UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

University of Cape Town

Master's Dissertation

Deep Hedging in Incomplete Markets

Author:
Jake Stangroom

Supervisor:
Mr. Melusi Mavuso

A dissertation submitted in fulfilment of the requirements for the degree of Master of Science

in the

Department of Statistical Sciences

November 28, 2023

Declaration of Authorship

I, Jake STANGROOM, declare that this dissertation titled, "Deep Hedging in Incomplete Markets" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Signed by candidate

Date: 01/12/2023

"*All models are wrong but some are useful.*"

George Box

Abstract

This dissertation presents an extensive analysis of the neural network approximation of mean-variance hedging with a comparison between the current neural network approaches and the theoretical solutions. These theoretical solutions provide a simulation-based performance benchmark for this comparison. Furthermore, this dissertation implements a financial market generator which allows for a realistic performance analysis based on both real and pseudo-real data; whereby, deep hedging is shown to offer highly competitive industry performance. Finally, the dissertation shows that deep hedging is effective for other quadratic criterion such as those similar to local risk-minimisation techniques.

Acknowledgements

Firstly, I would like to thank my supervisor, Mr. Melusi Mavuso, for the invaluable guidance throughout the dissertation. His knowledge and solutions enabled the dissertation to develop into its completed form.

Secondly, I would like to thank A/Prof Peter Ouwehand for discussions surrounding the dissertation. I look forward to our collaboration in the future.

I would like to thank the National Research Foundation and the Centre of Excellence: Mathematical and Statistical Sciences for both funding my degree at the University of Cape Town and providing learning experiences valuable to science professionals. Kindly note that the opinions expressed in this dissertation are those of the author and are not necessarily those of the National Research Foundation.

Dedicated to my family and friends for their integral contributions from a personal perspective. My mom and dad have always supported me through my studies and encouraged my choice to pursue higher degrees. Furthermore, my partner[1] has been the foundation to the high level of mental wellness I required and inspired me to pursue my academic goals. For this, I am eternally grateful.

[1] With our feline family

Contents

List of Figures

List of Tables

List of Abbreviations

EMM	**E**quivalent **M**artingale **M**easure
ELMM	**E**quivalent **L**ocal **M**artingale **M**easure
NFLVR	**N**o **F**ree **L**unch with **V**anishing **R**isk
FTAP	**F**undamental **T**heorem of **A**sset **P**ricing
SDE	**S**tochastic **D**ifferential **E**quation
MVH	**M**ean **V**ariance **H**edging
ESRE	**E**xpected **S**quare **R**eplication **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
DRNN	**D**eep **R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
OCE	**O**ptimised **C**ertainty **E**quivalents
VAE	**V**ariational **A**uto-**E**ncoder
CVAE	**C**onditional **V**ariational **A**uto-**E**ncoder

Chapter 1

Introduction

Machine learning techniques in mathematical finance have promoted a plethora of research. Throughout this dissertation we analyse the process known as *deep hedging*. Deep hedging offers a reinforcement learning solution to hedging claims in incomplete markets by approximation of the optimal strategy. By considering the quadratic criterion, our analysis utilises dynamic programming, martingale theory and industry benchmarks.

1.1 Literature Review

Black and Scholes, 1973 and Merton, 1973 introduced a pioneering approach to the theory of mathematical finance and derivative pricing. The approach utilised partial differential equations (PDEs) in order to establish replicating portfolios for contingent claims. Within no arbitrage theory the price of a replicating strategy defines the price of such derivatives from the law of one price. Consequently, the model provides a hedging approach for the complete model. Contrary to the above PDE-based approach, Harrison and Kreps, 1979 derived the Black-Scholes model from probability theory and martingales. In addition, Harrison and Pliska, 1981 established a link between market completeness and a class of martingales which follows a specific martingale representation property. Maintaining the completeness of the financial models, such as the Black-Scholes model, is not realistic in application. Therefore, research into the hedging of contingent claims in incomplete markets has received attention since introduction.

Hedging in incomplete markets requires a trader to associate a utility towards the profit and loss obtained from the inability to perfectly replicate a desired contingent claim. They must then aim to maximise her expected utility, yielding the most optimal intrinsic value and hedging strategy for her preferences. Beyond this, the exact details continue to be extensively researched. On one extreme, a possible incomplete market hedging solution is super-replication. Here, a trader constructs their hedging strategy such that the payoff of the contingent claim is super-hedged under all possible scenarios, or at least almost-surely. Consequently, when a trader charges a super-replication price for a contingent claim they can entirely eliminate the associated risk of the claim's

short position. The application of super-replication remains difficult in certain circumstances. More importantly, the super-replication price is generally too high from a competitive pricing perspective. For earlier research on super-replication techniques, see El Karoui and Quenez, 1995, Kramkov, 1996 and Föllmer and Kabanov, 1997.

Another, often more applicable approach, originated in Hodges and Neuberger, 1989, follows the notion of certainty equivalents. In particular, indifference pricing incorporates a utility function for the trader and aims to find the certainty equivalent which represents the amount which the trader would be indifferent to entering the position. The certainty equivalent is the indifference price associated with the given utility function. The hedging strategy is found naturally as part of the optimisation process, representing the best possible hedging strategy with respect to the performance evaluated by the chosen utility criterion. The problem surrounding this technique is the choice of utility function. Exploration of constant relative risk aversion (CRRA) and hyperbolic absolute relative risk aversion (HARA) utility functions for incomplete market hedging has been conducted by Malamud, Trubowitz, and Wüthrich, 2013 and Duffie et al., 1997 respectively. The study of convex risk measures was introduced by Föllmer, Schweizer, et al., 1990 to generalise the utility functions which resulted in the desired indifference pricing properties. A comparison of such convex risk measures was conducted by Ilhan, Jonsson, and Sircar, 2009 with particular emphasis on exotic options. A more recent study by Hodoshima, Misawa, and Miyahara, 2018 compares the indifference pricing method to the upcoming mean-variance approach under a mixture of Normal distributions which leads into the next approach for the pricing and hedging problem in incomplete markets. Despite the extensive research into a variety of utility functions, a practical problem remains; a general trader does not explicitly define their utility curve for contingent claims.

The final major approach for the pricing and hedging of contingent claims utilises a quadratic criterion for the risk profile. The approach is analogous to choosing particular martingale measures for the pricing. This idea was first popularised by Föllmer and Sondermann, 1986 and Duffie and Richardson, 1991. The quadratic criterion benefits from both a mathematical and economical perspective. Economically, the criterion represents a risk averse trader who aims to balance both the accuracy of their hedge and minimisation of the associated variance. Unlike many of the utility functions used for indifference pricing, the quadratic criterion equally penalises both an under and over hedge. From the mathematical perspective, the criterion offers tractability in the derivation of the optimal solution and the ability to follow a martingale approach. The use of the quadratic criterion can be further subdivided into two distinct approaches: *local risk-minimisation* and *mean-variance hedging*. A succinct guide through these approaches is provided by Schweizer, 1999 whilst Heath, Platen, and Schweizer, 2001a provided a comparison between the two methods under the Heston model, a stochastic volatility model introduced by Heston, 1993. Under the local risk-minimisation framework, a trader decides to follow a non-self-financing trading strategy which endures an associated cost with the goal of

minimising the cost. This was first explored by Föllmer and Sondermann, 1986 where the price process is a martingale. Schweizer, 1988 and Schweizer, 1991 then extended the framework to the general semi-martingale case. This work established the Föllmer-Schweizer decomposition and the link to the so-called minimal martingale measure. The obvious disadvantage to this method is that the trader must incur unplanned cash-flows throughout the life of the hedge since the strategy is not self-financing.

On the other hand, the mean-variance hedging approach maintains the self-financing property of the strategy and aims to minimise the expected square replication error. Bouleau and Lamberton, 1989, Duffie and Richardson, 1991 and Schweizer, 1994 pioneered and established the mean-variance hedging method, showing that the problem is a projection of the claim in the $L^2(\mathbb{P})$ Hilbert space onto the space of attainable claims. The continuous-time mean-variance hedging solution relies on a variance-optimal martingale measure, with Rheinländer and Schweizer, 1997 providing a generalised solution. In the discrete-time case, Černỳ, 2004 explored a dynamic programming solution as an alternative to the martingale theory construction. Much like with the process of indifference pricing, the mean-variance hedging with dynamic programming is plausible through the reduction of the problem with the Hamilton-Bellman-Jacobi equation. Additionally, given the quadratic nature of measure, mathematical convenience is also achieved. In particular, it is shown that the dynamic programming approach results in the explicit characterisation of the martingale solution. Unfortunately, the dynamic programming approach can be unfeasible in application due to the potentially high computation time; however, it does serve as a natural step when exploring hedging strategies of discontinuous price processes. In response, machine learning techniques are explored to approximate optimality to provide an application friendly approach.

The early 1990s saw the initial significant contributions made for neural networks in the field of option pricing. Malliaris and Salchenberger, 1993 and Hutchinson, Lo, and Poggio, 1994 explored neural networks to estimate the closing price of options. The networks receive the same variables as the Black-Scholes model and are compared against the Black-Scholes model under the mean-square error for performance evaluation. The associated hedging strategy for the networks is computed through sensitivity analysis. Following this, there has been a plethora of research into the application of neural networks for both pricing and hedging options. Neural networks provide a method for derivative pricing which is non-parametric in terms of the underlying model. Ruf and Wang, 2020 provided an elaborate overview of this literature ranging from model outputs to feature sets. Here, we focus on a few recent papers which serve as direct influences to this research.

Contrary to analysing the sensitivity of pricing networks to obtain the hedging strategy, receiving the hedging strategy directly from the network was originally explored by Carverhill and Cheuk, 2003 and Sutcliffe and Chen, 2011. Here, the networks were trained to output desired option sensitivities, such as the Delta and Vega, in a non-parametric manner. Despite these earlier attempts, Buehler et al., 2019 provided the most influential contribution to the neural

network hedging space, forming the basis of this proposed research. Beuhler approaches the hedging and pricing problem simultaneously under the indifference pricing framework. As mentioned, this framework can be reduced to a dynamic programming problem where both the optimal hedging strategy and price can be obtained from the optimisation. Consequently, they explore deep neural networks for the reinforcement learning problem with a semi-recurrent network. Additionally, market friction and various convex risk functions are incorporated and compared against the Black-Scholes performance in a Heston model simulation. The benefits of directly computing the hedging ratio from the network is briefly discussed in Ruf and Wang, 2021. The main advantage is the resolution of the fact that market models may share very similar prices but drastically different hedging strategies, shown by Lyons, 1995.

An important recent implementation follows data generation. Data generation is a prevalent problem in mathematical finance. The training of pricing and hedging models for real-world application requires real-world data; however, only one sample path is ever available. This causes a significant problem for deep learning-based pricing and hedging models which require data for which to be trained. Traditionally, data has been simulated under chosen models, such as the Black-Scholes or Heston model frameworks; however, generative networks present a more modern approach. Financial market data generators are in their infancy. Kondratyev and Schwarz, 2019 provided a market generator with a restricted Boltzmann Machine whilst Kondratyev, Schwarz, and Horvath, 2020 expanded on flaws associated with the restricted Boltzmann Machine. Henry-Labordere, 2019 and Wiese et al., 2020 explored GAN based market generators, whilst Buehler et al., 2020 provided the first VAE based market generator.

1.2 Problem Formulation

Consider a trader presented with an unattainable contingent claim, H, which they are required to hedge. Without a perfect hedge available, the trader is still expected to act optimally by minimising their associated risk. Whilst a few methods exist for hedging in incomplete markets, this dissertation focuses on hedging based on quadratic criterion with particular emphasis on mean-variance hedging. As such, the trader aims to minimise

$$\min_{\varphi} \mathbb{E}\left[(H - V_T(\varphi))^2\right], \tag{1.1}$$

where $V_T(\varphi)$ represents the portfolio value associated with the trading strategy φ at time T. The optimisation methods follow dynamic programming, martingale theory, and neural network approximation approaches. This dissertation analyses the neural network approximation technique.

The optimisation process of pricing and hedging in incomplete markets can be reformulated as a reinforcement learning problem since the problem originates from a dynamic programming problem. This reinforcement learning framework has become known as *deep hedging*. Throughout this dissertation, deep hedging is shown to approximate the optimal solution arbitrarily well in both theory

and application. This is first shown under simulation with optimal benchmarks. Here, underlying asset prices are simulated under the multinomial, Black-Scholes , Heston models and their optimal benchmarks are used to analyse the capabilities of deep hedging. These benchmarks show that deep hedging can approximate the optimal solution without any significant difference.

Under more realistic settings, the deep hedging approach can outperform industry benchmarks and deep hedging offers highly competitive industry performance. Firstly, we utilise financial market generators to create asset sample paths. The deep hedging framework is able to outperform a Black-Scholes hedging strategy under these $\mathbb{P}$ sample paths. Secondly, the quadratic criterion is changed to the quadratic daily profit and loss which relies on the option price process. Such a criterion offers an important insight for traders operating on a mark-to-market basis. Here, deep hedging is able to provide competitive performance against the Heston model, even when the option prices are generated under the Heston model. Whilst restricted to real samples, deep hedging models can suffer from overfitting; however, the training process is shown to be improved by incorporating financial market generation.

1.3 Outline

The dissertation contains five chapters. This introduction represents the first. The second chapter, 2, provides preliminaries required for the dissertation. This includes an introduction to mathematical finance, a brief summary of the neural networks presented in the dissertation and a succinct introduction to path signatures required for the data generation techniques.

Chapter 3 establishes the theoretical solutions to pricing and hedging in incomplete markets. These begin with indifference pricing before solving the quadratic hedging problem with both martingale theory and dynamic programming. In addition, a theoretical motivation for deep hedging is provided.

Chapter 4 presents our performance analysis of deep hedging. The results begin with simulation benchmarks to evaluate the approximation accuracy of deep hedging when the optimal or close-to-optimal solution is known. Following these benchmarks, the deep hedging approach is compared to industry performance on both pseudo-real and real paths.

Chapter 5 offers a conclusion to the analysis presented whilst providing pitfalls and areas of extension for the dissertation.

Chapter 2

Preliminaries

In this chapter, we discuss the foundations required to discuss the outlined problem of deep hedging from a theoretical perspective. For this, we begin with an introduction to mathematical finance, then proceed to an introduction to the neural network architectures used later on in the dissertation. Finally, we end the chapter with an introduction to path signatures, an essential component required for the data generation process covered in 4.3.

2.1 Mathematical Finance

The field of mathematical finance makes rigorous use of the theory of stochastic processes and probability theory. In this section, we cover the essential theory required for mathematical finance, and to discuss the theory of financial hedging in the required detail. We begin with an introduction to trading in continuous time, leading into a discussion of two important continuous-time market models.

2.1.1 Trading in Continuous Time

An introduction to trading in continuous time is generally a formal extension of the discrete-time trading theory. For readers who desire a rigorous development of the theory or have minimal prior knowledge of the field, we recommend Föllmer and Schied, 2011 for discrete theory. The focus of this dissertation is incomplete markets and the discretization of continuous-time models; hence, development of the continuous-time theory is necessary for the upcoming discussions. Consequently, this section guides the reader through an introduction to trading and introduces examples of complete market models. The essence of these models is the ability to perfectly replicate the payoff of a contingent claim by continuously re-balancing a self-financing portfolio. The construction cost of this replicating portfolio must then be the price of the contingent claim due to the law of one price. For a rigorous exploration of continuous-time mathematical finance, we refer the reader to Musiela and Rutkowski, 2010.

Introduction to Trading in Continuous Time

Throughout this introduction, we establish the definitions and notation required for future discussion. First, consider a continuous-time financial market which

consists of d *risky* assets and a single *risk-free* asset. Then, the $d+1$ asset prices form a stochastic process, which is denoted by

$$S := \left(S_t^0, S_t^1, \ldots, S_t^d\right). \tag{2.1}$$

Here, t represents the time, whilst S_t^i for $i = 0, \ldots, d$ denotes the i^{th} asset. It is common practice that the $i = 0$ asset represents the risk-free asset. If a constant interest rate is assumed, then the dynamics of the risk-free asset is

$$S_t^0 = S_0^0 e^{rt},$$

where r represents the risk-free rate of return over a unit period. Furthermore, assume without loss of generality that $S_0^0 = 1$.

The remainder of the d risky asset price processes are stochastic. Due to this natural separation, it can be beneficial to denote

$$\begin{aligned} B &:= S^0, \\ \underline{S} &:= \left(S^1, \ldots, S^d\right). \end{aligned}$$

The motivation for the notation B follows from the fact that the risk-free asset can be perceived as an investment in a riskless bank account. Finally, S is a $d+1$ dimensional cadlag semi-martingale, where we assume that each component is positive to avoid unnecessary mathematical complexities[1].

With the above notation, a financial market can be represented by the tuple of a filtered probability space and primary assets, denoted $((\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}), B, S)$. For the purposes of avoiding unnecessary technicalities in the mathematics, we assume $\mathcal{F}_0$ is trivial and restrict the market to a finite time interval, say $[0, T]$ for some future time T, without loss of generality in our case.

Now that we have a market tuple, let us make some simplifying assumptions in order to develop a theoretical basis for the trading. Assume that the following conditions exist:

1. Continuous trading.
2. Unrestricted borrowing and lending at the risk-free rate.
3. Frictionless market, which implies notions such as sufficient liquidity and the absence of transaction costs.

We will relax some of these assumptions once we have developed the theory to obtain more realistic market conditions.

Now let us introduce trading strategies and portfolios which combine these base market components. Begin by allowing for the natural assumption that we have the σ-algebra $\mathcal{F}_t \subseteq \mathcal{F}$ which represents the information available to the trader

[1]This assumption is realistic in most markets since do not expect negative prices for primary assets; however, the assumption can be relaxed given that the user incorporates concepts such as σ-martingales

at time t and that

$$\mathcal{F}_s \subseteq \mathcal{F}_t, \quad \forall s \leq t \leq T.$$

A *trading strategy* is a predictable process, φ_t, which is integrable with respect to S_t such that

$$\varphi_t = \left(\varphi_t^0, \varphi_t^1, \ldots, \varphi_t^d\right),$$

where φ_t^i represents the number of shares of the i^{th} asset held in the portfolio at time t. Hence, at time t, if a trader holds a portfolio with respect to the trading strategy φ, the value of the portfolio is calculated as

$$\begin{aligned} V_t(\varphi) &= \varphi_t \cdot S_t \\ &= \sum_{i=0}^{d} \varphi_t^i S_t^i. \end{aligned}$$

The process $V = V(\varphi) = \{V_t(\varphi) : t \in [0, T]\}$ is known as the *value process* of the portfolio. Following from the *law of one price*, the value of the portfolio at time t is the same as the construction cost of this portfolio at t.

Now, let us define the *cumulative gains process*, $G(\varphi) = \{G_t(\varphi) : t \in [0, T]\}$, for the trading strategy φ. This process captures the portfolio gains which have accumulated over the period $[0, t]$ directly from the trading strategy φ. Therefore,

$$\begin{aligned} G_0(\varphi) &:= 0, \\ G_t(\varphi) &:= \int_0^t \varphi_u \cdot dS_u, \quad \text{for } t > 0. \end{aligned}$$

Definition 2.1.1
A trading strategy is said to be self-financing if and only if

$$V_t(\varphi) = V_0(\varphi) + G_t(\varphi), \quad \forall t.$$

A self-financing strategy prevents both injections and withdrawals of investment capital, enforcing that the changes in portfolio value are the sole result of trading. For the remainder of the dissertation we assume that trading strategies are self-financing unless otherwise stated.

In the financial world, it is common practice to represent all of the market's asset prices relative to a single asset for comparative benefits. Whilst any asset can be chosen to be the basis asset, formally known as the *numeraire*, a common starting point is to use the risk-free asset. These prices are known as *discounted* asset prices and they allow us to easily handle the notion of *time value of money* by distinguishing between a single unit of currency throughout time. Mathematically, we denote the discounted asset price process, X, as follows:

$$X = \left(X^1, \ldots, X^d\right),$$

where

$$X^i := \frac{S^i}{S^0}, \quad \forall i = 0, 1, \ldots, d.$$

Here, S^0 is the *numeraire*, trivially resulting in $\frac{S^0}{S^0} = 1$. The associated discounted price system for the market is $(1, X) = (1, X^1, \ldots, X^d)$. In literature, $\bar{S}$ is traditionally also reserved for the discounted price system. Within the application of option pricing and hedging, it may become extremely beneficial to utilise other assets as the numeraire. Any asset can represent a numeraire given that the asset's price process is strictly positive. When this is the case, the resultant process is known as a numeraire-*deflated* price process rather than a discounted process. Throughout the remainder of the dissertation we will largely utilise the discounted process for mathematical convenience, supporting the following notions.

Under a self-financing trading strategy and discounted price process, we can redefine the trading strategy as

$$(\eta, \vartheta) := (\varphi^0, (\varphi^1_t, \ldots, \varphi^d_t)).$$

Under this notation, η represents the amount invested in the riskless asset, since the riskless asset has a discounted value of 1 for all t, and ϑ represents the strategy for the risky assets. Then, the discounted value and gains processes can be derived in relative terms as:

$$\begin{aligned} \bar{V}_t(\varphi) :&= (\eta, \vartheta)_t \cdot (1, X)_t \\ &= \eta_t + \sum_{i=1}^{d} \vartheta^i_t X^i_t, \\ \bar{G}_t(\varphi) :&= \int_0^t \vartheta \cdot dX_u. \end{aligned}$$

Firstly, note that the discounted gains process is only determined by the trading in the risky assets. This notion is consistent with our concern with the gains obtained from the risk taken by investing in a the risky assets in the portfolio. More importantly, we observe that, with the self-financing condition, the portfolio can be fully characterised by the initial value and the strategy for the risky assets, (ν_0, ϑ) where ν_0 denotes the initial value of the portfolio. This result can be shown as follows: let the riskless asset be the numeraire. Then

$$\begin{aligned} &\bar{V}_t(\varphi) = \bar{V}_0(\varphi) + \bar{G}_t(\varphi), \\ &\Rightarrow \varphi^0_t = \bar{V}_0(\varphi) + \sum_{i=1}^{d} \left[\int_0^t \varphi^i_u dX^i_u - \varphi^i_t X^i_t \right]. \end{aligned}$$

Consequently, it is possible to make any trading strategy on the risky assets self-financing by adjusting the strategy imposed on the riskless asset to satisfy the above equation. As a result, we will often try to utilise the discounted market where applicable for the mathematical convenience.

A final assumption we make before progressing is the inclusion of a finite credit line. Such a credit line is chosen to be any realistic value for the trader of concern, ensuring that the theory remains more realistic by preventing the value process becoming arbitrarily large and negative. The assumption is achieved through enforcing the following condition on the trading strategy.

Definition 2.1.2
A self financing trading strategy φ is admissible if there exists $a \in \mathbb{R}$ such that

$$G_t(\varphi) \geq a, \quad \forall t \geq 0.$$

We will extend our assumptions such that all trading strategies can be assumed to be admissible and self-financing unless otherwise stated.

Arbitrage Pricing Theory

An arbitrage opportunity is a fundamental concept in any area of finance. An arbitrage opportunity is a trading strategy which allows the trader to make a riskless profit. Explicitly, the trader can begin with no capital investment and have a positive probability of making a profit from the strategy whilst negating all risk of losing capital, at least almost surely.

Definition 2.1.3
A strategy φ is an arbitrage strategy or arbitrage opportunity if and only if

1. $V_0(\varphi) = 0$.
2. $V_T(\varphi) \geq 0, \ \ \mathbb{P} - a.s$
3. $\mathbb{P}(V_T(\varphi) > 0) > 0$.

This definition for an arbitrage opportunity also holds for any numeraire-deflated price process. It is clear that we want to avoid creating a market model which experiences arbitrage opportunities. A trader should not obtain excess returns without admitting the associated risk. We say that a market is *arbitrage-free* if there are no arbitrage opportunities in the market. Such an assumption to make is often fair since if an arbitrage opportunity exists in the market, the market is expected to aggressively exploit the opportunity which would result in its removal in a short time frame through supply and demand. Arbitrage Pricing theory explores the probability measures which ensure arbitrage-free market models.

Definition 2.1.4
Choose N to be a numeraire. Then a measure $\mathbb{Q}$ is called an equivalent (local) martingale measure (EMM) for the numeraire N if and only if

1. $\mathbb{Q} \approx \mathbb{P}$*; i.e.* $\mathbb{P}(A) = 0 \iff \mathbb{Q}(A) = 0, \quad \forall A \in \mathcal{F}$.
2. $\hat{S}_t = \left(\frac{S}{N}\right)_t$ *is a (local)* $\mathbb{Q}$*-martingale.*

If the numeraire process N is the riskless asset, then the EMM $\mathbb{Q}$ is known as a risk-neutral measure.

Suppose that $\mathbb{Q}$ is an EMM for some numeraire process N and denote the numeraire-deflated price process by $\hat{S}$. Then both $\hat{V}$ and $\hat{G}$ are also (local) $\mathbb{Q}$-martingales. This result follows since $\hat{G}$ is a stochastic integral with respect to $\hat{S}$ which is itself a (local) $\mathbb{Q}$-martingale. In addition, the equivalence condition for an EMM ensures that $\mathbb{P}$ and $\mathbb{Q}$ share the same arbitrage opportunities. Hence, an arbitrage opportunity remains an arbitrage when we change numeraire or EMM.

In order to express the Fundamental Theorem of Asset Pricing I (FTAP I) in the continuous-time framework, we must first cover one more concept.

Definition 2.1.5
A sequence of admissible and self-financing strategies (ν_0^k, ϑ^k) admits a free lunch with vanishing risk if

1. $\nu_0^k = 0$.
2. $V_T\left((\nu_0^k, \vartheta^k)\right) \geq -\frac{1}{k}, \ \ \forall \ k.$
3. $\exists \ \epsilon, \delta > 0$ *such that* $\mathbb{P}(V_T(\nu_0^k, \vartheta^k) > \delta) > \epsilon, \quad \forall k.$

Free lunch with vanishing risk strategies can be viewed as sequences of trading strategies which approximate arbitrage strategies. A market is said to adhere to the *no free lunch with vanishing risk* condition when no such sequence of strategies exist in the market. This is a strengthening of the arbitrage opportunity condition.

Theorem 1 (FTAP I)
For a financial market with non-negative assets, the following are equivalent:

1. *The market satisfies the NFLVR condition.*
2. *There exists an ELMM for X.*

In discrete-time theory, the absence of arbitrage opportunities is enough to ensure the existence of an EMM for X; however, the continuous-time framework requires the stronger NFLVR condition for the forward implication of theorem 1. Hence, we can obtain a NFLVR market, and thus an arbitrage-free market, through an EMM.

The Fundamental Theorem of Asset Pricing II (FTAP II) is our next objective. This is essential for the pricing and hedging process; but, we must develop the required concepts first.

Firstly, a *European contingent claim*, C, is a $\mathcal{F}_T$-measurable random variable in a financial market $((\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}), S)$. At the same time, let us denote the *discounted claim* as H, which is a contingent claim with respect to the discounted price process X.

Definition 2.1.6
We say a strategy (ν_0, ϑ) replicates C, or is a replicating strategy for C, if

$$V_T((\nu_0, \vartheta)) = C, \ \ \mathbb{P} - a.s.$$

If such a replicating strategy exits, we say that C is attainable or replicable.

Suppose that we have a contingent claim C which is attainable. Then we know the current (theoretical) price of this claim. In particular, if C can be replicated by the strategy (ν_0, φ) then the law of one price states that the price of C should be ν_0, i.e. the construction cost of the replicating portfolio, since $C = V_T((\nu_0, \vartheta))$. This notion is expressed by the following theorem.

Theorem 2 (Martingale Valuation)
Suppose that C is an attainable contingent claim, and that $\mathbb{Q}$ is an EMM for the numeraire N. If we denote C_t as the price at time $t < T$, then

$$C_t = N_t \mathbb{E}_{\mathbb{Q}} \left(\frac{C}{N_T} \middle| \mathcal{F}_t \right).$$

An important concept which surrounds the construction of the foundations of mathematical finance is the notion of a *complete* market.

Definition 2.1.7
A financial market $((\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}), X)$ is complete if every contingent claim is attainable.

This property will be a focusing relaxation for the exploration of this dissertation. Issues clearly arise for pricing claims in *incomplete* markets since we can no longer use theorem 2. We will explore this in the upcoming chapters. For now though, we will work with complete market models.

As the final step before we can state the second Fundamental Theorem of Asset Pricing (FTAP II), we must cover a martingale representation property. In particular, this representation permits that martingales can be decomposed into an initial value and and integral of a predictable process.

Definition 2.1.8
A stochastic process X is said to have the predictable representation property (PRP) with respect to $\mathbb{P}$ if every $\mathbb{P}$-local martingale M can be written as

$$M_t = M_0 + \int_0^t \varphi_s dX_s,$$

for some predictable X-integrable process φ

We are now able to utilise the FTAP II in order to classify whether market models are complete.

Theorem 3 (FTAP II)
Let $((\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}), X)$ be a financial market whose components are non-negative and assume that NFLVR holds. Then the following are equivalent:

1. *The market is complete.*
2. *There is a unique ELMM.*
3. *X satisfies PRP with respect to at least on ELMM for X.*

The martingale valuation theorem (2) assists the process of pricing contingent claims; however, it provides no insight into the process of finding a replication trading strategy. Property 3 of 3 brings us one step closer by showing that one exists without providing a procedure to find it. In particular, let H be a discounted contingent claim with $\mathbb{E}_{\mathbb{Q}}(|H|) < \infty$. Define the process $M = \{M_t : t = 0, 1, ..., T\}$ whereby $M_t := \mathbb{E}_{\mathbb{Q}}(H|\mathcal{F}_t)$. Then M is a martingale with $M_T = H$ and PRP tells us that there exists a predictable process φ^H such that

$$M_t = \mathbb{E}_{\mathbb{Q}}(H) + \int_0^t \varphi_s^H dX_s.$$

Then for $t = T$, the predictable process φ^H is a replicating strategy for H. With the addition of market frameworks, we can utilise this to establish a replicating strategy finding process.

2.1.2 Continuous-Time Models

Throughout this section, we will explore the continuation of the theory of arbitrage-free and complete market models. In particular, we have covered the foundations required for us to discuss some of the popular continuous-time market models. The Black-Scholes model is an important milestone for modern mathematical finance. The work pioneered the way for much of the exploration which followed it and the model remains popular almost 40 years on despite the rather restrictive assumptions which it imposes. The Heston model, originally proposed by Heston, 1993, relaxes the Black-Scholes assumption of constant volatility by introducing a stochastic volatility and has become another popular model in industry. Whilst we acknowledge the plethora of models available, we will be using these two models in our comparative process; hence, we will analyse these models for an understanding of continuous-time models.

Black-Scholes Model

The classical Black-Scholes model explores a market with a single risky asset and one source of noise which is an important foundation for understanding the invaluable continuous-time model. The discussion below follows closely from Mavuso, 2019.

Let us continue to work in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ which has been equipped with the filtration $\mathbb{F}$. The filtration is the natural filtration of a one-dimensional Brownian motion W_t. For technical reasons, the filtration is augmented to satisfy the usual conditions of continuity and completeness, see Protter, 2005 for why this is essential.

Black and Scholes suppose that the asset price process is modelled by a Geometric Brownian motion. In particular, the single risky asset S_t follows an Itô diffusion with dynamics

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \quad S_0 > 0. \tag{2.2}$$

Here, μ represents the *drift* and σ is the *volatility* of the asset price process. We must explicitly remark that the mean and volatility are time-homogeneous constants. Relaxing the constant volatility assumption forms a major area of research due to its rather restrictive and unrealistic nature. For now though, we remain with the more classical assumption for our foundation.

Additionally, the market is equipped with a riskless asset. An asset is riskless in the market if it has zero volatility; ensuring that the price process is deterministic. In reality, this asset is associated with a money market account which we will denote as $S_t^0 := B_t$ which has dynamics

$$dB_t = rB_t, \quad B_0 = 1,$$

for some risk-free rate of return r. Again, we make the classical assumption that this rate is constant. It is common to let B_t be the numeraire process and establish theory under the discounted price process and the risk-neutral measure. Following the theory of mathematical finance, since the asset prices are Itô diffusions, these processes are continuous semi-martingales which ensures that Itô calculus is valid. We must reiterate that when we describe the dynamics as a stochastic differential equation (SDE)

$$dS_t = \mu S_t dt + \sigma S_t dW_t,$$

we are referring to the common shorthand of a stochastic integral equation

$$S_t = S_0 + \int_0^t \mu S_u du + \int_0^t \sigma S_u dW_u, \quad \forall t \in [0, T], \tag{2.3}$$

whereby the second integral is an Itô integral.

Let us briefly return to the filtration of the probability space. We have defined the filtration to be the natural filtration of the one-dimensional Brownian motion; but, in reality, we only observe asset prices. It can easily be shown through Itô's formula that the unique solution to 2.3 is

$$S_t = S_0 e^{\sigma W_t + (\mu - \frac{1}{2}\sigma^2)t}, \quad \forall \ t \in [0, T]. \tag{2.4}$$

Hence, $S_t = f(W_t)$ for the appropriate Borel function f associated with 2.4. Applying the Doob-Dynkin Lemma informs us that $\mathbb{F}^S \subseteq \mathbb{F}^W$. In fact, Musiela and Rutkowski, 2010 succinctly shows that $\mathbb{F}^S = \mathbb{F}^W = \mathbb{F}$. Consequently, the filtration is generated by all the information of the assets prices available at the appropriate times; signifying the information available to the market. Hence, we make the assumption that the market has all the information and this information becomes known to all the traders at the same time.

Following the notions of the previous section, we would like to achieve a martingale process for the asset prices; yet, equation 2.4 clearly shows that the asset price process is only a martingale if the drift process is zero. Consequently, we must consider Girsanov's Theorem. In particular, we need to find the Girsanov kernel which causes the dynamics of discounted price process, $\bar{S}$, to be

driftless under the new measure $\mathbb{Q}$ defined by the Girsanov theorem. A simple application of Itô's formula shows us that

$$\begin{aligned} d\bar{S}_t = d\left(\frac{S_t}{B_t}\right) &= (\mu - r)\,\bar{S}_t dt + \sigma \bar{S}_t dW_t \\ &= (\mu - r)\,\bar{S}_t dt + \sigma \bar{S}_t d(\tilde{W}_t + \lambda_t dt) \\ &= (\mu - r + \sigma\lambda_t)\,\bar{S}_t dt + \sigma \bar{S}_t d\tilde{W}_t, \end{aligned}$$

where $\tilde{W}$ is a $\mathbb{Q}$ Brownian motion. Hence, we must choose the kernel, λ_t, such that $r - \mu = \sigma\lambda_t$ in order to ensure the discounted price process is driftless under $\mathbb{Q}$. Consequently,

$$\lambda_t = -\left(\frac{\mu - r}{\sigma}\right) \tag{2.5}$$

and the kernel is the negative *market price of risk*.

The importance of finding such a kernel follows from our desire to have an arbitrage-free market model. Recall the FTAP I 1 which implies that *the Black-Scholes Model is arbitrage-free if and only if there exists a risk-neutral measure, or more directly, if there exists an ELMM for the discounted asset prices.*

The next step of progression, after establishing an arbitrage-free model, is to explore the procedure of pricing and hedging within the Black-Scholes model. Clearly, we want to utilise the martingale valuation theorem 2; however, this requires that the contingent claim is attainable. The FTAP II 3 plays an essential role here. Since the kernel in 2.5 is unique, shown by Musiela and Rutkowski, 2010, the ELMM is unique and all contingent claims are attainable.

Let us briefly explore the problem of pricing contingent claims under the framework. It is at this point that we acknowledge the two approaches for pricing and hedging contingent claims. The first method follows the historical approach taken in Black and Scholes, 1973 whereby partial differential equations were set up such that the Feynman-Kac theorem was applicable. We do not follow this approach and rather continue with the martingale approach first shown by Harrison and Pliska, 1981.

Consider the contingent claim C which is not path dependent such that $C = g(S_T)$ for some Borel measurable function g. For mathematical ease, let us consider the discounted contingent claim $H = e^{-rT}C = e^{-rT}g(e^{rT}X_T) = h(X_T)$. Then pricing the contingent claim with the martingale valuation theorem reduced to solving

$$\begin{aligned} \pi(C) &= \mathbb{E}_{\mathbb{Q}}[e^{-rT}g(S_T)] \\ &= e^{-rT}\mathbb{E}_{\mathbb{Q}}[g(S_T)]. \end{aligned}$$

The more interesting avenue to explore is the problem of hedging. Let us first find the replicating strategy for the discounted claim H and then show that this replicating strategy replicates C. Define the martingale process, the claim's

price process, $M = \{M_t : 0 \leq t \leq T\}$ such that

$$\begin{aligned} M_t :=& \mathbb{E}_{\mathbb{Q}}\left(h(X_T)|\mathcal{F}_t\right) \\ =& \mathbb{E}_{\mathbb{Q}}\left(h(X_T)|X_t\right), \end{aligned}$$

since the discounted price process X is a Markov process. Following the Doob-Dykin Lemma, there must exist a function $F : [0, \infty) \times \mathbb{R} \to \mathbb{R}$ such that

$$M_t = F(t, X_t).$$

The replicating strategy is found by utilising Ito's Lemma and the Predicable Representation Property. In particular,

$$\begin{aligned} dM_t =& \frac{\partial F}{\partial t}dt + \frac{\partial F}{\partial x}dX_t + \frac{1}{2}\frac{\partial^2 F}{\partial x^2}d\langle X\rangle_t \\ =& \left(\frac{\partial F}{\partial t} + \frac{1}{2}\sigma^2 X_t^2 \frac{\partial^2 F}{\partial x^2}\right)dt + \sigma X_t \frac{\partial F}{\partial x}dW_t, \end{aligned}$$

where

$$\frac{\partial F}{\partial t} + \frac{1}{2}\sigma^2 X_t^2 \frac{\partial^2 F}{\partial x^2} = 0,$$

because the process M is a martingale, hence, driftless. With the PRP,

$$\begin{aligned} M_T &= F(T, X_T) = h(X_T) = H \\ &= \mathbb{E}_{\mathbb{Q}}\left(H\right) + \int_0^T \frac{\partial F}{\partial x}dX_t. \end{aligned}$$

Therefore, the replicating strategy for the discounted contingent claim C is given by $(\mathbb{E}_{\mathbb{Q}}(H), \varphi)$; the initial value and risky asset strategy respectively, where

$$\varphi_t \coloneqq \frac{\partial F}{\partial x}(t, X_t).$$

Since the strategy is self-financing, the strategy for the riskless asset is simply

$$\eta_t = F(t, X_t) - \varphi_t X_t.$$

Two problems still remain: the replicating strategy for C instead of H; the function F. If V once again denotes the value process of the trading strategy and we apply the above trading strategy, then

$$\begin{aligned} V_T =& \eta_T B_T + \varphi_T S_T \\ =& (F(T, X_T) - \varphi_T X_T)B_T + \varphi_T S_T \\ =& B_T F(T, X_T) \\ =& H B_T \\ =& C. \end{aligned}$$

Additionally, define $V(t, S_t) \coloneqq V_t = e^{rt}F(t, X_t) = e^{rt}F(t, e^{-rt}S_t)$ then

$$\frac{\partial F}{\partial X_t} = \frac{\partial V}{\partial S_t}.$$

Consequently, given a contingent claim and the value process for such a claim, the price and replicating strategy can be found through martingale methods.

Whilst the above discussion is restricted to a single risky asset, a generalised Black-Scholes framework exists. Such a framework considers a K-dimensional Brownian motion and d risky assets. See Ouwehand, 2015 for further details on this extension.

Heston Model

The Heston Model, Heston, 1993, is a popular model which aims to relax the assumption of constant volatility which is imposed by the classical Black-Scholes Model. In particular, the Heston Model is an example of a stochastic volatility model.

Stochastic volatility models aim to model the asset price dynamics with a volatility which has stochastic dynamics itself. In general, a stochastic volatility model is represented by

$$dS_t = \mu(t, S_t)dt + \sigma_t S_t dW_t, \tag{2.6}$$

$$d\sigma_t = a(t, \sigma_t)dt + b(t, \sigma_t)d\hat{W}_t, \tag{2.7}$$

where W and $\hat{W}$ are both one-dimensional Brownian motions with co-variation $d\langle W, \hat{W}\rangle_t = \rho dt$ for $\rho \in [-1, 1]$. σ_t is known as the instantaneous volatility. Note, that the functions μ, a, b are possibly functions of both t and S_t or σ_t respectively in this general setting.

In general, stochastic volatility models are **not** complete market models. To see this, consider the following. Let $dB_t = r_t dt$ such that B_t is the money market account; furthermore, let B be the numeraire process. Then $\mathbb{Q}$ is a risk-neutral measure if $\bar{S}_t = \frac{S_t}{B_t}$ is a (local) martingale under $\mathbb{Q}$. Supposing that such a measure exists, then Girsanov's theorem allows us to define $\mathbb{Q}$ by

$$\frac{d\mathbb{Q}}{d\mathbb{P}} = \mathcal{E}(\lambda \bullet \vec{W})_T,$$

where $\vec{W}$ is the two-dimensional Brownian motion $(W, \hat{W})$. Here, we require that the discounted price process is a $\mathbb{Q}$-(local) martingale. $(\vec{W}_t - \int_0^t \lambda_s ds)_t$ is a $\mathbb{Q}$-Brownian motion which implies that the kernel $\lambda_t = (\lambda_t^1, \lambda_t^2)^T$ is only restricted by

$$\lambda_t^1 = \frac{r_t - \mu(t, \sigma_t)}{\sigma_t},$$

whilst the second component remains unrestricted. Hence, there are infinitely

many appropriate Girsanov kernels; therefore, infinitely many risk-neutral measures in the model. By FTAP II, stochastic volatility models are generally incomplete. Here, the assumption about the existence of a risk-neutral measure was made; however, we can relax this assumption. If the component λ_t^1 for the kernel can be found to satisfy the above equation, then a risk-neutral measure exists.

Given that stochastic volatility models are incomplete, the process for pricing contingent claims in such a market must differ from that of the classical Black-Scholes model. The procedure of arbitrage pricing by finding a replicating strategy is no longer valid. In fact the general pricing process for stochastic volatility models is a rather extensive and unnecessary discussion for this dissertation. Hence, let us shift our focus to the particulars of the Heston Model.

The Heston model assumes dynamics such that

$$dS_t^1 = \mu S_t^1 dt + \sqrt{\nu_t} S_t^1 dW_t^s, \tag{2.8}$$
$$d\nu_t = \kappa(\theta - \nu_t)dt + \xi\sqrt{\nu_t} dW_t^\nu, \tag{2.9}$$

where ν_t represents the instantaneous *variance*. $\mu, \kappa, \theta, \xi, \nu_0, S_0^1 \in \mathbb{R}^+$; however, let $\mu = 0$ and consider a risk-neutral probability space with $\mathbb{Q}$ as the measure. Then, again W^s and W^ν are one-dimensional $\mathbb{Q}$-Brownian motions with $d\langle W^s, W^\nu \rangle_t = \rho dt$ for $\rho \in [-1, 1]$. The Feller condition, $2\kappa\theta > \xi^2$, is commonly imposed on the modelling process to ensure that the variance process is strictly positive.

A brief remark is directed towards the use of the Heston model. In contrast the the Black-Scholes model, the Heston model is generally used as a pricing model. That is, the model is generally represented under its risk-neutral dynamics and the real-world dynamics are not used directly. Later we deviate from this norm to establish an effective benchmark, as explained explicitly in 4.2.3.

In order to price and hedge contingent claims in the incomplete market created by the Heston model, we require the ability to trade in the volatility process. Since such a discussion is beyond the scope of this dissertation, the interested reader is referred to Section 5.2 of Buehler et al., 2019.

2.2 Neural Networks

Neural networks have become an essential component for a variety of predictive problems through the drastic improvements in computational technology. Consequently, it is no surprise to find neural networks as an avenue of exploration in the finance field. In this preliminary component, we will discuss the basics of neural networks and set the associated notation utilised throughout the dissertation. We start with neural networks, then recurrent networks and finally introducing variational auto-encoders. Importantly, we formulate neural networks under a stochastic processes framework. For a deeper discussion of neural networks, refer to Bishop and Nasrabadi, 2006.

2.2.1 Feed Forward Neural Networks

Let X be a random vector on $(\Omega, \mathcal{F}, \mathbb{P})$ and let Y be another random vector such that Y is measurable with respect to the sigma algebra of X, $\sigma(X)$. Further suppose that $X \in \mathbb{R}^B$ and $Y \in \mathbb{R}^M$. Intuitively, X represents the predictor variable and Y is the response variable of a hypothetical problem. Following the Doob-Dynkin Lemma, there must exist a function $f : \mathbb{R}^B \to \mathbb{R}^M$ such that

$$Y = f(X). \tag{2.10}$$

Neural networks often provide a solution to the approximation of this underlying function f. Consequently, let us cover the basic neural network structure.

Definition 2.2.1
A standard feed-forward neural network is a function $f^{\mathcal{NN}} : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ *such that*

$$f^{\mathcal{NN}}(x) = F_L \circ F_{L-1} \circ \cdots \circ F_1,$$

where

- L *is the number of layers of the network.*
- N_ℓ *for* $\ell = 1, \ldots, L-1$ *is the number of nodes in the hidden layer* ℓ *and* N_0, N_L *are the dimensions of the input and output layers respectively.*
- $F_\ell = \sigma_\ell \circ W_\ell$ *for* $\ell = 1, \ldots, L$.
- $\sigma_\ell : \mathbb{R} \to \mathbb{R}$ *is the activation function.*
- $W_\ell(x) = A^\ell x + b^\ell$ *for some weight and bias matrices* $A^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$, $b^\ell \in \mathbb{R}^{N_\ell}$.

Intuitively, a feed-forward neural network is a combination of layers which receives an N_0-dimensional input to produce an N_L-dimensional output. At each layer, the respective layer transforms the input by applying an activation function to an affine transformation of its input. The dimension of input for each of the hidden layers depends on the number of nodes chosen for the layers and is able to vary from the input and output dimensions of $f^{\mathcal{NN}}$. The choice of activation functions allows for the overall non-linear transformation of the input.

The Universal Approximation Theorem, by Hornik, 1991, is an invaluable contribution to the theory of neural networks. The theorem confirms that a feed-forward neural network can approximate any sufficiently regular function with arbitrary precision. More formally;

Theorem 4 (Universal Approximation Theorem, Hornik, 1991)
Suppose that σ is bounded and non-constant and let $C(\mathbb{R})$ denote the continuous functions on $\mathbb{R}$, then

1. *For any finite measure μ on $(\mathbb{R}^{d_0}, \mathcal{B}(\mathbb{R}^{d_0}))$, the set of all feed forward neural network mappings from $\mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$, denoted $\mathcal{NN}^{\sigma}_{\infty,d_0,d_1}$, is dense in $L^p(\mathbb{R}^{d_0}, \mu)$ for $1 \leq p \leq \infty$.*
2. *If in addition $\sigma \in C(\mathbb{R})$, then $\mathcal{NN}^{\sigma}_{\infty,d_0,d_1}$ is dense in $C(\mathbb{R}^{d_0})$ for the topology of uniform convergence on compact sets.*

Hence, the universal approximation theorem establishes the link between our problem formulation and neural networks. In particular, Hornik's theorem confirms that neural networks can approximate the function f in equation 2.10 with arbitrary precision; essentially providing that

$$Y \approx f^{\mathcal{NN}}(X).$$

2.2.2 Generalised Recurrent Based Neural Networks

Now let us extend the above introductory problem to incorporate a generalised recurrent structure. Let $X = \{X_0, X_1, \ldots, X_N\}$ be a discrete-time stochastic process on $(\Omega, \mathcal{F}, \mathbb{P})$ and let $Y = \{Y_0, Y_1, \ldots, Y_N\}$ be a process which is adapted to the natural filtration of X. For notational ease of the following dimensions, assume that X and Y are stochastic processes of random variables such that each X_n and $Y_n \in \mathbb{R}$ for $n = 0, 1, \ldots, N$. Then, by the Doob-Dynkin Lemma, there must exist $N + 1$ functions $f_0, \ldots, f_N$ such that

$$Y_n = f_n(X_0, \ldots, X_n), \quad \text{for each } n = 0, 1, \ldots, N,$$

where $f_n : \mathbb{R}^{n+1} \to \mathbb{R}$. Next, define the function $f : \mathbb{R}^{N+2} \to \mathbb{R}$ such that

$$f(\alpha_0, \ldots, \alpha_N, k) \coloneqq \sum_{n=0}^{N} f_n(\alpha_0, \ldots, \alpha_n) I_{\{n\}}(k), \quad \alpha_i, k \in \mathbb{R}.$$

By design, $f(\alpha_0, \ldots, \alpha_N, n) = f_n(\alpha_0, \ldots, \alpha_n)$ for all $n \in \{0, 1, \ldots, N\}$.

Now define an $N + 2$-dimensional stochastic process $C = \{C_0, \ldots, C_N\}$ such that

$$\begin{aligned} C_0 &= X_0 \boldsymbol{e}_1, \\ C_n &= C_{n-1} + X_n \boldsymbol{e}_{n+1} + \boldsymbol{e}_{N+2}, \end{aligned}$$

where $\boldsymbol{e}_i$ is the $N+2$-dimensional unit vector with value 1 for the i^{th} component and zero elsewhere. Then C is adapted to the natural filtration of X and

$Y_n = f(C_n)$ for every n. Furthermore, we can reformulate these equations into the more familiar recursive equations

$$\begin{aligned} C_n &= g(C_{n-1}, X_n), \\ Y_n &= f(C_n), \end{aligned}$$

where $g : \mathbb{R}^{N+2} \times \mathbb{R} \to \mathbb{R}^{N+2}$ is given by $g(c, x) = c + x\boldsymbol{e}_{n+1} + \boldsymbol{e}_{N+2}$.

Whilst this general recurrent framework allows for past information to be effectively utilised, its practical application is restricted. Above the fact that f is unknown, the framework quickly becomes computationally intractable as N increases. Consequently, we turn to the universal approximation theorem to resolve these issues and allow us to effectively utilise the recurrent framework. In particular, let $d \leq N + 2$ represent the (reduced) dimension of the process C and let us approximate the functions f and g by $f^{\mathcal{NN}}$ and $g^{\mathcal{NN}}$ respectively. Therefore,

$$\begin{aligned} Y_n &= f_n(X_0, \ldots, X_n) \\ &\approx f^{\mathcal{NN}}(g^{\mathcal{NN}}(C^{\mathcal{NN}}_{n-1}, X_n)) \\ &\approx f^{\mathcal{NN}}(C^{\mathcal{NN}}_n), \end{aligned}$$

where $C^{\mathcal{NN}}_n = g^{\mathcal{NN}}(C^{\mathcal{NN}}_{n-1}, X_n)$ is the neural network approximation for the d-dimensional state process C and $C^{\mathcal{NN}}_{-1}$ is a trainable network parameter.

The notion of recurrent based neural networks attempts to enhance the power of neural networks for sequential data problems, which is present in the above formulation. This is particularly apparent in time-series and natural language processing problems. Intuitively, an RNN retains, in a mathematical sense, some memory of previous inputs which is used in combination with the new input to provide an output. The use of neural networks to approximate the process Y described above is a flexible recurrent based neural network architecture. Simplifications and special cases of this general architecture have been popularised as the recurrent neural network (RNN) and long short-term memory (LSTM) models which we briefly discuss below.

Let the hidden state stochastic process $h = \{h_0, h_1, \ldots, h_{N-1}\}$, where each h_n is a random vector such that $h_n \in \mathbb{R}^d$ for all $n = 1, \ldots, N-1$, be defined by a function $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$

$$h_n \coloneqq g(h_{n-1}, X_n),$$

where $g(h, x) = \sigma(W_h h + W_x x + b_h)$. Here, σ is some activation function, traditionally $\sigma = \tanh$, and W_h, W_x and b_h are the trainable weight and bias matrices for the hidden state and input respectively.

Now define $f : \mathbb{R}^d \to \mathbb{R}$ such that

$$\begin{aligned} Y_n &= f_n(X_0, \ldots, X_n) \\ &\approx f^{\mathcal{NN}}(g(h_{n-1}, X_n)) \\ &\approx f^{\mathcal{NN}}(h_n). \end{aligned}$$

Intuitively, the hidden state process mathematically represents a form of memory of the previous inputs. This memory is then combined with the current input and transformed to achieve the response process Y. Despite the above representation following the simplest RNN structure, there exists a equivalent universal approximation theorem for RNNs under certain restrictions. In particular, if the pair (h_t, x_t) is Markovian, then the following theorem applies.

Theorem 5 (Universal Approximation Theorem for RNN, Schäfer and Zimmermann, 2006)
Let $f(\cdot) : \mathbb{R}^J \times \mathbb{R}^I \to \mathbb{R}^J$ be a measurable function and $g(\cdot) : \mathbb{R}^J \to \mathbb{R}^N$ be continuous; with the inputs $x_t \in \mathbb{R}^I$, the hidden states $h_t \in \mathbb{R}^J$, and the output $y_t \in \mathbb{R}^N$ for $t = 1, \ldots, T$. Then, any open dynamical system of the form

$$\begin{aligned} h_t &= g(h_{t-1}, x_t), \\ y_t &= f(h_t), \end{aligned}$$

can be approximated by an element of the function class $RNN^{I,N}(s)$ with an arbitrary precision, where s is a continuous sigmoidal activation function.

The LSTM architecture proposed by Hochreiter and Schmidhuber, 1997 is a more complex architecture compared to the classical RNN which is tedious to describe by the above generalised architecture. Since such a formulation is beyond the scope of this dissertation, we choose to simply describe the architecture briefly and display the architecture in figure 2.1. The LSTM model

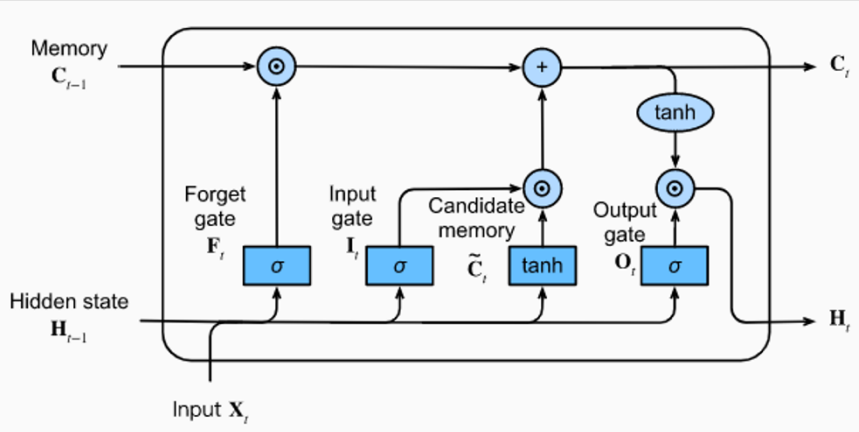


FIGURE 2.1: LSTM model architecture Zhang et al., 2021

resolves the issue of long-term dependencies inherent in the RNN model. This is achieved by allowing the model to have two memory states, one for short-term memory (*known as the hidden state,* $(h_n)_{n=0}^N$) and the second for long-term memory (*known as the cell state,* $(c_n)_{n=0}^N$). For each n, the cell state is updated

through *forget* and *remember/input* transformations which use c_{n-1}, h_{n-1}, X_n. To achieve an output from the model, c_n, h_{n-1}, X_n are transformed into the output h_n. Each of the above transformations follow a feed-forward neural network structure with specified activation functions. This could be formalised, albeit less elegantly than above, as follows. Suppose that $c_n \in \mathbb{R}^{d_c}$ and $h_n \in \mathbb{R}^{d_h}$ and let $g : \mathbb{R}^{d_c} \times \mathbb{R}^{d_h} \times \mathbb{R} \to \mathbb{R}^{d_c} \times \mathbb{R}^{d_h}$ be the neural network approximation of the composition of the cell and hidden state transformation. Then

$$\begin{aligned}(c_n, h_n) &= g(c_{n-1}, h_{n-1}, X_n),\\ Y_n &= f^{\mathcal{NN}}(h_n).\end{aligned}$$

2.2.3 Variational Auto-Encoders

Before discussing variational auto-encoders (VAEs) models, we must begin with an auto-encoder model description. Auto-encoders provide a dimensionality reduction tool based on neural network approximations which are beneficial in a plethora of applications such as denoising and signal processing; however, the generative limitations motivate the discussion of the VAE variant.

Auto-encoders

Begin with an N-dimensional random vector X on $(\Omega, \mathcal{F}, \mathbb{P})$ where N is relatively large and let Y be an M-dimensional random vector such that Y is measurable with respect to $\sigma(X)$. Now let $f : \mathbb{R}^N \to \mathbb{R}^M$ such that

$$Y = f(X),$$

where the existence of f follows from the Doob-Dynkin Lemma.

Now define the latent vector L to be a ℓ-dimensional random vector where $\ell < N$ and assume that $L \in m\sigma(X)$. Hence, there exists some functions $f_e : \mathbb{R}^N \to \mathbb{R}^\ell$ and $f_d : \mathbb{R}^\ell \to \mathbb{R}^M$. An auto-encoder utilises the universal approximation theorem to validate that the encoder and decoder functions, f_e and f_d respectively, can be approximated by feed-forward neural networks. The architecture is displayed in figure 2.2

Whilst the above provides the essential idea behind auto-encoders, we must take special note of our assumption that $L \in m\sigma(X)$. Such a random vector need not, *and does not*, exist for general a dimension ℓ. We motivate this assumption by the goal of dimensionality reduction which implies that we believe that $\sigma(Y) \subset \sigma(X)$. Hence, the choice of ℓ must be reasonable in application to match this motivation.

Variational auto-encoders

The VAE architecture was introduced by Kingma and Welling, 2013 as a variation of the auto-encoder which possessed generative characteristics. Hence, let us begin with a problem description for generation and the limitations inherent with the auto-encoder architecture.

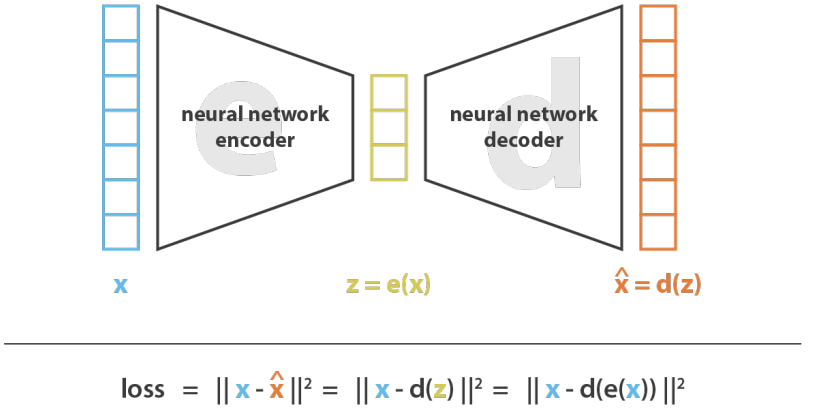


FIGURE 2.2: Auto-encoder architecture Rocca, 2019

Let X be an N-dimensional random vector on $(\Omega, \mathcal{F}, \mathbb{P})$. Without knowledge of the measure $\mathbb{P}$, we want to generate samples of X. In other words, let Y also be an N-dimensional random vector such that $X = Y$ $\mathbb{P}$-a.s. If the latent vector L is N-dimensional, then we have that $Y = f_d(f_e(X))$ for the identity functions f_e and f_d; however, if we have $\ell \ll N$, then neither f_e nor f_d is the identity. Theoretically, one could use samples of the random vector L to generate samples of Y from $Y = f_d(L)$; however, this presents the main limitation of an auto-encoder for the problem.

The encoded latent vectors within the auto-encoder are neither continuous nor maintain a regular structure. In application, clusters are developed in the latent space to enhance the decoder's performance. Consequently, samples from the latent space are both difficult to produce effectively and experience poor stability from the decoder. The VAE resolves this issue by ensuring that the latent space is both continuous and easy to sample from.

For the VAE architecture, we must add another set of components. Let $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ both represent ℓ-dimensional vectors and let $f_e : \mathbb{R}^N \to \mathbb{R}^\ell \times \mathbb{R}^\ell$ such that

$$(\boldsymbol{\mu}, \boldsymbol{\sigma}) = f_e(X).$$

Now, let $L|\boldsymbol{\mu}, \boldsymbol{\sigma} \sim \mathcal{N}_\ell(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. Additionally, we want $L \sim \mathcal{N}_\ell(\mathbf{0}, c\boldsymbol{I})$ for some $c \in \mathbb{R}$ which we discuss further soon. With this alteration to the encoding process, the latent space is now continuous and, since $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance vectors, the latent space is more interpretable.

As before, we approximate the functions f_e, f_d with the feed-forward networks $f_e^{\mathcal{NN}}, f_d^{\mathcal{NN}}$ to construct the VAE architecture which is visualised by figure 2.3 below. If $L \sim \mathcal{N}_\ell(\mathbf{0}, c\boldsymbol{I})$ then the grouping of vectors in the latent space is close and this ensures a sense of regularity. To train the network to achieve this, we must introduce another loss measure for the network.

Definition 2.2.2 (Kullback-Leibler Divergence)

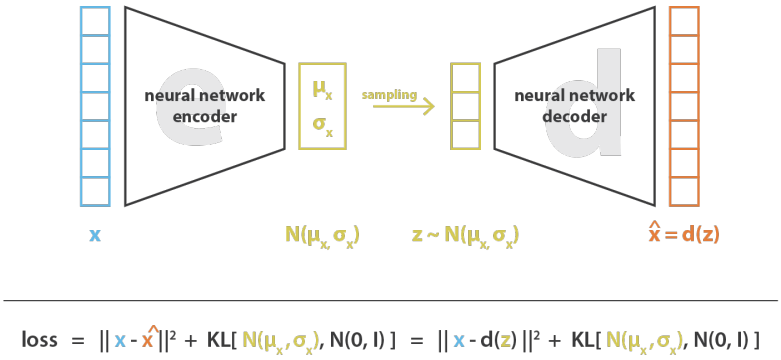


FIGURE 2.3: Variational auto-encoder architecture Rocca, 2019

Let $\mathbb{P}$ and $\mathbb{Q}$ be probability measures on $(\Omega, \mathcal{F})$ such that $\mathbb{P}$ is absolutely continuous with respect to $\mathbb{Q}$; then, the relative entropy from $\mathbb{Q}$ to $\mathbb{P}$ is defined as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \int_{\Omega} \ln\left(\frac{d\mathbb{P}}{d\mathbb{Q}}\right) d\mathbb{P},$$

where $d\mathbb{P}/d\mathbb{Q}$ is the Radon-Nikodym derivative of $\mathbb{P}$ with respect to $\mathbb{Q}$.

The Kullback-Leibler loss for the VAE architecture is the sum of the Kullback-Leibler divergence scores between the components of the conditional latent vector and a standard normal distribution. The computation can be simplified analytically to

$$\text{loss}_{KL} = \sum_{i=1}^{\ell} \left(\sigma_i^2 + \mu_i^2 - \ln(\sigma_i) - 1\right),$$

which is minimised when $\mu_i = 0$ and $\sigma_i = 1$.

In chapter 4.3, we use a variant of the VAE known as the conditional VAE or CVAE. As discussed, generating output vectors from a VAE relies on a sample of the random latent vector L; however, we have no control of this generation. The conditional variant, proposed by Sohn, Lee, and Yan, 2015, resolves this problem by allowing the VAE to be conditioned on some auxiliary input, enabling structured output predictions.

2.3 Path Signatures

In section 4.3, the use of rough paths signatures is implemented for the purposes of market generation. Path signatures have proven to be an extremely effective feature mapping for paths, with a plethora of state-of-the-art applications. Since the topic is far from trivial mathematics, we provide an introduction here. For a more detailed discussion, refer to Chevyrev and Kormilitzin, 2016.

The introduction begins with a formal introduction of paths and *rough* paths. After this, the path signature is defined and briefly discussed.

2.3.1 Paths

Before we can explore the field of path signatures, we must formalise the notion of a path. Mathematically, a path is a function of bounded variation which maps an interval to an N-dimensional Euclidean space. More specifically; we call f a path if $f : [a, b] \to \mathbb{R}^N$ for $a \leq b \in \mathbb{R}$. In most cases, the intervals which paths map are positive since they contain the time mapping of the function. Suppose that one has a Cartesian plane problem where both the horizontal and vertical position tracking is required; then, one could denote $g(t) = (x(t), y(t))$ where $x(t)$ and $y(t)$ denote the horizontal and vertical position at time t. This shows that in order to maintain all time information, we need $N+1$-dimensions to describe the trajectory of an N-dimensional path.

Now assume that f is not a function of bounded variation, then f is instead known as a rough path in this case. Rough path theory explores paths which are highly irregular and paths are not piece-wise differentiable. For this introduction, we assume paths have bounded variation to define notions; however, a natural extension to rough paths generally exists.

For notational ease, let us now denote the path $X : [a, b] \to \mathbb{R}^N$ and let $X_t := X(t)$ for $t \in [a, b]$. Then if X_t, Y_t are paths, we can define the path integral as

$$\int_a^b Y_t dX_t.$$

If both paths have bounded variation then the path integral is the well-known Riemann-Stieljes integral which can be interpreted as the area obtained by plotting Y_t against X_t for $t \in [a, b]$. When these paths are rough paths, the interpretation is more complex and beyond the scope of this dissertation. For a more detailed discussion of rough paths, refer to Friz and Hairer, 2020.

1. **Translation Invariance**
   The path integral
   $$\int_a^b Y_t dX_t$$

is invariant to the value of X_a. Let us define the vertically shifted path $Z_t := X_t + c$ for some $c \in \mathbb{R}$, then

$$\int_a^b Y_t dX_t = \int_a^b Y_t dZ_t.$$

2. **Reparameterisation Invariance**
   For the case where t represents time, reparametrisation invariance implies that the path integral is invariant to the (instantaneous) speed at which X_t is traversed. This arises from the fact that the path integral is an integral with respect to the time dependent path X_t instead of time t itself.

Whilst we will see these invariance properties again soon, the true building blocks of a path signature is defined by the next notion. We have defined the path integral over a fixed interval $[a, b]$, yet we can define path integrals as a path itself by varying the upper bound such that

$$Z_c = \int_a^c Y_t dX_t,$$

for $c \in [a, b]$. Then Z_c is a path defined by path integrals.

2.3.2 Signatures

Consider the N-dimensional path $X_t, t \in [a, b]$ such that $X_t = (X_t^1, \ldots, X_t^N)$. Then each X_t^i is the 1-dimensional co-ordinate paths which build up X and assume that each co-ordinate path is quantified in 1-dimensional units. The signature of the path X is an ordered set comprising of all the possible path integrals that can be constructed involving the combinations of the 1-dimensional components X_t^i.

The first level of these combinations explored by the signature is given by

$$S(X)_{a,t}^n := \int_a^t dX_c^n.$$

Let us briefly reconsider the path integral $\int_a^b Y_t dX_t$ defined above. If we let $Y_t = 1$, then

$$\int_a^b Y_t dX_t = \int_a^b X_t' dt = X_b - X_a.$$

Consequently, the N first order signature terms are simply the increments of $X^1, \ldots, X^N$ over the interval $[a, t]$ with $t \in [a, b]$. The collection of the first order terms is then $S(X)_{a,t}^1, \ldots, S(X)_{a,t}^N$. Each of these path integrals are quantified in 1-dimensional units, for example distance units. Again, we re-iterate that each of $S(X)_{a,t}^n$ is itself a path.

The second level of the path signature is defined by the path integrals involving two paths. The one path is X_t^m and the other path is the first order, increment path $S(X)_{a,t}^n$ of the path X_t^n. In particular, denote the N^2 second order components by $S(X)_{a,t}^{1,1}, \dots, S(X)_{a,t}^{N,N}$ such that

$$S(X)_{a,t}^{n,m} := \int_a^t S(X)_{a,c}^n dX_c^m.$$

Unlike the first order components, these path integrals are quantified using 2-dimensional units, for example area units. Intuitively, these second order integrals represent the area obtained when plotting $S(X)_{a,c}^n$ against X_c^m over the interval $[a,t], t \in [a,b]$.

The remainder of the path signature levels are then obtained by iteratively obtaining the path integrals. Consider the k^{th} level of the path signature. This level consists of N^k possible integrals of order k which has components denoted by $S(X)_{a,t}^{i_1,\dots,i_k}$ such that

$$S(X)_{a,t}^{i_1,\dots,i_k} := \int_a^t S(X)_{a,c}^{i_1,\dots,i_{k-1}} dX_c^{i_k}.$$

Here, we take a brief moment to clarify the multi-index $i_1, \dots, i_k$. For a given level k, i represents one of the N^k combinations, and so i_j represents the j^{th} component of the i^{th} combination.

The path signature $S(X)_{a,b}$ is then defined as the infinite ordered set of terms that we obtain by considering all levels of $k \geq 0$ and with respect to the path's domain $[a,b]$ such that

$$S(X)_{a,b} := \left(1, S(X)_{a,b}^1, \dots, S(X)_{a,b}^N, S(X)_{a,b}^{1,1}, \dots, S(X)_{a,b}^{N,N}, S(X)_{a,b}^{1,1,1}, \dots\right).$$

In the path signature, the initial term is set to 1 by convention, motivated by the fact that it represents the zeroth level. Obviously, in practice we cannot calculate and utilise the entire infinite set, so we take a truncation of the path signature. If we truncate the signature at level K, meaning that we only consider the first K level, then K is known as the order of the path signature.

Before continuing to the next section, let us explicitly look at what information a path signature captures from a probabilistic perspective. Suppose that we have N random variables $T^1, \dots, T^N$ which have the probability density functions $q^1, \dots, q^N$ and support $[a,b]$. Then we can define the CDF of the random variable T^n, denoted Q_t^n, as the probability that $T^n \leq t$. But then the CDF can be interpreted as a path, resulting in the N-dimensional path $Q_t := (Q_t^1, \dots, Q_t^N)$ of the random vector CDF. With such a path, let us see what each level of the signature describes:

1. The first level of the signature is defined as

$$S(Q)_{a,t}^n = Q_t^n - Q_a^n = \mathbb{P}(a < T^n \leq t).$$

2. The second level is given as

$$S(Q)_{a,t}^{n,m} = \int_a^t S(Q)_{a,c}^n Q_c'^m dc = \int_a^t \mathbb{P}(a < T^n \leq c) q^m(c) dc.$$

   But the latter equation can be interpreted as $a < T^n \leq T^m \wedge a \leq T^m \leq t$ which implies that

$$S(Q)_{a,t}^{n,m} = \mathbb{P}(a < T^n \leq T^m \leq t).$$

3. The k^{th} level can then be shown to be

$$S(Q)_{a,t}^{i_1,\dots,i_k} = \mathbb{P}(a < T_1^{i_1} \leq \cdots \leq T_k^{i_k} \leq t),$$

   where T_j^n is used to represent individually and identically distributed random variables with pdf q^n.

Now consider the signature

$$S(Q)_{a,b} = (1, \mathbb{P}(a < T_1^1 \leq b), \dots, \mathbb{P}(a < T_1^1 \leq T_2^1 \leq b), \dots).$$

We can see that each of the level one terms have a value of 1 due to the support. Analysing the remainder of the terms, it is clear that in the special case whereby the path is a CDF, the signature encodes information about the joint CDF. This motivates the notion that for a general path X, the path signature encodes information about the structure of the individual co-ordinate paths and their correlation structure.

When it comes to computing signatures for feature extraction, the log signature can be a more effective projection when compared to the normal signature. In order to define the log signature, we require another definition.

Definition 2.3.1 (Formal Power Series)
Let the power series be defined as

$$x = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \cdots e_{i_k},$$

for which $\lambda_0 > 0$. Then we can define the formal logarithm as the power series

$$\log x = \log(\lambda_0) + \sum_{n \geq 1} \frac{(-1)^n}{n} \left(1 - \frac{x}{\lambda_0}\right)^{\otimes n},$$

where $\otimes n$ denotes the n-th power with respect to the tensor product $\otimes$.

Definition 2.3.2 (Log Signature)
For a path $X : [a, b] \to \mathbb{R}^d$, the log signature of X is defined as the formal power series $\log SX_{a,b}$.

2.3.3 Pre-Processing

With a basic understanding of paths, rough paths and signatures; we now discuss the application of the above concepts. First, we discuss the observation of paths.

In the above section we defined paths over a continuous interval of time; however, in practice, we only observe data at discrete times. For this reason, let us define an N-dimensional *data stream* as a sequence of points in N-dimensional Euclidean space which we observe at discrete times $t_1 < \cdots < t_M$. Interpolation methods are then required to obtain a path from the data stream. The resultant interpolated path then follows the above formal definition of a path. Common examples of interpolation methods are: linear; previous point; next point; and quadratic interpolation. It is important to note that whilst a path may be rough, a stream must have bounded variation since it is a collection of discrete observations. This bounded variation property extends to the associated interpolated path of the stream.

Let us denote the data stream $\hat{X} = (x_1, \ldots, x_M)$ such that each $x_i \in \mathbb{R}^N$ with the observation timestamps $a = t_1, \ldots, t_M = b$. It is not necessary that the observation times are evenly spaced through time. To ensure that we continue to reinforce that the stream is discrete, we denote the stream's movement through time as $\hat{X}[i] = x_i$ and follow the path definition for $\hat{X}[i] = (\hat{X}^1[i], \ldots, \hat{X}^N[i])$ to represent the 1-dimensional components. As one might expect, to obtain the path signature of a stream, we must obtain the interpolated path X such that $X_{t_i} = \hat{X}[i]$ and then simply compute the path signature of this interpolated path. This does, however, imply that the signature of a stream depends on the choice of interpolation method.

Because the path signature consists of iterated path integrals, it inherits their properties of translation invariance and reparametrisation invariance. Since the signature is invariant to path reparameterisation with respect to t, the stream may be given an arbitrary, strictly increasing sequence as the observation timestamps. This implies that the timestamps need not be provided with their associated observations when either visualising a stream or computing its signature. For applications such as handwriting recognition, where we do not care about the absolute position or speed at which a path is executed, both translation invariance and reparametrisation invariance are useful properties. These beneficial properties for such a problem prove to be detrimental for a plethora of others which are sensitive to the speed of the path execution and the position of observations with respect to the origin. In these cases, pre-processing stream transformations allow one satisfy desired properties and utilise the power of the path signature. We discuss two important transformations below.

1. **Time-integrated Transformation**
   Beyond being invariant to reparameterisation and translation, path signatures are also invariant to retracing of path segments. If we transform the path X to include a strictly increasing co-ordinate path, then the transformation is sufficient to ensure that the path signature is sensitive to reparameterisation and retracing. For this reason, the time-integrated

transformation T_{time} includes the timestamps of of the observations such that

$$T_{\text{time}}(\hat{X}^j) = ((t_1, x_1), \dots, (t_M, x_M)).$$

2. **Lead-Lag Transformation**
   In order extract useful feature representations of paths, the lead-lag transformation is commonly a powerful transformation to implement. Consider the stream $\hat{X} = (x_1, \dots, x_M)$, then we define the lead transformation of component j as

$$T_{\text{lead}}(\hat{X}^j) = (x_1^j, x_2^j, x_2^j, x_3^j, \dots, x_M^j, x_M^j) = (u_1, \dots, u_{2M-1})$$

   and the lag transformation is given by

$$T_{\text{lag}}(\hat{X}^j) = (x_1^j, x_1^j, x_2^j, x_2^j, \dots, x_{M-1}^j, x_M^j) = (v_1, \dots, v_{2M-1}).$$

   Then, the lead-lag transformation of component j is the stack of the two transformations such that

$$T_{\text{lead-lag}}(\hat{X}^j) = ((u_1, v_1), \dots, (u_{2M-1}, v_{2M-1})).$$

Before concluding this introduction, we make a special remark on quadratic variation. We have discussed that the signature is able to extract important features of a path; however, quadratic variation is not one of these features, Gyurkó et al., 2013. This proves problematic for the application of signatures in finance since the quadratic variation of process forms a crucial component of the literature. As a result, we require a transformation which allows the signature to capture the quadratic variation and cross variation of the processes. The lead-lag transformation is such a transformation.

Chapter 3

Hedging in Incomplete Markets

The pricing and hedging process under the Heston Model alludes to complexities which become apparent with incomplete markets. In particular, without a unique ELMM, the use of different ELMMs results in different prices for contingent claims given the martingale valuation theorem. Similarly, the contingent claim may be unattainable, hence we cannot find a replicating strategy for the claim; therefore, the law of one price can no longer be relied on. In this chapter, we will explore the general process of pricing and hedging in incomplete markets. We begin by covering the indifference pricing theory within the Buehler et al., 2019 framework before focusing on the process of quadratic hedging.

Despite extensive research surrounding the incomplete markets, there is little consensus on a best solution. Consequently, a brief summary of some approaches is provided throughout this chapter. In particular, we explore the notion of indifference pricing under convex risk measures before honing in our focus on the mean-variance framework. For further reading, we refer the reader to Henderson and Hobson, 2004 and Schweizer, 1999 for the two frameworks respectively.

For the purposes of introducing market incompleteness and an assumption relaxation, consider transaction costs as a market friction. In particular, if a trader purchases a position $\psi \in \mathbb{R}^d$ at time t_k, then they will incur a transaction cost of $c_k(\psi)$ for some cost function $c_k(x) : \mathbb{R}^d \to \mathbb{R}^+$. Then the total transaction costs of a trading strategy φ is given by

$$C_T(\varphi) \coloneqq \sum_{k=0}^{n} c_k(\varphi_k - \varphi_{k-1}).$$

Hence, the terminal value of a portfolio with the inclusion of transaction costs is given by

$$\nu_0 + (\varphi \cdot S)_T - C_T(\varphi).$$

Some common cost functions are:

- Fixed transaction costs: for $c_k^i > 0$ and $\epsilon > 0$; set $c_k(\psi) \coloneqq \sum_{k=1}^{d} c_k^i I_{|\psi^i| \geq \epsilon}$.
- Proportional transaction costs: for $c_k^i > 0$; set $c_k(\psi) \coloneqq \sum_{k=1}^{d} c_k^i S_k^i |\psi^i|$.

The theory of indifference pricing is generally covered including transaction costs. Given this definition of transaction costs above, we follow this convention in our introduction of the indifference pricing framework below.

3.1 Indifference Pricing

With the absence of the law of one price, the notion of certainty equivalents is a natural solution to pricing unattainable contingent claims. In particular, a certainty equivalent is the the amount with which a trader would be indifferent between entering the claim's position and remaining uninvolved. This amount is based on expected utility for the trader; therefore, the choice of utility function is an existing problem which is discussed below. We begin this discussion with convex risk measures which summarises a large group of sufficient utility functions.

3.1.1 Convex Risk Measures

The notion of convex risk measures was formally introduced by Föllmer, Schweizer, et al., 1990 to generalise the measures which result in indifference prices with certain desired properties[1]. These properties are: non-linear, replicable claims are priced at the Black-Scholes values; prices are monotonic in the claim; and bid prices are concave whilst sell prices are convex in the claim. These measures are defined as follows. If $H : \Omega \to \mathbb{R}$ is a mapping such that $H(\omega)$ represents the discounted worth of a financial position at the end of the trading period given the scenario ω, then H is called a financial position. Denote the collection of financial positions in the market by $\mathcal{H}$, then we define a convex risk measure as follows.

Definition 3.1.1
Assume that $X, Y, H \in \mathcal{H}$. Then $\rho : \mathcal{H} \to \mathbb{R}$ is called a convex risk measure if the following properties hold:

1. *Monotonicity: if $X \leq Y$, then $\rho(X) \geq \rho(Y)$.*
2. *Cash invariance: if $m \in \mathbb{R}$, then $\rho(X + m) = \rho(X) - m$.*
3. *Convexity: $\rho(\lambda X + (1 - \lambda)Y) \leq \lambda\rho(X) + (1 - \lambda)\rho(Y)$.*

If $\rho(0) = 0$, then ρ is normalised which can be a beneficial property depending on the setting. Often normalisation can be assumed without loss of generality.

Indifference pricing requires that the trader find the minimal price to achieve an optimal hedging strategy given the optimality criterion of the convex risk measure. Consider the cash-invariance property of a convex risk function. For an $H \in \mathcal{H}$, the invariance property clearly shows that $\rho(H + \rho(H)) = 0$ which implies that $\rho(H)$ is the minimal amount which must be added to H such that $\rho(H + m) \leq 0$ for some $m \in \mathbb{R}$. This final condition implies that the position is acceptable. In particular, let $\rho : H \to \mathbb{R}$ be a convex risk measure and let

[1] For a detailed discussion on convex risk measures, see Föllmer and Schied, 2011

$H \in \mathcal{H}$ be a contingent claim. Now define

$$\pi(-H) = \inf_{\varphi \in \Psi} \rho(-H + (\varphi \cdot S)_T - C_T(\varphi)), \tag{3.1}$$

where Ψ represents the space of possibly restricted trading strategies. Then, $\pi(-H)$ represents the minimal amount that the trader needs to charge for the sale of a contingent claim H in order to achieve an acceptable terminal portfolio position if they hedge optimally. This optimal hedging strategy is the minimising strategy $\varphi \in \Psi$ above.

The indifference price of the contingent claim H, $p(H)$, is defined as the amount at which the trader becomes indifferent between entering into the financial position $-H$ and not entering the position. That is, a price p_0 must solve $\pi(-H + p_0) = \pi(0)$. Buehler et al., 2019 show that π is cash-invariant; therefore,

$$p_0 \coloneqq p(H) \coloneqq \pi(-H) - \pi(0). \tag{3.2}$$

With this definition of the indifference price, if the claim H is attainable and the market is frictionless, then the indifference price coincides with the price derived from replication.

The above summary of the indifference pricing approach is described as an optimisation problem. A trader needs to optimise their trading strategy according to the risk measure ρ. The indifference price is then the minimal capital required to reach an indifferent position from the perspective of the trader. Therefore, there exists an optimisation for both the hedging strategy and price through this approach. The problem of pricing and hedging in incomplete markets can be reduced to a dynamic programming problem whereby the optimal hedging strategy is a direct consequence of solving for the price. As proposed by Buehler et al., 2019, the trader can reformulate the problem as a reinforcement learning problem. Under such a framework, the hedging strategy is regarded as the set of actions and the portfolio value as a reward. This reinforcement learning framework is formalised in the following section. Before this progression, we continue with the theory of convex risk measures.

With the above summary of convex risk measures, the process of obtaining such a risk measure remains unexplained. For this purpose, consider optimised certainty equivalents (OCE) which allow a trader to move from a utility loss function to an associated convex risk measure.

Proposition 6 (Lemma 3.7, Buehler et al., 2019)
Let $\ell : \mathbb{R} \to \mathbb{R}$ be a loss function[2]. Then define the measure

$$\rho(H) = \inf_{\omega \in \mathbb{R}} \{\omega + \mathbb{E}[\ell(-H - \omega)]\}, \quad H \in \mathcal{H}. \tag{3.3}$$

Then ρ is a convex risk measure.

[2]Recall that a loss function must be: continuous; non-decreasing; and convex

If we let the loss function described above to be derived from a utility function u such that: $\ell(x) = -u(-x)$, then 3.3 is the OCE which Ben-Tal and Teboulle, 2007 explore in detail. Here, the negative of the OCE produces a robust family of convex risk measures.

For examples and further discussions on exponential utility indifference pricing, refer to Buehler et al., 2019.

Next, let us proceed from the theoretical indifference pricing discussion to formalise the notion of deep hedging.

3.1.2 Deep Hedging

In order to price and hedge contingent claims with machine learning techniques, let us reformulate the optimisation problem above in terms of the reinforcement learning framework. For this purpose, define the stochastic process $I = \{I_0, \dots, I_N\}$ on $(\Omega, \mathcal{F}, \mathbb{P})$ to be the information set, also known as the feature set, such that $\mathbb{F}^S \subseteq \mathbb{F}^I$. Here, I_n can be a random vector capturing various time n features about the market.

Firstly, Buehler et al., 2019 shows that the (possibly) constrained optimisation problem expressed by 3.1 can be expressed as an unconstrained optimisation problem. Hence, consider an OCE-risk measure defined by 3.3 and assume without loss of generality that there are no strategy constraints. Then

$$\pi(-H) = \inf_{\varphi \in \Psi} \inf_{\omega \in \mathbb{R}} \{\omega + \mathbb{E}\left[\ell(H - (\varphi \cdot S)_T + C_T(\varphi) - \omega)\right]\}, \tag{3.4}$$

where the trading strategy $\varphi = \{\varphi_1, \dots, \varphi_N\}$ is a predictable process. Define the process Y such that $Y_n \coloneqq \varphi_{n+1}$; then, Y is adapted to the natural filtration of the underlying price process S and the natural filtration of the information set I since $\mathbb{F}^S \subseteq \mathbb{F}^I$. Following our discussion in 2.2.2, a trader can approximate the process Y, and hence the strategy φ from a generalised recurrent neural network of the form

$$\begin{aligned} C_n^{\mathcal{NN}} &= f^{\mathcal{NN}}(g^{\mathcal{NN}}(C_{n-1}^{\mathcal{NN}}, I_n)), \\ \varphi_{n+1} \coloneqq Y_n &\approx f^{\mathcal{NN}}(C_n^{\mathcal{NN}}). \end{aligned}$$

Furthermore, $\omega \in m\mathcal{F}_0$ implying that ω can be simultaneously optimised by being represented as a neural network model parameter. We see this notion formally defined below.

Whilst the above formulation provides a sufficient motivation for the application of recurrent networks for hedging, section 4.2 of Buehler et al., 2019 provides a more detailed proof based on the same concepts shown in 2.2.2 which we briefly remark here.

Let M denote the size of a neural network, $f^{\mathcal{NN}_M}$, and π^M represent this neural network's approximation of π, then

$$\pi^M(-H) = \inf_{\bar{\theta}\in\Theta_M} \inf_{\omega\in\mathbb{R}} \left\{ \omega + \mathbb{E}\left[\ell(H - (\varphi^{\bar{\theta}} \cdot S)_T + C_T(\varphi^{\bar{\theta}}) - \omega) \right] \right\} \tag{3.5}$$

$$= \inf_{\theta\in\Theta} J(\theta), \tag{3.6}$$

where: Θ_M is the parameter space of the neural network $f^{\mathcal{NN}_M}$; $\varphi^{\bar{\theta}}$ is the hedging strategy from $f^{\mathcal{NN}_M}_{\bar{\theta}}$; $\Theta = \mathbb{R} \times \Theta_M$; $\theta = (\omega, \bar{\theta}) \in \Theta$; and

$$J(\theta) := \omega + \mathbb{E}\left[\ell(X - (\varphi^{\bar{\theta}} \cdot S)_T + C_T(\varphi^{\bar{\theta}}) - \omega) \right]. \tag{3.7}$$

Buehler et al., 2019 show that the optimal strategy can be approximated arbitrarily well in the space of strategies generated by neural networks and that the price found by the neural network converges to the theoretical price.

Proposition 7
Define Ψ_M as the set of trading strategies obtainable by a network with size M and let $\pi(M)$ follow 3.5. Then, for any $H \in \mathcal{H}$,

$$\lim_{M\to\infty} \pi^M(H) = \pi(H).$$

Proof. See Buehler et al., 2019, proposition 4.3. □

Now let us establish the reinforcement learning structure for the deep hedging topic and OCE-risk measures. Recall that reinforcement learning is a type of machine learning problem where the model is given some goal to maximise or minimise by making decisions; however, the optimal decisions are not provided to the model. Each reinforcement learning problem can be described by a set of necessary component. In table 3.1, we discuss these components in general and their direct counterparts in the deep hedging problem.

The final component of the deep hedging reinforcement learning problem is one which is not necessary for all reinforcement learning problems. There are different approaches to reinforcement learning which share the components of table 3.1 but differ on how the goal is achieved. Deep hedging follows the value-based approach which optimises a function which provides some measure of the expected future rewards. In the deep hedging framework, a convex-risk measure is analogous to such a function.

Consequently, the problem of hedging in incomplete markets can be clearly described by reinforcement learning. In the upcoming quadratic hedging, we need only change the risk measure to the appropriate quadratic criterion to adjust the deep hedging formulation. The theoretical motivation here is proved numerically throughout 4.2.

Component	Description	
	General Problem	Deep Hedging
Environment	Provides the agent with a state and the appropriate consequences to the agent's actions	The financial market
Agent	An entity which decides on an action to take within the environment	A trader
State	A representation of the current environment conditions which the agent must provide an action for	Current market information such as the underlying asset price, analogous to the information set at a given time
Action	An action which the agent performs within the environment	Implementing a trading strategy φ
Reward	The consequence of the actions taken by the agent	The current portfolio value of the strategy φ

TABLE 3.1: Reinforcement learning components for deep hedging

3.2 Quadratic Hedging

A natural criteria for measuring the hedging loss follows from the utilisation of a quadratic function. The theory of quadratic hedging was proposed by Föllmer and Sondermann, 1986 and further popularised by Schweizer, 1999 and his preceding work, where a plethora of beneficial properties arise under this framework. Throughout this chapter, we explore the two methods of quadratic hedging, with particular focus on mean-variance hedging. We begin by utilising martingale theory to obtain solutions before concluding with a mean-variance dynamic programming solution.

A trader may wonder why a quadratic hedging criterion is beneficial. The first reason follows the tractability of the approach in many frameworks. Throughout the remainder of this chapter, we will provide relatively general solutions to the quadratic hedging approach, with an emphasis placed on the dynamic programming approach covered in 3.2.3. The mean-variance dynamic programming solution provides a (semi-) tractable recursive algorithm which we use as a benchmark for the approximation accuracy of the neural network approach.

From an economical perspective, the quadratic hedging criteria assumes that the trader is neither the buyer or writer of the contingent claim at the time of optimisation. An example of this in application is the common scenario when a broker is required to provide a price quote for a client without the knowledge of whether the client is a buyer or seller. Furthermore, when a trader's information is known, a quadratic criteria can often be slightly adjusted to make it optimal for their preferences. For a more detailed discussion on the choice of quadratic

hedging, and mean-variance hedging in particular, refer to Bertsimas, Kogan, and Lo, 2001.

Recall that incomplete markets prevent the existence of replicating strategies for all contingent claims. We reiterate the notion from the indifference pricing approach that the criterion used to measure the risk of a strategy enables one to choose the optimal strategy which minimises that specific risk and that the associated valuation tool is a by-product of the solution.

As a precursor for what follows, we describe the main difference between the two quadratic hedging approaches. The risk-minimisation method aims to minimise the cost associated with a non-self-financing trading strategy which guarantees that the hedging portfolio matches the claim value at termination. The mean-variance approach continues to operate with self-financing trading strategies with the aim to minimise the quadratic loss between the hedging portfolio and the claim.

For the dual purpose of a reminder for the reader and an introduction of basic common notation and assumptions for the following sections, consider the following. Again, let X represent the discounted price process of the primary assets for the filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$. Let $\mathcal{Q}$ be the set of ELMMs for X and let $L(X)$ be the space of $\mathbb{R}^d$-valued X-integrable, predictable processes φ. The $L(X)$ space is important since it ensures that the gains process, captured by the stochastic integral $\int \varphi dX$, is well-defined for any $\varphi \in L(X)$. Additionally, note that φ generally denotes the entire strategy pair whilst ϑ denotes the risky positions. This notation coincides with that used in the beginning of chapter 2, hence our need to recall it, and more importantly the notation used in Schweizer, 1999. Now, we make the following assumptions:

$\mathcal{P} \neq \emptyset$ which prevents arbitrage by the FTAP I (theorem 1).

X is square-integrable.

The market is frictionless.

With the basic notation and assumptions introduced, we begin our discussion with the first quadratic hedging approach, risk-minimisation.

3.2.1 Risk-Minimisation

Our discussion of the risk-minimisation approach begins with the simplified case where X is a local-martingale with respect to the measure $\mathbb{P}$. We then briefly discuss the semi-martingale generalisation which gives rise to local risk-minimisation. A more detailed discussion of this generalisation goes beyond the scope of this dissertation. The interested reader is referred to Schweizer, 1999 and Föllmer and Schied, 2011 for such discussions. To this end, let us begin with more essential notation.

Let X be a local-martingale with respect to the real-world measure $\mathbb{P}$, let $\langle X \rangle = (\langle X^i, X^j \rangle)_{i,j=1,\dots,d}$ represent the matrix-valued covariance process of X, and let

$L^2(X)$ denote the space of $\mathbb{R}^d$-valued predictable processes ϑ such that

$$||\vartheta||_{L^2(X)} \coloneqq \left(\mathbb{E}\left[\int_0^T \vartheta_u^{Tr} d\langle X \rangle_u \vartheta_u \right] \right)^{\frac{1}{2}} < \infty, \tag{3.8}$$

which is a square-integrability condition on the strategy ϑ. Schweizer, 1999 shows that $\int \vartheta dX$ is both well-defined and in the space of square-integrable $\mathbb{P}$-martingales null at 0, $\mathcal{M}_0^2(\mathbb{P})$, for any $\vartheta \in L^2(X)$.

Next, we define a set of strategies which will be used for the risk-minimisation task moving forward. As such, they are known as RM strategies.

Definition 3.2.1 (RM Strategy)
A pair $\varphi = (\eta, \vartheta)$ where $\vartheta \in L^2(X)$ and $\eta = (\eta_t)_{0 \leq t \leq T}$ is an $\mathbb{R}$ adapted process is said to be an RM strategy if the value process, $V(\varphi) = \vartheta \cdot X + \eta$, is right-continuous and square-integrable. In other words, $V_t(\varphi) \in L^2(\mathbb{P})$ for each $t \in [0, T]$.

An RM strategy does not impose the self-financing condition which has been prevalent in our entire mathematical finance discussion so far. Therefore, suppose that a trader is required to hedge a contingent claim H; where they utilise the space of RM strategies. This can be trivially solved by the strategy $\varphi_t = (\eta_t, 0)$ with $\eta_t = HI_{\{t=T\}}$. Such a strategy clearly replicates the claim and is permitted since η_t is simply an adapted process; however, the trader has not mitigated any of their risk. For the purposes of addressing this issue, consider the following definitions.

Definition 3.2.2
The (cumulative) cost process $\zeta(\varphi)$ of an RM strategy $\varphi = (\eta, \vartheta)$ is given by

$$\zeta_t(\varphi) \coloneqq V_t(\varphi) - \int_0^t \vartheta_u dX_u, \quad 0 \leq t \leq T. \tag{3.9}$$

Additionally, the risk process of φ is given by

$$R_t(\varphi) \coloneqq \mathbb{E}\left[\left(\zeta_T(\varphi) - \zeta_t(\varphi) \right)^2 | \mathcal{F}_t \right], \quad 0 \leq t \leq T. \tag{3.10}$$

The cost process can be viewed as the current portfolio value less the gains made from trading; therefore, $\zeta_t(\varphi)$ is the cost associated with the price fluctuations when implementing the strategy φ over the interval $[0, t]$. This does not consider the transaction costs which may occur. The risk process is then a measure of the risk associated with the strategy φ by incorporating the cost of ensuring replication. Note that if a claim H is attainable, then there exists a self-financing RM strategy, φ, such that $V_T(\varphi) = H$. This φ will have a constant cost process and a risk process of zero. For incomplete markets, a risk-minimisation process is required.

Definition 3.2.3 (Risk-minimising Strategy)
A risk-minimising RM strategy is a strategy which minimises the risk process

over the space of RM strategies. In other words, φ is risk-minimising if

$$R_t(\varphi) \leq R_t(\tilde{\varphi}), \quad \mathbb{P}\text{-}a.s\ \forall t \in [0,T], \tag{3.11}$$

for any RM strategy $\tilde{\varphi}$ such that $V_T(\tilde{\varphi}) = V_T(\varphi)$.

Whilst there are more detailed definitions of risk-minimising strategies, see Lemma 2.2 Schweizer, 1999, they are equivalent under our restrictive framework. Whilst RM strategies are not self-financing, the notion of self-financing is still important.

An RM strategy φ is known as a *mean-self-financing* strategy if the associated cost process $\zeta(\varphi)$ is a $\mathbb{P}$-martingale. The mean-self-financing notion can be viewed as a strategy which is self-financing on average.

Proposition 8
Any risk-minimising RM strategy is also mean-self-financing.

We are now able to discuss risk optimisation. Recall the Kunita-Watanabe decomposition. If we define the space $\mathcal{I}^2(X) \coloneqq \{\int \vartheta dX | \vartheta \in L^2(X)\}$, Schweizer, 1999 shows that $\mathcal{I}^2(X)$ is a stable subspace of $\mathcal{M}_0^2(\mathbb{P})$. Therefore, if we have a contingent claim H, such that $H \in L^2(\mathbb{P})$, the projection result of the Kunita-Watanabe decomposition theorem shows that H can be uniquely expressed by

$$H = \mathbb{E}(H|\mathcal{F}_0) + \int_0^T \vartheta_u^H dX_u + L_T^H, \quad \mathbb{P}\text{-a.s.} \tag{3.12}$$

for some $\vartheta^H \in L^2(X)$ and $L^H \in \mathcal{M}_0^2$ which is strongly orthogonal to $\mathcal{I}^2(X)$. This decomposition provides the risk-minimisation result under the martingale assumption.

Proposition 9
Suppose that X is a $\mathbb{P}$-local martingale. Then every contingent claim $H \in L^2(\mathcal{F}_T, \mathbb{P})$ admits a unique risk-minimising RM-strategy φ^* with $V_T(\varphi^*) = H$ $\mathbb{P}$-a.s. In terms of the decomposition in 3.12, φ^* is given by

$$\vartheta^* = \vartheta^H, \tag{3.13}$$
$$V_t^* = V_t(\varphi^*) = \mathbb{E}(H|\mathcal{F}_t), \quad 0 \leq t \leq T, \tag{3.14}$$
$$\zeta(\varphi^*) = \mathbb{E}(H|\mathcal{F}_0) + L^H. \tag{3.15}$$

Furthermore, the risky strategy ϑ^H is defined by the Radon-Nikodym derivative

$$\vartheta^H = \frac{d\langle V, X\rangle}{d\langle X\rangle}. \tag{3.16}$$

Next, let us consider the generalised problem where X is a semi-martingale such that

$$X = X_0 + M + A,$$

where X_0 is a constant, M is a local $\mathbb{P}$-martingale which is null at 0, and A is a finite variation predictable process which is null at 0. Schweizer, 1999 show that

a general risk-minimisation solution does not exist in such a case. The remainder of the risk-minimisation section is concerned with solving an slightly altered problem. Furthermore, proposition 9 does not hold for contingent claims which experience a payoff stream of more than one payoff. For the latter extension, refer to Møller, 2001.

Now let us discuss a variant of the above risk-minimisation problem. The risk process defined by 3.10 is sometimes called the *remaining conditional risk* since it is based on the change in the terminal and time t cost process. Let us now define the *local risk process*. We first define the process in discrete time in order to convey the notion clearly. The continuous-time version requires the brief definition below.

Definition 3.2.4
A small perturbation Δ is an L^2 strategy, (δ, ϵ), such that δ is bounded, the variation of $\int \delta dA$ is bounded and $\delta_T = \epsilon_T = 0$. For any sub-interval $(s,t]$ of $[0,T]$, the small perturbation is defined as

$$\Delta|_{(s,t]} \coloneqq \left(\delta I_{(s,t]}, \epsilon I_{[s,t)}\right).$$

Definition 3.2.5
In a discrete-time setting, the local risk process of an RM strategy φ is given by

$$R_t^{loc}(\varphi) \coloneqq \mathbb{E}\left[\left(\zeta_{t+1}(\varphi) - \zeta_t(\varphi)\right)^2 | \mathcal{F}_t\right], \quad 0 \leq t \leq T. \tag{3.17}$$

Consequently, φ is known as the local risk minimising strategy if

$$R_t^{loc}(\varphi) \leq R_t^{loc}(\tilde{\varphi}), \quad \mathbb{P}\text{-}a.s\ \forall t \in [0,T], \tag{3.18}$$

for any RM strategy $\tilde{\varphi}$ such that $V_{t+1}(\tilde{\varphi}) = V_{t+1}(\varphi)$.
The continuous-time analogy for the local risk process is given by

$$r^{\tau}(\varphi, \Delta) \coloneqq \sum_{t_i, t_{i+1} \in \tau} \frac{R_{t_i}\left(\varphi + \Delta|_{(t_i, t_{i+1}]}\right) - R_{t_i}(\varphi)}{\mathbb{E}\left[\langle M\rangle_{t_{i+1}} - \langle M\rangle_{t_i} | \mathcal{F}_{t_i}\right]} I_{(t_i, t_{i+1}]}, \tag{3.19}$$

where $\langle M\rangle$ is the predictable quadratic variation of X. A strategy φ is the locally risk-minimising strategy if

$$\liminf_{n\to\infty} r^{\tau_n}(\varphi, \Delta) \geq 0, \quad (\mathbb{P} \otimes \langle M\rangle)\text{-}a.e.\ on\ \ \Omega \times [0,T],$$

for any small perturbation Δ and every increasing sequence $(\tau_n)_{n\in\mathbb{N}}$ of partitions tending to the identity.

The local risk definition clearly concerns a short-term optimisation problem because the risk is measured by local cost changes. In contrast to the above risk-minimisation problem, we solve the local risk-minimisation problem under the generalisation that the process X is a semi-martingale.

Whilst leaving the details for the reader in Föllmer and Schied, 2011 and Schweizer, 1999, the local risk-minimisation process requires the characterisation of minimal martingale measures.

Definition 3.2.6
An equivalent martingale measure $\hat{\mathbb{P}} \in \mathcal{Q}$ is called a minimal martingale measure if

$$\mathbb{E}\left[\left(\frac{d\hat{\mathbb{P}}}{d\mathbb{P}}\right)^2\right] < \infty,$$

and if every $\mathbb{P}$-martingale $M \in \mathcal{H}^2$ which is strongly orthogonal to X is also a $\hat{\mathbb{P}}$-martingale.

Proposition 10 (Corollary 10.28, Föllmer and Schied, 2011)
There exists at most one minimal martingale measure.

Föllmer and Schied, 2011, and Schweizer, 1999, show that under a minimal martingale measure $\hat{\mathbb{P}}$ the optimisation problem again reduces to the Kunita-Watanabe decomposition as above.

Proposition 11
Let X be a semi-martingale with respect to $\mathbb{P}$ and let $\hat{\mathbb{P}}$ be a minimal martingale measure. If $H \in L^2(\mathbb{P})$, then under $\hat{\mathbb{P}}$, H admits a Kunita-Watanabe decomposition

$$H = \hat{\mathbb{E}}\left[H|\mathcal{F}_0\right] + \int_0^T \vartheta_u^{H,\hat{\mathbb{P}}} dX_u + L_T^{H,\hat{\mathbb{P}}}, \tag{3.20}$$

such that $\vartheta^{H,\hat{\mathbb{P}}} \in L(X)$ and $L^{H,\hat{\mathbb{P}}}$ is a local $\hat{\mathbb{P}}$-martingale null at 0, which is strongly $\hat{\mathbb{P}}$-orthogonal to X. Furthermore, $L^{H,\hat{\mathbb{P}}}$ is a local $\mathbb{P}$-martingale which is strongly $\mathbb{P}$-orthogonal to X.

The decomposition described by proposition 11 is known as the Föllmer-Schweizer decomposition. In combination with proposition 10, the Föllmer-Schweizer decomposition provides the solution to the local risk-minimisation problem.

Proposition 12
Suppose that X is a semi-martingale with respect to $\mathbb{P}$. If the contingent claim $H \in L^2(\mathcal{F}_T, \mathbb{P})$ admits a Föllmer-Schweizer decomposition with respect to a minimal martingale measure $\hat{\mathbb{P}}$; then, H has a unique locally risk-minimising RM-strategy φ^* given by

$$\vartheta^* = \vartheta^{H,\hat{\mathbb{P}}}, \tag{3.21}$$

$$V_t^* = V_t(\varphi^*) = \hat{\mathbb{E}}(H|\mathcal{F}_t), \quad 0 \leq t \leq T. \tag{3.22}$$

It can be shown that if X is a local $\mathbb{P}$-martingale, then the locally risk-minimising strategy is also given by the Kunita-Watanabe decomposition. Furthermore, the risk-minimising and local risk-minimising strategies coincide under this simplification, proposition 10.34 Föllmer and Schied, 2011.

Before advancing to the mean-variance hedging approach, we make a more elaborate remark about the claim valuation based on the optimal hedging strategy. As mentioned previously, the valuation of H via an optimal hedging strategy is a by-product of the hedging process. Furthermore, $V_t(\varphi^*)$ can be viewed as the value and/or price of the claim H at time t; however, this intrinsic value is based on the subjectively chosen, quadratic criteria. Therefore, another trader with a different criteria need not agree on this intrinsic value process for H. The traders' intrinsic value for H is also unlikely to be the observed market price for the claim if it exists. This is an important distinction for the upcoming mean-variance hedging discussions and results in the following chapter.

3.2.2 Mean-Variance Hedging

Now, let us shift our focus to the mean-variance hedging (MVH) approach which originated in Bouleau and Lamberton, 1989 and was popularised by Schweizer, 1999, amongst others. Again, we start by solving the optimisation problem under the simple case where X is a local $\mathbb{P}$-martingale; after which, we explore the solution for the generalised case where X is a semi-martingale. Contrary to the risk-minimising approach, the MVH framework restricts itself to the space of self-financing strategies. A detailed comparison of the two approaches is explored by Heath, Platen, and Schweizer, 2001b.

Recall the cost process, $\zeta(\varphi)$, from definition 3.2.2. It is clear that any self-financing strategy is constant $\mathbb{P}$-a.s. In fact, we know that a self-financing strategy which replicates a contingent claim is only available if that contingent claim is attainable. Unsurprisingly, the measure of risk imposed under the MVH framework is the squared error between the claim and the value process at the maturity time T. Mathematically, let $\varphi = (\eta, \vartheta)$ be a self-financing strategy such that $\eta := V_0 + \int \vartheta dX - \vartheta \cdot X$. The replication error, $H - V_T(\varphi)$, is the time T loss experienced by a trader who pays the claim H whilst trading with the strategy φ. MVH attempts to minimise the expected square replication error.

As a mathematical link to risk-minimisation, recall the risk process defined in 3.2.2. Let the mean-variance cost process be defined as

$$\zeta_t^{mv}(\varphi) := \begin{cases} H - V_0 - \int_0^T \vartheta_u dX_u, & t = T, \\ 0, & 0 \leq t < T. \end{cases}$$

Then $\zeta_T^{mv}(\varphi)$ represents the cost of the RM strategy where: $H = V_T$, the initial capital is V_0, and the risky strategy is ϑ. Then $R_0(\varphi)$ is the expected square replication error and we need only minimise R_0 instead of minimising the entire process as with the risk-minimisation approach. Note, that we need not redefine the cost process ζ with ζ^{mv}. Instead one can consider the RM strategy where

$$\eta_t(\varphi) := \begin{cases} \int_0^t \vartheta_u dX_u - \vartheta_t \cdot X_t, & 0 \leq t < T, \\ H - \int_0^T \vartheta_u dX_u, & t = T. \end{cases}$$

Then, the risk process and minimisation process is equivalent.

More traditionally, the MVH problem is to find the self-financing strategy φ^* which minimises

$$\mathbb{E}\left[(H - V_T(\varphi))^2\right] = \left|\left|H - V_0 - \int_0^T \vartheta_u dX_u\right|\right|^2_{L^2(\mathbb{P})} \tag{3.23}$$

over all self-financing trading strategies. This has a far more mathematical intuition than the one provided by the above risk-minimisation analogy. Let $H \in L^2(\mathbb{P})$ and let Θ denote the space of strategies which are allowed for trading. The space Θ is discussed further on a case-by-case basis. The quadratic criterion ensures that the optimal strategy is obtained from the projection of H on to the space of attainable claims. The attainable claims space, $\mathcal{A}$, is given by the linear space spanned by $L^2(\mathcal{F}_0, \mathbb{P})$ for the initial capital and stochastic integrals given by the gains process such that

$$\mathcal{A} \coloneqq \mathbb{R} + \mathcal{G} = \left\{ V_0 + \int_0^T \vartheta_u dX_u \middle| (V_0, \vartheta) \in \mathbb{R} \times \Theta \right\},$$

where

$$\mathcal{G} \coloneqq G_T(\Theta) = \left\{ \int_0^T \vartheta_u dX_u \middle| \vartheta \in \Theta \right\}.$$

The solution to the L^2 projection of H requires that the space $\mathcal{A}$ is a closed subspace of $L^2(\mathbb{P})$; ensuring both the existence and uniqueness of this projection. It is clear that the issue of closedness of $\mathcal{A}$ depends on the space Θ.

In the case where X is a local $\mathbb{P}$-martingale, the Kunita-Watanabe decomposition directly provides us with the solution to this projection problem as before. For this martingale case, the space Θ only requires that $\vartheta \in L^2(X)$. In particular, we provide the solution in the following proposition.

Proposition 13
Suppose that X is a $\mathbb{P}$-local martingale and let $H \in L^2(\mathcal{F}_T, \mathbb{P})$ be a contingent claim. Then H can be decomposed as 3.12. Furthermore, since L^H is strongly orthogonal to $\mathcal{I}^2(X)$, the mean-variance optimal strategy for H is given by φ^* such that

$$V_0^* = \mathbb{E}[H|\mathcal{F}_0], \tag{3.24}$$
$$\vartheta^* = \vartheta^H. \tag{3.25}$$

Moreover, the mean-variance optimal strategy agrees with the risk-minimising and local risk-minimising strategy under the same conditions.

Next, let us consider the generalisation whereby X is a semi-martingale rather than a local $\mathbb{P}$-martingale. Again, we begin with essential definitions and assumptions. Define the space

$$\mathcal{P}_e^2 \coloneqq \left\{ \mathbb{Q} \in \mathcal{Q} \middle| \frac{d\mathbb{Q}}{d\mathbb{P}} \in L^2(\mathbb{P}) \right\} \subseteq \mathcal{Q}$$

of the square-integrable density ELMMs. Throughout the remainder of the

section, we make the assumptions that X is continuous, $\mathcal{P}_e^2 \neq \emptyset$, and $\mathcal{F}_0$ is trivial.

In a similar procedure to the local risk-minimisation solution, the mean-variance optimal strategy follows from the Kunita-Watanabe decomposition under a certain measure. For MVH, the *variance-optimal measure* is this associated measure; however, before we can define this measure, we must define Θ.

There are two main choices for Θ which exist in the MVH literature:

1. Gourieroux, Laurent, and Pham, 1998 define the space $\Theta' := \{\vartheta\}$ such that:
   (a) $\vartheta \in L(X)$.
   (b) $\int_0^T \vartheta_t dX_t \in L^2(\mathbb{P})$.
   (c) $\int \vartheta dX$ is a $\mathbb{Q}$-martingale for each $\mathbb{Q} \in \mathcal{P}_e^2$.
2. Rheinländer and Schweizer, 1997 define the space $\tilde{\Theta} := \{\vartheta\}$ such that:
   (a) $\vartheta \in L(X)$.
   (b) $\int \vartheta dX \in \mathcal{S}^2$, where $\mathcal{S}^2$ is the space of square-integrable semi-martingales,

$$\mathcal{S}^2 := \left\{ X = X_0 + M + A \,\middle|\, M \in \mathcal{M}_{loc}^2, \mathbb{E}\left(\left(\int_0^T |dA|_s \right)^2 \right) < \infty \right\}.$$

Whilst the space $\tilde{\Theta}$ is perceived as a more natural extension of the case where $\mathbb{P} \in \mathcal{Q}$; Θ' is shown, by Rheinländer and Schweizer, 1997, to contain $\tilde{\Theta}$ and if $G_T(\tilde{\Theta})$ is closed in L^2, then $\tilde{\Theta} = \Theta'$. Consequently, our choice for $\Theta := \Theta'$ for the semi-martingale case is motivated by its generality and flexibility. With a choice for Θ, we can now define the variance-optimal measure as follows.

Definition 3.2.7
The measure $\tilde{\mathbb{P}}$ is known as the variance-optimal measure if it is the unique $\mathbb{Q} \in \mathcal{P}_e^2$ which minimises

$$\left|\left| \frac{d\mathbb{Q}}{d\mathbb{P}} \right|\right|_{L^2(\mathbb{P})} = \sqrt{1 + Var\left(\frac{d\mathbb{Q}}{d\mathbb{P}} \right)}$$

over all $\mathbb{Q} \in \mathcal{P}_e^2$.

Furthermore, Gourieroux, Laurent, and Pham, 1998 show that the process

$$\tilde{Z}_t := \tilde{\mathbb{E}}\left(\frac{d\tilde{\mathbb{P}}}{d\mathbb{P}} \middle| \mathcal{F}_t \right) = \tilde{Z}_0 + \int_0^t \xi_u dX_u,$$

for some $\xi \in \Theta'$, which is an important process for the MVH solution seen in the upcoming proposition 14.

Now, given a claim $H \in L^2(\mathbb{P})$, the Cauchy-Schwartz inequality shows that $H \in L^1(\tilde{\mathbb{P}})$[3]. Under the variance-optimal ELMM $\tilde{\mathbb{P}}$, X is a continuous local martingale and consequently, H admits the Kunita-Watanabe decomposition such that

$$H = \tilde{\mathbb{E}}(H) + \int_0^T \vartheta_t^{H,\tilde{\mathbb{P}}} dX_t + L_T^{H,\tilde{\mathbb{P}}}, \tag{3.26}$$

with

$$\begin{aligned} V_t^{H,\tilde{\mathbb{P}}} :=& \tilde{\mathbb{E}}(H|\mathcal{F}_t), \\ =& \tilde{\mathbb{E}}(H) + \int_0^t \vartheta_u^{H,\tilde{\mathbb{P}}} dX_u + L_t^{H,\tilde{\mathbb{P}}}. \end{aligned}$$

This decomposition provides the optimal mean-variance optimal strategy stated by proposition 14 from Theorem 2.1 of Heath, Platen, and Schweizer, 2001b.

Proposition 14
Suppose that X is a semi-martingale with respect to $\mathbb{P}$, $H \in L^2(\mathbb{P})$, and H admits the Kunita-Watanabe decomposition in equation 3.26; then, H has a unique mean-variance optimal strategy given by

$$\begin{aligned} V_0^{mvo} &= \tilde{\mathbb{E}}(H), \\ \vartheta_t^{mvo} &= \vartheta_t^{H,\tilde{\mathbb{P}}} - \frac{\xi_t}{\tilde{Z}_t}\left(V_{t^-}^{H,\tilde{\mathbb{P}}} - \tilde{\mathbb{E}}(H) - \int_0^t \vartheta_u^{mvo} dX_u\right). \end{aligned}$$

As a final MVH result, we introduce the *minimal expected square replication error* attainable under the incomplete market described above.

Proposition 15
Define the density process

$$Z_t^{\tilde{\mathbb{P}}} := \hat{\mathbb{E}}\left(\frac{d\tilde{\mathbb{P}}}{d\mathbb{P}}\middle| \mathcal{F}_t\right)$$

and let $\varphi^{mvo} = (V_0^{mvo}, \vartheta^{mvo})$ be the mean-variance optimal strategy. The minimal expected square replication error (ESRE) of the claim H is given by

$$\mathbb{E}\left((H - V_T(\varphi^{mvo}))^2\right) = \mathbb{E}\left(\int_0^T \frac{Z_t^{\tilde{\mathbb{P}}}}{\tilde{Z}_t} d\langle L^{H,\tilde{\mathbb{P}}}\rangle_t\right).$$

Before concluding this section, we remark that finding the variance-optimal measure is still a difficult task in general. A special case, studied by Pham, Rheinländer, and Schweizer, 1998, solves this issue and makes the above results practical; however, we omit this special case here.

[3]Following from the fact that $d\tilde{\mathbb{P}}/d\mathbb{P} \in L^2(\mathbb{P})$

Whilst this section has covered the MVH solution provided by martingale theory; the following section discusses the solution provided by dynamic programming.

3.2.3 Dynamic Programming

In the previous sections, we have mentioned that the pricing and hedging problem can be reduced to a dynamic programming problem. Due to the structure of solving a dynamic programming problem, the solution is the optimal price and hedging strategy for a given discrete-time setup. For this reason, this section is dedicated to the solution of the discrete-time mean-variance hedging problem via dynamic programming; which serves as a baseline for the accuracy of the trading strategies in the following chapter. These solutions were originally provided in a similar vain by Bertsimas, Kogan, and Lo, 2001 and Černỳ, 2004.

Suppose that a trader faces the problem of mean-variance hedging, which is hedging in incomplete markets by minimising the expected square replication error under the real-world measure $\mathbb{P}$. Once again, define a trading strategy $\varphi \coloneqq (\varphi_1, \ldots, \varphi_T)$ and a contingent claim H. The trader must find the optimal strategy φ^* such that:

$$\varphi^* = \inf_{\varphi \in \Psi} \mathbb{E}\left[(V_T(\varphi) - H)^2\right], \tag{3.27}$$

where $V_t(\varphi)$ is the portfolio value at time t associated with the trading strategy φ. Here, Ψ is the space of trading strategies. On the surface, this appears to be a T-dimensional problem; yet, the mean-variance problem requires that Ψ is the set of all *self-financing* trading strategies. Hence, $V_T(\varphi) = V_0(\varphi) + \sum_{k=1}^{T} \varphi_k \Delta X_k$, where $V_0(\varphi)$ is the construction cost of the portfolio. Consequently, 3.27 in fact describes a $T+1$-dimensional problem

$$\inf_{\nu, \varphi \in \Psi} \mathbb{E}\left[(V_T(\varphi) - H)^2 | V_0(\varphi) = \nu\right]. \tag{3.28}$$

Now consider a dynamic programming approach to resolve the high dimensionality by restructuring into sub-problems. The value function at time t can be defined as the value of the expected square replication error conditioned on the information and portfolio value at time t if the optimal decisions, *trading strategies*, are made in future time steps.

$$\begin{aligned} J_T(\nu) &= \mathbb{E}\left[(V_T - H)^2 | \mathcal{F}_T, V_T = \nu\right] && (3.29)\\ &= \mathbb{E}\left[(\nu - H)^2 | \mathcal{F}_T, V_T = \nu\right] && (3.30)\\ &= (\nu - H)^2 && (3.31) \end{aligned}$$

and

$$J_t(\nu) = \inf_{\varphi_{t+1},\dots,\varphi_T} \mathbb{E}\left[(V_T - H)^2 | \mathcal{F}_t, V_t = \nu\right] \tag{3.32}$$

$$= \inf_{\varphi_{t+1},\dots,\varphi_T} \mathbb{E}\left[\left(\nu + \sum_{k=t+1}^{T} \varphi_k \Delta X_k - H\right)^2 | \mathcal{F}_t, V_t = \nu\right]. \tag{3.33}$$

Note that the terminal value function J_T does not have any decisions to make. This setup can easily be seen to reduce to the Bellman-Jacobi equation as follows:

$$J_T(\nu) = \mathbb{E}\left[(V_T - H)^2 | \mathcal{F}_T, V_T = \nu\right], \tag{3.34}$$

$$J_{T-1}(\nu) = \inf_{\varphi_T} \mathbb{E}\left[(\nu + \varphi_T \Delta X_T - H)^2 | \mathcal{F}_{T-1}, V_{T-1} = \nu\right] \tag{3.35}$$

$$= \inf_{\varphi_T} \mathbb{E}\left[\mathbb{E}\left((\nu + \varphi_T \Delta X_T - H)^2 | \mathcal{F}_T, V_T = \nu + \varphi_T \Delta X_T\right) | \mathcal{F}_{T-1}, V_{T-1} = \nu\right] \tag{3.36}$$

$$= \inf_{\varphi_T} \mathbb{E}\left[J_T(\nu + \varphi_T \Delta X_T) | \mathcal{F}_{T-1}, V_{T-1} = \nu\right], \tag{3.37}$$

$$J_{T-2}(\nu) = \inf_{\varphi_{T-1},\varphi_T} \mathbb{E}\left[(\nu + \varphi_{T-1}\Delta X_{T-1} + \varphi_T \Delta X_T - H)^2 | \mathcal{F}_{T-2}, V_{T-2} = \nu\right] \tag{3.38}$$

$$= \inf_{\varphi_{T-1}} \mathbb{E}\left[\inf_{\varphi_T} \mathbb{E}\left((\nu + \varphi_{T-1}\Delta X_{T-1} + \varphi_T \Delta X_T - H)^2 | \mathcal{F}_{T-1}, V_{T-1} = \nu + \varphi_{T-1}\Delta X_{T-1}\right) | \mathcal{F}_{T-2}, V_{T-2} = \nu\right] \tag{3.39}$$

$$= \inf_{\varphi_{T-1}} \mathbb{E}\left[J_{T-1}(\nu + \varphi_{T-1}\Delta X_{T-1}) | \mathcal{F}_{T-2}, V_{T-2} = \nu\right]. \tag{3.40}$$

Similarly, for any t in $0 \leq t < T, t \in \mathbb{N}$:

$$J_t(\nu) = \inf_{\varphi_{t+1}} \mathbb{E}\left[J_{t+1}(\nu + \varphi_{t+1}\Delta X_{t+1}) | \mathcal{F}_t, V_t = \nu\right]. \tag{3.41}$$

It is clear that the optimisation depends on the value of ν which is interpreted as the current portfolio value. But the above Bellman-Jacobi equations show that

$$\inf_{\nu,\varphi\in\Psi} \mathbb{E}\left[(V_T(\varphi) - H)^2\right] = \inf_{\nu} J_0(\nu), \tag{3.42}$$

assuming that $\mathcal{F}_0$ is trivial. Then ν represents the construction cost of the optimal hedging portfolio; and thus the fair price or intrinsic value of the contingent claim given by the mean-variance framework. Consequently, the $T+1$-dimensional problem of 3.28 has been reduced to $T+1$ many 1-dimensional sub-problems.

Now let us solve these sub-problems in a convenient manner. In particular, notice that each of the value functions J_t is quadratic. A proof by mathematical induction both simultaneously shows this fact and establishes a recursive

structure for the value function.

Proposition 16
The value function J_t is quadratic with respect to ν such that

$$J_t(\nu) = a_t(\nu - b_t)^2 + c_t, \tag{3.43}$$

where $a_t, b_t, c_t \in m\mathcal{F}_t$. In particular, at the terminal time

$$a_T = 1, \tag{3.44}$$
$$b_T = H, \tag{3.45}$$
$$c_T = 0. \tag{3.46}$$

For the non-terminal terms, first define p_t and q_t to assist the quadratic terms such that

$$p_t = \frac{\mathbb{E}[a_t b_t \Delta X_t | \mathcal{F}_{t-1}, V_{t-1} = \nu]}{\mathbb{E}[a_t \Delta X_t^2 | \mathcal{F}_{t-1}, V_{t-1} = \nu]}, \tag{3.47}$$
$$q_t = \frac{\mathbb{E}[a_t \Delta X_t | \mathcal{F}_{t-1}, V_{t-1} = \nu]]}{\mathbb{E}[a_t \Delta X_t^2 | \mathcal{F}_{k-1}, V_{t-1} = \nu]}. \tag{3.48}$$

Then the quadratic terms are defined by

$$a_t = \mathbb{E}[a_{t+1}(1 - q_t \Delta X_{t+1}) | \mathcal{F}_t, V_t = \nu], \tag{3.49}$$
$$b_t = \frac{1}{a_t}\mathbb{E}[a_{t+1}(b_{t+1} - p_t \Delta X_{t+1})(1 - q_t \Delta X_{t+1}) | \mathcal{F}_t, V_t = \nu], \tag{3.50}$$
$$\begin{aligned} c_t = {} & \mathbb{E}[c_{t+1} | \mathcal{F}_t, V_t = \nu] - a_t b_t^2 \\ & + \mathbb{E}[a_{t+1}(b_{t+1} - p_t \Delta X_{t+1})^2 | \mathcal{F}_t, V_t = \nu], \end{aligned} \tag{3.51}$$

for $t \in \{0, 1, \ldots, T-1\}$.

Proof. This proof follows backwards mathematical induction.
Base case: Let $t = T$, then

$$\begin{aligned} J_T(\nu) &= \mathbb{E}\left[(\nu - H)^2 | \mathcal{F}_T, V_T = \nu\right] \\ &= (\nu - H)^2 \\ &= a_T\left(\nu - b_T\right)^2 + c_T, \end{aligned}$$

where $a_T = 1, b_T = H, c_T = 0$; hence, $a_T, b_T, c_T \in m\mathcal{F}_T$. Therefore, the base case is clearly quadratic with respect to ν.
General case: Assume that J_t is quadratic for $t = k$; then $\exists\ a_k, b_k, c_k \in m\mathcal{F}_k$ such that:

$$J_k(\nu) = a_k\left(\nu - b_k\right)^2 + c_k.$$

Now we must show that this remains true for $t = k - 1$.

$$\begin{aligned} J_{k-1}(\nu) &= \inf_{\varphi_k} \mathbb{E}\left[J_k(\nu + \varphi_k \Delta X_k)|\mathcal{F}_{k-1}, V_{k-1} = \nu\right] \\ &= \inf_{\varphi_k} \mathbb{E}\left[a_k\left(\nu + \varphi_k \Delta X_k - b_k\right)^2 + c_k|\mathcal{F}_{k-1}, V_{k-1} = \nu\right]. \end{aligned}$$

Note that the $\mathbb{E}\left[J_k(\nu + \varphi_k \Delta X_k)|\mathcal{F}_{k-1}, V_{k-1} = \nu\right]$ is actually convex with respect to φ_k since a_k is always positive. We can then simply find the optimal strategy φ_k^* by differentiating and setting to zero as follows:

$$\begin{aligned} &\frac{\partial}{\partial \varphi_k} \mathbb{E}\left[J_k(\nu + \varphi_k \Delta X_k)|\mathcal{F}_{k-1}, V_{k-1} = \nu\right] \\ &= \mathbb{E}\left[\frac{\partial}{\partial \varphi_k} a_k\left(\nu + \varphi_k \Delta X_k - b_k\right)^2 + c_k|\mathcal{F}_{k-1}, V_{k-1} = \nu\right] \\ &= \mathbb{E}\left[2a_k\left(\nu + \varphi_k \Delta X_k - b_k\right) \times \Delta X_k|\mathcal{F}_{k-1}, V_{k-1} = \nu\right] \\ &\overset{\text{set}}{=} 0. \end{aligned}$$

Since both φ_k and $\nu \in m\mathcal{F}_{k-1}$

$$\varphi_k^* = \underbrace{\frac{\mathbb{E}[a_k b_k \Delta X_k|\mathcal{F}_{k-1}]}{\mathbb{E}[a_k \Delta X_k^2|\mathcal{F}_{k-1}]}}_{p_k} - \nu \overbrace{\frac{\mathbb{E}[a_k \Delta X_k|\mathcal{F}_{k-1}]}{\mathbb{E}[a_k \Delta X_k^2|\mathcal{F}_{k-1}]}}^{q_k}. \tag{3.52}$$

Substituting back into the value function,

$$J_{k-1}(\nu) = \mathbb{E}\left[a_k\left(\nu + \varphi_k^* \Delta X_k - b_k\right)^2 + c_k|\mathcal{F}_{k-1}, V_{k-1} = \nu\right].$$

Finally, we desire that $J_{k-1}(\nu) = a_{k-1}\left(\nu - b_{k-1}\right)^2 + c_{k-1}$ for some $\mathcal{F}_{k-1}$-measurable random variables $a_{k-1}, b_{k-1}, c_{k-1}$. Through tedious, yet simple, manipulation of the above expression, we obtain that

$$a_{k-1} = \mathbb{E}[a_k(1 - q_{k-1}\Delta X_k)|\mathcal{F}_{k-1}, V_{k-1} = \nu], \tag{3.53}$$

$$b_{k-1} = \frac{1}{a_{k-1}} \mathbb{E}[a_k(b_k - p_{k-1}\Delta X_k)(1 - q_{k-1}\Delta X_k)|\mathcal{F}_{k-1}, V_{k-1} = \nu], \tag{3.54}$$

$$\begin{aligned} c_{k-1} = \mathbb{E}[c_k|\mathcal{F}_{k-1}, V_{k-1} = \nu] - a_{k-1}b_{k-1}^2 \\ + \mathbb{E}[a_k(b_k - p_{k-1}\Delta X_k)^2|\mathcal{F}_{k-1}, V_{k-1} = \nu], \end{aligned} \tag{3.55}$$

where the components p_{k-1} and q_{k-1} are defined in equation 3.52, and are clearly $\in m\mathcal{F}_{k-2} \subseteq m\mathcal{F}_{k-1}$. Hence, each of $a_{k-1}, b_{k-1}, c_{k-1} \in m\mathcal{F}_{k-1}$ and $J_{k-1}(\nu)$ is quadratic, as required.

Conclusion: Therefore, by backwards induction, the value

function J_t is quadratic for all $t \in [0, T]$ with recursive components a_t, b_t, c_t. □

With the proof of the above proposition, the following corollary becomes clear; forming an essential part of the algorithmic solution.

Corollary 17
The following are the optimal dynamic programming decisions.

a) The components of the optimal dynamic trading strategy φ_t^*, also known as the optimal control, is given by

$$\varphi_t^* = p_t - V_t q_t. \tag{3.56}$$

b) The initial cost which minimises the square replication error is given by

$$\nu^* = V_0^* = b_0. \tag{3.57}$$

Proof. a) Obtained directly from 3.52.
b) If the optimal trading strategy φ^* is implemented, then

$$J_0(\nu) = a_0(\nu - b_0)^2 + c_0$$

which represents the minimal replication error given some initial cost ν. Therefore, optimal initial cost is given by

$$\arg\min_{\nu} J_0(\nu). \tag{3.58}$$

Therefore, $\nu^* = b_0$ is clear.

□

Following the results of proposition 16 and corollary 17, the optimisation process of mean-variance hedging in discrete time is computationally convenient. With a discrete structure of future price movements, generally captured in a tree format, a trader is able to compute the recursive coefficients of the value functions throughout the life of a contingent claim which is then used to compute the initial cost required to implement the optimal hedging strategy under the mean-variance framework. Whilst the algorithm is computationally convenient, it is far from computationally efficient. Consequently, the described dynamic programming approach quickly becomes intractable; preventing its use despite its accuracy.

The dynamic programming solution offers an effective method to test the approximation accuracy of the deep hedging approach. A numerical analysis of such a test is presented in the following chapter.

Chapter 4

Results

With the problem description and theoretical solutions provided for pricing and hedging European contingent claims in an incomplete market; we aim to explore the performance by means of numerical application. Section 4.1 provides a description of the common components used throughout the following numerical results. Immediately following this, Section 4.2 provides a comparative benchmark for the proposed models' performances. The three benchmarks which have been chosen is the dynamic programming solution for a computationally feasible scenario and the Black-Scholes and Heston models. Section 4.3 then aims to provide a more robust test of neural network models through the use of real-world data and data generation.

4.1 Setup

In an attempt to provide a consistent comparison with Buehler et al., 2019, we use the same experiment setup, unless otherwise stated. In particular, the trader is presented with a contingent claim with a maturity of 30 trading days whereby they utilise a dynamic hedging strategy with daily rebalancing. Hence, $T = 30/365, n = 30$ and the trading times are denoted $t_k = k/365$ for $k = 0, 1, \ldots, n$. The only restriction which we have made on the contingent claim is that it is of a European type; leading us to consider an at-the-money vanilla call option as our base contingent claim for the upcoming experiments. The initial underlying asset price, $S_0 = 100$ unless otherwise stated. The neural networks architectures which we explore in this dissertation are the well-known recurrent and LSTM networks which are described in Section 4.1.2. First, however, we make a brief detour to discuss the simulation aspect.

4.1.1 Discrete Process Simulation

The market models described in Section 2.1.2 are established in a continuous-time framework; however, we require a discretised version for the purposes of a data stream. In fact, we only require the price process at the trading times t_k. A detailed discussion on discrete simulation of continuous time processes can be found in Glasserman, 2013; however, for the purposes of this dissertation, we recall the well-known Euler-Mayurama method as described by the following proposition from Maruyama, 1955.

Proposition 18
Consider the SDE under $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t,$$

where B *is a* $\mathbb{P}$ *Brownian motion and* $X_0 = x_0$*, over the interval* $[0, T]$*. Then the Euler-Mayurama approximation of the solution to* X *is given by the Markov chain* Y *defined as follows. Partition the interval* $[0, T]$ *into* n *equal sub-intervals. If we desire* n *sub-intervals then the length of each interval is clearly* $\Delta t = T/n$ *and* $0 = t_0 < t_1 < \cdots < t_n = T$*. Then let* $Y_0 = x_0$ *and*

$$Y_{m+1} = Y_m + \mu(Y_m, t_m)\Delta t + \sigma(Y_m, t_m)\Delta B_m, \quad 0 \leq m \leq n,$$

where $\Delta B_m = B_{t_{m+1}} - B_{t_m}$*. Note that each* $\Delta B_m \sim \mathcal{N}(0, \Delta t)$*.*

Utilising proposition 18 allows us to simulate the required asset price processes for both the Black-Scholes and Heston model from Section 2.1.2.

4.1.2 Network Methodology

In this section, we briefly describe the architecture of the neural network models which are implemented in the upcoming sections. With the idea of deep hedging presented in Section 3.1.2, a trader follows a trading strategy which is constructed by a neural network. That is, at each trading time t_k, the trader chooses $\varphi_k \in \mathbb{R}^d$ as her trading strategy; whereby, each φ_k is obtained from a neural network model such that $\varphi_k = f_\theta^{\mathcal{NN}}(\gamma(I_0, \ldots, I_k))$, where θ is the parameter set for the network, I_k is the information at time k, and γ represents some transformation of the information set which will depend on the model architecture. The set θ contains two types of parameters. The first type, *hyper-parameters*, defines the structure of network. The second type, *model parameters*, represents the matrices of the network. The latter type is optimised through well-known gradient descent algorithms when training the neural network. Our approach to the former type requires a brief outline.

Hyper-parameter Tuning

The hyper-parameters of a model are the parameters of the model which are set externally to define the model architecture prior to the training process. The most intuitive examples of hyper-parameters, and the parameters which we tune in the following experiments, are:

1. The number of layers.
2. The number of nodes for each layer.
3. The activation functions for each layer.

Each of these hyper-parameters has the potential to significantly affect the performance of a trained model; yet, there are no a-priori optimal values for these hyper-parameters. Consequently, finding the optimal hyper-parameters for a given neural network is a persistent and open problem within any machine

learning application. This procedure is known as *hyper-parameter tuning* which aims to optimise some performance measure of a trained model. Formally, let $\mathcal{X}$ denote the space of hyper-parameters and let $f : \mathcal{X} \to \mathbb{R}$ be a measure of model performance which represents an objective function. Then hyper-parameter tuning aims to find x^* such that

$$x^* = \arg\min_{x \in \mathcal{X}} f(x). \tag{4.1}$$

In general, f is not an analytical expression, one cannot compute the derivatives, and f is computationally expensive.

Additionally, the observations of the objective function are generally stochastic in nature. Consider $x \in \mathcal{X}$, then due to the numerical optimisation of neural networks, the observed performance of the network is not constant over repeated computations of $f(x)$. Let Y be the random variable associated with the observations $y = f(x)$, then $\mathbb{E}[Y] = f(x)$. Therefore, the optimisation problem defined by 4.1 is highly non-trivial and a few popular methods exist for this purpose. Notable methods are the *grid-search*, *random-search*, and *Bayesian optimisation* tuning methods. These hyper-parameter tuning methods aim to balance the accuracy of the optimisation over the space $\mathcal{X}$ and the computational resource available.

First, let us consider the grid-search method. Define the space $\mathcal{X}' \subseteq \mathcal{X}$ such that $\mathcal{X}'$ is discrete. The grid-search method computes $f(x')$ for all $x' \in \mathcal{X}'$. Hence, the method requires that $\mathcal{X}'$ must be defined such that

$$x^* \approx \arg\min_{x' \in \mathcal{X}'} f(x'),$$

whilst remaining below an acceptable computation time.

The random-search algorithm follows a similar level of complexity with a few tweaks. Let $(x_n)_{n=1}^N$ denote N elements from the space $\mathcal{X}$ sampled at random. The accuracy of the tuning algorithm is limited by the choice of N such that

$$x^* \approx \arg\min_{n \in \{1,\dots,N\}} f(x_n),$$

whereby N provides a direct constraint on the computation time required for the algorithm.

Now let us consider the more complex Bayesian optimisation approach proposed by Shahriari et al., 2015 which defines a probability model to approximate the objective function which can easily be optimised. The mathematical formulation is as follows. Again, let Y represent the random variable for the observations, also known as the score, of the computationally expensive objective function f. Now define $\pi(Y)$ as the prior probability model for the score and an associated posterior probability model as $\pi(Y|x)$ with data $\mathcal{D}$. Here, the posterior is known as the *surrogate model* of f and $\mathcal{D}$ refers to the pair of the observation of the objective function and its associated hyper-parameter query. Finally, define the

acquisition function, $\alpha : \mathcal{X} \to \mathbb{R}$ such that

$$\alpha(x|\mathcal{D}) = \mathbb{E}[U(x, y, \mathcal{D})],$$

for some utility function which measures the amount of information with respect to the tuning process that the query x will provide. Hence, the acquisition function provides a measure of the expected information gained for a query x which can be used to choose new query points.

The Bayesian optimisation algorithm is as follows:

1. Initialise the surrogate prior distribution.
2. Choose hyper-parameter query elements $x \in \mathcal{X}$ which maximise the acquisition function.
3. Retrieve an observation y from each query of the objective function.
4. Obtain the posterior distribution which serves as the prior for the next iteration.
5. Repeat steps 2-4 for some number of iterations.
6. Appropriately optimise the final surrogate model to obtain the optimal query x^*_{BO} such that $x^*_{BO} \approx x^*$.

The choice of the prior distribution, data distribution and acquisition function allow for variants of the algorithm; however, Gaussian processes for the prior distribution is a common and convenient choice. For interested readers, refer to Shahriari et al., 2015 for more details.

In our experiments that follow, we utilise the Bayesian optimisation method for the hyper-parameter tuning method. The space $\mathcal{X}$ is defined on an experiment-by-experiment basis. For a computational resource restriction, we sample 50 query points for each of the network architectures and obtain the objective function after training for 100 epochs, unless otherwise stated. In order to prevent a bias for the limited training time, a constant learning rate per tuning process is used with the *Adam* optimiser, Kingma and Ba, 2014, which is a common choice of an extended stochastic gradient descent algorithm. A default learning rate of 0.001 is used unless otherwise stated. After narrowing down the hyperparameter queries, the best five queries are fully trained, and the final model is taken as the best performing network on the validation set.

Unless otherwise stated, 10^5 samples, with a 20% validation split, are used for training with batch sizes of 1500 and 10^4 test samples in accordance with Buehler et al., 2019. Furthermore, the final training of the models allows the models to train until they reach an early stopping criterion. Unless otherwise stated, the training reaches its stopping criterion if the network does not improve on its validation loss for 15 epochs. With a batching size of 1500, this is equivalent to 54 weight updates per epoch; providing the network sufficient time to escape a plateau. In addition, the learning rate is reduced by 10% if the validation error does not improve over 5 epochs to assist the optimisation

of the weights. With an overview of the parameter optimisation procedure, we take a closer look at the two model variants.

Model Architectures

As alluded to in the neural network formalisation of 2.2, the neural network architectures for the experiments follow the canonical RNN and LSTM structures with minor tweaks. We define these formally below.

Note that the machine learning models are implemented with Tensorflow in Python with GPU parallelisation. Each of these models have batch normalised versions which applies the batch normalisation technique to each layer for the purpose of better training on large datasets.

Recurrent Neural Network

The first network structure discussed is the deep RNN (DRNN) model. Recall the definition of the canonical RNN from 2.2.2 such that

$$\begin{aligned}\varphi_n &= f^{\mathcal{NN}}(g(h_{n-1}, I_n)) \\ &= f^{\mathcal{NN}}(h_n),\end{aligned}$$

where h_n is the hidden state and I_n is the information at time n.

A DRNN model stacks several RNN components together before returning the network output. Suppose that the DRNN stacks L recurrent layers. Each recurrent component is known as a *cell* and they are joined as follows:

$$\begin{aligned}h_n^\ell &= g^\ell(h_{n-1}^\ell, h_n^{\ell-1}), \\ h_n^1 &= g^1(h_{n-1}^1, I_n), \\ \varphi_n &= f^{\mathcal{NN}}(h_n^L),\end{aligned}$$

where g^ℓ and h^ℓ represent the activation function and hidden state of the ℓ^{th} cell respectively. The architecture can be visualised by Figure 4.1. A DRNN is able to preserve a wider variety of time dependent features through its several cells.

The feed-forward network $f^{\mathcal{NN}}$ is stacked above the final recurrent cell and transforms the final hidden state to the desired model output φ_n. Whilst freedom can be given to allow for a deep neural network here, canonical architectures simply use a single dense layer for $f^{\mathcal{NN}}$. Similarly, we found that a deep neural network did not perform significantly better than a dense layer for $f^{\mathcal{NN}}$; consequently restricting the $f^{\mathcal{NN}}$ to be a dense output layer for the DRNN architecture used throughout the results. This d-node output layer must produce the d-dimensional hedging strategy to implemented; therefore, a *ReLu* activation function is a common choice for the output layer. We make no such restriction and allow the hyper-parameter tuning process to decide from a variety of activation functions to be defined.

Long Short-Term Memory

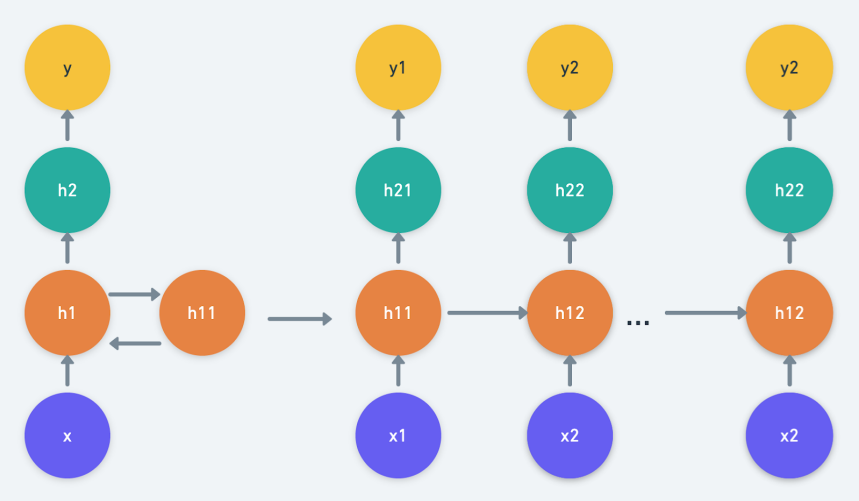


FIGURE 4.1: The architecture of a DRNN with two cells in recursive (left) and unrolled (right) format. Alla, 2021

Now let us consider a similar definition for a deep LSTM which forms the architecture for the second neural network model implemented throughout the experiments. Again denoting the recurrent component of the LSTM as an LSTM cell, a deep LSTM stacks, say L cells in the same format as above such that

$$\begin{aligned}(c_n^\ell, h_n^\ell) &= g^\ell(c_{n-1}^\ell, h_{n-1}^\ell, h_n^{\ell-1}),\\ (c_n^1, h_n^1) &= g^1(c_n^1, h_{n-1}^1, I_n),\\ \varphi_n &= f^{\mathcal{NN}}(h_n^L).\end{aligned}$$

Similarly to the DRNN model, we restrict the output transformation provided by $f^{\mathcal{NN}}$ to be a single layer feed-forward neural network. Furthermore, each LSTM cell has two types of activation functions in general, known as the activation function and the *recurrent* activation function. Recall, an LSTM cell has an input, forget, and output gate. The recurrent activation function is the activation function used for these gates whereas the activation function of an LSTM cell refers to the activation for the cell and hidden states directly. The activation functions proposed by Hochreiter and Schmidhuber, 1997 are *tanh* and *sigmoid* for the activation and recurrent activation respectively. In our experiments, we restrict the recurrent activation to this default function; however, we allow the activation function to vary.

With an architecture description, we can now consider the inputs for these neural networks explicitly.

The Information Set

The final component of the neural network methodology which we must discuss is the choice of the information set. The information set, denoted I_k, captures the information from the time t_k market information which represents the input

for the networks. The following discussion of the information set is motivated by Ruf and Wang, 2019.

When pricing or hedging options, a trader must clarify the goal of the network. In particular, will the model price and hedge a specific option or should the model be able to price and hedge a general option of a particular type. Suppose the former, such that a trader trains the network to hedge a call option with a specified maturity, strike price and initial stock price S_0; then, the network recognises the strike, and S_0 and maturity to a lesser degree, as some constant. The trader cannot expect the same network to hedge a call option with a different S_0 and strike without the appropriate changes. For the latter deployment, the trader must train the network to hedge call options with varying strikes and starting underlying prices; making both the underlying prices and the strike price integral components of the information/feature set. Including these two variables into the feature set can be achieved via two notable methods. The trivial solution is to represent both variables separately; yet, an alternative solution is to utilise the moneyness of the option, calculated as $M_t = S_t/K$ for the moneyness at time t. The latter method possesses several benefits over the former.

Firstly, Hutchinson, Lo, and Poggio, 1994 discuss the benefits of reducing the dimensionality of the feature set when using moneyness to represent the underlying price and the strike price simultaneously. Additionally, the option pricing function for parametric models are homogeneous of degree one with respect to the underlying price and the strike. This implies that moneyness is sufficient and in the neural network setting can reduce overfitting. Finally, moneyness has the potential to significantly improve the generalisation of the model for options which may vary in strike, Ruf and Wang, 2019. It is important to note, however, that these observations are based on pricing networks, of which the sensitivity can be used to provide the delta hedging strategy.

In addition to the underlying price and strike components, volatility measures could be considered. These volatility measures include historical volatility, volatility indexes and implied volatility; yet, many research contributions, including Buehler et al., 2019 do not include volatility in their feature sets.

Finally, we consider the feature set presented within Buehler et al., 2019. Here, $\log(S_t)$ is the only component of the feature set which is sufficient since they aim to only hedge a single specific option. The log transformation can be an important component depending on the setting. Such a transformation enforces stationarity of the underlying price paths. This is essential for our experiments in 4.3 which requires underlying price paths which do not share an initial price.

For the experiments below, we follow Buehler et al., 2019 in their choice to exclude any volatility measures within their feature set; however, we tweak their feature set regarding the underlying. Motivated by the potential generalisation of the model, we compare two feature sets. The first feature set is comprised of the pair (M_t, τ_t), where τ_t represents the time to maturity at time t. Such a choice of feature set follows the hedging model proposed by Ruf and Wang, 2021. The second feature set utilises the pair $(\log(M_t), \tau_t)$, exploiting the resultant

stationarity property. For convenience, we denote the feature sets as M and LM for moneyness and log moneyness respectively. The first feature set will only be used for the benchmarks for the reasons above. Furthermore, we do not explicitly test the generality of moneyness within this dissertation.

4.2 Benchmark

In the first numerical experiment section, we aim to measure the performance of the deep hedging models against known optimal and close-to-optimal hedging strategies. Such an experiment has the benefit of displaying the capability of deep hedging models to approximate the optimal hedging strategy; supporting the theoretical claim made by the problem description and the universal approximation theorem. For the benchmark section, we omit market frictions in order to compare the network models against the well-known, theoretically optimal strategies.

4.2.1 Dynamic Programming

Our first benchmark makes very restrictive assumptions on the market conditions for the purpose of creating a problem which is computationally tractable for the dynamic programming approach described in Section 3.2.3. The dynamic programming solution is generally evaluated via a tree structure; hence, our market assumptions are:

1. The price is only allowed to make discrete jumps between trading times.
2. The jump probabilities are time-homogeneous.

The tree of the primary asset price process can be defined mathematically. Consider a market with one primary risky asset, $d = 1$, and suppose that only three discrete jumps are possible. Let $\boldsymbol{J}_t = [j_1, j_2, j_3]^T$ denote the jump vector for time t. The jump vector defines the possible future prices by

$$X_{t+1} = X_t(1 + \boldsymbol{J}_t). \tag{4.2}$$

Associated with each of these jumps is a real-world probability, $\mathbb{P}$. Denote the probability vector at time t by $\boldsymbol{w}_t$ such that

$$\boldsymbol{w}_t = \Big[\mathbb{P}(\{\omega_{j_1}\}), \mathbb{P}(\{\omega_{j_2}\}), \mathbb{P}(\{\omega_{j_3}\})\Big]^T. \tag{4.3}$$

The assumptions made above require that $\boldsymbol{J}_t = \boldsymbol{J}_1$ and $\boldsymbol{w}_t = \boldsymbol{w}_1$ for each $t = 1, 2, \ldots, T$.

One can easily verify the dynamic programming solution by testing the algorithm within a complete market. For example, let $d = 1$ and let $\boldsymbol{J} \in \mathbb{R}^2$. Obviously; however, we want to use the solution to provide a benchmark in an

incomplete market. For this, we set the market conditions such that

$$\boldsymbol{J} = [1.05, 1.02, 1.0, 0.985, 0.95]^T, \tag{4.4}$$

$$\boldsymbol{w} = [0.10, 0.25, 0.30, 0.20, 0.15]^T \tag{4.5}$$

Whilst the tree structure is suitable for the benchmark, we need to sample paths from the tree to train and test the neural network models. Note that the sampling of such paths must be in accordance with the vector $\boldsymbol{w}$ since the network must learn to minimise the risk under the real-world measure $\mathbb{P}$.

For this benchmark, the neural network models are restricted to two hidden layers of variable size and activation functions. In particular, define

$$\mathcal{X} = \{(N_1, N_2, \sigma_1, \sigma_2, \sigma_{\text{out}}) : N_1, N_2 \in \mathfrak{N}; \sigma_1, \sigma_2 \in \Sigma_h; \sigma_{\text{out}} \in \Sigma_o\},$$

where $\mathfrak{N} = \{10, 30, 60\}$, $\Sigma_h = \{\tanh, \text{selu}, \text{elu}\}$ and $\Sigma_o = \{\tanh, \text{relu}, \text{swish}\}$.

With the dynamic programming benchmark setup, we can now analyse the deep hedging technique. To begin with, we model tuning process before evaluating the deep hedging performance. This is completed with both the M and LM-feature sets.

M-Feature Set

For the interested reader, the overall hyperparameter tuning procedure is summarised by the following Tensorboard. Within the Tensorboard, one can interactively view the training and validation performance of any of the queries $x \in \mathcal{X}$ as a function of the updating iterations or epochs under the `Scalars` tab[1]. An example of random queries from the tuning process can be seen in Figure 4.2. A large majority of the queries tested throughout the hyperparameter tuning process yield a similar validation loss performance after a few epochs, which can be seen on the examples in Figure 4.2. The final performance of each query is also tabulated under the `Hparams`.

Whilst the Tensorboard for the entire hyperparameter tuning process is rather overwhelming, the Tensorboard for the best performing models is intuitive and informative for those interested in the training of these final models. In particular, the above Tensorboard provides an in-depth look at the training and validation of the best models for each network type. Figure 4.3 shows the adaptive learning rates and the validation performance of the best networks for each of the model types respectively. It is clear from this figure that the performance of the networks does not significantly differ from each other when tested on the validation set. Furthermore, unlike the performance of 4.2, the log validation loss appears to have converged to either a local or global minima. We will soon see that this convergence is, *at least*, close-to-optimal. The optimal hyperparameter choices given by these models are given by Table 4.1.

[1]Tip: for the regex filter, *^RNN.*train* filters the training data for the RNN models. Adjust the model type and train and validation appropriately for desired plots.

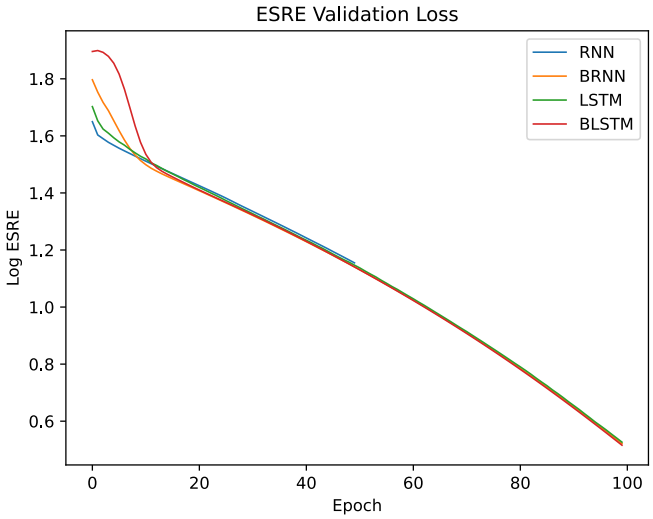


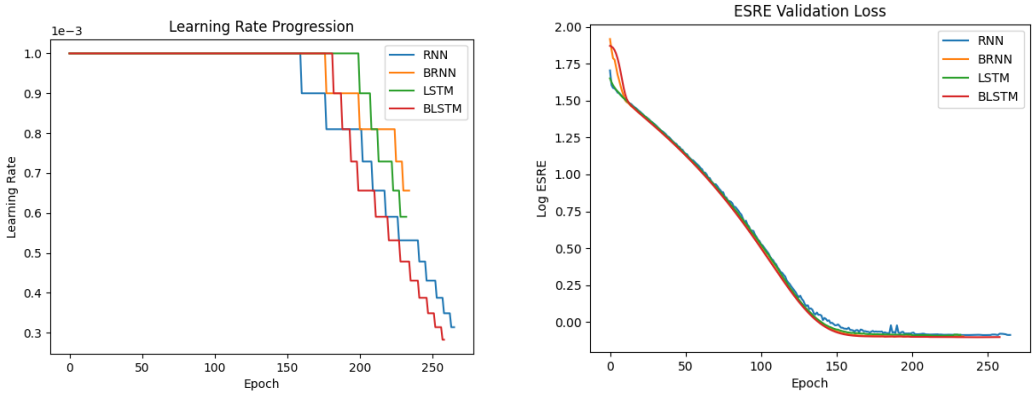
FIGURE 4.2: Example of the validation path of hyperparameter queries for the M dynamic programming benchmark

Hyperparameter	Optimal Value			
	RNN	Batch RNN	LSTM	Batch LSTM
N_1	60	10	60	30
N_2	60	60	60	60
σ_1	*elu*	*tanh*	*tanh*	*tanh*
σ_2	*tanh*	*tanh*	*elu*	*tanh*
σ_{out}	*tanh*	*tanh*	*tanh*	*swish*

TABLE 4.1: Optimal M-hyperparameters for neural network models for dynamic programming benchmark

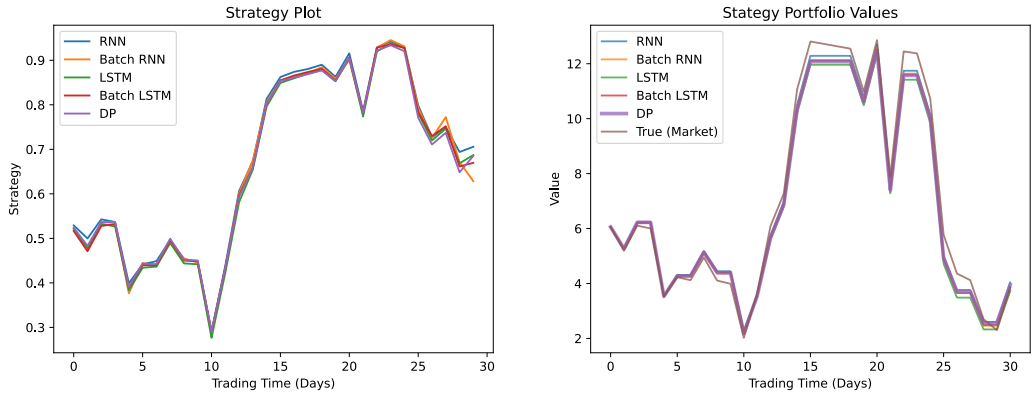
For the performance analysis, we begin by visualising the strategies and their associated hedging portfolios for a random test sample path given by each of these models against the optimal strategy computed via dynamic programming. These visualisation are found in 4.4. From Figure 4.4, it is clear that the recurrent based networks are able to achieve their reinforcement learning goal. Figure 4.4a shows that the trading actions chosen by the networks closely approximate the optimal decisions obtained from the dynamic programming approach without its drastic computational issues when scaling up the problem complexity. This fact is substantiated by Figure 4.4b which shows that the minor differences in trading decisions between all of the approaches yields largely the same hedging strategy. Whilst the true value of the option noticeably differs from the hedging portfolios, we reiterate that the DP approach is the optimal solution over the real-world measure and figure 4.4 is a visualisation of a single path only.

In order to evaluate the performance of the above approaches for the real-world measure, we display the histogram profit and loss (PnL) in figure 4.5 and the expected square replication errors tabulated in 4.2 below.



(A) Adaptive learning rate (B) Best validation loss per model type

FIGURE 4.3: Model fitting the best models for the M-feature set for the dynamic programming benchmark



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.4: The strategy and hedge for a random dynamic programming sample path

Figure 4.5 is relatively difficult to interpret clearly; but, this follows from the fact that each of the networks closely approximates the optimal solution as desired which is exacerbated by our current comparison of the batch and non-batch normalised model architectures. An important property which each histogram conveys is that the deep hedging approaches have a mean which is close to zero and a variance which does not appear significantly differ from that of the optimal dynamic programming solution. Numerical performance values are required to obtain a better analysis of the deep hedging quality. For this purpose, the benchmark Table 4.2 is far more informative. In particular, the dynamic programming approach obviously provides the lowest ESRE with the neural network approaches trailing close behind. This is a concise confirmation that the reinforcement learning problem can be well approximated by recurrent based neural networks, showing that the deep hedging approach is able to provide close-to-optimal hedge performance without the computational intractability of the optimal dynamic programming approach.

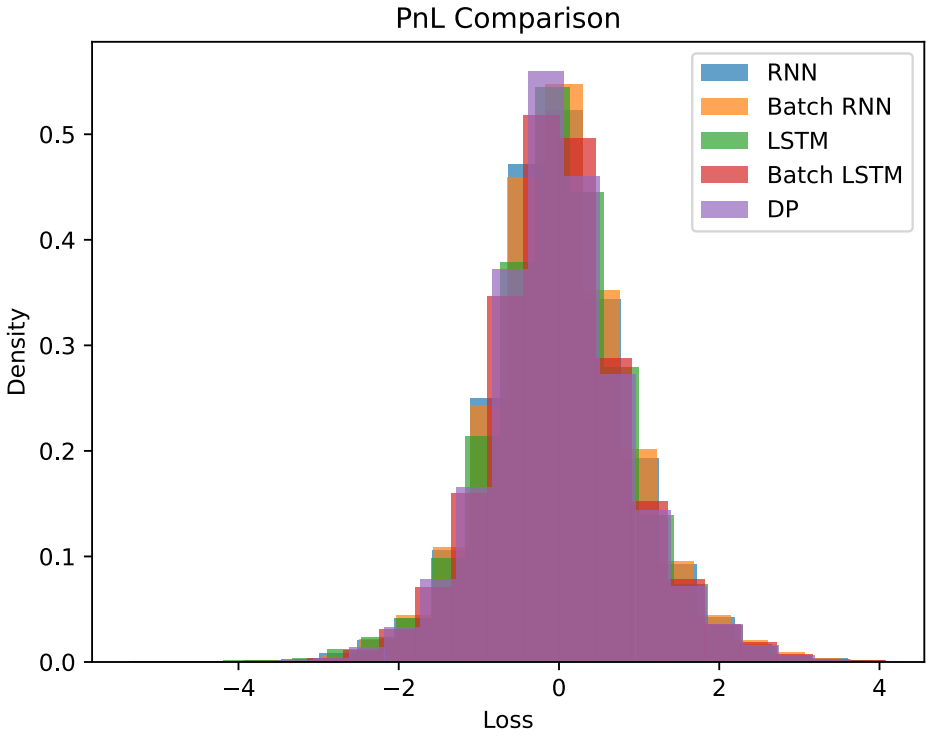


FIGURE 4.5: Profit and loss histogram for the dynamic programming benchmark

We take special note that the batch normalised versions of each network have a superior performance to those without. In the future experiments, we will treat batch normalisation as a hyperparameter but chose to show the difference explicitly within this benchmark for the purposes of a wide and robust comparison.

As mentioned, this benchmark has made use of the M-feature set for the network inputs. The dynamic programming benchmark clearly supports the notion that, **with appropriately constructed samples for training and testing purposes**, the use of the non-stationary moneyness input is sufficient to construct deep hedging models. Next, let us consider the LM-feature set for comparison.

LM-Feature Set

When changing the from the moneyness feature set to the log transformed moneyness feature set, the analysis of the deep hedging models for the dynamic programming benchmark follow closely from the M-feature set. This is an expected result based on the simulated data of the problem. As mentioned previously, the log transformation creates a stationary input which is an important property; however, the short time frame prevents stationarity from being an essential property.

The 200 query hyperparameter tuning process is summarised the following Tensorboard. For this tuning process, a higher learning rate of 0.01 was chosen

Approach	Expected Square Replication Error
Dynamic Programming	0.747
RNN	0.783
Batch RNN	0.762
LSTM	0.781
Batch LSTM	0.755

TABLE 4.2: Expected square replication error for dynamic programming benchmark

which resulted in a much quicker convergence of the networks. For example, Figure 4.6 shows the validation error of a random query from each model type. Compared to 4.2, such an analysis is clear. Furthermore, we can observe the

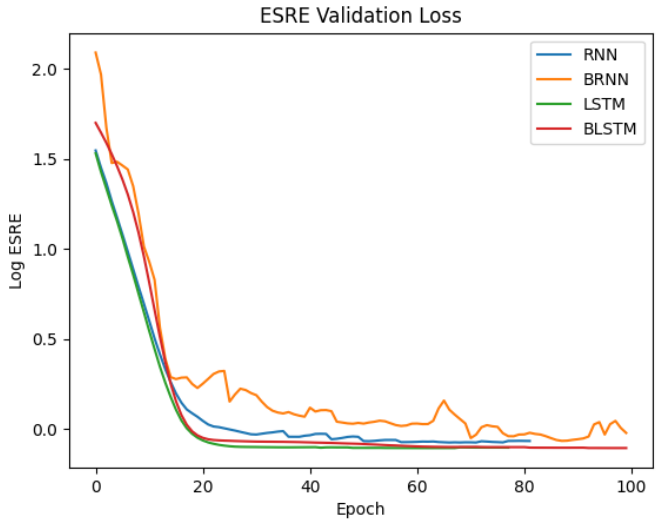
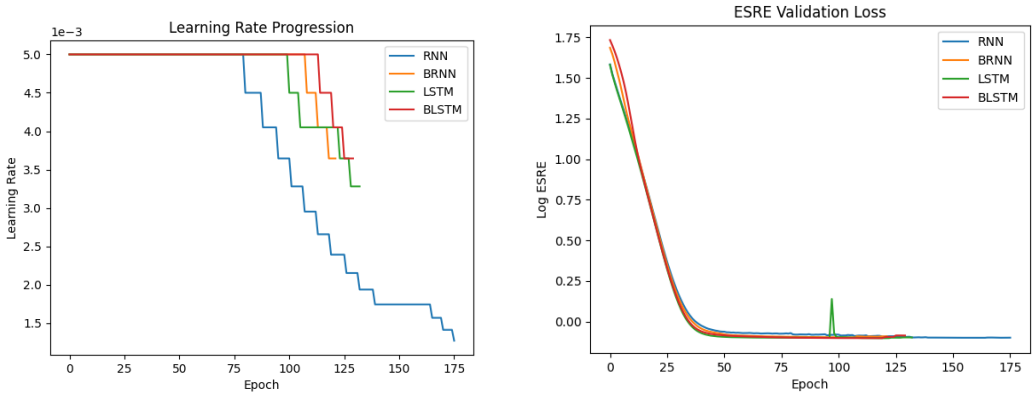


FIGURE 4.6: Example of the validation path of hyperparameter queries for the LM dynamic programming benchmark

difference in the ESRE on the validation set. We remark that the learning rate is adaptive and decreases when the network reaches a plateau in the validation loss which still allows for a sufficient search of minima for the purposes of a hyperparameter tuning guideline. The best queries are then trained at a learning rate of 0.005 with reduction on plateaus. For the training of these top performing queries, the reader can again refer to the following Tensorboard and can be visualised Figure 4.7b below. Again, we see a similar convergence of the models according to their ESRE on the validation set.The resulting optimal hyperparameters for each model is tabulated in 4.3 below. Whilst these optimal hyperparameter sets differ from those in 4.1, these differences are not necessarily significant. On one hand, the differences could follow since many of the hyperparameter queries result in very similar performance on the validation for both the M- and LM-feature sets. We further note that the higher learning rate for the hyperparameter tuning for the LM-feature set allows the Bayesian optimisation algorithm to observe more converged validation loss measures opposed to those of the M-feature set. Alternatively, the stationarity transformation could



(A) Adaptive learning rate

(B) Best validation loss per model type

FIGURE 4.7: Model fitting the best models for the M-feature set for the dynamic programming benchmark

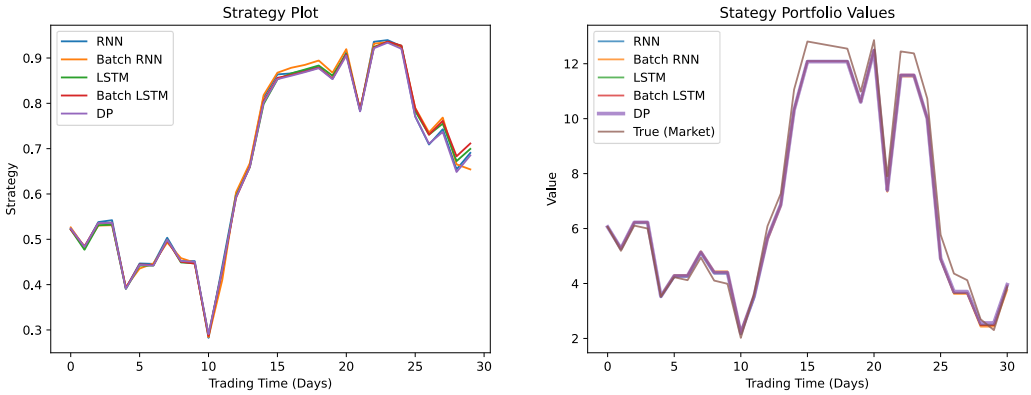
Hyperparameter	Optimal Value			
	RNN	Batch RNN	LSTM	Batch LSTM
N_1	10	10	60	60
N_2	10	10	10	10
σ_1	*tanh*	*tanh*	*elu*	*elu*
σ_2	*tanh*	*tanh*	*elu*	*tanh*
σ_{out}	*tanh*	*swish*	*tanh*	*swish*

TABLE 4.3: Optimal LM-hyperparameters for neural network models for dynamic programming benchmark

be reducing the complexity required to obtain the optimal performance; however, this single observation is not enough to deduce such a claim. We rather observe the hyperparameter differences throughout the remaining benchmarks.

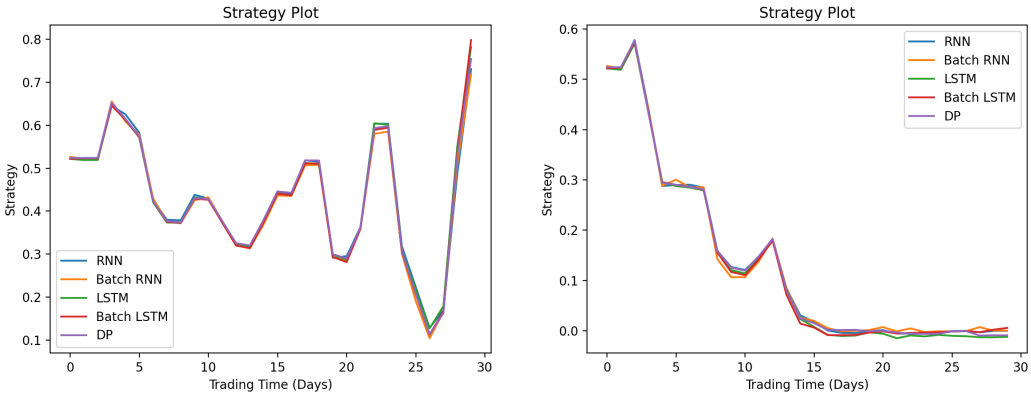
With respect to the hedging performance, the similarity of each model is clear from Figure 4.8 whereby the network hedging strategies are all close to optimal and the portfolio values are visually indistinguishable. We note that the visualisations provided by 4.4 and 4.8 arise from the same underlying sample path. An interesting observation surrounds the noticeable difference between the methods in the last five trading periods for both the M and LM-feature sets. If we view the strategy plot for a variety of other test sample paths, this phenomenon appears to be a trend; however, with varying degrees of magnitude. For example, refer to Figure 4.9. Here, the strategies shown in 4.9a appear to remain almost visually indistinguishable from the optimal solution whilst Figure 4.9b displays another example of a path where the final trading periods result in a larger difference between the network hedging strategies and the optimal solution. This is likely due to each input becoming frequent as the timeline progresses given the tree structure.

The overall performance of the hedging methods can be conveyed by histogram 4.10 of the PnL and the ESRE of each method tabulated in 4.4. Similar to the M-feature set, we find that the distributions of the LM PnLs are very similar



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.8: The strategy and hedge for a random dynamic programming sample path



(A) Sample path 22 (B) Sample path 13

FIGURE 4.9: The strategy and hedge for a random dynamic programming sample path

and we require Table 4.4 to make the inference on the performance differences of each method. Clearly, each of the networks do not significantly differ from the optimal ESRE, supporting the claim that deep hedging models can learn the optimal dynamic programming solution via reinforcement learning. Comparing the performance to the M-feature set table, 4.2, we find that the LSTM architecture remains to have a superior performance when compared to its direct RNN counterpart. On the other hand, the batch normalisation of the RNN network fails to outperform the non-batch normalised RNN architecture; however, the difference between the performance if minimal even when compared to those seen in Table 4.2. Regardless, the batch-normalised LSTM model remains the best performing model under both the M- and LM-feature sets.

The ability of deep hedging models to learn the optimal hedging strategy for the dynamic programming method under both the M- and LM-feature sets is clear. With that said, the LM-variant proves to be less restrictive with regard to how the financial data is generated due to its stationarity transformation.

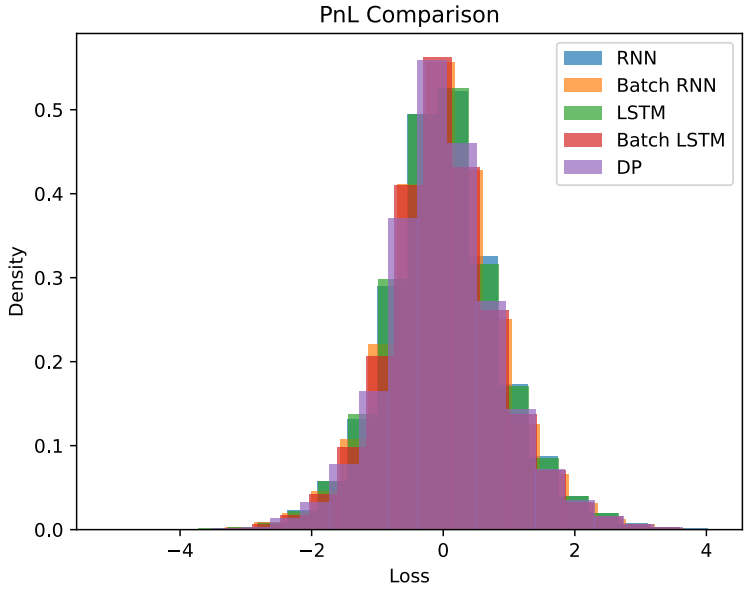


FIGURE 4.10: Profit and loss histogram for the dynamic programming LM-benchmark

Approach	Expected Square Replication Error
Dynamic Programming	0.747
RNN	0.761
Batch RNN	0.765
LSTM	0.755
Batch LSTM	0.753

TABLE 4.4: Expected square replication error for dynamic programming LM-benchmark

The benchmark performed above motivates the use of neural networks to approximate optimal hedging strategies. That said, the framework for this benchmark is extremely restrictive and far from the real world application of deep hedging. To generalise and advance towards the real world application, we consider a second benchmark based on the discretisation of the continuous-time Black-Scholes model.

4.2.2 Black-Scholes

Throughout this second performance benchmark we explore delta hedging in the Black-Scholes model. Despite the restrictive assumptions made by the Black-Scholes model, discussed in chapter 2.1.2, the model provides a very competitive performance in the real world and is an important benchmark for the industry. As a result, it is only natural to consider the Black-Scholes model as a benchmark for the neural network models. Whilst the Black-Scholes model is complete in continuous time, the market is incomplete when discretisation is implemented. Furthermore, discretising the continuous-time hedging solution

does not preserve the optimality in general; however, within the setup of this benchmark, the Black-Scholes delta hedging strategy closely approximates the optimal solution. Hence, it provides a great benchmark to measure the neural network models' performances.

For the simulation of the Black-Scholes market data, we choose the following market conditions:

$\mu = 0.10.$

$r_f = 0.0$ (w.l.o.g.).

$\sigma = 0.20.$

with the shared conditions stated in 4.1. More explicitly, the underlying asset prices are simulated from the Euler-Mayurama discretisation under the real-world measure such that

$$dS_t = \mu S_t dt + \sigma S_t dW_t,$$

which has risk-neutral dynamics

$$dS_t = r_f S_t dt + \sigma S_t dW_t.$$

The risk-neutral measure is used to compute the delta hedging strategies which represents the benchmark. This follows since these dynamics provide the best possible Black-Scholes delta hedging strategy for the Black-Scholes model with respect to the unknown volatility parameter, appropriate for a benchmark. We state this fact since this is **not** an entirely realistic choice. Whilst the goal of market calibration is to find the optimal market model parameters, the true parameters are neither known nor exactly obtainable in application.

Based on the fact that the number of optimal nodes per layer for the dynamic programming benchmark was commonly the upper bound of 60, we broaden the hyperparameter space $\mathcal{X}$ with the choice of a third hidden layer. In particular,

$$\begin{aligned}\mathcal{X} = \{(L, N_1, \dots, N_L, \sigma_1, \dots, \sigma_L, \sigma_{\text{out}}) : L \in \{2, 3\}; N_\ell \in \mathfrak{N};\\ \sigma_\ell \in \Sigma_h; \sigma_{\text{out}} \in \Sigma_o \ \text{ for } \ell = 1, \dots, L\},\end{aligned}$$

where $\mathfrak{N} = \{10, 30, 60\}$, $\Sigma_h = \{\tanh, \text{selu}, \text{elu}\}$ and $\Sigma_o = \{\tanh, \text{relu}, \text{swish}\}$. Here, we simply want to provide the networks with more flexibility by allowing, but not enforcing a third hidden layer.

M-Feature Set

Again, the Tensorboard of the overall hyperparameter tuning procedure is available. Figure 4.11 displays an example hyperparameter query for each model. As with the previous benchmark, we find that most queries produce similar validation plots. We further note that the networks are close to convergence within the 100 epoch restriction for the hyperparameter tuning algorithm. When observing the `Parallel Coordinates View` under the `Hparams` tab we find that

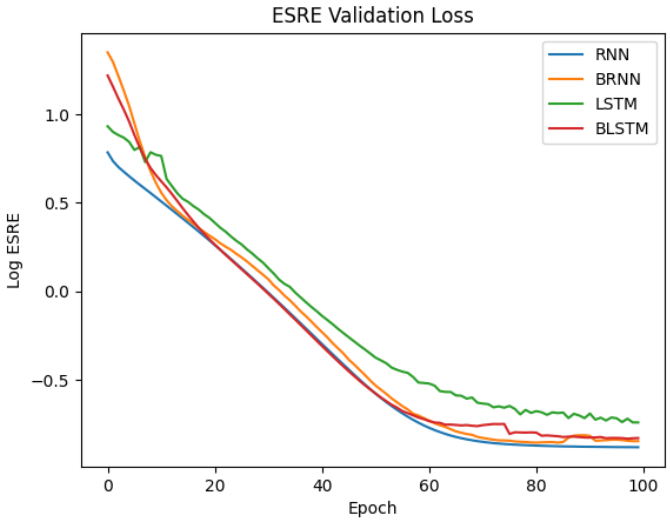
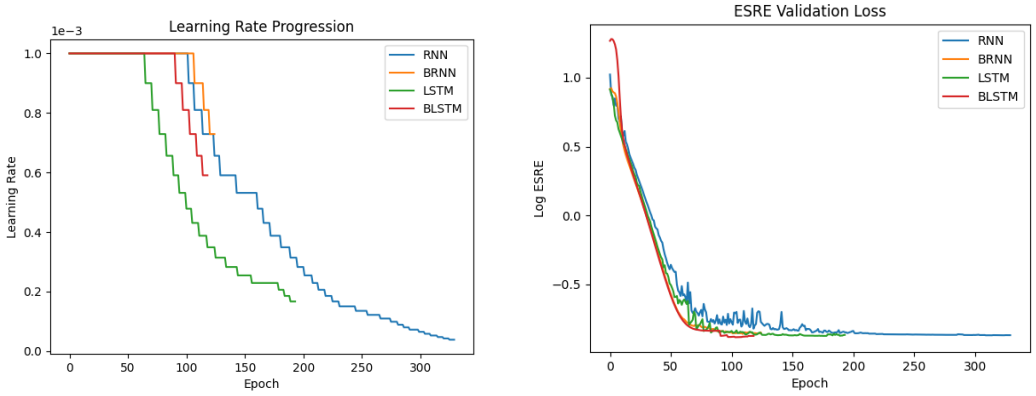


FIGURE 4.11: Example of the validation path of hyperparameter queries for the M Black-Scholes benchmark

incorporating a third hidden layer does not produce better performance on average. In fact, the performance of two and three hidden layer networks are largely intertwined. In Figure 4.12, we can view the training of the best model for each



(A) Adaptive learning rate

(B) Best validation loss per model type

FIGURE 4.12: Model fitting the best models for the M-feature set for the Black-Scholes benchmark

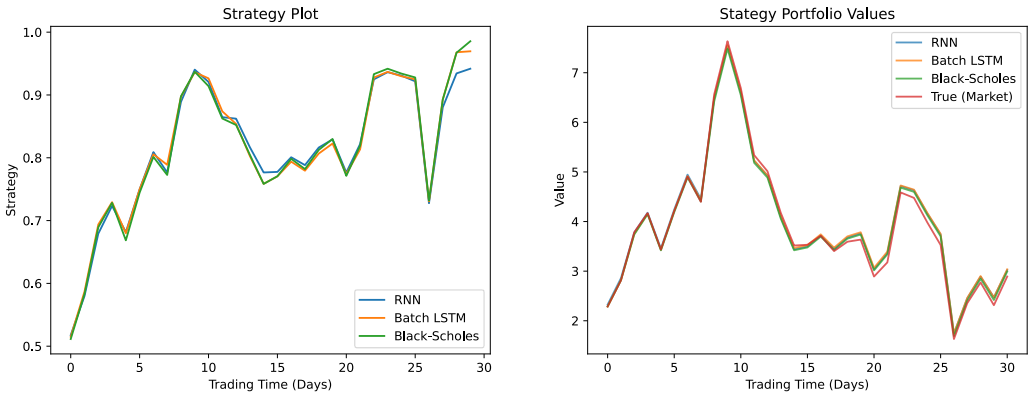
type. From this, we again see very similar performance of the models with convergence. Here, we find that whilst both the batched versions converged after around 140 epochs, the RNN model only converged after more than 329 epochs. The training of this model displays the importance of a learning rate reducer. For instance, the validation error fluctuates significantly over the 70-150 epoch whilst searching for the minima, which the much lower learning rate after 200 epochs is able to yield. The optimal hyperparameters are given by Table 4.5 with detailed training found in this Tensorboard. For the batch normalisation

Hyperparameter	Optimal Value	
	RNN	LSTM
Batch Normalisation	False	True
(N_1, N_2, N_3)	(60, 60, 60)	(10, 60, 10)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *elu*, *tanh*)	(*elu*, *tanh*, *elu*)
σ_{out}	*tanh*	*tanh*

TABLE 4.5: Optimal hyperparameters for neural network models for the Black-Scholes M-benchmark

hyperparameter in the Black-Scholes benchmark, we still tune the batch normalised and non-batch normalised versions independently. That is why we are still able to view both model types in Figure 4.12. The better performing version is then chosen to represent that architecture. Here, we again find that the batch normalisation is not favoured by the RNN structure.

With the best RNN and LSTM networks selected, we analyse their performance on the test set, beginning with a visualisation of the trading strategies and portfolio values in Figure 4.13 below. From Figure 4.13a, we find that the



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.13: The strategy and hedge for a random Black-Scholes sample path

deep hedging approaches closely follow the delta hedging strategy provided by the Black-Scholes model under the risk-neutral measure. As mentioned above, the delta strategy closely approximates the optimal solution which shows that the deep hedging approach appears to have the ability to learn the optimal solution within a discretisation of a continuous-time framework, in addition to the discrete dynamic programming framework above. Furthermore, the hedging portfolios of the three approaches in Figure 4.13b display an almost negligible difference between deep hedging and the Black-Scholes benchmark for this random sample path.

To analyse the performance with respect to the real-world measure, Figure 4.14 and Table 4.6 are required. The histogram of the PnLs in Figure 4.14 shows the reasonably symmetric distribution of the terminal profit and loss of the deep

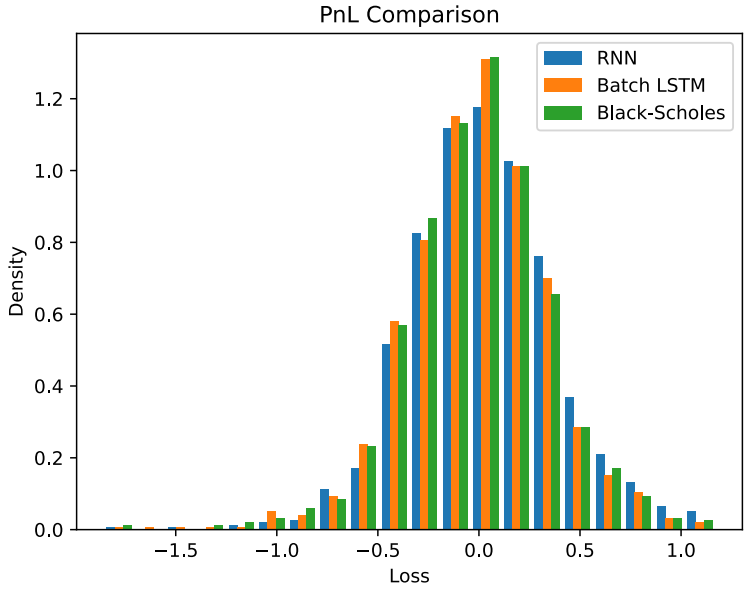


FIGURE 4.14: Profit and loss histogram for the Black-Scholes M-benchmark

Approach	Expected Square Replication Error
Black-Scholes	0.1238
RNN	0.1314
LSTM	0.1246

TABLE 4.6: Expected square replication error for Black-Scholes M-benchmark

hedging approach around zero. One could argue that the distributions is slightly skewed to the left; however, these are very similar in empirical distribution to the Black-Scholes benchmark. This confirms that our analysis of the single random sample path is accurate for general sample paths. From the results tabulated in 4.6, the Black-Scholes delta hedging strategy obtains the best ESRE on the test set which is expected from the benchmark. The important result is that the out-of-sample ESRE for both deep hedging models follow the benchmark's performance very closely. In particular, both deep hedging networks are within one one-hundredth of the benchmark ESRE. Consequently, the reinforcement learning procedure of deep hedging has been able to learn the value function of the dynamic programming problem[2].

LM-Feature Set

Now let us consider the LM-feature set under the Black-Scholes benchmark. The hyperparameter tuning process is summarised by this Tensorboard with

[2]The dynamic programming problem must not be confused with the dynamic programming solution. The former refers to the fact that all indifference pricing and quadratic hedging problems can be formulated as a dynamic programming problem. The latter is specific to solution provided in Section 3.2.3 used to obtain the dynamic programming benchmark.

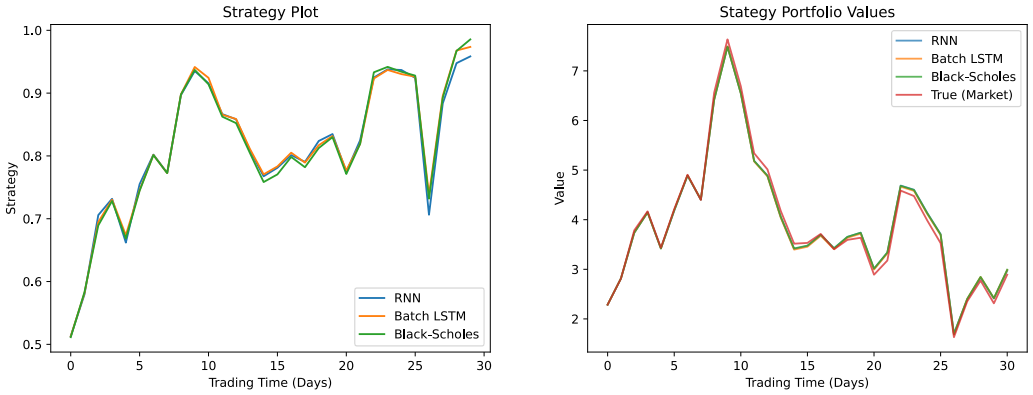
similar hyperparameter query plots as that seen in Figure 4.11 for the M-feature set. For this reason, we omit displaying another plot of example hyperparameter queries here, referring the interested reader to the Tensorboard. Again, the hyperparameter tuning procedure shows that networks with only two hidden layers perform competitively compared to those models with three hidden layers. The summary of the best models is provided by this Tensorboard which again mimics the M-feature set largely. Consequently, we only present the optimal hyperparameters in Table 4.7. These optimal hyperparameters support the

Hyperparameter	Optimal Value	
	RNN	LSTM
Batch Normalisation	False	True
(N_1, N_2, N_3)	(60, 60, 10)	(60, 60, -)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *tanh*, *tanh*)	(*tanh*, *tanh*, -)
σ_{out}	*tanh*	*tanh*

TABLE 4.7: Optimal hyperparameters for neural network models for the Black-Scholes LM-benchmark

claim that models with two hidden layers are competitive compared to those with three hidden layers since the best LSTM model only utilises two hidden layers; however, the largest available node sizes were chosen for these layers.

The MVH performance of the deep hedging models under the LM-feature set is comparable to the performance under the M-feature set. This analysis begins with Figure 4.15. Firstly, the RNN deep hedging strategies seen in Figure



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.15: The strategy and hedge for a random Black-Scholes sample path

4.15a appear slightly improved compared to those produced for Figure 4.13a, whilst the LSTM appears to have neither an improvement nor deterioration. The improvements towards the Black-Scholes benchmark yield visually negligible differences for the hedging portfolios of Figure 4.15b compared to those produced by the M-feature set. Furthermore, these improvements are only for

a single sample path and we cannot assume such improvements are achieved in general.

Comparing the LM histogram in Figure 4.16 to the M PnL of 4.14, the LM version appears to have slightly improved the RNN performance. This fol-

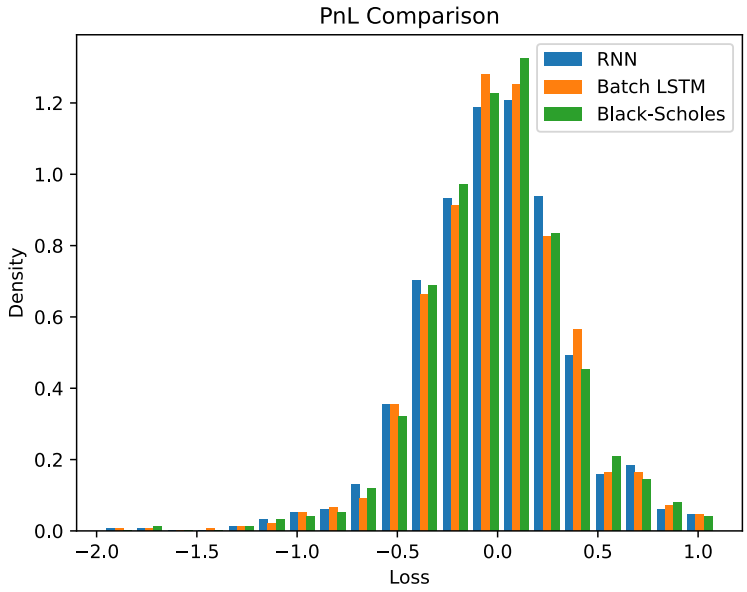


FIGURE 4.16: Profit and loss histogram for the Black-Scholes LM-benchmark

lows from the lower variance, and its performance mimicking the benchmark's performance closer. To quantify the potential improvements, we tabulate the ESRE in 4.8. From these quantities, we see that a slightly better performance

Approach	Expected Square Replication Error
Black-Scholes	0.1238
RNN	0.1286
LSTM	0.1261

TABLE 4.8: Expected square replication error for Black-Scholes LM-benchmark

of the RNN model with a decrease in ESRE from 0.1314 to 0.1286, validating our above analysis of the PnL. The LSTM shows a small increase in ESRE by 0.0015.

Whilst the LM-framework yields a difference deep hedging performance to the M-framework, the differences are extremely minor. There is no significant evidence to make the claim that either framework is better under the current simulation. On the other hand, both frameworks validate the application of deep hedging, confirming the ability for deep learning models to construct approximately optimal hedging strategies based on the Black-Scholes continuous-time market model.

As discussed in the preliminaries, 2.1.2, the Black-Scholes model is complete in a continuous-time framework; but, discretisation induces market incompleteness. The final benchmark we present is partly motivated by a desire to analyse deep hedging against a market model which is incomplete even in continuous time.

4.2.3 Heston

For our final benchmark, we analyse the deep hedging performance under the Heston market model. Such an analysis has two motivations. Firstly, 2.1.2 shows that the Heston market model has infinitely many ELMMs; therefore, the Heston model naturally creates an incomplete market. This creates an extra dimension of incompleteness when compared to discretising the Black-Scholes model. Secondly, the Heston model is another important model for the industry when stochastic volatility considerations are desired. For these reasons, the Heston model represents another relevant deep hedging benchmark to explore.

Unlike the Black-Scholes model, the Heston model dynamics are not traditionally used to simulate the underlying asset price process. Instead, the industry tends to use the Heston model for the purpose of a pricing model. We will discuss this further with our introduction of calibration in the upcoming numerical experiments; yet, we explicitly define the utilisation of the Heston model within this benchmark.

The discretised underlying asset prices process is simulated under the real-world measure $\mathbb{P}$ such that

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{\nu_t} S_t dW_t^s, \\ d\nu_t &= \kappa(\theta - \nu_t)dt + \xi\sqrt{\nu_t} dW_t^\nu, \end{aligned}$$

where W^s and W^ν are one-dimensional $\mathbb{P}$-Brownian motions with $d\langle W^s, W^\nu\rangle_t = \rho dt$. Through the Girsanov Theorem, for a risk-neutral measure $\mathbb{Q}$, the $\mathbb{Q}$-dynamics are

$$\begin{aligned} dS_t &= r_f S_t dt + \sqrt{\nu_t} S_t d\tilde{W}_t^s, \\ d\nu_t &= \kappa(\theta - \nu_t)dt + \xi\sqrt{\nu_t} d\tilde{W}_t^\nu, \end{aligned}$$

where $\tilde{W}^s$ and $\tilde{W}^\nu$ are $\mathbb{Q}$-Brownian motions with $d\langle \tilde{W}^s, \tilde{W}^\nu\rangle_t = \rho dt$. The pricing and hedging follows these $\mathbb{Q}$ dynamics. Consequently, the underlying price process and pricing model is represented by the Heston model.

The model parameters for the above dynamics are as follows.

$\mu = 0.10.$

$r_f = 0.0.$

$\kappa = 4.$

$\theta = 0.02.$

$\xi = 0.3.$

$\rho = -0.2.$

$\nu_0 = 0.02.$

The hedging strategy from the Heston model is the delta hedging strategy. Whilst Section 2.1.2 discuss more elaborate hedging through the introduction of a volatility asset, the use of delta hedging serves the required purpose of setting a benchmark when restricted to trading in the single underlying asset.

Now, let us consider the hyperparameter space $\mathcal{X}$. We continue to allow networks to have three hidden layers since three of the four of the above Black-Scholes networks utilised the third layer. In addition, we expand the number of nodes available per layer to include 80 and 100. The reason for this expansion is two-fold. On one hand, the upper bound of 60 nodes was still a common choice in the hyperparameter tuning process, possibly signalling a potential benefit to an increases node size. On the other hand, the batch normalisation action is incorporated into the hyperparameter tuning process directly from this point forward. Such an incorporation reduces the computational resources required for the tuning algorithm, supporting the choice of an expanded search space. Consequently, we define the search space as follows

$$\begin{aligned}\mathcal{X} = \{(B, L, N_1, \ldots, N_L, \sigma_1, \ldots, \sigma_L, \sigma_{\text{out}}) : B \in \{\text{True, False}\}; L \in \{2, 3\};\\ N_\ell \in \mathfrak{N}; \sigma_\ell \in \Sigma_h; \sigma_{\text{out}} \in \Sigma_o \ \text{ for } \ell = 1, \ldots, L\},\end{aligned}$$

where $\mathfrak{N} = \{10, 30, 60, 80, 100\}$, $\Sigma_h = \{\tanh, \text{selu}, \text{elu}\}$, $\Sigma_o = \{\tanh, \text{relu}, \text{swish}\}$ and B represents the variable for the inclusion of the batch normalisation process.

M-Feature Set

The Bayesian optimisation process represented by the 50 RNN queries and 50 LSTM queries is summarised by the following Tensorboard with a visualisation of example queries in Figure 4.17. Comparable to the query plots illustrated for the Black-Scholes benchmark in Figure 4.11, we find that a similar rate of convergence for the Heston benchmark is achieved. In particular, most of the queries in the Tensorboard plateau after around 65 epochs. The `Hparam` tab clearly shows that the direct incorporation of the batch normalisation into the tuning search results in far fewer non-batch normalised network queries compared to batch normalised queries. This follows since the acquisition function of the Bayesian optimisation algorithm does not assign a high measure of information towards the exclusion of the batch normalisation process after a few queries support the notion that the batch normalisation results in superior network performance. When analysing the performance associated with the number of hidden recurrent layers incorporated in the networks, we find that the top performing queries all contain three hidden layers. In particular, the top 13 performing queries with respect to the validation loss are three hidden layer models. The associated node size of these queries is implies that $\mathfrak{N} = \{10, 30, 60, 80, 100\}$ is a sufficient set since the choice of 100 nodes is rarely

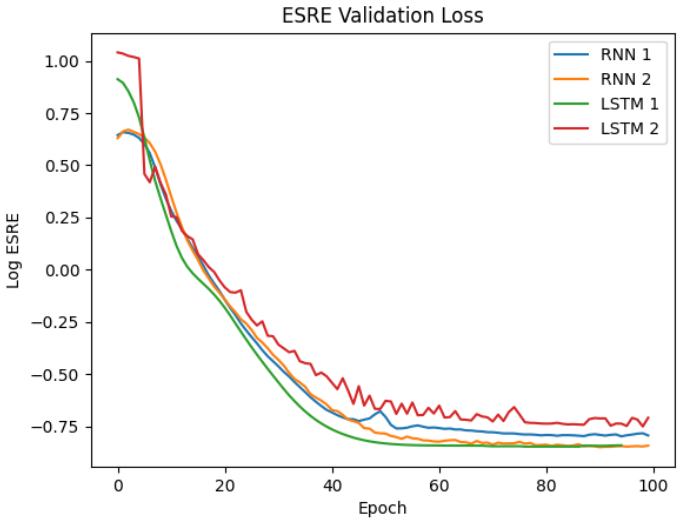
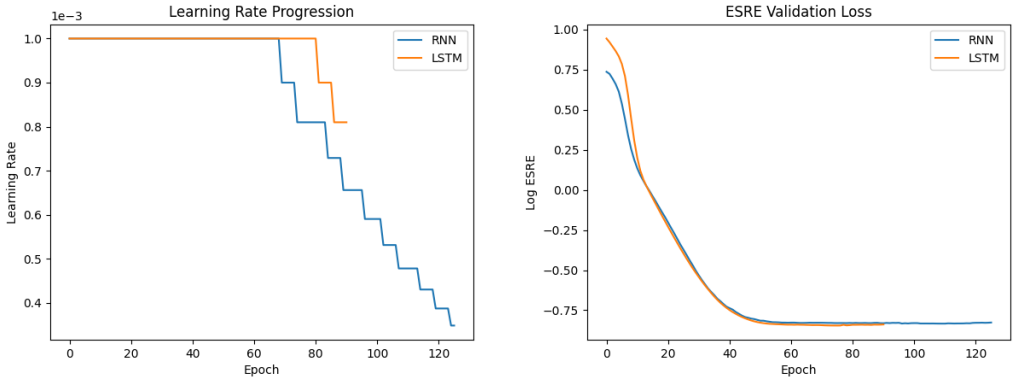


FIGURE 4.17: Example of the validation path of hyperparameter queries for the M Heston benchmark

chosen. In fact, the node size of the first layer, N_1, is consistently chosen as 10 for the top performing models, implying a good extraction of initial information with the small node size.

Tensorboard summarises the full training process of the best networks with 4.18 illustrating the training of the final networks for the benchmark. Figure 4.18



(A) Adaptive learning rate (B) Best validation loss per model type

FIGURE 4.18: Model fitting the best models for the M-feature set for the Heston benchmark

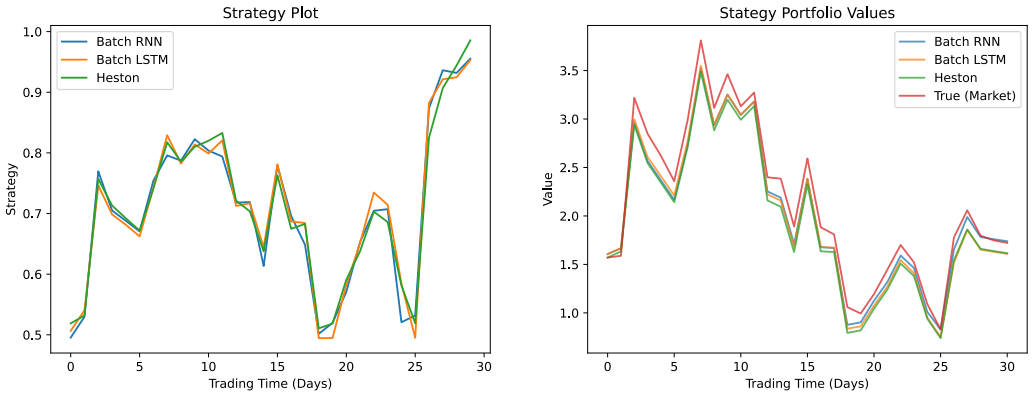
clearly shows that the two optimal deep hedging models have similar convergence and performance with the RNN utilising the lowered learning rates to fine-tune the parameters over a longer duration. The network hyperparameters for these models are tabulated in 4.9 below. As mentioned within the hyperparameter tuning analysis above, these models both utilise batch normalisation and three hidden recurrent layers with varying node sizes. The structure between the RNN and LSTM model is almost identical with the node size on the

Hyperparameter	Optimal Value	
	RNN	LSTM
Batch Normalisation	True	True
(N_1, N_2, N_3)	(10, 80, 60)	(10, 80, 100)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *elu*, *tanh*)	(*elu*, *elu*, *tanh*)
σ_{out}	*tanh*	*tanh*

TABLE 4.9: Optimal hyperparameters for neural network models for the Heston M-benchmark

final layer carrying the only difference. This observation; however, is not consistent throughout the previous experiments and we regard this as a coincidence which concludes our analysis of the hyperparameter tuning process, allowing us to proceed to the deep hedging analysis.

Analysing the deep hedging performance of these networks, we again begin with the visualisation of the resultant hedge for a single sample path displayed by Figure 4.19. The strategies in Figure 4.19a illustrates greater differences



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.19: The strategy and hedge for a random Heston sample path

between the deep hedging strategies and the Heston delta strategy compared to previous benchmarks. Despite this, the deep hedging strategies approximate the Heston benchmark and the differences in the strategies result in small differences in hedging portfolios. The LSTM hedging portfolio appears to be a very close approximation of the Heston portfolio whilst the RNN portfolio results in an almost perfect terminal hedge of the claim. The better performance observation of the RNN strategy is restricted to this sample path though. To evaluate the expected deep hedging performance, we refer to the PnL and ESRE in Figure 4.20 and Table 4.10 respectively. The PnL in 4.20 shows that the deep hedging approaches have a higher density very close to zero and very similar density around zero; however, they experience a higher density on the extremities. These extremities result in a worse ESRE relative to the Heston model as shown in 4.10. Whilst the difference is the ESREs is greater than the

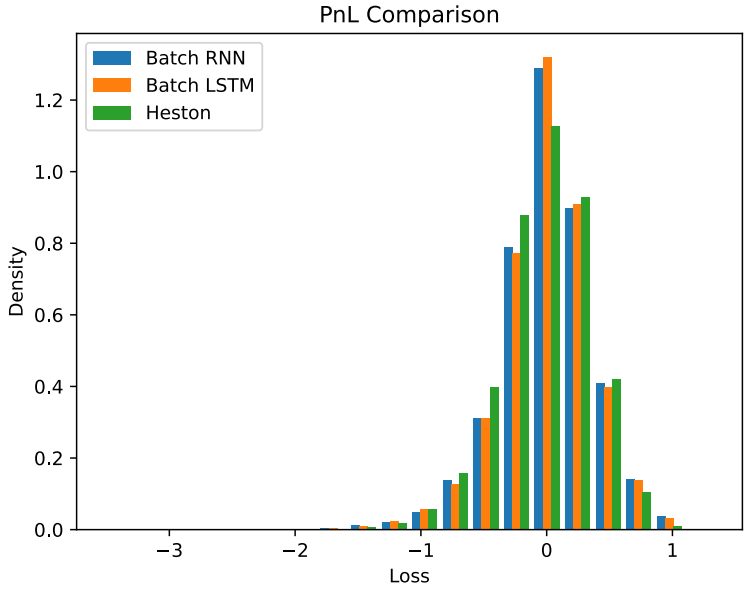


FIGURE 4.20: Profit and loss histogram for the Heston M-benchmark

Approach	Expected Square Replication Error
Heston	0.1318
RNN	0.1438
LSTM	0.1394

TABLE 4.10: Expected square replication error for Heston M-benchmark

Black-Scholes benchmark, the LSTM outperforms the RNN model again with all approaches sharing extremely similar performances based on the test set. The performance clearly shows that the deep hedging approach is robust enough to find the optimal hedging strategy under a Heston-based market framework.

LM-Feature Set

The hyperparameter tuning for the Heston LM-feature set is available in the following Tensorboard whilst a summary of the training of the optimal models can be found in this Tensorboard. For the sake of repetition, we again omit the plotting of the hyperparameter queries and the optimal networks due to their similarity to the M-feature set visualisations. For this, the reader is referred to the respective Tensorboards. Instead, we make the following observations about the network construction. Opposed to the M-feature set, we find that the performance of two hidden layer networks is competitive relative to those containing three hidden layers. In fact, seven of the top ten performing queries only contain two hidden layers. Secondly, we see that the choice of 100 node layer sizes dominate the top performing queries; however, the common choice of two hidden layers implies that the flexibility available to the networks is

sufficiently large for the learning process.

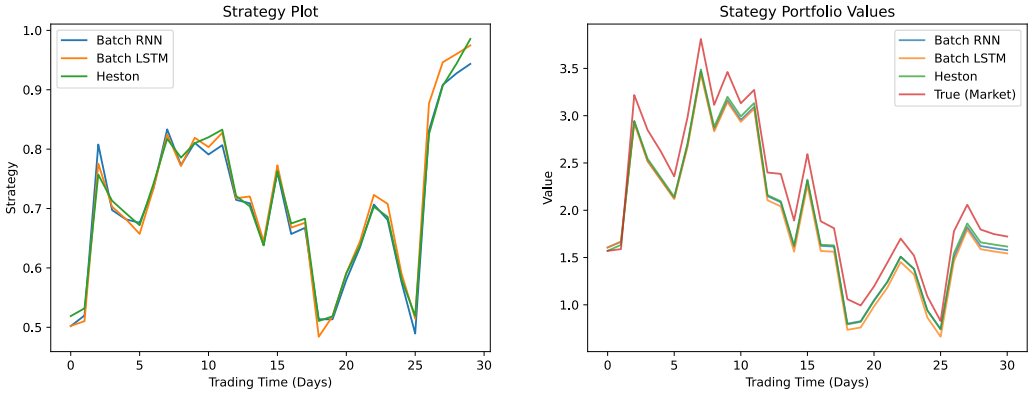
The optimal deep hedging architectures are summarised by Table 4.11. From

Hyperparameter	Optimal Value	
	RNN	LSTM
Batch Normalisation	True	True
(N_1, N_2, N_3)	(10, 30, -)	(100, 100, -)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *elu*, -)	(*elu*, *tanh*, -)
σ_{out}	*tanh*	*tanh*

TABLE 4.11: Optimal hyperparameters for neural network models for the Heston LM-benchmark

this, we again see a preference for the batch normalisation process; however, whilst the M-feature set optimal hyperparameters were extremely similar, the current choices experience greater differences. Let us now analyse the deep hedging performance of these networks.

The deep hedging strategies of the sample path reviewed in the M-feature set discussion are presented in Figure 4.21 below. From this figure, we can see



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.21: The strategy and hedge for a random Heston sample path

that the deep hedging performance is comparable to the M-feature set. In particular, the strategies in 4.21a are very similar to those provided in 4.19a and experience similar deviation from the Heston model strategy at the same time. These differences appear to be more pronounced with the LSTM model which results in the portfolio value of the RNN model achieving a slightly better approximation of the Heston benchmark. In both the M- and LM-feature set frameworks, the deep hedging networks experience their greatest deviation from the benchmark strategy when making large changes in the trading strategy.

Despite the better performance of the RNN on the above sample path, the overall performance on the test set continues to promote the LSTM architecture with the better deep hedging performance. This is illustrated by the slightly better

performance of the LSTM in Table 4.12. When compared to the deep hedging performance for the M-feature set in 4.10, we observe slight improvements for both the RNN and LSTM models. This appears to be a common observation within the provided benchmarks; however, the differences are neither significant nor expected given the discussion in 4.1.2.

Approach	Expected Square Replication Error
Heston	0.1318
RNN	0.1391
LSTM	0.1377

TABLE 4.12: Expected square replication error for Heston LM-benchmark

The PnL histogram in Figure 4.22 supports the minor differences between the two feature sets. A higher density at zero is again observed by the deep hedging

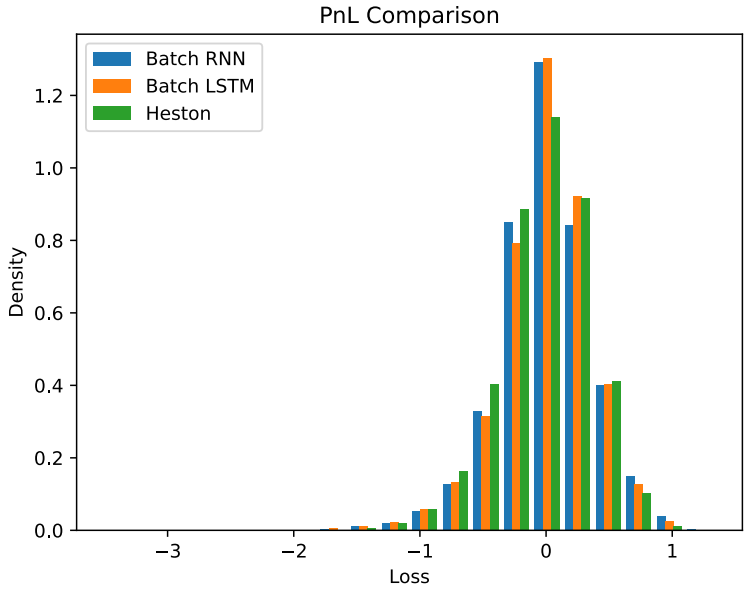


FIGURE 4.22: Profit and loss histogram for the Heston LM-benchmark

approaches compared to the benchmark; however, the extremities remain to cause the worse ESRE.

The culmination of the above three benchmarks have progressively relaxed simplifying assumptions made in the simulation of the underlying asset price process. In particular, we begin with a highly discrete simulation in the dynamic programming, before discretising the Black-Scholes and Heston continuous-time models. The Heston benchmark removed the inherent completeness of the Black-Scholes model under a continuous-time framework. In all of these benchmarks, we found that the deep hedging approach is sufficiently able to approximate the optimal reinforcement learning solution for the mean-variance

approach to hedging incomplete markets. This extends Buehler et al., 2019 whereby the focus was with indifference pricing and convex risk measures such as entropic risk and CVaR. In the numerical experiments to follow, we analyse deep hedging from a far more realistic perspective by utilising real-world data.

4.3 Numerical Experiments

Now that we have compared the deep hedging approach to model hedging approaches under ideal simulated market conditions, we analyse the robustness of each approach under real-world market conditions. The analysis is based on two experiment types. The first makes little change to the deep hedging process followed in the above benchmarks from a network perspective. The important change for the experiment relates to the market information whereby a method for simulating market conditions is explored. The second experiment follows a more traditional hedging analysis for real-world data which requires a change of risk measure to account for a single sample path.

4.3.1 Setup

Before proceeding with the experiments, we discuss the common setup for the experiments. These are the topic of calibration and the real data obtained.

Calibration

At its core, calibration is an option pricing tool which aims to match the observed market option prices to theoretical model prices. Such a process relies on finding the pricing model parameters which yield the observed market prices. To understand further, let us discuss the basic yet essential Black-Scholes example.

Suppose that a trader uses the Black-Scholes model for option pricing. Further assume that the calibration is with respect to vanilla call options such that the model price is given by:

$$c = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

where Φ is the CDF of the standard normal distribution and

$$d_1 := \frac{\ln\left(\frac{S_0}{K}\right) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$$
$$d_2 := d_1 - \sigma\sqrt{T}.$$

Therefore, for any given call option, the pricing model is dependent on the following set of parameters:

S – current spot price.

K – Strike price.

T – Time to maturity.

r – Risk-free rate.

σ – Volatility.

The important aspect about calibration is to notice that all of the above model parameters are known and observed in the market at any given time except for the volatility parameter, σ. It is then a trivial matter to find the value of σ which causes the model price to equal the market price for a single option. This volatility parameter is known as implied volatility, denoted σ_{imp}.

Definition 4.3.1
The implied volatility, σ_{imp}, of an option is the Black-Scholes volatility parameter which causes the observed market price and the Black-Scholes model price of the given option to coincide. Alternatively,

$$V^{market} \stackrel{!}{=} V^{BS}(\sigma_{imp}),$$

where V^{market} is the observed market price and $V^{BS}(\sigma_{imp})$ is the theoretical Black-Scholes model price with volatility σ_{imp}.

Note that for a given option, the price can be quoted with the implied volatility value alone and such a quotation is often used in industry.

Now consider the collection of options of an underlying asset available in the market for varying maturities and strikes. Each implied volatility is computed based on a single option and the implied volatility can change for each of the options in the collection. What emerges is known as a implied volatility surface which is a surface interpolating the implied volatility for an interval of maturities and strikes. For example, Figure 4.23 below shows a historical volatility surface of the S&P500 equity index. Calibration is the process of obtaining pricing

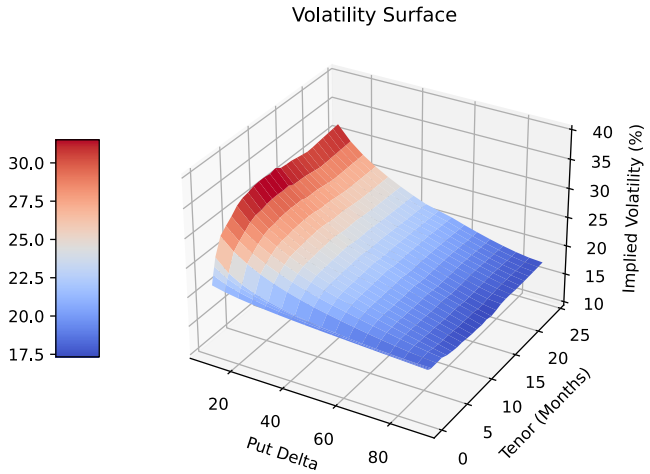


FIGURE 4.23: S&P500 volatility surface for 15 December 2022

model parameters which best approximate the observed volatility surface.

Further consider a trader implementing a Heston pricing model. It is clear that the unknown model parameters for Heston are $r, \kappa, \theta, \xi, \rho, v_0$ from equations 2.8 and 2.9. If we denote M_h as the collection of possible Heston parameters, then we can define Heston calibration as follows.

Definition 4.3.2
Calibration for the Heston model is the optimisation process with respect to the collection of market observable options, $\mathcal{O}$, such that

$$h^* = \arg\min_{h \in M_h} \sum_{i \in \mathcal{O}} d(V_i^{Heston}(h), V_i^{Market}),$$

where $V^{Heston}(h)$ is the Heston price given parameters h, V^{Market} is the observed market price and $d(.,.)$ is some metric chosen by the trader. A common choice for the metric is the usual metric on $\mathbb{R}$ such that $d(x, y) = |x - y|$.

Consequently, calibration is the process of finding the optimal pricing model parameters which minimises the difference between the model prices and the market prices. With these optimal parameters, a trader is then able to obtain the model price for an option which does not exist in the market; but, is consistent with current market sentiment.

The Data

For the real-world deep hedging analysis, we require an underlying asset, the S&P500. When choosing the underlying asset to represent the numerical experiments, we desire a liquid asset for two reasons. Firstly, a liquid asset is generally accompanied by liquid derivatives which is important for the comparisons made in 4.3.3 to follow. Secondly, the hedging analysis provided is more realistic for a liquid asset. Whilst the overall magnitude of the financial positions implemented by the hedging strategy could influence both the underlying asset price and derivative price if large enough, the assumption is made that the deep hedging strategies are implemented without significant influence on the market. This assumption is generally sufficient for liquid underlying assets.

The other factor under consideration for the underlying asset choice is the availability of data. We utilise the Refinitiv Eikon database for all of the historical data gathered throughout these numerical experiments. Whilst the historical prices of equity assets is readily available, the historical volatility surfaces are less common; however, the database does store limited daily historical volatility surfaces for selected global equity indices. Consequently, we choose the Standard & Poor's 500 index (S&P500) to represent our underlying asset for the following experiments. For the experiments, we utilise the past 20 years worth of historical closing prices of the index and are restricted to the daily volatility surfaces from the 8th of October 2018 to 22 December 2022.

The S&P500 is a United States stock index which is composed of 500 companies that are considered to be sector leaders by the index provider Standard & Poor's. As a result, the *risk-free* rate for the index is obtained from US Dollar (USD) interest rates. Since the options which we consider for the experiments are short

term, we gather the: overnight; one week; one month; three month; six month; and one year USD Libor rates.

Now that we have an understanding of the real data, we can explore our first deep hedging experiment based on real data.

4.3.2 Pseudo Real Data

As an extension of price process simulation under continuous-time market models, we now explore the use of financial market generation. The theory discussed in the previous chapters aim to minimise the risk under the real-world measure $\mathbb{P}$. The inherent problem with the real world is that we only observe one sample path from $\mathbb{P}$ which requires that the process of training deep learning models is adapted when compared to the above experiments. One such method is to create sample paths from $\mathbb{P}$.

Data Generation Process

Generating financial markets with signatures, proposed by Buehler et al., 2020, provides a novel incorporation of signatures and variational auto-encoders into financial data generation. With such an approach, one is able to generate underlying asset price paths such that the generated paths follow the real-world measure. We implement the same generative process to create sample paths to both train the deep hedging models and to test their performance.

Briefly, the method of Buehler et al., 2020 begins by transforming the partitions, say monthly partitions, of the observed real-world path with the lead-lag transformation. The log signatures of these transformed partitions are then used to train a conditional variational auto-encoder (CVAE). Consequently, the CVAE learns to simulate the log signature of these partitions where the simulation is conditioned of the previous partition's signature. Since signatures are unique, the log signatures generated by the CVAE can be inverted to produce paths which have the same distribution as the original partitions; and therefore, the real-world measure $\mathbb{P}$ whilst preserving contiguous properties. For a deeper understanding of this signature based financial market generator, refer to Buehler et al., 2020.

We take a moment to make three remarks about the above process. Firstly, the inversion of signatures, and log signatures by extension, is a highly non-trivial task. In particular, computationally efficient algorithms for signature inversion remains an open topic which is well beyond the scope of this dissertation. For the inversion, we follow the Buehler et al., 2020 method of evolutionary algorithms. Evolutionary algorithms solve problems by mimicking some population evolution. With some measure of fitness, a random collection of *possible solutions*, known as a population of individuals, initialise the algorithm. The best individuals in the population chosen by the measure of fitness are kept in the population, whilst the other individuals are removed. These 'fit' individuals become the *parents* to an introduction of new individuals into the population

which have mutations of some kind. This process of selection of the fittest individuals to form the basis for improvements repeats until a satisfactory individual represents the problem solution; mimicking the notion of biological evolution. For a true introduction to evolutionary algorithms, the reader is referred to Yu and Gen, 2010.

The second remark surrounds the claim that the generated paths are produced from the real-world measure $\mathbb{P}$. Consequently, we require a method for assessing the quality of the generated paths with respect to $\mathbb{P}$. For this purpose, Buehler et al., 2020 incorporate the results introduced by Gretton et al., 2012. Here, the authors explore a two-sample kernel test statistic to infer whether two samples share the same law. The test statistic is given as follows

$$T_U^2(X_1, \dots, X_m; Y_1, \dots, Y_n) \coloneqq$$
$$\frac{1}{m(m-1)} \sum_{i,j;i\neq j} k(X_i, X_j) - \frac{2}{mn} \sum_{i,j} k(X_i, Y_j) + \frac{1}{n(n-1)} \sum_{i,j;i\neq j} k(Y_i, Y_j),$$

where k is the signature kernel defined by Chevyrev and Oberhauser, 2018. Note that a metric based on the marginals of the two stochastic processes is insufficient because whilst two processes can share identical marginals, they may have different laws. Consequently, the above analysis on the law of the entire stochastic process is required.

The final remark relates to the inversion process. Recall that the path integral, and hence the signature, is translation invariant. Consequently, the inverted paths of the CVAE generated signatures represent the cumulative returns path instead of the price path; however, this is trivially obtained by starting the inverted path at the terminal price of the conditioning partition or a random value from an appropriate historical price range.

Returning to the deep hedging problem, we now provide a small contribution to the market generation topic by slightly expanding the use case of Buehler. Buehler et al., 2020 generate the underlying asset paths in a financial market which is sufficient for the purpose of creating training samples for deep hedging models; however, we want to continue our comparison between the performance of the deep hedging approach and hedging strategies produced by a market model. As discussed above, calibration is required under such a framework to obtain an information accurate hedge and mimic the trading strategies which a trader would have implemented should they have been in such a position. Consequently, the simulation of both the underlying asset and the associated implied volatility is necessary for the comparison; however, the implied volatility inclusion is accompanied by a set of problems.

Firstly, the implied volatility is not constant with respect to maturity or strike of an option; but, rather inherits the volatility smile property, discussed above, in general. Secondly, the implied volatility must be generated jointly with the underlying asset to preserve the inherent dependence. Therefore, a few choices of the generation could be made.

1. Generate an entire volatility surface for each trading period.

2. Carefully choose a single implied volatility to generate which tracks the implied volatility for the specific option throughout the market changes.
3. Generate calibrated market model parameters.

The first potential solution provides a simulation of the market information which a trader would use for her market model trading strategy; however, the generation of the surface is relatively high dimensional compared to the underlying asset generation. The second solution generates the implied volatility at the specific point on the surface which matches to the option which the trader is hedging; yet, calibrating a market model such as the Heston model to a single implied volatility is problematic with many solutions. The model parameters obtained from such a simulation would be erratic since the calibrated parameters would likely represent local minima. That brings us to our final solution. Generating model parameters solves the high dimensionality problem of the first solution whilst largely maintaining the entire surface information. Furthermore, the implied volatility surface under the current framework is used to calibrate a market model. Generating calibrated parameters directly skips the need calibrate after the fact. The restriction, however, is that the choice of a single market model must be chosen and maintained prior to the CVAE training; although, one could use the model parameters to reconstruct the volatility surface and calibrate to an alternative market model.

Let us explicitly define the input for such a CVAE. First, let $S_t, \mu_t, \kappa_t, \theta_t, \xi_t, \rho_t, v_{0_t}$ represent the path through time of the underlying asset and the Heston model parameters respectively. For the subscript of the model parameters, we are not implying that the dynamics of of the Heston model treat these parameters as time-dependent. Rather, these represent the calibrated parameter constants obtained when calibrating the model to the volatility surface at time t. Now recall the lead-lag transformation from Section 2.3. For the purposes of capturing the quadratic variation of paths, we must apply the lead-lag transformation; however, capturing the quadratic variation of the underlying asset is sufficient due to the dependence and joint generation. In order to maintain conformity, we use the lead transformation on the model parameters. Define X to be the multidimensional path such that

$$X_t = (\hat{T}_{\text{lead-lag}}(S_t), \hat{T}_{\text{lead}}(\mu_t, \kappa_t, \theta_t, \xi_t, \rho_t, v_{0_t})),$$

where $\hat{T}_{\text{lead}}$ represents the component-wise lead transformation for the sake of notation. For example, $\hat{T}_{\text{lead}}(X, Y) = (\hat{T}_{\text{lead}}(X), \hat{T}_{\text{lead}}(Y))$. We can define Z as the input for the CVAE such that

$$Z_p = S_k(X)_{t_p, t_{p+1}},$$

with the conditional input as Z_{p-1}, where $p = 0, \ldots, P$ denotes the partition, t_p is the start time of the partition p and $S_k(X)$ denotes the signature of X truncated at level k.

Whilst such an extension provides a possible solution to the joint generation of option prices and its underlying asset, the inversion of such signatures is

computationally intractable via evolutionary algorithms. The authors are currently unaware of an inversion method which is computationally efficient enough to manage the described signature inversion. Consequently, we only generate underlying asset paths for our current experiment. Under this generative framework, the input for the CVAE is defined such that

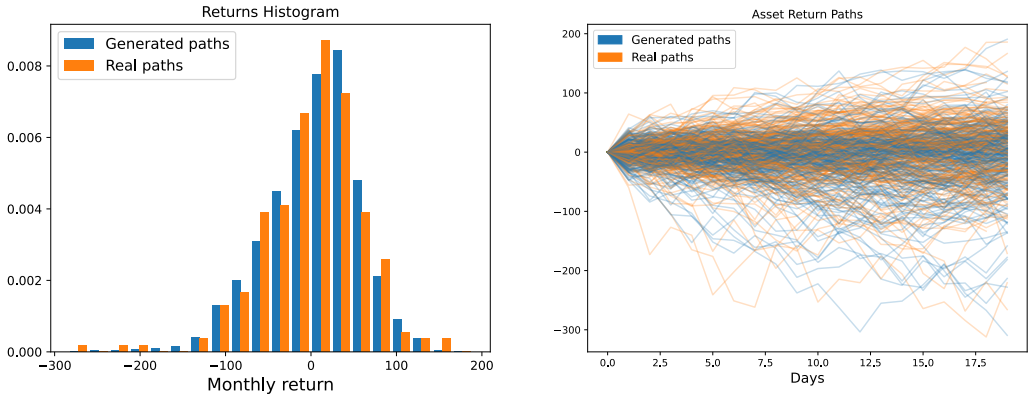
$$Z_p = S_k(\hat{T}_{\text{lead-lag}}(S_t))_{t_p,t_{p+1}},$$

which is the input defined by Buehler et al., 2020. With the theoretical framework defined, we must now describe the specifications of the data generation.

The Setup

As discussed above, we must partition the underlying asset paths. Since the benchmarks explored utilise monthly options, we choose to continue our deep hedging analysis with monthly options. Since a calendar month has roughly 22 business days, we partition the underlying asset data into calendar months, and generate underlying asset paths with a fixed length of 22 trading days via the inversion process. Such a partition and generation process allows for a sufficient number of partitions for the training of the CVAE and a realistic and convenient path length which will be further exploited in the second experiment.

One can infer the data generation accuracy by visualising both the histogram of the real and generated returns and a sample of the generated paths. Such a visualisation is provided in Figure 4.24 below. From 4.24a, we can see that the



(A) Histogram of the generated and real returns

(B) Extract of generated paths compared to the real paths

FIGURE 4.24: Visualisation of the path generation accuracy

empirical distributions of the generated and real returns over monthly partitions are highly comparable. Furthermore, Figure 4.24b serves as a sanity check to verify that the paths which produce these returns appear to follow the structure expected from an asset price path. Both of these visuals extend the theoretical motivation that the CVAE is a sufficient generative model.

For the experiment, we utilise 100 000 pseudo real price samples to train the deep hedging models and reserve 10 000 samples as a test set. This is consistent with

the number of samples used within the benchmarks above. With these generated paths, we analyse the MVH performance of the deep hedging approach. Whilst the process is almost identical to the benchmark implementation, there are some important differences. Firstly, the initial underlying asset price $S_0 \sim \mathcal{N}(2500, 50)$ which is used to convert the generated returns paths into price paths. For the current experiment, $S_0 = 2522.02$. This initial price remains consistent for all of the paths as required by the MVH framework, but, only the LM-feature set is used. As with the previous MVH benchmarks, each sample path is associated with a vanilla call option which is at-the-money at time $t = 0$. Secondly, since trading is only on business days, we set $T = 21/220$ and $n = 21$. Finally, the current framework does not provide an inherent benchmark or pricing model hedging comparison. There are no option prices throughout the life of the option which can be used to obtain a delta hedging strategy from, only the observed terminal payoff. Consequently, we compare deep hedging against two Black-Scholes delta hedging strategies. The first strategy is the Black-Scholes hedge where the volatility parameter is the constant volatility of the path. This represents a high performing strategy since the calculated volatility requires future information and is not necessarily a realistic Black-Scholes hedging performance. The second Black-Scholes strategy implements a variable volatility parameter based on a 20 trading day historical volatility. Futhermore, the price for the Black-Scholes benchmarks is the Black-Scholes price given by the future volatility. This allows for consistency in the comparison and implementation of the price-taker notion.

Our final choice for the experiment setup follows the hyperparameter space. The previous benchmarks explored a variety of hyperparameter spaces. Since this exploration was evolutionary, the hyperparameter space for this experiment shares the same space as the Heston benchmark. In particular, let

$$\begin{aligned}\mathcal{X} = \{(B, L, N_1, \ldots, N_L, \sigma_1, \ldots, \sigma_L, \sigma_{\text{out}}) : B \in \{\text{True, False}\}; L \in \{2, 3\};\\ N_\ell \in \mathfrak{N}; \sigma_\ell \in \Sigma_h; \sigma_{\text{out}} \in \Sigma_o \ \text{ for } \ell = 1, \ldots, L\},\end{aligned}$$

where $\mathfrak{N} = \{10, 30, 60, 80, 100\}$, $\Sigma_h = \{\tanh, \text{selu}, \text{elu}\}$, $\Sigma_o = \{\tanh, \text{relu}, \text{swish}\}$ and B represents the variable for the inclusion of the batch normalisation process. Each query is trained for 200 epochs with a starting learning rate of 0.01.

With the experiment setup concluded, we can now proceed to our experiment analysis.

The Analysis

As seen in the hyperparameter tuning of the Heston benchmark, the exploration of 50 RNN queries and 50 LSTM queries appears to be more than sufficient since the majority of queries result in an almost equivalent performance metric. Consequently, for this pseudo real data experiment, we consider 20 hyperparameter queries per model type. The full summary of these queries can be viewed in the following Tensorboard with an extract shown in 4.25 below. From the Tensorboard, and by extension the above extract, we find that the networks do not converge over the 200 epochs despite the higher learning rate and additional

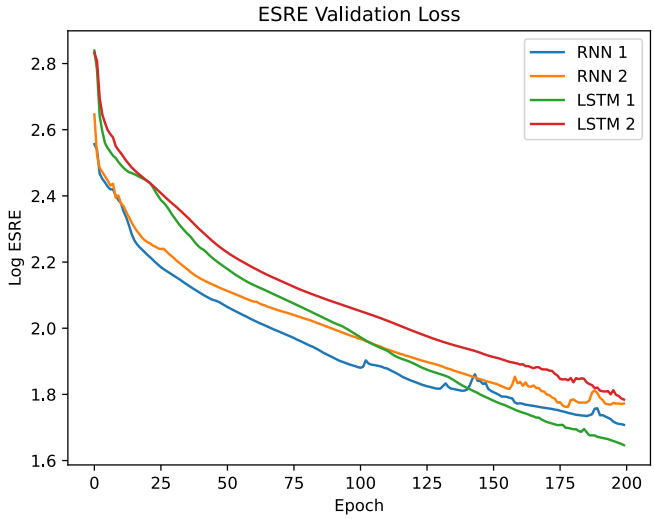
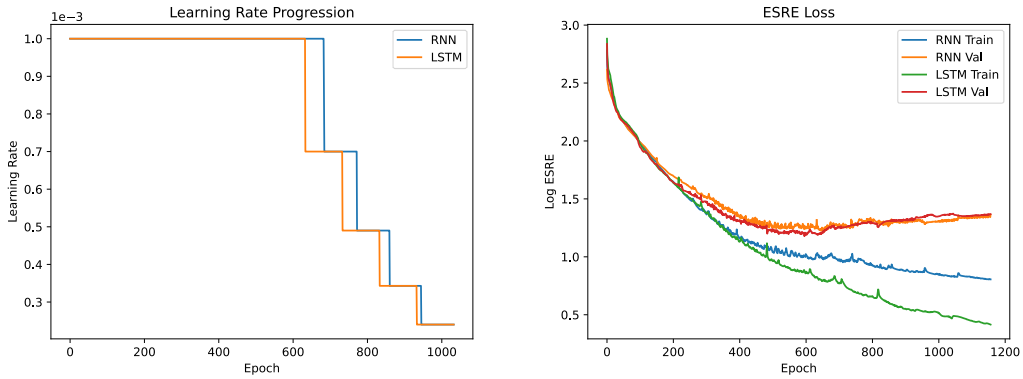


FIGURE 4.25: Example of the validation path of hyperparameter queries for experiment one

training epochs compared to the Heston benchmark. Whilst the top performing queries have a mixture of two and three hidden layers, the tuning process shows a strong preference towards a maximum node size of 100 in the initial hidden layer and a variety of node sizes for the following hidden layers. With initial hyperparameter tuning optimisation covered, we proceed with our final model choices.

Regardless of the convergence level, the tuning process provides an idea of promising queries which are trained fully. These are summarised in this Tensorboard with the training of the best performing networks shown in 4.26 and the optimal structure in 4.13 below. A notable difference between the model



(A) Adaptive learning rate

(B) Best validation loss per model type

FIGURE 4.26: Model fitting the best models for experiment one

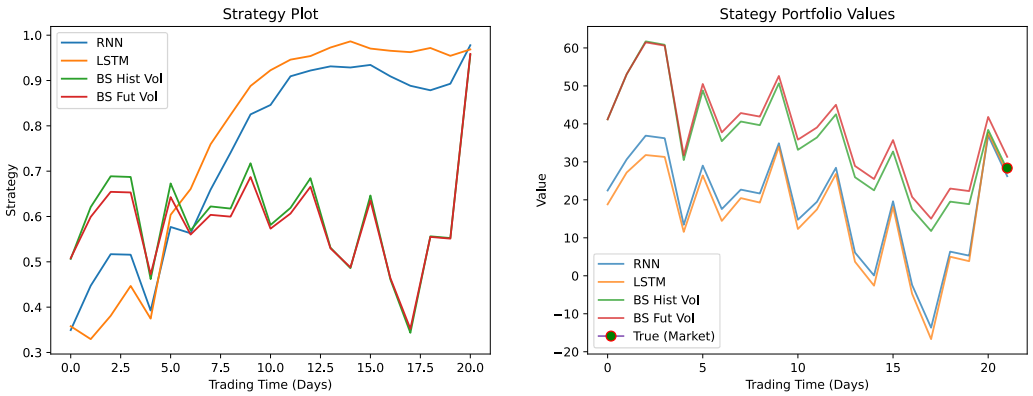
fitting in the above benchmarks and the first experiment accompanies the difference between the training and validation loss. In the benchmarks, there is generally an insignificant difference between the training and validation loss;

Hyperparameter	Optimal Value	
	RNN	LSTM
Batch Normalisation	False	False
(N_1, N_2, N_3)	(100, 100, -)	(100, 80, -)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *elu*, -)	(*tanh*, *selu*, -)
σ_{out}	*swish*	*swish*

TABLE 4.13: Optimal hyperparameters for neural network models for experiment one

hence, only the validation loss is visualised. For the pseudo real data experiment, the training and validation loss are distinct whereby the bias-variance trade-off is a more a important consideration. In particular, Figure 4.26b shows that the networks begin to overfit after approximately 600 epochs since the training loss continues to decrease whilst the validation loss passes its minimum value. The difference is an initial indication that the pseudo-real prices, and by association observed asset prices, are notably more difficult to hedge than the benchmark price processes. For the optimal hyperparameters, we find that batch normalisation was not favoured and that the choice of the initial hidden layer node size is 100. This could indicate the initial complexity in the pseudo-real prices which the network is required to learn. Both of these optimal models utilise only two hidden layers, implying that a sufficient hyperparameter space. Furthermore, the swish activation function on the final output layer is preferred which differs from the optimal output activation functions of the previous benchmarks. With these initial indications of deep hedging complexity for realistic asset price processes, let us now analyse the deep hedging performance directly.

To analyse the deep hedging performance on the pseudo-real data, we again begin with a visualisation of the deep hedging strategy and hedge portfolio in 4.27 below. As mentioned in the experiment setup, the MVH pseudo-real data



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.27: The strategy and hedge for a random pseudo-real sample path for experiment one

experiment contains two Black-Scholes strategies for comparison. These strategies are not assumed to be the optimal solution; however, represent industry benchmarks. Firstly, we find that the two deep hedging networks are relatively close approximations of each other although the LSTM appears to be less sensitive to changes in the input compared to the RNN strategy; although, both deep hedging strategies differ significantly from the Black-Scholes hedges. These Black-Scholes strategies appear to be similar for the single sample path. Whilst the visualisation of the hedge portfolios in 4.27b omits a true market value option price process, the observed terminal payoff of the option is plotted. From this, we see that both of the deep hedging networks and benchmarks achieve a terminal portfolio value close to the payoff for the random sample path. The hedging portfolios show that significant differences between the intrinsic option prices exists between the approaches. These figures may indicate that neither approach represents the optimal hedging solution.From a single sample path perspective, we are unable to make any further inference and now consider the deep hedging performance on the test set.

The histogram of the deep hedging PnLs illustrated in 4.28 shows the empirical distribution of the replication errors of the deep hedging strategies over the 10 000 test samples. From this PnL comparison we see that the two deep

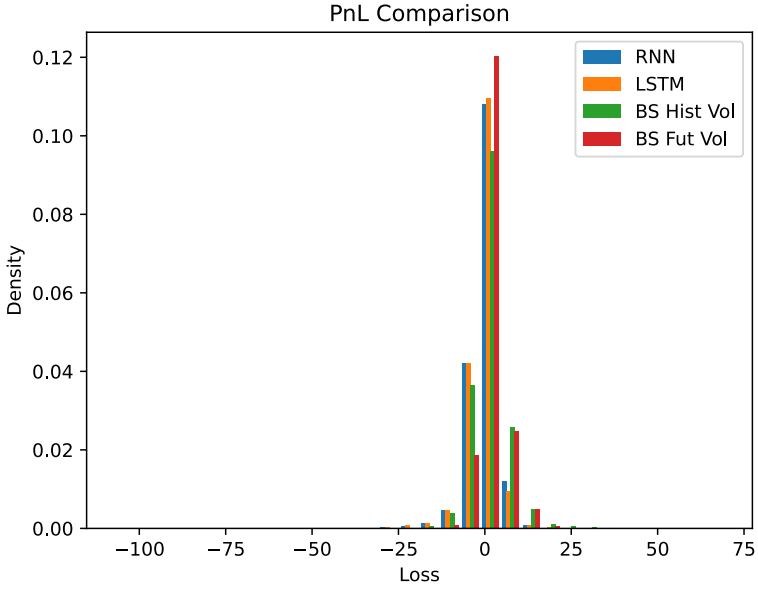


FIGURE 4.28: Profit and loss histogram for experiment one

hedging models provide comparable performance on the test set. The LSTM version appears to provide a slightly higher density around zero; however, neither network appears superior based on the empirical distributions. Compared to the benchmarks, the Black-Scholes future volatility benchmark provides a higher density at zero and comparable performance for the larger PnL values. The Black-Scholes historical volatility benchmark appears to be outperformed slightly with lower zero density and comparable density for the higher values. The numerical analysis tabulated in 4.14 below supports this observation. Here,

Approach	Expected Square Replication Error
BS Future	17.0200
BS Historical	27.8958
RNN	18.1237
LSTM	18.9320

TABLE 4.14: Expected square replication error for experiment one

we find the historical volatility benchmark experiences the worst performance by a significant margin. The future volatility benchmark; however, provides the best performance with both deep hedging models' performance trailing close behind. Interestingly, the RNN version slightly outperforms the LSTM on the test set.

An immediate observation based on both the loss values in Figure 4.28 and the ESRE values is that the replication error of based on the pseudo-real asset paths are significantly higher than those obtained in the above benchmarks. This ESRE is not necessarily indicative of a poor approximation of the optimal hedging strategies by the deep hedging and benchmark strategies.

Firstly, the asset paths for this experiment have $S_0 \approx 2500$ compared to $S_0 = 100$ for the benchmark price processes. The current paths consequently have a much higher magnitude and the absolute replication error may be significantly higher despite proportional similarities. For example, if we return to the Black-Scholes benchmark of 4.2.2 and make the alteration to have $S_0 = 2500$ instead of $S_0 = 100$ and $\sigma = 0.1$, then we find that the Black-Scholes strategy only provides a ESRE of 19.82. The fact that this example ESRE is close to the ESRE of the pseudo-real ESRE in 4.14 is merely a coincidence; however, it clearly shows that the ESRE cannot be taken by absolute value alone.

Secondly, the sample paths generated from the real-world measure $\mathbb{P}$ are likely to be more complex to hedge than those generated for the previous benchmarks. The benchmarks make restrictive assumptions about the underlying real-world measure. The pseudo-real paths make no such assumptions and, therefore, a higher level of hedging difficulty can be associated.

With the above two considerations, we can analyse the deep hedging performance as follows. The deep hedging models offer highly competitive performance for $\mathbb{P}$ asset sample paths. Whilst the future volatility benchmark outperformed both deep hedging models, such a benchmark is not necessarily a realistic industry performance. The benchmark utilises information which is only available within $\mathcal{F}_T$. Instead, the historical volatility benchmark offers a more realistic industry performance and the deep hedging models significantly outperform this benchmark. A realistic industry hedge commonly lies in between the historical and *crystal ball* benchmarks, where a trader utilises market sentiment of the future volatility similar to implied volatility. This motivates the realistic capabilities of deep hedging.

The RNN and LSTM architectures are highly comparable for deep hedging. We find that the relative difference between the two approaches is only 0.045 for the pseudo real MVH experiment which is similar to the relative differences observed throughout the benchmarks. Futhermore, the relatively symmetric distribution of the PnLs in Figure 4.28 implies that the deep hedging models have approximated either the optimal or close-to-optimal solution. Consequently, the first experiment supports deep hedging as an applicable approach to hedging real-world sample paths and provides a realistic performance of the approach under $\mathbb{P}$.

In the next experiment we will explore the deep hedging performance with observed real-world samples more directly and under a more traditional setting by changing our performance metric. Additionally, the following experiment is accompanied with a performance benchmark to compare the deep hedging performance against a realistic industry hedging performance.

4.3.3 Real Data

The final experiment is a deep hedging analysis on real data. Here, the deep hedging models are tested on the real-world sample paths to analyse their realistic performance. The benchmarks and the above pseudo-real data experiment establish the capabilities of the deep hedging approach, with the latter providing a close to realistic performance evaluation; however, common real-world implementation experiences some differences due to the single sample path problem. The main difference is the change from MVH towards local risk-minimisation.

The Risk Measure

Consider the scenario where there is only one observed sample path. Even if a trader partitions the available historical data according to the maturity of their desired option, a month in our case, there are very few samples for the deep hedging models to learn from. This problem is intensified under the MVH framework where the loss function is directly based on the terminal square replication error of the trading strategy. An industry solution for the single sample path problem is for the trader to consider a risk measure which related to local risk-minimisation instead of MVH. For this purpose, we require the following definition.

Definition 4.3.3
The daily PnL of a trading strategy φ is defined as

$$PnL_t^d(\varphi) = \Delta V_t(\varphi) - \Delta \pi_t,$$

where π_t is the market price of the option at time t and Δ is the change operator for the respective values.

In other words, the daily PnL measures the daily (assuming that the trading periods are daily) change in the portfolio value from the trading strategy and the change in the market prices. Therefore, a risk measure based on the daily PnL aims to minimise the difference in daily changes between the hedging portfolio

and the true option price. Such a risk measure provides a performance more appropriate for a single sample path since the single path has a PnL value for each trading period, whilst the MVH approach has one PnL value per path. This is a beneficial property even when a trader may be able to partition historical data into multiple sample paths since the number of sample paths remains relatively low for a deep learning problem.

A secondary reason for the use of the daily PnL follows from common industry standards. Traders for financial institutions are often required to operate on a mark-to-market structure. Such a structure requires the profit or loss of a position to be balanced on a periodic, often daily, basis. Consequently, the daily PnL would represent the daily balance under such a structure which is clearly of direct importance to the trader.

The daily PnL risk, known as the expected squared daily PnL, ESDPnL, which we implement is defined as

$$R^{daily}(\varphi) \coloneqq \mathbb{E}[\left(PnL^{d}(\varphi)\right)^{2} |\mathcal{F}_t],$$

Whilst not identical to the local risk-minimisation described in 3.2.1, the process is similar and draws direct influence from it.

With the appropriate change in the performance metric, we can now discuss the setup for the real data experiment.

The Setup

For this experiment we have two sub-experiments which we will compare directly. The first sub-experiment analyses deep hedging based purely on observed data whilst the second sub-experiment aims to enhance the training of the deep hedging models through the use of pseudo-real data. Both approaches are then compared on the same test set of real options data. Let us begin by describing the first sub-experiment setup.

As discussed in 4.3.1, we have roughly 4 years of daily options data for the S&P500 underlying asset. Unlike the previous experiment, we are unable to utilise the 20 years of historical closing prices for the S&P500 index since the daily PnL requires the market price of the option. As with all of the previous benchmarks and experiment, we assume the role of a trader which is hedging a one month vanilla call option. In practice, a month is roughly 22 trading days; therefore, we partition the available options data into 48 samples of 22 trading days. To achieve this partition, we restrict the historical S&P500 prices to the same range as the option data and partition into samples. For the market option prices, we calibrate a Heston model to the historical volatility surfaces on a daily basis and use the calibrated model to obtain a market calibrated option price which aligns to the correct maturity and strike which the trader is concerned with. At the same time, we are able to compute the Heston delta via finite differencing methods which serves as a realistic industry comparison. Note that the Heston hedging strategies represent a benchmark for the numerical experiment since the market option prices are obtained from a daily calibrated

Heston model. Therefore, the option prices reflected in the experiment are not necessarily the observed option prices. Such a difference is necessary since many of the option prices required do not exist in the market, or at least the historical data is unavailable. Futhermore, pricing an option from a calibrated model is common practice for non-existent options.

With the above data, we can now train and test the deep hedging approach and compare the performance against the delta hedging strategy provided by the Heston model. For this, we allocate 8 monthly partitions as the test, the final 8 partitions available to mimic an application setting, whilst the remaining 40 partitions compose the training data for the deep hedging approach. A problem which the reader may be concerned with at this stage is the very low number of training samples, especially for a deep reinforcement learning problem despite the change of risk measure. The second sub-experiment aims to provide a potential solution.

With the ability to generate pseudo-real sample paths from the method described in 4.3.2, our second sub-experiment explores the enhancement of the training process of the deep hedging models. In particular, we generate pseudo-real sample paths for the underlying asset and simulate an associated option price process as follows.

1. At the start of each sample, choose a random starting date from the range of available calibrated Heston parameters.
2. Generate option prices from the contiguous set of Heston parameters beginning with the above starting date.

Whilst the above simulation of market option prices is sub-optimal since the simulation of the parameters and the underlying asset is independent, the discussion of their joint simulation in 4.3.2 currently prevents a better solution. Therefore, the second sub-experiment explores the following question. Can the use of possibly sub-optimal pseudo-real samples for training enhance the strictly real-world performance of deep hedging models?

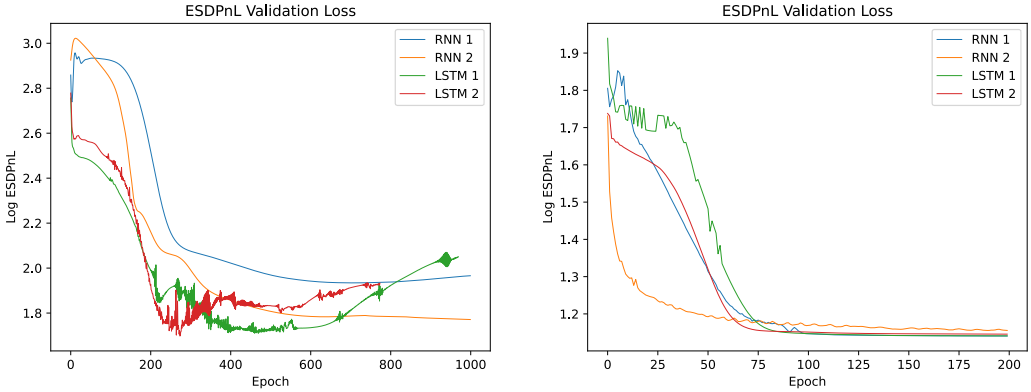
The second sub-experiment utilises 20 000 samples for the training process and is analysed on the same test set as the first sub-experiment. For these samples, the starting asset prices follow $S_0 \sim \mathcal{N}(2000, 200)$. Beyond this, the setup between the two sub-experiments remain consistent. As with the above experiment, the LM-feature set since S_0 is not constant. Finally, the Heston benchmark's hyperparameter space is used for the deep hedging models.

Now that we have described the two sub-experiments for this real data experiment, we can now proceed to analysing the results and evaluating the realistic real-world performance of deep hedging.

The Analysis

As with the first numerical experiment, the number of hyperparameter queries explored by the Bayesian optimisation process is reduced compared to the benchmarks. In particular, the first sub-experiment (sub1), where the deep

hedging models are trained on the observed sample paths alone, explores 30 queries per architecture whilst the second sub-experiment (sub2), where the deep hedging models are trained with pseudo real sample paths, explores 20 queries per architecture. For Sub1, each query is trained for 1000 epochs since each epoch only contains 40 samples. For Sub2, however, each query is trained for 200 epochs following its sample size of 20 000. The Sub1 Tensorboard and Sub2 Tensorboard are available respectively for those readers interested in the full hyperparameter tuning summary; however, an appropriate extract is displayed in Figure 4.29 below. The sub1 validation ESDPnL extract in 4.29a



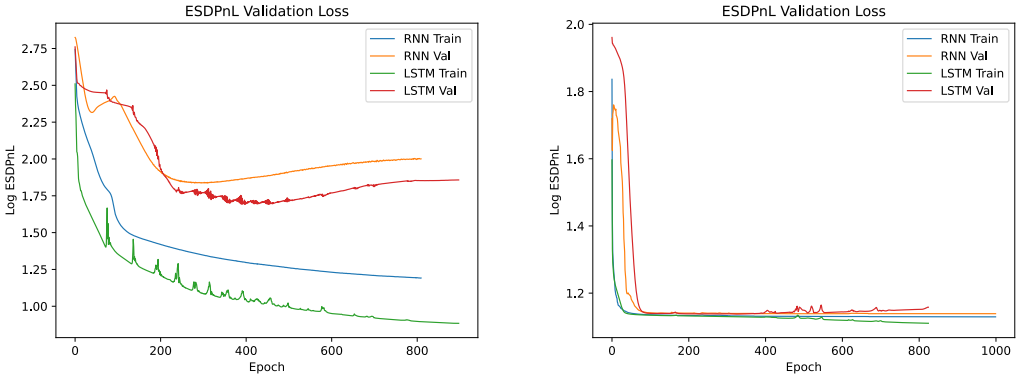
(A) Extract for Sub1 (B) Extract for Sub2

FIGURE 4.29: Hyperparameter tuning validation loss extract for experiment two

summarises a common trend prevalent in the sub1 tuning. The 1000 epochs available to each query appears to be sufficient with almost all of the queries either reaching a plateau or even displaying signs of overfitting by an increase in validation loss after achieving a minimum. In a similar vain, the sub2 validation ESDPnL extract in 4.29b summarises the level of convergence associated with the deep hedging models in sub2. Comparing both sub-experiments, the use of pseudo real sample paths for the training process appears to help prevent overfitting of the models; however, this is expected due to the additional number of paths and the same generative source.

A brief analysis of the hyperparameters in both Tensorboards shows a strong preference towards batch normalisation by both sub-experiments. The tuning of sub1 shows that the most of the top performing queries contain only two hidden layers whilst the best queries for sub2 contain three hidden layers. Such a preference could follow from the easier overfitting within sub1 due to the lower sample size. To further analyse these hyperparameters, we consider the fitting of the top performing queries for each sub-experiment.

The model fitting of the top hyperparameter queries are presented in the sub1 Tensorboard and sub2 Tensorboard respectively with the best models displayed in Figure 4.30 below. As with the hyperparameter tuning in Figure 4.29 and the model fitting in experiment one, the sub1 model fitting in Figure 4.30a shows that a significant difference exists between the training and validation



(A) Sub1 fit (B) Sub2 fit

FIGURE 4.30: Model fitting the best models for experiment two

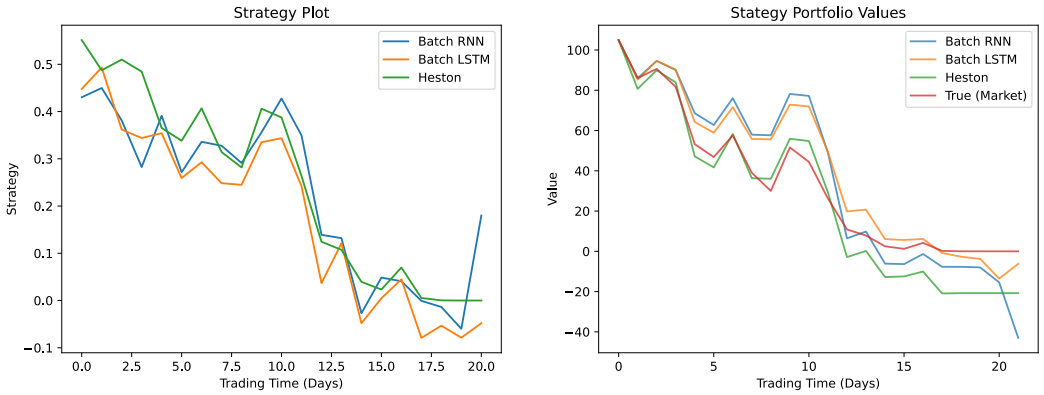
loss values. Whilst the training loss continues to decrease, the validation loss is indicative of the deep hedging models overfitting for a high number of epochs. The model fitting for sub2 in Figure 4.30b displays a less significant difference between the training and validation loss values. The parameters for these optimal models are tabulated in 4.18. As seen in the hyperparameter tuning

Hyperparameter	Optimal Value	
	RNN	LSTM
Sub-experiment One		
Batch Normalisation	True	True
(N_1, N_2, N_3)	(10, 10, -)	(10, 100, -)
$(\sigma_1, \sigma_3, \sigma_3)$	(*elu*, *elu*, -)	(*elu*, *tanh*, -)
σ_{out}	*swish*	*tanh*
Sub-experiment Two		
Batch Normalisation	True	True
(N_1, N_2, N_3)	(100, 10, 80)	(10, 100, 100)
$(\sigma_1, \sigma_3, \sigma_3)$	(*tanh*, *elu*, *tanh*)	(*elu*, *elu*, *tanh*)
σ_{out}	*tanh*	*tanh*

TABLE 4.15: Optimal hyperparameters for neural network models for experiment two

analysis, two and three hidden layers are the optimal choice for the first and second sub-experiments respectively. An important sub1 model fitting observation is that the RNN models are consistently outperformed by the LSTM models in the sub1 Tensorboard. This is explored further by the following deep hedging analysis.

With the analysis of the deep hedging performance with respect to the daily PnL, we compare each sub-experiment individually before an overall comparison. Consequently, we begin the sub1 deep hedging analysis with a single sample path visualisation in Figure 4.31 below. As explained by the experiment introduction, the Heston delta strategy represents an industry benchmark



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.31: The strategy and hedge for a real sample path for sub-experiment one

to compare the deep hedging performance against. Furthermore, a true market price for the option is available as opposed to the terminal payoff restriction for experiment one in section 4.3.2.

From Figure 4.31a, we notice that the deep hedging strategies appear to follow the general structure of the Heston strategy. Interestingly, both of the deep hedging models provide strategies which are negative towards the end of the option life despite delta commonly restricted to the unit interval. Since we do not explicitly make such a restriction with the deep hedging models, although one could easily achieve the restriction through the output activation function, this could be an initial indication of overfitting of the deep hedging models. Beyond this; however, it is difficult to deduce which strategy is a closer approximation of the Heston strategy although the significant jump of the RNN model at the end trading period appears to be an outlier that will prove important in the histogram to follow. Similarly, the resultant hedge portfolios in Figure 4.31b do not clearly indicate a hedge which offers a better daily PnL for the single observed sample path. Instead, we analyse the models with respect to the test set.

Rather than the distribution of the terminal PnL values, we must now consider the distribution of the daily PnL values which is illustrated in Figure 4.32 below. The histogram shows that the Heston model provides a higher daily PnL density close to zero when compared to the deep hedging models. As a result, the Heston model appears to outperform the RNN based on the daily PnL when combined with the higher density of the RNN model for larger PnL values. As mentioned with the single sample path analysis, the RNN appears to produce outliers which are present with neither the Heston nor LSTM models. When comparing the LSTM and Heston models, the performance is closer. Whilst the Heston model produces a higher density close to zero, the Heston strategy also produces a higher density for large PnL values compared to the LSTM model. Consequently, we analyse the numerical performance.

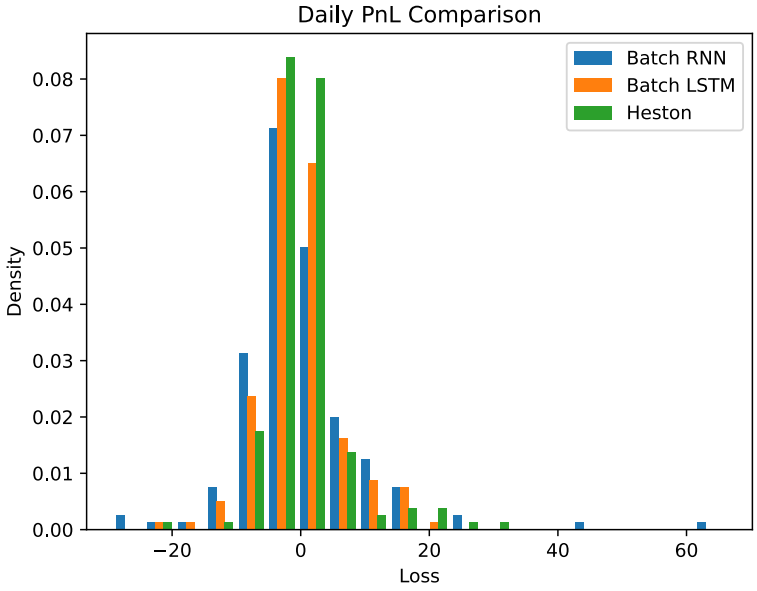


FIGURE 4.32: Profit and loss histogram for experiment two, sub-experiment one

The ESDPnL performance of the deep hedging and Heston models is tabulated in 4.16. The numerical comparison solidifies that the Heston model outperforms

Approach	Expected Square Daily PnL
Heston	38.5925
RNN	97.0366
LSTM	36.1055

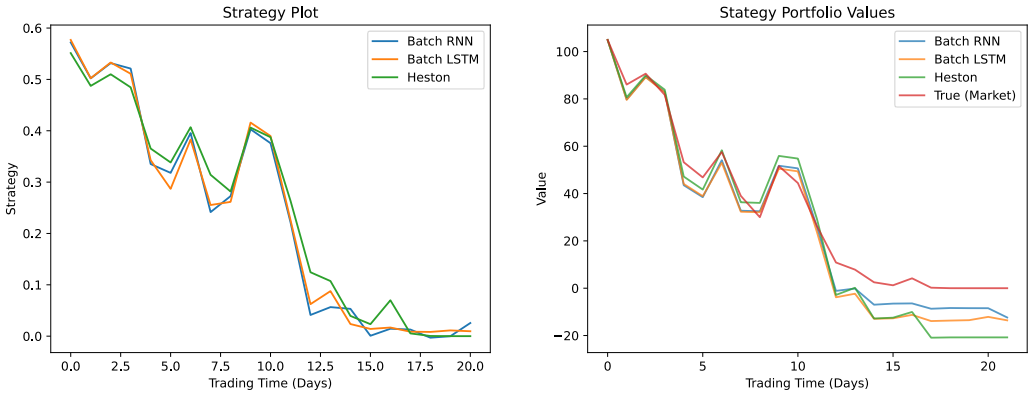
TABLE 4.16: Expected square daily PnL for sub-experiment one

the RNN model on the test set which is implied by the above histogram. The significant difference between the two ESDPnL values is largely based on the outlier values which are magnified when using the squared difference. Furthermore, the LSTM model outperforms the Heston benchmark on the test set. The Heston ESDPnL over all 48 sample paths is approximately 41, as we will see in the sub2 analysis, which implies that the LSTM model offers a competitive performance against the Heston model; however, the drastic difference between the RNN and LSTM model could be indicative of a high overfitting risk associated with deep hedging with few sample paths.

The RNN model achieved a 28.01 ESDPnL versus the LSTM's 13.38 ESDPnL on the training data with 68.60 and 49.16 based on the validation set respectively. The lack of data available causes the validation performance to be unstable and overfitting difficult to avoid. Despite the RNN model yielding poor performance against the LSTM based on the training and validation data, the significantly higher ESDPnL in 4.16 exemplifies the problem of overfitting whilst the LSTM displays the realistic capabilities deep hedging if avoided. In an attempt to

address this problem, we now consider the deep hedging performance of the second sub-experiment.

For the initial analysis of the second sub-experiment, we analyse the deep hedging performance based on all 48 real samples available. Again, we begin with the single sample visualisation in Figure 4.33 below. The same sample path as 4.31



(A) Hedging strategies provided by each model (B) Associated hedging portfolios

FIGURE 4.33: The strategy and hedge for a real sample path for sub-experiment two

is used for the sub2 analysis. From 4.33a, it is clear that the sub2 deep hedging strategies are a closer approximation of the Heston strategy for the given path. Secondly, we find that the sub2 models do not yield the negative strategies as seen with the sub1 models despite the absence of mathematical restriction. Similarly, the hedging portfolios in 4.33b illustrate the closer approximation of both the Heston hedge and the true market by both deep hedging strategies. Based on the single sample path, the sub2 models appear to be a significantly better approximation of the Heston benchmark; consequently, let us consider the 48 real samples as our sub2 test set.

The daily PnL histogram in 4.34 below continues to display the close approximation of the Heston delta hedging strategy by the deep hedging models. Firstly, we cannot directly compare the histogram of 4.32 and 4.34 since they are based on different data sets; however, due to the approximation accuracy, it is not clear which sub2 approach is superior. For this, we require the numerical analysis tabulated in 4.17 below. Again, we find that the LSTM deep hedging

Approach	Expected Square Daily PnL
Heston	41.0810
RNN	43.5294
LSTM	36.2508

TABLE 4.17: Expected square daily PnL for sub-experiment two

model outperforms the Heston strategy. Secondly, we find that the RNN model greatly improves its performance when compared to sub1 and offers competitive performance with the LSTM and Heston approaches.

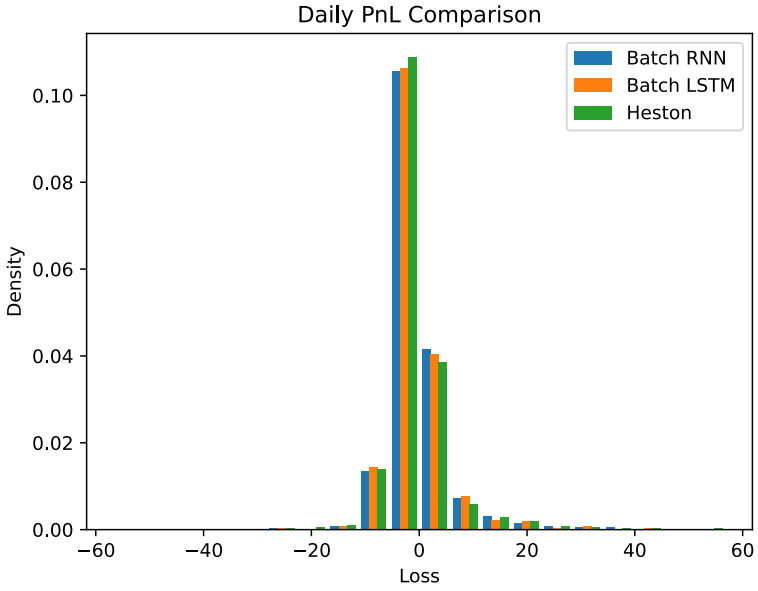


FIGURE 4.34: Profit and loss histogram for experiment two, sub-experiment two

These results appear to show that the deep hedging approach can approximate the optimal solution. Furthermore, the results obtained from sub2 support the claim that the pseudo samples can enhance the performance of deep hedging models even if these samples are sub-optimal from a generation perspective. To compare the performance of the sub1 and sub2 approaches directly, our final analysis is to compare their performance on the same test set.

Given the training data required by sub1, we obtain the numerical performance of sub2 on the 8 samples utilised as a test set by sub1 and tabulate the comparison in 4.18 below. Despite the sub2 LSTM outperforming the Hes-

Approach	Expected Square Daily PnL	
	Sub-experiment 1	Sub-experiment 2
Heston	38.5925	
RNN	97.0366	40.4643
LSTM	36.1055	39.658

TABLE 4.18: Expected square daily PnL comparison for experiment two

ton benchmark on the collection of all real samples, the sub2 LSTM is slightly outperformed by the Heston model on the 8 sample test set. Therefore, the sub1 LSTM outperforms the sub2 LSTM. On the other hand, the sub2 RNN drastically outperforms the sub1 RNN due to the poor performance of the sub1 RNN discussed above. Whilst the sub2 models do not outperform the Heston hedge on the small test set, the consistency implies that the additional samples of the sub2 framework helps to prevent overfitting in the deep hedging models. Given the performance measure instability for a small number of samples, it

is likely that the sub2 models could outperform the sub1 models on a larger out-of-sample test set.

An important remark about the performance throughout this experiment is that the Heston benchmark is obtained under ideal circumstances. The market prices of the options are obtained from a daily calibrated Heston model which also produces a delta strategy. Consequently, some of the variability in observed option prices may be removed. Therefore, the Heston strategy represents a very strong benchmark which the deep hedging strategies provide competitive performance against. Additionally, the deep hedging strategies may not necessarily be approximating the benchmark since the benchmark need not be the optimal solution to the optimisation problem. The LSTM outperforming the Heston model in Table 4.17 supports the claim that the LSTM model is a closer approximation of the optimal solution than the Heston strategy. We now summarise our analysis for the second experiment.

Experiment two provides an important analysis of deep hedging from an application perspective. Firstly, experiment two extends the analysis of deep hedging under the MVH framework to a local risk-minimisation based measure, contributing to the convex risk measures explored by Buehler et al., 2019. Such a risk measure is important for many traders who rely on trading within a mark-to-market structure. Secondly, the experiment is able to illustrate the solution to few real-world sample paths provided by pseudo-real data generation. Building on the analysis of experiment one in 4.3.2, the second experiment is able to provide a realistic deep hedging performance comparison for industry application. The deep hedging approach is shown to be competitive against market model based hedging approaches with the ability to outperform even the best market model hedges as described by the above remark.

Chapter 5

Conclusion

Throughout the dissertation we have analysed the problem of hedging within incomplete markets and offered an extension to the deep hedging solution presented by Buehler et al., 2019. We began by introducing the required theory for both mathematical finance and machine learning before covering the incomplete market and deep hedging problems in greater detail. The Results section in chapter 4 numerically analyse the performance capabilities of deep hedging compared to the more classical and theoretical counterparts. Here, the deep hedging approach is shown to have extremely competitive performance in all scenarios ranging from basic tree-based market processes to real-world markets. From this, a few important conclusions have been made.

5.1 Conclusions

An essential observation from the benchmarks presented in 4.2 is that the deep hedging approach can approximate the theoretically optimal solution arbitrarily well without knowledge of such a solution. From the dynamic programming benchmark, the optimal strategy can be computed within the incomplete market and the difference between the deep hedging approach and the optimal solution is shown to be insignificant. The Black-Scholes and Heston benchmarks extend the incompleteness of the market; whereby, the deep hedging approach is again shown to perform as effectively as the theoretical approximations of the optimal hedging strategies. Interestingly, throughout these benchmarks, the log transformation shows no significant improvement to the deep hedging feature set. Furthermore, the LSTM architecture consistently outperforms the RNN architecture; however, the difference is often insignificant.

Following the real data experiments, a few interesting observations are present. Firstly, the deep hedging approach appears to significantly outperform a realistic Black-Scholes hedging strategy based on pseudo-real asset paths. Even when providing the Black-Scholes model with future information, the deep hedging models are able to provide competitive performance against the Black-Scholes model. Such a result strongly supports the ability of the deep hedging approach to improve incomplete market hedging within industry application.

The second experiment illustrates that the deep hedging approach can follow the daily PnL/local risk-minimisation framework. For strictly real-data utilisation,

deep hedging faces a lack of data problem both within this dissertation and industry application. Whilst the LSTM deep hedge was able to outperform the Heston benchmark here, the observation must be prefaced with the fact that a very small test sample size was used which allows for significant variance. An interesting analysis presented is that of the second sub-experiment whereby the training of the deep hedging models is enhanced by pseudo-real data generation. Data generation provides a great opportunity to solve the lack of real-world asset samples. Despite the sub-optimal options market generation, the pseudo-real data was able to effectively train the deep hedging models. With this, the LSTM model was able to outperform the Heston benchmark when tested on all 4 years worth of monthly samples. Furthermore, the pseudo-real data was able to improve the stability of the deep hedging model performance by effectively expanding its training samples and reducing its tendency to overfit. A major pitfall of this experiment is the lack of options data. Although this pitfall is still present in industry, we believe that further analysis could provide more support for the enhancement.

An interesting extension still to be explored is the use of more powerful reinforcement learning network architectures; for example, a Transformer-based architecture. Such an architecture could provide better performance, especially for the more complex real-data processes. In addition, extensions into the generation of financial markets and options markets appear to be interesting avenues for improving the performance of deep hedging models in industry application. The inclusion of market frictions such as transaction costs could be explored to further obtain a realistic deep hedging performance as well as additional claims other than vanilla call options.

Finally, we remark that the analysis of these results were based of quadratic criterion. A trader may argue that such a criterion is undesirable as they need not treat profit and loss equally; however, beyond extending Buehler et al., 2019, the choice was two-fold. Firstly, the quadratic criterion, MVH in particular, offers both a dynamic programming and martingale solution. Secondly, traders often follow a risk mitigation perspective whereby the replication of the claim is viewed without greater concern for whether amounts were profits or losses. Consequently, we arrive at our recommendations for such traders.

5.2 Recommendations

The conclusions outlined above provide a recommendation to industry traders. Deep hedging under real-world market conditions can provide either improved or highly competitive performance compared to both the Black-Scholes and Heston hedging approaches. The use of real-world measure data generation to assist in the training of the deep hedging model proves highly beneficial in training deep hedging models and should be considered when using deep hedging in industry application.

Whilst Buehler et al., 2019 explores transaction costs in their deep hedging approach, we have omitted this friction within the numerical experiments which

represents a pitfall of the analysis; although the work by Buehler shows that deep hedging is robust enough to include transaction cost. Another pitfall of the deep hedging approach would be utilisation under market conditions which are significantly different from any of those contained in the observed history. The model will not be trained for such conditions which could result in poor hedging performance. These pitfalls should be considered before applying deep hedging with the recommendations outlined above.

Bibliography

Alla, Samhita (2021). *Multi-Layer Deep RNN [image]*. Online; accessed December 10, 2022. URL: https://blog.paperspace.com/advanced-recurrent-neural-networks-deep-rnns/.

Ben-Tal, Aharon and Marc Teboulle (2007). "An old-new concept of convex risk measures: the optimized certainty equivalent". In: *Mathematical Finance* 17.3, pp. 449–476.

Bertsimas, Dimitris, Leonid Kogan, and Andrew W Lo (2001). "Hedging derivative securities and incomplete markets: An ϵ-arbitrage approach". In: *Operations research* 49.3, pp. 372–397.

Bishop, Christopher M and Nasser M Nasrabadi (2006). *Pattern recognition and machine learning*. Vol. 4. 4. Springer.

Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3, pp. 637–654. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831029.

Bouleau, Nicolas and Damien Lamberton (1989). "Residual risks and hedging strategies in Markovian markets". In: *Stochastic processes and their applications* 33.1, pp. 131–150.

Buehler, Hans et al. (2019). "Deep hedging". In: *Quantitative Finance* 19.8, pp. 1271–1291.

Buehler, Hans et al. (2020). "Generating financial markets with signatures". In: *Available at SSRN 3657366*.

Carverhill, Andrew P and Terry HF Cheuk (2003). "Alternative neural network approach for option pricing and hedging". In: *Available at SSRN 480562*.

Černỳ, Aleš (2004). "Dynamic programming and mean-variance hedging in discrete time". In: *Applied Mathematical Finance* 11.1, pp. 1–25.

Chevyrev, Ilya and Andrey Kormilitzin (2016). "A primer on the signature method in machine learning". In: *arXiv preprint arXiv:1603.03788*.

Chevyrev, Ilya and Harald Oberhauser (2018). "Signature moments to characterize laws of stochastic processes". In: *arXiv preprint arXiv:1810.10971*.

Duffie, Darrell and Henry R Richardson (1991). "Mean-variance hedging in continuous time". In: *The Annals of Applied Probability*, pp. 1–15.

Duffie, Darrell et al. (1997). "Hedging in incomplete markets with HARA utility". In: *Journal of Economic Dynamics and Control* 21.4, pp. 753–782. ISSN: 0165-1889. DOI: https://doi.org/10.1016/S0165-1889(97)00002-X. URL: https://www.sciencedirect.com/science/article/pii/S016518899700002X.

El Karoui, Nicole and Marie-Claire Quenez (1995). "Dynamic programming and pricing of contingent claims in an incomplete market". In: *SIAM journal on Control and Optimization* 33.1, pp. 29–66.

Föllmer, Hans and Yuri M Kabanov (1997). "Optional decomposition and Lagrange multipliers". In: *Finance and Stochastics* 2.1, pp. 69–81.

Föllmer, Hans and Alexander Schied (2011). *Stochastic finance: an introduction in discrete time*. Walter de Gruyter.

Föllmer, Hans, Martin Schweizer, et al. (1990). *Hedging of contingent claims under incomplete information*. Rheinische Friedrich-Wilhelms-Universität Bonn.

Föllmer, Hans and Dieter Sondermann (1986). "Hedging of non-redundant contingent claims, Contributions to Mathematical Economics, A. Mas-Colell and W. Hildenbrand ed". In: *North-Holland, Amsterdam* 205, p. 223.

Friz, Peter K and Martin Hairer (2020). *A course on rough paths*. Springer.

Glasserman, Paul (2013). *Monte Carlo methods in financial engineering*. Vol. 53. Springer Science & Business Media.

Gourieroux, Christian, Jean Paul Laurent, and Huyên Pham (1998). "Mean-variance hedging and numeraire". In: *Mathematical finance* 8.3, pp. 179–200.

Gretton, Arthur et al. (2012). "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1, pp. 723–773.

Gyurkó, Lajos Gergely et al. (2013). "Extracting information from the signature of a financial data stream". In: *arXiv preprint arXiv:1307.7244*.

Harrison, J Michael and David M Kreps (1979). "Martingales and arbitrage in multiperiod securities markets". In: *Journal of Economic theory* 20.3, pp. 381–408.

Harrison, J Michael and Stanley R Pliska (1981). "Martingales and stochastic integrals in the theory of continuous trading". In: *Stochastic processes and their applications* 11.3, pp. 215–260.

Heath, David, Eckhard Platen, and Martin Schweizer (2001a). "A comparison of two quadratic approaches to hedging in incomplete markets". In: *Mathematical finance* 11.4, pp. 385–413.

– (2001b). "A comparison of two quadratic approaches to hedging in incomplete markets". In: *Mathematical finance* 11.4, pp. 385–413.

Henderson, Vicky and David Hobson (2004). "Utility indifference pricing-an overview". In: *Volume on Indifference Pricing*.

Henry-Labordere, Pierre (2019). "Generative models for financial data". In: *Available at SSRN 3408007*.

Heston, Steven L (1993). "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The review of financial studies* 6.2, pp. 327–343.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hodges, Stewart D and Anthony Neuberger (1989). "Optimal replication of contingent claims under transaction costs." In: *The Review of Futures Markets* 8.2, pp. 223–239.

Hodoshima, Jiro, Tetsuya Misawa, and Yoshio Miyahara (2018). "Comparison of utility indifference pricing and mean-variance approach under normal mixture". In: *Finance Research Letters* 24, pp. 221–229. ISSN: 1544-6123. DOI: https://doi.org/10.1016/j.frl.2017.09.008. URL: https://www.sciencedirect.com/science/article/pii/S1544612317304300.

Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2, pp. 251–257.

Hutchinson, James M, Andrew W Lo, and Tomaso Poggio (1994). "A nonparametric approach to pricing and hedging derivative securities via learning networks". In: *The journal of Finance* 49.3, pp. 851–889.

Ilhan, Aytaç, Mattias Jonsson, and Ronnie Sircar (2009). "Optimal static-dynamic hedges for exotic options under convex risk measures". In: *Stochastic Processes and their Applications* 119.10, pp. 3608–3632.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kondratyev, Alexei and Christian Schwarz (2019). "The market generator". In: *Available at SSRN 3384948*.

Kondratyev, Alexei, Christian Schwarz, and Blanka Horvath (2020). "Data anonymisation, outlier detection and fighting overfitting with Restricted Boltzmann Machines". In: *Outlier Detection and Fighting Overfitting with Restricted Boltzmann Machines (January 27, 2020)*.

Kramkov, Dmitrij O (1996). "Optional decomposition of supermartingales and hedging contingent claims in incomplete security markets". In: *Probability Theory and Related Fields* 105.4, pp. 459–479.

Lyons, Terry J (1995). "Uncertain volatility and the risk-free synthesis of derivatives". In: *Applied mathematical finance* 2.2, pp. 117–133.

Malamud, Semyon, Eugene Trubowitz, and Mario V Wüthrich (2013). "Indifference pricing for CRRA utilities". In: *Mathematics and financial economics* 7.3, pp. 247–280.

Malliaris, Mary and Linda Salchenberger (1993). "A neural network model for estimating option prices". In: *Applied Intelligence* 3.3, pp. 193–206.

Maruyama, Gisiro (1955). "Continuous Markov processes and stochastic equations". In: *Rendiconti del Circolo Matematico di Palermo* 4, pp. 48–90.

Mavuso, Melusi (2019). *Mathematical Finance*. University of Cape Town, Department of Statistics.

Merton, Robert C (1973). "Theory of rational option pricing". In: *The Bell Journal of economics and management science*, pp. 141–183.

Møller, Thomas (2001). "Risk-minimizing hedging strategies for insurance payment processes". In: *Finance and Stochastics* 5.4, pp. 419–446.

Musiela, M and M Rutkowski (2010). "Martingale Methods in Financial Modelling ser". In: *Stochastic Modelling and Applied Probability*. Springer Berlin Heidelberg.

Ouwehand, Peter (2015). *Stochastic Financial Modelling in Continuous–Time*. African Institute of Financial Markets and Risk Management.

Pham, Huyên, Thorsten Rheinländer, and Martin Schweizer (1998). "Mean-variance hedging for continuous processes: new proofs and examples". In: *Finance and Stochastics* 2.2, pp. 173–198.

Protter, Philip E (2005). "Stochastic differential equations". In: *Stochastic integration and differential equations*. Springer, pp. 249–361.

Rheinländer, Thorsten and Martin Schweizer (1997). "On L^2-projections on a space of stochastic integrals". In: *The Annals of Probability* 25.4, pp. 1810–1831.

Rocca, Joseph (2019). *Understanding Variational Autoencoders*. Online; accessed December 10, 2022. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

Ruf, Johannes and Weiguan Wang (2019). "Neural networks for option pricing and hedging: a literature review". In: *arXiv preprint arXiv:1911.05620*.

– (2020). "Neural networks for option pricing and hedging: a literature review". In: *Journal of Computational Finance, Forthcoming*.

– (2021). "Hedging with linear regressions and neural networks". In: *Journal of Business & Economic Statistics* just-accepted, pp. 1–33.

Schäfer, Anton Maximilian and Hans Georg Zimmermann (2006). "Recurrent neural networks are universal approximators". In: *International Conference on Artificial Neural Networks*. Springer, pp. 632–640.

Schweizer, Martin (1988). "Hedging of options in a general semimartingale model". PhD thesis. ETH Zurich.

– (1991). "Option hedging for semimartingales". In: *Stochastic processes and their Applications* 37.2, pp. 339–363.

– (1994). "Approximating random variables by stochastic integrals". In: *The Annals of probability*, pp. 1536–1575.

– (1999). *A guided tour through quadratic hedging approaches*. Tech. rep. SFB 373 Discussion Paper.

Shahriari, Bobak et al. (2015). "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175.

Sohn, Kihyuk, Honglak Lee, and Xinchen Yan (2015). "Learning structured output representation using deep conditional generative models". In: *Advances in neural information processing systems* 28.

Sutcliffe, Charles and Fei Chen (2011). "Pricing and Hedging Short Sterling Options Using Neural Networks". In: *Available at SSRN 1929767*.

Wiese, Magnus et al. (2020). "Quant gans: Deep generation of financial time series". In: *Quantitative Finance* 20.9, pp. 1419–1440.

Yu, Xinjie and Mitsuo Gen (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media.

Zhang, Aston et al. (2021). "Dive into Deep Learning". In: *arXiv preprint arXiv:2106.11342*.