I

Hedging Autocallable Notes with Deep Reinforcement Learning

Julian Schady

Yuri Lauryshyn, Reilly Pickard

April 2024

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

EXECUTIVE SUMMARY

Autocallable Notes are a popular type of structured and are attractive because they offer higher returns than traditional bonds while still offering the principal protection that bonds offer (Cuim et al. 2023). This greater return comes from the fact that a standard autocallable note pays a coupon (C) if the underlying asset of the note (S) is greater than a value called the coupon barrier (CB) at an observation date (OD). An autocallable note also gets automatically called if the underlying asset is greater than a value, which is called the autocall barrier (AB), at a given observation date. If the autocallable note is automatically called at an observation date or the note reaches its maturity date, the investor will receive the face value of the note. The structure of the autocallable note's payoff leads to difficult problems within the area of risk management. The Delta, which is the change in the price of the autocallable note with respect to the change in the price of the underlying asset and is a measure used for hedging of autocallable note, can increase to large values when the underlying asset approaches the barriers because of the discontinuous nature of the payoffs. The Gamma, which is the change in the notes Delta with respect to the change in the underlying asset, around the barriers also increases and decreases to large and small amounts because of the movement of Delta around the barriers. Delta and Delta-Gamma Neutral dynamic hedging is difficult because of these barriers and can lead to a large standard deviation in profit and loss distributions, as large amounts of the underlying or hedging option may be bought when hedging near the barriers. Therefore, strategies that address this issue would be valuable as they would lead to more optimal hedging. Dynamic Delta and Delta-Gamma hedging can be framed as a sequential decision-making process within a stochastic environment. This type of problem is able to be addressed by an area within Machine Learning called Reinforcement Learning. This paper uses a Deep Reinforcement Learning (DRL) algorithm called Deep Deterministic Policy Gradient (DDPG) to Delta and Delta-Gamma hedge two different autocallable notes. Three experiments were conducted in which a DDPG model was trained to either Delta hedge or Delta Gamma hedge an autocallable note. Each trained model was then compared to the baseline Delta Neutral or Delta-Gamma Neutral model. This comparison was done using each model's Profit and Loss distribution over many hedging trials. The results of the experiments show the viability of using DRL to hedge an autocallable note. Another experiment was done to explore the importance of hyperparameter tuning when training a DDPG model. In this experiment the optimal hyperparameters for a DDPG Delta-Gamma

hedging model were found and compared to the baseline hedging model. Future work is suggested to explore the use of different DRL algorithms, hedging different autocallable note structures, using different underlying processes, and using different hedging options when Delta-Gamma hedging.

ABSTRACT

This paper utilizes deep reinforcement learning (DRL), specifically the deep deterministic policy gradient (DDPG) algorithm, to dynamically Delta hedge and Delta-Gamma hedge two types of autocallable notes. The autocallable note has discontinuous payoffs due to the presence of barriers. This discontinuous payoff leads to difficulties in risk management. Specifically, the Delta and Gamma of an autocallable note around the barriers can increase and decrease to large amounts. When Delta and Delta-Gamma neutral hedging these large values cause large purchasing of whatever is being used to hedge. To address the difficult hedging problem, DRL was utilized to learn viable Delta and Delta-Gamma hedging strategies for two types of autocallable notes. The DDPG model was trained and tested with asset paths generated using Geometric Brownian Motion (GBM). It was found that the DDPG model had a smaller standard deviation in its profit and loss distribution compared to the regular Delta neutral and Delta-Gamma neutral baseline model. The importance of hyperparameter tuning was also explored. The optimal hyperparameters of a DDPG model to Delta-Gamma hedge a real-world autocallable note structure was found. The implication of these results shows the viability of using DRL in dynamically hedging autocallable notes. It also shows the value of continual research and exploration into this topic.

ACKNOWLEDGMENTS

I sincerely thank my supervisor, Yuri Lauryshyn, for his direction and guidance throughout the year. I have been introduced and learned many new concepts while working on this thesis. The weekly meetings with Yuri gave me great insight and motivation for this project, and I am very grateful.

I would also like to thank Reilly Pickard for his guidance and assistance on this project. Reilly helped me so much in setting up a DDPG agent, debugging it, and tuning hyperparameters. I have learned a lot from his assistance this year, and I am incredibly grateful for his advice.

I would also like to thank Julio DeJesus and Finn Wredenhagen for their insights and guidance throughout the year during our weekly meetings. I learned a lot by listening to their advice on my projects and during discussions on other projects. I am very grateful to have had their guidance this year.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Autocallable Note

An automatically callable note, more commonly known as an autocallable note, is a type of structured product. They were first introduced in the United States in 2003 by BNP Paribas and have seen a steady increase in use since their introduction (Deng et al. 2011). Autocallable notes have also become common in Canada, where they are issued by all five major Canadian banks. Autocallable notes are attractive because they offer principal protection similar to bonds yet have higher returns (Cuim et al. 2023). A standard autocallable note pays a coupon (C) if the underlying asset of the note (S) is greater than a value which is called the coupon barrier (CB) at an observation date (OD). Multiple coupons can be paid throughout the note's lifetime. An autocallable note also gets automatically called if the underlying asset is greater than the value which is called the autocall barrier on an observation date. If the autocallable note is automatically called or the note reaches its maturity date, the investor will receive the face value of the note. Observation dates can be at any set time intervals, such as monthly or quarterly. The equations that outline the amount of the coupon received at an observation date and if the note is called at an observation date are expressed as,

$$Coupon(t_i) = C * 1_{\{s_{t_i} \geq CB\}} * 1_{\{max_{j=1,2,\ldots\ldots,i-1}\ s_{t_i} \leq AB\}}, \textit{ (Equation 1)}$$

$$Auto\ Called(t_i) = Face\ Value * 1_{\{s_{t_i} \geq AB |\ t_i = Maturity\}} * 1_{\{max_{j=1,2,\ldots\ldots,i-1}\ s_{t_i} \leq AB\}}. \textit{ (Equation 2)}$$

In Equation 1 C is the coupon amount, s_{t_i} is the underlying asset price at time t_i, AB is the autocall barrier, and CB Is the coupon barrier. A visual may help to better explain the structure of an autocallable note. Figure 1 visualizes the evolution of an underlying stock price with an initial value of 100 and an example autocallable structure. In the figure, the red dotted lines represent the observation dates of the autocallable note with monthly observation dates and a maturity of one year. The orange line represents the Coupon Barrier, and the green line is the Autocall Barrier. The underlying stock price is above the coupon barrier at the last four observation dates; therefore, a coupon is paid on those four dates. Since the underlying stock is never larger than the auto call barrier at an observation date, it is not redeemed early.

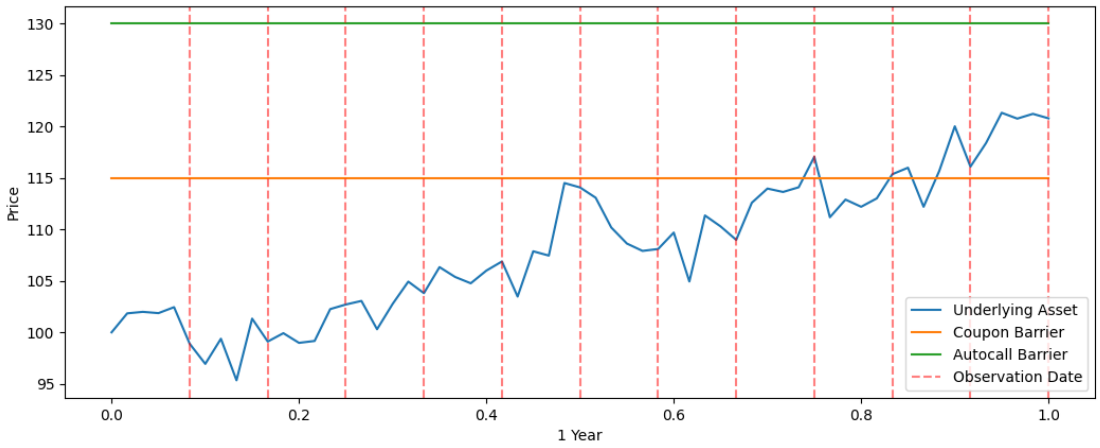

Figure 1: Stock Price Evolution Against Autocallable Note Observation Dates, Coupon Barrier, and Autocall Barrier

The price of an autocallable is the present value of the expected cashflows from coupons and the face value. Pricing an autocallable is path-dependent, meaning the probability of receiving a coupon on an observation date is conditional on whether the note was automatically called previously. This path dependency leads to several different strategies being common in pricing an autocallable note. An analytical solution for the price of European options was derived via the Black and Scholes (BS) option pricing model (Black and Scholes 1973). Guillaume (2015) derived an explicit expression for the price of several different autocallable notes under the BS model. This paper is useful to show that analytical solutions are possible; however, for this application, they are not viable as they are not generalizable for different structures of notes. Deng et al. (2011) present a flexible partial differential equation to model autocallable notes and use the finite difference method to price the notes. The finite difference is an accurate and efficient method of pricing but also does not generalize to price many different note structures. The most common strategy to price autocallable notes is using Monte Carlo simulations. Lee and Hong (2021), Fries and Joshi (2008), and Koster and Rehmet (2018) all present different Monte Carlo simulation strategies to price autocallable notes. Monte Carlo simulations are a very good method for pricing; however, for this application their downfall is they are computationally expensive. Finally, binomial and trinomial trees offer another way to price a variety of options, including barrier options (Hull 2012). Binomial trees were used by Li and Zhang (2022) to price a type of autocallable note. Pricing trees are advantageous in this scenario as they can store the

price of the autocallable note at many different points of time and prices of the underlying asset. The pricing strategy used in this thesis will be explained in Chapter 4.

The discontinuous nature of the payoff structure of autocallable notes leads to challenges in risk management. Figure 2 helps to visualize the effect of the discontinuous payoffs on the price of the autocallable option. In the figure the blue line is the price of the underlying asset, and the orange line is the price of the autocallable note. The orange dotted line is the first observation date. At the first observation date the underlying is very close to the coupon barrier, which causes the price of the autocallable note to spike up.

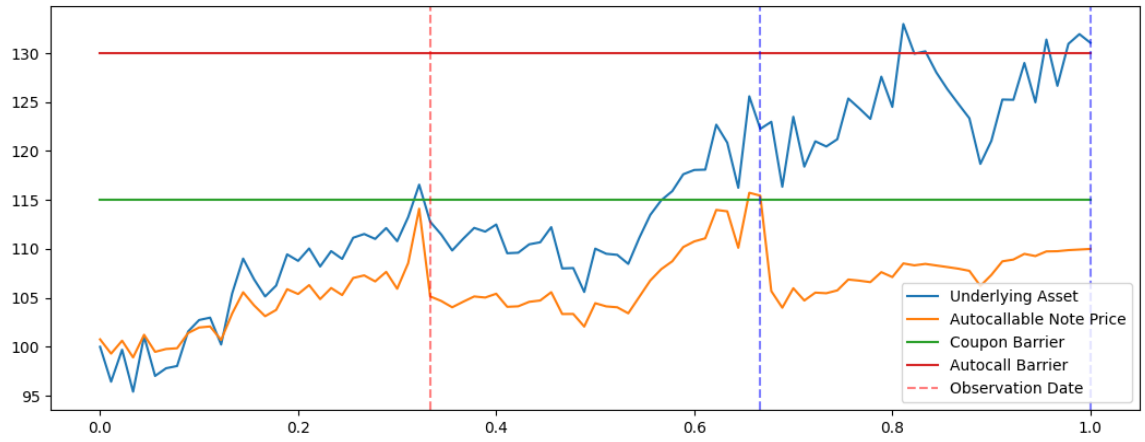

Figure 2: Price of Autocallable Note and Underlying Asset

The Delta of an option is defined as the rate of change of the price of the option with respect to the price of the underlying asset (Hull 2012). In the case of autocallable notes, the Delta around the barriers rapidly increases near barriers to a large number because the price of the autocallable can increase rapidly with only a small increase in the underlying asset. Figure 3 helps visualize this problem, as it plots Delta right before the barrier as a function of the underlying asset price. The first increase occurs at the coupon barrier, where Delta increases rapidly to a large number. The rapid decrease occurs at the autocall barrier, which happens because the price of the autocallable note decreases rapidly since it is redeemed early.

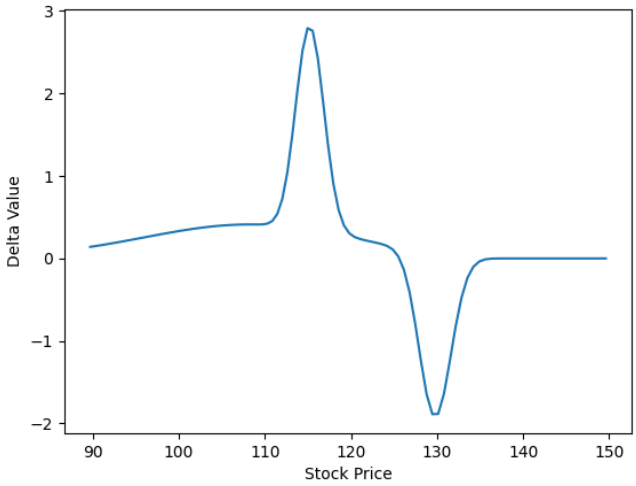

Figure 3: Delta of Autocallable Note Before Observation Date versus Underlying Asset Price

The Gamma of an option is defined as the rate of change of the Delta of an option with respect to the price of the underlying asset (Hull 2012). The Gamma around the barriers also increases and decreases to large and small amounts because of the movement of Delta around barriers. This behaviour is presented in Figure 3, which shows the value of Gamma of an autocallable note against the price of the underlying asset.

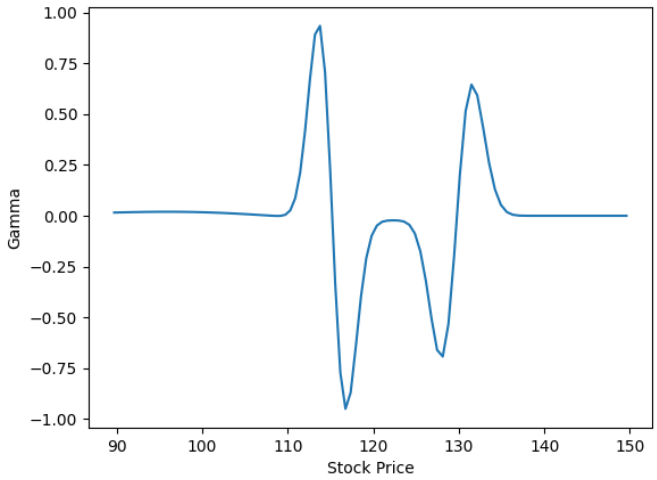

Figure 4: Gamma of Autocallable Note before Observation Date against Underlying Asset

Delta and Delta-Gamma Neutral dynamic hedging can be difficult because of these barriers and can lead to a large variance in Profit and Loss as large amounts of the underlying or

hedging option may be bought when hedging near the barriers. This presents the problem of finding an optimal way to dynamically hedge an autocallable note.

1.2 Reinforcement Learning

The optimal way to dynamically hedge an autocallable note can be framed as a sequential decision-making process in a stochastic environment. Reinforcement Learning (RL) is an area of machine learning that is particularly well suited to solving these types of problems. This section will explain the basics of RL and review different RL algorithms.

Reinforcement learning (RL) is an area of machine learning that is structured for modelling sequential action-making and learning an optimal action-making strategy. In this structure, there is an agent and an environment. The environment is a set of states, denoted as S. The agent can observe the current state it is in, denoted as s, can make an action, denoted as a, and will receive a reward, denoted as r, based on the action. After each action and reward is received, the agent transitions to a new state. The goal of training an RL agent is for it to learn how to make the optimal action for its current state (Sutton and Barton 2018). This is done by training the agent to make actions that maximize future expected rewards, which is clearly expressed as,

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \dots + \gamma^{T-1} r_T, \textit{ (Equation 3)}$$

by Sutton and Barto (2018), where r_t is the reward at time t, and γ is the discount factor. The discount factor is necessary to avoid infinite future returns and also make rewards closer in time more impactful. The agent's policy, denoted by π, is a mapping of the set of possible states to the set of possible actions, meaning it outlines what actions the agent will take given its current state. The optimal policy denoted by $\pi *$ is the policy that maximizes G_t. To determine how good an agent's policy is an equation called the Q-value is used. The Q-value of policy π is summarized by Sutton and Barto (2018) as,

$$Q^{\pi}(s, a) = \mathbb{E}[G_t | s_t, a_t], \textit{ (Equation 4)}$$

which is the expected value of the future rewards conditional on the state s_t and action a_t from the agent's policy. Many different algorithms present different strategies to learn the optimal policy. The first categorization of algorithms is into model-free algorithms and model-based

algorithms. Model-based algorithms try to learn a model of the environment, meaning they try and learn to predict state transitions and rewards. Whereas model-free algorithms do not try to learn the model of the environment (Achiam and Abbeel 2018). Model-free algorithms can be further split into value-based methods and policy-based methods.

Value-based methods aim to find the optimal action-value function by directly evaluating state-action pairs and determining which are more valuable in terms of cumulative future rewards. An important value-based RL algorithm was introduced by Watkins (1989) called Q Learning. The goal of Q-Learning is to learn the Q-value stemming from the optimal policy, denoted as $Q^{\pi*}(s,a)$. Q-Learning allows $Q^{\pi*}(s,a)$ to be updated while following a suboptimal policy, which helps to explore more of the environment. Because the Q-function is updated while following a suboptimal policy and not with the current optimal policy, it is characterized as an off-policy method. The Q-Learning update equation is a version of the Bellman equation and was expressed in Sutton and Barto (2018) as,

$$Q(s,a) = Q(s,a) + \alpha\big(r' + \gamma * max_{a'}Q(s',a') - Q(s,a)\big) \textit{ (Equation 5).}$$

In Equation 5 r' is the reward received after taking action a in state s, (s',a') is the state-action pair at the next time step, and α is the learning rate. The intuition is that as the Q-function is updated, the policy is improved. The max function is used to iterate through all a' that could be taken after it has transitioned to s' (Sutton & Barto, 2018). The Q function is continually updated until the optimal Q-value is found. One notable limitation of Q-Learning is the requirement of a lookup table to store the Q-value for each (s,a). For high-dimensional problems with a large number of states and actions this becomes infeasible to do. This limitation was addressed by Mnih et al. (2013) by using a Deep Neural Network (DNN) to approximate the Q-function instead of using a tabular method. The Q-function was denoted as $Q(s,a;\ \theta)$, where theta represented the weights of the neural network. Changing from a tabular method to a DNN allowed for much higher dimensional state and action spaces. The DNN is trained using stochastic gradient descent, which updates θ based on a loss function. Because of the use of deep neural networks this method has become known as Deep Q-Learning. Deep Q-Learning is the basis for most modern value-based deep reinforcement learning algorithms.

Policy-based methods are different from value-based methods because they represent a policy explicitly, denoted as $\pi_\theta(a|s)$, where theta represents the policy parameters. These methods optimize the parameters of the policy θ directly by gradient ascent or by using a performance objective function. One basic policy-based algorithm is known as policy gradient. Policy Gradient works by computing the gradient of the performance objective function and updating the policy parameters next. The performance objective function is described in Sutton and Barto (2018) as,

$$J(\theta) = \sum_{s\in S} d^{\pi}(s) \sum_{a\in A} \pi_\theta(a|s)\, Q^{\pi}(s,a) \textit{ (Equation 6),}$$

where $d^{\pi}(s)$ is the stationary distribution for π. The advantages of a policy-based algorithm are that the algorithm is directly optimizing the policy, which is compared to value-based algorithms that indirectly optimize the policy by optimizing the Q-function. This trade-off makes value-based algorithms usually more sample efficient when they do work, because they can reuse data more effectively than policy optimization techniques. However, policy-based have the advantage of being able to learn stochastic policies, have the ability handle continuous action spaces and being more stable in high dimensional environments. (Achiam and Abbeel 2018)

Due to the environment of this specific problem, another issue with the DQN algorithm is that the use of the argmax function makes it infeasible to have continuous action spaces. This problem is addressed by policy-based deep reinforcement algorithms. Deep Deterministic Policy Gradient (DDPG), introduced by Lillicrap et al. (2015), is an DRL algorithm that addresses this problem. DDPG combines the advantages of both value-based algorithms and policy-based algorithms by using two neural networks, the actor-network and the critic-network. The actor-network determines the optimal actions given the current state and the critic network estimates the state-action value function or Q-function (Cassimero 2023). The critic operates similarly to the DQN algorithm. Differentiating the current Q-value with respect to the actions brings it to converge to the optimal Q-value $Q^{\pi*}(s,a)$ (Lillicrap 2015). The critic's Q-value for each iteration is updated through stochastic gradient descent on the loss function as (Sutton and Barton 2018),

$$L_i(\theta_i) = \mathbb{E}_{s,a}\left[\left(y_i - Q(s,a;\,\theta_i)\right)^2\right] \textit{ (Equation 7).}$$

The target y_i, is computed using the current policy estimate of the actor's current policy π expressed by Sutton and Barto (2018) as,

$$y_i = \mathbb{E}_{s,a}[r' + \gamma * Q(s', \pi_i(s');\ \theta_{i-1})]. \textit{ (Equation 8)}$$

Q-Value updated through stochastic gradient descent on the loss function in Equation 7 is used in the actor's gradient ascent operation. Because there is no use of an argmax function this allows DDPG to be used in environments with continuous action spaces.

1.4 Objective

The objective of this thesis is to investigate the use of deep reinforcement learning to Delta and Delta-Gamma hedge an autocallable note dynamically. This objective is split into sub-objectives to test the viability of deep reinforcement learning in hedging autocallable notes and to find optimal training hyperparameters for the hedging DRL algorithm.

1.5 Deliverables

1. The first deliverable is a generalizable pricing function that accurately prices autocallable notes of different structures, coded in Python.

2. The second deliverable is a baseline hedging model that calculates the Delta neutral or Delta-Gamma neutral position and takes that position, coded in Python.

3. The third deliverable is a DDPG model coded in Python using the TensorFlow library.

4. The fourth deliverable is a training procedure for the DDPG model, coded in python.

5. The fifth deliverable is a testing procedure also coded in Python.

1.6 Outline of Chapters

The rest of the thesis is structured into different chapters. Chapter 2 contains a literature review to explore the current research on this problem. Chapter 3 contains the methodologies of the experiments conducted. Chapter 4 presents the results of the experiments. Chapter 5 presents the Conclusions stemming from the results. Finally, Chapter 6 presents recommendations for future work.

2. LITERATURE REVIEW

The purpose of this section is to explore what strategies have been used to dynamically hedge autocallable notes using Deep Reinforcement Learning (DRL). Due to the scarce amount of work using DRL to hedge autocallable notes, this section will be split into separate sections for non-RL hedging strategies for autocallable notes, how RL is used to hedge vanilla options and how RL is used to hedge autocallable notes.

2.1 Non-Reinforcement Learning Hedging

As discussed in Chapter 1, the Delta of an autocallable note can increase rapidly to a large number, around barriers. There are non-RL strategies to address the large Delta values stemming from the barriers. The purpose of exploring non-RL methods is to get a better sense of what is being done currently to hedge autocallable notes in order to compare results. Cui et al. (2023) considered a dynamic delta neutral hedging approach in the presence of transaction costs, to hedge an autocallable note. To address the problem of the product's Delta becoming too large near the barriers, they apply payoff modification to the barriers. This payoff modification smooths the payoff of the autocallable from a jump to a smooth payoff. As a result, the Delta changes more smoothly and does not rapidly increase to large numbers. Without a large increase, the Delta is much easier to hedge, which leads to a decrease in transaction cost (Cui, Li and Zhang 2023). The one caveat is determining how drastically the payoff should be modified. Modifying to little does not address the problem of large Delta values near the barrier, whereas too large of a modification leads to the modified payoff being too dissimilar to the original payoff at the barrier. If the modified payoff is too dissimilar, then a Delta neutral hedge on it would not be close to a Delta neutral position on the unmodified payoff. A systematic approach to deciding this trade-off is provided which minimizes the CvaR of the hedging loss. This paper is important for validating the existence of the problem of large Delta values near the barrier. It could also provide a base model for comparison to an RL hedging model.

2.2 Reinforcement Learning Hedging of Vanilla Options

The use of Reinforcement Learning (RL) in dynamically hedging European options has been written about to a far greater extent than autocallable notes. Although the actual hedging of the European options may look different, it is still very similar in scope. Therefore, exploring this literature will help better the understanding of the choice of different RL algorithms, what to include in the state and action spaces, what to use as a reward function and possible baseline models. One of the earlier papers to use a model-free RL algorithm to solve the problem of dynamic option hedging was Kolm and Ritter (2019). Kolm and Ritter (2019) used a policy-based RL algorithm called SARSA, which allows for a continuous action space. SARSA is not a deep RL algorithm, so usually the Q-values are stored in a table. Because the hedging environment has continuous state variables, a linear function was used instead. This is why deep RL models are more common for this problem. Pickard and Lauryshyn (2023) provide a review of 17 studies addressing dynamic vanilla option hedging through deep reinforcement learning (DRL). The studies it addresses include Halperin (2019), Kolm and Ritter (2019), Du et al. (2020), Vittori et al. (2021), Cao et al. (2021), Pham et al. (2021), Giurca and Borovkova (2021), Xiao et al. (2021), Kim (2021), Assa et al. (2021), Murray et al. (2022), Xu and Dai (2022), Mikkilä and Kanniainen (2023), Cao et al. (2023), Zheng et al. (2023), Canelli et al (2023) and finally Fathi and Hientzsch (Fathi and Hientzsch 2023). The review gives invaluable insight and trends regarding the choice of DRL algorithms, state and action spaces, reward functions, data generation processes and results for each study. A valuable insight about choices of DRL algorithm is that most papers opt for policy-based algorithms as they allow for continuous action spaces, with Deep Deterministic Policy Gradient (DDPG) being the most common algorithm (Pickard and Lauryshyn 2023). In regard to state spaces, insights about optimal variables to include were found. Almost all studies included the price of the underlying asset and the time to maturity in the state space. A significant number of studies included the current hedging amount and the price of the option in the state space (Pickard and Lauryshyn 2023). The action space is always buying or selling a continuous or discrete amount of the underlying asset or hedging option. The majority of studies use a variant of the mean-variance metric to maximize returns while minimizing variance, expressed as,

$$R_t = \delta w_t - \lambda(\delta w_t)^2 \quad \textit{(Equation 9)},$$

where λ is a measure of risk aversion, and δw_t is the incremental wealth of the hedging portfolio at time t. Successful results are also achieved with the return, Var and CvaR reward function formulations (Pickard and Lauryshyn 2023). Geometric Brownian Motion (GBM) is the most common data generation process, with some studies opting to use stochastic volatility models such as the SABR or the Heston models. In terms of inclusion of transaction costs, some studies include these, and some do not. It is found that in the studies that include transactions costs, the RL agents are more cost-conscious (Pickard and Lauryshyn 2023). Although this review focused on vanilla options compared to autocallable notes, it is still very valuable in determining the structure of our model.

2.3 Reinforcement Learning Hedging of Autocallable Note

The use of reinforcement learning (RL) to hedge autocallable notes is quite a new pursuit, so there is a small amount of literature available. The earliest experiment was by Xu et al. (2023), in which they utilized the RL algorithm SARSA to hedge an option called a Phoenix option. A Phoenix option is a name for a particular structure of autocallable note. SARSA is a value-based RL and uses an on-policy update rule, meaning that it performs its action based on its current most optimal policy and updates from there. The paper also used GBM to simulate asset paths. The paper used the Sharpe ratio as their reward function, which is a risk-adjusted measure of return. The use of the Sharpe ratio makes more sense as they are hedging in the presence of transaction costs. They do not compare to a baseline model but evaluate their hedging results using the Sharpe ratio, where they achieve good results. In their conclusion, they suggest further exploration of different weights, performance measures and RL algorithms. This paper is important as it is the first use of RL to hedge autocallable notes, and it also gives good insight into the choice of reward functions and the RL algorithm. The drawback to using Sharpe Ratio is that its intuition does not fit in a no transaction cost environment, as the goal of hedging is not to maximize the return, but to minimize loss. The drawback to SARSA is that it is not a deep RL algorithm so it stores the Q-function tabularly which can be computationally expensive.

The most recent paper to use RL in hedging autocallable notes was by Noh (2024). This paper used DDPG as its RL algorithm which provides the advantage of Deep RL while having a

continuous action space. This paper uses Monte Carlo simulations under a SABR underlying process to price the autocallable as well as a finite difference method to calculate Delta and Gamma. The state space contains the underlying stock price, the gamma of the portfolio and the time to the next observation date. The intuition of putting the portfolio gamma in the state space is to help the agent learn to decide the percentage of the Gamma that needs to be hedged. The underlying stock is not used to hedge in this paper. Instead, they run two experiments where an American option and a digital option are used to hedge. The American option was priced using the Longstaff Schwartz algorithm. A digital option is chosen as it also has a barrier type payoff similar to an autocallable. The reward function used in Noh (Noh 2024), $R_i = -\kappa|V_i H_i| + (P_i^- - P_{i-1}^+)$ where V_i is the value of the d option at time $i\Delta t$, H_i is the position taken in option, κ is the transaction cost, P_i^- is portfolio's market value before time $i\Delta t$, P_{i-1}^+ is portfolio's market value after time $i\Delta t$. The two RL strategies were assessed against Delta neutral and Delta Gamma neutral strategies and were found to have a lower CvaR and VaR than those methods. This indicated a positive result and potential for the use of deep RL in hedging autocallable notes. The importance of this paper was the use of American and digital options for hedging, it indicates that the unique structure of the autocallable note might call for hedging options with different structures. It would be useful to compare how hedging with options compares to hedging with the underlying asset.

2.4 Summary

In summary, this section reviews current non-RL autocallable note hedging strategies, reviews literature on RL strategies for hedging Vanilla options and reviews the small amount of the literature regarding RL strategies for hedging an autocallable note. The main takeaways from this section are the insights on the choice of RL algorithm, state space, action space, and finally formulation of reward function. It also highlights the gap in research in this specific area which is a big motivation for this thesis.

3. METHODOLOGY

Implementing a deep reinforcement learning (DRL) agent to dynamically Delta and Delta-Gamma hedge an autocallable note requires data of stock prices over time, a mechanism for pricing an autocallable note, a baseline hedging model for Delta and Delta-Gamma hedging, a design of a DRL agent, a training procedure and finally a testing procedure. The data of stock prices will be generated using Monte Carlo simulations under a Geometric Brownian Motion process. The strategies used to price autocallable notes are binomial and trinomial Trees. The baseline hedging model will be a Delta neutral and Delta-Gamma neutral hedging model. The design of the DRL agent involves the choice of DDPG as the DRL algorithm used as well as the construction of the actor and critic neural networks, choice of state and action spaces, choice of reward function, and tuning of hyperparameters. The training procedure involves the use of GBM to generate data to train the agent over a set number of episodes. Finally, the testing of the agent will be done by comparing results to the baseline model.

3.1 Data

Data is especially important when it comes to training and testing reinforcement learning algorithms. In the hedging environment, the agent specifically needs data of stock prices over time. Using Monte Carlo simulations under a Geometric Brownian Motion (GBM) process is the most common data generation technique in RL hedging literature (Pickard and Lauryshyn 2023). The advantages of using GBM, is it is simple, and it is easy to generate lots of data. Some disadvantages are that a constant volatility is chosen, and it can not mode jumps in stock prices, which are both present in the real-life stock market (Glasserman 2003). Other alternative sources of data would be to use real-world stock price data or generate data from a more complex process such as the Heston or SABR model, which incorporates stochastic volatility. Under GBM, the underlying stock price evolves according to the equation expressed in Glasserman (2003) as,

$$S_t = S_0 \, exp\left(\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right) \textit{(Equation 10),}$$

where μ is the risk-free rate, σ is the volatility, t is the time, and W_t is Brownian Motion under the risk-neutral measure. A sample path of stock prices over time can be generated by taking time steps Δt and calculating S_t.

3.2 Pricing Method

Binomial and trinomial pricing trees were chosen to price the autocallable notes, as they store discrete autocallable prices at different nodes on the tree. Each node is an index of a specific underlying asset price at a certain point in time. If a price of the autocallable is needed that is between stock prices nodes, then linear interpolation between nodes is used. Pricing trees are well suited for RL hedging as the tree only needs to be calculated once before training occurs and can be used throughout training. A binomial tree is a very simple way to price an option, which only relies on the assumption of no arbitrage. A multi-step binomial tree of stock prices can be generated using the formulas as described in Hull (2012),

$$u = e^{\sigma\sqrt{(\Delta t)}}, d = e^{-\sigma\sqrt{(\Delta t)}},\ P_u = \frac{e^{r\Delta t}-d}{u-d}, P_d = (1 - P_u)$$ *(Equation 11).*

At each step, the stock price S_t can either go up by the factor u with probability P_u or down by the factor d with probability P_d. The price of the option at time t can be calculated as, from the risk-neutral valuation principle, it should be equal to the expected payoff discounted back to time t. This formula can be described as,

$$f = e^{-t\Delta t}[P_u f_u + P_d f_d]\,,\ f_u = e^{-t\Delta t}[P_u f_{uu} + P_d f_{ud}],\ f_d = e^{-t\Delta t}[P_u f_{du} + P_d f_{dd}]$$

(Equation 12),

Where f is the price of the autocallable note at the initial time or at the root of the tree, f_u is the price after 1 time step when the underlying asset price increased and f_d is the price after one time step when the underlying asset price decreased. Furthermore, f_{uu} is the price of an autocallable note after two times steps, where the underlying asset increased at both steps and any variable f_{xx} proceed in the same way. This process can be repeated for any number of steps. In the case of autocallable notes, we know the value of the note at maturity. Therefore, we can build a pricing tree from maturity to the present. When an observation date is encountered, if on the tree the underlying asset at that node is above the coupon barrier, the coupon is added onto the value

of the autocallable at that node. If the underlying asset is above the autocall barrier, then the value of the autocallable note at that node is set to the face value plus the coupon.

A trinomial tree is a more developed version of a binomial tree, where instead of a stock having the option to go up or down at a time step it also has the option to go on a middle path. This middle path may be the same price or increase by the risk-free rate. A trinomial tree can be a more accurate version of a binomial because more nodes are present. A multi-step trinomial tree of stock prices can be generated using the formulas described in Hull (2012) as,

$$u = e^{\sigma\sqrt{2\Delta t}}, d = e^{-\sigma\sqrt{2\Delta t}}, m = 1, \qquad P_u = \left(\frac{e^{r\Delta t} - d}{u - d}\right)^2, P_d = \left(\frac{u - e^{r\Delta t}}{u - d}\right)^2,$$

$$P_m = 1 - (P_d + P_u), (\textit{Equation 13}).$$

Where m is the change of the stock price in the middle path, and P_m is the probability of the stock taking that path. Just as in the binomial tree, the price of the option at time t can be calculated as, from the risk-neutral valuation principle, it should be equal to the expected payoff discounted back to time t. This formula can be described as,

$$f = e^{-t\Delta t}[P_u f_u + P_m f_m + P_d f_d] \textit{ (Equation 14).}$$

The trinomial tree can be built from maturity to the initial time in the same way as the binomial tree. The trinomial tree was chosen over the binomial tree for pricing as it has more nodes, therefore leading to a more accurate price (Hull 2012).

3.4 Autocallable Note Structures

Two different structures of autocallable notes have been chosen. The first structure is presented in Table 1 was constructed with the initial goal of testing the viability of using DRL to dynamically hedge an autocallable note.

Table 1: Autocallable Note Structure One.

Variable	Value
Initial Stock Price	100$
Face Value	100$
Coupon	10% of Face Value
Observation Dates	Tri-Annually
Coupon Barrier	115% of The Initial Stock Price
Autocall Barrier	130% of The Initial Stock Price
Maturity	1 Year

A maturity of one year and three observation dates was chosen as a simpler note structure will make it easier to train an initial DRL model. A large coupon of 10% was chosen so the effects of the barrier in hedging would be very clear. This structure does not represent an autocallable present in the market but helps to get a sense of how training a DRL agent to hedge an autocallable note would proceed.

The second structure of autocallable note is similar to notes present in the real-world market. The structure is presented in Table 2. The major changes are that the coupon value is smaller at 2% but the observation dates are more frequent, being monthly. The Maturity is also longer, at 7 years.

Table 2: Autocallable Note Structure Two

Variable	Value
Initial Stock Price	100$
Face Value	100$
Coupon	2% of Face Value
Observation Dates	Monthly
Coupon Barrier	115% of Initial Stock Price
Autocall Barrier	130% of Initial Stock Price
Maturity	7 Year

3.5 Neutral Hedging Baseline

The most common baseline model used in the literature for DRL for option hedging was to use Black-Scholes Delta Gamma, which are the Delta Neutral and Delta-Gamma Neutral positions (Pickard and Lauryshyn 2023). Because the autocallable note is being priced using trees and not an analytical solution, there is no available analytical formula of Delta or Gamma for an autocallable note. Under the binomial trees of stock prices and autocallable note prices, an estimate of Delta and Gamma can be calculated using the formula expressed as (Hull 2012),

$$\Delta_0 = \frac{(f_u - f_d)}{(S_u - S_d)}, \quad \Gamma = \frac{(\Delta_u - \Delta_d)}{(S_u - S_d)} \quad (\textit{Equation 15}),$$

where f_u and f_d are the same as defined above. Δ is Delta, and Γ is Gamma. Based on the trinomial trees of stock prices and autocallable note prices, the estimates of Delta and Gamma are given by (Hull 2012),

$$\Delta_0 = e^{\left(-r\frac{\Delta t}{2}\right)}\left(\frac{(p_u f_u + p_d f_m - p_u f_m - p_d f_d)}{\left(Se^{\left(r\frac{\Delta t}{2}\right)} - Se^{\left(-r\frac{\Delta t}{2}\right)}\right)}\right) \Gamma_0 = e^{\left(-r\frac{\Delta t}{2}\right)}\left(\frac{(p_u \Delta_u + p_d \Delta_m - p_u \Delta_m - p_d \Delta_d)}{\left(Se^{\left(r\frac{\Delta t}{2}\right)} - Se^{\left(-r\frac{\Delta t}{2}\right)}\right)}\right) \text{ (Equation 16).}$$

When Delta-Gamma hedging with the underlying asset and a hedging option, the value of Delta is affected by both the underlying asset and the hedging option. The amount invested in the hedging option should follow the equation outlined by Hull (2012) as,

$$\Gamma_t = \frac{\Gamma_t^g}{\Gamma_t^h} \text{ (Equation 17),}$$

where Γ_t^g is the Gamma of the option to be hedged and Γ_t^h is the Gamma of the hedging option. As the hedging option impacts Delta the formula for the amount invested in the underlying asset α_t, should follow the equation outlined by Hull (2012) as,

$$\alpha_t = \Delta_t^g - \Gamma_t \Delta_t^h \text{ (Equation 18),}$$

where Δ_t^g is the Delta of the option to be hedged and Δ_t^h is the Delta of the hedging option. This scheme will be followed for the baseline model when Delta-Gamma hedging.

3.6 Reinforcement Learning Model

Structure

The choice of what deep reinforcement learning (DRL) agent to use was decided based on the specific needs of a hedging environment and what is most common in DRL hedging literature. The action space of a hedging environment needs to be continuous because the action is buying a certain amount of an underlying asset or hedging option. That is why value-based methods such as DQN were not chosen. DDPG was chosen as the DRL algorithm as it allows for continuous action space, and it was commonly used in DRL hedging literature (Pickard and Lauryshyn 2023).

DDPG uses two neural networks for the actor and for the critic. A diagram of the actor network is presented in figure 5.

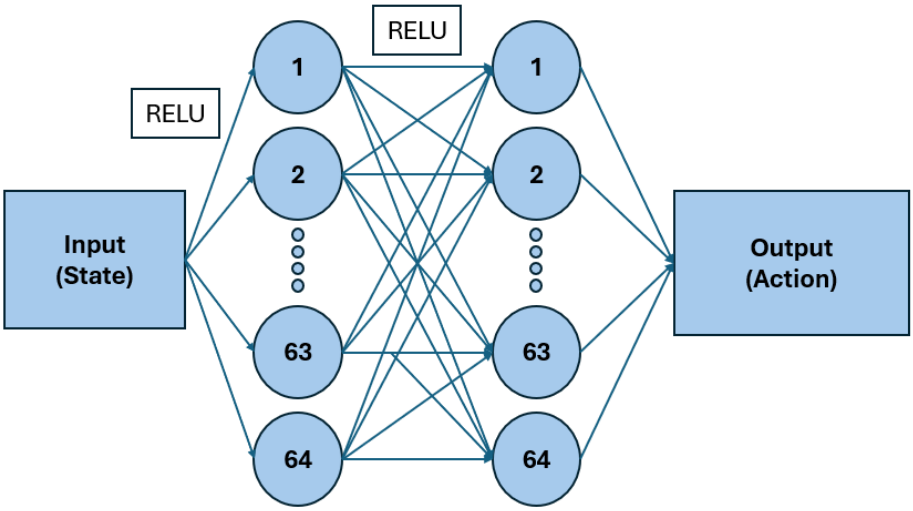

Figure 5: Actor Network Architecture

The actor network takes in the state as the input. It has two fully connected layers consisting of 64 neurons each. The RELU activation function is used on the two fully connected layers and a linear activation function is used on the output layer. The output of the actor network is the action as the actor networks job is to store the policy. A diagram of the critic network architecture can be seen in figure 6.

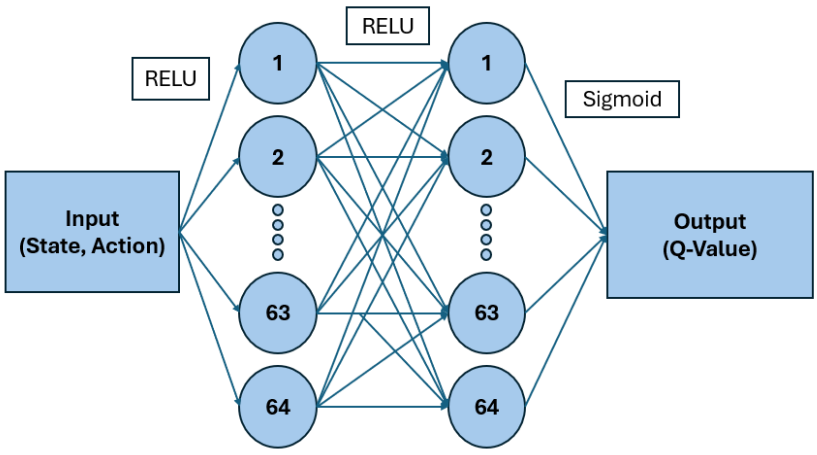

Figure 6: Critic Network Architecture

The critic network takes in the state and action as an input. It also has two fully connected layers consisting of 64 neurons each. A RELU activation function is used for the first to layers and a

sigmoid activation function is used for the output. The output of the critic network is the Q-Value, as the goal of the critic network is to store the Q-function. When Delta hedging, the state space includes the time to maturity, the price of the underlying asset, and the current number of shares held. The action space is the amount of the underlying asset to buy or sell. When Delta-Gamma hedging the state space includes the same variables as in the case of Delta hedging with the addition of the current amount of the hedging option held. The action space also includes the amount of the underlying asset to buy r sell with the addition of a variable for the amount of hedging option to buy or sell. The addition of the price of the autocallable note and the price of the hedging option was considered, however, it was decided against as the DRL agent should be able to learn the correct amount to hedge based off the reward function. The reward function for the Delta hedging agent is given by,

$$R_t = -|A_t[S_{t+1} - S_t] - [C_{t+1} - C_t]| \text{ (Equation 19),}$$

where A_t is the amount invested in the underlying stock at time t, S_t is the underlying asset at time t and C_t is the price of the autocallable at time t. This equation is the absolute value of the difference between the value invested in the underlying stock and the value of the autocallable note. A negative is put in front of the absolute difference because the RL agent will try and find a policy that maximizes the future reward, and the best possible hedge would result in an absolute difference of zero. The reward function for the Delta-Gamma hedging agent is given by,

$$R_t = -|A_t[S_{t+1} - S_t] + B_t[H_{t+1} - H_t] - [C_{t+1} - C_t]| \text{ (Equation 20).}$$

This reward function is the same as the previous reward function with the addition of B_t which is the amount invested in the hedging option and H_t, which is the price of the hedging option at time t. Again, the optimal hedge would result in a reward function that is zero. In both cases, transaction costs are not considered.

Another important part in setting up the DDPG agent is the use of a Replay Buffer. A replay buffer stores previous experiences to be sampled. It stores each state, action, reward and next state together in a tuple. Mini batches of these tuples are sampled when the actor and critic neural networks are updated. The size of the Replay buffer is set to 10,000, so it will store the past 10,000 experiences. The reason a replay buffer is used is to ensure the data used to update the neural networks are independently distributed. When random batches are sampled from a replay buffer, the samples are independently distributed.

3.7 Reinforcement Learning Model Training

Before training the RL model the choice of hyperparameters is very important. The specific hyperparameters involved in training a DDPG agent are the number of episodes trained over, the actor and critic learning rates, the sample mini-batch size, the target update factor τ and the discount factor γ. For all experiments, the sample mini batch size was set to 64, the target update factor was set to 0.001, and the discount factor was set to 0.999. In contrast, the number of episodes, the actor learning rate and the critic learning rate were tuned for each experiment. Careful selection of the learning rate is critical in balancing the stability of the model with efficiency. For each hedging episode, the number of rebalancing points is set. For all experiments except the last, there are three equally spaced rebalancing points before each observation date, the closest rebalancing step being two days from the observation date. The training episode starts with the issuing of the autocallable note, where the underlying stock is at an initial price. At each rebalancing step, the stock price evolution is generated by the GBM process giving the RL agent the new state; the RL agent then makes action, and the state action, reward and next state are stored in the replay buffer to sample and train the actor and critic networks. Figure 7 depicts the training procedure of the DDPG model for the dynamic hedging environment.

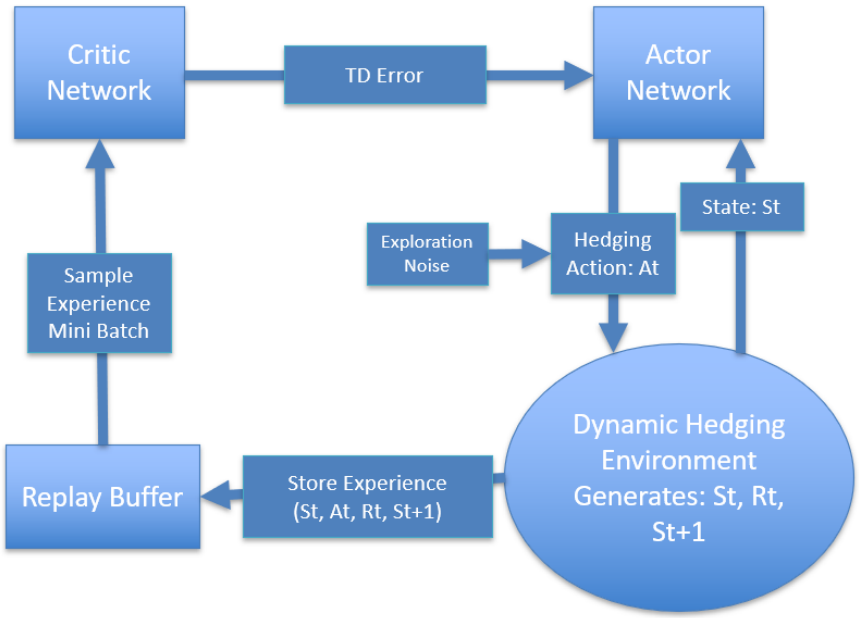

Figure 7: Training procedure for DDPG Model for Hedging

3.8 Reinforcement Learning Model

Testing

The RL testing procedure is to run the trained agent in the same hedging environment. The RL agent is run in the same hedging environment for a number of trials as well as the baseline hedging model. Then, the profit and loss results are compared for the two models. The final profit and loss of both models are calculated by tracking a "bank account" position throughout the hedging episode. The bank account position is denoted by B. For Delta hedging at the initial time $t = 0$, the bank account position is described as,

$$B_0 = C_0 - S_0 A_0, \qquad B_t = B_{t-\Delta t} e^{r\Delta t} - (A_t - A_{t-\Delta t}) S_t,$$

$$B_T = B_{T-\Delta t} e^{r\Delta t} + S_T A_{T-\Delta t} - C_T \quad (Equation\ 21),$$

where C_t is the price of the autocallable note at time t, S_t is the price of the underlying stock and A_t is the hedging value at time t. For Delta-Gamma hedging, an additional term is added, presented as,

$$B_0 = C_0 - S_0 A_0 - D_0 H_0, \qquad B_t = B_{t-\Delta t} e^{r\Delta t} - (A_t - A_{t-\Delta t}) S_n - (H_t - H_{t-\Delta t}) D_t,$$

$$B_T = B_{T-\Delta t} e^{r\Delta t} + S_T A_{T-\Delta t} + D_T H_{T-\Delta t} - C_T \quad (Equation\ 22),$$

where the additional term D_t is the amount hedged in the hedging option at time t and H_t is the price of the hedging option at time t. This procedure is done 10,000 times for each experiment to get a distribution of P and L for the RL model and the baseline model. Both are expected to have a mean of zero, indicating no arbitrage. A smaller standard deviation of the P and L distribution indicates a better hedging model, as there are fewer large losses.

3.9 Summary

In summary, Chapter 3 outlines the data generation method, pricing strategies, autocallable note structure, the hedging baseline model for comparison, the deep reinforcement learning (DRL) model structure, training procedure and finally testing procedure. Then Geometric Brownian Motion process was used to generate stock price paths because of its commonality in DRL hedging literature. Trinomial trees were the method chosen for pricing the

autocallable note. Delta and Delta-Gamma neutral hedging model was used as the baseline comparison model. The DRL algorithm chosen was DDPG. The setup of the DDPG agent was explained. Finally, the training and testing procedure were outlined.

4.0 ANALYSIS AND RESULTS

This section presents the different experiments conducted and discusses their results. The section is divided into three individual experiments. In the first experiment a DDPG model was trained using a GBM process to dynamically Delta hedge Autocallable Note 1. In the second experiment a DDPG model was trained using a GBM process to dynamically Delta-Gamma hedge Autocallable Note 1. In third experiment the DDPG model was trained to Delta-Gamma hedge the autocallable with structure is presented in Table 2. Additionally, many models were trained with different hyperparameters using a Grid Search. These models were compared to find the optimal hyperparameters.

4.1 Experiment 1: Delta Hedging

A DDPG agent was trained to Delta hedge an autocallable note with structure presented in Table 1, by following the training procedure in Section 3.7. The goal of this initial experiment was to test the viability of the DRL agent in Delta hedging. For this experiment the model was trained over 4000 episodes with an actor learning rate of 1-e5 and a critic learning rate of 1-e6. The experiment was run using the testing procedure outlined in Section 3.8. The testing procedure results of the profit and loss distribution of the RL hedging model are presented in Figure 8 and the Delta neutral model is presented in Figure 9

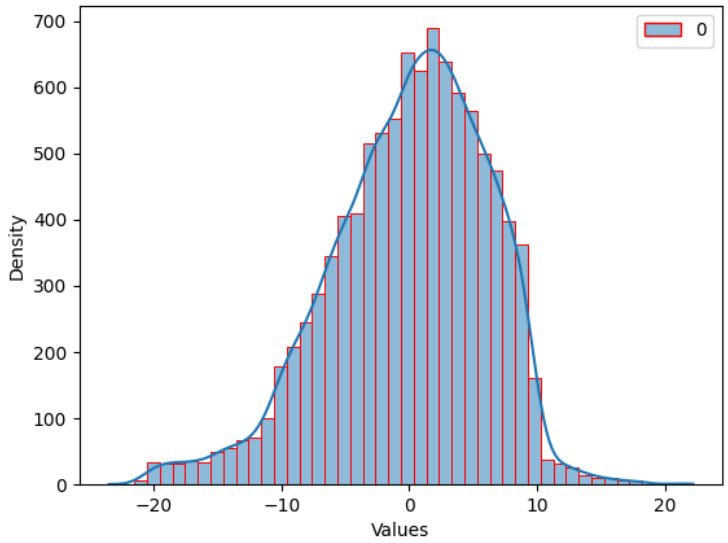

Figure 8: Profit and Loss Distribution of RL Autocallable Note Hedging Model Over 10,000 Trials

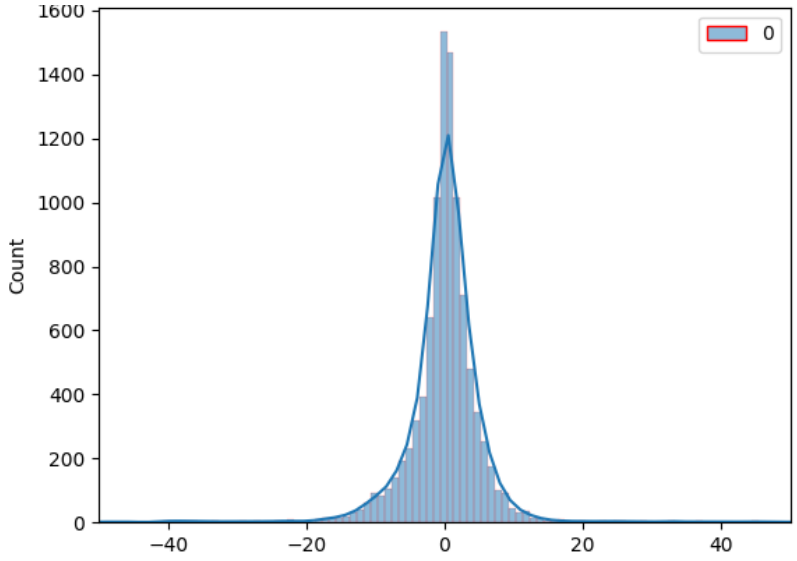

Figure 9: Profit and Loss distribution of Delta Neutral Autocallable Note Hedging Model over 10,000 Trials

The mean of the profit and loss distribution presented in Figure 8 is -0.006501 and the standard deviation is 6.16211. The mean of the profit and loss distribution presented in Figure 9 is 0.016382 and the standard deviation is 9.101137. Both means are close to zero which is the expected result as a bias to one side would indicate improper pricing and the presence arbitrage. The standard deviation of the RL profit and loss distribution is smaller than the Delta neutral profit and loss standard deviation which is a good result. Figures 8 and 9 give us a visual indicator of this result with Figure 9 containing larger bounds where losses and profits go all the way to -40 and 40. This indicates that the Delta neutral model has more extreme losses. Figure 10 shows the sample path of the underlying asset along with the two hedging models' action below over 30 rebalancing steps. An interesting result occurs at the second observation date. The underlying asset is very close to the barrier at the observation date, just under it. In the second graph the orange line representing the Delta neutral model jumps up at this point, whereas the RL model does not as much. This may indicate the difference between the Delta Neutral and RL models actions near the barrier, which may lead to the difference in standard deviation.

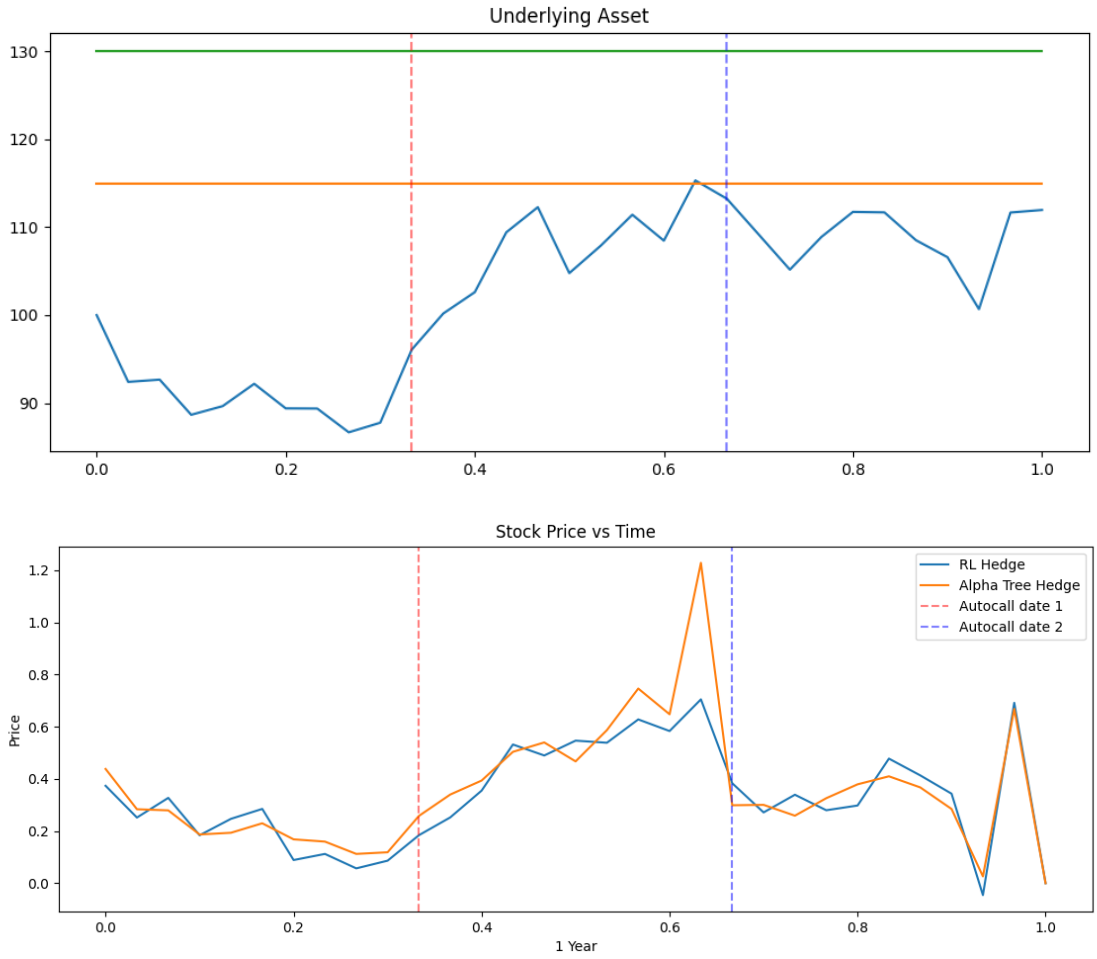

Figure 10: Underling Asset, RL Hedging Model and Delta Neutral Models Action

4.2 Experiment 2: Delta-Gamma Hedging

A DDPG agent was trained to Delta-Gamma hedge an autocallable note with structure presented in Table 1, by following the training procedure in Section 3.7. The goal of this initial experiment was to test the viability of the DRL agent in Delta-Gamma hedging. A European Call Option was used as the hedging option do to it being very simple. For this experiment the model was trained over 3000 episodes with an actor learning rate of 1-e5 and a critic learning rate of 1-e6. The experiment was testing using the testing procedure outlined in Section 3.8. The testing procedure results of the profit and loss distribution of the RL hedging model are presented in Figure 11 and the Delta neutral model are presented in Figure 12.

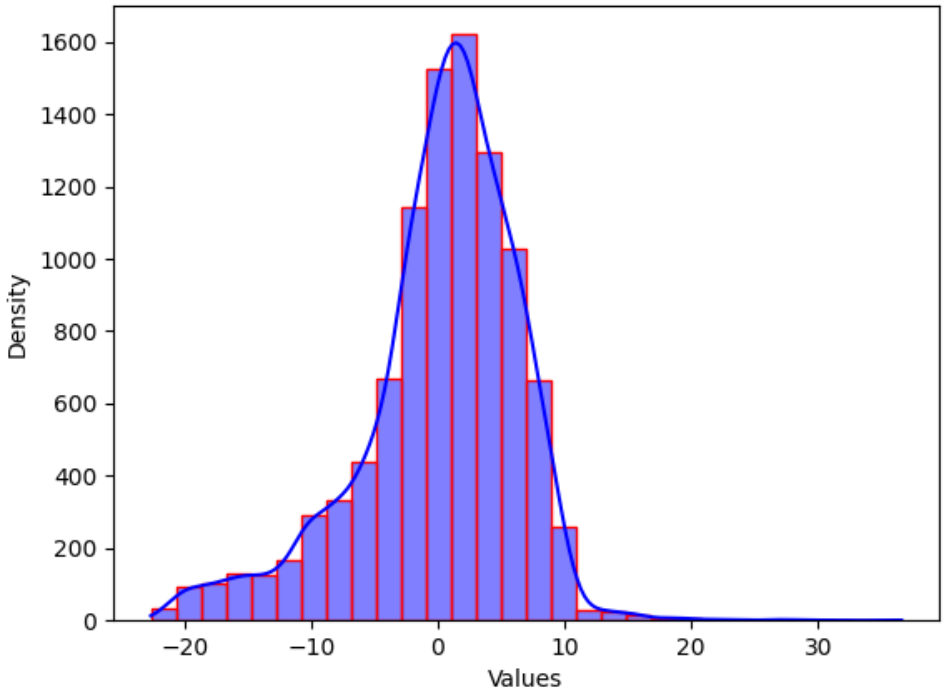

Figure 11: Profit and Loss Distribution of Delta-Gamma RL hedging Model.

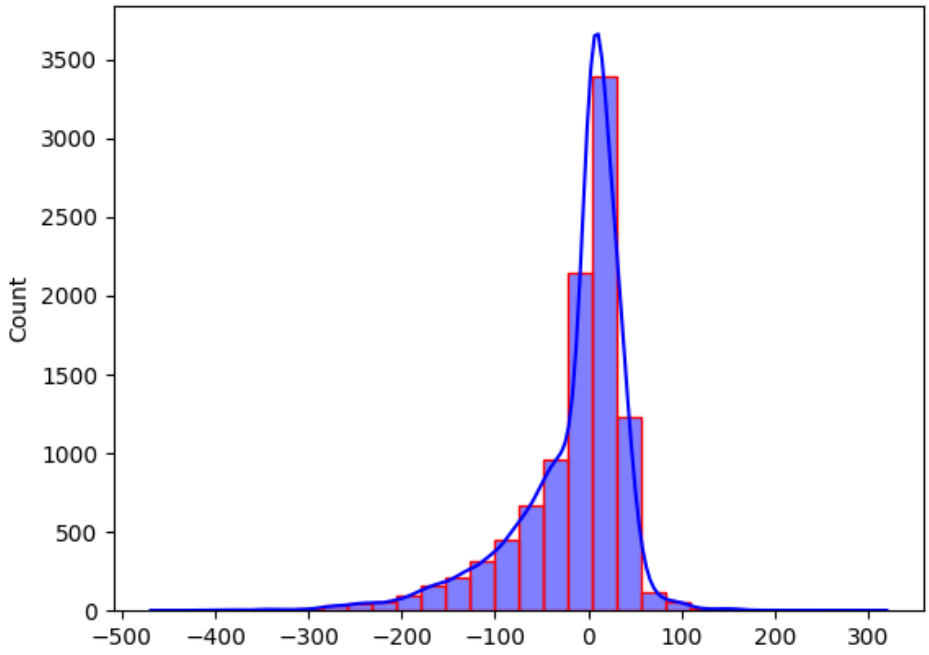

Figure 12: Profit and Loss Distribution of Delta-Gamma Neutral Hedging Model

The mean of the profit and loss distribution presented in Figure 11 is -0.03137 and the standard deviation is 6.2658. The mean of the profit and loss distribution presented in Figure 12 is 0.1365 and the standard deviation is 61.0744. Both means are close to zero which is the expected result as a bias to one side would indicate improper pricing and the presence arbitrage. The standard deviation of the RL profit and loss distribution is much smaller than the Delta-Gamma neutral profit and loss standard deviation which is a good result. The Delta-Gamma Neutral hedge standard deviation is so large as the Gamma of the European call option used to hedge is much different to the autocallable note's Gamma near the barriers. This leads to large purchases of the hedging option, and large purchases or shorting of the underlying asset.

4.3 Experiment 3: Delta-Gamma Hedging, Optimal Hyperparameter

In the final experiment multiple models were trained to Delta-Gamma hedge an autocallable note with structure presented in Table 2. In order to tune hyperparameters a grid search was used, which is a loop through possible combinations of different hyperparameters. The models were trained over 1000 episodes, 2000 episodes, and 4000 episodes. These results can be presented in Table 3 where the columns in dictate the Actor learning rate and the rows are the critic learning rate. In each box the top number is the mean and the bottom number is the standard deviation of their profit and loss distribution.

Table 3: Hyperparameter Tuning Grid Search Results

Mean And Standard Deviation Test Result, 1000 Episodes	Actor Learning rate: 1-e4	Actor Learning rate: 5-e5	Actor Learning rate: 1-e5
Critic Learning rate: 1-e4	0.02782, 15.9024	0.03498, 15.2864	-0.34987, 16.2928
Critic Learning rate: 1-e5	0.02485, 14.7812	-0.1005, 15.0345	0.0846, 16.336

Critic Learning rate: 1-e6	0.3498, 15.7342	-0.1387, 15.0492	0.0945, 17.3561
Mean And Standard Deviation Test Result, 2000 Episodes	Actor Learning rate: 1-e4	Actor Learning rate: 5-e5	Actor Learning rate: 1-e5
Critic Learning rate: 1-e4	0.02997, 17.2483	0.29386, 18.3462	0.2343, 18.9487
Critic Learning rate: 1-e5	-0.04846, 15.3498	-0.2938, 16.2984	0.03496, 15.29386
Critic Learning rate: 1-e6	-0.03785, 14.0628	0.0585, 14.0134	-0.02385, 13.9085
Mean And Standard Deviation Test Result, 4000 Episodes	Actor Learning rate: 1-e4	Actor Learning rate: 5-e5	Actor Learning rate: 1-e5
Critic Learning rate: 1-e4	-0.06573, 17.38724	0.4774, 16.2873	-0.04283, 20.237
Critic Learning rate: 1-e5	0.0467, 15.373	-0.34786, 16.493846	0.03476, 16.4879
Critic Learning rate: 1-e6	0.06885, 12.0486	-0.08985, 13.0384	0.04242, 10.4536

The optimal hyperparameters found for this experiment were for the DDPG model trained over 4000 episodes, with an actor learning rate of 1-e5 and a critic learning rate of 1-e6. A note for this result is that the optimal hyperparameters are at the highest number of episodes tried as well as the lowest critic and actor learning rates. This indicates that greater episodes should be tried to see if they are more optimal. However greater episodes with a smaller learning rate come at the cost of increased training time. Therefore, it can be justified that this optimal hyperparameter is sufficient for the goal of this experiment. Its results were compared to the Delta-Gamma neutral model by using the test procedure outlined in Section 3.8. Figure 13 is the profit and loss distribution for the RL hedging model and Figure 14 is the profit and loss distribution of the

Delta-Gamma Neutral hedge model. Both profit and loss distributions were consisting of 4000 trials.

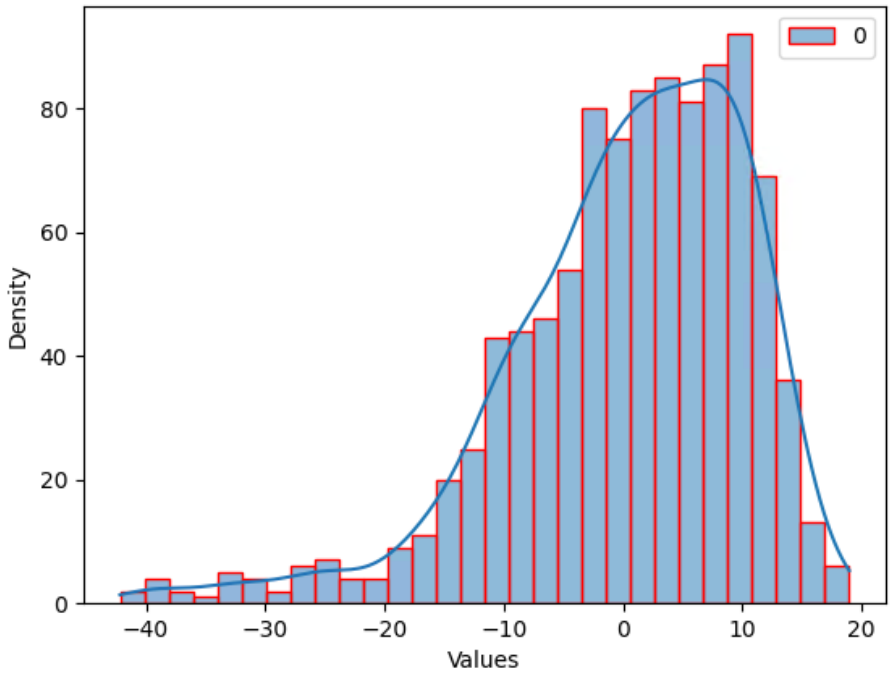

Figure 13: Profit and Loss Distribution for RL Delta-Gamma Hedging Model

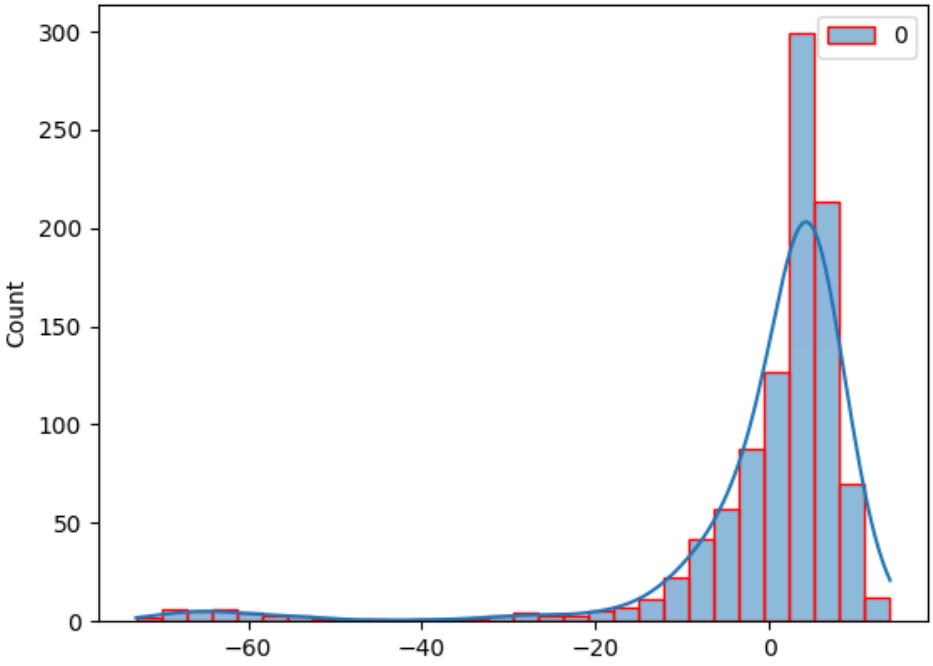

Figure 14: Profit and Loss Distribution of Delta-Gamma Neutral Hedging Model

The mean of the profit and loss distribution presented in Figure 13 is -0.02611 and the standard deviation is 10.1114. The mean of the profit and loss distribution presented in Figure 14 is 0.3346 and the standard deviation is 12.9410. Both means are close to zero which is the expected result as a bias to one side would indicate improper pricing and the presence arbitrage. The standard deviation of the RL profit and loss distribution is a little smaller than the Delta-Gamma neutral profit and loss standard deviation which is a good result. The standard deviation of the Delta-Gamma neutral model is mu closer to the RL model's because in the previous experiment the coupon was much larger at 10% whereas the coupon for this autocallable note is 2%. This makes the Gamma between the hedging option and the autocallable note near the barrier more similar, so less extreme hedging purchases are made. The Result of this experiment is that a DRL model can be trained to Delta-Gamma hedge an autocallable note with a realistic structure and that it can perform competitively with the Delta-Gamma neutral hedge.

4.4 Summary

The summary of the results from experiment 1 is that the viability of a RL hedging model on an autocallable note with structure presented in Table 1 was proven as the RL hedging model was able to achieve a lower standard deviation compared to the Delta neutral model. The results from experiment 2 show the viability of a RL hedging agent to Delta-Gamma hedge an autocallable with structure presented in Table 1. As the RL hedging model was able to achieve a lower standard deviation in its profit and loss distribution compared to the Delta-Gamma hedging model. Finaly the result of the third experiment, is the testing of optimal hyperparameters of the RL hedging agent to Delta-Gamma hedge an autocallable note with structure presented in Table 2. The result of the optimal hyperparameter model was compared to the Delta Gamma neutral model and achieved a lowered standard deviation in its profit and loss distribution.

5.0 CONCLUSIONS

In summary, this paper furthers the current Deep Reinforcement Learning (DRL) hedging field that is primarily focused on Vanilla options and extends it to focus on a structured note known as an autocallable note. This work utilized the Deep Deterministic Policy Gradient (DDPG) Algorithm for its DRL algorithm as it has presented successful use in the hedging field. With data generated using Monte Carlo simulations of the Geometric Brownian Motion (GBM) process, the DDPG model was trained for three different experiments. The first experiment trained a DDPG model to Delta hedge an autocallable note with the structure presented in Table 1. The result of that experiment showed the viability of DRL to Delta hedge an autocallable note. The second experiment trained a DDPG model to Delta-Gamma hedge an autocallable note with structure presented in Table 1. The result of this experiment also showed the viability of Delta Gamma hedging using a DRL model. Finally, the third experiment trained a DDPG model to Delta-Gamma hedge an autocallable note with structure presented in table2. The experiment also explored the optimal actor learning rate, critic learning rate and number of episodes trained over. The result of this experiment showed the viability of a DDPG model to Delta-Gamma hedge an autocallable note that is similar to notes present in the Real world. It also showed the importance of hyperparameters in training a DDPG model for hedging an autocallable note. These results imply that DRL agents can be used effectively to hedge different autocallable notes. They also suggest lots of opportunity for further research outlines in the section below.

6.0 FUTURE WORK

There is ample opportunity for further research into DRL hedging autocallable notes. Suggested areas of further research would be exploring different DRL algorithms, exploring different autocallable note structures, exploring different data generation methods, further experiments for Delta hedging autocallable notes with Real world structures, and using different hedging options when Delta-Gamma hedging. DDPG is an effective and well used DRL algorithm. But there are many other RL algorithms that have their own unique advantages when it comes to efficiency, and stability. Exploration of value-based RL algorithms would be useful as DDPG is a policy-based RL algorithm. The structure of the autocallable note presented in Table 2 is more similar to structures present in the Real world. However, there are many other structures available. Two examples of autocallable note features that could be explored is coupon snowballing and using a basket of underlying assets instead of just 1. Coupon snowballing is when if a coupon is not received on an observation date it rolls onto the next observation date. This would have interesting and difficult affects on the Delta and Gamma of an autocallable note. Exploring the use of a DRL model to hedge these notes would provide valuable information about hedging these types of options. Often in the real-world autocallable notes rely on a basket of stocks as the underlying instead of just one stock. This greatly increase the complexity compared to just having one underlying. Exploring hedging these autocallable note structures using DRL would be very valuable as they are a common product. In this paper an experiment to test a DDPG model on delta hedging an autocallable note of structure presented in Table 2 was not done. It would be good to round out this experiment to test this. Finally, in this paper when Delta-Gamma hedging a European call option was used. I the future further research using different options to Delta-Gamma hedge would be useful to as they might be easier to hedge with. For example, hedging with another type of barrier option might have similar gamma values therefore making hedging with it easier.

REFERENCES

Achiam,J, and P Abbeel. 2018. *Introduction to RL.* Accessed March 2024. https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below.

Assa, H., C. Kenyon, and H. Zhang. 2021. "Assessing Reinforcement Delta Hedging." *SSRN preprint.*

Bellefroid, Maxime de. 2024. "The Derivatives Academy." 16.1-16.2.

Black, F, and M Scholes. 1973. "The Pricing of Options and Corporate Liabilities." *https://doi.org/10.1086/260062.* 81 (3): 637–54.

Cannelli, L., G. Nuti, M. Sala, and O. Szehr. 2023. "Hedging Using Reinforcement Learning: Contextual k-Armed Bandit versus Q-Learning." *J. Financ. Data Sci.* 9, 100101.

Cao, J., J. Chen, J. Hull, and Z Poulos. 2021. "Deep Hedging of Derivatives Using Reinforcement Learning." *J. Financ. Data Sci.* 3, 10-27.

Cao, J., J. Chen, S. Farghadani, J. Hull, Z. Poulos, Z. Wang, and J. Yuan. 2023. "Gamma and Vega Hedging Using Deep Distributional Reinforcement Learning." *Front. Artif. Intell* 6, 1129370.

Cassimero, G. 2023. *A Deep Dive into Actor-Critic methods with the DDPG Algorithm.* April. Accessed March 2024. https://medium.com/geekculture/a-deep-dive-into-the-ddpg-algorithm-for-continuous-control-2718222c333e#:~:text=The%20DDPG%20algorithm%20combines%20the,function%20(Q%2Dfunction).

Cui, Yeda, Lingfei Li, and Gongqiu Zhang. 2023. *Pricing and Hedging Autocallable Products by Markov Chain Approximation.* SSRN. doi:CUI, Yeda and Li, Lingfei and Zhang, Gongqiu, Pricing and Hedging Autocallable Products by Markov Chain Approximation (August 31, 2023). Available at SSRN: https://ssrn.com/abstract=4557397 or http://dx.doi.org/10.2139/ssrn.4557397.

Cuim, Yeda, Lingfei Li, and Gongqui Zhang. 2023. *Pricing and Hedging Autocallable Products by Markov Chain.* Social Science Research Network.

Deng, Geng, Joshua Mallett, and Craig McCann. 2011. *Modeling Autocallable Structured Products.* Journal of Derivatives and Hedge Funds. doi:Deng, Geng and McCann, Craig J. and Available at SSRN: https://ssrn.com/abstract=1981308.

Du, J., M. Jin, P.N. Kolm, G. Ritter, Y. Wang, and B Zhang. 2020. "Deep Reinforcement Learning for Option Replication and Hedging." *J. Financ. Data Sci.* 44-57.

Fathi, A., and B. Hientzsch. 2023. "A Comparison of Reinforcement Learning and Deep Trajectory Based Stochastic Control Agents for Stepwise Mean-Variance Hedging." *arXiv 2023, arXiv:2302.07996.*

Fries, C, and M Joshi. 2008. "Conditional Analytic Monte-Carlo Pricing Scheme of Auto-Callable Products." *Available at SSRN: https://ssrn.com/abstract=1125725 or http://dx.doi.org/10.2139/ssrn.1125725.*

Giurca, B., and S Borovkova. 2021. "Delta Hedging of Derivatives Using Deep Reinforcement Learning." *SSRN.*

Glasserman, P. 2003. *Monte Carl Methods In Financial Engineering.* New York: Springer.

Guillaume, T. 2015. "Analytical valuation of autocallable notes." *International Journal of Financial Engineering 2 (2).*

Halperin, I. 2019. "The QLBS Q-Learner Goes NuQLear: Fitted Q Iteration, Inverse RL, and Option Portfolios." *Quant. Financ.* 19, 1543–1553.

Hull, J. 2012. "Options, Futures, and Other Derivatives. Eighth edition." *Prentice Hall,.* https://search.library.wisc.edu/catalog/9910112878402121.

Kim, H. 2021. "Deep Hedging, Generative Adversarial Networks and Beyond." *arXiv 2021, arXiv:2103.03913.*

Kolm, P, and G Ritter. 2019. "Dynamic Replication and Hedging: A Reinforcement Learning Approach." *The Journal of Financial Data Science* 159-171.

Koster, F, and A Rehmet. 2018. "Monte Carlo Payoff Smoothing for Pricing Autocallable Instruments." *Journal of Computational Finance, Vol. 21, No. 4, Available at SSRN: https://ssrn.com/abstract=3122314.*

Lee, M, and J Hong. 2021. "Semi closed-form pricing autocallable ELS using Brownian bridge." *Communications for Statistical Applications and Methods 28 (3)* 251–265.

Li, Yuqian, and Leyi Zheng. 2022. "A binomial pricing method for snowball autocallable." *Proc. SPIE 12330, International Conference on Cyber Security, Artificial Intelligence, and Digital Economy (CSAIDE 2022), 123300N (23 August 2022); https://doi.org/10.1117/12.2646572.*

Lillicrap T., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra. 2015. "Continuous control with deep reinforcement learning." *arXiv:1509.02971, https://doi.org/10.48550/arXiv.1509.02971.*

Mikkilä, O., and J. Kanniainen. 2023. "Empirical Deep Hedging." *Quant. Financ.* 23, 111–122.

Minh, V. 2013. "Playing Atari With Deep Reinforcement Learning." *NIPS Deep Learning Workshop,.*

Murray, P., B. Wood, H. Buehler, M. Wiese, and M. Pakkanen. 2022. "Deep Hedging: Continuous Reinforcement Learning for Hedging of General Portfolios across Multiple Risk Aversions." *In Proceedings of the ICAIF' 22: Proceedings of the Third ACM*

International Conference on AI in Finance, New York, NY, USA, 2–4 November 2022; Association for Computing Machinery: New York, NY, USA, 361-368.

Noh, J. 2024. "Hedging Beyond the Mean: A Distributional Reinforcement Learning Perspective for Hedging Portfolios with Structured Products." *Available at SSRN: https://ssrn.com/abstract=4709441 or http://dx.doi.org/10.2139/ssrn.4709441.*

Pham, U., Q. Luu, and H. Tran. 2021. "Multi-Agent Reinforcement Learning Approach for Hedging Portfolio Problem." *Soft Comput.* 25, 7877–7885.

Pickard, R, and Y Lauryshyn. 2023. "Deep Reinforcement Learning for Dynamic Stock Option Hedging: A Review." *Mathematics, 11, 4943. https://doi.org/10.3390/math11244943.*

Sutton, R, and A Barton. 2018. *Reinforcement Learning, An Introduction, Second Edition.* Cambridge, Massachussetts: MIT press, ISBN: 978-0-262-19398-6.

Vittori, E., M. Trapletti, and M Restelli. 2021. "Option Hedging with Risk Averse Reinforcement Learning." *In Proceedings of the ICAIF' 20: Proceedings of the First ACM International Conference on AI in Finance, New York, NY, USA, 15–16 October 2020; Association for Computing Machinery: New York, NY, USA;.*

Watkins, C, and P Dayan. 1989. "Q-Learning." *Machine Learning, https://doi.org/10.1007/BF00992698* 279--292.

Xiao, B., W. Yao, and X. Zhou. 2021. "Optimal Option Hedging with Policy Gradient." *In Proceedings of the 2021 International Conference on Data Mining Workshops (ICDMW), Auckland, New Zealand* 1112–1119.

Xu, H, C Xu, H Yan, and Y Sun. 2023. "Structured products dynamic hedging based on reinforcement learning." *J Ambient Intell Human Comput 14, 12285–12295 (2023). https://doi.org/10.1007/s12652-023-04657-y* 12285–12295.

Xu, W., and B. Dai. 2022. "Delta-Gamma–Like Hedging with Transaction Cost under Reinforcement Learning Technique." *J. Deriv.* 29, 60–82.

Zheng, C., J. He, and C. Yang. 2023. "Option Dynamic Hedging Using Reinforcement Learning." *arXiv 2023, arXiv:2306.10743.*