UNIVERSITY OF GOTHENBURG
SCHOOL OF BUSINESS, ECONOMICS AND LAW

Deep Learning and the Heston Model: Calibration & Hedging

Oliver Klingberg Malmer & Victor Tisell

July 2020

A thesis presented for the degree of
Bachelor of Science in Statistics

Thesis advisor: Alexander Herbertsson
School of Business, Economics and Law
Department of Economics

Contact:
victor@tritonic.se or gusmalmeol@student.gu.se

Abstract

The computational speedup of computers has been one of the defining characteristics of the 21st century. This has enabled very complex numerical methods for solving existing problems. As a result, one area that has seen an extraordinary rise in popularity over the last decade is what is called *deep learning*. Conceptually, deep learning is a numerical method that can be "taught" to perform certain numerical tasks, without explicit instructions, and learns in a similar way to us humans, i.e. by trial and error. This is made possible by what is called *artificial neural networks*, which is the digital analogue to biological neural networks, like our brain. It uses interconnected layers of *neurons* that activates in a certain way when given some input data, and the objective of *training* a artificial neural network is then to let the neural network learn how to activate its neurons when given vast amounts of training examples in order to make as accurate conclusions or predictions as possible.

In this thesis we focus on deep learning in the context of financial modelling. One very central concept in the financial industry is pricing and risk management of financial securities. We will analyse one specific type of security, namely the *option*. Options are financial contracts struck on an underlying asset, such as a stock or a bond, which endows the buyer with the optionality to buy or sell the asset at some pre-specified price and time. Thereby, options are what is called a financial *derivative*, since it derives its value from the underlying asset. As it turns out, the concept of finding a fair price of this type of derivative is closely linked to the process of eliminating or reducing its risk, which is called *hedging*. Traditionally, pricing and hedging is achieved by methods from probability theory, where one imposes a certain model in order to describe how the underlying asset price evolves, and by extension price and hedge the option. This type of model needs to be *calibrated* to real data. Calibration is the task of finding parameters for the stochastic model, such that the resulting model prices coincide with their corresponding market prices. However, traditional calibration methods are often too slow for real time usage, which poses a practical problem since these models needs to be re-calibrated very often. The hedging problem on the other hand has been very difficult to automate in a realistic market setting and suffers from the simplistic nature of the classical stochastic models.

The objective of this thesis is thus twofold. Firstly, we seek to calibrate a specific probabilistic model, called the *Heston model*, introduced by Heston (1993) by applying neural networks as described by the *deep calibration* algorithm from Horvath et al. (2019) to a major U.S. equity index, the S&P-500. Deep calibration, amongst other things, addresses the calibration problem by being significantly faster, and also more universal, such that it applies to most option pricing models, than traditional methods.

Secondly, we implement artificial neural networks to address the hedging problem by a completely data driven approach, dubbed *deep hedging* and introduced by Buehler et al. (2019), that allows hedging under more realistic conditions, such as the inclusion of costs associated to trading. Furthermore, the deep hedging method has the potential to provide a broader framework in which hedging can be achieved, without the need for the classical probabilistic models.

Our results show that the deep calibration algorithm is very accurate, and the deep hedging method, applied to simulations from the calibrated Heston model, finds hedging strategies that are very similar to the traditional hedging methods from classical pricing models, but deviates more when we introduce transaction costs. Our results also indicate that different ways of specifying the deep hedging algorithm returns hedging strategies that are different in distribution but on a pathwise basis, look similar.

Keywords: deep learning, deep hedging, deep calibration, option pricing, stochastic volatility, Heston model, S&P 500 index options, incomplete markets, transaction costs.

Acknowledgements

We would like to express our special thanks and gratitude to our advisor Alexander Herbertsson, who enabled this thesis by his commitment to the subject, guidance, thoughtful discussions and constructive comments.

Contents

1 Introduction

A *financial derivative* is a financial instrument that derives its value from some other financial asset. In order for dealers of financial derivatives to operate efficiently and provide liquidity to the marketplace, dealers must be able to minimize and ideally eliminate their risk. The practice in which this is achieved is called *hedging*. A hedge is a offsetting position to an agents portfolio of instruments and/or assets which serves to reduce risk. The theoretical motivation of how such a strategy should be achieved is heavily intertwined with derivative pricing theory. The traditional framework in which pricing and hedging operates is a result of Black & Scholes (1973) and is generally referred to as the *Black-Scholes* framework (BS), which is a coherent mathematical structure that allows for closed form solutions to pricing and hedging, which will be described in greater detail later on in this thesis. However, since the introduction of BS, financial markets, including derivatives markets, has expanded and developed extensively. The evolution of the derivatives market, which has closely been in line with the development of modern computational technology, has greatly impacted the way dealers operate and by extension how effective capital allocation has become, since computationally intensive methodology enables numerical solutions to complex problem formulations. As a result of the increased complexity of financial markets, the flaws of the central assumptions of the BS-model have become more apparent. This thesis seeks to research a numerical solution to both hedging and pricing, that is independent of classical methods, such as the Black-Scholes model.

Motivated by the limitations of the Black-Scholes model, other models has been suggested. Some of these frameworks model volatility as a separate process, such as the *Heston model*. The Heston model, introduced by Heston (1993), is an expansion of the Black-Scholes model which explicitly takes the non-constant volatility, i.e. the variation of any price series measured by the standard deviation of log-returns, into account. However, complex financial models are often less parsimonious as they depend on multidimensional parameter spaces, introducing a *model calibration* problem. Model calibration is the process by which the parameters of a certain model is estimated, generally by minimizing the difference between model generated data and observed data. For the specific Heston model, fast numerical methods has been suggested that performs well, however, are not generalizable to more complex models such as rough volatility models and models with jumps in volatility. Even if the empirical properties of such models are promising, the computational costs are large. Thus, the limiting the applicability of such models. Therefore, there exist a need for faster and, ideally, more accurate calibration methods, will be the partial objective of this thesis.

Hedging of financial derivatives should be performed with realistic market assumptions. When a financial institution trades a security, the institution incurs a cost, generally referred to as a *transaction cost*. If one combines the existence of transaction costs with the statistical properties of realistic market conditions, classical models are insufficient descriptions of reality, meaning they will produce non-systematic errors. The research of hedging under the conditions given by realistic market assumption has long been an important subject for practitioners. However, the inclusion of transaction costs omits analytical solutions to optimal hedging weights. Thus, there exist a need for numerical methods to solve the solutions to optimal hedging weights with inclusion of transaction costs. As such, the central property of numerical solutions to the hedging problem should be model independence, as this exclude the assumption of unrealistic statistical properties of financial markets, i. e. the method has to be data driven and independent of explicit human input to correct non-systematic errors.

Modern computers have enabled accessibility of complex numerical methods for solving existing problems. One such group of methods are called *machine learning*. Machine learning (ML) is the process in which computer systems use algorithms and statistical models to perform some task, independently of explicit instructions of how to perform it. In general, machine learning algorithms utilize training data, on which the machine "learns" the pattern of the data and builds a function such that it seeks to optimally perform a certain task. This model is then generally used *out of sample* in which the quality of the predictions are validated. One example of such models are *artificial neural networks (ANN)*. Artificial neural networks are computational systems that are reminiscent of biological neural networks in brains. As such, the learning procedure of ANN's are vaguely similar to that of human brains. Thus, such a network learns, without task specific instructions, by "feeding" some input and activating *neurons* in the network to come to a conclusion about what that the outcome should be. For example, in image recognition, a researcher might want to classify images of hand written numbers. The network is given some input image that is manually *labelled* with what number that image represents. The network then learns how it should change its parameters to classify correctly by evaluating how well each parameter combination performs the classification task. As one might suspect, this method requires very large data sets to be able to learn how to solve the task sufficiently well. Hence, even though the theory of ANN's have existed since McCulloch & Pitts (1943), its practical application has been limited until recently as the computational speedup of modern computers is substantial, especially for complex neural networks with large dimensions, and thus many parameters.

This thesis will attempt to address some of the deficiencies of both model calibration and hedging of the Heston model by means of deep learning, i.e. the application of multi-layered artificial neural networks. Based on the results of Hornik (1991), in which deep neural networks are shown to have universal approximating properties, any continuous function, and its derivatives, can be approximated arbitrarily well by a deep neural network, numerical methods, utilizing ANN's, for model calibration was introduced by Horvath et al. (2019) and is called *deep calibration*. Similar methods apply to hedging, i.e. neural networks can act as approximators for optimal hedging strategies. We will consider optimality in the same sense as Buehler et al. (2019), which are the originators of the method. We build a neural network calibration and hedging method on S&P 500 market data and attempt to address the *compatibility* of the methods. Thus, the objective of the thesis is to implement and study methods of numerical approximation for any full-scale front-office financial model under more realistic market assumptions than classical methods would allow, by means of deep learning. We implement the methodology for the specific Heston model, however, it is easily extendable to other financial models. Hence, we shall calibrate a Heston model on the S&P 500 by *deep calibration*, and then simulate trajectories which are used to *train* the *deep hedging* method. We then evaluate the performance of each separately by means of descriptive statistics and visualization.

Our numerical studies in the Heston model, using simulated prices, indicate that the deep calibration algorithm can approximate traditional calibration methods very well. However, as one might expect, the calibration error grows when applied to real market data, since the Heston model itself cannot perfectly capture real market dynamics, although the error is still small enough to ensure a "reasonable" fit to market data. Furthermore, we also show that when performing unconstrained calibration, where the resulting parameter space violates what is known as the *Feller condition*, which is a condition on the Heston model parameter space that ensures that the discretized version of the variance process remains positive. As such, violations of the Feller condition means that there exists possibilities for the simulated variance process being negative, which is theoretically impossible, and therefore introduces problems when simulating

the Heston model. We deal with the problem introduced by the violation of the Feller condition by simply constraining the parameter space, which as previously mentioned, will introduce larger calibration error. However, as the simulation of sample paths is integral to the deep hedging algorithm, the Feller condition becomes a "necessary condition" to satisfy.

Our numerical results from the *deep hedging* algorithm shows that generated strategies are consistently very close to a classical hedging strategy, used as a benchmark. Moreover, different profit/loss (pnl) distributions, i.e. the frequencies of hedging strategy pnl evaluated over a large number of sample paths, differ between risk measures. Hence, we are able to highlight some pros and cons of deep hedging strategies over different measures of risk, by comparing descriptive statistics such as their respective mean and variance etc. For example, when the risk measure is a quadratic function, the pnl distribution is very similar to the benchmark strategy. Furthermore, we also introduce transaction costs and show their impact on the deep hedging strategy and compare it to when costs are excluded. Our results indicate that the main difference when one introduces transaction costs is that the mean of the hedging strategy pnl distribution shifts upwards.

The thesis is structured as follows. In Section 2 we introduce the concept of financial derivatives, specifically options. The theoretical background of pricing of financial derivatives, probabilistic frameworks in which pricing and hedging are achieved and finally we provide a theoretical background to hedging, which will rationalize our choice of risk measure, is discussed in Section 3. Furthermore, the section includes a short description of classical model calibration which provides a contextual background to deep calibration. Section 4 expands previous discussions of hedging theory into discrete time. Furthermore, we introduce the concept of convex risk measures and optimal hedging under such measures. We also formulate the theoretical background of optimality under a shortfall measure weighted by a loss function. In Section 5, the theoretical background of neural networks needed to understand the numerical methods for hedging and calibration is covered. In Section 6 and 7 we explain the properties and the implementation of the AI-methods used for calibration and hedging respectively. In Section 8 we present the numerical results of the calibration method and the hedging strategy in relation to a traditional hedge obtained by traditional methods. Lastly, we provide conclusions and suggestions for future research in Section 9.

2 Background to Financial Derivatives

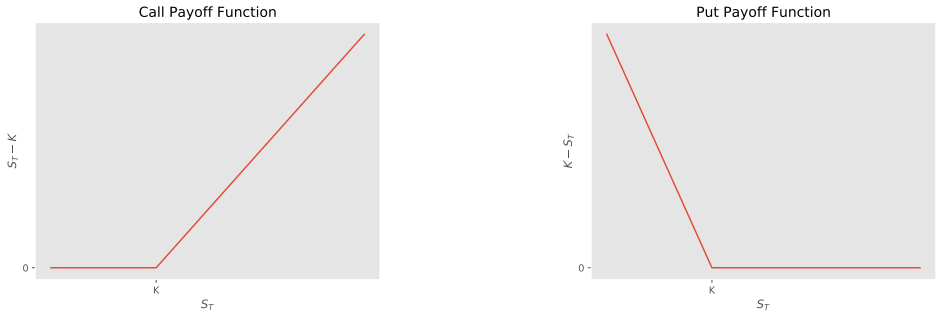
This section will introduce the concept of financial derivatives, specifically options and their properties and also serve to introduce the reader to the financial concepts in the formulation of the stated objective of the thesis.

A *financial derivative* is a financial instrument that derives its value from some other financial asset. The simplest form of a financial derivative is a *forward* contract. A forward is a contract between a buyer and a seller to transact at a future time at the current forward price. This contract is obligatory for both parties. Another slightly more complex form of derivative are *options*. These contracts come in many forms but we will only consider European vanilla options as these represent the derivative in its most simple form. A European vanilla option is a contract between a buyer and seller that bestows the buyer the right, not the obligation, to buy or sell a security at a pre-specified date and price. Other types of options differ in settlement type. The date of settlement is referred to as maturity T and the pre-specified price is referred to as strike K. There exist two types of vanillas, call options and put options, where call options

bestows the buyer right to buy the underlying asset and vice versa for put options. Because of the structure of these contracts, the payoff is non-linear with respect to the state-variable, which considers the space in which the underlying asset changes, in financial terms, the state considered is time. In order to gain a better understanding of the concept of options, one can consider the mathematical formulation of their payoff structure at time T,

$$call := (S_T - K)^+, \; put := (K - S_T)^+ \tag{1}$$

where S_T denotes the terminal value of the underlying asset at maturity. From Equation (1) it becomes evident that, as $K < S_T$, the call option will have a positive payoff and the put option will have a payoff that equals zero. For the call option, this scenario is called *in the money (ITM)* and for the put option this is called *Out of the money (OTM)*, both of these scenarios stems from the concept of *moneyness* which considers $\frac{S_t}{K} = M$. A option is considered at the money (ATM) if $M = 1$, that is if the current price of the underlying is equal to the exercise price. A option is considered to be in the money (ITM) if there exist a monetary gain in exercising it. A call is ITM if $M > 1$ and a put is ITM if $M < 1$ and a option is out of the money if the converse is true. The loss that the seller of the put option will be equal to strike of the option adjusted for the cash received when sold at the inception of the contract. Conversely, the payoff for the call option holder is equivalent to the strike, see Figure 1 for more details regarding option payoffs.



(a) Call option payoff.

(b) Put option payoff.

Figure 1: Payoff functions at maturity T for European call and put options with strike K.

3 Theoretical Perspective on Pricing, Hedging & Probabilistic Modelling

In this section we will introduce the reader to the concept of option pricing, stochastic modelling and two special cases of such models, namely the popular *Black-Scholes model* and the *Heston model*, which is the stochastic model that we will consider for calibration and hedging. We will cover the various tools needed in order to understand such models, for example stochastic processes and volatility. Furthermore, the reader will be introduced to *hedging* in its most general form and thus provide a solid theoretical background to the reader in order to understand the objective of the thesis.

Options are a special case of what is called *contingent claims*. Contingent claims are generally defined as a financial derivative whose payoff is *contingent*/dependent on the realization of

some future uncertain event regarding the underlying asset. Pricing and risk management of such claims is a central concept for financial practitioners and academics alike. In order to understand modern financial markets, in which derivatives are an essential part, one needs to understand their *valuation*. In order to model contingent claims, one needs to consider some form of *probabilistic model* of the underlying asset S. In such a model, the price of contingent claims reflects the known information about the terminal distribution of S under some probability measure.

In order to understand subsequent sections on stochastic models for financial assets, we need to introduce the notion of *probability spaces* and *filtered probability spaces*. A probability space is a measurable space, on which we can define a probability measure. A measurable space is defined by a tuple $(\Omega, \mathcal{F})$, where Ω is any set, usually called the sample space in probability theory, and $\mathcal{F} \subseteq 2^{\Omega}$, where 2^{Ω} is the power set of Ω, is whats called a σ-algebra. A σ-algebra is a collection of subsets of Ω, which denotes all possible events, that includes Ω and the empty set $\emptyset$ that is closed under complements and countable unions. A probability measure $\mathbb{P}$ on the measurable space $(\Omega, \mathcal{F})$ is a function $\mathbb{P} : \mathcal{F} \mapsto [0, 1]$ such that $\mathbb{P}(\Omega) = 1$, $\mathbb{P}(\emptyset) = 0$ that satisfies countable additivity for pairwise disjoints sets in $\mathcal{F}$. We now have the triple $(\Omega, \mathcal{F}, \mathbb{P})$, which is a probability space. Furthermore, since asset prices, and by extension, options evolve over time and adapts to the information in the market place, we need to introduce the concept of *filtration*. A filtration $\mathbb{F} := (\mathcal{F}_t)_{t \in \mathbb{T}}$ is formally defined as a collection of σ-algebras ordered by some index set, which for our purposes will be represented as time, provided that $\mathcal{F}_s \subseteq \mathcal{F}_t \subseteq \mathcal{F}$ for any $s \leq t$. The intuition behind filtration is that it represents all relevant information about the probability space at each point in time. What we now have is what is called a *filtered probability space* $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$. For further information see e.g. Protter (2005).

The general task of option pricing under the real measure $\mathbb{P}$ can then be described by *discounting* the expected payoff in which the discount rate is determined by an individual investors risk preference. Under the assumption of *complete markets* (see Section 3.5) and the absence of systematic risk-free profits (absence of arbitrage) one does not need to adjust the expectation individually but can incorporate all investors risk premia under a *equivalent measure*, $\mathbb{Q}$, such that $\mathbb{P} \sim \mathbb{Q}$ which means that the probability measures assign zero probability to the same sets, see e.g. Björk (2009) for further details. Thus, pricing of options can be achieved by discounting the expected payoffs of the probability distribution under the equivalent or *risk-neutral* measure ($\mathbb{Q}$). Thereby, one can consider a European option with strike K, maturity T and estimate its value

$$V_t^{call} = e^{-r(T-t)} \mathbb{E}_{\mathbb{Q}} \left[\max(S_T - K)^+ | \mathcal{F}_t \right] \tag{2}$$

$$V_t^{put} = e^{-r(T-t)} \mathbb{E}_{\mathbb{Q}} \left[\max(K - S_T)^+ | \mathcal{F}_t \right] \tag{3}$$

in which $T - t$ denotes the *time to maturity*. The task described by Equation (2) and (3) amounts to evaluating the probability distribution of S_T under $\mathbb{Q}$. Here we denote the payoff functions $\max(S_T - K)^+$ by $g(S_T; K, T)$ and $\max(K - S_T)^+$ by $h(S_T; K, T)$. When $t = 0$, then $\mathcal{F}_t = \mathcal{F}_0$ which means that the conditional expectation $\mathbb{E}_{\mathbb{Q}}[u(S_T; K, T)|\mathcal{F}_0] = \mathbb{E}_{\mathbb{Q}}[u(S_T; K, T)]$ in which u is any deterministic function. This means that the conditional expectations admits the representation

$$V_0^{call} = e^{-rT} \mathbb{E}_{\mathbb{Q}} \left[g(S_T; K, T) \right] = e^{-rT} \int_{-\infty}^{\infty} g(x; K, T) f_{S_T}(x) dx \tag{4}$$

$$V_0^{put} = e^{-rT} \mathbb{E}_{\mathbb{Q}} \left[h(S_T; K, T) \right] = e^{-rT} \int_{-\infty}^{\infty} h(x; K, T) f_{S_T}(x) dx. \tag{5}$$

One can see in Equation (4) and (5) that the only stochastic element present in the pricing task at $t = 0$ is the terminal distribution of S. In order to model this stochastic element, *stochastic processes* needs to be introduced. Stochastic processes are ordered stochastic variables over a continuous space, that we define as time. Formally, a stochastic process is a family of stochastic variables defined on the probability space. The simplest form of such a process is the *Brownian motion/Wiener process* (see Figure 2 for sample path.)

Definition 3.1. Wiener process. The Wiener process $(W_t)_{t\geq 0}$ is a stochastic process with the following properties:

1. $W_0 = 0$. The initial point of the process is zero.
2. Increments in W are independent.
3. W_t is almost surely continuous.
4. $W_t \sim \mathcal{N}(0, t)$. All increments of W are normally distributed with $\mathbb{E}[W_t - W_s] = 0$ and $Var(W_t - W_s) = t - s$ for $t \geq s$

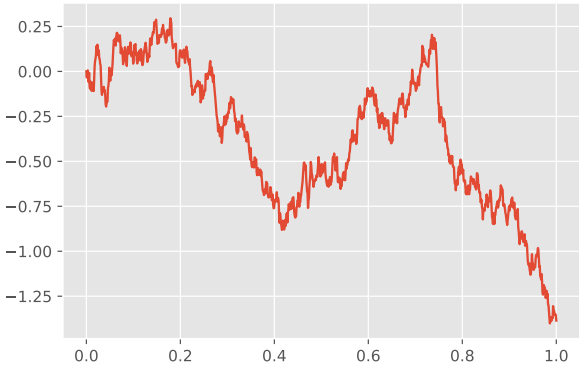


Figure 2: Sample trajectory of Brownian motion/ Wiener process $(W_t)_{t\in(t,T]}$ with $W_0 = 0$, independent normally distributed increments with $W_t \sim \mathcal{N}(0, t)$ for each t.

A Wiener process is sometimes also referred to as *Brownian motion*. One informal way of representing a stochastic process, is by what is called a *stochastic differential equations* (SDE's). A SDE is the representation of a stochastic process in its differential form, and differs from an ordinary differential equation in the sense that one or more terms is a stochastic process, typically some random noise, for example dW, which is the representation of Brownian motion with the property that $dW \sim \mathcal{N}(0, dt)$ and can be conceptualized as the limit of Brownian increments. In order to solve such equations, stochastic calculus is applied.

3.1 The Black-Scholes Model

The Black-Scholes (BS) model is a analytically coherent framework of market dynamics and is the bedrock on which most derivative pricings are based. The BS-model allows for closed form

solutions to V_t^{call} and V_t^{put} given by Equations (2) and (3) respectively, where $\mathcal{F}_t = \sigma(W_s : s \leq t)$ and is called the filtration generated by the Wiener process. The spot process S follows the SDE

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{6}$$

where W is a Wiener process, see Definition 3.1. In order to solve this equation, stochastic calculus must be applied. Consider its integral form

$$S_t = S_0 + \int_0^t rS_u du + \int_0^t \sigma S_u dW_u$$

and when Itô calculus is applied the resulting stochastic process is a *geometric Brownian motion*

$$S_t = S_0 e^{\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t}. \tag{7}$$

If one would consider the SDE under the real measure $\mathbb{P}$, the risk free rate r should be substituted for the instantaneous drift as assets do no-longer have the same expected return Black & Scholes (1973). After solving the BS *partial differential equation*, the following expression is obtained.

Definition 3.2. Black-Scholes Equation

$$C^{BS}(S_t, \sigma, K, T) = \mathcal{N}(d_1)S_t - \mathcal{N}(d_2)Ke^{-r(T-t)} \tag{8}$$

in which $C^{BS}(S_0, \sigma, K, T) = V^{call}$, $\mathcal{N}(\cdot)$ denotes the standard normal cumulative distribution function and

$$d_1 = \frac{\ln(S_t/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \qquad d_2 = d_1 - \sigma\sqrt{T-t} \tag{9}$$

where time to maturity is denoted as $T - t$. The only unknown quantity under $\mathbb{Q}$ is the volatility parameter σ.

As mentioned in the introduction, there exist severe limitations of the BS-model.

- **The stochastic differential equation.** The stochastic differential equation (6) that Black-Scholes utilizes is a oversimplification of the way that stock prices evolve. By extension, this implies that prices are continuous and returns are log-normally distributed. As empirical data suggests, asset returns are more fat-tailed than that of the Gaussian distribution. This results in distortions in the probability that a contract expires in the money (ITM) and, by extension, in the price of the option.
- **Volatility.** The only unknown quantity of the model is the volatility, σ. This quantity is thereby the a critical component of the model and is considered to be constant. Market data indicates that this is not the case as it varies across strike and maturity of the option.
- **Discretization.** Real life trading and hedging takes place at discrete times while Black-Scholes assumes continuous time and approximations leads to suboptimal solutions for practitioners.

In order to deal with these inherent deficiencies, new models has been proposed which focus primarily on the concept of *volatility*.

3.2 Stochastic and Implied Volatility

Volatility is the main measurement of risk associated with any financial asset. *Realized volatility*, σ_R, is defined as the standard deviation of continuous compounding historical returns. Let $S_{t_1}, S_{t_2}, \ldots, S_{t_n}$ be observations of a stock price S_t at n different time points $t_1 < t_2 < \cdots < t_n$ then the *realized volatility* over n time point is defined as

$$\sigma_R = \sqrt{\left[\frac{1}{n-1}\sum_{i=1}^{n}(r_i - \bar{r})^2\right]} \tag{10}$$

where $r_i = \ln\left(\frac{S_{t_i}}{S_{t_{i-1}}}\right)$ and denotes the log return of the financial asset over the non-finite time interval $t - u$. Furthermore $\bar{r}$ denotes the mean log return of the asset

$$\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i \quad .$$

Volatility estimates can be split into two main fields. *Historical volatility* and *Implied volatility (IV)*. Historical volatility is calculated by estimating the standard deviation of returns over some finite historical time interval as in Equation (10). Thereby arguing that the best estimate of future realized volatility is historical realized volatility. In contrast, *implied volatility* (σ_{BS}) is the main measure of *anticipated*/perceived price risk and is defined as the input value of σ into Equation (8) that will generate a theoretical quote that is equal to the market or a quote generated by another model. The IV for a unique option can be found by extracting σ using the Black-Scholes pricing formula, such that

$$\sigma_{BS} := \left\{\sigma \ : \ C^{BS}(T, K, \sigma) = \mathcal{P}(K, T)\right\} \tag{11}$$

where $\mathcal{P}(K, T)$ denotes an option price and can be represented by a market quote or by a quote generated by any model at $t = 0$. This equation is solved by numerical solvers, for example the algorithm proposed by Stefanica & Radoicic (2017). The implied volatility *surface* is the volatility such that Equation (11) holds *across* a grid of strikes and maturities. Formally, one can define the (call option) implied volatility surface as

$$\Sigma_{BS} = \left\{\{\sigma_{T_j,K_k}\}_{j=0,k=0}^{m,n} \ : \ C^{BS}(T_j, K_k, \sigma_{T_j,K_k}) = \mathcal{P}(K_k, T_j)\right\} \tag{12}$$

where T_j denotes the jth maturity and K_k denotes the kth strike. Emprical data for the call option market implied volatility surface on the 8th of November 2019 for the S&P 500 index is shown in Figure 3.

The market implied volatility surface is highly complex and to have any hope of replicating it, one must treat volatility with extra care. *Stochastic volatility (SV)* is a joint model of prices and variance that has been suggested to model variable volatility surfaces by allowing prices to depend on the variance process and by extension, volatility. Hence, volatility is modelled as a latent stochastic process. In SV models, one generally tries to capture the *mean-reverting* property of volatility and generate the stochastic element by a standard Brownian motion that is correlated to the Brownian motion that produces the spot price process. These properties allows stochastic volatility models to be more dynamic than the Black-Scholes model and thus differ in pricing. There exist a vast amount of SV-models, this paper will only consider one such model, namely the *Heston model*.

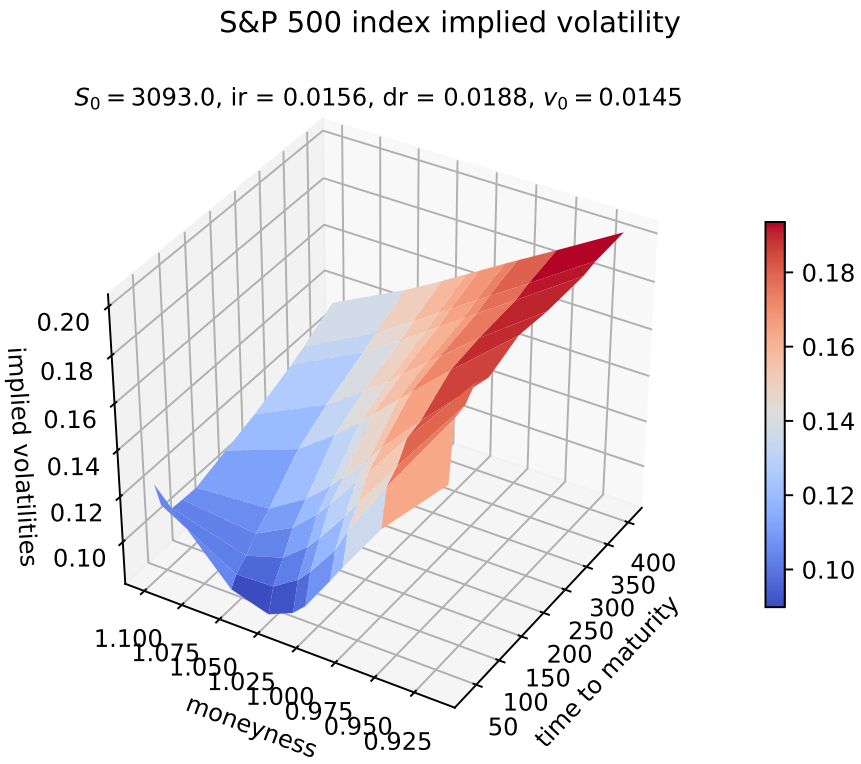


Figure 3: 2019-11-08 Implied volatility surface of S&P 500 index call options. Each point in the grid is the Black-Scholes implied volatility (IV) that generate the market price, so the IV surface is the set of volatilites that generate all market prices from the Black-Scholes model, as in Equation (12), such that $\sigma_{BS} \in \Sigma_{BS}$.

3.3 The Heston Model

One of the major assumptions of the Black-Scholes model, as noted in Subsection 3.1, is constant volatility. However, in reality we typically have skewed volatility surfaces. Many different models has been proposed in order to try and correct this shortfall of the BS framework. One such model is the *Heston model*. The Heston model, introduced by Heston (1993), is an extended version of the Black-Scholes model that takes stochastic volatility into account in which variance is modelled as a hidden Cox-Ingersoll-Ross process. The Heston model takes the leverage effect in to account as volatility and returns are correlated and is modeled by two correlated Brownian motions. One difference between the Heston model and other stochastic volatility models is the existence of a fast and semi closed solutions to European option pricing, as we shall see later in this section, which gives an indication as to why it is so popular Gatheral (2015).

Definition 3.3. Heston SDE's:

$$dS_t = rS_t dt + \sqrt{V_t} S_t d\tilde{W}_t^{(S)} \tag{13}$$

$$dV_t = \kappa(\theta^{\mathcal{H}} - V_t)dt + \sigma\sqrt{V_t dt} d\tilde{W}_t^{(V)} \tag{14}$$

where $\tilde{W} = \left(\tilde{W}^{(S)}, \tilde{W}^{(V)}\right)$ is a *correlated* 2-dimensional $\mathbb{Q}$-Wiener process and

$$d\tilde{W}_t^{(S)} d\tilde{W}_t^{(V)} = \rho dt \text{ so that } corr\left(\tilde{W}_t^{(S)}, \tilde{W}_t^{(V)}\right) = \rho\,.$$

The parameters in Equations (13) and (14) are interpreted as in Table 1.

Parameter	Interpretation
κ	Variance mean reversion speed
θ	Long term variance
V_0	Initial variance
σ	Volatility of variance
ρ	Spot v.s. variance correlation

Table 1: Parameter interpretations

The Heston model allows for semi-analytical solutions to option pricing. This is obtained by standard arbitrage arguments, risk neutrality and the Itô formula where one obtains a *partial differential equation (PDE)* which looks like

$$\begin{aligned}\frac{\partial C}{\partial t} + \frac{S^2 V}{2}\frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC + [\kappa(\theta - V) - \lambda V]\frac{\partial C}{\partial V} \\ + \frac{\sigma^2 V}{2}\frac{\partial^2 C}{\partial V^2} + \rho\sigma S\tilde{V}\frac{\partial^2 C}{\partial S \partial V} = 0\end{aligned} \tag{15}$$

where λ is the market price of volatility risk, for further description see Heston (1993). European call options that satisfies the PDE in Equation (15) are subject to various boundary conditions implied by rational choice and other factors. For the full mathematical description of these boundary conditions see Heston (1993). Intuitively, these constraints can be described for a call option $C(S, V, t)$ where $S \in [0, \infty]$, $V \in [0, \infty]$ and $t \in [0, T]$ as

- **Terminal payoff**. The terminal payoff of each call option is described by $C(S, V, T) = max(S - K)^+$.
- **Lower and upper boundary w.r.t spot price.** The lower bound, $C(0, V, t)$, is zero. As prices does not have a upper boundary, only the *delta*, which measure spot price risk for a option and, for a call option, defined as $\delta_C = \frac{\partial C}{\partial S}$, is bounded by 1.
- **Upper limit w.r.t volatility**. The upper bound for the option is equivalent to the spot price.

By analogy with the Black-Scholes call option pricing formula in Equation (8), Heston (1993) makes the following ansatz and propose a solution to the original Heston PDE in Equation (15) that respects the above boundary conditions

$$C(s, v, t) = sP_1(s, v) - e^{-r(T-t)}P_2(s, v). \tag{16}$$

If one consider the logarithm as a change of variables, i.e. $x = \ln s$, and substitute the proposed solution in Equation (16) into the Heston PDE in Equation (15), one finds that $P_1(s, v)$ and $P_2(s, v)$ must satisfy the PDE

$$\begin{aligned} \frac{1}{2}v\frac{\partial^2 P_j}{\partial x^2} + \rho\sigma v\frac{\partial^2 P_j}{\partial x \partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 P_j}{^2} + (r + u_j v)\frac{\partial P_j}{\partial x} \\ + (\kappa\theta^{\mathcal{H}} + -b_j v)\frac{\partial P_j}{\partial v} + \frac{\partial P_j}{\partial t} = 0, \qquad j = 1, 2 \end{aligned} \tag{17}$$

where

$$u_1 = \frac{1}{2},\ u_2 = -\frac{1}{2}, b_1 = k + \lambda - \rho\sigma,\ b_2 = k + \lambda.$$

In order for Equation (16) to satisfy the terminal payoff condition, the PDEs in Equation (17) (Equation (12) p.330 in Heston (1993)) are subject to the condition

$$P_j(s, v) = Pr(\ln S_T > \ln K | \ln S_t = s, V_t = v) \qquad j = 1, 2 \tag{18}$$

where S_t and V_t are driven by slightly altered versions of the SDEs in Equation (13) and (14), see p.331 Equation (14) in Heston (1993).

Thereby, $P_1(s, v)$ and $P_2(s, v)$ represents the probability of expiring in the money conditional on realizations of $\ln S_t$ and V_t, analogously to d_1 and d_2 in the Black-Scholes formula in Equation (8). However, the probabilities in Equation (18) are not analytically tractable, yet their *characteristic functions* satisfy the same PDEs as $P_1(s, v)$ and $P_2(s, v)$. Recall that the characteristic function defines the probability distribution of any random variable. Hence the characteristic functions can be used in order to evaluate the in-the-money (ITM) probabilities in Equation (18). The characteristic function of the log asset price is

$$\begin{aligned} f_j(\ln S_T, V, T; \phi) &= \mathbb{E}\left[e^{(i\phi \ln S_T)}\right] \\ &= \exp\left(C(T; \phi + D(T; \phi)V + i\phi \ln S_T\right) \end{aligned}$$

where D, C, and ϕ are functions such that the characteristic function satisfies Equation (17). The probabilities in (18) can be obtained semi-analytically by the inversion theorem, introduced by Gil-Pelaez (1951), as

$$P_j(\ln S_T, V, T; \ln(K)) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{\exp\left(-i\phi \ln(K)\right) f_j(\ln S_T, V, T; \phi)}{i\phi}\right] d\phi \tag{19}$$

where $Re(z)$ denotes the real part of any complex number z. The solution to Equation (19) can be obtained by fast numerical integration, hence the popularity of the Heston model in the industry Rouah (2013).

As the construction of hedging strategies for both the benchmark strategy and the neural network strategy depend on realizations of sample paths, one needs to be capable of simulating such paths in a effective manner. There exists two main methods for simulation, *discretization* and *exact simulation*.

Discretization is a method in which one samples from a approximation of stochastic differential equations (SDE) at discrete time points. For the Heston model, numerous discretization methods has been proposed, the most simple of which is the *Euler-Maruyama* method. The Euler-Maruyama (EM) method approximates the Heston SDE's by a *Markovian model* and thereby only depend on the previous state. This property of the EM scheme makes the simulation

procedure very simple as one can iteratively sample from S_t and V_t. Time is partitioned into small intervals and as these intervals become smaller, the approximation converges to the true solution to the SDE. The resulting model is

$$S_t = S_{t-1} + rS_{t-1}\Delta t + \sqrt{V_{t-1}}S_{t-1}\sqrt{\Delta t}Z_{t-1}^{(S)} \tag{20}$$

$$V_t = V_{t-1} + \kappa(\theta^{\mathcal{H}} - V_{t-1})\Delta t + \sigma\sqrt{V_{t-1}\Delta t}Z_{t-1}^{(V)} \tag{21}$$

$$\sqrt{\Delta t}Z_t^{(V)}, \sqrt{\Delta t}Z_t^{(S)} \sim \mathcal{N}(0, \Delta t) \qquad corr\left(Z^{(S)}, Z^{(V)}\right) = \rho.$$

However, if $2\kappa\theta^{\mathcal{H}} \leq \sigma^2$, V_t is not strictly positive. This condition is known as the *Feller-condition*. When the Feller condition is violated, an error is introduced in the cumulative distribution of the integrated volatility over time, see e.g. Bégin et al. (2015). This means that under this violation, the standard Euler-Maruyama scheme will not reflect the true variance process, and by extension, introduce a bias in the spot process as well. Thereby, the simulation scheme has been modified to handle this *discretization error*, by assigning functions that forces V_t to be positive. One can restate the variance process in the EM scheme as

$$V_t = f_1(V_{t-1}) + \kappa(\theta - f_2(V_{t-1}))\Delta t + \sigma\sqrt{f_3(V_{t-1})\Delta t}Z_t^{(V)}.$$

There exist at least three different schemes to assign functions, *reflection*: $f_1(V) = f_2(V) = f_3(V) = |V|$, *partial truncation* : $f_1(V) = f_2(V) = V, f_3(V) = V^+$ and *Full Truncation*: $f_1(V) = V, f_2(V) = f_3(V) = V^+$ in which $V^+ = max(V, 0)$. We shall see that our calibrated parameter combination does indeed violate this condition, thus introduce a bias in the simulation procedure which will affect the estimated hedging weights. However, one can easily constrain the search space for the calibration method to take these boundary conditions into account. This will be expanded upon in Section 6.

However, even when the Feller-condition is satisfied, if one considers the deviation of the *monte-carlo* price from the semi-analytical price in Equation (16) as a measure of convergence, there still exists some non-negligible bias in the EM scheme. In order to circumvent such biases one can consider *exact simulation* as introduced by Broadie & Kaya (2006). Exact simulation utilizes the distributional properties of V_t, described in Cox et al. (1985), which are such that the conditional distribution of $V_t|V_u$ for $u < t$ follows a *non-central chi-square distribution*. However, even though the exact simulation algorithm is very accurate since it does not explicitly rely on approximation of a continuous process by a discrete process, it suffers from some drawbacks related to its complexity and computational inefficiency. Hence, other discrete approximations has been introduced inspired by the exact simulation algorithm, see e.g. the *quadratic exponential* scheme introduced by Andersen (2007),which has similar accuracy as the exact simulation scheme but retains the computational efficiency, and relative simplicity, of the Euler-scheme Mrázek & Pospísil (2017). We shall however utilize the *Moment-matching scheme*, introduced in Andersen & Brotherton-Ratcliffe (2005). Which is a approximation where the variance process in the Euler-type approximation is adjusted such that its first two moments match that of a log-normal distribution. The discretization takes the form

$$V_t = \left(\theta^{\mathcal{H}} + \left(V_{t-1} - \theta^{\mathcal{H}}\right)e^{-\kappa\Delta t}\right)\exp\left(-\frac{1}{2}\Gamma_{t-1}^2 + \Gamma_{t-1}Z_t^{(V)}\right) \tag{22}$$

where

$$\Gamma_{t-1} = \ln\left(1 + \frac{\sigma^2 V_{t-1}\left(1 - e^{-2\kappa\Delta t}\right)}{2\kappa\left(\theta^{\mathcal{H}} + \left(V_{t-1} - \theta^{\mathcal{H}}\right)e^{-\kappa\Delta t}\right)^2}\right).$$

The reason we choose this scheme is because our approach is very dependent on computational efficiency since, as earlier noticed, our hedging method depend on a large number of realizations. Hence, we chose a discrete time process and the reason we chose the moment matching scheme is that we believe that it is the best in terms of balance between complexity, computational efficiency and approximating capability, see e.g. Rouah (2013) pp.202.

As the Heston model does not assume constant volatility, the implied volatility is variable and thereby more inline with empirical reality, see Figure 4, which displays a implied volatility surface generated from the Heston model.

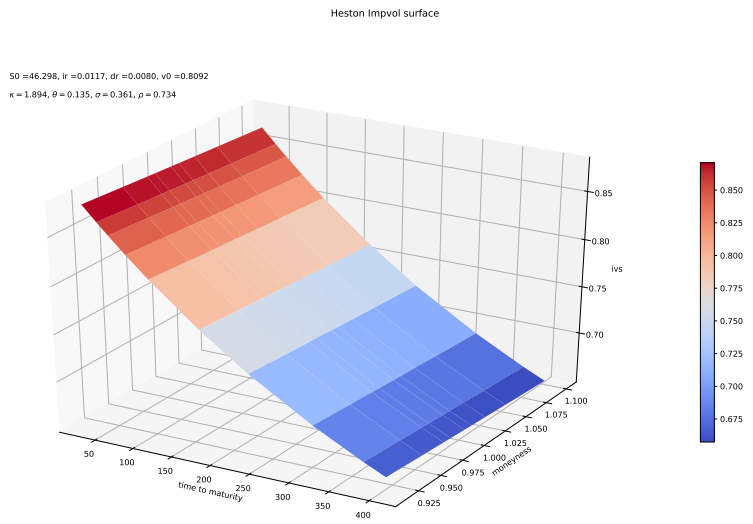


Figure 4: Example of implied volatility surface from a Heston model with parameters $\kappa = 1.894, \theta^{\mathcal{H}} = 0.135, \sigma = 0.361$ and $\rho = 0.734$. This parameter combination generates a rather simple implied volatility (IV) surface. The model has the capacity to generate skewness in all dimensions. For example, the model can capture what is called *volatility term structure* and *smile*. See Cont & Fonseca (2001).

The inability to generate variable implied volatility surfaces is, as earlier noted, one of the main shortcomings of the Black-Scholes model, that is, the Black-Scholes model cannot realistically be calibrated to available market data. In the Heston model however, the calibration can be done much better compared to the Black-Scholes model, that is one can find a parameter vector such that the fit of the generated implied volatility surface roughly matches that of the market Gatheral (2015). Generally, sellers of deep out-of-the-money (OTM) calls charges a larger risk premium relative to more at the money (ATM) options. This phenomena can be explained by the fact that a short call position has unlimited downside and thereby, for the seller to either hedge this exposure or accumulate unlimited downside risk, the seller will likely charge higher premium to finance a hedge or accumulate risk. Since the Heston model is able to produce a volatility surface that is roughly in line with the market, the Heston model to adapt better to changes in the distribution of returns than models that treat volatility as a constant.

Another nice property of the Heston model is the intuitive nature of its parameters and by extension, their impact on the implied volatility surface. $\theta^{\mathcal{H}}$ essentially shifts the implied volatility surface higher and by extension increases prices as higher long term mean variance implies a wider terminal distribution and more uncertainty, warranting higher prices. The correlation parameter, ρ, effects the skewness of the spot return distribution and hence the skewness in the surface w.r.t strikes. A negative ρ induce negative skewness in the terminal return distribution and vice versa since lower spot returns will be accompanied by higher volatility and by extension makes the terminal spot return more leptokurtic. The volatility of variance parameter, σ, will affect the kurtosis of the the terminal spot return distribution and by extension cause a steeper IV surface. When κ is not too large, then high values of σ cause more violent fluctuations in the volatility process and hence stretch the tails of the return distribution in both directions Mrázek & Pospísil (2017).

3.4 Classical Parameter Calibration

In this subsection we will introduce the concept of parameter calibration, which will serve as a basis for the discussion on deep calibration. The notation and setup of this subsection are presented as in Horvath et al. (2019).

Consider the Heston model parameterized by a set of parameters $\theta = (\kappa, \theta^{\mathcal{H}}, \sigma, \rho)$ so that $\theta \in \Theta$. We will analyze options characterized by strike K and maturity T The market pricing function $\mathcal{P}^{MKT}$ then takes two parameters T and K and outputs market prices of options characterized by their strike and maturity, which we denote by $\mathcal{P}^{MKT}(T, K)$ for each T and K. We then impose some kind of model, with an associated pricing function that maps the model parameters and option characteristics to prices, which in the case of the Heston model is Equation (16) for European call options. The common property of all model calibration methods is that they iteratively evaluate the model pricing function, or some approximation of it, on each instance of model parameters θ until a small enough distance, described by some appropriate distance function, between model prices and market prices is obtained. Formally, this can be described by the minimization

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}}\, d\left(P(\theta, T, K), \mathcal{P}^{MKT}(T, K)\right) \tag{23}$$

where d is some distance function. However, direct implementation of the model pricing function can be very slow or analytically intractable, especially for more complicated models. Then it may be more computationally efficient to consider some numerical approximation of the model pricing function $\tilde{P}$, and calibrate with this approximation instead. In this thesis we will use neural networks in the numerical approximation of $\tilde{P}$, as will be elaborated upon in Section 6. The optimization in Equation (23) can then be approximated by

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}}\, d\left(\tilde{P}(\theta, T, K), \mathcal{P}^{MKT}(T, K)\right). \tag{24}$$

One can let the function d in Equation (24) be related to the squared distance and as proposed in Horvath et al. (2019), then the calibration in Equation (24) becomes

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{k=1}^{n} \sum_{j=1}^{m} \omega_{T_j, K_k} \left(\tilde{P}(\theta, T_j, K_k) - \mathcal{P}^{MKT}(T_j, K_k)\right)^2 \tag{25}$$

where the dimensionality of the grid is given by $n \times m$, that is, n maturities and m strikes, and the weights ω is pre-specified and can reflect the relative importance of a specific option with respect

to liquidity etc. Classical model parameter calibration describes the process by which Equation (24) is minimized by some specified distance function. The minimization of the distance function in Equation (25) produces a non-linear least squares problem and is solved by numerical least squares algorithms such as the *Levenberg-Marquardt* algorithm. For a full description of the algorithm, the reader is referred to Mrázek & Pospísil (2017). In essence, the algorithm minimises the residual of the pricing map for each parameter combination θ and evaluates the improvement for changes in the parameters in which the increments is computed by solving a normal equation. This method introduces severe calibration bottlenecks as the normal equations have to be recalculated as the true pricing function is unknown and in option models where analytical solutions are sparse, the normal equations also have to be recomputed frequently. This means that the calibration task becomes unnecessarily computationally expensive.

Recall that the partial objective of this thesis is to implement a calibration method using neural networks. In essence, we will approximate the model pricing function $P(\theta, T, K)$ by a neural network. However, there are a some differences to equation (25) which will be detailed in Section 6.

3.5 Continuous time Martingale Pricing & Dynamic Replication

To understand the relation between hedging and pricing of financial derivatives as presented in Section 3.3 and 3.1, one must consider so called *dynamic replication*. This section will introduce the theoretical background to hedging of general contingent claims, both in the complete market case and the incomplete case, which will act as the basis of the discussion on numerical approximations of optimal hedging strategies by neural networks as such strategies can be seen as the implementation of the strategies in this section, with machine learning.

Our discussion of dynamic hedging is based on the work of Föllmer & Schweizer (1990). Consider a financial market where prices can be described by a stochastic process $S = (S_t)_{t\geq 0}$ on some filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$, as described in Section 2. If the market is *complete*, any *contingent claim* (a derivative whose future payoff depends on the value of the underlying asset), H, is a variable on the probability space at terminal time T and its payoff can be generated by a dynamic strategy based on S. If markets are free of arbitrage, there must exist a probability measure $\mathbb{Q} \sim \mathbb{P}$ such that S is a martingale under the equivalent measure and that $\mathbb{Q}$ and $\mathbb{P}$ share the same null sets Björk (2009).

A martingale is defined by the conditional expectation, $\mathbb{E}[S_t|\mathcal{F}_u] = S_u$ for all $u < t$, where $\mathcal{F}_t \in \mathbb{F}$. Under the martingale approach to derivative pricing, the price at time t of a contingent claim on S at maturity, is given by the conditional expectation $H_t = \mathbb{E}(H_T|\mathcal{F}_t)$ in which the terminal payoff H is denoted by H_T. According to the martingale pricing theory,the conditional expectation of the price change of H given the filtration is zero which in turn implies that H is a martingale under the probability measure and that successive changes in H are uncorrelated. The economic argument is thereby that all the information in the past that is useful for forecasting future prices, is already discounted in the current price and thus exclude arbitrage opportunities. This is a weaker assumption than that of Fama (1970), in which all information useful for forecasting the probability distribution of the next period is contained in the current price, generally referred to as the *random walk hypothesis*.

The martingale pricing theory assumes that S is a *semimartingale* under $\mathbb{Q}$, i.e. can be decomposed as a sum of a *local* martingale and a adapted process with bounded variation.

$$S = S_0 + M + A$$

where M is a local martingale and A is a adapted to the filtration, i.e. A_t is $\mathcal{F}_t$-measurable for each t. For a formal definition of a local martingale, see Protter (2005). The result of this is that one can define a Itô integral on S, with maximal generality Protter (2005). Consider the contingent claim H on the stochastic process S, then H can be considered a random loss incurred and a stochastic variable on the probability space of all square Lebesgue integrable functions at T that is

$$H \in \mathcal{L}^2(\Omega, \mathcal{F}_T, \mathbb{P}).$$

To hedge against this claim, a portfolio strategy must be used which involves S and a risk free money market instrument $Y = 1$, i.e. price variation is non existent and the risk free return is zero. Furthermore, amounts of stock and money market instruments, δ is a predictable process, in which predictability means that δ_t is $\mathcal{F}_{t^+}$ adaptive, and η is a adaptive process, defined over the same time interval as S. The value of the portfolio, Π_t is given as

$$\Pi_t = \delta_t S_t + \eta_t$$

and the cost C can be represented by a stochastic integral over S

$$C_t = \Pi_t - \int_0^t \delta_u dS_u. \tag{26}$$

We are only interested in a replicating portfolio such that $\Pi = H$, i.e. only admits hedging strategies that replicates the terminal payoff of the claim. Suppose further that H can be written under the Itô interpretation as

$$H = H_0 + \int_0^T \delta_u dS_u \tag{27}$$

if δ satisfies the technical integrabillity conditions of square integrable processes, a strategy can be defined as

$$\eta := \Pi - \delta \cdot S, \qquad \Pi_t = H = H_0 + \int_0^t \delta_u dS_u, \qquad (0 \leq t \leq T)$$

where we define $\delta \cdot S$ as $\delta \cdot S = \int_0^T \delta_u dS_u$. When one considers the definition of the cost process in Equation (26), simple mathematical manipulation leads to the observation that

$$\begin{aligned} \Pi_t &= C_t + \int_0^t \delta_u dS_u, \\ H_t = H_0 + \int_0^t \delta_u dS_u &\iff C_t = H_0 \end{aligned} \tag{28}$$

is *self financing* as $C_t = C_T = H_0$. Self financing refers to the concept of a portfolio which you do not consume or insert capital such that the purchase of new assets must be financed by the sale of a current asset Björk (2009). In our case this means that the Itô differential of the hedging portfolio Π is the integrand in the stochastic integral in Equation (28). Intuitively, this means that changes in the value of the replicating portfolio is only given by changes in the value of the price of the asset. Since Π is self financing, we have that Π produces the claim from the initial capital injection and thereby no further risk and costs arises and H is completely hedged by the strategy on S under the assumption of complete markets, for more details see Föllmer & Schweizer (1990). Hence, options are redundant as the replicating portfolio ,Π, perfectly replicates the cash flow of the contingent claim/option. Therefore, the value of H should equal the value of Π.

However, in *incomplete markets*, perfect replication is no longer possible as most claims carry intrinsic/residual risk. Intuitively this means that there is a replication error at maturity T, in the sense that there is a difference between the value of the replicating portfolio Π_T and the payoff of the contingent claim H_T which leaves a "gap-risk" for the seller of the derivative. In a complete market it will always be possible, as seen above, to find a replication strategy δ such that $\Pi_T = H_T$ almost surely. However, in a incomplete market, perfect replication is not attainable by definition since each claim carries intrinsic risk. For a incomplete market, the problem is not described by risk-elimination, but by risk-minimization as the residual risk is unhedgeable. As the strategy is now only minimizing risk, we are interested in finding a admissible hedging strategy such that it minimizes the residual risk

$$\mathbb{E}[(C_T - C_t)^2 | \mathcal{F}_t]$$

as the replication is not perfect. This implies that the objective of the hedging strategy is to minimize the error of replication. The cost process C for the hedging strategy is no longer self-financing as $C_T \neq C_t$, in fact it will be self-financing in only its expectation

$$\mathbb{E}[C_T - C_t | \mathcal{F}_t] = 0$$

and implies that H no longer admits to Equation (27) as one needs to consider that $C_t \neq C_T$ and is in fact a martingale. Thereby, the risk minimizing strategy needs to be modified in order for $\Pi_T = H$ according to the *Kunita-Watanabe decomposition*

$$H = H_0 + \int_0^T \delta_u dS_u + L_T \tag{29}$$

in which $L = \{L_t\}_0^T$ represents a *orthogonal* martingale process such that L is orthogonal to S and thus gives rise to the *unhedgable*/residual risk inherent in market incompleteness. A martingale is orthogonal if and only if their product is also a martingale. See e.g. Protter (2005) for orthogonality in terms of risk-minimization. The risk minimizing strategy is obtained

$$\eta = \Pi - \delta \cdot S$$

However the replicating portfolio

$$\begin{aligned} \Pi_t &= \mathbb{E}[H | \mathcal{F}_t] \\ &= H_0 + \int_0^t \delta_u dS_u + L_t \end{aligned} \tag{30}$$

no longer *perfectly* replicates the contingent claim due to the square integrable orthogonal martingale L. See Föllmer & Schweizer (1990) for further description of hedging in incomplete markets. One of the goals of this thesis is to numerically approximate δ in Equation (30) in discrete time by a so called a artificial neural network, via utility maximization and risk measures which would theoretically allow us to transcend specific model implied dynamics of S, for more details, see Section 4.

We shall now consider how one might obtain the hedging strategy $\{\delta_t\}_0^T$ by classical methods, which relies on specific model assumptions. To reiterate, we are interested in finding the hedging strategy that minimizes risk. Conceptually, the purpose of δ is to choose a "optimal" weight of S in the portfolio Π such that the replication error is, ideally, eliminated. As discussed earlier,

complete market hedging allows the strategy on S to perfectly replicate H such that the risk is totally eliminated. However, when hedging is conducted in incomplete markets, such weighting of S only minimizes the risk as the claim carries intrinsic risk as a result of L in Equation (3.5). If we assume the absence of arbitrage and complete markets, then a replicating portfolio should be risk free and should thereby earn the risk free rate. In the case of the Black-Scholes model, where dS is defined as in Equation (6), one can find analytic expressions for the hedging portfolio by letting $\Pi = -H + \frac{\partial H}{\partial S}S$ and applying Itô's lemma for two variables to the claim H on S. Using the arbitrage argument and complete markets we know that $\Delta\Pi = -\Delta H + \frac{\partial H}{\partial S}\Delta S$ should earn the risk free rate if it has no risk. As it turns out, $\Delta\Pi = r\Pi\Delta t$, which means that δ in Equation (28), in the case of the Black-Scholes model, is determined by

$$\delta_t = \frac{\partial H_t}{\partial S_t} \tag{31}$$

for each $t \in [0, T]$ and eliminates ΔW in the Itô differential equation for $\Delta\Pi$. For full derivation see Black & Scholes (1973). The practice in which this is achieved is called *delta-neutralization* and δ is called the hedging parameter. This result is very intuitive since one want to hedge consecutive changes in S which is only dependent on one Brownian motion, i.e. only one source of risk exist. δ is what is called a *greek*. Greeks are formally defined as the partial derivative of the claims value with respect to some factor that effects the value of the claim.

In the Heston model, where volatility a latent stochastic process, there exist another source of risk, namely $\tilde{W}_t^{(V)}$ in Equation (14), which will need to be hedged. Thus any portfolio containing S or claims on S will carry volatility risk. This means that, when one constructs a hedging portfolio H, one needs to insert a second hedging parameter in order to neutralize both Brownian motions in Equation (14). We will not cover the details of this concept, however it should be noted that using a delta hedging strategy in a stochastic volatility model will always ignore volatility risk.

As noted earlier, the calculation of delta is directly dependent on which stochastic differential equation (SDE) is used to describe the spot process in differential form. In the Black-Scholes model where S follows a geometric Brownian motion as in Equation (7), δ is calculated as $\mathcal{N}(d_1)$, where d_1 is defined as in Equation (9). For the Heston model, δ is calculated as P_1 in Equation (18). However, there exist model independent approximations of the hedge ratio that utilize *numerical differentiation*, specifically *finite difference* methods. Consider any continuous function $f(x)$, its first order derivative can be computed as

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}.$$

By approximating the limit with a small enough ε, the numerical derivative can be found. We shall now define H as a call option C. To reiterate, we are interested in approximating the partial derivative $\delta_t = \frac{\partial C_t}{\partial S_t}$ for all $t \leq T$, thereby we need to calculate the price of the call option $C_{\mathcal{H}(\theta)}(S_t + \varepsilon, V, T-t)$ at each t and $C_{\mathcal{H}(\theta)}(S_t - \varepsilon, V, T-t)$ by the fourier pricing method in Equation (16) and approximate the partial derivative by choosing a sufficiently small ε

$$\delta_t \approx \frac{C_{\mathcal{H}(\theta)}(S_t + \varepsilon, V, K, T-t) - C_{\mathcal{H}(\theta)}(S_t - \varepsilon, V, K, T-t)}{2\varepsilon} \tag{32}$$

for each $t \in [0, T]$. The results is the approximated hedging strategy such that δ defines the relative amount of stock that needs to be held in order to replicate the payoff of the claim.

This hedging strategy is maintained in practice by continuous recalculation of a positions delta and rebalanced accordingly. This hedging strategy will serve as the *benchmark* to which the neural network strategy is compared. However, as earlier noted, this method will still leave the volatility risk of C unhedged. We are aware of the fact that this will limit the capacity of the benchmark strategy to replicate the payoff of C. We have chosen this approach as we believe that the introduction of contingent claims on V is beyond the technical scope of this thesis.

4 Optimal Hedging in Discrete Time using Convex Risk Measures & the Quadratic Criterion

This section will present the theoretical background to hedging in incomplete and discrete time markets and expand the previous discussions on hedging, where we formulate the hedging problem as a numerical optimization task and introduce the concept of optimality under monetary risk measures, of which we introduce three measures with different properties. The formulation of the hedging problem as a numerical minimization task is crucial since it will enable the use of artificial neural networks as described in Buehler et al. (2019).

Expanding on hedging in incomplete markets in Subsection 3.5, we need to consider that time is not continuous in any practical implementation of hedging strategies and thereby the claim H is not admissible to the representation in Equation (29). Thereby, H needs to be discretized with respect to time so that

$$H = H_0 + \sum_{i=1}^{n} \delta_{t_i} \Delta S_{t_i} + L_T \tag{33}$$

in which $\Delta S_{t_i} = S_{t_i} - S_{t_{i-1}}$ approximates dS_t in the limit as $\Delta \to 0$ and $t \in [0, T]$. To reiterate, S_t represents the spot process for the financial security and δ represents a unique self-financing hedging strategy in which δ_t represents the hedging ratio based on the knowledge of $S_{t_{i-1}}$. Let

$$G_T(\delta) = \sum_{i=1}^{n} \delta_{t_i} \Delta S_{t_i}$$

represent the cumulative profit/loss of the dynamic hedging strategy and the cost process C is defined as

$$C_T(\delta) = \sum_{i=0}^{n} c_i(\delta_{t_{i+1}} - \delta_{t_i}).$$

If H is considered a random variable on the filtered probability space of square integrable functions, $H \in \mathcal{L}^2(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$. Then $H - c - G_T(\delta)$ represents the *net loss* incurred with initial capital c when transaction costs are not considered, i.e. $C_T = 0$. The risk associated to the size of the difference $H - c - G_T(\delta)$ is often referred to as *shortfall risk* or *replication error risk*. The simplest representation of H, and indeed how we will represent it, is a financial liability resulting from the sale of a European vanilla option. We will consider $H = -Z = -(S_T - K)^+$ with maturity T and strike K. The goal of a agent under incomplete markets in discrete time with no frictions would then be to minimize this incurred net loss.

In order to motivate the structure that has been introduced above, one needs to consider the concept of *utility based hedging*. In a complete market with no transaction costs and continuous trading, there exists a fair price, p_0 and a strategy δ such that $-Z + p_0 + G_T(\delta) - C_T(\delta) = 0$ holds $\mathbb{P}-$almost surely Föllmer & Leukert (2000). This price, p_0 ensures that the agent is indifferent

between the accumulation of risk, i.e. taking a position in $-Z$ and to holding no position at all. This in turn implies that the initial capital insertion, $c = p_0$ for the strategy to be self financing, and for the position to make sense. However, under the current market setting, one can not find a price such that the portfolio is completely risk free, as there exist residual risk by means of the orthogonal martingale L. However, one can approximate optimal hedging strategies under some *monetary risk measure* such that p_0 is approximated. This risk measure should have appropriate properties such that it reflects a financial agents monetary views on risk. As such there exists various utility constraints on such a measure. For a monetary risk measure to be a "good" measure of risk they either have to be *coherent* or *convex* risk measures, where coherency is a more strict property than convexity as all coherent risk measures are also convex measures of risk. When one considers the risk measures in terms of a hedging problem, one can consider these measures as evaluating the *shortfall*/replication error risk. The main idea in Buehler et al. (2019) is to use neural networks to approximate the strategies obtained from minimizing shortfall risk via convex measures. To this end we need to introduce the concept of convex risk measures. The following notation and setup originates from Buehler et al. (2019).

Definition 4.1. Assume that $X, X_1, X_2 \in \mathcal{X}$, represents financial positions ($-X$ represents a liability), where $\mathcal{X}$ denotes a given linear space of functions $X : \Omega \mapsto \mathbb{R}$. Then $\rho : \mathcal{X} \mapsto \mathbb{R}$ is a *convex risk measure* if it adheres to the following criteria:

1. Monotonicity: if $X_1 \geq X_2$ then $\rho(X_1) \leq \rho(X_2)$.
   A more favourable position requires less cash injection.
2. Convex: $\rho(\lambda X_1 + (1-\lambda)X_2) \leq \lambda\rho(X_1) + (1-\lambda)\rho(X_2)$, for $\lambda \in [0,1]$.
   Diversification lowers risk.
3. Cash-invariance: $\rho(X + c) = \rho(X - c)$.
   Additional cash injection reduces the need for more such injections

If ρ adheres to Definition 4.1, then one can consider the optimization problem,

$$\pi(-X) := \inf_{\delta \in \Delta} \{\rho(-X + G_T(\delta) - C_T(\delta))\} \tag{34}$$

where Δ denotes the set of all *constrained* hedging strategies, and means that π is a convex risk measure as π is *monotone decreasing* and *cash invariant* and if $C_T(\cdot)$ and Δ are convex Buehler et al. (2019).

To reiterate, we seek to minimize the risk associated with the replication error at maturity for a financial liability $-Z$ which admits to the representation in Equation (33) by choice of $\delta \in \Delta$. In order to apply convex risk measures to evaluate the replication error, we first have to assume that p_0 is observable, which we will proxy as the risk neutral price under $\mathbb{Q}$. If $\rho(-Z)$ denotes the minimal amount of capital to be added to the position $-Z$ in order for it to be acceptable under ρ, then $\pi(-Z)$ denotes the minimal amount to be charged in order to replicate $-Z$. Thus one can define the *indifference price* as p_0, as it is the solution to $\pi(-Z + p_0) = \pi(0)$. If π is a convex risk measure, one can utilize the its translation invariance property such that $p_0 = \pi(-Z) - \pi(0)$. This implies that approximating optimal hedging strategies δ of Z under the risk measure π, involves approximation of the price of Z. For further information, see Buehler et al. (2019).

We now proceed by introducing the various risk measures which we will use for the numerical approximation of optimal hedging strategies. Once again we follow the notation and setup in

Buehler et al. (2019). One reasonable way of minimizing risk could be to maximize the *expected utility* of the replication. Consider the *entropic* risk measure

$$\rho(Z) = \frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda Z)] \tag{35}$$

in which $\lambda > 0$ denotes the risk aversion parameter. Note that the expectation in Equation (35) is the *expected exponential utility* of Z. Proving that Equation (35) is a convex risk measure is nontrivial, and a detailed proof can be found in e.g. in Föllmer & Schied (2008).

The optimal hedging strategy under the entropic risk measure is obtained by substituting ρ in Equation (34) with the entropic risk measure

$$\pi(-Z) = \inf_{\delta \in \Delta} \left\{ \frac{1}{\lambda} \log \mathbb{E}\left[\exp(-\lambda(-Z + G_T(\delta) - C_T(\delta)))\right] \right\} \tag{36}$$

Another reasonable way to compute risk could be to consider the tail of the profit/loss distribution, at some quantile, and compute its expectation, this is called *conditional value at risk* or *expected shortfall* (ES). Expected shortfall is a "better" risk measure than the entropic risk measure, see Equation (35), in the sense that it is so called *coherent*. Coherent measures of risk are a stronger form of risk measures since, a convex risk measure that satisfies positive homogeneity, i.e. if $\rho(\lambda X) = \lambda \rho(X)$, and ρ satisfies the properties in Definition 4.1, then ρ is also a coherent risk measure. Expected shortfall $ES_\alpha(X)$ represents the expected *value at risk* at the α quantile of the loss distribution given that it has been exceeded, hence, it is also called average value at risk or conditional value at risk. The measure can be defined as

$$ES_\alpha(X) = \frac{1}{1-\alpha} \int_0^{1-\alpha} VaR_\gamma(X) d\gamma$$

where

$$VaR_\gamma(X) = \inf \{m \in \mathbb{R} \ : \ \mathbb{P}[X \leq -m] \leq \gamma\}$$

is the $1-\alpha$-level value at risk, and is nothing more than the average of the $1-\alpha$-quantile of the loss distribution. However, the formulation that is generally used for optimization purposes, and to prove coherency, is formulated in terms of the loss *tail mean*

$$ES_\alpha(X) = -\bar{x}_{(\alpha)}$$

where $\bar{x}_{(\alpha)}$ is the average in the α-quantile of the return distribution and is formally described as

$$\bar{x}_{(\alpha)} = \alpha^{-1} \left(\mathbb{E}[X \mathbf{1}_{\{X \leq x_{(\alpha)}\}}] + x_{(\alpha)}(\alpha - \mathbb{P}[X \leq x_{(\alpha)}]) \right) \tag{37}$$

in which $x_{(\alpha)} := \inf \{x \in \mathbb{R} \ : \ \mathbb{P}[X \leq x] \geq \alpha\}$ and represents the lower quantile of X and $\mathbf{1}_A$ is a indicator function over the set A. Note that $VaR_\alpha = -x_{(1-\alpha)}$. For proof of coherence which implies convexity, see Acerbi & Tasche (2002). If we consider the risk measure in terms of our current hedging problem, i.e. represent X as the replication of $-Z$, then one can reformulate the optimization in Equation (34) as

$$\pi(-Z) = \inf_{\delta \in \Delta} ES_\alpha(-Z + G_T(\delta) - C_T(\delta)). \tag{38}$$

Recall that p_0 is endogenously given in the marketplace, furthermore, according to Buehler et al. (2019), one can also fix a *loss function* $\ell : \mathbb{R} \mapsto [0, \infty)$. As we are interested in minimizing the loss at maturity, the optimal δ can be considered a minimizer to

$$\pi(-Z) = \inf_{\delta \in \Delta} \{\mathbb{E}[\ell(-Z + p_0 + G_T(\delta))]\} \tag{39}$$

which tells us that a numerical approximation of the optimal hedging strategy can be found by minimizing the shortfall risk weighted by the loss function. For a loss function to be appropriate for the purpose of measuring shortfall risk, one usually specifies a convex loss function as the convexity of the loss entails risk aversion. This formulation of the problem allows investors to systematically find an efficient hedge and interpolate between the extremes of full shortfall risk and maximal risk elimination by changing the risk aversion parameter. The optimal hedge under this shortfall measure does not only minimize the probability of shortfall occurrence, but also its size. For further explaination of the concept of *variance optimal hedging*, which is an utility based hedging strategy, see Schweizer (1995), in which the loss function is defined as $\ell(x) = x^2$. This is called *quadratic hedging* or *mean-variance hedging* which is variance optimal in the sense that optimality under this shortfall measure returns the hedging strategy with the smallest variance of the replication error

$$\pi(-Z) = \inf_{\delta \in \Delta} \left\{ \mathbb{E}\left[(-Z + p_0 + G_T(\delta))^2 \right] \right\} \tag{40}$$

and since the optimal quadratic shortfall hedge is optimal in terms of variance of terminal replication error.

$$\mathbb{E}[(-Z + p_0 + G_T(\delta))^2] = Var(-Z + G_T(\delta)) \tag{41}$$

However, neither $Var(X)$ or $\mathbb{E}[X^2]$ can be considered convex risk measures since $Var(X)$ is not convex when X, Y in Definition 4.1 are correlated, only when independent, and its monotonocity property can only be considered in a informal sense. If X is *preferable* to Y, then the variance of X is lower than the variance of Y. However, variance is translation invariant.

We have now considered three scenarios in which the optimal hedging problem has been formulated as a numerical optimization problem by finding optimal hedging strategies δ such that either the expected shortfall, entropic risk or a shortfall risk weighted by a loss function is minimized for the hedge portfolio. Hence we need to introduce the numerical method in which we seek to achieve approximate optimality under these measures. We will use so called *artificial neural networks* for this task.

5 Theoretical Background to Neural Networks

This section will introduce the reader to the theoretical background of Neural Network needed for the implementation done later in this thesis, and is structured as follows. First, the background and context of the use, as well as the formal definition of Neural Networks will be explained. Lastly, various parts and properties of Neural Networks will be covered, such as some terminology, neurons and the learning properties.

5.1 Background

An artificial neural network (ANN/NN) is a system inspired by the a brains neural network and is used to approximate functions. It consists of interconnected groups of nodes, which represents the neurons of the brain, with connections that represents the synapses of the brain. ANN are unique in its ability to "learn" from given examples, without being programmed with task-specific objectives.

The notation and setup given in this subsection is partly taken from Horvath et al. (2019). As noted earlier, ANN is used to approximate functions. Consider the function F^*, which is not available in closed form but can be approximated by a given input data x and output data $y = F^*(x)$. A feed forward network, which is the most simple ANN architecture, defines

$y = F(x, w)$ and the training of the neural network determines the optimal values of the networks parameters $\hat{w}$, which creates the best approximation $F^*(\cdot) \approx F(\cdot, \hat{w})$, given the input data y and output data x. In a more formal manner, the definition of NN is given by definition 5.1.

Definition 5.1. Neural Networks (Horvath et al. (2019)):
Let $L \in \mathbb{N}$ denotes the number of layers in the NN and the ordered list $(N_1, N_2 \ldots, N_L) \in \mathbb{N}^L$ denote the neurons (nodes) in each layer respectively. Furthermore, define the functions acting between layers as

$$\begin{aligned} w^l : \mathbb{R}^{N_l} \mapsto \mathbb{R}^{N_{l+1}} \text{ for some } 1 \leq l \leq L-1 \\ x \mapsto A^{l+1}x + b^{l+1} \end{aligned} \tag{42}$$

in which $A^{l+1} \in \mathbb{R}^{N_{l+1} \times N_l}$. b^{l+1} represents the *bias term* vector and each $A^{l+1}_{(i,j)}$ denotes the *weight* connecting neuron i of layer l with neuron j in layer $l+1$. If we then denote the collection of functions in the form of Equation (42) on each layer to $w = (w^1, w^2 \ldots, w^L)$ then the sequence w is the network weights. Then the NN $F(\cdot, w) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ is defined as:

$$F := F_L \circ \cdots \circ F_1 \tag{43}$$

where each component is in the form $F_l := \sigma_l \circ w^l$, i.e. $F_l := \sigma_l(w^l(x))$ as $\circ$ denotes a function composition. The first term in this, $\sigma_l : \mathbb{R} \mapsto \mathbb{R}$, denotes the *activation function* and is applied component wise to the vector w^l. w^l denotes the output given by w^{l-1}.

The justification of the use of ANN in this thesis is derived from the results of Hornik (1991) and is presented in Theorem 1 below. The theorem shows that if the objective is to approximate a real valued continuous function, a simple neural network with at least three layers, i. e. at least one hidden layer, will be successfully, and thus justifying the use of ANN in this thesis.

Theorem 1. *Universal Approximation Theorem (Hornik (1991)): Let $\sigma : \mathbb{R} \mapsto \mathbb{R}$ be a activation function. Let I_m denote the m dimensional hypercube $[0,1]^m$. The space of $\mathbb{R}$ valued functions on I_m is denoted by $C(I_m)$. Given any $\varepsilon > 0$ and any $g \in C(I_m)$, there exists an integer, N, constants $v_i, b_i \in \mathbb{R}$ and $w_i \in \mathbb{R}^m$ for $i = 1, .., N$ such that*

$$F(x) = \sum_{i=1}^{N} v_i \sigma\left(a_i^T x_i + b_i\right)$$

where

$$|F(x) - g(x)| < \epsilon, \quad \text{for all } x \in I_m.$$

Thereby, any continuous real valued function can be approximated arbitrarily well by a simple neural network consisting of at least 3 layers.

5.2 Neurons

As noted earlier, the main component of ANN's are neurons. There are two different types of artificial neurons, perceptrons and sigmoid neurons. Perceptrons are the oldest version of artificial neurons, developed in the 50s by Frank Rosenblatt in the paper Rosenblatt (1958) and influenced by McCulloch and Pitts paper McCulloch & Pitts (1943). The perceptron takes binary input $x_1, x_2, \ldots, x_N$ and produces an single output. However, there are limitations of the perceptrons. Since the output is binary, the change in weights will make big changes in the output of a neural network consisting of perceptrons. These will be inefficient for learning,

the process of adjusting model parameters in order if the network to attaining a more accurate result. The sigmoid neuron does not have this error. A sigmoid neuron, shown in figure 5, has an output value *between* 0 and 1.

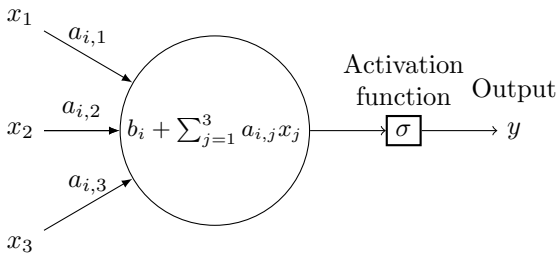


Figure 5: Example of a sigmoid neuron with three input variables (x_1, x_2 and x_3). The neuron adds the bias with the sum of the the input multiplied with the assign weights, which then sends through the sigmoid activation function σ.

It has weights $a_1, a_2, ...$ just like the perceptron, and an overall bias b. The output take the value $\sigma(x \cdot a + b)$, where σ is the *activation function*, in the format of a sigmoid function. A sigmoid function is a monotonic function which has a derivative shaped as a bell curve. One example of a sigmoid function is the logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Hence, the output from these neurons with input $x_1, x_2, ..., x_n$, weights $a_1, a_2, ..., a_n$ and bias b is

$$\frac{1}{1 + \exp(-\sum_{i=1}^{n} a_i x_i + b)}.$$

The selection of an activation function in this thesis is based on the work of Hornik (1991). The following is the interpretation of Hornik (1991) given by Horvath et al. (2019):

Theorem 2. *Universal Approximation Theorem for derivatives (Hornik (1991)): Let $F^* \in \mathcal{C}^n$ (the function has n orders of derivatives) and $F : \mathbb{R}^{d_0} \mapsto \mathbb{R}$ and $\mathcal{NN}^{\sigma}_{d_0,1}$ be the set of single-layer neural networks with activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$, input dimension $d_o \in \mathbb{N}$ and output dimension 1. Then, if the (non-constant) activation function is $\sigma \in \mathcal{C}^n(\mathbb{R})$, then $\mathcal{NN}^{\sigma}_{d_0,1}$ arbitrarily approximates F^* and all its derivatives up to order n.*

The results from Theorem 2 is that when selecting a activation function, its smoothness is of importance, where smoothness is the number of derivative orders of the function. For example, if $\sigma(x) = x$ then $\sigma \in \mathcal{C}^1(\mathbb{R})$, thus the NN will not be able to arbitrarily approximate the function and all its derivatives if $F^* \in \mathcal{C}^l$ and $l > 0$. But for example, if $\sigma(x) = \sigma_{Elu}(x) = \alpha(e^x - 1)$ and $\sigma_{Elu} \in \mathcal{C}^{\infty}(\mathbb{R})$ then the activation function is smooth and will be sufficient to arbitrarily approximate all functions, i.e $F^* \in \mathcal{C}^{\infty}$.

5.3 Learning

As mentioned earlier, what makes ANN attractive is its ability to "learn". The goal of an ANN is to approximate a function $f : \mathbb{R}^n \mapsto \mathbb{R}^n$, which, in simple terms, is done by training with labeled data, i. e. the input is fed to the ANN and outputs are produced, these are then compared with

the real values. To put this in mathematical terms, let's consider a *cost function* $C(\cdot)$, network parameters w, input $x_1, x_2, \cdots$ and $y_1, y_2, \cdots$, where $y_i = F^*(x_i)$ for all i, then the optimal weights, i.e. the weights that approximates $F(x, \hat{w}) \approx F^*(x)$ is found by,

$$\hat{w} = \underset{\text{all } w}{\operatorname{argmin}} \sum_{i=1}^{N_{Train}} C(F(x_i, w) - y_i).$$

Note that the learning is made using a *training set* with size N_{Train}. The solution $\hat{w}$ is found using *gradient descent* and *backpropagation*, which will be described in the next subsections.

5.3.1 Gradient Descent

Gradient-Based learning utilizes the gradient of a function. The gradient can be described as the slope, if the function has a one variable then the gradient is the same as the derivative. Let's consider the function $f : \mathbb{R}^m \mapsto \mathbb{R}^m$, then the gradient is denoted as ∇f. If we want to find the minimum of f then we could move in the opposite direction of the gradient ∇f, since the gradient always point towards the steepest change towards the maximum of the function. The method of gradient descent uses iterative steps in the opposite direction of the gradient to reach the minimum of the function. The step size of these iterations are stated when building the NN and is called the *learning rate*. According to Murphy (2012), when setting the learning rate there is a trade-off between slow learning and the possibility of overshooting. In the context of ANN, using the notation above, the target function is

$$J(w) = \sum_{i=1}^{N_{Train}} C(F(x_i, w) - y_i) \tag{44}$$

The general method of gradient descent described so far can be executed in different ways, using different methods. The most common one is *batch gradient descent*. This method uses all the examples z, where z are pairs of (x, y), to find one gradient descent. In mathematical terms with the objective function $F(\cdot)$, as written in Ruder (2016), this can be expressed as follows

$$w = w - \eta \cdot \nabla_w J(x, w)$$

where w is the network parameters, η is the learning rate and $\nabla_w J(x)$ is the gradient of the objective function with respect to w. A problem with this method is that it is applied to the entire data set and this causes inefficiencies for very large data sets.

To solve this problem, *stochastic gradient descent* (SGD) is introduced, according to Bottou (2012). An iteration of SGD instead takes the form of

$$w = w - \eta \nabla_w J(w; z^{(i)})$$

where $z^{(i)}$ is a randomly selected example, in the training set and $J(w; z^{(i)})$ denotes $C(F(x_i, w) - y_i)$. The model calculates the gradient and then immediately makes the correction to the model parameters for each example. The result is a more efficient algorithm for big data sets but not as reliable for smaller data sets since the cost will fluctuate based on the randomly picked examples.

One problem with SGD is that vectorization can not be applied as it is constrained by memory, and thus requires a step by step approach. To solve this, *Mini-batch gradient descent* is introduced. Mini-batch gradient descent is a mixture of ordinary gradient descent and SGD.

Instead of taking one example for each step, it takes a small amount of examples (mini-batches), $z^{(i:i+n)}$ in the form a sequence of examples z.

$$w = w - \eta \cdot \nabla_w J\left(w; z^{(i:i+n)}\right)$$

where n is the size of the mini-batch, and $J(w; z^{(i:i+n)})$ denotes

$$\sum_{i=1}^{n} C(F(x_i, w) - y_i). \tag{45}$$

Besides allowing vectorization in the code to make it work more efficiently, it also results in a smoother convergence.

An extension of SGD and mini-batch gradient descent is *adaptive momentum optimizer (Adam)*, first described in Kingma & Ba (2014), but the following description is given by Ruder (2016). Adam computes an adaptive learning rate for each parameter, instead of a constant one, in contrast to gradient descent methods. It uses an exponentially decaying average of past squared gradients v_t as well as an exponentially decaying average of past gradients m_t

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \cdot \nabla_w J(w_t) \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) \cdot (\nabla_w J(w_t))^2 \end{aligned}$$

where β_1 and β_2 is parameters, recommended to be set to 0.9 and 0.999 respectively. Vectors m_0 and v_0 is set to 0. m_t and v_t are estimates of the first moment and second moment of the gradients receptively. Especially in the beginning, there will be a bias towards 0. To solve this, Adam optimizer computes bias-corrected estimates.

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2} \end{aligned}$$

Finally, the updated network parameters are set to

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

where ϵ is a parameter suggested to be set to 10^{-8}.

An extended version of Adam is *Nestrov-accelerated Adaptive Moment Estimation (Nadam)*. Nadam modifies the momentum term m_t in Adam, according to the description given by Ruder (2016). To get a grasp of how Nadam works, one needs to understand *Nestrov Accelerated Gradient (NAG)*. If we first consider the momentum update rule:

$$\begin{aligned} g_t &= \nabla_{w_t} J(w_t) \\ m_t &= \gamma m_{t-1} + \eta g_t \\ w_{t+1} &= w_t - m_t \end{aligned}$$

we have that momentum involves taking a step in the direction of the current gradient, as well as in the direction of the previous momentum vector. NAGs steps are more accurately in the

gradient direction by updating the parameters with the momentum step before calculating the gradient.

$$\begin{aligned} g_t &= \nabla_{w_t} J(w_t - \eta m_{t-1}) \\ m_t &= \eta m_{t-1} + \eta g_t w_{t+1} = w_t - m_t \end{aligned} \tag{46}$$

Further, this can be modified by applying the look-ahead momentum vector to update the current parameters

$$\begin{aligned} g_t &= \nabla_{w_t} J(w_t) \\ m_t &= \gamma m_{t-1} + \eta g_t \\ w_{t+1} &= w_t - (\gamma m_t + \eta g_t) \end{aligned}$$

instead of applying the momentum vector twice, one time for updating gradient g_t and one time for updating the parameters w_{t+1}. NAG can now be applied to Adam by replacing the previous momentum vector with the current one. First, recall the expanded Adam-equation with the notation $g_t = \nabla_{w_t} J(w_t)$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right)$$

then $\frac{m_{t-1}}{1-\beta_1^t}$ can be replaced with the bias-corrected estimate $\hat{m}_{t-1}$ and we get

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_{t-1} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right).$$

Lastly, just as done in Equation (46), Nestrov momentum is added by replacing the previous steps estimated momentum with the current steps estimated momentum.

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_t + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right).$$

5.3.2 Backpropagation

The following subsection is based the description given by Nielsen (2015). While gradient descent is the method used for learning, backpropagation is the method used for finding the gradient. Let a_{ji}^l denote the weight for the connection from the ith neuron in the $(l-1)$th layer to the jth neuron in the lth layer. Let b_j^l denote the bias of the jth neuron in the lth layer and denote the activation of the same neuron be denoted as x_j^l. For a given activation function $\sigma(\cdot)$

$$x_j^l = \sigma \left(\sum_n a_{jn}^l x_n^{l-1} + b_j \right)$$

for notational convenience, we can consider matrices and vectors instead of scalars. Let A^l denote the weight matrix, b^l denote the bias vector and $x^l = \sigma(A^l x^{l-1} + b^l)$ denote the activation vector. Furthermore, let $z^l = A^l x^{l-1} + b^l$ denote the weighted inputs, hence

$$x^l = \sigma(z^l)$$

is only a function of the weighted input vectors. The objective is to find the partial derivatives

$$\frac{\partial C}{\partial a} \quad \text{and} \quad \frac{\partial C}{\partial b}$$

where C is the cost function, that evaluates the error of the model compared to the real value, y. In this example a quadratic cost function is used,

$$C = \frac{1}{2n} \sum_x ||y - F(x)||^2$$

To find these, backpropagation uses each neurons error. The error of the jth neuron in the lth layer is

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

This is called an error because the size of the partial derivative describes if C is close to zero and can be rewritten as

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

and represent it in matrix form

$$\delta^L = \nabla_x C \odot \sigma'(z^L)$$

where $\nabla_a C$ is a vector whose elements are $\partial C / \partial a_j^L$ and $\odot$ is the Hadamard product (elementwise product). The next equation in the backpropagtion process is

$$\delta^L = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

which describes the error in one layer in terms of the error in the next layer. We also get that the error is dependent on the change in the bias:

$$\delta_j^l = \frac{\partial C}{\partial b_j^l}$$

Lastly, the backpropagation uses an equation for the rate of change of the cost with respect to weights:

$$\frac{\partial C}{\partial a_{jk}^l} = a_k^{l-1} \delta_j^l$$

The complete backpropagation algorithm is presented in Algorithm 1.

Algorithm 1: Backpropagation

Result: Gradient is obtained by
Set activation x^l for the input layer. **for** *each* $l = 1, 2, ..., L$ **do**
  Compute $z^l = a^l x^{l-1} + b^l$ and $x^l = \sigma(z^l)$;
end
Compute vector $\delta^L = \nabla_x C \odot \sigma'(z^L)$;
Backpropagate the error **for** *each l = L-1, L-2,...,2* **do**
  Compute $\delta^l = ((a^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
end
Compute gradient using $\frac{\partial C}{\partial a_{jk}^l} = a_k^{l-1} \delta_j^l$

6 Deep Calibration

In this section we will consider the calibration of a Heston model through finding a approximation of the market implied volatility surface by means of neural networks and then use numerical

techniques to find parameters that best generate such a surface. As outlined in the introduction, see Section 1, the objective of this thesis is to analyse a coherent algorithm for both calibration and hedging of the Heston model. As such, this section will introduce the first partial objective of the thesis. Furthermore, this section relies on the theoretical background of the Heston model, see Subsection 3.3, calibration as presented in Subsection 3.4 and neural networks in Section 5. Although the framework and the context in which neural networks is used are presented in this section, the model specifications are disclosed in Subsection 8.1.

Deep calibration is the translation of the classical calibration problem into artificial intelligence. This was first introduced by Hernández (2016) and further developed by Horvath et al. (2019) to solve computationally intensive calibration of complex stochastic volatility models. We will only consider the application of such a calibration method to the Heston model, see Subsection 3.3. Intuitively what we are trying to achieve in this section is to find a neural network parameterization that approximates the implied volatility map, then find the model parameters that best generates that surface.

To reiterate, consider the Heston model $\mathcal{H}(\theta)$ that is parameterized by the parameter vector θ, belonging to the parameter space Θ, that is $\theta \in \Theta$ and its associated pricing function P, described in Equation (16). If the Heston model parameters is calibrated correctly, it can produce option prices roughly similar to market prices. However, for computational or analytical tractability reasons, it may be better to consider some approximation of the model pricing function, $\tilde{P} \approx P$. The classical calibration task can be described as choosing a parameter vector $\hat{\theta} \in \Theta$ that minimizes the distance between model quotes $P(\hat{\theta}, T, K)$, or a approximation thereof $\tilde{P}(\hat{\theta}, T, K)$, and market prices $P^{MKT}(T, K)$, as in Equation (23) or (24). However, we will first approximate the Heston model pricing function in terms of the implied volatility surface by a neural network, then apply classical calibration procedures in order to obtain the parameter set that best generates the trained volatility surface, weighted by some appropriate distance function. In other words, we will approximate the implied volatility surface generated by the Heston model with a neural network. Then we will use this neural network in the calibration of the Heston model by techniques similar to classical calibration as described in Subsection 3.4. The major advantage of using the neural network approach is that one can train a neural network as a approximator for the entire implied volatility surface. This, amongst other factors, means that the calibration using the trained neural network will be significantly faster compared to the classical calibration using the fast Fourier transform for the characteristic functions and then solve for the Black-Scholes implied volatility. This is crucial in real time pricing, since risk management, computations and calibrations of liquid options ,such as S&P500 options, often have to be done multiple times per minute during real time trading.

Motivated by Theorem 1 which states that any continuous function can be approximated by a simple neural network with at least 3 layers, i.e. at least 1 hidden layer, we know that the implied volatility approximation task can be solved by implementing a *feed forward network*, as described in Section 5. In Section 3.4 we discussed the calibration of the stochastic model through approximating the market pricing function by choosing a set of parameters that minimizes the sum of squared errors relative to market quotes. We shall now redefine the problem in terms of training a neural network to approximate the market implied volatility surface and find the parameters that generate the surface by the *two step approach* in Horvath et al. (2019). Let $\tilde{P}(\theta, T, K)$ and $P^{MKT}(T, K)$ in Equation (24) be replaced by $\Sigma_{BS}^{\mathcal{H}(\theta)} = \left\{\sigma_{BS}^{\mathcal{H}(\theta)}(T_j, K_k)\right\}_{j=1,k=1}^{n,\,m}$, such that $\Sigma_{BS}^{\mathcal{H}(\theta)}$ represents the *Black-Scholes implied volatility surface*, see Equation (12), gen-

erated by the Heston model parameterized by $\theta \in \Theta$, and $\Sigma_{BS}^{MKT} = \left\{\sigma_{BS}^{MKT}(T_j, K_k)\right\}_{j=1,k=1}^{n,\,m}$ are the implied volatilities to the corresponding market prices. Note that Σ_{BS}^{MKT} and $\Sigma_{BS}^{\mathcal{H}(\theta)}$ are actually approximations, since finding the inverse of Black-Scholes pricing function in Equation (8), relies on numerical approximations. The problem of finding optimal parameter sets can then be expressed as

$$\hat{\theta} = \underset{\theta\in\Theta}{\operatorname{argmin}}\, d\left(\Sigma_{BS}^{\mathcal{H}(\theta)}, \Sigma_{BS}^{MKT}\right)$$

in which $d : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \mapsto \mathbb{R}_+$ denotes an appropriate distance function. We are first interested in approximating the Heston model implied volatility map $\Sigma_{BS}^{\mathcal{H}(\theta)}$, by training a neural network as an approximator to the IV surface. Let $F := \{F(w; \theta, T_j, K_k)\}$ denote a neural network approximation of the IV-surface generated by the Heston model, parameterized by w (weights and biases), which can be formally expressed as learning the map

$$F : (w, \theta) \mapsto \Sigma_{BS}^{\mathcal{H}(\theta)} \tag{47}$$

by finding a optimal parameterization $\hat{w}$ of the neural network. In view of the notation of Section 5, let $\{\theta_i, K, T\}$ be a collection of input data and let $y_i = \Sigma_{BS}^{\mathcal{H}(\theta_i)}$ be the corresponding collection of output data. Recall from Section 5 that training the neural network function $F(w, \theta, K, T)$ on the data (θ_i, T, K, y_i) means finding optimal weights and biases w such that for any θ we have that $F(w, \theta, K, T) \approx \Sigma_{BS}^{\mathcal{H}(\theta)}$. Training the neural network on the training set (θ_i, K, T), where θ_i is chosen from a suitable set in the parameter space and the pair (K, T) are chosen over some standard grid of market implied volatilities and a neural network loss function $d(\cdot)$ as $d(x, y) = \sum_i \sum_j (x_{ij} - y_{ij})^2$, means finding optimal w and can be described by the optimization

$$\hat{w} := \underset{all\ w}{\operatorname{argmin}} \sum_{i=1}^{N_{Train}} \sum_{j=1}^{n} \sum_{k=1}^{m} \eta_{j,k} \left(F(w; \theta_i, T_j, K_k) - \sigma_{BS}^{\mathcal{H}(\theta_i)}(T_j, K_k)\right)^2 \tag{48}$$

where η_{jk} denotes weights that can be assigned in order to reflect specific approximation objectives for each point on the grid. For example can assign smaller η to at-the-money options and larger to tail options, however we shall not "favour" any option, hence $\eta_{j,k} = 1$ for all j, k.

We next describe the input data of the training set in more detail. Recall that we want to train a feed forward neural network over N_{Train} pseudo-random parameter samples (θ_i) generated by pre-specified prior distributions. We have specified the number of training iterations as $N_{train} =$ 100 000. For each iteration, we also have to simulate collections of market observable quantities, i.e. spot price S_0, risk free rate r, dividend rate q and initial variance V_0, as these are also inputs for the Heston model. We have chosen *uninformative* prior distributions, specifically a collection of beta and uniform distributions over reasonable boundaries for all quantities and parameters. Boundaries are specified in order to limit the training examples to a reasonable set of Heston model parameters and market obvservables, see Table 2 and these boundaries were chosen by

Parameter	Boundary
κ	(0.001, 15.)
$\theta^{\mathcal{H}}$	(0.001, 6.)
σ	(0.005, 4.)
ρ	(-0.999, 0.999)

Table 2: Calibration boundary conditions for optimization algorithm.

literature review, see Crisóstomo (2014). These parameter samples are then divided into training, validation and testing data, where validation data is used to save network parameterizations and thus evaluate the minimization in Equation (48) out-of-sample, and testing data is used as second step to "validate the validation". In order to find a minimum of the loss function, one needs specify a optimizer, we shall consider the *Nestrov accelerated adaptive momentum (Nadam)* optimizer for this task, outlined in Subsection 5.3.1. Recall that, conceptually, Nadam optimizer is a variant of gradient descent that is more likely to find global minimum than stochastic gradient descent since it utilizes the magnitude of the gradient. If the gradient is large, the optimizer will take larger steps since it builds momentum and accelerates, thereby, hopefully, "overshoot" local minimum.

Once the optimal w are obtained and the implied volatility map is learned, we shall denote the trained network as $\tilde{F}(\theta) = F(\hat{w}, \theta)$ and we denote *pixel-wise* prediction as $\tilde{F}(\theta)_{j,k} = F(\hat{w}, \theta, T_j, K_k)$. If one retains the "least squares" distance function in Equation (48), the calibration problem is formulated as finding parameters such that one minimizes the difference between the neural network approximation of the IV-surface and the market.

$$\hat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{F}(\theta)_{j,k} - \sigma_{BS}^{MKT}(T_j, K_k) \right)^2 \tag{49}$$

A more intuitive formal description of the calibration step can be presented as

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{\sigma}_{BS}^{\mathcal{H}(\theta)}(T_j, K_k) - \sigma_{BS}^{MKT}(T_j, K_k) \right)^2, \tag{50}$$

where $\tilde{\sigma}_{BS}^{\mathcal{H}(\theta)}(T_j, K_k) = \tilde{F}(\theta)_{j,k} = F(\hat{w}, \theta, T_j, K_k)$. We deem this to be a more intuitive explanation as it expresses the optimization in terms of a more explicit classical calibration problem.

There exist several optimizers that can solve the minimization in Equation (49). These optimizers differs mainly on computing time. We have selected *differential evolution* (DE) for this task. DE is, as the name suggests, a *evolutionary algorithm*, inspired by natural evolution of species, that can solve numerical optimization problems in a wide variety of fields. The reader is referred to Appendix A for more detailed description of the DE-optimization method.

As we have discussed in Subsection 3.3, simulation schemes are somewhat limited by the Feller condition.The Feller condition is based on the observation that if $2\kappa\theta^{\mathcal{H}} \leq \sigma^2$, then there is a non-zero probability of sampling negative values of V. In order to avoid negative values of the variance process, one can restrict the parameter search space such that the condition is met by, dynamically, adjusting the upper boundaries of Θ such that $2\hat{\kappa}\hat{\theta}^{\mathcal{H}} \leq \hat{\sigma}^2$ is met. If the feller condition is applied to Equation (49), one obtains the constrained optimization

$$\begin{aligned} \hat{\theta} := & \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{F}(\theta)_{j,k} - \sigma_{BS}^{MKT}(T_j, K_k) \right)^2 \\ & \text{Subject to : } 2\kappa\theta^{\mathcal{H}} \leq \sigma^2 \end{aligned} \tag{51}$$

where we remind the reader that $\hat{\theta} = (\hat{\kappa}, \hat{\theta}^{\mathcal{H}}, \hat{\sigma}, \hat{\rho})$ denotes the calibrated parameter vector. However, the Feller condition has created quite the divide in both practice and industrial applications. Option prices generated by the model is still valid even upon violation Cui et al. (2017),

and the price of the underlying asset S is still a martingale under the risk neutral measure Ribeiro & Poulsen (2013), but simulations may be impaired for severe violations. In fact, the unconstrained calibration in Equation (49) returns better fit than Equation (51) according to Ribeiro & Poulsen (2013) and is also what our results indicate, see Section 8.1. Furthermore, it is well established, especially in foreign exchange markets , see e.g. Table 6.3 pp.99 Clark (2011), that unconditional calibration usually violates the Feller condition, and is also usually the case when calibrating to various equity indices, see Table 6.2 pp.156 Rouah (2013). Furthermore, we see that on p.1283 in Buehler et al. (2019), the Feller condition is also ignored. Therefore, we have decided to report unconstrained calibration results, i.e. using the optimization in Equation (51), and the Feller constrained minimzation in Equation (51), in the numerical results Section 8.1.

7 Deep Hedging

This section will provide the reader with the theoretical background to the methodology of hedging using neural networks by reformulating the hedging objective, in terms of finding optimal parameter sets. As such, the following reasoning heavily depend on hedging theory, covered in Section 3.5 and 4 as well as understanding of neural networks, see Section 5. First, we will define the market setting and conditions in which we will apply the deep hedging algorithm in Section 7.1. Then, in Section 7.2, we will formally describe the deep hedging algorithm in more detail, including an example of how the neural network approximates optimal hedging strategies for a specific risk measure.

As the name suggests, *deep hedging* is the process by which derivatives are hedged exclusively with deep learning techniques and was initially researched and implemented by Buehler et al. (2019). The approach is totally different from the classic replication techniques (see Section 3.5). In many ways however, *deep hedging* is still very classical as it can be seen as the implementation of hedging in incomplete markets, see Section 4, with machine learning. Thus, a problem that was inaccessible before, becomes accessible by the virtue of modern computational technology and the subsequent rise in the applicability of AI.

Deep hedging is a natural consequence of the practical application of automated hedging techniques. Initially, these automated hedging strategies was implemented through the use of classic greek models. However, classical hedging models like the Black-Scholes model, assumes conditions that conflict with empirical reality such as constant volatility. The problem with classical models is that they presume that the model specific parametric description of reality is a good enough approximation. For example, the Heston model assumes that the dynamics between instantaneous variance and spot returns evolve according to the system of SDE's in Equations (13) and (14). Deep hedging on the other hand, is a purely data driven approach that attempts to automatically implement hedging strategies that circumvents the replication theory and by extent, does not rely explicitly on any model describing market dynamics. Furthermore, deep hedging lends it self to be applicable to the cases in which the classical hedging models do not work very well. For example in markets where there is not a lot of buyers and sellers such as the OTC-derivative market. We will however only consider the case of liquid European vanilla options.

7.1 Market Setting & Objective

This section will introduce the basic market setting on which we will define the deep hedging algorithm similarly to Buehler et al. (2019). This means we reintroduce the hedging portfolio

and terminal pnl distribution and the discrete versions of the stochastic integrals in Section 3.5, i.e. trading strategy and transaction costs.

We will consider a discrete time market with terminal time T and trading is only allowed at $0 = t_0, \ldots, t_n = T,\ t \in [0, T]$. We assume that $(S_{t_i})_{i=0,\ldots,n}$ evolves by the discrete version of the Heston model parameterized by $\mathcal{H}(\hat{\theta})$, where $\hat{\theta}$ is obtained by deep calibration as described in Section 6.The filtered probability space is fixed on $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ in which $\mathbb{F} := (\mathcal{F}_{t_i})_{i=0,\ldots,n}$ is the filtration generated by the discrete versions of the Brownian motions in Equation (13) and (14) and $\mathcal{F}_{t_i}$ contains all relevant information at time t_i. Furthermore, S is adapted to the filtration. The contingent claim that we seek to hedge $-Z$ is a $\mathcal{F}_T$ measurable random variable on the filtered probability space. Intuitively, the above conditions means that we seek to hedge in a incomplete market setting as described in Section 3.5.

Recall from Section 4 that the strategy applied in order to hedge $-Z$ is implemented by trading in S with the sequence of weights $\delta = (\delta_{t_i})_{i=0,\ldots,n}$, as we ignore volatility risk for ease of implementation. If one represents $-Z$ as a liability and injects cash at t_0, the terminal wealth is defined as

$$\mathrm{PnL}_T(Z, p_0, \delta) = -Z + p_0 + G_T(\delta) - C_T(\delta) \tag{52}$$

in which the agent sells a contingent claim and

$$G_T(\delta) = \sum_{i=0}^{n} \delta_{t_i} \Delta S_{t_i} \tag{53}$$

denotes the gains from the hedging strategy and where C is defined as the cost of trading in a discrete market

$$C_T(\delta) = \sum_{i=0}^{n} \varepsilon \mid \delta_{t_i} - \delta_{t_{i-1}} \mid \tag{54}$$

and p_0 represents the initial cash injection, which we have shown in Subsection 3.5 to be equivalent to the market price that the trader receives when selling the option. In a complete market, Equation (52) would be zero, however, this is not the case in our current setting, hence optimizing a strategy will imply finding a price that minimizes the PnL of the hedging portfolio under some suitable risk measure. Furthermore, we have specified the settings, see Section 4, in which we will describe optimality in terms of various risk measures, mainly convex measures of risk.

In order to find close-to-optimal hedges under these measures, we need to employ some numerical technique. As stated earlier, we shall consider neural networks for this task.

7.2 Approximation of Optimal Hedging Strategies by Neural Networks

This section will cover the theoretical background to the deep hedging algorithm, and link it to our previous discussions on hedging under convex risk measures in Section 4 and neural networks in Section 5. First, in order to justify the application of neural networks as *approximators* of optimal hedging strategies, we will reformulate the *constrained* hedging problem , see Section 4, in terms of a *unconstrained* problem, as presented in Theorems 1 and 2. We have chosen to only reformulate the hedging problem for the *entropic risk* in Equation (36), because of notational convenience and in order to follow the setup of Buehler et al. (2019). However, the methodology presented below easily extends to the other risk measures in Section 4. Then we utilize the unconstrained set of admissible hedging strategies in order to formulate the hedging problem under

convex risk measures as a optimization of neural network parameters. Lastly, we illustrate how the neural network minimizes the discrete objective function using stochastic gradient descent and mini-batch training.

As discussed in Section 4, expected shortfall, entropic risk, and the quadratic criterion[1] are reasonable monetary measures of shortfall risk in incomplete markets as discussed in Section 4. recall that the entropic risk measure is convex and expected shortfall is a coherent risk measure, which implies convexity. When one consider optimal hedging strategies under such measures, one can seek optimality in terms of minimizing the PnL-distribution in Equation (52), which is called minimizing shortfall risk or replication error. We shall now, at least in part, formulate close-to-optimal hedges under the entropic measure as a problem such that Theorem 1 and 2 can be applied, where the notation originates from Buehler et al. (2019). First some technical conditions must be considered regarding constrained and unconstrained hedging strategies. The idea is that the constrained strategy δ in Equation (36) is replaced with a unconstrained strategy $D \circ \delta$ so that the unconstrained formulation of the entropic hedging objective in Equation (36) can be described as

$$\pi(-Z) = \inf_{\delta \in \Delta^u} \frac{1}{\lambda} \log \mathbb{E}(\exp(-\lambda[-Z + G_T(D \circ \delta)) - C_T(D \circ \delta)]) \tag{55}$$

where Δ^u is the unconstrained set of admissible hedging strategies. The reader is referred to Buehler et al. (2019) for further discussions with respect to these technical conditions. One can now seek to approximate Equation (55), by neural networks. Following the notation and setup of Buehler et al. (2019), we have that the space of admissible *neural network* hedging strategies $\Delta_M \subset \Delta^u$, to be the space of all possible output data from neural networks with a architecture that Buehler et al. (2019) calls *semi-recurrent* neural networks

$$\begin{aligned} \Delta_M &:= \left\{ (\delta_{t_i})_{i=0,\dots,n-1} \in \Delta^u \ : \ \delta_{t_i} = F_{t_i}(S_{t_i}, \delta_{t_{i-1}}),\ F_{t_i} \in \mathcal{NN}_{M,d_0=2,d_1=1} \right\} \\ &= \left\{ (\delta^{\tilde{w}}_{t_i})_{i=0,\dots,n-1} \in \Delta^u \ : \ \delta^{\tilde{w}}_{t_i} = F^{\tilde{w}_{t_i}}(S_{t_i}, \delta_{t_{i-1}}),\ \tilde{w}_{t_i} \in \widetilde{\mathcal{W}}_{M,d_0=2,d_1=1} \right\}. \end{aligned} \tag{56}$$

Here, $(F_{t_i})_{i=1,\dots,n-1}$ is a sequence of feed forward neural network (FNN) functions, indexed time, in a abstract space of all semi-recurrent networks $\mathcal{NN}^{\sigma_{Elu}}_{M,d_0=2,d_1=1}$ with parameter dimension M, input dimension 2 and output dimension 1 and $\widetilde{\mathcal{W}}_{M,d_0=2,d_1=1}$ is its corresponding the abstract space for its parameters. Intuitively, the right hand sides in Equation (56) means that we define a sequence of neural networks ordered by time such that for each t_i, $i = 0, 1, \dots, n-1$, we have a neural network F_{t_i}, parameterized by $\tilde{w}_{t_i}$ whose output δ_{t_i}, is an input in the neural network at the next time step $F_{t_{i+1}}$ parameterized by $\tilde{w}_{t_{i+1}}$, which output feeds into the next FNN in the sequence, up until t_{n-1}. In other words, the neural network $\mathcal{NN}^{\sigma_{Elu}}_{M,d_0,d_1}$ is the abstract space of sequences of feed forward neural networks $(F_{t_i})_{i=0,1,\dots,n-1}$ parameterized by $\widetilde{\mathcal{W}}_{M,d_0,d_1}$, such that each element in the sequence $(F_{t_i})_{i=0,1,\dots,n-1}$ is parameterized by a corresponding element in the sequence $(\tilde{w}_{t_i})_{i=0,\dots,n-1}$ where the resulting neural network is denoted by $(F^{\tilde{w}_{t_i}})_{i=0,\dots,n-1}$. For visualization of the neural network architecture, see Figure 12. Equation (56) tells us that the set of admissible neural network hedging strategies is all the unconstrained hedging strategies that are outputs of semi-recurrent neural networks with parameter dimension M. Thus, the

[1] As noted in Section 4, the quadratic criterion is neither a convex or coherent monetary risk measure. However, approximating $-Z$ in $\mathcal{L}^2$, i.e. the space of square Lebesgue integrable functions, by stochastic integrals $G_T(\delta)$ of a given discrete stochastic process S, leads to the variance-optimal hedging strategy, which is equivalent to optimality under the quadratic criterion. Hence, approximating optimality of the expectation of the PnL-distribution for a hedging portfolio, weighted by a quadratic shortfall measure is a prudent setup for deep hedging and was also used in Buehler et al. (2019) as an example.

problem of finding neural network hedging strategies can be reformulated as parameterising a semi-recurrent neural network. Note that the reason why each feed forward network only have to depend on the "current" S_{t_i} and the previous hedging weight $\delta_{t_{t-1}}$ is that S is a discrete Markovian process and δ_t is $\mathcal{F}_t$-measurable for all t, however the setup and notation could be easily extended to cover non-Markovian processes, as shown in Buehler et al. (2019).

In the case of the entropic risk measure, we replace Δ^u with Δ_M in Equation (55) and obtain the formulated numerical optimization of finding optimal parameter combinations for the neural network

$$\begin{aligned}\pi(-Z) &= \frac{1}{\lambda}\log \inf_{\delta\in\Delta_M} \mathbb{E}(\exp(-\lambda[-Z + G_T(D\circ\delta) - C_T(D\circ\delta)]) \\ &= \frac{1}{\lambda}\log \inf_{\tilde{w}\in\widetilde{\mathcal{W}}_M} \mathbb{E}(\exp(-\lambda[-Z + G_T(D\circ\delta^{\tilde{w}}) - C_T(D\circ\delta^{\tilde{w}})]).\end{aligned} \tag{57}$$

Let $J(\tilde{w}) = \mathbb{E}(\exp(-\lambda[-Z + G_T(\delta^{\tilde{w}}) - C_T(\delta^{\tilde{w}}))])$. Then the approximately optimal hedging strategy in terms of the sequence $\tilde{w} = (\tilde{w}_{t_i}, \tilde{w}_{t_2}, \ldots, \tilde{w}_{t_n})$ describing neural network parameters for each feed forward neural network at each time t_i for $i = 1, \ldots, n$, under the entropic measure is defined as

$$\pi^M(-Z) = \frac{1}{\lambda}\log \inf_{\tilde{w}\in\widetilde{\mathcal{W}}_M} J(\tilde{w}). \tag{58}$$

The objective of approximating a optimal strategy is reduced to the constrained problem of finding optimal parameters (weights and biases) for our neural network such that $\tilde{w}$ minimizes the loss function $J(\tilde{w})$. This strategy can arbitrarily well approximate strategies in δ since

$$\lim_{M\to\infty} \pi^M(-Z) = \pi(-Z) \tag{59}$$

in which, as earlier, M denotes the dimensions of the parameter space for the neural network. One result of this is the convergence of the hedging price, defined as $\pi(-Z) - \pi(0)$ to the semi-analytical price of Z, obtained by fourier pricing as described in Section 3.3. The reader is referred to Buehler et al. (2019) for proof of the approximation as it is beyond the technical scope of this thesis.

We now proceed by formally describing the computation of $\tilde{w}$ in (58), by means of a neural network. Here we utilize the theoretical background from Section 5 in order to formally define the process in by which the risk measure is minimized by choice of $\tilde{w}$. As discussed in Section 5, the process in by which a neural network learns is by finding the gradient of the cost function and step in the opposite direction of the gradient, changing the network parameters along the way, and, hopefully, find a global minimum of the cost function.

Since we have discretized the stochastic integrals into sums in Equation (53), we now have a discrete sample space $\Omega = (\omega_1, \ldots, \omega_N)$. Let $J(\tilde{w})$ denote the expectation in Equation (57), it can now be computed as

$$\begin{aligned}J(\tilde{w}) &= \sum_{\omega\in\Omega} \exp(-\lambda[-Z(\omega) + G_T(D\circ\delta^{\tilde{w}})(\omega) - C_T(D\circ\delta^{\tilde{w}})(\omega)])\mathbb{P}(\omega) \\ &= \sum_{i=1}^{N} \exp(-\lambda[-Z(\omega_i) + G_T(D\circ\delta^{\tilde{w}})(\omega_i) - C_T(D\circ\delta^{\tilde{w}})(\omega_i)])\mathbb{P}(\omega_i).\end{aligned} \tag{60}$$

However, since training neural networks relies on vast amounts of data, N will be very large and therefore it may be computationally infeasible to perform gradient descent. To address this

problem, one may consider mini-batch training, as discussed in Section 5, where the concept is that one selects m subsets $(\Omega^{(1)}, \ldots, \Omega^{(m)})$ where $\Omega^{(j)} = (\omega_1^{(j)}, \ldots, \omega_{N_{Batch}}^{(j)}) \subset \Omega$ by uniformly sampling over Ω m-times such that each *mini-batch* has N_{Batch} elements. One can then compute the expectation of each batch separately as

$$J_j(\tilde{w}) = \frac{N}{N_{Batch}} \sum_{i=1}^{N_{Batch}} \exp(-\lambda[-Z(\omega_i^{(j)}) + G_T(D \circ \delta^{\tilde{w}})(\omega_i^{(j)}) - C_T(D \circ \delta^{\tilde{w}})(\omega_i^{(j)})])\mathbb{P}(\omega_i^{(j)}) \quad (61)$$

for each $j = 1, \ldots, m$ and where $\frac{N}{N_{Batch}}$ is a scalar value to make sure that the expectation becomes the mini-batch sample average since the probabilities in Equation (61) are uniform with $\mathbb{P}(\omega^{(j)}) = 1/N$, as on page 1283 in Buehler et al. (2019). Evaluating the expectation in Equation (61) for each mini-batch is much more computationally efficient than computing the expectation over the entire sample space as in Equation (60) for each iteration in the minimization procedure. One can now as earlier Minimizing the target function $J(\tilde{w})$ by stochastic gradient descent, as described in Section 5.3.1, over each mini-batch can be described by starting with a initial guess and stepping in the opposite direction of the gradient for each mini-batch until local minimum is potentially obtained, or more formally

$$\tilde{w}^{(j+1)} = \tilde{w}^{(j)} - \eta_j \nabla J_j(\tilde{w}^{(j)}), \quad (62)$$

for some small $\eta_j > 0$ and initial guess $\tilde{w}^{(0)}$. Stochastic gradient descent finds the local minimum of J as the number of batches $j \to \infty$. Equation (62) describes the *simple* gradient descent algorithm, however, keeping computational efficiency in mind, and the discussion in Section 5.3.1, resulted in us choosing adaptive momentum optimizer (Adam) instead, which updates the network parameters $\tilde{w}$ with

$$\tilde{w}_{j+1} = \tilde{w}_j - \frac{\eta}{\sqrt{\frac{v_j}{1-\beta_2}} + \epsilon} \cdot \frac{m_j}{1-\beta_1}$$

where

$$m_j = \beta_1 m_{j-1} + (1-\beta_1)\nabla_w J(\tilde{w}_j)$$
$$v_j = \beta_2 v_{j-1} + (1-\beta_2(\nabla_w J(\tilde{w}_j)))^2.$$

The above methodology easily extends to the rest of the monetary risk measures that we consider in this thesis, i.e. expected shortfall and the quadratic criterion, see Section 4 as Equation (61) can be generalised to be used with other risk measures, see Buehler et al. (2019).

8 Implementation & Numerical Results

This section will discuss the implementation of the deep calibration and hedging respectively. In Section 8.1 we first discuss practical considerations when training the deep calibration algorithm, introduced in Section 6, such as the feed forward neural network architecture, number of training examples etc. Then we proceed by illustrating some results from deep calibration of the Heston model, both to synthetic data out of sample and to real market in order to illustrate the accuracy of the calibration process. We will also illustrate differences in terms of fitting when the Feller condition, as discussed in Section 6, is both enforced and violated, .

In Section 8.2, we consider results from the deep hedging algorithm. We illustrate and discuss the semi-recurrent architecture of the neural network, as introduced in Equation (56) and

discussed in the subsequent text in Section 7. Then we proceed illustrating simulated terminal profit/loss distributions, i.e. realizations of Equation (52), for all risk measures introduced in Section 4 and compare the results to the benchmark delta hedging strategy introduced in Section 3.5. Furthermore, we will also vary the level of risk aversion for expected shortfall and the entropic risk measure, i.e. varying α and λ in Equation (37) and (35) respectively. We will also display and comment on descriptive statistics for each simulated pnl distribution and compare them to the benchmark delta hedging strategy. Then, we illustrate pathwise characteristics of the deep hedging strategies and, yet again compare them to the benchmark strategy. Lastly we illustrate the impact of introducing transaction costs on profit and loss distribution of the expected shortfall deep hedging strategy and display sample path and contrast them towards when transaction costs are not included. We also discuss the "joint" methodology of deep calibration and hedging.

We implement both deep calibration and deep hedging through Python in which we primarily use the packages Numpy, QuantLib and Tensorflow. The resulting code is available at the project GitHub repository, which is extensively based on the code made available at Teichmann (2019), especially the deep calibration algorithm, the availability of which we are grateful.

8.1 Calibration Method

In this subsection we will discuss the implementation and results of the deep calibration algorithm, As discussed in Subsection 3.4 and Section 6, the calibration of the Heston model is the process in which one estimates the optimal parameter combination in terms of a distance function from parameter implied quotes to market quotes. The calibration task, as described in Section 6, is performed by implementation of a 4-layered feed forward neural network with 30 neurons in the two hidden layers, introduced in Section 5 in which the output layer is the volatility surface $\tilde{\sigma}_{BS}^{\mathcal{H}(\theta)}(T_j, K_k)$ where $j = 1, \ldots, 10$ and $k = 1, \ldots, 9$. As such we specify a 9×10 implied volatility grid of approximately equidistant points in moneyness ranging from 0.91 to 1.1. The corresponding maturities T_j were specified for a grid ranging from 31 days to 406 days. The reason as to why these are not equidistant is the fact that we calibrate to market data, hence they are subject to real world maturities and moneyness. The input layer consists of the model parameter combinations $\theta = (\kappa, \theta^{\mathcal{H}}, \sigma, \rho)$, see Figure 6 for network architecture. We generate 100 000 synthetic parameter combinations from the uninformative prior distributions over the parameter space as described in Section 6, the data is then split into 80% training data, 10% validation data, 10% testing data. Validation data is used to make an unbiased evaluation of the model fit, and update the network hyperparameters accordingly. Testing data is used to monitor the loss function out of sample, i.e. as a second step. The rest of the observations is used for training. As we shall perform mini-batch training, the training samples are split into batches of size 1024, for efficient training, we set the number of epochs to 250, based on the discussion in Section 5 and set σ_{ELU} as the activation function.

We then employ the differential evolution algorithm as outlined in Section 6 (for details see Appendix A) in order to find the optimal parameter combinations that minimize the loss in Equation (49) from the neural network predicted implied volatility surface, $\tilde{F}$, in order to obtain parameter combinations $\hat{\theta} = (\hat{\kappa}, \hat{\theta}^{(\mathcal{H})}, \hat{\sigma}, \hat{\rho})$. The methodology can be roughly split into 4 consecutive sections, as follows:

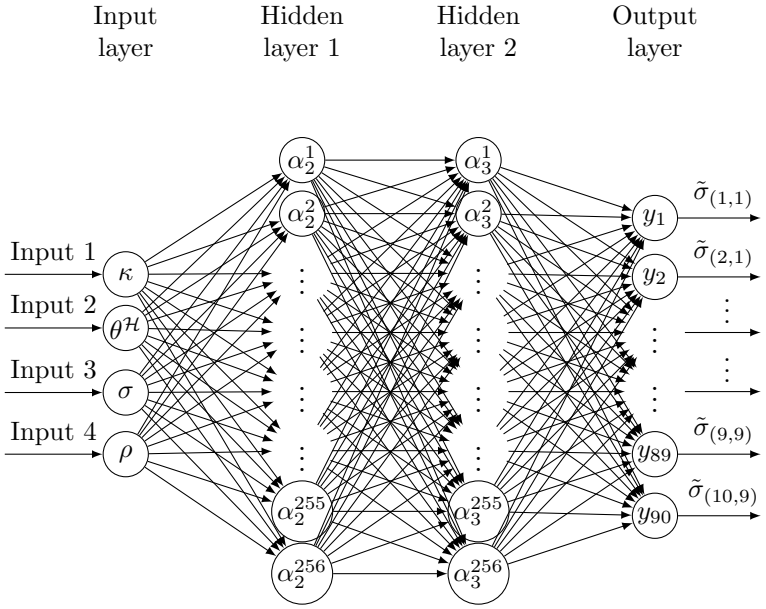


Figure 6: Neural Network Architecture for model calibration. The input layer is the Heston model parameters and output layer represents the implied volatility surface. We have two hidden layers with 256 neurons each. The objective for the neural network is to learn the map from parameters to implied volatilities as in Equation (47) where we weight the neural network output error as in Equation (48). The learning procedure is achieved by backpropagation and stochastic gradient descent, outlined in Section 5. The trained neural network is then used in the differential evolution algorithm, which is conceptually covered in Appendix A, to find the Heston model parameters that best generate the trained implied volatility map.

1. **Data handling**. Implied volatility data is extracted from option-chain by the trial and error and data is structured to fit the network.

2. **Heston model generation and simulation**. Synthetic pseudo-random parameter combinations $\theta_{train} \in \Theta$ are simulated from priors where analytical prices and volatility surfaces are generated.

3. **Feed-forward Neural Network creation and training**. Generation and training of Neural Network to learn the map from parameters to implied volatilities.

4. **Model calibration**. Calibration step by differential evolution in order to approximate optimal parameter combinations that generates the neural network predicted implied volatility surface.

As the partial stated objective of the thesis is to calibrate the Heston model to market data, we consider the calibration of S&P 500 call options on 2019-11-08. One important factor to consider when one calibrates to empirical data is the inability of the pricing model to fully capture the real terminal distribution of S and V and by extension, not be able to fully capture

the states of nature that generates the implied volatility surface that one seeks to calibrate. In order to illustrate the approximating properties of neural networks and the accuracy of the deep calibration algorithm, we first show results from calibration to synthetic data, where we know the Heston parameter vector θ beforehand and show the error of the calibration. Only after we have considered the validity of the method, shall we apply the deep calibration algorithm to empirical data. Hence, we now proceed to analyse the approximating capabilities of the neural network in terms of calibration of simulated parameter combinations, see Figure 7.

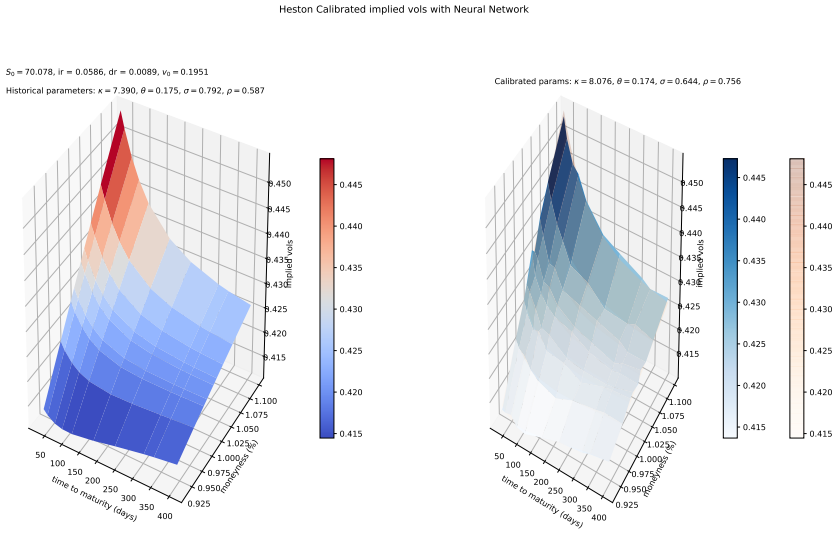


Figure 7: Synthetic and calibrated Heston surface. Right hand side plots both calibrated (blue scale) and synthetic volatility surface (red scale). Synthetic parameters; $\kappa = 7.390$, $\theta^{(\mathcal{H})} = 0.175$, $\sigma = 0.792$, $\rho = 0.587$. Calibrated parameters; $\hat{\kappa} = 8.076$, $\hat{\theta}^{(\mathcal{H})} = 0.174$, $\hat{\sigma} = 0.644$, $\hat{\rho} = 0.756$.

As one can see in Figure 7, the network seems to be able to replicate the simulated implied volatility surface, hence the method is accurate on random volatility surfaces generated from the same prior distributions as used in the training example. In order to more formally analyse errors over the entire grid we can calculate relative errors which allow us to consider whether the errors in the calibration is constant over maturities, see Figure 8 for visualisation. In Figure 8 we see e.g. that the maximal relative error is 0.2%, the minimal relative error is -0.3% and the mean relative error is approximately 0%.

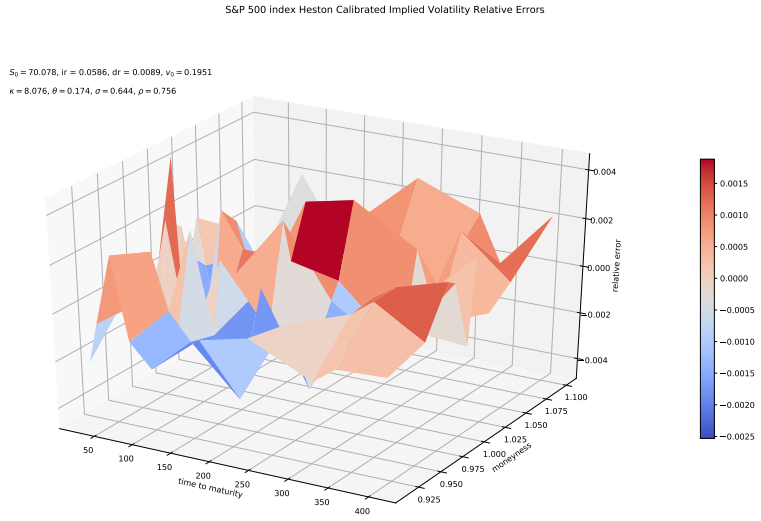


Figure 8: Relative error of calibrated implied volatility surface, i.e. a visualisation of $(\epsilon_{1,1}, \ldots, \epsilon_{n,m})$ where $\epsilon_{j,k}$ is defined as the relative percentage error of the calibrated implied volatility surface in Equation (49), relative to the data. In this case, the data is the random samples of our prior distributions with boundary conditions specified as in Table 2. Synthetic parameters; $\kappa = 7.390$, $\theta^{(\mathcal{H})} = 0.175$, $\sigma = 0.792$, $\rho = 0.587$. Calibrated parameters; $\hat{\kappa} = 8.076$, $\hat{\theta}^{(\mathcal{H})} = 0.174$, $\hat{\sigma} = 0.644$, $\hat{\rho} = 0.756$. Clearly, the relative errors are very small since the largest error is approximately 0.15 % which can also be deduced from the fit in Figure 7.

We have now visualized the approximating capabilities of the neural network, and validated its numerical proficiency in terms of the total calibration task, i.e. learning the map from parameters to volatility surface and the inverse map by calibration through differential evolution, see Section 6 for more details. As such, we are confident that the deep calibration algorithm should also be able to calibrate to empirical data. The empirical data we use is the implied volatility surface of the S&P 500 index on the 8th of November 2019 and has already been introduced to the reader in Figure 3.

As earlier discussed in Section 6, the variance process in the Heston model is constrained by the Feller condition, which becomes very apparent when performing simulations as the variance process can become negative. However, the option prices generated by the model are still legitimate when the Feller condition is violated, and as it turns out, empirical calibration often imply parameter calibrations that violates this condition, for further details see the discussion in Section 6. This has led to somewhat of a divide in weather or not to enforce the condition. Therefore we have decided to show the performance of the deep calibration algorithm for both cases, i.e. using Equation (49) when the Feller condition is ignored, visualized in Figure 9 and its relative errors in Figure 10 and use Equation (51) when the condition is enforced, where the relative errors are displayed in Figure 11.

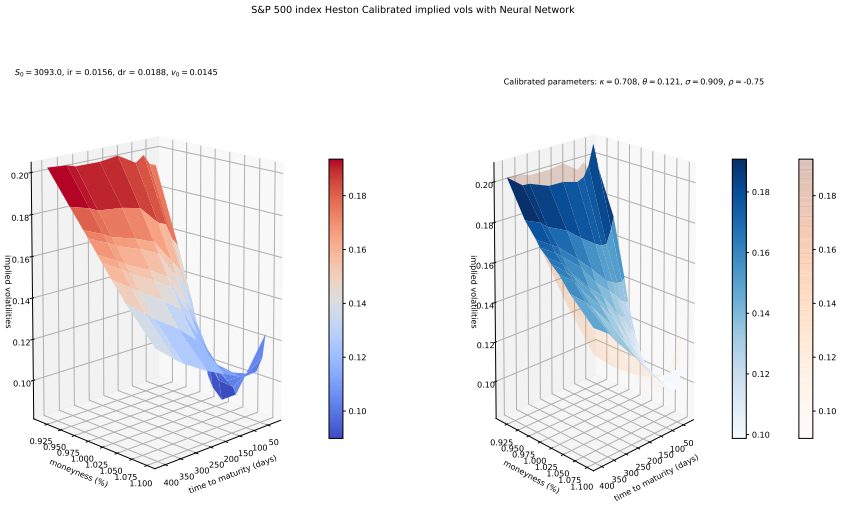


Figure 9: Left hand side visualises the implied volatility surface in Figure 3. The right hand side visualises the calibrated implied volatility surface (blue scale) versus the *actual* volatility surface (red scale). The calibrated parameters for the Heston model on SPX 2019-11-08 are; $\hat{\kappa} = 0.708,\ \hat{\theta}^{(\mathcal{H})} = 0.121,\ \hat{\sigma} = 0.909,\ \hat{\rho} = -0.75$.

As can be seen in Figure 9, the overall term and strike structure of the implied volatility (IV) surface is captured by the deep calibration algorithm. Furthermore, one can see that the tail options seem to be fairly well replicated by the calibrated parameter combinations.

Evidently, the Heston model pricing model inefficiency for negative jump risk, see e.g. Gatheral (2015) and Pan (2001), seems correct as the model seems to be less able to calibrate short term ITM options than short term OTM options, see Figure 10. The overall results are also largely in line with the findings of Horvath et al. (2019), in the sense that the deep calibration algorithm is very accurate, although we do not show computing time.

As one can see in when comparing Figure 10 and 11, the Feller constrained parameter calibration is less capable than the the unconstrained calibrated parameter combination to replicate the market IV surface. In Figure 10 we see e.g. that the maximal relative error is 23% and the minimal relative error is -0.13% . Which can be contrasted to the maximal relative error of 33.3% and minimal relative error of -32% for the Feller constrained calibration in Figure 11. Furthermore, the mean relative error across the whole grid is 1.8% in Figure 10, which can be contrasted to 2.38% in Figure 11. Both calibrations could however be altered slightly since one can specify the weights $\eta_{j,k}$ in Equation (48) to counteract this, if this error is indeed a consistent feature.

Recalling our discussion of the Feller condition and various simulation schemes, one has to make a choice, utilize more complicated, and potentially slower discretization schemes that better deal deal with the Feller condition (but may still be inadequate for severe violations) *or* utilize the constrained calibrated parameter combination such that one can obtain unbiased trajectories.

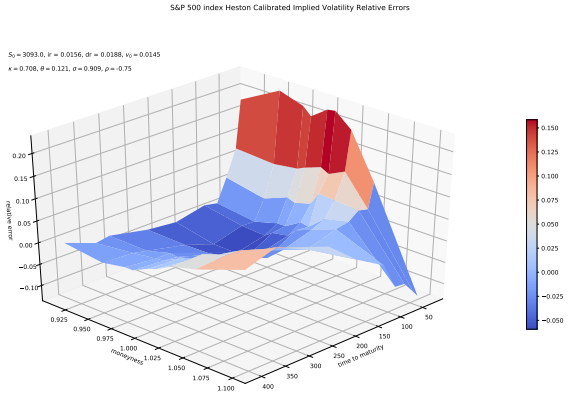


Figure 10: Relative error of the *unconstrained* calibrated implied volatility surface, i.e. a visualisation of $(\epsilon_{1,1}, \ldots, \epsilon_{n,m})$ where $\epsilon_{j,k}$ is defined as the relative percentage error of the implied volatility surface generated by calibrated parameters in Equation (49), relative to market data. Final value of the differential evolution algorithm (for details, see Appendix A); 0.008. Resulting Heston model parameter values; $\hat{\kappa} = 0.708$, $\hat{\theta}^{(\mathcal{H})} = 0.121$, $\hat{\sigma} = 0.909$, $\hat{\rho} = -0.75$.

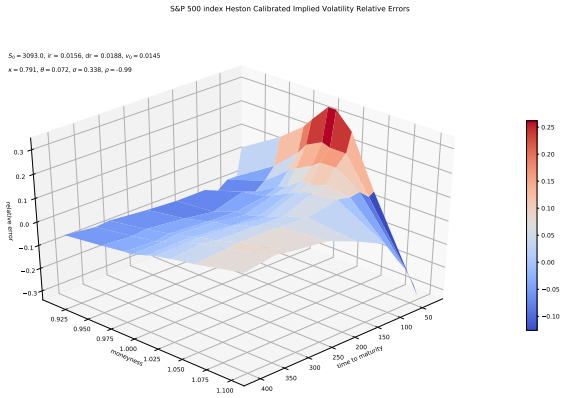


Figure 11: Relative error of the *Feller-constrained* calibrated implied volatility surface, i.e. a visualisation of $(\epsilon_{1,1}, \ldots, \epsilon_{n,m})$ where $\epsilon_{j,k}$ is defined as the relative percentage error of the implied volatility surface generated by calibrated parameters in Equation (51), relative to market data. Final value of the differential evolution algorithm (for details, see Appendix A); 0.0145. Resulting Heston model parameter values; $\hat{\kappa} = 0.791$, $\hat{\theta}^{(\mathcal{H})} = 0.072$, $\hat{\sigma} = 0.338$, $\hat{\rho} = -0.99$. Note that scales differ relative to Figure 10.

8.2 Hedging Method

This section will present our numerical results from the deep hedging algorithm, as described in Section 7. First, we remind the reader of the central aspects of the deep hedging algorithm. Then we discuss the neural network architecture and link it to the theoretical background in Section 7 as a minimizer of replication error under a convex risk measure. We also discuss some implementation related considerations such as number of neurons, activation functions etc. Then we proceed by presenting numerical results, consisting of simulated hedging profit/loss distributions and associated descriptive statistics compared to the benchmark strategy introduced in Section 4. We also compare various deep hedging strategies to the benchmark strategy on a pathwise basis, and lastly analyse the effects of including transaction costs in the deep hedging algorithm.

The deep hedging framework, detailed in Section 7, is the process by which optimal hedging strategies are approximated by minimizing the replication error of a hedging portfolio, under some monetary risk measure, as in Equation (34), by using the approximating capabilities of artificial neural networks. The theory in Buehler et al. (2019) is developed for convex risk measures, which were introduced in Definition 4.1, but can be extended to cover other risk measures, such as coherent risk measures, see e.g. Föllmer & Schied (2008) for further details.
As seen in Section 7, the task of finding optimal hedging strategies, as in Equation (34), can be reformulated as finding optimal parameters for a sequence of neural network approximations of the hedging strategy at each time-step, as $F_{t_i} \in \mathcal{NN}^{\sigma}_{M,d_0,d_1}$ in Equation (56). This sequential chain of neural networks is what Buehler et al. (2019) calls a *semi-recurrent* neural network, where the output at each time-step is the current hedging strategy, which is the input, along with a simulated value of S into the next feed forward network in the sequence. More concretely, one can visualize the semi-recurrent neural network as a series of feed forward networks indexed by time, t_i to t_n. At each t_i, the input layer consists of the *simulated* underlying asset price S_{t_i} for each $t_i \in (t_1, \ldots, t_n)$, generated by the Heston model calibration described in Section 6 and implemented as in Subsection 8.1, and the output at t_{i-1} of the approximately optimal hedging weights under some monetary risk measure, as discussed in Section 4. We have decided to specify each feed forward network with two hidden layers, 30 neurons and σ_{Elu} as activation function, as seen in See Figure 12.

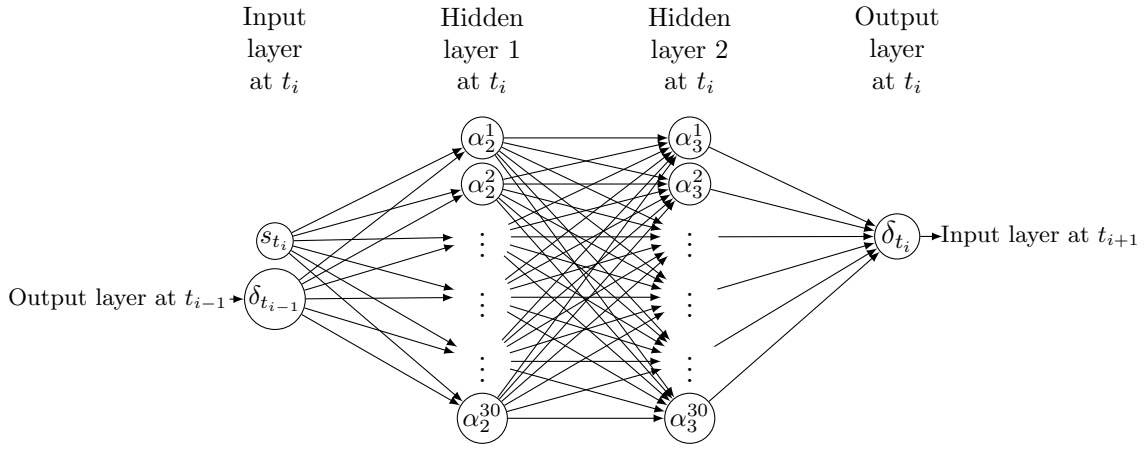
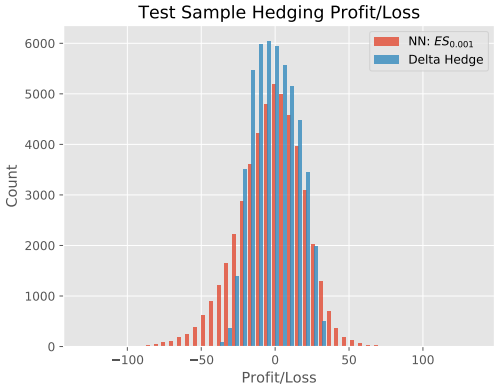


Figure 12: Deep hedging semi-recurrent neural network, formally described in Equation (56) and discussed in Section 7. The semi-recurrent neural network is a sequence $(F^{\tilde{w}_{t_i}})_{i=1,\dots,n}$ of feed forward networks indexed by time and parameterized by a vector $\tilde{w} \in \widetilde{\mathcal{W}}_M$ containing weights and biases. The input for $F^{\tilde{w}_{t_i}}$ is a simulation of the underlying asset S_{t_i}, from Equation (20), which in our case evolves according to Heston model dynamics where the variance process is discretized according to the moment matching scheme in Equation (22).

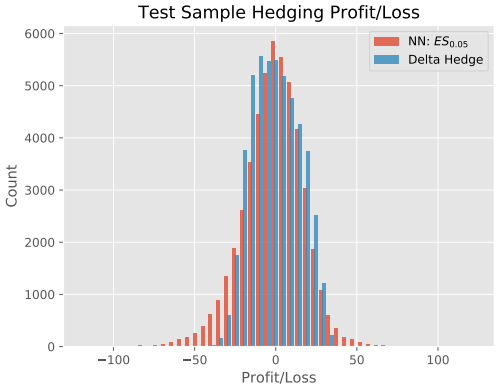
As discussed earlier in Section 4, we shall consider the approximation of optimal hedging strategies under both convex and non-convex monetary risk measures. The various risk measures will serve as the neural network loss function, described in Section 5. We consider optimality under *expected shortfall (ES)* and the *entropic risk measure*, with varying risk aversion parameters α and λ respectively, see Equations (38) and (35). We shall also consider optimality under a non-convex measure, namely the *quadratic criterion*, see Equation (40). The neural network minimizes the replication error by stochastic gradient descent, using mini-batch training, described formally for the entropic risk measure in Equation (62) and (61) respectively, however, the methodology easily extends to all measures under consideration.

Before presenting the numerical results of the hedging algorithm, one need to consider the fact that the method suffers from the so called *black-box* problem. Namely, that we cannot know the reason as to why the model gives a certain output as the dimensions of deep neural networks are too large. Hence, this will impair our ability to discuss the results in detail.

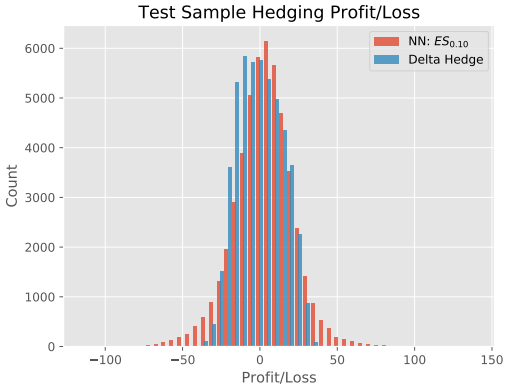
We now present the results of the deep hedging algorithm. All of the following hedging strategies will be evaluated at the spot value $S_0 = 3093.08$, and option parameters $T = 31/365$, $K = 3100$, for the calibrated SPX Heston model. Firstly, we simulate trajectories of S_t, by the moment-matching scheme, which are then used to train a neural network strategy $\delta^{\tilde{w}} \in \Delta^u$ as a minimizer to (34), where Δ is replaced by the set of unconstrained strategies Δ^u. This numerical approximation of a optimal hedging strategy $\delta^{\tilde{w}}$ under some convex risk measure is then used to produce profit/loss-distributions of the hedging strategy as in Equation (52). The hedging profit/loss (pnl) distribution, see Equation (52), for deep hedging strategies are *benchmarked* to the pnl-distribution of a simple delta hedging strategy, see Equation (31) (and preceding discussion in Section 3.5) calculated numerically as in Equation (32), evaluated over the same trajectories as the deep hedging strategies.



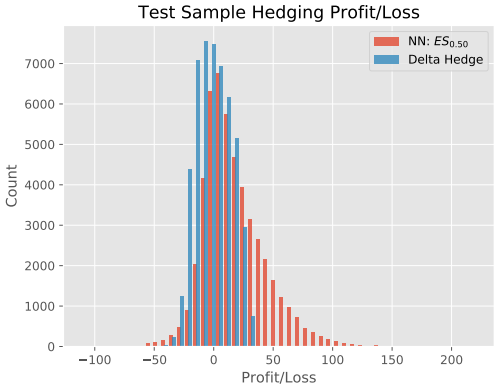
(a) Expected shortfall PnL when $\alpha = 0.01$.



(b) Expected shortfall PnL when $\alpha = 0.05$.



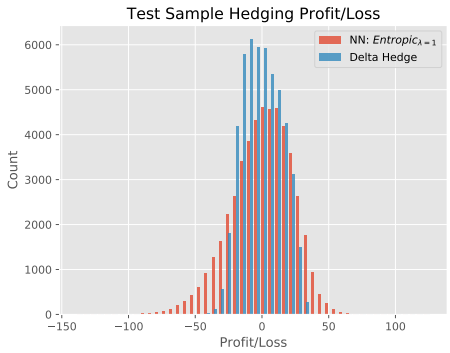
(c) Expected shortfall PnL when $\alpha = 0.10$.



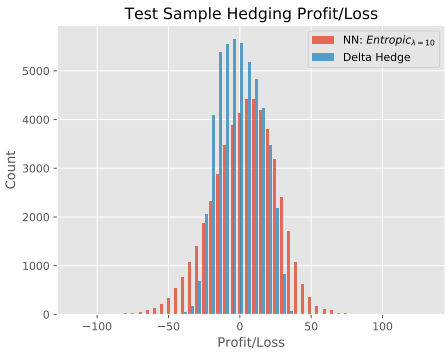
(d) Expected shortfall PnL when $\alpha = 0.50$.

Figure 13: Simulated terminal pnl-histograms for delta hedge (blue) and expected shortfall (see e.g. Equation (38)) deep hedging strategies with varying α-quantile (red) evaluated over 50 000 trajectories and $t_n = 30$ days. Pnl's are calculated as in Equation (52) where δ is replaced by $\delta^{\tilde{w}}$ for deep hedging strategies. The benchmark delta hedge δ (see Section 3.5 for details) is approximated numerically as in Equation (32) by finite difference. $-Z$ in Equation (52) is the terminal payoff when selling a call option, see Equation (16).

In Figure 13, one can see that a increase in risk aversion, larger α, shifts the distribution to the right, as the PnL distributions in Figure 13 imply that the agent can charge the indifference price, i.e. $p_0^{\tilde{w}} = \pi^M(-Z) - \pi^M(0)$ and not fair market prices of $-Z$, which is also noted by Buehler et al. (2019). We can therefore, without fear of contradiction, say that increase in risk aversion generally causes prices to converge towards the risk neutral price, as it is consistent with the results in Buehler et al. (2019). Furthermore, we can also see that, in general, decreases in risk aversion generally causes the deep hedging strategy to look more like the benchmark delta hedging strategy.



(a) Entropic measure PnL when $\lambda = 1$.



(b) Entropic measure PnL when $\lambda = 10$.

Figure 14: Simulated terminal pnl-histograms for delta hedge (blue) and entropic (see Equation (55)) deep hedging strategies with varying risk aversion parameter λ (red) evaluated over 50 000 trajectories and $t_n = 30$ days. Pnl's are calculated as in Equation (52) where δ is replaced by $\delta^{\tilde{w}}$ for deep hedging strategies. For the benchmark delta hedge δ (see Section 3.5 for details) is approximated numerically as in Equation (32) by finite difference. $-Z$ in Equation (52) is the terminal payoff when selling a call option, see Equation (16).

An increase in the risk aversion parameter λ in Equation (57) corresponds to a increase in risk aversion, which is evidently rather subtle in Figure 14, since one could only see a slight decrease in variance in Figure 14b relative to Figure 14a.

We next consider approximations of optimal hedging strategies under the quadratic criterion, that is, when the optimization is given as in Equation (40). As earlier noted, the quadratic criterion is not a convex risk measure, however, it is a common practice to consider hedging in incomplete markets under the quadratic criterion as it returns variance optimal hedging strategies, see e.g. Föllmer & Schweizer (1990). It is also a well established practice in finance to consider expectation of portfolio returns, weighted by a loss function of the form $\ell(x) = x^p$ for $p \geq 1$, see e.g. Föllmer & Leukert (2000). This is also used by Buehler et al. (2019), see p. 1288. In Figure 15 we illustrate the simulated pnl-distribution for the quadratic criterion, in which we can see that it is very similar to the benchmark delta hedge.

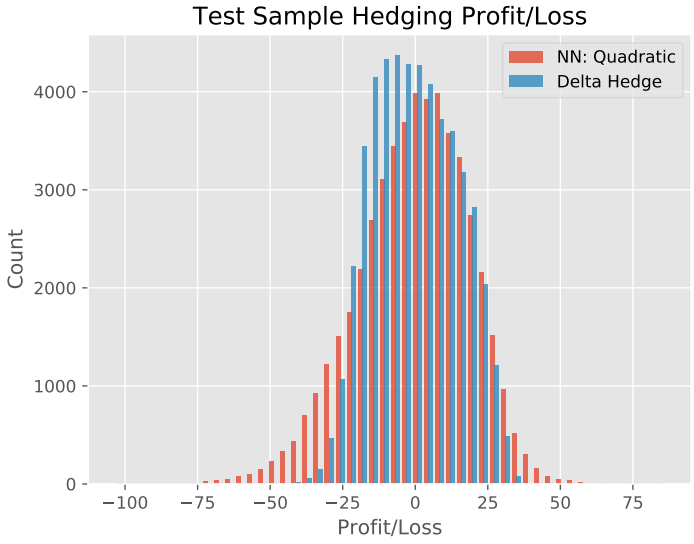


Figure 15: Simulated terminal pnl-histograms for delta hedge (blue) and quadratic (see Equation (40)) deep hedging strategies (red) evaluated over 50 000 trajectories and $t_n = 30$ days. Pnl's are calculated as in Equation (52) where δ is replaced by $\delta^{\tilde{w}}$ for deep hedging strategies. For the benchmark delta hedge δ (see Section 3.5 for details) is approximated numerically as in Equation (32) by finite difference. $-Z$ in Equation (52) is the terminal payoff when selling a call option, see Equation (16).

From Figure 15 we can see that the variance of the close-to-variance-optimal hedging strategy is larger than the variance of the delta hedging strategy. Figures 13 - 15 visualizes the PnL distributions of all considered risk measures and Table 3 gives the corresponding descriptive statistics for all hedging strategies.

Model type	$p_0^{\tilde{w}}$	Min, Max	Mean	Std.dev	Skewness	Kurtosis
$ES_{0.01}$	39.55	$(-130.374,\ 135.67)$	-2.379	22.02	-0.365	0.85
$ES_{0.05}$	41.06	$(-119.22,\ 123.08)$	-1.22	18.75	-0.25	1.64
$ES_{0.10}$	42.7	$(-114.72,\ 139.45)$	2.79	19.47	-0.029	2.152
$ES_{0.50}$	58.79	$(-110.51,\ 220.07)$	17.616	26.74	0.957	2.186
Quadratic	40.89	$(-103.626,\ 86.051)$	-0.336	19.10	-0.38	0.292
$\text{Entropic}_{\lambda=1}$	41.02	$(-139.022,\ 126.094)$	0.3041	22.83	-0.45	0.468
$\text{Entropic}_{\lambda=10}$	43.04	$(-116.051,\ 131.920)$	3.763	22.61	-0.187	0.5209
Delta Hedge	40.82	$(-50.158,\ 37.386)$	-0.215	14.51	0.088	-0.804

Table 3: Descriptive statistics for pnl of hedging strategies, given by Equation (52) and evaluated over all risk measures introduced in Section 4 and illustrated in Figures 13 to 15. Calculations are based on test sample with 50 000 trajectories of S when $t_n = 30$ days. Skewness and kurtosis are the third and fourth standardized moments of the pnl-distribution respectively. Notice the mean hedging PnL is dependent on risk aversion, α, and the mean hedging PnL is roughly equivalent to the deep hedging price *premium* relative to the market price, which we approximate as the fair price under $\mathbb{Q}$, see Equation (16). The resulting fair market price is 40.82.

As one can see from Table 3, most statistics are quite similar, which is also supported by Figures 13 - 15 since their simulated distributions are also quite similar. However, the implied hedging price, i.e. p_0 in Equation (52), of expected shortfall with $\alpha = 0.5$ diverges substantially from the risk neutral price, indicating that this is more risk averse parameterization than the rest. Other differences are for example the differences in min-max values for different risk aversions, the same can be said for their standard deviations. The neural network strategy with the lowest standard deviation is expected shortfall with $\alpha = 0.01$ and the strategy with the lowest maximum loss (minimum value) is the expected shortfall with $\alpha = 0.50$. The risk measure that seemingly best approximate the fair market price is the quadratic criterion, see Table 3, however, these are surely negligible differences. One of the main conclusions that we draw from Table 3 is that using the expected shortfall strategy in the deep hedging algorithm makes the algorithm very flexible. This is because training can be made to target different objectives such as minimizing worst case scenarios by choosing a large α or be very "risk neutral" by specifying a smaller α. One other main conclusion that may be drawn is that no strategy seem able to outperform the benchmark delta strategy.

In Figure 16 we see the realization of the hedging strategy for individual sample paths, for the collection of expected shortfall measures.

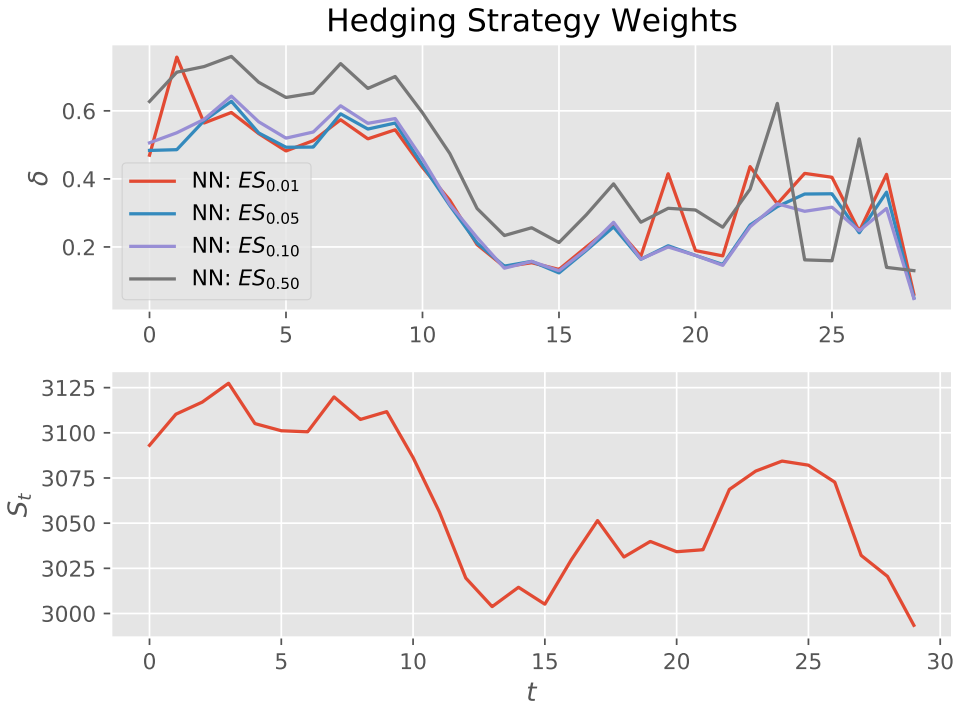


Figure 16: Comparisons of expected shortfall strategies with different α-quantiles, see e.g. Equation (4) over one trajectory of S (red line in sub figure), evolving according to Equation (20) with discretized variance process V as in Equation (22), where $t_n = 30$ days. Each trajectory of the deep hedging algorithm is computed as $(\delta^{\tilde{w}_{t_i}})_{i=1,\dots,n}$ in Equation (56). Note that although deep hedging strategies have similar weights over time, initial hedging weights differ, since $p_0^{\tilde{w}}$ in Equation (52) differs, due to differences in risk measure, see e.g Table 3.

Figure 16 display deep hedging strategies $\delta^{\tilde{w}}$ when using expected shortfall as risk measure, see e.g. Equation (4) over a given trajectory of S, evolving according to Heston model dynamics with parameters from the constrained deep calibration in Equation (51), presented in Section 8.1. As one can see, the expected shortfall strategies are rather similar, especially $ES_{0.05}$ and $ES_{0.10}$. Furthermore, notice that expected shortfall with higher α has higher initial hedging weight. This is due to the fact that $p_0^{\tilde{w}}$ in Equation (52), differ between risk measures as seen in Table 3. Furthermore one can deduce from Figure 16 that deep hedging strategies evolve similarly over time, even though $ES_{0.50}$ seems less numerically stable, which is likely due to insufficient observations in training dataset, as mentioned earlier in the section.

In Figure 17, we show the performance of the entropic neural network approximation, where one can see that the strategies seem more volatile than expected shortfall strategies. We cannot deduce any significant differences that would allow us to systematically discriminate between risk measures in on a per sample path basis, as the risk aversion parameters, λ in Equation (36) and α in Equation (38) are not comparable.

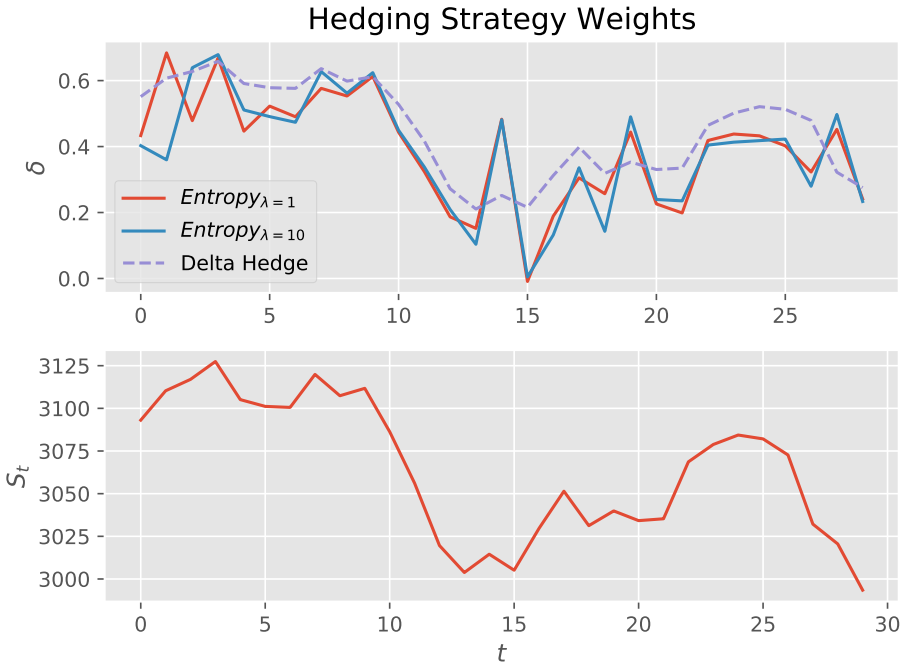


Figure 17: Comparisons of entropic strategies with different λ-values to benchmark, see e.g. Equation (55), over one trajectory of S (red line in sub figure), evolving according to Equation (20) with discretized variance process V as in Equation (22), where $t_n = 30$ days. Each trajectory of the deep hedging algorithm is computed as $(\delta^{\tilde{w}_{t_i}})_{i=1,\dots,n}$ in Equation (56) and the benchmark delta hedging strategy $(\delta_{t_i})_{i=1,\dots,n}$ is approximated according to Equation (32) for each t_i as a discrete version of Equation (31), the theoretical implications of which has been discussed in Section 3.5. Note that although deep hedging strategies have similar weights over time, initial hedging weights differ, since $p_0^{\tilde{w}}$ in Equation (52) differs, due to differences in risk measure, see e.g Table 3.

We now proceed by presenting the results of the approximate variance optimal hedge, see Figure 18. Here one can see that, in general, all hedging strategies are rather similar in terms of their respective sample paths. However, we detect a pattern across all neural network approximations that the strategies are more volatile than the delta hedge, we believe that this is likely due to numerical instability from insufficient sample size for the training of the semi-recurrent neural network in Equation (56), discussed earlier in the section.

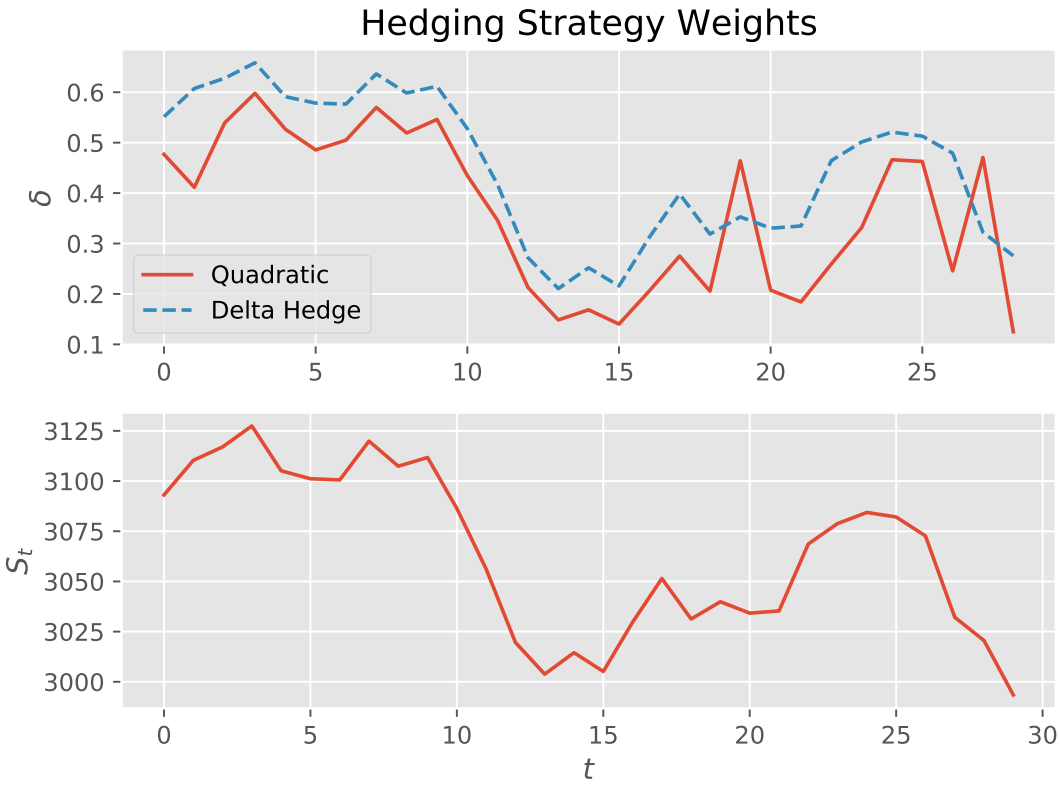


Figure 18: Comparison of quadratic hedging to benchmark delta strategy, see Equation (40), over one trajectory of S (red line in sub figure), evolving according to Equation (20) with discretized variance process V as in Equation (22), where $t_n = 30$ days. Each trajectory of the deep hedging algorithm is computed as $(\delta^{\tilde{w}_{t_i}})_{i=1,\dots,n}$ in Equation (56) and the benchmark delta hedging strategy $(\delta_{t_i})_{i=1,\dots,n}$ is approximated according to Equation (32) for each t_i as a discrete version of Equation (31), the theoretical implications of which has been discussed in Section 3.5. Note that although deep hedging strategies have similar weights over time, initial hedging weights differ, since $p_0^{\tilde{w}}$ in Equation (52) differs, due to differences in risk measure, see e.g Table 3.

In light of these various visualizations, Figures 13 - 18, and statistics in Table 3, we can begin to form some conclusions about the various deep hedging strategy specifications. The most flexible risk measure, and arguably most intuitive, that we consider is seemingly expected shortfall, as varying the α-quantile in Equation (4) allows an economic agent to alter risk aversion based on individual risk preference. However our results suggest that larger α-quantile may more

easily exhibit numerical instability. In our view, the intuitive nature of the expected shortfall risk aversion parameter makes it more easily implemented than the entropic risk measure. The quadratic criterion performs best in terms of learning the deep hedging price, i.e. $p_0^{\tilde{w}}$ is closest to the risk neutral price in Equation (16) for the quadratic criterion, even though by a slim margin. However, in terms of profit/loss performance, the simple delta hedging strategy, presented in Section 3.5, has seemingly superior performance to all deep hedging strategy specifications, see e.g. Table 3. We believe that this is likely due to us simulating from a Heston model, thereby out of sample performance on real market data would likely be different.

In Figure 19 we illustrate the impact of *proportional transaction costs*, as presented in Equation (54), on the pnl distribution for $ES_{\alpha=0.05}$, i.e. a non-zero $C_T(\delta^{\tilde{w}})$ in Equation (52) where we set $\varepsilon = 0.01$.

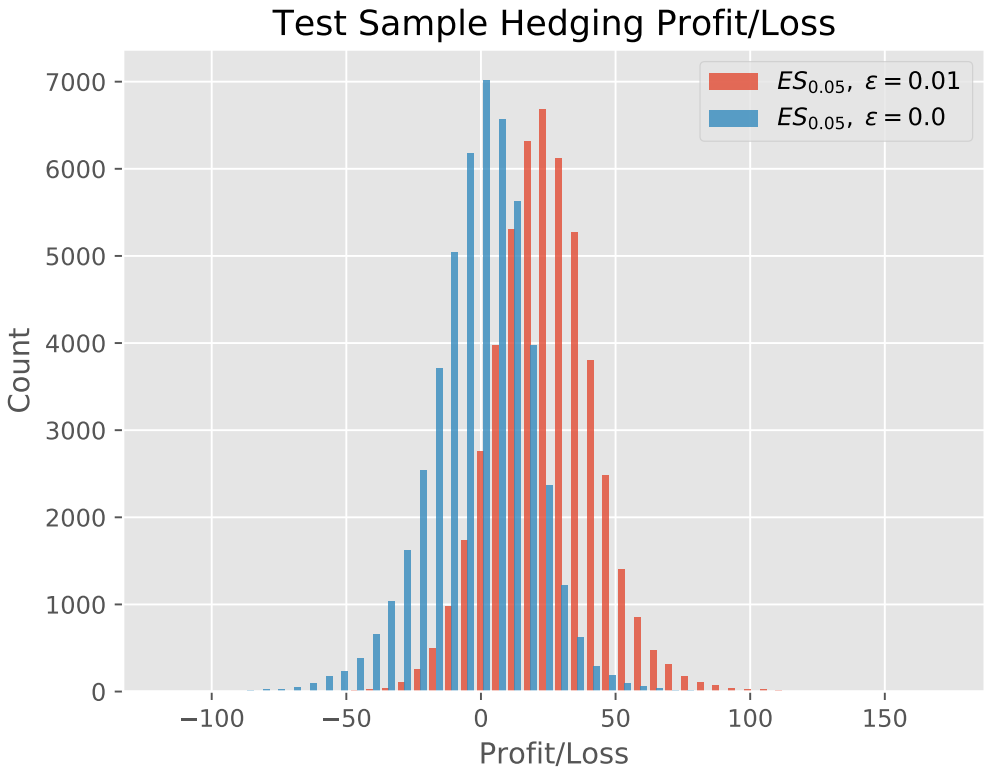


Figure 19: Simulated terminal pnl-histograms for deep hedging-expected shortfall strategy (see e.g. Equation (38)), without transaction costs,(blue) and expected shortfall deep hedging strategies with varying with proportional transaction costs (red) evaluated over 50 000 trajectories and $t_n = 30$ days. Pnl's are calculated as in Equation (52) where δ is replaced by $\delta^{\tilde{w}}$ for deep hedging strategies, and when transaction costs are included, $C_T(\delta^{\tilde{w}}) > 0$, given by proportional transaction costs in Equation (54) where $\varepsilon = 0.01$. $-Z$ in Equation (52) is the terminal payoff when selling a call option, see Equation (16).

The deep hedging price of the $ES_{0.05,\varepsilon=0.01}$ strategy, $p_0^{\tilde{w},\varepsilon} = 52.22$ in Equation (52), which can be contrasted to the risk neutral price of 40.82 and the $ES_{0.05,,\varepsilon=0.00}$ price of 41.06. Hence, as one might expect, transaction costs will shift the smallest acceptable price higher since there is a implicit cost associated to trading. Next, Figure 20 shows the impact of transaction costs on a per trajectory basis.

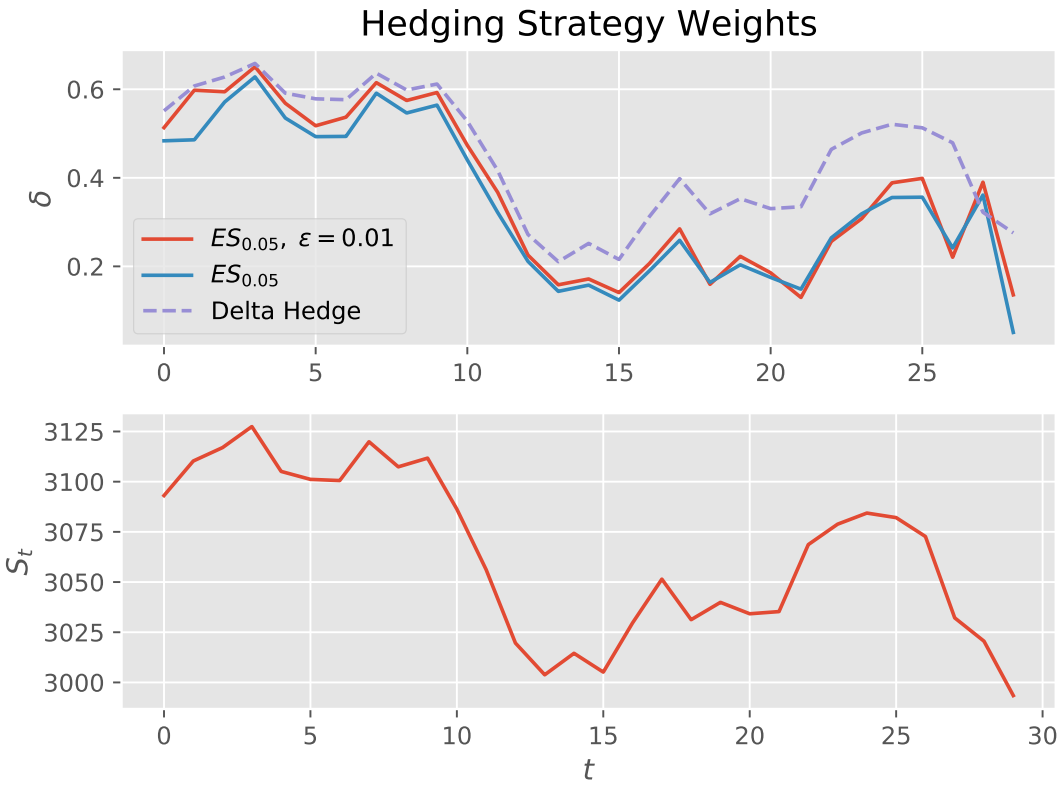


Figure 20: Comparison of expected shortfall strategy, see e.g. Equation (4), with and without transaction costs, to benchmark delta strategy over one trajectory of S (red line in sub figure). S is evolving according to Equation (20) with discretized variance process V as in Equation (22), and $t_n = 30$ days. Each trajectory of the deep hedging algorithm is computed as $(\delta^{\tilde{w}_{t_i}})_{i=1,\dots,n}$ in Equation (56). When transaction costs are included (red line in super figure), $\varepsilon = 0.01$ in Equation (54) representing transaction cost of 1% for each trade (daily rebalancing). The benchmark delta hedging strategy $(\delta_{t_i})_{i=1,\dots,n}$ is approximated according to Equation (32) for each t_i as a discrete version of Equation (31), the theoretical implications of which has been discussed in Section 3.5. The delta hedging strategy ignores transaction costs. Note that although deep hedging strategies have similar weights over time, initial hedging weights differ, since $p_0^{\tilde{w}}$ in Equation (52) differs, due to $C_T(\delta^{\tilde{w}}) > 0$ in Equation (34).

9 Conclusions & Suggestions for Future Research

Complex stochastic models introduce the need for model independent numerical methods for calibration and hedging. In this thesis we have investigated the application of artificial neural networks for calibration and hedging of stochastic volatility. The purpose of this thesis, was to combine the method of Horvath et al. (2019) with the deep hedging algorithm, introduced by Buehler et al. (2019) such that one can hedge and calibrate a stochastic model using both methods, which has been successfully implemented. Interestingly, both of these methods are generalizable to all types of stochastic models in need for hedging and calibration. Thus, the need for model specific methodology can be, largely, averted since one can approximate, both the pricing map of the model and the hedging formula with artificial neural networks. However, our current formulation of the hedging method relies on the ability to simulate from the calibrated stochastic volatility model, which introduce some necessary considerations. First of all, one must consider some form of accurate discretization in order to sample correctly from the spot process with sufficient speed for training to be practical. Secondly, the formulation of the hedging method is dependent on which *sources of risk* is described by the stochastic model and will thus affect which financial instruments are utilized for hedging. For example, if one were to seek to completely hedge all risk in a stochastic volatility model, then one would have to introduce a hedge of $W^{(V)}$ as well as a hedge of $W^{(S)}$, see e.g. Equation (13) and (14). As such, our formulation of the hedging method still has some important model dependencies. One potentially interesting subject for research could be utilizing the approximating capabilities of neural networks to train a simulation scheme which samples from a discretized version of S. Here one could potentially train a neural network to sample at all given time-points in a interval simultaneously by for example using model parameters as input data and the exact simulation algorithm of Broadie & Kaya (2006) in the output layer.

The applicability of deep calibration and hedging is very general, as pointed out by Horvath et al. (2019), since the training step of both can be conducted offline and thus the prediction of stochastic model parameters and hedging strategy is fast. Furthermore, we show that both methods approximate the pricing function and optimal hedging strategies respectively. One possible area of future research that could be interesting is the eventual regulatory implication of applying numerically optimal hedging approximations in terms of capital requirements. Even though this research would not be strictly statistical, it would almost surely effect the spread and speed of the adaptation for the hedging methodology, since it would provide a clear path to implementation from a operations perspective.

However, there exist a problem with the current formulation of the hedging methodology that might impair the practical applicability of the joint algorithm. Namely the fact that the hedging model is not well equipped to handle fundamental regime switches for the underlying market, as it has to be retrained in order to handle these. This could be solved if a regime switching model was calibrated in the first place, however this implies that switching probabilities and circumstances are constant. Hence, if one would consider fitting the neural network hedging strategy, *only* using real empirical data, then the network would have to be retrained, which is a slow procedure and thus not realistic. Furthermore, only considering empirical data for training would produce severe distortions of the neural network hedge since the output is very sensitive to the quality of the data. Therefore, the method is not very suitable to sparse data sets, especially if the feature set is very large. These problems indicates that there is quite a lot of problems that need to be addressed in order for the method to be applied to OTC derivatives, where relevant data is naturally much more sparse.

One interesting possible solution to OTC derivatives is the recent research into *market generators*, see e.g. Wiese et al. (2019), Wiese et al. (2020) or Kidger et al. (2020), which introduces a non-parametric way of learning financial time-series in a completely data driven way. These methods may prove to be able to circumvent the need for parametric stochastic models, such as the Heston model, in the deep hedging algorithm to simulate financial time-series. This is, in our view, of immense importance for the general quantitative finance community. Since it may be able to let the finance industry move away from the classical Markovian models. However, to our knowledge, there are still many important topics of research such as ensuring no-arbitrage condition holds etc.

Furthermore, although the aggregate difference of hedging PnLs between risk measures are quite large, individual differences on a per trajectory basis is quite small. However, for our example, none of the hedging strategies seem to outperform the simple delta hedging strategy which we believe is due to the fact that we are simulating from a Heston model, nonetheless, they are more practical since they allow for the inclusion of market frictions such as transaction costs.

A Differential Evolution

Differential evolution (DE) is, as the name suggests, a *evolutionary algorithm*, inspired by natural evolution of species, that can solve numerical optimization problems in a wide variety of fields. In the case of our problem, the DE algorithm aims at evolving a population of NP 4-dimensional parameter vectors, usually called individuals, which denotes the set of candidate solutions. Let $\theta_{i,G} = \left\{\kappa_i, \theta_i^{\mathcal{H}}, \sigma_i, \rho_i\right\},\ i = 1, \dots NP$ where G denotes the number of generations .The initial guess of the population covers the entire space by uniformly randomly selecting individuals within the search space, constrained by the parameter bounds, see Table 2. Then, the algorithm employs a *mutation operation*, generated by some *mutation strategy*, and produce a mutant vector, $\mathbf{V}_{i,G} = \left\{\nu_{i,G}^{\kappa}, \nu_{i,G}^{\theta^{\mathcal{H}}}, \nu_{i,G}^{\sigma}, \nu_{i,G}^{\rho}\right\}$ for each individual $\theta_{i,G}$, which is called the *target vector*. After the mutation, a *crossover operation* is performed, where a *trial vector* $\mathbf{U}_{i,G}$ is produced, where each element is selected from either the mutation or the target vector based on a pre specified *crossover probability*. The elements in the trial vector are then passed through a loss function, which in our case is the squared distance function in Equation (49), where the loss is evaluated and mutations accepted if the loss becomes smaller. This procedure is repeated until some stopping criterion on the loss function is satisfied. If the calibration problem in Equation (49) is convergent, then the optimal parameter vector, $\hat{\theta}$ is found. For a more complete description of the algorithm, see e.g. Qin et al. (2009). Some advantages of the DE-algorithm is it does not require any significant assumptions of the optimization problem and it can search large spaces for solutions. However, the algorithm does not guarantee global optima.

References

Acerbi, C. & Tasche, D. (2002), 'On the coherence of expected shortfall', *Journal of Banking Finance* **26**, 1487–1503.

Andersen, L. B. G. (2007), 'Efficient simulation of the heston stochastic volatility model'.

Andersen, L. B. G. & Brotherton-Ratcliffe, R. (2005), 'Extended libor market models with stochastic volatility', *Journal of Computational Finance* **9**.

Björk, T. (2009), *Arbitrage Theory in Continuous Time*, third edn, Oxford University Press.

Black, F. & Scholes, M. (1973), 'The pricing of options and corporate liabilities', *Journal of political economy* **81**, 637–654.

Bottou, L. (2012), Stochastic gradient tricks, *in* G. Montavon, G. B. Orr & K.-R. Müller, eds, 'Neural Networks, Tricks of the Trade, Reloaded', Lecture Notes in Computer Science (LNCS 7700), Springer, pp. 430–445.

Broadie, M. & Kaya, Ö. (2006), 'Exact simulation of stochastic volatility and other affine jump diffusion processes', *Operations research* **54**, 217–231.

Buehler, H., Ganon, L., Teichmann, J. & Wood, B. (2019), 'Deep hedging', *Quantitative Finance* **19**, 1271–1291.

Bégin, J.-F., Bédard, M. & Gaillardetz, P. (2015), 'Simulating from the heston model: A gamma approximation scheme', *Monte Carlo Methods and Applications* **21**, 205–231.

Clark, I. J. (2011), *Foreign Exchange Option Pricing: A Practitioner's Guide*, Wiley Finance, John Wiley & Sons, Inc.

Cont, R. & Fonseca, J. D. (2001), 'Dynamics of implied volatility surfaces', *Quantitative finance* **2**, 45–60.

Cox, J. C., Ingersoll, J. E. & Ross, S. A. (1985), 'A theory of the term structure of interest rates', *Econometrica: Journal of the Econometric Society* pp. 385–407.

Crisóstomo, R. (2014), 'An analysis of the heston stochastic volatility model: Implementation and calibration using matlab', *Documentos de trabajo (CNMV)* pp. 1–46.

Cui, Y., del Baño Rollin, S. & Germano, G. (2017), 'Full and fast calibration of the heston stochastic volatility model', *European Journal of Operational Research* **263**(2), 625–638.

Fama, E. F. (1970), 'Efficient capital markets: A review of theory and empirical work', *The Journal of Finance* **25**, 383–417.

Föllmer, H. & Leukert, P. (2000), 'Efficient hedging: Cost versus shortfall risk', *Finance and Stochastics* **4**, 117–146.

Föllmer, H. & Schied, A. (2008), 'Convex and coherent risk measures', *Encyclopedia of Quantitative Finance* pp. 355–363.

Föllmer, H. & Schweizer, M. (1990), 'Hedging of contingent claims', *Applied stochastic analysis* **5**, 389–415.

Gatheral, J. (2015), *The Heston Model*, John Wiley Sons, Ltd, chapter 2, pp. 15–24.

Gil-Pelaez, J. (1951), 'Note on the inversion theorem', *Biometrika* **38**(3-4), 481–482.

Hernández, A. (2016), 'Model calibration with neural networks'.

Heston, S. L. (1993), 'A closed-form solution for options with stochastic volatility with applications to bond and currency options', *The Review of Financial Studies* **6**, 327–343.

Hornik, K. (1991), 'Approximation capabilities of multilayered feedforward networks', *Neural Networks* **4**, 251–257.

Horvath, B., Muguruza, A. & Tomas, M. (2019), 'Deep learning volatility'.

Kidger, P., Morrill, J., Foster, J. & Lyons, T. (2020), 'Neural controlled differential equations for irregular time series', *arXiv preprint arXiv:2005.08926* .

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *Computing Research Repository* **abs/1412.6980**.

McCulloch, W. S. & Pitts, W. H. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysic* **5**, 115–133.

Mrázek, M. & Pospísil, J. (2017), 'Calibration and simulation of heston model', *Open Mathematics* **15**, 679–704.

Murphy, K. P. (2012), *Machine Learning: A Probabilistic Perspective*, MIT Press.

Nielsen, M. A. (2015), *Neural Networks and Deep Learning*, Determination Press.

Pan, J. (2001), 'The jump-risk premia implicit in options: Evidence from an integrated time-series study', *Journal of Financial Economics* **63**, 3–50.

Protter, P. E. (2005), *Stochastic differential equations*, Springer, Berlin, Heidelberg, pp. 249–361.

Qin, A. K., Huang, V. L. & Suganthan, P. N. (2009), 'Differential evolution algorithm with strategy adaptation for global numerical optimization', *IEEE Transactions on Evolutionary Computation* **13**, 398–417.

Ribeiro, A. & Poulsen, R. (2013), 'Approximation behoves calibration', *Quantitative Finance Letters* **1**(1), 36–40.

Rosenblatt, F. (1958), 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review* **65**(6), 386–408.

Rouah, F. D. (2013), *The Heston Model and Its Extensions in Matlab and C*, Wiley Finance, John Wiley & Sons, Inc.

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747* .

Schweizer, M. (1995), 'Variance-optimal hedging in discrete time', *Mathematics of Operations Research* **20**, 1–32.

Stefanica, D. & Radoicic, R. (2017), 'An explicit implied volatility formula', *International Journal of Theoretical Applied Finance* **20**.

Teichmann, J. (2019), 'Machine learning in finance'.
URL: *https://people.math.ethz.ch/ jteichma/index.php?content=teach_mlf2019*

Wiese, M., Bai, L., Wood, B. & Buehler, H. (2019), 'Deep hedging: learning to simulate equity option markets', *Available at SSRN 3470756* .

Wiese, M., Knobloch, R., Korn, R. & Kretschmer, P. (2020), 'Quant gans: deep generation of financial time series', *Quantitative Finance* pp. 1–22.