UNIVERSITY OF GOTHENBURG
SCHOOL OF BUSINESS, ECONOMICS AND LAW

Neural Networks as Approximators for Heston Model Calibration & Hedging

Oliver Klingberg Malmer & Victor Tisell

March 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 recent development in the application of deep learning methods, enabled by modern computational capacity, has resulted in increased adaptation of such methods by the financial industry. In this thesis we focus on the on one such method, namely *deep learning*, in the context of financial modelling. Deep learning enables numerical approximations to previously unattainable problems that has long eluded financial researchers, for example non-parametric minimization of risk (hedging) under realistic market conditions and calibration of complex financial models. By utilizing the universal approximating properties of neural networks presented by Hornik (1991), we study and implement methods for estimating and hedging a parametric stochastic model, called the Heston model and used to describe the volatility and price of an asset, introduced in Heston (1993). We build the calibration method by training a option pricing function, which is then used to find optimal parameters for the Heston model such that the distance between the stochastic model and the market is minimized, as described by Horvath et al. (2019). We then proceed by approximating optimal trading strategies that minimize the profit/loss distribution of a hedging portfolio, as introduced in Buehler et al. (2019). We find that the modelling attempt was successful, since we illustrate that the approximating properties hold for both the calibration and the hedging task. However, we show that the stochastic model have some inherent deficiencies with respect to constraints to the parameter space, which introduce some potential loss of accuracy or simulation speed, which will effect the *deep hedging* algorithm. Furthermore, we compare hedging strategy performance for different specifications of the deep hedging, in which we find substantial statistical differences in profit/loss-distributions and show the impact of transaction costs. We also highlight its potential superior applicability under realistic market conditions, relative to a theoretical hedging strategy.

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 as they will always produce some non-systematic overrides. The research of hedging under such conditions has long been an important subject for practitioners. However, the inclusion of such circumstances omits analytical solutions to optimal hedging weights. Thus, there exist a need for numerical methods to solve this problem. As such, the central property of such methods should be model independence, as this exclude the assumption of unrealistic statistical properties of financial markets. Thus, the method has to be data driven and independent of explicit human input to correct non-systematic overrides.

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 results show that one can arbitrarily well approximate the calibration task of a Heston model by the *deep calibration* approach. We also show that *deep hedging* arbitrarily well approximate optimal hedging strategies. Furthermore, our results indicate that the different specifications of the hedging method have large impact on the profit/loss (PnL) distribution of the strategy, hence we are able to identify, potentially more desirable specifications of the method in terms of their context and practical applicability when transaction costs are both included and excluded. We also find some practical limitations that could impair the compatibility of the methods, when applied to the Heston model. In its current form, the speed of the training procedure for the deep hedging algorithm is dependent on the speed of simulating trajectories and the accuracy of the calibration, when applied to empirical data. Because of this fact, the joint algorithm is

dependent on which simulation method is used *if* the calibrated parameter combination violates what is known as the *Feller-condition*, which restricts the parameter space of the Heston model based on the fact that, at a certain threshold, there is a non-zero probability of sampling negative variances, which some simulation schemes are better equipped to handle than others. However, some of the more complex and accurate simulation schemes are slower than the simpler ones. We deal with this problem by enforcing the condition in the calibration procedure, however, there are some downsides associated to performing constrained calibration in this manner. The constrained model is less accurate than the unconstrained model, which by extension, means that the constrained parameters are less able to reflect current market conditions than their unconstrained counterpart and thus also likely impair performance of the trained deep hedging algorithm when applied to real market data.

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

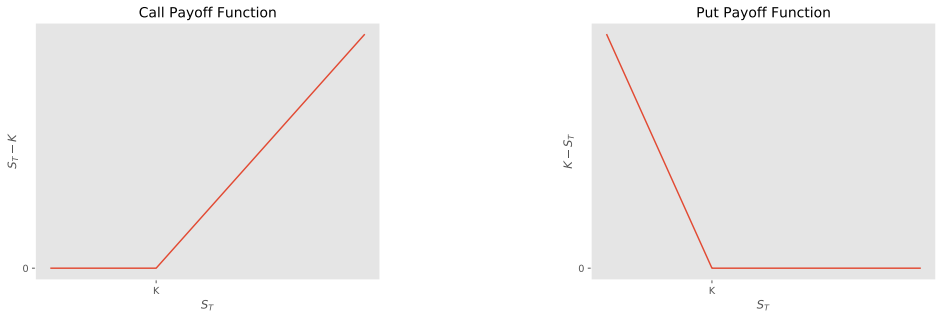
This section will introduce the concept of financial derivatives, specifically options and their properties and also serve to introduce the reader to the financial concepts in the formulation of the stated objective of the thesis.

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. Consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ in which Ω denotes the *sample space*, $\mathcal{F} := (\mathcal{F})_{t\in\mathbb{T}}$ is the *filtration* such that $\mathcal{F}_t$ represents all information available at t. $\mathbb{P}$ denotes the real probability measure on the space such that $\mathbb{P}[X \in A] \in [0, 1]$ for any reasonable set $A \subseteq \mathbb{R}$.

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 $\mathbb{P} \sim \mathbb{Q}$ (Björk (2009)). 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/Brownian motion 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. $W_t \sim \mathcal{N}(0, t) \iff dW_t \sim \mathcal{N}(0, dt)$. All increments in W are normally distributed with $\mathbb{E}[dW_t] = 0$ and $Var(dW_t) = dt$ as $Var(W_t) = t$

Figure 2: Example of sample path of Brownian Motion. Increments are normally distributed with variance dt and the initial point of the process is zero.

Furthermore, In order to describe and analyze stochastic processes, *stochastic differential equations* (SDE's) must be considered. A SDE differs from ordinary differential equations in that one or more terms is a stochastic process, typically some random noise, for example dW. 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 $\mathcal{F}_t = \sigma(W_s : s \leq t)$. 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 & 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

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

where T_j denotes the jth maturity and K_k denotes the kth strike. Tests of the predictive power of implied volatility through regression by Canina & Figlewski (1993) shows that implied volatility is not a very good predictor of future volatility and that historical volatility is a better proxy. In contrary to previous findings, implied volatility carries very limited amount of information about the future. However, for our purposes, its predictive capacity has little significance as it only serves as a alternative market quote for prices.

Stochastic volatility (SV) is a joint model of prices and variance in which prices 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*.

3.3 The Heston Model

The *Heston model* 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.

Definition 3.3. Heston SDE's:

$$dS_t = rS_t dt + \sqrt{V_t} S_t d\tilde{W}_t^{(S)} \tag{12}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t dt} d\tilde{W}_t^{(V)} \tag{13}$$

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

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

The parameters in Equations (12) and (13) 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

$$\begin{aligned}\frac{\partial C}{\partial t} + \frac{S^2 V}{2}\frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC + [\kappa(\theta - V) - \lambda V]\frac{\partial C}{\partial V} \\ + \frac{\sigma^2 V}{2}\frac{\partial^2 C}{\partial V^2} + \rho\sigma SV\frac{\partial^2 C}{\partial S \partial V} = 0\end{aligned} \tag{14}$$

where λ is the market price of volatility risk, for further description see Heston (1993). European call options that satisfies the PDE in Equation (14) 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.

Consider the general pricing task for call options in Equation (2) at $t = 0$, one can find a pricing function analogous to that of Black-Scholes model in Equation (8), denoted as C, in the following manner

$$\begin{aligned}C(S, K, T) &= e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[(S_T - K)^+\right] \\ &= e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[(S_T - K)\mathbf{1}_{S_T > K}\right] \\ &= e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[(S_T\mathbf{1}_{S_T > K}\right] - Ke^{-rT}\mathbb{E}^{\mathbb{Q}}\left[\mathbf{1}_{S_T > K}\right] \\ &= S_t P_1 - Ke^{-rT}P_2\end{aligned} \tag{15}$$

where $\mathbf{1}$ is the indicator function, P_1 and P_2 represent the probability of expiring in the money conditional on $\ln S_t$ and V_t such that

$$P_j = Pr(\ln S_t > \ln K | lnS_t, V_t), \qquad j = 1, 2. \tag{16}$$

However, these probabilities are obtained using different probability measures. The expectation under $\mathbb{Q}$ of a indicator function returns the *risk-neutral* probability of the indication, and in our specific case, the risk-neutral ITM-probability

$$\mathbb{E}^{\mathbb{Q}}\left[\mathbf{1}_{S_T > K}\right] = \mathbb{Q}(S_T < K) = \mathbb{Q}(\ln S_T > \ln K | \ln S_t, V_t) = P_2.$$

In order to evaluate $e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[S_T\mathbf{1}_{S_T > K}\right]$, one must change the original measure to another measure $\mathbb{Q}^S$, see e.g. Rouah (2013) for derivation. We can now state the pricing function as

$$\begin{aligned}C_{\mathcal{H}(\theta)}(S, V, T) &= S_t P_1 - Ke^{-rT}P_2 \\ &= S_t\mathbb{Q}^S(S_T > K | \ln S_t, V_t) - Ke^{-rT}\mathbb{Q}(S_T > K | \ln S_t, V_t).\end{aligned} \tag{17}$$

The pricing function in Equation (17) is derived by substituting the proposed solution in Equation (17), into Equation (14) which yields new PDE's, the reader is yet again referred to Heston (1993) for mathematical description. However, these probabilities are not analytically tractable, yet their *characteristic functions* satisfies the so called *adjusted* PDE's . 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 (16). The characteristic functions of the ITM probabilities are defined as

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

where D, C, and ϕ are functions such that the characteristic function satisfies the PDE, see p.331 Heston (1993). The probabilities in (16) can be obtained semi-analytically by the *Gil-Pelaez* inversion theorem as

$$P_j(\ln S_T, V, T; \ln(K)) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{\exp\left(-i\phi \ln(K)\right) f_j(\ln S_T, V, T; \phi)}{i\phi}\right] d\phi \tag{18}$$

where $Re(z)$ denotes the real part of any complex number z. The solution to Equation (18) 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

$$S_t = S_{t-1} + rS_{t-1}\Delta t + \sqrt{V_{t-1}} S_{t-1}\sqrt{\Delta t} Z_{t-1}^{(S)} \tag{19}$$

$$V_t = V_{t-1} + \kappa(\theta - V_{t-1})\Delta t + \sigma\sqrt{V_{t-1}\Delta t} Z_{t-1}^{(V)} \tag{20}$$

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

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 3 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 (17) as a measure of convergence, there still exists some non-negligible bias in the EM scheme. In order to (largely) avoid 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* with d degrees of freedom and non-centrality parameter λ, i.e. $V_t|V_u \sim \chi\prime_d^2(\lambda)$, where $d = 4\kappa\theta^{\mathcal{H}}/\sigma^2$ and $\lambda = \frac{4\kappa e^{-\kappa(t-u)}}{\sigma^2(1-e^{-\kappa(t-u)})}V_u$. In order to follow the exact simulation scheme, first one has to have explicit expressions for S_t and V_t given S_u and V_u, hence the representation of the Heston SDE's in Equations (12) and (13) in their integral form

$$S_t = S_u \exp\left[r(t-u) - \frac{1}{2}\int_u^t V_s ds + \rho\int_u^t \sqrt{V_s}dW_s^{(1)} + \sqrt{1-\rho^2}\int_u^t \sqrt{V_s}dW_s^{(2)}\right]$$

$$V_t = V_u + \kappa\theta^{\mathcal{H}}(t-u) - \kappa\int_u^t V_s ds + \sigma\int_u^t \sqrt{V_s}dW_s^{(1)} \tag{21}$$

where $W^{(1)}$ and $W^{(2)}$ are *independent* Brownian motions. If one seeks to sample from the marginal distribution of (S_t, V_t), Broadie & Kaya (2006) propose the following algorithm.

- *Step 1.* Generate a sample from the distribution of V_t given V_u, utilizing the fact that the marginal distribution is a non-central chi-square.
- *Step 2.* Generate a sample from the distribution of $\int_u^t V_s d_s|V_t, V_u$.
- *Step 3.* Recover $\int_u^t \sqrt{V_s}dW_s^{(1)}|V_t, V_u, \int_u^t V_s d_s$ from Equation (21).
- *Step 4.* Generate a sample from $S_t|\int_u^t \sqrt{V_s}dW_s^{(1)}, \int_u^t V_s d_s$.

For explicit instructions on how to sample and recover these quantities the reader is referred to Broadie & Kaya (2006). 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 intensity. Hence, other discrete approximations has been introduced inspired by the exact simulation algorithm, e.g. the *Quadratic exponential* scheme introduced by Andersen (2007), which are more widely used in the industry as they have similar performance as the exact simulation scheme but retains the computational efficiency, and relative simplicity, of the Euler-scheme. 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 3.

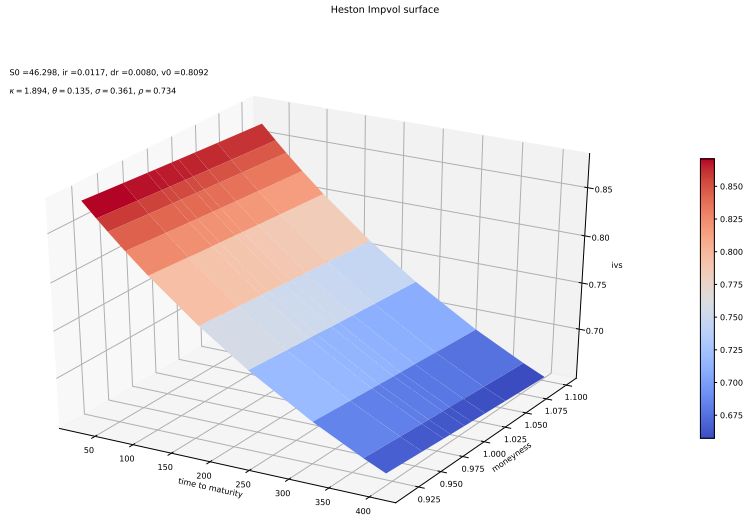


Figure 3: Example of implied volatility surface from a Heston model with parameters $\kappa = 1.894, \theta^{\mathcal{H}} = 0.135, \sigma = 0.361$ and $\rho = 0.734$. This parameter combination generates a rather simple implied volatility (IV) surface. The model has the capacity to generate skewness in all dimensions. For example, the model can capture what is called *volatility term structure* and *smile*. See Cont & Fonseca (2001).

The ability of the Heston model to fit the implied volatility surface reasonably well is the main reason that the Heston model is so popular for practitioners, besides the Black-Scholes model. Recall that the Black-Scholes model explicitly assume constant variance, which by extension means that the variance process in Equation (13) is redundant and thereby assumes a flat implied volatility (IV) surface. A variable IV surface transports a lot of information, it is even more informative than prices. The skewness of a IV surface tells how much risk-premium is attached to each strike. Generally, sellers of deep out-of-the-money (OTM) calls charges a larger premium than 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. This means that the implied volatility surface

does not only reflect market expectations of future volatility, the specific shape of the surface can reveal information about the shape of the terminal distribution as the shape of the market IV surface indicates leptokurtic distribution of spot returns and by extension, the divergence from normality and the Black-Scholes framework. This allows the Heston model to adapt better to changes in the distribution of spot 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

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 $\theta \in \Theta$. Furthermore, we shall analyze options characterized by strike K and maturity T. Consider the market pricing function $\mathcal{P}^{MKT}$, such that market prices are obtained by $\mathcal{P}^{MKT}(T, K)$, and the model pricing function P is the Heston model pricing function, see Equation (17). The general task of parameter calibration can be stated as learning the pricing map

$$P(\theta, T, K) \mapsto \mathcal{P}^{MKT}(T, K) \tag{23}$$

thus, we are interested in finding a parameter combination that *accurately* maps model prices to market prices. We are interested in finding a approximate pricing function $\tilde{P}$ such that $\tilde{P} \approx \mathcal{P}^{MKT}$ by some numerical method. Hence, we can state the approximation objective as

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

in which $d(\cdot)$ denotes some appropriate distance function. Consider $d(x, y) = \sum_i \sum_j (x_{ij} - y_{ij})^2$, Then the calibration function in Equation (24) is restated as

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

where 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. Obviously, 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 IV map 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, which is where the advantage of the neural network approach becomes evident.

3.5 Continuous time Martingale Pricing & Dynamic Replication

To understand the relation between hedging and pricing, 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 such strategies, 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 $S = \{S_t\}_0^T$ on some filtered probability space $(\Omega, \mathcal{F}, \mathbb{P})$. 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, which means 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 $u < t$. 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 square integrable process

$$S = S_0 + M + A$$

where M is a local martingale and A is a $\mathcal{F}$-adapted process, i.e. $\{A_t\}$ *adapts* to the information in the filtration such that A is dependent on $\mathcal{F}$, since $\{A_t\}$ is $\mathcal{F}_t$-measurable A cannot reveal more information than the what the filtration contains. Note that is the inverse is true we call the process *predictable*. A local martingale is formally defined as a martingale process such that, for a sequence of *stopping times* τ_n, for which $\mathbf{1}_{\{\tau_n>0\}}$, M is a martingale. The result of this is that S is a "good" integrable process and thus representable under the Itô interpretation. 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 at T

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

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 $\mathcal{F}_{t+i}$ 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

$$\Pi_t = \delta S_t + \eta_t$$

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

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

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

$$\Pi_t = C_t + \int_0^t \delta_u dS_u,$$

$$H_t = H_0 + \int_0^t \delta_u dS_u \iff C_t = H_0$$

is self financing as $C_t = C_T = H_0$, i.e. does not require any insertion of capital beyond the initial input H_0 and optimal as it produces the claim H from the initial capital 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). Thus, 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. The concept of market incompleteness describes the uncertainty arising with respect to asymmetric information, which by extent gives rise to non-constant volatility, however we will only consider the concept of market incompleteness by asymmetric information as the specific case of variable volatility is beyond the scope of this thesis, see Föllmer & Schweizer (1990) for detailed description.

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*

$$H = H_0 + \int_0^T \delta_u dS_u + L_T \tag{28}$$

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

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

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.

We shall now consider the predictable process $\delta := \{\delta_t\}_0^T$. 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 stochastic element of H, 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). In the case of the Black-Scholes model, dS is defined as in Equation (6) which allows analytical expressions of how δ eliminates dW and is thus void of risk and through absence of arbitrage, earn the risk free rate. The practice in which this is achieved is called *delta-neutralization*. One can show by Itô calculus that δ or the so called *hedging parameter* is the partial derivative of the value of H with respect to the spot value, $\delta := \frac{\partial H}{\partial S}$ where t is left out for notational convenience. 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 (13), 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 portfolio of H such that the instantaneous rate of change is equal to the risk free rate, i.e. $d\Pi = r\Pi$, one has to include a second hedging parameter and contingent claim. Consider the portfolio

$$\Pi = -C(S, V, T) + \delta S + \phi U(S, V, T)$$

in which U is also a contingent claim on S in Equation (13) and represents claim such that ϕU neutralizes $W_t^{(V)}$ in Equation (13). The hedging parameters δ, ϕ must be $\phi = \frac{\partial H}{\partial V} / \frac{\partial U}{\partial V}$ and $\delta = \phi \frac{\partial U}{\partial S} + \frac{\partial H}{\partial S}$ in order for both sources of risk $W^{(S)}$ and $W^{(V)}$ to be eliminated and can be derived by the Heston PDE, see Heston (1993). We shall however exclude volatility related risk for computation of optimal hedging strategies from our analysis since the practical difficulty in implementation of, either representing U as a claim on S, similar to C, or considering U as a direct claim on V, represented by a *over-the-counter (OTC)* hedging vehicle, called a *variance swap*, see e.g. Buehler et al. (2019) for its practical implications in terms of hedging. Furthermore, we believe that the complexity of hedging with OTC-derivatives is beyond the technical scope of this thesis. Hence, our "optimal" hedging strategies are actually *pseudo* optimal since $W^{(V)}$ is ignored. Going forward, when we refer to "approximation of optimal hedging strategies under the Heston model", the reader has to keep the reasoning presented above in mind. We shall now introduce the computation of spot risk hedging, recall the *delta-neutral* hedge.

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 (16). 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 (17) and approximate the partial derivative by choosing a sufficiently small ε

$$\delta_t \approx \frac{C_{\mathcal{H}(\theta)}(S_t + \varepsilon, V, T - t) - C_{\mathcal{H}(\theta)}(S_t - \varepsilon, V, T - t)}{2\varepsilon} \tag{29}$$

for each $t \in [0, T]$. The resulting stochastic process 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 very important 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 (28). Thereby, H needs to be discretized with respect to time so that

$$H = H_0 + \sum_{i=1}^{n} \delta_{t_i} \Delta S_{t_i} + L_T \tag{30}$$

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 probability space that is square integrable, $H \in \mathcal{L}^2(\Omega, \mathcal{F}, \mathbb{P})$. 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 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 $\mathbb{P}.as.$ 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 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,

$$\pi(-Z) := \inf_{\delta \in \Delta} \{\rho(-Z + G_T(\delta) - C_T(\delta))\} \tag{31}$$

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).

In order to apply the optimization in Equation 31 to our current setting in which 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 (30) by choice of $\delta \in \Delta$, 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 under the real probability measure $\mathbb{P}$. 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. One reasonable way of minimizing risk could be to maximize the *expected utility* of the replication. Consider the *entropic* risk measure

$$\rho(Z) = \frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda Z)] \tag{32}$$

in which $\lambda > 0$ denotes the risk aversion parameter. Note that the expectation in Equation (32) is the *expected exponential utility* of Z. In order to prove that when one evaluates the position $X \in \mathcal{X}$ by the expected utility $\mathbb{E}[u(X)]$, one obtains a convex risk measure, then we have to mention the concept of *acceptance sets*, which are sets of positions that are acceptable to a regulator. The following notation and setup originates from Föllmer & Schied (2008). Any monetary risk measure induces a acceptance set

$$\mathcal{A}_\rho = \{X \in \mathcal{X} \ : \ \rho(X) \leq 0\}.$$

It turns out that the acceptance set completely determines ρ in the sense that

$$\rho(X) = \inf\{m \in \mathbb{R} \ : \ m + X \in \mathcal{A}_\rho\}.$$

If $\mathcal{A}_\rho$ satisfies the following properties

$$\begin{aligned} \mathcal{A}_\rho \cup \mathbb{R} &\neq \emptyset, \\ \inf\{m \in \mathbb{R} \ : \ m + X \in \mathcal{A}_\rho\} &> -\infty, \text{ for all } X \in \mathcal{X}, \\ X \in \mathcal{A}_\rho, \ Y \in \mathcal{X}, Y \geq X &\implies Y \in \mathcal{A}_\rho \end{aligned}$$

then $\rho_{\mathcal{A}}$ is a monetary risk measure. If $\mathcal{A}_\rho$ is a convex set, then so is the associated risk measure $\rho_{\mathcal{A}}$. In the case where one consider a risk measure with the expected utility $\mathbb{E}[u(X)]$, where the utility function $u : \mathbb{R} \mapsto \mathbb{R}$ is concave and strictly increasing, then the set for a given threshold c

$$\mathcal{A}_\rho = \left\{X \in \mathcal{X} \ : \ \mathbb{E}^{\mathbb{Q}}[u(X)] \geq u(c)\right\}$$

is valid and a convex acceptance set associated to a convex risk measure ρ. This is called a *utility-based shortfall* monetary risk measure. If $u(x) = -e^{-\lambda x}$ and $c = 0$, one obtains the entropic risk measure in Equation (32). For further details, see Föllmer & Schied (2008). The optimal hedging strategy under the entropic risk measure is then obtained by substituting ρ in Equation (31) with the entropic risk measure

$$\pi(-Z) = \inf_{\delta \in \Delta} \left\{ \frac{1}{\lambda} \log \mathbb{E}\left[\exp(-\lambda(-Z + G_T(\delta) - C_T(\delta)))\right] \right\} \tag{33}$$

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 (32), in the sense that it is so called *coherent*. Coherent measures of risk are a stronger form of risk measures since, a convex 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 α-level value at risk, and is nothing more than the average of the α-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

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

in which $x_{(\alpha)} := \inf\{x \in \mathbb{R} : \mathbb{P}[X \leq x] \geq \alpha\}$ and represents the lower quantile of X and $\mathbf{I}$ is a indicator function. 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 (31) as

$$\pi(-Z) = \inf_{\delta \in \Delta} ES_\alpha(-Z + G_T(\delta) - C_T(\delta)). \tag{34}$$

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

$$\pi(-Z) = \inf_{\delta \in \Delta} \{\mathbb{E}[\ell(-Z + p_0 + G_T(\delta))]\} \tag{35}$$

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 smallest variance of the replication error

$$\pi(-Z) = \inf_{\delta \in \Delta} \left\{\mathbb{E}\left[(-Z + p_0 + G_T(\delta))^2\right]\right\} \tag{36}$$

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

$$\mathbb{E}[(-Z + p_0 + G_T(\delta))^2] = Var(-Z + G_T(\delta)) \tag{37}$$

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) 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 $F^*(\cdot) \approx F(\cdot, \hat{w})$ for the function $F^*(\cdot)$. 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

$$\begin{aligned} w^l : \mathbb{R}^{N_l} \mapsto \mathbb{R}^{N_{l+1}} \text{ for some } 1 \leq l \leq L-1 \\ x \mapsto A^{l+1}x + b^{l+1} \end{aligned} \tag{38}$$

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 (38) 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:

$$F := F_L \circ \cdots \circ F_1 \tag{39}$$

where each component is in the form $F_l := \sigma_l \circ W^l$. The first term in this, $\sigma_l : \mathbb{R} \mapsto \mathbb{R}$, denotes the *activation function*. 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. The theorem shows that if the objective is to approximate a real valued continuous function, a simple neural network with at least three layers, ie 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 $\mathbb{R}$ values 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*

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

such that

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

Thereby, any continuous real valued function can be approximated 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 4, has an output value *between* 0 and 1. It has weights $A_1, A_2, ...$ just like the perceptron, and an overall

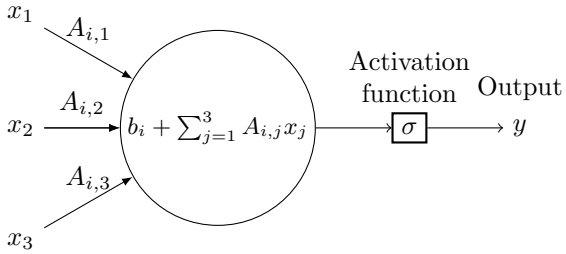


Figure 4: Example of a sigmoid neuron with three input variables (x_1, x_2 and x_3). The neuron adds the bias with the sum of the the input multiplied with the assign weights, which then sends through the sigmoid activation function σ.

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, smoothness meaning 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 Terminology - Epochs, Iterations & Batches

To be able to understand some further ANN-concepts introduced later in this thesis, we first introduce some of the terminology required.

Recall that a NN learns to approximate a given function by adjusting its parameters. This is done after some data has been run through the model, also called a forward pass and backward pass. The definition of an *epoch* in NN is when an entire data set is passed forward and backward through a network once. To sufficiently train the network a data set might have to be run through the network more than once, hence, multiple epochs are required. However in many cases the data set is too large and needs to be divided into pieces, which are called *batches*. Batches are passed through the network multiple times, the number of times it is passed through the network is the number of *iterations*.

5.4 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, done by training with real values, ie the input is fed to the ANN and outputs are produced, these are then compared with the real values. If the ANN produced values that are far away from the real value then the model parameters will have to change. One could argue that the learning of an ANN is made up be two main parts. The first part is the *cost function*, which is the function that compares the produced output with the desired value and thus evaluates the quality of the prediction. The second part is how the weights should change. This is a really complicated task since the number of weights can be rather large. One way to do this is by using *gradient descent* and *backpropagation*.

5.4.1 Gradient Descent

Gradient-Based learning is based on *gradient descent*. Gradient descent uses the derivative to minimize the cost function. For example, $y = f(x)$. Then $f(x - \epsilon\ sign(f'(x))$ is smaller than $f(x)$ for small enough ϵ. $f(x)$ can be reduced to the minimum by iterating small changes in x with the opposite sign of the derivative. 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.

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, 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

$$w = w - \eta \cdot \nabla_w F(w)$$

where w is the network parameters, η is the learning rate and $\nabla_w F(w)$ 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. 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

$$w = w - \eta \nabla_w F(w; z^{(i)}) \tag{40}$$

where $z^{(i)}$ is a randomly selected example, in the training set. 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).

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

where n is the size of the mini-batch. 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

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

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:

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

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.

$$\begin{aligned} g_t &= \nabla_{w_t} F(w_t - \eta m_{t-1}) \\ m_t &= \eta m_{t-1} + \eta g_t w_{t+1} = w_t - m_t \end{aligned} \tag{41}$$

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

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

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 $g_t = \nabla_{w_t} F(w_t)$

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

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 (41), Nestrov momentum is added by replacing the previous steps estimated momentum with the current steps estimated momentum.

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

5.4.2 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^l_{ji} 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^l_j denote the bias of the jth neuron in the lth layer and denote the activation of the same neuron be denoted as a^l_j. For a given activation function $\sigma(x)$

$$a^l_j = \sigma \left(\sum_n A^l_{jn} a^{l-1}_n + b_j \right)$$

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

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

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

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

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

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

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

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

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:

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

The complete backpropagation algorithm is presented in Algorithm 1.

Algorithm 1: Backpropagation

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

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 perfected 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. Thus we seek to apply neural networks by finding a optimal set of parameters that replicate the "image" of the implied volatility surface on a per pixel basis and then find the parameter set that generate that surface.

To reiterate, consider the Heston model $\mathcal{H}$ that is parameterized by $\theta \in \Theta$ with the pricing function P, described in Equation (17), that generates prices which are approximately equivalent to market prices, i.e. $P \approx \mathcal{P}$ in which the market pricing function is $\mathcal{P}$. The classical calibration process can be described as choosing a parameter vector $\hat{\theta} \in \Theta$ that minimizes the distance between the market quotes and the model implied quotes as in Equation (24) in Section 3.4. However, we shall consider first learning the model functional in terms of implied volatility surface, then applying classical calibration procedures to obtain the parameter set that best generates the trained volatility surface, for some appropriate distance function.

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 this problem 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 $P(\theta, T, K)$ and $\mathcal{P}(T, K)$ in Equation (23) 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 (3.2), generated by the Heston model parameterized by $\theta \in \Theta$, and $\Sigma_{BS}^{MKT} = \left\{\sigma_{BS}^{MKT}(T_j, K_k)\right\}_{j=1,k=1}^{n,\,m}$ respectively. 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 $d(\cdot)$ denotes an appropriate distance function. We are first interested in approximating the risk-neutral pricing function in the Heston model, see Equation (17), and by extension obtain implied volatilities, by training a neural network as an approximator to the IV surface. Let $F := \{F(\omega; \theta, T_j, K_k)\}$ denote a neural network approximation of the IV-surface generated by $\mathcal{H}(\theta)$ parameterized by ω (weights and biases). Then we are interested in approximating the map

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

by finding a optimal parameterization $\hat{\omega}$ of the neural network. If one specifies the neural network loss function $d(\cdot)$ as $d(x,y) = \sum_i \sum_j (x_{ij} - y_{ij})^2$, the training of the network can be described by the optimization

$$\hat{\omega} := \underset{\omega \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{N_{Train}} \sum_{j=1}^{n} \sum_{k=1}^{m} \eta_{j,k} \left(F(\omega; \theta_i, T_j, K_k) - \tilde{\sigma}_{BS}^{\mathcal{H}(\theta_i)}(T_j, K_k) \right)^2 \tag{42}$$

in which η_{jk} denotes weights that are assigned manually to each option in the grid and signals the relative importance of a IV "pixel", for example one might want to minimize the distance for OTM or ITM to a larger extent than to that of ATM options, however we shall not "favour" any option, hence $\eta_{j,k} = 1$ for all j, k.

We want to train a feed forward neural network over N_{Train} pseudo-random parameter samples (θ_i) generated by pre-specified prior distributions. Furthermore, prior distributions for the observable quantities, spot price S_0, risk free rate r, dividend rate q and initial variance V_0 must be specified as well. For both parameters and observables, we have chosen *uninformative* prior distributions, specifically a collection of beta and uniform distributions over reasonable boundaries. For the model parameters, boundaries shall be specified in order to limit the search space of the neural network to a "reasonable space", see Table 2. These boundaries were chosen by lit-

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.

erature 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 (42) 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 5.4.1. To reiterate, 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 optimized weights are obtained and the implied volatility map is learned, we shall denote the trained network as $\tilde{F}(\theta) = F(\hat{\omega}, \theta)$ and we denote *pixel-wise* prediction as $\tilde{F}(\theta)_{j,k} = F(\hat{\omega}, \theta, T_j, K_k)$. If one retains the "least squares" distance function in Equation (42), 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.

$$\hat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{F}(\theta)_{j,k} - \sigma_{BS}^{MKT}(T_j, K_k) \right)^2 \tag{43}$$

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

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{\sigma}_{BS}^{\mathcal{H}(\theta)}(T_j, K_k) - \sigma_{BS}^{MKT}(T_j, K_k) \right)^2 \tag{44}$$

where $\tilde{\sigma}_{BS}^{\mathcal{H}(\theta)}(T_j, K_k) = \tilde{F}(\theta)_{j,k} = F(\hat{\omega}, \theta, T_j, K_k)$. 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 (43). 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. 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 prespecified *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 (43), 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 (43) 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.

However, as we have discussed in Subsection 3.3, simulation schemes are bounded by the Feller condition. Remember that 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 solve this one can restrict the search space such that the condition is met by, dynamically, adjusting the upper boundaries of $\Theta \in [\Theta_{min}, \Theta_{max}]$ such that $\Theta_{max} = \left\{\left(\kappa_{max}, \theta_{max}^{\mathcal{H}}, \sigma_{max}, \rho_{max}\right) \ : \ 2\kappa_{max}\theta_{max}^{\mathcal{H}} \leq \sigma_{max}^2\right\}$. Hence, one can now apply these calibration boundaries in the second step of the method in Horvath et al. (2019). Hence if this condition is applied, the calibration search space is reduced and the calibration of $\hat{\theta}$ is constrained such that the optimal solution is no longer a minimizer to Equation (43) and as such, the calibration task must be reformulated as a constrained optimization

$$\begin{aligned} \hat{\theta} &:= \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \sum_{k=1}^{n} \left(\tilde{F}(\theta)_{j,k} - \sigma_{BS}^{MKT}(T_j, K_k)\right)^2 \\ \text{Subject to} &: \Theta_{max} = \left\{\left(\kappa_{max}, \theta_{max}^{\mathcal{H}}, \sigma_{max}, \rho_{max}\right) \ : \ 2\kappa_{max}\theta_{max}^{\mathcal{H}} \leq \sigma_{max}^2\right\}. \end{aligned} \tag{45}$$

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.

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 independent 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 hedging strategies was implemented through the use of classic greek models. However, when trading under real market conditions, such classical models always introduce some overrides. The problem is that these errors in the classic derivative models are not systematic and thereby rely on explicit human configurations to alleviate such overrides.

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 produce such overrides. Furthermore, this technique lends it self to be applicable to the cases in which the classical hedging models do work very well. This is generally the case in markets in illiquid markets such as the OTC-derivative market. We will however only consider the case of liquid European vanilla options. Furthermore, this means that the modeller is completely free to specify the market conditions in which, hopefully, the optimal hedging strategy is obtained.

7.1 Market Setting & Objective

We will consider a discrete time market with terminal time T and trading is only allowed at $0 = t_0, \dots, t_n = T,\ t \in [0, T]$. The probability space is fixed on $(\Omega, \mathcal{F}, \mathbb{P})$ in which $\mathcal{F}$ denotes the complete feature set and is thereby defined as the information which is generated by the process that describes all information available at each t. Furthermore, we will assume that S is described by the spot process in a Heston model parameterized by $\mathcal{H}(\theta)$, which is obtained by deep calibration as described in Section 6 and is $\mathcal{F}$ adapted, which means we will consider hedging in a incomplete market. The contingent claim which we seek to hedge $-Z$ is a random variable that is $\mathcal{F}_T$ measurable.

Recall from Section 4 that the strategy applied in order to hedge $-Z$ is implemented by trading in S with the weighting $\delta = \{\delta_t\}_0^T$, as we ignore volatility risk for ease of implementation. If one represents Z as a call option and injects cash at t_0, the terminal wealth is defined as

$$\mathrm{PnL}_T(Z, p_0, \delta) = -Z + p_0 + G_T(\delta) - C_T(\delta) \tag{46}$$

in which the agent sells a call option and

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

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

$$C_T(\delta) = \sum_{i=0}^{n} \varepsilon(\delta_{t_{i+1}} - \delta_{t_i}) \tag{47}$$

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 (46) 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. 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. However, as previously mentioned, we really consider "pseudo optimality" since the market setting implies that S is dependent on V and the hedging strategies does not take this into account, hence the optimal hedge will also have to include a strategy on some claim on V or another claim on S that depend on V.

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

In order to justify the application of neural networks as *approximators* of optimal hedging strategies, one first need to reformulate the *constrained* hedging problem 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 (33), 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.

As discussed in Section 4, expected shortfall, entropic risk, and the quadratic criterion are reasonable monetary measures risk. Specifically, the entropic risk measure is convex, expected shortfall is a coherent risk measure. When one consider optimal hedging strategies under such measures, one can seek optimality in terms of minimizing the PnL-distribution in Equation (46), which is called minimizing shortfall risk or replication error. As noted in Section 4, the quadratic criterion is neither a convex or coherent monetary risk measure. However, approximating $-Z$ in $\mathcal{L}^2$ 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. 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, that are beyond the technical scope of this thesis. However, the idea is that one can define a $\mathcal{F}_T$ measurable map D, such that one can project the unconstrained strategies Δ^u onto the constrained set of strategies δ. Here the idea is that $\Delta = (D \circ \delta^u) \subset \Delta^u$, then the unconstrained formulation of the entropic hedging objective in Equation (33) can be described as

$$\pi(-Z) = \inf_{\delta \in \Delta^u} \frac{1}{\lambda} \log \mathbb{E}(\exp(-\lambda[-Z + G_T(D \circ \delta)) - C_T(D \circ \delta)]) \tag{48}$$

where $D \circ \delta$ is the constrained projection of δ^u onto δ. Note that $D \circ \delta = \delta$, 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 (48), by neural networks. Recall that the filtration $\mathcal{F}$ gives rise to $\mathcal{F}_t$ that denotes the richest possible *feature set* at each t and thus contains all information about the probability space, as such the hedging strategy will have to contain such information. Thus, the feature set that we will choose is only based upon the realization of the Heston model on the probability space, as S is a $(\mathcal{F}, \mathbb{Q})$-Markov process. Following the notation and setup of Buehler et al. (2019), we have that the space of admissible *neural network* hedging strategies $\Delta_M \subset \Delta^u$, in which M denotes the dimensions of parameter combinations for the neural network.

$$\begin{aligned} \Delta_M &:= \left\{ \{\delta^\theta_{t_i}\}_0^n \in \Delta^u \ : \ \delta_{t_i} = F_{t_i}(S_{t_i}, \delta_{t_{i-1}}),\ F_{t_i} \in \mathcal{NN}_{M,d0,d1} \right\} \\ &= \left\{ \{\delta^\theta_{t_i}\}_0^n \in \Delta^u \ : \ \delta^\theta_{t_i} = F^{\theta_{t_i}}(S_{t_i}, \delta_{t_{i-1}}),\ \theta_{t_i} \in \Theta_{M,d0,d1} \right\} \end{aligned} \tag{49}$$

in which $F^{\theta_{t_i}}$ denotes a fixed neural network $\mathcal{NN}$ such that $F^{\theta_{t_i}} \in \mathcal{NN}^{\sigma_{Elu}}_{M,d0,d1}$ and θ is the network parameters, i.e. weights and biases, $\theta = (w, b)$ (see Section 5). For the case of the entropic risk measure, replace Δ^u with Δ_M in Equation (48) and obtain the formulated numerical optimization of finding optimal parameter combinations for the neural network

$$\begin{aligned}\pi(-Z) &= \frac{1}{\lambda} \log \inf_{\delta \in \Delta_M} \mathbb{E}(\exp(-\lambda[-Z + G_T(D \circ \delta) - C_T(D \circ \delta)]) \\ &= \frac{1}{\lambda} \log \inf_{\theta \in \theta_M} \mathbb{E}(\exp(-\lambda[-Z + G_T(D \circ \delta^\theta) - C_T(D \circ \delta^\theta)]).\end{aligned}$$

Let $J(\theta) = \mathbb{E}(\exp(-\lambda[-Z + G_T(\delta^\theta) - C_T(\delta^\theta))])$. Then the approximately optimal hedging strategy under the entropic measure is defined as

$$\pi^M(-Z) = \frac{1}{\lambda} \log \inf_{\theta \in \Theta_M} J(\theta). \tag{50}$$

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 θ minimizes the loss function $J(\theta)$. This strategy can arbitrarily well approximate strategies in δ since

$$\lim_{M \to \infty} \pi^M(-Z) = \pi(-Z) \tag{51}$$

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 θ in (50), 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 θ. 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 direction gradient, changing the network parameters along the way, and, hopefully, find a global minimum of the cost function. That is to say that if we let J be the function whose minimum we want to find then starting with a initial guess for the network parameters $\theta^{(0)}$

$$\theta^{(j+1)} = \theta^{(j)} - \eta_j \nabla J_j(\theta^{(j)}), \tag{52}$$

for some small $\eta_j > 0$, finds the local minimum of J as $j \to \infty$. Equation 52 describes the simple *gradient descent* algorithm, however, considerations of computational efficiency and avoidance of local minima, discussed in Section 5.4.1, we have chosen to use adaptive momentum optimizer (Adam) instead, which updates the network parameters θ with

$$\theta_{j+1} = \theta_j - \frac{\eta}{\sqrt{\frac{v_j}{1-\beta_2}} + \epsilon} \cdot \frac{m_j}{1-\beta_1}$$

where

$$\begin{aligned}m_j &= \beta_1 m_{j-1} + (1-\beta_1)\nabla_w J(\theta_j) \\ v_j &= \beta_2 v_{j-1} + (1-\beta_2(\nabla_w J(\theta_j))^2.\end{aligned}$$

Furthermore, under the entropic risk measure ,the function J_j is given by

$$J_j(\theta) = \sum_{m=1}^{N_{Batch}} \exp(-\lambda \left[-Z + G_T(\delta^\theta - C_T(\delta^\theta))\right] (\omega_m^{(j)}) \frac{N}{N_{Batch}} \mathbb{P}[\{\omega_m^{(j)}\}]. \tag{53}$$

The algorithm avoids getting stuck in a local minimum because as it uses random batch-sampling. The function is then based on the average value of the observations in the sample, where the samples are chosen with uniformly probabilities. 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 53 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. Firstly, we specify and discuss the parameterization each neural network, i.e. architecture, activation etc., then we proceed by visualizing, interpreting and discussing the results of each method in terms of their objective, respectively. Furthermore, we also discuss the joint methodology of utilizing neural networks for both tasks. We implemented both methods 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 method, the availability of which we are grateful

8.1 Calibration Method

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. Maturities was 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. See Figure 5 for network architecture. We have decided to generate 100000 synthetic parameter combinations from the prior distributions over the parameter space as described in Section 6, the data is then split into 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 5. We set σ_{ELU} as the activation function.

We then employ the differential evolution algorithm, see Section 6, in order to find the optimal parameter combinations that minimize the loss in Equation (43) from the neural network predicted implied volatility surface, $\tilde{F}$. The calibrated parameter combinations $\hat{\theta} = (\hat{\kappa}, \hat{\theta}^{(\mathcal{H})}, \hat{\sigma}, \hat{\rho})$ are now obtained. The methodology can be roughly split into 4 consecutive sections.

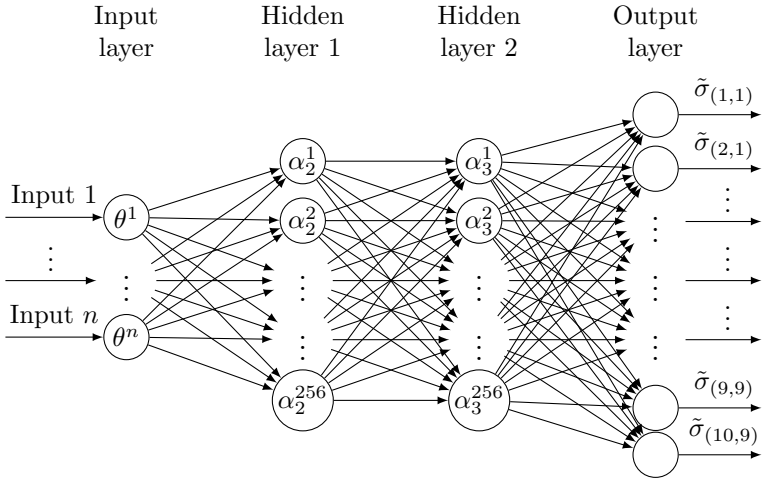


Figure 5: Neural Network Architecture for model calibration. The input layer is the Heston model parameters and output layer is the implied volatility surface. Here one learns the parameters to implied volatility map. We are interested in finding the inverse relation for empirical data. Hence, the differential evolution step.

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 extracted 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. Hence, in order to show the approximating properties of neural networks in terms of calibration, one has to isolate this inherent inability of the pricing model. As such, we shall first consider the calibration of the Heston model to synthetic implied volatility data generated by prior distributions over the parameter space. 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 6.

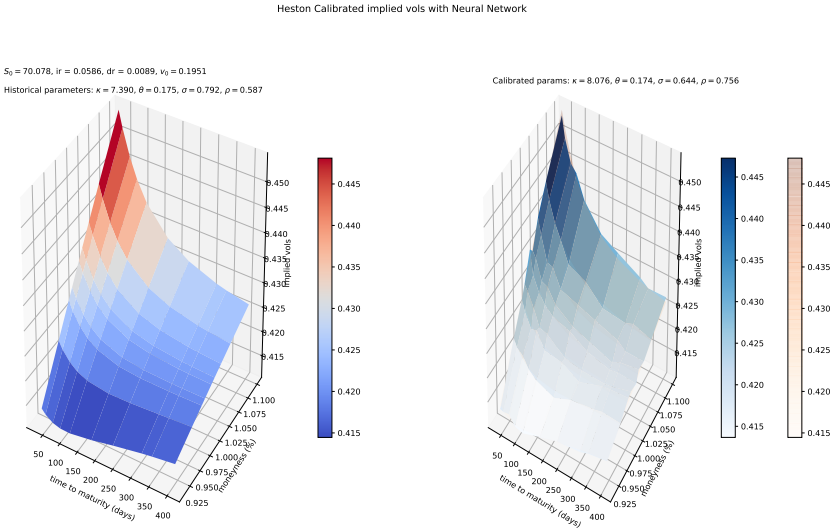


Figure 6: Synthetic and calibrated Heston surface. Right hand side plots both calibrated (blue scale) and synthetic volatility surface (red scale). Synthetic parameters: $\kappa = 7.390, \theta^{(\mathcal{H})} = 0.175, \sigma = 0.792, \rho = 0.587$. Calibrated parameters: $\hat{\kappa} = 8.076, \hat{\theta}^{(\mathcal{H})} = 0.174, \hat{\sigma} = 0.644, \hat{\rho} = 0.756$. Here one can see that the approximating capabilities of the method is very strong.

As one can see in Figure 6, 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. However, one need to consider whether the difference in the calibrated volatility surface is constant over maturities, see Figure 7.

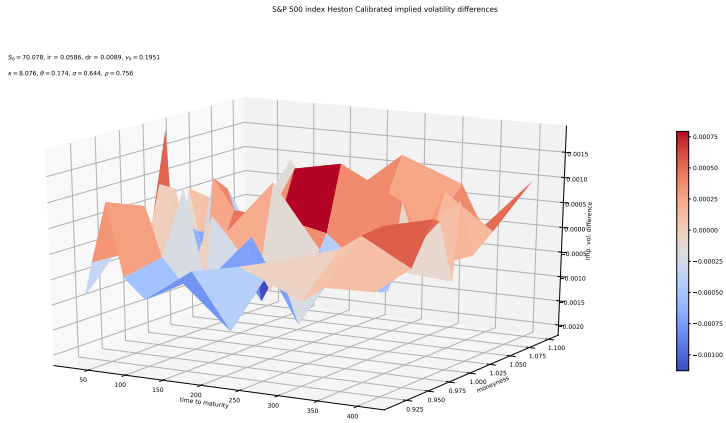


Figure 7: Difference between calibrated surface and simulated surface, i.e. visualization of distance function over surface in Equation (43), from model presented in Figure 6. Clearly, errors are very small. No clear underestimation trend present, however, not constant calibration error.

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 neural network calibration algorithm should also be able to calibrate to empirical data. Firstly however, we shall briefly analyse the image of the implied volatility surface from S&P 500 call options on 2019-11-08.

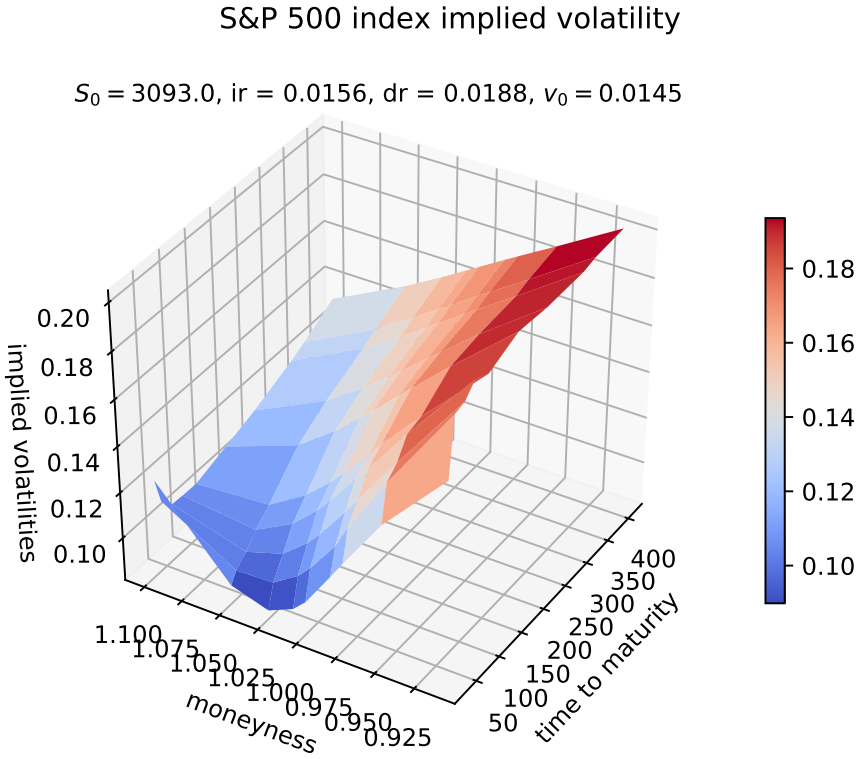


Figure 8: The call implied volatility surface is heavily skewed with respect to tail options, see for example the shape of the short maturity left tail options, i.e. out-of-the-money (OTM) call options with relatively short time to maturity. We suspect that the Heston model will not be able to capture such shapes.

As one can see in Figure 8, the pricing model may be too "rigid" in order to replicate the inherent market pricing premium for short term tail options Rouah (2013). This is a well known shortfall of the Heston model. The price of short term tail options has a form of negative jump risk premium attached which is highly related to the state of volatility in markets, i.e. when markets are in a state of higher volatility, the negative jump risk premium is larger and thus the implied probability of such jump occurrence is larger than in states of lower volatility, for more details see Pan (2001). Thus, for a model to be more in line with empirical data, one might include stochastic volatility models with jumps. The Heston model quite easily extends to accommodate jumps and has been extensively researched in the literature, see Bates (1993).

Figure 9 displays the performance of our calibration method to empirical data.

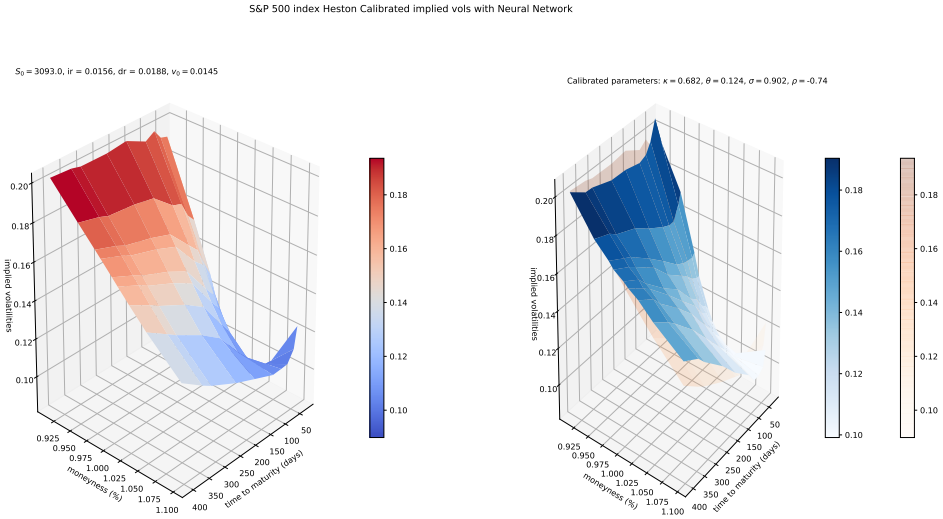


Figure 9: Left hand side visualises the implied volatility surface in Figure 8. The right hand side visualises the calibrated implied volatility surface (blue scale) versus the *actual* volatility surface (red scale). The calibrated parameters for the Heston model on SPX 2019-11-08 are: $\hat{\kappa} = 0.682, \hat{\theta}^{(\mathcal{H})} = 0.124, \hat{\sigma} = 0.902, \hat{\rho} = -0.74$

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. As we earlier considered the calibrated volatility difference plot, see Figure 7, we shall also analyse the the same relationship here.

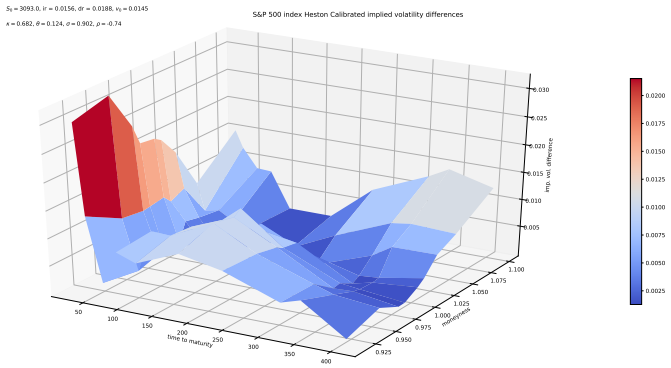


Figure 10: Difference between calibrated surface and market surface, i.e. visualization of distance function over surface in Equation (43), from model presented in Figure 9.

Evidently, the pricing model inefficiency for negative jump risk is correct as the model seems to be less able to calibrate short term ITM options than short term OTM options, see Figure 10. However, it should be noted that deeper in the money *call* options generally have higher implied volatilities as such options are more exposed to negative jump risk, and thus it is natural that the error is larger because of the fact the volatility is higher. It should however be noted that the maximal error is only 2%. Furthermore, there seems to be a slight stabilisation trend with respect to the term structure of the surface, for example the 406 day calibration error is flatter with respect to the strike space than shorter term options and is supported by the fact that the shorter term calibration error is generally larger than their longer term equivalents. However, we do not believe that these errors are largely systematic and thus one cannot draw to large conclusions with respect to these errors. Therefore, we state that we have accomplished the the partial objective of calibration through utilization of neural networks. The results are also largely in line with the findings of Horvath et al. (2019).

However, before considering the numerical results of the hedging method, we have to consider the *integration* of the deep calibration method into deep hedging. As discussed in Section 3.3, as the hedging algorithm is dependent on realizations of the spot process and the realization of sample paths of the model is dependent on which simulation scheme is used. If the feller condition $2\kappa\theta^{\mathcal{H}} \leq \sigma^2$ is violated, then there will exist a non-zero probability of sampling negative values of the variance process and the cumulative distribution function (CDF) of the Monte-Carlo option price will diverge from the semi-analytical price CDF, according to Bégin et al. (2015). However, as discussed in Section 3.3, the performance of simulation methods, in light of the feller condition, differs severely. The more simple Euler-type schemes, such as the Euler-Maruyama scheme in Equation (20), are less able to deal with violations of the feller condition since a larger frequency of observations passes through the "control functions". The exact simulation scheme is unaffected by this violation, however, its practical applicability is somewhat limited by its computational inefficiency and complexity. We will however implement the so called *moment-matching* scheme, where the spot process is a simple Euler-discretization, however the first two moments of the variance process is matched to that of a log-normal distribution, as in Equation (22). When the feller condition is not taken into account, the *unconstrained* calibration in Equation (43) returns a parameter combination that violates this condition, as such one would have to resort the exact simulation algorithm. This would however severely impair training time of the deep hedging algorithm, as the simulation time would be severely impaired. However, if one precedes with the *Feller-constrained* calibration in Equation (45), then the moment-matching scheme could be utilized instead, which would mean a significant improvement in training time for hedging model.

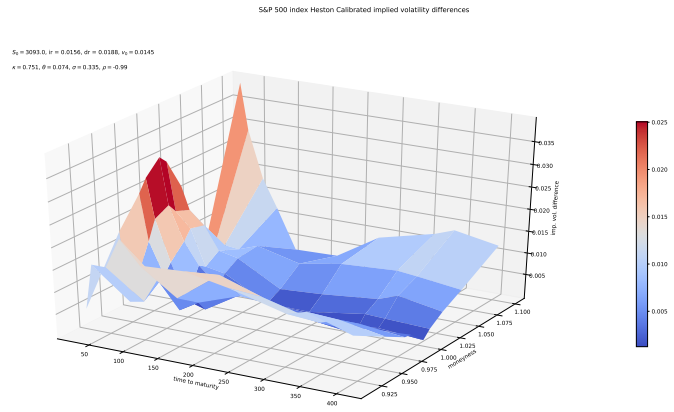


Figure 11: Difference between Feller-constrained calibrated surface, see Equation (45), and market surface. Order of magnitude larger, Final value of the differential evolution algorithm: 0.0145 compared to 0.008 for model presented in Figure 9.

As one can see in Figure 11, the feller constrained parameter space is less capable than the earlier calibrated parameter combination to replicate the market IV surface. Hence, one has to make a choice with respect to the integration, utilize more complicated, and potentially slower discretization schemes *or* utilize the constrained calibrated parameter combination such that one can obtain unbiased trajectories.

8.2 Hedging Method

The deep hedging framework, as introduced in Section 7, utilizes a more complex version of the feed forward network, called *semi-recurrent* neural networks. A semi-recurrent neural network architecture can be seen as a sequence of deep feed-forward networks, where each output serves as input in the next time step, and the current step depend on the previous outputs. Thus, one can visualize the recurrent neural network as a series of feed forward networks over time, t_i to t_n. At each t_i, the input layer consists of the *simulated* underlying asset price S_t for each $t = (t_i, \ldots, t_n)$, 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. These input parameters are then fed forward through a neural network with 2 hidden layers and 30 neurons each. See Figure 12 for architecture visualization. However, it should be noted that the deep hedging methodology easily extends to full-scale recurrent neural networks (RNN), where the recurrence is inherent in each cell, as described in Boden (2002). A RNN utilizes a "internal state" in which all cells saves a lagged output. This makes the network "remember" earlier states and data. It is widely used in language processing and other areas when using sequential data, where the order matters. But, when introducing this internal state, more parameters will have to be used, hence the processing is slower.

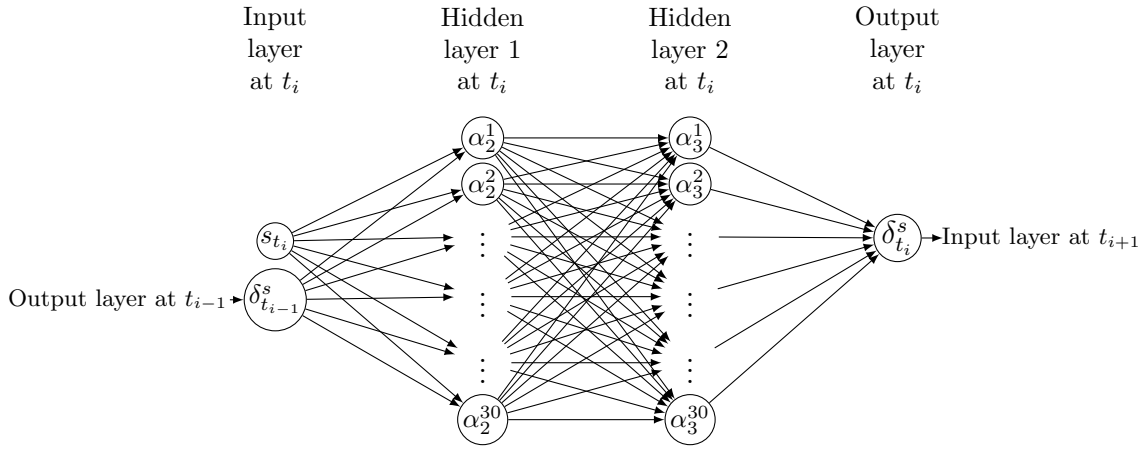


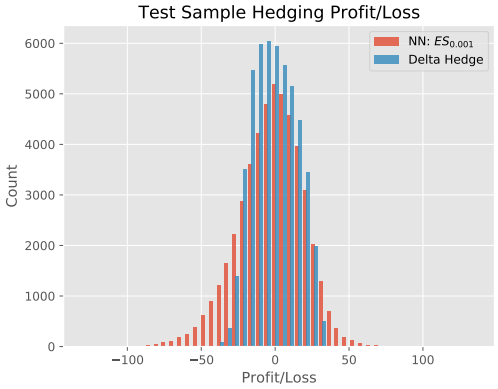
Figure 12: Deep hedging semi-RNN. Here one can clearly see the recurrent nature of the network as the hedging weight $\delta^S_{t_{i-1}}$ affects the hedging weights in the output layer. If we would have taken volatility related risk into account, there would have to be one more input neuron containing $\delta^V_{t_{i-1}}$ and output neuron $\delta^V_{t_i}$.

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, see Section 7 for details regarding the optimization procedure. We shall consider optimality under *expected shortfall (ES)* and the *entropic risk measure*, with varying risk aversion parameters α and λ respectively, see Equations (34) and (32). We shall also consider optimality under a non-convex measure, namely the *quadratic criterion*, see Equation (36).

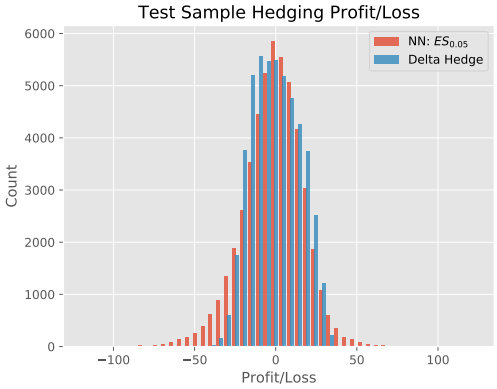
In section 7, we have shown that optimizing parameters for a neural network, when the loss function is a monetary risk measure of the hedging PnL, is equivalent to approximating unconstrained optimal hedging strategies under the measure, see Equation (49). For minimization by Adam optimizer of the entropic measure, see Equation (53), 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. Therefore, this section will be more descriptive than what would otherwise be warranted.

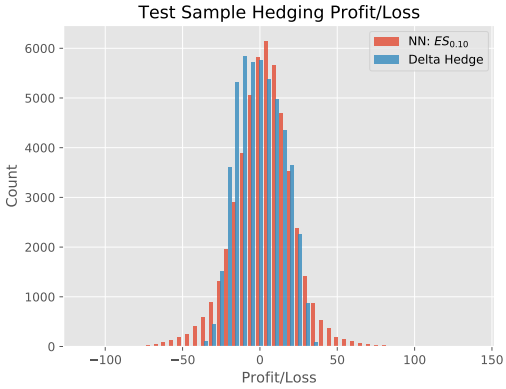
We shall now proceed by presenting the results of the deep hedging algorithm. All of the following hedging strategies will be evaluated at the spot value $S_0 = 3093.08$, and option parameters $T = 31/365$, $K = 3100$ as it represents the contract SPXW191211C03100, for the calibrated SPX Heston model. Firstly, we simulate trajectories of S_t, by the moment-matching scheme, which are then used to train a neural network strategy $\delta^\theta \in \delta^u$ as a minimizer to (31), where Δ is replaced by the set of unconstrained strategies. This numerical approximation of a optimal hedging strategy under some risk measure is then used to produce PnL-distributions of the hedging strategy as in Equation (46). These strategies are then compared to the PnL-distribution of a simple delta hedge, evaluated over the same trajectories.



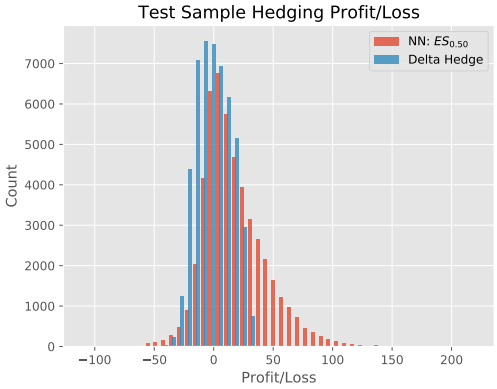
(a) Expected shortfall PnL when $\alpha = 0.01$.



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



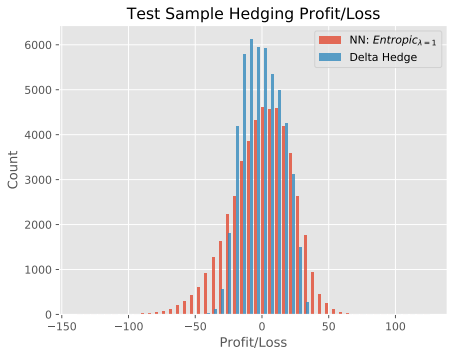
(c) Expected shortfall PnL when $\alpha = 0.10$.



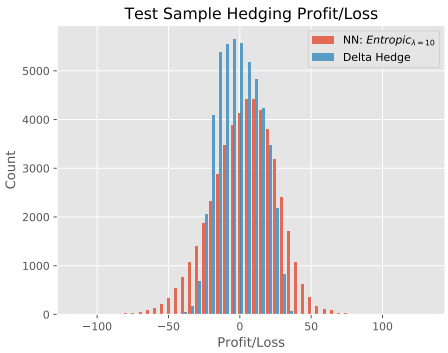
(d) Expected shortfall PnL when $\alpha = 0.50$.

Figure 13: PnL distributions for expected shortfall with different risk aversion parameters. Increase in risk aversion leads to a higher mean hedge PnL and thus increase the probability of earning money since the lowest acceptable price of Z increases.

In Figure 13, one can see that a increase in risk aversion, shifts the distribution to the right, i.e. would imply higher expected profit, as the PnL distributions in Figure 13 imply that the agent can charge the indifference price, i.e. $p_0^\theta = \pi^M(-Z) - \pi^M(0)$ and not fair market prices of $-Z$, which is also noted by Buehler et al. (2019). This is due to the fact that as α goes to 0, the agent approaches risk neutrality, hence the price converges to the risk neutral price of $-Z$ in (17). This means that, under real market conditions, the above PnL distributions would not be attainable. We shall not expand upon this concept in our thesis, however, researching differences in PnL distributions for different risk measures when only the fair market price is traded, is a interesting item for future research.



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



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

Figure 14: Histograms over hedging PnL for entropic measures. Seemingly no large difference of strategies in terms of their respective shapes. However, variance is larger for both cases relative to the delta hedge, which holds true for all considered approximations.

We now consider approximations of optimal hedging strategies under the quadratic criterion. This measure is, as earlier noted, not a convex risk measure, however, it is well known that optimal hedging strategies under the quadratic criterion 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 for example Föllmer & Leukert (2000). See Figure 15 for PnL-histogram of approximation of variance optimal hedge.

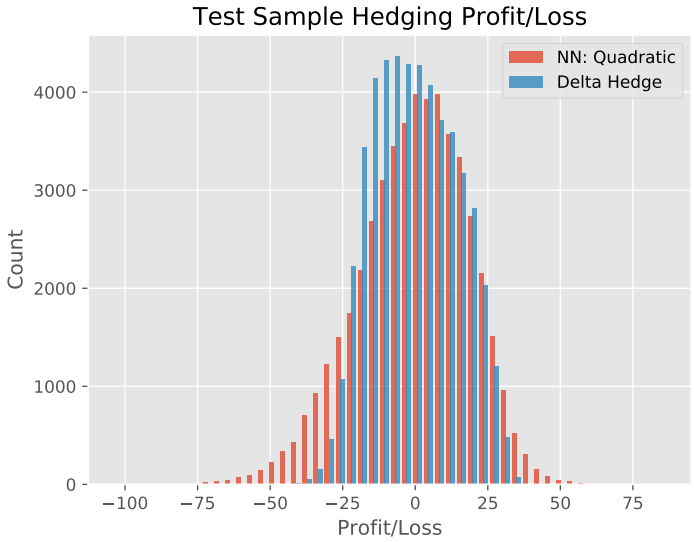


Figure 15: Hedging PnL for Quadratic criterion. One can see that the empirical distribution of the neural network strategy, seems to larger sample variance than the delta hedge.

Conversely, the variance of the close-to-variance-optimal hedging strategy has is larger than

the variance of the delta hedging strategy. As we have now visualized the PnL distributions of all considered risk measures, we now consider descriptive statistics for the hedging strategies, see Table 3.

Model type	p_0^θ	Min, Max	Mean	Variance	Skewness	Kurtosis
$ES_{0.01}$	39.55	$(-130.374,\ 135.67)$	-2.379	484.94	-0.365	0.85
$ES_{0.05}$	41.06	$(-119.22,\ 123.08)$	-1.22	351.48	-0.25	1.64
$ES_{0.10}$	42.7	$(-114.72,\ 139.45)$	2.79	379.21	-0.029	2.152
$ES_{0.50}$	58.79	$(-110.51,\ 220.07)$	17.616	$714,908$	0.957	2.186
Quadratic	40.89	$(-103.626,\ 86.051)$	-0.336	364.819	-0.38	0.292
$\text{Entropic}_{\lambda=1}$	41.02	$(-139.022,\ 126.094)$	0.3041	521.34	-0.45	0.468
$\text{Entropic}_{\lambda=10}$	43.04	$(-116.051,\ 131.920)$	3.763	511.07	-0.187	0.5209
Delta Hedge	40.82	$(-50.158,\ 37.386)$	-0.215	210.493	0.088	-0.804

Table 3: Descriptive statistics for PnL of hedging strategies, given by Equation (46), over test sample with 50000 trajectories. Notice the mean hedging PnL is very dependent on risk aversion 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 (17). The resulting fair market price is 40.82.

From Table 3 one sees the numerical properties for neural network approximations of optimal hedging strategies, *across* risk measures, one can see that the PnL-distributions are quite similar, when one exclude parameterization of relatively high risk aversion for expected shortfall. The largest difference seems to be the max and min of their respective hedging PnL, however, as noted earlier, this is also dependent on the level of risk aversion. The smallest maximum loss of all neural network hedging strategies is the quadratic criterion. The risk measure that seemingly best approximate the fair market price is the quadratic criterion, see Table 3, however, these are surely negligible differences.

In Figure 13 we see the realization of the hedging strategy for individual sample paths, for the collection of expected shortfall measures. Recall the definition of expected shortfall in terms of hedging, namely the expectation of the α-quantile for the empirical return distribution of the hedging strategy. It is also called conditional value at risk the proceeding definition is the conditional expectation given that VaR_α has been exceeded. Hence, when one seeks optimality of hedging strategies under ES_α measures, then larger alpha *should* lead to more risk averse trading in S, in light of the position in $-Z$.

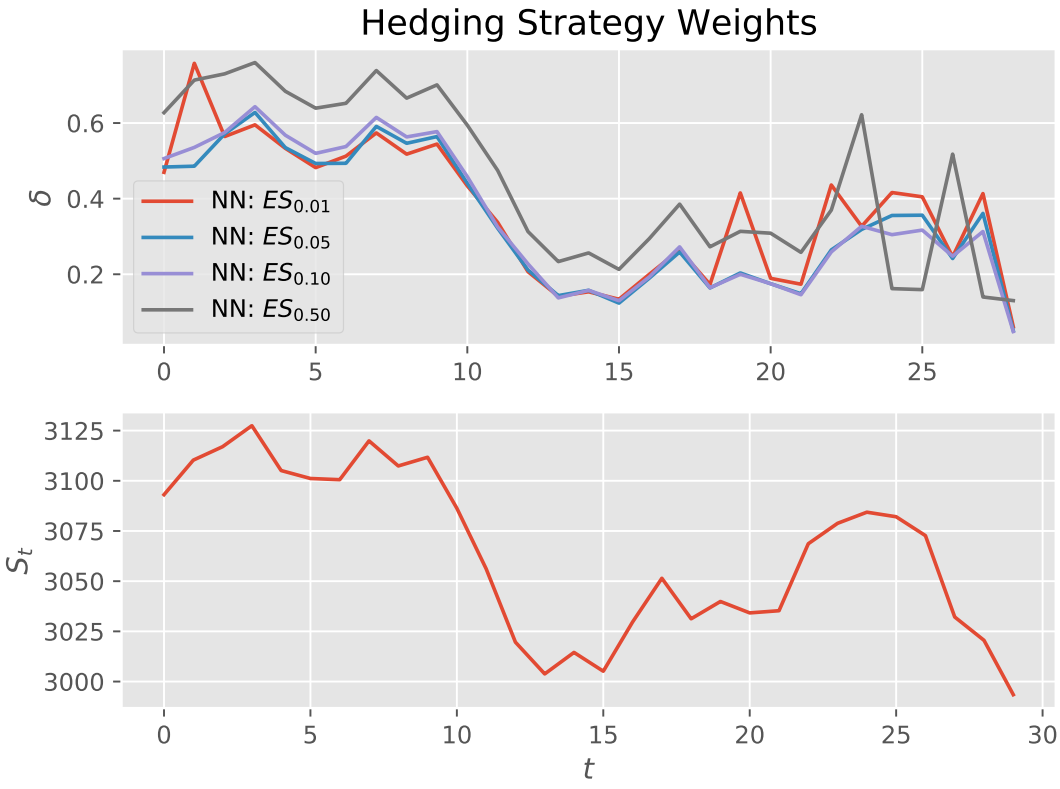


Figure 16: Comparison of ES hedging strategies over one realisation of S. Close-to-optimal hedging weights have similar shape of sample paths, however, initial hedging weights differ, since p_0^θ differs.

As one can see in Figure 16, the expected shortfall strategies are rather similar, especially optimality under $ES_{0.05}$ and $ES_{0.10}$, divergence is also dependent on time since the strategies seemingly diverge in larger magnitude for larger t.

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 (33) and α in Equation (34) are not comparable. However, one can see substantial differences in terms of their PnL-distributions.

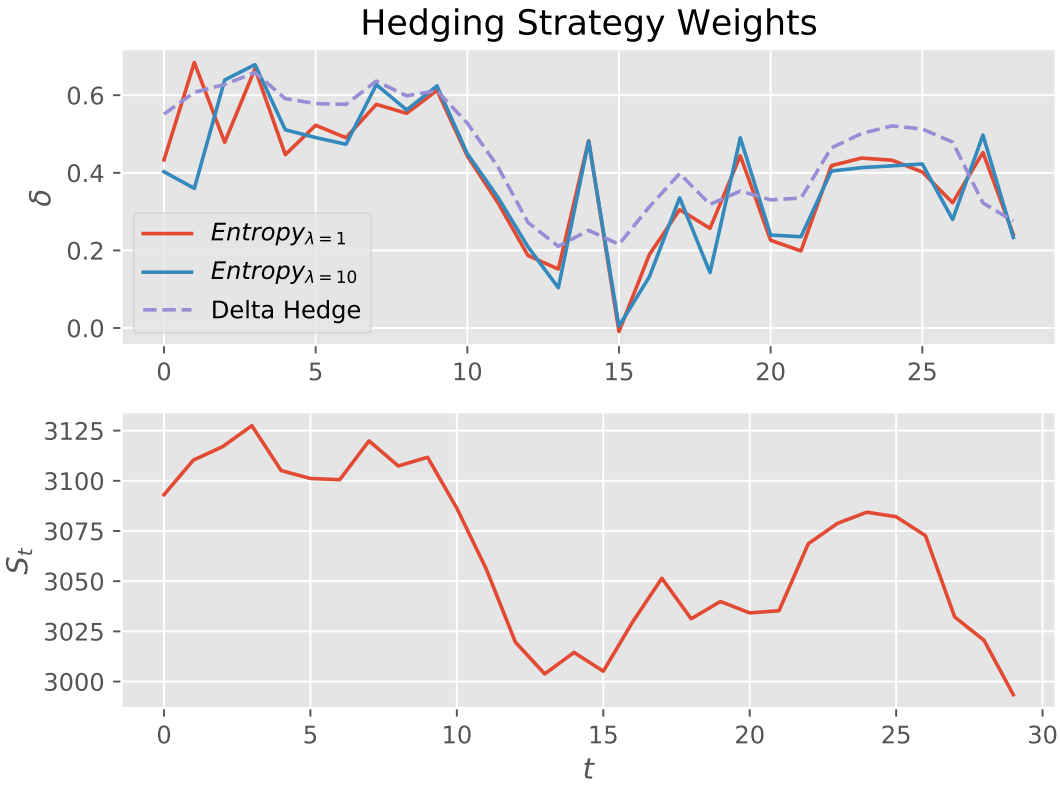


Figure 17: Realization of entropic hedging strategies, compared to delta strategy. Entropic strategies seem more volatile.

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 insufficient sample size.

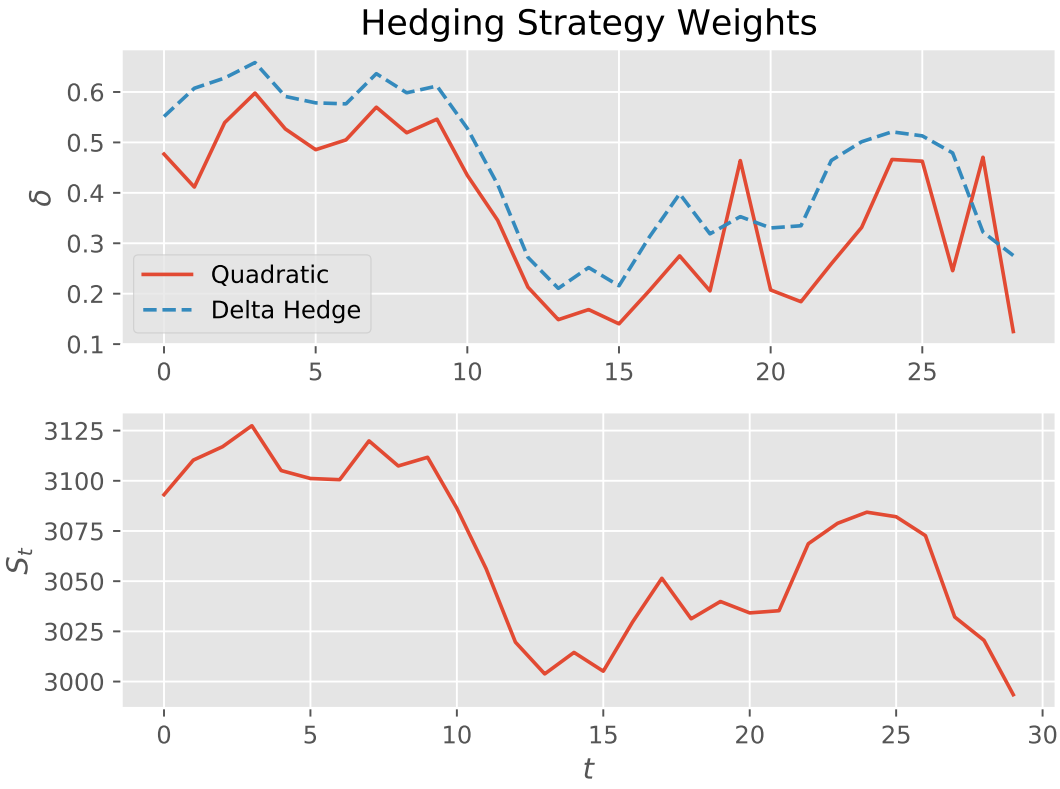


Figure 18: Realization of Quadratic hedge approximation compared to delta hedging strategy. The resulting approximation is more volatile than the delta hedge. However, this is seemingly a property for all neural network strategies. Larger sample size might be needed.

If we were to choose a risk measure, in light of these various visualizations and statistics, see Figures 13 to 18 and Table 3, and if we were to consider the flexibility of the measures, we would choose between either the *quadratic criterion* or *expected shortfall*. The quadratic criterion has superior performance in terms of learning the deep hedging price, however from Table 3, one can see that it seems to *under-charge* the fair market price, even though by a small margin. In contrast to the quadratic criterion, expected shortfall has a more flexible structure which allows agents to specify their risk preference levels and optimality is dependent on individual risk preference. Furthermore, ES_α does not seem to systematically under or over charge, as it varies with risk preference. However, its numerical performance is seemingly worse than the quadratic criterion. However it should be noted that ES prediction is heavily dependent on VaR prediction and is thus very sensitive to tail miss specifications.

We now illustrate the impact of *proportional transaction costs*, as presented in Equation (47), on the PnL distribution for some expected shortfall measure. We have chosen to illustrate this with $\alpha = 0.05$ since this entails close to risk neutrality and thus the divergence of the deep hedging price, and thus the non-centrality of the PnL distribution is mostly due to the impact of transaction costs, see Figure 19.

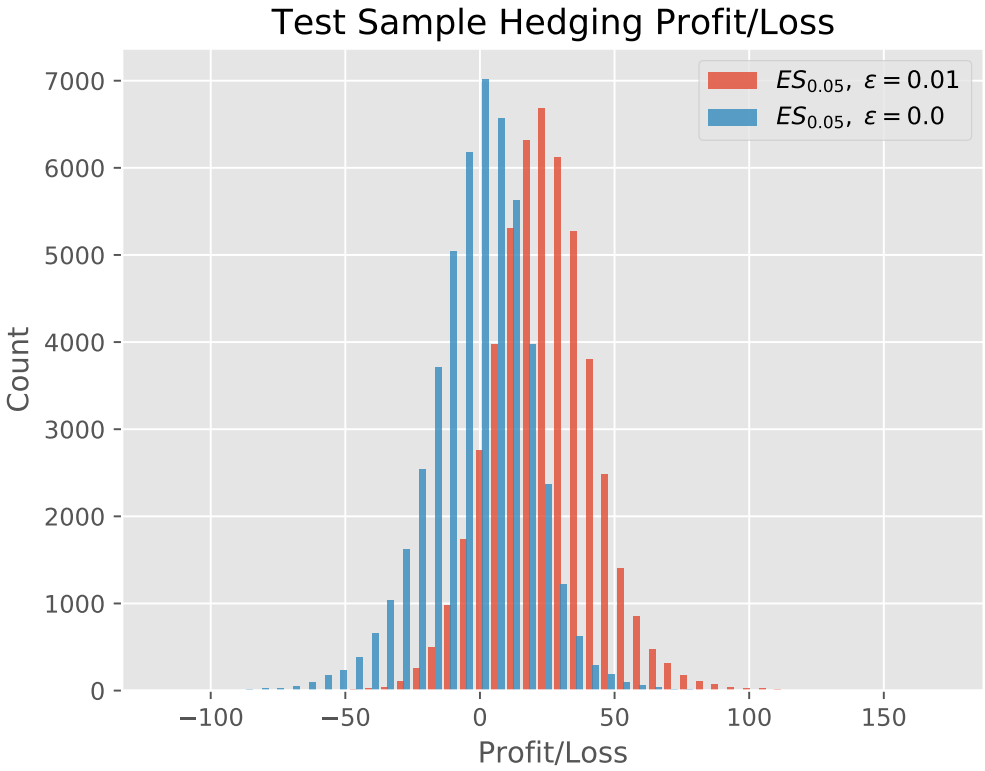


Figure 19: PnL distribution for $ES_{0.05}$ with and without proportional transaction costs when $\varepsilon = 0.01$,i.e cost of 1 % trade rebalancing, see Equation (47).

The deep hedging price of the $ES_{0.05,\varepsilon=0.01}$, $p_0^{\theta,\varepsilon} = 52.22$ which can be contrasted to the risk neutral price of 40.82 and the $ES_{0.05}$ price of 41.06. Hence, as one might expect, transaction costs will shift the smallest acceptable price higher since there is a implicit cost of trading. As financial institutions seek to transfer their cost of servicing client *onto* the counter-party, their lowest acceptable price, in light of their risk preference, will be higher than if there were no such cost. Next, Figure 20 shows the impact of transaction costs on a per trajectory basis.

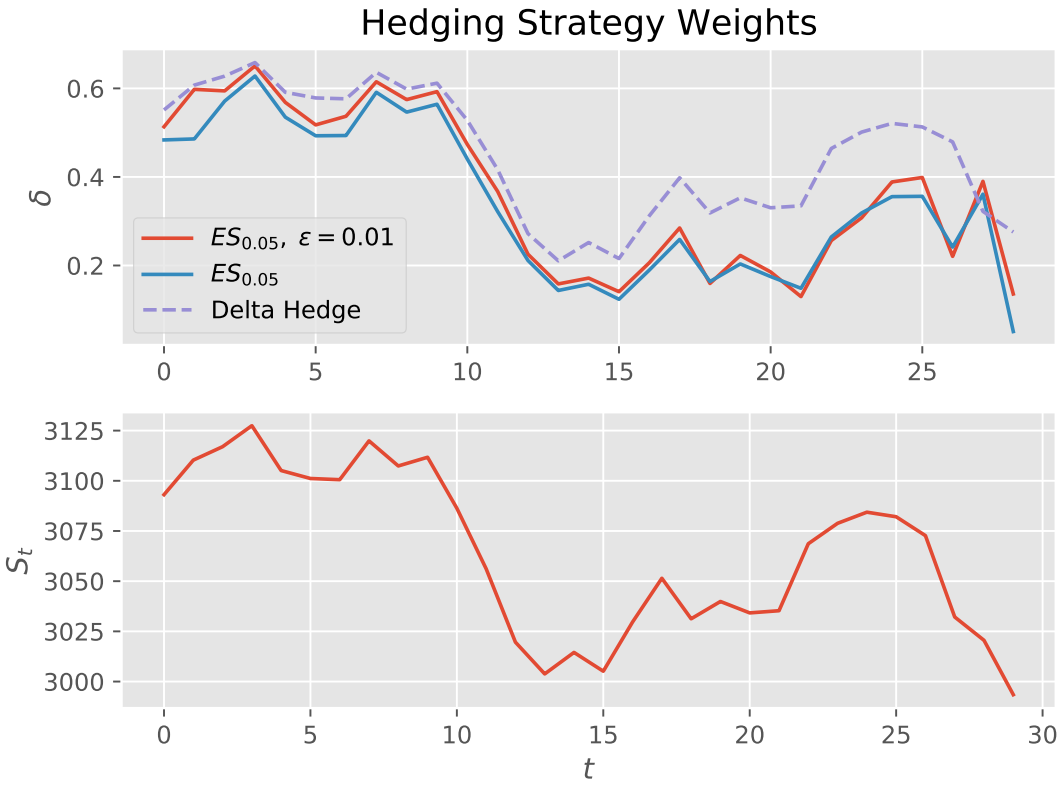


Figure 20: Hedging weights on per trajectory basis. Close-to-optimal hedging weights have small impact, although not constant.

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 researched the application of artificial neural networks for calibration and hedging of stochastic volatility. The purpose of this thesis, introduced in Section 1, 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 build a coherent front office model, 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 choices that must be made. 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 (12) and (13). As such, our formulation of the hedging method still has some important model dependencies, which is, to our knowledge, yet to be solved and could be interesting subjects for future research. If one could utilize the universal approximating properties of ANN's to sample from the calibrated distribution of a discrete version of the stochastic volatility model, then our current formulation would be truly model independent.

The applicability of deep calibration and hedging is very large 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 arbitrarily well approximate the pricing function and optimal hedging strategies respectively. The hedging method does also allow practitioners to comply with regulation with respect to capital requirements as the loss function for the semi-recurrent neural network could be specified in terms of monetary risk measures that the bank is need to be in compliance with. This is also an area that could be interesting for future research, namely the eventual regulatory implication of applying numerically optimal hedging approximations in terms of capital requirements when optimality is determined by the same risk measures that contribute to the capital requirement assessment. 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, the 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. However, if one could simulate the feature sets from some model, sparse data sets could be populated by model interpolations and thus solve this problem. 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.

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, nonetheless, they are more practical since they allow for the inclusion of market frictions such as transaction costs.

References

Acerbi, C. & Tasche, D. (2002), 'On the coherence of expected shortfall', *Journal of Banking Finance* **26**, 1487–1503.
URL: *https://doi.org/10.1016/S0378-4266(02)00283-2*

Andersen, L. B. G. (2007), 'Efficient simulation of the heston stochastic volatility model'.
URL: *http://dx.doi.org/10.2139/ssrn.946405*

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

Bates, D. S. (1993), 'Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options', *The Review of Financial Studies* **9**, 69–107.
URL: *https://doi.org/10.1093/rfs/9.1.69*

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

Black, F. & Scholes, M. (1973), 'The pricing of options and corporate liabilities', *Journal of political economy* **81**, 637–654.
URL: *http://dx.doi.org/10.1086/260062*

Boden, M. (2002), 'A guide to recurrent neural networks and backpropagation', *In the Dallas Project, Technical Report T2002:03* .

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.
URL: *http://leon.bottou.org/papers/bottou-tricks-2012*

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.
URL: *https://doi.org/10.1080/14697688.2019.1571683*

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.
URL: *https://doi.org/10.1515/mcma-2015-0105*

Canina, L. & Figlewski, S. (1993), 'The informational content of implied volatility', *The Review of Financial Studies* **6**, 659–681.
URL: *https://doi.org/10.1093/rfs/5.3.659*

Cont, R. & Fonseca, J. D. (2001), 'Dynamics of implied volatility surfaces', *Quantitative finance* **2**, 45–60.
URL: *https://doi.org/10.1088/1469-7688/2/1/304*

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.
URL: *https://arxiv.org/pdf/1502.02963.pdf*

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.
URL: *https://doi.org/10.1007/s007800050008*

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.

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.
URL: *https://doi.org/10.1016/0893-6080(91)90009-T*

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

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.
URL: *https://www.cs.cmu.edu/ ./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf*

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.
URL: *http://neuralnetworksanddeeplearning.com*

Pan, J. (2001), 'The jump-risk premia implicit in options: Evidence from an integrated time-series study', *Journal of Financial Economics* **63**, 3–50.
URL: *https://doi.org/10.1016/S0304-405X(01)00088-5*

Protter, P. E. (2005), *Stochastic differential equations*, Springer, Berlin, Heidelberg, pp. 249–361.
URL: *https://doi.org/10.1007/978-3-662-10061-5*$_6$

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.
URL: *https://doi.org/10.1109/TEVC.2008.927706*

Rosenblatt, F. (1958), 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review* **65**(6), 386–408.
URL: *https://doi.org/10.1037/h0042519*

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* .
URL: *https://arxiv.org/pdf/1609.04747.pdf*

Schweizer, M. (1995), 'Variance-optimal hedging in discrete time', *Mathematics of Operations Research* **20**, 1–32.
URL: *https://doi.org/10.1287/moor.20.1.1*

Stefanica, D. & Radoicic, R. (2017), 'An explicit implied volatility formula', *International Journal of Theoretical Applied Finance* **20**.
URL: *http://dx.doi.org/10.2139/ssrn.2908494*

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