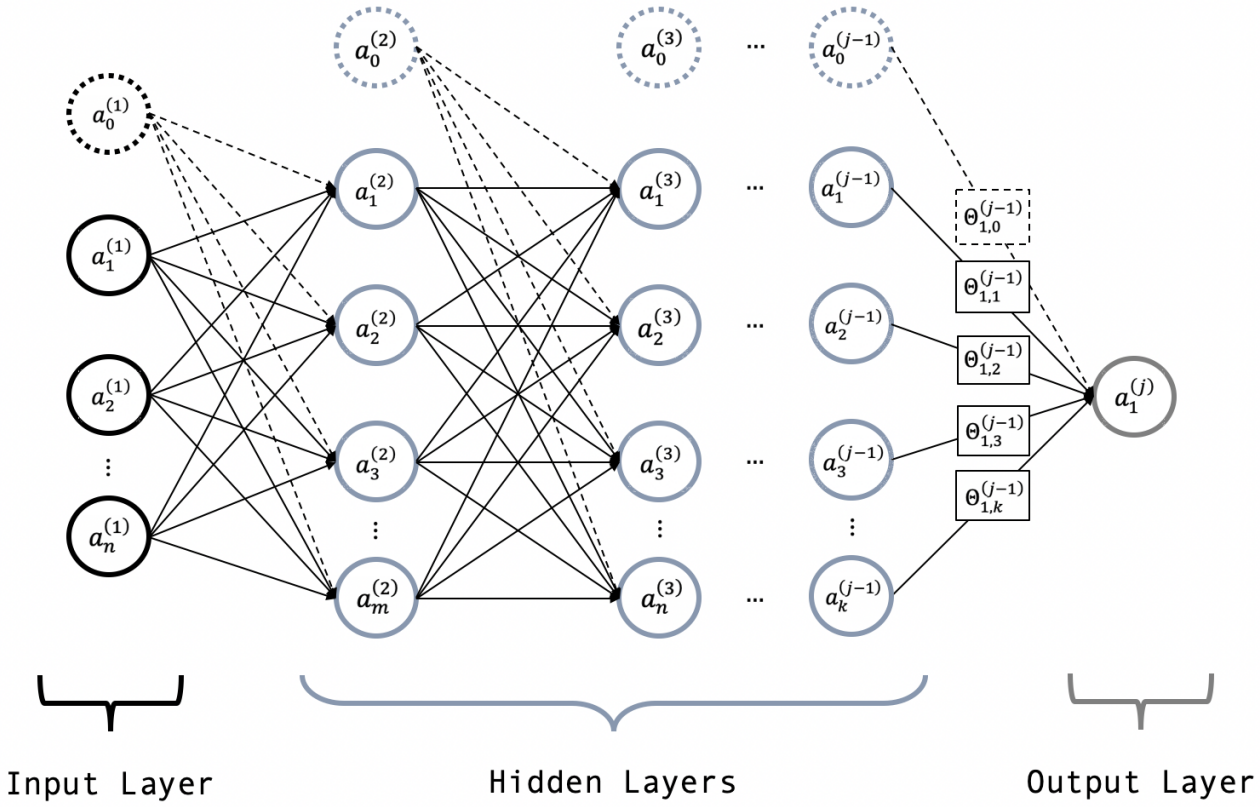

FIGURE 3.2: Structural representation of an artificial neural network. Each arrow corresponds to a weight like shown in the last layer, but they have been omitted for visual purposes

3.4 Properties of Artificial Neural Networks

ANNs belong to the family of nonparametric statistical models, meaning they do not rely on assumptions about the parameters of the underlying probability distribution in contrast to the vast majority of financial models. Consequently, ANNs allow the probability distribution to take any form as given by data, which proves valuable for a range of purposes where traditional parametric models fail to provide an accurate fit. This feature is particularly appropriate for empirical option pricing, since market participants generally disregard the lognormal return assumption underlying the BSM model due partly to the property that stock returns tend to have fatter tails than the normal distribution (Taleb, 2008).

ANNs are nonlinear models capable of fitting complex nonlinear data and providing nonlinear predictions. This feature among others are the underpinnings of the universal approximation theorem first proved by Cybenko (1989). This theorem states that ANNs with a finite number of layers and neurons can approximate any continuous function. However, the theorem does not advise how many layers or neurons are needed to achieve this, meaning there is no guarantee to find the optimal solution (Kubat, 2015).

In general, artificial neural networks are considered highly non-transparent due to the complexity of the connections between perceptrons and the nonlinear activation functions. Unlike simpler forms of machine learning such as linear regression, the parameters of the model are extremely difficult to interpret and the relationship between the input and output is thus often criticized for being a black box (Benitez, Castro, & Requena, 1997). Generally, researchers and practitioners disregard statistical inference of ANNs and instead trust the model predictions if the out-of-sample prediction errors are sufficiently small for a specific purpose and if the bias-variance tradeoff is sufficiently balanced.

3.5 Forward Propagation

Though visually more appealing, the graphical illustration in Figure 3.2 of forward propagation in an MLP lacks the mathematics needed to understand how the algorithm can be implemented in a computer. For this purpose, a vectorized representation is advantageous since it simplifies equations while being computationally efficient. The following notation is adapted from Ng and Katanforoosh (2018).

Following the notation of Figure 3.2, each perceptron, except those given directly from the data in the input layer, are calculated by applying an activation function to a weighted sum of the perceptrons in the preceding layer,

$$a_i^j = g\left(\Theta_{i,0}^{(j-1)} a_0^{(j-1)} + \Theta_{i,1}^{(j-1)} a_1^{(j-1)} + \ldots + \Theta_{i,n}^{(j-1)} a_n^{(j-1)}\right) \tag{3.1}$$

where $g(\cdot)$ is an activation function and the bias unit $a_0^{(j-1)} \equiv 1$. The above can be represented using matrix notation by denoting the input of the $g(\cdot)$ function by z_i^j,

$$\mathbf{z}^j = \mathbf{\Theta}^{(j-1)}\mathbf{a}^{(j-1)} = \begin{bmatrix} \Theta_{1,0}^{(j-1)} & \Theta_{1,1}^{(j-1)} & \cdots & \Theta_{1,k}^{(j-1)} \\ \Theta_{2,0}^{(j-1)} & \Theta_{2,1}^{(j-1)} & \cdots & \Theta_{2,k}^{(j-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{i,0}^{(j-1)} & \Theta_{i,1}^{(j-1)} & \cdots & \Theta_{i,k}^{(j-1)} \end{bmatrix} \begin{bmatrix} a_0^{(j-1)} \\ a_1^{(j-1)} \\ \vdots \\ a_k^{(j-1)} \end{bmatrix} = \begin{bmatrix} z_1^j \\ z_2^j \\ \vdots \\ z_i^j \end{bmatrix} \tag{3.2}$$

The vector of perceptions at layer j including the added bias unit is then

$$\mathbf{a}^j = \begin{bmatrix} a_0^{(j)} \\ a_1^{(j)} \\ a_2^{(j)} \\ \vdots \\ a_i^{(j)} \end{bmatrix} = \begin{bmatrix} 1 \\ g\left(z_1^{(j)}\right) \\ g\left(z_2^{(j)}\right) \\ \vdots \\ g\left(z_i^{(j)}\right) \end{bmatrix} \tag{3.3}$$

This notation is applicable to each hidden layer, but also the output layer where the parameter matrix, $\mathbf{\Theta}^{(j-1)}$, simplifies to a vector since the perceptrons of the penultimate layer are channeled into a single output perceptron as illustrated in Figure 3.2.

3.6 Activation Functions

The $g(\cdot)$-function, known as the activation function, can take several forms. For classification problems, the sigmoid activation function known from logistic regression is often applied since it outputs a value between 0 and 1 that can be interpreted as a probability. For regression problems, the Rectified Linear Unit, or ReLU is a commonly applied activation function. This activation function coincidentally looks like the payoff of a call option as seen in Figure 3.3. As such, the ReLU activation function will only output positive or zero values in accordance with option prices always being positive or zero. For these reasons, and due to trial-and-error, this thesis applies the ReLU activation function in the hidden layers. Concretely, the ReLU

function is

$$g(x) = \max(x, 0) \tag{3.4}$$

For the output perceptron, a simple activation function denoted linear unit is applied, which simply is

$$g(x) = x \tag{3.5}$$

The linear unit thus simplifies to the weighted sum of the perceptrons in the preceding layer, $z^{(j-1)}$, essentially equivalent to no activation function in the output layer.

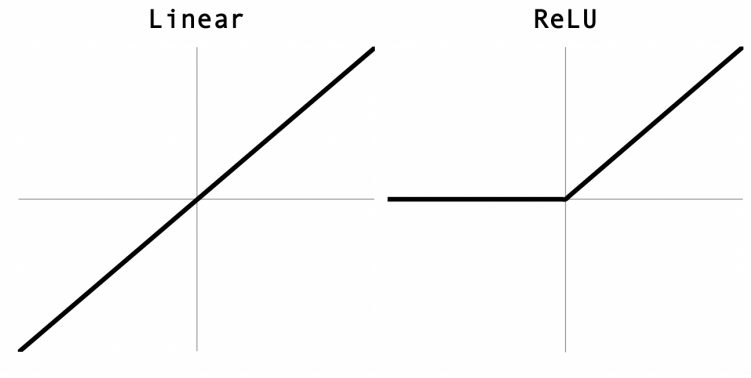

FIGURE 3.3: Graphical illustration of the two activation functions used in this thesis.

3.7 Gradient Descent Algorithm

As mentioned in section 3.1, the learning process of the ANN occurs through minimization of a cost function by changing its parameters. When dealing with regression-type problems, the most frequently used cost function is the mean squared error (MSE)

$$J(\Theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \widehat{y_i})^2 \tag{3.6}$$

The cost functions of ANNs are normally minimized using variations of the gradient descent algorithm, which uses the gradients of the cost function in order to find the global minimum using an iterative process. The gradient is defined as a vector composed of the partial derivatives

of the cost function with respect to all parameters of the model

$$\nabla J\left(\boldsymbol{\Theta}\right) = \begin{bmatrix} \frac{\delta J(\boldsymbol{\Theta})}{\delta \theta_{1,0}} & \frac{\delta J(\boldsymbol{\Theta})}{\delta \theta_{1,1}} & \dots & \frac{\delta J(\boldsymbol{\Theta})}{\delta \theta_{i,k}} \end{bmatrix}^T \tag{3.7}$$

The gradient informs us which multidimensional 'direction' has the steepest upward slope and it can therefore be used to determine in which direction a step would lead to the cost function being reduced the most; that is the opposite direction of the gradient. Concretely, the gradient descent algorithm initiates by choosing random parameters and calculating the cost and gradient related to these parameters.

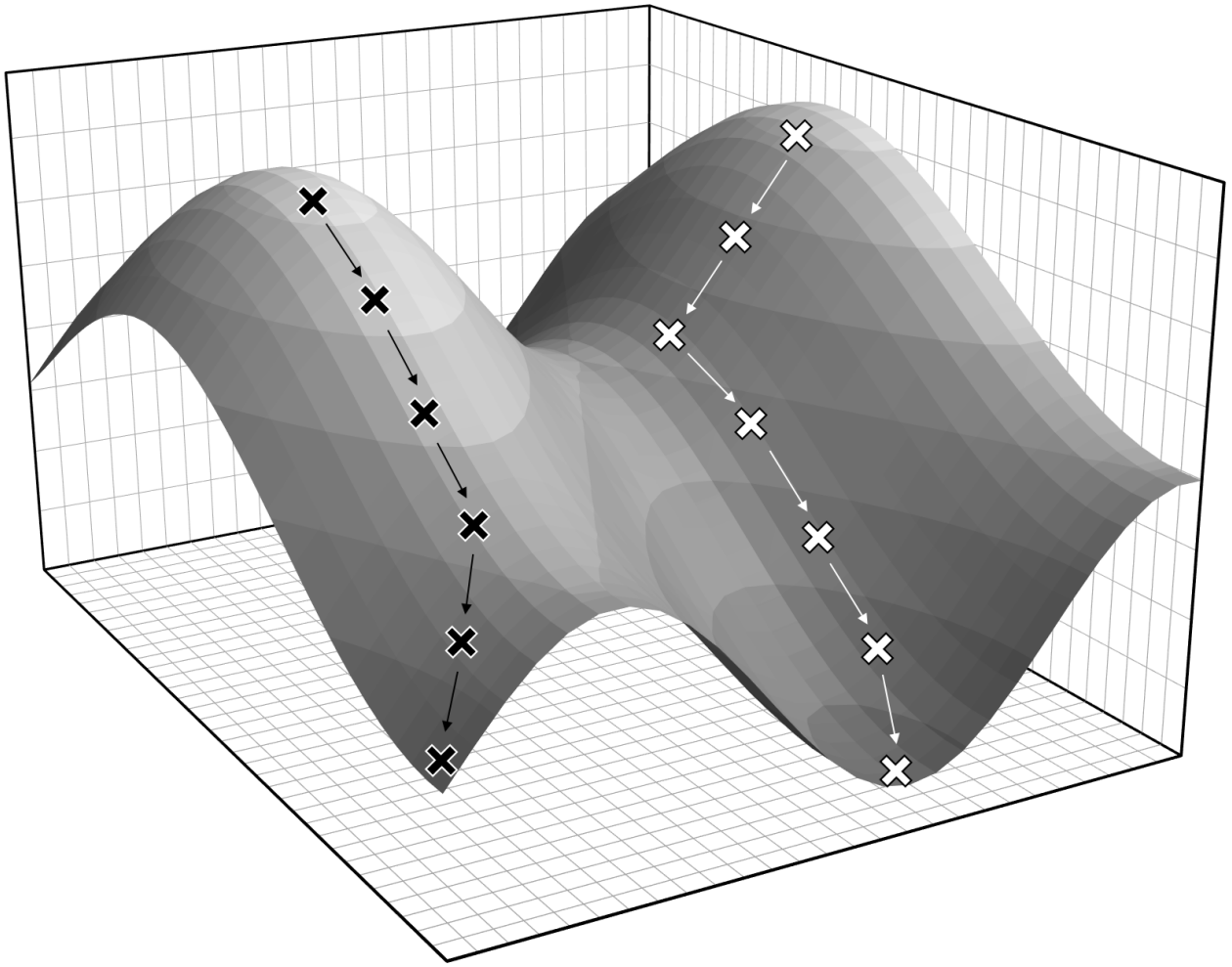

FIGURE 3.4: Three-dimensional illustration of the gradient descent algorithm. Notice, how two initial starting points can lead to different solutions.

A learning rate, η, is chosen, which determines the step-size of each iteration of the gradient descent algorithm. The learning rate affects the rate of convergence and the ability for the optimization algorithm to escape local minima. The parameters are then updated iteratively as

$$\Theta_j = \Theta_j - \eta \nabla J\boldsymbol{\Theta} \tag{3.8}$$

until the gradient approaches zero which indicates a local minimum has been reached. Notice, that there is no guarantee of reaching the global minimum unless the cost function is convex.

One approach to training an ANN using the gradient descent algorithm is to adjust the parameters of the model for each observation; an approach with slow convergence since each observation affects the model parameters. Another approach is to compute the average gradient of the entire dataset at once, which is a computationally expensive task. The compromise is to use batches of training examples, which ensures computationally efficient and fast convergence of the optimization algorithm. When the entire training data has been processed through the ANN, the model is said to have been through one epoch and the decision of batch size and the number of epochs to train the ANN needs to be decided before training initiates. In section 4.4, I discuss how the grid search method is used to identify the optimal batch size and number of epochs.

Chapter 4

Methodology

In this chapter, I describe data collection and preprocessing in detail, followed by a description of the cross-validation and the grid search methods used to identify the optimal hyperparameters of the ANNs. Next, I present the methods used to forecast volatility and the metrics used to measure the performance of the ANNs. Finally, I introduce the additional variables expected to improve the ANN. The data preparation, model training and analysis are conducted in the Python programming language primarily using the open source TensorFlow (Google, 2019) and SciKit-Learn (Pedregosa et al., 2011) packages.

4.1 Data

The data underlying the analysis of this thesis is retrieved from two sources: OptionMetrics (retrieved through WRDS) and The Federal Reserve Bank of St. Louis Economic Data (FRED). The options under consideration are index call options on the S&P 500 and the Dow Jones Industrial Average from the Chicago Board of Options Exchange (Cboe) from the period 2010 to 2017. I include options on two different indices in order to examine how well the ANNs generalize to options on different indices. The options on these indices have European exercise style which allows for comparison to the BSM model. In addition, these options are highly

liquid and traded across a large assortment of strike prices and time-to-maturities, which is an important factor since ANNs require large amounts of data. I focus the analysis solely on call options while assuming that the put-call parity holds. In order words, the put-call parity can be used to transform call option prices into put options, and the model is, therefore, able to price both of these option types.

The dataset includes bid and ask prices, strike prices, close prices of the underlying indices, dividend yields, and maturity dates. The only missing BSM input is a proxy for the risk-free rate, which I instead obtain from FRED. Specifically, I choose the 1-month Treasury constant maturity rate with a daily frequency as a proxy for the risk-free rate, since it is generally considered one of the closest proxies of a risk-free investment.

I collect bid and ask prices at market close and calculate the mid option price as the average of the bid and ask prices. I calculate the time-to-maturity as the number of calendar days between the maturity date and the current date of an option price. OptionMetrics calculates the continuously-compounded dividend yield as the implied index dividend based on a regression of options and the put-call parity (OptionMetrics, 2015).

The data from OptionMetrics includes implied volatility, but since this measure is the volatility that makes the BSM option pricing formula equal to the observed market price, using the implied volatility of each option as input would essentially result in estimating the BSM model. Instead, the volatility input consists of volatility forecasts using several approaches as described in section 4.5.

I obtain financial time series of the assets under consideration to conduct these volatility forecasts. The data is obtained as close prices from OptionMetrics with a daily frequency for the period 2000 to 2017. I include data on the financial time series starting in 2000 because I want volatility predictions for the options in 2010 to be based on sufficient historical data. The returns are calculated without dividend adjustments, such that the dividend yield can be used as input to the ANN similar to the BSM option pricing model.

The option data is limited to options with a time-to-maturity up to two years while the moneyness of the options is restricted to the interval 0.8-1.5. In Chapter 5, I conduct a sensitivity analysis of the time-to-maturity and moneyness in order to identify any issues related to the option pricing capability of the ANNs and to assess whether the ANN can capture the volatility surface present in option markets.

The option data is described further in Figure 4.1, where the distribution of moneyness, time-to-maturity and the number of options per year is illustrated. Figure 4.1a shows that the majority of observations are distributed around near-the-money options with the number of options declining the further out-the-money the options become. Figure 4.1b shows that the time-to-maturity of the options is concentrated around the shorter maturities, while Figure 4.1c shows that the number of options available in the data increases considerably during the period, which could be due to a higher traded volume, increased data collection or a combination of both.

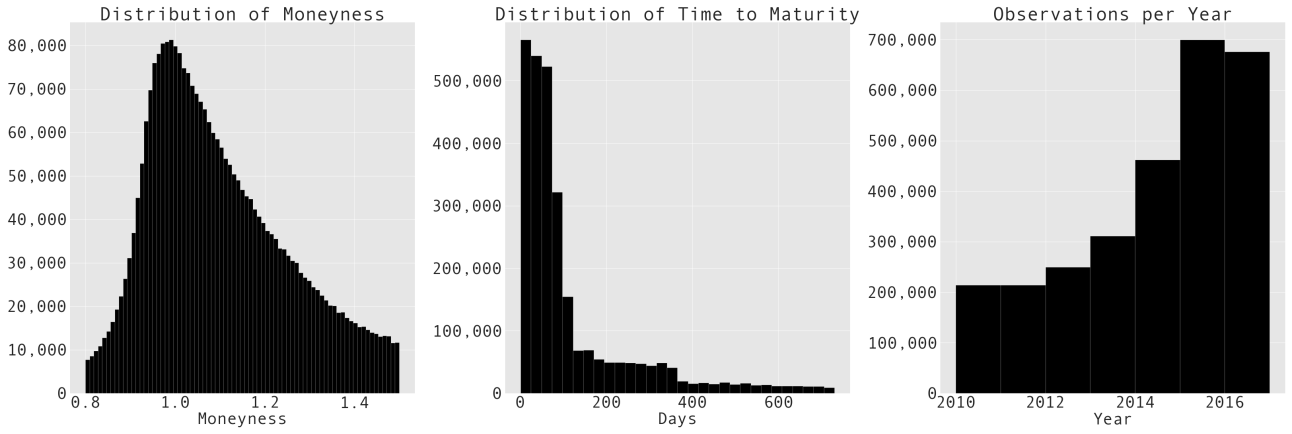

FIGURE 4.1: Histograms of the distribution of input data in the artificial neural network model.

4.2 Data Preprocessing

The data has been preprocessed in two important ways. Firstly, the data has been normalized by the strike price in accordance with previous literature on the subject. Secondly, feature scaling is applied in order to ensure that the input variables are on the same scale.

4.2.1 Dimensionality Reduction

Generally, ANNs tend to overfit training data resulting in poor out-of-sample predictions on the validation data. The primary reason for this is related to the typically large number of weights that need to be optimized during training. A simple method to reduce the dimensionality of ANNs is to reduce the number of inputs while maintaining the information within these input variables. By assuming the return distribution of the underlying asset is independent of the level of the underlying price of the asset, the BSM option pricing formula is homogeneous of degree one in both the price of the underlying and the strike price of the option (Hutchinson et al., 1994). This implies that we can focus on the moneyness of the option, S/K, while also dividing the price of the option by the strike price

$$\frac{C(S,K)}{K} = C(\frac{S}{K}, 1) \tag{4.1}$$

By doing this, the input variables are reduced by one resulting in optimization of fewer weights. Whereas exploiting this feature of the BSM model may not successfully translate to the 'market pricing formula', I keep the assumption since it is in agreement with previous literature and since it is reasonable from the perspective of financial option pricing theory.

4.2.2 Feature Scaling

Feature scaling is a commonly applied method in machine learning. It can take several forms but shared between them is the idea of scaling the features of the model – the input variables – to the same scale, such that adjustments to weights have comparable effects on all features. For instance, if one feature is 100 times larger than another, such as having the moneyness centered around 1, but the interest rate centered around 0.01, the adjustments to the weights in the gradient descent algorithm are uneven, and the effect of the interest rate diminishes.

An example of a feature scaling method commonly applied in financial theory is standardization, where the features are scaled to have zero mean and unit variance,

$$x_i = \frac{x_i - \bar{x_i}}{\sigma_i} \tag{4.2}$$

The underlying assumption of standardization is that each feature is normally distributed. This is often assumed in financial contexts, but it is an unrealistic assumption for the features of this model since the moneyness and time-to-maturity evidently are not normally distributed as seen in Figure 4.1a-b. An alternative feature scaling method is normalization, where the features are scaled to the interval [0,1],

$$x_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{4.3}$$

While this method has application to non-gaussian features, it is highly sensitive to outliers, since a single very large outlier will drag the feature scaled data towards this outlier. This issue is addressed in the robust scaling method.

Instead of relying on the minimum and maximum of the data like normalization does, robust scaling scales using the interquartile range. The interquartile range is the range between the first quartile (25th quantile) and the third quartile (75th quantile). This makes it robust to outliers. Concretely, the robust scaling method is

$$x_i = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{4.4}$$

where $Q_1(\cdot)$ is the first quartile and $Q_3(\cdot)$ is the third quartile. This feature scaling method resulted in the best results for the benchmark ANN and is used throughout the thesis for this reason.

4.3 Cross-Validation

It is common practice in predictive modeling to divide data into groups of training data and validation data in order to alleviate the issues of overfitting and to increase the generalizability as described in section 3.2. This concept is denoted cross-validation. The overall goal is to identify a model that generalizes well to new data by using only relevant signals and patterns in the data without overfitting by finding patterns in noise. Concretely, the data is split such that a percentage of the data is used to train the model, while the remainder is used to validate the model predictions out-of-sample. In this thesis, I will use a split of 90% for training and 10% for validation of the model. I will also refer to the validation data as out-of-sample data.

I apply an extension to the cross-validation method denoted k-fold cross-validation. In this approach, the dataset is randomly partitioned into k folds of approximately equal size, while letting a single fold be the validation data and estimating the model on the remaining k–1 folds (James, Witten, Hastie, & Tibshirani, 2017). Letting $k = 10$, I train the ANNs ten times, each time with a different fold behaving as the validation data. I then average the ten resulting loss functions to obtain a more precise estimate of the model's loss function, and I compute the standard deviation as an estimate for the variability of the model. I will refer to this method as 10-fold cross-validation. The primary benefit of the 10-fold cross validation method is that it provides insights into the bias and variance of a model. Applying the bias-variance tradeoff framework using the 10-fold cross-validation method allows for improved optimization of hyperparameters that result in a model with higher generalizability as described in section 4.4.

4.4 Optimization of Hyperparameters

Besides training the ANN, it is necessary to identify hyperparameters, which are parameters that are chosen before the training is initiated (Bergstra & Bengio, 2012). For instance, the number of layers and the number of perceptrons in each layer are examples of hyperparameters

that need to be chosen before the ANN can be trained and the parameters of the model can be optimized.

Hyperparameters affect the accuracy of the ANN to a large extent. In order to identify the optimal hyperparameters, I apply the grid search approach. In grid search, the ANN trains on all possible combinations of a specified subset of hyperparameters and the algorithm returns the optimal combination of hyperparameters with respect to the loss function (Bergstra & Bengio, 2012). I apply the 10-fold cross-validation method described in section 4.3 within the grid search algorithm such that the bias-variance tradeoff is considered in the optimization of hyperparameters.

I train the ANN and optimize hyperparameters on a Quadro P6000 commercial grade GPU from NVIDIA using cloud computing on Paperspace (*Paperspace*, 2019). GPUs can compute in parallel making them much faster at training ANNs than CPUs and the Quadro P6000 furthermore significantly outperforms the computing power of a personal computer resulting in faster optimization of hyperparameters and the ability to test more combinations. However, the ANNs still take a long time to train due to the large amount of data and the large number of parameters. Concretely, the benchmark ANN presented in Chapter 5 takes 2:30h to train once on the Quadro P6000, while the total uninterrupted training time for the entire analysis was approximately eleven days.

The subset of hyperparameters chosen for the grid search algorithm are:

$$\text{Perceptrons} = \{64, 128, 256\} \tag{4.5}$$

$$\text{Layers} = \{1, 2, 3\} \tag{4.6}$$

$$\text{Batch Sizes} = \{512, 1024, 4096\} \tag{4.7}$$

$$\text{Epochs} = \{50, 100, 200\} \tag{4.8}$$

The grid search with 10-fold cross-validation tries all possible combinations of the chosen hyperparameters ten times, leading to the ANN being trained $3^4 \times 10 = 810$ times. While the Quadro P6000 GPU is able to run parallel computations, the optimization of hyperparameters

still took approximately eleven days of uninterrupted training. As apparent, the dimensionality of the grid search approach increases rapidly as the number of tested hyperparameters increase. This is the reason I limit testing to four hyperparameters for three potential values.

I now present the optimal hyperparameters found using the grid search approach. The results apply to the Benchmark ANN, which is the optimized ANN based on the typical BSM input variables. In Chapter 5, I attempt to improve this Benchmark ANN by extending it to include additional input variables.

The ANNs in this thesis consist of three hidden layers consisting of 128 perceptrons each. The perceptrons in the hidden layers contain the 'ReLU' activation function, while the perceptron in the output layer contains the 'linear' activation function. There are five perceptrons in the input layer of the Benchmark ANN, which are 1) moneyness, 2) time-to-maturity, 3) volatility, 4) risk-free rate, and 5) dividend yield. I find the optimal batch size for the benchmark ANN to be 4096, while I find the optimal number of epochs to be 200.

Given the grid search algorithm finds the optimal number of epochs and the optimal number of layers to be the upper bound of the test values, the optimal hyperparameters are likely above these values. Despite this, I have decided to continue with these values due to constraints on time and costs of using cloud computing services.

4.5 Volatility Forecasting

The BSM model takes six inputs of which five are given exogenously, while the volatility input needs to be chosen with discretion. This implies that prediction of option prices is highly reliant on the chosen volatility estimate. Mandelbrot (1963) provides evidence that forecasting volatility is possible to a relatively high extent due to the high degree of conditional heteroscedasticity, or volatility persistence, present in most financial time series. Numerous methods for forecasting volatility are available in the literature, but only the most prominent methods are examined in this thesis. Concretely, the volatility forecasting methods under consideration are: historical volatility of different window lengths, GARCH(1,1), realized volatility, and the VIX index.

The models are examined as input into the Benchmark ANN in Chapter 5 and the best model is identified with respect to minimization of the mean absolute percentage error (MAPE) as described in section 4.8.

4.5.1 Historical Volatility

Annualized historical volatility is calculated from the return series of the financial assets using a 10-day, 30-day, and 60-day rolling window. The 30-day historical volatility is calculated as

$$\sigma_t = \sqrt{252}\sqrt{\frac{1}{T-1}\sum_{i=30}^{T-1}(r_{t-i} - \bar{r})^2} \tag{4.9}$$

and the 10-day and 60-day historical volatility are calculated analogously. The 10-day, 30-day and 60-day historical volatility are computed in order to evaluate the importance of shorter-term vs. longer-term volatility. This volatility forecasting model is the simplest and is considered the benchmark model to which the more advance methods are compared.

4.5.2 GARCH(1,1)

Generalized Autoregressive Conditional Heteroscedasticity, or GARCH, is commonly applied for option pricing and risk management purposes (Bollerslev, 1986). The model pulls the forecasted volatility towards an assumed long-term variance.

The GARCH(1,1) is the most commonly used variation of the GARCH(r, m) model, where r denotes the r most recent observations on r_t^2 and q denotes the q most recent estimates of variance. The GARCH(1,1) forecast of volatility is defined as

$$\sigma_t = \sqrt{\alpha_0 + \alpha_1 r_{t-1}^2 + \beta\sigma_{t-1}^2} \tag{4.10}$$

where α_0, α_1, and β are parameters from maximum likelihood optimization, σ_{t-1}^2 is the variance forecast one period ago, and r_{t-1}^2 is an estimate of variance under the assumption of stationarity.

The aim of maximum likelihood estimation is to identify the optimal parameters to fit a probability distribution to data. Given a time series of returns, $r_0, r_1, \ldots, r_{t-1}$, and the assumption that returns are normally distributed with an expected daily return of zero and a variance of σ_t^2, that is $N(0, \sigma_t^2)$, the variance simplifies to

$$\sigma_t^2 = \frac{1}{T-1}\sum_{t=1}^{T}(r_{t-1} - \bar{r})^2 \rightarrow \sigma_t^2 = \frac{1}{T-1}\sum_{t=1}^{T} r_{t-1}^2 \tag{4.11}$$

Following (Hull, 2018), the optimal parameters of the GARCH(1,1) are found as

$$\max_{\alpha_0,\alpha_1,\beta} \sum_{t=1}^{T}\left[-\ln(\sigma_t^2) \ - \frac{r_t^2}{\sigma_t^2}\right] \tag{4.12a}$$

$$\text{subject to } \alpha_1 + \beta < 1 \text{ and } a_0 > 0 \tag{4.12b}$$

Forecasts are generated recursively, such that the data used for estimating the GARCH(1,1) is expanding with time. In other words, data from $[1, t]$ is used to forecast $t + 1$, while data from $[1, t + 1]$ is used to forecast $t + 2$. To ensure that sufficient data is available to generate the first forecast of volatility on January 1, 2010, the data used to estimate the GARCH(1,1) begins ten years earlier on January 1, 2000. Finally, the forecasts are one-step-ahead forecasts, meaning the model recursively forecasts volatility one trading day ahead.

4.5.3 Realized Volatility

Realized volatility is a measure of volatility based on intraday returns sampled at a high frequency such as 5 min returns (Corsi, 2009). While the GARCH(1,1) has been widely accepted for volatility forecasting in the literature and in practice, Andersen, Bollerslev, Diebold, and Labys (2003) found that time series forecast of realized volatility significantly outperforms the GARCH(1,1) out-of-sample.

The standard measure of realized volatility is the square-root of the sum of squares of intraday log-returns. Given a set of equally-spaced intraday prices, $P_{t-j}\Delta$, the daily realized

volatility is defined as

$$RV_t^d = \sqrt{r_{t-j\Delta}^2} \tag{4.13}$$

where $\Delta = 1d/M, r_{t-j\Delta} = P_{t-j\Delta} - P_{t-(j+1)\Delta}$ is the continuously compounded intraday returns sampled at a frequency of Δ, and subscript j indexes the time within day t (Corsi, 2009). The daily realized volatility data is retrieved with a Δ-frequency of 5 min from the 'Realized Library' published by the Oxford-Man Institute at the University of Oxford (*Oxford-Man Realized Volatility Library*, 2019). The Oxford-Man Institute publishes realized volatility measures for 31 assets as of March 2019, including the S&P 500 and Dow Jones Industrial Average indices.

Realized volatility is not a forecast but rather a measure of the volatility realized within a day. According to a review of realized volatility forecast models by Wan Shin (2018), the heterogeneous autoregressive realized volatility model (HAR-RV) proposed by Corsi (2009) is the most prominent realized volatility forecast model in recent literature, and for this reason, I will focus on this model.

The HAR-RV attempts to model the effect of realized volatility at daily, weekly and monthly frequencies. It has been found to successfully reproduce some of the primary empirical characteristics of financial return series such as volatility persistence and fat tail distributions (Corsi, 2009). The heterogeneous autoregressive model of realized volatility (HAR-RV) is defined as

$$RV_{t+1} = \beta_0 + \beta_1 RV_t^d + \beta_2 RV_t^w + \beta_3 RV_t^m + \epsilon_{t+1} \tag{4.14}$$

where $RV_t^d = RV_t, \quad RV_t^w = \frac{1}{5}(RV_t + \ldots + RV_{t-4}), \quad RV_t^m = \frac{1}{22}(RV_t + \ldots RV_{t-21})$. The forecasts are conducted as one-step ahead recursive forecast corresponding to the method applied to the GARCH(1,1) forecasts.

4.5.4 VIX Index

The Chicago Board Options Exchange (Cboe) releases a volatility index commonly referred to as the 'fear gauge' or by its ticker: the VIX. It is a measure of the expected risk-neutral annualized volatility over the following 30 calendar days implied from mid-quote real-time

prices of S&P 500 index options (Cboe, 2019). The VIX is thus a forward-looking estimate of volatility, and it is often used by traders to assess the riskiness of the financial markets in the next 30 days. Among others, Cboe also releases a volatility index for the Dow Jones Industrial Average index called by its ticker, VXD.

The VIX and VXD are calculated according to

$$\sigma = \sqrt{\frac{2}{T}\sum_{i}\frac{\Delta K_i}{K_i^2}e^{rT}Q(K_i) - \frac{1}{T}\left[\frac{F}{K_0} - 1\right]^2} \tag{4.15}$$

where F is the forward index level derived from index option prices, T is the time to maturity of an option, r is the risk-free rate, K_0 is the first strike below the forward index level, K_i is the strike price of ith out-of-the-money option, $Q(K_i)$ is the midpoint of the bid-ask spread for each option with strike K_i, and $\Delta K_i = (K_{i+1} - K_{i-1})/2$ (Cboe, 2019).

4.6 Implied Volatility

The implied volatilities of the options in the data are found numerically using the Newton-Raphson minimization algorithm, since the BSM option pricing formula cannot be solved analytically for σ_{IV}. The Newton-Raphson method is an iterative algorithm for finding the root of differentiable functions and it is often used to extract the implied volatility from the BSM formulas. Concretely, our objective is to minimize the loss function

$$(c_{obs}(K,T) - c_{BS}(K,T,\sigma_{IV}))^2 \tag{4.16}$$

such that

$$c_{obs}(K,T) = c_{BS}(K,T,\sigma_{IV}) \tag{4.17}$$

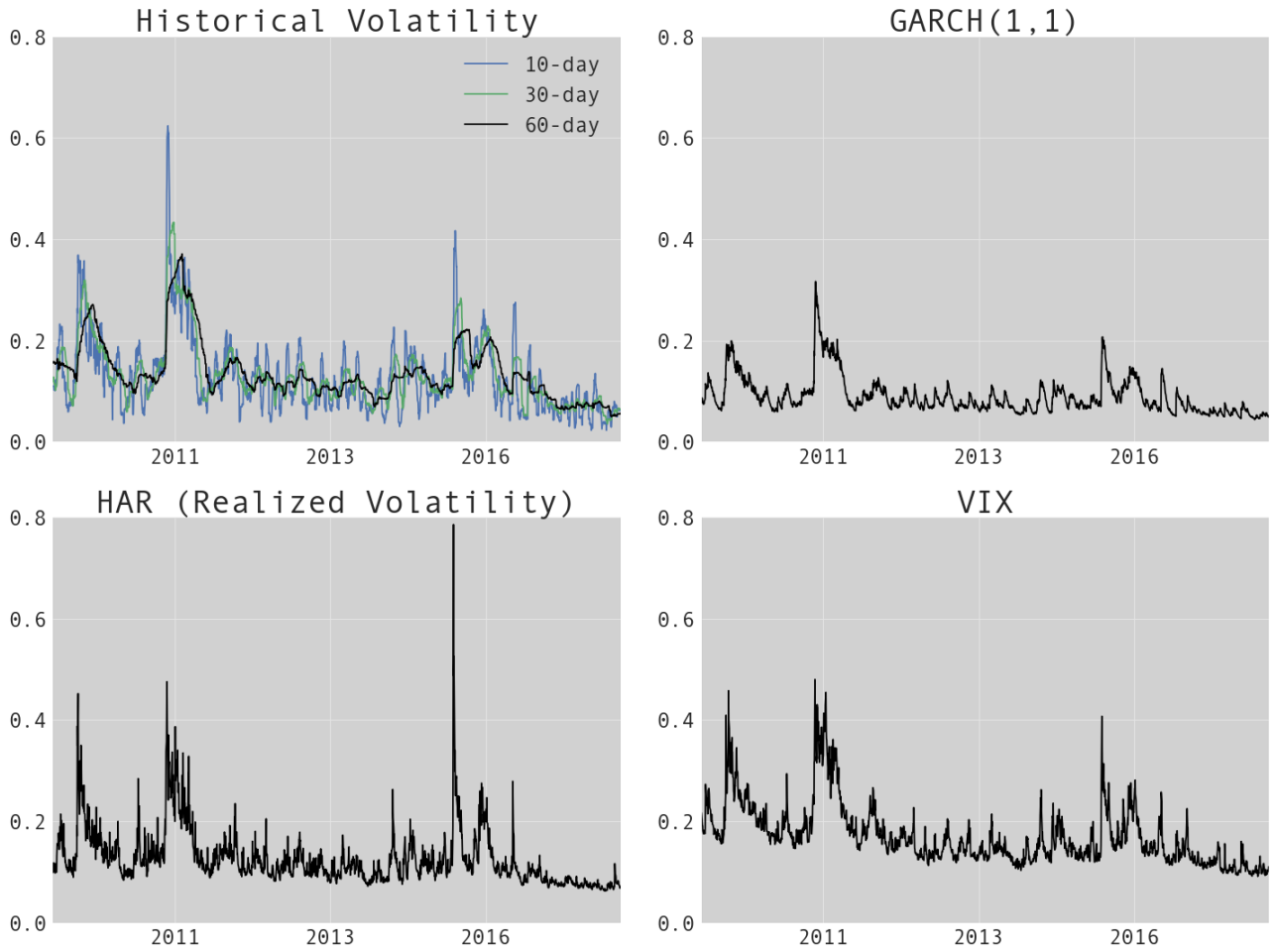

FIGURE 4.2: Time series plots of the four volatility forecasting models integrated into the artificial neural network model of this thesis.

Define x_0 as an initial guess of the root of the function $f(x_n)$ and $f'(x_n)$ as the derivative of the function $f(x_n)$, then the Newton-Raphson algorithm updates the initial guess as

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4.18}$$

The guess of the root of the function is updated iteratively according to

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.19}$$

until the loss function is minimized sufficiently. I choose the required precision of the loss function to be e-07. The partial derivative of the BSM formulas with respect to volatility is

called vega and it is

$$\nu = \frac{\delta c_{BS}}{\delta \sigma} = S_t e^{-q(T-t)} N'(d_1)\sqrt{T-t} \tag{4.20}$$

Replacing $f'(x_n)$ by ν simplifies computations, and the algorithm is able to converge efficiently with few iterations.

4.7 Practitioner Black-Scholes Model

As mentioned in section 2.9, the Practitioner Black-Scholes (PBS) model is the BSM model with volatility determined by a deterministic volatility function (DVF), which is a quadratic function of the strike price and time-to-maturity

$$\sigma_{IV} = \max\left(a_0 + a_1 K + a_2 K^2 + a_3 T + a_4 T^2 + a_5 KT, 0.01\right) \tag{4.21}$$

I obtain the parameters of the DVF using OLS regression on a cross-section of implied volatilities. I run the regressions on all available options in the training data during each trading day in the period 2010 to 2017. Next, I use the DVF to obtain predictions of implied volatility and pass these predictions into the BSM formula to obtain the PBS prices. The regressions are run on options in the training data only, while the validation period is used to test the out-of-sample performance of the PBS model.

4.8 Evaluation Measurements

In order to quantify the relative performance of different hyperparameters, volatility forecast models, etc., the following measurements are recorded: mean squared error (MSE) and mean absolute percentage error (MAPE). These measures are heavily applied in the literature making comparison possible. This is likely due to their nonparametric nature making them well-suited for evaluation of the nonparametric ANNs.

The MSE measure is used as the loss function when training the ANNs as described in section 3.7. However, it is difficult to interpret the mean of squared errors, so the MAPE is recorded alongside the MSE. The MAPE can be interpreted as the average percentage error regardless of the sign of errors. In other words, positive and negative errors do not cancel out but are instead taken as absolute errors. I consider MAPE as the primary evaluation measure when choosing the optimal volatility forecasting model, hyperparameters, etc. since MSE was used in the training process, while MAPE can be seen as a validation evaluation measurement. The mean squared error (MSE) is calculated as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.22}$$

and the mean absolute percentage error (MAPE) is calculated as

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.23}$$

4.9 Additional Option Pricing Variables

In order to capture the effect of demand pressure on option prices, I present four variables that I hypothesize contain information about the demand for and supply of options. I investigate whether either of these four variables has explanatory power in conjunction with the BSM inputs already present in the Benchmark ANN. The following section describes the economic rationale behind each variable, the source of the data, and the calculation methodology.

The first three variables are proxies of market participants' perception of recession risk while the fourth variable is a measure of inventory risk as measured by bid-ask spreads of options. The reason I have included three variables with a similar purpose is that no one variable is consistently able to predict a recession. Therefore, I include three variables that are all considered relatively good indicators of the perceived recession risk.

4.9.1 Cyclically Adjusted Price-Earnings (CAPE)

The CAPE ratio is the work of Robert Shiller and was first published in his book Irrational Exuberance (Shiller, 2005). It is a measure of the expensiveness of the S&P 500 index similar to how the price-earnings ratio is the expensiveness of individual stocks. It measures how much it costs to invest in the S&P 500 index per $1 of net income of the constituents of the index. Historically, the CAPE ratio has proved a valuable indicator of the expensiveness of the S&P 500, and thus it has been used as a predictor of market cycles and asset bubbles.

The CAPE is calculated as the price level of the S&P 500 index divided by the average inflation-adjusted earnings for the S&P 500 constituents over the last ten years (Shiller, 2005). It uses averages of earnings over ten years in order to smooth out short-term fluctuations and to focus on longer market cycles. I obtain data from Robert Shiller's website (Shiller, 2019) with a daily frequency.

Options are often used as a risk management tool particularly to hedge against market downturns. Therefore, the demand for and supply of put options is expectedly dependent on the market cycle meaning put options would increase in price when the market is relatively expensive, and the risk of a recession is increasing. This effect should, in theory, affect the prices of both call and put options due to the put-call parity mentioned in section 2.1

4.9.2 TED Spread

The TED spread is an indicator of credit risk in the U.S. economy as measured by the spread of interest rates on interbank loans over T-bills (Constable & Wright, 2011). A narrow TED spread indicates confidence in the economy since commercial banks are willing to lend to each other at a rate close to the risk-free rate. Vice versa, a wide TED spread indicates uncertainty as liquidity is being withdrawn resulting in increased rates on interbank loans. This typically results in an economic slowdown as the risk appetite of financial institutions decreases.

Concretely, the TED spread is measured in basis points and is calculated as the 3-month dollar-denominated LIBOR rate less the 3-month T-bill rate. I obtain data from FRED (FRED, 2019b) with a daily frequency. Historically, a TED spread between 10 and 50 bps has indicated confidence and economic growth, a claim supported by Boudt, Paulus, and Rosenthal (2017), who show how a level of the TED spread above 48 bps typically suggests that market participants perceive the economy to be in, or close to, a financial crisis. Recent examples of the TED spread widening beyond this 'safe' range include 1990, 2000, and 2008, where a recession followed shortly after. However, the TED spread has also provided false positives, where widening spreads were not followed by economic slowdown such as in 1987.

I hypothesize that the demand for put options as crash insurance increases when the TED spread widens due to the increased perceived credit risk in the economy implied by a widening TED spread. However, the effect of a widening TED spread may also affect the supply of these options by market makers, since market makers would be less willing to make markets in put options in such scenarios, where they themselves are worried about the risk of a recession.

4.9.3 Yield Curve

The Treasury yield curve illustrates the yield on Treasuries of different maturities (Constable & Wright, 2011). Under normal conditions, the yield curve is upward-sloping, implying a higher yield on longer-term Treasuries than on shorter-term Treasuries. However, periods of recessions have historically often been preceded by an inversion of the yield curve, meaning a higher yield on short-term Treasuries than on long-term Treasuries.

One economic explanation for the inversion of the yield curve is reinvestment risk. Notably, investors lose confidence in short-term investment opportunities and prefer to allocate funds at higher long-term yields. This mechanic drives a flattening and potentially inversion of the yield curve as the yield decreases on the long-term Treasuries in relatively high demand, while the yield increases on the short-term Treasuries in relatively low demand.

Importantly, inverted yield curves do not cause recessions but instead signal an increased perceived risk among investor and a flight-to-safety mentality. For instance, the recession of 1990 was preceded by an inverted yield curve, but the catalyst that actually started the recession was likely the Iraqi invasion of Kuwait and the associated increase in oil prices that followed (Andolfatto & Spewak, 2019). In other words, adverse events are more likely to cause a recession in times when the yield curve is inverted, and investors are cautious. A negative slope of the yield curve should thus be seen as an indicator of decreasing confidence among investors and as a signal of an increased risk of a recession similar to the TED spread.

In the literature, the slope of the yield curve is often measured as the difference between yields on 10-year Treasuries and 2-year Treasuries (Estrella & Mishkin, 1996). I obtain this data from FRED with a daily frequency using the '10-Year Treasury Constant Maturity Minus 2-Year Treasury Constant Maturity' data (FRED, 2019a).

The yield curve's historically high predictive power of recessions has caused traders and investors to follow any changes carefully. I hypothesize that this effect affects the pricing of options due to an increased demand for crash insurance and a decreased willingness of market makers to act as counterparties by supplying the increased demand.

4.9.4 Liquidity

Liquidity is a term used to describe how easy it is to buy and sell an asset with regards to price, transaction cost, and timeframe (Boudt et al., 2017). Highly liquid assets are characterized by low transaction costs, fast and easy access to buyers and sellers and the ability to buy and sell close to the intrinsic value of the asset.

Liquidity of stocks is commonly measured by the traded volume since this figure provides insight into how easy it is to find a buyer or seller at any given moment for a specific quantity of shares. However, the mechanic of option trading as well as the large range of different maturities and strike prices available for each option, makes the traded volume a bad measure of option liquidity. Indeed, options with a certain maturity and strike price can have no traded

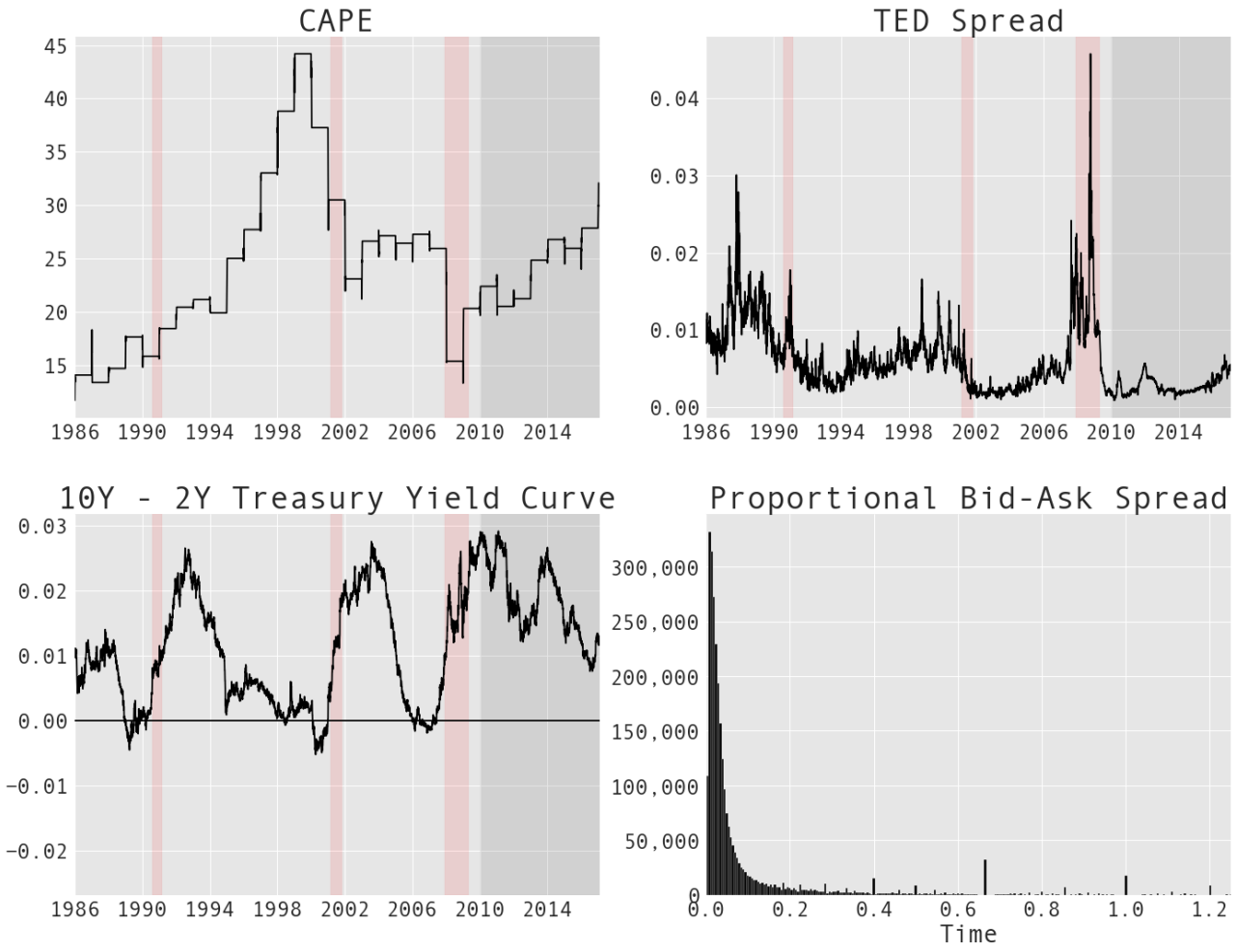

FIGURE 4.3: Illustrations of the additional variables in the artificial neural network model. The areas marked in dark grey are the period under consideration while the areas marked in red are global financial crises.

volume within a day and still be liquid due to a high willingness of market makers to make market. A large quantity of the options used in this analysis does in fact not have any traded volume during a whole day despite the S&P 500 index options being the most heavily traded options in the world (Fournier, 2013). Because options are derivative securities whose value derive from the underlying asset, the price of options does in fact change regardless of any trading activity.

Option prices are quoted by market makers, who provide liquidity by being prepared to take positions in the markets. The liquidity of options is thus the willingness of market makers to take the opposite side of a trade. Therefore, liquidity of options is most commonly

measured by the bid-ask spread, since it measures how willing market makers are to make market in specific options. Specifically, the more market makers are competing for trades on an option, the narrower the bid-ask spread becomes and the higher the liquidity of that option is.

The specific measure of option liquidity used in this thesis is the 'proportional bid-ask spread'. This representation of the bid-ask spread ensures that the bid-ask spread of options at different price scales can be compared as the bid-ask spread is transformed into a percentage spread. A large proportional bid-ask spread translates to a low liquidity option and vice versa. The proportional bid-ask spread is calculated as

$$\frac{\text{Ask Price} - \text{Bid Price}}{\text{Mid Price}} \tag{4.24}$$

where the mid price is defined as (Ask Price − Bid Price)/2.

The hypothesis underlying the inclusion of a liquidity measure in the ANN for option pricing is based on the inventory risk stemming from an imbalance between the supply and demand of options. Market makers will increase the bid-ask spread of an option when they are unwilling to supply the demand for an option. In this way, the effect of demand for and supply of options is directly affecting the bid-ask spread, and I hypothesize that the proportional bid-ask spread can increase the explanatory power of the Benchmark ANN by utilizing information regarding the liquidity of an option.

Chapter 5

Empirical Results

In this chapter, I present the empirical results of the ANN models compared to the BSM model and PBS model. Initially, I examine which volatility forecasting model results in the lowest pricing error in order to identify the Benchmark ANN. I then investigate how well the Benchmark ANN performs compared to the PBS and BSM in terms of capturing the volatility surface. Next, I assess whether there is any improvement from including additional option pricing variables to the Benchmark ANN model. Finally, I end the chapter with a discussion of the results including a discussion of real-world implications of the results. As mentioned in section 4.6, I focus the analysis on MAPE as opposed to MSE, since the MSE was used to train the ANNs, while the MAPE acts as a validation evaluation measure. In addition, I will focus the analysis on the S&P 500 index options and use the options on Dow Jones Industrial Average to assess how well the model generalizes to other index options.

5.1 Volatility Forecasting Model

The Benchmark ANN is the product of the optimized hyperparameters and the best performing volatility forecasting model. In Chapter 4, I found the optimal hyperparameters of the

Benchmark ANN to be: three hidden layers of 128 perceptrons with the ReLU activation function and a batch size of 4096. I apply the same method to the problem of finding the best performing volatility forecasting model. Specifically, I race the six volatility forecasting models against each other using the grid search approach to find the model that best price the options under consideration when added as an additional input to the ANN.

Table 5.1 contains the mean and standard deviation of the MSE and MAPE estimates resulting from the 10-fold cross-validation for each of the volatility forecasting models for S&P 500 (SPX) and Dow Jones Industrial Average (DJX) call options, respectively. I compare these estimates to the BSM model with the equivalent volatility forecast.

Volatility Dependence. Interestingly, the ANN model prevails the corresponding BSM model for all volatility forecasting models for both the SPX and DJX call options as measured by lower MSE and MAPE. In addition, the variation between volatility forecasting models of the ANN and the BSM is smaller suggesting that the ANN model is less reliant on the volatility parameter than the BSM model. For instance, the out-of-sample MAPE of the BSM model on DJX options lies in the interval from 25.57% to 48.15%, while the corresponding range for the ANN model is 10.82% to 14.46%. This observation demonstrates that the ANN model prevails the BSM model regardless of the volatility forecasting model used since the upper bound of the interval of the ANN model is lower than the lower bound of the interval of the BSM model. This observation is true for both DJX options and SPX options and both in the training and validation data. This initial result suggests that the ANN model prevails the BSM model regardless of the volatility forecasting model.

A likely explanation for the BSM model's higher dependence on volatility is related to a difference in how the models utilize the information available in input parameters. While the BSM model interprets all information contained within input parameters as relevant, the ANN model is able to assign weights to input parameters allowing it to cherry-pick relevant information from the inputs and disregard noise. At the same time, the ANN model can learn which parameters are most important from data while taking linear and nonlinear relationships between input parameters into account. On the other hand, volatility is the only input chosen

SPX	Mean Squared Error (MSE)				Mean Absolute Percentage Error (MAPE)			
	Training		Validation		Training		Validation	
	ANN	BSM	ANN	BSM	ANN	BSM	ANN	BSM
10-Day Hist	6.86E-07 (2.02E-07)	1.31E-04	6.96E-07 (2.02E-07)	1.30E-04	0.2087 (0.0280)	0.4970	0.2122 (0.0290)	0.5082
30-Day Hist	7.29E-07 (3.50E-07)	1.09E-04	7.40E-07 (3.50E-07)	1.07E-04	0.1994 (0.0218)	0.3921	0.2022 (0.0221)	0.3949
60-Day Hist	7.35E-07 (3.58E-07)	9.88E-05	7.41E-07 (3.45E-07)	9.73E-05	0.2339 (0.0808)	0.3627	0.2376 (0.0811)	0.3646
GARCH(1,1)	6.51E-07 (1.56E-07)	1.71E-04	6.66E-07 (1.56E-07)	1.69E-04	0.2287 (0.0862)	0.2476*	0.2326 (0.0873)	0.2483*
HAR-RV	6.41E-07 (2.53E-07)	1.23E-04	6.57E-07 (2.52E-07)	1.22E-04	0.2205 (0.0663)	0.2899	0.2233 (0.0661)	0.2911
VIX	5.22E-07* (1.96E-07)	5.95E-05*	5.30E-07* (1.96E-07)	5.89E-05*	0.1293* (0.0386)	0.7073	0.1313* (0.0385)	0.7122

DJX	Mean Squared Error (MSE)				Mean Absolute Percentage Error (MAPE)			
	Training		Validation		Training		Validation	
	ANN	BSM	ANN	BSM	ANN	BSM	ANN	BSM
10-Day Hist	3.3E-6 (6.0E-7)	2.3E-4	3.4E-6 (5.9E-7)	2.3E-4	0.1211 (0.022)	0.4096	0.1197 (0.021)	0.4112
30-Day Hist	3.8E-6 (2.0E-6)	1.9E-4	3.8E-6 (2.0E-6)	1.9E-4	0.1474 (0.079)	0.3366	0.1446 (0.076)	0.3334
60-Day Hist	3.1E-6 (7.7E-7)	1.7E-4	3.1E-6 (7.7E-7)	1.7E-4	0.1244 (0.0374)	0.2924	0.1216 (0.0366)	0.2894
GARCH(1,1)	4.1E-6 (2.7E-6)	3.0E-4	4.1E-6 (2.7E-6)	3.0E-4	0.1183 (0.036)	0.2576*	0.1169 (0.034)	0.2557*
HAR-RV	3.4E-6 (7.3E-7)	2.0E-4	3.5E-6 (7.4E-7)	1.9E-4	0.1278 (0.049)	0.3041	0.1257 (0.048)	0.2926
VXD	2.6E-6* (3.5E-7)	1.0E-4*	2.7E-6* (3.5E-7)	1.1E-4*	0.1092* (0.0193)	0.4930	0.1082* (0.0185)	0.4815

TABLE 5.1: Comparison of volatility forecasting models in the artificial neural network (ANN) model and the Black-Scholes-Merton (BSM) model.

with discretion in the BSM model whereas all other input parameters are given exogenously implying that the volatility parameter of the BSM model is essential.

Bias-Variance Tradeoff. The argument presented in the paragraph above is only valid if the ANN is well-trained in relation to neither underfitting nor overfitting the data. I examine this by analyzing the out-of-sample performance of the ANNs using the cross-validation method mentioned in section 4.3 and the bias-variance tradeoff mentioned in section 3.2.

The MSE and MAPE are comparable across the training and validation data for both SPX and DJX options. This observation suggests that the ANNs generalize well to unseen data implying that the models do not suffer from overfitting. This is a good sign considering the tendency of ANNs to overfit data due to the large number of parameters. Additionally, the ANN model appears to generalize well to different index options as seen from the (even lower) pricing errors on the options of the Dow Jones Industrial Average index.

The mean and standard deviation of the MSE and MAPE from the 10-fold cross-validation present in Table 5.1 can be seen as measures of the bias and variance of the models, respectively. While the ANN model has lower biases compared to the BSM model, the ANN model introduces variance as a result of the complex optimization process underlying the training of the model. The variance of the ANN model means that the predicted price of the same option may change each time the model is trained; an undesirable feature that we wish to minimize. On the other hand, the BSM model always provides the same option price for a given set of input variables.

The bias-variance tradeoff is, as implied by its name, a tradeoff between bias and variance. The question is therefore whether the lower bias of the ANN model more than offsets the adversarial effect of the variance they introduce. The standard deviation of the out-of-sample MAPE estimates resulting from the 10-fold cross-validation of the ANNs is between 2.21% and 8.73% for the SPX options and between 1.85% and 7.60% for the DJX options. To avoid information overload, I will only highlight the most interesting insights from Table 5.1 regarding the bias-variance tradeoff.

For the SPX options, the MAPE of the ANN model with GARCH(1,1) volatility is only marginally lower than the corresponding BSM model, but the standard deviation is the highest among all models at 8.73%; a bad tradeoff between bias and variance. On the other hand, the ANN model with VIX (for SPX options) and VXD (for DJX options) volatility introduces only a modest standard deviation in relation to the substantially lower bias as measured by MAPE. Indeed, the VIX/VXD volatility forecast results in the lowest bias of the six volatility forecasting models, and I define the volatility of the Benchmark ANN as VIX and VXD for the SPX and DJX options, respectively. Specifically, the Benchmark ANN for SPX options achieves an out-of-sample MAPE of 13.13% with a standard deviation of 3.85% while the Benchmark ANN for DJX options achieves an out-of-sample MAPE of 10.82% with a standard deviation of 1.85%. The key takeaway from this result is that the ANN model can price options with a substantially lower bias than the BSM model while only introducing a modest variance; a promising result for the ANN model. In reference to the research question of this thesis, namely whether the ANN model is able to price options better than traditional methods, the initial results are in favor of the ANN model.

Benchmark BSM Model. Interestingly, the BSM model with VIX/VXD volatility has the highest MAPE of all the volatility forecasting models in contrast to the ANN model. Concretely, the out-of-sample MAPE is 71.22% for the BSM model with VIX as volatility for SPX options, while it is 48.15% for DJX options with VXD as volatility in the BSM model. Instead, the BSM model with GARCH(1,1) volatility achieves the lowest MAPE for both SPX and DJX options. For this reason, I define the first comparison model to the Benchmark ANN as the BSM model with GARCH(1,1) volatility.

The reason for this discrepancy between which volatility forecasting model performs best in the ANN model and the BSM model is challenging to know with certainty. On the one hand, since the VIX/VXD volatility measure is created from implied volatilities of index options on SPX and DJX options, it is not surprising that it is useful for option pricing. On the other hand, option prices are likely to reflect that the GARCH(1,1) volatility forecasting model is widely used in practice (Rouah & Vainberg, 2007). Seemingly the VIX/VXD measure of future

volatility is not directly appropriate in the BSM model, but the ANN model manages to extract useful information from this measure despite.

While Table 5.1 shows evidence of improvements of the ANN model to the BSM model, it also conveys the message that there is room for improvement. MAPEs of 13.13% and 10.82% for the Benchmark ANN on SPX and DJX options, respectively, is substantially better than the BSM model but it is still a sizeable error from an economic perspective. I will discuss the real-world implication of the results further in section 5.4.

5.2 Benchmark Artificial Neural Network

The ANNs contain 50,433 weights applied to nonlinear activation functions making them extremely difficult, if not impossible, to understand and interpret. Instead, I investigate how reliable the model predictions are in order to understand under which circumstances the model predictions are reliable and under which circumstances the model has problems. Concretely, I initiate this section by investigating the precision of the ANNs as measured by linear regression of the predicted prices against the observed prices and compare the results to Hutchinson et al. (1994). Next, I investigate how well the ANNs are able to incorporate the volatility surface by investigating how well the ANNs price options of different moneyness and time-to-maturity.

5.2.1 Regression

Following in the footsteps of Hutchinson et al. (1994), the first paper to explore the use of machine learning to price options, I present an ordinary least squares (OLS) regression of the predicted option prices by the Benchmark ANN against the observed market prices for SPX options in Table 5.2. In addition, I illustrate the predicted option prices in a scatter plot against the observed market prices in Figure 5.1.

The fitted line of the OLS regression has an intercept of 0.0579, a slope of 1.0006 and an R^2 of 0.99997 for the SPX options and approximately the same for the DJX options. The

slope and R^2, both close to one, suggest that the model is making very accurate predictions, whereas the intercept of approximately 0.0579 suggests that the Benchmark ANN consistently prices options a bit too expensive. However, seen relative to option prices ranging from \$0 to \$800, the mispricing of approximately five cents is trivial.

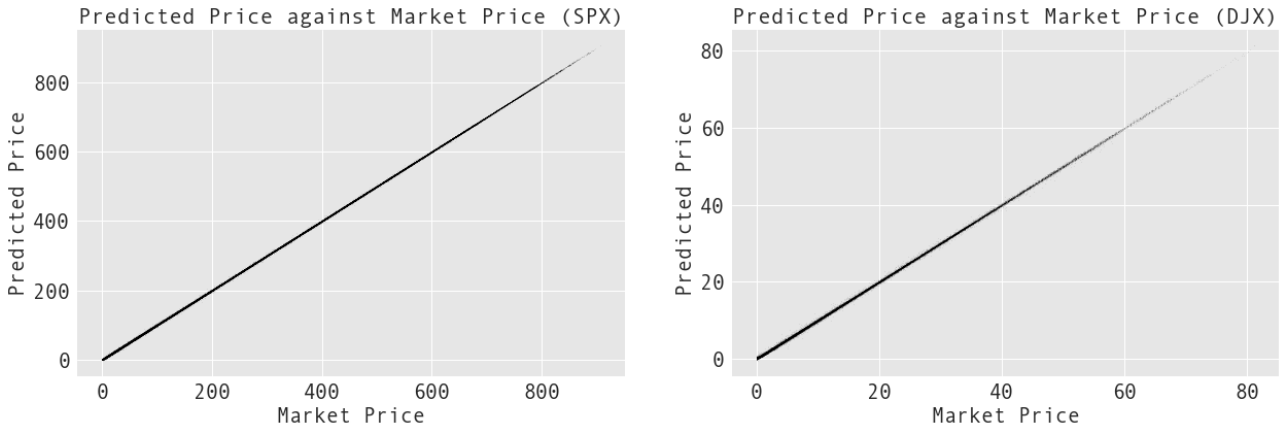

FIGURE 5.1: Scatter plot of the predicted option pricing against the observed option price.

R^2	0.999974
# obs	360,348

	Coefficient	Std. Err.	t-test	p-value	95% confidence interval	
Constant	0.0579	0.002	23.485	0.0000	0.053	0.063
Slope	1.0006	8.44E-06	1.19E+05	0.0000	1.001	1.001

TABLE 5.2: Regression output from regressing the predicted option pricing against the observed option price.

The Benchmark ANN developed in this thesis achieves a higher R^2 than the one of Hutchinson et al. (1994) of 95.53% suggesting the Benchmark ANN can price options more accurately. This improvement is not surprising given the exponential growth in computing power and the vastly increased data availability; possibly the two most important success factors of ANNs.

The distribution of percentage residuals of the ANN models, presented in Figure 5.2, exhibit a bell curve shape with a visibly higher kurtosis than the normal distribution. Approximately 94% of options are priced with an error of less than 5% while approximately 88% of

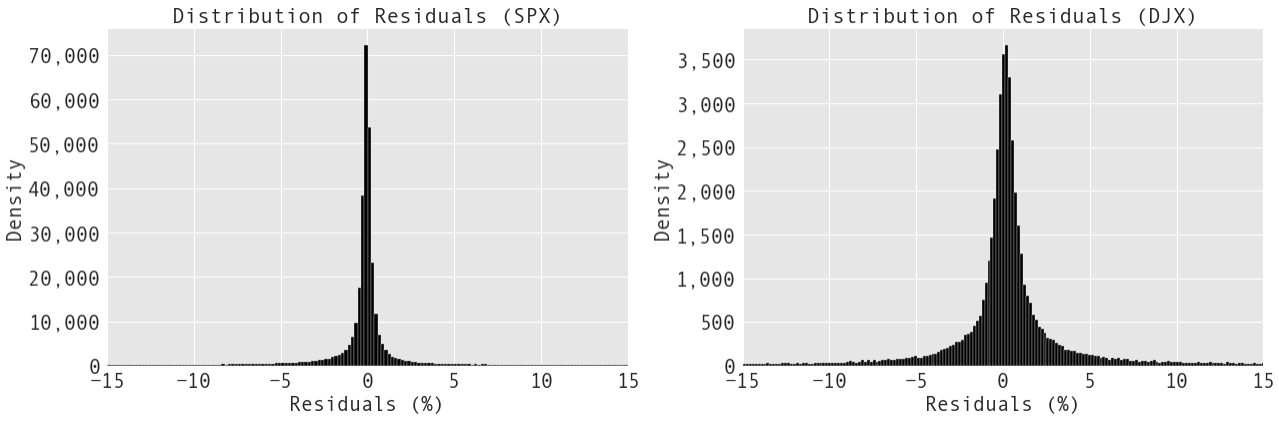

FIGURE 5.2: Distribution of percentage residuals of the artificial neural network model.

the options are priced with an error of less than 1%. This result suggests that the ANN model price the majority of options with a reasonably small percentage error from the true observed market price.

However, the ANN model has some outliers that negatively affect overall pricing errors. Concretely, 1% of options are priced with an error of greater than 38% while 0.5% of percentage residuals exceed 57%. These outlier residuals were not visible in the histograms in Figure 5.2, and I have therefore restricted the x-axis to ± 15%.

To summarize, the percentage residuals imply that almost nine out of ten options are priced with a percentage error of 1% or less; a promising result. However, the outliers are sizeable, and the model seems to have issues, whose origin I will examine in further detail in the next section.

5.2.2 Moneyness

Table 5.3 shows the performance of the ANN model and the two comparison models for different moneyness for SPX and DJX index call options. For both SPX and DJX options, the PBS model exhibits lower pricing errors than the BSM model, while the ANN model exhibits substantially lower pricing errors than both the PBS and BSM. This result is consistent across all moneyness and the training and validation data.

SPX	Mean Squared Error (MSE)						Mean Absolute Percentage Error (MAPE)					
	Training			Validation			Training			Validation		
	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM
FOTM: 0.80 - 0.90	3.2E-7	1.6E-4	5.1E-5	3.5E-7	1.6E-4	5.2E-5	0.3931	0.8241	0.9290	0.4006	0.8279	0.9357
OTM: 0.90 - 0.98	2.2E-7	3.0E-5	9.2E-5	2.3E-7	3.0E-5	9.3E-5	0.3582	0.7061	0.7371	0.3630	0.7033	0.7382
ATM: 0.98 - 1.02	3.7E-7	2.8E-5	1.9E-4	3.8E-7	2.7E-5	1.9E-4	0.0974	0.3233	0.4038	0.1017	0.3231	0.4038
ITM: 1.02 - 1.10	3.5E-7	2.2E-5	2.4E-4	3.5E-7	2.2E-5	2.4E-4	0.0070	0.0354	0.1288	0.0070	0.0354	0.1284
DITM: 1.10 - 1.50	4.9E-7	1.6E-5	1.7E-4	4.9E-7	1.5E-5	1.7E-4	0.0022	0.0097	0.0289	0.0022	0.0095	0.0285
ALL: 0.80 - 1.50	5.2E-7	2.9E-5	1.7E-4	5.3E-7	2.9E-5	1.7E-4	0.1293	0.2303	0.2476	0.1313	0.2294	0.2483
DJX	Mean Squared Error (MSE)						Mean Absolute Percentage Error (MAPE)					
	Training			Validation			Training			Validation		
	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM
FOTM: 0.80 - 0.90	8.9E-7	7.5E-5	5.9E-5	9.3E-7	8.3E-5	6.0E-5	0.3531	0.6728	0.8607	0.3407	0.6965	0.8783
OTM: 0.90 - 0.98	1.0E-6	2.4E-5	2.0E-4	1.0E-6	2.5E-5	2.1E-4	0.2655	0.4360	0.7202	0.2554	0.4435	0.7183
ATM: 0.98 - 1.02	1.4E-6	2.1E-5	3.7E-4	1.5E-6	2.1E-5	3.8E-4	0.0954	0.2542	0.4546	0.0996	0.2682	0.4606
ITM: 1.02 - 1.10	1.7E-6	1.8E-5	5.3E-4	1.8E-6	1.7E-5	5.3E-4	0.0149	0.0368	0.1963	0.0154	0.0359	0.1957
DITM: 1.10 - 1.50	2.4E-6	1.9E-5	4.4E-4	2.4E-6	1.9E-5	4.5E-4	0.0049	0.0118	0.0577	0.0050	0.0120	0.0585
ALL: 0.80 - 1.50	2.6E-6	2.5E-5	3.0E-4	2.7E-6	2.6E-5	3.0E-4	0.1092	0.1782	0.2576	0.1082	0.1850	0.2557

TABLE 5.3: Evaluation of the three models in the training and validation data on S&P 500 and Dow Jones Industrial Average index options per moneyness.

The three models have in common that the pricing error as measured by MAPE is a decreasing function of moneyness. Said differently, the more in-the-money (ITM) an option becomes, the better able are all three models of pricing the option. One cause of this result is likely the fact that option prices under Black-Scholes assumptions approach $S - Ke^{-r(T-t)}$ in the limit as $S \to \infty$ as mentioned in section 2.6. The effect of this is that option prices start to behave linearly as the moneyness increases.

This feature of option prices is also evident from Figure 5.3 that shows the predicted option price from the ANN model on SPX options and the observed SPX option prices both plotted against moneyness. From these plots, it is apparent that both observed option prices

and predicted option prices behave linearly for higher moneyness, while at-the-money options show the greatest variation.

FIGURE 5.3: Predicted option prices and observed option prices both plotted against moneyness.

Another reason for this behavior of the ANN model could be related to the composition of the training data used to train the model, namely that approximately 68% of the SPX options are ITM or DITM while this number is approximately 65% for the DJX options as seen in Table 5.4. This overrepresentation of ITM and DITM options in the data indicates that the ANNs have an incentive to focus on minimizing the pricing error of ITM or DITM options since this results in the most considerable reduction of the overall error.

Count (% of Total)	SPX		DJX	
	Training	Validation	Training	Validation
FOTM: 0.80 - 0.90	135,618 (4%)	15,031 (4%)	35,812 (7%)	3,948 (7%)
OTM: 0.90 - 0.98	536,347 (17%)	59,911 (17%)	76,586 (16%)	8,463 (16%)
ATM: 0.98 - 1.02	375,513 (12%)	41,799 (12%)	56,063 (12%)	6,293 (12%)
ITM: 1.02 - 1.10	668,129 (21%)	73,647 (20%)	96,604 (20%)	10,756 (20%)
DITM: 1.10 - 1.50	1,527,413 (47%)	169,947 (47%)	221,289 (45%)	24,572 (45%)
ALL: 0.80 - 1.50	3,243,020	360,335	486,354	54,032

TABLE 5.4: Composition of data per group of moneyness in the training and validation data for the options under consideration in this thesis.

The Benchmark ANN model thus captures the volatility smirk of index options better than the comparison models as seen by consistently lower pricing error across all moneyness.

However, the pricing error is still a decreasing function of moneyness implying that the model has issues related to pricing out-the-money options.

5.2.3 Time-To-Maturity

Table 5.5 shows the pricing errors of the three models on SPX and DJX option for different groups of time-to-maturity up to two years. The ANN model prevails the two comparison models for almost all time-to-maturities as measured by both MSE and MAPE.

SPX	Mean Squared Error (MSE)						Mean Absolute Percentage Error (MAPE)					
	Training			Validation			Training			Validation		
	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM
0-14 days	2.7E-7	1.5E-6	1.8E-6	2.7E-7	1.6E-6	1.8E-6	0.1964	0.5021	0.1565	0.2013	0.4906	0.1577
14-28 days	2.2E-7	2.8E-6	5.6E-6	2.3E-7	2.7E-6	5.5E-6	0.1186	0.3564	0.2065	0.1197	0.3513	0.2067
28-90 days	2.6E-7	3.8E-6	3.0E-5	2.6E-7	3.8E-6	2.9E-5	0.0645	0.1531	0.2263	0.0643	0.1538	0.2274
90-365 days	6.2E-7	3.3E-5	3.1E-4	6.4E-7	3.1E-5	3.1E-4	0.0516	0.1588	0.3487	0.0529	0.1588	0.3497
365-550 days	1.3E-6	1.3E-4	1.3E-3	1.4E-6	1.3E-4	1.3E-3	0.0224	0.2080	0.4489	0.0244	0.2018	0.4474
550-730 days	1.8E-6	5.5E-4	2.5E-3	1.9E-6	5.7E-4	2.5E-3	0.0168	0.6407	0.4933	0.0179	0.6706	0.4957
ALL	5.2E-7	2.9E-5	1.7E-4	5.3E-7	2.9E-5	1.7E-4	0.1293	0.2303	0.2476	0.1313	0.2294	0.2483
DJX	Mean Squared Error (MSE)						Mean Absolute Percentage Error (MAPE)					
	Training			Validation			Training			Validation		
	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM	ANN	PBS	BSM
0-14 days	1.4E-6	2.1E-6	2.8E-6	1.6E-6	2.1E-6	2.7E-6	0.1600	0.3073	0.1026	0.1632	0.3319	0.1070
14-28 days	1.4E-6	3.1E-6	8.1E-6	1.4E-6	3.1E-6	7.5E-6	0.1311	0.2504	0.1284	0.1326	0.2707	0.1304
28-90 days	1.4E-6	5.9E-6	4.3E-5	1.4E-6	5.8E-6	4.3E-5	0.1265	0.1437	0.2261	0.1202	0.1464	0.2233
90-365 days	2.2E-6	2.0E-5	4.6E-4	2.4E-6	1.9E-5	4.6E-4	0.0793	0.1390	0.4007	0.0823	0.1383	0.4046
365-550 days	4.5E-6	5.9E-5	1.4E-3	4.6E-6	5.8E-5	1.4E-3	0.0667	0.1556	0.4781	0.0677	0.1557	0.4858
550-730 days	6.2E-6	1.9E-4	2.4E-3	6.7E-6	2.1E-4	2.5E-3	0.0488	0.4457	0.5378	0.0489	0.4922	0.5363
ALL	2.6E-6	2.5E-5	3.0E-4	2.7E-6	2.6E-5	3.0E-4	0.1092	0.1782	0.2576	0.1082	0.1850	0.2557

TABLE 5.5: Evaluation of the three models in the training and validation data on S&P 500 and Dow Jones Industrial Average index options per time-to-maturity.

Interestingly, the ANN model is a decreasing function of time-to-maturity while the opposite is true for the BSM model. The MAPE of the PBS model is instead decreasing for time-to-maturities up to 90 days, while it is increasing for time-to-maturities between 90 and 730 days. Precistly what causes this difference in pricing errors across time-to-maturities for the three models is difficult to say based on Table 5.5. Nevertheless, it is an important insight into the pricing capabilities of the models as it shows how confident one should be on an option price prediction given by each model for a given time-to-maturity.

While the PBS model is modeling the volatility term structure observed in market prices, the benefits of doing so seem negligible compared to the BSM with GARCH(1,1) volatility. Both of these models have a large variation in MAPE between different time-to-maturity, and this inconsistency in pricing across time-to-maturity is an undesirable feature. The ANN is seemingly more reliable in pricing options of different time-to-maturity, but it does have issues related to pricing shorter-term options.

The conclusion from Table 5.5 is therefore that the ANN model is better able to capture the volatility term structure of index options than the comparison models, while the model still has some issues related to pricing of shorter-term options.

In order to conclude on the ANN models' ability to incorporate the volatility surface, I have prepared a sensitivity analysis in Table 5.6 that combines the moneyness and time-to-maturity and evaluates the Benchmark ANN model on out-of-sample MAPE for SPX options.

The general pattern apparent from Table 5.6 is that the ANN model prices options of longer time-to-maturity and higher moneyness better. The highest pricing error occurs for the options of the shortest time-to-maturity that are the most out-of-the-money, while the DITM options appear to be consistently priced across different time-to-maturities with an out-of-sample MAPE of approximately 0.7%.

		Moneyness				
		0.80 - 0.90	0.90 - 0.98	0.98 - 1.02	1.02 - 1.10	1.10 - 1.50
Time-To-Maturity	0-14 days	3.5380	1.3981	0.3008	0.2013	0.0078
	14-28 days	1.5938	0.5767	0.0583	0.1197	0.0068
	28-90 days	0.7339	0.2339	0.0643	0.0263	0.0064
	90-365 days	0.3280	0.0744	0.0529	0.0155	0.0076
	365-550 days	0.0992	0.0230	0.0244	0.0131	0.0077
	550-730 days	0.0572	0.0205	0.0180	0.0117	0.0080

TABLE 5.6: Sensitivity analysis of the artificial neural networks ability to capture the volatility surface.

5.3 Additional Option Pricing Variables

After analyzing the benchmark ANN, henceforth denoted the ANN (B), I continue to an analysis of whether additional variables can increase the explanatory power of the model and help predict option prices. I call this new model with an additional predictive variable the ANN (A). As mentioned in section 4.9, the four variables are hypothesized to capture the effect of supply and demand on option prices.

I use the same approach to evaluate the explanatory power of the additional option pricing variables as I used to identify the best volatility forecasting model: grid search with 10-fold cross-validation. I retrain the ANNs using the additional input variable and report the results in Table 5.7. Since I already showed that the ANN (B) prevails the BSM model and the PBS model, I will only compare the ANN (A) to the ANN (B) to evaluate whether there is any added explanatory power to the ANN (B) from including the additional option pricing variables.

CAPE. Table 5.7 show little to no improvement in the model bias from including the CAPE ratio to the ANN (B) implying that the ANN struggles to identify relevant information from the CAPE ratio. Instead, the CAPE ratio introduces substantially more variance to the 10-fold cross-validation estimates. These results are consistent across the SPX and DJX options.

SPX	Mean Squared Error (MSE)				Mean Absolute Percentage Error (MAPE)			
	Training		Validation		Training		Validation	
	ANN (A)	ANN (B)	ANN (A)	ANN (B)	ANN (A)	ANN (B)	ANN (A)	ANN (B)
CAPE	2.75E-07	5.22E-07	2.81E-07	5.30E-07	0.1248	0.1293	0.1265	0.1313
	(1.34E-08)	(1.96E-07)	(1.23E-08)	(1.96E-07)	(0.0791)	(0.0386)	(0.0803)	(0.0385)
TED Spread	3.57E-07	5.22E-07	3.62E-07	5.30E-07	0.1587	0.1293	0.1610	0.1313
	(1.98E-07)	(1.96E-07)	(1.98E-07)	(1.96E-07)	(0.1019)	(0.0386)	(0.1025)	(0.0385)
Liquidity	3.39E-07	5.22E-07	3.45E-07	5.30E-07	0.0881	0.1293	0.0891	0.1313
	(2.04E-08)	(1.96E-07)	(2.08E-08)	(1.96E-07)	(0.0106)	(0.0386)	(0.0107)	(0.0385)
Yield Curve	2.97E-07	5.22E-07	3.01E-07	5.30E-07	0.1132	0.1293	0.1148	0.1313
	(1.19E-07)	(1.96E-07)	(1.19E-07)	(1.96E-07)	(0.0394)	(0.0386)	(0.0400)	(0.0385)
DJX	Mean Squared Error (MSE)				Mean Absolute Percentage Error (MAPE)			
	Training		Validation		Training		Validation	
	ANN (A)	ANN (B)	ANN (A)	ANN (B)	ANN (A)	ANN (B)	ANN (A)	ANN (B)
CAPE	2.35E-06	2.60E-06	2.48E-06	2.70E-06	0.1034	0.1092	0.1033	0.1082
	(2.04E-07)	(3.5E-7)	(2.04E-07)	(3.5E-7)	(0.0316)	(0.0193)	(0.0304)	(0.0185)
TED Spread	2.28E-06	2.60E-06	2.41E-06	2.70E-06	0.0960	0.1092	0.0961	0.1082
	(9.36E-08)	(3.5E-7)	(8.97E-08)	(3.5E-7)	(0.0199)	(0.0193)	(0.0188)	(0.0185)
Liquidity	2.28E-06	2.60E-06	2.42E-06	2.70E-06	0.0811	0.1092	0.0810	0.1082
	(2.04E-07)	(3.5E-7)	(1.94E-07)	(3.5E-7)	(0.0101)	(0.0193)	(0.0089)	(0.0185)
Yield Curve	2.25E-06	2.60E-06	2.37E-06	2.70E-06	0.0887	0.1092	0.0892	0.1082
	(8.90E-08)	(3.5E-7)	(8.87E-08)	(3.5E-7)	(0.0188)	(0.0193)	(0.0180)	(0.0185)

TABLE 5.7: Evaluation of the additional option pricing variables in the artificial neural network model in comparison to the Black-Scholes-Merton model.

This result implies that the expensiveness of the stock market, specifically the S&P 500 index, does not affect the pricing of options according to the ANN model. One reason for this result could be that the CAPE ratio contains little to no relevant information for the pricing of options. However, the black-box nature of ANNs implicates that such conclusions are challenging to make with certainty. Another reason could be that the ANN fails to extract the relevant information. Instead, I can conclude that the CAPE ratio shows little predictive power of option prices for the given specifications of the benchmark ANN in terms of hyperparameters and the given period under consideration.

TED Spread. The results of including the TED spread in the ANN (B) is conflicting across SPX and DJX options. For the SPX options, the TED spread substantially worsens

both the bias and the variance of the model. For the DJX options, the TED spread slightly improves the bias with a comparable variance to the ANN (B). However, I do not consider these conflicting results convincing enough to conclude that the TED spread increases the explanatory power of the ANN (B).

This result is unexpected since the economic rationale for the TED spread having an influence on option prices, based on the notion of supply/demand option pricing, is convincing. Especially because the TED spread can be interpreted as a measure of the risk appetite among commercial and investment banks, who are the typical dealers of options, and who would be expected to limit the supply of options in times when they are more constrained. A likely explanation for this could be that the ANN model fails to extract useful information from the TED spread in the period under consideration. Specifically, the period from 2010 to 2017 does not include any U.S. recessions causing the TED spread to behave relatively steady as apparent from Figure 4.3. The ANNs simply may not have had enough relevant data during the period under consideration to understand and interpret the signals from the TED spread.

Liquidity. The liquidity measure considerably lowers the bias and the variance of the ANN (B) for both SPX and DJX options. Concretely, the out-of-sample MAPE is 8.91% for SPX options while it is 8.10% for DJX options both with a standard deviation of close to 1% suggesting a beneficial tradeoff between bias and variance. These MAPEs should be seen in relation to an out-of-sample MAPE of 18.50% for the PBS model and 25.57% for the BSM model; a substantial improvement in empirical option pricing to traditional methods.

Seemingly, option prices are affected by the proportional bid-ask spread used in this thesis as a measure of option liquidity. This implies that the size of the bid-ask spread influences the mid-price of an option, likely due to an imbalance between market demand and supply. Concretely, market makers increase the bid-ask spread of an option when they are unwilling to supply the demand for an option, thus increasing the transaction costs.

However, increased transaction costs are not sufficient to explain the improvements of including the liquidity measure in the ANN (A) model, since the model is optimized to predict

the mid-price: Ceteris paribus, an increase in the bid-ask spread does not change the mid-price, since it is the average of the bid and ask price. However, the proportional bid-ask spread seemingly does affect the mid-price of options implying that unwillingness among market makers to supplyt the increased demand for an option not only effects the transaction costs but increase the mid-price of the option as well.

Yield Curve. Table 5.7 show evidence of improvements to the ANN (B) model from using information about the slope of the yield curve. The improvement is not as substantial as the improvement due to the liquidity measure, but the results still do imply an improvement in bias without an increase in variance.

The slope of the yield curve is generally considered reliable in predicting a forthcoming recession (Constable & Wright, 2011) and Table 5.7 suggests that this signal translates to option pricing as well. A likely cause of this relationship is the effect of supply and demand, concretely that market participants have increased demand for options as insurance to protect against adverse market moves when the slope of the yield curve is decreasing thus driving up the prices of these options. At the same time, market makers are less willing to supply the market with crash insurance given they too perceive an increased probability of a financial crisis.

5.4 Discussion and Suggested Further Research

The primary feature of ANNs that make them suitable for option pricing is their ability to learn the dynamics of the option markets without restrictive assumptions like the BSM model. While the BSM relies on assumptions such as the underlying asset following a geometric brownian motion and the volatility being constant, the ANN instead learns these dynamics directly from the data. ANNs do not require specification of a partial differential equation, complex continuous-time stochastic calculus, assumptions about perfect capital market, etc. to obtain the price of an option, unlike the BSM model. Granted, ANNs are not simple models either, but they have the advantage of being versatile and useful to a range of different applications within

most scientific fields without considerable adjustment. This versatility likely allows for the ANN model to apply to different derivatives such as American options, exotic options or even interest rate derivatives. Indeed, I would suggest conducting further research on the pricing of different derivatives using ANNs or other machine learning methods given the required data is available in the necessary quantities.

While the ANN model prevails both the BSM model and the PBS model, the errors are still significant from an economic perspective, specifically for OTM options. Conceivably the most prominent concern regarding the use of ANNs to price options is the black-box concept. Even though the ANN model exhibits lower pricing errors than the BSM model and the PBS model, the black-box nature of the model may worry market participants using the model. While the BSM model consistently misprices options, likely as a result of the unrealistic assumptions, option dealers are aware of the limitations of the model and are able to make adjustments that take the market dynamics better into account. On the other hand, the inner workings of the ANN model are concealed by the complexity of the model stemming from the massive number of parameters connecting nonlinear functions. Instead, option dealers must blindly trust the predictions of option prices of the ANN model and disregard any wish to know what causes a specific prediction. This implies a tradeoff between the vast improvements in the pricing error of the ANN model and the black-box nature of the model. Further research could examine the outliers of the ANN model versus the BSM model and evaluate whether either model is more reliable in 'extreme' times versus in 'normal' times.

As far as I am aware, the notion of including additional variables in an ANN option pricing model is novel, and the initial results of doing so are promising. Especially the liquidity measure proved to have a positive effect on the pricing error of the ANN model as well as the slope of the yield curve. The improvements to the ANN model of including these two measures suggest that the ANN model is able to capture the effect of supply and demand on option pricing to a degree. The favorable tradeoff between bias and variance suggested that ANN (A) was able to capture this effect reliably and consistently.

It would be interesting to explore further if any other variables have predictive power in

the ANN model or conversely if any of the traditional BSM inputs can be omitted from the ANN model. In relation to this, there is potential to further reduce the dimensionality of the ANN by normalizing input variables using insights from financial theory. For example, one could use the futures price of the underlying asset, $F_0 = S_0 e^{(r-q)T}$, and in this way reduce the number of input variables by two – the interest rate and the dividend yield – without losing any information. However, this approach introduces assumptions to the ANN model instead of letting the ANN model discover such relationships on its own. In addition, there may exist relationships between the interest rate and the other variables that the ANN model is unable to learn if these variables are embedded in the specification of the underlying asset as a futures contract.

The versatile nature of ANNs allows for easy adjustments to the inputs of the model. Arguably, this is one of the major advantages of ANN in relation to option pricing. Besides this, I find that the universal approximation theorem stating that ANNs can approximate any continuous function and the notion of ANNs being non-parametric are the primary reasons underlying the improvements of the ANN models to the traditional methods. ANNs surely have potential to improve on traditional option pricing models.

Chapter 6

Conclusion

The Black-Scholes-Merton option pricing model suffers from unrealistic assumptions that affect its empirical option pricing capabilities. In this thesis, I explore whether non-parametric artificial neural networks are able to price options on the S&P 500 and the Dow Jones Industrial Average indices better than traditional methods.

I develop an artificial neural network option pricing model and compare its empirical performance on index options to the Black-Scholes-Merton model and the Practitioner Black-Scholes model. I examine whether the artificial neural network is able to incorporate the effect of the volatility surface and supply and demand in option markets.

I find that the artificial neural network model prices index options on the S&P 500 index and the Dow Jones Industrial Average index with substantially lower pricing error than the comparison models. Approximately nine out of ten options are priced with less than 1% error, but large outliers pull the mean absolute percentage error to approximately 12%. Despite these outliers, this is still a substantial improvement to the Black-Scholes-Merton model's mean absolute percentage error of approximate 25% and the Practitioner Black-Scholes model's mean absolute percentage error of approximately 21%.

I present evidence that the artificial neural network model is better able to capture the volatility surface known from option markets since the model prices options better than the

comparison models across different moneyness and time-to-maturity. However, the pricing error is still a decreasing function of moneyness and time-to-maturity implying that the pricing errors get larger the more out-the-money the options becomes and the longer the time-to-maturity is.

I show that including the proportional bid-ask spread as a measure of the liquidity premium and the slope of the yield curve as variables to the artificial neural network model improves the pricing error. In particular, the proportional bid-ask spread model achieves an out-of-sample mean absolute percentage error of approximately 8.5%, close to three times better than the Black-Scholes-Merton model. I conclude that the improvement from using the liquidity measure is likely due to the demand pressure on index put options shown by Garleanu et al. (2008) and Constantinides and Lian (2015).

The primary limitation of the artificial neural network is its black-box nature. Concretely, the model is highly opaque due to a large number of parameters and nonlinear activation functions. This implies a tradeoff between the opaqueness of the artificial neural network model and the unrealistic assumptions of the Black-Scholes-Merton model.

I conclude that the artificial neural network model is a promising option pricing model that prevails both the Black-Scholes-Merton model and the Practitioner Black-Scholes model substantially. The ANN models developed in this thesis result in lower pricing errors than the comparison models while indicating the ability to capture the volatility surface and the effect of supply and demand on option pricing. In particular, the ability to add additional variables with predictive power is a promising feature of the artificial neural network model.

I believe artificial neural networks have great potential in the field of option pricing. I would suggest further research be conducted on additional variables that could help increase the predictive power of the artificial neural network model, particularly variables that could help decrease the relatively high pricing error of out-the-money options and options with long time-to-maturity.

Appendix A

Data Preparation

```python
# Import Libraries
import pandas as pd
import numpy as np
from copy import copy
import matplotlib.pyplot as plt
import scipy.stats as si
import statsmodels.api as sm
from py_vollib.black_scholes_merton.implied_volatility import implied_volatility

# Import Data
data = pd.read_csv('Data/Option Metrics/option_metrics.csv')

# Data Preparation
data = data[data['ticker']=='SPX'] # SPX or DJX
data = data[data['cp_flag']=='C']
data['strike_price'] = data['strike_price'] / 1000
data = data[data['best_bid'] > 0]
data['price'] = (data['best_bid'] + data['best_offer']) / 2
data['spread'] = data['best_offer'] - data['best_bid']
data = data.drop(['best_bid','best_offer'],axis=1)
data['date'] = pd.to_datetime(data['date'], format = '%Y%m%d')
data['exdate'] = pd.to_datetime(data['exdate'], format = '%Y%m%d')
```

```
data['T'] = ((data['exdate'] - data['date']).dt.days)/365
data = data.set_index('date')
del(data['exdate'])
data = data[(data['T'] <= 2)]
data['Year'] = data.index.year
data = data[(data['Year'] >= 2010)]

# Merge Interest Rate (r), Continous Dividends (q), Underlying (S), and
    Volatility (sigma)

# 1M Treasury Rate (r)
r = pd.read_csv('Data/Option Metrics/1M Treasury Rate.csv', parse_dates=True,
    index_col='DATE',na_values='.')
r.dropna(inplace=True)
r['r'] = np.float64(r['DGS1MO']) / 100
del(r['DGS1MO'])
data = data.merge(r, left_index=True, right_index=True, how='left')
del(r)

# Continous Dividends (q)
q = pd.read_csv('Data/Option Metrics/div.csv', index_col='date', parse_dates=
    True)
q['rate'] = q['rate'] / 100
q = q[q['ticker']=='SPX'] # SPX or DJX
q = pd.DataFrame(q['rate'])
data = data.merge(q, left_index=True, right_index=True, how='left')
del(q)

# Underlying (S)
S = pd.read_csv('Data/Option Metrics/Indices_ts.csv', parse_dates=True,
    index_col='date')
S = S[S['ticker']=='SPX'] # SPX or DJX
S = pd.DataFrame(S['close'])
data = data.merge(S, left_index=True, right_index=True, how='left')
del(S)
data['M'] = data['close'] / data['strike_price']
```

```
55 data = data[(data['M'] <= 1.5)]
56 data = data[(data['M'] >= 0.8)]
57
58 # Volatility (sigma)
59 vol = pd.read_csv('Data/Option Metrics/SPX_vol.csv', index_col='date',
       parse_dates=True) # SPX or DJX
60 data = data.merge(vol, left_index=True,right_index=True, how='left')
61 del(vol)
62
63 # Merge Additional Variables
64
65 # CAPE ratio
66 cape = pd.read_csv('Data/Option Metrics/Additional Variables/cape.csv',
       na_values='.', index_col='date', parse_dates=True)
67 cape = cape.resample(rule='D').ffill()
68 data = data.merge(cape, left_index=True, right_index=True, how='left')
69 del(cape)
70
71 # TED spread
72 TEDspread = pd.read_csv('Data/Option Metrics/Additional Variables/TEDspread.csv'
       , dtype='a', na_values='.', index_col='DATE')
73 TEDspread.dropna(inplace=True)
74 TEDspread['TEDRATE'] = np.float64(TEDspread['TEDRATE']) / 100
75 data = data.merge(TEDspread, left_index=True, right_index=True, how='left')
76 del(TEDspread)
77
78 # Yield Curve: 10Y - 2Y
79 yield_curve = pd.read_csv('Data/Option Metrics/Additional Variables/T10Y2Y.csv',
        dtype='a', na_values='.', parse_dates=True, index_col='DATE')
80 yield_curve['T10Y2Y'] = np.float64(yield_curve['T10Y2Y']) / 100
81 yield_curve = yield_curve.resample(rule='D').ffill()
82 data = data.merge(yield_curve, left_index=True, right_index=True, how='left')
83 del(yield_curve)
84
85 # BSM Call Option Price
86 def BSM_call(S,K,T,r,q,sigma):
```

```
87      d1 = (np.log(S/K) + (r - q + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
88      d2 = d1 - sigma * np.sqrt(T)
89          return S * np.exp(-q*T) * si.norm.cdf(d1, 0.0, 1.0) - K * np.exp(-r * T)
        * si.norm.cdf(d2, 0.0, 1.0)
90
91  # Practitioner Black-Scholes
92  iv = np.zeros((len(data),1))
93  for i in range(len(iv)):
94      try:
95          iv[i] = implied_volatility(data['price'][i], data['close'][i], data['
        strike_price'][i], data['T'][i], data['r'][i], data['rate'][i], 'c')
96      except:
97      iv[i] = np.nan
98  data['IV'] = iv
99  data.dropna(inplace=True)
100
101 reg = pd.DataFrame(data['strike_price'])
102 reg['K^2'], reg['T'], reg['T^2'], reg['KT'] = data['strike_price'] ** 2, data['T
        '], data['T'] ** 2, data['strike_price'] * data['T']
103 reg = sm.add_constant(reg)
104 reg['y'] = data['IV']
105 def regress(reg, X, y):
106     results = sm.OLS(reg[y],reg[X]).fit()
107     return results.params
108 params = reg.groupby(by=reg.index).apply(regress, ['const','strike_price','K^2',
        'T','T^2','KT'], ['y'])
109 data = data.merge(params,left_index=True, right_index=True)
110 data['Practitioner Vol'] = data.const + data['strike_price_y'] * data['
        strike_price_x'] + data['K^2'] * (data['strike_price_x'] ** 2) + data.T_y *
        data.T_x + data['T^2'] * (data.T_x **2) + data.KT * data['strike_price_x'] *
        data.T_x
111 data = data[(data['Practitioner Vol'] > 0.01)]
112
113 # Data Export
114 df = pd.DataFrame()
```

```
115 df['S'], df['K'], df['M'], df['T'] = data['close'],  data['strike_price_x'],
    data['M'], data['T_x']
116 df['vol10'], df['vol30'], df['vol60']  = data['vol10'], data['vol30'], data['
    vol60']
117 df['GARCH'], df['HAR-RV'], df['VIX'] = data['GARCH'], data['HAR-RV'], data['VIX'
    ] # VIX or VXD
118 df['IV'], df['Practitioner Vol'] = data['IV'], data['Practitioner Vol']
119 df['r'], df['q'] = data['r'], data['rate']
120 df['price'] = data['price']
121 df['ID'] = data['optionid']
122 df['price'] = df['price'] / df['K']
123 df['CAPE'], df['TED'], df['Yield Curve'], df['Liquidity'] = data['cape'], data['
    TEDRATE'], data['T10Y2Y'], (data['spread'] / df['K']) / df['price']
124 df['Year'] = data['Year']
125 df['BSM (vol10)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['vol10'
    ])) / df['K']
126 df['BSM (vol30)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['vol30'
    ])) / df['K']
127 df['BSM (vol60)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['vol60'
    ])) / df['K']
128 df['BSM (GARCH)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['GARCH'
    ])) / df['K']
129 df['BSM (HAR-RV)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['HAR-
    RV'])) / df['K']
130 df['BSM (VIX)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df['VIX'])) 
    / df['K']
131 df['BSM (Practitioner)'] = (BSM_call(df['S'],df['K'],df['T'],df['r'],df['q'],df[
    'Practitioner Vol'])) / df['K']
132
133 df.to_csv('Data/Option Metrics/Market_Data_SPX_call.csv')
```

Appendix B

Volatility Forecasts

```
# Import Libraries
import pandas as pd
from matplotlib import pyplot as plt
from arch import arch_model
from arch.univariate import HARX
import numpy as np

# Import Data
data = pd.read_csv('Data/Option Metrics/Indices_ts.csv', parse_dates=True,
    index_col='date')
data = data[data['ticker']=='SPX'] # SPX or DJX
data = pd.DataFrame(data['close'])
data['return'] = data['close'].pct_change()
data.dropna(inplace=True)

# Historical Volatility (Rolling)
vol = pd.DataFrame(columns = ['vol10','vol30','vol60'])
vol['vol10'] = data['return'].rolling(window = 10).std() * np.sqrt(252)
vol['vol30'] = data['return'].rolling(window = 30).std() * np.sqrt(252)
vol['vol60'] = data['return'].rolling(window = 60).std() * np.sqrt(252)
vol.dropna(inplace=True)

```

```
22 # Recursive GARCH
23 returns = data['return'] * 100
24 garch = arch_model(returns)
25 end_loc = np.where(returns.index >= '2001-1-1')[0].min()
26 forecasts = {}
27 for i in range(len(returns)-end_loc+1):
28     res = garch.fit(last_obs=i+end_loc, disp='off')
29     temp = res.forecast().variance
30     fcast = temp.iloc[i+end_loc-1]
31     forecasts[fcast.name] = fcast
32 df = pd.DataFrame(forecasts).T
33 df = np.sqrt(df/100)
34 vol['GARCH'] = df
35
36 # HAR-RV
37 rv = pd.read_csv('Data/realized_vol.csv', index_col = 'date', parse_dates = True
       )
38 rv = rv[rv['Symbol']=='.DJI']
39 rv = pd.DataFrame(rv['rv5'])
40 har = HARX(rv, lags=[1, 5, 22])
41 end_loc = np.where(rv.index >= '2001-1-1')[0].min()
42 forecasts = {}
43 for i in range(len(rv)-end_loc+1):
44     res = har.fit(last_obs=i+end_loc, disp='off')
45     temp = res.forecast().mean
46     fcast = temp.iloc[i+end_loc-1]
47     forecasts[fcast.name] = fcast
48 df = pd.DataFrame(forecasts).T
49 df = np.sqrt(df * 252)
50 vol['HAR-RV'] = df
51
52 # VIX (SPX) / VXD (DJIA)
53 vix = (pd.read_csv('Data/Option Metrics/Additional Variables/VIX.csv',
       parse_dates=True, index_col='Date')) / 100
54 vol['VIX'] = vix['vix'] # VIX or VXD
55
```

```
56 # Export to CSV
57 vol.to_csv('Data/Option Metrics/SPX_vol.csv') # SPX or DJX
```

Appendix C

Artificial Neural Network

```
# Import Libraries
from keras.models import Sequential
from keras.layers import Dense
from keras import backend
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.model_selection import train_test_split
from keras.models import load_model
from sklearn.preprocessing import RobustScaler
from keras.callbacks import History
import seaborn as sns
import time as time
sns.set(color_codes=True)
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
from keras.utils import CustomObjectScope
from keras.initializers import glorot_uniform
from copy import copy
import statsmodels.api as sm
import scipy.stats as si

```

```
23 # Accuracy Function
24 def CheckAccuracy(y,y_hat):
25     stats = dict()
26     stats['diff'] = y - y_hat
27     stats['mse'] = np.mean(stats['diff']**2)
28     print("Mean Squared Error:                 ", stats['mse'])
29     stats['mape'] = np.mean(np.abs(y - y_hat)/y
30     print("Mean Absolute Percentage Error:     ", stats['mape'])
31     return
32
33 # Import Data
34 df = pd.read_csv('Data/Option Metrics/Market_Data_SPX_call.csv', parse_dates=
       True, index_col = 'Unnamed: 0')
35 X_train, X_test, y_train, y_test = train_test_split(df[['M','T','r','q', 'vol10'
       ,'vol30','vol60','GARCH','HAR-RV','VIX', 'BSM (vol10)','BSM (vol30)','BSM (
       vol60)','BSM (GARCH)','BSM (HAR-RV)','BSM (VIX)', 'Year','K']].values, df[['
       price']].values, test_size = 0.10, random_state = 110493)
36 strike_train = np.resize(X_train[:,-1],(len(X_train),1))
37 year_train = np.resize(X_train[:,-2], (len(year_train),1))
38 BSM_train = np.resize(X_train[:,10:16], (len(BSM_train),6))
39 X_train = X_train[:,0:5]
40 strike_test = np.resize(X_test[:,-1], (len(strike_test),1))
41 year_test = np.resize(X_test[:,-2], (len(year_test),1))
42 BSM_test = np.resize(X_test[:,10:16], (len(BSM_test),6))
43 X_test = X_test[:,0:5]
44 vol_train = np.resize(X_train_sc[:,4:10], (len(vol_train),6))
45 X_train = X_train_sc[:,0:4]
46 vol_test = np.resize(X_test_sc[:,4:10], (len(vol_test),6))
47 X_test = X_test_sc[:,0:4]
48
49 # 10-fold cross-validation | Volatility Forecasting Model
50 for k in range(10):
51     mse_train, mse_test, mse_BSM_train, mse_BSM_test = np.zeros(6), np.zeros(6),
       np.zeros(6), np.zeros(6)
52     mape_train, mape_test, mape_BSM_train, mape_BSM_test = np.zeros(6), np.zeros
      (6), np.zeros(6), np.zeros(6)
```

```
53      diff_train, diff_test, diff_BSM_train, diff_BSM_test = np.zeros((len(X_train),6)), np.zeros((len(X_test),6)), np.zeros((len(X_train),6)), np.zeros((len(X_test),6))
54      modelList = [
55          ('vol10_SPX.h5','vol30_SPX.h5','vol60_SPX.h5','GARCH_SPX.h5','HAR-RV_SPX.h5','VIX_SPX.h5')]
56      errorList_csv = [
57          ('diff_train_SPX_1.csv','diff_test_SPX_1.csv')]
58
59      for i in range(6):
60          print('Fold', k+1, ',- Model', i+1)
61          model = Sequential()
62          model.add(Dense(128, input_dim=5))
63          model.add(Dense(128, activation='relu'))
64          model.add(Dense(128, activation='relu'))
65          model.add(Dense(128, activation='relu'))
66          model.add(Dense(1, activation='linear'))
67          model.compile(loss='mse',optimizer='adam')
68          X_train_sc = np.append(X_train, np.resize(vol_train[:,i], (len(vol_train),1)), axis = 1)
69          X_test_sc = np.append(X_test, np.resize(vol_test[:,i], (len(vol_test),1)), axis = 1)
70          model.fit(X_train_sc, y_train_sc, batch_size=4096, epochs=200, verbose = 0)
71          # Predictions (Train)
72          y_train_hat = sc_y.inverse_transform(np.float64(model.predict(X_train_sc)))
73          # Predictions (Test)
74          y_test_hat = sc_y.inverse_transform(np.float64(model.predict(X_test_sc)))
75          # ANN Evaluation Metrics
76          diff_train[:,i:i+1] = y_train - y_train_hat
77          mse_train[i] = np.mean(diff_train[:,i:i+1]**2)
78          mape_train[i] = np.mean(np.abs(diff_train[:,i:i+1]/y_train))
79          diff_test[:,i:i+1] = y_test - y_test_hat
80          mse_test[i] = np.mean(diff_test[:,i:i+1]**2)
```

```
81          mape_test[i] = np.mean(np.abs(diff_test[:,i:i+1]/y_test))
82          # BSM Evaluation Metrics
83          diff_BSM_train[:,i:i+1] = y_train - BSM_train[:,i:i+1]
84          diff_BSM_test[:,i:i+1] = y_test - BSM_test[:,i:i+1]
85          mse_BSM_train[i] = np.mean(diff_BSM_train[:,i:i+1]**2)
86          mse_BSM_test[i] = np.mean(diff_BSM_test[:,i:i+1]**2)
87          mape_BSM_train[i] = np.mean(np.abs(diff_BSM_train[:,i:i+1]/y_train))
88          mape_BSM_test[i] = np.mean(np.abs(diff_BSM_test[:,i:i+1]/y_test))
89          # Save Model
90          model.save(modelList[k][i])
91      print('Model', i+1, 'Completed')
92      print(mse_train)
93      print(mse_test)
94      print(mape_train)
95      print(mape_test)
96      print(mse_BSM_train)
97      print(mse_BSM_test)
98      print(mape_BSM_train)
99      print(mape_BSM_test)
100     # Save Residuals
101     print('Saving Residuals')
102     errorList = [diff_train,diff_test, diff_BSM_train, diff_BSM_test]
103     for j in range(4):
104     df = pd.DataFrame(errorList[j])
105     df.to_csv(errorList_csv[k][j])
```

Bibliography

Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed.). The MIT Press.

Amihud, Y., Mendelson, H., & Pedersen, L. H. (2006). Liquidity and asset prices. *Foundations and Trends® in Finance*, *1*(4), 269-364.

Amilon, H. (2003). A neural network versus black-scholes: A comparison of pricing and hedging performances. , *22*(4), 317–335.

Andersen, T., Bollerslev, T., Diebold, F., & Labys, P. (2003). Modeling and forecasting realized volatility. *Econometrica*, *71*(2), 579-625.

Andolfatto, D., & Spewak, A. (2019). *Does the yield curve really forecast recession?*

Bates, D. S. (2003). Empirical option pricing: a retrospection. *Journal of Econometrics*, *116*(1-2), 387-404.

Benitez, J. M., Castro, J. L., & Requena, I. (1997, Sep.). Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks*, *8*(5), 1156-1164.

Bergstra, J., & Bengio, Y. (2012, February). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*, 281–305.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637-54.

Bollen, N. P. B., & Whaley, R. E. (2004). Does net buying pressure affect the shape of implied volatility functions? *Journal of Finance*, *59*(2), 711-753.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307-327.

Boudt, K., Paulus, E. C., & Rosenthal, D. (2017). Funding liquidity, market liquidity and ted spread: A two-regime model. *Journal of Empirical Finance*, *43*(C), 143-158.

Brown, R. (2019, 05). A brief account of microscopical observations made in the months of june, july and august, 1827. *On the Particles Contained in the Pollen of Plants; and on the General Existence of Active Molecules in Organic and Inorganic Bodies*, 161-173.

Cboe. (2019). *Cboe volatility index - vix® whitepaper* (Tech. Rep.). Author.

Christoffersen, P., & Jacobs, K. (2004). The importance of the loss function in option valuation. *Journal of Financial Economics*, *72*(2), 291 - 318.

Constable, S., & Wright, R. E. (2011). *The wsj guide to the 50 economic indicators that really matter: From big macs to "zombie banks," the indicators smart investors watch to beat the market.* HarperBusiness.

Constantinides, G. M., & Lian, L. (2015, May). *The supply and demand of s&p 500 put options* (Working Paper No. 21161). National Bureau of Economic Research.

Corsi, F. (2009, Spring). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, *7*(2), 174-196.

Cybenko, G. (1989, December 1). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, *2*(4), 303–314.

Dumas, B., Fleming, J., & Whaley, R. E. (1998). Implied volatility functions: Empirical tests. *Journal of Finance*, *53*(6), 2059-2106.

Estrella, A., & Mishkin, F. S. (1996). The yield curve as a predictor of u.s. recessions. *Current Issures in Economics and Finance*, *2*(7).

Fournier, M. (2013, 01). Inventory risk, market-maker wealth, and the variance risk premium. *SSRN Electronic Journal*.

FRED, F. R. B. o. S. L. (2019a). *10-year treasury constant maturity minus 2-year treasury constant maturity [t10y2y].*

FRED, F. R. B. o. S. L. (2019b). *Ted spread.* Retrieved April 2, 2019, from `https://fred.stlouisfed.org/series/TEDRATE`

Garleanu, N., Pedersen, L. H., & Poteshman, A. M. (2008). Demand-based option pricing. *The Review of Financial Studies*, *22*(10), 4259-4299.

Gençay, R., & Qi, M. (2001, 08). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *Neural Networks, IEEE Transactions on*, *12*, 726 - 734.

Google. (2019). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Gradojevic, N., Gençay, R., & Kukolj, D. (2009, April). Option pricing with modular neural networks. *Trans. Neur. Netw.*, *20*(4), 626–637.

Herculano-Houzel, S. (2009, 11). The human brain in numbers: A linearly scaled-up primate brain. *Frontiers in human neuroscience*, *3*, 31.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, *6*(2), 327-43.

Hull, J. C. (2018). *Options, futures, and other derivatives.* Pearson.

Hutchinson, J. M., Lo, A., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, *49*(3), 851-89.

Itô, K. (1944). Stochastic integral. *Proceedings of the Imperial Academy*, *20*(8), 519-524.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning: with applications in r* (1st ed. ed.). Springer.

Kubat, M. (2015). *An introduction to machine learning* (1st ed.). Springer Publishing Company, Incorporated.

Lajbcygier, P. R., & Connor, J. T. (1997). Improved option pricing using artificial neural networks and bootstrap methods. *International Journal of Neural System*, *8*(4), 457-471.

Liang, X., Zhang, H., Xiao, J., & Chen, Y. (2009, August). Improving option price forecasts with neural networks and support vector regressions. *Neurocomput.*, *72*(13-15), 3055-3065.

Mandelbrot, B. (1963). The variation of certain speculative prices. *The Journal of Business*, *36*.

Mcculloch, W., & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 127–147.

Merton, R. (1973). Theory of rational option pricing. *Bell Journal of Economics*, *4*(1), 141-183.

Ng, A., & Katanforoosh, K. (2018, October). *Deep learning cs229 lecture notes.*

OptionMetrics. (2015). *Ivydb us* (Tech. Rep.). Author.

Oxford-man realized volatility library. (2019). Retrieved March 28, 2019, from `http://www.oxford-man.ox.ac.uk`

Paperspace. (2019). Retrieved May 10, 2019, from `https://www.paperspace.com`

Park, H., Kim, N., & Lee, J. (2014). 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.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Poitras, G. (2008, 01). The early history of option contracts. *Vinzenz Bronzin's Option Pricing Models: Exposition and Appraisal*. doi: 10.1007/978-3-540-85711-2_24

Rouah, F. D., & Vainberg, G. (2007). *Option pricing models and volatility using excel-vba.* John Wiley and Sons, Inc.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986, October). Learning representations by back-propagating errors. *Nature*, *323*, 533–.

Shiller, R. (2005). *Irrational exuberance* (2nd ed.). Princeton University Press 2000.

Shiller, R. (2019). *Shiller data.* Retrieved April 2, 2019, from `http://www.econ.yale.edu/~shiller/data.htm`

Taleb, N. N. (2008). *The black swan. the impact of the highly improbable.* Random House Inc.

Wang, Y.-H. (2009). Nonlinear neural network forecasting model for stock index option price: Hybrid gjr–garch approach. *Expert Systems with Applications*, *36*(1), 564 - 570.

Wan Shin, D. (2018, 09). Forecasting realized volatility: A review. *Journal of the Korean Statistical Society*, *47*.

Yang, S.-H., & Lee, J. (2011, March). Predicting a distribution of implied volatilities for option pricing. *Expert Syst. Appl.*, *38*(3), 1702-1708.