Vrije Universiteit Amsterdam

Master Thesis Business Analytics

Option Hedging Under the Heston Model Using Deep Reinforcement Learning

Author:
Romy Rouwendaal

VU supervisor:
Asst. Prof. Vincent François-Lavet

VU Second Reader:
Prof. dr. Ger Koole

EY Supervisor:
M.Sc. Christophe Meunier Charette

July 5, 2021

Option Hedging Under the Heston Model Using Deep Reinforcement Learning

Romy Rouwendaal

Master Thesis Business Analytics

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

EY
Antonio Vivaldistraat 150
1083 HP Amsterdam

Preface

This thesis is written in fulfillment of the requirements for the degree of Master of Science in Business Analytics at the Vrije Universiteit Amsterdam (VU). Business Analytics combines insights from mathematics, computer science, and economics to improve products, services, and processes using data. This thesis focuses on the application of reinforcement learning to option hedging. The research is conducted during a six-month internship at EY. EY is a global organization that offers assurance, audit, tax, financial, and business advisory services to automotive, financial, government, entertainment, mining, real estate, technology, and telecommunication industries.

This thesis would not have been possible without the help, support, and guidance of several people. First of all, I would like to express my gratitude to EY for providing me the opportunity to conduct this research. In particular, I would like to thank my external supervisor Christophe Meunier Charette for his dedicated guidance and supervision during the fulfillment of this work. The frequent meetings and exchange of his insights and experiences were very meaningful and enabled me to gain knowledge in the field of quantitative finance. I would also like to express my gratitude and appreciation to Vincent François-Lavet, who gave me useful insights into the reinforcement learning domain, and whose support and advice have been contributed to this research. In addition, I would like to thank Jeroen van Kasteren, PhD Candidate at the VU, for his encouragement, discussions, and inspirational feedback on the writing of the thesis. Furthermore, I would like to thank Ger Koole for being the second reader. Lastly, I would like to thank Peter Stol, Scientific computing (hpc) lead at the VU, for providing me access to and guidance in working with a computational cluster at the VU. Without his help, some of the training jobs would still have been running at this time.

Executive summary

Problem definition - Hedging is a way to reduce financial risk exposure. Traditional hedging models are often based on unrealistic assumptions, such as continuous trading. These assumptions enable the construction of a portfolio that replicates a risk exposure perfectly at each point in time. However, the realistic setting of discrete trading and market frictions introduces hedging errors and transaction costs. To construct an appropriate portfolio, a trade-off has to be made at each point in time between both the error and the costs. Hence, the hedging task involves a sequential decision-making process and is amenable to reinforcement learning (RL). The goal of this research is to investigate to what extent RL can be used to hedge a plain vanilla call option, a digital option, and several types of the barrier call option under the Heston model.

Methodology - A separate RL agent is trained on a plain vanilla, digital, and different types of the barrier call option. The agents are trained on simulated data, as learning requires a lot of data. For this, it is assumed that the asset price is driven by a stochastic volatility process, described by the Heston model. To reflect realistic trading, time is discretized. The hedging objective is defined as a mean-variance optimization problem, which includes a risk aversion parameter and captures the trade-off between the expected wealth and the variance (i.e. risk) resulting from hedging. The hedging problem is embedded in a Markov Decision Process to recast it as an RL problem. The actor-critic Deep-Deterministic-Policy-Gradient algorithm is used to train the agents. As the agents must hedge the options by only taking positions in the underlying asset, the hedging problem comes down to delta hedging and the performance of the RL approach is compared with this strategy. The terminal portfolio values of both strategies are compared based on 10,000 (almost surely) out-of-sample simulated stock price paths. The performance of the RL agent is evaluated for each option type in the absence of transaction costs, in an environment with costs, and for some modified input parameters.

Results - The results show that it is possible to use RL to hedge a plain vanilla, digital, up-and-in, and down-and-out barrier call option. Both in the absence and presence of transaction costs, the agent is able to hedge these options more optimally than the delta hedging strategy. In this research setting, RL is less effective in hedging an up-and-out barrier call option and the agent did not learn how to hedge a down-and-in barrier option.

Recommendation - The application of RL to option hedging seems to be very promising. A key strength of this approach is that it is more flexible compared to traditional hedging models since the RL agent does not make any assumptions about the environment. For further research, it is recommended to improve the practicability and generality of the RL approach. It would for example be useful to investigate the application of RL to the hedging of more complicated option types, a portfolio of options, and using historical data.

Contents

Mathematical Notations and Abbreviations

$\mathbb{E}(\cdot)$	Expected value
$\mathbb{V}(\cdot)$	Variance
$\mathbb{P}(\cdot)$	Probability
$\min_x$	Minimum over variable x
$\max_x$	Maximum over variable x
$\operatorname{argmin} f(x)$	The value of x for which the value of the function $f(\cdot)$ is minimized
$\operatorname{argmax} f(x)$	The value of x for which the value of the function $f(\cdot)$ is maximized
$\log(x)$	Logarithm of x
$\mathbb{R}$	Set of real numbers
∇	Gradient, differential operator
$\sum_x$	Sum over values of x
$\int_x$	Integral over values of x
$\in$	Is an element of
$N(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
α	Learning rate
ϵ	Small value
θ, w	Model parameters
γ	Discount factor
μ	Mean
σ	Standard deviation
RL	Reinforcement learning
MDP	Markov Decision Process
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{T}$	Transition function
$\mathcal{R}$	Reward function
s, s'	States
a	Action
r	Reward
t	Discrete time step
dt	Infinitesimal change in time
Δt	Change in time
T	Final time step of an episode, option's maturity
s_t	State at time t
r_t	Reward at time t
a_t	Action at time t
$R(s, a, s')$	Instantaneous reward for taking action a in state s and ending up in state s'
$T(s, a, s')$	The probability of ending up in state s' when taking action a in state s
π	Policy
π^*	Optimal policy
$\pi(s)$	Action a taken in state s under *deterministic* policy
$\pi(a\|s)$	Probability of taking action a in state s under *stochastic* policy π
$\pi_\theta(a\|s)$	Probability of taking action a in state s under parameterized *stochastic* policy π
$\mu_\theta(s)$	Action a taken in state s under parameterized *deterministic* policy
$V^\pi(s)$	Value of state s under policy π
$V^*(s)$	Value of state s under optimal policy
$Q^\pi(s, a)$	Value of taking action a in state s under policy π
$Q^*(s, a)$	Value of taking action a in state s under optimal policy
$Q(s, a)$	Estimate of the value of action a in state s
$Q^w(s, a)$	Estimate of the value of taking action a in state s given parameters w
$J(\pi_\theta)$	Performance objective
$\rho^\pi(s)$	Discounted state distribution in the limit

D	Replay memory
C	Capacity
$\mathbb{P}$	Real-world (physical) probability measure
$\mathbb{Q}$	Risk-neutral probability measure
W	Wiener process
S	Price of a risky asset
B	Price of a riskless asset
K	Strike price
$h(S_T)$	Payoff function
$C_t(S_t)$	Option value
Δ	Delta, $\frac{\partial C(t,S_t)}{\partial S}$
(X_t, Y_t)	Portfolio consisting of X_t units of the risky asset and Y_t units of the riskless asset
$T-t$	Remaining time until the option expires (time to maturity)
κ	Risk-aversion parameter
H_t	Value of hedging strategy
c_t	Transaction costs
PnL_t	Profit or loss of hedging from time $t-1$ to t
Π_t	Portfolio value at time t, $C_t(S_t) - H_t$
UAI	Up-and-in barrier call option
UAO	Up-and-out barrier call option
DAI	Down-and-in barrier call option
DAO	Down-and-out barrier call option
OTM	Out-the-money
ATM	At-the-money
ITM	In-the-money
BM	Brownian Motion
GBM	Geometric Brownian Motion
BS	Black-Scholes

1 Introduction

Due to instability and variability in the financial market, firms are exposed to a variety of risks, such as market risk (including currency risk, interest rate risk, and price risk), credit risk, and liquidity risk. Managing these risks is a critical factor in company success, as they can negatively impact the financial performance of a firm [87]. Financial risk exposure can be reduced by hedging, which uses financial instruments that properly offset changes in the fair value or cash flows of the hedged item. Hedging typically involves the use of financial derivatives, which are financial contracts whose values are derived from other financial assets, like forwards, futures, swaps, and options [70].

Since the price of financial assets changes over time, frequent rebalancing of the hedged positions may be required. This is called dynamic hedging. Most derivative pricing models, such as the well-known Black-Scholes model [9], are complete market models. In a complete market, there are no transaction costs and there is a price for every derivative security in every possible state of the world. Option pricing models also generally require the assumption of continuous time trading[1]. However, the presence of market frictions, such as transaction costs, liquidity constraints, and inseparable positions and risks, is much more realistic [72]. In this setting, continuous trading of arbitrarily small amounts of the underlying asset is not possible or can be costly. As a result, the hedged portfolio is only rebalanced at discrete times. This creates a replication error since the hedged position does not always perfectly replicate the financial contract. Therefore, a trade-off has to be made at each point in time between the replication error costs and the transaction costs. The optimal hedging strategy involves thus a dynamic multi-stage decision-making, taking the possible future states of the world and the corresponding actions in these states into account. Considering hedging as a sequential decision-making process makes it amenable to reinforcement learning (RL).

RL is an area of machine learning concerned with how an intelligent agent should take actions in an environment to maximize its expected rewards. This research focuses on the application of RL to the hedging of equity options. These are option contracts that give the holder the right, but not the obligation, to buy or sell an underlying stock at an agreed-upon price and date. At each point in time, the RL agent should take a position in the underlying stock to properly offset the changes in the option value. The advantage of the RL approach is that the agent does not need any prior information about the option characteristics or stock price dynamics. It learns what positions to take by interacting with an environment: at each point in time, the agent takes an action that is applied to an environment, which in turn provides feedback to the agent on how good or bad the action taken is. Starting with random actions, the agent improves his policy based on this feedback.

Research about reinforcement learning for option hedging is quite novel. Previous research, such as [13][14][65][80], has mainly focused on vanilla options, which are options without any additional features. It is known that options with a more complicated structure or additional terms, such as a digital call option[2] and a barrier call option[3], can be harder to hedge. Reinforcement learning may be useful for such options. In addition, some of the earlier studies such as [36][80] modelled the stock prices with constant volatility. Stochastic volatility is more representative of the real market and can be modelled using e.g. the Heston model [37]. Because of the potential and novelty, this research aims to investigate to what extent reinforcement learning can be used to hedge a plain vanilla, digital and different types of a barrier call option under the Heston model.

[1] A market in which traders can trade at any time when the market is open.

[2] An option type that pays out a fixed amount of cash if the underlying asset price ends above or on the option's strike price.

[3] An option type that can only be exercised if the underlying asset price has reached or has not reached some level.

The remainder of this paper is structured as follows. Chapter 2 gives a brief introduction to reinforcement learning and discusses some of the relevant algorithms in more detail. Chapter 3 explains the concepts of options and hedging and introduces the relevant financial models to model asset prices and price options. Chapter 4 discusses related work that applied reinforcement learning to option hedging. Chapter 5 explains how the option hedging problem is recast as a reinforcement learning problem and outlines the methodology for constructing the reinforcement learning algorithm, as well as the evaluation of the model. The experimental setup is described in Chapter 6. The results are shown and discussed in Chapter 7. Finally, Chapter 8 concludes this research and Chapter 9 provides a discussion of the paper.

2 Reinforcement Learning

Humans make thousands of decisions per day to choose e.g. products, services, careers, and to survive as species. Typically, more than one decision is involved in decision-making. These situations are called *sequential decision-making*. The decisions have long-term effects, but might not all encompass an immediate reward. For example, when cooking a cake, the ingredients are added after each other to the dough. However, one does not know whether too much or too little is added until the cake is out of the oven [77]. Sequential decision tasks are often hard to solve because it might be difficult to infer each action's outcome and/or the set of actions might be too large to be presented to the decision-maker. Here, tools can help the decision-maker to differentiate between the possible actions and to focus on the preferred ones. Reinforcement learning is one of these ways.

Reinforcement learning (RL) is a subset of machine learning that enables an artificial agent to learn how to behave in a (stochastic) environment. The goal of the agent is to perform actions that maximize a numerical reward over time [43]. The agent should discover which actions yield the most reward by interacting with its environment. This is a trial and error process, where the agent performs a number of actions in the environment and receives feedback on the amount of reward that these actions yield. Actions may affect not only the immediate reward but also the opportunities available to the agent at later times and hence the subsequent rewards. As a consequence, an agent might choose an action that maximizes future rewards, although the immediate reward associated with this might look sub-optimal. To take the right action, the indirect, delayed consequences of actions should thus be taken into account which might require foresight or planning [75]. Here, the agent uses its experience to improve its performance over time.

Reinforcement learning is different from the other two machine learning paradigms: supervised learning and unsupervised learning. In contrast to supervised learning, an RL agent is not trained on a training set of labeled examples. Although unsupervised learning and RL both involve the training on unlabeled data, their goals are different. Unsupervised learning intends to discover hidden patterns in the data. Although uncovering patterns in an agent's experience can be useful in RL, it does not necessarily lead to an optimal policy that maximizes the long-term reward [75].

This chapter gives a brief introduction to RL. First, the formal framework of a Markov Decision Process (MDP) is defined, which provides a mathematical framework for modeling decision-making, accompanied by the definition of value functions and policies. Several relevant RL algorithms are then discussed.

2.1 Formulation via Markov Decision Process (MDP)

In reinforcement learning (RL), an agent observes a state s_t of the environment and takes an action a_t at each point in time. This action changes the environment. The agent observes then a new state s_{t+1} and receives a reward r_t from the resulting environment. In general, it is expected that the environment is stochastic, i.e. taking the same action in a given state on two separate occasions may yield a different subsequent state or reward. A representation of the agent-environment interaction is shown in Figure 1.

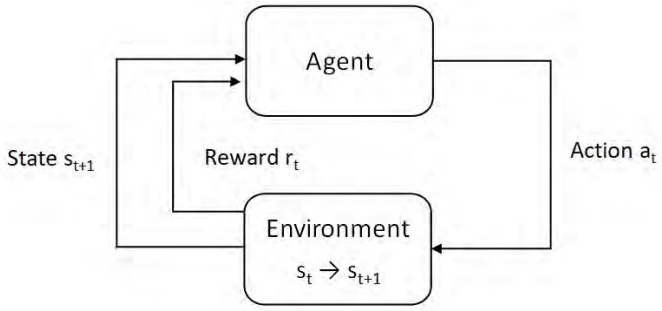


Figure 1: Representation of the agent-environment interaction in reinforcement learning, adapted from [75].

The reinforcement learning problem can be modelled as a Markov decision process (MDP). An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, consisting of [84]

- The **state space** $\mathcal{S}$. It defines the set of possible states which can be discrete or continuous.
- The **action space** $\mathcal{A}$. It defines the set of possible actions which can be discrete or continuous.
- The **transition function** $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. It probabilistically specifies the next state of the environment $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. The probability is represented by $T(s, a, s')$.
- The **reward function** $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. It specifies the instantaneous reward for taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and ending up in state $s' \in \mathcal{S}$. The reward is represented by $R(s, a, s')$.
- The **discount factor** $\gamma \in [0, 1]$. This value indicates the importance of future rewards. The agent prioritizes rewards in the immediate future if this value is close to zero. If γ approaches 1, future rewards are more strongly taken into account than with a lower γ. If the time horizon is finite and the rewards bounded, $\gamma = 1$ is theoretically possible. If the time horizon is infinite, $\gamma < 1$ ensures convergence of the sum of future rewards [75].

The model is Markovian if the state transitions only depend on the current state and not on any previous actions or states, i.e. $\mathbb{P}\big(s_{t+1} \big| s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0\big) = \mathbb{P}\big(s_{t+1} \big| s_t, a_t\big) = T(s_t, a_t, s_{t+1})$. This means that the current state contains all relevant information that is important to make a decision [84].

2.1.1 Solving Markov Decision Processes

Solving a Markov Decision Process (MDP) means computing an optimal policy π^*. A policy $\pi \in \Pi$ represents the strategy of which action $a \in \mathcal{A}$ to take in state $s \in \mathcal{S}$. A policy can either be deterministic or stochastic. A deterministic policy is a function defined as $\pi : \mathcal{S} \rightarrow \mathcal{A}$ and outputs an action for a particular state. A stochastic policy is defined as $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ and outputs a probability distribution over actions for a particular state. Deterministic policies, as well as a finite action and state space, are considered first.
The optimal policy is often computed by learning value functions. There are two types of value functions:

- The **state value function** $V^\pi(s) : \mathcal{S} \rightarrow \mathbb{R}$ represents an estimate of how good it is for an agent to be in a given state. The discounted expected return when starting in

state $s \in \mathcal{S}$ and following policy $\pi \in \Pi$ thereafter can be defined as

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^i r_{t+k} | s_t = s, \pi\Big], \ \gamma \in [0, 1). \tag{1}$$

- The **state-action value function** $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents an estimate of how good it is for an agent to take a certain action in a certain state. The discounted expected return when taking an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and following policy $\pi \in \Pi$ thereafter can be defined as

$$Q^{\pi}(s, a) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^i r_{t+k} | s_t = s, a_t = a, \pi\Big], \ \gamma \in [0, 1). \tag{2}$$

Value functions satisfy certain recursive properties that can be expressed in terms of the so-called *Bellman Equation* [6]. For example for the state-value function, the expected value of a state can be decomposed into the instantaneous reward and the weighted discounted rewards of the possible subsequent states:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \ | s_t = s, \pi\Big] \\ &= \mathbb{E}\Big[r_t + \gamma V^{\pi}(s_{t+1}) | s_t = s, \pi\Big] \\ &= \sum_{s' \in \mathcal{S}} T(s, \pi(s), s')\Big(R(s, \pi(s), s') + \gamma V^{\pi}(s')\Big). \end{aligned} \tag{3}$$

The optimal value function is unique and can be expressed in the *Bellman optimality equation*. This equation states that the value of a state under an optimal policy must be equal to the expected return for the best action in that state:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s, a, s')\Big(R(s, a, s') + \gamma V^*(s')\Big). \tag{4}$$

Using the optimal value function, the optimal policy selects the action that has the highest expected return based on the successor states:

$$\pi^*(s) = \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s, a, s')\Big(R(s, a, s') + \gamma V^*(s')\Big). \tag{5}$$

For the state-action value function, the optimal value and policy can be derived in the same way and these are defined as:

$$Q^*(s, a) = \sum_{s' \in S} T(s, a, s')\Big(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')\Big), \tag{6}$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s, a). \tag{7}$$

Q^* and V^* are related in the following way:

$$Q^*(s, a) = \sum_{s' \in S} T(s, a, s')\Big(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} V^*(s')\Big), \tag{8}$$

$$V^*(s) = \max_{a' \in \mathcal{A}} Q^*(s, a). \tag{9}$$

An important difference between both value functions is that the Q-function does not sum over the transition probabilities to select an optimal action as in Equation 5. This makes the access to the transition function unnecessary; (an estimate of) the Q-function suffices.

Different algorithms have been developed to find the optimal policy π^*. An important difference between these is the distinction between model-based and model-free algorithms. This difference is visually shown in Figure 2. Model-based techniques aim to find the optimal strategy in the presence of a model of the MDP. This model consists of knowledge about the transition function and the reward function. It may be known in advance or can be learnt by interacting with the environment. The existence of a model allows an agent to explicitly plan ahead by considering the long-term outcomes of the possible actions and to more carefully select actions. In this way, planning can be used to construct a value function or policy. However, in practice, the planning can be computationally expensive and it is often not possible to find an exact model of the environment [62]. Model-free algorithms do not use such a model. Instead, they rely only on the interaction with the environment. As model-free methods are more popular and have been more extensively developed and tested than model-based methods, the next section discusses some model-free algorithms.

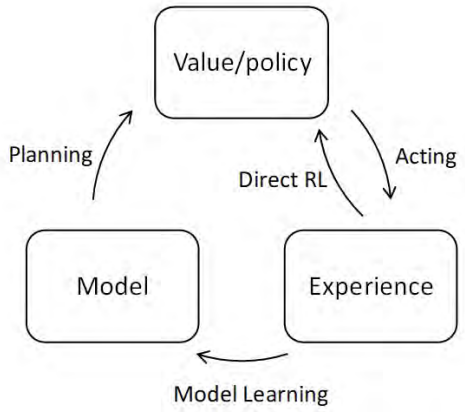


Figure 2: Representation of model-based and model-free reinforcement learning, adapted from [75].

2.2 Model-free Methods

Reinforcement learning is mainly concerned with approaches that do not rely on learning a model: model-free methods. Because a model of the Markov Decision Process (MDP) is unknown, the agent must try out different actions to obtain information from the environment and to be able to produce an optimal policy. This results in an exploration-exploitation trade-off. To maximize the reward, the agent must prefer actions that it has tried in the past and led to high rewards (*exploitation*). At the same time, it should discover actions that it has not selected (often) before which may result in making better decisions in the future (*exploration*) [47]. By balancing the trade-off, the agent tries to learn or estimate the value functions (*value-based*) or policy (*policy-based*) directly. Here, the transition and reward functions are captured implicitly. The following subsections discuss some of the value-based en policy-based algorithms in more detail.

2.2.1 Value-based

2.2.1.1 Temporal Difference: Q-learning

Value-based methods are exemplified by the temporal difference (TD) method [34]. Temporal difference was developed by Richard Sutton in 1988 [76]. It is based on bootstrapping: the value of a state ($V(s)$) can be learnt based on the estimates of other values. Using the Bellman equation (see Equation 3), $V(s)$ can be expressed in the immediate reward and the

current estimate of the value function of the next state [75]:

$$V(s) = r + \gamma V(s'). \tag{10}$$

As shown in the previous subsection, computing an optimal action in a state based on the learnt V-function requires a weighted summation of all successor states using the transition function. As this function is often unknown in model-free algorithms, the Q-function is learnt instead. Combining Equation 10 with the relationship between $Q^*(s,a)$ and $V^*(s)$ as presented in Equation 9 gives a similar expression for the Q-function:

$$Q(s,a) = r + \gamma \max_{a' \in \mathcal{A}} Q(s',a'). \tag{11}$$

The updates are performed iteratively after experiencing a transition from state s to s', based on the action a, while receiving reward r. This transition can be summarized into the *experience tuple* (s,a,r,s'). Using the update rule in Equation 11, the estimate of $Q(s,a)$ is each step fully updated. The use of a learning rate $\alpha < 1$ creates a "smoother" update. This rate determines to what extent the estimate is updated. Including the learning rate in Equation 11 gives the update rule used by the Watkins' Q-learning algorithm [82]:

$$Q(s,a) = Q(s,a) + \alpha\Big(r + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a)\Big). \tag{12}$$

The pseudocode of the Q-learning algorithm is presented in Algorithm 1. If the states and actions are respectively visited and performed frequently enough and if the learning rate is adjusted properly, $Q(s,a)$ is guaranteed to converge to $Q^*(s,a)$.

Algorithm 1 Q-learning [82]

Initialise action-value function $Q(s,a)$ arbitrarily
for each episode=1,..,M **do**
  Initialise state s
  for each step of episode t=1,..,T **do**
    With probability ϵ select a random action a, otherwise select $a = \max_a Q(s,a)$
    Execute action a and observe reward r and state s'
    Set $Q(s,a) = Q(s,a) + \alpha\Big(r + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a)\Big)$
    Set $s = s'$
  until s is terminal

2.2.1.2 Q-learning with Function Approximator: Deep Q-learning

Q-learning uses a tabular representation (vector or matrix) to store the Q-values for each state-action pair and to estimate the optimal Q-function. Such a representation is infeasible if a reinforcement learning problem has a large action space and/or a large or continuous state space. The state-action value function $Q^\pi(s,a)$ can then be approximated by representing it with a parameterized function $Q^w(s,a)$ such that

$$Q^w(s,a) \approx Q^\pi(s,a), \tag{13}$$

where w represents the set of parameters of the function approximator.

Deep Q-learning uses a deep[4] neural network $Q^w(s,a)$ as a parameterized estimate of the

[4] "Deep" means that the artificial neural network (ANN) has multiple layers between the input and output layers.

state-action value function, where w are the weights of the neural network. This network is called a Deep Q-network (DQN) and maps states to Q-values for all possible actions. The Q-network can be trained by minimizing the error between the estimated $Q^w(s,a)$ and the true Q-function $Q(s,a)$. Since the true Q-function is not known, an estimate for $Q(s,a)$ via $Q(s',a')$ as shown in Equation 11 can be used as target value. However, the target, $r + \gamma \max_{a' \in A} Q^w(s',a')$, and the prediction, $Q^w(s,a)$, are dependent as they both rely on w. To make the training more stable, a separate network is used to estimate the target: $Q^{\tilde{w}}$. This network has the same architecture as the main Q-network, but different weights. The weights of the main Q-network are updated by minimizing the loss:

$$L = \left(r + \gamma \max_{a' \in A} Q^{\tilde{w}}(s',a') - Q^w(s,a)\right)^2. \tag{14}$$

The Q-network's weights are copied to the target network every N iterations. The target network's weights are held fixed between the updates. The architecture of this algorithm is depicted in Figure 3 and the pseudocode is presented in Algorithm 2.

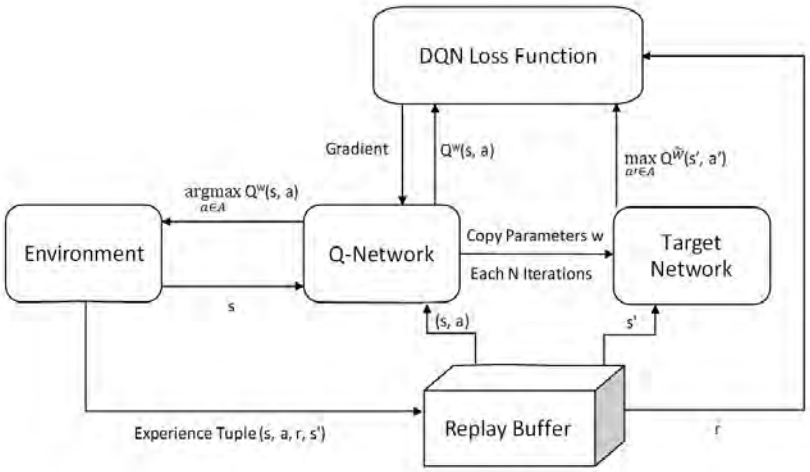


Figure 3: A flowchart of the Deep Q-network (DQN) algorithm with a replay buffer and a target network, adapted from [58].

A second technique applied by deep Q-learning to enhance the learning stability is the experience replay mechanism [50]. Subsequent experiences (s,a,r,s') are highly correlated, as the environment changes slowly over time. Learning from each of these experience tuples in sequential order violates the assumption that the data are independent and identically distributed (i.i.d.), made by the optimization algorithm which is used to train the networks. With experience replay, experience tuples are stored in a first-in-first-out (FIFO) buffer D with finite capacity C. Mini-batches of tuples are sampled uniformly at random from the buffer when training the neural network. This does not only remove correlations among the observations, but also increases the sample efficiency, as it allows to learn multiple times from individual experience tuples.

Algorithm 2 Deep Q-learning with Experience Replay and Target Network [56]

Initialise replay memory D to capacity C
Initialise action-value network $Q^w(s,a)$ with random weights w
Initialise target action-value network $Q^{\tilde{w}}$ with weights $\tilde{w} = w$
for each step of episode t=1,..,M **do**
  Initialise state s_t
  for each step of episode t=1,..,T **do**
    With probability ϵ select a random action a_t, otherwise select $a_t = \max_a Q^w(s_t, a)$
    Execute action a_t and observe reward r_t and state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Set $s_{t+1} = s_t$
    Sample a random mini-batch of N transitions (s_j, a_j, r_j, s_{j+1}) from D

$$\text{Set } y_j = \begin{cases} r_j \text{ for terminal state } s_{j+1} \\ r_j + \gamma \max_{a'} Q^{\tilde{w}}(s_{j+1}, a') \text{ for non-terminal state } s_{j+1} \end{cases}$$

    Update the Q-function by minimizing the loss: $L = \frac{1}{N} \sum_j \big(y_j - Q(s_j, a_j)\big)^2$

2.2.2 Policy-based

The algorithms described so far can only learn a single deterministic action from a discrete set of actions, i.e. a deterministic policy in discrete action space. In problems with a continuous action space, policy-based algorithms have proven more practical [67]. Instead of deriving the policy from a value function, policy-based methods directly model and optimize the policy. The policy is usually modelled with a parameterized function with respect to a n-dimensional vector θ: $\Pi = \{\pi_\theta : \theta \in \mathbb{R}^n\}$. The goal is to find policy parameters that maximize a performance objective $J(\pi_\theta)$ which calculates the expected reward of a policy. The parameters are learnt by performing a policy search or gradient ascent. The class of algorithms that perform a policy search is exemplified by the evolutionary algorithm (EA) approach [34]. The remainder of this section focuses on the class of algorithms that performs gradient ascent, also known as policy gradient methods. The stochastic policy gradient theorem is discussed first, followed by the deterministic one.

2.2.2.1 Stochastic Policy Gradient

In gradient policy methods, the traditional approach to handling continuous action spaces has been to use a parameterized stochastic policy

$$\pi_\theta(a|s) = \mathbb{P}(a|s;\theta) \tag{15}$$

that describes a probability distribution of taking an action given a state associated with the policy. At each time step, a policy distribution $\pi_\theta(a|s_t)$ is constructed from which an action a_t is sampled, i.e. $a_t \sim \pi_\theta(\cdot|s_t)$. Here, the policy distribution is often represented with a normal distribution $N(\mu_\theta(s), \sigma_\theta(s))$. Starting from an initial state, an agent follows a policy to generate a sequence (*trajectory*) of states, actions and rewards $s_0, a_0, r_0, .., s_T, a_T, r_T$. The goal is to learn a policy that maximizes the expected return from the start distribution[74]:

$$J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(a|s) r(s,a) da ds = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\Big[r(s,a)\Big], \tag{16}$$

where

$$\rho^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s|s_0, \pi) \tag{17}$$

is the discounted state distribution (state visitation frequency) in the limit and

$$r(s,a) = \int_{s' \in \mathcal{S}} T(s,a,s')R(s,a,s'). \tag{18}$$

The policy parameters that maximize this objective function can be found using gradient ascent. Here, the parameters are moved in the direction of the performance gradient $\nabla_\theta J(\pi_\theta)$, i.e. the direction that leads to a greater cumulative reward:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}), \tag{19}$$

where k is the iteration number. Computing the performance gradient is hard, as it depends on the state distribution. This distribution is difficult to estimate due to uncertainty in the environment. The *Policy Gradient Theorem* (see [74] for a proof) simplifies this computation by changing $r(s,a)$ to $Q^\pi(s,a)$ and moving the gradient operator inside the integral:

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) da ds. \tag{20}$$

In this way, the policy gradient ($\nabla_\theta \pi_\theta(s,a)$) does not depend on the gradient of the state distribution and can be estimated from experience. For this, the likelihood ratio trick can be applied to estimate gradients from expectations:

$$\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} = \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s). \tag{21}$$

Combining Equation 20 and 21 gives:

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a) da ds \\ &= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} \Big[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a) \Big]. \end{aligned} \tag{22}$$

2.2.2.2 Deterministic Policy Gradient

Contrary to prior beliefs, Silver et al. [71] showed that policy gradient for deterministic policies

$$a = \mu_\theta(s) \tag{23}$$

exists. Using the discounted state distribution ρ^μ analogously to the stochastic one, the performance objective $J(\mu_\theta)$ can again be written as an expectation:

$$J(\mu_\theta) = \int_{\mathcal{S}} \rho^\mu(s) r\big(s, \mu_\theta(s)\big) ds = \mathbb{E}_{s \sim \rho^\mu} \Big[r\big(s, \mu_\theta(s)\big) \Big]. \tag{24}$$

Silver et al. provided a *Deterministic Policy Gradient Theorem* analogously to the Policy Gradient Theorem to derive the deterministic policy gradient:

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) \big|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s \sim \rho^\mu} \Big[\nabla_\theta \mu_\theta(s) Q^\mu(s,a) \big|_{a=\mu_\theta(s)} \Big]. \end{aligned} \tag{25}$$

Expanding the gradient of the Q-value using the chain rule gives:

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s \sim \rho^\mu} \Big[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) \big|_{a=\mu_\theta(s)} \Big]. \tag{26}$$

It can be observed that the deterministic policy gradient integrates over the state space only, rather than over both the action and state space as the stochastic policy gradient does. As

a result, calculating the deterministic policy gradient may require fewer samples compared to the stochastic policy gradient.
Based on the Stochastic and Deterministic Policy Gradient Theorems, several policy gradient algorithms have been developed that estimate the expectation of the policy gradient by sampling [20]. These algorithms consist of a policy evaluation step and a policy improvement step. The former concerns the estimation of the $Q^{\pi}(s, a)$. The policy improvement step of the algorithm takes then a gradient step to optimize the policy with respect to this estimation. The estimation of $Q^{\pi}(s, a)$ should be bias-free and have a low variance to ensure convergence and a stable gradient [16]. A "simple" approach is to estimate it from the trajectories' returns while following policy π_{θ}. The REINFORCE algorithm [86] relies on this approach. A particular limitation of this algorithm is that it requires many trajectories and that it can exhibit high variance. A more efficient approach is to use a value-based approach to estimating Q^{π}, which is applied in actor-critic methods.

2.2.3 Actor-Critic

Actor-critic algorithms combine elements of value-based and policy-based methods [35]. They consist of two models which are learnt simultaneously. An *actor* (policy-based) learns a parameterized policy $\pi_{\theta}(a|s)$ or $\mu_{\theta}(s)$ and a *critic* (value-based) learns a parameterized value function $Q^{w}(s, a)$ The critic evaluates the actor's current policy and provides feedback (a reinforcing signal) to the actor. The policy evaluation can be done by e.g. temporal difference (TD) learning as described in Section 2.2.1. The actor updates its policy based on this signal, in the direction of performance improvement. This framework is visualized in Figure 4.

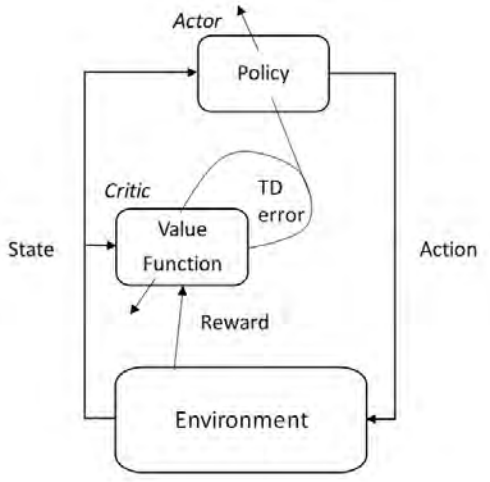


Figure 4: Representation of an actor-critic model, adapted from [81].

Several actor-critic methods have been developed for deterministic and stochastic policies. Among these, the Deep Deterministic Policy Gradient (DDPG) algorithm is discussed in the next paragraph.

2.2.3.1 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) [49] is an actor-critic, model-free algorithm that can handle continuous state and action spaces. DDPG combines Deep Q-Network (DQN) and Determistic Policy Gradient (DPG). It consists of an actor network $\mu^{\theta}(s)$, which learns a parameterized deterministic policy using DPG, and a critic network $Q^{w}(s, a)$

which is learnt using deep Q-learning. As in DQN, DDPG employs the use of a replay buffer and a separate target network to improve the stability of the learning. Next to the target Q-network ($Q^{\tilde{w}}(s,a)$), DDPG also has a target policy network ($\mu^{\tilde{\theta}}(s)$).

Compared to DQN and DPG, DDPG has two major adjustments. First, it applies soft updates on the parameters of the target network to the learnt networks:

$$\tilde{w} = \tau w + (1-\tau)\tilde{w}, \tag{27}$$

$$\tilde{\theta} = \tau\theta + (1-\tau)\tilde{\theta} \tag{28}$$

for some $\tau << 1$. In this way, the parameters change slowly, rather than being temporarily frozen as in DQN. This has proven to improve the learning stability.[5] Secondly, it adds a noise sampled from a noise process N to the deterministic policy:

$$a = \mu_\theta(s) + N. \tag{29}$$

This is to ensure exploration, as the deterministic policy gradient might not explore the full state and action space. The pseudocode is presented in Algorithm 3. The DDPG algorithm has been extended in several ways, such as the algorithms introduced in [4][29][52].

Algorithm 3 Deep Deterministic Policy Gradient (DDPG) [49]

Initialise replay memory D to capacity C
Initialise actor network $\mu^\theta(s)$ and critic network $Q^w(s,a)$ with random weights θ and w
Initialise target actor network $\mu^{\tilde{\theta}}(s)$ and target critic network $Q^{\tilde{w}}(s,a)$ with weights $\tilde{\theta} = \theta$ and $\tilde{w} = w$
for each episode t=1,..,M **do**
  Initialise a random process N for action exploration
  Receive initial state s_1
  for each step of episode t=1,..,T **do**
    Select action $a_t = \mu^\theta(s_t) + N_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Set $s_{t+1} = s_t$
    Sample a random mini-batch of N transitions (s_j, a_j, r_j, s_{j+1}) from D

$$\text{Set } y_j = \begin{cases} r_j \text{ for terminal state } s_{j+1} \\ r_j + \gamma Q^{\tilde{w}}\left(s_{j+1}, \mu^{\tilde{\theta}}(s_{j+1})\right) \text{ for non-terminal state } s_{j+1} \end{cases}$$

    Update the critic by minimizing the loss:

$$L = \frac{1}{N}\sum_j \left(y_j - Q^w(s_j, a_j)\right)^2$$

    Update the actor policy using the sampled policy gradient:

$$\nabla_\theta J \approx \frac{1}{N}\sum_j \nabla_a Q^w(s,a)\big|_{s=s_j, a=\mu(s_j)} \nabla_\theta \mu^\theta(s)\big|_{s=s_j}$$

    Update the target networks:

$$\tilde{w} = \tau w + (1-\tau)\tilde{w}$$

$$\tilde{\theta} = \tau\theta + (1-\tau)\tilde{\theta}$$

[5] This is because the learning task becomes more similar to a supervised learning problem, for which robust solutions exist.

3 Option Pricing and Hedging

This section explains the concepts of options and hedging. As these rely on the prices of financial assets, some approaches to model asset prices are introduced first in Section 3.1. The concept of an option is introduced in Section 3.2, as well as the characteristics of some option types. Section 3.3 discusses some methods to value options. Finally, the principle of hedging is explained in Section 3.4. As this research focuses on equity options, all parameters in these sections are positive real numbers (defined on $\mathbb{R}_+$), unless stated otherwise[6].

3.1 Asset Price Dynamics

Future prices of financial assets are uncertain and fluctuate over time. Their behavior can be modelled using a stochastic process X, which is a set of random variables that are time-dependent. Important classes of stochastic processes are Markov processes, in which the future development of the process only depends on the current realizations and not on the past history, i.e. [19]

$$\mathbb{P}\big(X_t = x_t \big| X_{t-1} = x_{t-1}, .., X_0 = x_0\big) = \mathbb{P}\big(X_t = x_t \big| X_{t-1} = x_{t-1}\big). \tag{30}$$

Other important classes are the Martingales, which are processes in which the expectation of X_t given the past is equal to the most recent observation i.e.

$$\mathbb{E}\big[X_t \big| X_{t-1}, .., X_0\big] = X_{t-1}. \tag{31}$$

The following subsections introduce some stochastic processes to model the dynamics of the asset price.

3.1.1 Brownian Motion

One of the fundamental building blocks of a stochastic processes is the Brownian Motion (BM). A BM, also called a Wiener process, is a real-valued stochastic process W with the following properties [22]:

- $W_0 = 0$.
- Independent increments: $W_t - W_s$ and $W_v - W_u$ are independent for any $0 \leq s < t \leq u < v$.
- Stationary increments: $W_t - W_s$ has the same distribution as W_{t-s} for any $0 \leq s < t$.
- Gaussian increments: $W_t - W_s \sim N(0, t-s)$ for any $0 \leq s < t$.
- Continuity: W_t is continuous in t.

Because of the stationary and independent increments property, the BM is a Markov process. It is also a martingale, since

$$\mathbb{E}\big[W_t \big| W_s\big] = \mathbb{E}\big[W_s \big| W_s\big] + \mathbb{E}\big[W_t - W_s \big| W_s\big] = W_s. \tag{32}$$

The BM cannot be used to describe the dynamics of a financial asset directly, as it allows for negative values. Instead, the non-negative variation of the BM, called the Geometric Brownian motion, is widely used.

[6]Note that in recent times, negative strike S and spot prices K have been observed for some commodities and derivatives, as well as a negative interest rate r. The models described in the following sections are in these cases still feasible, but adjustments may be required.

3.1.2 Geometric Brownian Motion

A Geometric Brownian Motion (GBM) with drift (mean) μ and volatility σ is a stochastic process S with intial value S_0 that can be described by the following SDE [22]:

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{33}$$

where S_t is the asset price at time t, μ is the drift (mean), σ is the volatility and W_t is a Wiener process. Using stochastic calculus[7], the logarithm of the asset price given an initial value S_0 can be described as

$$\ln(S_t) = \ln(S_0) + (\mu - \frac{1}{2}\sigma^2)t + \sigma W_t, \tag{34}$$

which is normally distributed: $\ln(S_t) \sim N\big(\ln S_0 + (\mu - \frac{1}{2}\sigma^2)t, \sigma^2 t\big)$. This yields the closed form solution for S_t:

$$S_t = S_0 e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W_t}. \tag{35}$$

The future asset price only depends on the current price and random noise. Hence, the future price is independent of the past, making the GBM a Markov Process. S_t is log-normally distributed with

- $\mathbb{E}[S_t] = S_0 e^{\mu t}$.
- $\mathbb{V}[S_t] = S_0^2 e^{2\mu t}(e^{\sigma^2 t} - 1)$.

S_t is not a martingale except for $\mu = 0$, as $\mathbb{E}[S_t|S_s] \neq S_s$. Classical option pricing models assume that the price of a financial asset behaves as a GBM. However, empirical studies show that the GBM is not suitable to model the dynamics of the price of assets adequately [73]. This is because GBM assumes constant volatility. This is not representative of the real market, where heteroscedastic volatility can be observed.

3.1.3 Stochastic Volatility

As the constant volatility assumption may lead to modelled prices that are not representative of the real market, several approaches have been proposed to capture the uncertainty in the behavior of the volatility of the asset's price [61]. The most well-known one is the Heston model introduced by Steven Heston in 1993 [37]. This model uses a GBM to describe the dynamics of the price of a financial asset:

$$dS_t = \mu S_t dt + \sqrt{\nu_t} S_t dW_t^S, \tag{36}$$

but the SDE involves a non-constant instantaneous variance ν_t. This instantaneous variance is modelled as a mean-reverting Cox-Ingersoll-Ross (CIR) process [1] in which the variance reverts to the long-time average θ with rate κ as defined in the following SDE:

$$d\nu_t = \kappa(\theta - \nu_t)dt + \epsilon\sqrt{\nu_t} dW_t^\nu, \tag{37}$$

where ϵ is the volatility of the instantaneous variance and dW_t^S and dW_t^ν are two correlated Wiener processes with correlation coefficient $\rho \in [-1, 1]$ such that

$$dW_t^S dW_t^\nu = \rho dt. \tag{38}$$

In continuous time, the parameters have to satisfy the so-called Feller constraint to ensure that the instantaneous variance does not become negative [1]:

$$2\kappa\theta > \epsilon^2. \tag{39}$$

[7] By applying Itô's lemma, see Equation 80, on $x = \ln(S)$.

Contrary to the GBM, a solution for S_t cannot directly be obtained, as the SDE for the asset price depends on a second SDE. Instead, the asset price can be obtained by applying a numerical approximation. This requires simulating *discretized* versions of the SDEs where the instantaneous variance is simulated first and used to simulate the asset price. Both paths are discretised using a discretization step size Δt resulting in $m = \lfloor \frac{T}{\Delta t} \rfloor$ discretization steps. The most simple and common discretization scheme is the Euler scheme [25][54]. Applying this scheme to the variance process yields

$$\nu_{t+1} = \nu_t + \kappa(\theta - |\nu_t|)\Delta t + \epsilon\sqrt{|\nu_t|}dW^{\nu}_{t+1}, \tag{40}$$

where dW^{ν}_{t+1} is the Brownian increment $W^{\nu}_{t+1} - W^{\nu}_t$. As this increment is normally distributed with mean 0 and standard deviation $\sqrt{\Delta t}$, one can simulate the increment by generating a standard normal random variable and multiplying it by $\sqrt{\Delta t}$. Note that this scheme involves the absolute value of the instantaneous variance ν_t to avoid negative values. It was shown in [85] that the asset price S_{t+1} is lognormally distributed, i.e. behaves as a GBM, conditional on the values of S_t and ν_t. Hence, the Heston process is a Markov process and the exact solution of the conditional asset price dynamics can be used to simulate the value of S_t [11]:

$$S_{t+1} = S_t e^{(\mu - 0.5|\nu_t|)\Delta t + \sqrt{|\nu_t|}dW^{S}_{t+1}}, \tag{41}$$

where

$$dW^{S}_{t+1} = \rho\, dW^{\nu}_{t+1} + \sqrt{1-\rho^2}\, dW_{t+1} \tag{42}$$

and dW^{ν}_{t+1} and dW_{t+1} are independent. The Heston model has been extended in several ways, such as the addition of jumps to the stock price process (stochastic volatility jump diffusion model [5]) or a state-dependent stochastic volatility (stochastic local volatility model [2]).

3.2 Option types

After having explained the concept of asset price modelling, this section continues with the principles of options. Options are contracts that give the holder the right to buy (*call option*) or sell (*put option*) an underlying asset for a certain price at or before a specific date [40]. The price of the underlying asset in the contract is called the *exercise price* or *strike price*, denoted by K. The date in the contract is called the *expiration date* or *maturity date*. The remaining time until the contract expires is denoted by $T-t$, where $t = 0$ is at inception and $t = T$ at maturity. An investor can take a long position or a short position in an option. The former involves buying the option and the latter selling it. Options can be traded on an exchange (exchange-traded options) or directly between two parties (over-the-counter). The market price of an option is called the option value.

There exists many different option types. A call/put option is often referred to as a vanilla option due to the lack of additional features. Any option with a more complicated payoff structure or additional terms or conditions is called an exotic option. Within these option types, a distinction can be made between European-style and American-style options. The former class consists of options that can only be exercised at maturity, whereas the options belonging to the latter can be exercised at any time before or at maturity. In theory, there exists an unlimited number of possible exotic options, but in practice, there are only a few that are often used [28]. Two of these are the digital option and the barrier option. The next subsections describe the characteristics of these options, next to these of the plain vanilla option. Here, the focus is on European-style call options.

3.2.1 Plain Vanilla Call Option

The European-style plain vanilla option has a single expiration date, exercise price, and no additional features [40]. As the holder of a plain vanilla call option has the right to buy the

underlying asset for price K at time T, it is profitable to exercise the option if the price of the underlying asset is above the strike price. If this is not the case, the option will not be exercised, as it is more profitable to buy the underlying asset at the current market price. The profit resulting from taking a position in an option is captured in the payoff function. For the plain vanilla call option, the payoff function, denoted by $h(S_T)$, is defined as:

$$h(S_T) = \begin{cases} S_T - K \text{ if } S_T \geq K \\ 0 \text{ if } S_T < K \end{cases} \tag{43}$$

The corresponding payoff diagram for a call option with strike $K = 100$ is shown in Figure 5. The possible gain of a vanilla call option is unlimited and the possible loss is limited by the option value as paid when entered into the contract.

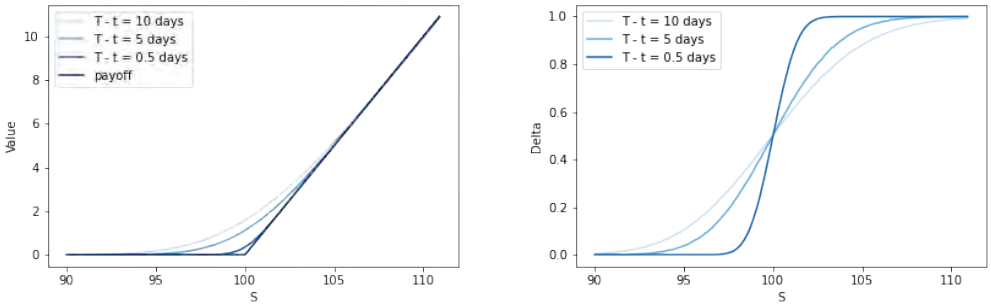


Figure 5: The value (left) and delta (right) of the plain vanilla call option as a function of the underlying asset price S, different maturities $T - t$ and a strike price of $K = 100$.

The sensitivity of the option value $C_t(S_t)$ with respect to the input parameters is measured by financial measures which are known as the *Greeks* [45]. The most important ones are the delta, gamma and vega, which are defined as:

- Delta: $\Delta_t = \frac{\partial C_t(S_t)}{\partial S}$.
- Gamma: $\Gamma_t = \frac{\partial^2 C_t(S_t)}{\partial S^2}$.
- Vega: $\nu_t = \frac{\partial C_t(S_t)}{\partial \sigma}$.

Given a certain change in the input parameter, the option price changes by this small amount multiplied with the respective Greek. For example, if the stock price changes by dS, the call price changes by (approximately) $dS\Delta$. As will be motivated in Section 3.4, the remainder of these subsections will focus on the delta, which measures the sensitivity of the option value with respect to a price change in the underlying asset.
For a vanilla call option, the delta ranges between 0 and 1. Its behaviour is shown in Figure 5. The delta is closely related to moneyness, which is defined as $\frac{S}{K}$. An option is [40]:

- In-the-money (ITM) if the option would be exercised (positive intrinsic value; $S > K$).
- At-the-money (ATM) if the option holder is indifferent between exercising and not exercising the option ($S = K$).
- Out-the-money (OTM) if the option would not be exercised (zero intrinsic value; $S < K$).

Call options that are far OTM have a delta near 0; the option value is not sensitive to changes in the underlying asset price, as it is very unlikely that the option will end up in the money. As the OTM option moves further away ITM, the option becomes more sensitive

to changes in the underlying asset, and the delta value increases. The delta of an ATM call option is close to 0.5 because there is approximately a 50% chance of the underlying asset going up and hence ending up ITM and a 50% chance of going down and ending up OTM. The more ITM the call option is, i.e. S_t is sufficiently greater than K, the more the option behaves like the underlying asset and hence the closer the delta becomes to 1, reflecting a one-to-one reaction to price changes in the underlying asset.

3.2.2 Digital Call Option

The digital call option is an exotic option which pays out a fixed amount of cash Q if the underlying asset price ends above or on the option's strike price at maturity [66]:

$$h(S_T) = \begin{cases} Q \text{ if } S_T \geq K \\ Q \text{ if } S_T < K \end{cases} \tag{44}$$

As the payoff can only take two potential values, the digital call option is also called a binary option; the digital option is basically a binary bet on the direction of the underlying asset price. The payoff diagram of a digital call option with a payout of $Q = 1$ and strike $K = 100$ is shown in Figure 6. It can be observed that the payoff is discontinuous in the underlying asset price. In addition, the payoff is equal to the delta of the vanilla call option. In other words, a digital option is an option on the delta of a vanilla call option and its delta is equivalent to the gamma of a vanilla call option. Furthermore, the payoff of the option remains the same once it expires ITM, irrespective of how deep ITM the option is. These particularities are also reflected in the behaviour of the option delta.

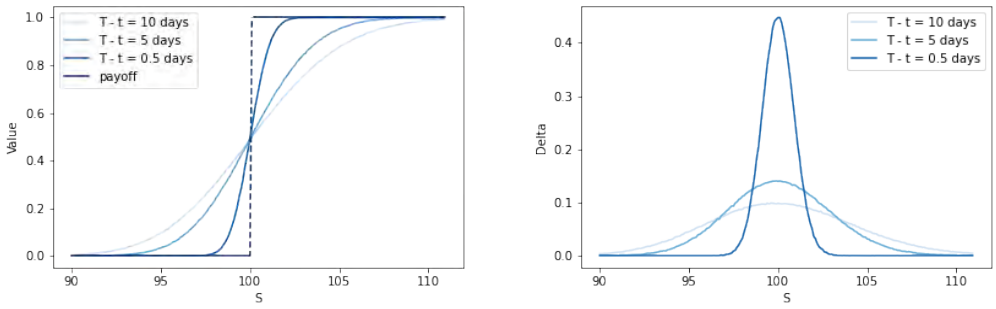


Figure 6: The value (left) and delta (right) of the digital call option as a function of the underlying asset price S, different maturities $T - t$, payout of $Q = 1$ and a strike price of $K = 100$.

The delta behaviour of the digital call option is illustrated in Figure 6. Just like for the plain vanilla call option, the delta of an OTM digital call option moves closer to zero as the option approaches maturity since the probability to end up ITM decreases. Contrary to the plain vanilla call option, the delta of the digital option also moves closer to zero when the option is (deep) ITM, rather than converging to 1. This is because once the underlying asset price is sufficiently greater than the strike, the trader is indifferent to a further price increase due to the fixed payoff. As the payoff function has a discontinuity point in $S = K$, the derivative of the present value of this payoff with respect to the asset price (i.e. the digital's delta) is infinite at this point. As time passes, the delta can take high values that change extremely fast as the asset price approaches the strike; a tiny move in the underlying asset price has a large impact on the price of the option. This may lead to high replication errors in discrete trading [78]. This, next to the fact that it is impossible to hold an infinite amount of the underlying asset in practice, makes the delta-hedging not a very appealing strategy to replicate the digital call option. In practice, the digital call option is often hedged as a call spread, more specifically a *bull spread*. This spread consists of a long position in

a European call option with an exercise price of $K - \epsilon$ and a short position in a European call option with an exercise price of $K + \epsilon$. The option value and delta of this strategy are calculated as follows:

$$C_{\text{spread}} = \frac{C(t, K - \epsilon) - C(t, K + \epsilon)}{2\epsilon}, \tag{45}$$

$$\Delta_{\text{spread}} = \frac{\Delta(t, K - \epsilon) - \Delta(t, K + \epsilon)}{2\epsilon}. \tag{46}$$

If $\epsilon \to 0$, the call spread's delta behaves as the digital call option's delta. For $\epsilon > 0$, the delta is smoother than that of a digital call.

3.2.3 Barrier Call Option

A barrier option is an exotic, path-dependent option [40]. The payoff is the same as that for a vanilla call, but only if the underlying asset price has reached or has not reached some predetermined barrier (trigger-level) during the option's lifetime. Barrier options can be divided into knock-out and knock-in options. A knock-out option is an option that is cancelled, i.e. becomes worthless, as soon as the barrier is reached. A knock-in option is an option that becomes activated as soon as, i.e. is worthless until, the barrier is reached. When initialising the option, the barrier can be set above or below the underlying asset price, meaning that the underlying asset price has to respectively rise above or fall below the barrier. The former is indicated by *up* and the latter by *down*. The four basic forms of the barrier option are *down-and-out (DAO)*, *down-and-in (DAI)*, *up-and-out (UAO)* and *up-and-in (UAI)*. Denoting the barrier by B, the minimum and maximum asset price during the option's lifetime by $S_{\min} = \min\{S_0, .., S_T\}$ and $S_{\max} = \max\{S_0, .., S_T\}$, the payoff structures of these options are [83]:

$$h(S_T)_{\text{DAO}} = \begin{cases} S_T - K \text{ if } S_T \geq K \text{ and } S_{\min} > B \\ 0 \text{ if } S_T < K \text{ or } S_{\min} \leq B \end{cases} \tag{47}$$

$$h(S_T)_{\text{DAI}} = \begin{cases} S_T - K \text{ if } S_T \geq K \text{ and } S_{\min} \leq B \\ 0 \text{ if } S_T < K \text{ or } S_{\min} > B \end{cases} \tag{48}$$

$$h(S_T)_{\text{UAO}} = \begin{cases} S_T - K \text{ if } S_T \geq K \text{ and } S_{\max} < B \\ 0 \text{ if } S_T < K \text{ or } S_{\max} \geq B \end{cases} \tag{49}$$

$$h(S_T)_{\text{UAI}} = \begin{cases} S_T - K \text{ if } S_T \geq K \text{ and } S_{\max} \geq B \\ 0 \text{ if } S_T < K \text{ or } S_{\max} < B \end{cases} \tag{50}$$

Barrier options are cheaper than the corresponding plain vanilla call option because of the higher probability of expiring worthless. The closer the barrier is set to the asset price at initiation, the higher the probability that the barrier is knocked out (knocked in) and thus the cheaper (more expensive) the option. This behaviour is also reflected in the in-out parity, which states that the sum of the prices of a European-style knock-in and knock-out barrier option with the same barrier and strike is equal to the price of the European vanilla option with that same strike price [55]. This means that before the barrier has been reached and prior to maturity, both knock-in and knock-out options have a positive value which is strictly below that of the corresponding plain vanilla call option. After reaching the barrier, the knock-out option is worthless (i.e. its value and delta are zero) and the knock-in option behaves (i.e. has the same delta and value) as the plain vanilla call option. If the plain vanilla call option has a positive payoff at maturity, either the knock-in option or the knock-out option has the same payoff, depending on whether or not the barrier has been reached during the lifetime of the options.

Although barrier options are attractive because of their lower price compared to vanilla options, they are more complicated to hedge. The payoff diagram of the UAO barrier call option with a barrier of $B = 105$ and strike $K = 100$ is shown in Figure 7. It can be

observed that the payoff function has a discontinuity point in $S = B$. This leads to a delta being very sensitive to changes in the underlying asset price around the barrier (i.e. the discontinuity point) near maturity. Whenever the barrier has not been hit, the delta of a DAI/UAO barrier call option can take both positive and negative values, which is visible in Figure 7. This is because an increase in the underlying asset price has two competing effects: it increases the probability of ending ITM, but decreases/increases the knock-in/knock-out probability. It can also be explained by the fact that the barrier options can be decomposed into vanilla call options and a digital call option. For example, a UAO barrier call option can be broken down into a long vanilla call option with strike K, a short vanilla call option with strike B, and a short digital call option with strike B. If the stock price is just below the barrier B, the digital call option is near-the-money, which translates into a *negative* delta spike. Changes from a positive to a negative delta at/before the barrier makes the hedging of respectively the DAI and UAO option even more difficult, as it would require an investor to take alternating long and short positions in the underlying asset. This is not the case for the UAI/DAO barrier call option, where a change in the underlying asset price increases both the chance of getting kicked in/out and ending up ITM/OTM. However, these options have a higher delta before/below the barrier compared to the corresponding vanilla call option because of their discontinuity in the payoff. This can also be explained by decomposing the payoff of these options into vanilla call options and digital call options. For example, a UAI barrier call option can be broken down into a long call option and digital call option, both with strike B. If the stock price is just below the barrier B, the digital call option is near-the-money, which translates into a *positive* delta spike.

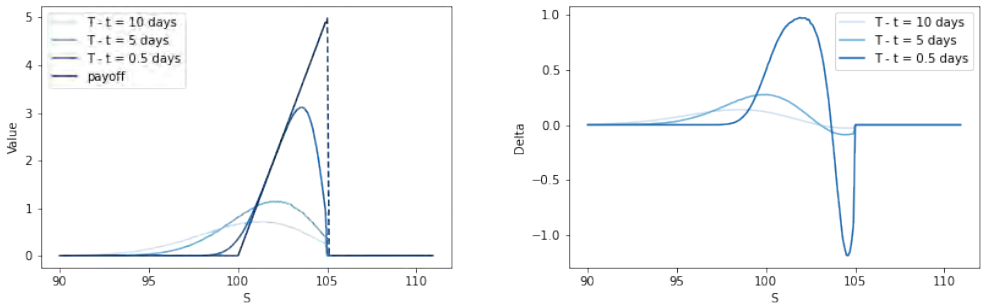


Figure 7: The value (left) and delta (right) of the up-and-out (UAO) barrier call option as a function of the underlying asset price S, different maturities $T - t$, barrier $B = 105$ and a strike price of $K = 100$.

3.3 Option Pricing

The value of an option depends on several factors, such as the underlying asset price, strike price, volatility, and remaining time to maturity. Hence, the precise value can be difficult to establish. Different pricing models have been developed aimed at incorporating these variables [69]. These methods use a model to describe the dynamics of the underlying asset price and a model to calculate the option value as a function of the assumed asset price behavior. The latter is based on risk-neutral valuation. This valuation states that under the risk-neutral measure $\mathbb{Q}$, the price of an option equals the expected value of its future payoffs, discounted by the risk-free rate [42]:

$$C_t(S_t) = e^{-r(T-t)}\mathbb{E}_Q\big[h(S_T)\big], \tag{51}$$

where $C_t(S_t)$ is the value of the call option at time t, r the risk-free rate, $T-t$ the remaining maturity and $h(S_T)$ the option payoff. Calculating an option price under the real-world (physical) probability measure $\mathbb{P}$ involves the likelihood of outcomes of an underlying asset

price and risk premia to compensate investors for bearing different types of market risk, which differ per investor. The latter is avoided in the risk-neutral valuation. The risk-neutral measure $\mathbb{Q}$ is constructed based on the model of the underlying asset price in such a way that it earns the risk-free rate in expectation. Option pricing models that are based on this valuation can be divided into analytical methods and numerical methods. These are described in more detail in the following subsections.

3.3.1 Analytic Methods

Analytical models aim to obtain a closed-form solution for the option value by evaluating the expression in Equation 51. For example, the value of a plain vanilla call option with strike K and maturity T at time t is given by [17]:

$$C_t(S_t) = e^{-r(T-t)} \int_{-\infty}^{\infty} \max\{S_T - K\} q(S_T) dS_T, \tag{52}$$

where $q(S_T)$ is the risk-neutral density for the underlying asset S_t at T. One of the most widely known option pricing models that solved this equation analytically is the Black-Scholes (BS) model, introduced by Fischer Black and Myron Scholes in 1973 [40]. This model assumes that the underlying asset behaves as a GBM. Under this model, a risk-neutral probability measure $\mathbb{Q}$ is obtained by applying the following change of probability:

$$W_t = \tilde{W}_t - \kappa t,$$

where $\tilde{W}_t$ is a Brownian Motion under the new probability measure $\mathbb{Q}$ and $\kappa = \frac{\mu - r}{\sigma}$ [24]. Under this measure, the dynamics of S_t can be written as:

$$\begin{aligned} dS_t &= \mu S_t dt + \sigma S_t dW_t \\ &= \mu S_t dt + \sigma S_t d\tilde{W}_t - \sigma \frac{\mu - r}{\sigma} S_t dt \\ &= r S_t dt + \sigma S_t d\tilde{W}_t. \end{aligned} \tag{53}$$

Applying the properties of the GBM gives $\ln(S_t) \sim N\big(\ln S_0 + (r - \frac{1}{2}\sigma^2)t, \sigma^2 t\big)$. Using this distribution and solving the integral in Equation 52 yields the following price for a European plain vanilla call option with maturity T, strike price K, constant risk-free interest rate r and volatility σ:

$$C_t(S_t) = \Phi(d_1) S_t - \Phi(d_2) K e^{-r(T-t)}, \tag{54}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution and

$$d_1 = \frac{\ln \frac{S}{K} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \tag{55}$$

$$d_2 = \frac{\ln \frac{S}{K} + (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}. \tag{56}$$

A full derivation can be found in for example [30]. Based on this option price, a closed-form solution for the Greeks can be obtained by taking the respective derivatives. For example, the delta for a plain vanilla call under the Black-Scholes model is given by:

$$\Delta = \frac{\partial C_t(S_t)}{\partial S} = N(d1). \tag{57}$$

The option value and delta of other option types can be derived in the same spirit, by using the obtained risk-neutral probability measure and by changing the option payoff in Equation 52.

3.3.2 Numerical Methods

This subsection reviews some approaches to obtain the option value and corresponding Greeks numerically.

3.3.2.1 Option Value

There are several price processes for S_t and payoff structures for which a closed-form solution for the option value is difficult to obtain. For example in the Heston model, a semi-closed form solution is derived for the plain vanilla call option and digital call option in [37] and [48], but such a formula does not exist for the barrier option. In that case, numerical methods can be applied to obtain an estimation of the option value. The most common ones for option pricing are Monte Carlo methods, finite difference methods, and tree models (such as the well-known binomial tree). The remainder of this paragraph focuses on Monte Carlo pricing.

Monte Carlo methods are a class of computational techniques that rely on repeated random sampling to estimate the possible outcomes of an uncertain event. A numerical result, such as the sample mean, can be obtained based on these outcomes. These algorithms are mainly used for numerical integration and optimization. Monte Carlo pricing is based on the former, as the determination of the option value involves an expectation (hence integral, see Equation 52). It combines the Monte Carlo method with the risk-neutral valuation. This technique generates N random paths $S_{t+\Delta t}, ..., S_T$ of the underlying asset price. This is done according to the risk-neutral asset price dynamics using a discretization step size of Δt. An estimate of the option price is obtained by calculating the discounted payoff on each path and taking the average over the paths [10]:

$$C_t(S_t) \approx \frac{1}{N} e^{-r(T-t)} \sum_{i=1}^{N} h^i(S_T). \tag{58}$$

The Monte Carlo pricing approach is outlined in Algorithm 4 for the plain vanilla call option under the Heston model.

Algorithm 4 Monte Carlo pricing for plain vanilla call under Heston model

Initialise the initial stock price S_0, instantaneous variance ν_0, discretization step Δt and the parameters κ, θ, ϵ, ρ of the Heston model

for each simulation $n = 1, .., N$ **do**

for each discretization step $t = \Delta t, .., T$ **do** ▷ Simulate a random path $S_{t+\Delta t}, ..., S_T$

$$Z_\nu, Z_S \overset{i.i.d.}{\sim} N(0,1)$$

$$Z_S = \rho Z_\nu + \sqrt{1-\rho^2} Z_S$$

$$\nu_t = \nu_t + \kappa(\theta - |\nu_t|)\Delta t + \epsilon\sqrt{|\nu_t|} Z_\nu$$

$$S_t = S_t e^{(\mu - 0.5|\nu_t|)\Delta t + \sqrt{|\nu_t|} Z_S}$$

Calculate the payoff of each path: $h^i(S_T) = \max\{S_T^i - K, 0\}$

Estimate option value: $C_t(S_t) \approx \frac{1}{N} e^{-r(T-t)} \sum_{i=1}^{N} h^i(S_T)$

The accuracy of the Monte Carlo estimator can be described by the standard error, i.e. the standard deviation of the estimator:

$$\frac{\sigma_{C_t(S_t)}}{\sqrt{N}}. \tag{59}$$

The accuracy increases in $\sqrt{N}$: the accuracy can be doubled by quadrupling the number of simulations. The latter may be computationally expensive. Fortunately, there exist meth-

ods that increase the accuracy without increasing N. These are called variance reduction techniques. One of them is the Antithetic Variable Technique [46]. Simulating the underlying asset prices $S_t^1, S_t^2, .., S_t^N$ at a discretized time step t requires generating the random variables $W_1, W_2, ..., W_N$. The Antithetic Variable Technique also takes the antitheses of these variables, resulting in the sequence $-W_1, -W_2, ..., -W_N$ and prices $S_t^{1-}, S_t^{2-}, .., S_t^{N-}$. In this way, each number is used twice, hence increasing efficiency. Compared to doubling the number of simulations, the Antithetic Variable Technique also leads to an increase in precision. Applying this technique results in a standard error being $\leq \frac{C_t(S_t)}{\sqrt{2N}}$, which is less than or in the worst case equal to the standard error if the number of simulations would have been doubled. The gain in precision is due to the fact that the covariance of simulated values is often negative, resulting in a lower variance of the estimator [46].[8]

3.3.2.2 Greeks

If a closed-form solution does not exist for the option value, a numerical differentiation is required to compute the Greeks. A natural approach to this numerical problem is to use the Monte Carlo finite difference method as approximation of the derivative [33]. In case of the delta, i.e. the first order derivative of the option value with respect to the underlying asset's price, the Monte Carlo estimator can be calculated using forward finite difference

$$\Delta \approx \frac{C_t(S_t + \epsilon) - C_t(S_t)}{\epsilon}, \tag{60}$$

for some small $\epsilon > 0$, or by central finite difference

$$\Delta \approx \frac{C_t(S_t + \epsilon) - C_t(S_t - \epsilon)}{2\epsilon}. \tag{61}$$

Here, $C_t(S_t), C_t(S_t + \epsilon)$ and $C_t(S_t + \epsilon)$ are determined using Monte Carlo pricing. With forward difference, the best possible convergence rate is $M^{-\frac{1}{4}}$. Central difference leads to a rate of $M^{-\frac{1}{3}}$, and $M^{-\frac{1}{2}}$ if the same random numbers are used for both Monte Carlo estimators. The latter is the best possible rate that can be achieved for a Monte Carlo method [31].

An important drawback of the finite difference method is that it may perform very poorly when there is a discontinuity in the payoff function, as is the case for e.g. a barrier option and digital option [26]. For these options, it has been shown that the use of stochastic calculus of variations, also known as the Malliavin calculus, resulted in a lower variance and faster convergence and hence outperformed the finite difference method [7][26]. A reason for this is the slow mean square convergence of $C_t(S_t + \epsilon)$ to $C_t(S_t)$ in the finite difference method, which is linear in ϵ for discontinuous payoff functions. Secondly, the Malliavin technique uses a smoothened payoff and circumvents in this way the computation of derivatives. However, for options with a continuous payoff function, the finite difference method outperformed the Malliavin simulation, as their mean-square convergence of $C_t(S_t + \epsilon)$ to $C_t(S_t)$ is quadratic in ϵ [7].

The Malliavin technique performs an integration by parts formula and obtains a weight function π which is independent of the payoff function. In this way, the Greeks can be expressed numerically as [88]

$$\frac{\partial C_t(S_t)}{\partial \alpha} \approx e^{-r(T-t)} \mathbb{E}_Q \Big[h(S_t)\pi \Big], \tag{62}$$

where α is the corresponding input parameter. The Malliavin weight differs per Greek and option model. For example, Fournié [26] showed that under the Black-Scholes model, the

[8] $\mathbb{V}(X_1 + X_2 + .. + X_n) = \sum_{i=1}^n \mathbb{V}(X_i) + \sum_{j=1}^n \sum_{k=1}^n Cov(X_j, X_k)$.

Malliavin weight of Delta is $\Delta_{MV} = \frac{W_T}{S_0 \sigma T}$ and that the delta can be computed by

$$\Delta_{\text{BS}} \approx e^{-r(T-t)} \mathbb{E}_Q \Big[h(S_T) \int_t^T \frac{1}{S_u \sigma T} dW_u \Big] = e^{-r(T-t)} \mathbb{E}_Q \Big[h(S_t) \frac{W_T}{S_t \sigma T} \Big]. \tag{63}$$

As described in [79], the delta under the Heston model using Malliavin calculus is given by

$$\Delta_{\text{Heston}} \approx e^{-r(T-t)} \mathbb{E}_Q \Big[h(S_t) \int_t^T \frac{1}{S_u T \sqrt{1-\rho^2} \sqrt{\nu_u}} dW_u \Big]. \tag{64}$$

3.4 Hedging

As previously described, the option value changes as the values of its input parameters change. Hedging aims to offset these changes and hence reduce the associated risk. The hedging strategy consists of the construction of a replicating portfolio that has the same cash flows (static replication) or the same Greeks (dynamic replication) as the to be hedged derivative. Every derivative can be replicated perfectly if the market is complete, i.e. if [40]

- There are no transaction costs, i.e. there is no difference between the sell and buy price and there are no additional costs.
- Long and short positions are allowed for all assets.
- Fractional holdings are allowed for all assets.
- The market is completely liquid, i.e. it is possible to buy/sell unlimited amounts on the market.

This section describes the hedging strategy under the Black-Scholes model and the Heston model.

3.4.1 Black-Scholes model

In the Black-Scholes model [9], there is just one source of risk, i.e. the dynamics of the risky asset, driving the randomness in the market. In this model, any derivative can be replicated (the market is complete) with two primary assets: a risky asset (the option's underlying asset) and a riskless asset (such as a bond or a saving account). It is assumed that the risky asset behaves as a GBM, following the SDE as described in Equation 33. The dynamics of the riskless asset are described by the ordinary differential equation (ODE):

$$dB_t = r B_t dt, \tag{65}$$

where r is the interest rate. By convention, $B_0 = 1$ such that $B_t = e^{rt}$ and the relative profit is constant (r). In this market, a trading strategy is the construction of a portfolio (X_t, Y_t) consisting of $X_t \in \mathbb{R}$ units of the option's underlying asset and $Y_t \in \mathbb{R}$ units of the riskless asset. X_t and Y_t can be positive (long position) or negative (short position) [24]. They are adapted processes, called F_t measurable. This means that at time t, they only depend on the information available before time t. At time t, the value of the portfolio is denoted by V_t:

$$V_t = X_t S_t + Y_t B_t. \tag{66}$$

The value of the portfolio over an infinitesimal time interval changes by:

$$dV_t = V_{t+1} - V_t = dX_{t+1} S_t + dY_{t+1} B_t + X_{t+1} dS_t + Y_{t+1} dB_t, \tag{67}$$

where $dX_{t+1} = X_{t+1} - X_t$ and $dY_{t+1} = Y_{t+1} - Y_t$. The portfolio is self-financing if the construction at time zero is financed by the sale of the option and if there is no further cash flowing in or out the strategy. For example, if an investor buys an additional amount of

$X_{t+1} - X_t$ shares of the risky asset at time t, costing $(X_{t+1} - X_t)S_t$, the position in the riskless asset changes by [44]

$$(Y_{t+1} - Y_t)B_t = -(X_{t+1} - X_t)S_t. \tag{68}$$

In other words:

$$dX_{t+1}S_t + dY_{t+1}B_t = 0, \tag{69}$$

such that

$$dV_t = X_{t+1}dS_t + Y_{t+1}dB_t. \tag{70}$$

To hedge an option, the payoff of the trading strategy should be exactly equal to the payoff of the option:

$$V_T = X_T S_T + Y_T B_T = h(S_T). \tag{71}$$

The portfolio that satisfies this condition is called a hedging strategy. To find the corresponding portfolio $(X_t,\ Y_t)$, it is convenient to work with martingales. As described in Section 3.1.2, S_t is not a martingale under the $\mathbb{P}$-measure. The discounted asset price is a martingale under both the $\mathbb{P}$ and $\mathbb{Q}$-measure, when discounted by the correct discount factor. As previously described, the former involves the incorporation of risk premia which differ per investor. Hence, the $\mathbb{Q}$-measure is often used and considered in the remainder of this subsection.
Define the discounted value of the risky asset and the portfolio by respectively $\tilde{S}_t = e^{-rt}S_t$ and $\tilde{V}_t = e^{-rt}V_t$. The dynamics of the discounted portfolio can be described as follows:

$$\begin{aligned} d\tilde{V}_t &= d(e^{-rt})V_t + e^{-rt}dV_t \\ &= -re^{-rt}V_t dt + e^{-rt}(Y_t dB_t + X_t dS_t) \\ &= -re^{-rt}(Y_t B_t + X_t S_t)dt + e^{-rt}(Y_t r B_t dt + X_t dS_t) \\ &= X_t(-re^{-rt}S_t dt + e^{-rt}dS_t) \\ &= X_t d\tilde{S}_t. \end{aligned} \tag{72}$$

Hence, the discounted value of the self-financing portfolio at time t can be written as

$$\tilde{V}_t = \tilde{V}_0 + \int_0^t X_u d\tilde{S}_u = V_0 + \int_0^t X_u d\tilde{S}_u. \tag{73}$$

The self-financing hedging portfolio can be determined by 1) finding a $\mathbb{Q}$ measure under which the discounted asset price $\tilde{S}_t$ is a martingale and 2) finding a process X_t that satisfies the equation [24]

$$V_T = V_0 + \int_0^T X_u d\tilde{S}_u = h(S_T). \tag{74}$$

The position in the riskless asset Y_t can then be determined using the relationship described in Equation 66.

A unique hedging portfolio exists in the Black-Scholes framework. Based on the dynamics of S_t under the $\mathbb{Q}$-measure presented in Equation 53, the dynamics of the discounted value of the underlying asset under this measure are described by the following SDE:

$$\begin{aligned} d\tilde{S}_t &= d(e^{-rt})S_t + e^{-rt}dS_t \\ &= -re^{-rt}S_t dt + e^{-rt}(rS_t dt + \sigma S_t d\tilde{W}_t) \\ &= e^{-rt}S_t \sigma d\tilde{W}_t \\ &= \sigma \tilde{S}_t d\tilde{W}_t. \end{aligned} \tag{75}$$

As this SDE does not have a drift term (there is no dt-term), $\tilde{S}_t$ is a martingale under $\mathbb{Q}$. Using this expression, the value of discounted value of $\tilde{V}_t$ in Equation 73 can be written as

$$\tilde{V}_t = V_0 + \int_0^t X_u d\tilde{S}_u = V_0 + \int_0^t X_u \sigma \tilde{S}_u d\tilde{W}_u. \tag{76}$$

As this is an integral with respect to a Brownian motion, the discounted value of a self-financing portfolio is also martingale [19]. Combining this property with the fact that $V_T = h(S_T)$ gives

$$\tilde{V}_t = \mathbb{E}_Q\Big[V_T e^{-rT}\Big|F_t\Big] = \mathbb{E}_Q\Big[h(S_T)e^{-rT}\Big|F_T\Big] \tag{77}$$

and

$$V_0 = \mathbb{E}_Q\Big[h(S_T)e^{-rT}\Big]. \tag{78}$$

This means that the value of the self-financing portfolio at initiation is equal to the conditional expectation of the discounted terminal value under $\mathbb{Q}$ with respect to the information at time t [24]. The value of the hedging portfolio can be written as a function of t and S_t in the form $H(t, S_t)$ and Itô's lemma can be used to derive the unique hedging portfolio. This lemma states that the dynamics of process X_t that satisfies the SDE [19]

$$dX_t = \mu_t dt + \sigma_t dW_t \tag{79}$$

can be approximated by:

$$df = \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial x}dx + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}dx^2 + .. \tag{80}$$

Applying this lemma to the self-financing portfolio gives

$$\begin{aligned} dV_t &= \frac{\partial H}{\partial t}(t, S_t)dt + \frac{\partial H}{\partial S}(t, S_t)dS + \frac{1}{2}\frac{\partial^2 H}{\partial S^2}(t, S_t)(dS)^2 \\ &= \frac{\partial H}{\partial t}(t, S_t)dt + \frac{\partial H}{\partial S}(t, S_t)(rS_t dt + \sigma S_t d\tilde{W}_t^{)} + \frac{1}{2}\frac{\partial^2 H}{\partial S^2}(t, S_t)\sigma^2 S_t^2 dt, \end{aligned} \tag{81}$$

and for the discounted portfolio value

$$\begin{aligned} d\tilde{V}_t &= -re^{-rt}V_t dt + e^{-rt}dV_t \\ &= e^{-rt}\Big(\frac{\partial H}{\partial t}(t, S_t) + rS_t\frac{\partial H}{\partial S}(t, S_t) + \frac{1}{2}\sigma^2\frac{\partial^2 H}{\partial S^2}(t, S_t) - rH\Big)dt + e^{-rt}\sigma S_t\frac{\partial H}{\partial S}(t, S_t)d\tilde{W}_t. \end{aligned} \tag{82}$$

As $\tilde{V}_t$ is a martingale, the drift term is equal to 0, i.e.

$$\frac{\partial H}{\partial t}(t, S_t) + rS_t\frac{\partial H}{\partial S}(t, S_t) + \frac{1}{2}\sigma^2\frac{\partial^2 H}{\partial S^2}(t, S_t) - rH = 0 \tag{83}$$

resulting in

$$d\tilde{V}_t = e^{-rt}\sigma S_t\frac{\partial H}{\partial S}d\tilde{W}_t = \sigma\tilde{S}_t\frac{\partial H}{\partial S}d\tilde{W}_t \tag{84}$$

or in the integrated form

$$\tilde{V}_t = V_0 + \int_0^t \frac{\partial H}{\partial S}\sigma\tilde{S}_u d\tilde{W}_u. \tag{85}$$

Combining Equation 76 and 85 yields

$$X_t = \frac{\partial H}{\partial S}(t, S_t). \tag{86}$$

In the case of the plain vanilla call option, $H = C$ and the number of shares of the risky asset is equal to the Δ:

$$X_t = \frac{\partial H}{\partial S}(t, S_t) = \frac{\partial C}{\partial S}(t, S_t) = N(d_1). \tag{87}$$

This means that in the Black-Scholes model, the delta at time t is the number of shares required to replicate a vanilla call option at that time.

3.4.2 Heston Model

In the Heston model, there are two sources of risk driving the randomness in the market due to stochastic volatility. As volatility is not a tradeable asset, the market described by this model is not complete: an option cannot be replicated with the assets available in this market [3]. The market can become complete by adding a volatility-dependent asset, such as a European option. The self-financing hedging portfolio corresponding to such a market can be derived in the same spirit of the replication approach applied in the Black-Scholes model. One of such derivations is given in [15]. This derivation is more complicated, as it contains an additional asset, and is not included in this thesis.

4 Prior Works on Reinforcement Learning for Option Hedging

The idea to optimize financial objective functions via reinforcement learning (RL) was already proposed in 1998 by Moody et al. [57]. The first attempts to apply RL to hedge options modelled the problem with a discrete state space and action space. Among these, Watts introduced a basis risk hedging strategy using R in 2015L. It used the SARSA algorithm[9] to find an optimal strategy for hedging a non-traded asset. The hedging objective was to maximize the utility of the terminal payoff of the hedging strategy, called the cash flow formulation. This approach used an exponential objective (utility) function with a risk tolerance value to capture the variance of the hedging costs
Q-learning was applied to the hedging problem in 2017 by Halperin [36]. This research assumed that the complete market assumptions held, neglected transaction costs, and assumed that the agent had an exponential utility. The accounting approach was applied, which evaluates the hedging strategy at each point in time.

A continuous state space was addressed by Ritter and Kolm in 2018 [65]. They considered a discrete action space, quadratic transaction costs, accounting formulation and used the SARSA algorithm to train the agent. They assumed that the agent had a quadratic utility: their objective function was equal to the mean hedging cost plus a constant times the variance of the hedging cost[10].

The allowance for a continuous action space was proposed by Buehler et al. in 2019 [13]. They used the deep Q-learning algorithm. Contrary to the previous approaches, which considered the Black-Scholes (BS) model and hence assumed that the stock price behaved as a Geometric Brownian Motion (GBM), they assumed that the market was driven by the Heston model. In addition, trading was allowed in both stock and a variance swap[11] and the hedging strategy was only valued at maturity using the cash flow approach. Several risk measures were considered and embedded in a neural network environment. Their work was extended in 2019 [12]. Here, they presented an approach to calculating the price and optimal hedging strategies for portfolios of derivatives. They tried different neural network architectures and considered both a market driven by the Heston model and the Black-Scholes model in the presence of trading costs and liquidity constraints. The Conditional Value at Risk (CVaR) coherent risk measure[12] was used as the objective function.
In the same year, Cao et al. [14] proposed another objective function. To better estimate the variance of the hedging costs, they considered two Q-functions, one for the first moment (mean) of the costs and another for the second moment (variance). Their research assumed that the volatility of the underlying asset price was stochastic and the transaction costs proportional. They applied the Deep Deterministic Policy Gradient (DDPG) algorithm, formulated two ways to define the hedger's problem, and considered both the accounting formulation and cash flow formulation, as well as a hybrid form.
A different actor-critic algorithm was applied by Vittori et al. in 2020 [80]: the Trust Region Volatility Optimization (TRVO)[13] algorithm. Using this algorithm, the variance of the hedging costs was automatically bounded, as the algorithm's constraint guaranteed improvement in a risk-averse manner. They considered quadratic transaction costs and the

[9] In contrast to Q-learning[82], which only selects the action a_t and estimates the largest value of $Q(s_{t+1}, a_{t+1})$ to update $Q(s_t, a_t)$, SARSA selects two actions a_t and a_{t+1} and uses $Q(s_{t+1}, a_{t+1})$ to update $Q(s_t, a_t)$.

[10] This objective is a mean-variance optimization problem with a risk aversion parameter.

[11] An over-the-counter derivative that exchanges payments related to future realized price variance against fixed rates.

[12] A risk measure, also called the expected shortfall, that quantifies the average value of a loss in an investment if the loss exceeds a given confidence level.

[13] A risk-averse variant of the TRPO actor-critic algorithm that is similar to DDPG, but aims to maximize a specific objective function subject to a constraint on the difference between the old and new policies.

Black-Scholes model, and applied the accounting formulation.

In summary, the application of RL to option hedging is quite novel. Although the implementation of the hedging problem is not identical for the aforementioned papers, these studies had a similar goal, setting, and overall conclusions. Among these studies, a transition is visible from the modelling of the hedging problem using discrete state and action space towards continuous ones (in line with the timeline of RL). In addition, the RL algorithms have become more sophisticated. Starting with constant volatility in the Black-Scholes model, more papers considered stochastic volatility. In addition, the transaction costs were initially excluded and later modelled in a proportional or quadratic way. All papers included the variance, i.e. risk, of the hedging costs using one of the following different measures: CVAR, an exponential, or a quadratic utility function. Most studies considered one single vanilla call option, except [12] which extended the range of hedging instruments available in the market and aimed at hedging a portfolio of options. All papers showed that RL could effectively be used for the defined option hedging problem.

5 Methodology

In this research, the goal of the reinforcement learning (RL) agent is to replicate an option with certain characteristics and a stock as the underlying asset. The options chosen and assumptions made regarding these options are described in Section 5.1. To recast the option hedging problem as an RL problem, the framework is embedded in a Markov Decision Process (MDP). Recall the representation of the agent-environment interaction in the RL setting: at each time step, the agent receives a state from the environment s_t, takes an action a_t that is applied to the environment, and receives a reward r_t plus a new state s_{t+1} from the environment. In the option hedging problem, the action is the position to take in the underlying asset. To be able to take this action, the agent must know the current stock price, the time left until the option expires, and its previous position in the stock. These variables are captured in the state. The reward represents how well the hedging portfolio replicates the option and includes possible transaction costs. The exact determination of the environment is described in Section 5.2. Section 5.3 describes the RL algorithm chosen to learn the option hedging problem. Finally, the approach to evaluating the performance of the RL agent is described in Section 5.4. The implementation of these approaches is presented in Chapter 6.

5.1 Options

As options, the plain vanilla, digital and barrier call option are chosen to be trained on. The latter two are chosen due to their discontinuity in the payoff, which makes the hedging more complicated compared to the former, see Section 3.2.2 and 3.2.3.
As in [65], the options have a maturity of 10 trading days ($T = 10$) with 5 periods per day ($D = 5$). The latter means that the state transitions are defined in discrete steps, reflecting realistic trading. As a result, an episode consists of 51 ($T \times D\ +1$ at expiration) time steps. The episode ends when the option expires; the environment is then reset.
The options are initialised at-the-money (ATM), as these are the most interesting. ATM plain vanilla and digital call options can get in-the-money (ITM) or out-of-the-money (OTM) in a short period of time if the stock price moves up or down. For an ATM barrier call option, the trigger level has not been hit yet, but can be reached during the option's lifetime.
For simplicity, it is assumed that the to be hedged options cannot be traded, i.e. are held until maturity. As the contract size of most stock option contracts is standardized at 100 shares, it is assumed that the agent should hedge one single call option that represents an option to buy 100 shares of the underlying stock.
As in earlier studies, except [13] and [12], the agent must hedge the option by only taking positions in the underlying asset and a riskless asset. Hence, the agent cannot hedge any volatility exposure and the hedging problem comes down to delta hedging.
To calculate the rewards and to compare the agent's performance with that of a delta hedging strategy, the value and the delta of the to be hedged option are calculated at each time step. As the stock price is simulated according to the Heston model, as will be motivated in Section 5.2.3, a closed-form solution is not available for all the option types. Hence, the option value and delta are determined using Monte Carlo pricing.

5.2 Environment

To define the environment and recast the option hedging problem as an RL problem, the problem is embedded in a Markov Decision Process (MDP), defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ as explained in Section 2.1. The following subsections cover the implementation of each of these variables.

5.2.1 State $\mathcal{S}$

As described in Section 2.1, the state should contain all information that is important for the agent to make a decision. As this research uses an RL algorithm that uses neural networks, the state is scaled since feature scaling improves the convergence of such algorithm [23]. The following variables are chosen as state components and scaled in the following way:

1. The stock price S_t. This variable is required to determine the expected payoff of the option, where the option value and hence appropriate action depend on. The stock price S_t depends on the initial stock price S_0. To adjust for this dependency, the logarithmic return with respect to the initial stock price is calculated: $\log(\frac{S_t}{S_0})$. This is a commonly used normalization technique applied in finance and represents the percentage change in price [39]. Furthermore, the change in the stock price depends on the time interval between two trading points. It is known that the variance of price changes increases as time increases. If the stock price is modelled as a GBM like in the Black-Scholes model, see Section 3.1.2, the logarithmic returns are normally distributed and their variance increase with the square root of time. This is also known as the *square root of time rule*. In that case, the scaled state component $\log(\frac{S_t}{S_0})/(\sigma\sqrt{\Delta t})$ adjusts for the time interval length. As the stock prices are simulated according to the Heston model, $\log(\frac{S_t}{S_0})/\sqrt{\nu_t \Delta t}$ yields an approximate scaling, as the distribution of logarithmic returns is normal only for small returns under this model, see Section 3.1.3 and [21]. [14]

2. The time to maturity $T - t$. Since the hedging problem has a finite horizon, the remaining time until the option expires is required to be able to take an appropriate action. This value impacts the option value, hence the appropriate hedging portfolio, as well as the number of subsequent steps for which the agent should maximize the expected reward. The time to maturity is scaled between 0 and 1, where the value 1 is at initiation ($t = 0$) and 0 is at expiration ($t = T$).

3. The previous position in the underlying asset a_{t-1}. This variable is required to determine the amount of the underlying asset bought or sold at time t. This amount is defined as the difference between a_t and a_{t-1} and important to take into account when taking an action in the presence of transaction costs, as this yields a negative reward. The previous position is divided by the contract size such that the value is represented per one underlying share and between the option's minimum and maximum action, see section 5.2.2.

In other words, the state can be denoted by $s_t = (\log(\frac{S_t}{S_0})/\sqrt{\nu_t \Delta t}, a_{t-1}, T - t)$. In this way, the state transitions only depend on the current state and not on any previous actions or states, hence satisfying the Markov property.
Note that the state does not need to contain the option delta or option parameters such as the strike, as the agent should infer these from the state variables and the reward provided by the environment. When training on different option types simultaneously, it is necessary to add an option-specific feature. The agent would otherwise not be able to infer what option type it should hedge. In this setting, the intrinsic value is added as a state variable. This value represents the value of the option if it would be exercised now. The intrinsic value is scaled in the same spirit of the stock price scaling: the logarithmic return of this value is calculated with respect to the strike price and adjusted for the fact that the logarithm of zero (if the intrinsic value is zero) is undefined. The square root of time rule is applied to adjust for the time interval length. This gives the scaled intrinsic value: $\log(\frac{K+ \text{ intrinsic payoff}}{K})/\sqrt{\nu_t \Delta t}$.

[14] Note that a min-max scaling could have been applied as well, but this would require the computation of a time-dependent mean and variance prior to the training or the computation of a running average and variance throughout an episode and through training. Next to the fact that this is more computationally expensive, this scaling would be less generic, as it is parameter-specific and might not hold when e.g. the initial stock price changes.

5.2.2 Action $\mathcal{A}$

The action represents the position to take in the underlying asset. This action is represented per one underlying share. The action space is continuous: the underlying can be bought and sold in any fractions ($a_t \in [a_{\min}, a_{\max}]$). To define the action space, the minimum and maximum action are determined for each of the option types. In the Black-Scholes model, the optimal action in a given state should be equal to the option's delta, see Section 3.4.1. In discrete time, the presence of transaction costs and/or under the Heston model, a pure delta hedge strategy might not be optimal. However, a similar tracking is to be expected in this research since the hedging task comes down to delta hedging. Hence, the action boundaries are determined based on the delta distribution of the options. As discussed in Section 3.2, the delta of a plain vanilla call is always between 0 and 1, but might take more extreme values in the case of a digital or barrier call option. However, a larger action space requires longer exploration and hardens the training. Also from a risk perspective, it might be better to bound the minimum and maximum action to avoid large positions and hence to limit the risk. Therefore, extreme possible values that rarely occur are discarded for the digital and barrier call option. This is done by extracting the deltas at each time step for 10,000 episodes and taking the 1st or 99th percentile as respectively the minimum or maximum action value.

5.2.3 Transition Function $\mathcal{T}$

The full state vector is part of the transition from state s_t to state s_{t+1} when taking action a_t. In this research, the remaining time to maturity $T - t$ and previous position a_{t-1} are deterministic. The time between two trading points (Δt) is chosen to be fixed and the action taken at time t (a_t) flows directly in the new state that the agent will observe as a_{t-1}. The stochastic transition factor is the newly observed (scaled) stock price S_t. The stock price transition can be modelled implicitly by providing samples from the transition distributions. For this, real historical stock price data or simulated data can be used. As the RL agent needs to see a large amount of data to learn, the former approach is less suitable. It requires either market data up to several years ago or the use of overlapping windows to divide the time series in trajectories. The former may not be representative of the current market and introduce high variability, and the latter may introduce correlation and the chance of overfitting [15]. Hence, the RL agent is trained on simulated data in this research. This approach can produce an infinite amount of data and avoids overfitting due to its stochastic nature. To simulate the stock prices, it is assumed that the market follows the Heston dynamics; the stock price is simulated according to Equation 40, 41 and 42.

5.2.4 Reward Function $\mathcal{R}$

With continuous trading and in the absence of market frictions such as transaction costs, it is possible to set up a dynamic hedge portfolio that replicates the option perfectly at each point in time. In that case, there is no risk involved as the portfolio value Π, i.e. the option value minus the offsetting hedge value, is always zero and thus has no variance. The more realistic setting of discrete trading introduces a hedging error, as portfolio rebalancing takes place at discrete times while asset prices move in continuous time [60]. Most investors proved to have a risk aversion, meaning that they want to reduce the amount of risk they take for a level of return [18]. Hence, the hedging objective should not solely aim at finding the strategy that maximizes the expected rewards, but it should also take the risks arising from the hedging strategy into account. The investor's attitude towards risk is subjective and can be hardly justified by economic reasoning.[16] The degree of risk aversion can be

[15] When a model tries to predict a trend in a training data set that is too noisy, the model corresponds too closely to the training data set but has a poor fit with new data set.

[16] The regulatory authority cannot change the investors' risk aversion degree, but it can control it. To protect financial institutions, their investors, clients, and the economy as a whole, regulators control the

captured in a utility function. Here, the investor seeks to maximize the expected utility of the final wealth $\mathbb{E}[u(w_T)]$, where w_T is the final wealth including the trading costs and u the investor's utility function. Ritter argued that this optimization problem can be hard to solve for some nonlinear utility functions, but that the optimal policy π[17] is also optimal for the simpler mean-variance optimization problem [64]:

$$\max_{\pi} \mathbb{E}[w_T] - \frac{\kappa}{2}\mathbb{V}[w_T]. \tag{88}$$

Here, κ denotes the risk-aversion parameter[18]. Cao et al. [14] defined two alternative formulations to represent the final wealth: the accounting formulation and the cash flow formulation [14]. Both approaches are implemented as described in the following two paragraphs.

5.2.4.1 Accounting Formulation

In the accounting formulation, the hedged position is valued at each time step. Here, the final wealth is decomposed as a sum of period-by-period hedging costs which are provided to the RL agent at each time step in the form of a reward. This gradual feedback incorporates domain knowledge (i.e. a pricing model) and eases the learning of the hedging problem. Hence, this formulation can be seen as reward shaping. This approach reflects real trading; calculating the option price and delta is often standard and already implemented in the trading environment. An investor would likely rebalance its portfolio on e.g. a daily basis and evaluate his performance frequently, not only when the option expires.

According to the accounting formulation [14], the profit or loss (PnL) of hedging from time $t-1$ to t is defined as the change in the portfolio value $\Pi_t - \Pi_{t-1}$ plus the trading costs c_t resulting from the portfolio rebalancing:

$$\mathrm{PnL}_t = \Pi_t - \Pi_{t-1} - c_t. \tag{89}$$

Denoting the option value by $C_t(S_t)$ and the value of the hedging strategy by H_t, Π_t can be written as:

$$\Pi_t = C_t(S_t) - H_t. \tag{90}$$

It is assumed that the hedging strategy is self-financing, see Equation 73, such that the hedge value at time t is equal to:

$$\begin{aligned} &H_t = \textstyle\sum_{k=0}^{t-1} a_k(S_{k+1} - S_k) = H_{t-1} + a_{t-1}(S_t - S_{t-1}), 0 < t \leq T, \\ &H_0 = C_0(S_0) \end{aligned} \tag{91}$$

where a_t is the position in the stock at time t. The transaction costs are defined in the same way as proposed in [65]:

$$\begin{aligned} &c_t = \text{Ticksize} \times \left(|a_t - a_{t-1}| + 0.01(a_t - a_{t-1})^2\right),\ 0 < t < T, \\ &c_0 = \text{Ticksize} \times \left(|a_0| + 0.01(a_0)^2\right). \end{aligned} \tag{92}$$

Here, $a_t - a_{t-1}$ represents the amount of the underlying that is bought or sold at time t and the tick size is the smallest amount that the price of a financial instrument can

capital requirements, i.e. the capital that financial institutions need to maintain solvent. The required capital is expressed as a percentage of the institutions' risk-weighted assets, which makes it unattractive to take very high risk. Investors with a lower risk aversion could still be willing to take on risk, but the cost of the associated capital is such that it is not worth it.

[17] The optimal sequence of positions taken in the underlying asset.

[18] The risk aversion parameter is positive for risk-averse investors (additional risk reduces the utility), 0 for risk-neutral investors (a change in risk does not affect the utility) and negative for risk-seeking investors (additional risk increases the utility).

move. The cost function is convex: the transaction costs do not grow proportionally to the number of traded shares. By expressing the final wealth as a sum of individual profit or losses resulting from hedging and assuming that these are independent, the mean-variance optimization problem can be rewritten as a sum:

$$\max_{\pi} \sum_{t=0}^{T} \big(\mathbb{E}[\mathrm{PnL}_t] - \frac{\kappa}{2}\mathbb{V}[\mathrm{PnL}_t]\big). \tag{93}$$

To transition the mean-variance problem into a reinforcement learning problem, the following approximate reward function proposed by [65] is used:

$$R_t \approx \mathrm{PnL}_t - \frac{\kappa}{2}\big(\mathrm{PnL}_t\big)^2. \tag{94}$$

Maximizing the sum of the expected rewards using this function approximates the mean-variance problem in Equation 88 and is the same as maximizing the discounted expected return in the general RL setting presented in Equation 4 with $\gamma = 1$. This reward function has a quadratic shape and the maximum is obtained at $\mathrm{PnL}_t = \frac{1}{\kappa}$, since $\frac{\partial}{\partial \mathrm{PnL}_t} R_t = 1 - \kappa \mathrm{PnL}_t$. As the hedge portfolio should replicate the option as good as possible, a portfolio change of zero is optimal. Hence, the maximum reward should be obtained at this point. To correct for this, the PnLs are shifted in this research:

$$\mathrm{PnL}_t = \mathrm{PnL}_t + \frac{1}{\kappa} \tag{95}$$

such that $\frac{\partial}{\partial \mathrm{PnL}_t} R_t = -\kappa \mathrm{PnL}_t$ and the maximum is attained at $\mathrm{PnL}_t = 0$. This adjustment shifts the reward function, but does not impact its quadratic shape nor the hedging objective. Furthermore, the absolute value and the standard deviation of the portfolio, and hence of the PnLs which flow into the rewards, increase over time (within an episode). With a discount factor close to 1, large negative rewards are accumulated, making the learning difficult. To ease the learning, the PnLs are scaled on a per time step basis. As the PnL is defined on $t-1$ to t, its standard deviation is approximated by the standard deviation of the stock price over this time interval.[19] Under the Heston model, S_t conditioned on S_{t-1} and ν_{t-1} behaves as a GBM, see Section 3.1.3. Using the variance of this distribution yields the scaled value $\mathrm{PnL}_t \big/ \sqrt{S_{t-1}^2 e^{2r\Delta t}(e^{\nu_{t-1}\Delta t} - 1)}$ with r the risk-free rate and Δt the time interval between two trading points.

5.2.4.2 Cash Flow Formulation

In the cash flow formulation, only the cash inflows and outflows resulting from trading in the underlying stock and a potential final cash flow if the option is exercised at maturity are used. The final wealth w_T can thus be defined as the option value minus the hedge value at maturity plus the total cost of trading in the underlying stock up to maturity:

$$w_T = C_T(S_T) - H_T - \sum_{t=0}^{T-1} c_t, \tag{96}$$

where $C_T(S_T)$ is the payoff of the option at maturity, H_T the value of the hedge portfolio at maturity and c_t the transaction costs at time t as defined in respectively Equation 90 and

[19] Normalizing based on the entire history of the PnLs would yield an on an average good result, but does not take the time dependency into account. Hence, an approximate, time-varying scaling based on the distribution of the portfolio is applied. The variance of the portfolio Π consists of the variance of the option, the variance of the hedge, and the covariance between the two. The latter is difficult to determine analytically but can be computed empirically. This involves the computation of a running average, which is computationally expensive. Instead, an approximate scaling is applied based on the variance of the underlying stock, on which both the variance of the option and the hedge value depend. This scaling takes the time-dependency into account.

92. In this way, the mean-variance problem as defined in Equation 88 can directly be used as a reward function. The final wealth is again shifted by $\frac{1}{\kappa}$ to ensure that the maximum is attained at $w_T = 0$. The wealth is scaled in a similar way as in the accounting formulation. The cash flow formulation does not require any domain knowledge and may avoid biases introduced by such. However, the resulting sparse rewards may harden the learning of the hedging problem.

5.2.5 Discount factor γ

To train the RL agent, a discount factor of 0 was initially used. As the agent's goal is to maximize the sum of expected future rewards, the discount factor was then increased to take future rewards into account. A higher discount factor better reflects the real-world setting, but also leads to the propagation of errors and instabilities during training. As proposed in [27], the discount factor was increased through learning as an attempt to reduce these instabilities. Starting with an initial discount factor of $\gamma_0 = 0.9$, the increased discount factor is calculated using the inverse time decay schedule according to:

$$\gamma = 1 - \frac{0.1}{1 + \text{decay rate} \times \frac{\text{step}}{\text{decay step}}}, \tag{97}$$

where the step is the current training step (iteration) and the decay step is the number of training steps after which a full decay rate is applied. As this procedure requires longer training, a trade-off was considered between the performance of the policy, and the stability and speed of the learning process. Here, the goal was to find a discount factor that is close to one, but which does not hurt the hedging performance too much. To achieve this, a validation step was performed every 200 iterations. This step compared the performance of the policy under the *current* γ with the policy under the *previous best* γ. The average return and standard deviation of both policies were calculated based on 350 simulated episodes.[20] The *best* policy was then chosen based on the following approximated one-sided t-test: if the agent's average performance was worse than the previous best average performance with 95% confidence, i.e. if

$$z_{\text{score}} = \frac{\mu_{\text{current return}} - \mu_{\text{previous best return}}}{\sigma_{\text{current return}}} \leq -1.96, \tag{98}$$

the best agent was not updated.

5.3 Reinforcement Learning Algorithm

The critical RL aspect left is the algorithm to train the agent. An actor-critic algorithm is appropriate, as it allows for the continuous state space involved in this option hedging problem. Within this class, the Deep Deterministic Policy Gradient (DDPG) algorithm is chosen, because it can also handle the hedging problem's continuous action space. As described in Section 2.2.3.1, this algorithm uses neural networks as function approximators. The model parameters of this algorithm are updated based on mini-batches which are sampled from the replay buffer and by applying gradient descent. Several gradient descent variants exist. Initializing the algorithm requires the choice of this optimization technique algorithm, the design of the neural networks, and the determination of several hyperparameters, such as the learning rate and exploration rate. These settings are described in Section 6.3.

5.4 Evaluation

This section describes the approach to evaluating the performance of the RL agent, i.e. to measure the agent by how well it does at the hedging task. As the option hedging problem

[20] Taking the standard error into account, this number seemed to be an appropriate representation of the population and at the same time not too computationally expensive.

comes down to delta hedging in this research, the performance of the RL agent is compared with a pure delta hedge strategy. This strategy takes the delta of the option as the position in the underlying stock at each point in time to replicate the option. This strategy is chosen as a benchmark, as it is more realistic and suitable compared to a random acting reinforcement learning agent. After training the agent, 10,000 stock price paths are simulated and the profit and losses (PnLs) of both strategies are tracked on these paths. These paths are almost surely out-of-sample due to the random seeds and associated stochasticity used in the training. The distribution of the PnLs over time and the terminal portfolio value, i.e. accumulated PnL within an episode, of the strategies are compared. The two-sample Kolmogorov-Smirnov test is used to test whether the distributions come from the same distribution. This test is nonparametric, i.e. does not assume that the data are sampled from a prescribed distribution, and compares the cumulative distributions of two data sets [8]. To test if the sample means and the average standard deviation within an episode of both strategies are significantly different or if one is significantly lower/higher, a two-sample t-test is performed. This test assumes normality for small samples but is also valid for large samples from non-normal distributions [8]. Since the PnL values of both strategies are tracked on the same stock price paths, the samples are dependent and a paired two-sample t-test is performed. The tests are conducted for each of the option types separately.

For each option type, a separate RL agent is trained in an environment with and without transaction costs. The hedging performance of the RL agent compared to that of a delta hedge is analysed in these two environments and also compared between the two different settings. To test the generalizability of the RL agent, its performance is evaluated on similar options with a modified strike price, initial stock price, initial instantaneous variance, or initial barrier level compared to the option on which the agent had been trained. Here, the RL agent is not *retrained*. As the option price at inception differs for an option with a modified characteristic, so does its PnLs. To adjust for this, the results are analysed based on the ratio of the RL agent's performance to delta hedge performance, rather than comparing the absolute differences. Furthermore, to determine whether a single RL agent can be used to hedge different option types, a separate RL agent is trained on different option types simultaneously by adding the scaled intrinsic value as state component as described in Section 5.2.1. As the inclusion of different option types makes the hedging problem more complex, it is decided to add the digital call option and the UAI barrier call option, instead of all the barrier option types. As the UAI behaves as a vanilla call option once the barrier has been hit, the agent sees the vanilla call feature more often. To adjust for this, the training environments are divided over the options using a ratio of 1:2:2 for the digital call, plain vanilla, and UAI barrier call option respectively. This means that the RL agent sees the digital call option twice as much compared to the other two option types. Finally, the possible impact of design choices related to the neural network on the performance of the RL agent is investigated. For this, the RL agent is trained and evaluated on a grid of different values of the following hyperparameters/functions: the random seed when initializing the neural networks, the learning rate for the actor/critic network, the discount factor, and the optimization algorithm to update the model parameters.

Next to comparing the performance of the RL agent with that of delta hedging, the value function is plotted for a simulated episode to visualise the performance of the agent. For this, the state is observed at each time step within the episode. Given this state, the critic's estimate of the value function is extracted for a range of actions, as well as the actor's selected action. The value function is then visually shown using a contour plot, which shows for each time step how the state-action value changes as a function of the action given the observed state.

6 Experimental Setup

As previously described, the option hedging problem is embedded in a Markov Decision Process (MDP) to recast it as a reinforcement learning (RL) problem. To implement the corresponding financial environment and the Deep Deterministic Policy Gradient (DDPG) algorithm to train the agent, the library *TensorFlow Agents* is used in *Python*. This is a library for reinforcement learning. Implementing an environment in Python requires the construction of a *step(action)* and a *reset()* method. The former applies an action to the environment and returns the state, reward, and discount factor for the next time step. The latter starts a new episode and provides an initial time step. These methods are implemented according to the MDP described in Section 5.2. The corresponding pseudocode is presented in Algorithm 5. The parameters chosen to implement the environment are described in Section 6.2. To return the state and reward for the next step, the class *Option* is implemented. This class implements the Heston model to simulate the stock price (paths). It also contains a subclass for each option type which includes the parameters specific for that option type, as well as the method for calculating the option's delta and fair value. The subclass for the barrier option is divided into four sub-subclasses for the up-and-in (UAI), up-and-out (UAO), down-and-in (DAI), and down-and-out (DAO) barrier option type. The chosen option parameters and the approach to calculating the option delta and value are described in Section 6.1. The initialisation of the DDPG agent and the neural networks requires the determination of certain hyperparameters. These are presented in Section 6.3.

Algorithm 5 Reset and Step function

$C_t(S_t) = \frac{1}{10{,}000} e^{-r(T-t)} \sum_{i=1}^{10{,}000} h^i(S_T)$ ▷ Option value
$\Delta = \frac{\partial C_t(S_t)}{\partial S}$ ▷ Estimate using finite difference or Malliavin
if Reset **then**
  $a_t = \Delta$
  $c_t = \text{Ticksize} \times (|a_t| + 0.01(a_t)^2)$ ▷ Transaction costs
  $H_t = C_t(S_t)$ ▷ Hedge value
  $\Pi_t = C_t(S_t) - H_t$ ▷ Portfolio value
  $\text{PnL}_t = -c_t$
else
  $c_t = \text{Ticksize} \times (|a_t| + 0.01(a_t - a_{t-1})^2)$ ▷ Transaction costs
  $H_t = H_{t-1} + a_t(S_t - S_{t-1})$ ▷ Hedge value
  $\Pi_t = C_t(S_t) - H_t$ ▷ Portfolio value
  $\text{PnL}_t = \Pi_t - \Pi_{t-1} - c_t$
  $r_t = \text{PnL}_t - \frac{\kappa}{2}(\text{PnL}_t)^2$ ▷ Reward
$\nu_{t+1} = \nu_t + \kappa(\theta - |\nu_t|)\Delta t + \epsilon\sqrt{|\nu_t|} dW^{\nu}_{t+1}$ ▷ Instantaneous variance
$dW^{S}_{t+1} = \rho\, dW^{\nu}_{t+1} + \sqrt{1-\rho^2}\, dW_{t+1}$
$S_{t+1} = S_t e^{(\mu - 0.5|\nu_t|)dt + \sqrt{|\nu_t|} dW^{S}_{t+1}}$ ▷ Stock price
$s_{t+1} = \left(\log\left(\frac{S_{t+\Delta t}}{S_0}\right)/\sqrt{\nu_t \Delta t}, a_t, T - t - \Delta t\right)$ ▷ Next state
if Reset **then**
  return s_{t+1}
else
  return s_{t+1}, r_t

6.1 Options

This section presents the parameters chosen to define the options and the approach to calculating their value and delta.

6.1.1 Option Parameters

As mentioned before, the options are initialised at-the-money (ATM) and have a maturity of 10 trading days. Their minimum and maximum action are determined as described in

Section 5.2.2. Apart from these parameters, the fixed payoff of the digital call option is set at $Q = 1$. Furthermore, the barrier option requires the determination of its trigger level which is critical for the learning of the agent. For the knock-in barrier options, a barrier that is too low (UAI)/high (DAI) would reduce the hedging problem to a plain vanilla call option hedging task. At the same time, a value that is too high (UAI)/low (DAI) means that the option is rarely kicked in and hence expires worthless most of the time. As a result, the agent would learn to not take any position at all. The opposite holds for the knock-out barrier option. The option would behave like a plain vanilla call option if the barrier is too high (UAO)/too low (DAO). If the value is too low (UAO)/high (DAO), the option would often be kicked out and hence expiring worthless most of the time. For this reason, barrier values that are knocked in (knock-in option being activated) or *not* knocked out (knock-out option being alive) in approximately 62% of the times seemed to be reasonable. The corresponding barrier values are determined by simulating 10,000 episodes and tracking the fraction of time that the barrier is hit for a range of values. The resulting barriers as well as the other options' input parameters and the initial value are presented in Table 1.[21]

Table 1: The hyperparameters used for the option types.

Option	Min action	Max action	K	T (days)	Additional feature	Initial value
Vanilla	0	1	100	10	-	1.58
Digital	0	1	100	10	Fixed payoff = 1	0.49[22]
UAI	0	1	100	10	Barrier = 101.5	1.55
UAO	-0.25	0.5	100	10	Barrier = 103.2	0.25
DAI	0	1	100	10	Barrier = 98.3	0.22
DAO	0	1	100	10	Barrier = 96.7	1.55

Furthermore, the portfolio value is 0 for the knock-out barrier options once their barrier is hit. At the same time, the hedge value ($H_t = H_{t-1} + a_{t-1}(S_t - S_{t-1}),\ 0 < t \leq T$) is likely to be positive due to the self-financing property. To match the option value after the option is kicked out, the agent will take negative positions in the underlying stock if the stock price increases or positive positions if the stock price decreases. This is not desirable; one should not invest in the underlying stock anymore when the barrier is hit, as the option is then worthless. To avoid the agent taking any position in the stock after hitting the barrier, the episode is terminated at this moment.

6.1.2 Option Pricing

The option value and delta are determined using Monte Carlo simulations. For this, 10,000 stock price paths (from the current time t until maturity T) are simulated at each time step according to Equation 40, 41 and 42 and by applying the Antithetic Variable Technique as described in Section 3.3.2. As the training requires many simulated episodes, a larger discretization step is chosen than Δt to reduce the computation time when calculating the fair value. To minimize the noise due to simulation, a relative discretization is applied: for each time step, the remaining time to maturity is divided into 10 equally spaced intervals. As the estimated option price and delta of the barrier call option depend on respectively the minimum (DAI and DAO) or maximum (UAI and UAO) stock price, these values are monitored when simulating the stock price paths. The option price is then estimated by

[21] It can be observed that the minimum and maximum action is different for the UAO barrier call option. This is because the delta of this option type can take negative and positive values, as explained in Section 3.2.3. Although this is theoretically also the case for the DAI barrier option, the extracted delta values of this option were between 0 and 1 in this research setting. It is also visible that the initial values of the knock-in and knock-out barrier option do not sum up to the price of the vanilla call option as stated by the in-out parity. This is because the options have different barrier levels.

[22] This value reflects the fact that the digital's payoff is equal to the delta of the plain vanilla call option (see Section 3.2.2), which is close to 0.5 ATM.

calculating the average discounted payoff of the simulated stock price paths based on the option's payoff structure as described in Section 3.2. This is summarized in Table 2.

Based on the simulated stock price paths, the option's delta is approximated using central finite difference method or Malliavin calculus. As described in Section 3.3.2, the former approach outperforms the latter when the payoff function is continuous and the reverse holds when there is a discontinuity in the payoff function. This is supported by Figure 8, which shows the convergence of the delta of a plain vanilla and a digital call option as a function of the sample size n using the two different approaches. Hence, the central finite difference method is applied to estimate the plain vanilla call option's delta and Malliavin calculus is used for the digital and knock-out barrier call option. As the knock-in barrier call options behave like a plain vanilla call option once the barrier is hit, their delta is determined using Malliavin calculus when the barrier has not been hit yet and using central finite difference after it has been hit. This is summarized in Table 2. The central difference approximation to the delta is calculated according to Equation 61. Here, ϵ is set to 0.01. The stock price paths of $S_t + 0.01$ and $S_t - 0.01$ are calculated using the same simulated random standard normal variables as used for S_t. The delta approximation using Malliavin calculus is calculated according to Equation 64.

Table 2: The formula and approach used to estimate the option value and delta per call option type.[23]

Option	**Option value**	**Delta**
Vanilla	$e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}$	Central finite difference
Digital	$e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\mathbb{1}_{S_T^i \geq K}$	Malliavin calculus
UAI	Inactivated: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}\mathbb{1}_{S_{\max}^i \geq B}$	Malliavin calculus
	Activated: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}$	Central finite difference
UAO	Alive: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}\mathbb{1}_{S_{\max}^i < B}$	Malliavin calculus
	Kicked out: 0	0
DAI	Inactivated: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}\mathbb{1}_{S_{\min}^i \leq B}$	Malliavin calculus
	Activated: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}$	Central finite difference
DAO	Alive: $e^{-r(T-t)}\frac{1}{M}\sum_{i=1}^{M}\max\{S_T^i - K, 0\}\mathbb{1}_{S_{\min}^i > B}$	Malliavin calculus
	Kicked out: 0	0

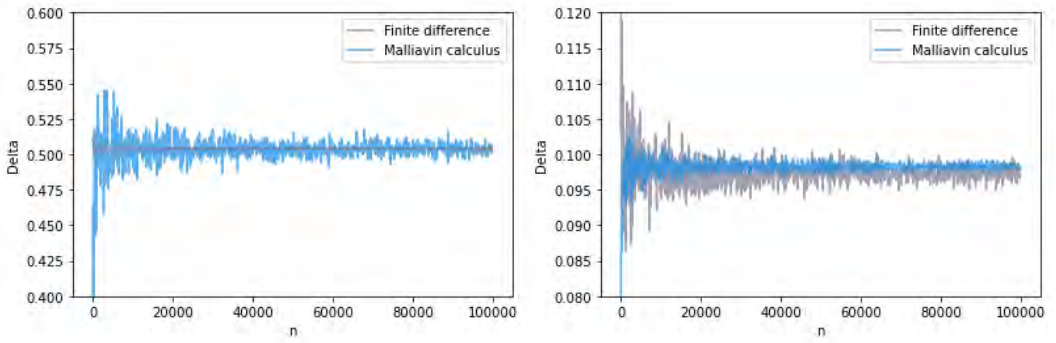


Figure 8: Convergence of the delta of a plain vanilla call option (left) and a digital call option (right) as a function of sample size n when using finite difference and Malliavin calculus.

[23]The indicator function $\mathbb{1}$ of an event is a random variable that takes value 1 when the event happens and value 0 when the event does not happen

6.2 Environment

To implement the reward function as described in Section 5.2.4, a value of 2 is used for the risk aversion parameter κ. This parameter is based on [80], which showed that the optimum of the efficient frontier, representing the PnL gain over the reward volatility, is attained close to this value. For the transaction costs defined in Equation 92, a tick size of 0.1 is used, as in [65]. To represent the final wealth, the accounting formulation is used, as the RL agent did not learn using the cash flow formulation given the trained set of parameters.
The Heston model is implemented to simulate the stock price and hence to define the transition function, as well as the stock price paths to calculate the option's value and delta. For practical applicability, the model parameters are based on [41]. These parameters are shown in Table 3 and are not calibrated to the real market. Calibration involves choosing the model parameters such that the stock price predictions are close to the observed market data. This is a task on its own and the focus of this research is on the general useability of RL to hedging, rather than fitting the RL agent to market data.

Table 3: The hyperparameters used in the Heston model.

Hyperparameter	Value
θ	0.1
ρ	-0.2
κ	1.5
ϵ	0.2
r	0
ν_0	0.2^2
S_0	100.0

The RL agent was first trained on the following grid of fixed discount factors:
$\gamma \in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.91, 0.95, 0.99]$. A discount factor of $\gamma = 0.9$ seemed to work for most option types. The discount factor was then increased through training in combination with the validation step as described in Section 5.2.5. Using a decay rate of 0.1, the following grid was tested for the decay step: [100, 150, 180, 200]. A higher discount value could be obtained in this way for the plain vanilla option, but it resulted in instabilities for the other option types. Finding an appropriate set of hyperparameters that would avoid these instabilities would require training on different grids or extending the learning cycle (i.e. more training steps). As a fixed discount factor of $\gamma = 0.9$ seemed to work for most option types, the analyses performed in this research are based on this value. The impact of the discount factor is visually shown in Section 7.4.3.

To speed up the training, 32 training environments run in parallel. An additional environment is created for the evaluation of the RL agent during the training. This environment uses a fixed seed, such that the evaluation takes place on a fixed stock price path to ensure that the improved hedging performance cannot be attributed to stochasticity. As training the RL agent requires intensive computational work, access to the GPU usage on a VU BAZIS HPC cluster is granted to train multiple RL agents in parallel and to speed up the training runs.

6.3 Reinforcement Learning Model

The initialisation and training of the DDPG algorithm require the determination of certain parameters. These values are shown in Table 4. This section describes and motivates the choice of the most important ones.

Recall that the DDPG algorithm has four neural networks: an actor network, which proposes an action given an observed state, a critic network, which predicts the expected Q-function

Table 4: The hyperparameter values used in the DDPG algorithm.

Hyperparameter	**Value**
Replay memory size	8192
Mini-batch size	128
Warmup episodes	64
Time steps/episode	50
Total iterations	5000
Train episodes/iteration	32
Test episodes/iteration	1
Train steps/iteration	1
Test steps/iteration	1
Exploration	Time inverse decay with initial value 0.5, decay rate 1.0, decay step 250.
	Minimum value clipped at 0.05
TD error loss	Element-wise Huber loss
Target update rate	0.999
Target update period	1 episode
Actor learning rate	Time inverse decay with initial value 0.005, decay rate 1.0 and decay step
	250, Rectified Adam optimizer
Critic learning rate	Time inverse decay with initial value 0.05, decay rate 1.0 and decay step
	50, Rectified Adam optimizer
Actor network	[Dense(256), Dense(256), Dense(1)] [”relu”, ”relu”, ”tanh”]
Critic network	State: [Dense(256), Dense(256)] [”relu”, ”relu”] Action: - Joint: [Dense(256), Dense(256), Dense(1)] [”relu”, ”relu”, ”linear”]
Gamma	0.9

for a state-action pair, and two target networks which are updated slowly to keep the estimated targets stable. For these networks, there are several structures and depths possible. Adding more layers allows for efficient representations of the interactions within the input data [32]. However, very deep neural networks or complicated structures require a lot of hyperparameters that need to be tuned, as well as a lot of data to be trained on which can become very challenging and costly. The most basic deep neural network, which is the multi-layered perceptron (MLP), seemed to be sufficient for this research and is hence chosen. An MLP consists of interconnected neurons and can be divided into three main layers: an input layer, the hidden layers that perform computations on the input data, and the output layer that returns an output. The number of hidden layers of neurons to use per layer cannot be calculated analytically and has to be determined e.g. using a grid of values. For the actor, two hidden layers are chosen. To output one single action, the output layer consists of one neuron. To bound the actions, the hyperbolic tangent (tanh) is used as an activation function, which takes any real value as input and outputs values in the range -1 to 1. The last layer is initialised between -0.003 and 0.003, to prevent getting 1 or -1 output values in the initial stages, which would squash the gradients to zero due to the tanh function. As the critic outputs a single estimated Q-value for a state-action pair, it has both the state as input as well as the action. The state is passed through the same layers as in the actor network, but instead of returning an action, it is concatenated with the action. Two hidden layers are added on top of the concatenated ones to let the MLP extract more abstract features. To output a single real value, the output layer of the critic consists of one neuron with a linear activation function. The hidden layers of the actor and critic network consist of 256 neurons. The Rectified Linear Unit (ReLu) function is used as an activation function for these layers, which is the default activation for MLPs [63] and outputs the input directly if it is positive and zero otherwise.[24]

[24]Note that no dropout activation is applied after the non-linear activation functions in this research. This technique randomly drops out neurons during trading to reduce overfitting and improve generalization

The weights of the networks are updated based on mini-batches which are sampled from the replay buffer and by applying gradient descent. The stochastic gradient descent (SGD) method has been widely used [51] as an optimizer. It calculates the gradient, based on a single sample or mini-batch instead of the entire training data as in gradient descent. The RL agent was tested on this optimizer, as well as on two of its variants: the Adam and Rectified Adam (RAdam) optimizer. A disadvantage of SGD is that due to frequent updates, the steps taken towards the minima are noisy. This may lead to a slower convergence and a gradient descent into the wrong direction. The Adam optimizer circumvents these problems by using an adaptive learning rate for each parameter: it accelerates the SGD in the relevant direction. A disadvantage of this optimizer is the fact that the adaptive learning rate can be quite large in the early stage of the training, leading to a bad convergence. The rectified Adam (RAdam) adjusts for this by rectifying the variance of the adaptive learning rate. Based on the other hyperparameters and a grid of learning rates, the RAdam seemed to perform best and is hence chosen in this research.

Regarding the exploration, recall that the DDPG algorithm adds a noise term to the action generated by the actor network to ensure the exploration. In the early stage of the training, the agent has not learnt yet and the standard deviation of this noise term is set at 0.5 to let the agent explore and learn more about the environment. As the training progresses, the agent learns more about the environment and the standard deviation decays using an inverse time decay schedule so that the likelihood of exploration becomes less probable and such that the agent exploits the environment more and more.

One of the most critical hyperparameters left is the learning rate of the networks. A rate that is too high may result in large gradients and divergent behaviour, whereas a value that is too small will slow down the training and may get stuck with a high training error [32]. In this research, the commonly employed technique learning rate annealing is applied [59]. Here, the training starts with a relatively high learning rate to move away from the randomly intialised parameters towards a range of appropriate parameters, and lowers during the training to explore a more optimal set of weights. The learning rate of the critic is set higher than the actor's rate because if the actor changes faster than the critic, the estimated Q-value is based on past policies and does not truly represent the current action. For the initial value of the time inverse decay schedule, a grid of [0.001, 0.005, 0.01] and [0.01, 0.05, 0.1] for respectively the actor and critic was tested. A step size grid of respectively [100, 250, 500] and [10, 50, 100] was tested. For the actor/critic, a rate of 0.005/0.05 that decays $\frac{1}{250}/\frac{1}{50}$ iterations with a rate of 1.0 seemed to work best. Using this schedule, these rates have a value of 2e-4 and 5e-4 in the final stage of the training.[25] The impact of the learning rate on the RL agent's performance is discussed in more detail in the Section 7.4.1.

error [38]. As the networks are trained on simulated data with a random seed, the chance of overfitting is negligible.

[25]It should be noted that the RAdam optimizer allows for a less sensitive configuration due to its adaptive learning rate.

7 Results

This chapter presents and evaluates the empirical performance of the reinforcement learning (RL) agents trained on the plain vanilla, digital, and barrier call option. The analyses are conducted based on 10,000 (almost surely) out-of-sample simulated episodes. The results of the RL agents are compared to those of the delta hedging strategy. For this, statistical tests are performed at a significance level of 1%.[26] The performance of the agents in the absence of transaction costs is discussed in Section 7.1. Section 7.2 discusses the impact of transaction costs. The robustness and flexibility of the agents are tested in Section 7.3. Finally, the impact of some of the hyperparameters chosen to train the agents is analysed in Section 7.4.

7.1 Results Without Transaction Costs

This section analyses the performance of the RL agents in a cost-free environment. Table 5[27] shows a summary of the performance of the RL agents and the corresponding delta hedging strategies per option type in terms of the final portfolio value Π_T. This table shows the average portfolio value (the mean hedging error μ), the average of the absolute portfolio value (the mean absolute hedging error MAE), the standard deviation (σ), and the 5^{th} and 95^{th} percentile of the portfolio values. μ indicates whether the strategies are on average good at hedging the options. For example, if the hedging errors reveal a symmetrical distribution closely centered near zero, the hedging strategy results in both gains and losses, but the hedging error is on average zero. The mean absolute hedging error does not consider the cancelling out of gains and losses, but concerns whether or not the strategy can offset any potential loss or gain. Based on this table, it seems that reinforcement learning is quite effective in hedging a plain vanilla call, digital, UAI barrier, and DAO barrier option. The performance on the UAO barrier and DAI barrier call option seems to be worse. The following subsections show and discuss the results per option type in more detail.

Table 5: The performance of the reinforcement learning agent versus delta hedging in terms of the terminal portfolio value per call option type in the absence of transaction costs.

	RL hedging					**Delta hedging**				
Option	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\mathbf{5^{th}pct}$	$\mathbf{95^{th}pct}$	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\mathbf{5^{th}pct}$	$\mathbf{95^{th}pct}$
Vanilla	-80.81	80.85	36.27	-145.12	-25.78	-157.10	157.10	61.69	-258.23	-62.89
Digital	1.68	32.25	37.68	-56.17	57.69	-2.79	43.28	55.06	-75.90	106.60
UAI	-113.39	113.44	42.77	-180.87	-50.23	-156.61	156.61	64.07	-265.82	-62.22
UAO	24.30	37.40	48.00	-44.53	110.75	0.54	34.57	60.79	-58.60	127.06
DAI	13.32	50.94	111.30	-24.82	252.10	-28.57	38.74	46.35	-124.19	37.66
DAO	-69.06	69.36	40.16	-133.89	-8.35	-149.35	149.35	62.37	-259.77	-69.59

7.1.1 Plain Vanilla Call Option

This subsection considers the simple European call option in the absence of transaction costs. An out-of-sample simulated episode of the trained RL agent is shown in Figure 9. It seems that the agent learnt how to hedge the option. Due to the discretization error, the option value and hedge value, presented in the lower right subplot, do not and will likely

[26] The probability of rejecting the null hypothesis given that the null hypothesis was assumed to be true. The null hypothesis is rejected if the p-value, i.e. the probability of obtaining a result at least as extreme as the observed one, is lower than this threshold. The result of an experiment is then said to have statistical significance.

[27] It is difficult to compare the results between the different call option types. This is because the hedging problem, scale of the portfolio value, and hence hedging error are different for each option due to the different option characteristics.

not coincide. Nevertheless, it is clearly visible that the hedge value tracks the value of the option and picks up its spikes. In addition, it learnt to replicate the delta: its position tracks the delta position, even though this value was not provided to the agent.

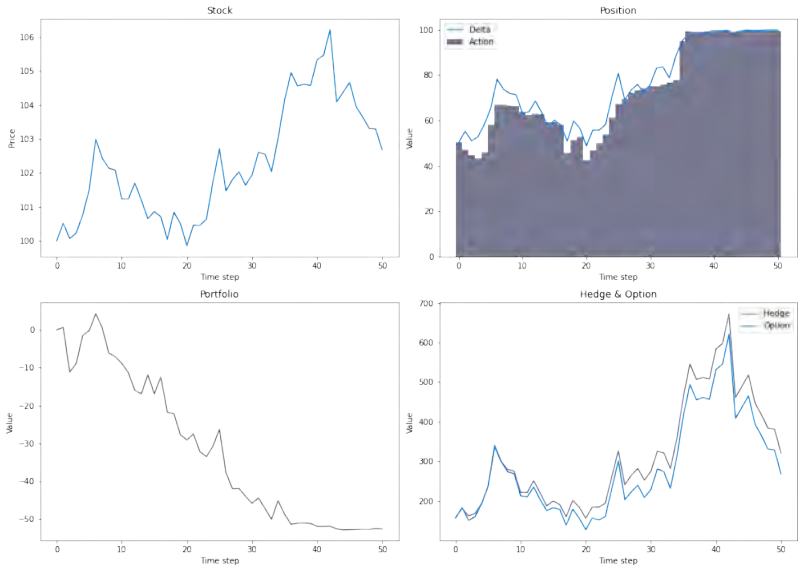


Figure 9: A single simulated episode of an RL agent trained on a plain vanilla call option. The stock price evolution is presented in the upper-left plot. The delta (blue) and agent's position (grey) are presented in the upper-right plot. The lower-left plot shows the portfolio value and the difference between the hedge value and option value is presented in the lower-right plot.

The fact that the agent learnt how to hedge the vanilla call option is confirmed by the value function, which is shown in Figure 10. This figure is based on the same simulated episode as in Figure 9. The figure shows the critic's estimated Q-values based on the scaled rewards (rather than the absolute hedging errors) per time step and action. For each time step, the actions that result in the highest rewards are displayed in green and the ones that obtain the lowest rewards are displayed in red. During the learning, the actor and critic interplay. At the beginning of the training, the critic is still inaccurate. This results in the actor taking quite random actions. As the training progresses, the accuracy of the critic increases. As a result, the actor is better able to select the best action which maximizes the value function. Destabilizing of the learning is in the same way visible in the value function. A wrong step by the actor or critic might adversely affect the other, resulting in a less accurate value function. In the case of an accurate critic and actor, one would expect a value function with distinct contour levels in which the actions taken by the agent have the highest values and where the estimated Q-values decrease as one moves away from the optimal actions. This is the case for the plotted value function. In addition, the estimated Q-values are accurate. As the option is initialised at-the-money (ATM), the option can go either in-the-money (ITM) or out-of-the-money (OTM). Due to this uncertainty, both a full investment and no investment at all in the underlying stock are then not optimal. As time passes, the green area, i.e. actions with the highest values, converges to the top. This is in line with the stock price movements because from step $\sim$35 onwards, the stock price is sufficiently high that the option is likely to end up ITM. To hedge the option then correctly, one should fully invest in the underlying stock. Smaller positions are less optimal, which is reflected in the value function showing a lower estimated Q-value for these actions.

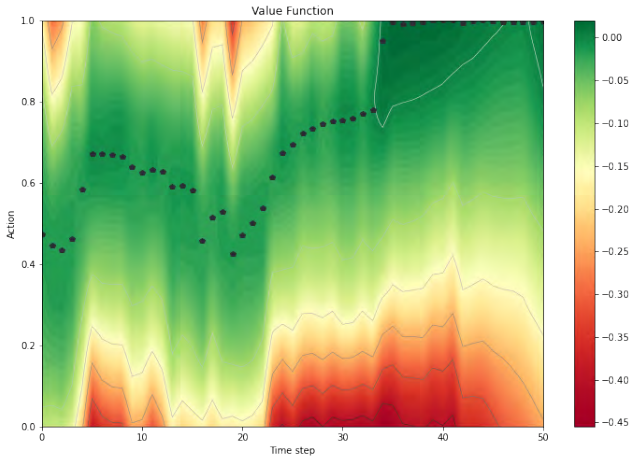


Figure 10: Value function of the single simulated episode per one share of the vanilla call option's underlying stock.

To further analyse the performance of the trained RL agent, the distribution of the agent's terminal portfolio value and that of the delta hedging strategy are shown in Figure 11. It can be observed that both distributions mostly take negative values. They seem to have a normal shape, which is left-skewed for the delta hedge strategy. The distribution of the RL agent is more shifted and centered towards the right, i.e. towards zero, compared to the delta hedging one. In addition, the average terminal portfolio value of the RL agent is higher than that of the delta hedge. This result is statistically significant, supported by the paired two-sample t-test. The RL agent's average absolute portfolio value is significantly lower than that of the delta hedge. Besides, the two-sample Kolmogorov-Smirnov test rejects the null hypothesis that both samples come from a population with the same distribution.

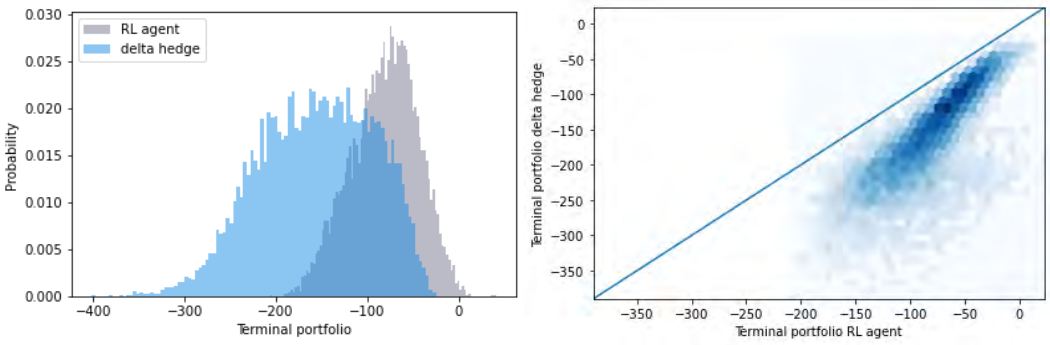


Figure 11: Distributions of the terminal portfolio value of the RL agent and delta hedge (left) and their relationship on a per episode basis (right) for the vanilla call option.

The hexbin plot in Figure 11 (on the right) shows the relationship between the terminal portfolio value of the delta hedging strategy (Y-axis) and the corresponding value of the RL agent (X-axis) on a per episode basis. A clustering of most points along the 45-degree line would indicate a similar performance for both strategies. The plot shows that the higher concentration of points, represented by the darker areas, is located on the center-right, under the diagonal. As a portfolio value of zero would be optimal, i.e. no hedging error, this confirms that the RL agent results in lower hedging errors than the delta-hedging strategy on a per episode basis. Furthermore, the darker areas are quite centered; there are no high

concentrations of points located far away from the center. This indicates that there are no clusters of outliers, i.e. episodes in which one of the strategies is doing much better than the other.

To compare the performance of both strategies throughout the episode, the standard deviation of the PnL (individual profit and losses) within an episode is depicted for both strategies in Figure 12. The distribution of the RL agent is narrower than that of the delta hedging strategy. This is confirmed by the Kolmogorov-Smirnov test, which indicates that the distributions are from different populations. In addition, the RL agent has a significantly lower volatility than the delta hedging one. Furthermore, the distributions of the strategies are skewed to the right, i.e. encompass a right tail with larger values of the standard deviation. To investigate where the larger values come from, the episodic portfolio values of the RL agent and delta hedge are binned into gridded hexagons in the right subplot of Figure 12. Here, the bin colours represent the bin's average episodic standard deviation of the PnLs, rather than the number of points in the bin as in the hexbin in Figure 11. The darker the colour, the higher the standard deviation of the bin. It can be observed that the lightest bins are located in the upper right corner of the plot, i.e. near the point (0, 0). The corresponding episodes result in a low terminal portfolio value, have thus smaller hedging errors, and have a smaller standard deviation of the PnLs within the episode. It is likely that these paths end far ITM or OTM and are hence easier to hedge; they entail less uncertainty and the optimal position in the underlying stock fluctuates less. Therefore, paths that are harder to hedge are expected to create greater hedging errors and a bigger standard deviation of the PnLs within an episode. The latter is supported by the hexbin, as the bin colour is darker if one moves below the diagonal.

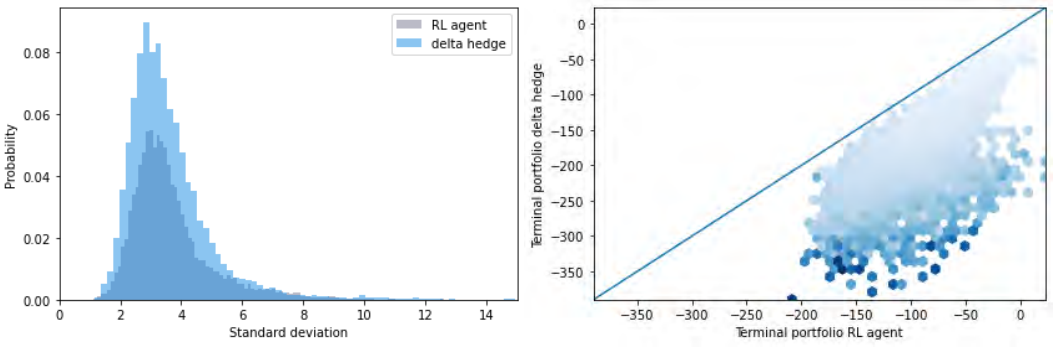


Figure 12: Distributions of the standard deviation of the PnL within an episode (left) for the RL agent and delta hedge, and the binned portfolio values of both strategies coloured by the average standard deviation within the bins' episodes (right) for the vanilla call option.

To take a closer look at the distribution of the PnL through the option's lifetime, a density plot of the RL agent's distribution over time is shown in Figure 13. This plot also shows a density plot of the difference of the PnL between the two strategies over time on a per episode basis. It follows that the average performance of the delta hedge strategy is quite stable throughout the episode. For the RL agent, the median and 45-55$^{\text{th}}$ percentile are at the beginning close to zero, decrease and show more dispersion over time, and increase again halfway to maturity. A possible explanation for this behaviour is that as time passes, there is often less uncertainty whether the option will end up OTM or ITM. As a result, the agent might be better able to know what action results in the highest reward. However, this is not the case if near maturity, the stock price is near-the-money. Hedging is harder in this case; more extreme PnLs and a less distinct value function can then be observed. The average difference in the absolute value of the PnL between the RL agent and delta hedging strategy on a per episode basis, depicted in the right subplot, is negative. This means that the RL agent also results in a lower hedging error and hence outperforms the

delta hedging strategy on a per time step basis. The two-sample paired t-test indicates that this outperformance is statistically significant, except for the last two time steps. The latter is supported by the figure, which shows that the distribution of the difference disperses in the last $\sim$ 10 time steps and that the average difference converges towards zero in these time steps.

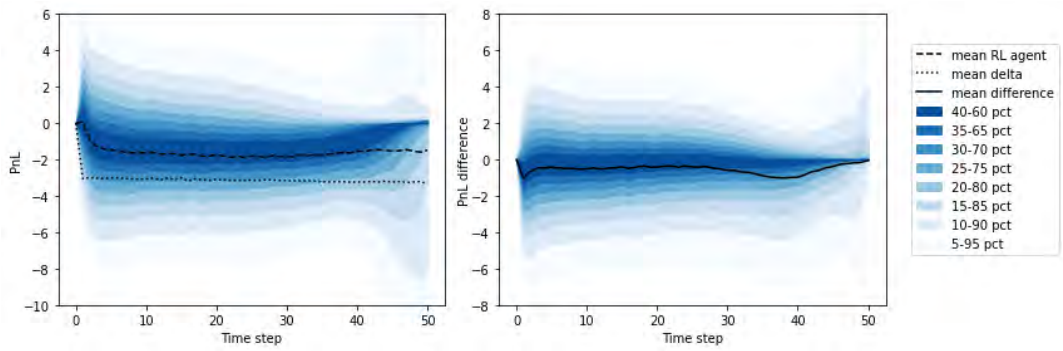


Figure 13: Distribution of the PnL of the RL agent (left) and the difference between the absolute portfolio of the RL agent and delta hedging on a per episode basis (right) through the plain vanilla call option's lifetime. The colours show the corresponding percentiles.

The results show that the RL agent is able to hedge the plain vanilla call in the absence of transaction costs. It outperforms the delta hedge strategy with a lower average (absolute) hedging error and a lower variance.

7.1.2 Digital Call Option

Following the vanilla call, the tests described in the previous subsection are also conducted on the digital call option. It seems that the RL agent is also able to hedge this option type. Overall, the hedge value seems to track the option value and the value function is accurate. For a digital option, the most complex decision to take is when the stock moves near the strike just before maturity. Near maturity, the delta of the option is zero, except near the strike. As a result, one should invest a large amount in the stock if its price is equal or close to the strike, sell everything if it moves away from the strike, and so on. The large positions are very risky, as the stock can end up just OTM, resulting in no payoff and making the investment worthless. An example of such a pin risk is shown in Figure 14, along with the corresponding value function in Figure 15. Around time step 43, the stock price reaches the strike, decreases then, and approaches the strike again. This leads to a delta spike in the upper right plot and a value function being less distinct and showing lower Q-values from this time step onwards. It becomes especially problematic when the stock price decreases in the last time steps. Although a large position is required to match the option value, buying shares of the underlying stock results in a decreased hedge value ($= H_{t-1} + a_t(S_t - S_{t-1})$). This is because the stock price difference between two time steps is negative in that case. As a result, the portfolio value also decreases, leading to a high loss at maturity. The agent might need to see more often these events to understand that it should not do anything. But even then, the stock price might end up just ITM, resulting in a high loss. The agent would not encounter this problem when the stock price is near the strike, but increases just before maturity. Although a large investment in the underlying stock is still risky in that case, investing in the underlying stock results in a higher hedge value and a smaller hedging error since the stock price difference is then positive. This highlights the particularity of the digital option: the fact that the payoff doesn't depend on the direction of the movement in price. This makes hedging complicated. Although the overall behaviour of the agent is good, actions in such events remain difficult to take and may result in high losses.

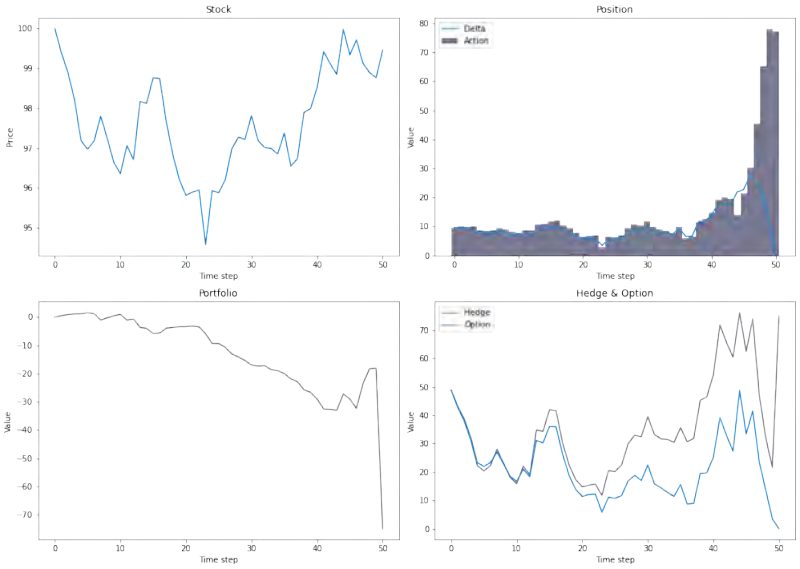


Figure 14: A single episode of a digital call option with an example of a pin risk.

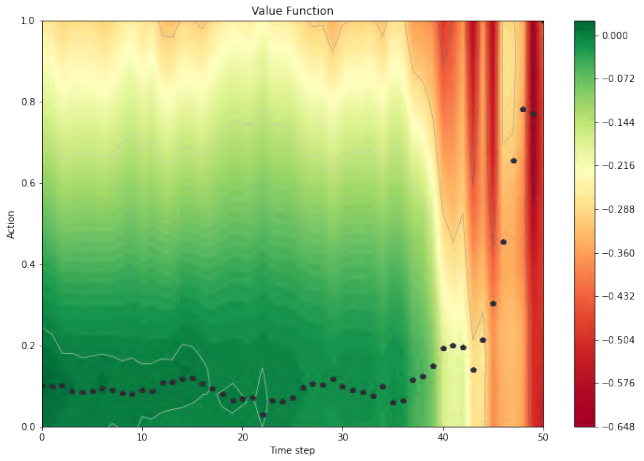


Figure 15: Value function of the single simulated episode per one share of the digital option's underlying stock.

To compare the performance of the RL agent with that of the delta hedging strategy, the distributions of the terminal portfolio value of both strategies are shown in Figure 16. The distribution of the RL agent shows two lumps below and above zero and is more centered around zero compared to the distribution of the RL agent trained on the plain vanilla call option. The distribution of the delta hedging strategy has a similar negative lump but has a longer right tail instead of a second positive lump. The two-sample Kolmogorov-Smirnov test rejects the null hypothesis that both samples come from a population with the same distribution. On average, the RL agent and delta hedge have respectively a negative and positive terminal portfolio value. The t-test confirms that the RL agent's average portfolio value is significantly higher than that of the delta hedge. In addition, the RL agent results in a significantly lower average of the absolute portfolio values (MAE) compared to the hedging strategy. The hexbin plot in Figure 16 shows one clear cluster of points that lies

more or less on the diagonal. This suggests a similar performance for both strategies. A second, less concentrated cluster can be observed on the right, which is also centered around the diagonal. Compared to the plain vanilla's hexbin, the points are less spread out.

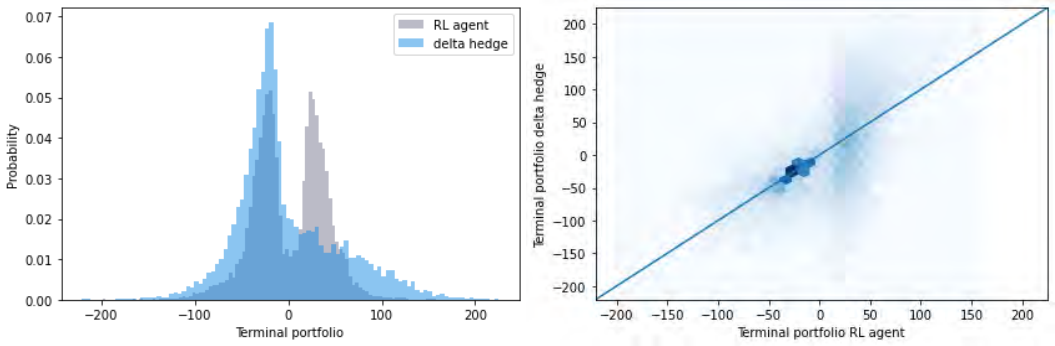


Figure 16: Distributions of the terminal portfolio value of the RL agent and delta hedge (left) and their relationship on a per episode basis (right) for the digital call option.

When looking at the PnLs within an episode obtained by both strategies, the RL agent has a lower standard deviation whose distribution is narrower compared to the delta hedge. These two observations are statistically significant, supported by the two-sample t-test and the Kolmogorov-Smirnov test, and illustrated in the left subplot of Figure 17. The binned portfolio values of both strategies, coloured by the average standard deviation within the bins' episodes, are shown in the right subplot. A similar pattern can be observed as for the vanilla call: the lightest bins are located in the center around (0,0), i.e. are likely to contain episodes that are easier to hedge. Paths that result in higher portfolio values encompass a higher standard deviation.

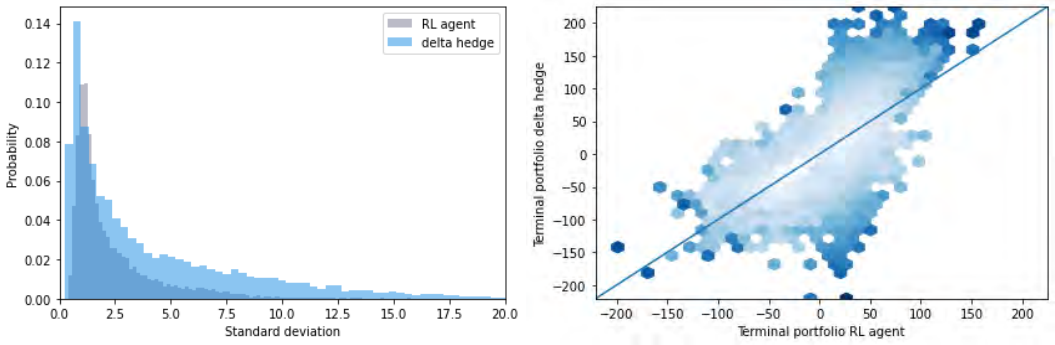


Figure 17: Distributions of the standard deviation of the PnL within an episode (left) for the RL agent and delta hedge, and the binned portfolio values of both strategies coloured by the average standard deviation within the bins' episodes (right) for the digital call option.

The distribution of PnL through the option's lifetime is presented in Figure 18 to further analyse the agent's performance on a per episode basis. The distribution of the RL agent as well as the difference in the absolute portfolio value between the two strategies disperses as time passes. The widths increase in particular in the last five time steps. Before this point, the distributions are narrower and more concentrated around zero compared to the results for the plain vanilla call option. Next to the fact that the initial digital option value is lower compared to the plain vanilla one, the observed behaviour can be explained by the option's delta. The sensitivity of the option value, i.e. the delta, is fairly constant low, except near expiration. As described in Section 3.2.2, the delta can take high values for near-the-money options at this point and may change extremely fast, leading to high replication errors. This

is reflected in the figure. When comparing the average PnL value of the RL agent and delta hedge over time, the average of the former is slightly higher, except for the last few time steps. The absolute PnL value is significantly lower for the delta strategy, except for the last time step where it is significantly lower for the RL agent. There is no significant difference at the 47^{th} and 48^{th} time step. This, combined with the fact that the greatest replicating errors occur in the last few time steps explains the lower absolute terminal portfolio value for the RL agent.

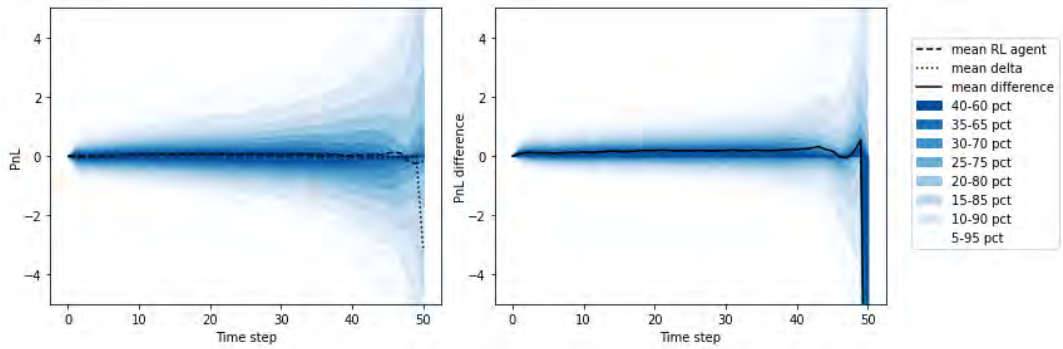


Figure 18: Distribution of the PnL of the RL agent (left) and the difference between the absolute portfolio of the RL agent and delta hedging on a per episode basis (right) through the digital call option's lifetime.

The results show that the RL agent is able to hedge the digital call option in the absence of transaction costs. It significantly outperforms the delta hedging strategy in terms of the (average) absolute portfolio value and standard deviation.

7.1.3 Barrier Call Option

Finally, the tests are conducted on the barrier call option. Within this option type, it seems that the RL agent is able to hedge the up-and-in (UAI) and down-and-out (DAO) barrier call option. It picked up the dynamics of these option types and the delta spikes when the stock price approaches the barrier near maturity. Similar to the digital call option, difficulties arise when the stock moves close to the barrier (DAO)/just before maturity (UAI). Compared to the delta hedging strategy, the average terminal portfolio value of both options is significantly higher. In addition, the average absolute portfolio values are significantly lower than that of the delta hedge. The standard deviation of the PnL within an episode is for both also significantly lower than the delta hedging one. The distributions of the terminal portfolio value have a similar shape as the plain vanilla call option, although the DAO barrier option seems to take slightly more positive values and the UAI barrier option slightly more negative values. However, the options' distributions of the PnL's standard deviation within an episode deviate from that of the vanilla call option. These distributions are presented in Figure 19: the distribution is more shifted to the right/left for the UAI/DAO barrier call option. For the options' terminal portfolio value, as well as the standard deviation within the episode, the Kolmogorov-Smirnov test shows that the samples of the RL agent and the delta hedge come from a population with a different distribution.

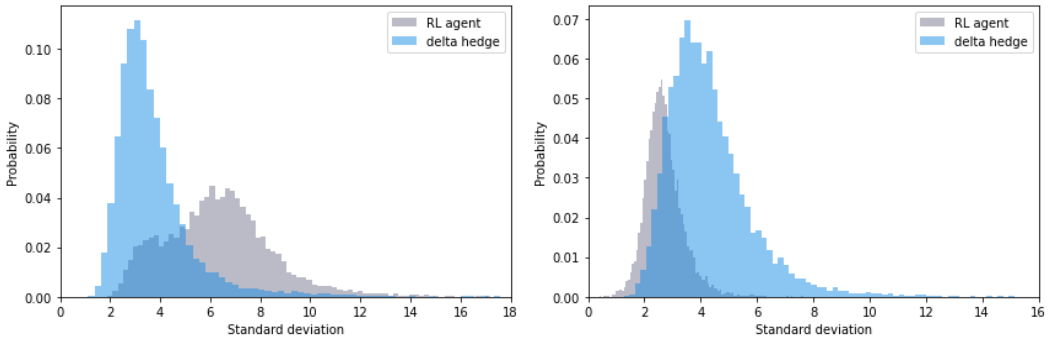


Figure 19: Standard deviation of the PnL within an episode for the up-and-in (left) and down-and-out (right) barrier call option.

Another striking aspect is the distribution of the PnL through the UAI barrier option's lifetime, illustrated in Figure 20. The average PnL of the delta hedging strategy is quite stable again throughout time. The average value of the RL agent is at the beginning of the episode lower compared to the plain vanilla call and increases towards zero as time passes. At the same time, the PnL distribution is in the beginning quite wide but narrows and converges towards zero over time. An explanation for this behaviour might be that in the beginning of the episode, there is an additional uncertainty due to the barrier compared to the plain vanilla call option. This uncertainty often decreases as time passes: the stock moves either up, reaching the barrier or increasing the likelihood that it will be reached, or down, making it less likely that the option will reach the barrier and end ITM. As a result, the agent might be better able to take the correct action as time passes [28]. This is also visible in the right subplot: the difference in the absolute PnL decreases over time. Although the RL agent's average PnL value is significantly less negative (closer to zero) compared to the delta hedge, except for the first 9 time steps, the absolute PnL is significantly lower for only one time step.
The corresponding distribution for the DAO barrier option is similar to that of the plain vanilla option. It seems to be narrower for the last time steps, but this is difficult to interpret this chart as the episodes are terminated once the barrier is kicked out.

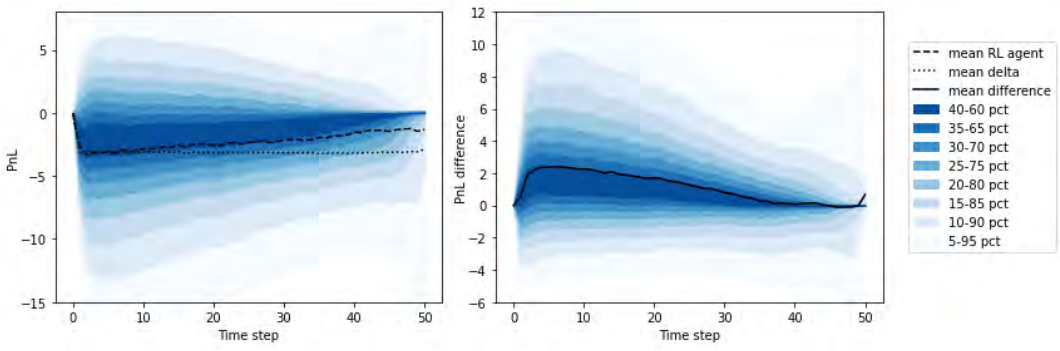


Figure 20: Distribution of the PnL of the RL agent (left) and the difference between the absolute portfolio of the RL agent and delta hedging on a per episode basis (right) through the up-and-in barrier call option's lifetime.

The RL agent did not learn how to properly hedge the up-and-out (UAO) barrier option. It learnt to pick up the extreme delta spikes, e.g. if the stock price approaches the barrier

[28] Note that this does not hold for the events where the barrier has not been kicked in yet and the stock price moves up near the strike near maturity.

near maturity. However, the RL agent systematically under- or over-hedges in the simulated episodes. As a result, the hedge value shows a less accurate tracking of the option value. In addition, due to the allowance of negative values, the agent occasionally takes negative actions, even though the price is increasing without being too close to the barrier. This is visually shown in Appendix A. The delta hedging strategy outperforms the RL agent. This is visible in Figure 21, which shows that the average PnL of the former oscillates more around zero except for the last few time steps. In addition, the (absolute) value of the terminal value is significantly lower for the delta hedging strategy.

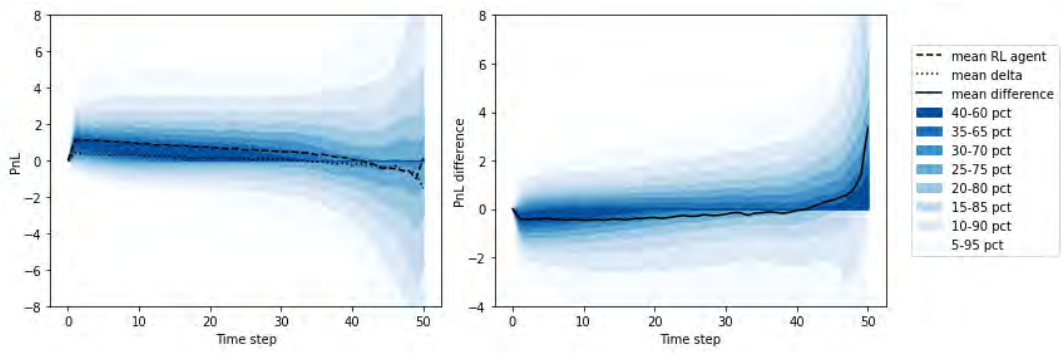


Figure 21: Distribution of the PnL of the RL agent (left) and the difference between the absolute portfolio of the RL agent and delta hedging on a per episode basis (right) through the up-and-out barrier call option's lifetime.

The performance of the RL agent is even worse for the down-and-in (DAI) barrier call option. The agent learnt to rarely take a position in the underlying stock rather than the dynamics of this option. As will be explained later in this subsection, the option ends often OTM. Hence, small positions in the underlying stock do often not lead to high portfolio values. However, the hedging behaviour itself is incorrect: the hedge value does not match the option value throughout the option's lifetime and the agent does not take a larger position in the underlying stock if the barrier has been kicked in and if the option ends ITM. This is visible in Table 5: although the average portfolio value of the DAI option deviates significantly less from zero than the delta hedging one, its standard deviation and average absolute portfolio are significantly higher. The fact that the distribution of the DAO option is more spread out is also reflected in the values of the 5-95$^{\text{th}}$ percentile. The inability of the RL agent to hedge the DAI option correctly is also supported by the inaccurate value function. This is blurred and does not show a converging path of optimal actions throughout the episode. This is visually shown in Appendix B.

The fact that the agent is not able to hedge the UAO and DAI barrier option correctly is in line with the more complicated delta behaviour of these options compared to those of the DAO and UAI option as described in Section 3.2.3. In addition, these options have a smaller payoff range or one that is more difficult to enable in this research setting.[29] The UAO barrier option only results in a payoff if the stock price ends between the strike price and the barrier level, while the barrier has not been exceeded during the option's lifetime. The payoff range is even more difficult to enable for the DAI barrier option. Here, the stock price has to move down to reach the barrier and then move above the strike to get a payoff. Even though the barrier is reached and the option is kicked in in 66% of the simulated episodes, it is less likely that the downward movement is followed by such an upward movement within the option's lifetime of only 10 trading days. These events might let the agent learn to take small positions or no position at all in the underlying stock, hence incorrect hedging. To validate the impact of the smaller payoff range on the hedge performance of

[29]This is reflected in a lower initial value compared to the other option types, as shown in Table 1.

the UAO and DAI option, an adjusted version of the UAO barrier option was tested first. In this setting, the option was knocked out if the price of the original UAO barrier value was exceeded for more than x% during the option's lifetime. Here, a threshold of 100% represented the original barrier option and 0% the plain vanilla call option. However, these tests did not show any conclusive results. This might be explained by the fact that the agent must learn the additional impact of the threshold value, which makes the hedging problem more complex.
A second attempt made to validate the impact of the smaller payoff range on the hedge performance was to let the UAO barrier option end more often ITM by increasing the barrier level. For this, four RL agents were trained on barriers that were respectively 25%, 50%, 75%, and 98% less often kicked out compared to the original UAO barrier call option. The corresponding barrier values and minimum and maximum action of these options were determined in the same way as described in Section 5.2.2. From a financial perspective, it is to be expected that a higher barrier eases the hedging task, as the option is less often kicked out and the delta behaviour is more similar to that of a plain vanilla call option. However, no consistent pattern could be observed between the barrier level and hedge performance of the RL agent. A possible explanation for this might be that as the option is less and less often kicked out, the agent is seeing too few such events. It might be unable to correctly approximate the dynamics that realise the rare behaviour, resulting in a great training loss that is used when calculating the cumulative reward for other state-action pairs. This can then in turn lead to accumulated errors and destabilization of the learning. A solution might be to use a prioritised replay buffer, in which experiences that led to an important difference between the expected reward and the obtained reward (i.e. the rare events) are more frequently sampled.
For the UAO option, one could also argue that the incorrect hedging can be attributed to the fact that the option is terminated once the barrier is hit. As a result, the agent sees the states less often, as future states are no longer discovered after hitting the barrier. However, this is also the case for the DAO option, but the RL agent is still able to hedge this option correctly.
Finally, the hedging of a DAI barrier option has an additional complexity from an RL perspective. Based on the observed state, the agent cannot derive whether the barrier has been hit during the option's lifetime. This is not the case for the knock-out options, as these are terminated once the barrier is hit. For the UAI barrier call, it may only be an issue when the stock price is between the strike and the barrier. This price range is relatively small and the agent might still be able to approximate the chance of being kicked in given the remaining time to maturity, the previous position, and the scaled stock price. As this range is much wider for the DAI barrier option, further research could investigate the impact of adding a state component, indicating whether the barrier has already been hit, would improve the hedging performance of this option.

The results indicate that RL can be used to correctly hedge a UAI and DAO barrier option in the absence of transaction costs. Both RL agents outperformed the corresponding delta hedging strategy. The performance is worse for the UAO option and the RL agent did not learn how to hedge the DAI barrier option.

7.2 Impact of Transaction Costs

This section analyses the performance of the RL agents in a cost environment. A summary of the performance of the RL agent and the delta hedging strategy in terms of the final portfolio value Π_T is presented per option type in Table 6. Also in the presence of transaction costs, the RL can be used to hedge a plain vanilla, digital, UAI barrier, and DAO barrier option. For these options, the introduction of transaction costs reinforces the conclusions regarding the outperformance of the RL agent over the delta hedging strategy. These agents result in an average portfolio that deviates significantly less from zero, a significantly lower

average absolute portfolio value, and a significantly lower standard deviation of the portfolio value than the corresponding hedging strategy. The RL agents also outperform the delta hedge in terms of the standard deviation of the PnL within an episode, whose distributions are narrower and show a shorter right tail compared to the corresponding delta hedge strategy. For the UAO barrier option, the RL agent picked up some of the delta spikes, but the delta hedging strategy again outperformed the agent. The RL agent is still not able to hedge the DAI option: it incorrectly learnt to take a full investment in the underlying stock during the entire lifetime of the option. The remainder of this section focuses on the options that the RL agent can hedge (the vanilla, digital, UAI and DAO barrier call option).

Except for the UAI barrier call option, the distributions of the options' portfolio value are negatively shifted. This is likely because the transaction costs flow directly into the PnL, resulting in the terminal portfolio taking more negative values. The results of the plain vanilla call option and digital call option are further analysed, as the results of the UAI and DAO barrier option leverage on these.

Table 6: The performance of the reinforcement learning agent versus delta hedging in terms of the terminal portfolio value per call option type in the presence of transaction costs.

	RL hedging					**Delta hedging**				
Option	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\mathbf{5^{th}}$**pct**	$\mathbf{95^{th}}$**pct**	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\mathbf{5^{th}}$**pct**	$\mathbf{95^{th}}$**pct**
Vanilla	-107.32	109.15	40.37	-164.41	-51.01	-190.18	190.18	72.24	-310.53	-79.62
Digital	-18.50	29.80	31.82	-67.77	31.03	-19.29	45.09	52.40	-99.18	79.66
UAI	-88.17	88.46	40.39	-155.70	-26.87	-188.78	188.78	73.94	-316.09	-80.62
UAO	32.76	47.91	63.45	-45.77	159.11	-9.29	35.62	54.48	-75.96	110.61
DAI	0.22	296.80	396.02	-698.68	642.33	-44.08	50.32	57.47	-164.82	26.99
DAO	-77.37	77.65	42.87	-141.69	-7.65	-174.44	174.44	74.00	-305.45	-78.93

For the plain vanilla call option, the introduction of transaction costs introduces a delay and reduction of the actions taken by the agent. As the trading of arbitrary small amounts of the underlying stock is costly, the agent does not always act immediately when the delta spikes up (down), but waits to see if the delta returns to the previous lower (higher) value due to stock price movements. The agent hedges the position if the stock price increases (decreases) even further. These results are in line with what is shown in [80] and supported by Table 7. This table presents the average position and the average absolute change in position within an episode for the agents that had been trained respectively in the absence and presence of transaction costs. It follows that these average values are indeed lower for the RL agent that had been trained in the presence of transaction costs. The paired t-test indicates that these results are significant.

Table 7: The average position and average absolute change in position of two RL agents trained in the absence/presence of transaction costs on a per episode basis for the vanilla call option.

Environment	**Mean**	**Absolute change**
No costs	48.91	3.08
Costs	48.57	2.97

To further analyse the performance between the agents in a cost-free and a cost environment, the RL agent that had been trained in the absence of transaction costs is tested in an environment with transaction costs. Similarly, the RL agent that had been trained in the presence of transaction costs is tested in an environment without transaction costs. For both agents, the resulting average portfolio value as well as the average absolute portfolio value is presented in Table 8. The agent being trained without costs results in a significantly higher average portfolio value and a significantly lower absolute portfolio value in both a cost and

a cost-free environment. The former was to be expected: when employing a policy in the same environment as it was learnt, the performance is expected to better than that of a policy that was learnt in a different environment. For this reason, the results regarding the cost environment are striking. To further analyse this, the difference in the average costs between the two agents in the cost/cost-free environment is calculated for each episode, accounting for the fact that the agents do not always start from the same position. For a specific agent, the difference in its portfolio value between the cost-free and cost environment can be explained solely by the transaction costs resulting from trading in the underlying stock. This is because the agent is in each scenario tested on the same set of paths: an agent sees the same states in the cost-free and cost environment and takes hence the same actions in both. Therefore, if the difference in the average portfolio value between two agents (trained in different environments) in a cost environment is equal to the adjusted average transaction costs, the difference in performance can be attributed solely to the difference in transaction costs resulting from different positions taken. However, this is not the case for the two agents that were trained in different environments; the difference between their average portfolio values may be explained by a better performance of the agent trained without costs. As the introduction of transaction costs changes the hedging problem, the agents might not have reached the same level of accuracy during training. By tuning the hyperparameters of the RL algorithm and/or increasing the number of iterations, the agent trained with costs should be able to discover the right actions. In this way, it should be feasible to obtain a policy that is at least as good as a policy that is learnt in the absence of costs.

Table 8: Results of the two RL agents (one being trained in the absence, the other in the presence of transaction costs and both tested on a cost and cost-free environment) in terms of the average portfolio value (μ, left) and the average absolute portfolio (MAE, right) for the vanilla call option.

Trained \ Tested	No costs	Costs
No costs	-80.81	-104.93
Costs	-84.21	-107.86

Trained \ Tested	No costs	Costs
No costs	80.85	104.93
Costs	87.73	109.78

The results for the digital option are at first sight striking. Although the average portfolio value of this option is more negative in the presence of transaction costs, the average absolute portfolio value is significantly lower. This might again be explained by the option's delta. As previously described and shown, the delta is fairly constant low, except near expiration where the delta can change extremely fast. In the absence of transaction costs, the agent learnt to track the delta: it took large investments if the stock price approached or exceeded the strike price and sold it again if the stock price moved away from the strike in these last time steps. In the presence of transaction costs, large moves in the agent's position are very costly. Due to the convexity of the cost function as defined in Equation 92, it is less costly to build up a position over a longer period of time, rather than taking a large position at once in the last time steps. The higher the fees, the less willing the agent is to take such large positions in the underlying stock just before maturity. As a result, the presence of transaction costs smooths out the actions taken by the agent. This is supported by Figure 22. It presents the behaviour of the RL agent on the same simulated episode as shown in the absence of transaction costs in Figure 14. The positions and moves in the amount hold are quite lower in the last few time steps compared to the situation without transaction costs.

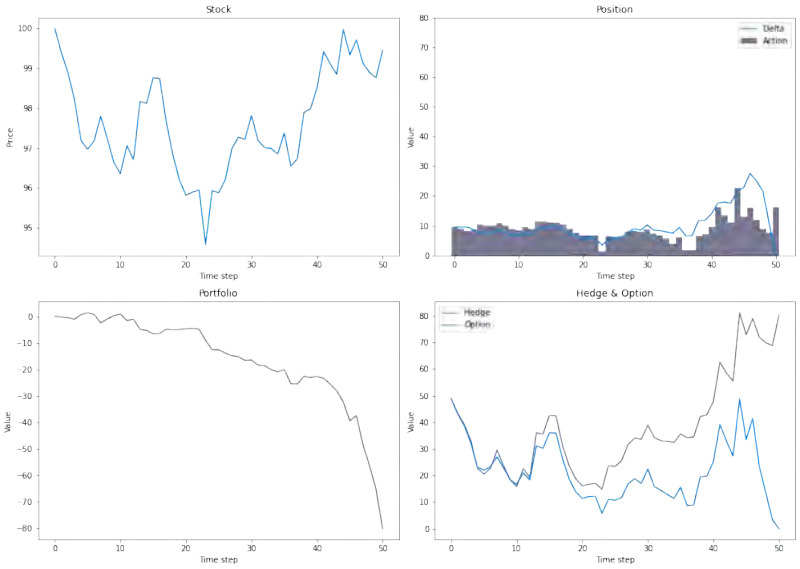


Figure 22: A single episode of a digital call option with an example of a pin risk in the presence of transaction costs.

The distribution of the agent's position in the digital option's underlying stock, presented in Figure 23, supports the fact that the presence of transaction costs smooths out the action space. It follows that distribution in the absence of transaction costs is narrower compared to the situation with transaction costs, except for the last few time steps. The distribution disperses earlier in the absence of transaction costs and takes higher values compared to that in the presence of transaction costs. This is reflected in the mean of the position over time: it is higher for the first ~ 30 time steps in the presence of transaction costs, after which it is higher for the environment without transaction costs. Furthermore, the average position and the average absolute change in position on a per episode basis, presented in Table 9, are significantly higher in the presence of transaction costs.

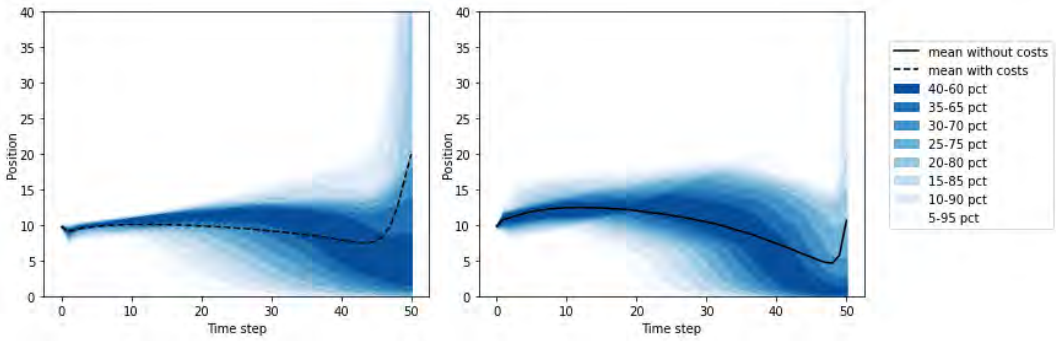


Figure 23: Distribution of the position in the underlying stock of the digital call option of the RL agent in the absence of transaction costs (left) and the presence of transaction costs (right).

Table 9: The average position and average absolute change in position of two RL agents trained in the absence/presence of transaction costs on a per episode basis for the digital call option.

Environment	Mean	Absolute change
No costs	9.66	1.43
Costs	10.09	1.52

As the transaction costs might avoid large moves in the agent's position in the last time steps, it might also help to avoid large hedging errors in these steps. To validate this assumption, the RL agent that had been trained in the absence/presence of transaction costs is tested on an environment with/without transaction costs. The results in terms of the average (absolute) portfolio value are shown in Table 10. On a per episode basis, the average absolute portfolio value is significantly lower for the RL agent being trained in a cost environment. This holds both when it is tested in the absence and presence of transaction costs. So, the presence of transaction costs seems to lead to a smoother sequence of actions taken by the agent. This seems to avoid large hedging errors near/at maturity and in this way to an improved hedging performance of the digital call option. This is in line with the call spread approach used in practice to hedge digital calls, see Section 3.2.2. Again, tuning the hyperparameters and/or increasing the number of iterations should allow the agent that is trained in a cost-free environment to learn the smoothed hedging strategy and hence to obtain a policy that is at least as good as the one that is learnt in the presence of transaction costs.

Table 10: Results of the two RL agents (one being trained in the absence, the other in the presence of transaction costs and both tested on a cost and cost-free environment) in terms of the average portfolio value (μ, left) and the average absolute portfolio (MAE, right) for the digital call option.

Trained \ **Tested**	**No costs**	**Costs**
No costs	1.68	-7.08
Costs	-8.61	-17.77

Trained \ **Tested**	**No costs**	**Costs**
No costs	32.25	32.92
Costs	28.33	29.70

For the UAI barrier call option, the average portfolio value deviates significantly less from zero and the average absolute portfolio value is significantly lower in the presence of transaction costs compared to the environment without transaction costs. This might be explained by the fact that the UAI barrier option acts as a digital option when the barrier has not been hit. Following the same line reasoning as for the digital call option, the introduction of transaction costs might avoid large moves in the agent's position when the stock price moves near the barrier. This might in turn avoid large hedging errors. Compared to the digital option, the distribution of the position in the underlying stock over time shows different behaviour. This is illustrated in Figure 24. The distribution starts earlier to disperse on both sides in the absence of transaction costs. This might be explained by the additional payoff uncertainty introduced by the barrier. As a result, the agent might keep a more constant position at the beginning of the option's lifetime in the presence of transaction costs. As time passes, the uncertainty often decreases, resulting in the agent taking larger positions to hedge the option correctly.

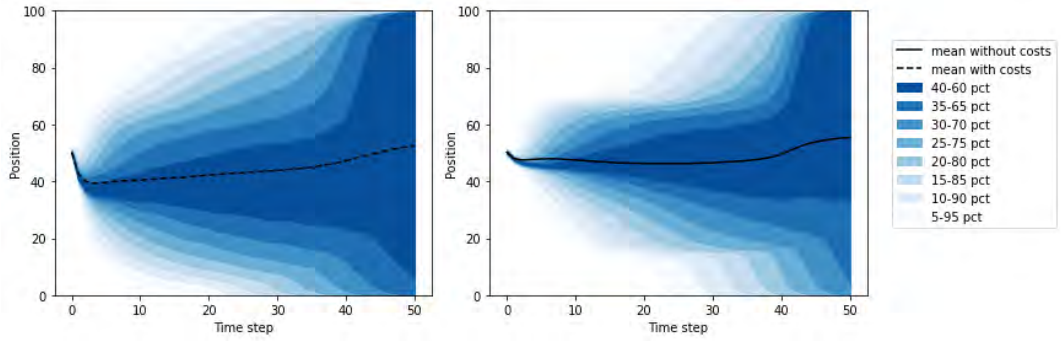


Figure 24: Distribution of the position in the underlying stock of the UAI barrier call option of the RL agent in the absence of transaction costs (left) and the presence of transaction costs (right).

7.3 Generalizability

This section analyses the generalizability of the RL agents. The robustness against changes in some of the parameters of the option is evaluated in Section 7.3.1. Section 7.3.2 describes the agent's flexibility with respect to the learning of multiple option types simultaneously.

7.3.1 Modified Option Characteristics

The previous sections tested the performance of an RL agent on an option that had the same characteristics as the one on which the agent had been trained. When changing one of the option's parameters, the option value and hence optimal hedging behaviour change. This subsection tests the robustness of the agent against changes in the strike price, initial value of the underlying stock, initial value of the instantaneous variance, and barrier level. For this, the RL agent trained on the plain vanilla call is used to investigate the impact of changes in the first three parameters, and the RL agent trained on the UAI barrier call option is used for the latter. The agents are not re-trained and tested on environments without transaction costs. As the option value changes the PnL scaling, the ratio of the RL agent's terminal value to the delta hedging one is presented on a per episode basis. This is done to better compare the hedging performance of the two strategies across the modified option parameters.

7.3.1.1 Modified Strike Price

The behavior of the RL agent in a test environment with a different strike price is shown in Table 12. Recall that the agent had been trained on an option with strike $K = 100$. For the delta hedging strategy, it follows that the average portfolio value decreases in absolute value as the strike moves further away from $K = 100$ where the option is ATM. This was to be expected, as an option that is initialised with a sufficiently low (ITM)/high (OTM) strike is easier to hedge compared to an option that is initialised ATM. It involves less uncertainty, because it is likely that the option will remain ITM/OTM, and the optimal position, hence hedging error, will change less throughout an episode. The opposite pattern is visible for the RL agent. Although its average portfolio value does not change drastically, the absolute hedging error and standard deviation are the lowest for the strike on which the agent had been trained and these increase as the strike moves away from this value. The diminishing performance for the RL agent combined with an improving portfolio value for the delta hedging results in an increasing ratio between both when one moves away from $K = 100$. Based on these results, it follows that the RL agent is robust against small changes in the strike price (2-3%) and still significantly outperforms the delta hedging strategy in these cases. However, the agent's performance gets worse for larger changes in the initial strike price. This is visually shown in Appendix C. The delta hedging strategy is beneficial in these cases. This can be explained by the fact that the strike price is not explicitly captured in the state. When being trained on a fixed strike price, the agent was able to derive the payoff structure and hence the strike from the obtained rewards. When the strike changes, the initial position would be lower/higher compared to the trained environment in which the option was ATM. Based on this, the agent may understand that the option is ITM/OTM and hence adjust the current position accordingly. However, if the strike deviates too much, without being captured in the state, the agent will likely under- or over-hedge at some point in the episode, leading to an accumulation of hedging errors. Training on a range of strikes simultaneously might improve the agent's generalizability, although it may also average out the payoff and result in learning instabilities. As the strike price is explicitly captured in the option moneyness, capturing this variable in the state may also be considered to improve the agent's generalizability.

Table 11: The mean and standard deviation of the RL agent's and delta hedging's terminal portfolio value, and their ratio on a per episode basis for different strikes K. For these strategies, as well as the ratio between both, the scenario with the lowest portfolio's standard deviation is highlighted in grey.

	RL hedging			Delta hedging			Ratio RL/delta	
K	μ	MAE	σ	μ	MAE	σ	μ	σ
95	-85.00	146.32	178.85	-68.30	68.31	64.24	13.28	346.30
96	-84.80	133.67	155.85	-92.83	92.83	68.72	2.27	4.09
97	-83.89	116.81	126.45	-117.00	117.00	69.50	1.18*	1.21
98	-82.85	98.38	92.48	-137.86	137.86	67.17	0.73	0.57
99	-81.60	84.52	57.11	-151.59	151.59	62.83	0.54	0.31
100	-80.81	80.85	**36.27**	-157.10	157.10	**61.69**	0.52	**0.13**
101	-80.04	81.88	56.03	-152.82	152.82	63.71	0.52	0.28
102	-79.35	91.97	89.76	-140.20	140.20	68.07	0.65	0.50
103	-79.14	108.81	122.29	-121.55	121.55	71.25	0.97	0.87
104	-78.99	124.86	150.01	-99.59	99.59	71.21	1.54	1.54
105	-78.89	137.22	172.27	-77.19	77.19	67.09	2.54	2.77

7.3.1.2 Modified Initial Stock Price

A similar is pattern is visible when changing the initial value of the underlying stock S_0. This is presented in Table 12. Recall that the RL agent had been trained on $S_0 = 100$, where the option is ATM. The option gets more OTM/ITM as S_0 moves away from this value. Following the same reasoning for the modified strike price, this results in a lower average portfolio value for the delta hedging strategy. Just like for the modified strike, the absolute hedging error and standard deviation of the RL agent are the lowest for the initial stock price ($S_0 = 100$) on which the agent had been trained. The performance in terms of the absolute portfolio diminishes as S_0 moves away from this value, resulting in an increasing ratio between both strategies. It seems that the RL agent is robust against small changes (2-3%) in the initial stock value. Although S_0 is captured in the state to calculate the logarithmic returns $\left(\log(\frac{S_t}{S_0})\right)$, the delta hedging strategy significantly outperforms the agent for a larger change in the initial stock price. The impact is similar to that what is shown for the modified strike Appendix C. A reason might be that for two different values of S_0, an equal percentage change in the stock price outputs the same state variable, but may not yield the same option moneyness and hence optimal action. Including the option moneyness in the state might not only improve the agent's generalizability with respect to the strike but possibly also to the (initial) stock price.

Table 12: The mean and standard deviation of the RL agent's and delta hedging's terminal portfolio value and their ratio on a per episode basis for different initial values of the underlying stock S_0.

	RL hedging			Delta hedging			Ratio RL/delta	
S_0	μ	MAE	σ	μ	MAE	σ	μ	σ
95	-74.92	132.84	168.36	-67.92	67.92	62.13	2.91	3.25
96	-75.79	122.09	147.91	-91.91	91.91	67.88	1.67	1.70
97	-76.77	107.13	121.36	-116.04	116.04	69.33	1.01	0.92
98	-77.75	90.74	89.33	-136.70	136.70	66.73	0.66	0.51
99	-79.23	81.10	55.78	-151.18	151.18	63.07	0.52	0.28
100	-80.81	80.85	**36.27**	-157.10	157.10	**61.69**	0.52	**0.13**
101	-82.40	85.28	57.35	-153.20	153.20	63.42	0.54	0.31
102	-84.45	99.63	92.90	-141.29	141.29	68.31	0.72	0.56
103	-86.33	118.67	127.36	-122.64	122.64	71.58	1.13	1.12
104	-88.11	136.59	157.83	-100.52	100.52	71.84	2.00	3.06
105	-89.23	150.88	182.63	-77.83	77.84	69.17	12.32	487.33

7.3.1.3 Modified Initial Instantaneous Variance

Contrary to the modified strike and initial stock price, the RL agent shows the same behaviour as the delta hedging strategy when changing the initial value of the instantaneous variance ν_0. This is shown in Table 13. For both strategies, the terminal portfolio increases in absolute value as ν_0 increases. This can be explained by the fact that a lower ν_0 makes the hedging task easier since the expected stock price movements are smaller. As a result, the option value and hence the optimal hedging position tend to be more steady. This gives smaller hedging errors. Higher volatility leads to more sudden price jumps, which introduces more uncertainty in the option value and creates more hedging errors in the discrete setting. Based on these results, it seems that a single training may be sufficient to properly hedge a call option with any realistic value of ν_0: the RL agent is able to deal a volatility change of around 30% (from $\sqrt{0.2} \approx 0.44$ to $\sqrt{0.075} \approx 0.27$ and to $\sqrt{0.375} \approx 0.57$). For this range, the RL agent significantly outperforms the delta hedging strategy in terms of the (ratio of the) average terminal portfolio value as well as its standard deviation. The RL agent is more robust against changes in ν_0 compared to a modified K or S_0. A reason for this might be that the instantaneous variance is explicitly captured in the state to calculate the variance of the logarithmic returns ($\sqrt{\nu_t \Delta t}$). This adjusts for the impact of a modified ν_0, i.e. the expected stock price movement between two trading points due to the instantaneous variance. Note that the instantaneous variance is modelled as a mean-reverting process, see Section 3.1.3. This means that the instantaneous eventually reverts to the long-time average θ, regardless of the value of ν_0. Hence, for options with a longer lifetime, it may be interesting to investigate the impact of a modified θ or modified volatility of the instantaneous variance ϵ.

Table 13: The mean and standard deviation of the RL agent's and delta hedging's terminal portfolio value and their ratio on a per episode basis for different initial values of the instantaneous variance ν_0.

	RL hedging			**Delta hedging**			**Ratio RL/delta**	
$\boldsymbol{\nu_0}$	μ	**MAE**	$\boldsymbol{\sigma}$	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$
0.075	-36.06	36.12	**16.05**	-66.11	66.11	**26.80**	0.56	0.16
0.1	-44.11	44.16	19.75	-83.05	83.05	33.18	0.54	0.14
0.125	-52.81	52.85	23.71	-100.94	100.94	40.03	0.53	0.13
0.15	-61.92	61.96	27.82	-119.36	119.36	47.09	0.52	0.13
0.175	-71.28	71.32	32.02	-138.14	138.14	54.38	0.52	0.13
0.2	-80.81	80.85	36.27	-157.10	157.10	61.69	0.52	0.13
0.225	-90.46	90.50	40.56	-176.16	176.16	69.06	0.51	0.13
0.25	-100.19	100.23	44.89	-195.34	195.34	76.47	0.51	**0.13**
0.275	-109.98	110.02	49.23	-214.54	214.54	83.83	0.51	0.13
0.3	-119.82	119.87	53.59	-233.86	233.86	91.31	0.51	0.13
0.325	-128.79	128.92	58.43	-251.82	251.82	98.91	0.62	0.44

7.3.1.4 Modified Barrier Level

For the UAI barrier call option, the behaviour of the RL agent and delta hedging strategy in a test environment with a different barrier level is shown in Table 14. From a financial perspective, it is to be expected that a lower barrier level results in a lower absolute value of the terminal portfolio value. A lower barrier means that the barrier is more frequently kicked in. When the barrier is kicked in, the UAI barrier option behaves as a plain vanilla call option and is easier to hedge. For both the RL agent as well as the delta hedging strategy, it can be observed that the standard deviation of the terminal portfolio value is indeed lower for lower barrier levels. For delta hedging strategy, the (absolute) average value also gets worse as the barrier level increases, except for the first four barrier levels. Such a

pattern is not visible for the RL agent. Here, a good performance on a lower barrier level was also to be expected. The agent was able to hedge the option correctly when the barrier on which the agent had been trained was hit. As the option behaves as a plain vanilla call option from this moment onwards, the agent learnt the dynamics of the vanilla call option. For the lower tested barriers, the RL agent significantly outperformed the delta hedging strategy.[30] This also holds for a barrier that is slightly higher than the one the agent had been trained on. However, the delta hedging strategy seems to outperform the agent for a larger barrier.

Table 14: The mean and standard deviation of the RL agent's and delta hedging's terminal portfolio value and their ratio on a per episode basis for different UAI barrier levels B.

	RL hedging			**Delta hedging**			**Ratio RL/delta**	
$\boldsymbol{B}$	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$
95	-114.87	114.91	**39.00**	-157.10	157.10	61.69	0.83	0.39
100.5	-114.74	114.78	39.12	-157.07	157.07	**61.64**	0.83	0.39
101	-114.22	114.26	39.94	-156.76	156.76	61.98	0.82	0.38
101.5	-113.39	113.44	42.77	-156.61	156.61	64.07	0.82	0.37
102	-112.26	112.32	48.21	-156.60	156.60	68.59	0.80	**0.36**
102.5	-110.78	110.89	56.10	-157.18	157.18	76.33	0.79	0.36
103	-108.80	108.99	64.73	-158.55	158.55	86.91	0.77	0.36
103.5	-108.07	108.43	76.81	-161.65	161.65	101.47	0.76	0.47
104	-106.73	107.41	88.41	-162.93	162.98	113.10	0.76	1.25
104.5	-105.65	107.13	99.89	-163.34	163.64	123.17	0.77	2.04
105	-106.04	109.53	112.11	-163.68	164.59	134.36	0.84	2.12
105.5	-106.37	113.66	125.03	-161.25	163.31	141.79	3.24	49.30

7.3.2 RL Agent Trained On Different Option Types Simultaneously

In this research setting (given the fixed number of training iterations), the first attempts seem to indicate that it is difficult to train the RL agent on the plain vanilla, digital, and UAI barrier call option type simultaneously. For these options, the RL agent learnt to pick up the extreme delta spikes/drops in the options' dynamics during the training. However, it systematically under- or over-hedges and the hedging performance is worse than that of the RL agents trained on the options separately. This is supported by the fact that the DDPG algorithm did not converge: the actor and critic loss increased at some point during training. The fact that the RL agent has to learn multiple option dynamics and derive what type to hedge in a specific test environment hardens the hedging task. It is likely that it needs to see more training examples and requires a different set of hyperparameters (such as a lower learning rate) to learn the correct hedging strategy. As the RL agent did pick up some of the delta spikes/drops, additional runs using grids of hyperparameters, division of the option types over the training environments and a different (scaling of the) added scale component are required to conclude whether it can be used to hedge different option types simultaneously. This question is especially important if one wants to hedge a portfolio of options. Using different RL agents for each of the options separately would not be appropriate in this case, since it would not capture the correlation between the different options.

7.4 Impact of Hyperparameters

This section analyses the impact of the learning rate, optimization technique, discount factor, and seeds chosen to initialise the weights of the neural networks on the hedging

[30] A barrier level of 99.5 means that the option is already kicked in at inception. This level is added as a validation.

performance. The analyses are performed on the plain vanilla call option.

7.4.1 Learning Rate

This subsection investigates the impact of the learning rate on the hedging performance of the RL agent. This is done for the actor learning rate, as the rate for the critic did not seem to influence the performance that much. Using the time inverse decay scheme, nine different agents are trained on a grid of the following decay initial rates and decay steps: [0.001, 0.005, 0.01] and [100, 250, 500]. A higher value of the initial decay rate/decay step means a larger/faster update of the learning rate. The values of the resulting learning rates are shown per iteration in Figure 7.4.1. A log scale is applied for visualization purposes.

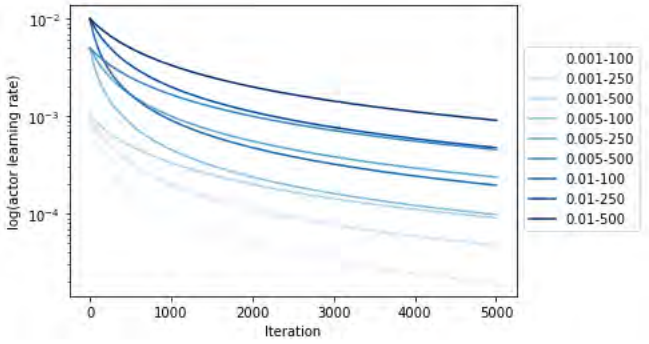
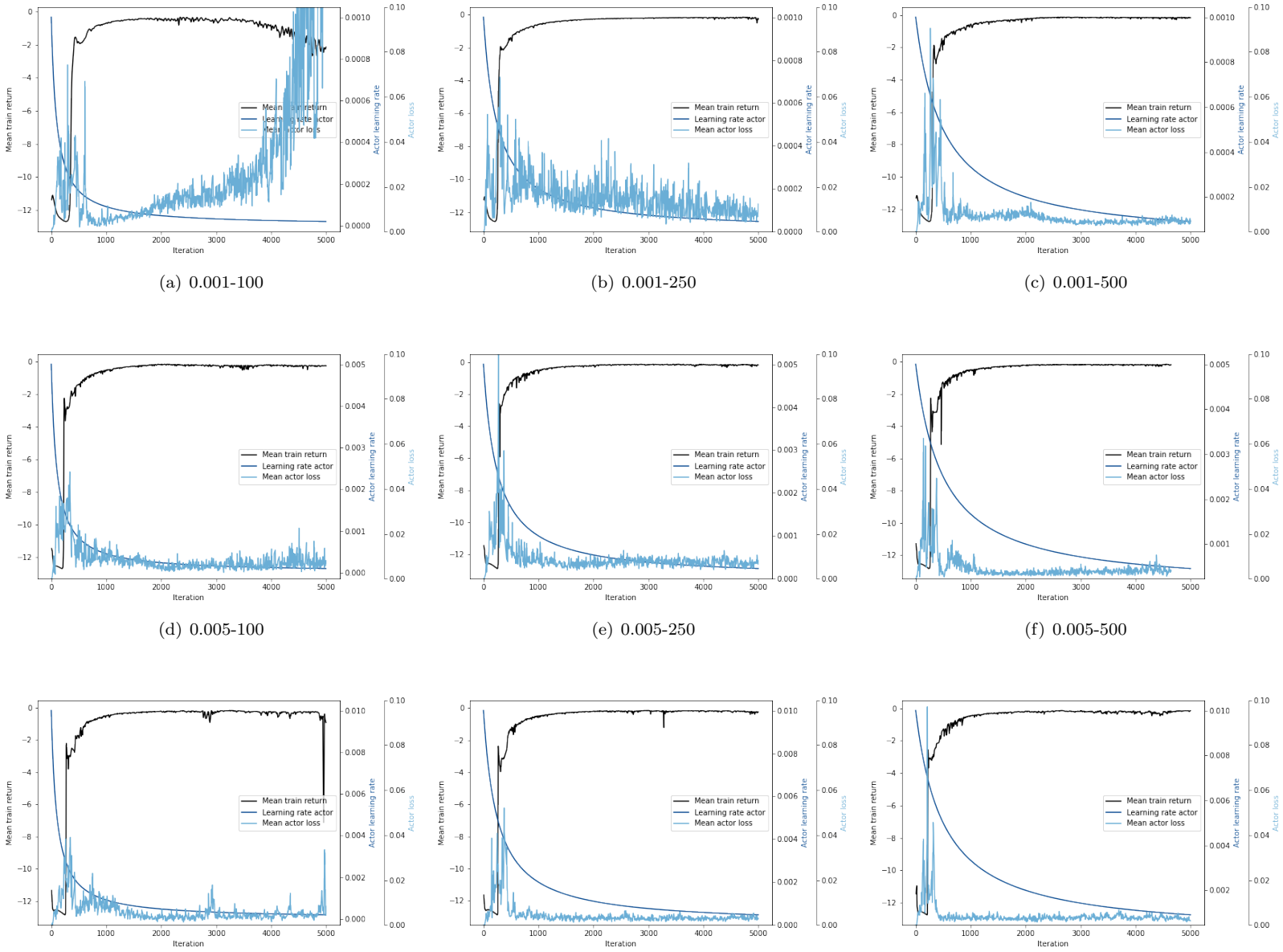


Figure 25: The learning rate of the actor (represented on a log scale) during training for different initial decay rates (represented by the first number in the legend) and decay steps (represented by the second number in the legend).

For each of these schemes, the results of the RL agent in terms of the portfolio value are presented in Table 15. The average train return[31], actor loss and learning rate during training are visualized in Figure 26. It can be observed that the actor loss has not converged for some of the decay schemes, in particular for the smallest initial learning rate (0.001) that decays the fastest (decay step of 100). The small weight updates may result in getting stuck for too long at a local minimum or saddle point, causing a high training error. The initial learning rate of 0.001 combined with a decay step of 250 reflects the slow converge caused by a small learning rate: the actor loss shows a downward trend, but takes higher values during training compared to the other converged plots. The average (absolute) portfolio of this scheme is also quite high. Extending the training cycle might lead to convergence and hence similar results to these of the other converged schemes. The performance improves when slower decaying this learning rate, by using a decay step value of 500. For the highest initial learning rate (0.01), the algorithms have converged, except in combination with a decay step of 100. The actor loss of this scheme shows a spike at the end of the training together with a drop in the average train return. Despite the convergence of the other two decay steps with the initial learning rate, the optimal average (absolute) portfolio value is not achieved at these schemes. This may be explained by the fact that a higher learning rate can cause the model to converge too quickly to a suboptimal solution. Although an initial learning rate of 0.005 seems to result in a slightly increasing actor loss at the end of the training when combined with a decay step of 100 or 500, this value seems to give an on average *good enough* weights.[32] It follows that the convergence of the algorithm and the performance of the RL agent are quite sensitive to the learning rate for the actor. Extending the grid of initial decay rates and decay steps might result in an even better convergence and performance of the model.

[31] The average undiscounted reward of an episode.

[32] Note that an initial decay rate of 0.001 combined with a decay step of 500 seems to perform better in terms of the portfolio value for the plain vanilla call option compared to a learning rate of 0.005 with a decay step of 250. However, the latter performs better on average when considering all the option types.



(a) 0.001-100 (b) 0.001-250 (c) 0.001-500

(d) 0.005-100 (e) 0.005-250 (f) 0.005-500

(g) 0.01-100 (h) 0.01-250 (i) 0.01-500

Figure 26: The average return, actor loss and actor learning rate for different inverse time decay schemes (initial decay rate - decay steps).

Table 15: The performance of the RL agent in terms of the terminal portfolio value for different initial decay rates and decay steps of the actor's learning rate.

Initial decay rate - decay step	μ	MAE	σ	5^{th}pct	95^{th}pct
0.001-100	-144.5	149.6	78.6	-246.8	15.8
0.001-250	-113.0	113.0	31.8	-168.1	-65.5
0.001-500	-79.0	79.1	30.9	-130.6	-29.3
0.005-100	-79.6	80.4	35.4	-136.4	-25.5
0.005-250	-80.8	80.8	36.2	-144.9	-24.8
0.005-500	4.1	98.5	123.0	-145.4	238.4
0.01-100	-133.5	134.4	46.2	-199.3	-63.5
0.01-250	-96.1	96.1	38.2	-160.3	-36.6
0.01-500	-88.0	88.1	33.8	-149.0	-37.4

7.4.2 Optimizer

This subsection investigates the impact of the optimization technique for minimizing the expected loss of the actor and critic network. Next to the Rectified Adam (RAdam) optimizer, which is chosen in this research as an optimization technique, the Stochastic Gradient Descent (SGD) and Adam optimizer are tried. The results in terms of the terminal portfolio value are shown in Table 16. It can be observed that the RAdam optimizer results in the lowest average absolute portfolio value and the lowest standard deviation, followed by the Adam optimizer. The SGD optimizer has the highest average hedging error, as well as the highest standard deviation. The differences in the average portfolio values are statistically significant. In addition, the distribution of the portfolio value is the most spread out for the SGD optimizer and the least for the RAdam. These patterns are supported by Figure 27, which shows the average actor loss and return for the different optimization techniques. It can be observed that although the SGD optimizer has the lowest actor loss through training, it results in the lowest average train return. The average train return is the highest throughout training for the RAdam optimizer. The same pattern is visible for the minimum train return and the maximum train return, which are the highest for the RAdam optimizer followed by the Adam technique. These results are in line with what is described in Section 6.3. They confirm the fact that the adaptive learning rate, applied by the Adam optimizer, leads to gradient descent in a better direction of improvement compared to SGD. Furthermore, it is visible that the Adam optimizer results in an actor loss that is more oscillating compared to the SGD and RAdam and that the average train return shows a slower convergence. This confirms the fact that rectifying the variance of this rate, applied by RAdam, leads to faster convergence. It follows that given the fixed number of training iterations, the RL agent learnt how to hedge the call option for all three optimization techniques, but the performance and speed of convergence are sensitive to the choice of the technique.

Table 16: The performance of the RL agent in terms of the terminal portfolio value for for the Stochastic Gradient Descent (SGD), Adam and Rectified Adam (RAdam) optimizer.

Optimizer	μ	MAE	σ	5^{th}pct	95^{th}pct
SGD	-196.50	196.50	56.69	-291.77	-109.78
Adam	-115.67	115.86	55.25	-219.36	-45.77
RAdam	-80.79	80.83	36.24	-144.92	-24.83

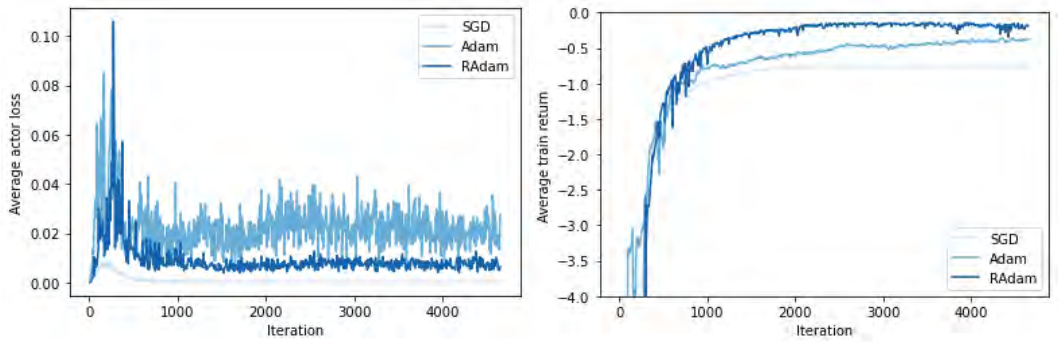


Figure 27: The average actor loss (left) and return (right) for the Stochastic Gradient Descent (SGD), Adam and Rectified Adam (RAdam) optimizer.

7.4.3 Discount factor

This subsection investigates the impact of the discount factor on the hedging performance of the RL agent. Eight different agents are tested on a grid of the following discount factors: $\gamma \in [0, 0.25, 0.5, 0.75, 0.9, 0.925, 0.975, 0.999]$.[33] For each discount factor, the results of the RL agent in terms of the portfolio value are presented in Table 17. Up to and including a factor of 0.9, the agent is able to hedge the option correctly and outperforms the delta hedging strategy. The average (absolute) portfolio value, as well as the standard deviation, is similar for these tested discount factors, and a nonzero discount factor leads to significantly lower absolute hedging errors. However, at some point, a discount factor higher than 0.9 results in error propagation during training and higher hedging costs: the absolute portfolio, as well as the standard deviation, increases significantly. These observations are supported by Figure 28, which shows the average actor loss and return for a discount factor $\gamma \in [0, 0.9, 0.925, 0.999]$. The agents that had been trained on a discount factor of 0 and 0.9 show a similar convergence of the actor loss. The average return is slightly higher for the discount factor of 0.9 compared to the factor of 0. The actor loss for the discount factor of 0.925 starts to increase after approximately 2000 iterations. In addition, the corresponding average return is lower than that of lower discount factors. Despite the higher actor loss and lower average return, the RL agent that had been trained on this factor, as well as on 0.975, was still able to pick up some of the delta and option spikes. This is also reflected in Figure 28, since the actor loss is significantly lower and the average train return is significantly higher for this discount factor than that for a discount factor of 0.999. For the latter, the RL agent incorrectly learnt to rarely take a position in the underlying stock, just like for the down-and-in barrier call option. This results in extreme gains and losses, that cancel out and hence lead to an on average "good" portfolio value, but a very high average absolute portfolio value. The corresponding actor loss shows diverging behaviour and the average return initially decreases and then stabilizes at a level that is significantly lower than that of the other discount factors.

[33] Recall that with a higher discount factor, future rewards are more strongly taken into account compared to a lower discount factor.

Table 17: The performance of the RL agent in terms of the terminal portfolio value for different discount factors.

Discount factor	μ	MAE	σ	5^{th}pct	95^{th}pct
0	-88.64	88.98	33.76	-148.13	-37.61
0.25	-81.16	81.16	28.12	-132.37	-39.72
0.5	-77.60	77.63	31.35	-131.40	-28.19
0.75	-85.47	86.40	36.06	-146.51	-32.12
0.9	-80.79	80.83	36.24	-144.92	-24.83
0.925	-157.42	157.44	62.25	-268.74	-64.36
0.975	-156.66	159.53	86.99	-281.16	7.01
0.999	-2.44	185.38	236.05	-160.67	494.42

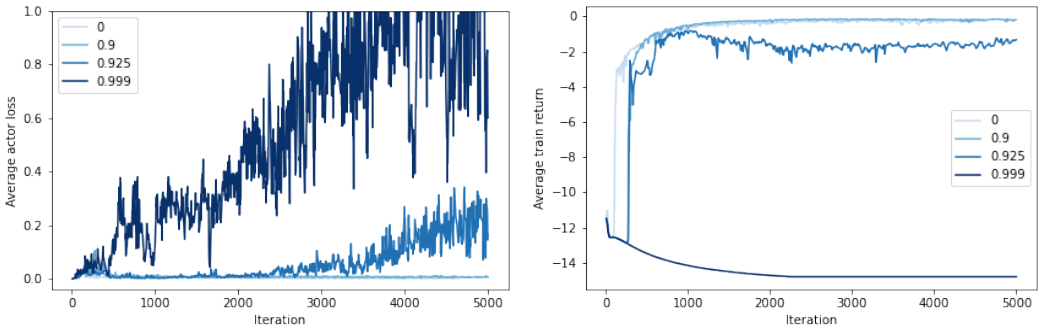


Figure 28: The average actor loss (left) and return (right) for the different discount factors.

Based on these results, it follows that a nonzero discounted factor leads to a better performance compared to a discount factor of zero. This factor can be stretched up to a point, after which the training destabilizes. The discount factor can be pushed even further towards 1 by softening the errors through training and tuning the hyperparameters. An example of this, along with a representation of the sensitivity to the hyperparameters, is shown in Figure 29. This figure shows the average return, actor loss, and discount factor for two different attempts aimed at increasing the discount factor through training. For the left subplot, the discount factor was increased up to approximately 0.97. Using the corresponding inverse time decay scheme, the actor loss and average return converged. The RL agent learnt to effectively hedge the plain vanilla call. The right subplot shows a diverging behaviour: the average actor starts to increase at an early stage of the training, leading to a propagation of instabilities. At the same time, the average train return is quite oscillating and shows a downward trend. This highlights the fact that the discount factor should not be increased at a rate that is too high. The longer the planning horizon, i.e. modelled by a higher discount factor, the more complex finding an optimal policy becomes and the greater the computational expense [27].

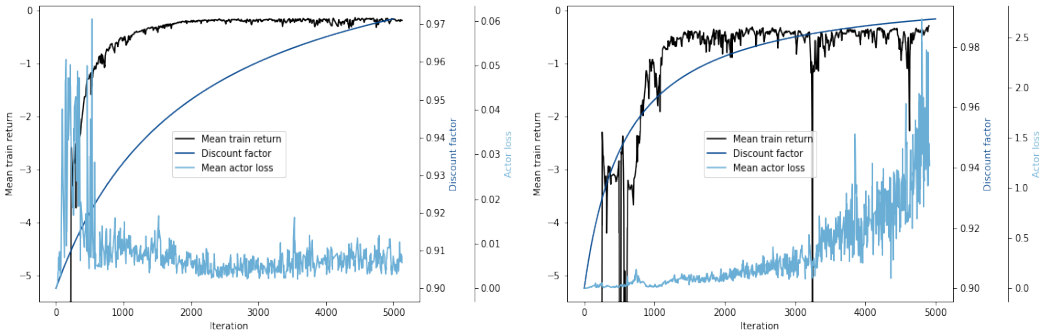


Figure 29: The average return, actor loss and discount factor for two different inverse time decay schemes (initial rate = 0.1, decay rate = 0.1 and 0.3 and decay steps = 200 and 180), starting from $\gamma_0 = 0.9$.

7.4.4 Seeds for Neural Networks' Weight initialisation

Weight initialisation is the procedure of setting the neural network's weights, which define the starting point for the training of the neural network. This subsection analyses the impact of the seed used in this procedure. For this, five different RL agents are trained on the plain vanilla call option using different seeds[34] for initializing the critic (seed=0-4) and actor (seed=6-10) network. All the trainings converged. The results in terms of the terminal portfolio value are shown in Table 18 and the corresponding distributions of the terminal portfolio value are presented in Figure 30. It follows that the distributions of the portfolio value are quite overlapping and similar. Although the second setting shows a higher peak and a slightly more concentrated distribution, none of the distributions show extreme values or an extremely deviating center towards the right or left. Despite the average portfolio values are significantly different for the tested seeds, they lie more or less in the same range (around -85). The values of the standard deviation also lie in the same range (around 35). In addition, the difference between the worst and best average portfolio value is less than half of the standard deviation. Furthermore, the RL agent learnt to hedge the vanilla call option and significantly outperformed the delta hedge for all tested combinations of seeds. Hence, it seems that in this research setting, the choice of the seeds for the neural networks' weight initialisation does not impact the overall conclusion regarding the RL agent's hedgeability and results in similar distributions of the portfolio value. Some deviation in the portfolio values was to be expected. As mentioned in [53], a neural network that is initialised with a different set of weights results in different starting points for the optimization process and might potentially result in a different final set of weights with different performance characteristics. In addition, due to stochasticity during training, the agents were not trained on the same stock price paths and/or order. This might also have led to some deviation in the portfolio values between the different agents.

[34] Apart from the fact that a fixed seed is used to initialise the weights, the weights are initialised in the same way as in the experimental setup: they are drawn from a uniform distribution within [-limit, limit], with limit = $\sqrt{3 \times scale/n}$ and n the number of input neurons.

Table 18: The performance of the RL agent in terms of the terminal portfolio value for different seeds used to initialise the neural network's weights.

Seed (actor - critic)	$\boldsymbol{\mu}$	**MAE**	$\boldsymbol{\sigma}$	$5^{\textbf{th}}$**pct**	$95^{\textbf{th}}$**pct**
1 (6-0)	-88.08	88.11	34.73	-149.85	-36.57
2 (7-1)	-77.05	77.29	30.32	-128.59	-28.01
3 (8-2)	-86.51	86.66	34.85	-147.15	-31.86
4 (9-3)	-90.74	90.99	38.06	-155.40	-30.80
5 (10-4)	-81.07	81.33	37.32	-141.37	-16.31

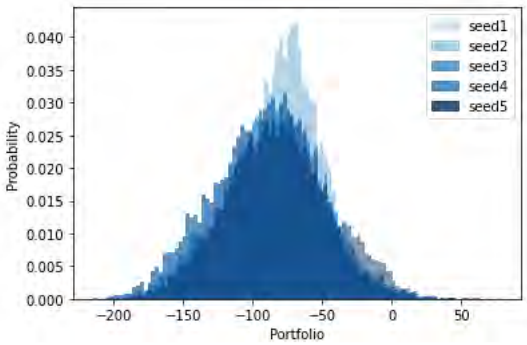


Figure 30: Distributions of the terminal portfolio value for different seeds used to initialise the neural network's weights.

8 Conclusion

The goal of this research was to investigate to what extent reinforcement learning (RL) can be used to hedge a plain vanilla, digital and different types of a barrier call option under the Heston model. To answer this research question, the hedging problem was embedded in a Markov Decision Process (MDP) and separate RL agents were trained on the different option types, both in the absence and presence of transaction costs. To evaluate the performance of the agents, the terminal portfolio values of 10,000 (almost surely) out-of-sample simulated stock price paths were compared with these of the delta hedging strategy.

The results showed that reinforcement learning can be used to effectively hedge a plain vanilla, a digital, an up-and-in (UAI), and a down-and-out (DAO) barrier call option under the Heston model. The RL agents picked up the dynamics of these options and learnt to replicate them. Both in the absence and presence of transaction costs, the agents were able to hedge these options more optimally than the delta hedging strategy in terms of the average (absolute) portfolio value and its standard deviation. The introduction of transaction costs led to more hedging errors in the case of the vanilla and DAO barrier call option, while it resulted in an improved hedging performance for the digital and UAI barrier call option. Except for the digital call option, the presence of transaction costs reduced the trading of arbitrary small amounts of the underlying stock: the agents' actions were smoother and expressed a delay compared to the delta hedge. This is because these moves are very costly and the risk aversion parameter, which is captured in the reward function, forces the agents to control the volatility of the period-by-period hedging costs (that include the transaction costs). For the digital call option, the introduction of transaction costs avoided large moves in the agent's position in the last time steps. Instead, the agent learnt to smooth out the actions and to build up a position throughout the digital option's lifetime. This seemed to avoid high replication errors in these last time steps and to improve the hedging performance.

Furthermore, the plain vanilla call option is robust against (and still outperformed the delta hedging strategy for) small changes in the underlying stock's strike K and the initial stock price S_0, and for a large range of initial instantaneous variance values ν_0. The RL agent trained on the UAI barrier call option was also robust against lower barriers and a slightly higher barrier than the one it had been trained on. In addition, the results showed that the RL agent trained on the vanilla call option was sensitive to some of the hyperparameters chosen to train the agent. First of all, the (speed of) convergence of the RL algorithm, as well as the hedging performance, is sensitive to the actor learning rate and the optimization technique. A nonzero discount factor resulted in a significantly lower portfolio value compared to a factor of zero, but the training of most option types destabilized when it exceeded the value of 0.9. Finally, the choice of the seeds used for the neural networks' weight initialization did not impact the overall conclusion regarding the RL agent's hedgeability. Since the same state representation and option valuation approach is used for all option types, similar results regarding the agent's robustness and flexibility are expected for the other option types that the agent learnt to hedge.

In this research setting, the performance of the RL agent was worse on the up-and-out (UAO) barrier call option and the agent did not learn how to hedge the down-and-in (DAI) barrier option. The RL systematically under- or over-hedges in the case of the former and learnt to take a full investment or no investment at all in the underlying stock of the latter. Next to the fact that the delta behaviour of these options is more complicated compared to the DAO and UAI option, this might be explained by the fact that these options have a payoff range that is more difficult to enable, which makes hedging harder.

9 Discussion

This section provides the main contribution, advantages, limitations, and potential improvements for further research in the field of option hedging using reinforcement learning (RL).

This study builds on earlier studies that investigated the application of RL to option hedging. These mainly focused on the plain vanilla call option. As their implementation details are generic, it is difficult to compare the experimental results. Despite this fact, these studies drew the same discussion as this research: the RL agent outperformed delta hedging for the plain vanilla call option in both a cost and a cost-free environment. To the best of my knowledge, this is the first study that investigated the application of RL to the hedging of a digital call option and four types of the barrier call option separately.

A major advantage of the RL approach is that an agent does not need any information about the strike price, the stock price process, the volatility process of the stock price, the option pricing formula, the payoff function, the option delta, and transaction cost function. The agent does not make any assumptions about these: it learns to hedge an option as good as possible by interacting with the environment. This means that RL can be used to hedge any option with a specific payoff function.[35] In this way, RL is way more flexible compared to mathematical models aimed at providing pricing formulas and hedging strategies. These models are often based on unrealistic assumptions, such as continuous trading, or do not provide a closed-form solution for all option types. Further research on different option types, such as options with a very complex payoff structure, may yield new insights and improve the practical utility. Another valuable extension would be to investigate American-style options[36] and options on stocks that pay dividends[37]. RL is also more flexible in the sense that additional constraints, e.g. related to the position in the underlying asset, can easily be incorporated by modifying the available actions.

A limitation of this research is its practicability and generality. The agents were trained on a specific hedging problem: on a specific option type and a specific environment with a fixed set of parameters. For example, although the Heston assumption is reflective of the real market, the RL agent might not be able to adapt properly to a new, possibly better stochastic process that models the stock price differently. Changes in the parameters/environment would probably require retraining the agent. The same holds for large changes in the option input parameters. Furthermore, the first attempt made to train an RL agent on multiple option types simultaneously did unfortunately not succeed. Further research on practicability and generality, both in terms of the option/environment conditions and the number of options that can be hedged, would be useful.
Another point regarding the practicability is that although RL is a promising approach in option hedging and risk management in general, it might be considered as being riskier than traditional models. Implementing an RL algorithm for option hedging involves the determination of e.g. what to include in the model inventory, how to measure the risks associated with the hedging strategy, the degree of risk aversion, and the model validation. As a result, an RL algorithm is often more complex compared to traditional models. More specialized skills are required to be able to implement the algorithm, understand the actions taken by the agent, and interpret the results. For example, many banks often operate in jurisdictions with stringent regulatory requirements [68]. As RL may introduce potential regulatory risk and model risk, traditional models are likely to be preferred. This might change if more regulatory instructions arise so that model-risk management can be enhanced, the added

[35] However, the results of the RL agent on the DAI and UAO option showed that RL does not always outperform delta hedging. Finding a good policy might require a more extensive hyperparameter search or a modified state.

[36] Options which can be exercised at any time before and including the expiration date.

[37] A dividend is a share of a company's profits distributed in a pro-rata distribution to the shareholders.

risk can be mitigated and banks can benefit more from the use of machine learning. RL could have an important role here.

This research can be extended and improved in a few other ways. First of all, it focused on delta hedging, as the RL agent could only trade in the underlying asset and a riskless asset. The hedging task can easily be changed by including other assets, such that the agent cannot only hedge delta exposure but also e.g. volatility exposure. As the option can then better be replicated, this may lead to an even better hedging performance of the option. Further investigations could lead to valuable insights. Besides, the performance of the RL agent in the presence of transaction costs was compared with that of delta hedging. This strategy does not take the transaction costs into account. Further research could consider a more advanced benchmark that accounts for this.

The accounting formulation used to define the hedging objective led to a stable learning. However, a disadvantage of this formulation is the requirement of a pricing model and a possible bias introduced by such. Further research could focus on the cash flow formulation, which does not require a pricing model and might in this way be more flexible. Another promising adjustment might be to use a pre-trained agent based on the accounting formulation and to adjust this model under the cash flow formulation (known as *transfer learning*). Another point regarding the agent's flexibility is the time between two trading points. This value was chosen to be fixed in this research. An extension of this research could be to make these intervals stochastic, for example by adding some noise. In this way, the agent may be better able to hedge options with different maturities and hence different values of the time to maturity than the one it was trained on. It might also be interesting to investigate the impact of restricting the agent to only trade if the stock reaches a specified price level or after the occurrence of a certain price change.
A related extension would be to train the agents on a finer discretization of time. The performance of the delta hedging strategy is expected to increase as the rebalancing frequency increases because the discretization error decreases. Nevertheless, Cao et al. [14] showed that the RL agent consistently outperformed delta hedging and that the difference between both performances increased as the frequency of hedging increased. It would be interesting to see if the same applies to call options with a discontinuity in the payoff function.

Another interesting line of research would be to train the RL agent on actual historical stock price data instead of simulated data. This would allow the agent to exploit the *model free* component of RL, as it will learn a hedging strategy independent of any simulated stock price process. In this way, the agent might learn patterns in the stock prices that are not captured by the stochastic process. However, as described earlier, training the RL agent requires a lot of data. A solution would be to use intraday option/stock data, but this can introduce a correlation between the samples. Another idea is to simulate the stock prices according to a stochastic process that is calibrated to the real market and to test the performance of the agent on market data.

To train the agent, a fixed discount factor of 0.9 was chosen in this research. To better reflect the hedging objective, further research can focus on obtaining a higher discount factor without causing instabilities. For this, different parameter values, decaying schemes, and scaling of the state features can be investigated. In addition, the DDPG algorithm was used to train the agent and a grid of hyperparameters was conducted to tune the algorithm. Further research could try an improved version of this algorithm and conduct a more extensive grid search. In addition, the performance of the algorithm could be improved by implementing recently proposed techniques, such as a prioritized replay buffer and batch normalization. The impact of different neural network architectures would also be interesting to investigate.

References

1. Albrecher, H., Mayer, P., Schoutens, W. & Tistaert, J. The little Heston trap. *Wilmott,* 83–92. ISSN: 1540-6962 (2007).
2. Alexander, C. & Nogueira, L. Stochastic Local Volatility. *Proceedings of the 2nd IASTED International Conference, MIT, Cambridge, MA* (2008).
3. Alziary, B. & Takáč, P. On the Heston Model with Stochastic Volatility: Analytic Solutions and Complete Markets. *Electronic Journal of Differential Equations* **168** (2017).
4. Barth-Maron, G. *et al.* Distributed Distributional Deterministic Policy Gradients. *CoRR* **abs/1804.08617.** arXiv: 1804.08617 (2018).
5. Bates, D. S. Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. *The Review of Financial Studies* **9,** 69–107. ISSN: 08939454, 14657368 (1996).
6. Bellman, R. *Dynamic Programming* 1st ed. (Princeton University Press, Princeton, NJ, USA, 1957).
7. Benhamou, E. Efficient Computation of Greeks for Discontinuous Payoffs by Transformation of the Payoff Function. *The Journal of Computational Finance* (2004).
8. Bijma Fetsje, e. a. *An Introduction to Mathematical Statistics* (Amsterdam University Press, 2017).
9. Black, F. & Scholes, M. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy* **81,** 637–54 (1973).
10. Bolia, N. & Juneja, S. Monte Carlo methods for pricing financial options. *Sadhana* **30,** 347–385 (2005).
11. Broadie, M. & Kaya, Ö. Exact Simulation of Stochastic Volatility and Other Affine Jump Diffusion Processes. *Operations Research* **54,** 217–231 (2006).
12. Buehler, H. *et al.* Deep Hedging: Hedging Derivatives Under Generic Market Frictions Using Reinforcement Learning. *SSRN Electronic Journal* (2019).
13. Bühler, H., Gonon, L., Teichmann, J. & Wood, B. *Deep Hedging* 2018. arXiv: 1802.03042.
14. Cao, J., Chen, J., Hull, J. C. & Poulos, Z. Deep Hedging of Derivatives Using Reinforcement Learning. *Risk Management Analysis in Financial Institutions eJournal* (2019).
15. Chen, B. Calibration of the Heston Model with Application in Derivative Pricing and Hedging (2007).
16. Chou, P.-W., Maturana, D. & Scherer, S. A. *Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution* in *ICML* (eds Precup, D. & Teh, Y. W.) **70** (PMLR, 2017), 834–843.
17. Crisostomo, R. An Analysis of the Heston Stochastic Volatility Model: Implementation and Calibration Using Matlab. *SSRN Electronic Journal* (2015).
18. Cristian, P., Musetescu, R., Brasoveanu, I. & Draghici, A. Empirical evidence on risk aversion for individual romanian capital market investors. *Review of Economic and Business Studies* **1,** 91–101 (2008).
19. Das, K. Introduction to stochastic process (2018).
20. Degris, T., Pilarski, P. M. & Sutton, R. S. *Model-Free reinforcement learning with continuous action in practice* in *American Control Conference (ACC)* (2012), 2177–2182.

21. Dragulescu, A. & Yakovenko, V. Probability Distribution of Returns in the Heston Model with Stochastic Volatility. *Quantitative Finance* **2,** 443–453 (2002).

22. Durrett, R. *Stochastic Calculus: A Practical Introduction* (Crc Press, 1996).

23. Ekenel, H. K. & Stiefelhagen, R. Analysis of Local Appearance-Based Face Recognition: Effects of Feature Selection and Feature Normalization. *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06),* 34–34 (2006).

24. Etheridge, A. *A Course in Financial Calculus* (Cambridge University Press, 2002).

25. Euler, L. *Institutiones Calculi Integralis* 110–113 (B. G. Teubner, Leipzig-Berlin, 1768).

26. Fournié, E., Lasry, J.-M., Lions, P.-L. & Lebuchoux, J. Applications of Malliavin calculus to Monte-Carlo methods in finance. II. *Finance and Stochastics* **5,** 201–236 (2001).

27. François-Lavet, V., Fonteneau, R. & Ernst, D. How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies (2015).

28. Franke, J., Härdle, W. K. & Hafner, C. in *Statistics of Financial Markets* 145–157 (Springer, Berlin, 2019). ISBN: 978-3-030-13750-2.

29. Fujimoto, S., van Hoof, H. & Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *CoRR* **abs/1802.09477.** arXiv: 1802.09477 (2018).

30. Garven, J. Derivation and Comparative Statics of the Black-Scholes Call and Put Option Pricing Formula (2009).

31. Glynn, P. W. *Optimization of Stochastic Systems via Simulation* in *Proceedings of the 21st Conference on Winter Simulation* (Association for Computing Machinery, Washington, D.C., USA, 1989), 90–105. ISBN: 0911801588.

32. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* ISBN: 0262035618 (The MIT Press, 2016).

33. Gracianti, G. Computing Greeks by Finite Difference using Monte Carlo Simulation and Variance Reduction Techniques. *Berkala MIPA* **25** (2012).

34. Grefenstette, J., Moriarty, D. & Schultz, A. Evolutionary Algorithms for Reinforcement Learning. *Journal of Artificial Intelligence Research* **11** (2011).

35. Grondman, I., Busoniu, L., Lopes, G. A. D. & Babuska, R. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42,** 1291–1307 (2012).

36. Halperin, I. QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *SSRN Electronic Journal* (2017).

37. Heston, S. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *Review of Financial Studies* **6,** 327–343 (1993).

38. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* **abs/1207.0580.** arXiv: 1207.0580 (2012).

39. Hudson, R. & Gregoriou, A. Calculating and Comparing Security Returns is Harder than you Think: A Comparison between Logarithmic and Simple Returns. *International Review of Financial Analysis* **38** (2010).

40. Hull, J. *Options, futures, and other derivatives* 6. ed., Pearson internat. ed. XXII, 789. ISBN: 978-0-13-197705-1 (Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 2006).

41. Jain, R. *Option Pricing and Hedging with Deep Learning* (2019).

42. Jarrow, R., Patie, P., Srapionyan, A. & Zhao, Y. Risk-neutral pricing techniques and examples. *Mathematical Finance* (2018).

43. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* **4,** 237–285 (1996).

44. Kakushadze, Z. Phynance. *Universal Journal of Physics and Application* **9,** 64–133. ISSN: 2331-6543 (2015).

45. Keith Cuthbertson, D. N. & O'Sullivan, N. *Derivatives: Theory and Practice* (John Wiley Sons Ltd., 2019).

46. Kroese, D., Taimre, T. & Botev, Z. Handbook of Monte Carlo Methods (2011).

47. Kumaraswamy, R., Schlegel, M., White, A. & White, M. *Context-Dependent Upper-Confidence Bounds for Directed Exploration* in *NeurIPS* (2018).

48. Lazăr, V. Pricing Digital Call Option in the Heston Stochastic Volatility Model. *Studia Universitatis Babeş-Bolyai. Mathematica* **48** (2003).

49. Lillicrap, T. *et al.* Continuous control with deep reinforcement learning. *CoRR* (2015).

50. Lin, L.-J. *Reinforcement Learning for Robots Using Neural Networks* PhD thesis (USA, 1992).

51. Liu, L. *et al.* On the Variance of the Adaptive Learning Rate and Beyond. *CoRR* **abs/1908.03265.** arXiv: 1908.03265 (2019).

52. Lowe, R. *et al.* Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *CoRR* **abs/1706.02275.** arXiv: 1706.02275 (2017).

53. Madhyastha, P. & Jain, R. On Model Stability as a Function of Random Seed. *CoRR* **abs/1909.10447.** arXiv: 1909.10447 (2019).

54. Maruyama, G. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo* **4,** 48–90 (1955).

55. Mishura, Y. *Financial Mathematics* 83–121 (ISTE Press - Elsevier, 2016).

56. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning (2013).

57. Moody, J. & Saffell, M. *Reinforcement Learning for Trading* in *NIPS* (1998).

58. Muteba, F., Djouani, K. & Olwal, T. Deep Reinforcement Learning Based Resource Allocation For Narrowband Cognitive Radio-IoT Systems. *Procedia Computer Science* **175,** 315–324 (2020).

59. Nakkiran, P. Learning Rate Annealing Can Provably Help Generalization, Even for Convex Problems. *ArXiv* **abs/2005.07360** (2020).

60. Peeters, B., Dert, C. & Lucas, A. *Black Scholes for portfolios of options in discrete time: the price is right, the hedge is wrong* WorkingPaper TI 2003 90/2 (Tinbergen Instituut (TI), 2003).

61. Privault, N. *Stochastic Finance: An Introduction with Market Examples* 1–421. ISBN: 9780429168796 (2013).

62. Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).

63. Ramachandran, P., Zoph, B. & Le, Q. V. Searching for Activation Functions. *CoRR* **abs/1710.05941.** arXiv: 1710.05941 (2017).

64. Ritter, G. Machine Learning for Trading. *Microeconomics: General Equilibrium Disequilibrium Models of Financial Markets eJournal* (2017).

65. Ritter, G. & Kolm, P. Dynamic Replication and Hedging: A Reinforcement Learning Approach. *SSRN Electronic Journal* (2018).

66. Röman, J. R. M. *Analytical Finance: Volume I. The Mathematics of Equity Derivatives, Markets, Risk and Valuation* (Palgrave Macmillan, 2017).

67. Ryu, M., Chow, Y., Anderson, R., Tjandraatmadja, C. & Boutilier, C. CAQL: Continuous Action Q-Learning. *CoRR* **abs/1909.12397.** arXiv: 1909.12397 (2019).

68. Saunders, A. & Cornett, M. M. *Financial institutions management: a risk management approach* (McGraw-Hill, 2008).

69. Sevcovic, D., Stehlíková, B. & Mikula, K. Analytical and numerical methods for pricing financial derivatives. *Analytical and Numerical Methods for Pricing Financial Derivatives,* 1–309 (2011).

70. Shapiro, A. C. *Multinational financial management* (New York: Wiley, 2010).

71. Silver, D. *et al.* Deterministic Policy Gradient Algorithms. *31st International Conference on Machine Learning, ICML 2014* **1** (2014).

72. Staum, J. Chapter 12 Incomplete Markets. *Handbooks in Operations Research and Management Science* **15** (2007).

73. Stojkoski, V., Sandev, T., Basnarkov, L., Kocarev, L. & Metzler, R. Generalised geometric Brownian motion: Theory and applications to option pricing (2020).

74. Sutton, R., Mcallester, D., Singh, S. & Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Adv. Neural Inf. Process. Syst* **12** (2000).

75. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

76. Sutton, R. S. Learning to Predict By the Methods of Temporal Differences. *Machine Learning* **3,** 9–44 (1988).

77. Tartaglia, E. M., Clarke, A. & Herzog, M. What to Choose Next? A Paradigm for Testing Human Sequential Decision Making. *Frontiers in Psychology* **8** (2017).

78. Tichý, T. Model Dependency of the Digital Option Replication – Replication under an Incomplete Model (in English). *Czech Journal of Economics and Finance (Finance a uver)* **56,** 361–379 (2006).

79. Tsumurai, S. Malliavin Differentiability of CEV-Type Heston Model. *Journal of Mathematical Finance* **10,** 173–199 (2020).

80. Vittori, E., Trapletti, M. & Restelli, M. Option Hedging with Risk Averse Reinforcement Learning. *CGN: Risk Management* (2020).

81. Wang, H., Zhang, P. & Liu, Q. An Actor-critic Algorithm Using Cross Evaluation of Value Functions. *IAES International Journal of Robotics and Automation (IJRA)* **7,** 39 (2018).

82. Watkins, C. J. C. H. & Dayan, P. Q-learning. *Machine Learning* **8,** 279–292. ISSN: 1573-0565 (1992).

83. Westermark, N. Barrier Option Pricing Degree Project in Mathematics, First Level (2009).

84. Wiering, M. & Van Otterlo, M. *Reinforcement Learning: State of the Art* ISBN: 978-3-642-27644-6 (Springer, 2012).

85. Willard, G. A. Calculating Prices and Sensitivities for Path-Independent Derivatives Securities in Multifactor Models. *The Journal of Derivatives* **5** (1997).

86. Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* **8,** 229–256 (1992).

87. Yang, L.-R., Su, M.-C. & Lin, J.-Z. A Rule Extraction Based Approach in Predicting Derivative Use for Financial Risk Hedging in Construction Companies. *Expert Syst. Appl.* **37,** 6510–6514 (2010).

88. Yilmaz, B. Computation of option greeks under hybrid stochastic volatility models via Malliavin calculus. *Modern Stochastics: Theory and Applications* **5,** 1–21 (2018).

Appendices

The following appendices provide some additional visualisations and explanations on the performance of two reinforcement learning (RL) agents that had been trained on respectively an up-and-out barrier and a down-and-in barrier. In addition, for the RL agent trained on a vanilla call option, an additional analysis is included on the impact of a modified strike and a different discount factor.

A Performance RL agent on the up-and-out-barrier call option

Figure A.1 shows a single simulated episode of the trained RL agent on the up-and-out (UAO) barrier call option. It follows that the agent picked up some of the option (delta) spikes. This is also visible in the corresponding value function, which is presented in Figure A.2 and shows distinct contour levels through the option's lifetime. The value function also reflects the particularity of the UAO option: if the stock price approaches the barrier around the 42^{th} time step, the delta as well as the action taken by the agent drops. In addition, the value function is less distinct and shows lower Q-values around this point in time. Although the agent picked up some of the option dynamics, the agent does not properly hedge the option. It is visible that the agent under-hedges the option. In addition, it takes short positions in the underlying stock (negative actions) while a long position is to be expected (e.g. before time step 30).

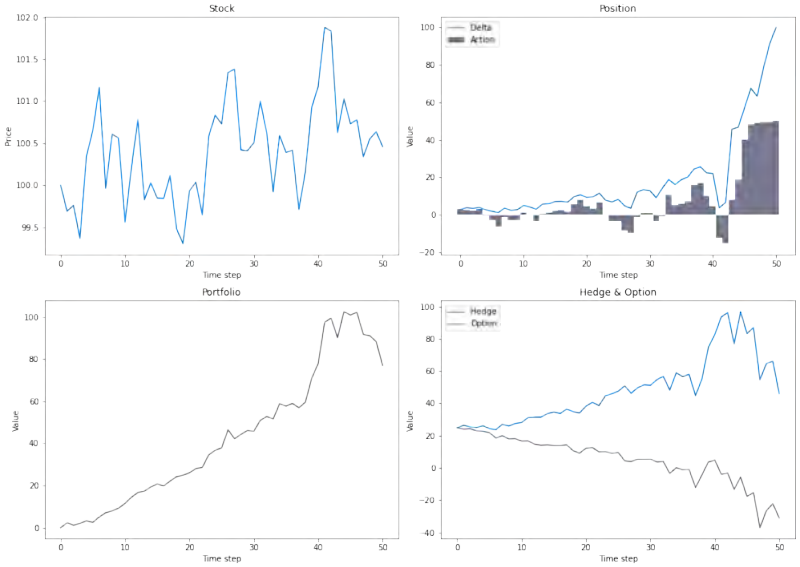


Figure A.1: A single episode of an up-and-out (UAO) barrier call option.

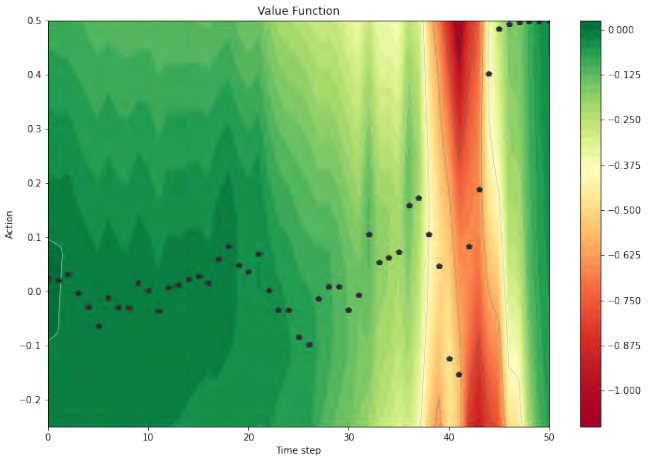


Figure A.2: Corresponding value function of the single simulated episode per one share of the up-and-out (UAO) barrier call option's underlying stock.

B Performance RL agent on the down-and-in-barrier call option

Figure B.1 shows a single simulated episode of the trained RL agent on the down-and-in (DAI) barrier call option. It can be observed that the agent did not learn how to properly hedge this option. The agent does not invest in the underlying stock, except for the position at initiation (due to the self-financing condition). As a result, the hedge value does not track the option value throughout the option's lifetime. The inability of the agent to hedge the option is reflected in the corresponding value function, which is shown in Figure B.2. As the option goes ITM, one would expect a green area of actions with the highest values that converges towards the top of the value function. However, no distinct contour levels can be observed through the option's lifetime. The estimated Q-values are inaccurate, which results in poor performance of the agent.

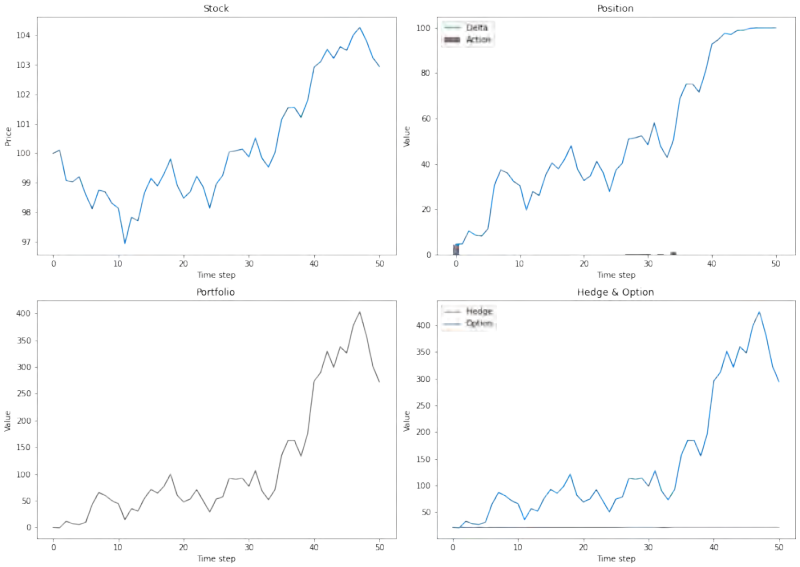


Figure B.1: A single episode of a down-and-in (DAI) barrier call option.

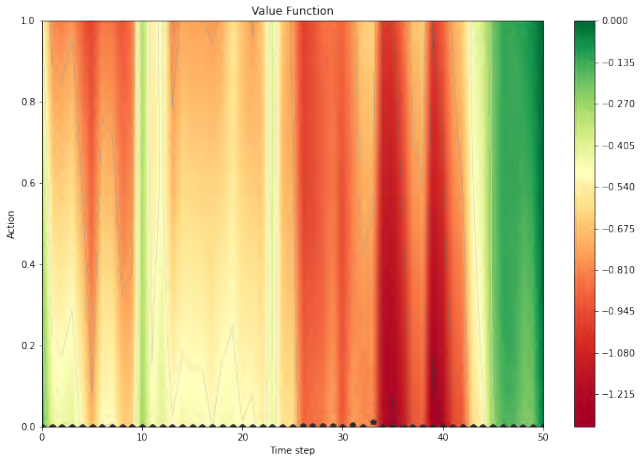
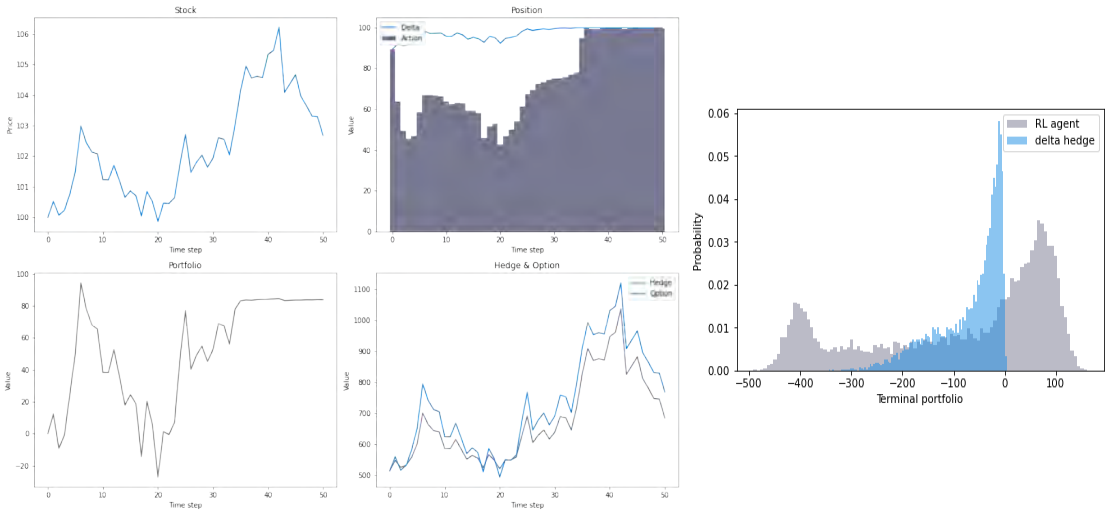


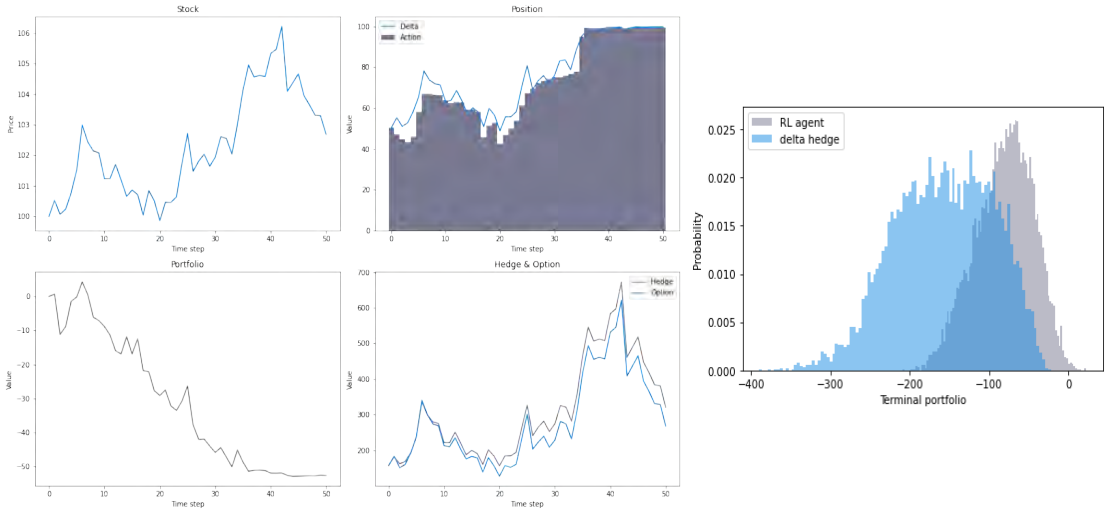
Figure B.2: Corresponding value function of the single simulated episode per one share of the down-and-in (DAI) barrier call option's underlying stock.

C RL agent tested on vanilla call options with a modified strike

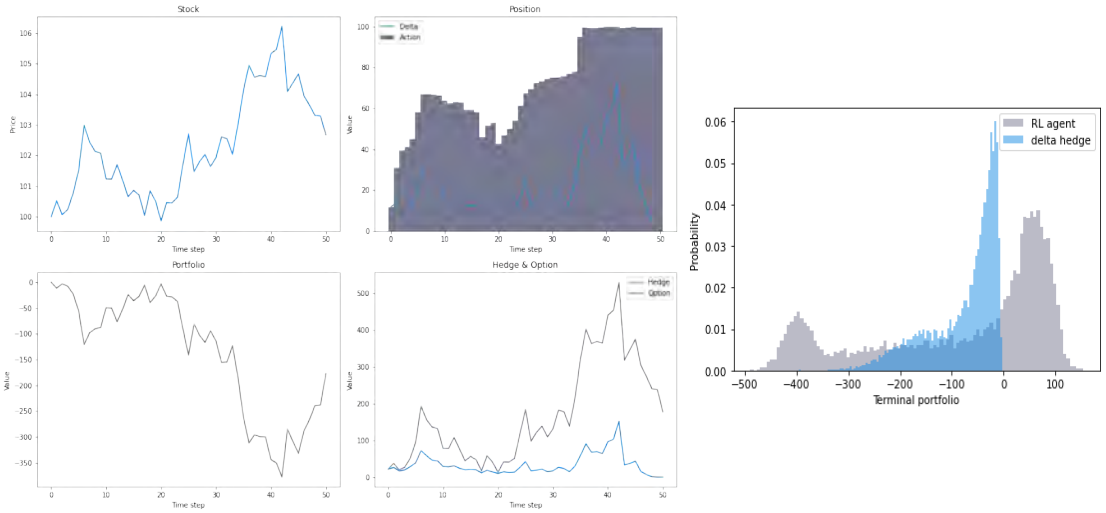
Figure C.1 visualises the impact of a modified strike price on the RL agent's hedge performance for a vanilla call option with a strike price of $K = 95$, $K = 100$ and $K = 105$. For these strikes, it shows the delta behaviour and actions taken by the agent for a particular episode, as well as the distribution of the portfolio value based on 10,000 simulated episodes. It can be observed that apart from the first time steps, the actions taken by the agent are similar for all three call options. The former can be explained by the fact that the position in the underlying stock is set equal to the option value at initiation due to the self-financing property. The call option with $K = 95/K = 105$ starts and ends ITM/OTM. As a result, the delta is higher/lower than for the call option with $K = 100$. Since the agent employs a similar strategy in this episode for all three strikes, it respectively under-hedges/over-hedges in the case of $K = 95/K = 105$. The other way around also holds: the RL agent over-hedges/under-hedges a call option with $K = 95/K = 105$ that starts and ends OTM/ITM. As a result, the agents' average portfolio values, shown in Table 11, are similar. However, the average absolute portfolio value is significantly higher for the vanilla call options with a modified strike. In addition, the distribution of the strike on which the agent had been trained is narrower than these of the modified strikes. The fact that the agent employs a similar strategy in a particular episode for different strikes reflects the fact that the RL agent is not robust against large changes in the strike price. This is also visible in the value function, which is similar for the three tested call options.



(a) $K = 95$



(b) $K = 100$

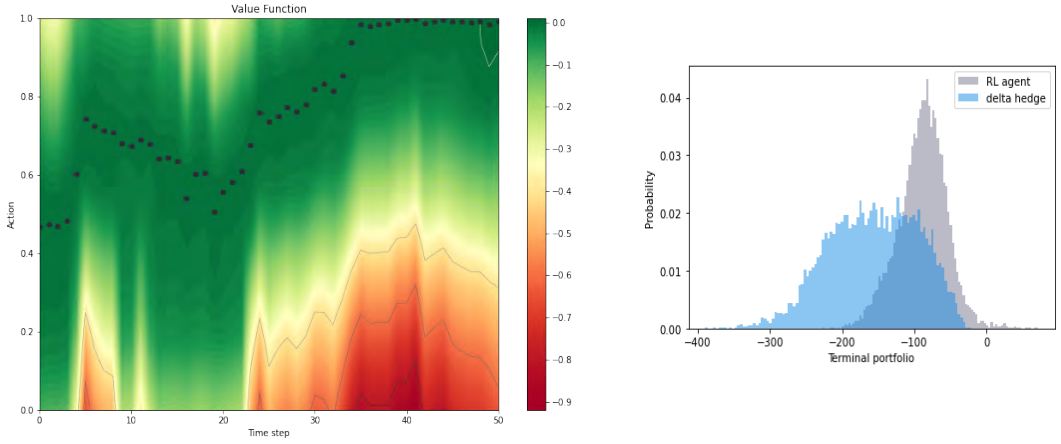


(c) $K = 105$

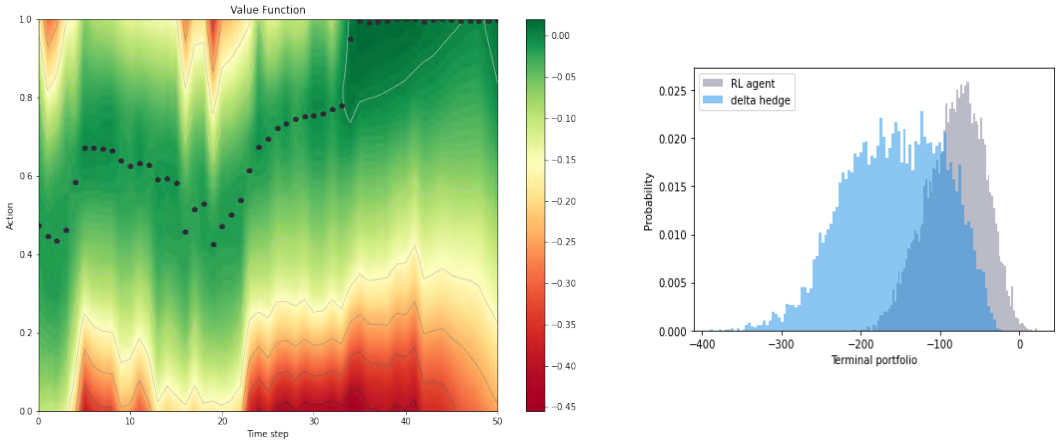
Figure C.1: A single episode (left) and distribution of the portfolio value (right) for a vanilla call option with strike (a) $K = 95$, (b) $K = 100$ and (c) $K = 105$.

D RL agent trained on a vanilla call option using different discount factors

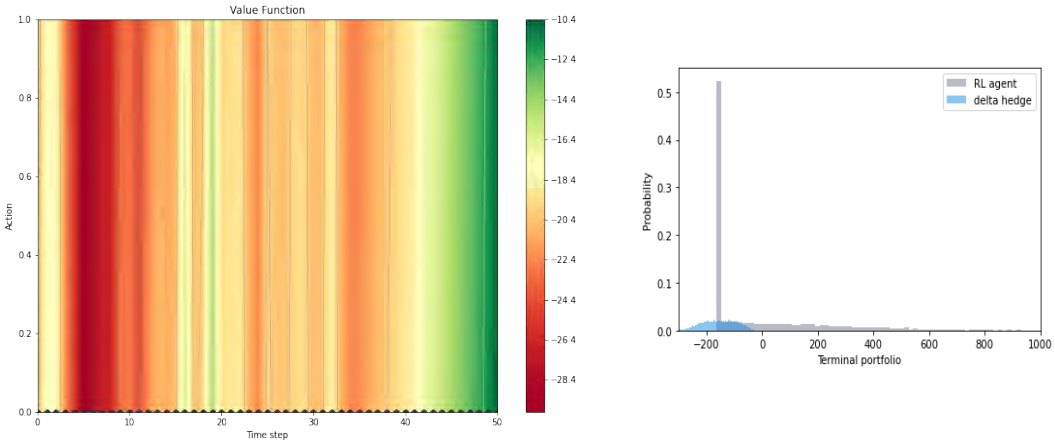
Figure D.1 visualises the impact of the discount factor used to train the RL agent on the agent's performance for a vanilla call option. Contrary to Appendix C, a separate agent is trained on each of the tested discount factors. Hence, the agents' value functions are different. It can be observed that the results of the agents trained on a discount factor of $\gamma = 0$ and $\gamma = 0.9$ are similar: both value functions are accurate and show similar optimal actions (represented by the black dots), and the portfolio distributions have a similar shape. Note that the estimated Q-values have a different colour scaling, as the future rewards are not taken into account in case of $\gamma = 0$. For $\gamma = 0.999$, the agent did not learn how to hedge the call option properly. Instead, the RL agent incorrectly learnt to rarely take a position in the underlying stock. This is visible in the value function, which is inaccurate and in which the optimal action is achieved at an action of zero for the entire episode. This results in extreme gains and losses, which is reflected in the wide distribution of the agent's portfolio value.



(a) $\gamma = 0$



(b) $\gamma = 0.9$



(c) $\gamma = 0.999$

Figure D.1: The value function of a single episode (left) and distribution of the portfolio value (right) for three RL agents trained on a vanilla call option and a discount factor of (a) $\gamma = 0$, (b) $\gamma = 0.9$ and (c) $\gamma = 0.999$.