Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

Pricing and Hedging of Derivatives by Unsupervised Deep Learning

Author: Peixuan Qin (CID : 01772192)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2019-2020

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr.Mikko Pakkanen, who has provided me with his selfless support and constantly advise. He has helped me build up the necessary knowledge of this project and his feedback is very essential.

I would also like to express my deepest appreciation to all the lecturers who helped me to address problems I encountered during the courses, it is very honored for me to be a part of this MSc programme.

Finally, my special thank goes to my family and my boyfriend who always stand by my side and support me to finish this work during this difficult epidemic period.

Contents

1 Introduction **3**

1.1 Outline 4

2 Hedging Approaches **4**

2.1 Market setting 4

2.2 Delta Hedging 5

2.3 Utility Function Method 6

2.3.1 Risk measure 6

2.3.2 Indifference price 7

2.3.3 Exponential Utility Indifference Pricing 7

3 Feedforward Neural Network and Long Short Term Memory **8**

3.1 Feedforward Neural Network (FNN) 8

3.1.1 Basic structure 8

3.1.2 Activation Functions 9

3.1.3 Loss Functions 10

3.1.4 SGD and Backpropagation 10

3.2 Long Short Term Memory (LSTM) 12

4 Deep Hedging **15**

4.1 Model setting 15

4.2 Advantages and Drawbacks 16

5 Experiment **17**

5.1 Data generation 17

5.2 Result 17

5.2.1 European call option 19

5.2.2 Asian Call Option 25

5.2.3 Digital Call Option (Cash or nothing) 27

5.2.4 Digital Call Option(Asset or nothing) 31

5.2.5 Barrier Option 35

6 Conclusion **42**

1 Introduction

Pricing and hedging derivatives in the market are crucial problems in the financial industry. In an ideal market without frictions, a trader can solve a hedging problem using trivial mathematical derivation based on option pricing model. For instance, in perfect Bachelier model or Black Scholes model, we can obtain unique option price and hedging strategy by changing the measure. However, these theories and results cannot be applied directly to real market, as there are many other frictions and uncertainties in reality, e.g. transaction costs, various signals, liquidity constraints, temporary and permanent price impact during liquidation. These factors requires trader to adjust the Greeks computed by a complete market model by their own knowledge and preferences, which might influence the efficiency of the hedging and accuracy of pricing. To learn and simulate the real market to the fullest, researchers have built up many efficient models, for example, Rogers and Singh (2010) and Bank et al. (2017) study market with temporary price impact, and present theoretical results of the trading strategy.

Apart from these, some machine learning based model-free methods are rising in pricing and hedging problem, and this paper mainly draws on following two studies. Halperin (2017) studies the hedging problem using Q-learning with stock price under Black Scholes assumptions and without trading cost. Buhler et al. (2018) proposes a novel approach named deep hedging which has been tested to be very efficient on hedging, it presents a framework even for hedging portfolio of derivatives in market with frictions. There are also many other reinforcement learning methods applied in risk management and quantitative finance sectors, and researches have shown some promising results in, e.g. portfolio optimization (Moody and Wu (2002)) and algorithmic trading (Du et al. (2016)).

Deep hedging is an unsupervised learning-based approach to determine optimal hedging strategies for options and other derivatives. It was originally devised by researchers at JP Morgan and ETH Zurich in 2017. This approach does not assume a complete market and accommodates multiple frictions. Fundamentally, it is based on the idea to represent the hedging strategy as a neural network, whose feature sets may contain current and past market data, including price, volatility, signal and etc. Subsequently, the network is trained to optimize the trader's PnL, according to some performance measure, e.g., squared error, convex risk measure or exponential utility function (Ilhan et al. (2009) and Föllmer and Leukert (2000)). Finally, we use new features to do the prediction, which would give us the optimal hedging strategy and indifference price. It has been proved that deep feedforward networks satisfy universal approximation properties, see, e.g., Hornik (1991). Moreover Buhler et al. (2018) suggest some explanations of the outstanding efficiency of neural networks at approximating hedging strategies as in Bolcskei et al. (2019). In this paper, we generate the market price following a discrete-time version of geometric Brownian motion and adapt Black Scholes model to be our benchmark.

1.1 Outline

This rest of the paper is organised as follows. In Part 2, some hedging methods such as delta hedging and hedging under risk measure are demonstrated. Part 3 introduces the basic structure of feedforward neural network and LSTM. Our deep hedging model setting is illustrated by an example of hegding European call option in Part 4. Part 5 contains results and comments for various option types. Part 6 illustrates the main conclusion of the experiment and some pros and cons of the model.

2 Hedging Approaches

2.1 Market setting

Consider a discrete-time financial market with finite time horizon 1 and trading time step $0 = t_0 < t_1 < \ldots < t_T = 1$. Fix a finite probability space $\Omega = \{\omega_1, \ldots, \omega_N\}$ and a probability measure $\mathbb{P}$ such that $\mathbb{P}[\{\omega_i\}] > 0$ for all i. We define the set of all real-valued random variables over Ω as $\mathscr{X} := \{X : \Omega \to \mathbb{R}\}$.

We denote by I_k with values in $\mathbb{R}$ any new market information available at time t_k, including market prices, trading cost, news, balance sheet information, any trading signals, risk limits etc. The filtration $\mathbb{F} = (\mathscr{F}_k)_{k=0,\ldots,T}$, is generated by the process $I = (I_k)_{k=0,\ldots,}$. For instance, $\mathscr{F}_k$ represents all information available up to t_k. Note that each $\mathscr{F}_k$ -measurable random variable can be written as a function of $I_0, \ldots, I_k$.

In this paper, we assume the Black Scholes market only contains derivatives over one asset whose price is given by a $\mathbb{F}$-adapted stochastic process $S = (S_k)_{k=0\ldots T}$. In order to hedge a option within the maturity, we may trade in its underlying S using an $\mathbb{F}$-adapted stochastic process $\delta = (\delta_k)_{k=0\ldots T-1}$. Here, δ_k denotes the agent's holdings of the asset at time t_k. We also define $\delta_{-1} = \delta_T := 0$ for notational convenience.

2.2 Delta Hedging

Delta, $\Delta_t = \frac{\partial C_t}{\partial S_t}$, measures the exposure of a derivative to changes in the value of the underlying. An option is usually exposed to the risk of the price fluctuation of the asset, whereas delta hedging is an option trading strategy that aims to reduce the directional risk associated with price movements.

Mathematically, we can derive the Taylor expansion of the value of an option $C(S)$, for a change ϵ in the underlying S.

$$C(S+\epsilon) = C(S) + \epsilon C'(S) + \frac{1}{2}\epsilon^2 C''(S) + \dots \tag{1}$$

where $C'(S) = \Delta$ and $C''(S) = \Gamma$. For small change in the price of underlying, we can ignore the terms after the first order derivatives and use quantity Δ to determine how many underlying to buy or sell to create a hedging portfolio.

For example, if we long a stock call option with delta of 0.45 at time t, then the option value decreases by \$0.45 when underlying price falls by \$1. In order to offset this risk, we manage to reach a delta neutral position where the overall delta is as close to zero as possible. This will minimize the option's price movements influenced by the underlying. To do this, we set up a dynamic trading strategy to hedge the payoff of the option at time t.

$$C_t(S_t) = x_{t-1} + \Delta_t(S_t - S_{t-1})$$

where $x_{t-1} = C_{t-1}(S_{t-1})$ is the previous wealth and $\Delta_t = 0.45$ is the hedge ratio.

Furthermore, it is easy to infer from equation (1) that there is also delta-gamma hedging. This is a hedging strategy that combines delta and gamma to mitigate both the risk of price change and of volatility of this change. But for simplicity, we will not discuss in this paper.

While delta hedging allows traders to hedge the risk of adverse price changes in a portfolio, in reality, continuous rebalancing leads to very high accrued transaction costs. In addition, delta hedging relies on taking derivative of the payoff in certain pricing model, so it is not suitable for data-driven risk management. However, we shall refer to this strategy as a benchmark to evaluate the hedging performance by neural network.

2.3 Utility Function Method

2.3.1 Risk measure

In an idealized complete market, such as binomial model, with continuous-time trading, no transaction costs, and unconstrained hedging, for any payoff X there exists a unique replication strategy δ and a fair price $p_0 \in R$ such that $p_0 + (\delta \cdot S)^T - X = 0$ holds $\mathbb{P}-$a.s, where

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

We denote by $\mathcal{H}^u$ the unconstrained set of such trading strategies.

This strategy is the delta hedging we discuss above. In this paper, we assume a market without trading coststhen the strategy should have the terminal wealth

$$\mathrm{PnL}_T(X, p_0, \delta) := -X + p_0 + (\delta \cdot S)_T$$

If δ and p_0 do not have analytical results, we can work out the replication strategy by utility function method.

First, we have to specify an optimality criterion which defines an acceptable "minimal price" for any position. Such a minimal price is going to be the minimal amount of cash we need to add to our position in order to implement the optimal hedge and such that the overall position becomes acceptable in light of the various costs and constraints. We focus here on optimality under convex risk measures as studied e.g. in Xu (2006) and Ilhan et al. (2008).

Definition 1 *Assume that $X, X_1, X_2 \in \mathcal{X}$ represent asset positions. We call $\rho : \mathcal{X} \to \mathbb{R}$ a convex risk measure if it has following properties:*

1. *Monotonicity: if $X_1 \geq X_2$, then $\rho(X_1) \leq \rho(X_2)$. This implies that a portfolio with greater future returns has less risk.*

2. *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 implies that a merge of portfolios does not create extra risk.*

3. *Translation invariance: $\rho(X + c) = \rho(X) - c$ for $c \in \mathbb{R}$. This implies that the addition of a sure amount of cash reduces the risk by the same amount. In particular, when $c = \rho(X), \rho(X+c) = 0$, in other words, $\rho(X)$ is the least amount of capital that needs to be added to the position X in order to make it acceptable in the sense that $\rho(X + c) \leq 0$.*

2.3.2 Indifference price

Now, let $\rho : \mathscr{X} \to \mathbb{R}$ be such a convex risk measure and consider an optimization problem

$$\pi(X) := \inf_{\delta \in \mathscr{H}} \rho(X + (\delta \cdot S)_T) \tag{2}$$

It is not difficult to conclude that π is also monotone decreasing and translation invariant, see Buhler et al. (2018). We can then define an optimal replication strategy $\delta \in \mathscr{H}$ as a minimizer of equation (2). Recalling the interpretation of *Translation invariance*, $\rho(X)$ as the minimal amount of c that has to be added to the risky position X to make it acceptable for the risk measure ρ, this also means that $\pi(X)$ is simply the minimal amount that the agent needs to charge in order to make her terminal position acceptable, if she hedges optimally.

If we defined this as the minimal price, then we would exclude the possibility that having no liabilities may actually have positive value. We then define the Indifference price $p(X)$ as the amount of cash that a trader needs to charge in order to be indifferent between the position X and making no trade at all(this indicates $X = 0$). Thus the indifference price would be the solution p to $\pi(-Z + p) = \pi(0)$. By the *Translation invariance* property, we know that $\pi(-Z + p) = \pi(-Z) - p$, so the indifference price

$$p = \pi(-Z) - \pi(0) \tag{3}$$

We can see that if there are no trading constraints and transaction costs, the indifference price p coincides with the initial price of a replication portfolio if it exists and is unique (Buhler et al. (2018)).

2.3.3 Exponential Utility Indifference Pricing

We define price q using exponential utility function $U(x) := -\exp(-\lambda x), x \in \mathbb{R}$ with risk aversion λ. From the discussion above, q should satisfy

$$\begin{aligned} \sup_{\delta \in \mathscr{H}} \mathbb{E}[U(q - X + (\delta \cdot S)_T)] &= \sup_{\delta \in \mathscr{H}} \mathbb{E}[U((\delta \cdot S)_T)] \\ \exp(-\lambda q) &= \frac{\sup_{\delta \in \mathscr{H}} \mathbb{E}[U(-X + (\delta \cdot S)_T)]}{\sup_{\delta \in \mathscr{H}} \mathbb{E}[U((\delta \cdot S)_T)]} \\ q &= \frac{1}{\lambda} \log\left(\frac{\sup_{\delta \in \mathscr{H}} \mathbb{E}[U(-X + (\delta \cdot S)_T))]}{\sup_{\delta \in \mathscr{H}} \mathbb{E}[U((\delta \cdot S)_T)]}\right) \end{aligned} \tag{4}$$

In this case, if the trader charges q amount of cash, sells option with payoff X and trades in the market according to δ, she then obtains the same expected utility as not selling the option.

Now if we choose

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

as the entropic risk measure, we can obtain p from equation (3)

$$\begin{aligned}
p &= \pi(-X) - \pi(0) \\
&= \inf_{\delta\in\mathscr{H}} \rho(-X + (\delta\cdot S)_T) - \inf_{\delta\in\mathscr{H}} \rho((\delta\cdot S)_T) \\
&= \frac{1}{\lambda}\log\left(\frac{\inf_{\delta\in\mathscr{H}}\mathbb{E}[\exp(-\lambda(-X+(\delta\cdot S)_T))]}{\inf_{\delta\in\mathscr{H}}\mathbb{E}[\exp(-\lambda((\delta\cdot S)_T))]}\right) \\
&= \frac{1}{\lambda}\log\left(\frac{\inf_{\delta\in\mathscr{H}}\mathbb{E}[\exp(-\lambda(-X+(\delta\cdot S)_T))]}{\inf_{\delta\in\mathscr{H}}\mathbb{E}[\exp(-\lambda((\delta\cdot S)_T))]}\right) \\
&= q
\end{aligned} \tag{5}$$

Thus, we prove that price q obtained using exponential utility with risk averse λ is equivalent to the indifference price p.

3 Feedforward Neural Network and Long Short Term Memory

3.1 Feedforward Neural Network (FNN)

3.1.1 Basic structure

Neural network are functions constructed by composing alternatingly affine and none-linear functions. Mathematically, a feed-forward neural network can be defined as in Pakkanen (2019)

Definition 2 *Let $I, O, r \in \mathbb{N}$. A function $f : \mathbb{R}^I \to \mathbb{R}^O$ is a feed-forward neural network (FNN) with $r-1 \in \{0,1,\dots\}$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i-th hidden layer for any $i = 1,\dots,r-1$, and activation functions $\boldsymbol{\sigma}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}, i = 1,\dots,r$, where $d_r := O$, if*

$$\boldsymbol{f} = \boldsymbol{\sigma}_r \circ \boldsymbol{L}_r \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{L}_1$$

where $L_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, for any $i = 1,\dots,r$, is an affine function

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

parameterised by weight matrix $W^i = \left[W^i_{j,k}\right]_{j=1,\dots,d_i,k=1,\dots,d_{i-1}} \in \mathbb{R}^{d_i \times d_{i-1}}$ and bias vector $\boldsymbol{b}^i =$

$\left(b^i_1,\ldots,b^i_{d_i}\right) \in \mathbb{R}^{d_i}$, *with* $d_0 := I$. *We shall denote the class of such functions* $\boldsymbol{f}$ *by*

$$\mathcal{N}_r(I,d_1,\ldots,d_{r-1},O;\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_r)$$

We call $r,d_1,\ldots,d_{r-1}$ the hyperparameters of the FNN and the weights and biases in $\boldsymbol{\theta} = (W^1,\ldots,W^r,\boldsymbol{b}^1,\ldots,\boldsymbol{b}^r)$ parameters of the network, together with the activation functions $\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_r$, they determine the architecture of the network.

3.1.2 Activation Functions

There are some common activation functions that have been testified to be quite effective to build up the network. For example, *Identity*: $g(x) = x$, *Sigmoid*(σ) :$g(x) = \frac{1}{1+e^{-x}}$, *ReLU* : $g(x) = \max\{x,0\}$ and *Hyperbolic tangent(tanh)*: $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Figure 1 are the graphs of these activation functions.

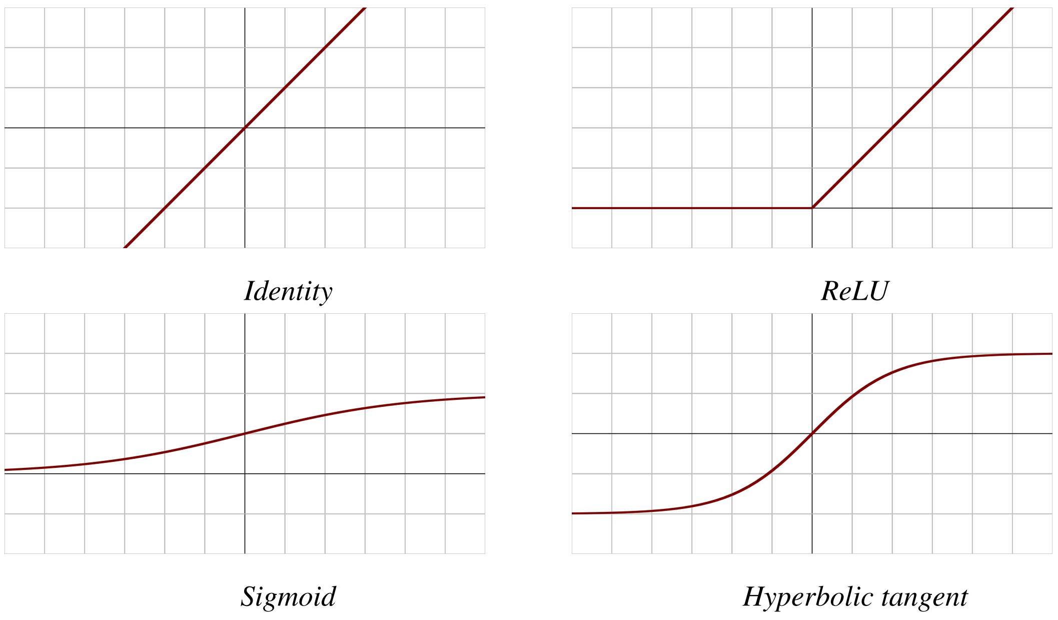

Figure 1: Common activation functions

The *Sigmoid* and the *tanh* are said to be saturating, which means that their output is bounded. They are widely used in the output layer to produce bounded output values in some particular range, such as probability (0 to 1). In contrast, *Identity* can be used to output unbounded result. *ReLU* is now commonly used in hidden layers. *ReLU* and its derivative are mathematically very simple, making them numerically efficient. Although the derivative of *ReLU* is undefined at zero, we can usually substitute it with value 1 and the in-continuity does not jeopardize the effectiveness of the network much in the experiments by Glorot et al. (2011).

3.1.3 Loss Functions

The notions of task and optimality are characterised by a loss function.

$$\ell : \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}$$

Given input $\boldsymbol{x} \in \mathbb{R}^I$ and reference value $\boldsymbol{y} \in \mathbb{R}^O$ we compute the realised loss as $\ell(\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{y})$. In theory, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are a random vector $(\boldsymbol{X},\boldsymbol{Y})$ from a known distribution, we could try to seek optimal $\boldsymbol{f}$ by minimising the risk

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

However, in practice, this distribution is usually unknown. Hence, we resort to empirical methods with samples $\boldsymbol{x}^1,\ldots,\boldsymbol{x}^N$ of $\boldsymbol{x}$ and $\boldsymbol{y}^1,\ldots,\boldsymbol{y}^N$ of $\boldsymbol{y}$ for some large $N \in \mathbb{N}$. For the sake of SGD, we also consider averaging over any subset, minibatch, $B \subset \{1,\ldots,N\}$ of samples. Notice that once the architecture and the activation functions have been fixed, the network $\boldsymbol{f_\theta} = \boldsymbol{f}$ is then fully determined by the parameter vector $\boldsymbol{\theta}$ we need to optimize. So, we calculate

$$\mathscr{L}_B(\boldsymbol{\theta}) := \frac{1}{\#B}\sum_{i\in B}\ell\left(\boldsymbol{f_\theta}\left(\boldsymbol{x}^i\right),\boldsymbol{y}^i\right)$$

as the minibatch empirical risk, where $\#B$ is the number of elements in set B.

In this paper we use two types of loss function: squared loss and exponential utility function.

3.1.4 SGD and Backpropagation

Stochastic gradient descent(SGD) is a gradient descent method using over different minibatches. Consider first minimizing a differentiable objective function $F : \mathbb{R}^d \to \mathbb{R}$ over $\boldsymbol{x}(t) \in \mathbb{R}^d, t > 0$, the gradient descent method achieve the optimal solution by an iterative algorithm

$$\boldsymbol{x}(t+\eta) = \boldsymbol{x}(t) - \eta\nabla F(\boldsymbol{x}(t))$$

given initial $\boldsymbol{x}(0)$, where

$$-\nabla F = \frac{\boldsymbol{x}(t+\eta)-\boldsymbol{x}(t)}{\eta}$$

where η is the step size or learning rate. Under certain assumptions on F, which guarantee the existence of a unique minimizer and are stronger than mere convexity, it is shown in Schmidhuber (2015) that $x(t)$ tends to the minimizer as t $\to \infty$.

While this method seems to be trivial to implement, in practice, it is computationally costly to directly minimize the risk with large N. Not to mention that gradient descent may also lead to overfitting. To this end, we apply the SGD to the training of network as in Pakkanen (2019). Firstly, we randomly split the training data into k minibatches: $B_1,\dots,B_k \subset \{1,\dots,N\}$, and each minibatch contains m samples, so that $N = km$ and $\bigcup_{i=1}^{k} B_i = \{1,\dots,N\}$. Then the parameter vector $\boldsymbol{\theta}$ is updated by

$$\boldsymbol{\theta}_i := \boldsymbol{\theta}_{i-1} - \eta \nabla_{\boldsymbol{\theta}} \mathscr{L}_{B_i}(\boldsymbol{\theta}_{i-1}), \quad i = 1,\dots,k$$

with initial $\boldsymbol{\theta}$ set in advance. After that, we regard one round of training through all the minibatch as an epoch. This algorithm is then repeated over multiple epochs, each round with newly spilt minibatches, while initialising $\boldsymbol{\theta}$ with the last value of the previous epoch.

If the learning rate η were set and fixed at a rather high value in the beginning, the route of SGD might overshoot or zigzag, whereas low η could also lead to very slow convergence otherwise. To tackle this issue, several refinements of SGD have been proposed, to adaptively tune η or to seek update gradient directions better than $\nabla_{\boldsymbol{\theta}}\mathscr{L}$. Some well-known refinements of SGD include the Nesterov momentum method by Nesterov (1983) and Ilya Sutskever and Hinton (2013), RMSProp by Tieleman and Hinton (2012) and Adam by Kingma and Ba (2014), the last of which is perhaps the most renowned optimisation method in deep learning at the moment. In Adam, the parameter update direction of $\nabla_{\boldsymbol{\theta}}\mathscr{L}$ is determined by an exponentially weighted moving average (EWMA) of previous gradients, while the learning rate η is adjusted using an EWMA of squares of previous gradients.

To compute the gradient of the empirical risk $\nabla_{\boldsymbol{\theta}}\mathscr{L}_B(\boldsymbol{\theta})$, we apply a special case of algorithmic differentiation - backpropagation. This algorithm provides a way to efficiently compute the exact value of the gradient in feedforward neural network, but for chosen $\boldsymbol{\theta}$ only. Here, we only highlight the main ideas and results as in Higham and Higham (2019) and skip the proof and calculation. In short, backpropagation is based on chain rule. The minibatch enpirical risk, by linearity, can be written as

$$\nabla_{\boldsymbol{\theta}}\mathscr{L}_B(\boldsymbol{\theta}) = \frac{1}{\#B}\sum_{i\in B} \nabla \ell\left(\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i\right), \boldsymbol{y}^i\right)$$

So our aim is to work out the expressions of loss function's derivatives with respect to all the weights and biases. We now introduce some notations for presenting the results. For $\boldsymbol{x} \in \mathbb{R}^I$,

$$\begin{aligned}
\boldsymbol{z}^{\boldsymbol{i}} = \left(z_1^i,\dots,z_{d_i}^i\right) &:= \boldsymbol{L}_i\left(\boldsymbol{a}^{i-1}\right) = W^i \boldsymbol{a}^{i-1} + \boldsymbol{b}^i, \quad && i = 1,\dots,r \\
\boldsymbol{a}^i = \left(a_1^i,\dots,a_{d_i}^i\right) &:= \boldsymbol{g}_i\left(\boldsymbol{z}^i\right), && i = 1,\dots,r, \\
\boldsymbol{a}^0 &:= \boldsymbol{x}
\end{aligned}$$

so that $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{a}^r$ and $\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) = \ell(\boldsymbol{a}^r, \boldsymbol{y})$. In addition, we introduce a term called adjoint $\boldsymbol{\delta}^i = \left(\delta^i_1, \ldots, \delta^i_{d_i}\right) \in \mathbb{R}^{d_i}$ by

$$\delta^i_j := \frac{\partial \ell}{\partial z^i_j}, \quad j = 1, \ldots, d_i$$

for any $i = 1, \ldots, r$. Using the chain rule, we can derive all the derivatives by a backward recursive process with the help of adjoints.

$$\begin{aligned}
\boldsymbol{\delta}^r &= \boldsymbol{g}'_r\left(\boldsymbol{z}^r\right) \odot \nabla_{\hat{y}} \ell\left(\boldsymbol{a}^r, \boldsymbol{y}\right) \\
\boldsymbol{\delta}^i &= \boldsymbol{g}'_i\left(\boldsymbol{z}^i\right) \odot \left(W^{i+1}\right)' \boldsymbol{\delta}^{i+1}, && i = 1, \ldots, r-1 \\
\frac{\partial \ell}{\partial b^i_j} &= \delta^i_j, && i = 1, \ldots, r, j = 1, \ldots, d_i \\
\frac{\partial \ell}{\partial W^i_{j,k}} &= \delta^i_j a^{i-1}_k, && i = 1, \ldots, r, j = 1, \ldots, d_i, k = 1, \ldots, d_{i-1},
\end{aligned}$$

where $\odot$ stands for the component-wise Hadamard product of vectors. In this way, we can estimate the derivatives of empirical risk and update the parameter $\boldsymbol{\theta}$ in the SGD.

3.2 Long Short Term Memory (LSTM)

Recurrent neural networks(RNN) are based on David Rumelhart's work in 1986, and are now widely used in translation, speech recognition, image processing and other areas. What differentiates RNN from FNN is that the output of RNN is subsequently fed back to the network, which makes RNN a little more complicated than FNN. It is more straightforward to understand the structure when we unfold the RNN to FNNs.

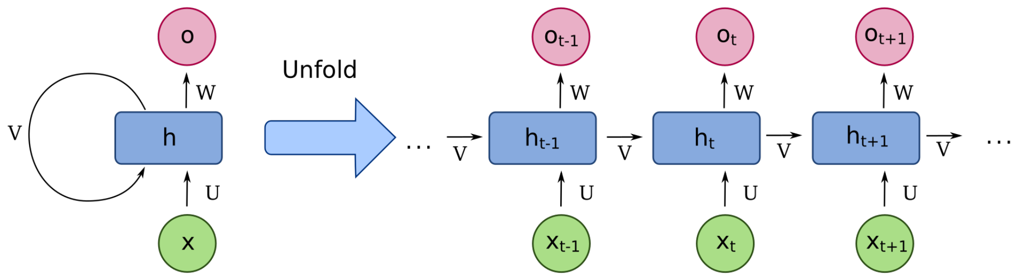

Figure 2: Recurrent neural network

We can see from Figure 2 that the loop in the structure of RNN ensures information to pass on on. When unfolding RNN, it can be thought of multiple copies of the same network, each passing a message from one step of the network to the next. So each round of training remembers the information in previous training, and this network is particular suitable for predicting something that is path dependent.

However, RNN also has its drawbacks. Gradient vanishing and exploding are very likely to occur during the backpropagation, making RNN unable to learn to connect the information(Nielsen (2015)). This issue is due to the fact that, unfolding RNN with N time steps is a very deep feedforward network that duplicates N times, and in backpropagation the gradient in early layers is the product of terms from all the later layers. When there are many layers, the gradient tends to be unstable and may eventually explode or vanish. Also, because the product that gives the gradient contains many instances of the same weight, the product goes either exponentially small or exponentially big. In other words, the fact that the unrolled RNN is composed of duplicates of the same network makes the unrolled network's "unstable gradient problem" more severe than in a normal deep feedforward network.

Having said that, long short term memory network(LSTM) - a special kind of RNN developed by Hochreiter and Schmidhuber (1997) - is able to tackle the vanishing and exploding gradient problem discussed above, that is, is capable of learning long-term dependencies. Currently, LSTM is perhaps the most famous and popular RNN and it works tremendously well on a large variety of problems. We would now give a brief introduction of it.

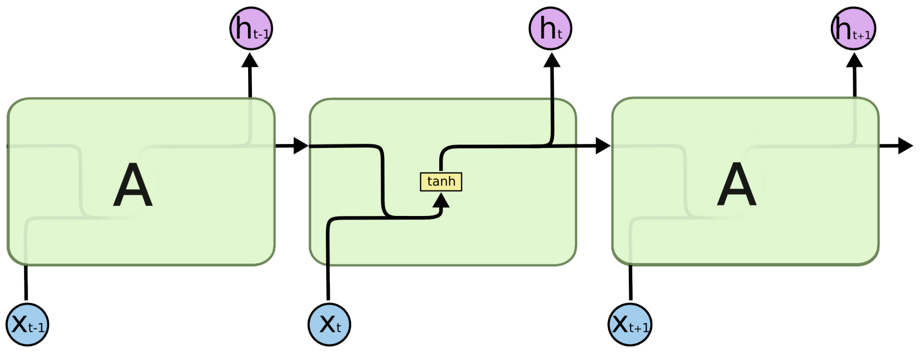

Figure 3: The repeating module in a standard RNN contains a single layer

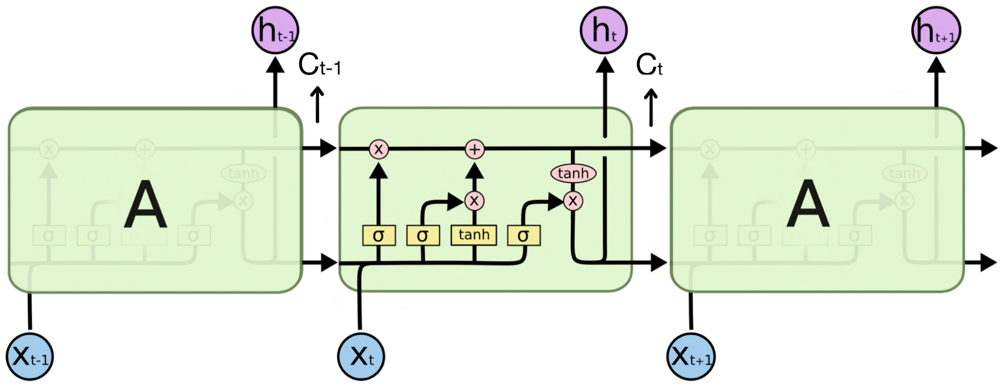

Figure 4: The repeating module in an LSTM contains four interacting layers

Figure 3 and Figure 4 give the structures of RNN and LSTM respectively, where x_t represents input and h_t stands for output. The key insight of LSTM is that we create a cell state C_t (upper horizontal line in Figure 4) - a conveyor belt that stores and updates history information and then pass it to the next step, with only some minor linear interactions. To make sure we preserve and pass the information appropriately, we introduce three gates into LSTM layer. These three gates are composite functions with the form of a ordinary layer $f(x) = \sigma(Wx+b)$, where $\sigma(\cdot)$ is *Sigmoid* activation function, W a weight matrix and b bias vector. Now we explain the details of these three gates and how they function in a simple LSTM layer.

- Forget gate layer f_t looks at h_{t-1} and x_t, and decides what information we should leave aside when entering into C_t.

  $$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

  In some cases, we want to forget useless properties of old subject.

- Input gate layer i_t also looks at h_{t-1} and x_t, and decides what new values we should store into C_t.

  $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

  Here, a *tanh* layer creates this new values as a vector $\tilde{C}_t$

  $$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  With the forget gate and the input gate, we then update the cell state C_{t-1} into C_t by combining the old and new information

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Output gate o_t again looks at h_{t-1} and x_t,

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

and decides what information we should pass on from cell state C_t to the output h_t

$$h_t = o_t * \tanh(C_t)$$

4 Deep Hedging

Having the introduction of hedging and neural network showing above, we can now present the framework of deep hedging.

4.1 Model setting

First, we set up the FNN model and begin with a simple test on European call option where we can easily compare the deep hedging strategy with Black Scholes delta hedging approach.

Deep hedging works in discrete time, so we divide interval $[0,1]$ into T subintervals with grid points $t_n = \frac{n}{T}, n = 0, 1, 2, \ldots, T$ and interval $h = \frac{1}{T}$.

Our aim is to use feedforward neural network to develop a self-financing trading strategy to hedge the payoff $(S_T - K)^+$. Such a strategy is specified by its initial wealth x_0 and position γ_t in the underlying stock at time t for any $t = 0, 1, \ldots, T$. Assuming zero interest rate and zero transaction cost, by the self-financing property, the terminal wealth of the strategy can be expressed as

$$V_T = x_0 + \sum_{t=1}^{T} \gamma_{t-1}(S_t - S_{t-1})$$

Now with x_0 and price path S_t fixed, we aim to match the V_T to the payoff $(S_T - K)^+$, that is, train neutal network whose output is γ_t so that the

$$\begin{aligned} \text{PnL} &= V_T - (S_T - K)^+ \\ &= x + \sum_{t=1}^{T} \gamma_{t-1}(S_t - S_{t-1}) - \left(S_0 + \sum_{t=1}^{T} (S_t - S_{t-1}) - K \right)^+ \end{aligned} \tag{6}$$

approaches 0 as close as possible.

By adaptedness, γ_t must be a function of the past prices $S_t, S_{t-1}, \ldots, S_0$ only. We represent trading strategy γ as a neural network, whose inputs are the available market data and output is the hedging position

$$\gamma_t = f_t(S_t, S_{t-1}, \ldots, S_0)$$

where f_t is a neural network for any $t = 0, 1, \ldots, T-1$. In fact, since S is a Markov process, we can rewrite γ as a simpler neural network $f : [0,1] \times \mathbb{R} \to \mathbb{R}$ so that

$$r_t = f\left(\frac{t}{T}, S_t\right), \quad t = 0, 1, \ldots, T-1$$

Finally, we specify a feedforward neural network in Keras as follow

$$f \in \mathcal{N}_4(2, 100, 100, 100, 1; ReLU, ReLU, ReLU, Sigmoid)$$

where we choose *Sigmoid* as the output activation function due to the financial intuition that the hedging position should be between 0 and 1. We adapt both squared loss and exponential utility function as its loss function, since the later one can achieve hedging and pricing at the same time. During the training of FNN, we set epochs = 5, batch size = 10 and choose Adam as the optimizer.

For LSTM, we specify a model containing one LSTM layer with 10 units and a output layer with *Sigmoid* function. However, we observe that if we use the same training procedure as in FNN to train the LSTM, the network converges quite slowly. To accelerate the speed of convergence, we separate the training of LSTM into two rounds. In the first round, we set epochs = 5, batch size = 50, and Adam as the optimisation method with a relatively large initial leaning rate = 0.01. Then in the second round, we change the learning rate into 0.001 and others remain the same.

For large T the price process S_t is close to the continuous Black Scholes price process, thus the hedging ratio γ_t should in theory be close to the Black Scholes delta hedging position

$$\gamma_t^{\mathrm{BS}} = \frac{\partial}{\partial S} \mathrm{C}\left(S_t, K, 1 - \frac{t}{T}\right)$$

which amounts to perfect replication, PnL= 0.

4.2 Advantages and Drawbacks

Compared to delta hedging, deep hedging is in principle model-free and data-driven. The real data in the market are very likely insufficient for training the network, however, in practice, we require a market simulator to generate the data we need. One of the common ways to tackle this is model calibration. We first need to set up a dynamic model for the price of underlying and calibrate it using market data. For instance, we intend to calibrate the Heston model with the price process satisfying

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S$$

in which the volatility v_t is a CIR process

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}dW_t^v$$

where μ is drift, $\theta > 0$ is long term mean, $\kappa > 0$ is mean reversion and W_t^S and W_t^v are two standard Brownian motions with correlation ρ. First, we use quote data of call and put option in the market to determine μ, θ, κ and ρ. Then, we simulate enough price paths and train the network. Finally, we use this network to test real data.

Another strength of deep hedging is that various market imperfections, such as trading cost, are relatively straightforward to incorporate to the framework, while in analytical method they are uaually very hard to dealt with.

5 Experiment

5.1 Data generation

We generate a price path S_t by first generating T independent $N(0,1)$ variables $W_1, W_2, \ldots, W_T$ and computing

$$S_{t+1} = S_t + \mu h S_t + \sigma\sqrt{h} S_t W_{t+1}$$

where μ is drift and σ is volatility. We use S_t to approximate the price in continuous Black Scholes model.

In the experiment, we simulate $N = 100,000$ price paths and set the original parameters

$$\mu = 0.0, \sigma = 0.5, T = 100, S_0 = 1$$

5.2 Result

We examine deep hedging on five types of options: European call option, Asian call option, digital option(cash or nothing), digital option(asset or nothing) and barrier option(up and out). Part of results can be seen from the tables below.

Table 1: Indifference price of different option using FNN and LSTM model

	Analytical / MC price	$\lambda = 0.1$		$\lambda = 0.5$		$\lambda = 1$	
		FNN	LSTM	FNN	LSTM	FNN	LSTM
European calll option	0.1974	0.1973	0.1977	0.1976	0.1978	0.1976	0.1978
Asian call option	0.1131	0.1141	0.1139	0.1143	0.1136	0.1144	0.1138
Digital option (cash or nothing)	0.4013	0.4040	0.4051	0.4082	0.4107	0.4116	0.4160
Digital option (asset or nothing)	0.5987	0.6004	0.6030	0.6041	0.6106	0.6079	0.6191
Barrier option (B = 1.5)	0.0229	0.0238	0.0232	0.0244	0.0242	0.0292	0.0254
Barrier option (B = 2.0)	0.0891	0.0908	0.0910	0.0930	0.0930	0.0956	0.0954
Barrier option (B = 3.0)	0.1701	0.1710	0.1714	0.1736	0.1750	0.1763	0.1797
Barrier option (B = 3.0) (important sampling)	0.1701	0.16158	0.1690	0.17117	0.1698	0.17504	0.1726

	Analytical / MC price	$\lambda = 5$		$\lambda = 10$		$\lambda = 15$	
		FNN	LSTM	FNN	LSTM	FNN	LSTM
European calll option	0.1974	0.1981	0.1983	0.1990	0.1991	0.1988	0.1997
Asian call option	0.1131	0.1153	0.1143	0.1160	0.1155	0.1169	0.1163
Digital option (cash or nothing)	0.4013	0.4367	0.4435	0.4625	0.4776	0.4855	0.4754
Digital option (asset or nothing)	0.5987	0.6367	0.6414	0.6625	0.7299	nan	0.7761
Barrier option (B = 1.5)	0.0229	0.0292	0.0301	0.0353	0.0363	0.0425	0.0427
Barrier option (B = 2.0)	0.0891	0.1145	0.1144	0.1293	0.1287	0.1416	0.1387
Barrier option (B = 3.0)	0.1701	0.1860	0.1941	0.1923	0.1893	0.1924	0.1924
Barrier option (B = 3.0) (important sampling)	0.1701	0.18576	0.1864	0.18576	0.1894	0.19784	0.1924

For all 7 options, the indifference prices generated by both FNN and LSTM network with risk aversion $\lambda = 0.1, 0.5, 1$ are quite close to the corresponding analytical prices, with the maximum difference less than 0.01 (barrier option when $B = 3.0$ and $\lambda = 1$), indicating that two neural network models indeed give reasonable option price in Black Scholes market.

In case samples of crossing the barrier are not sufficient for the training in up and out option with $B = 3.0$, we apply importance sampling by adding a positive drift to the price path, and a weight (Radon-Nikodym derivative) $Z_1 = \mathscr{E}(\frac{\mu}{\sigma}W_1) = \exp(-\frac{\mu}{\sigma}W_1 - \frac{1}{2}(\frac{\mu}{\sigma})^2)$ to the empirical risk

$$E[\mathrm{PnL}(S)^2 Z_1] \qquad \text{and} \qquad E[U[\mathrm{PnL}(S)]Z_1]$$

1326 sample paths cross over the barrier when there is no drift, and 5072 sample paths reach above the barrier when the drift $\omega = 0.3$. We can see that, in general, the indifference price with importance sampling are closer to the analytical price compared to the case without drift. Notice that larger drift such as $\omega = 1$ may cause abnormal price, in this case, Z_1 cannot adjust the empirical risk correctly.

However, for digital and barrier option, when $\lambda = 5$, the indifference price is evidently higher than the analytical price. For instance, the indifference price of asset or nothing option obtained from FNN and LSTM are 0.6367 and 0.6414, and the analytical price is 0.5987, the difference between the analytical and numerical results is larger compared to the case when λ is small, and this difference tends to be bigger as the trader becomes more risk averse. In addition, the loss function of asset or nothing option ends up positive infinite in FNN model when $\lambda = 15$, which means that the network fails to converge. In fact, a higher price indicates that the network cannot achieve the optimal hedge as delta hedging does and it needs more initial cash to start the trading. Also, the comparison in Figure 16, 17, 21 and 22 confirm that deep hedging ratio fails to track delta hedging ratio closely when λ is large. Adjusting the structure of the network and subdividing the time interval makes little effort to improve the pricing error, and these two approaches lead to longer training time. This pricing problem is due to the fact that the exponential utility function with large λ is quite sensitive, which makes training more difficult. Meanwhile, payoffs of digital and barrier option are not continuous, they may contain jumps, say, from 0 to 1 or 2 to 0, and this would cause instability in loss function, leading to numerically calculating error and sometimes even convergence issue during the training of the network

5.2.1 European call option

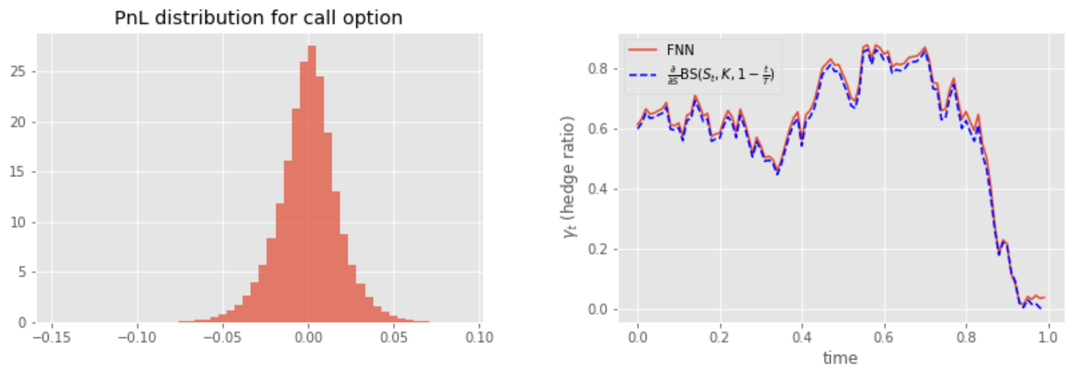

Figure 5: FNN - PnL histogram for call option and hedge ratio for a price path

When hedging call option in FNN with squared loss, the distribution of PnL is the histogram in Figure 5. The line graph on the right side shows two hedging strategies of a particular price path. We can infer that deep hedging matches delta hedging quite well in this case.

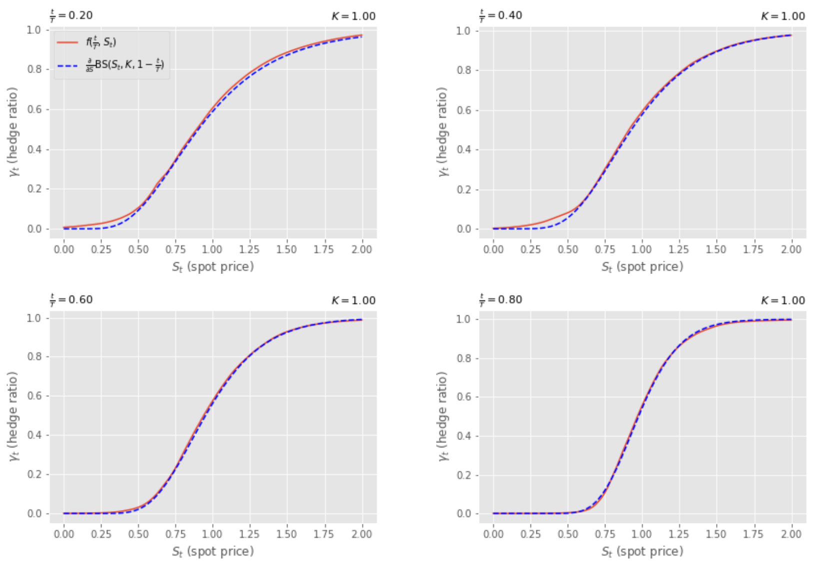

Figure 6: Comparison of hedge ratio between the deep hedging strategy and the delta hedging strategy

While deep hedging position is supposed to follow Black Scholes delta hedging position as close as possible in theory, there is a slight deviation between these two ratios when $\frac{t}{T} = 0.2$ as in Figure 6. This is due to the fact that all the price processes start at 1, and few price paths can drop below 0.5 in such a limited time, leading to lack of sample paths for training with small S_t. Important sampling may help to solve the inadequacy of samples. While similar problem happens when $\frac{t}{T} = 0.2, S_t > 0.6$, deep hedging performs quite well in other scenarios.

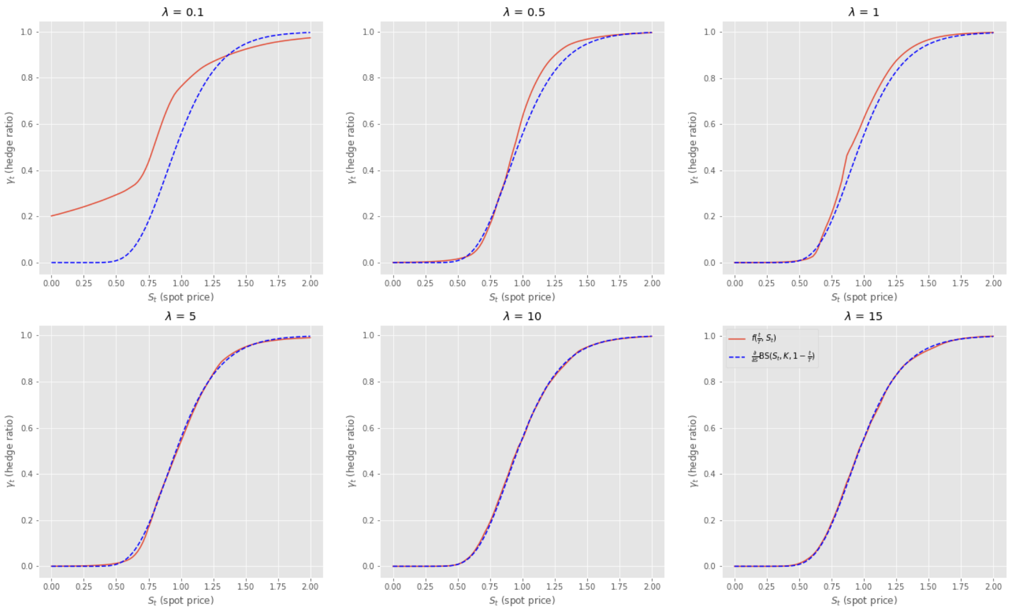

Figure 7: FNN - Comparison of strategy between deep hedging and delta hedging with different levels of risk aversion λ at time $\frac{t}{T} = 0.7$.

In order to include trader's risk aversion, we change the loss function to exponential utility function, and we can obtain both hedge ratio and indifference price by equation (3) at the same time. In each option type, we train FNN and LSTM models with risk aversion $\lambda = 0.1, 0.5, 1, 5, 10, 15$.

Risk preference λ stands for trader's tolerance of risk. Low λ indicates low level of risk aversion, whereas high λ high level of risk aversion, in which case we would seek a strategy close to Black-Scholes delta hedging. (The limit $\lambda \to 0$ approximates risk-neutrality.) We can see from Figure 7 that when $\lambda = 0.1$, the hedge ratio γ_t given current price S_t deviates from γ_t^{BS} by roughly 0.2, but they tend to be closer as λ grows. As $\lambda > 5$, they seems to have no difference, that is, deep hedging with high λ agrees with delta hedging in Black Scholes model.

Figure 8 and 9 are the the routes of hedge ratio for a particular price path with different risk aversion using FNN and LSTM. In general, low λ leads to less hedging than high λ. In particular, though the trend of hedging strategies with lower λ in LSTM model is similar to the optimal hedge, in detail they varied a lot from each other.

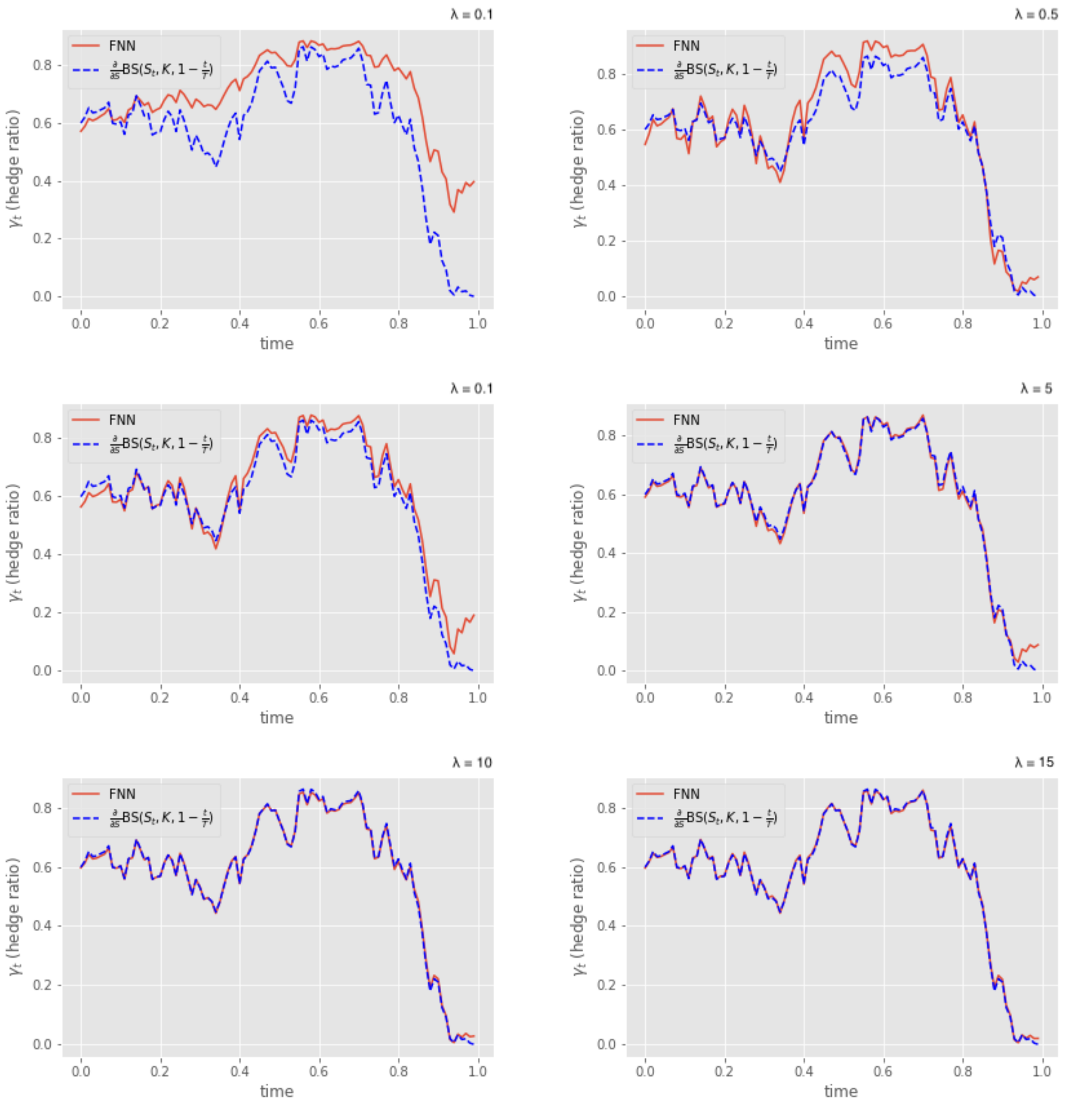

Figure 8: FNN - hedge ratio with different λ

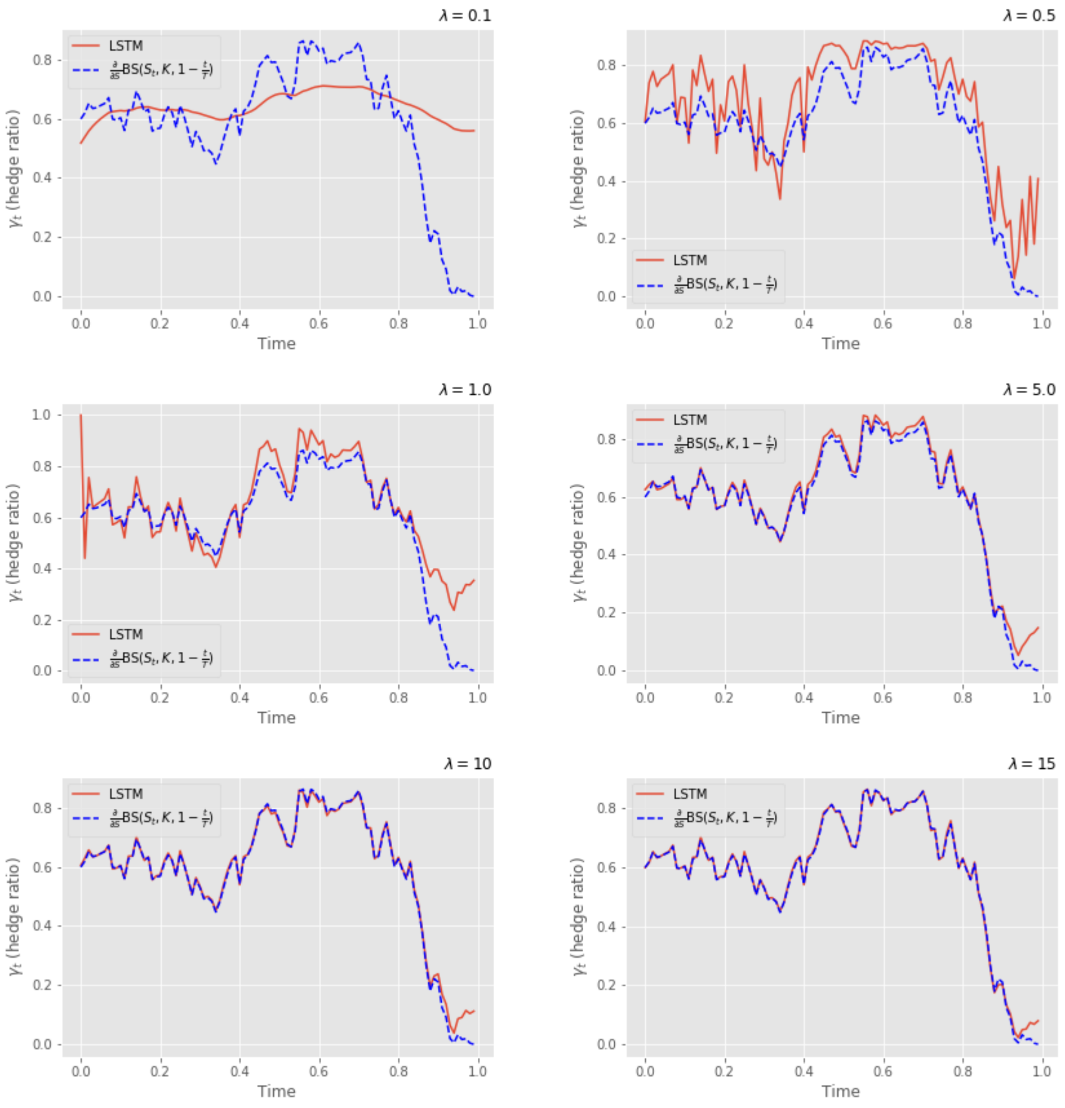

Figure 9: LSTM - hedge ratio with different λ

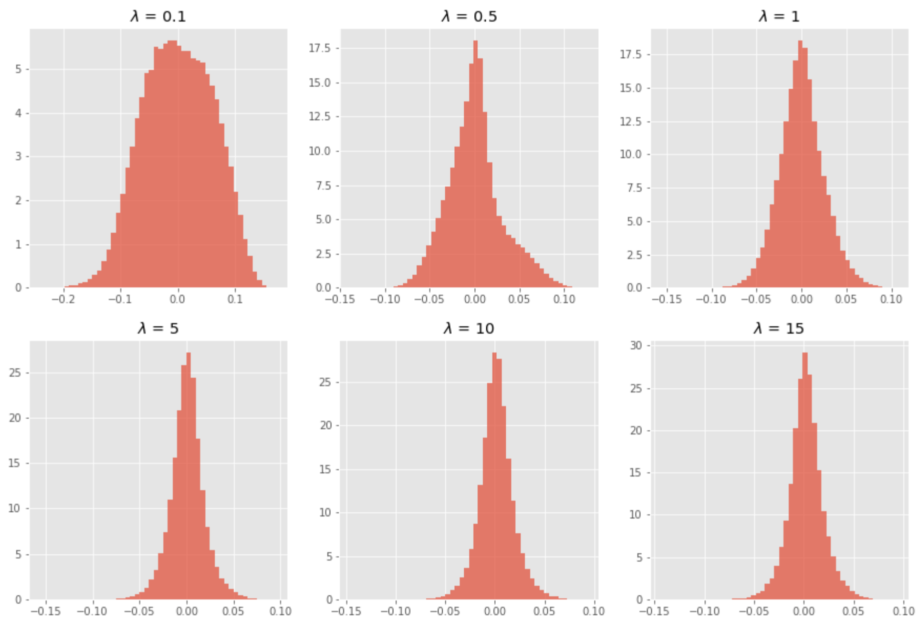

Figure 10: FNN - PnL histogram for call option with different lambda

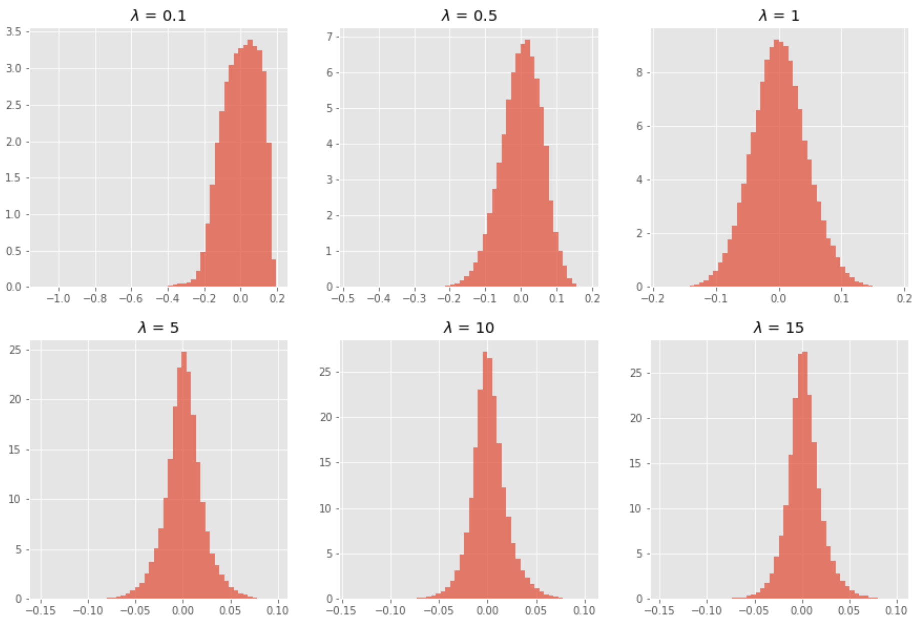

Figure 11: LSTM - PnL histogram for call option with different lambda

Figure 10 and 11 are the distributions of PnL with various risk aversions in FNN and LSTM.

Similarly, when λ is big, the trader manages to achieve full hedging, hence the distribution of PnL is consistent with Figure 6. Although the peak is also around 0 when $\lambda = 0.1$ and 0.5, these two trading strategies care less about the risk and are trying to make some profits, which leads to the asymmetry in the histograms.

5.2.2 Asian Call Option

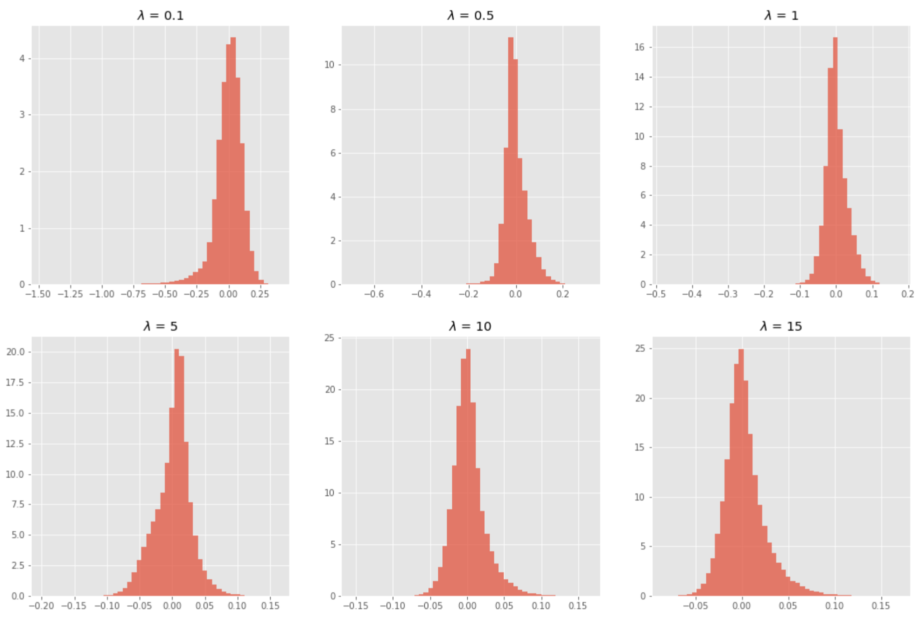

Figure 12: FNN-PnL histogram for Asian option with different lambda

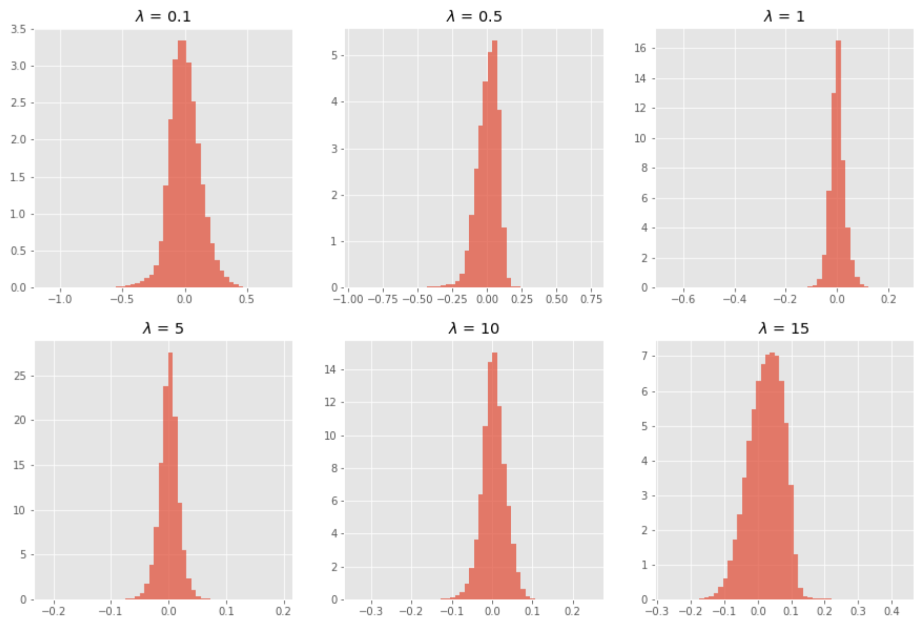

Figure 13: LSTM-PnL histogram for Asian option with different lambda

The histograms in the above show that PnL for Asian option in two models have little difference, they both peak at 0 and grow more concentrated as the trader becomes more risk averse. Additionally, from table 1, we can see that the indifference price from LSTM preforms constantly better than that from FNN model. This can be explained by the fact that the payoff of Asian call is path dependent, and LSTM can preserve the history price information by cell states, so that the learning is more effective than FNN.

5.2.3 Digital Call Option (Cash or nothing)

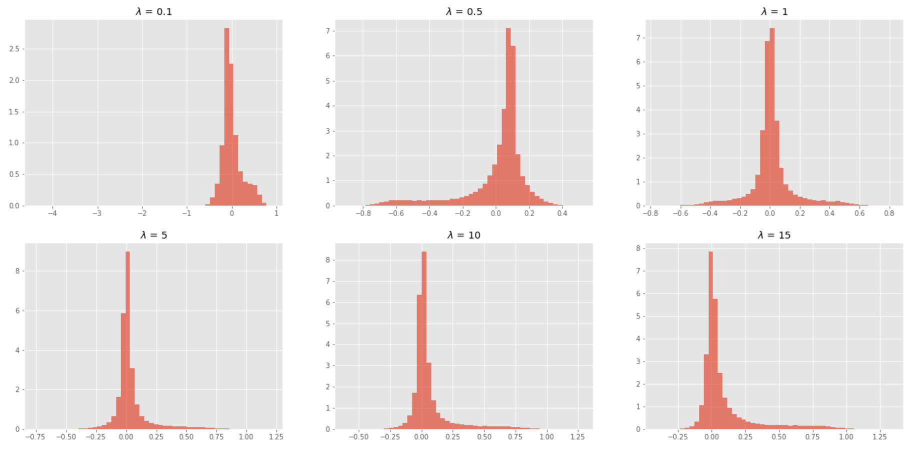

Figure 14: FNN - PnL histogram for digital option (cash or nothing) with different lambda

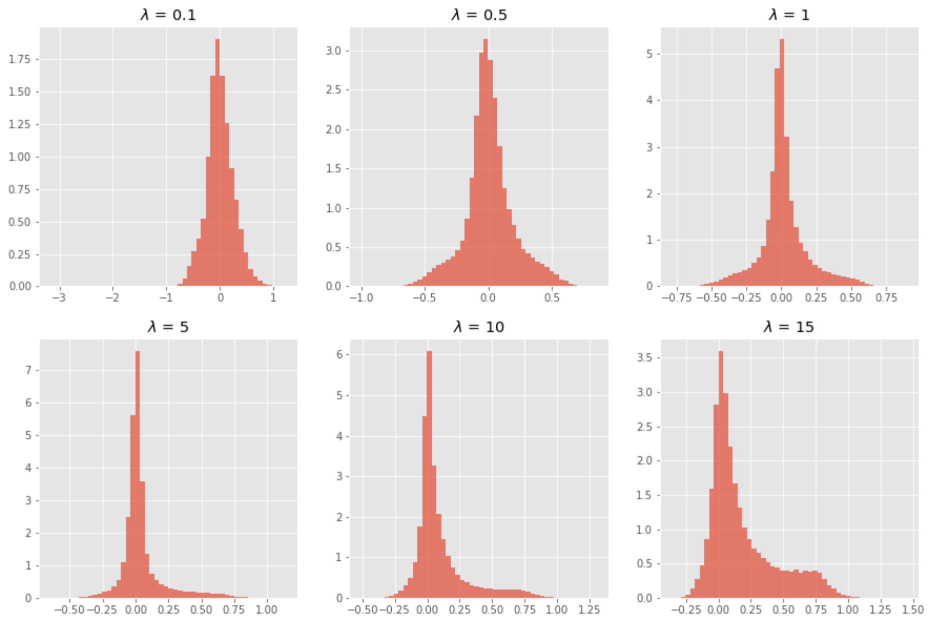

Figure 15: LSTM - PnL histogram for digital option (cash or nothing) with different lambda

When $\lambda = 1$ and 5, the deep hedging strategy using two neural networks follows the benchmark successfully. However, it deviates from the benchmark on account of the numerical error due to the large λ. Furthermore, the network tends to hedge less when trader is more risk neutral,

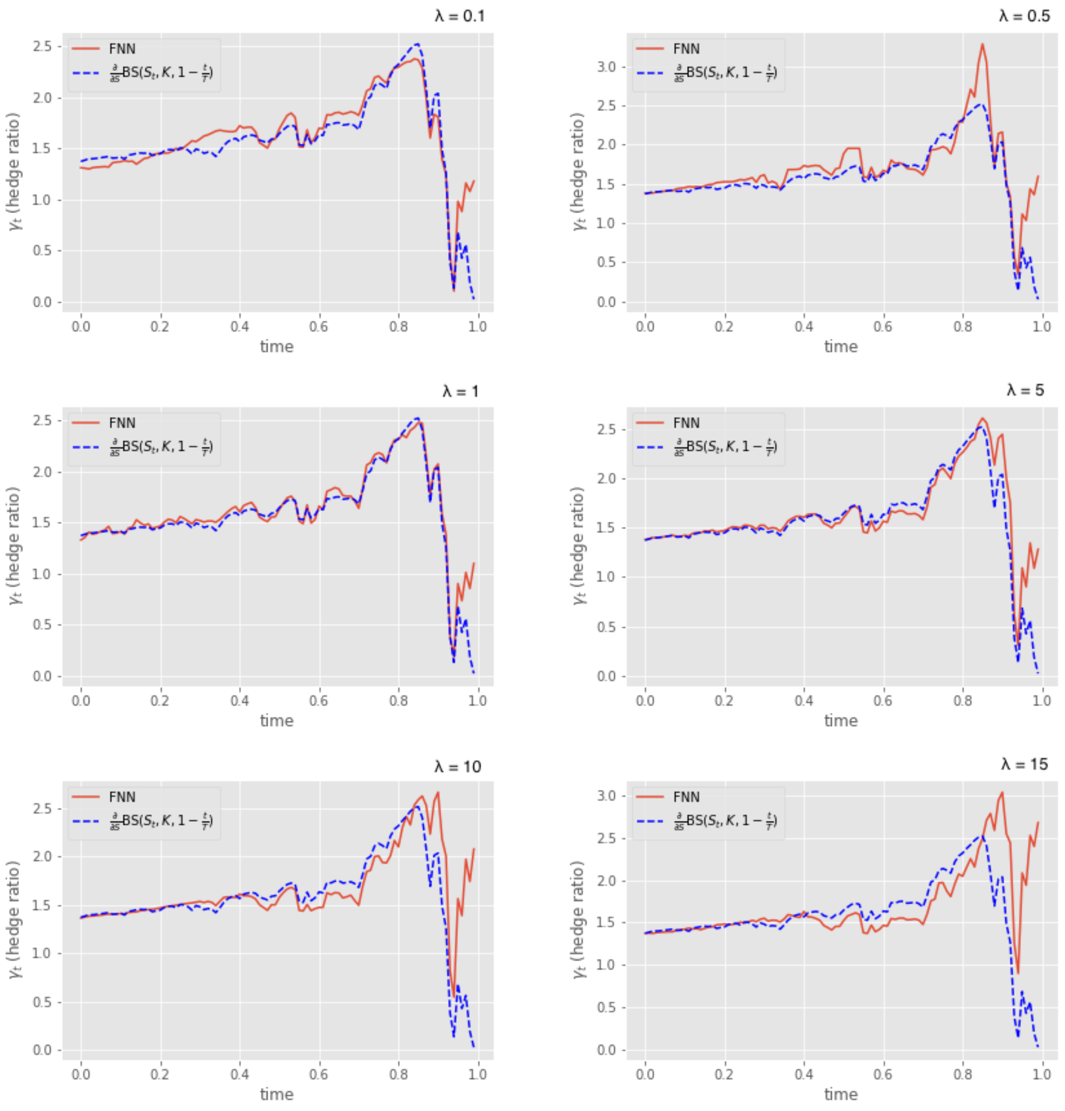

Figure 16: FNN - hedge ratio for digital option (cash or nothing) with different λ

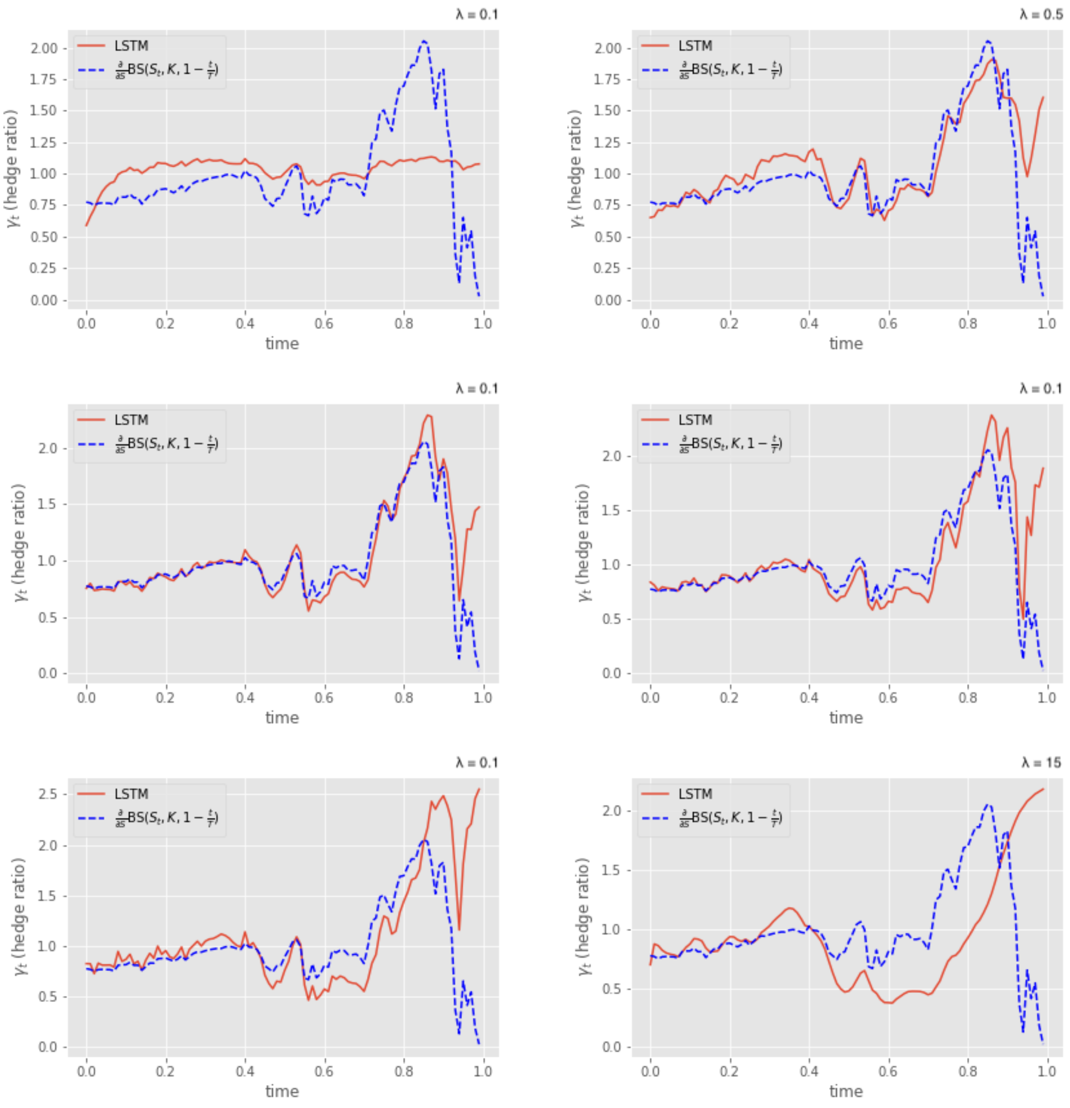

Figure 17: LSTM - hedge ratio for digital option (cash or nothing) with different λ

and this is much more notable in LSTM model, where the hedge ratio is roughly a horizontal line when $\lambda = 0.5$. In this case, the trader desire less about having 0 PnL. As a result, we observe less concentrated PnL distribution when λ is small in both Figure 15 and Figure 14. Apart from that, we can see heavy tails on the right when λ is big and some with different shape. This is due to the poor robustness of the models when facing option payoff that might includes big jumps, and in each time of running the model, the PnL distribution differ from each other.

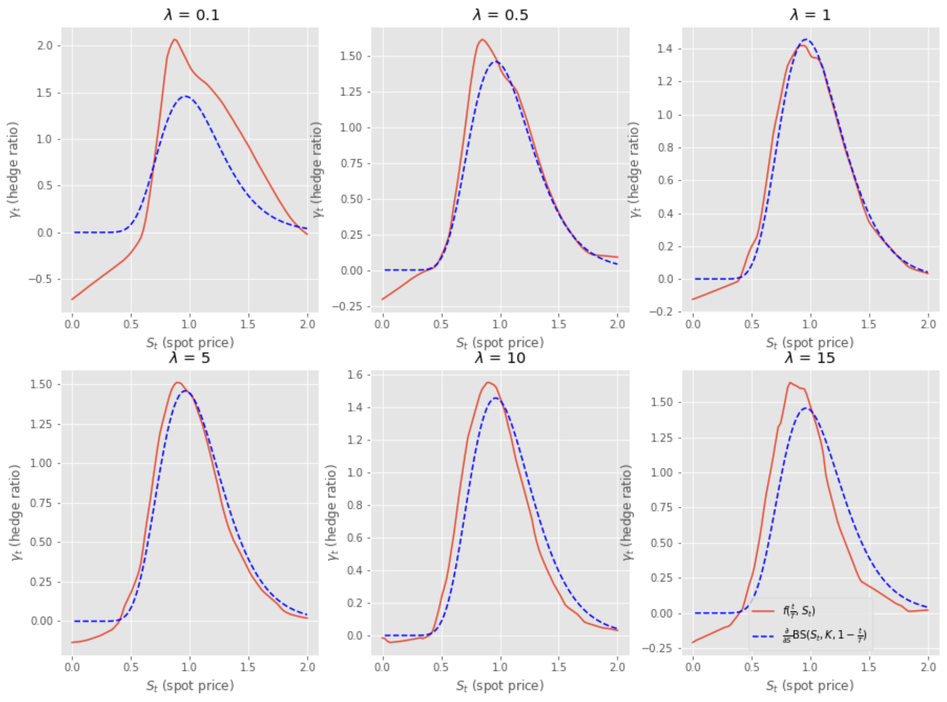

Figure 18: Cash or nothing option's comparison between FNN and delta hedging when $\frac{t}{T} = 0.7$

Since LSTM is a recurrent neural network, which depends on the entire history prices, we cannot use a single price and a particular time to predict the hedge ratio for the digital option. Nevertheless, Figure 18 is the comparison of deep hedging ratio using FNN and delta hedging ratio at $\frac{t}{T} = 0.7$, and it supports the statement regarding risk aversion above.

5.2.4 Digital Call Option(Asset or nothing)

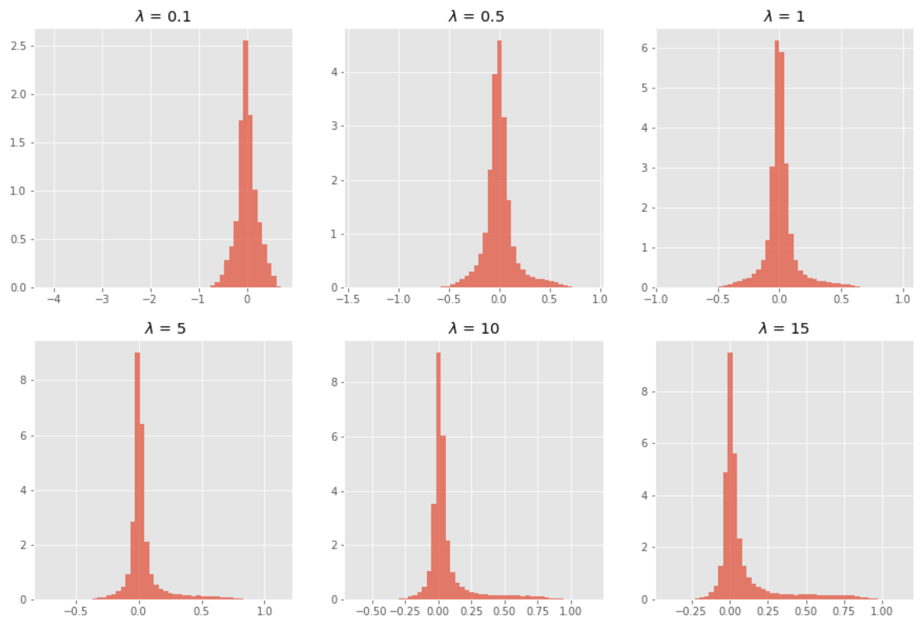

Figure 19: FNN - PnL histogram for digital asset option with different lambda

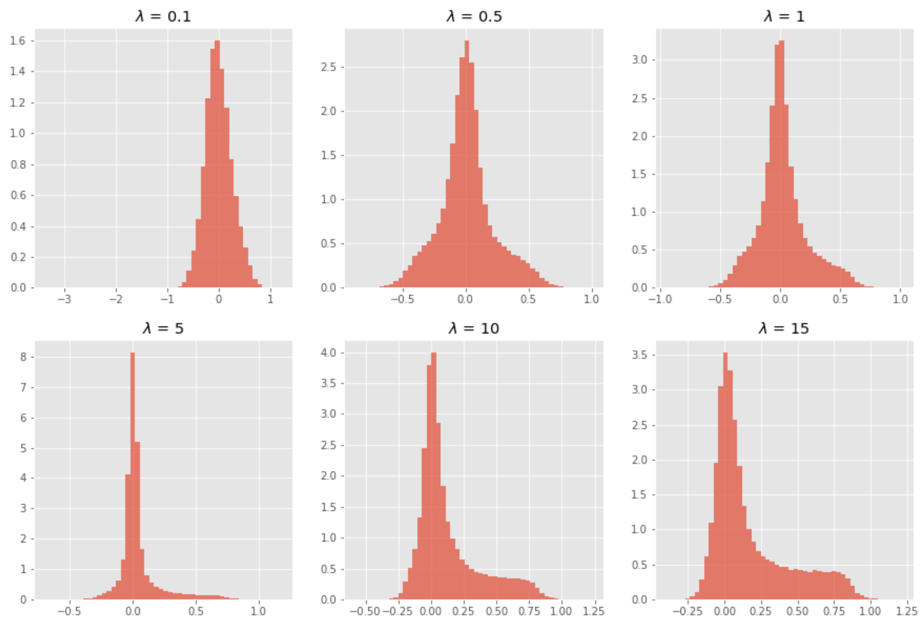

Figure 20: LSTM - PnL histogram for digital asset option with different lambda

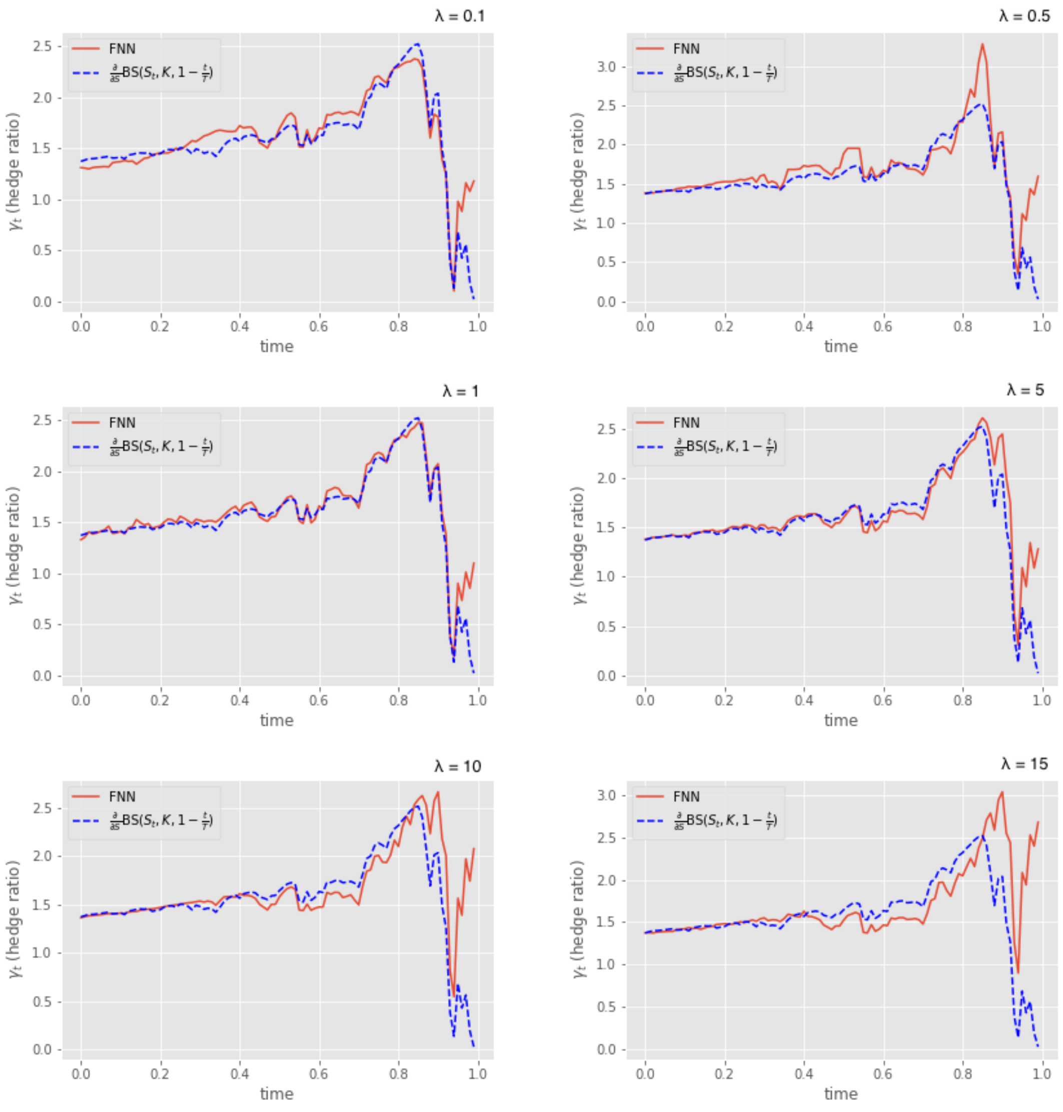

Figure 21: FNN - hedge ratio for digital option (asset or nothing) with different λ

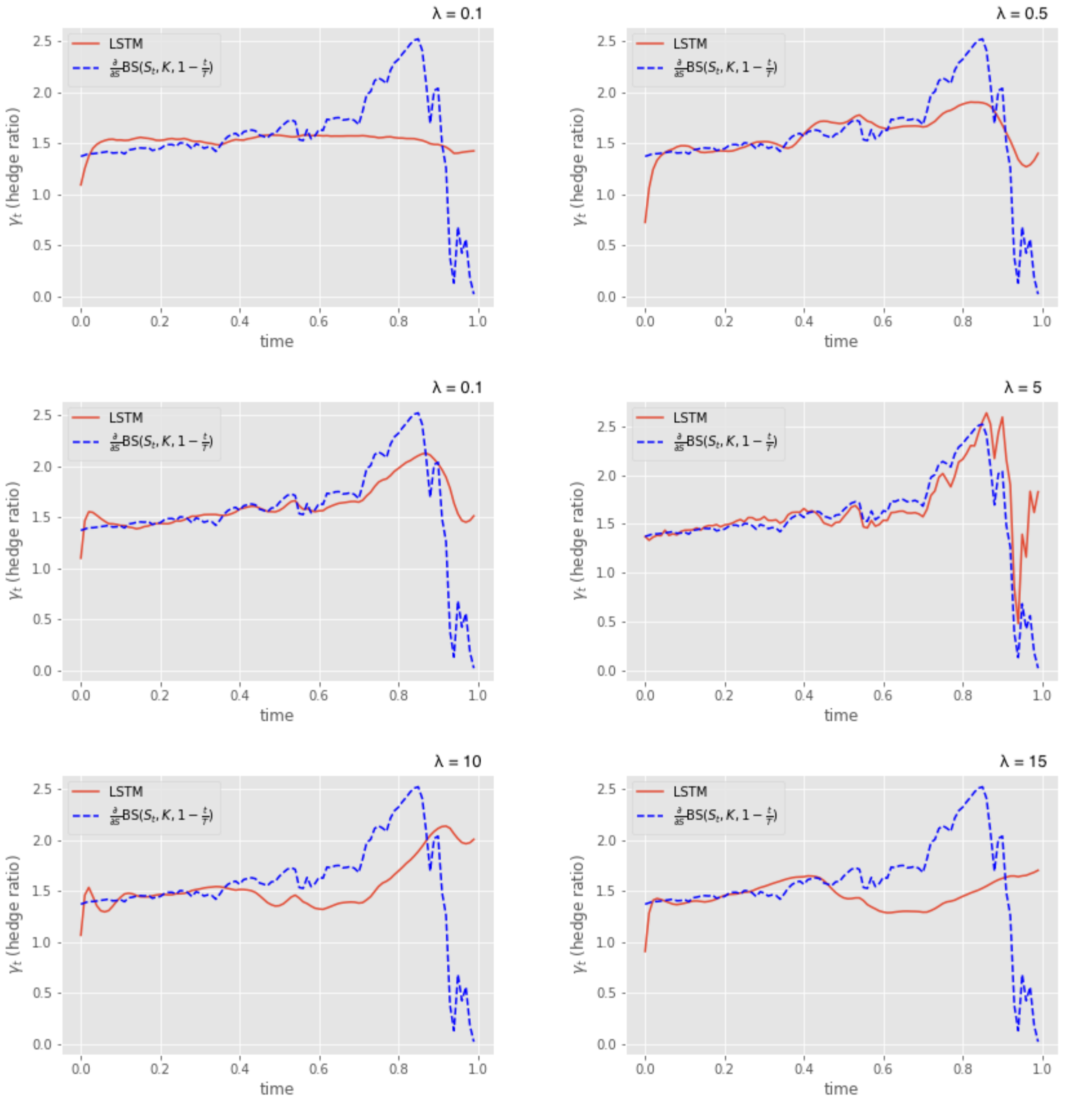

Figure 22: LSTM - hedge ratio for digital option (asset or nothing) with different λ

As for the asset or nothing option, the PnL distribution and the hedge ratios are similar to the cash or nothing option. However, the payoff of asset or nothing option is very likely to have bigger jump than the former option,which makes the training of the network less stable. As a result, LSTM model hedge far less than in cash or nothing option: It does a little hedge in the case where $\lambda = 10$ and 15, and the PnL histograms show that the hedging portfolios end up more profit than when $\lambda = 5$. In the experiment for digital option, the network sometimes goes against the risk preference due to the numerical problem.

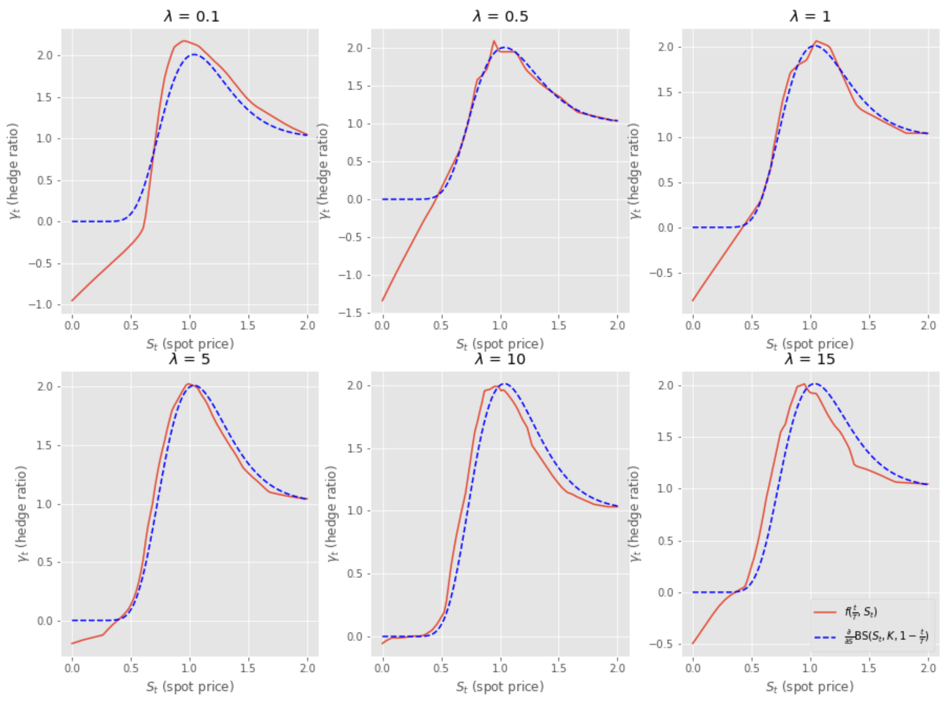

Figure 23: Comparison between FNN strat and BS digital asset

5.2.5 Barrier Option

When testing barrier option using FNN model, we add the running minimum as one of the features to track whether the barrier is crossed, whereas in LSTM, we skip this step since LSTM can reserve the past information on its own. We investigate the hedging strategies of different sample paths to illustrate the results in barrier options.

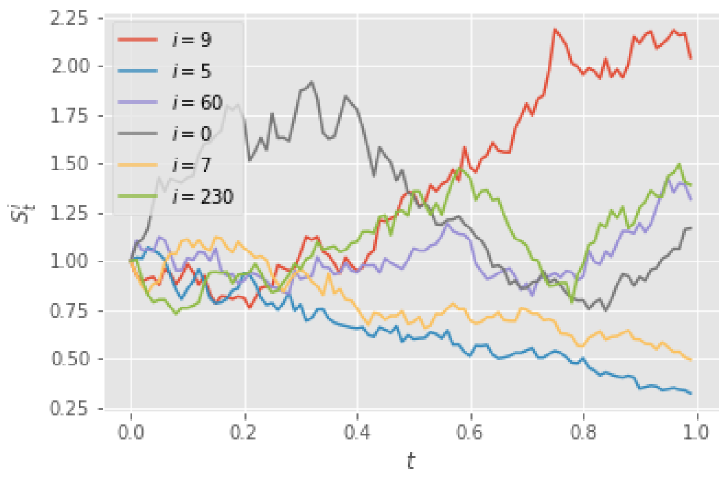

Figure 24: Several price paths

Figure 24 demonstrates 6 sample price paths which have three types of payoff:

1. Out of the money: payoff = 0, $S_T < K$ and $S_{max} < B$, e.g. No.5 and No.7.
2. In the money: payoff = $S_T - K > 0$, $S_T > K$ and $S_{max} < B$, e.g. No.60 and No.230.
3. Cross the barrier: payoff = 0, $S_{max} > B$, e.g. No.0 and No.9.

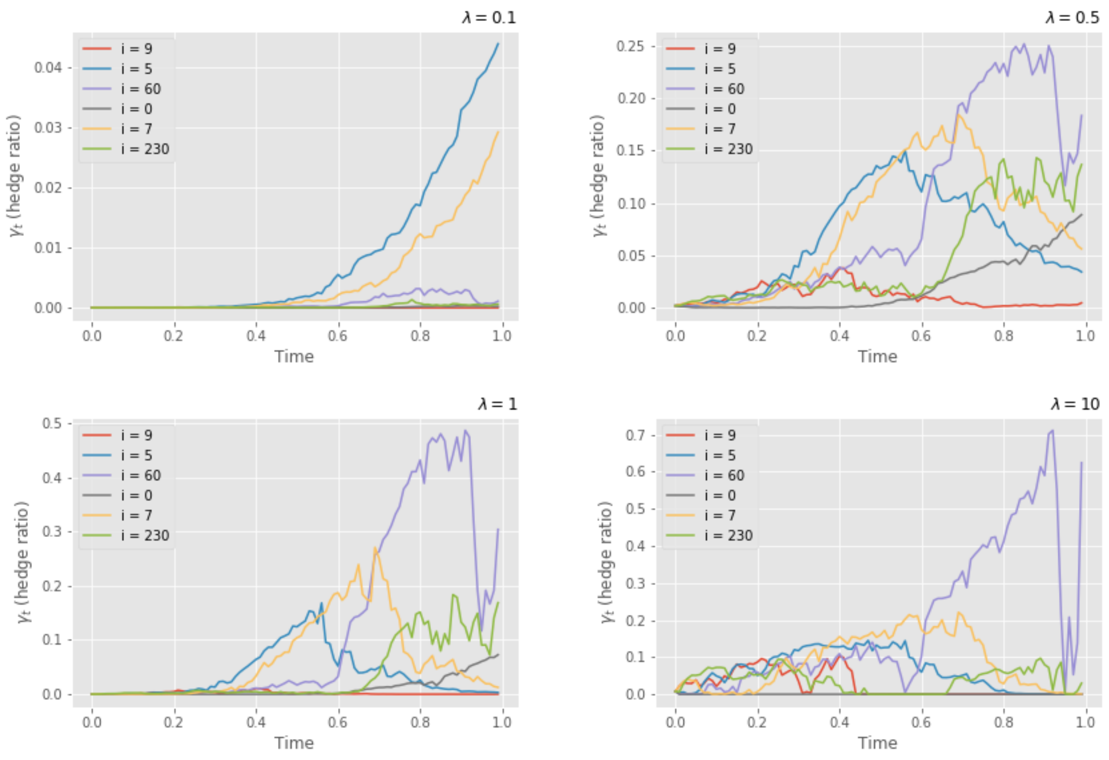

Figure 25: FNN - Barrier option's hedge ratio of several price paths ($B = 1.5$)

In the first graph in Figure 25, trader is almost risk neutral so she only trades a little amount for hedging and tries to make some profits by hedging the type 1 sample, No.5 and No.7. When $\lambda = 0.5$, the network learns that No.0 and No.9 path have crossed the barrier before expiry so they remains a low hedging level, whereas the network starts to hedge the price paths(No.60 and No.230) that may have a positive payoff. When $\lambda = 1$, as No.60 price path fluctuates between 0.75 and 1.25 before the final moment, the network tries to hedge this sample price to achieve 0 PnL, but when this price path goes near the barrier(which may indicate the jump in payoff), the hedge ratio declines dramatically on account of the sensitivity of the exponential utility function. Meanwhile, the network begins to hedge No.230 path after the path falls from the barrier at $\frac{t}{T} = 0.6$ yet maintains a lower level of ratio than that of path No.60, due to the fact that trader prefers less risk in this case. When $\lambda = 10$, the hedge ratio of sample No.60 plummets at the time when it is close to the barrier, and bounces back immediately as the price turns down. Other price paths remains a relatively low hedging ratio when the trader is very risk averse.

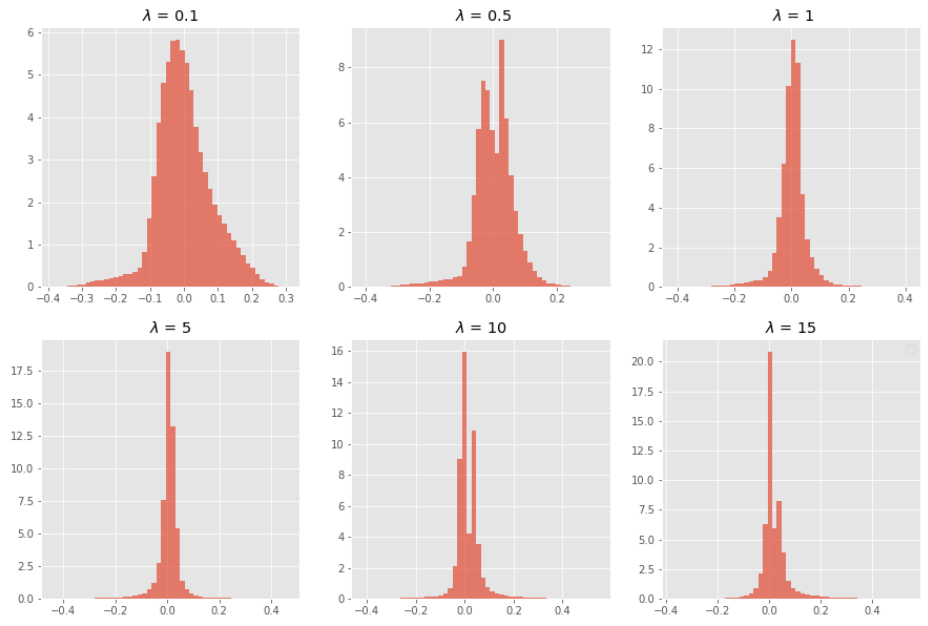

Figure 26: FNN - PnL histogram for Barrier option ($B = 1.5$) with different lambda

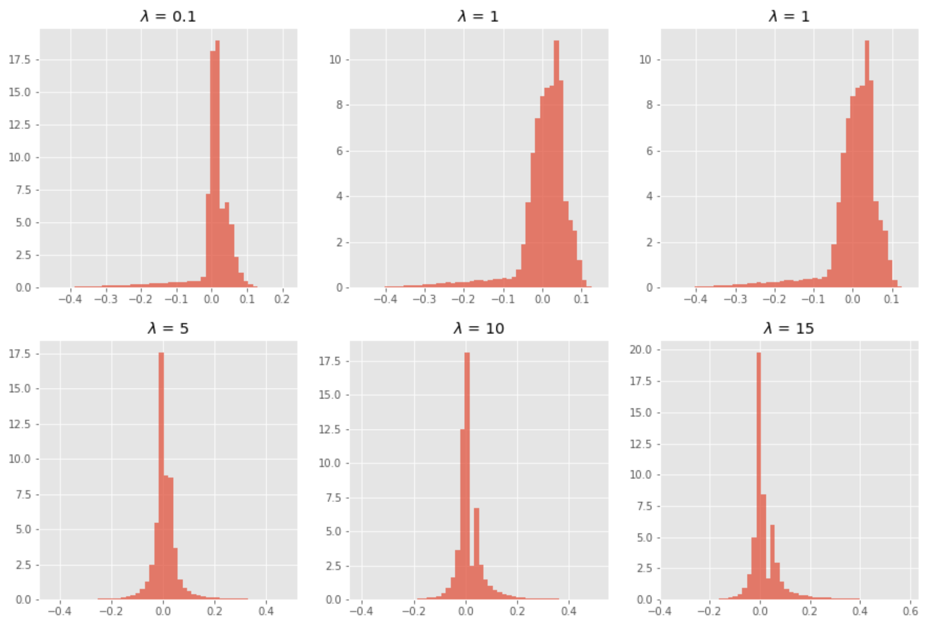

Figure 27: LSTM - PnL histogram for Barrier option ($B = 1.5$) with different lambda

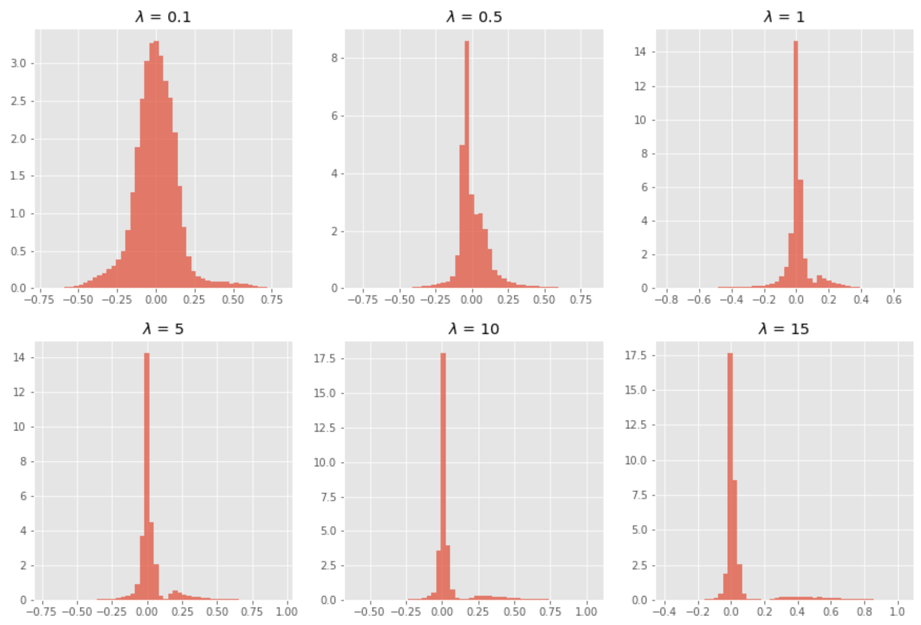

Figure 28: FNN - PnL histogram for Barrier option ($B = 2$) with different lambda

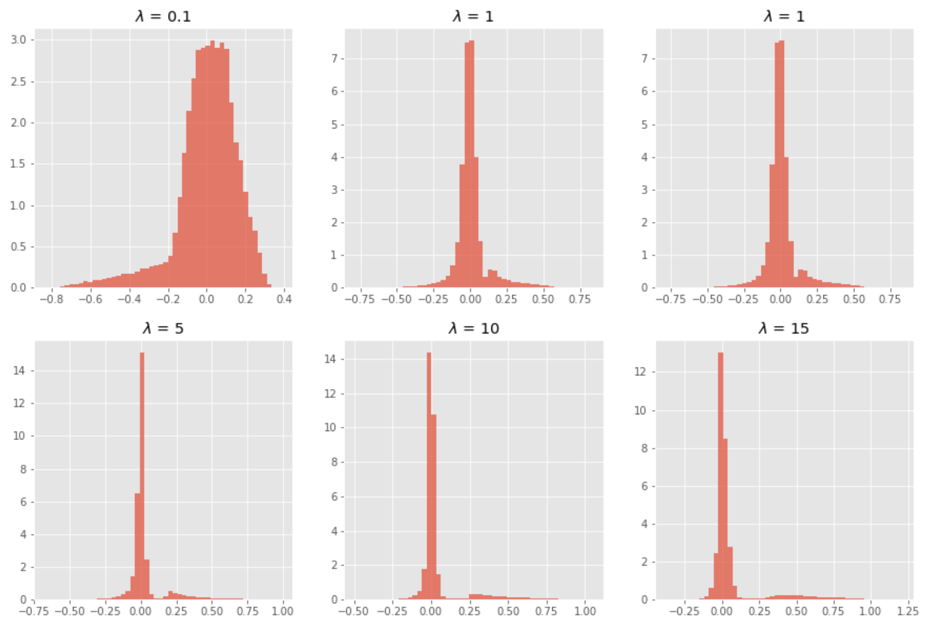

Figure 29: LSTM - PnL histogram for Barrier option ($B = 2$) with different lambda

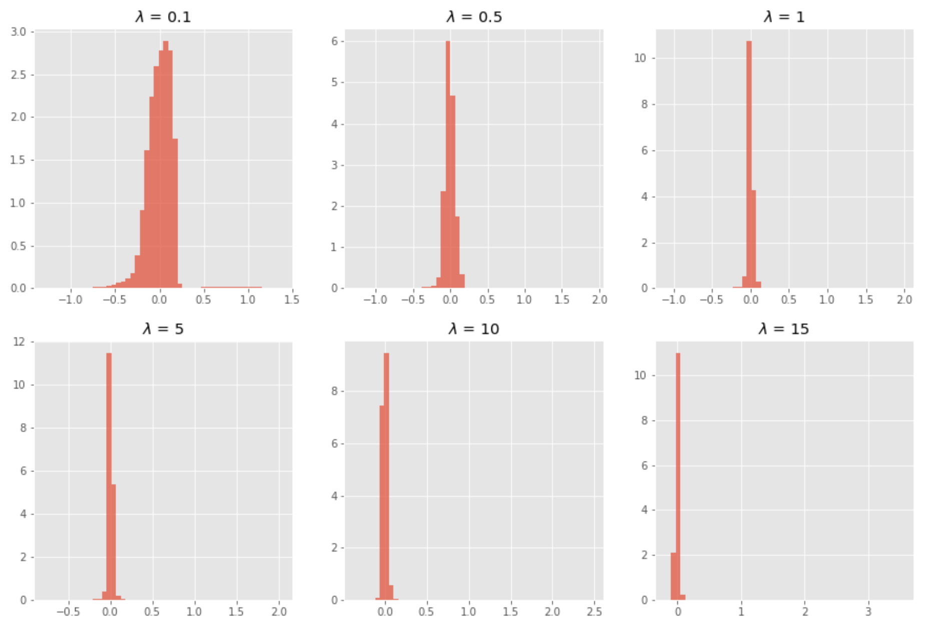

Figure 30: FNN - PnL histogram for Barrier option ($B = 3$) with different lambda

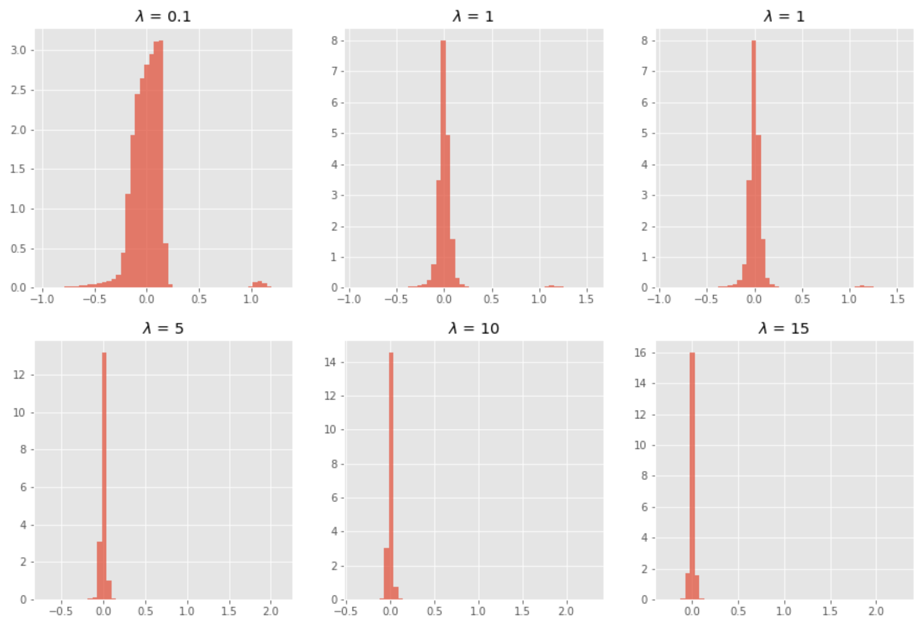

Figure 31: LSTM - PnL histogram for Barrier option ($B = 3$) with different lambda

Figure 26, 27, 28, 29, 30, 31 are PnL distributions when hedging Barrier option with various barriers using FNN and LSTM. Similar to the former PnL histograms, PnLs get more concentrated as λ gets larger. However, we can observe that there is more than one peak at 0 and most of the histograms have heavy tails on the positive side. To investigate the composition of PnLs, we split them into three parts according to the three types of price paths, and the details can be seen in Figure 32, 33, 34, 35. Out of the money: positive and negative PnLs are almost even and peak at 0, network gives up hedging the option that is unlikely to have positive payoff. Cross the barrier: most of the PnLs are positive, peak at 0.1 when $\lambda = 0.5$, and then the mode moves towards right as the trader gets more risk averse. Meanwhile, the sample number of the peak decreases. This is main reason that a flat and low tail in the histogram above moving towards bigger values. In the money: The PnLs gather gradually towards 0 as the trader getting more risk averse, network is trying to hedge the derivative as usual. These three parts of PnL combine and form the histograms above.

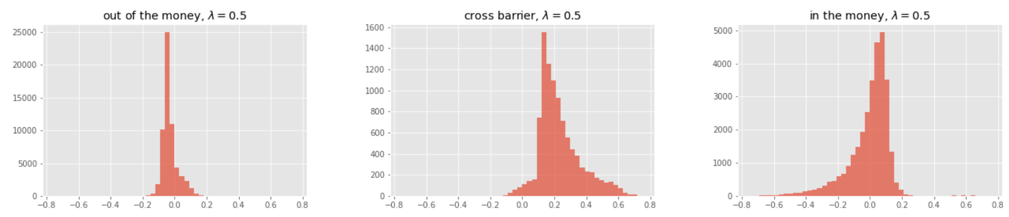

Figure 32: Separate PnL for barrier option with $\lambda = 0.5$

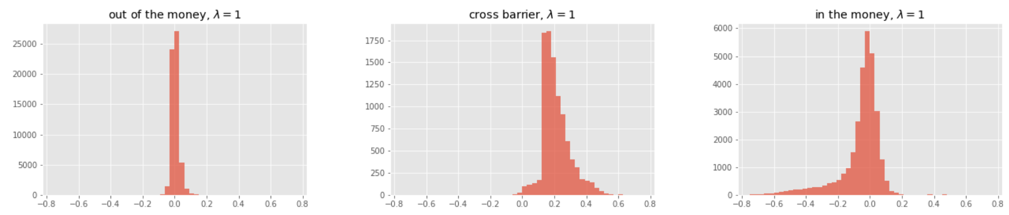

Figure 33: Separate PnL for barrier option with $\lambda = 1$

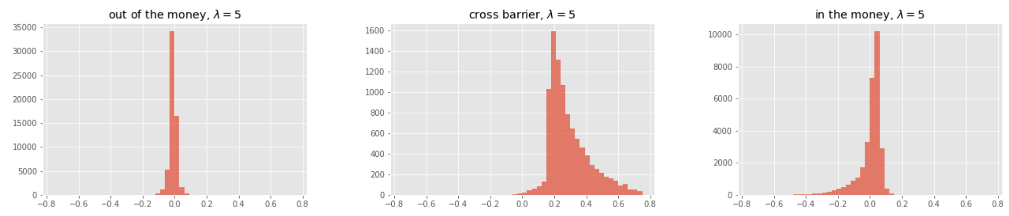

Figure 34: Separate PnL for barrier option with $\lambda = 5$

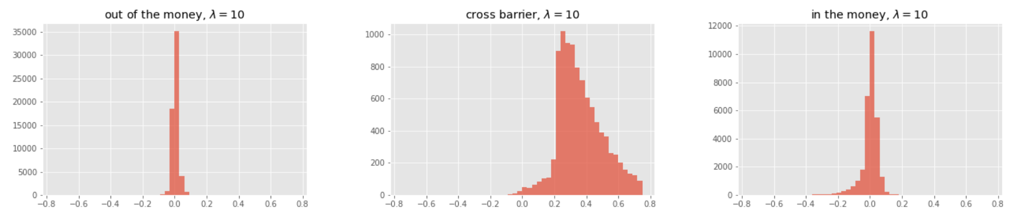

Figure 35: Separate PnL for barrier option with $\lambda = 10$

6 Conclusion

We draw several conclusions from the content above. Firstly, with proper risk preference, both FNN and LSTM network are efficient in pricing and hedging the derivatives in a Black Scholes market, and the results are in accordance with the theoretical results. Although LSTM takes longer time to train, it is suitable for path dependent derivatives such as Asian option and barrier option. Secondly, importance sampling does help when the samples are not sufficient in some price range. However, one limitation is that when the trader is very risk averse and the payoff contains jumps, the numerical problem with respect to the exponential utility function may occur like the discussion in the result of digital option, making the training of the network more challenging.

Next step one could try to apply the model to the markets with frictions and other market information such as signals and liquidity restrictions , and also change the risk measure to CVar as in Buehler et al. (2019).

References

Bank, P., Soner, H. M. and Voß, M. (2017). Hedging with temporary price impact, *Mathematics and Financial Economics* .

Bolcskei, H., Grohs, P., Kutyniok, G. and Petersen, P. (2019). Optimal approximation with sparsely connected deep neural networks, *SIAM Journal on Mathematics of Data Science* **1**(1): 8–45.

Buehler, H., Gonon, L., Teichmann, J. and Wood, B. (2019). Deep hedging, *Quantitative Finance* **19**(8): 1271–1291.

Buhler, H., Gonon, L., Teichmann, J. and Wood, B. (2018). Deep hedging, *Papers* .

Du, X., Zhai, J. and Lv, K. (2016). Algorithm trading using q-learning and recurrent reinforcement learning, *positions* **1**: 1.

Föllmer, H. and Leukert, P. (2000). Efficient hedging: cost versus shortfall risk, *Finance and Stochastics* **4**(2): 117–146.

Glorot, X., Bordes, A. and Bengio, Y. (2011). Deep sparse rectifier neural networks, *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.

Halperin, I. (2017). Qlbs: Q-learner in the black-scholes(-merton) worlds, *Papers* .

Higham, C. F. and Higham, D. J. (2019). Deep learning: An introduction for applied mathematicians, *SIAM Review* **61**(4): 860–891.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural computation* **9**(8): 1735–1780.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks, *Neural networks* **4**(2): 251–257.

Ilhan, A., Jonsson, M. and Sircar, R. (2008). Optimal static-dynamic hedges for exotic options under convex risk measures, *Ssrn Electronic Journal* .

Ilhan, A., Jonsson, M. and Sircar, R. (2009). Optimal static-dynamic hedges for exotic options under convex risk measures, *Stochastic Processes and their Applications* **119**(10): 3608–3632.

Ilya Sutskever, James Martens, G. D. and Hinton, G. (2013). On the importance of initialization and momentum in deep learning, *Proceedings of The 30th International Conference on Machine Learning* pp. 1139–1147.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization, *Computer Science* .

Moody, J. and Wu, L. (2002). Optimization of trading systems and portfolios, *Computational Intelligence for Financial Engineering*.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o(1/k^2),$Dokl.akad.naukSssr$**269**.

Nielsen, M. A. (2015). *Neural networks and deep learning*, Vol. 2018, Determination press San Francisco, CA.

Pakkanen, M. (2019). Deep learning, *Lecture notes* pp. 17–39.

Rogers, L. C. G. and Singh, S. (2010). The cost of illiquidity and its effects on hedging, *Mathematical Finance* **20**(4): 597–615.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview, *Neural networks* **61**: 85–117.

Tieleman, T. and Hinton, G. (2012). Rmsprop, *COURSERA — Neural Networks for Machine Learning, Lecture 6.5* .

Xu, M. (2006). Risk measure pricing and hedging in incomplete markets, *Annals of Finance* **2**(1): 51–71.