Computer Science

&

Data Science

Capstone Report - Fall 2023

Optimizing Derivative Hedging Strategies: Reinforcement Learning Approach and Deep Neural Networks Approach

Fifi Zhao,
Binxi Li

supervised by
Mathieu Laurière

Preface

The inspiration for this project stemmed from the persistent challenges faced in optimizing derivative hedging strategies in the dynamic stock market. There is a growing importance of integrating advanced technologies to navigate the complexities of dynamic market conditions.
This endeavor was driven by the acknowledgment of a critical gap in traditional hedging approaches—specifically, the need to consider not only the mean cost but also the standard deviation of hedging costs. The inspiration drew from the pursuit of more nuanced and effective strategies in managing derivative portfolios.
The target audience for this research spans professionals in quantitative finance, risk management, and algorithmic trading. It is particularly relevant for those seeking innovative and adaptable methodologies to optimize derivative hedging outcomes. This project holds significance for practitioners aiming to enhance their decision-making processes in derivative markets, providing them with insights into cutting-edge techniques that can significantly improve performance and risk management.
In essence, this project is vital for professionals and researchers alike, offering optimized hedging models with the utilization of neural network that aligns with the evolving demands of the financial industry. Two different approaches: Reinforcement Learning and Deep Neural Networks, coupled with a unique consideration of hedging costs, positions this research as a valuable resource for those at the forefront of quantitative finance and algorithmic trading.

Acknowledgements

We extend our sincere appreciation to all those who contributed to the realization of this research endeavor. This work has been made possible through the collaborative efforts and support of various individuals and institutions.
We express our gratitude to our colleagues and peers for their valuable insights, discussions, and feedback throughout the research process. Their contributions have played a pivotal role in refining the methodologies and enhancing the overall quality of this paper.
We would like to acknowledge the guidance and mentorship provided by Professor Mathieu Lauriere, whose expertise in Computer Science has been instrumental in shaping the direction of our research. Their unwavering support and encouragement have been a source of inspiration.
Furthermore, we extend our thanks to Professor Guo Li for providing the necessary resources facilitated the execution of this research.
This research stands as a collective effort, and we are grateful for the collaborative spirit that has enriched the journey from conception to completion.

Abstract

The area of our research focuses on optimizing derivative hedging strategies, especially when trading costs are associated with underlying assets. Derivative hedging strategies play a crucial role in mitigating risk and maximizing returns in volatile financial markets. To illustrate optimal hedging involves dynamic multi-stage decision making, we will use an innovative idea in hedging: Reinforcement Learning (RL) and Deep Neural Net(DNN). These advanced techniques offer a promising direction for derivative hedging, aligning with the dynamic nature of financial markets and surpassing traditional hedging approaches by incorporating transaction costs.

Keywords

Capstone; Computer science; NYU Shanghai; Deep Learning; Reinforcement Learning; Hedge; Financial Derivatives

Contents

1 Introduction

1.1 Context

Financial derivatives are essential tools in modern finance, allowing investors and institutions to manage risk and speculate on asset price movements. Derivative hedging strategies play a crucial role in mitigating risk and maximizing returns in volatile financial markets. Traditional hedging methods often rely on static rules and heuristics, which may not adapt well to dynamic market conditions. This research proposal aims to investigate the use of advanced techniques such as Reinforcement Learning (RL) and Deep Neural Networks (DNN) to optimize derivative hedging strategies. Reinforcement Learning (RL) and Deep Neural Networks (DNNs) can play significant roles in optimizing the hedging of financial derivatives by offering enhanced decision-making capabilities, adaptability, and improved risk management. Below are roles that RL and DNN can play in the hedging:

1. Dynamic Decision Making: RL can adapt to changing market conditions by learning optimal hedging actions over time. It allows the hedging strategy to be dynamic and responsive, adjusting to variations in asset prices, volatility, and other relevant factors.

2. Risk Management: Both RL and DNNs can help in assessing and managing risk more effectively. RL can optimize hedging strategies to minimize downside risk, while DNNs can model and predict market volatility, helping in the determination of risk levels.

3. Complex Pattern Recognition: DNNs excel at recognizing complex patterns and relationships in financial data. They can identify subtle market signals and correlations that might be challenging for human traders or traditional models to detect.

4. Enhanced Predictions: DNNs can provide more accurate predictions of future asset prices or volatility, allowing for better anticipation of market movements. This, in turn, helps in setting up hedges that are more precisely tailored to expected market behavior.

5. Reducing Transaction Costs: RL can optimize trading decisions to minimize transaction costs associated with hedging. It can determine when and how to adjust hedge positions, potentially reducing trading expenses.

6. Optimizing Hedging Strategies: RL can systematically optimize the choice of derivatives and their quantities for hedging purposes. It can find a balance between reducing risk and

maximizing returns, leading to more efficient hedging strategies. The article [1], illustrates that reinforcement learning approach outperforms delta hedging when there are trading costs.The other paper [2] focuses on hedging using improved DNNs including feature selection, weight reuse, ensemble learning, and the outcome is better than Longstaff-Schwartz algorithm when d 20. We would like to base on the two papers to reproduce and do experimental improvements.

1.2 Objective

This project is aiming to investigate and develop reinforcement learning-based and deep neutral network-based methodologies for optimizing the hedging strategies in derivatives trading. The research has a specific focus on optimizing the two approaches by addressing the impact of trading costs and variations in underlying asset price processes, in order to enhance the efficiency and effectiveness of hedging decisions. Then, by evaluating and comparing the performance of the optimized reinforcement learning approach and deep neutral networks approach across various market conditions, we can provide insights into the situations where one approach may outperform the other. And this project will be segmented into the essential phases listed below:

1. Considering the influence of asset trading costs, develop and implement a framework for optimizing derivative hedging strategies using Reinforcement Learning and Deep Neural Networks Learning and Deep Neural Networks

2. Present the result of the hedging strategy optimized by Reinforcement Learning and Deep Neural Networks for the scenarios where the derivative price follows stochastic volatility process and geometric Brownian motion.

3. Evaluate and compare the performance of Reinforcement Learning-based hedging strategies, Deep Neural Networks-based hedging strategies and traditional hedging strategies.

4. Compare that when the asset price process incorporates a stochastic volatility, the results from using a simple Black Scholes model to revalue options at each step and using stochastic volatility model to revalue options at each step.

2 Related Work

Nowadays, with the development of modern finance, hedging becomes an important tool for traders in the varying market. It derives from particular assumptions about the stochastic process governing the underlying price dynamics, and it is not only considered theoretical valuation concepts but also, more crucially for our analysis, tool for mitigating risk [1].

However, the wise selection of a hedging strategy holds such paramount importance that an inadequately designed hedging approach might overlook various trading costs and result in numerous risks, potentially leading to detrimental consequences[2]. Therefore, developing a feasible and optimized model to mitigate the risk involved in derivative trading is of great importance. This stimulates our project to develop an optimized hedging model

The most traditional model for hedging strategies is Black-Scholes model:

- Black-Scholes model: Foad and Tommi examined conditional-mean hedging within a fractional Black-Scholes pricing framework, taking into account proportional transaction costs. A precise formula for the conditional-mean hedging portfolio was developed[3]:

$$\pi_{t_i}^N = \frac{S_{t_i} e^{\hat{X}_{t_{i+1}}} \Phi(\hat{d}^+_{t_{i+1}(t_i)}) - K\Phi(\hat{d}^-_{t_{i+1}}(t_i)) - V_{t_i}^{\pi^N} + kS_{t_i}|\Delta\pi_{t_i}^N|}{\Delta\hat{S}_{t_{i+1}}(t_i)} \tag{1}$$

However, in Cao's paper, they evaluated the performance of delta hedging in both constant and inconstant stochastic volatility environments, and the result shows that delta hedging works noticeably less well in inconstant stochastic volatility environments[4]. Meanwhile, in Cao's paper, they compare the performance of delta-hedging and the reinforcement learning method that they developed, and shows that reinforcement learning method greatly outperforms the delta hedging in almost all of the environments[4]. Furthermore, in Lauri's paper, the comparison between DNN model and black-schole model delivers the same message that the neural network is superior to delta hedging with out-of-the-money options.[5]

Thus, all of the above shows that newly-developed machine learning tool can greatly improve the performance of traditional hedging models, since more factors can be included and the methods developed by machine learning can adapt to a more diverse market environment. This stimulates our project to utilized the newly-developed machine learning tool, which can taking all of the possible trading costs and influencing factors into consideration, to develop a more optimized hedging model.

DNN

With multiple hidden layers between the input and output layers, deep neural networks are enabled to efficiently encode complex and fluctuating functions with high levels of non-linearity[6]. Since Financial markets are nonlinear and dynamic, and deep neural networks are well-suited to model and capture these complex nonlinear relationships, enabling the identification of optimal hedging strategies based on historical data and real-time information. Buehler's paper developed a deep neural network to derive the hedging strategies for high-dimension American option. Each network learns the difference of the price functions between consecutive time intervals. The result shows their DNN algorithm is superior than the Longstaff-Schwartz algorithm when $d \geq 20$[7]. Meanwhile, Patrick's paper also illustrates their deep neural network model yields highly precise pricing and dynamic hedging strategies, with small replication errors[8].

Thus, all of this shows DNN is proficient in predicting price movements and assessing hedging strategies, and its performance substantially enhances when dealing with high-dimensional data. This stimulates us to develop and analyze a more optimized DNN model for hedging strategies.

3 Solution

3.1 Data

In this project, we use synthetic stock price dataset to model market data. The synthetic stock price dataset is used for both training and performance testing process.

We assume that the stock price follows geometric Brownian motion. Based on a certain initial stock price, the path of stock price can be generated. The stock price at time t is:

$$S_t = S_0 \cdot e^{(r-\frac{\sigma^2}{2})\cdot t+\sigma\cdot W_t} \tag{2}$$

where S_t is the stock price at time t, S_0 is the initial stock price (at time $t=0$), r is the drift , σ is the volatility of the stock, t is time, W_t is Brownian motion.

According to Black-Schole's, the price C of the call option at maturity time T is calculated as:

$$C = S_0 N(d_1) - Xe^{-rT} N(d_2), \tag{3}$$

where C is the call option price, S_0 is the current stock price, X is the option's strike price, T

is the time to expiration in years, r is the risk-free interest rate, $d_1 = \frac{\ln(S_0/X)+(r+\frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$, $d_2 = d_1 - \sigma\sqrt{T}$,$N(d_1 or d_2)$ is the cumulative distribution function of the standard normal distribution.

3.2 Reinforcement Learning Approach

This paper focuses on applying reinforcement learning (RL) to hedge European call Option, which involves training an agent to make decisions on how to adjust a portfolio dynamically in response to market movements in order to minimize risk. The aim of our reinforcement learning is to maximize the expected reward of the investment, and in other words minimize the expected cost of hedging. The reward can be expressed by:

$$C_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T, \tag{4}$$

where γ is discount factor, R_t is cash flow at time node t, and T is time to expiration.

Different trader tends to evaluate the reward of the asset and make decisions at different frequencies. And a series of decisions are to made by the trader at different time nodes in the stochastic changing market. Every time the trader makes the decision, it involves both the states and the actions. In our model, the states include the current stock price, time to expiration, and holding of the stock. And the sets of all possible actions are the actions of buying or selling shares of the underlying asset to adjust holding proportion. The value of every state-action pair is estimated by the Q-function $Q(S_t, A_t)$. In other words, the Q-function quantifies the quality of an action taken in a specific market condition, considering the potential future rewards.

In order to maximize the reward, an agent is involved in the process. The agent is the learning system in the RL framework, equipped with policies (π) that dictate its behavior action of selling or buying (A_t) in response to the observed states of the market environment (S_t). The agent's objective is to learn and update policy that maximizes the Q-function (the cumulative reward of trading over time). The agent's reinforcement learning process for policy update are shown in figure 1.

Let the current optimal policy be denoted as π^*, that indicates that if the market state is in S_t, the trader would choose $A_t = \pi^*(S_t) = \arg\max_{a_t \in A} Q(S_t, a_t)$. Once all state-action pairs have been studied, and the state-action pairs converges, the optimal policy at the time is the optimal policy for the overall question.

To sum up, the reinforcement learning approach in our paper involves two major stages: Q-

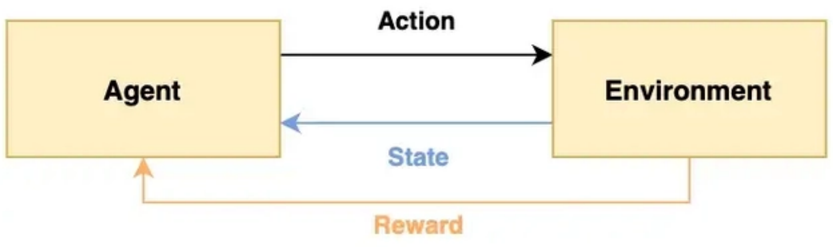


Figure 1: Reinforcement Learning Architecture

function estimation and optimal policy update.

3.2.1 Policy Update Strategy

At the beginning of the training process, since there isn't enough data for Q-function to converge, Q-function at that time could not successfully estimate the real value of each state-action pair. Thus, the optimal policy shouldn't be followed completely followed at the beginning of the training.

Thus, in our model, the developing of the learning process divides into 2 stages: exploitation and exploration. In other words, with probability $1-\epsilon$, the agent chooses the action with the highest Q-value. And with probability ϵ, the agent chooses a random action regardless of its Q-value. This process can be shown by the following equation:

$$\pi(S_t) = \begin{cases} \forall a_t \in A & \text{with } p = \epsilon \\ \arg\max_{a_t \in A} Q(S_t, a_t) & \text{with } q = 1-\epsilon \end{cases}$$

The ϵ decreases and converges to zero as the training proceeds.

3.2.2 Hedging Cost Calculation

In our paper, we quantitatively assess the rewards of hedging, in other words, the negative transaction costs of hedging through accounting $P\&L$ formulation.

More specifically, when using accounting $P\&L$ formulation, the negatives costs of hedging are calculated as

$$R_{i+1} = V_{i+1} - V_i + H_i(S_{i+1} - S_i) - \beta|S_{i+1}((H_{i+1} - H_i)|, \tag{5}$$

where S_i is the stock price at time node $i\delta t$, V_i is the value of derivative position at time node $i\delta t$, H_i is the holding of stock from i_{th}time node to $(i+1)_{th}$, and β is the transaction cost rate given by the proportion of transaction cost to the transacted value.

3.2.3 Objective Function

The key purpose of our model is to minimize of cost of hedging , which is the negative reward. In other to quantify and evaluate the desirability of the model's actions in a given environment, we introduce the objective function. The objective function $J(t)$ is defined as the expectation of the hedging cost since time node t (C_t) plus the constant λ multiplied by the standard deviation of the hedging cost since time node t. More specifically,

$$J(t) = E(C_t) + \lambda * \sigma_{C_t}, \tag{6}$$

where $\sigma_{C_t} = \sqrt{E({C_t}^2) - E(C_t)^2}$ be the standard deviation of the hedging cost since time node t. And the ultimate goal of our project is to minimize $J(0)$.

3.2.4 Q-function

Two Q-functions are introduced and updated in our model. The first one Q_1 is the Q-function for the estimation of the expectation of cost for each state-action pair. And the second one Q_2 is the Q-function for the estimation of the expectation of square cost. Thus, $J(t)$ can be rewrote as:

$$J(S_t, A_t) = Q_1(S_t, A_t) + \lambda * \sqrt{Q_2(S_t, A_t) - Q_1(S_t, A_t)^2} \tag{7}$$

And the goal of the algorithm in our paper is find the policy that can distinguish action a_t to carry out at state S_t to minimize $J(S_t, a_t)$.

The structures of how the Q_1, Q_2, π are updated and interact with each other, can be vividly shown in figure 3. And in the following part of this section, we will detailed explain this structure shown in the figure 3.

For Q_1, we utilize the Deep DPG updating algorithm. In other words, we add a parameter vector ω to Q function and utilize gradient ascent algorithm to to update a_t in direction that can fastest increase value of Q-function. The loss function for Q1 can be more specifically expressed as:

$$L_1 = (R_{t+1} + \gamma Q_1(S_{t+1}, \pi(S_{t+1})) - Q_1(S_t, A_t; \omega_1))^2 \tag{8}$$

where γ is discount rate, R_{t+1} is immediate reward at $t+1$. And the estimation of value of Q_2

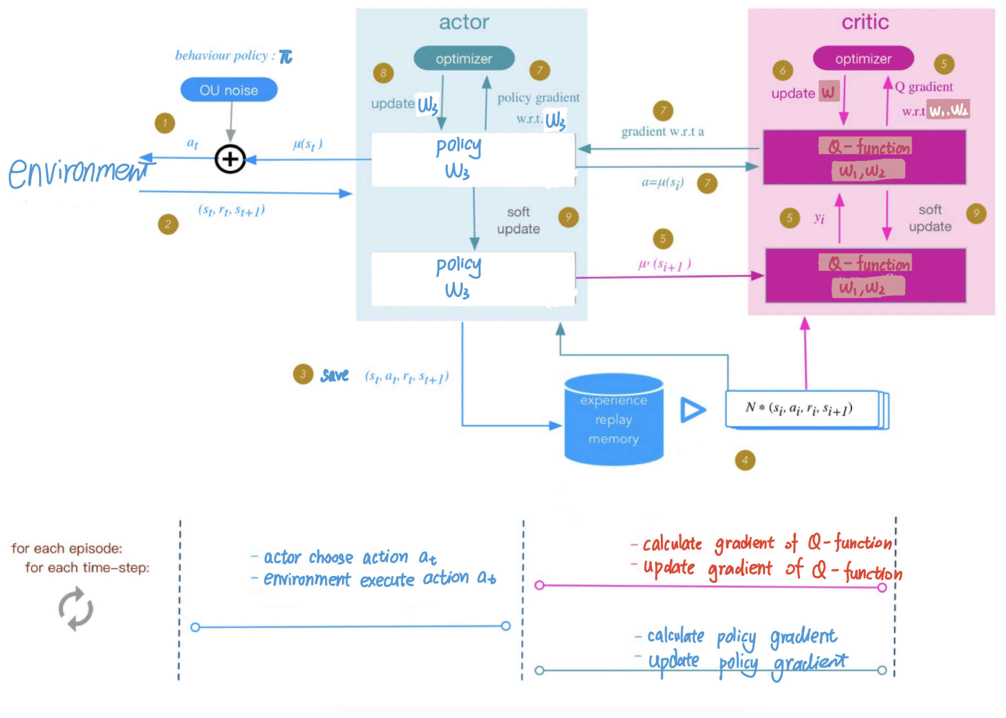


Figure 2: Updating Architecture

is updated by:

$$\omega_1 \leftarrow \omega_1 + \alpha \nabla_{\omega_1} Q_1(S_t, \pi(S_t; \theta_1)). \tag{9}$$

For Q_2, we also utilize the Deep DPG updating algorithm. If the greedy action was followed, the expected value of Q_2 equals to $E[(R_{t+1} + \gamma Q_1(S_{t+1}, a_t))^2]$. Thus, loss function for Q_2 is

$$L_2 = ({R_{t+1}}^2 + \gamma^2 Q_2(S_t, \pi(S_{t+1})) + 2\gamma R_{t+1} Q_1(S_{t+1}, \pi(S_{t+1})) - Q_2(S_t, A_t; \omega_2))^2 \tag{10}$$

where γ is discount rate, R_{t+1} is immediate reward at $t+1$.

And the estimation of value of Q_2 is updated by:

$$\omega_2 \leftarrow \omega_2 + \alpha \nabla_{\omega_2} Q_2(S_t, \pi(S_t; \theta_2)). \tag{11}$$

And finally, we will update the policy, it is also updated through Deep DPG to find the policy that minimizes (S_t, A_t). We add a parameter vector ω_3 to the policy, i.e. $\pi(S_t; \theta)$:

$$\omega_3 \leftarrow \omega_3 + \alpha \nabla_{\omega_3} J(S_t, \pi(S_t; \omega_3)). \tag{12}$$

3.3 Neural Networks Approach

3.3.1 NNHedge

For our experiments, we adopted NNHedge, which was created by [9]. It is a deep learning framework for neural derivative hedging. The advantages of NNHedge are listed below:

1. Integration with PyTorch and Numpy

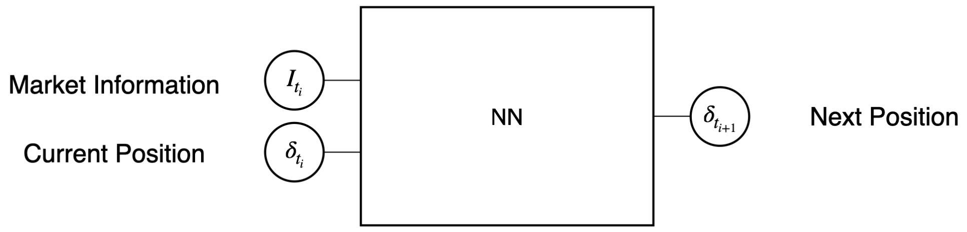
2. Seamless Pipelines:

   NNHedge provides seamless pipelines for model building and assessment. The framework simplifies the process of creating and evaluating neural networks for derivative hedging.

3. Easity of understanding NNHedge includes a variety of instruments, pre-made neural networks, data loaders, and training loops. This pre-built functionality can be advantageous for users who may not have an advanced understanding of deep learning APIs, as it relieves the necessity for such expertise.

3.3.2 Neural Approximation of Black Scholes Model

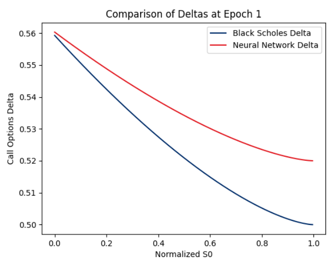
In the neural approximation of Black Scholes model, we use the Black Scholes Model delta as a label to train the neural networks. Our aim is to get approximation of the Black Scholes Delta and improve the result. It is illustrated by the following graph:



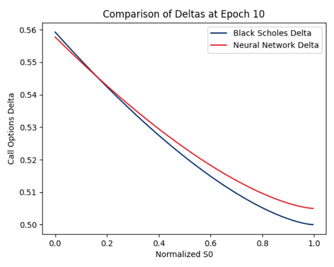
(a) A hedging strategy represented by a neural network

1. Vanilla SNN: Approximating BSM delta experiment.

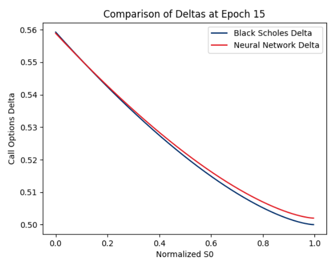
   For this experiment, we designed a Simplified Neural Network containing two linear layers and a sigmoid function as the final layer, with a total of 13 trainable parameters for the SNN. Below is a structure of our simplified neural network. Even with a small structure, our SNN fully replicates the analytical Black-Scholes delta within just 15 epochs.



(b) SNN at epoch 1



(c) SNN at epoch 10



(d) SNN at epoch 15

Figure 3: Neural Approximation of BS model

3.3.3 Neural Improvement of Black Scholes Model

Our improvement of calculation positions of underlying assets lies on the span of price data. Unlike how the Black - Scholes model uses one price data Sn to calculate Δ_n, we use short sequences $[S_{n-2} : S_n]$ called span. The length of spans are hyperparameters. We choose from 3, 5, 7 as the span in our study.

We train our neural networks to minimize profit and loss of the European call option:

$$PnL = \delta(S_1 - S_0 - 5) + C_0 - (S_1 - K)^+$$

which accounts for the transaction cost in the process of hedging. After training the models, we use Entropic Loss Measure, mean of Profit and Loss for quantitative assessment of their performance.

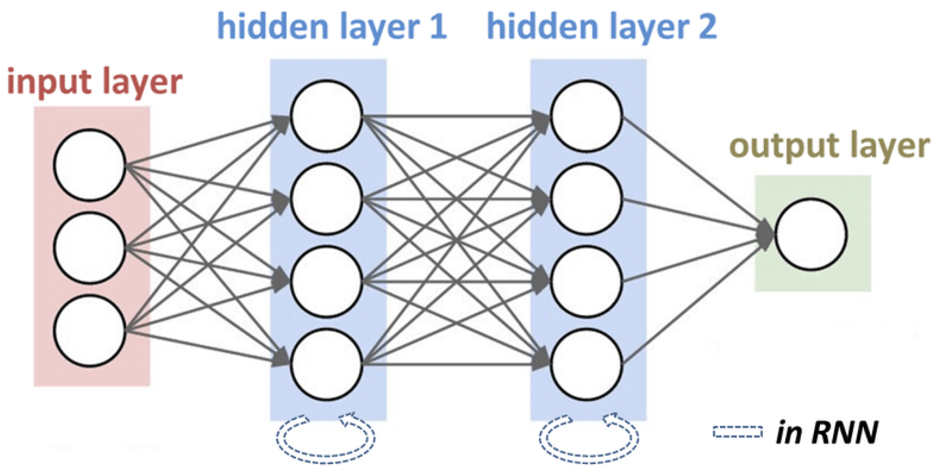
3.3.4 Neural Networks Architectures

1. RNN

   RNN stands for Recurrent Neural Network. It's a type of artificial neural network designed for sequential data. Unlike traditional feedforward neural networks, RNNs have connections that form a directed cycle, allowing them to maintain a memory of previous inputs. This recurrent nature makes them effective for tasks where context or order matters, as they can capture dependencies and patterns over time.

   Definition of RNN: Y is the output

$$H_t = \phi_h(W_{xh} \cdot X_t + W_{hh} \cdot H_{t-1} + b_h)$$
$$Y_t = W_{hy} \cdot H_t + b_y$$

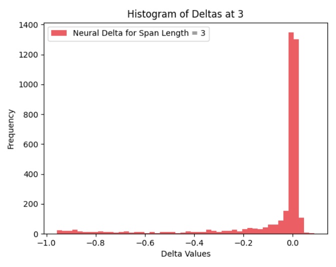


(a) Recurrent Neural Networks

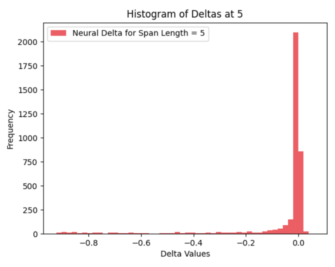
In the given definition, X_t denotes the current input at time step t and W_{xh}, W_{hh} is the weight matrix for the input and past hidden state correspondingly. At each step, past hidden state H_{t-1} is recursively included into H_t allowing RNN to trace all past hidden states $H_{0:t-1}$

We will use RNN to undergo training on the recently generated span datasets. We opt for a basic RNN structure, consisting of 3 RNN layers and a total of 166 parameters. It's noteworthy that the sequence length in our dataset falls within the range of 3, 5, 7.

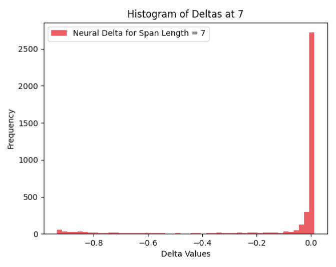
Below are histogram plots of the neurally generated delta.



(b) Span = 3



(c) Span = 5



(d) Span = 7

Figure 4: Neural Delta by RNN model

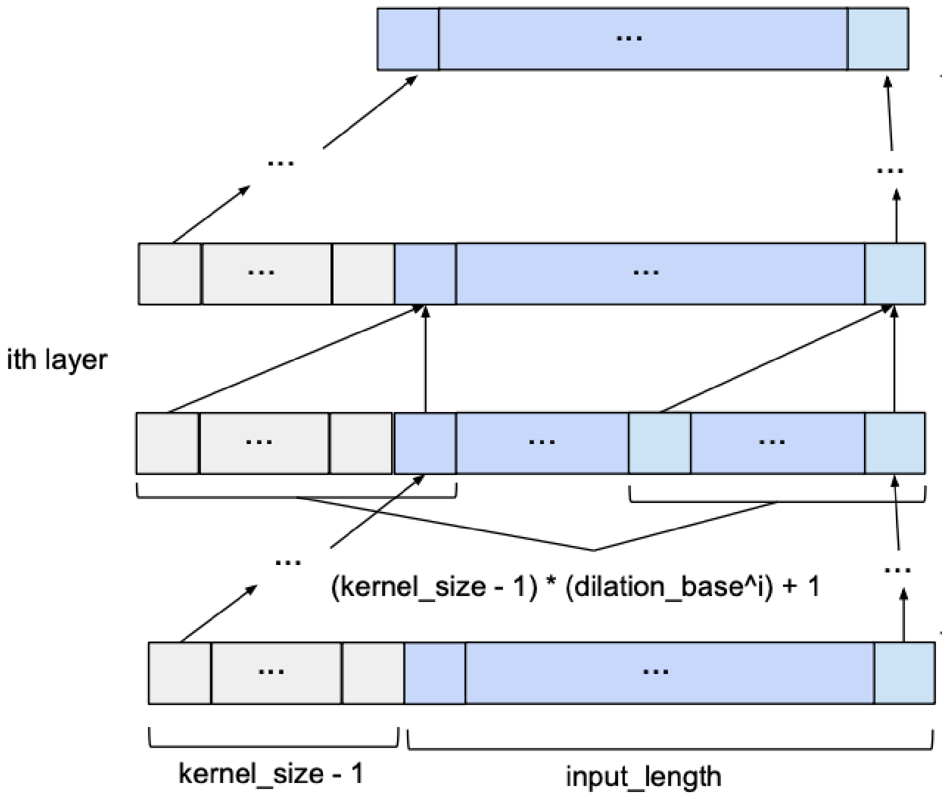
2. Temporal Convolutional Neural Networks

Advantage of TCN to RNN: In practical applications, RNN faces a serious internal design issue: due to the fact that the network can only process one time step at a time, the computation of the next step must wait until the processing of the previous step is completed. This means that RNN cannot undergo large-scale parallel processing like CNN.The TCN model is based on the CNN model and made improvement in the memory of history using Dilated Convolution (also known as Atrous Convolution) and Residual blocks, enabling the model to capture a broader context and retain historical information.

Definition:

$$H(s) = (x *_d h)(s) = \sum_{i=0}^{k-1} h(i)_{s-d \cdot i}$$

d: is the dilation factor h: is the filter function mapping indices 0,1,...k-1 to real numbers



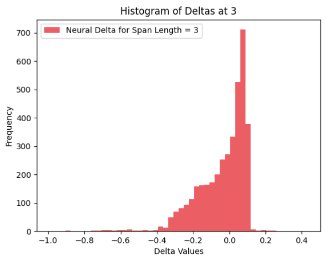
(a) 1-D Dilated Convolution

Components: Our TCN model consists of three 1-dimensional convolutional layers and a simple feed-forward network. Besides, our four transformation layers collectively have 126 trainable parameters.

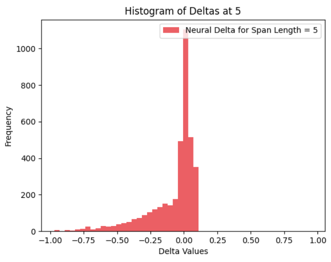
These are histograms of the neural delta by TCN model. They successfully converged the Profit and loss value to when we increase the span length.

3. Attention Based Neural Networks

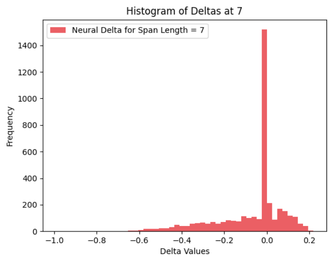
Attention mechanisms is a relatively recent neural architecture in machine learning mimic cognitive attention. They compute "soft" weights for individual words, specifically for their



(b) Span = 3



(c) Span = 5



(d) Span = 7

Figure 5: Neural Delta by TCN model

embeddings within a context window. This process can occur in parallel in recurrent neural networks. Unlike "hard" weights, which are pre-trained and fixed, these "soft" weights are flexible and can be adjusted dynamically during each runtime. In our model, self-attention calculates a priority weight for each input sequence, which teaches the neural network to concentrate on certain parts of the sequence, improving the efficiency of handling longer sequences.

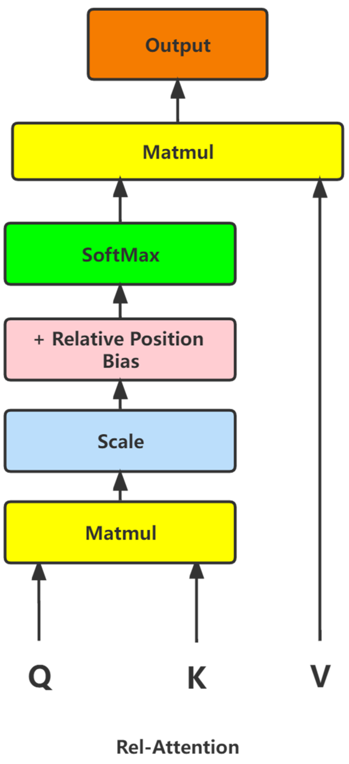
Definition: Let $X \in \mathbb{R}^{n\times d}$ and $W_K \in \mathbb{R}^{d\times d_k}$, $W_Q \in \mathbb{R}^{d\times d_Q}$, $W_V \in \mathbb{R}^{d\times d_V}$. While $Q = XW_Q$, $K = XW_K$, $V = XW_V$, self-attention S can be defined as:

$$S = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V$$

Softmax is a normalization function applied row-wise, and d representing the embedding dimension of the representation vector. The attention mechanism is unresponsive towards the sequence of the input, allowing model parallelism and faster training.

A distinctive feature of the attention mechanism is its ability to provide detailed insights into the model's weights. In our experiment, we employ back-calculation on the query layer to estimate the average significance of each sequence during the generation of the neural delta. The term "criticality" of a sequence reflects the extent to which each sequence contributes to generating the output. Specifically, when the Span Length (SL) is 3, and W_Q: $W_{SL} \rightarrow W_{10}$ for input X $\in [0, x_{SL-1}]$, the query is equation

$$\text{query} = softmax(\mathrm{W}_Q \cdot X) = softmax\left(\begin{bmatrix} W_{Q11}\cdot X_1 + W_{Q12}\cdot X_2 + W_{Q13}\cdot X_3 \\ W_{Q21}\cdot X_1 + W_{Q22}\cdot X_2 + W_{Q23}\cdot X_3 \\ \vdots \\ W_{Q101}\cdot X_1 + W_{Q102}\cdot X_2 + W_{Q103}\cdot X_3 \end{bmatrix}\right) = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{10} \end{bmatrix}$$



(a) AttentionNet Model

To obtain the multiplication of query and the value layer W_V: $W_{SL} \to W_{10}$

$$\text{output} = \omega \cdot (W_V^T \cdot X) = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{10} \end{bmatrix} \begin{bmatrix} W_{V11} \cdot X_1 + W_{V12} \cdot X_2 + W_{V13} \cdot X_3 \\ W_{V21} \cdot X_1 + W_{V22} \cdot X_2 + W_{V23} \cdot X_3 \\ \vdots \\ W_{V10,1} \cdot X_1 + W_{V10,2} \cdot X_2 + W_{V10,3} \cdot X_3 \end{bmatrix}^T \tag{9}$$

Then by calculating $\sum_{n=1}^{10}(\omega_n \cdot W_{1n},)$

one can estimate how much x_1, the first sequence of the input, contributes in generating the output.

4 Results and Discussion

4.1 Reinforcement Learning Result

We commence our analysis by training and evaluating the performance of our model using data-set 1, as detailed in the Methods section of our paper. Specifically, we operate under the assumption that the stock price adheres to geometric Brownian motion with volatility value 0.2. And we set discount rate γ as 1, transaction cost rate β as 1%.

Our examination involves the computation of key metrics for the hedging strategy produced by our reinforcement learning model for two short-term at-the-money call options. This includes the determination of the mean hedging cost, standard deviation of hedging cost, and the value of the objective function $J(0)$. In our experiment, $J(0)$ is calculated by $\lambda = 1.5$. Simultaneously, we conduct a parallel assessment for delta hedging of the same two short-term at-the-money call options.This comparative analysis serves to establish a baseline for evaluating the efficacy of the reinforcement learning model.

For the convergence of our policy, we record $J(0)$ every 1000 episodes, and we notice that $J(0)$ converges during the process of training. After 47,000 episodes, $J(0)$ arrives at the platform area, and that is the final result of our $J(0)$ in each circumstances.

The first testing scenario is for at-the-money call option with 1-month maturity time. The comparison of performance between reinforcement learning and delta hedging is shown in Table 1:

Table 1: Performance Comparison for At-the-money Call Option with 1-month Maturity Time

Frequency	Mean Cost(Delta)	Cost S.D.(Delta)	Mean Cost(RL)	Cost S.D.(RL)	$\Delta J(0)$
1 day	153%	49%	97%	62%	16.1%
2 days	119%	51%	89%	63%	6.1%
3 days	95%	52%	81%	59%	2.0%
7 days	87%	55%	77%	60%	0.3%

The first testing scenario is for at-the-money call option with 3-month maturity time. The comparison of performance between reinforcement learning and delta hedging is shown in Table 2:

Table 2: Performance Comparison for At-the-money Call Option with 3-month Maturity Time

Frequency	Mean Cost(Delta)	Cost S.D.(Delta)	Mean Cost(RL)	Cost S.D.(RL)	$\Delta J(0)$
1 day	111%	34%	67%	36%	25.3%
2 days	89%	33%	65%	35%	15.2%
3 days	78%	31%	59%	35%	10.3%
7 days	63%	35%	51%	40%	2.5%

In Table 1 and 2, $\Delta J(0)$ refers to the RL model's improvement on the value of objective function, compared with Delta hedging. It is calculated by

$$\Delta J(0) = \frac{J(0)_{Delta} - J(0)_{Rl}}{J(0)_{Delta}} \tag{13}$$

The results showcased in both Table 1 and Table 2 unequivocally affirm a marked improvement in the performance of our reinforcement learning model. This enhancement is particularly highlighted by a noteworthy reduction in the mean cost of hedging and objective function $J(0)$ across various trading frequencies and call options with distinct maturity times. While it's worth noting a slight increase in the standard deviation of the trading cost, this is overshadowed by the significant decrease in the mean cost of trading, ultimately contributing to an overall reduction in $J(0)$. The findings underscore the effectiveness of our model in fostering greater efficiency and cost-effectiveness in hedging practices.

Furthermore, it is notable that as can be shown in the tables, as the hedging frequency increases, the change in the objective function $\Delta J(0)$ exhibits a significant upswing. In essence, a higher trading frequency accentuates the reinforcement learning model's pronounced advantage over traditional delta hedging.

Meanwhile upon comparing the $\Delta J(0)$ values presented in both Table 1 and Table 2 across all trading frequencies, a discernible pattern emerges: the $\Delta Y(0)$ values exhibit a notable increase as the maturity time of the stock extends. For example, the improvement of $J(0)$ for a 1-month at-the-money call option at a 1-day frequency trading, which shows a 16.1% enhancement. Remarkably, for a 3-month at-the-money call option at the same trading frequency, the improvement escalates to 25.5%. These nuanced comparisons emphasize the increasingly pronounced advantage of our model in minimizing hedging costs with an extended maturity period.

4.2 Deep Neural Network Result

4.2.1 Deep Neural Network Model Comparison

In this section, we conduct a comparative analysis of the models mentioned earlier— RNN, TCN, AttentionNet, employing quantitative approaches. We utilize metrics such as Entropic Loss Measure (ERM), mean of Profit and Loss (PnL). The primary objective for the trained neural networks in each metric is to minimize the absolute value. The outcomes of our experiments are presented in Below. Each value in the table represents the average result of three identical models trained with different random seeds and different volatility number. We operate under the assumption that the stock price adheres to geometric Brownian motion with risk-free rate as 0.1, initial stock price 100, strike price 100. Besides, we used a train size of 100000 and a time horizon of a whole year.

Entropic Loss Measure is given by: $pu(X) = \frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda x)]$ where the risk aversion coefficient is $\lambda > 0$.

Table 3: Model Comparison

Model	Length	Eentropic Loss Measure (ERM)	Mean of Profit and Loss (PnL)
RNN	3	0.9934679	5.206425184
RNN	5	0.991345	5.130143578
RNN	7	0.99028375	5.070120747
RNN (Avg.)		0.9916988833	5.063191465
TCN	3	0.9928923	5.077245421
TCN	5	0.9927955	4.956978114
TCN	7	0.9919996	4.853005633
TCN((Avg.)		0.9925624667	4.962409723
ANN	3	0.99543333	4.813687
ANN	5	0.99396366	4.80211704
ANN	7	0.9910694	4.627855025
ANN (Avg.)		0.9931316767	4.7478863555
Black Scholes			6.937027423

From the table, we can observe that RNN model has the lowest Entropic loss measure, but with the highest cost. and Attention Net has the highest entropic loss and lowest mean of cost among neural networks.

This is attributed to the the advantages of Attention Net over TCN and RNN:

(1) Capture Long-Range Dependencies:

Attention mechanisms allow the model to selectively focus on different parts of the input sequence, enabling it to capture long-range time steps more effectively. Therefore, ANN hedging cost at span $= 7$ is the lowest. RNNs, on the other hand, may struggle with vanishing or exploding gradient problems, limiting their ability to capture dependencies across distant time steps.

(2) Effective Handling of Irrelevant Information:

Attention mechanisms can assign low weights to irrelevant parts of the input sequence, in our case, it allocates different weights of the span length contributing to the delta , effectively improving the model's robustness. RNNs may have a tendency to process all information equally.

(3) Parallelization:

Attention mechanisms enable parallelization during training, making it computationally more efficient compared to sequential processing in RNNs and TCNs. This can lead to faster training times and improved performance, especially when dealing with large datasets.

In addition, the Black Scholes Model mean of Profit and loss is higher than deep learning models even without considering the transaction costs. Hence, the cost of hedging using neural networks is lower than that of traditional hedging, meaning that deep learning methods have advantages in hedging, The reason the mean profit and loss (PnL) of the Black Scholes Model is higher than that of deep learning models might be due to the simplified assumptions inherent in the Black-Scholes framework. The Black-Scholes Model is a closed-form solution that assumes constant volatility, risk-free interest rates, and normally distributed returns. These assumptions, while providing a theoretical framework, might not fully capture the complexities and dynamics of real financial markets.

On the other hand, deep learning models, are designed to learn from historical data and adapt to the complexities of market behavior. They consider more factors and can capture non-linear relationships that the Black-Scholes Model may overlook. This adaptability and ability to handle more complex scenarios could lead to a more accurate representation of market dynamics and potentially better hedging outcomes in real-world situations.

Futhermore, all model types demonstrate enhanced performance as the span length increases, suggesting that historical data holds greater influence than current data in the generation of neural delta, since historical data often contains long-term patterns and trends that are essential for understanding the broader market dynamics. Longer spans allow models to capture and learn from these extended patterns, enabling them to make more informed predictions.

5 Discussion

Our challenges in this project and solutions are listed there:

1. Data Quality and Availability:

   Challenge: Obtaining high-quality financial data for training neural networks, especially for delta hedging, can be challenging due to market noise, incomplete datasets, or limited historical data.

   Solution: We addressed this challenge by using the geometric brownian motion under 10000, 100000 times of trias, and different volatility, interest rate to approximate the real world scenario, and meanwhile avoiding the situation of incomplete datasets and huge oscillation.

2. Overfitting and Generalization:

   Challenge: Neural networks are prone to overfitting, where the model performs well on the training data but fails to generalize to new, unseen data.

Solution: We applied cross-validation and thorough testing on out-of-sample data , ensuring the model's ability to generalize to different market conditions.

3. Dynamic Market Conditions:

   Challenge: Financial market is not perfect, there are always transaction costs, but it was not included in the traditional delta hedging method.

   Solution: We added the term - 5 delta and β representing the transaction cost in our neural network models, since in the real market situation.

If we do it from scratch, we would make following improvements:

1. Prepare Denoised Financial Real Market Data to Train:

   Employ Denoising techniques, such as advanced signal processing methods to effectively filter out noise and outliers in the financial data. This could involve exploring deep learning models specifically designed for denoising tasks.

2. Train a Larger Number of Episodes

   Due to time constraints, we halted the training process upon reaching the initial data platform. Nevertheless, there is a potential for the emergence of a second platform with additional training episodes. Consequently, we plan to conduct further training to assess the likelihood of encountering a second platform and to explore the possibility of achieving improved learning outcomes.

3. Explore Other Advanced Neural Network Models

   Investigate the integration of ensemble methods that combine predictions from multiple neural network models, which can enhance robustness and generalization, providing a more stable and reliable framework for delta hedging in dynamic financial markets.

6 Contribution

I have contributed to the neural network hedging strategies design, and participated in the Reinforcement Learning strategy design. Meanwhile, I have done literature review of those scholarly papers, and the paper writing part.

7 Conclusion

In conclusion, this paper has delved into the realm of optimizing derivative hedging strategies with the application of Neural Network. We use 2 different neural networks: Reinforcement Learning with neural network(RL) approach and a Deep Neural Networks (DNN) approach. Through rigorous analysis and experimentation, we have demonstrated the efficacy of this innovative methodologies in enhancing hedging performance when transaction cost is taken into consideration.

The Reinforcement Learning model demonstrated a notable enhancement in the hedging strategy, substantiated by a substantial reduction in the objective function $J(0)$ across diverse trading frequencies and call options with varying maturity times. Notably, our RL model plays a pivotal role in significantly decreasing the mean cost of hedging, particularly for call options with extended maturity periods and in the context of higher trading frequencies. This emphasizes its adaptive prowess and efficacy in optimizing strategies amidst dynamic market conditions.

Meanwhile, the significance of for our RL model lies in the innovative inclusion of not only the mean cost of hedging but also the standard deviation of it in our objective function. Traditionally, the emphasis has predominantly been on minimizing the mean cost, neglecting the critical aspect of risk represented by the standard deviation. By incorporating both metrics into our objective function, we bring forth a more comprehensive and nuanced evaluation of derivative hedging strategies. Our approach recognizes that a strategy with a lower mean cost might still be undesirable if it exhibits high variability, potentially leading to unexpected and unfavorable outcomes.

Simultaneously, the Deep Neural Networks approach provided a sophisticated framework for capturing complex relationships within the derivative market. Within the NNHedge framework, we explored Recurrent Neural Networks (RNN), Temporal Convolutional Neural Networks (TCN), and Attention-Based Neural Networks. The focus was on addressing the challenges posed by dynamic market conditions, transaction costs, and the need for effective risk management. The basic idea behind neural networks for hedging is to learn the δ for hedging. The RNN model outperformed TCN and AttentionNet in terms of Entropic Loss Measure with the lowest value, which demonstrates that RNN has the capacity to mitigate the risk in hedging. For mean of Profit and loss, AttentionNet demonstrated advantages in the hedging cost, due to its focusing on relevant span sequence, and enabling parallelization during training.

Both two advanced techniques showed better performance than the traditional hedging method. As we navigate the complexities of derivative markets, the findings from this study underscore the potential of cutting-edge technologies in reshaping traditional financial strategies. The presented approaches offer a glimpse into the future of derivative hedging, where intelligent learning systems and advanced neural networks work in tandem to navigate uncertainties and optimize outcomes.

This pioneering aspect of our research not only contributes to the advancement of derivative hedging strategies but also opens avenues for future investigations into the interplay between cost minimization and risk mitigation. The implications extend beyond our specific methodologies, paving the way for a more holistic and robust framework for optimizing financial strategies in dynamic and complex market environments.

Due to the limit of time, we may improve the following in the future:

1. Comparison Between RL and DNN Approaches:

   Provide a more in-depth comparison between the RL and DNN approaches. Discuss the strengths and weaknesses of each method in different scenarios. Understanding the trade-offs between these approaches can help practitioners choose the most suitable method based on their specific needs and constraints.

2. Sensitivity Analysis:

   Perform a sensitivity analysis on key parameters or assumptions made in the models. This could include varying transaction costs, market conditions, or other relevant factors to assess the robustness of the proposed strategies. Understanding how sensitive the results are to changes in these parameters adds credibility to the findings.

Direction can be extended:

1. Multi-Asset Hedging: Extend the analysis to include multiple types of derivatives or financial instruments. Investigate how the proposed neural network approaches perform in a multi-asset environment and explore potential synergies or challenges.

2. Integration with Traditional Models:

   Investigate hybrid approaches that integrate the strengths of neural network models with traditional derivative pricing models. Assess whether combining these approaches can provide a more robust and accurate framework for derivative hedging.

References

[1] R. Frey and A. Stremme, "Market volatility and feedback effects from dynamic hedging," *Mathematical Finance*, vol. 7, no. 4, pp. 351–374, 1997.

[2] G.-G. S. Fletcher, "Hazardous hedging: the (unacknowledged) risks of hedging with credit derivatives," *Rev. Banking & Fin. L.*, vol. 33, p. 813, 2013.

[3] F. Shokrollahi and T. Sottinen, "Hedging in fractional black–scholes model with transaction costs," *Statistics & Probability Letters*, vol. 130, pp. 85–91, 2017.

[4] J. Cao, J. Chen, J. Hull, and Z. Poulos, "Deep hedging of derivatives using reinforcement learning," *The Journal of Financial Data Science*, vol. 3, no. 1, pp. 10–27, dec 2020. [Online]. Available: https://doi.org/10.3905%2Fjfds.2020.1.052

[5] L. Stark *et al.*, "Machine learning and options pricing: A comparison of black-scholes and a deep neural network in pricing and hedging dax 30 index options," 2017.

[6] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks." *Journal of machine learning research*, vol. 10, no. 1, 2009.

[7] Y. Chen and J. W. Wan, "Deep neural network framework based on backward stochastic differential equations for pricing and hedging american options in high dimensions," *Quantitative Finance*, vol. 21, no. 1, pp. 45–67, 2021.

[8] S. Becker, P. Cheridito, and A. Jentzen, "Pricing and hedging american-style options with deep learning," *Journal of Risk and Financial Management*, vol. 13, no. 7, p. 158, 2020.

[9] G. Son and J. Kim, "Neural networks for delta hedging," 2021.