IMPERIAL

Imperial College London

Department of Mathematics

Machine Learning-Driven Hedging Strategies with Neural Stochastic Differential Equations

Author: Manuel Lazaro Alonso (CID: 02357669)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2023-2024

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

First, I would like to thank my industry supervisor Gordon Lee, for giving me the chance to work in a project as interesting and fun as this one. In addition, I want to thank him for guiding and helping me throughout the project, and for pushing me harder to achieve greater results. Also, I want to thank my internal supervisor Cris Salvi for his advises, and help.

Access to data was made possible through funding provided by the MSc in Mathematics and Finance, Imperial College London.

Finally but not least, I would like to thank my family, especially my Mom and Dad for their unconditional support and love.

Abstract

This dissertation presents a machine learning-driving framework offering a practical implementation of pricing and hedging with real market data for incomplete markets with transaction costs. Our framework consist in implementing a Neural Stochastic Differential Equation (Neural SDE) as market generator, for our hedging models based on neural networks. We implement this model-free and data-driven framework with SP500 market data to simple options like European Call options, and more complex products like Asian options, Lookback options, and even multivariate settings like Rainbow options.

With this framework we are able to measure the statistical arbitrage of the market data, and develop optimal strategies to take advantage of it. Also, we develop two different types of hedging strategies, based in our risk preference. We can choose between a hedging strategy that while is hedging a liability, it also gets influenced by the statistical arbitrage of the market; and a hedging strategy that disregards the statistical arbitrage in the market, and only focus in hedging the derivative. In addition, we incorporate a risk aversion parameter to our trading strategies, to personalize even more our risk preferences.

Contents

1 Pricing and Hedging in Imperfect Markets **11**
- 1.1 Black-Scholes . . . 11
  - 1.1.1 Pricing a European Call Option . . . 12
  - 1.1.2 Delta Hedging in BS . . . 13
- 1.2 Settings for a Discrete Incomplete Market . . . 14
  - 1.2.1 Trading Strategies . . . 15
- 1.3 Convex Risk Measures and Indifference Pricing . . . 15
  - 1.3.1 Convex Risk Measure . . . 15
  - 1.3.2 Indifference Pricing . . . 16
  - 1.3.3 Exponential Utility Indifference Pricing . . . 16

2 Neural Networks (NN) **18**
- 2.1 Feed Forward Neural Networks (FNN) . . . 18
  - 2.1.1 General Architecture . . . 18
  - 2.1.2 Activation functions . . . 19
  - 2.1.3 Universal Approximation . . . 21
- 2.2 Training and Optimizers . . . 21
  - 2.2.1 Loss Function . . . 21
  - 2.2.2 Stochastic Gradient Descent . . . 22
  - 2.2.3 Adam Optimizer . . . 24
  - 2.2.4 Back-Propagation . . . 24
- 2.3 Recurrent Neural Network (RNN) . . . 25
  - 2.3.1 Standard Recurrent Neural Network . . . 25
  - 2.3.2 Long Short-Term Memory (LSTM) . . . 26

3 Market Generator: Neural SDE **28**
- 3.1 Neural SDE . . . 28
- 3.2 Non-Adversary training with Signature Kernels . . . 28
  - 3.2.1 Signature Transformation . . . 28
  - 3.2.2 Signature Kernels . . . 29
  - 3.2.3 Signature Kernel Scores . . . 30

4 Implementation of our Deep Hedging framework **32**
- 4.1 Implementation of the Deep Hedging model . . . 32
  - 4.1.1 Estimating the Indifference Price . . . 32
  - 4.1.2 Estimating the Hedging Ratios under a Risk-Neutral Measure . . . 33
  - 4.1.3 Architecture and training optimization . . . 34
- 4.2 Implementation of the Neural SDE . . . 35
  - 4.2.1 Evaluation Metrics . . . 35

5 Numerical Results and Discussion **37**
5.1 Black-Scholes Benchmark 37
5.2 European Call Option of the SP500 41
5.2.1 The Neural SDE: Generating SP500 data 42
5.2.2 Deep Hedging Model 44
5.3 Exotic Options 55
5.3.1 Asian Option 55
5.3.2 Lookback Option 60
5.3.3 Rainbow Option 65

A More results **74**
A.1 LSTM for European Call in SP500 74
A.2 Generated data results for Asian Option 80
A.3 Generated data results for Lookback Option 82

Bibliography **86**

List of Figures

1.1 The histogram of the PnL of following the Black-Scholes delta hedging strategy. 14

2.1 Graphical representation of a neural network with $r = 3, I = d_0 = 4, d_1 = 6, d_2 = 4$ and $O = d_3 = 3$. Source from Deep Learning lecture notes [18] . . 19
2.2 Representing graphically the RNN structure folded and unfolded. Source from [19] 26
2.3 The graphical representation of a LSTM cell structure. Source from [12]. . . 27

3.1 On the left are the individual channels (γ^x, γ^y) of a 2D path γ. In the middle are the channels reparametrized under $\varphi : t \mapsto t^2$. On the right are the path γ and its reparametrized version $\gamma \circ \varphi$. The two curves overlap, meaning that the reparametrization φ represents irrelevant information if one is interested in understanding the shape of γ. Figure taken from [36]. . 29

5.1 The hedging ratios over asset price at the 20th step time for different risk aversions and cost levels for the BS model using FNN. 39
5.2 The hedging ratios over asset price at the 20th step time for different risk aversions and cost levels for the BS model using the LSTM. 39
5.3 The hedging ratios over time for different risk aversions and cost levels for the BS model for FNN. 40
5.4 The hedging ratios over time for different risk aversions and cost levels for the BS model for LSTM. 40
5.5 The PnLs for the BS model for different risk aversion with no transaction costs for FNN. 41
5.6 The PnLs for the BS model for different risk aversion with no transaction costs for LSTM. 41
5.7 Qualitative plot of the paths generated with the paths of real market data. 42
5.8 The figure compares the marginal distribution of the paths for real and generate data for different times. 42
5.9 Qualitative plot of the autocorrelation scores with 95% interval at different lags. 43
5.10 The cross-correlation matrices of the generated and real data. 44
5.11 The hedging ratios over asset price at time step $t = 20$ following the Delta Z strategy for different risk aversion and cost levels. 46
5.12 The hedging ratios over asset price at time step $t = 20$ following the Delta Q strategy for different risk aversion and cost levels. 46
5.13 The hedging ratios over asset price at time step $t = 20$ following the Delta 0 strategy for different risk aversion and cost levels. 46
5.14 We compare the hedging ratio over price plot along different times for the strategy delta 0 between different risk aversions $\lambda = 1$ and $\lambda = 10$ and for different cost levels. 47

5.15 We compare the hedging ratio over price plot along different times between both strategies Z and Q for $\lambda = 1$ and for different cost levels. 48
5.16 We compare the hedging ratio over price plot along different times between both strategies Z and Q for $\lambda = 10$ and for different cost levels. 49
5.17 The hedging ratios over time for the Delta Z strategy for different risk aversions and cost levels. 50
5.18 The hedging ratios over time for the Delta Q strategy for different risk aversions and cost levels. 50
5.19 The hedging ratios over time for the Delta 0 strategy for different risk aversions and cost levels. 50
5.20 The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for generated data. 51
5.21 The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for generated data. 51
5.22 The PnLs of the strategy Q and Z for different risk aversions and no transaction costs for generated data. 52
5.23 The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real market data. 53
5.24 The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for real market data 53
5.25 The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real market data 53
5.26 The PnLs of the Delta 0 strategy for different risk aversion and no transaction costs for generated and real data. 54
5.27 The predicted hedging surface for a Asian Option by $\boldsymbol{f}$ for $\lambda = 10$ and no transaction costs. 56
5.28 The hedging surface next to its 2D representation of the hedging ratios with respect the asset price for the real market data for a Asian Option. 57
5.29 The hedging ratios over time for the strategy Delta Q for path 3 in the real market dataset for Asian Call using FNN. 58
5.30 The hedging ratios over time for the strategy Delta Q for path 3 in the real market dataset for Asian Call using LSTM. 58
5.31 The PnLs for the Delta Z and Q strategies for the Asian option using real data for FNN. 59
5.32 The PnLs for the Delta Z and Q strategies for the Asian option using real data for LSTM. 60
5.33 The predicted hedging surface for a Lookback Option by $\boldsymbol{f}$ for $\lambda = 10$ and no transaction costs. 61
5.34 The hedging surface next to its 2D representation of the hedging ratios with respect the asset price for the real market data for a LookBack Option. . . 62
5.35 Hedging ratios over time for Lookback option using FNN. 63
5.36 Hedging ratios over time for Lookback option using FNN. 63
5.37 The PnLs for the Delta Z and Q strategies for the Lookback option using real data for FNN 65
5.38 The PnLs for the Delta Z and Q strategies for the Lookback option using real data for LSTM 65
5.39 Qualitative plots of the paths generated with the paths of real market data for both indices. 66
5.40 The figure compares the marginal distribution of the paths for real and generate data for different times for SP500 data. 66

5.41 The figure compares the marginal distribution of the paths for real and generate data for different times for NASDAQ data. 67
5.42 Plot with autocorrelation scores with confidence intervals at different lags for SP500 and NASDAQ. 67
5.43 The cross-correlation matrices of the generated and real data for SP500. . . 68
5.44 The cross-correlation matrices of the generated and real data for NASDAQ. 68
5.45 The hedging surfaces pair for both assets for $\lambda = 10$ and no transaction costs for the Rainbow Option. At the **left** we represent the pairs that follows the strategy Delta Z , at the **right** the pairs that follow the strategy Delta Q. . 69
5.46 The hedging ratios for both assets over time following the Delta Z strategy for different risk aversion and without transaction costs. 70
5.47 The hedging ratios for both assets over time following the Delta Q strategy for different risk aversion and without transaction costs. 70
5.48 The hedging ratios for both assets over time following the Delta 0 strategy for different risk aversion and without transaction costs. 70
5.49 The PnLs for the Delta Z and Delta Q strategies for the generated data. . . 72
5.50 The PnLs for the Delta Z and Delta Q strategies for the real data. 72

A.1 he hedging ratios over asset price at time step $t = 20$ following the Delta Z strategy for different risk aversion and cost levels for LSTM. 74
A.2 he hedging ratios over asset price at time step $t = 20$ following the Delta Q strategy for different risk aversion and cost levels for LSTM. 74
A.3 he hedging ratios over asset price at time step $t = 20$ following the Delta 0 strategy for different risk aversion and cost levels for LSTM. 75
A.4 The hedging ratios over time for the Delta Z strategy for different risk aversions and cost levels for LSTM. 75
A.5 The hedging ratios over time for the Delta Q strategy for different risk aversions and cost levels for LSTM. 75
A.6 The hedging ratios over time for the Delta 0 strategy for different risk aversions and cost levels for LSTM. 76
A.7 The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for generated data for LSTM. 77
A.8 The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for generated data for LSTM. 77
A.9 The PnLs of the strategy Z and Q for different risk aversions and no transaction costs for generated data for LSTM. 77
A.10 The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real data for LSTM. 78
A.11 The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for real data for LSTM. 79
A.12 The PnLs of the strategy Z and Q for different risk aversions and no transaction costs for real data for LSTM. 79
A.13 The PnLs for the Delta Z and Q strategies for the Asian option using generated data for FNN. 81
A.14 The PnLs for the Delta Z and Q strategies for the Asian option using generated data for LSTM. 81
A.16 The PnLs for the Delta Z and Q strategies for the Asian option using generated data for LSTM. 83
A.15 The PnLs for the Delta Z and Q strategies for the Asian option using generated data for FNN. 83

List of Tables

2.1 Common one-dimensional activation functions and their properties. Adapted from Wikipedia[41] . . . 20
2.2 Multi-dimensional activation functions and their properties. Adapted from Wikipedia[41] . . . 20

5.1 Table with Indifference Prices, Super-Hedging Ratios, VaR, and CVaR for different risk aversion levels and transaction costs using FNN and LSTM for the BS model. . . . 38
5.2 The table shows the different KS average scores and average Type I error for different times along the paths marginal distributions. . . . 43
5.3 The table shows the autocorrelation scores with 95% confidence intervals at different lags. . . . 43
5.4 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies for generated data. . . . 45
5.5 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real market data. . . . 52
5.6 Comparison of $\rho^{\theta}(0)$, Super-Hedging Ratios, and VaR between Generated Data and Real Data. . . . 54
5.7 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Z, for Real data for the Asian Option. . . . 59
5.8 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Q, for Real data for the Asian Option. . . . 60
5.9 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Z for a Lookback option . . . 64
5.10 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Q, for a Lookback option. 64
5.11 Comparison of SP500 and NASDAQ across different time steps with average KS score and Type I errors. . . . 67
5.12 Autocorrelation scores with confidence intervals at different lags for SP500 and NASDAQ. . . . 68
5.13 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for generated data for the Rainbow Option. . . . 71
5.14 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real data for the Rainbow Option. 71

A.1 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for generated data for LSTM. . . . 76

A.2 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real data for LSTM. 78
A.3 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM, following the strategy Delta Z, for the Asian Option for the Generated Data. 80
A.4 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Q, for the Asian Option for Generated Data. 80
A.5 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Z, for generated data. . 82
A.6 Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Q, for generated data. . 82

Introduction

The challenge of pricing and hedging derivatives in financial markets is one of fundamental in topics Quantitative Finance. Derivatives are crucial financial instruments for hedgers to mitigate the risk of volatile underlyings assets. Especially for entities like banks or market makers, which may act in the sell-side offering liquidity to the market, hedging and pricing derivative products are their main responsibilities. The first quantitative model that provide a standardized and mathematically rigorous way to price and hedge derivatives was the Black-Scholes model, first introduced in [3] in 1973. During the last 30 years this framework was the cornerstone of the financial mathematics for both practitioners and academia. The model is based under the assumption of frictionless markets, with perfect information hypothesis, and continues trading strategies. In these idealized conditions, market participants can trade without incurring in market fractions such as transaction costs, liquidity constraints, or market impact, and the market is "*complete*", meaning every contingent claim can be perfectly hedged. However, real-world markets deviate significantly from these perfect market assumptions. Trading is often subject to frictions, and to the impossibility to trade at continuous time. Hence, markets are typically incomplete, where not all risks can be perfectly hedged. These imperfections necessitate the development of more sophisticated models and methods for pricing and hedging in incomplete markets.

In recent years, the rise of machine learning, particularly neural networks, has opened new avenues for addressing these challenges. Neural networks, inspired by the human brain's architecture, have proven their ability to learn from large datasets and adapt to new information, making them invaluable tools in various domains, from image and speech recognition to natural language processing [15, 17]. One of the most significant contributions of neural networks is in the field of finance, where they are used for tasks such as algorithmic trading [28], financial forecasting [27, 30], detect financial bubbles [1], and optimize trading strategies. More concretely, in 2019 Hans Buehler et al. develop the *Deep Hedging* model in [4], where they achieve to develop successful hedging strategies using neural networks in an incomplete market framework. They used the numerical framework based in indifference pricing and convex risk measures for incomplete markets already is studied in [33, 26, 42] and incorporated into a deep learning framework.

In these past recent years the Deep Hedging model became so successful between practitioners and academia that became a recurrent framework. Researchers extend and implemented the model into different problems, and frameworks. For example, in [5] they extend the model into a risk-neutral measure; in [6, 34, 7] they extend the model into a reinforcement learning setup; in [2] they implement it using another numerical framework based in quantile super-hedging; in [24] they implement it for rough volatility models; and they even implement it into a quantum computer in [35].

However, to effectively apply a deep hedging framework to real market data, we require a reliable market generator due to the inherent irregularities and noise in the data. Moreover, deep hedging approaches necessitate a large number of simulated market paths to evaluate and optimize hedging strategies. One promising solution to this problem is the use of Neural Stochastic Differential Equations (Neural SDEs), which were first in-

troduced in the context of continuous-time generative modeling of irregular time series in [40]. Neural SDEs have demonstrated state-of-the-art performance, particularly when trained adversarially as Wasserstein Generative Adversarial Networks (Wasserstein-GANs) [30]. Despite their success, training Neural SDEs in an adversarial setting is notoriously unstable, often suffering from issues such as mode collapse, and requiring sophisticated techniques like weight clipping and gradient penalty to maintain stability. To address these challenges, the authors in [27] propose a novel approach to training Neural SDEs non-adversarially using signature kernels, which are rooted in rough path theory. This method has shown significant success, offering a more stable and efficient training process.

The objective of this thesis is to develop a comprehensive machine learning-driven framework that can take market data as input and efficiently price and hedge any derivatives under real-world conditions, like transaction costs. This framework will integrate the Neural SDE develop in [27] as the market data generator for our deep hedging models. Our framework is model-free and fully data-driven. We aim to learn the optimal trading strategies of the market, and optimise the hedging strategies of any complex derivative product. Our objective is to develop two hedging strategies depending in our risk preference. One based in taking into advantage the statistical arbitrage of the market while hedging the derivative, and other based on hedging the liability in a risk-neutral measure. In both strategies we will incorporate a risk aversion parameter.

We have structure the thesis in 5 chapters. Chapter 1 compares the mathematical backgrounds of traditional models like Black-Scholes with the Convex Risk Measures and Indifference pricing framework. Chapter 2 presents the theoretical background in which neural networks are based, and shows two types of neural networks: Feed Forward Neural Networks and Recurrent Neural Networks. These two types will be use in the implementation of our framework. Chapter 3 shows the Neural SDE architecture and explains the concepts behind training the model with signature kernel scores. Chapter 4 explains the implementation of the Deep Hedging model and Neural SDE model for the SP500 data. Finally, Chapter 5 provides the results and discussion of our framework for a European Call Option, and other more complex products like Asian Options, Lookback Options and Rainbow Options.

Chapter 1

Pricing and Hedging in Imperfect Markets

This chapter addresses the problem of pricing and hedging in incomplete markets through the mathematical approach of using convex risk measures and indifference pricing. This approach offer a robust framework for handling market imperfections and providing more realistic hedging strategies for real data.

We will develop the mathematical foundation for this approach, comparing it with the classical Black-Scholes model, where its hedging strategy are based in computing "greeks" i.e. the derivatives of the liability price with respect a parameter of the model; whereas this other approach is "greekless".

1.1 Black-Scholes

The Black-Scholes (BS) model considers a probability space $(\Omega, \{\mathcal{F}_t\}_{t=0}^T, \mathbb{P})$, with $\mathbb{P}$ as the real world measure and, as we mentioned before, the model doesn't assume transaction costs and the market is complete. The model assumes that the price of the risky asset follows a geometric Brownian motion with constant drift μ and volatility σ. This can be mathematically represented by the stochastic differential equation (SDE):

$$dS_t = S_t(\mu\, dt + \sigma\, dW_t), \qquad S_0 > 0, \tag{1.1.1}$$

where $(W_t)_{t\geq 0}$ is the Brownian motion on $(\Omega, \{\mathcal{F}_t\}_{t=0}^T, \mathbb{P})$, and S_0 is the initial price.

The solution to this SDE gives the asset price a distribution over time like a log-normal:

$$S_t = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right). \tag{1.1.2}$$

The model also assumes a riskless bond with a fixed risk-free rate r and its deterministic price B_t evolves as:

$$dB_t = B_t r dt, \qquad B_0 = 1, \tag{1.1.3}$$

which is equivalent to $B_t = e^{rt}$. Our portfolio or value process V_t will be defined by,

$$V_t(\phi) = \phi_t^B B_t + \phi_t^S S_t, \tag{1.1.4}$$

where the trading strategy $\phi = (\phi^B, \phi^S)$ is a pair of stochastic processes on $(\Omega, \{\mathcal{F}_t\}_{t=0}^T, \mathbb{P})$.

1.1.1 Pricing a European Call Option

A European call option gives the holder the right, but not the obligation, to buy the underlying asset at a specified strike price K on the option's expiration date T. The contingent claim or payoff of a European call option at maturity time is:

$$V(T) = \max(S_T - K, 0). \tag{1.1.5}$$

Since, we are working in a complete market, where every contingent claim is attainable, we can price the derivative by replication. Our portfolio $V_T(\phi)$ at the final time T will match the derivative payoff $V(T)$. Also, we will work only with self-financing strategies, ie $V_t(\phi) = V_0(\phi) + G_t(\phi)$, where $V_0(\phi)$ is the initial wealth, and $G_t(\phi)$ is the gain process defined as,

$$G_t(\phi) = \int_0^t \phi_u^B dB_u + \int_0^t \phi_u^S dS_u, \tag{1.1.6}$$

or, in differential terms,$dV_t(\phi) = dG_t(\phi)$ ie, $dV_t(\phi) = \phi_t^B dB_t + \phi_t^S dS_t$. These strategies doesn't allow injection in capital during the trading time, only at the beginning $V_0(\phi)$. From this self-financing condition we can get its trading strategy,

$$\phi_t^S = \frac{\partial V}{\partial S}(t, S_t),\ \phi_t^B = (V_t - \phi_t^S S_t)/B_t, \tag{1.1.7}$$

so, the self-financing portfolio can be writing in differential terms as,

$$dV_t = \phi_t^B dB_t + \phi_t^S dS_t = \left(V(t, S_t) - \frac{\partial V}{\partial S}(t, S_t)S_t\right) rdt + \frac{\partial V}{\partial S}(t, S_t)S_t(\mu dt + \sigma dW_t). \tag{1.1.8}$$

Following the theory of pricing by replication, we assume that the price of the derivative at time t is $V(t, S_t)$ a function of time and the stock price. We apply Ito's Lemma to $V(t, S_t)$,

$$dV(t, S_t) = \left(\frac{\partial V}{\partial t}(t, S_t) + \frac{\partial V}{\partial S}(t, S_t)\mu S_t + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(t, S_t)\sigma^2 S_t^2\right) dt + \frac{\partial V}{\partial S}(t, S_t)\sigma S_t dW_t. \tag{1.1.9}$$

Then, by equating the self-financing portfolio (1.1.8) and Ito's Lemma (1.1.9), we get the famous Black-Scholes partial difference equation (PDE),

$$\frac{\partial V}{\partial t}(t, S_t) + \frac{\partial V}{\partial S}(t, S_t)rS_t + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(t, S - t)\sigma^2 S_t^2 = rV(t, S_t) \tag{1.1.10}$$

with (1.1.5) as boundary condition. We notice that the constant drift μ from the real world measure $\mathbb{P}$ it is gone, which means that the replicating price won't depend on it. Now, we could price the derivative by solving the PDE with any numerical method like finite difference methods. But also, we can price it with an analytical solution by using the the Feynman-Kac theorem, which allows to interpret the solution of a parabolic PDE (1.1.10) in terms of expected values of a diffusion process,

$$V(t, S_t) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[V(S_T)|\mathcal{F}_t]. \tag{1.1.11}$$

The expectation is taken with respect to the so-called martingale measure $\mathbb{Q}$, i.e. a probability measure $\mathbb{Q} \sim \mathbb{P}$ under which the risky stock price $\frac{S_t}{B_t} = e^{-r_t}S_t$ with the risk-free bond price B_t as a numeraire is a martingale, which is equivalent to $S(t)$ having drift rate r under $\mathbb{Q}$:

$$S_t = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t^Q\right) \tag{1.1.12}$$

This is equivalent to apply the Girsanov's theorem to the real world measure $\mathbb{P}$ to get the risk-neutral measure $\mathbb{Q}$, which gives the classic BS SDE with the risk-free rate r as drift,

$$dS_t = S_t(r\,dt + \sigma\,dW_t), \qquad S_0 > 0 \tag{1.1.13}$$

By the Fundamental Theorem of Asset Pricing (FTAP), we can affirm that BS market model is free of arbitrage since, since it has an equivalent martingale measure $\mathbb{Q}$. The concept of free arbitrage is essential in asset pricing, since ensures no free-money. An arbitrage strategy is a self-financing strategy that will make money with positive probability without assuming any risks, ie $\mathbb{P}(V_t(\phi) > 0) > 0$. Moreover, the corollary of the FTAP states that a market model is arbitrage free and complete, if and only if, there exist a unique martingale measure $\mathbb{Q}$. Closing the circle where we started from, making this theory really robust mathematically.

Now pricing consists in just computing the expected payoff of the option under the measure $\mathbb{Q}$ (1.1.11), which gives the following analytical price:

$$V(t, S_t) = S_t N(d_1) - Ke^{-r(T-t)}N(d_2) \tag{1.1.14}$$

where:

- $d_1 = \frac{\ln(S_t/K)+(r+\frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$
- $d_2 = d_1 - \sigma\sqrt{T-t}$
- $N(\cdot)$ is the cumulative distribution function of the standard normal distribution.

1.1.2 Delta Hedging in BS

In the BS world, as we seen before, pricing amounts to replicate the expected payoff of the derivative with an optimal self-financing strategy, ie $V_T = V_0 + G_T(\phi_{BS})$, where V_0 is the price of the derivative at time $t = 0$, ϕ_{BS} is the optimal Black-Scholes delta hedging strategy defined at (1.1.7). For an European Call option it would be,

$$\phi_t^S = \frac{\partial V}{\partial S}(t, S_t) = N(d_1) \tag{1.1.15}$$

This is the greek *Delta* of the BS model.

If we could trade any infinitesimal ratio of stocks at continuous time, we could perfectly hedge any derivative under the BS world. However, if we are under discrete time, where we can trade only at certain time steps $0 = t_0 < t_1 < ... < t_{i-1} < t_i < ... < t_n = T$, this perfect hedging cannot be achieved even in a BS market. The BS model in discrete time would be,

$$S_{t_i} = S_{t_{i-1}} \exp\left((r - \frac{1}{2}\sigma^2)(t_i - t_{i-1} + \sigma(t_i - t_{i-1})^{1/2} Z_{i+1}\right), S_0 > 0 \tag{1.1.16}$$

where $\{Z_{t_i}\}_{i=0}^n$ are random variables identical independent distributed (iid) as random normal distributions $\mathcal{N}(0, 1)$.

We provide a numerical example to illustrate this. Where we simulate 307200 paths of 30 time steps, with parameters $r = 0$, $\sigma = 0.1$, $K = 100$, and $S_0 = 100$. In the figure 1.1 we plot the histogram of the PnL of the BS delta hedging strategy:

$$PnL = V(0, S_0) + \sum_{i=0}^{T-1} \phi_i^{BS}(t_i, S_{t_i})(S_{t_{i+1}} - S_{t_i}) - \max(S_T - K, 0)$$

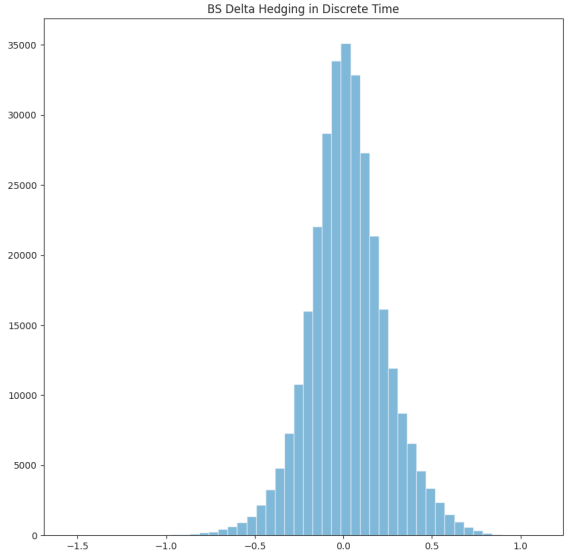


Figure 1.1: The histogram of the PnL of following the Black-Scholes delta hedging strategy.

We notice that the strategy doesn't hedge perfectly the derivative, otherwise the only PnL would be 0, but super-hedge them with a ratio of 51.7%, ie the probability that PnL ≥ 0.

In summery, the Black-Scholes model provide us a robust benchmark for pricing and hedging derivatives. However, its notable assumptions limited its practical use in real markets with transaction costs, arbitrage, and more complex stock price movements.

1.2 Settings for a Discrete Incomplete Market

In this section, we define a general mathematical setting for pricing and hedging in incomplete markets. We follow a similar notation to [4].

We consider a discrete-time financial market with a finite time horizon T and trading dates $0 = t_0 < t_1 < \cdots < t_n = T$. The market is modelled on a finite probability space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ with a probability measure $\mathbb{P}$ such that $\mathbb{P}[\{\omega_i\}] > 0$ for all i. The set of all real-valued random variables over Ω is denoted by $\mathcal{X} = \{X : \Omega \to \mathbb{R}\}$.

At each trading date t_k, new market information can be represented by $I_k \in \mathbb{R}^r$, which it could include market costs, mid-prices of liquid instruments, news, balance sheet information, trading signals, and risk limits. The process $I = (I_k)_{k=0}^n$ generates the filtration $\mathcal{F} = (\mathcal{F}_k)_{k=0}^n$, where $\mathcal{F}_k$ represents all information available up to time t_k.

The market contains d hedging instruments whose mid-prices are given by an $\mathbb{R}^d$-valued $\mathcal{F}$-adapted stochastic process $S = (S_k)_{k=0}^n$. These instruments can include both primary assets, such as equities, and secondary assets, such as liquid options. Also, it can be extended for hedging instruments that may not be tradable before a future point in time due to liquidity restrictions.

Our portfolio of derivatives, which we want to hedge, are representing our liabilities. They are represented as an $\mathcal{F}_T$-measurable random variable Z. This portfolio can include a mix of liquid and over-the-counter (OTC) derivatives, with maturity T being the maximum maturity of all instruments, at which point all payments are known. We exclude instruments with true optionally, such as American options.

For simplicity, we assume that all intermediate payments are accrued using a locally risk-free overnight rate, which allows us to assume zero rates. Also, we assume all currency spot exchanges happen at zero cost, settling in our reference currency.

1.2.1 Trading Strategies

To hedge the liability Z at T, we trade S using an $\mathbb{R}^d$-valued $\mathcal{F}$-adapted stochastic process $\delta = (\delta_k)_{k=0}^{n-1}$ with $\delta_k = (\delta_k^1, \delta_k^2, \ldots, \delta_k^d)$, which will be our hedging ratios. All trading is assumed to be self-financed, with an initial injection of cash p_0 into our portfolio, which will be the price of the derivative. We denote by $\mathcal{H}$ the set of all admissible hedging strategies. The agent's wealth at time T is given by

$$PnL_T(Z, p_0, \delta) = -Z + p_0 + (\delta \cdot S)_T - C_T(\delta), \tag{1.2.1}$$

where

$$(\delta \cdot S)_T = \sum_{k=0}^{n-1} \delta_k \cdot (S_{k+1} - S_k).$$

it is the gain process of our hedging strategies, and the trading costs are $C_T(\delta) = \sum_{k=0}^{n} c_k(\delta_k - \delta_{k-1})$, where $\delta_{-1} = 0$.

Our setup can include various market frictions:

- **Proportional transaction costs:** For $c_k^i > 0$, $c_k(n) = \sum_{i=1}^d c_k^i S_k^i |n_i|$.
- **Fixed transaction costs:** For $c_k^i > 0$ and $\epsilon > 0$, $c_k(n) = \sum_{i=1}^d c_k^i 1_{|n_i| \geq \epsilon}$.

These frictions account for proportional costs, fixed costs, and can be extended for more complex structures, such as price impact modelling.

1.3 Convex Risk Measures and Indifference Pricing

In incomplete markets, the classical approach we showed in the previous section of pricing derivatives using the theory of pricing by replication is no longer applicable. In real markets, not every contingent claim can be perfectly hedged, and markets have frictions such as transaction costs, liquidity constraints, and other limitations play a significant role. To address this, we use the concept of convex risk measures and indifference pricing, which provides an adaptable framework that doesn't depend on modelling market dynamics.

Here we presented this framework similar to how they did it in [4], since it is easy to extend to the deep hedging implementation. Although, a similar numerical framework was already studied in earlier papers like [33, 26, 42].

1.3.1 Convex Risk Measure

Convex risk measures allow us to quantify the risk of holding a particular portfolio by assigning a value that represents the minimal amount of capital that needs to be added to make the portfolio acceptable. Unlike traditional approaches, convex risk measures are quite versatile. They can take into account market frictions and statistical arbitrage, making them ideal for pricing incomplete markets.

Definition 1.3.1. A *convex risk measure* $\rho : \mathcal{X} \to \mathbb{R}$ is a function that satisfies the following properties for any asset positions $X, X_1, X_2 \in \mathcal{X}$:

- **Monotonicity**: If $X_1 \geq X_2$, then $\rho(X_1) \leq \rho(X_2)$. This implies that a more favorable position requires less capital injection.
- **Convexity**: $\rho(\alpha X_1 + (1-\alpha)X_2) \leq \alpha\rho(X_1) + (1-\alpha)\rho(X_2)$ for $\alpha \in [0,1]$. This reflects the principle that diversification reduces risk.
- **Cash-Invariance**: $\rho(X + c) = \rho(X) - c$ for any $c \in \mathbb{R}$. Adding cash to a position reduces the required capital by the same amount.

We call ρ normalized if $\rho(0) = 0$.

1.3.2 Indifference Pricing

Given a convex risk measure ρ, we define the minimal pricing problem for a contingent claim X as:

$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta)) \tag{1.3.1}$$

The following proposition found in [4, Proposition 3.2] justifies the use of convex measures since the minimal pricing problem will be a convex risk measure too, and later on this would be important for the neural network implementation.

Proposition 1.3.2. *The functional π is monotone decreasing and cash-invariant. If $C_T(\cdot)$ and $\mathcal{H}$ are convex, then π is a convex risk measure.*

In this context, $\rho(-Z)$ represents the minimal amount of capital required to make the liability $-Z$ acceptable under the risk measure ρ. Similarly, $\pi(-Z)$ is the minimal price the agent needs to charge to ensure that hedging the liability $-Z$ becomes acceptable. These concepts provide the foundation for indifference pricing, where the price of a claim is determined by the amount of capital required to remain indifferent between holding the claim or not.

Definition 1.3.3 (Indifference pricing)**.** The price $p(Z)$ of a contingent claim Z is the amount of cash p_0 that makes the agent indifferent between selling the claim or not, i.e., solving:

$$\pi(-Z + p_0) = \pi(0)$$

By cash-invariance, this leads to the *indifference price* being defined as:

$$p(Z) := \pi(-Z) - \pi(0) \tag{1.3.2}$$

Arbitrage opportunities in this setting arise when there exists a trading strategy $\delta \in \mathcal{H}$ such that the portfolio has a non-negative payoff with a positive probability of strictly positive returns, i.e. $0 \leq X + (\delta \dot{S})_T - C_T(\delta) =: (*)$ while $\mathbb{P}[(*) > 0] > 0$. In case such opportunity exists, we obviously have $\rho(X) < 0$. Depending on the cost function and our constraints on $\mathcal{H}$, we may be able to invest an unlimited amount into this strategy. In this case, we get $\pi(0) = -\infty$. If this applies to $X = 0$, we call such a market irrelevant, since it would mean that for any X, $\pi(X) = -\infty$. This concept doesn't only apply to classical arbitrage but also to statistical arbitrage. If in average there is any trading strategy in the market that assures to make profit, and you can exploit it indefinitely you will have an irrelevant market.

For example for a market where $0 > \pi(0) > -\infty$, it means that following the strategy that minimizes $\rho(0)$, you will make money on average i.e. statistical average, but you can not scale it indefinitely. Notice that this is the power of indifference pricing, where you can account for this, so you don't need a martingale measure $\mathbb{Q}$ modelling the stock price to price a derivative.

1.3.3 Exponential Utility Indifference Pricing

One specific case of indifference pricing is based on the exponential utility, this framework was already studied [23, 14], and consist as follows. For an exponential utility function $U(x) = -\exp(-\lambda x), x \in \mathbb{R}$, with risk aversion parameter $\lambda > 0$, the indifference price

$p(Z)$ of the contingent claim Z satisfies:

$$\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[U\left((\delta\cdot S)_T - C_T(\delta)\right)\right] = \sup_{\delta\in\mathcal{H}} \mathbb{E}\left[U\left(p(Z) - Z + (\delta\cdot S)_T - C_T(\delta)\right)\right]$$

$$\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left((\delta\cdot S)_T - C_T(\delta)\right)\right)\right] = \sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left(p(Z) - Z + (\delta\cdot S)_T - C_T(\delta)\right)\right]\right)$$

$$\exp(\lambda p(Z)) = \frac{\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left(-Z + (\delta\cdot S)_T - C_T(\delta)\right)\right)\right]}{\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left((\delta\cdot S)_T - C_T(\delta)\right)\right)\right]}$$

$$p(Z) = \frac{1}{\lambda}\log\left(\frac{\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left(-Z + (\delta\cdot S)_T - C_T(\delta)\right)\right)\right]}{\sup_{\delta\in\mathcal{H}} \mathbb{E}\left[\exp\left(-\lambda\left((\delta\cdot S)_T - C_T(\delta)\right)\right)\right]}\right) \tag{1.3.3}$$

This results leads to the definition to the entropic risk measure as follows:

$$\rho(X) = \frac{1}{\lambda}\log\mathbb{E}\left[\exp(-\lambda X)\right] \tag{1.3.4}$$

Since, this measure gives the same indifference pricing :

$$p(Z) = \pi(-Z) - \pi(0)$$
$$= \frac{1}{\lambda}\log\left(\sup_{\delta\in H}\mathbb{E}\left[U\left(-Z + (\delta\cdot S)_T + C_T(\delta)\right)\right]\right) - \frac{1}{\lambda}\log\left(\sup_{\delta\in H}\mathbb{E}\left[U\left((\delta\cdot S)_T + C_T(\delta)\right)\right]\right)$$

Chapter 2

Neural Networks (NN)

Neural networks consist of interconnected layers of artificial neurons, where each neuron processes input data through a series of weights and activation functions to produce an output. These models are designed to recognize patterns in data, enabling them to perform complex tasks that were once thought to be the exclusive domain of human intelligence.

This chapter will delve into the mathematical foundation of neural networks, starting with the basics of Feed Forward Neural Networks (FNN), which are the most straightforward type of neural network. We will explore their general architecture, the role of activation functions, and the concept of universal approximation, which underpins their ability to model complex functions. Additionally, we will discuss the importance of loss functions to adapt them into a variety of different problems, and the various optimization techniques used to improve the performance of training neural networks.

In the latter part of the chapter, we will focus on Recurrent Neural Networks (RNNs), which are particularly suited for sequence prediction tasks. A special type of RNN, the Long-Short Term Memory (LSTM) network, will be introduced due to its effectiveness in capturing long-term dependencies in time series data.

2.1 Feed Forward Neural Networks (FNN)

This section is inspired from the Deep Learning notes [18].

2.1.1 General Architecture

Definition 2.1.1. Let $I, O, r \in \mathbb{N}$. A function $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$ is a *feedforward neural network*(FNN) with $r-1 \in \{0, 1, \ldots\}$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i-th hidden layer for any $i = 1, \ldots, r-1$, and activation functions $\sigma_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$, $i = 1, \ldots, r$, where $d_r := O$, if

$$\boldsymbol{f} = \sigma_r \circ \mathbf{L}_r \circ \cdots \circ \sigma_1 \circ \mathbf{L}_1, \tag{2.1.1}$$

where $\mathbf{L}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, for any $i = 1, \ldots, r$, is an affine function given by

$$\mathbf{L}_i(\mathbf{x}) := \mathbf{W}^i \mathbf{x} + \mathbf{b}^i, \quad x \in \mathbb{R}^{d_{i-1}},$$

parameterized by the weight matrix $W^i = [W^i_{j,k}]_{j=1,\ldots,d_i, k=1,\ldots,d_{i-1}} \in \mathbb{R}^{d_i \times d_{i-1}}$ and the bias vector $\mathbf{b}^i = (b^i_1, \ldots, b^i_{d_i}) \in \mathbb{R}^{d_i}$, with $d_0 := I$. We shall denote the class of such functions $\mathbf{f}$ by

$$\mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots, \sigma_r). \tag{2.1.2}$$

If $\sigma_i(x) = (g(x_1), \ldots, g(x_{d_i}))$, $\mathbf{x} = (x_1, \ldots, x_{d_i}) \in \mathbb{R}^{d_i}$, for some $g : \mathbb{R} \to \mathbb{R}$, we write g in place of σ_i in (2.2) and (2.3).

The architecture of the FNN is completely characterized by:

- the hyparameters: $r, d_1, \ldots, d_r$,
- the actual parameter weights $W^1, \ldots, W^r$ and biases $b^1, \ldots, b^r$,
- The activation functions: $\sigma_1, \ldots, \sigma_r$.

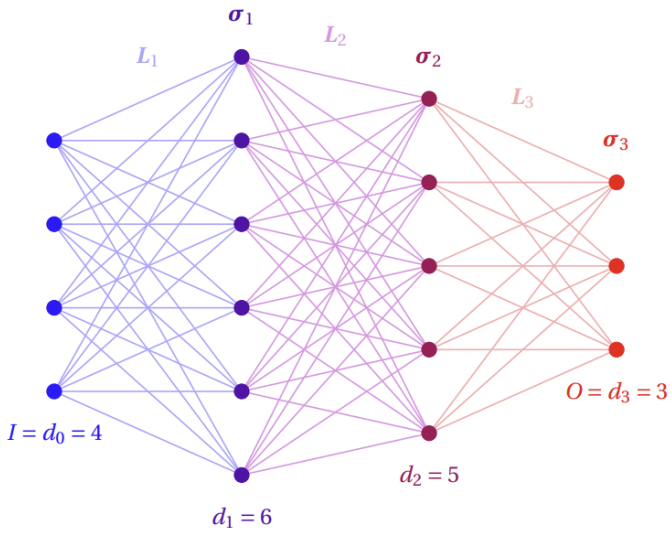


Figure 2.1: Graphical representation of a neural network with $r = 3, I = d_0 = 4, d_1 = 6, d_2 = 4$ and $O = d_3 = 3$. Source from Deep Learning lecture notes [18]

2.1.2 Activation functions

Activation functions are key in the architecture of a neural network. They allow the FNN to be non-linear, without them the concatenation of multiple layers would be another linear function, making them useless. This non-linearity allows the neural network to fit data from a variety of different problems. Normally, activation functions can be characterized in two categories: one-dimension and multidimensional activation functions. We build the table 2.1 with some examples of the most commonly used one dimension activation functions, and the table

The *Identity* is normally used only in the output layer, if we expect an unbounded optput. The *ReLU* is one of the most popular activation functions for the hidden layers nowadays, and its derivative are mathematically very simple, making them numerically efficient (although it is not defined at 0, we can usually substitute it with value 1 without any significant issues). There is its continues differentiable counterpart **Softplus**, but it works as well as ReLU and it is a more complicated derivative. There are also popular some of the ReLU variants like *PReLU* or *SiLU*, that allow to take into consideration negative values, which can be useful in some contexts. There are saturating activation functions like *Sigmoid*, *Hyperbolic tangent*, or *Gaussian*. These are suited when you are looking for a bound solution, like in a classification problem.

Regarding the most common multi-dimensional activation functions, we have the *Softmax*, which is used to build a multiclass classifier, and the *Maxout*, which is similar to a multi-dimensional ReLU.

Table 2.1: Common one-dimensional activation functions and their properties. Adapted from Wikipedia[41]

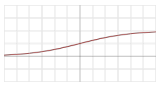
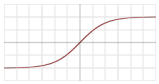
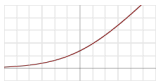
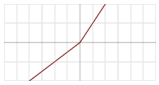
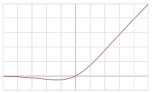
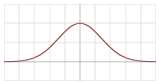
Name	Plot	Function $g(x)$	Derivative of $g'(x)$	Range	Smoothness
Identity		x	1	$(-\infty, \infty)$	C^∞
Sigmoid		$\frac{1}{1+e^{-x}}$	$g(x)(1-g(x))$	$(0,1)$	C^∞
Hyperbolic Tangent (tanh)		$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1-g(x)^2$	$(-1,1)$	C^∞
Rectified LinearUnit (ReLU)		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$	$[0, \infty)$	C^0
Softplus		$\ln(1+e^x)$	$\frac{1}{1+e^{-x}}$	$(0, \infty)$	C^∞
Parametric Rectified Linear Unit (PReLU)		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} \alpha & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0
Sigmoid Linear Unit (SiLU)		$\frac{x}{1+e^{-x}}$	$\frac{1+e^{-x}+xe^{-x}}{(1+e^{-x})^2}$	$[-0.278\ldots, \infty)$	C^∞
Gaussian		e^{-x^2}	$-2xe^{-x^2}$	$(0, 1]$	C^∞

Table 2.2: Multi-dimensional activation functions and their properties. Adapted from Wikipedia[41]

Activation	Definition	Derivative	Range	Smoothness
Softmax	$g_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$, $i = 1, \ldots, d$	$\frac{\partial g_i(\mathbf{x})}{\partial x_j} = \begin{cases} g_i(\mathbf{x})(1-g_i(\mathbf{x})), & i = j \\ -g_i(\mathbf{x})g_j(\mathbf{x}), & i \neq j \end{cases}$	$(0,1)^d$	C^∞
Maxout	$g(\mathbf{x}) = \max\{x_1, \ldots, x_d\}$	$\frac{\partial g(\mathbf{x})}{\partial x_i} = \begin{cases} 1, & x_i > \max_{j \neq i} x_j \\ 0, & x_i < \max_{j \neq i} x_j \end{cases}$	$\mathbb{R}$	C

2.1.3 Universal Approximation

The Universal Approximation Theorem is a fundamental concept in the theory of neural networks. It provides the mathematical justification for why machine learning, particularly deep learning, is so effective. The theorem states that a feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$, given appropriate activation functions. This result is profound as it assures that neural networks have the capacity to model complex functions, which is why they are so powerful in multiple applications, including finance.

The following theorem is a slight reformulation of [32, Theorem 1 and Proposition 1].

Theorem 2.1.2 (Universal approximation property)**.** *Let $g : \mathbb{R} \to \mathbb{R}$ be a measurable function such that:*

1. *g is not a polynomial function,*
2. *g is bounded on any finite interval,*
3. *the closure of the set of all discontinuity points of g in $\mathbb{R}$ has zero Lebesgue measure.*

Moreover, let $K \subset \mathbb{R}^I$ be compact and $\epsilon > 0$. Then:

(i) For any $u \in C(K, \mathbb{R})$, there exist $d \in \mathbb{N}$ and $f \in \mathcal{N}_2(I, d, 1; g, Id)$ such that

$$\|u - f\|_{sup,K} < \epsilon.$$

(ii) Let $p \geq 1$. For any $v \in L^p(K, \mathbb{R})$, there exist $d' \in \mathbb{N}$ and $h \in \mathcal{N}_2(I, d', 1; g, Id)$ such that

$$\|v - h\|_{L^p(K)} < \epsilon.$$

Remark 2.1.3. It is important to note that theorem 2.1.2 holds only for networks with a single hidden layer. However, in practice this theorem can be extend it to deeper networks [43, 20].

Remark 2.1.4. Another important observation is that the theorem 2.1.2 does not inform on how the neural network f and h should look like, it only guarantees their existence. However, this result does not ensure the convergence of solutions, as challenges such as numerical noise, local minima, or poorly conditioned problems may impede convergence. Consequently, this theorem merely serves as a theoretical reassurance that approximating functions using neural networks is indeed possible.

2.2 Training and Optimizers

2.2.1 Loss Function

Training is one of the most challenging and important parts when you are building your neural network model. Once our architecture is selected, now we have to find the optimal weights and biases with the training process. The loss function is key in this process, since evaluates the output of the neural network to match our expected needs from the model. The choice of an appropriate loss function is crucial because it directly influences the optimization process and the ability of the network to learn from data. Basically, the training of neural networks models is done by optimizing the loss function $\ell : \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}$.

Given a neural network $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$, $\mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots, \sigma_r)$, and input $\mathbf{x} \in \mathbb{R}^I$ and reference value $\mathbf{y} \in \mathbb{R}^O$, we compute the realised loss as $\ell(\boldsymbol{f}(\mathbf{x}), \mathbf{y})$. Mathematically,

if $\mathbf{x}$ and $\mathbf{y}$ are a realisation of a joint random vector $(\mathbf{X}, \mathbf{Y})$, we could theoretically try to find an optimal $\boldsymbol{f}$ by minimising the risk

$$\mathbf{E}[\ell(\boldsymbol{f}(\mathbf{X}), \mathbf{Y})]$$

However, in practice we don't really know the distribution of $(\mathbf{X}, \mathbf{Y})$, so we need to work with the empirical minibatch risk

$$\mathcal{L}_B(\theta) := \frac{1}{\#B} \sum_{i \in B} \ell(\boldsymbol{f}_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{2.2.1}$$

This definition allows for the average over any subset of samples or minibatch, $B \subset \{1, \dots, N\}$. Also, we are noticeable the weights and biases in this definition $\theta := (W^1, \dots, W^r; \mathbf{b}^1, \dots, \mathbf{b}^r)$, since from the training perspective the loss function is a function of θ.

Depending whether your problem involves in that the neural network replicates the label or reference value $\mathbf{y}$, or not, we have two different problems: supervised or unsupervised learning. In supervised training our objective is that $\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{x})$ matches its label $\mathbf{y}$. Some of the most used loss functions in supervised learning are:

- Absolute loss: $\ell(\hat{y}, y) = |\hat{y} - y|$. It targets the median. Reducing the loss function would mean to reduce the difference between both medians of $\mathbf{y}$ and $\hat{\mathbf{y}}$
- Squared loss: $\ell(\hat{y}, y) = (\hat{y} - y)^2$. It targets the mean. Reducing the loss function would mean to reduce the difference between both means of $\mathbf{y}$ and $\hat{\mathbf{y}}$
- Binary cross-entropy: $\ell(\hat{y}, y) = -y \log \hat{y} - (1-y) \log(1-\hat{y}), \quad \hat{y} \in (0,1), \quad y \in \{0,1\}$. It is useful for classification problems with binary choices. Minimising categorical cross-entropy amounts to minimising the corresponding Kullback–Leibler divergence, which is an information theoretic measure of the discrepancy between the empirical distribution of labels and the distribution predicted by the network.

For unsupervised training there are not common choices, since each problem requires its personalised loss function. Now, the reference value $\mathbf{y}$ are not the expected values of the output of the neural network, but values that would be useful to quantise loss function that we want to optimise with respect the $\mathbf{f}_\theta(\mathbf{x})$.

2.2.2 Stochastic Gradient Descent

After formulating the loss function, the next crucial step involves devising a method to obtain a optimizer. A naive approach would be to work out the gradient of a generic differentiable objective function $F : \mathbb{R}^d \to \mathbb{R}$, and solve $\nabla \mathcal{F}(\mathbf{x}) = 0$. However, in practice this approach would be infeasible because of the big number of parameters to fit. The usual approach is using the the gradient flow,

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla F(\mathbf{x}(t)), \quad t > 0 \tag{2.2.2}$$

with initial condition $\mathbf{x}(\mathbf{0}) \in \mathbb{R}^d$. Under certain assumptions on F, it can be shown that $\mathbf{x}(t)$ tends to the minimiser as $t \to \infty$ [37]. In the context of optimizing our neural network, we will discrete the differential equation (2.2.2) for a step size or learning rate $\eta > 0$:

$$\frac{\mathbf{x}(t+\eta) - \mathbf{x}(t)}{\eta} \approx -\nabla F(\mathbf{x}(t))$$

$$\mathbf{x}(t+\eta) \approx \mathbf{x}(t) - \eta \nabla F(\mathbf{x}(t))$$

This Euler approximation motivates the gradient descent, which is an iterative algorithm that step by step looks for a minimiser with gradient updates

$$\mathbf{x}_{\text{new}} := \mathbf{x}_{\text{old}} - \eta \nabla F(\mathbf{x}_{\text{old}})$$

given some initial condition $\mathbf{x}_0$. However, for our purpose this method infeasible. Computing the gradient of $\mathcal{L}(\boldsymbol{f}_\theta)$ can be computationally costly with a large training dataset N, also it can lead to an overfitted network $\mathbf{f}_\theta$. To circumvent this limitation, we can split the data into random minibatches, and compute the gradient of the loss function of that minibatch to update the wieghts and biases of the network. This optimizer firstly introduced in [25] in 1993, it is called Stochastic Gradient Descent (SGD), and it is the core optimizer in neural networks. Other optimizers more popular nowadays are extension and improvements of this one.

In SGD, as mention before we split the data into k minibatches with fix size $m \ll N$, such that $N = km$, for $k \in \mathbb{N}$. We then sample uniformly minibatches $B_1, \dots, B_k \subset \{1, \dots, N\}$, such that $\#B_i = m$ for any $i = 1, \dots, k$, without replacement, that is, $B_1, \dots, B_k$ are disjoint with $\bigcup_{i=1}^{k} B_i = \{1, \dots, N\}$. Starting form initial condition θ_0, the parameter vector θ is updated like

$$\theta_i := \theta_{i-1} - \eta \nabla_\theta \mathcal{L}_{B_i}(\theta_{i-1}), \quad i = 1, \dots, k \tag{2.2.3}$$

where $\mathcal{L}_{B_i}(\theta)$ is minibatch empirical risk corresponding to B_i. If we consider that the training data consist of iid samples, we can interpret the minibatch gradient $\nabla_\theta \mathcal{L}_{B_i}(\theta_{i-1})$ as a noisy but unbiased estimate of the full gradient. One pass through the entire training data set by (2.2.3) constitutes a training epoch. This procedure is then repeated over multiple epochs, with new minibatches, while initialising θ_0 with the last value of the previous epoch. For clarity, we summarise the entire SGD algorithm in pseudocode in Algorithm 1.

Algorithm 1 Stochastic Gradient Descent

Input: # epochs: n_e
Input: # batches: m
Input: batch size: k
Input: Initial Parameters: θ_0
Input: learning rate: η
Output: trained network parameters $\theta_m^{n_e}$
▷ Loop through all the epochs
for i in $\{1, 2, \dots, n_e\}$ **do**
    randomly select m samples with k data sets without replacement
    if $i == 1$ **then**
        $\theta_0^i = \theta_0$ ▷ initialize the starting weight matrix
    else
        $\theta_0^i = \theta_m^{i-1}$ ▷ take the last training result as initial condition
    end if ▷ Loop through all minibatches
    for j in $\{1, 2, \dots, m\}$ **do**
        $\theta_{j+1}^i = \theta_j^i - \epsilon \nabla_\theta \mathcal{L}_{B_j}(\theta_j^i)$ ▷ gradient update for each minibatch j
    end for
end for
return $\theta_m^{n_e}$

2.2.3 Adam Optimizer

Adam (Adaptive Moment Estimation) is one of the most popular optimization algorithms. It was firstly introduced in [31], it is an enhance version of the SGD that adapts the learning parameter individually for different models. At each time step approximates the first and second moment of the gradient by computing the exponential-weighted moving average (EWMA) of previous gradients. The first moment estimate represent the direction of update and the second moment estimate is the adjustment to the learning rate at each time step. For clarity, we summarise the Adam algorithm in pseudocode at Algorithm 2. We write it with a looping process based in some converge criteria, instead of a certain number of epochs. Also, note that the direct estimation through the EWMA always generates bias towards zero at the initial time step of training, so the algorithm corrects this bias.

Adam combines the benefits of two other popular algorithm like AdaGrad [16] and RMSProp [13]. AdaGrad is known for its effectiveness in dealing with sparse data (when most of the gradient values are zero), while RMSProp is known for its ability to adapt learning rates based on recent gradient information, making it ideal for noisy and non-stationary objective functions. Adam inherits the strengths of both methods, making it robust and versatile.

Algorithm 2 Adam algorithm, where all operations on vectors are element-wise computations, and default input $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$

Input: step size: α
Input: exponential decay rates for moment estimates: $\beta_1, \beta_2 \in [0, 1)$
Input: loss function with respect to parameters θ: $\ell_t(\theta)$
Input: Initial parameter vector: θ_0
Input: initial 1^{st} moment vector: $\mathbf{m}_0 = 0$
Input: initial 2^{nd} moment vector: $\mathbf{v}_0 = 0$
Output: updated model parameter vector: θ_t
$t = 0$ ▷ loop from the beginning of the time stamp
while θ_t not converge **do**
    $t = t + 1$
    $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla_\theta \ell_t(\theta_{t-1})$ ▷ update biased first moment estimate
    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \left(\nabla_\theta \ell_t(\theta_{t-1})\right)^2$ ▷ update biased second moment estimate
    $\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}$ ▷ bias-corrected first order estimate
    $\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}$ ▷ bias-corrected second order estimate
    $\theta_t = \theta_{t-1} - \frac{\alpha \hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}$
end while
return θ_t

2.2.4 Back-Propagation

Now we will discuss the details of computing the gradient of the loss function. The backpropagation algorithm is an effective gradient calculation method based on the chain rule. The computation starts at the output layer and is propagated back layer by layer to compute the gradient of the loss function with respect to each parameter.

To show backpropagation theoretically, we first introduce the adjoint and chain rule. Assume that we are training a FNN $\boldsymbol{f}_\theta \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots, \sigma_r)$ by SGD, where σ_i is the component-wise application of a one-dimensional activation function $g_i : \mathbb{R} \to \mathbb{R}$,

for $i = 1, \ldots, r$. Denote the derivative of $\mathbf{g}_i$ as $\mathbf{g}_i'$. For $\mathbf{x} \in \mathbb{R}^I$,

$$\begin{aligned}
\mathbf{z}_i = (z_i^1, \ldots, z_i^{d_i}) &:= \mathbf{W}_i \mathbf{a}_{i-1} + \mathbf{b}_i, \quad i = 1, \ldots, r, \\
\mathbf{a}_i = (a_i^1, \ldots, a_i^{d_i}) &:= \mathbf{g}_i(\mathbf{z}_i), \quad i = 1, \ldots, r, \\
\mathbf{a}_0 &:= \mathbf{x},
\end{aligned}$$

so then $\boldsymbol{f}_\theta(\mathbf{x}) = \mathbf{a}^r$ and $\ell(\mathbf{f}_\theta(\mathbf{x}), y) = \ell(\mathbf{a}_r, y)$. The adjoint $\delta_i = (\delta_i^1, \ldots, \delta_i^{d_i}) \in \mathbb{R}^{d_i}$ is defined as

$$\delta_j^i := \frac{\partial \ell}{\partial z_j^i}, \quad j = 1, \ldots, d_i,$$

for any $i = 1, \ldots, r$.

Definition 2.2.1 (Chain rule)**.** For differentiable $G : \mathbb{R}^d \to \mathbb{R}$ and $\mathbf{F} = (F_1, \ldots, F_d) : \mathbb{R}^{d'} \to \mathbb{R}^d$, define $H(\mathbf{x}) = G(\mathbf{y})$ with $\mathbf{y} = (y_1, \ldots, y_d) = \mathbf{F}(\mathbf{x})$, that is, $H = G \circ \mathbf{F} : \mathbb{R}^{d'} \to \mathbb{R}$. Then

$$\frac{\partial H}{\partial x_i}(\mathbf{x}) = \sum_{j=1}^{d} \frac{\partial G}{\partial y_j}(\mathbf{y}) \frac{\partial F_j}{\partial x_i}(\mathbf{x}).$$

Proposition 2.2.2 (Backpropagation)**.** *Using the chain rule and the adjoint, we can derive a backward recursive procedure for the components of* $\nabla_\theta \ell(\boldsymbol{f}_\theta(\mathbf{x}), y)$*:*

$$\begin{aligned}
\delta_r &= g_r'(\mathbf{z}_r) \odot \nabla_{\hat{y}} \ell(\mathbf{a}_r, y), \\
\delta_i &= g_i'(\mathbf{z}_i) \odot (\mathbf{W}_{i+1}^\top \delta_{i+1}), \quad i = 1, \ldots, r-1, \\
\frac{\partial \ell}{\partial b_i^j} &= \delta_i^j, \quad i = 1, \ldots, r, \quad j = 1, \ldots, d_i, \\
\frac{\partial \ell}{\partial W_i^{jk}} &= \delta_i^j a_{i-1}^k, \quad i = 1, \ldots, r, \quad j = 1, \ldots, d_i, \quad k = 1, \ldots, d_{i-1},
\end{aligned}$$

where $\odot$ *stands for the component-wise Hadamard product of vectors.*

2.3 Recurrent Neural Network (RNN)

2.3.1 Standard Recurrent Neural Network

If we want to work with time series data, we can add an extra temporal dimension to the input data, but FNN doesn't capture the information of previous data. For example for financial time series, it has been show that volatility is mostly path-dependant up to a 90% [21]. This motivate us to use recurrent neural network (RNN), which constantly take advantage of the output at each time step of the input data. For example, if we have a vector of sequential data $\mathbf{x} \in \mathbb{R}^{n \times T}$ as input into the RNN, it will process $\mathbf{x}_i \in \mathbf{x}$ one at a time and learn from the output generated by previous data, $\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$. In other words, it can capture the correlation between the current and previous data input, which is beneficial for processing the time series data. Mathematically we can defined as follows:

Definition 2.3.1 (Recurrent Neural Networks)**.** Let $n, m, k, T \in \mathbb{N}$, define the hidden state at each time step $t = 1, \ldots, T$ as $\mathbf{h}_{(t)} : \mathbb{R}^m \to \mathbb{R}^m$, weight matrices for hidden-to-hidden connection as $\mathbf{W} \in \mathbb{R}^{m \times m}$, for input-to-hidden connection as $\mathbf{U} \in \mathbb{R}^{m \times n}$ and hidden-to-output connection as $\mathbf{V} \in \mathbb{R}^{k \times m}$, and bias vector $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^k$. Denote component-wise activation functions as $\sigma_h, \sigma_o : \mathbb{R}^m \to \mathbb{R}^m$, we can represent the hidden state and output at each time stamp t as:

$$\mathbf{h}_{(t)} = \sigma^h \left(\mathbf{W}\mathbf{h}_{(t-1)} + \mathbf{U}\mathbf{x}_{(t)} + \mathbf{b} \right), \tag{2.3.1}$$

$$\mathbf{o}_{(t)} = \sigma^o \left(\mathbf{V}\mathbf{h}_{(t)} + \mathbf{c} \right). \tag{2.3.2}$$

Now, we observe that the RNN is a more complex architecture compared to the FNN. We have a extra hidden state $\mathbf{h}_t$, that takes into account the previous hidden state $\mathbf{h}_{t-1}$ and the input at that time $\mathbf{x}_t$. We can interpret the RNN in two ways, folded and unfolded, as shown on the left-hand and right-hand sides of the figure 2.2.

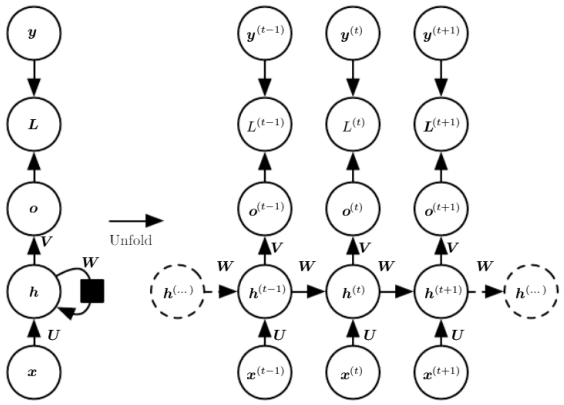


Figure 2.2: Representing graphically the RNN structure folded and unfolded. Source from [19]

For folded RNN (left-hand side of the figure 2.2), it represents the overall structure of the neural network with states that represent the whole time-series vector as one input sample. So, we would understand the RNN similar to a FNN with a dense layer h, a activation function σ, and finally computing the loss function L. By unfolding the RNN, we show how actually works the RNN by representing the sample data as one time step vector inputs into the neural network. In the right-hand side of the figure 2.2 shows how hidden states depends in their previous hidden state until reaching the first time step. In summery, the folded perspective shows how is the model overall, where we can even add more recurrent layer to the model if it is required by complexity of the problem. On the other hand, the unfolded perspective shows you how the recurrent layer actually works.

However, we can face the Vanishing Gradient Problem in the training process for long senquences of data. This problem arise when the gradients shrink exponentially as they are propagated backward through time. This typically happens when the network's activation functions (such as the sigmoid or tanh functions) squash their inputs to small values in the range of $(0, 1)$ for sigmoid or $(-1, 1)$ for tanh. During backpropagation, these small derivatives are multiplied together across many time steps. If the network has many layers or steps, the product of these small numbers can become extremely small, effectively leading the gradient to "*vanish*". When the gradient vanishes, it becomes too small to contribute meaningfully to the weight updates, particularly for layers or time steps far back in the network. As a result, the network fails to learn long-range dependencies because the earlier layers receive very little signal to adjust their weights.

2.3.2 Long Short-Term Memory (LSTM)

One of the most popular RNN is the Long Short-Term Memory (LSTM) algorithm, firstly develop by [22]. It solves the vanishing gradient problem by altering the connection weight matrices between hidden states at each time stamp, avoiding a situation that the same weight matrix repeatedly multiplied by itself. The crucial step is to let the network *forget* some of the older states. In other words, LSTM can learn when and where to remove irrelevant information from the past.

The overall structure of LSTM is similar to the regular RNN regarding the unfolded structure. The different between the standard RNN is the computation within the hidden

or cell state. The special setting in LSTM is the *memory cell state* $\mathbf{C}_t$ which is the stable information flow carrying the relevant historical and current data at all time stamps and will not fade away as t goes further. The LSTM cell structure is illustrated in figure 2.3. It is updated at every time step by three control gates: the forget gate, the input gate and output gate.

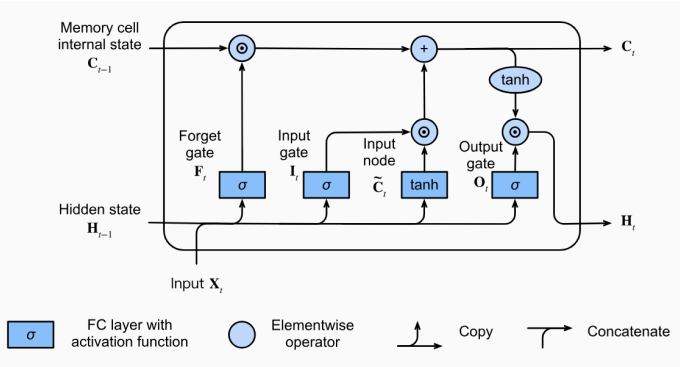


Figure 2.3: The graphical representation of a LSTM cell structure. Source from [12].

The **forget gate** $\mathbf{F}_t$ takes the previous hidden states $\mathbf{H}_{t-1}$ and current data $\mathbf{x}_t$ as inputs, determines the relevance of the information and *forgets* the previous irrelevant information in $\mathbf{C}_{t-1}$ by applying a sigmoid activation function indicating the level of importance.

The **input gate** $\mathbf{i}_t$ is going to decide the *update* of the current cell states, determining how much current information should be obtained based on the new candidate memory cell state $\tilde{\mathbf{C}}_t$ also by applying a sigmoid activation function, determining the level of importance. The final output from this gate will add to $\mathbf{C}_{t-1}$.

At this stage, we have our new cell state $\mathbf{C}_{t-1}$ updated to $\mathbf{C}_t$ by forgetting irrelevant information from the past and adding new information based on current data. A similar procedure is then followed by the **output gate** $\mathbf{O}_t$, which also determines the level of importance by applying a sigmoid activation function to $\mathbf{x}_t$ and $\mathbf{H}_{t-1}$. Combining the indicator with the current cell state $\mathbf{C}_t$, we have our output $\mathbf{H}_t$.

We can defined mathematically this procedure as follows:

Definition 2.3.2 (LSTM forward propagation)**.** Let $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_c, \mathbf{W}_o \in \mathbb{R}^{m\times m}, \mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_c, \mathbf{U}_o \in \mathbb{R}^{m\times n}, \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^m$, $m, n, T \in \mathbb{N}_+$, and $\mathbf{x}_t \in \mathbb{R}^n, \mathbf{H}_t \in \mathbb{R}^{m\times 1}$, where $t = 1, \ldots, T$, and a component-wise sigmoid function (σ) and hyperbolic tangent (tanh).

Define $\mathbf{F}_t, \mathbf{I}_t, \mathbf{C}_t, \mathbf{O}_t : \mathbb{R}^{m\times 1} \to \mathbb{R}^{m\times 1}$ such that:

$$\mathbf{F}_t = \sigma(\mathbf{W}_f\mathbf{H}_{t-1} + \mathbf{U}_f\mathbf{x}_t + \mathbf{b}_f), \tag{2.3.3}$$

$$\mathbf{I}_t = \sigma(\mathbf{W}_i\mathbf{H}_{t-1} + \mathbf{U}_i\mathbf{x}_t + \mathbf{b}_i), \tag{2.3.4}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c\mathbf{H}_{t-1} + \mathbf{U}_c\mathbf{x}_t + \mathbf{b}_c), \tag{2.3.5}$$

$$\mathbf{O}_t = \sigma(\mathbf{W}_o\mathbf{H}_{t-1} + \mathbf{U}_o\mathbf{x}_t + \mathbf{b}_o), \tag{2.3.6}$$

where $\mathbf{F}_t$ is the **forget gate**, $\mathbf{I}_t$ is the **input gate**, $\tilde{\mathbf{C}}_t$ is the *candidate memory* and $\mathbf{O}_t$ is the **output gate**.

$$\mathbf{C}_t = \mathbf{F}_t\mathbf{C}_{t-1} + \mathbf{I}_t\tilde{\mathbf{C}}_t, \tag{2.3.7}$$

and the final output at time t is

$$\mathbf{H}_t = \mathbf{O}_t \tanh(\mathbf{C}_t). \tag{2.3.8}$$

Chapter 3

Market Generator: Neural SDE

This chapter will explore the model of Neural SDE that the authors in [27] developed.

3.1 Neural SDE

We will introduce first the neural SDE generator with the same notation as the authors in the original paper did [27]. We take $(\Omega, \mathcal{F}, \mathbb{P})$ as the underlying probability space. Let $T > 0$ and $d_x \in \mathbb{N}$. Denote by $\mathcal{X}$ the space of continuous paths of bounded variation from $[0, T]$ to $\mathbb{R}^{d_x}$ with one monotone coordinate i.e. the time dimension. For any random variable X with values on $\mathcal{X}$, we denote by $\mathbb{P}_X := \mathbb{P} \circ X^{-1}$ its law.

Let $W : [0, T] \to \mathbb{R}^{d_w}$ be a d_w-dimensional Brownian motion and $a \sim \mathcal{N}(0, I_{d_a})$ be sampled from d_a-dimensional standard normal. The values $d_w, d_a \in \mathbb{N}$ are hyperparameters describing the size of the noise, and initial noise respectively. A Neural SDE is a model with the following architecture

$$Y_0 = \xi_\theta(a), \quad dY_t = \mu_\theta(t, Y_t)dt + \sigma_\theta(t, Y_t) \circ dW_t, \quad X_t^\theta = A_\theta Y_t + b_\theta \tag{3.1.1}$$

for $t \in [0, T]$, with $Y : [0, T] \to \mathbb{R}^{d_y}$ the strong solution, if it exists, to the Stratonovich SDE, where

$$\xi_\theta : \mathbb{R}^{d_a} \to \mathbb{R}^{d_y}, \quad \mu_\theta : [0, T] \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y}, \quad \sigma_\theta : [0, T] \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y \times d_w}$$

are regular neural networks, and $A_\theta \in \mathbb{R}^{d_x \times d_y}, b_\theta \in \mathbb{R}^{d_x}$. The dimension $d_y \in \mathbb{N}$ is a hyperparameter describing the size of the hidden state. If $\mu_\theta, \sigma_\theta$ are Lipschitz and $\mathbb{E}_a[\xi_\theta(a)] < \infty$ then the solution Y exists and is unique.

Given a target $\mathcal{X}$-valued random variable X^{true} with law $\mathbb{P}^{X^{\text{true}}}$, the goal is to train a Neural SDE so that the generated law $\mathbb{P}^{X^\theta}$ is as close as possible to $\mathbb{P}^{X^{\text{true}}}$, for some appropriate notion of closeness that we will discuss in the following section.

3.2 Non-Adversary training with Signature Kernels

In this section we will explain briefly the mathematical concepts behind the training process, for more detail on the mathematical theory we encourage the reader to read [9].

3.2.1 Signature Transformation

In order to understand a signature kernel and how it can be used to train a neural SDE, we need first to introduce the signature transformation. Signature transformations are a core concept in rough path theory. They arise naturally as the solution of a controlled

differential equation (CDE), ie $dy_t = f(y_t)dX_t$, with the classical method of Picard iteration [9, Section 1.1.1]. The CDE's challenge consists in modelling the response over an interval of time generated by the interaction of a driving signal with a nonlinear control system. Usually, signature transformations are introduced in a general context of Banach spaces [9], but here we will introduce them in our context of the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ previously defined.

Definition 3.2.1 (Signature Transformation)**.** The *signature map* $S : \mathcal{X} \to \mathcal{T}$ is defined for any path $x \in \mathcal{X}$ as the infinite collection $S(x) = (1, S^1(x), S^2(x), \dots)$ of iterated Riemann-Stieltjes integrals

$$S^k(x) := \int_{0<t_1<\cdots<t_k<T} dx_{t_1} \otimes dx_{t_2} \otimes \cdots \otimes dx_{t_k}, \quad k \in \mathbb{N},$$

where $\otimes$ is the standard tensor product of vector spaces and $\mathcal{T} := \mathbb{R} \oplus \mathbb{R}^{d_x} \oplus (\mathbb{R}^{d_x})^{\otimes 2} \oplus \dots$.

The keys on why the signature transformations are so useful in a context of modelling paths are their analytical properties. Firstly, they exhibit invariance under reparameterizations; this essentially allows the signature transformation to act as a filter that removes an infinite dimensional group of symmetries given by time reparameterization. Secondly, the uniqueness of signature transformation for a certain class of paths, ensuring a one-to-one identifiability of a path with its signature transformation. These analytical properties tackles an important obstacle that most machine learning models have to face, the potential symmetry present in the data.

Here we will present a classical example of symmetry under reparametrization used in [36]. Consider the reparametrization $\varphi : [0, 1] \to [0, 1]$ given by $\varphi(t) = t^2$ and the path $\gamma : [0, 1] \to \mathbb{R}^2$ defined by $\gamma_t = (\gamma_t^x, \gamma_t^y)$ where $\gamma_t^x = \cos(10t)$ and $\gamma_t^y = \sin(3t)$. As it is clearly depicted in Figure 3.1, both channels (γ^x, γ^y) of γ are individually affected by the reparametrization φ, but the shape of the curve γ is left unchanged. Building an invariance into a model is usually very hard. However, the signature transformation wouldn't have this problem.

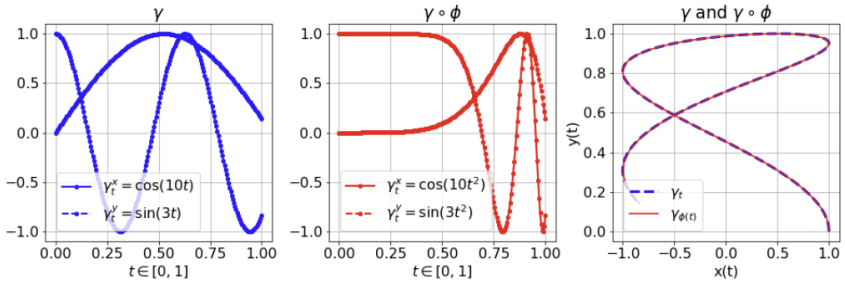


Figure 3.1: On the left are the individual channels (γ^x, γ^y) of a 2D path γ. In the middle are the channels reparametrized under $\varphi : t \mapsto t^2$. On the right are the path γ and its reparametrized version $\gamma \circ \varphi$. The two curves overlap, meaning that the reparametrization φ represents irrelevant information if one is interested in understanding the shape of γ. Figure taken from [36].

3.2.2 Signature Kernels

One of the main drawbacks of incorporating the signature transformation for our model is their expensive computational cost, and using approximations of it i.e. the truncated signature transformation. However, there is a way of tackle this using a kernel trick.

Kernel methods are well-established tools in machine learning which are fundamental to support vector machine models for classification, nonlinear regression and outlier detection involving small or moderate-sized data sets, [39, 8, 38]. The essence of these methods is to achieve better separation between labelled data by embedding a low-dimensional feature space X into a higher dimensional one H, which is commonly assumed to be a Hilbert space, by means of a feature map $\psi : X \to H$. The associated kernel is a function $K : X \times X \to \mathbb{R}$ with the property that $\langle \psi(x), \psi(y) \rangle_H = K(x, y)$ for all x and y in X. In many situations, the kernel can be efficiently evaluated with no reference to the feature map, a property commonly referred to as kernel trick. This property allows one to benefit from the advantages of working in a higher dimensional feature space without the associated drawbacks. With this notion we will introduce the signature kernel.

Definition 3.2.2 (Signature Kernel)**.** The signature kernel $k_{\text{sig}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric positive semidefinite function defined for any pair of paths $x, y \in \mathcal{X}$ as

$$k_{\text{sig}}(x, y) := \langle S(x), S(y) \rangle_{\mathcal{T}},$$

where $\langle \cdot, \cdot \rangle_{\mathcal{T}}$ is a canonical Hilbert-Schmidt inner product on $\mathcal{T}$ [1].

In [36] the authors provide a novel kernel trick to the signature kernel's problem. They proved that solving a signature kernel is the same as solving a Goursat partial differential equation (PDE). The signature kernel satisfies the following:

$$k_{\text{sig}}(x, y) = f(T, T) \quad \text{where} \quad f(s, t) = 1 + \int_0^s \int_0^t f(u, v) \langle dx_u, dy_v \rangle_1, \tag{3.2.1}$$

which can be transformed into the following hyperbolic PDE:

$$\frac{\partial^2 f(s, t)}{\partial s \partial t} = \langle \dot{x}_s, \dot{y}_t \rangle_1 f(s, t)$$

where $\dot{x}_s = \frac{dx_p}{dp}\Big|_{p=s}$ and $\dot{y}_t = \frac{dx_q}{dq}\Big|_{q=t}$ are the derivatives of x and y at time s and t, respectively.

This trick reduces the problem of computing the signature for both paths, and then their inner product, into solving a linear hyperbolic PDE. For our results we will use the same finite difference numerical method to solve it as they did in [36].

3.2.3 Signature Kernel Scores

We will denote by $\mathcal{H}$ the unique reproducing kernel Hilbert space (RKHS) of k_{sig}. From now on we endow $\mathcal{X}$ with a topology with respect to which the signature is continuous; see [10] for various choices of such topologies. Denote by $\mathcal{P}(\mathcal{X})$ the set of Borel probability measures on $\mathcal{X}$. The following proposition connects the signature kernel to any probability measure.

Proposition 3.2.3. *[27, Proposition 3.1] The signature kernel is characteristic for every compact set $\mathcal{K} \subset \mathcal{X}$, i.e., the map $\mathbb{P} \mapsto \int k_{sig}(x, \cdot)\, \mathbb{P}(dx)$ from $\mathcal{P}(\mathcal{K})$ to $\mathcal{H}$ is injective.*

Remark 3.2.4. In particular, Proposition 3.1 implies that the signature kernel is cc-universal, i.e., for every compact subset $\mathcal{K} \subset \mathcal{X}$, the linear span of the set of path functionals $\{k_{\text{sig}}(x, \cdot) : x \in \mathcal{K}\}$ is dense in $\mathcal{C}(\mathcal{K})$ in the topology of uniform convergence.

[1] More exactly, any inner product $\langle \cdot, \cdot \rangle_1$ on $\mathbb{R}^{d_x}$ yields a canonical Hilbert-Schmidt inner product $\langle \cdot, \cdot \rangle_k$ on $(\mathbb{R}^{d_x})^{\otimes k}$ for any $k \in \mathbb{N}$, which in turn yields, by linearity, a family of inner products $\langle \cdot, \cdot \rangle_{\mathcal{T}}$ on $\mathcal{T}$. By a slight abuse of notation, we use the same symbol to denote the Hilbert space obtained by completing $\mathcal{T}$ with respect to $\langle \cdot, \cdot \rangle_{\mathcal{T}}$.

Definition 3.2.5 (Signature Kernel Score)**.** We define the *signature kernel score* $\phi_{\text{sig}} : \mathcal{P}(\mathcal{X}) \times \mathcal{X} \to \mathbb{R}$ for any $\mathbb{P} \in \mathcal{P}(\mathcal{X})$ and $y \in \mathcal{X}$ as

$$\phi_{\text{sig}}(\mathbb{P}, y) := \mathbb{E}_{x,x' \sim \mathbb{P}}[k_{\text{sig}}(x, x')] - 2\mathbb{E}_{x \sim \mathbb{P}}[k_{\text{sig}}(x, y)]. \tag{3.2.2}$$

A highly desirable property to require from a score is its strict properness, consisting in assigning the lowest expected score when the proposed prediction is realised by the true probability distribution. The following proposition justifies that the signature score is strict properness.

Proposition 3.2.6. *[27, Proposition 3.2] For any compact $\mathcal{K} \subset \mathcal{X}$, ϕ_{sig} is a strictly proper kernel score relative to $\mathcal{P}(\mathcal{K})$, i.e.,$\mathbb{E}_{y \sim \mathbb{Q}}[\phi_{sig}(\mathbb{Q}, y)] \leq \mathbb{E}_{y \sim \mathbb{Q}}[\phi_{sig}(\mathbb{P}, y)]$for all $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{K})$, with equality if and only if $\mathbb{P} = \mathbb{Q}$.*

This score quantifies how "close" are the paths y to the probability measure $\mathcal{P}$. This sense of distance is often called the signature kernel *maximum mean discrepancy* (MMD), and it is the RKHS distance between two probability measures $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{K})$ defined as

$$\mathcal{D}_{k_{\text{sig}}}(\mathbb{P}, \mathbb{Q})^2 = \mathbb{E}_{y \sim \mathbb{Q}}[\phi_{\text{sig}}(\mathbb{P}, y)] + \mathbb{E}_{y,y' \sim \mathbb{Q}}[k_{\text{sig}}(y, y')].$$

The authors also proposed a consistent and unbiased estimator of ϕ_{sig} given m sample paths $\{x^i\}_{i=1}^m \sim \mathbb{P}$ as follows:

$$\hat{\phi}_{\text{sig}}(\mathbb{P}, y) = \frac{1}{m(m-1)} \sum_{j \neq i} k_{\text{sig}}(x^i, x^j) - \frac{2}{m} \sum_i k_{\text{sig}}(x^i, y). \tag{3.2.3}$$

With all this theoretical background the authors in [27] proposed the following setting for training a Neural SDE. Given a target $\mathcal{X}$-valued random variable X^{true} with law $\mathbb{P}_{X^{\text{true}}}$. Recall the notation $\mathbb{P}_{X^\theta}$ for the law generated by the Neural SDE equation 3.1.1. The training objective is given by

$$\min_\theta \mathcal{L}(\theta) \quad \text{where} \quad \mathcal{L}(\theta) = \mathbb{E}_{y \sim \mathbb{P}_{X^{\text{true}}}} \left[\phi_{\text{sig}} \left(\mathbb{P}_{X^\theta}, y \right) \right].$$

Combining equations 3.1.1, 3.2.1, 3.2.2, and 3.2.3, the generator-discriminator pair can be evaluated by solving a system of linear PDEs depending on sample paths from the Neural SDE; in summary:

Generator: $X^\theta \sim \text{SDESolve}(\theta)$ **Discriminator:** $\mathcal{L}(\theta) \approx \text{PDESolve}\left(X^\theta, X^{\text{true}}\right)$.

Chapter 4

Implementation of our Deep Hedging framework

In the previous chapters we introduced the theoretical background in which our framework is rooted on. This chapter presents the implementation of the models that we use for the chapter 5 of numerical results.

4.1 Implementation of the Deep Hedging model

The indifference pricing framework presented in chapter 1 is a robust and versatile theory for pricing and hedging, however lacks in providing a numerical optimizer for the optimization problem $\pi(X)$ defined in the equation 1.3.1. A great solution to this problem is the use of neural networks to find the optimal hedging strategies to minimise $\rho(X)$. We have shown in chapter 2 with the Universal Approximation Theorem 2.1.2 that the neural networks are versatile functions that can in theory approximate anything. Furthermore, the loss function needs to be a convex function to ensure convergence, thus the convex risk measure $\rho(X)$ works perfectly as the loss function of the neural network model. The authors in [4] realise this, and develop the first Deep Hedging model.

4.1.1 Estimating the Indifference Price

Now we our hedging ratios, defined at the subsection 1.2.1, will be the output of our neural networks $\delta^\theta \in \mathcal{H}$. Our portfolio's wealth will depend on our neural networks.

$$PnL_T(Z, p_0, \delta^\theta) = -Z + p_0 + (\delta^\theta \cdot S)_T - C_T(\delta^\theta),$$

For our numerical experiments we use the following proportional transaction costs:

$$C_T(\delta^\theta) = \sum_{i=1}^{d} k \left(\delta^\theta_{0,i} S_{0,i} + \sum_{t=2}^{T} \left[\delta^\theta_{t-1,i} - \delta^\theta_{t-2,i} \right] S_{t-1,i} + \delta^\theta_{T,i} S_{T,i} \right)$$

with k as the proportional cost rate.

As mentioned before now, the loss function of our neural networks will be the entropic risk measure 1.3.4. For the general framework, we will trained two neural networks. The first one $\boldsymbol{f}$ will be trained using the entropic risk measure of the liability Z that we want to hedge $\rho^\theta(-Z)$, and the second one $\boldsymbol{f_0}$ with the entropic risk measure of the market $\rho^\theta(0)$. Our loss function ℓ will be defined as:

$$\ell(\delta^\theta, S; Z, \lambda, k) = \exp\left(-\lambda \left(-Z + (\delta^\theta \cdot S)_T - C_T(\delta^\theta) \right) \right), \tag{4.1.1}$$

so the logarithmic of our empirical minibatch risk $\mathcal{L}_B(\theta)$ defined in 2.2.1 will match $\lambda\rho^\theta(-Z)$. Therefore, the entropic risk measure will be computed by:

$$\rho^\theta(-Z) = \frac{1}{\lambda}\log\left(\mathcal{L}_B(\theta)\right) = \frac{1}{\lambda}\log\left[\frac{1}{\#B}\sum_{i\in B}\ell(\delta_i^\theta, \mathbf{S}_i; Z, \lambda, k)\right] \tag{4.1.2}$$

where B is the minibatch defined next to the equation 2.2.1. We note that, as we said before, the delta ratios δ_i^θ are the outputs of the neural networks, and the path prices of the underlyings $\mathbf{S}_i$ are the reference values that they will use to evaluate their outcome. This is indeed an unsupervised learning problem, although this problem can be also reformulated to be a reinforcement learning setup, see [6] for more details.

We also note that as our neural networks are trained and optimized the $\rho^\theta(-Z)$ will be minimised until convergence to $\pi^\theta(-Z)$. Therefore, our estimate of the indifference price of the liability Z is:

$$\hat{p}(Z) = \pi^\theta(-Z) - \pi^\theta(0) = \frac{1}{\lambda}\log\left[\frac{\left(\frac{1}{\#B}\sum_{i\in B}\ell(\boldsymbol{f}^\theta(\mathbf{X}_i), \mathbf{S}_i; Z, \lambda, k)\right)}{\left(\frac{1}{\#B}\sum_{i\in B}\ell(\boldsymbol{f_0}^\theta(\mathbf{X}_i), \mathbf{S}_i; 0, \lambda, k)\right)}\right] \tag{4.1.3}$$

where X will be our input to the neural network that it may vary depending the liability or the neural network.

4.1.2 Estimating the Hedging Ratios under a Risk-Neutral Measure

A naive approach to estimate the hedging ratios given some stock price paths would be just to use the deltas form the $\boldsymbol{f}$ neural network i.e. $\delta_Z^\theta = \boldsymbol{f}(\mathbf{X})$. However, this would only properly work in a driftless market, i.e. $\pi(0) = 0$. In a general market generator it is not guarantee that the hedging strategy δ_Z^θ would completely hedge the risk of the derivative. It may focus on taking advantage of the statistical arbitrage of the market, instead of focusing in hedging the risks of the liability. If we want to find the optimal hedging strategy to hedge the liability risks, we need to take out the statistical arbitrage from the market by changing the measure to a risk neutral measure. The following theorem adapted from [5, Theorem 2.2] provides the theoretical background for this change of measure.

Theorem 4.1.1 (Robustly removing the Drift under Transaction Costs and Trading Constraints)**.** *Assume that the market is constrained and that transaction costs c_t are super-additive. Suppose that $\delta^* \in \mathcal{H}$ is a (not necessarily unique) policy that minimizes*

$$\mathbb{E}_{\mathbb{P}}\left[e^{-\lambda G(\delta)}\right],$$

satisfying $\mathbb{E}_{\mathbb{P}}\left[e^{-\lambda G(\delta^*)}\right] > 0$, where $G(\delta) = (\delta \cdot S)_T - C_T(\delta)$ is the gain process.

Then, the market under the measure $\mathbb{Q}^*$ given by

$$\frac{d\mathbb{Q}^*}{d\mathbb{P}} = \frac{e^{-\lambda G(\delta^*)}}{\mathbb{E}_{\mathbb{P}}\left[e^{-\lambda G(\delta^*)}\right]}$$

is free from statistical arbitrage for any transaction cost $c_t' \geq c_t$ and any tighter constraints $\mathcal{H}_t' \subseteq \mathcal{H}_t$.

Further more if we consider a setting with zero transaction costs $c = 0$, then this measure $\mathbb{Q}^*$ it is actually the minimal entropy martingale measure (MEMM) in the sense that it minimizes the relative entropy

$$H(\mathbb{Q} \mid \mathbb{P}) = \mathbb{E}_{\mathbb{Q}} \left[\log \frac{d\mathbb{Q}}{d\mathbb{P}} \right]$$

over all equivalent martingale measures $\mathbb{Q}$. More detail of pricing under the MEMM can be found here [33].

In our deep hedging implementation, we have actually already find the optimal δ^*. We note that minimising $\mathbb{E}_{\mathbb{P}} \left[e^{-\lambda G(\delta)} \right]$ is what are we are doing with the $\boldsymbol{f_0}$ neural network, which is optimizing the entropic risk measure of the market. Thus, $\boldsymbol{f_0}(\mathbf{X}) = \delta_0^\theta = \delta^*$. Therefore, in our setting this change of measure $\mathbb{Q}^*$ would mean to reweight each path from our market generator so they follow the probability:

$$q^* = \frac{e^{-G(\delta_0^\theta)}}{\sum e^{-G(\delta_0^\theta)}}$$

However, the following corollary found in [5, Corollary 4.1] allows us another simpler approach in our Deep Hedging problem.

Corollary 4.1.2 (Optimal policy for Risk Neutral Deep Hedging)**.** *Assume transaction costs are zero. Under the statistical measure $\mathbb{P}$ suppose that δ_Z^θ is a solution to the Deep Hedging problem for Z, and that δ_0^θ is an optimal statistical arbitrage policy.*

Then the policy $\delta_{\mathbb{Q}^*} = \delta_Z^\theta - \delta_0^\theta$ is a solution to the Deep Hedging problem under the minimal entropy martingale measure $\mathbb{Q}^*$.

Remark 4.1.3. This corollary 4.1.2 is **key** in the implementation of our hedging models. The strategy develop by the neural network $\boldsymbol{f}$,δ_Z^θ, we will call it the strategy **Delta Z**, which it will combine the influence of the statistical arbitrage of the market with the need of hedging the liability. In the other hand, **Delta Q**, $\delta_{\mathbb{Q}^*}$ is the difference between the strategies developed by $\boldsymbol{f}$ and $\boldsymbol{f_0}$ i.e . $\delta_Z^\theta - \delta_0^\theta$, and they will be under the risk neutral measure, disregarding the possible statistical arbitrage in the model, and only focusing in hedging the liability. We also note that from now on we will name the optimal statistical arbitrage strategy build by $\boldsymbol{f_0}$ as **Delta 0**.

4.1.3 Architecture and training optimization

For our numerical experiments we will use two neural networks. A FNN, defined and described in 2.1, and a RNN, more specific a LSTM layer, previously explained and described in 2.3.2. The FNN will be constituted by the following structure:

$$f_{FNN} \in \mathcal{N}_4(2, 100, 100, 100, 1; ReLU, ReLU, ReLU, sigmoid),$$

The network takes in a 2-dimensional input at each time stamp, i.e., input $\mathbf{X} = (X_{i,j})_{N \times T}$ where $X_{i,j} = \left(\frac{j}{T}, S_j^{(i)} \right), i = 1, \ldots, N$ and $j = 0, \ldots, T-1$. This input is designed for pricing European Call Options, for other path dependent options, we could use some extra label in the input. Our FNN structure has 3 hidden layers, for which each of them has 100 units, and equipped with $ReLU$ as the activation function. The output layer has a 1 dimension output, as mention before, representing the hedge ratio for an underlying asset. If we would like to extend it to d multi-asset hedging, the dimensional output would be obviously d. The output is bounded by the sigmoid so the hedging ratios are in between (0,1), since financial institution hedging ratios for call options should be in that range.

In the other hand, the RNN is constituted by one dimension input layer, one LSTM layer of 20 units i,e, 20 hidden dimensions; followed by a sigmoid activation function and one output dimension layer. The input of the RNN $\mathbf{X}$ can be only the stock path or also

with the time label like in FNN. Both are valid option, unlike the FNN architecture, which requires the temporal label too.

For the training process we split the data in minibatches of size 256, and we use the Adam optimizer described in 2.2.3 with learning rate 10^{-3}. We train the data 40 epochs for the FNN and 55 epochs for the RNN to ensure convergence throughout all the different risk aversion levels λ and different cost levels k.

4.2 Implementation of the Neural SDE

If we want to use this Deep Hedging model in a practical setup, we need a market generator that can replicate not only the real market data return distribution but also the path's distribution. A perfect choice for this matter is the Neural SDE model develop in [27], and already presented and explain in detail in chapter 3.

For our numerical experiments we use the SP500[1] hourly data from 01/04/2007 to 24/03/2021. We split the data in paths of 30 time steps with stripe length of one time step. Each path is standardized, i.e. $\hat{X}_t = \frac{X_t - \mu_T}{\sigma_T}$, where μ_T is the mean value of the whole path and σ_T the standard deviation. Then, we apply a filter to get rid off the noise data, we eliminate the 0.985 percentile of the most total variation paths and absolute return at time step 5 to eliminate the paths with a big variation at the beginning. Also, we subtract the initial value of the path, so each path start at 0; and normalised the time label to be in between $[0, 2]$, when feed it into the discriminator. These transformations are made to improve training the model. We organised the paths in random batches of 128 paths from the whole data set that we will feed into the training process.

For the architecture described in 3.1.1, we used an initial noise size $d_a = 5$, a noise size of $d_w = 8$, a hidden state size of $d_y = 16$. The neural networks μ_θ and σ_θ have a depth of 3 hidden layers of 32 size with LipSwish[2] as activation function and tanh as final activation function, thus their architecture are

$$\mu_\theta \in \mathcal{NN}(16, 32, 32, 32, 16; \mathit{LipSwish}, \mathit{LipSwish}, \mathit{LipSwish}, \mathit{tanh})$$

and

$$\sigma_\theta \in \mathcal{NN}(16, 32, 32, 32, 128; \mathit{LipSwish}, \mathit{LipSwish}, \mathit{LipSwish}, \mathit{tanh}).$$

For the SDE solver we used the Euler–Maruyama method for an Ito SDE integration with $dt = 1$ over $[0, 29]$.

Regarding the discriminator described in 3.2.3, we used a radial basis function kernel or RBF kernel, i.e. $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$, with $\gamma = 1$ and $\|\cdot\|^2$ as the squared Euclidean distance. For the PDE solver we used a dyadic order of 1, which is related to the mesh size of PDE solver used in the loss function, for more detail [36, Section 3.1].

For the training process, we use the Adam optimizer with learning rate of 10^{-4}, and we run the process for 4000 steps, where at each step we evaluate one batch, and optimise the model. Regarding the backpropagation we did the same approach as the authors in the original paper " optimise-then-discretise", more detail about the approach in [29, section 5.2.3].

4.2.1 Evaluation Metrics

We will use three metrics to quantify how close are the path distributions of the generated paths $X^\theta \sim \mathbb{P}_{X^\theta}$ to the real paths $X^{\text{true}} \sim \mathbb{P}_{X^{\text{true}}}$.

[1] The S&P 500 is a stock market index that measures the performance of 500 of the largest publicly traded companies in the United States, representing a broad cross-section of the U.S. economy.

[2] The LipSwish is a similar activation function to SiLU, $\mathit{LipSwish}(x) = x\sigma(x)/1.1$ where σ is the sigmoid function.

1. **The Kolmogorov-Smirnov (KS) test**: The KS test is a non-parametric test that compares the cumulative distribution functions (CDFs) of two samples, to discuss whether both samples come from the same distribution. The null hypothesis H_0 is that the two samples are from the same distribution. The KS score is

$$D_t = \sup_x \left| \hat{F}_{t,X^{\text{true}}}(x) - \hat{F}_{t,X^\theta}(x) \right|$$

where $\hat{F}_{t,X}$ are the empirical cumulative distributions function of the paths distributions of real and generated data up to time t. The rejection of the null hypothesis H_0 happens at α level of significance if

$$D_t > \sqrt{-\log\left(\frac{\alpha}{2}\right) \cdot \frac{1}{N}}$$

where N is the total number of paths in the test, for our setup it will be one batch $N = 128$. We repeated this test 5000 times at the 5% significance level and reported the average KS score along with the average Type I error for different time marginals t.

2. **Autocorrelation**: To measure temporal dependencies or correlation from the real data and the generated one, we will compare both autocorrelation functions

$$\text{ACF}_\ell = \frac{1}{N\sigma^2} \sum_{t=\ell}^{N} (X_t - \mu)(X_{t-\ell} - \mu),$$

where μ is the average of the path X_t over [0,N] and σ^2 its corresponding variance. We will provide the results in a table for different lags, and a plot with the 95% confidence interval.

3. **Cross-correlation**: We provide average cross-correlation scores $(r_t, r^2_{t,\ell})$ between the returns associated to $X_t \sim \mathbb{P}_{X^\theta}$ and the squared, lagged returns $r^2_{t,\ell}$. We present the scores in matrix form. Finally, we provide the mean sqaured error (MSE) between the matrix obtained from $\mathbb{P}_{X^{\text{true}}}$ and the one obtained from the generator.

Chapter 5

Numerical Results and Discussion

In this chapter we present our numerical results of our framework in different setups. First we show how the deep hedging model replicates the Black-Sholes model with ease for different risk aversions and cost levels and both neural networks FNN and LSTM. Then we show the results of the neural SDE model for the SP500 hourly, and its implementation to the Deep Hedging model with a European Call Option as liability. We test the Deep Hedging model not only with the generated data but also with the real data. We compare both hedging strategies the Delta Z, δ_Z^θ, and Delta Q, $\delta_{\mathbb{Q}^*}$. Also, we study the statistical arbitrage of the data with $\pi^\theta(0)$ and how it is affected with different risk aversions and transaction costs levels.

After we have study our models in a elementary setup like the European Call Option, we will explore other exotic options like Arithmetic Asian Call Options, LookBack Options and Rainbow Options. We will compare both FNN and LSTM especially for the path-depended options to check if LSTM leverages from its sequential depended setup. We will study the hedging surfaces of the different options with real market data.

To compare the different deep hedging models and strategies we will provide tables with the following metrics. First the indifference price, we will consider that a lower price is better than a higher one since the NN didn't rely just in making profitable the PnL by increasing the derivative's price but in properly hedging it. Secondly, we consider the super-hedging ratio of the PnL i.e. $\mathbb{P}(\text{PnL} \geq 0)$. Obviously, the greater super-hedging ratio the better PnL distribution, but we will consider too the left tail of the distribution with the $\alpha = 95\%$ Value at Risk (VaR),i.e. the $1-\alpha$ quantile of the PnL distribution, and $\alpha = 95\%$ Conditional Value at Risk (CVaR) or Expected Shortfall (ES), which is the expected VaR from different confidence levels in range of $(\alpha, 1)$. These last two metrics offer us a way to measure the risk in the hedging strategies.

5.1 Black-Scholes Benchmark

We check first how the deep hedging model behaves under different parameters in a well known model as the Black Scholes with analytical formula for pricing and hedging, already explained in section 1.1. We use as market generator the BS discrete formula 1.1.16, with $S_0 = 1$, $r = 0$, $\sigma = 0.5$. We simulate for $T = 31$ time steps, and $N = 10^5$ number of paths and we will consider a European Call Option as liability Z with strike $S_0 = K$.

Since it is a driftless market we don't need to take into account the $\pi(0)$ in the indifference pricing formula 1.3.2, so we don't necessarily train the $\boldsymbol{f_0}$. If we would train it, the $\pi^\theta(0) \approx 0$ as it gets optimized and the hedging ratios δ_0^θ would tend to 0 too. So, it is pointless to consider them in this scenario. Therefore, we will train only $\boldsymbol{f}$ for different risk aversions $\lambda \in \{0.1, 1, 5, 10\}$ and transaction costs $k \in \{0\%, 0.05\%, 0.5\%, 5\%\}$. The indifference price in this setup is $\hat{p}(Z) = \pi^\theta(-Z)$.

Parameters		Indifference Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
0.1	0.00%	**0.1980**	0.1982	0.4860	**0.5298**	**-0.1817**	-0.2128	**-0.2172**	-0.3324
	0.05%	0.1991	**0.1987**	**0.6258**	0.5149	-0.1728	**-0.1510**	**-0.2077**	-0.2384
	0.50%	0.2045	**0.2044**	0.6677	**0.7542**	-0.4083	**-0.4801**	**-0.7142**	-0.7645
	5.00%	**0.2066**	**0.2066**	**0.7336**	**0.7336**	**-0.8086**	**-0.8086**	**-1.3185**	**-1.3185**
1.0	0.00%	**0.1980**	0.1984	0.5413	**0.5721**	**-0.0623**	-0.0791	**-0.0881**	-0.1111
	0.05%	0.1995	**0.1994**	0.4984	**0.5520**	**-0.0594**	-0.0738	**-0.0793**	-0.1033
	0.50%	0.2100	**0.2079**	**0.5445**	0.5408	**-0.0950**	-0.1113	**-0.1295**	-0.1595
	5.00%	0.2706	**0.2666**	0.6589	**0.7479**	**-0.3004**	-0.3822	**-0.5613**	-0.6059
5.0	0.00%	**0.2000**	0.2010	**0.5332**	0.5047	**-0.0486**	-0.0546	**-0.0680**	-0.0760
	0.05%	**0.2014**	0.2485	0.5282	**0.6581**	**-0.0480**	-0.1801	**-0.0694**	-0.2361
	0.50%	0.2135	**0.2122**	**0.5514**	0.5412	**-0.0564**	-0.0614	**-0.0772**	-0.0889
	5.00%	0.3105	**0.2966**	**0.6261**	0.5877	-0.1211	**-0.1114**	-0.1761	**-0.1595**
10.0	0.00%	0.2025	**0.2024**	0.5601	**0.5714**	**-0.0447**	-0.0458	**-0.0647**	-0.0664
	0.05%	0.2039	**0.2041**	0.5504	**0.5644**	**-0.0438**	-0.0459	**-0.0633**	-0.0660
	0.50%	0.2163	**0.2156**	0.5707	**0.5840**	**-0.0463**	-0.0501	**-0.0663**	-0.0737
	5.00%	0.3237	**0.3081**	**0.6417**	0.6158	-0.0773	**-0.0750**	**-0.1076**	-0.1081
Black-Scholes		**0.1974**		**0.5066**		**-0.0509**		**-0.0723**	

Table 5.1: Table with Indifference Prices, Super-Hedging Ratios, VaR, and CVaR for different risk aversion levels and transaction costs using FNN and LSTM for the BS model.

In the table 5.1 we observe the different results for the different parameters and neural networks. We note that the prices are quite similar to the analytical BS, and they increase as we increase the risk aversion and the cost level. The super-hedging ratio values are almost always better than the BS model solution with more than 50% as usual. We see that, in average, these values increase as we increase our risk aversion values and cost levels. Regarding the VaR and CVaR, we observe as we expected that the values would increase significantly as we increase the risk aversion. This is due to the entropic risk measure, as we increase the risk aversion the measure becomes way less favourable to big losses.

Comparing both neural networks FNN and LSTM in average we don't notice a better model, since for some parameters one neural network worked better than the other one. For example, we see that LSTM worked a little better in $\lambda = 1$ and $\lambda = 10$ with lower prices and higher super-hedging ratios. Also, we notice that in average the FNN has thinner left tails in the PnL distribution than LSTM. However, training neural networks it is not an exact science, therefore we cannot conclude that one type of NN is better than the other one in this set up.

In the figures 5.1 and 5.2 we represent the hedging ratios over the asset price at the time step 20 for the different NN. We observe how the neural networks learn unsupervised to match the BS solution by minimising the $\pi^\theta(-Z)$, especially for low cost levels. We see how both NN mimic better the BS solution as we increase the risk aversion, especially for high transaction cost.

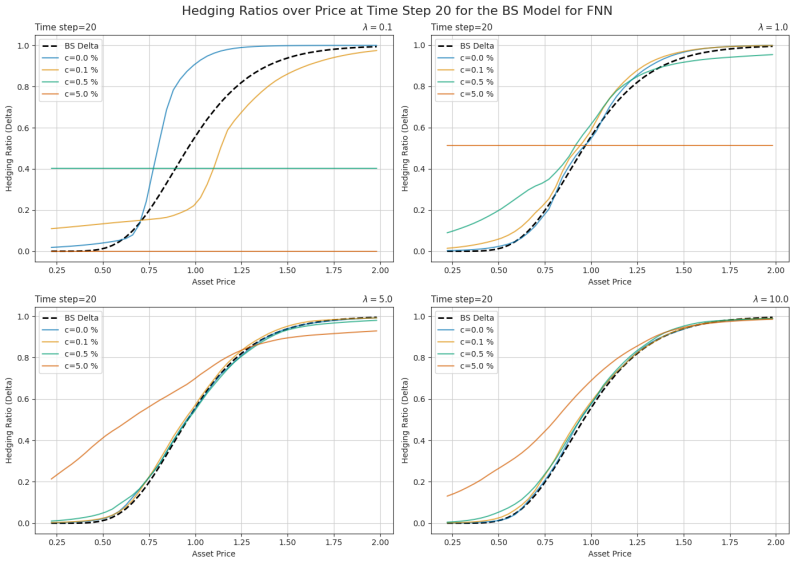


Figure 5.1: The hedging ratios over asset price at the 20th step time for different risk aversions and cost levels for the BS model using FNN.

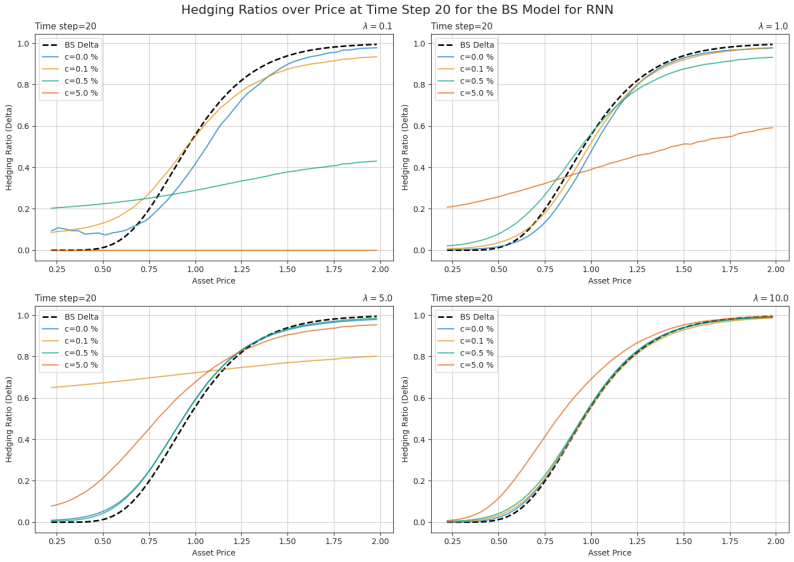


Figure 5.2: The hedging ratios over asset price at the 20th step time for different risk aversions and cost levels for the BS model using the LSTM.

In figures 5.3, 5.4 we represent the hedging ratios over time for one particular path for both NN. Both NN matches the BS solution as the λ is increased. However, we see how they converge to the BS solution with different smoothness. We notice that the LSTM hedging ratios over time are less spiky than FNN and BS. This is due to the nature of the LSTM, since its outputs don't depends only in the current price of the stock price like the BS model or FNN, but on the whole trajectory. But at the end converges to the optimal solution too.

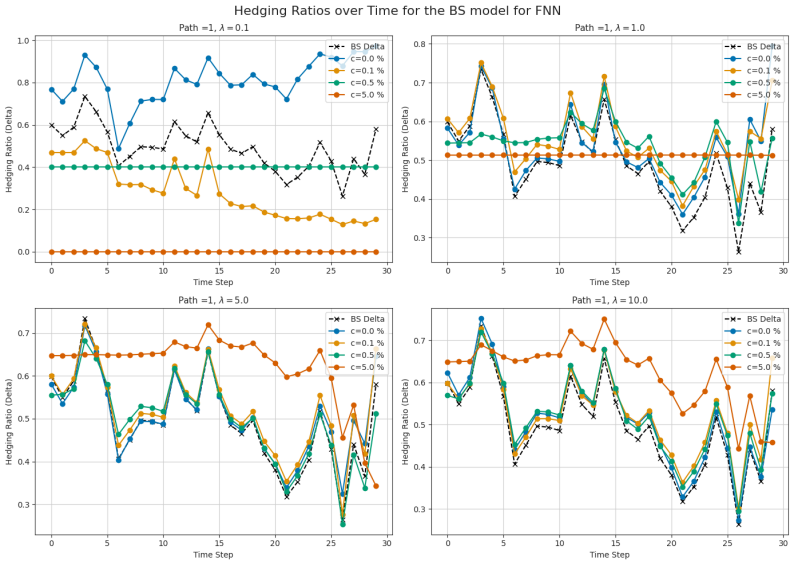


Figure 5.3: The hedging ratios over time for different risk aversions and cost levels for the BS model for FNN.

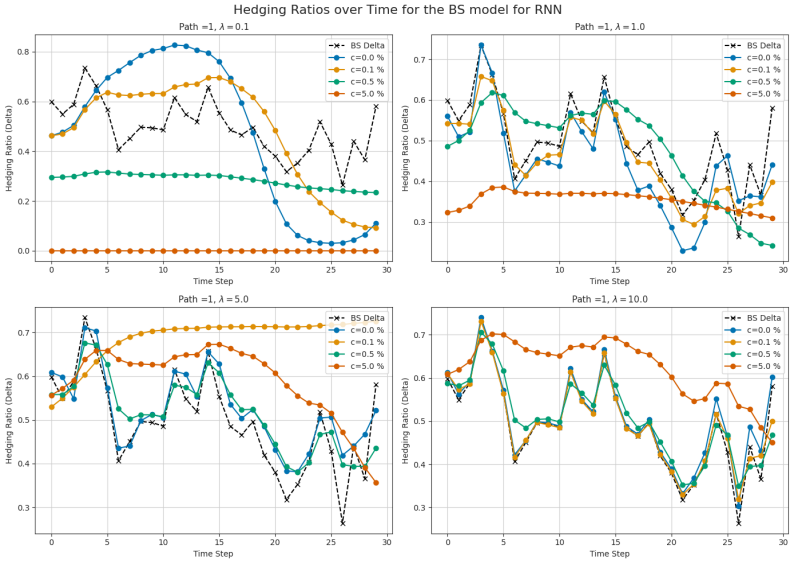


Figure 5.4: The hedging ratios over time for different risk aversions and cost levels for the BS model for LSTM.

In figures 5.5 and 5.6 we provide the PnLs of the different risk aversion with no transaction cost comparing them to the benchmark solution of BS. We observe that except for $\lambda = 0.1$, the neural network's PnLs distributions improve their right tail compared to the BS solution, and in addition the left tail for $\lambda = 5$ and $\lambda = 10$ as we can check in the table 5.1. Therefore, we improved the BS distribution with both NN for high λ.

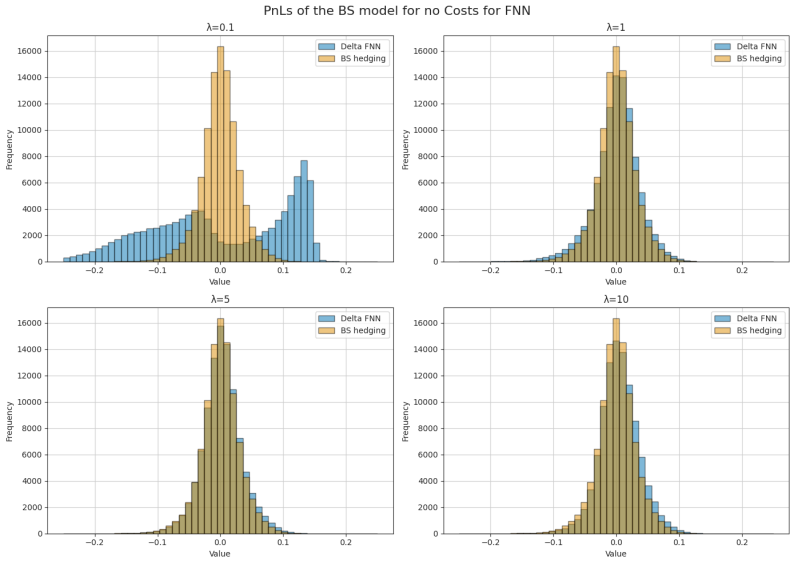


Figure 5.5: The PnLs for the BS model for different risk aversion with no transaction costs for FNN.

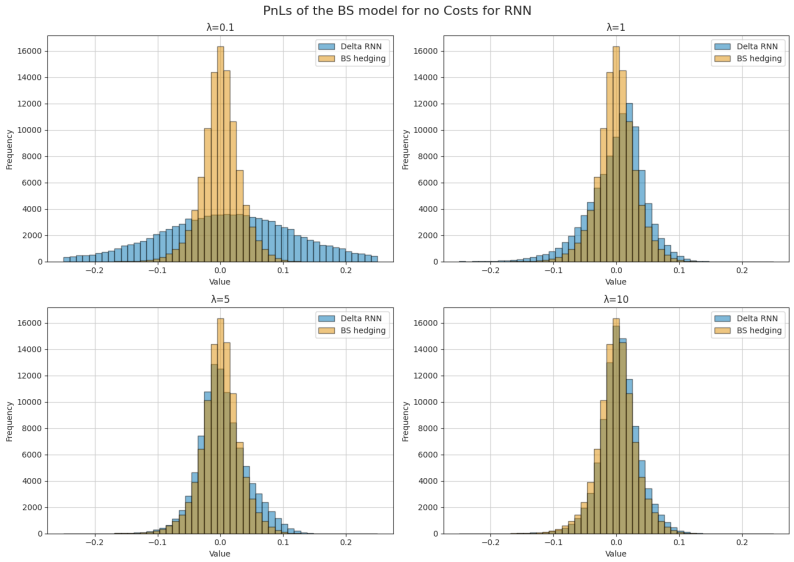


Figure 5.6: The PnLs for the BS model for different risk aversion with no transaction costs for LSTM.

In summery in this section, we have shown the power of the Deep Hedging framework in a well known benchmark. We successfully reproduce the BS pricing and hedging framework with a BS market generator, and extend it to different proportional transaction costs and risk aversions. Also, we note how the table 5.1, and the figures 5.1, 5.3, 5.5, are different faces of the same coin, with those we display enough information to characterised a hedging strategy.

5.2 European Call Option of the SP500

After successfully validating the Deep Hedging framework in the BS model, we proceed to test it in a practical setup with real market data like the SP500 index.

5.2.1 The Neural SDE: Generating SP500 data

We implement the Neural SDE as explained in section 4.2. In the figure 5.7 we show a qualitative plot of 80 paths generated from the Neural SDE next to 80 SP500 data paths. As mentioned before, all the paths are standarized and initialized to 0. In figure 5.8 we compare the marginal distribution of 10^5 generated paths with the real market paths for different time steps $t = 5, 15, 29$. This is the qualitative plot of the table 5.2, which shows the KS average scores and the Type I error for 5% significant level. We note that the marginal distribution of paths at the beginning it is substantially worst that the one at the end. This is due to the market data being quite noisy at the beginning, so it is hard to the Neural SDE to replicated it properly. Although, 8.6% of Type I error at time $t = 3$ is still a good result, the Neural SDE adapts better to the marginals distribution as time advances with 5.3% at time $t = 15$, and at time $t = 27$ achieves a great result with 3.4% of Type I error.

Also, it is worth mentioning that this marginal distribution at final time resembles a couple of the "stylised facts" of the financial markets [11]. Since, it has a more peaky distribution towards 0, and with fatter tails than a Gaussian distribution.

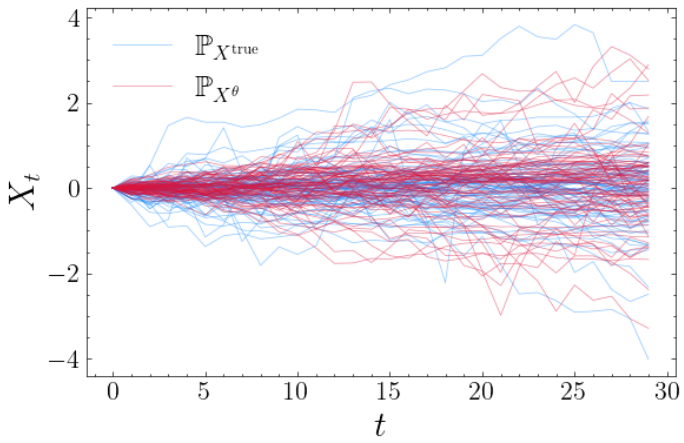


Figure 5.7: Qualitative plot of the paths generated with the paths of real market data.

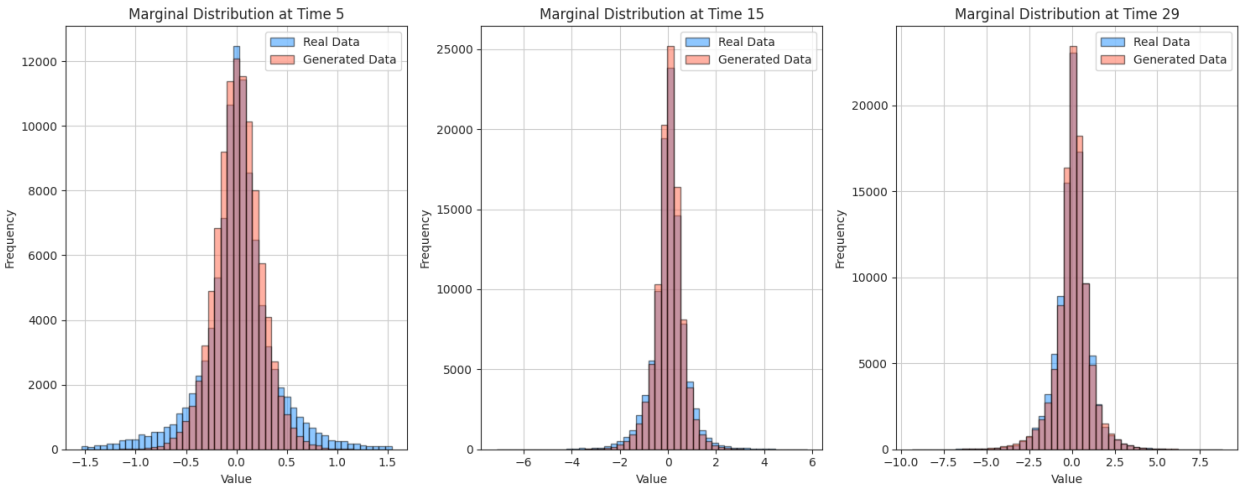


Figure 5.8: The figure compares the marginal distribution of the paths for real and generate data for different times.

$t = 3$		$t = 9$		$t = 15$		$t = 21$		$t = 27$	
KS	% Reject	KS	% Reject	KS	% Reject	KS	% Reject	KS	% Reject
0.1251	**8.6%**	**0.1116**	**5.7%**	**0.1093**	**5.3%**	**0.1080**	**5.2%**	**0.1051**	**3.4%**

Table 5.2: The table shows the different KS average scores and average Type I error for different times along the paths marginal distributions.

In figure 5.9 and table 5.3 we show the autocorrelation scores with a 95% of confidence interval for different lags for the generated and real data. In the table we show up to the 5th lag, and at the figure up to the 14th lag. We get quite similar average autocorrelation scores and confidence intervals all across the different lags. This means that on average the generated data replicates the same autocorrelation behaviour as the real data.

	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$
Generated Data	0.7631 ± 0.2671	0.5751 ± 0.3956	0.4227 ± 0.4537	0.2980 ± 0.4677	0.1949 ± 0.4557
Real Data	0.7573 ± 0.2852	0.5720 ± 0.4115	0.4228 ± 0.4720	0.3015 ± 0.4810	0.2012 ± 0.4664

Table 5.3: The table shows the autocorrelation scores with 95% confidence intervals at different lags.

Finally, in figure 5.10 we present the cross-correlation matrices between the returns process r_t and the lagged squared returns process $r^2_{t-\ell}$ for lags $\ell = \{0, 1, 2, 3, 4, 5\}$. The MSE between both matrices is 0.031003. This is another successful data correlated metric.

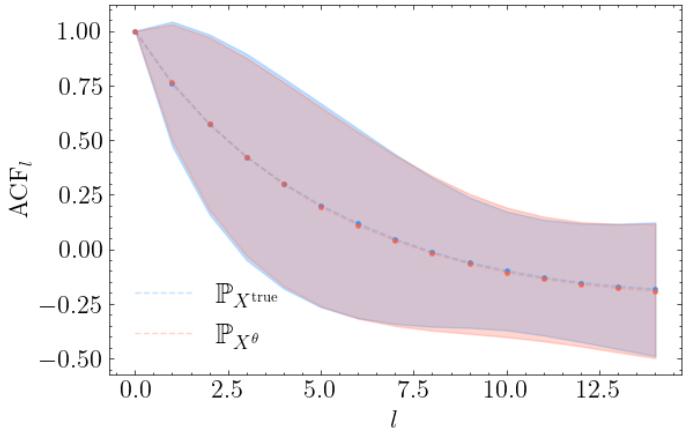


Figure 5.9: Qualitative plot of the autocorrelation scores with 95% interval at different lags.

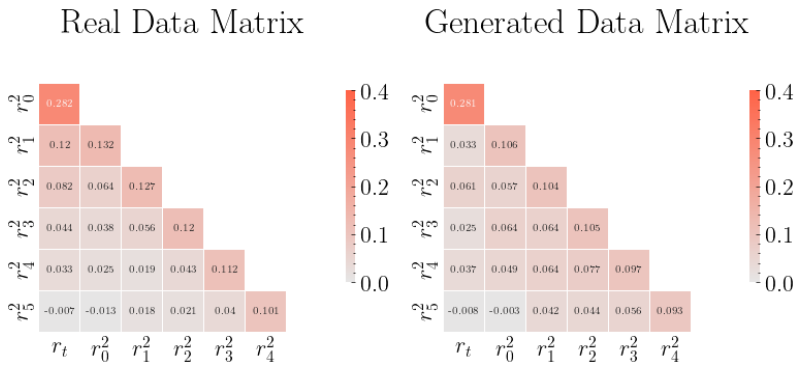


Figure 5.10: The cross-correlation matrices of the generated and real data.

In summery, we successfully achieve a market generator that can replicate up to a high degree of confidence the path's distribution and the correlations metrics of the real market data.

5.2.2 Deep Hedging Model

First, we adapt the data from the Neural SDE implementation. We use as training data for our deep hedging models $N = 10^5$ paths of generated data initialised to $S_0 = 1$ and scaled to a factor of 0.08, so the values of the paths are in between $[0.3506, 1.7017]$. For the validation data of the model we use $N = 10^5$ of the SP500 data paths, also initialised to $S_0 = 1$ and scaled with the same factor of 0.08, so their values are in between $[0.3637, 1.6772]$.

In the previous section 5.1 we have seen that the models with $\lambda = 0.1$ didn't performed that well, so from now on we are using the following set of risk aversion $\lambda \in \{1, 5, 10, 15\}$ and the same transaction costs $k \in \{0\%, 0.05\%, 0.5\%, 5\%\}$. Since now we are not working in a driftless market, we will train the both nueral networks $\boldsymbol{f}$ and $\boldsymbol{f_0}$. We will use the indifference pricing formula 4.1.3. First we will use an European Call option with strike $K = S_0$ as liability before going into more exotic options in the next section. We will be comparing both strategies, Delta Z, where the model takes into account the statistical arbitrage of the market, and Delta Q, where the model disregard the possible statistical arbitrage of market and only focus on hedging the liability. Also, we will study how $\boldsymbol{f_0}$ optimise the optimal statistical arbitrage strategy Delta 0 in the market.

We will compare our results with how we would traditionally address this problem, using a Monte Carlo estimator for the option price, and the Black-Sholes model for developing the hedging strategies.

Parameters		Indif Price	Super-Hedging Ratio		VaR		CVaR	
λ	k	Gen Data	Delta Z	Delta Q	Delta Z	Delta Q	Delta Z	Delta Q
1.0	0.00%	0.0290	**0.7197**	0.6025	-0.0820	**-0.0594**	-0.1343	**-0.0814**
	0.05%	0.0290	**0.7002**	0.6118	-0.0790	**-0.0661**	-0.1304	**-0.1007**
	0.50%	0.0309	**0.6802**	**0.6802**	**-0.0931**	**-0.0931**	**-0.1540**	**-0.1540**
	5.00%	0.0309	**0.6802**	**0.6802**	**-0.0931**	**-0.0931**	**-0.1540**	**-0.1540**
5.0	0.00%	0.0292	**0.6570**	0.6039	-0.0513	**-0.0380**	-0.0697	**-0.0548**
	0.05%	0.0298	**0.6350**	0.6152	-0.0442	**-0.0404**	-0.0617	**-0.0570**
	0.50%	0.0345	**0.6764**	**0.6764**	**-0.0555**	**-0.0555**	**-0.0900**	**-0.0900**
	5.00%	0.0373	**0.7201**	**0.7201**	**-0.0868**	**-0.0868**	**-0.1476**	**-0.1476**
10.0	0.00%	0.0302	**0.6349**	0.6236	-0.0375	**-0.0359**	-0.0524	**-0.0501**
	0.05%	0.0311	**0.6368**	0.6258	-0.0385	**-0.0374**	-0.0533	**-0.0535**
	0.50%	0.0371	**0.6674**	**0.6674**	**-0.0487**	**-0.0487**	**-0.0692**	**-0.0692**
	5.00%	0.0527	**0.7992**	**0.7992**	**-0.0713**	**-0.0713**	**-0.1322**	**-0.1322**
15.0	0.00%	0.0315	**0.6538**	0.6440	-0.0349	**-0.0343**	**-0.0501**	-0.0503
	0.05%	0.0326	**0.6671**	0.6544	-0.0382	**-0.0351**	-0.0540	**-0.0501**
	0.50%	0.0391	**0.6797**	**0.6797**	**-0.0435**	**-0.0435**	**-0.0613**	**-0.0613**
	5.00%	0.0752	**0.8236**	**0.8236**	**-0.0451**	**-0.0451**	**-0.0899**	**-0.0899**
Black-Scholes		**0.0297(MC)**	**0.6333**		**-0.0353**		**-0.0590**	

Table 5.4: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies for generated data.

First, we show how the NNs perform in the data that they were train on at table 5.4. At first glance we observe how the indifference price is quite similar to the MC price throughout the different risk aversion and low cost levels. Comparing both strategies we clearly see that the Delta Z has better super-hedging ratios than the Delta Q or BS. Also we notice that the three strategies have a significant super-hedging ratio which is always above than 60% and even 70%. On the other hand, the Delta Q strategy has better VaR and CVaR than Delta Z. Also, we observe that the strategies are virtually the same as the cost level increases, and as the risk aversion increase, they also become similar.

These result are actually what we would expect. The delta Z takes advantage of the statistical arbitrage while trying to hedge the liability, thus the super-hedging ratio is bigger. However, as we increase the transaction cost in the market the statistical arbitrage in the market disappears, and Delta Z and Q became the same strategy. In addition, when we increase the risk aversion the NNs are less susceptible to the arbitrage due to its possible bigger losses, so both strategies became more similar. We will see these concepts throughout the different figures of the section.

In the figures 5.11, 5.12 and 5.13 we show the hedging ratios over the asset price at time step $t = 20$, for the different risk aversion, cost levels and strategies. We see how the statistical arbitrage influences the Delta Z strategy specially for $\lambda = 1$, and how increasing the risk aversion the hedging ratios tend to match the BS solution. In contrast, for Delta Q the hedging ratios resemble the BS solution across all the different risk aversions. Regarding the strategy Delta 0, we see how the hedging ratios decreases significantly as we increase the risk aversion.

It is worth noticing that when we state that the hedging ratios resemble a BS solution, this doesn't mean to match exactly the same BS solution, but with a similar form, since we are not in a BS market, so the hedging solution shouldn't fully match BS one.

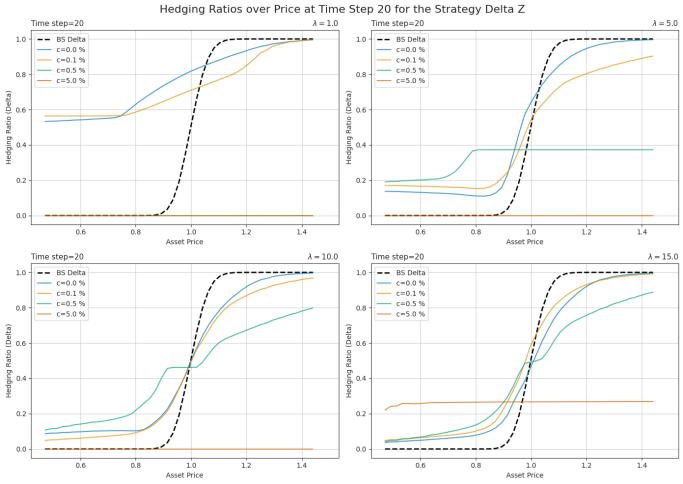


Figure 5.11: The hedging ratios over asset price at time step $t = 20$ following the Delta Z strategy for different risk aversion and cost levels.

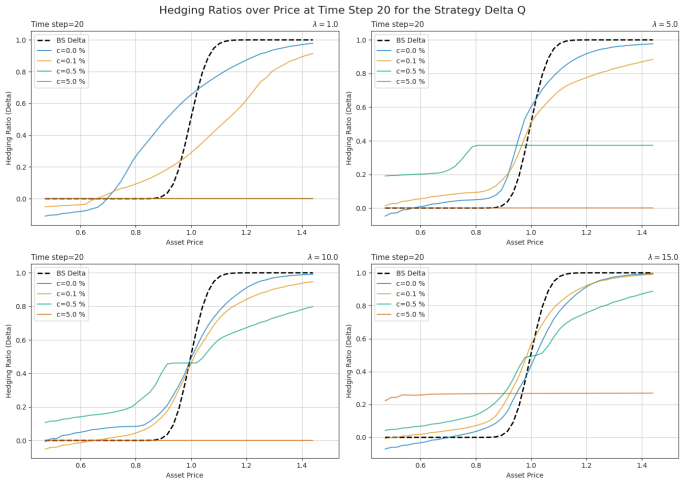


Figure 5.12: The hedging ratios over asset price at time step $t = 20$ following the Delta Q strategy for different risk aversion and cost levels.

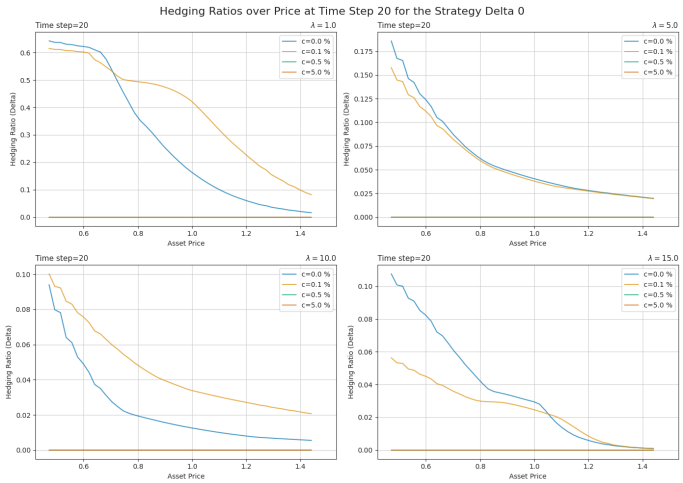


Figure 5.13: The hedging ratios over asset price at time step $t = 20$ following the Delta 0 strategy for different risk aversion and cost levels.

In the figure 5.14 we show how the Delta 0 strategy works for different risk aversions. For $\lambda = 1$ we see that the optimal statistical arbitrage at the beginning is to buy regardless to the asset price with ratio 1, and as the time advances the strategy sells its position depending the price of the asset. The NN $\boldsymbol{f_0}$ learns that in average the stock price will finish around 1, so towards the end of the time path it sells when asset price is high and buys when it is low. For $\lambda = 10$, $\boldsymbol{f_0}$ learns this strategy but is more cautious to bigger loss, so it is more conservative with substantially lower hedging ratios.

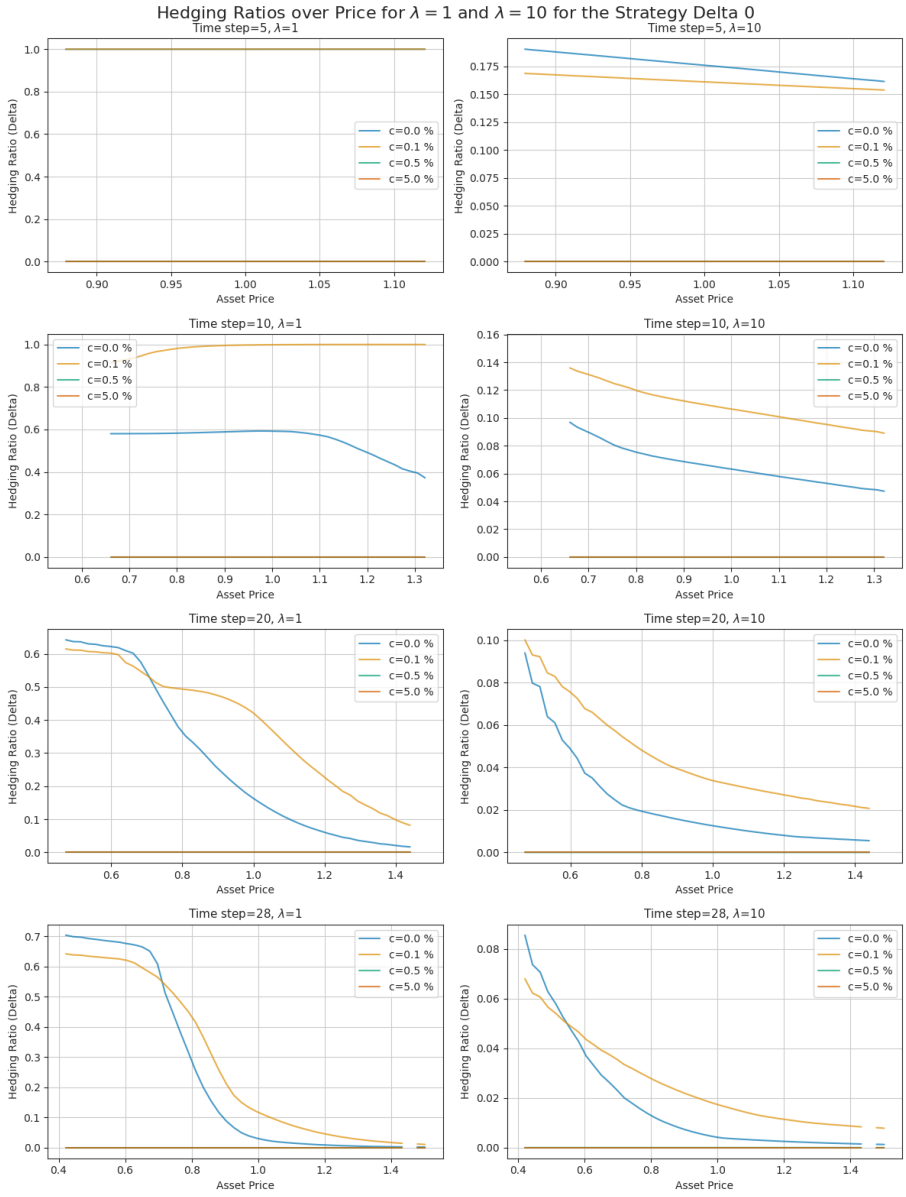


Figure 5.14: We compare the hedging ratio over price plot along different times for the strategy delta 0 between different risk aversions $\lambda = 1$ and $\lambda = 10$ and for different cost levels.

In figures 5.15, and 5.16 we show the hedging ratios over price for different time steps comparing both strategies Delta Z and Q for two different risk aversions. For $\lambda = 1$ the hedging strategy Delta Z gets influence by the statistical arbitrage, and the need to hedge the liability. Therefore, it will combine both strategies, starting buying with ratio 1 like Delta 0. As the time advances the hedging ratio will go toward 1 for high prices of the asset due to hedging, and for low price the hedging strategy will follow a similar policy

to the Delta 0. In the Delta Q we get rid of the statistical influence over Delta Z getting a more alike solution to the BS model. For $\lambda = 10$ Delta Z is barely influence of the statistical arbitrage, thus it resemble the BS solution except for low prices. And Delta Q it is even more similar to the BS model.

Also, it is worth mentioning that we get some small negative values in the Delta Q. This is a consequence of $\boldsymbol{f}$ and $\boldsymbol{f_0}$ not being equally optimise or fully optimise. As we commented before training neural networks is not an exact science.

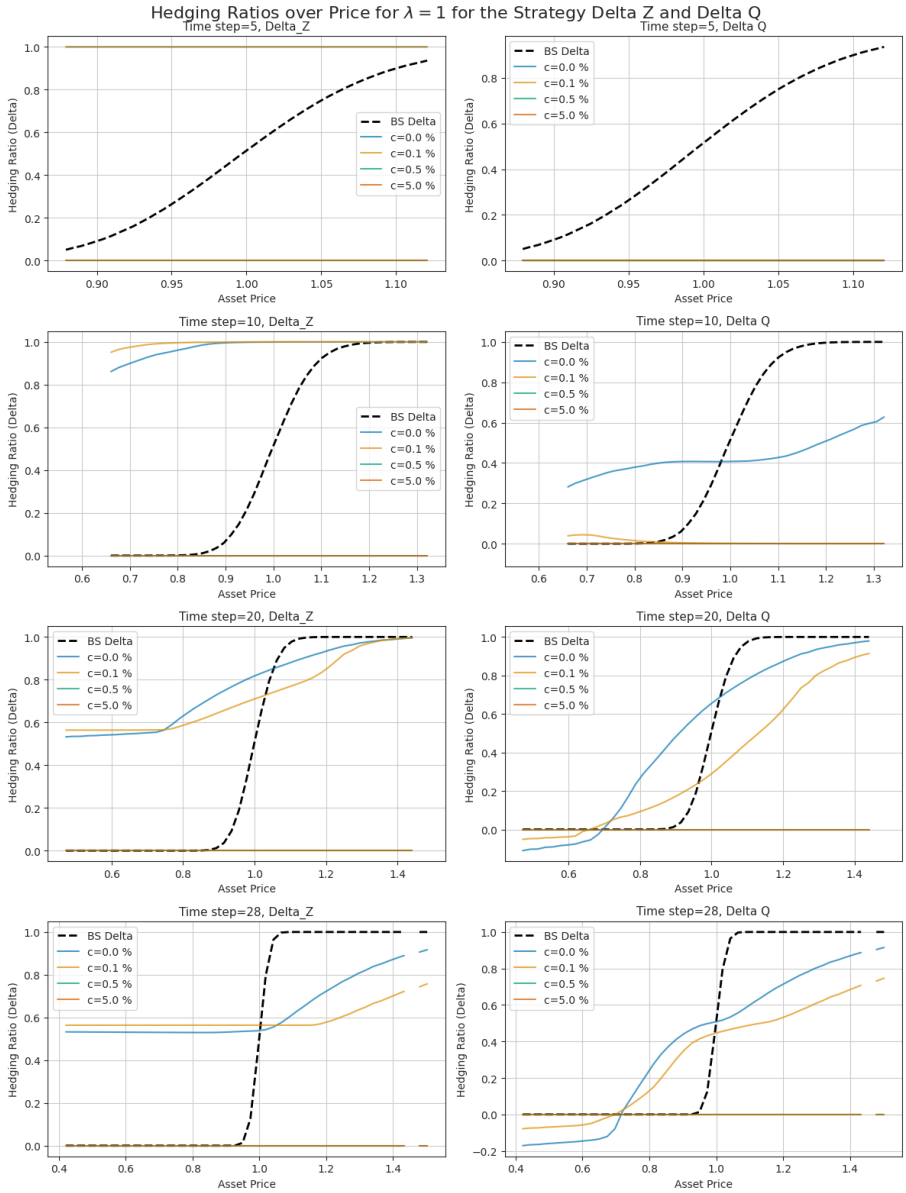


Figure 5.15: We compare the hedging ratio over price plot along different times between both strategies Z and Q for $\lambda = 1$ and for different cost levels.

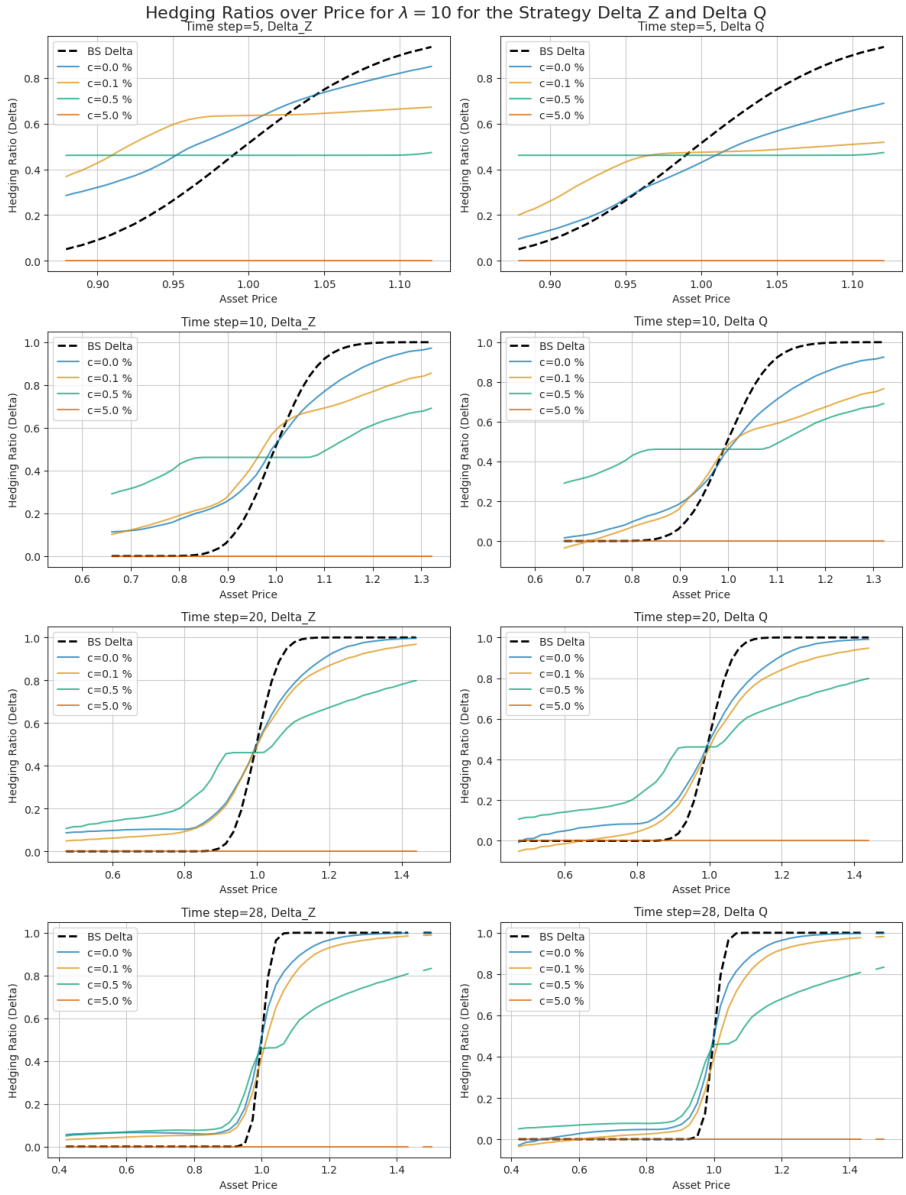


Figure 5.16: We compare the hedging ratio over price plot along different times between both strategies Z and Q for $\lambda = 10$ and for different cost levels.

In the figures 5.17, 5.18 and 5.19 we represent the hedging ratios over time for one real data path over different risk aversion, cost levels and strategies. As mention before, the Delta 0 strategy will start with a high ratio and will progressively sell its position. Therefore, the Delta Z will start also high and progressively will adapt better to the hedging strategy as we increase the risk aversion. The opposite happens for Delta Q where it starts at a low ratio, and progressively will adapt better to the hedging strategy as we increase the risk aversion. Comparing both Deltas Z and Q, we see that Q adapts quicker from the beginning to the hedging strategy.

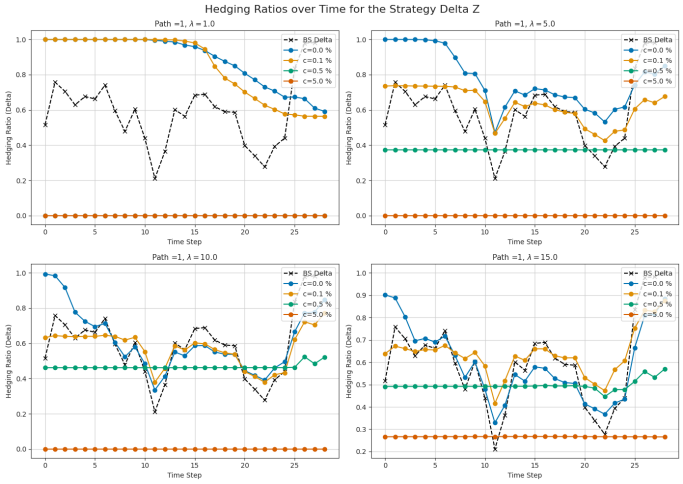


Figure 5.17: The hedging ratios over time for the Delta Z strategy for different risk aversions and cost levels.

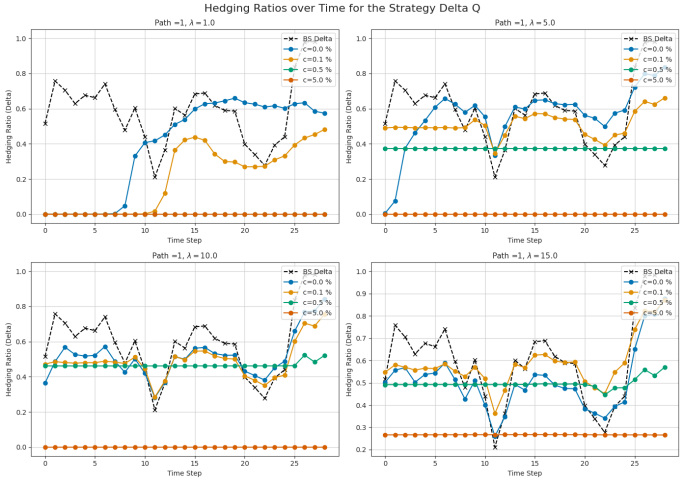


Figure 5.18: The hedging ratios over time for the Delta Q strategy for different risk aversions and cost levels.

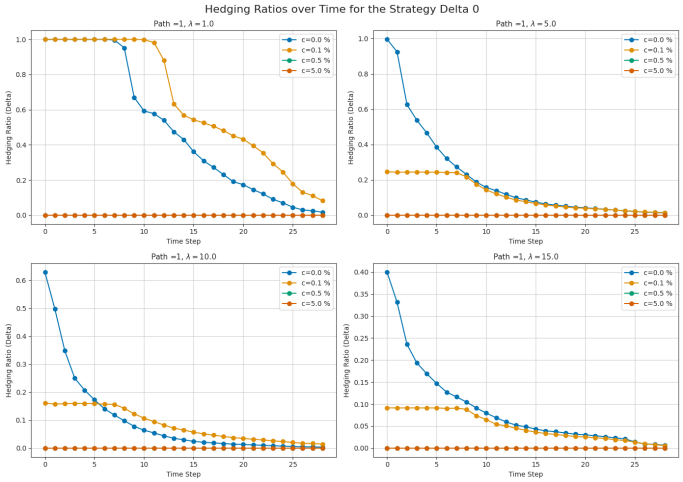


Figure 5.19: The hedging ratios over time for the Delta 0 strategy for different risk aversions and cost levels.

In the figures 5.20, 5.21 and 5.22 we compare the PnLs of the Delta Z and Q between each other and the BS strategy with the generated data. These reflects the results of table 5.4. The Delta Z and Q have better right tail than the BS strategy, and even better left tail for high risk averse levels, as we can check at the table. Also, as we expected both Q and Z tend to be really similar for high risk aversion levels, although Z still more positive than Q.

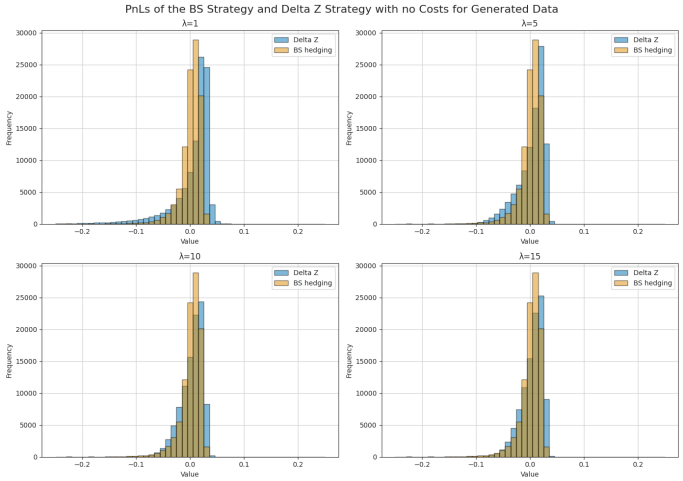


Figure 5.20: The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for generated data.

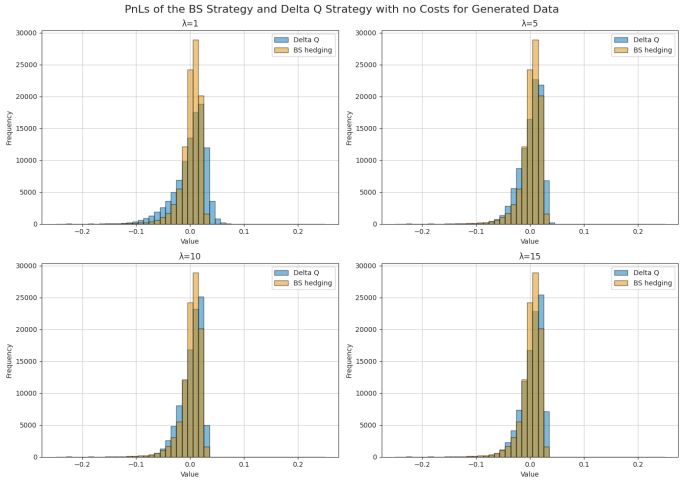


Figure 5.21: The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for generated data.

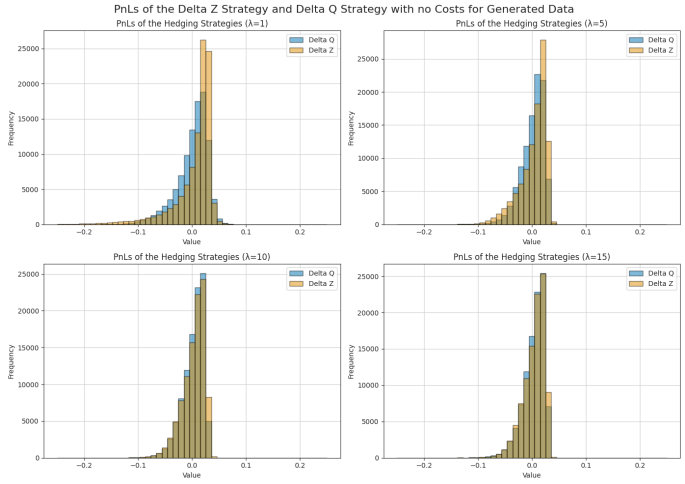


Figure 5.22: The PnLs of the strategy Q and Z for different risk aversions and no transaction costs for generated data.

After showing the results for the generated data, we proceed with the real market data results in the table 5.5. The results with real data for the hedging strategies are quite similar or even better than the results with generated data. Although, we note that the indifference prices of the different risk aversions are slightly more disappeared around the MC price than the generated data. But, this is something we would expect as the NNs weren't fully optimised for this data. This still confirms again how accurate the market generator simulating real market data.

Parameters		**Indif Price**	**Super-Hedging Ratio**		**VaR**		**CVaR**	
λ	k	**Real Data**	**Delta Z**	**Delta Q**	**Delta Z**	**Delta Q**	**Delta Z**	**Delta Q**
1.0	0.00%	0.0284	**0.7178**	0.6071	-0.0850	**-0.0618**	-0.1387	**-0.0906**
	0.05%	0.0283	**0.6975**	0.6298	-0.0844	**-0.0709**	-0.1375	**-0.1144**
	0.50%	0.0306	**0.6792**	**0.6792**	**-0.0896**	**-0.0896**	**-0.1511**	**-0.1511**
	5.00%	0.0306	**0.6792**	**0.6792**	**-0.0896**	**-0.0896**	**-0.1511**	**-0.1511**
5.0	0.00%	0.0286	**0.6754**	0.6044	-0.0597	**-0.0406**	-0.0855	**-0.0619**
	0.05%	0.0298	**0.6415**	0.6155	-0.0460	**-0.0383**	-0.0688	**-0.0603**
	0.50%	0.0345	**0.6691**	**0.6691**	**-0.0518**	**-0.0518**	**-0.0887**	**-0.0887**
	5.00%	0.0368	**0.7144**	**0.7144**	**-0.0834**	**-0.0834**	**-0.1449**	**-0.1449**
10.0	0.00%	0.0306	**0.6593**	0.6341	-0.0423	**-0.0348**	-0.0653	**-0.0567**
	0.05%	0.0311	**0.6442**	0.6333	-0.0390	**-0.0360**	-0.0588	**-0.0583**
	0.50%	0.0372	**0.6660**	**0.6660**	**-0.0462**	**-0.0462**	**-0.0714**	**-0.0714**
	5.00%	0.0515	**0.7911**	**0.7911**	**-0.0687**	**-0.0687**	**-0.1303**	**-0.1303**
15.0	0.00%	0.0325	**0.6859**	0.6691	-0.0370	**-0.0326**	-0.0591	**-0.0546**
	0.05%	0.0330	**0.6813**	0.6661	-0.0381	**-0.0341**	-0.0581	**-0.0537**
	0.50%	0.0394	**0.6868**	**0.6868**	**-0.0417**	**-0.0417**	**-0.0645**	**-0.0645**
	5.00%	0.0742	**0.8181**	**0.8181**	**-0.0432**	**-0.0432**	**-0.0889**	**-0.0889**
Black-Scholes		**0.0294(MC)**	**0.6380**		**-0.0353**		**-0.0590**	

Table 5.5: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real market data.

In figures 5.23, 5.24, and 5.25, we plot the PnLs for the different strategies and risk aversions for the real market data. We get analogous results to the ones with generated data.

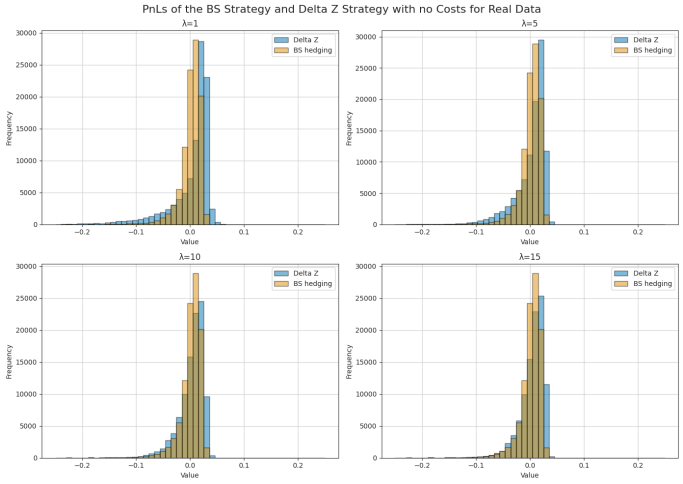


Figure 5.23: The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real market data.

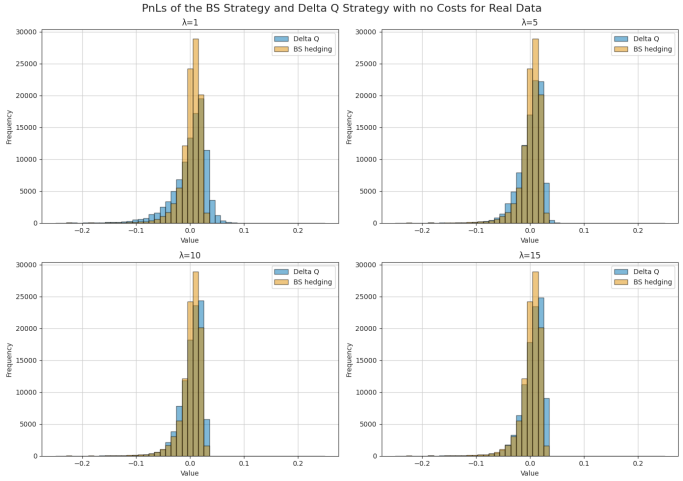


Figure 5.24: The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for real market data

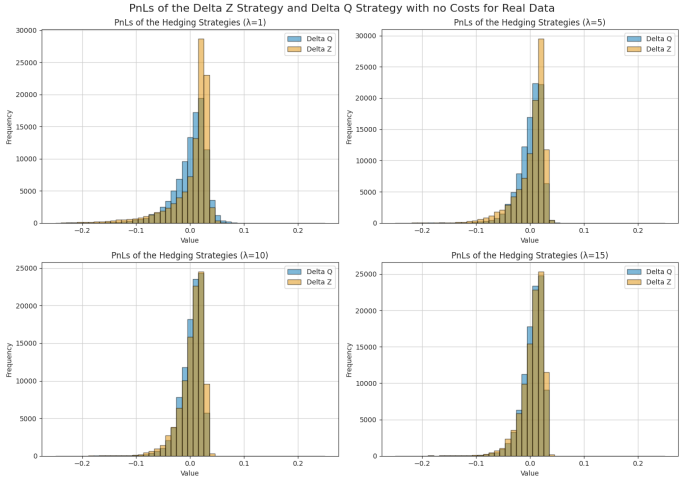


Figure 5.25: The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real market data

In the table 5.6 and figure 5.26, we compare the optimal statistical arbitrage strategy of the generated market in the real data market. The measure of statistical arbitrage $\pi^{\theta}(0)$ it is reduced as we increase the risk aversion, and totally erase when we increase significantly the level costs. Although $\pi^{\theta}(0)$ is optimise for the generated market, it can sense some statistical arbitrage in the real data for $\lambda = 1$. The super-hedging ratios are above 50% for transaction costs lower than 0.1%. The strategy Delta 0 performs a little better in the generated data, as we would expect, the results still quite similar to the real data. Again another metric that corroborates that the generated data is quite close to the real one with similar statistical arbitrage.

Parameters		$\pi^{\theta}(0)$		**Super-Hedging Ratio**		**VaR**	
λ	k	Gen Data	Real Data	Gen Data	Real Data	Gen Data	Real Data
1.0	0.00%	-2.26e-03	-7.64e-04	0.5645	0.5479	-5.89e-02	-7.77e-02
	0.05%	-1.23e-03	3.55e-04	0.5551	0.5358	-7.90e-02	-9.30e-02
	0.50%	4.04e-11	3.85e-11	0.4357	0.4224	-2.32e-10	-3.32e-10
	5.00%	5.82e-11	6.84e-11	0.0000	0.0007	-1.03e-10	-1.42e-10
5.0	0.00%	-7.92e-04	6.16e-04	0.5453	0.5346	-2.40e-02	-3.97e-02
	0.05%	-2.55e-04	9.95e-05	0.5450	0.5303	-1.39e-02	-1.83e-02
	0.50%	4.27e-11	3.86e-11	0.4726	0.4502	-2.69e-10	-3.68e-10
	5.00%	5.21e-11	5.89e-11	0.0000	0.0005	-7.44e-11	-1.06e-10
10.0	0.00%	-3.94e-04	4.40e-04	0.5406	0.5353	-1.22e-02	-2.14e-02
	0.05%	-1.24e-04	1.45e-04	0.5475	0.5326	-9.89e-03	-1.27e-02
	0.50%	4.38e-11	3.02e-11	0.4915	0.4636	-2.57e-10	-3.56e-10
	5.00%	8.50e-11	9.69e-11	0.0000	0.0005	-1.33e-10	-1.86e-10
15.0	0.00%	-2.68e-04	3.24e-04	0.5530	0.5349	-9.67e-03	-1.53e-02
	0.05%	-8.89e-05	5.20e-05	0.5504	0.5346	-5.93e-03	-7.41e-03
	0.50%	5.59e-11	6.03e-11	0.4030	0.3947	-6.50e-10	-6.47e-10
	5.00%	9.97e-11	1.06e-10	0.0000	0.0016	-1.74e-10	-2.04e-10

Table 5.6: Comparison of $\rho^{\theta}(0)$, Super-Hedging Ratios, and VaR between Generated Data and Real Data.

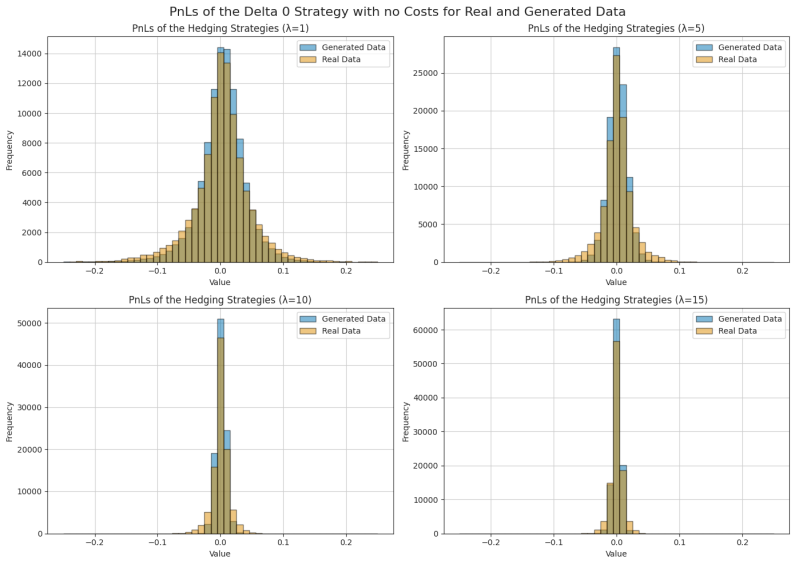


Figure 5.26: The PnLs of the Delta 0 strategy for different risk aversion and no transaction costs for generated and real data.

In summery, in this section we proved that the Neural SDE market generator can be implemented into the deep hedging problem effectively. We studied the market's statistical arbitrage and the strategies to fully take advantage of it; to combine it with hedging a liability; and to disregard it and just focus on hedging the derivative. Also, we note that all the results exposed here are for the FNN. We have compute the results for LSTM too, but there were quite similar to the FNN so we attached them in the appendix A.1.

5.3 Exotic Options

After checking that our framework worked in a simple European Call Option, we will implement it in more exotic products that depend on the full trajectory of the price path, or in multiple assets. We will compare both FNN and LSTM to see if the RNN leverages from its sequential depended setup. In this section we will be more focus on learning the hedging surfaces of the exotic derivatives than their influence by the statistical arbitrage, hence we will be plotting the figures with $\lambda = 10$ to illustrate the proper hedging strategies of the derivatives. Also, when plotting the hedging ratios over time we will be using the Delta Q strategy as we can see more clearly the hedging strategy from the beginning.

5.3.1 Asian Option

First, we test our framework with the same Neural SDE market generator for a Arithmetic Asian call option with fixed strike $K = S_0$. Its payoff at maturity time T=30 is:

$$Z_T = \max(A(0,T) - K, 0)$$

where $A(0,T) = \frac{1}{T}\sum_{i=0}^{T} S_i$ is the arithmetic average of the asset price from $t = 0$ to $t = T$. This derivative doesn't even have an analytical price formula in BS. The usual alternatives are numerical methods like lattice methods or finite difference methods. Here we simply use a Monte Carlo estimator as price reference.

In this setup the input of the NNs $\boldsymbol{f}$ and $\boldsymbol{f_0}$ will have an extra label $A(0,t)$ the running arithmetic average of the path, hence the NN will have all the information needed up to time t to price and hedge the derivative.

The deep hedging model allows us to study with ease the hedging surface of path dependant derivatives. In the figure 5.27 we present the hedging surface predicted by the neural network $\boldsymbol{f}$ after being trained with the generated data. At the beginning, the hedging surface resembles a European call option. As time advances it evolves as the NN learns how to adapt the hedging ratios depending the spot prices, the running average and the time left. It is worth mention that there are some regions on the surface that the NN extrapolates since there weren't enough market data on those region, like for example the region of high running average with low spot prices.

In figure 5.28 we represent the hedging surface next to its 2D representation implemented to the real market data. We note how to important is the 3D representation to be able to understand the hedging strategy as a whole picture.

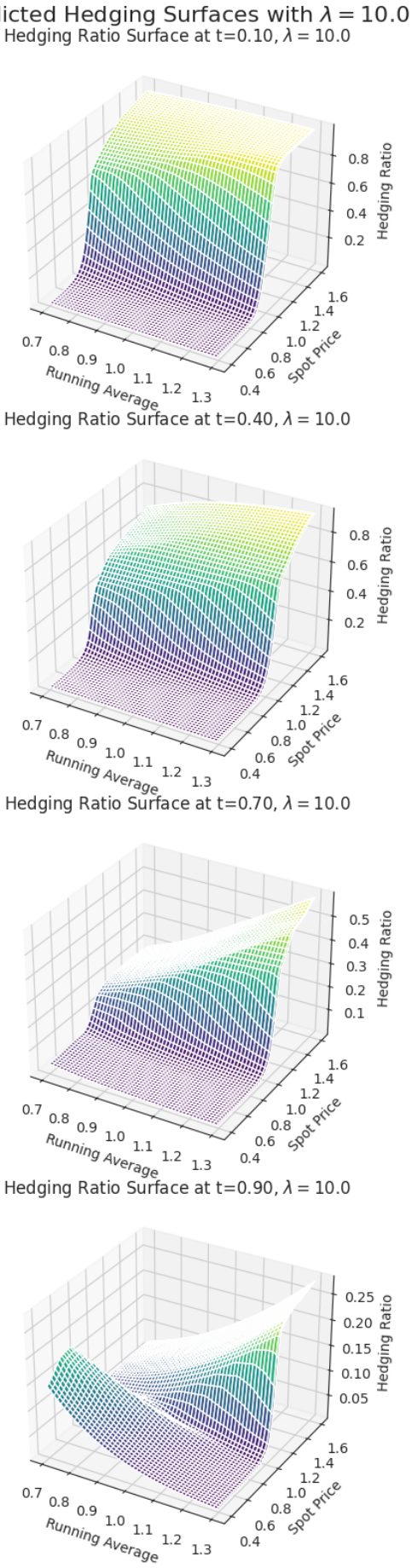


Figure 5.27: The predicted hedging surface for a Asian Option by $\boldsymbol{f}$ for $\lambda = 10$ and no transaction costs.

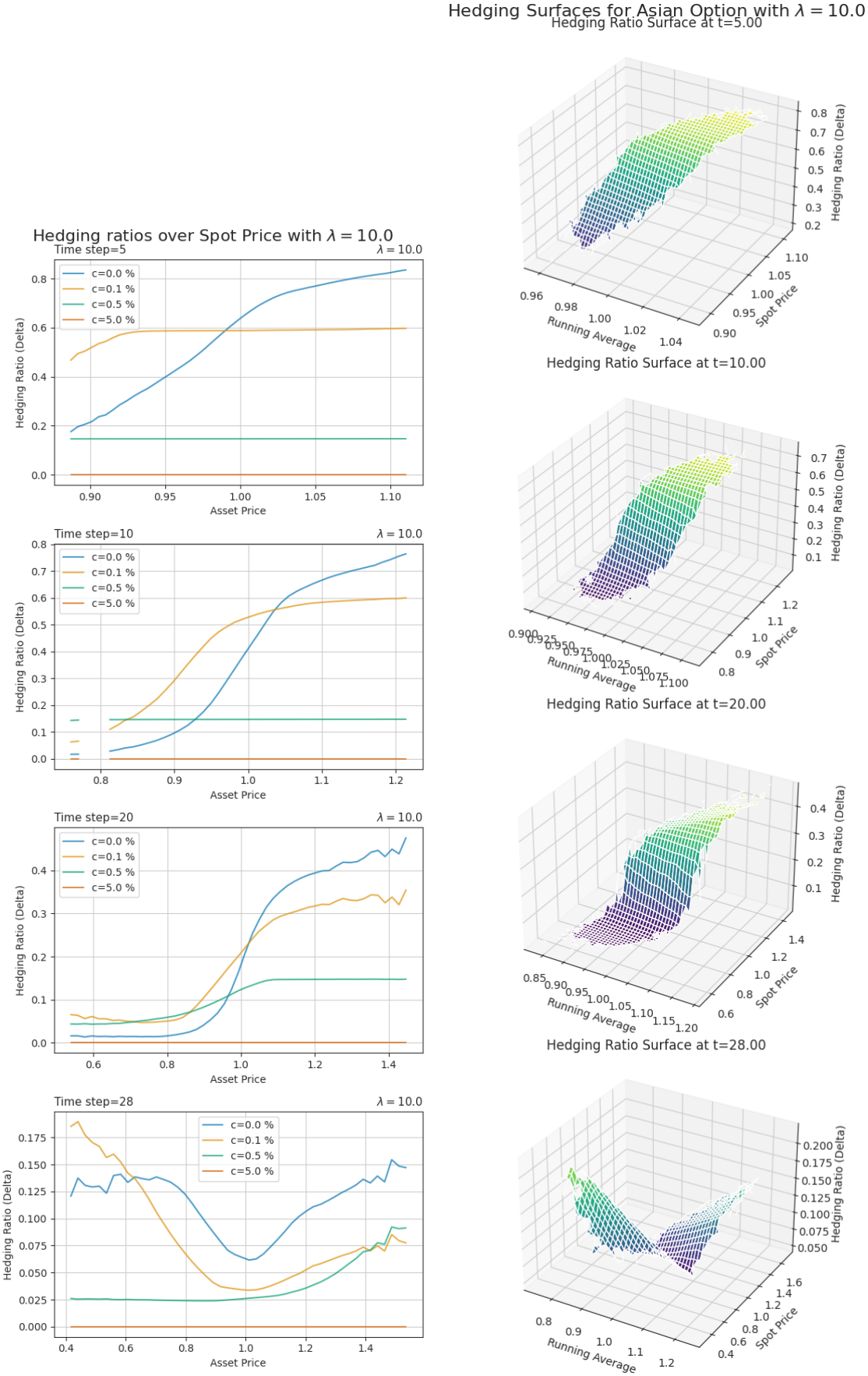


Figure 5.28: The hedging surface next to its 2D representation of the hedging ratios with respect the asset price for the real market data for a Asian Option.

In the figures 5.29 and 5.30 we plot the hedging ratios over time for a path in the real market dataset using FNN and LSTM respectively. We note that the hedging ratios over time decreases for this type of derivative, not only this particular path. We can also observe it in the hedging surfaces in figure 5.27. In fact, we notice that the hedging ratios are smaller compare to other options. This is due to the smaller payoff of this option compared to other options.

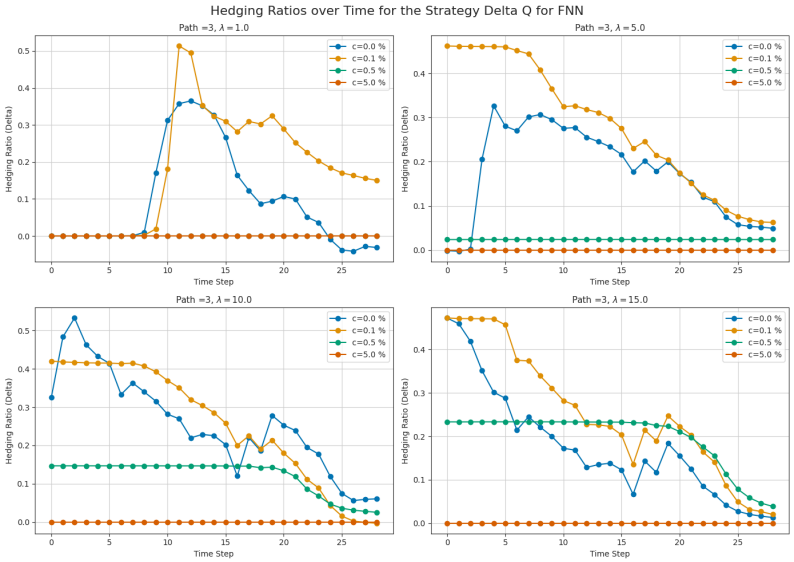


Figure 5.29: The hedging ratios over time for the strategy Delta Q for path 3 in the real market dataset for Asian Call using FNN.

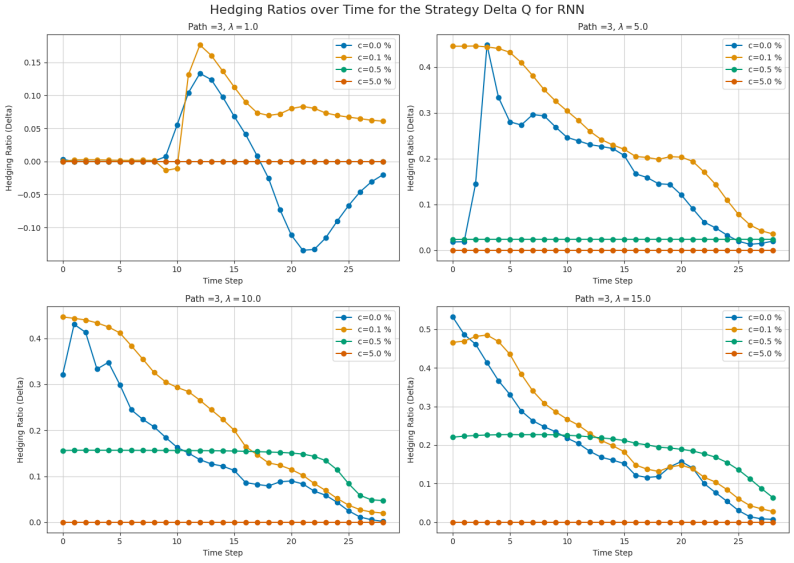


Figure 5.30: The hedging ratios over time for the strategy Delta Q for path 3 in the real market dataset for Asian Call using LSTM.

In the tables 5.7 and 5.8 we compare our results for real market data between the FNN and LSTM for Delta Z strategy and Delta Q respectively. We got super-hedging ratios above the 60% for both strategies and NNs, and even 70% for some Delta Z. As expected, we got higher super-hedging ratios for Delta Z, but grater VaR and CVaR for Delta Q. We didn't get any significant different between FNN and LSTM, moreover we get really

similar results for high cost levels. We plot this results in the figures 5.31, and 5.32 where we visually compare the PnLs of the Delta Z and Q strategies for different risk aversion for FNN and LSTM respectively. We also computed the results for the generated data but we got quite similar results to the validation data, thus we present them in the appendix A.2.

Parameters		**Indif Price**		**Super-Hedging Ratio**		**VaR**		**CVaR**	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	**0.0162**	**0.0162**	0.6882	**0.7096**	-0.0820	**-0.0719**	-0.1342	**-0.1193**
	0.05%	0.0162	**0.0161**	0.6854	**0.7034**	-0.0810	**-0.0711**	-0.1312	**-0.1167**
	0.50%	**0.0172**	**0.0172**	**0.6849**	**0.6849**	**-0.0544**	**-0.0544**	**-0.0886**	**-0.0886**
	5.00%	**0.0172**	**0.0172**	**0.6849**	**0.6849**	**-0.0544**	**-0.0544**	**-0.0886**	**-0.0886**
5.0	0.00%	**0.0157**	0.0158	**0.7110**	0.7047	-0.0494	**-0.0476**	-0.0755	**-0.0730**
	0.05%	0.0170	**0.0169**	**0.7206**	0.7198	-0.0420	**-0.0394**	-0.0663	**-0.0623**
	0.50%	**0.0190**	**0.0190**	**0.7078**	**0.7078**	**-0.0504**	**-0.0504**	**-0.0833**	**-0.0833**
	5.00%	**0.0191**	**0.0191**	**0.7032**	**0.7032**	**-0.0526**	**-0.0526**	**-0.0867**	**-0.0867**
10.0	0.00%	0.0172	**0.0166**	**0.7020**	0.6896	-0.0364	**-0.0358**	-0.0557	**-0.0548**
	0.05%	0.0175	**0.0174**	**0.6955**	0.6645	-0.0348	**-0.0286**	-0.0515	**-0.0439**
	0.50%	**0.0207**	**0.0207**	**0.7197**	0.7113	**-0.0376**	**-0.0374**	**-0.0646**	-0.0632
	5.00%	**0.0223**	**0.0223**	**0.7322**	**0.7322**	**-0.0494**	**-0.0494**	**-0.0836**	**-0.0836**
15.0	0.00%	0.0180	**0.0179**	0.6801	**0.7022**	-0.0272	**-0.0272**	-0.0436	**-0.0436**
	0.05%	0.0180	**0.0179**	0.6736	**0.6793**	-0.0262	**-0.0261**	-0.0382	**-0.0393**
	0.50%	**0.0219**	**0.0218**	**0.7076**	0.7076	**-0.0309**	-0.0311	**-0.0527**	-0.0523
	5.00%	**0.0272**	**0.0272**	**0.7738**	**0.7738**	**-0.0444**	**-0.0444**	**-0.0786**	**-0.0786**
Monte Carlo		**0.0162**							

Table 5.7: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Z, for Real data for the Asian Option.

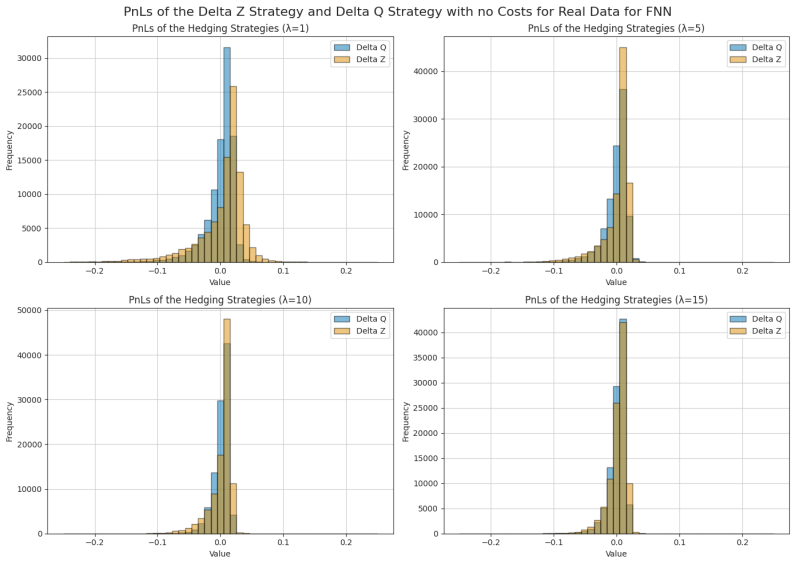


Figure 5.31: The PnLs for the Delta Z and Q strategies for the Asian option using real data for FNN.

Parameters		Indif Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	**0.0162**	**0.0162**	0.6345	**0.6506**	**-0.0472**	-0.0538	**-0.0756**	-0.0865
	0.05%	**0.0162**	0.0161	0.6022	**0.6648**	-0.0483	**-0.0504**	-0.0762	**-0.0812**
	0.50%	**0.0172**	**0.0172**	**0.6849**	**0.6849**	**-0.0544**	**-0.0544**	**-0.0886**	**-0.0886**
	5.00%	**0.0172**	**0.0172**	**0.6849**	**0.6849**	**-0.0544**	**-0.0544**	**-0.0886**	**-0.0886**
5.0	0.00%	0.0157	**0.0158**	**0.6093**	0.6135	**-0.0362**	-0.0343	**-0.0558**	-0.0530
	0.05%	0.0170	**0.0169**	0.6429	**0.6333**	-0.0290	**-0.0275**	-0.0421	**-0.0401**
	0.50%	**0.0190**	**0.0190**	**0.7078**	**0.7078**	**-0.0504**	**-0.0504**	**-0.0833**	**-0.0833**
	5.00%	**0.0191**	**0.0191**	**0.7032**	**0.7032**	**-0.0526**	**-0.0526**	**-0.0867**	**-0.0867**
10.0	0.00%	0.0172	**0.0166**	**0.6425**	0.6286	-0.0232	**-0.0256**	-0.0375	**-0.0409**
	0.05%	0.0175	**0.0174**	0.6467	**0.6342**	-0.0273	**-0.0255**	-0.0396	**-0.0366**
	0.50%	**0.0207**	**0.0207**	**0.7197**	0.7113	**-0.0376**	**-0.0374**	**-0.0646**	-0.0632
	5.00%	**0.0223**	**0.0223**	**0.7322**	**0.7322**	**-0.0494**	**-0.0494**	**-0.0836**	**-0.0836**
15.0	0.00%	0.0180	**0.0179**	**0.6580**	0.6541	-0.0229	**-0.0220**	-0.0385	**-0.0359**
	0.05%	0.0180	**0.0179**	**0.6491**	0.6498	**-0.0232**	-0.0230	**-0.0352**	-0.0345
	0.50%	**0.0219**	**0.0218**	**0.7067**	**0.7076**	**-0.0309**	-0.0311	**-0.0527**	-0.0523
	5.00%	**0.0272**	**0.0272**	**0.7738**	**0.7738**	**-0.0444**	**-0.0444**	**-0.0786**	**-0.0786**
Monte Carlo		**0.0162**							

Table 5.8: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Q, for Real data for the Asian Option.

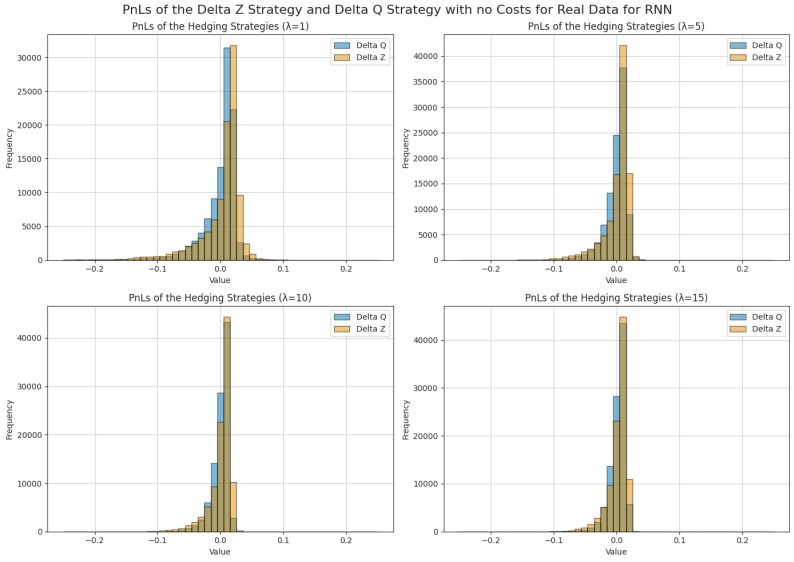


Figure 5.32: The PnLs for the Delta Z and Q strategies for the Asian option using real data for LSTM.

5.3.2 Lookback Option

Now, we test our framework for a Lookback Call Option with floating strike. Its payoff at maturity $T = 30$ is:

$$Z_T = \max(S_T - S_{\min,T}, 0)$$

where $S_{\min,T} = \min_{0 \le \tau \le T} S_\tau$ is the running minimum of the price path. In this setup the input of the NN $\boldsymbol{f}$ and $\boldsymbol{f_0}$ will have an extra label $S_{\min,t}$ the running minimum value of the path.

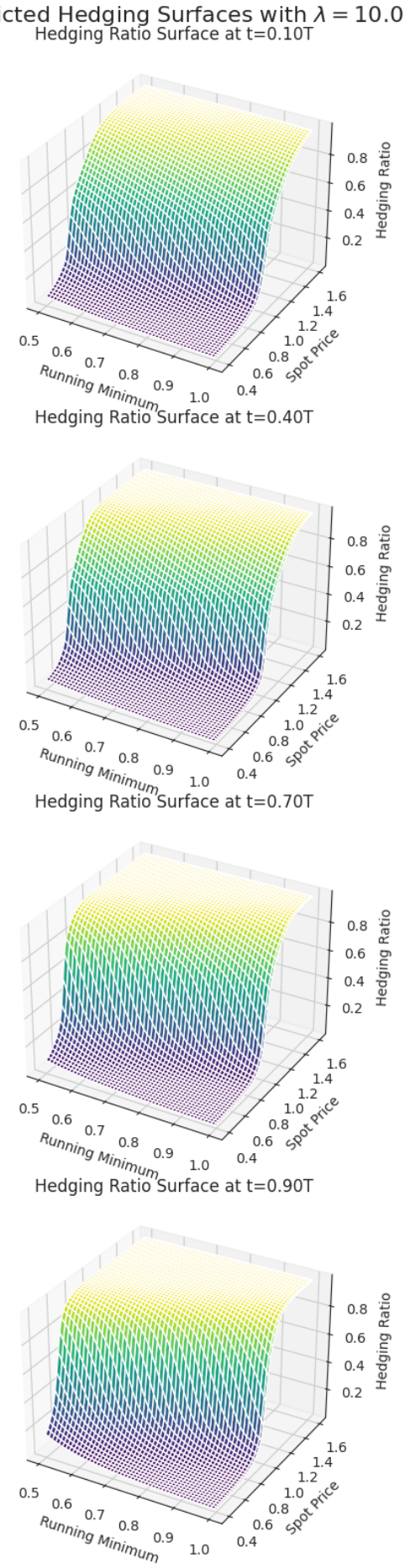


Figure 5.33: The predicted hedging surface for a Lookback Option by $\boldsymbol{f}$ for $\lambda = 10$ and no transaction costs.

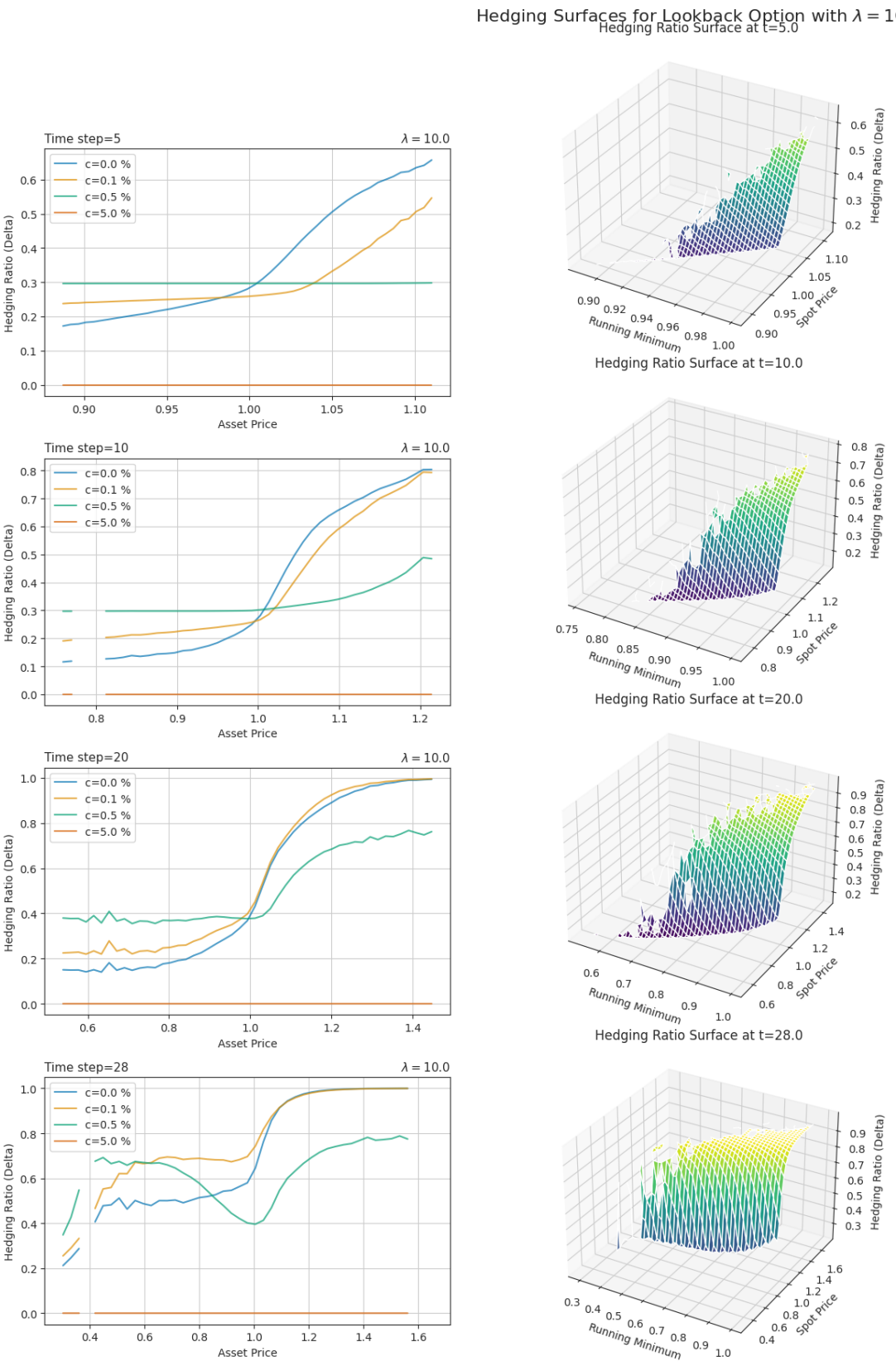


Figure 5.34: The hedging surface next to its 2D representation of the hedging ratios with respect the asset price for the real market data for a LookBack Option.

In the figure 5.33 we illustrate the predicted hedging surface by the FNN. The surface

resembles to the European call, however its slope it is diagonal in the plane, which makes sense as the NN learned to hedge the liability by the difference between the spot price and the running minimum. In figure 5.34, we show the hedging ratio surface applied to the real data next to its 2D representation.

We note that the hedging ratio doesn't go to 0 although the value is low. This is not because of statistical arbitrage since we are using $\lambda = 10$, this is a consequence of the payoff not being 0 most of the times, hence the NN learns to correctly hedge the liability even in low prices. Also, We note that hedging the ratios start low since the NN doesn't know if the asset price will increase or not, once the price increases the hedging ratio will increase too. However, if the prices decreases approaching the minimum the ratios will decrease too. We show an example of this concept in the figures 5.35 and 5.36 where we plot the hedging ratios over time for one particular for FNN and LSTM, respectively.

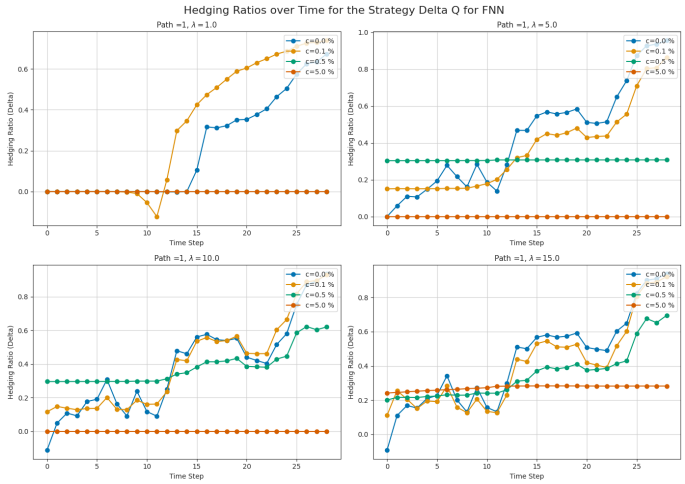


Figure 5.35: Hedging ratios over time for Lookback option using FNN.

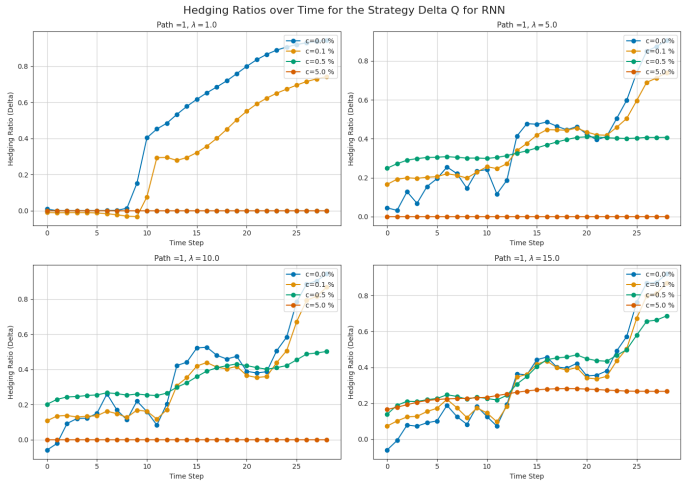


Figure 5.36: Hedging ratios over time for Lookback option using FNN.

We present the results for this option for the validation data in the tables 5.9 and 5.10 for the Delta Z and Q strategies respectively. In both tables we compare the FNN and LSTM, and as for the Asian option, we didn't find any sensible difference between both NN. Regarding the two strategies we got the same results as with previous options. And again we got a great super-hedging ratio over 60% across all the different strategies, neural networks, risk aversion levels and cost levels. The PnLs of the strategies for the

different NN are plotted in the figures 5.37 and 5.38. And again we show the results of the generated data at the appendix A.3, as there were quite similar to the real ones.

Parameters		**Indif Price**		**Super-Hedging Ratio**		**VaR**		**CVaR**	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	0.0476	**0.0480**	**0.6537**	0.6657	**-0.1042**	-0.1108	**-0.1629**	-0.1758
	0.05%	0.0479	**0.0479**	0.6484	**0.6509**	**-0.1022**	-0.1025	**-0.1564**	-0.1596
	0.50%	0.0501	**0.0501**	**0.6521**	**0.6521**	**-0.0996**	**-0.0996**	**-0.1652**	**-0.1652**
	5.00%	0.0501	**0.0501**	**0.6521**	**0.6521**	**-0.0996**	**-0.0996**	**-0.1652**	**-0.1652**
5.0	0.00%	**0.0505**	0.0512	**0.6534**	0.6567	**-0.0713**	-0.0715	**-0.1129**	-0.1161
	0.05%	0.0510	**0.0513**	0.6402	**0.6468**	**-0.0629**	-0.0628	-0.0994	**-0.1017**
	0.50%	**0.0556**	0.0554	**0.6729**	0.6587	**-0.0726**	-0.0715	**-0.1196**	-0.1107
	5.00%	0.0580	**0.0580**	**0.7037**	**0.7037**	**-0.0917**	**-0.0917**	**-0.1573**	**-0.1573**
10.0	0.00%	**0.0548**	0.0557	0.6918	**0.7025**	**-0.0574**	-0.0568	**-0.0938**	-0.0969
	0.05%	0.0552	**0.0563**	0.6831	**0.6986**	**-0.0546**	-0.0558	**-0.0896**	-0.0937
	0.50%	**0.0614**	0.0606	**0.6919**	0.6977	**-0.0628**	-0.0643	**-0.1006**	-0.1011
	5.00%	**0.0787**	**0.0787**	**0.8078**	**0.8078**	**-0.0710**	**-0.0710**	**-0.1366**	**-0.1366**
15.0	0.00%	**0.0603**	0.0623	0.7477	**0.7581**	-0.0485	**-0.0492**	-0.0851	**-0.0870**
	0.05%	**0.0612**	0.0629	0.7405	**0.7580**	**-0.0486**	-0.0500	**-0.0837**	-0.0863
	0.50%	**0.0687**	0.0677	0.7552	**0.7482**	-0.0556	**-0.0542**	-0.0933	**-0.0907**
	5.00%	**0.1076**	0.1069	**0.8445**	0.8426	**-0.0478**	-0.0500	**-0.0971**	-0.0997
Monte Carlo		**0.0495**							

Table 5.9: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Z for a Lookback option .

Parameters		**Indif Price**		**Super-Hedging Ratio**		**VaR**		**CVaR**	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	0.0476	**0.0480**	**0.6345**	0.6450	-0.0782	**-0.0736**	-0.1289	**-0.1169**
	0.05%	0.0479	**0.0479**	0.6246	**0.6275**	-0.0728	**-0.0712**	-0.1188	**-0.1179**
	0.50%	0.0501	**0.0501**	**0.6521**	**0.6521**	**-0.0996**	**-0.0996**	**-0.1652**	**-0.1652**
	5.00%	0.0501	**0.0501**	**0.6521**	**0.6521**	**-0.0996**	**-0.0996**	**-0.1652**	**-0.1652**
5.0	0.00%	0.0505	**0.0512**	0.6502	**0.6585**	**-0.0580**	-0.0603	**-0.0957**	-0.1021
	0.05%	0.0510	**0.0513**	0.6426	**0.6501**	**-0.0624**	-0.0618	-0.0999	**-0.1022**
	0.50%	**0.0556**	0.0554	**0.6729**	0.6587	**-0.0726**	-0.0715	**-0.1196**	-0.1107
	5.00%	0.0580	**0.0580**	**0.7037**	**0.7037**	**-0.0917**	**-0.0917**	**-0.1573**	**-0.1573**
10.0	0.00%	**0.0548**	0.0557	0.6858	**0.7037**	**-0.0547**	-0.0549	**-0.0911**	-0.0947
	0.05%	0.0552	**0.0563**	0.6834	**0.7009**	**-0.0552**	-0.0569	**-0.0910**	-0.0977
	0.50%	**0.0614**	0.0606	**0.6919**	0.6977	**-0.0628**	-0.0643	**-0.1006**	-0.1011
	5.00%	**0.0787**	**0.0787**	**0.8078**	**0.8078**	**-0.0710**	**-0.0710**	**-0.1366**	**-0.1366**
15.0	0.00%	**0.0603**	0.0623	0.7442	**0.7563**	**-0.0473**	-0.0507	**-0.0840**	-0.0891
	0.05%	**0.0612**	0.0630	0.7380	**0.7569**	**-0.0494**	-0.0511	**-0.0851**	-0.0891
	0.50%	**0.0687**	0.0677	0.7552	**0.7482**	-0.0556	**-0.0542**	-0.0933	**-0.0907**
	5.00%	**0.1076**	0.1069	**0.8445**	0.8426	**-0.0478**	-0.0500	**-0.0971**	-0.0997
Monte Carlo		**0.0495**							

Table 5.10: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Q, for a Lookback option.

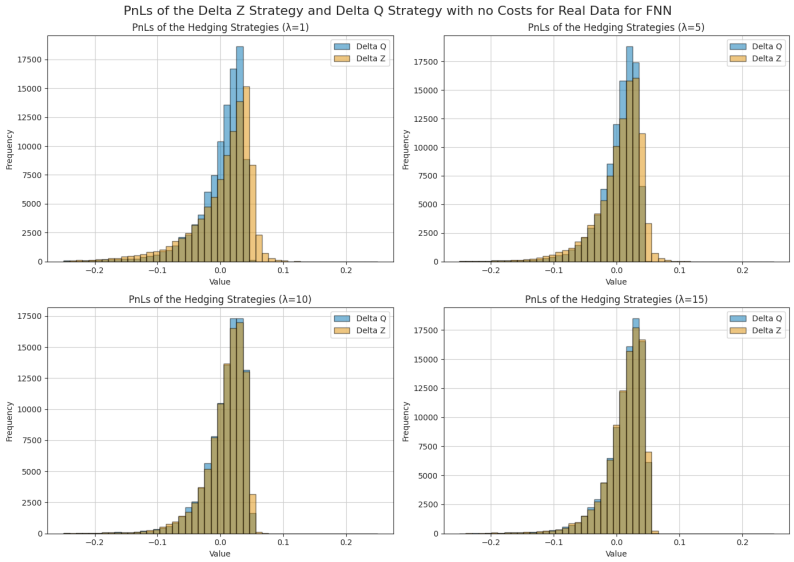


Figure 5.37: The PnLs for the Delta Z and Q strategies for the Lookback option using real data for FNN

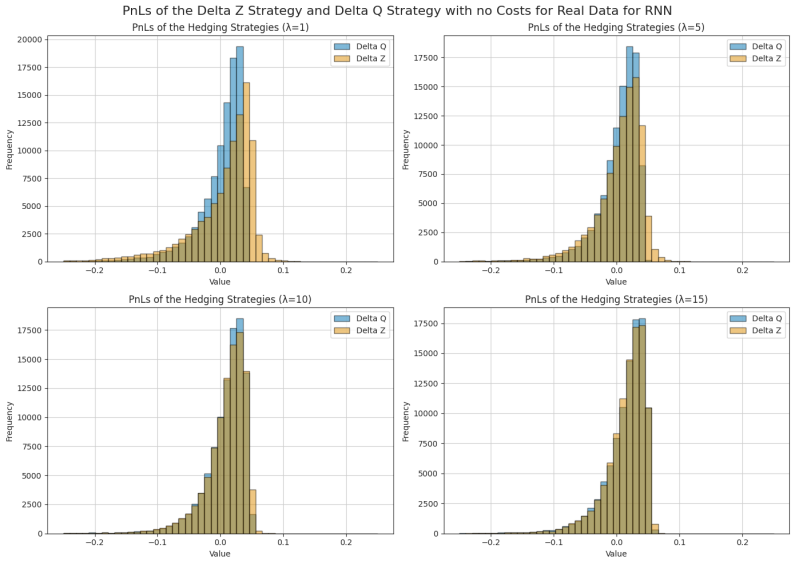


Figure 5.38: The PnLs for the Delta Z and Q strategies for the Lookback option using real data for LSTM

In summery, in these two subsection we have shown how the deep hedging model properly builds the hedging strategies for path-dependent derivatives like Asian Options or Lookback options. Also, we note that there wasn't significant different between using FNN or LSTM. This is due to the fact that we complemented our inputs with labels that could describe the payoff derivative at all time, thus there was no need to use the LSTM path-dependent architecture. Although, practically it was useful to have another solution to compare to ensure that neural networks were fully optimise.

5.3.3 Rainbow Option

In this last subsection, we will implement our framework to a multivariate problem like a Rainbow Option, where the payoff depends on more than one asset. More concretely we

will price and hedge a basket option, a type of Rainbow option which payoff is based on the overall performance of all assets in the basket. Its payoff at maturity time T for two assets S_1 and S_2 is

$$Z_T = \max(\frac{1}{2}(S_{1,T} + S_{2,T}) - K, 0).$$

As underlyings we chose the hourly SP500 data, as previously, and the hourly NASDAQ[1] data from 01/04/2007 to 24/03/2021.

The Neural SDE results

We followed the same implementation explained in section 4.2, but now with an extra dimension representing the other price path. In figure 5.39 we show a qualitative plot of the generated and real paths for both indices.

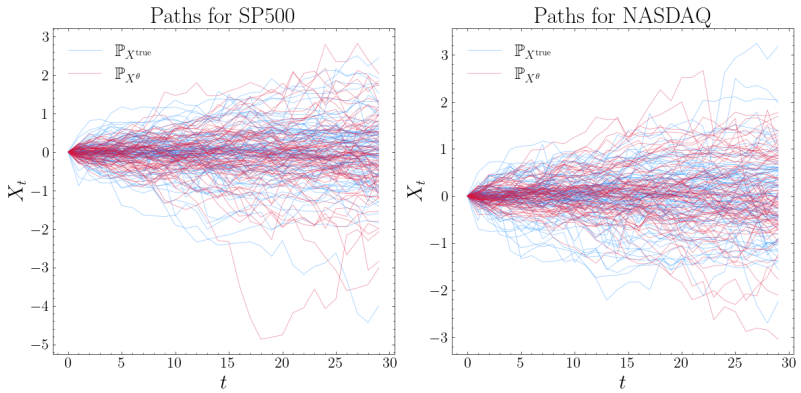


Figure 5.39: Qualitative plots of the paths generated with the paths of real market data for both indices.

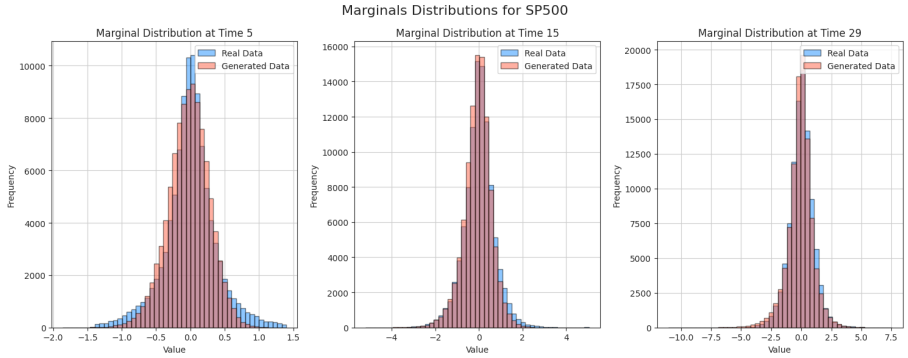


Figure 5.40: The figure compares the marginal distribution of the paths for real and generate data for different times for SP500 data.

[1]The NASDAQ is a stock market index that tracks the performance of 100 of the largest non-financial companies listed on the NASDAQ stock exchange, with a strong focus on technology and innovative industries.

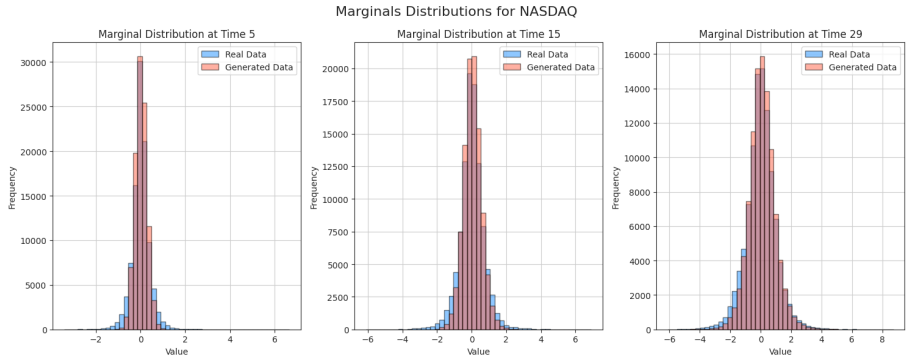


Figure 5.41: The figure compares the marginal distribution of the paths for real and generate data for different times for NASDAQ data.

	$t = 3$		$t = 9$		$t = 15$		$t = 21$		$t = 27$	
	KS	% Reject	KS	% Reject	KS	% Reject	KS	% Reject	KS	% Reject
SP500	**0.1421**	**25.3%**	**0.1363**	**23.3%**	**0.1279**	**15.7%**	**0.1228**	**11.9%**	**0.1241**	**12.4%**
NASDAQ	**0.1224**	**11.8%**	**0.1144**	**6.1%**	**0.1126**	**5.7%**	**0.1123**	**6.3%**	**0.1135**	**7.2%**

Table 5.11: Comparison of SP500 and NASDAQ across different time steps with average KS score and Type I errors.

In the table 5.11 we provide the average KS scores and Type I error for both indices. For this Neural SDE implementation we didn't get as good results as the previous one. We got better results for the NASDAQ data, than the SP500 data. This can be qualitatively understood with the marginal distribution plots in the figures 5.40, and 5.41.

The autocorrelation scores in table 5.12 and figure 5.42 show similar scores and confidence intervals between generated data and real data, but not as precise as the previous implementation.

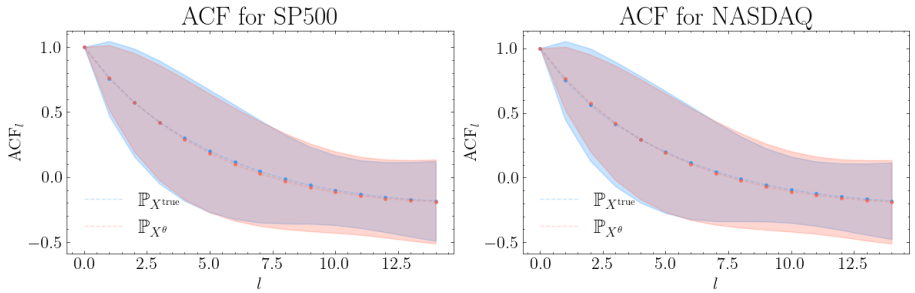


Figure 5.42: Plot with autocorrelation scores with confidence intervals at different lags for SP500 and NASDAQ.

	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$
SP500					
Gen Data	0.7641 ± 0.2491	0.5734 ± 0.3789	0.4178 ± 0.4426	0.2904 ± 0.4629	0.1844 ± 0.4556
Real Data	0.7574 ± 0.2870	0.5713 ± 0.4150	0.4216 ± 0.4728	0.3005 ± 0.4834	0.1997 ± 0.4699
NASDAQ					
Gen Data	0.7666 ± 0.2442	0.5777 ± 0.3740	0.4228 ± 0.4422	0.2962 ± 0.4641	0.1909 ± 0.4582
Real Data	0.7512 ± 0.3035	0.5627 ± 0.4343	0.4142 ± 0.4813	0.2947 ± 0.4891	0.1966 ± 0.4716

Table 5.12: Autocorrelation scores with confidence intervals at different lags for SP500 and NASDAQ.

The cross-correlation scores of both indices are shown in the figures 5.43 and 5.44. For the SP500 data we got a MSE=0.055886, and for the NASDAQ data we got MSE=0.036143.

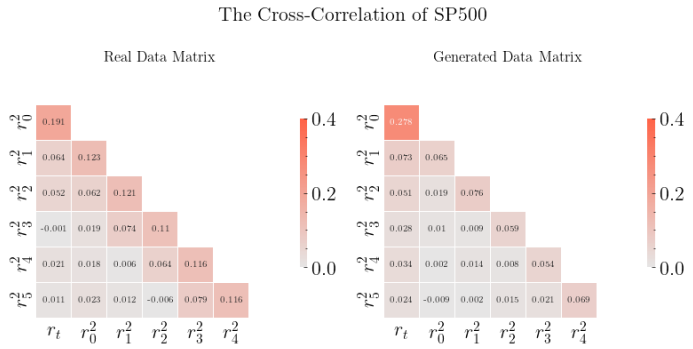


Figure 5.43: The cross-correlation matrices of the generated and real data for SP500.

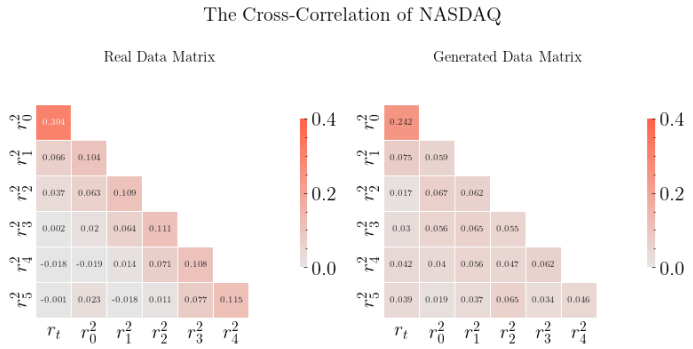


Figure 5.44: The cross-correlation matrices of the generated and real data for NASDAQ.

Although the results of the implementation of the Neural SDE aren't as accurate as the first implementation, they are still reasonable results.

The Deep Hedging results

For the Deep Hedging implementation, we initialised the paths to $S_0 = 1$, and scaled them with a factor of 0.08 as with the previous implementation. We chose the strike of the basket option as $K = S_0$.

In this setup the input of the NNs $\boldsymbol{f}$ and $\boldsymbol{f_0}$ will have an extra label for the other asset price, and the output will be two dimensional representing hedging ratios of both assets.

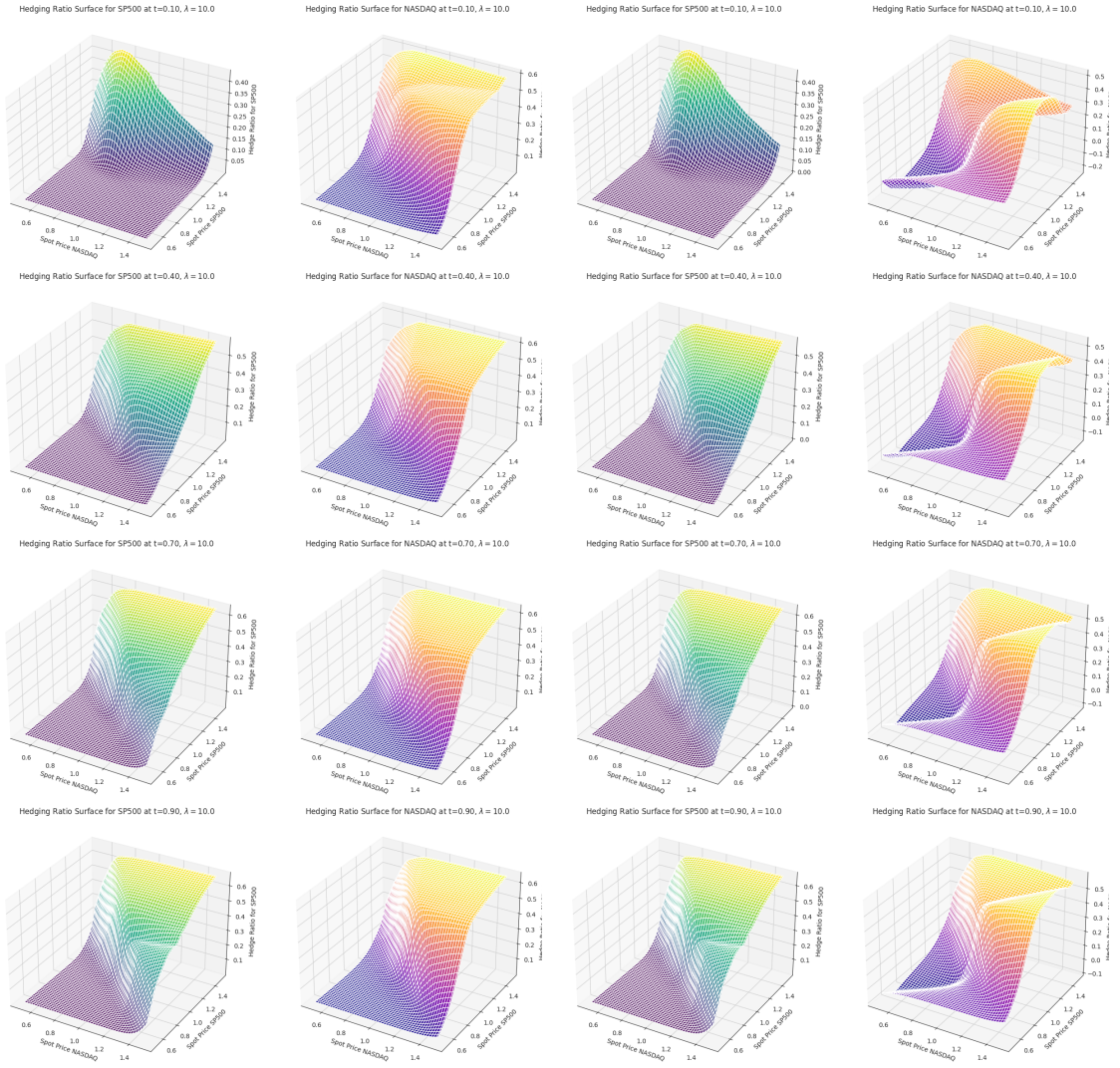


Figure 5.45: The hedging surfaces pair for both assets for $\lambda = 10$ and no transaction costs for the Rainbow Option. At the **left** we represent the pairs that follows the strategy Delta Z , at the **right** the pairs that follow the strategy Delta Q.

Now we have hedging surface pairs for the hedging ratios. In the figure 5.45 we compare how the pairs evolve over time for the strategies Delta Z and Q. The hedging surfaces don't quite resemble the Call Option structure, since the hedging ratios of one asset are not only influence by the price of its asset but also by the price of the other price. The NN learns some sense of correlation between both assets, as it tries to optimize $\rho^{\theta}(-Z)$. Comparing both strategies we note a small difference at the NASDAQ hedging surface in the middle of the slope, where the statistical arbitrage was. Nothing changes at the SP500 hedging surface. Therefore, the neural networks only notice statistical arbitrage on the NASDAQ index. We can clearly see this in the figure 5.48, where we plot the hedging ratios over time for both assets for a path following the Delta 0 strategy.

In figures 5.46, and 5.47 we show the hedging ratios over time for the strategies Delta Z and Q respectively. Comparing both figures we observe that as we get rid of the statistical arbitrage the hedging ratios tend to be in a similar range.

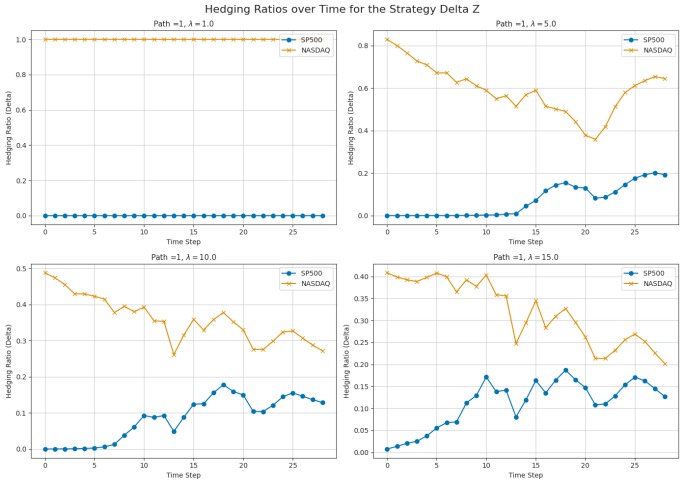


Figure 5.46: The hedging ratios for both assets over time following the Delta Z strategy for different risk aversion and without transaction costs.

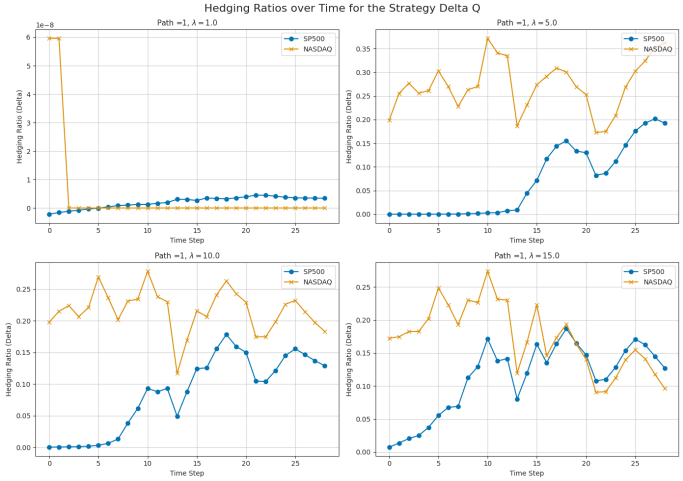


Figure 5.47: The hedging ratios for both assets over time following the Delta Q strategy for different risk aversion and without transaction costs.

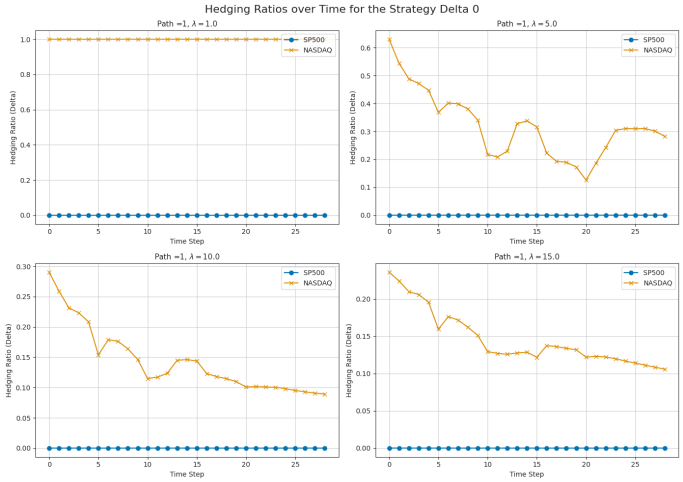


Figure 5.48: The hedging ratios for both assets over time following the Delta 0 strategy for different risk aversion and without transaction costs.

Parameters		Indif Price	Super-Hedging Ratio		VaR		CVaR	
λ	k	Gen Data	Delta Z	Delta Q	Delta Z	Delta Q	Delta Z	Delta Q
1.0	0.00%	0.0165	0.5565	**0.6660**	-0.0965	**-0.0554**	-0.1364	**-0.0824**
	0.05%	0.0165	0.5489	**0.6660**	-0.0975	**-0.0554**	-0.1374	**-0.0824**
	0.50%	0.0172	**0.6733**	**0.6733**	**-0.0547**	**-0.0547**	**-0.0816**	**-0.0816**
	5.00%	0.0172	**0.6733**	**0.6733**	**-0.0547**	**-0.0547**	**-0.0816**	**-0.0816**
5.0	0.00%	0.0162	**0.5757**	0.5562	-0.0457	**-0.0267**	-0.0610	**-0.0350**
	0.05%	0.0169	0.5674	**0.6133**	-0.0405	**-0.0333**	-0.0534	**-0.0462**
	0.50%	0.0188	**0.6881**	**0.6881**	**-0.0531**	**-0.0531**	**-0.0800**	**-0.0800**
	5.00%	0.0188	**0.6881**	**0.6881**	**-0.0531**	**-0.0531**	**-0.0800**	**-0.0800**
10.0	0.00%	0.0167	0.5724	**0.5756**	-0.0247	**-0.0202**	-0.0337	**-0.0269**
	0.05%	0.0175	0.5654	**0.5778**	-0.0256	**-0.0225**	-0.0339	**-0.0306**
	0.50%	0.0208	**0.7151**	**0.7151**	**-0.0466**	**-0.0466**	**-0.0720**	**-0.0720**
	5.00%	0.0212	**0.7097**	**0.7097**	**-0.0507**	**-0.0507**	**-0.0776**	**-0.0776**
15.0	0.00%	0.0170	**0.5858**	0.5730	-0.0202	**-0.0165**	-0.0278	**-0.0226**
	0.05%	0.0179	**0.5703**	0.5771	-0.0211	**-0.0188**	-0.0279	**-0.0251**
	0.50%	0.0224	**0.6943**	**0.6943**	**-0.0331**	**-0.0331**	**-0.0514**	**-0.0514**
	5.00%	0.0242	**0.7359**	**0.7359**	**-0.0477**	**-0.0477**	**-0.0746**	**-0.0746**
Monte Carlo		**0.0168**						

Table 5.13: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for generated data for the Rainbow Option.

Parameters		Indif Price	Super-Hedging Ratio		VaR		CVaR	
λ	k	Real Data	Delta Z	Delta Q	Delta Z	Delta Q	Delta Z	Delta Q
1.0	0.00%	0.0174	0.5325	**0.6617**	-0.1194	**-0.0595**	-0.1694	**-0.0907**
	0.05%	0.0174	0.5256	**0.6617**	-0.1204	**-0.0595**	-0.1703	**-0.0907**
	0.50%	0.0186	**0.6723**	**0.6723**	**-0.0583**	**-0.0583**	**-0.0895**	**-0.0895**
	5.00%	0.0186	**0.6723**	**0.6723**	**-0.0583**	**-0.0583**	**-0.0895**	**-0.0895**
5.0	0.00%	0.0199	0.5649	**0.5761**	-0.0640	**-0.0313**	-0.0901	**-0.0470**
	0.05%	0.0199	0.5581	**0.6163**	-0.0513	**-0.0376**	-0.0737	**-0.0554**
	0.50%	0.0206	**0.6898**	**0.6898**	**-0.0563**	**-0.0563**	**-0.0876**	**-0.0876**
	5.00%	0.0206	**0.6898**	**0.6898**	**-0.0563**	**-0.0563**	**-0.0876**	**-0.0876**
10.0	0.00%	0.0199	0.5807	**0.6114**	-0.0353	**-0.0263**	-0.0514	**-0.0411**
	0.05%	0.0208	0.5780	**0.6102**	-0.0346	**-0.0283**	-0.0498	**-0.0421**
	0.50%	0.0233	**0.7127**	**0.7127**	**-0.0469**	**-0.0469**	**-0.0750**	**-0.0750**
	5.00%	0.0237	**0.7164**	**0.7164**	**-0.0533**	**-0.0533**	**-0.0845**	**-0.0845**
15.0	0.00%	0.0201	0.5989	**0.6352**	-0.0293	**-0.0228**	-0.0432	**-0.0368**
	0.05%	0.0214	0.5947	**0.6200**	-0.0288	**-0.0248**	-0.0425	**-0.0372**
	0.50%	0.0252	**0.6959**	**0.6959**	**-0.0349**	**-0.0349**	**-0.0555**	**-0.0555**
	5.00%	0.0279	**0.7497**	**0.7497**	**-0.0491**	**-0.0491**	**-0.0803**	**-0.0803**
Monte Carlo		**0.0182**						

Table 5.14: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real data for the Rainbow Option.

In the tables 5.13 and 5.14 we provide the results for the generated and real data respectively. Firstly, we notice that for the results with the generated data we get a similar indifference price to the Monte Carlo price for low cost levels. The real data results are a little less accurate with the indifference price, but within the range of what we would expect. In addition, as we would expect, Delta Q has greater VaR and CVaR than Delta Z.

The first surprising result we get is that the strategy Delta Q have a greater super-hedging ratio around more than 60%, compared to Delta Z with ratios around 55% for both real and generated results. One plausible explanation is that the neural networks didn't properly learn the statistical arbitrage in the market, since the complexity of the model increased and they got stuck performing suboptimally only taking the statistical arbitrage of one of the assets out of the two assets. In figure 5.48 we showed how neural networks don't take into advantage the possible statistical arbitrage of the SP500 data. And in figure 5.45 we see how subtle was the difference between both hedging surfaces following Z and Q. Other plausible explanation could be due to the Neural SDE implementation of the SP500 data wasn't as a accurate as the first one, and couldn't properly reflect the statistical arbitrage of the data. Training $\boldsymbol{f_0}$ for only the generated SP500 data we got $\pi^{\theta}(0) = 3.95e-11$ which is close to 0 but it is not negative, but its Delta 0 strategy super-hedging ratio is still over 50% with 54.37%.

Regardless of why the strategy Z performs suboptimally, the strategy Delta Q properly hedge the risk of the liability, as we can observe in the figures 5.49 and 5.50. This shows how robust is the Delta Q approach that can capture the proper hedging strategy regardless of the possible statistical arbitrage captured by the model.

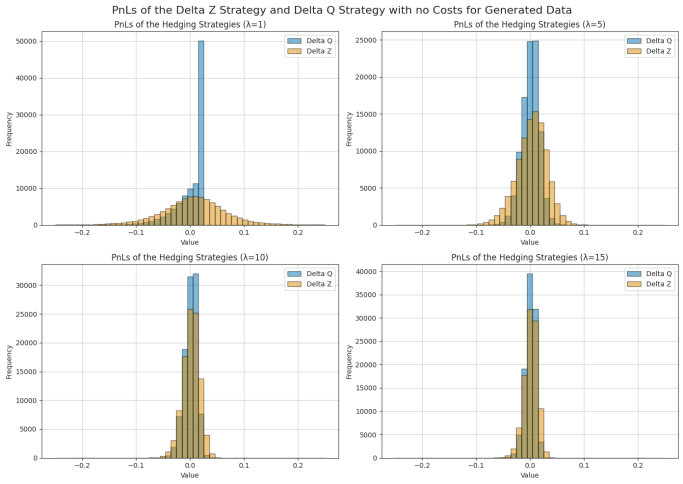


Figure 5.49: The PnLs for the Delta Z and Delta Q strategies for the generated data.

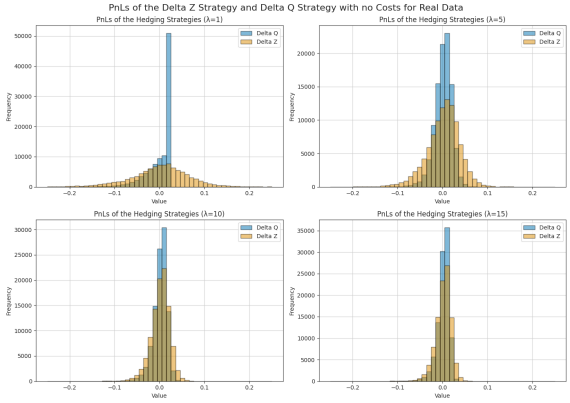


Figure 5.50: The PnLs for the Delta Z and Delta Q strategies for the real data.

In summery, in this subsection we have implemented a Rainbow option into our framework, and although the statistical arbitrage of the market wasn't fully capture by the model, we still successfully developed its hedging strategy.

Conclusion

This thesis has successfully developed and implemented a machine learning-driven framework for pricing and hedging derivatives in incomplete markets with real market data. We developed a model-free, and fully data-driven deep hedging framework that incorporates a Neural SDE trained on signature kernels as market generator.

We developed optimal trading strategies taking advantage of the statistical arbitrage of the SP500 data by training a deep hedging model with the generated data from the Neural SDE, showing how useful this market generator can be. Also, we have developed hedging strategies for the SP500 data for simple derivatives like European Call options, and more complex path-dependant derivatives like Arithmetic Asian Call options or Lookback options. We showed the hedging surfaces of these exotic derivatives with the SP500 data. Furthermore, we implemented the framework in a multivariate setting to price and hedge a Rainbow option based in the market data of the indices SP500, and NASDAQ, showing its hedging surfaces pairs too.

For all this setups we have develop two different hedging strategies based in different risk measures. We have develop the strategy Delta Z that takes into account the statistical arbitrage of the market while it hedges the liability, and the Delta Q strategy based in the risk-neutral measure that disregards the statistical arbitrage of the market, and only focus in hedging the liability. In addition, we implemented different risk aversion and cost levels for all these strategies. Our results showed how if the generated data properly reflect the statistical arbitrage of the model, the Delta Z strategy will have better super-hedging values than the Delta Q. In the other hand, Delta Q had better VaR and CVaR values, than Delta Z. Also, we have seen how robust is the strategy Delta Q, that regardless of the statistical arbitrage captured by the framework, it will give the risk-neutral strategy.

Our results for the European Call Option with SP500 data, are actually quite similar to the traditional approaches with BS hedging and Monte Carlo pricing, however over framework incorporates frictions of the market like transaction costs, and allows to adjust the risk aversion of the agent.

For the implementation of the deep hedging model we used two types of neural networks FNN and LSTM. Our results along the different setups didn't find any real difference between both performances.

Some further continuation of our work could be implementing the framework into other exotics derivatives like Autocallables options or Barrier options, where we could note the LSTM leveraging from its sequential depended setup. Another, possible further implementation for the framework could to apply it to more volatile markets like cryptocurrencies, to test the framework in more hostile markets.

Appendix A

More results

A.1 LSTM for European Call in SP500

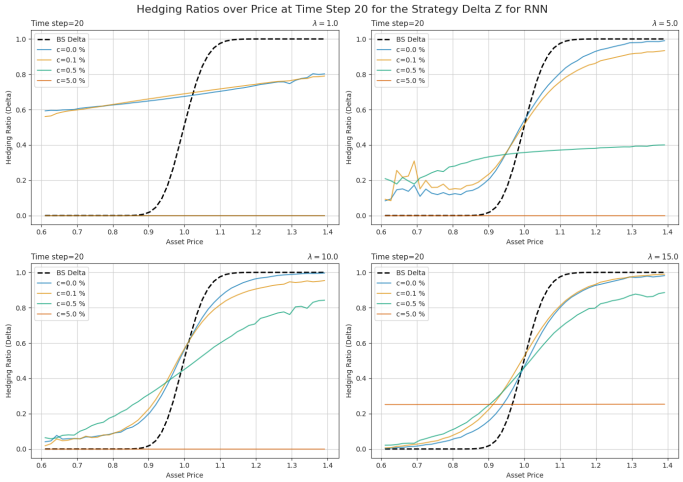


Figure A.1: he hedging ratios over asset price at time step $t = 20$ following the Delta Z strategy for different risk aversion and cost levels for LSTM.

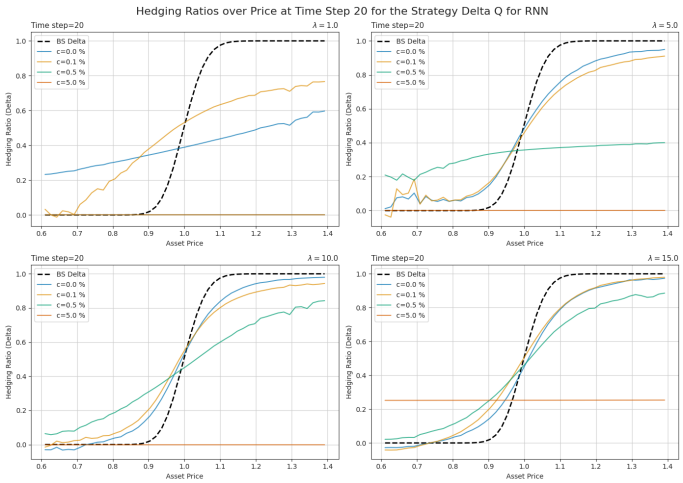


Figure A.2: he hedging ratios over asset price at time step $t = 20$ following the Delta Q strategy for different risk aversion and cost levels for LSTM.

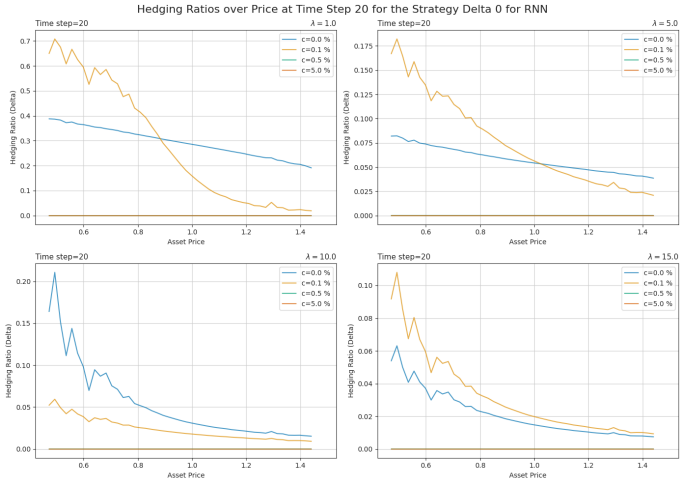


Figure A.3: he hedging ratios over asset price at time step $t = 20$ following the Delta 0 strategy for different risk aversion and cost levels for LSTM.

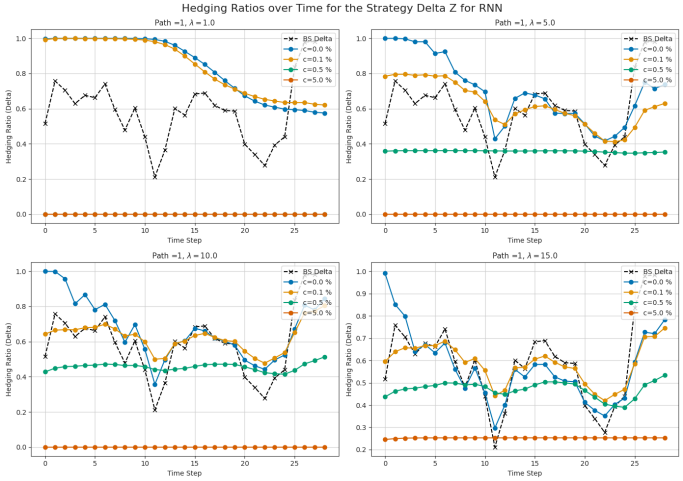


Figure A.4: The hedging ratios over time for the Delta Z strategy for different risk aversions and cost levels for LSTM.

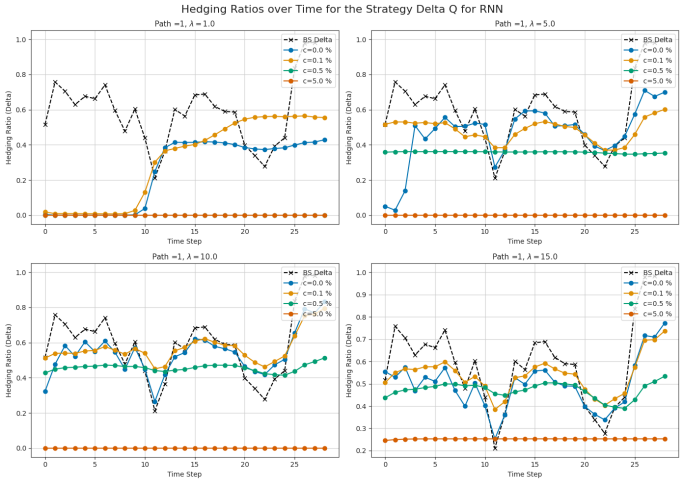


Figure A.5: The hedging ratios over time for the Delta Q strategy for different risk aversions and cost levels for LSTM.

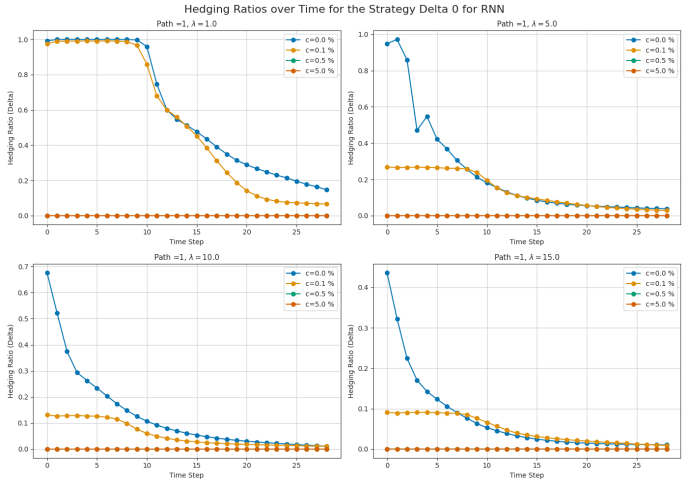


Figure A.6: The hedging ratios over time for the Delta 0 strategy for different risk aversions and cost levels for LSTM.

Parameters		**Indif Price**	**Super-Hedging Ratio**		**VaR**		**CVaR**	
λ	k	**Gen Data**	**Delta Z**	**Delta Q**	**Delta Z**	**Delta Q**	**Delta Z**	**Delta Q**
1.0	0.00%	0.0290	**0.7097**	0.6228	-0.0779	**-0.0650**	-0.1293	**-0.1122**
	0.05%	0.0291	**0.7051**	0.6021	-0.0773	**-0.0622**	-0.1285	**-0.1059**
	0.50%	0.0309	**0.6802**	**0.6802**	**-0.0931**	**-0.0931**	**-0.1540**	**-0.1540**
	5.00%	0.0309	**0.6802**	**0.6802**	**-0.0931**	**-0.0931**	**-0.1540**	**-0.1540**
5.0	0.00%	0.0292	**0.6423**	0.5965	-0.0474	**-0.0395**	-0.0661	**-0.0643**
	0.05%	0.0298	**0.6376**	0.6150	-0.0448	**-0.0409**	-0.0632	**-0.0628**
	0.50%	0.0345	**0.6785**	**0.6785**	**-0.0554**	**-0.0554**	**-0.0907**	**-0.0907**
	5.00%	0.0373	**0.7201**	**0.7201**	**-0.0868**	**-0.0868**	**-0.1476**	**-0.1476**
10.0	0.00%	0.0302	**0.6488**	0.6231	-0.0403	**-0.0343**	-0.0567	**-0.0573**
	0.05%	0.0312	**0.6516**	0.6380	-0.0410	**-0.0379**	**-0.0567**	-0.0560
	0.50%	0.0369	**0.6689**	**0.6689**	**-0.0466**	**-0.0466**	**-0.0677**	**-0.0677**
	5.00%	0.0527	**0.7992**	**0.7992**	**-0.0713**	**-0.0713**	**-0.1322**	**-0.1322**
15.0	0.00%	0.0316	**0.6509**	0.6430	-0.0347	**-0.0345**	**-0.0502**	-0.0505
	0.05%	0.0325	**0.6599**	0.6505	-0.0376	**-0.0354**	**-0.0521**	-0.0522
	0.50%	0.0388	**0.6770**	**0.6770**	**-0.0415**	**-0.0415**	**-0.0600**	**-0.0600**
	5.00%	0.0752	**0.8302**	**0.8302**	**-0.0458**	**-0.0458**	**-0.0912**	**-0.0912**
Black-Scholes		**0.0297(MC)**	**0.6333**		**-0.0353**		**-0.0590**	

Table A.1: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for generated data for LSTM.

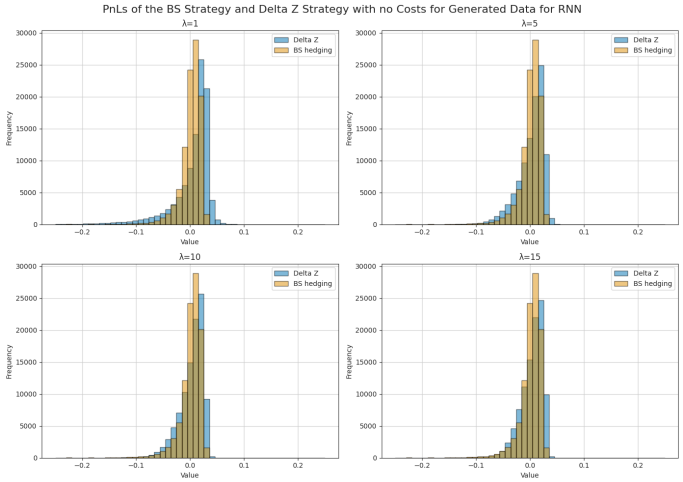


Figure A.7: The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for generated data for LSTM.

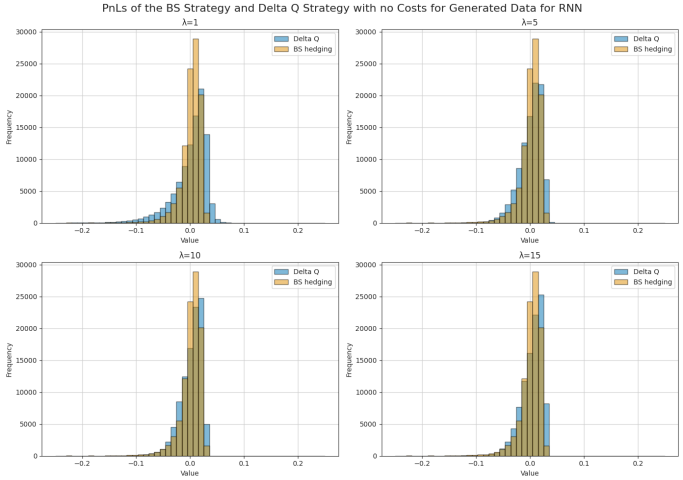


Figure A.8: The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for generated data for LSTM.

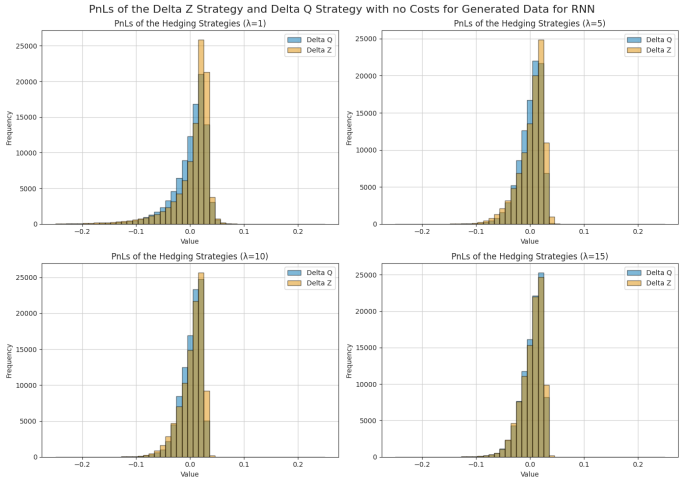


Figure A.9: The PnLs of the strategy Z and Q for different risk aversions and no transaction costs for generated data for LSTM.

Parameters		Indif Price	Super-Hedging Ratio		VaR		CVaR	
λ	k	Gen Data	Delta Z	Delta Q	Delta Z	Delta Q	Delta Z	Delta Q
1.0	0.00%	0.0285	**0.7073**	0.6289	-0.0822	**-0.0676**	-0.1357	**-0.1280**
	0.05%	0.0283	**0.7025**	0.6096	-0.0826	**-0.0644**	-0.1355	**-0.1253**
	0.50%	0.0306	**0.6792**	**0.6792**	**-0.0896**	**-0.0896**	**-0.1511**	**-0.1511**
	5.00%	0.0306	**0.6792**	**0.6792**	**-0.0896**	**-0.0896**	**-0.1511**	**-0.1512**
5.0	0.00%	0.0287	**0.6608**	0.6075	-0.0557	**-0.0426**	-0.0833	**-0.0826**
	0.05%	0.0298	**0.6422**	0.6129	-0.0470	**-0.0382**	-0.0711	**-0.0710**
	0.50%	0.0345	**0.6715**	**0.6715**	**-0.0518**	**-0.0518**	**-0.0892**	**-0.0892**
	5.00%	0.0368	**0.7144**	**0.7144**	**-0.0834**	**-0.0834**	**-0.1449**	**-0.1450**
10.0	0.00%	0.0308	**0.6800**	0.6369	-0.0465	**-0.0332**	-0.0708	**-0.0739**
	0.05%	0.0313	**0.6599**	0.6392	-0.0409	**-0.0367**	**-0.0611**	-0.0604
	0.50%	0.0368	**0.6660**	**0.6660**	**-0.0430**	**-0.0430**	**-0.0687**	**-0.0687**
	5.00%	0.0515	**0.7911**	**0.7911**	**-0.0687**	**-0.0687**	**-0.1303**	**-0.1303**
15.0	0.00%	0.0331	**0.6892**	0.6760	-0.0369	**-0.0325**	**-0.0600**	-0.0612
	0.05%	0.0330	**0.6730**	0.6614	-0.0363	**-0.0336**	**-0.0560**	-0.0566
	0.50%	0.0388	**0.6808**	**0.6808**	**-0.0391**	**-0.0391**	**-0.0617**	**-0.0617**
	5.00%	0.0741	**0.8239**	**0.8239**	**-0.0434**	**-0.0434**	**-0.0899**	**-0.0899**
Black-Scholes		**0.0294**	**0.6380**		**-0.0353**		**-0.0590**	

Table A.2: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between Delta Z and Delta Q strategies, for real data for LSTM.

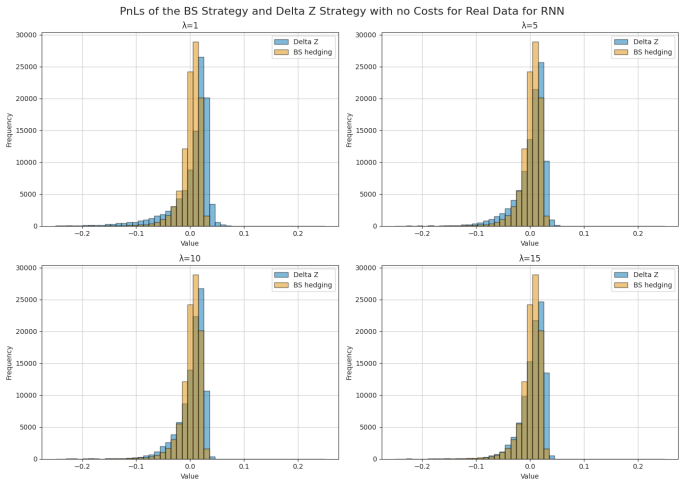


Figure A.10: The PnLs of the strategy Z and BS for different risk aversions and no transaction costs for real data for LSTM.

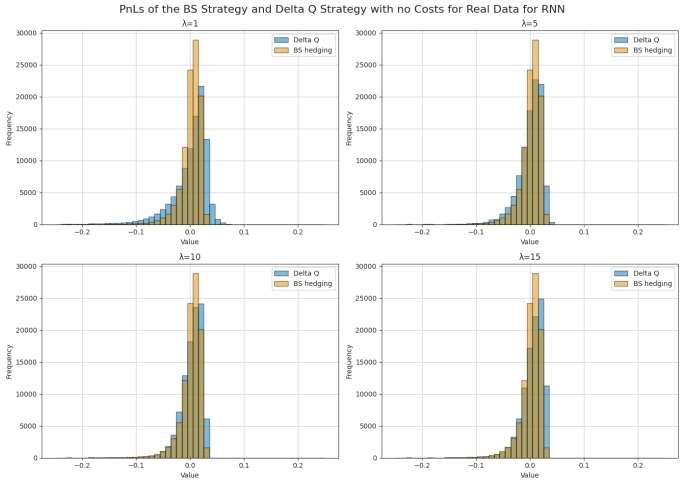


Figure A.11: The PnLs of the strategy Q and BS for different risk aversions and no transaction costs for real data for LSTM.

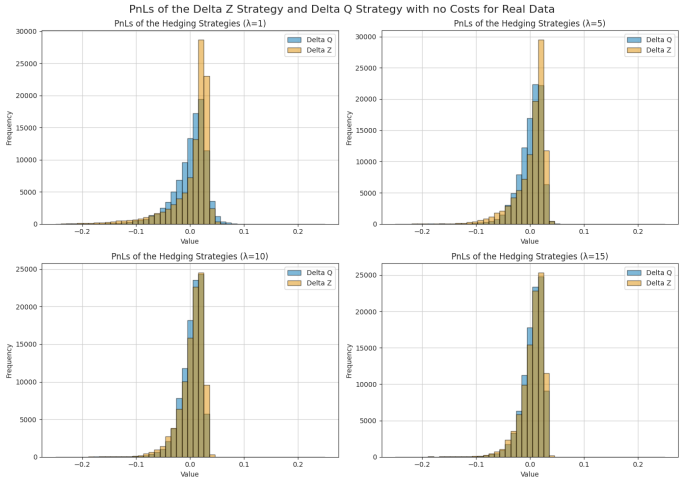


Figure A.12: The PnLs of the strategy Z and Q for different risk aversions and no transaction costs for real data for LSTM.

A.2 Generated data results for Asian Option

Parameters		Indif Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	**0.0144**	0.0145	0.6949	**0.7109**	-0.0716	**-0.0584**	-0.1171	**-0.0949**
	0.05%	**0.0144**	0.0145	0.6907	**0.7058**	-0.0708	**-0.0578**	-0.1142	**-0.0923**
	0.50%	0.0152	**0.0152**	**0.6605**	**0.6605**	**-0.0452**	**-0.0452**	**-0.0701**	**-0.0701**
	5.00%	**0.0152**	**0.0152**	**0.6605**	**0.6605**	**-0.0452**	**-0.0452**	**-0.0701**	**-0.0701**
5.0	0.00%	**0.0140**	0.0140	**0.6997**	0.6776	-0.0345	**-0.0316**	-0.0515	**-0.0479**
	0.05%	0.0144	**0.0144**	0.7175	**0.7136**	-0.0336	**-0.0311**	-0.0517	**-0.0474**
	0.50%	**0.0163**	**0.0163**	**0.6806**	**0.6806**	**-0.0419**	**-0.0419**	**-0.0656**	**-0.0656**
	5.00%	**0.0163**	**0.0163**	**0.6756**	**0.6756**	**-0.0441**	**-0.0441**	**-0.0690**	**-0.0690**
10.0	0.00%	**0.0141**	0.0141	0.6580	**0.6365**	-0.0236	**-0.0218**	-0.0330	**-0.0311**
	0.05%	0.0148	**0.0147**	**0.6949**	0.6399	-0.0287	**-0.0216**	-0.0417	**-0.0317**
	0.50%	**0.0174**	**0.0174**	**0.6786**	0.6678	**-0.0306**	-0.0300	**-0.0488**	-0.0474
	5.00%	**0.0181**	**0.0181**	**0.6976**	**0.6976**	**-0.0424**	**-0.0424**	**-0.0672**	**-0.0672**
15.0	0.00%	**0.0144**	0.0143	0.6066	**0.6267**	-0.0171	**-0.0186**	-0.0238	**-0.0262**
	0.05%	0.0148	**0.0148**	0.6376	**0.6447**	**-0.0196**	-0.0193	**-0.0265**	-0.0271
	0.50%	**0.0181**	**0.0181**	**0.6541**	0.6516	**-0.0248**	-0.0245	**-0.0383**	-0.0378
	5.00%	**0.0202**	**0.0202**	**0.7234**	**0.7234**	**-0.0402**	**-0.0402**	**-0.0651**	**-0.0651**
Monte Carlo		**0.0144**							

Table A.3: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM, following the strategy Delta Z, for the Asian Option for the Generated Data.

Parameters		Indif Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	**0.0144**	0.0145	0.5919	**0.6149**	**-0.0355**	-0.0454	**-0.0514**	-0.0679
	0.05%	**0.0144**	0.0145	0.5677	**0.6304**	-0.0383	**-0.0395**	-0.0540	**-0.0588**
	0.50%	0.0152	**0.0152**	**0.6605**	**0.6605**	**-0.0452**	**-0.0452**	**-0.0701**	**-0.0701**
	5.00%	**0.0152**	**0.0152**	**0.6605**	**0.6605**	**-0.0452**	**-0.0452**	**-0.0701**	**-0.0701**
5.0	0.00%	**0.0140**	0.0140	0.5767	**0.5881**	-0.0245	**-0.0246**	**-0.0340**	-0.0343
	0.05%	**0.0144**	0.0144	0.6249	**0.6130**	**-0.0224**	-0.0212	**-0.0318**	-0.0291
	0.50%	**0.0163**	**0.0163**	**0.6806**	**0.6806**	**-0.0419**	**-0.0419**	**-0.0656**	**-0.0656**
	5.00%	**0.0163**	**0.0163**	**0.6756**	**0.6756**	**-0.0441**	**-0.0441**	**-0.0690**	**-0.0690**
10.0	0.00%	**0.0141**	0.0141	0.5907	**0.5989**	-0.0164	**-0.0200**	-0.0236	**-0.0281**
	0.05%	**0.0148**	0.0147	**0.6159**	0.6029	-0.0211	**-0.0192**	-0.0280	**-0.0256**
	0.50%	**0.0174**	**0.0174**	**0.6786**	0.6678	**-0.0306**	-0.0300	**-0.0488**	-0.0474
	5.00%	**0.0181**	**0.0181**	**0.6976**	**0.6976**	**-0.0424**	**-0.0424**	**-0.0672**	**-0.0672**
15.0	0.00%	**0.0144**	0.0143	0.6000	**0.5912**	-0.0175	**-0.0162**	-0.0252	**-0.0233**
	0.05%	0.0148	**0.0148**	0.5974	**0.5985**	**-0.0163**	-0.0159	**-0.0222**	-0.0223
	0.50%	**0.0181**	**0.0181**	**0.6541**	0.6516	**-0.0248**	-0.0245	**-0.0383**	-0.0378
	5.00%	**0.0202**	**0.0202**	**0.7234**	**0.7234**	**-0.0402**	**-0.0402**	**-0.0651**	**-0.0651**
Monte Carlo		**0.0144**							

Table A.4: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM following the strategy Delta Q, for the Asian Option for Generated Data.

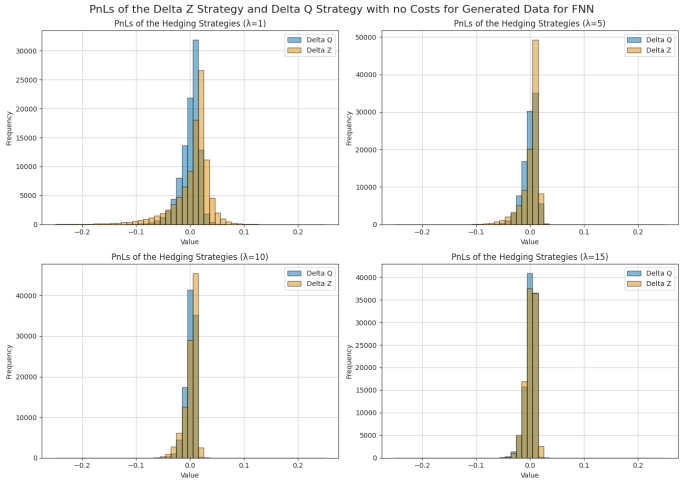


Figure A.13: The PnLs for the Delta Z and Q strategies for the Asian option using generated data for FNN.

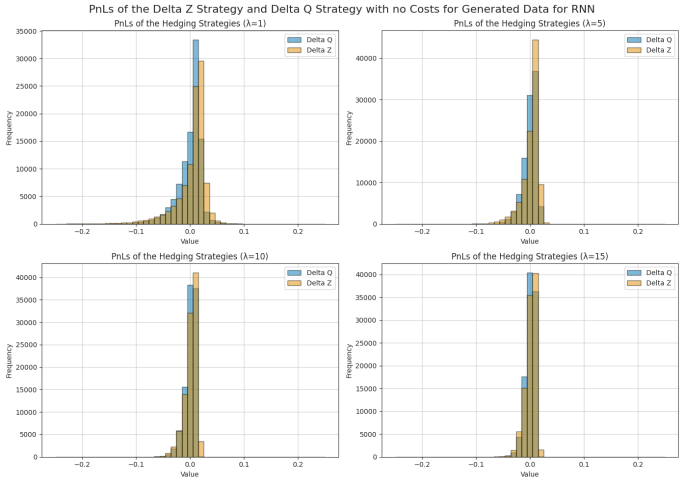


Figure A.14: The PnLs for the Delta Z and Q strategies for the Asian option using generated data for LSTM.

A.3 Generated data results for Lookback Option

Parameters		Indif Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	0.0510	**0.0513**	**0.6767**	0.6947	**-0.1075**	-0.1072	**-0.1672**	-0.1701
	0.05%	0.0511	**0.0512**	0.6668	**0.6743**	**-0.1034**	-0.1072	**-0.1609**	-0.1701
	0.50%	0.0528	**0.0528**	**0.6522**	**0.6522**	**-0.1072**	**-0.1072**	**-0.1701**	**-0.1701**
	5.00%	0.0528	**0.0528**	**0.6522**	**0.6522**	**-0.1072**	**-0.1072**	**-0.1701**	**-0.1701**
5.0	0.00%	0.0539	**0.0536**	**0.6616**	0.6490	**-0.0711**	-0.0692	**-0.1073**	-0.1006
	0.05%	0.0542	**0.0542**	0.6510	**0.6541**	**-0.0687**	-0.0699	**-0.1009**	-0.1013
	0.50%	0.0585	**0.0583**	**0.6827**	0.6670	**-0.0812**	-0.0788	**-0.1254**	-0.1147
	5.00%	0.0612	**0.0612**	**0.7060**	**0.7060**	**-0.0988**	**-0.0988**	**-0.1617**	**-0.1617**
10.0	0.00%	0.0574	**0.0573**	**0.6735**	0.6756	**-0.0596**	-0.0599	**-0.0876**	-0.0886
	0.05%	0.0582	**0.0582**	0.6711	**0.6705**	**-0.0594**	-0.0611	**-0.0878**	-0.0875
	0.50%	0.0642	**0.0638**	0.6851	**0.6921**	**-0.0697**	-0.0711	**-0.0983**	-0.1007
	5.00%	0.0816	**0.0816**	**0.8025**	**0.8025**	**-0.0784**	**-0.0784**	**-0.1413**	**-0.1413**
15.0	0.00%	0.0621	**0.0616**	0.7167	**0.7034**	**-0.0544**	-0.0550	**-0.0832**	-0.0806
	0.05%	0.0626	**0.0628**	**0.7022**	0.7060	**-0.0541**	-0.0571	**-0.0809**	-0.0823
	0.50%	0.0699	**0.0696**	0.7207	**0.7185**	**-0.0636**	-0.0616	**-0.0898**	-0.0885
	5.00%	0.1101	**0.1091**	**0.8267**	0.8227	**-0.0568**	-0.0589	**-0.1032**	-0.1052
Monte Carlo		**0.0516**							

Table A.5: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Z, for generated data.

Parameters		Indif Price		Super-Hedging Ratio		VaR		CVaR	
λ	k	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**	**FNN**	**LSTM**
1.0	0.00%	0.0510	**0.0513**	**0.6236**	0.6558	**-0.0818**	-0.0827	**-0.1183**	-0.1325
	0.05%	0.0511	**0.0512**	**0.6172**	0.6200	**-0.0775**	-0.0765	**-0.1134**	-0.1133
	0.50%	**0.0528**	**0.0528**	**0.6522**	**0.6522**	**-0.1072**	**-0.1072**	**-0.1701**	**-0.1701**
	5.00%	**0.0528**	**0.0528**	**0.6522**	**0.6522**	**-0.1072**	**-0.1072**	**-0.1701**	**-0.1701**
5.0	0.00%	**0.0539**	0.0536	0.6391	**0.6376**	**-0.0629**	-0.0644	**-0.0922**	-0.0915
	0.05%	**0.0542**	**0.0542**	0.6351	**0.6417**	**-0.0672**	-0.0679	**-0.0926**	-0.0957
	0.50%	**0.0585**	0.0583	**0.6827**	0.6670	**-0.0812**	-0.0788	**-0.1254**	-0.1147
	5.00%	**0.0612**	**0.0612**	**0.7060**	**0.7060**	**-0.0988**	**-0.0988**	**-0.1617**	**-0.1617**
10.0	0.00%	**0.0574**	0.0573	**0.6603**	0.6648	**-0.0587**	-0.0586	**-0.0846**	-0.0856
	0.05%	**0.0582**	**0.0582**	0.6627	**0.6666**	**-0.0590**	-0.0618	**-0.0852**	-0.0869
	0.50%	**0.0642**	0.0638	0.6851	**0.6921**	**-0.0697**	-0.0711	**-0.0983**	-0.1007
	5.00%	**0.0816**	**0.0816**	**0.8025**	**0.8025**	**-0.0784**	**-0.0784**	**-0.1413**	**-0.1413**
15.0	0.00%	**0.0621**	0.0616	**0.7098**	0.6963	**-0.0536**	-0.0560	**-0.0811**	-0.0800
	0.05%	**0.0626**	0.0628	0.6976	**0.7026**	**-0.0545**	-0.0579	**-0.0799**	-0.0820
	0.50%	**0.0699**	0.0696	0.7207	**0.7185**	**-0.0636**	-0.0616	**-0.0898**	-0.0885
	5.00%	**0.1101**	0.1091	**0.8267**	0.8227	**-0.0568**	-0.0589	**-0.1032**	-0.1052
Monte Carlo		**0.0516**							

Table A.6: Comparison of Indifference Prices, Super-Hedging Ratios, VaR, and CVaR between FNN and LSTM strategies following Delta Q, for generated data.

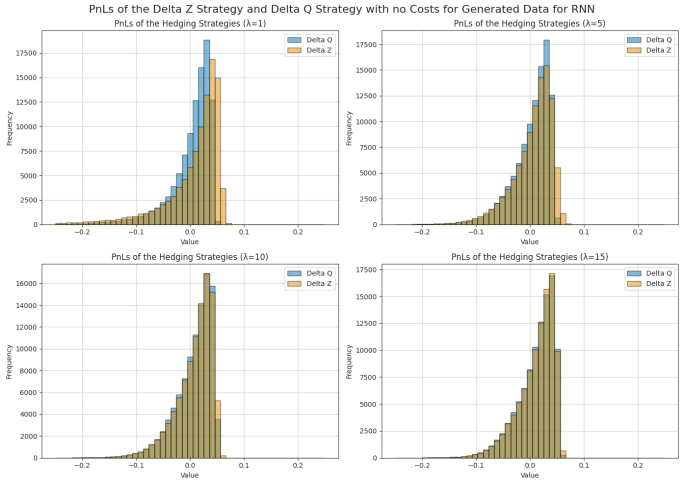


Figure A.16: The PnLs for the Delta Z and Q strategies for the Asian option using generated data for LSTM.

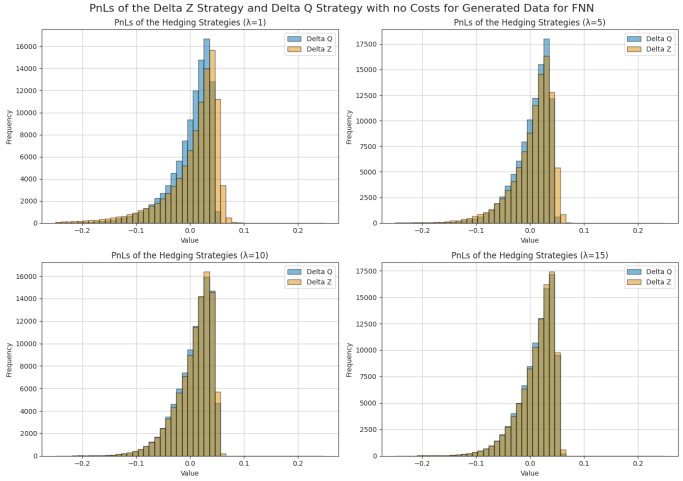


Figure A.15: The PnLs for the Delta Z and Q strategies for the Asian option using generated data for FNN.

Bibliography

[1] F. Biagini, L. Gonon, A. Mazzon, and T. Meyer-Brandis, *Detecting asset price bubbles using deep learning*, Mathematical Finance, n/a (2024).

[2] F. Biagini, L. Gonon, and T. Reitsam, *Neural network approximation for super-hedging prices*, Mathematical Finance, 33 (2023), pp. 146–184.

[3] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy, 81 (1973), pp. 637–654.

[4] H. Buehler, L. Gonon, J. Teichmann, and B. Wood, *Deep hedging*, QUANTITATIVE FINANCE, 19 (2019), pp. 1271–1291.

[5] H. Buehler, P. Murray, M. S. Pakkanen, and B. Wood, *Deep hedging: Learning risk-neutral implied volatility dynamics*, 2021.

[6] H. Buehler, P. Murray, and B. Wood, *Deep bellman hedging*, 2024.

[7] J. Cao, J. Chen, J. Hull, and Z. Poulos, *Deep hedging of derivatives using reinforcement learning*, arXiv preprint arXiv:2103.16409, (2021).

[8] C. Carmeli, E. D. Vito, A. Toigo, and V. Umanità, *Vector valued reproducing kernel hilbert spaces and universality*, 2008.

[9] T. Cass and C. Salvi, *Lecture notes on rough paths and applications to machine learning*, 2024.

[10] T. Cass and W. F. Turner, *Topologies on unparameterised path space*, 2022.

[11] R. Cont, *Empirical properties of asset returns: stylized facts and statistical issues*, Quantitative finance, 1 (2001), p. 223.

[12] D2L.ai, *Long short-term memory (lstm).* `https://d2l.ai/chapter_recurrent-modern/lstm.html`, 2023. Accessed: 2024-08.

[13] Y. Dauphin, H. Vries, J. Chung, and Y. Bengio, *Rmsprop and equilibrated adaptive learning rates for non-convex optimization*, arXiv, 35 (2015).

[14] M. H. Davis, V. G. Panas, and T. Zariphopoulou, *European option pricing with transaction costs*, SIAM Journal on Control and Optimization, 31 (1993), pp. 470–493.

[15] L. Deng, *Deep learning: from speech recognition to language and multimodal processing*, APSIPA Transactions on Signal and Information Processing, 5 (2016), p. e1.

[16] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, Journal of machine learning research, 12 (2011).

[17] Y. Goldberg, *Neural network methods for natural language processing*, Springer Nature, 2022.

[18] L. Gonon, *Lecture notes of deep learning*, 2023. Lecture notes, Deep Learning, Imperial College.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[20] I. Gühring, G. Kutyniok, and P. Petersen, *Error bounds for approximations with deep relu neural networks in ws,p norms*, Analysis and Applications, 18 (2020), pp. 803–859.

[21] J. Guyon and J. Lekeufack, *Volatility is (mostly) path-dependent*, Quantitative Finance, 23 (2023), pp. 1221–1258.

[22] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[23] S. Hodges, *Optimal replication of contingent claims under transaction costs*, Review Futures Market, 8 (1989), pp. 222–239.

[24] B. Horvath, J. Teichmann, and Ž. Žurič, *Deep hedging under rough volatility*, Risks, 9 (2021), p. 138.

[25] S. ichi Amari, *Backpropagation and stochastic gradient descent method*, Neurocomputing, 5 (1993), pp. 185–196.

[26] A. Ilhan, M. Jonsson, and R. Sircar, *Optimal static-dynamic hedges for exotic options under convex risk measures*, Stochastic Processes and their Applications, 119 (2009), pp. 3608–3632. Funding Information: The second author was partially supported by NSF grant DMS-044965. Third author's work was partially supported by NSF grants DMS-0456195 and DMS-0807440.

[27] Z. Issa, B. Horvath, M. Lemercier, and C. Salvi, *Non-adversarial training of neural sdes with signature kernel scores*, in Advances in Neural Information Processing Systems, vol. 36, Curran Associates, Inc., 2023, pp. 11102–11126. NeurIPS 2023, the Thirty-seventh Annual Conference on Neural Information Processing Systems, New Orleans, LA, USA, December 10-16, 2023.

[28] S. Jansen, *Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python*, Packt Publishing Ltd, 2020.

[29] P. Kidger, *On neural differential equations*, 2022.

[30] P. Kidger, J. Foster, X. Li, H. Oberhauser, and T. Lyons, *Neural sdes as infinite-dimensional gans*, 2021.

[31] D. Kingma, *Adam: a method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[32] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks, 6 (1993), pp. 861–867.

[33] J. Lim, *A numerical algorithm for indifference pricing in incomplete markets*, SSRN Electronic Journal, (2006).

[34] E. Lütkebohmert, T. Schmidt, and J. Sester, *Robust deep hedging*, Quantitative Finance, 22 (2022), pp. 1465–1480.

[35] S. Raj, I. Kerenidis, A. Shekhar, B. Wood, J. Dee, S. Chakrabarti, R. Chen, D. Herman, S. Hu, P. Minssen, et al., *Quantum deep hedging*, Quantum, 7 (2023), p. 1191.

[36] C. Salvi, T. Cass, J. Foster, T. Lyons, and W. Yang, *The signature kernel is the solution of a goursat pde*, SIAM Journal on Mathematics of Data Science, 3 (2021), p. 873–899.

[37] F. Santambrogio, *Euclidean, metric, and wasserstein gradient flows: an overview*, Bulletin of Mathematical Sciences, 7 (2017), pp. 87–154.

[38] B. K. Sriperumbudur, K. Fukumizu, and G. R. G. Lanckriet, *Universality, characteristic kernels and rkhs embedding of measures*, 2010.

[39] I. Steinwart, *On the influence of the kernel on the consistency of support vector machines*, J. Mach. Learn. Res., 2 (2002), p. 67–93.

[40] B. Tzen and M. Raginsky, *Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit*, 2019.

[41] Wikipedia contributors, *Activation function*. `https://en.wikipedia.org/wiki/Activation_function`, 2024. Accessed: August 10, 2024.

[42] M. Xu, *Risk measure pricing and hedging in incomplete markets*, Annals of Finance, 2 (2006), pp. 51–71.

[43] D. Yarotsky, *Error bounds for approximations with deep relu networks*, Neural Networks, 94 (2017), pp. 103–114.