mathematics

MDPI

Review

Deep Reinforcement Learning for Dynamic Stock Option Hedging: A Review

Reilly Pickard [1,*] and Yuri Lawryshyn [2]

[1] Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada
[2] Department of Chemical Engineering and Applied Chemistry, University of Toronto, Toronto, ON M5S 3E5, Canada; yuri.lawryshyn@utoronto.ca
* Correspondence: reilly.pickard@mail.utoronto.ca

Abstract: This paper reviews 17 studies addressing dynamic option hedging in frictional markets through Deep Reinforcement Learning (DRL). Specifically, this work analyzes the DRL models, state and action spaces, reward formulations, data generation processes and results for each study. It is found that policy methods such as DDPG are more commonly employed due to their suitability for continuous action spaces. Despite diverse state space definitions, a lack of consensus exists on variable inclusion, prompting a call for thorough sensitivity analyses. Mean-variance metrics prevail in reward formulations, with episodic return, VaR and CvaR also yielding comparable results. Geometric Brownian motion is the primary data generation process, supplemented by stochastic volatility models like SABR (stochastic alpha, beta, rho) and the Heston model. RL agents, particularly those monitoring transaction costs, consistently outperform the Black–Scholes Delta method in frictional environments. Although consistent results emerge under constant and stochastic volatility scenarios, variations arise when employing real data. The lack of a standardized testing dataset or universal benchmark in the RL hedging space makes it difficult to compare results across different studies. A recommended future direction for this work is an implementation of DRL for hedging American options and an investigation of how DRL performs compared to other numerical American option hedging methods.

Keywords: reinforcement learning; neural networks; dynamic stock option hedging; quantitative finance; financial risk management

MSC: 91G20

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

Academic Editors: Xinwei Cao, Tran Thu Ha, Dunhui Xiao, Vasilios N. Katsikis, Ameer Hamza Khan and Shuai Li

Received: 28 November 2023
Revised: 8 December 2023
Accepted: 11 December 2023
Published: 13 December 2023

1. Introduction

In recent years, artificial intelligence (AI) and machine learning (ML) methodologies have garnered significant attention within the financial sector. Among these AI and ML techniques, reinforcement learning (RL) has emerged as a robust paradigm with the potential to enhance intricate decision-making processes in volatile and uncertain financial landscapes. Notably, RL has demonstrated substantial promise in the domain of financial option hedging [1]. Moreover, the combination of RL and neural networks (NNs) has proven to be effective in RL tasks in complex environments [2]. The combination of RL and NNs is called deep RL (DRL).

Traditional financial option hedging makes use of the Black–Scholes (BS) [3] option pricing model [4]. The BS model enables one to calculate the price of both call and put options as a function of the current underlying asset price, the strike price, the volatility, the risk-free rate and the time to expiry [3]. Option hedgers give specific interest to the BS Delta (Δ) parameter [1]. The BS Delta is the first partial derivative of the BS option price with respect to the underlying asset price [4]. Using the BS model, the risk associated with a financial option contract can be hedged effectively by holding Delta shares in the underlying asset [4]. This hedge position is financed through a risk-free bank account,

and the Delta hedge and bank account positions form a portfolio that perfectly replicates the option position [4]. As the BS model is derived in continuous time, all risks can be mitigated by continuously rebalancing the replication portfolio, and all losses (gains) of the option are offset by the gains (losses) of the synthetic portfolio [3]. The BS Delta is one of the BS Greeks, and other BS Greeks of interest to hedgers are the BS Gamma (first partial derivative of the BS Delta with respect to the underlying asset price) and the BS Vega (first partial derivative of the option price with respect to the underlying volatility) [4].

Although the continuous replication of option positions offers a strong theoretical foundation, several assumptions made in the BS Delta method fail to align with realistic financial markets. The first limitation is the assumption of continuous rebalancing, as real-world markets operate in a discrete fashion [5]. Discrete hedging requires maintaining the Delta position from one rebalance point to the next, resulting in a potential divergence between the hedge portfolio's value and the option value [5]. A second limitation in the BS Delta model is the omission of transaction costs. In the presence of transaction costs, frequent rebalancing through the purchasing of the underlying asset might incur higher losses than an unhedged position [6]. Moreover, the instantaneous market impact induced by the buying and selling of the underlying asset is not considered in the BS Delta method [7]. Additionally, the BS model is derived by letting the underlying asset price process evolve as a geometric Brownian motion (GBM) with a constant volatility assumption [3], and this asset price evolution may not be realistic for real markets [8].

With market frictions considered, the development of an effective dynamic hedging strategy is a progressive decision-making procedure in an uncertain environment. Consequently, DRL offers a potential solution. This paper presents a comprehensive review of the existing literature concerning the utilization of DRL to optimize dynamic stock option hedging. The paper proceeds as follows: Section 2 introduces RL and DRL, Section 3 discusses similar review work in the RL and financial RL fields, and Section 4 outlines the methodology used to select relevant articles and gives an overview of how the articles are analyzed. Section 5 analyzes the selected papers, with specific interest being given to the choice of the RL algorithm, the selection of state and action spaces, the choice of reward formulations, the data generation processes and the key results of these works.

2. Reinforcement Learning

2.1. Fundamentals

To conceptually introduce RL, it is helpful to first consider the multi-arm bandit (MAB) problem, just as is done in Sutton's and Barto's introductory RL textbook [9]. The multi-arm bandit problem considers an agent in front of a multi-lever slot machine. Pulling the levers leads to stochastic rewards, and the goal of the agent is to maximize the cumulative reward over a set number of lever pulls. Each lever pull may be modeled as a discrete time step. The agent must therefore learn to balance exploitation, by pulling a lever that has returned large rewards, and exploration, by pulling other levers and assessing their potential returns [9]. Although in this scenario the information provided to the agent at each time step is identical, a contextualized MAB (CMAB) problem provides the agent with some form of information before each lever pull [9].

CMAB sets the stage for RL, wherein an agent first observes the current state of the environment, denoted as s and belonging to the environment's state space $\mathcal{S}$, before selecting an action a from the set of permissible actions $\mathcal{A}$ [9]. After each action, a reward $r \, \epsilon \mathcal{R} \epsilon \mathbb{R}$ is observed, where $\mathcal{R}$ is the set of possible rewards. The premise of RL is the maximization of future expected rewards, G, also called the return, which, at a given time t for a finite problem of T time steps, is summarized in [9] as

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots + \gamma^{T-1} r_T, \tag{1}$$

The discount factor γ accounts for the uncertainty of the rewards at future time steps [9]. The discount factor may also be used to reflect the time value of money for a financial problem if r_t represents the cash flow or profit at time t [1]. The action a taken by

the agent in state s is determined by the agent's policy, π. The policy may be deterministic, mapping a state directly to an action, $\pi : s \rightarrow a$, or stochastic, where for each state the resulting action is sampled from a probability distribution, $\pi : s \rightarrow \mathbb{P}_\pi(a)$ [10]. The optimal policy is denoted as π^* and leads to a maximum return. The value obtained by taking an action a in state s and thereafter following a policy π is quantified by the Q-value [9], defined as

$$Q^\pi(s,a) = \mathbb{E}[G|s,a] \in \mathbb{R}. \tag{2}$$

The Q-value informs the agent how optimal it is to take action a in state s and thereafter follow a policy π, where optimality is defined in terms of the expected return from that state–action pair. Note that the Q-function is the collection of Q-values for each state–action pair [9]. Therefore, an RL agent acquires a map of optimal actions to take for each state by learning the Q-function stemming from the optimal policy, $Q^{\pi^*}(s,a)$ [9]. The Q-value for state–action pairs can be updated in an off- or on-line manner. An off-line method, such as the first-visit Monte-Carlo (MC) method, updates the current Q-value as

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G - Q(s,a)). \tag{3}$$

In first-visit MC, G is the cumulative reward following the first occurrence of the pair (s,a) in a simulated episode, and α is an adjustable learning rate hyperparameter [9]. MC methods require the completion of an episode to update the Q-value, which may be computationally intensive and lead to slowed learning [9]. To avoid waiting for episode termination to update the value function, an on-line learning method may be used. An example of an on-line method is temporal difference (TD) learning, which updates the Q-value after each action. The update rule for TD learning is

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r' + \gamma Q(s',a') - Q(s,a)), \tag{4}$$

where r' is the reward from taking a in state s, and $Q(s',a')$ is the Q-value of the next state–action pair [9]. Using the next state, next action and reward to update the Q-value in this on-line manner is the SARSA (s,a,r',s',a') algorithm, an on-policy TD learning method [9]. In on-policy methods, the agent utilizes data from experiences generated by the current estimation of the optimal policy to learn the Q-function. Conversely, off-policy methods involve using a sub-optimal policy to generate experiences for updating $Q^{\pi^*}(s,a)$ [11].

SARSA is an example of a value-based RL method, as the aim of SARSA is to find the optimal Q-function by directly evaluating which state–action pairs are more promising in terms of expected cumulative future rewards [9]. Value methods such as SARSA do not explicitly consider the optimal policy [12]. Conversely, policy-based methods keep the current estimate of the optimal policy in memory during the learning process, and updates are performed on this optimal policy estimate rather than the value function [12]. Using a value method rather than the policy method may impose the constraint of discrete action spaces. The decision between continuous and discrete actions are crucial in dynamic hedging problems, i.e., buying a continuous number of shares versus limiting the RL agent to buy some discrete number of shares. As such, the relevant RL methods described in this paper are classified into value- and policy-based algorithms. This classification method is consistent with classification techniques in the RL literature [13]. Although several RL methods are available for use, the methods described in this review are included due to their use in dynamic hedging problems.

2.2. Value-Based RL Methods

The SARSA method offers a straightforward way to grasp how state value functions are learned. However, SARSA is an on-policy method, and on-policy methods have biases that hinder the exploration process [9]. This bias stems from using the current optimal policy to select the next action, resulting in under-exploration of the environment, even if there is some built-in probability ε that the next action is random as opposed to greedy [9].

A breakthrough result by Watkins in 1989 [14] introduced Q-learning, an off-policy value method that allows $Q^{\pi^*}(s,a)$ to be updated while following a suboptimal policy. Q-learning allows for increased efficiency and has been found to speed up the convergence to the optimal value function [9]. Q-learning updates the current Q-value as

$$Q(s,a) = Q(s,a) + \alpha\big(r' + \gamma max_a Q\big(s',a'\big) - Q(s,a)\big), \tag{5}$$

noting that both SARSA and Q-learning are tabular methods, which, in RL, refers to storing the Q-function in a lookup table [9]. Tabular methods are intractable for high-dimensional problems, as storing values in a lookup table of size $\mathcal{S} \times \mathcal{A}$ can become computationally expensive and impractical as the dimensions of S and A increase. An approximation of the Q-function eliminates the need to store $\mathcal{S} \times \mathcal{A}$ Q-values. To eliminate the need for a lookup table, an NN may be used to approximate the Q-function. Recall that the field concerning the use of an NN to aid the computation of an RL solution is called DRL. DRL facilitates the effective handling of high-dimensional and continuous state spaces. Tesauro [15] was the first to incorporate NNs for RL in 1994, using an NN to approximate the optimal strategy for the game of backgammon. Despite this success, DRL was not popularized until the 2013 work of Mnih et al. [2], who used an NN to approximate the optimal Q-function. The approximation is denoted as $Q(s,a;\,\theta)$, where θ signifies the synaptic weights connecting various nodes within the neural network, referred to as a Q-network [2]. Ref. [2] shows that this method of deep Q-networks (DQN) can be used to train an agent to master multiple Atari games. The Q-network is trained by minimizing the loss function, which, for a deterministic policy and iteration i, is given by

$$L_i(\theta_i) = \mathbb{E}_{s,a}\Big[\big(y_i - Q(s,a;\,\theta_i)\big)^2\Big], \tag{6}$$

where $y_i = \mathbb{E}_{s,a}\,[r' + \gamma max Q(s',a';\,\theta_{i-1})\,]$ is the target Q-value for iteration i [2]. The network weights are optimized using stochastic gradient descent (SGD) [2]. A thorough review of gradient descent techniques to optimize NNs is given in [16], who describes SGD as the differentiation of the loss function with respect to the network weights to find $\nabla_{\theta_i} L_i(\theta_i)$, where ∇_{θ_i} refers to the gradient being taken with respect to the network weights. The updated weights are given by

$$\theta_i = \theta_{i-1} - \eta \nabla_{\theta_i} L_i(\theta_i) \tag{7}$$

where η is the learning rate for the descent algorithm [16].

In addition to the Q-network approximation, another major component of DQN is the use of the experience replay buffer [2]. Experience replay, originally proposed for RL by Lin [17] in 1992, is a technique that involves the storage of encountered transitions (state, action and reward) in a memory buffer. These stored transitions are then uniformly sampled during each training step [17]. Ref. [18] provides a thorough explanation of how experience replay is used in RL. The use of experience replay significantly enhances the algorithm's sample efficiency by allowing data to be reused for multiple training iterations, rather than discarding them right after collection [18]. Additionally, experience replay enhances the network's stability throughout the training process [18]. Ref. [2] popularized the use of experience replay in RL, using the replay buffer to uniformly sample the target Q-value for iteration i (y_i). During the inner loop of the Q-value iteration, mini-batch updates are applied by sampling data from the replay buffer [2].

An extension to DQN called distributed deep Q-networks (distributed DQN) may be made by representing the action value function as a distribution of possible cumulative returns [19]. By allowing $Z^\pi(s,a)$ to be a mapping of state–action pairs to return distributions, Ref. [19] showed that $Z^\pi(s,a)$ may be updated similarly to $Q^\pi(s,a)$.

2.3. Policy-Based RL Methods

To update the Q-function in value methods such as DQN, a max operation on the Q-value over all next possible actions is required. When the action space is continuous, the valuation of each potential next action becomes intractable. To handle problems with continuous action spaces, ref. [20] introduced a policy-based algorithm, deep deterministic policy gradient (DDPG), that utilizes an actor–critic architecture. The critic evaluates the current Q-value, and the differentiation of $Q^{\pi}(s, a;\ \theta)$ with respect to the actions allows for convergence to the optimal solution $Q^{\pi^*}(s, a;\ \theta)$ [20]. For every action produced by the actor network, a Q-value is computed by the critic network, which is trained by SGD [20]. This Q-value output from the critic is then passed back to the actor network, and the process repeats. The actor network converges to a deterministic policy by performing gradient ascent on the current Q-value with respect to the network weights [20].

Although DDPG uses SGD to optimize the critic, the key change in the critic network from DQN is in the computation of the target y_i, which is needed to compute $L_i(\theta_i)$. In DDPG, the current policy estimate produced by the actor's policy output π_i is used in the calculation of y_i instead of the max operation used in DQN. For DDPG, the target is $y_i = \mathbb{E}_{s,a}\left[r' + \gamma Q(s', \pi_i(s');\ \theta_{i-1})\right]$ [20]. Eliminating the requirement to maximize the Q-value by considering all possible actions allows for continuous action spaces to be considered. Note that the work of [20] is an extension of the findings of [21], which introduced the key equations for policy gradient algorithms.

Other policy methods found in the analyzed literature are proximal policy optimization (PPO) and trust region policy optimization (TRPO). PPO is an on-policy algorithm that attempts to learn the optimal policy directly by using a surrogate objective [22]. The surrogate objective is generally the clipped ratio of the new policy to the old policy [22]. The ratio is given by

$$ratio(\theta) = \frac{\pi(s, a;\ \theta)}{\left(\pi^{old}(s, a;\ \theta)\right)}, \tag{8}$$

and the PPO algorithm aims to maximize

$$J(\theta) = E[min(ratio(\theta) \times A(s, a), clip(ratio(\theta), 1 - \varepsilon, 1 + \varepsilon) \times A(s, a)], \tag{9}$$

where $A(s, a) = Q(s, a) - V(s)$ is the advantage function, and ε is the clipping hyperparameter, which is tuned to limit large policy changes when $ratio(\theta)$ is large [22]. Clipping refers to keeping the quantity inside the given bounds. Values above the bounds are mapped to the maximum value $(1 + \varepsilon)$, and values below the bounds are mapped to the minimum $(1 + \varepsilon)$. Note that $V(s)$ is the state value function and represents the expected return from a given state [9]. TRPO also uses $ratio(\theta)$ and $A(s, a)$, and the maximization objective in TRPO is given by

$$E[ratio(\theta) \times A(s, a) - \beta KL[\pi_{old}, \pi]], \tag{10}$$

where β and KL are constants, noting that KL is the Kullback–Leibler divergence between the old and new policy estimates [23].

3. Similar Work

Researchers have conducted numerous comprehensive reviews on RL. Ref. [24] gives an overview of model and model-free RL algorithms and illustrates RL's tractability challenges. Ref. [25] sheds light on the DRL achievements of the early 2010s, such as the breakthroughs of [2,20,21]. Other notable reviews dedicated to RL and DRL are given in [26–28]. In terms of leveraging RL for practical use, Ref. [29] gives a thorough review of current RL applications and highlights RL trends in recommendation systems, gaming, automated vehicles, natural language processing, web security, robotics, energy management, healthcare and finance. In their review, Ref. [29] briefly discusses RL applicability in trading and credit risk but does not mention options or hedging.

In addition to these reviews of the entire field of RL, there are works that consider the use of RL in finance. A recent paper [12] provides a comprehensive overview of RL usage for optimal execution and market making in electronic markets, portfolio optimization, automated trading, order routing and option hedging. Ref. [12] only briefly discusses RL for option hedging and only references six of the reviewed articles from this paper [1,5,30–33].

Ref. [34] also reviews the use of RL in finance. In addition to discussing RL in financial technology (FinTech), Ref. [34] reviews RL applications in wealth management, retirement plan optimization, cryptocurrency price prediction, stock trading and portfolio management. Other works that take a comprehensive look at RL in finance include [35,36]. In addition to these broader surveys, narrower reviews have examined the use of RL applications in specific financial subdomains, such as algorithmic trading [37,38], market making [39], economics [40,41] and portfolio optimization [42].

In terms of reviewing RL methods for dynamic option hedging, which is the goal of this paper, only [43] was found. However, Ref. [43] only analyzes six of the works found in this paper. Moreover, Ref. [43] only provides a general analysis of the RL algorithms used for option hedging and does not give a detailed analysis of the implementation and rationale of different state and action spaces. Further, Ref. [43] does not review any studies that use empirical data.

4. Methodology

The objective of this review is to perform a thorough analysis of DRL utilization in dynamic stock option hedging. To identify pertinent studies, the initial step involved a manual search on Google Scholar and the University of Toronto online library, employing the keywords "reinforcement learning" and "hedging". The search process placed emphasis on the identification of studies that structure the dynamic hedging problem within an RL framework, with clear definitions given for the state, action and reward. Additionally, the selected articles employ an RL algorithm or its variant to address the dynamic hedging problem and carry out testing to compare the hedging results with established benchmarks. These inclusion attributes excluded works framed with model-based dynamic programming (DP), which led to the omission of the frequently cited works of Buehler et al. (2019, 2020) [44,45] in the financial RL literature. Refs. [44,45] use neural networks to solve the dynamic hedging problem from a DP perspective, which is not aligned with the objectives of this review.

As this review only considers the use of DRL for the dynamic hedging of stock options, similar analyses using DRL to hedge variable annuities [46], credit index options [47] and other contingent claims [48] were excluded. Further, thesis works were excluded. Overall, 15 papers meeting the criteria were found using the search criteria. Additionally, Ref. [49], which was found in the reference list of [12,50], was found when searching for papers that use RL in combination with employed generative adversarial networks (GANs) to solve the dynamic hedging problem. Therefore, a total of 17 papers were included in this analysis.

Each paper assessed in this review was analyzed in terms of its application of specific components within the field of DRL. The DRL ingredients of focus include the selection of the underlying algorithm, the definition of state and action spaces, the formulation of the reward function, the processes behind generating training and testing data and the establishment of a relevant comparison benchmark. Given the lack of consistent symbol use in the literature, the reader should note that this paper defines a specific symbol for each relevant variable for ease of comparison. Respectively, symbols C_t, K, S_t, Δ_t, ν_t, Γ_t, σ_t and n_t, represent the option price, strike price, stock price, BS Delta (BSD), BS Vega (BSV), BS Gamma (BSG), volatility and current shares held in the hedging position. The t subscript in each variable listed above alludes to the fact that the symbol represents a value at time step t. Table 1 lists the DRL methods, states, actions, rewards, data generation processes, and benchmarks used for each paper in chronological order. Note that the listings of Table 1 are generalized, and detail is provided where necessary in Section 5.

As shown in Table 1, only 10 of the works are peer-reviewed. This represents the novelty of the field. Not listed in Table 1 is the type of option being hedged, as all papers except [51] restrict their analyses to European options. Ref. [51] uses empirical American option prices for training and testing data but does not modify its DRL approach in any manner when considering American versus European options. Further, Ref. [51] uses the BS Delta strategy as a performance benchmark and does not compare the DRL agent performance to a strategy more apt for American options. As such, a deeper investigation into the use of DRL for hedging American and other exotic options is a worthy avenue for future research. As is discussed in Section 5.4, most of the papers in this review train the DRL agent with simulated asset price data that follow some stochastic process, such as GBM or a stochastic volatility model. If the option is European, the computation of the BS option price and the BS Greeks required for agent training and testing are trivial to obtain. However, the BS model does not consider the potential for early exercise [3]. When pricing and hedging an option wherein exercising early may be optimal for the holder, such as an American put option on a non-dividend paying stock, there is no analytic pricing formula, and numerical procedures must be used [4]. Therefore, an investigation into how DRL compares to other numerical methods for hedging and pricing American options is a compelling direction for future research.

Table 1. Summary of analyzed works.

Source	Reviewed	Method	State	Action	Reward	Train Data	Test Data	Benchmark
[5,30]	Yes	Q-Learning	S_t, τ, σ_t	Disc.	$\delta w_t - \lambda(\delta w_t)^2$	GBM	GBM	BSD
[31]	Yes	SARSA	S_t, τ, n_t	Disc.	$\delta w_t - \lambda(\delta w_t)^2$	GBM	GBM	BSD
[32]	Yes	DQN Pop-Art PPO	S_t, τ, n_t, K	Disc.	$\delta w_t - \lambda(\delta w_t)^2$	GBM	GBM	BSD
[52]	No	TRVO	C_t, S_t, Δ_t, n_t	Cont.	δw_t	GBM	GBM	BSD
[1]	Yes	DDPG	S_t, τ, n_t	Cont.	min *initial* $\mathbb{E}[w_t] + \lambda\sqrt{\mathbb{V}[w_t]}$,	GBM, SABR	GBM, SABR	BSD, Bartlett
[53]	Yes	IMPALA	S_t, τ	Disc.	+1, −1	HSX, HNX	HSX, HNX	Market Return
[49]	No	DQN, DDPG	$C_t, S_t, \Delta_t, n_t, \sigma_t$	Disc.	$\delta w_t - \lambda(\delta w_t)^2$	GBM, Heston	GBM, Heston, S&P	BSD, Wilmott
[54]	No	PG w/Baseline	S_t, τ, n_t	Disc.	δw_t	GBM, Heston	GBM, Heston, S&P	BSD
[50]	No	Dir. Policy Search	S_t, τ	Cont.	CVaR	GBM, GAN	GBM, GAN	BSD
[55]	No	DDPG	S_t, τ, n_t	Cont.	Payoff	GBM	GBM	BSD
[56]	No	Actor Critic	C_t, S_t, n_t, τ	Cont.	δw_t	Heston	Heston	BSD
[51]	Yes	DDPG	$S_t, \tau, \Delta_t, n_t, \mathrm{K}, \nu_t, \Gamma_t$	Cont.	$\delta w_t - \lambda(\delta w_t)^2$	S&P, DJIA	S&P, DJIA	BSD
[57]	Yes	TD3	S_t, τ, n_t, σ_t	Cont.	δw_t	GBM, Heston, S&P	GBM, Heston, S&P	BSD
[58]	Yes	D4PG-QR	$S_t, \Gamma_t^{port}, \nu_t^{port}, \Gamma_t^{hedge}, \nu_t^{hedge}$	Cont.	CvaR and modified mean-var.	SABR	SABR	BSD, BSDG, BSDV

Table 1. *Cont.*

Source	Reviewed	Method	State	Action	Reward	Train Data	Test Data	Benchmark
[59]	No	DDPG, DDPG-U	$S_t, \tau, n_t, \sigma_t, \Delta_t, \frac{dC}{dt}$	Cont.	$\delta w_t + \lambda Var[\delta w_t]$	GBM, S&P	GBM, S&P	BSD
[33]	Yes	CMAB	S_t, τ, n_t	Disc.	$\delta w_t - \lambda(\delta w_t)^2$	GBM	GBM	CMAB vs. DQN
[60]	No	DDPG	$C_t, S_t, \Delta_t, n_t, \tau$	Cont.	Min. c_t	GBM	GBM	BSD

5. Analysis

The comprehensive analysis in this review is organized into five distinct sections to provide a comprehensive and in-depth assessment of each paper. In Section 5.1, an examination of the DRL method employed in each paper is presented. Section 5.2 delves into the formulation of state and action spaces, closely examining the modeling choices, including the dimensionality of the state space and the representation of the action space. Section 5.3 is dedicated to the analysis of reward functions within the selected papers, including an investigation of their theoretical foundations and implications on the learning process. Section 5.4 highlights the methodologies employed for both training and testing data generation, providing a comprehensive exploration of simulation processes and, where relevant, providing the origins of empirical data. Last, Section 5.5 synthesizes the key findings and conclusions from each paper, offering a cohesive overview of the collective contributions of the reviewed studies.

5.1. RL Methods

This report focuses on how DRL is used to optimize the dynamic hedging problem. However, there are two studies [5,30,31] that do not formally use DRL, i.e., they approximate the optimal value function or policy with an NN. However, Refs. [5,31] are included in this analysis because they were the first two studies to formulate the dynamic hedging problem with a standard RL framework (state, action and reward). In [5], the optimal hedging problem is initially framed as a Markov decision process (MDP) that may be solved with model-based DP. Through DP, the study computes the optimal hedge strategy recursively at each time step, with the aid of basis function expansions to approximate the continuous state solution. Ref. [5] extends this DP result with a more traditional RL approach, relaxing the requirement for a precise model of the environment dynamics. Ref. [5] represents one argument of the Q-function, which can be found using Q-learning as the option price, and a second argument represents the optimal hedge for a given state–action pair. Therefore, the model introduced in [5], which is called Q-learning Black–Scholes (QLBS), allows the estimated option price and optimal hedge to be reflected with a single model. Ref. [5] then shows that QLBS can be extended to approximate a parametrized optimal Q-function by using fitted Q-learning (FQI). Note that Ref. [30] expands upon the work of [5] with numerical experiments. Thus, although Refs. [5,30] use function approximation techniques to solve the RL problem, the studies do not use an NN.

Ref. [31] was the next to frame the dynamic option hedging problem with RL, using SARSA as the RL method. Recall that the SARSA method requires a discrete look-up table for all Q-values. To allow for a higher-dimensional environment, Ref. [31] applies a non-linear regression of the form $y = Q(X)$ to approximate the Q-function, where X is the current state–action pair (s, a). Akin to [30], Ref. [31] uses a non-NN function approximation method. However, Refs. [5,30,31] are pioneering works regarding the use of RL to optimize the dynamic hedging problem.

Moving now to studies that use DRL, recall that DQN uses NNs to approximate the Q-function, thereby allowing for a continuous state space. However, recall that DQN requires a max operation over all possible next actions to approximate the Q-value target,

a condition that makes DQN intractable when a continuous action space needs to be searched [20]. In the context of hedging problems that ideally necessitate the acquisition the ability to buy various different amounts of shares depending on the underlying asset price, policy methods, which accommodate continuous action spaces, were more prevalent across the analyzed papers.

Ref. [49] uses both DQN and DDPG and compare the performance of each. The result of this comparison is discussed in detail with the rest of the results in Section 5.5. Another comparison of value- and policy-based methods is given in [32], which compares DQN to PPO. Ref. [32] also looks at a third method, called DQN pop-art, which clips the rewards of DQN to limit large gradient changes. The limitations of DQN are further examined in [33], which compares DQN to a contextualized multi-arm bandit (CMAB) approach. Ref. [33] describes that CMAB considers all rewards as being independent and identically distributed for each time step, meaning that the action selection does not impact future rewards.

Ref. [1] also points to the limitations of DQN in the hedging environment and employs a DDPG formulation with two separate Q-functions to serve as critics. The first Q-function evaluates the expected hedging cost, whereas the second Q-function evaluates the squared hedging cost, which is a measure of the hedging cost variance [1]. Refs. [51,55,60] also use the DDPG algorithm. Ref. [60] compares the performance of DDPG to the NN approximation of a model-based DP strategy. Ref. [59] modifies the DDPG algorithm, allowing the DDPG actor network to not only produce the action but also estimate the associated variance derived from the reward function. The critic network therefore assesses both the anticipated reward and variance for each actor's output. Consequently, an optimization of the critic network leads to high weights being placed on actions that lead to high expected rewards and low expected variance [59]. Ref. [59] calls this modified DDPG method DDPG with uncertainty (DPPG-U).

Ref. [57] found DDPG to be unstable in the option hedging environment, as it states that DDPG "seemed to either not converge at all or started to learn and then later diverge". Ref. [57] may have encountered this poor implementation of DDPG for various reasons. A lack of convergence may be due to the choice of algorithm hyperparameters (learning rates, discount factor, exploration parameter, batch size and more). The optimal hyperparameters for each training set and environment are different, and wrong hyperparameter choices can significantly impact the learning process [61]. Moreover, the phenomenon of early learning followed by divergence may be an occurrence of catastrophic forgetting, a phenomenon that occurs in non-stationary environments when new experiences encountered by the RL agent erase previously learned optimal actions, as described in [62]. Further, Fujimoto et al., 2018, describe that the DDPG critic network may overestimate Q-values, leading to unnecessary exploitation by the actor. Ref. [63] proposes a twin delayed DDPG (TD3) algorithm that uses a double critic. The idea of a double critic (double Q-learning) for DRL is discussed thoroughly in [64]. The TD3 update uses the minimum target value of the two critics, and the action range is clipped to smooth the actor's policy update [57]. The authors of Ref. [57] employ the TD3 algorithm in their study.

A further extension to DDPG is in [58], which uses a distributional DDPG variant. Distributional DDPG is akin to the distributional DQN algorithm proposed in [19], which defines $Z(s,a)$ as the distribution of discrete fixed-point rewards to which new generated values can be allocated. Ref. [65] shows that distributional DQN may be implemented for continuous action spaces with an actor–critic method, wherein the critic estimates $Z(s,a)$ and the actor updates the policy accordingly. For the RL hedging agent, [58] uses quantile regression (QR) to estimate $Z(s,a)$, a distributional DDPG variant first proposed in [66]. Ref. [58] makes this choice because two of its risk measures, value at risk (VaR) and conditional value at risk (CVaR), can be efficiently calculated with a quantile-based representation of outputs [58]. This method is called distributed distributional deep deterministic policy gradient with QR (D4PG-QR). Ref. [58] was the first to employ D4PG-QR to optimize dynamic hedging.

Another novel method in the DRL for option hedging literature is a TRPO algorithm variant that [52] proposes. Ref. [52] allows for a varied risk aversion parameter to be included as an input, allowing the RL agent to be trained to make decisions based on various risk appetites. Adding a risk aversion parameter to the state avoids having to retrain the agent each time the risk aversion level changes. Ref. [52] calls this TRPO variant trust region volatility optimization (TRVO). Ref. [56] introduces another method that incorporates variable risk aversion, proposing the construction of a linear combination of the state vector to learn general policies for a given hedging portfolio. Ref. [56] details that this linear combination is mapped into actions. By including the risk aversion parameter as a state feature, Ref. [56] explains that the policy is learned with respect to risk appetite. Ref. [56] uses an NN to optimize the actor and critic outputs.

Other papers that use policy methods are [50,53,54]. Ref. [54] uses a process wherein the return from an MC episode that follows a certain policy is computed and compared to the current baseline state value function. The policy gradient is then computed based on the comparison, and a new policy is generated [54]. The state value function is parametrized and trained alongside the policy, which is akin to an actor–critic method [54]. Ref. [50] also employs a policy search but does so directly, and the RL agent does not learn the value function at all. Ref. [50] cites [9], stating that this direct policy search allows for asymptotic convergence of deterministic policies, adequate exploration and use in continuous action spaces.

A final method used in the analyzed literature is importance weighted actor–learner architecture (IMPALA) [53]. The IMPALA method was first introduced in [67], which describes IMPALA as a distributed DRL technique wherein multiple actors operate in parallel. Each actor maintains its own respective value function estimate, and each actor therefore generates a separate trajectory [67]. The learner aggregates and updates the policy based on the distribution of all actor experiences [67]. Ref. [53] cites [67] in saying that the IMPALA method enables for better data efficiency and stability.

5.2. State and Action Spaces

With the RL algorithms for each study now presented, different state and action spaces may be investigated. Regarding the state space formulation for the dynamic hedging problem, all the analyzed studies include the stock price in the state space. Further, all works except [58] consider the current holding (previous action) in the state space. Ref. [58] makes no mention of why the current holding is not included. All authors include the time to expiration of the option in the state space except for [49,52,58]. Ref. [58] includes the BS Vega and Gamma in the state, whereas Refs. [49,52] include the BS Delta. As described in Section 1, the BS Vega, BS Gamma and BS Delta are all BS Greeks, and the BS Greeks are all functions of the BS option price and time [3]. In addition to [49,52], the state spaces of [51,59,60] also include the BS Delta in the state space in their studies. Several papers explicitly explain that the BS Delta can be deduced from the option and stock prices, and its inclusion in the state space only serves to unnecessarily augment the input size [1,31,32]. However, Ref. [49] claims that the largest contingent of state information possible leads to increased robustness. Further, Ref. [59] expresses that a larger state space leads the agent to learn further complexities inside the training data. Ref. [60] performs a comparison of DRL agents trained with BS Delta and BS Delta-free state spaces to show that the inclusion of the BS Delta leads to better learning in the dynamic hedging environment.

Refs. [49,52,56,60] include the option price in the state vector. Option price inclusion is notable in [49], which also includes volatility, the BS Delta, the stock price and the number of shares. Recall that the BS model option price is dependent only on time, stock price, volatility and strike price [3]. Therefore, by including the option price in the state, the agent of [49] can learn the time and strike price indirectly. Concerning volatility, Refs. [57,59] place volatility in their respective state spaces. Ref. [59] includes volatility alongside the BS Delta, stock price and time.

Given that many state space variables for a hedging problem are correlated through the BS model, it is natural to wonder about redundancy and to examine if the state space can be reduced. To examine this analytically, a feature engineering study or sensitivity analysis can be conducted on the input parameters. This is partly discussed in [59], which creates a relationship matrix between input variables but only includes volatility and the BS Greeks (Theta, Gamma, Vega and Delta). Refs. [5,30] transform the volatility, stock price, drift μ and time into one stationary variable, X_t. This transformation is given as $X_t = -(\mu - \sigma^2/2)t + \log(S_t)$ [5,30].

Ref. [56] mentions using news feeds and market sentiment as input features, but it is not explicitly mentioned how this state inclusion can be accomplished. Market sentiment would be an interesting variable for a sensitivity analysis, as one would think that market sentiment and stock price are highly correlated. Although enabling a model to include extra features is by no means a bad idea, it is valuable to consider the tradeoff between information and efficiency. Human traders do consider a vast array of variables when making hedging decisions, but humans do not face the computational burdens of automated agents. Expanding the number of states considered by a RL agent inevitably extends the time required for the agent to explore and compute the true value function or policy for each state [9].

Looking now toward action spaces, Refs. [5,30] formulate a problem in which the agent can act by selecting a discrete number of underlying stocks to buy or sell. A discrete action space is also used in [31,32], which both let the number of contracts of 100 shares be L, and the DRL agent may take a trading action in the discrete range of $\{-100\text{ L}, \ldots, 100\text{ L}\}$. As [33] uses a muti-arm bandit approach (CMAB), it uses 51 actions (bandits to select), representing 25 buy actions for the underlying stock, 25 sell actions and 1 hold action. In the second experiment in [33], short selling is prohibited, and the action space is limited to 26 choices. Refs. [49,53,54] also frame the action at each time step as the buying of some discrete number of shares.

The rest of the analyzed articles use a continuous action space wherein the agent can buy or sell a continuous number of underlying shares. Although many of these continuous action spaces are described in a broad manner, Refs. [51,57–59] all specifically mention that the action space has upper and lower bounds, and the action is a continuous fraction between the maximum and minimum hedge. Ref. [58] details that the maximum hedge for their agent is determined by ensuring that at least one of the two following conditions is met:

1. The ratio of post-hedging the BS Gamma to pre-hedging the BS Gamma is within the range of [0, 1].
2. The ratio of post-hedging the BS Vega to pre-hedging the BS Vega is within the range of [0, 1].

Ref. [57] uses a clipping formula for its outputted action, as mentioned in the description of the TD3 method in the previous section. Ref. [57] writes that the action at time step t is given by $a_t = clip(\mu(s_t|\phi_t), a_{low}, a_{high})$, where $\mu(s_t|\phi_t)$ is the parametrized policy.

5.3. Reward Formulations

Many of the analyzed works use a mean-variance formulation for the reward function. The mean-variance objective is a core concept of modern portfolio theory developed in [68]. The goal of a mean-variance reward function is reflective of the name, as the mean-variance objective is to maximize the expected mean reward while minimizing the expected variance of the reward. A mean-variance objective is the core idea of hedging, wherein traders try to limit the potential for large losses while still trying to maximize the potential for gains. A generalized version of the mean-variance reward for a single time-step t is given as

$$r_t = \delta w_t - \lambda \delta {w_t}^2, \tag{11}$$

recalling that λ is a measure of risk aversion. The mean-variance reward formulation is used in [5,30–33,49,51,59]. Ref. [1] uses a variation of the mean-variance reward formulation, letting w_t be the hedging loss from time t onward and defining

$$\Upsilon(t) = \mathbb{E}[w_t] + \lambda\sqrt{\mathbb{V}[w_t]}. \tag{12}$$

Ref. [1] trains the DDPG agent to minimize $\Upsilon(0)$, which is a minimization of cumulative hedging losses for an entire episode. Recall that Ref. [1] uses DDPG with two separate Q-function critics. The first critic evaluates $\mathbb{E}[w_t]$, whereas the second critic evaluates the expected variance, $\mathbb{V}[w_t]$. Refs. [52,54,56,57] all use the incremental wealth (δw_t) as the reward at each time step and do not consider the variance.

Refs. [53,55] both look ahead to the end of an episode before computing the reward. In [55], the episode reward is the difference between the cash held at expiry and the payoff of the short position in the call option (they are short, so the loss is $(S_T - K)_+$). In [53], entire episodes are simulated, and the hedging agent is rewarded a +1 if the profit is positive and -1 if the profit is negative. Ref. [50] also waits to the end of the episode to compute the reward. The reward in [50] is the CVaR of the payoff on the call option in which the agent has shorted plus the sum of the profits generated over the episode. CVaR and VaR are risk measures [4]. VaR finds a single point estimate of the maximum potential loss given a threshold (confidence level or risk aversion level) [4], whereas CVaR quantifies the expected loss beyond the single point estimate found with VaR [69]. Ref. [58] also incorporates CVaR and VaR into the reward, and the distributional nature of these risk measures is one of the underlying reasons behind the choice of its distributional DRL algorithm choice (D4PG-QR). Specifically, Ref. [58] looks at a 30-day trading period and tries to minimize the loss plus 1.645 times the standard deviation, while also attempting to minimize the VaR and CVaR at 95% confidence levels. Note that $N^{-1}(0.95) = 1.645$, which explains the significance of the standard deviation multiplier in the reward function used in [58].

5.4. Data Generation Processes

Once the RL algorithm, state space, action space and reward formulation are all configured, the RL agent needs data for both training and testing. In the dynamic hedging environment, data stem from the underlying asset price process. The asset price dynamics, S_t, for a continuous GBM process are given by $S_t = S_0\, e^{((r-\frac{\sigma^2}{2})t+\sigma W_t)}$. This equation may be discretized to obtain a simulation model of the asset price process in discrete time:

$$S_{t+\Delta t} = S_t e^{(r-\frac{\sigma^2}{2})\Delta t+\sigma\mathcal{N}(0,\,\sqrt{\Delta t})}. \tag{13}$$

The discretization size is given by $\Delta t = \frac{T}{N}$, where T is the time to maturity and N is the number of discretization steps. For each newly generated stock price, the BS model may be used to compute the BS option price and the BS Greeks, providing the hedging agent all requisite information at every time step. The generation of stock paths using GBM is straightforward, and due to its pivotal role as the underlying process for BS option price and Delta calculations, GBM is employed for training and testing data generation for at least one experiment in nearly every analyzed study, with only three exceptions [53,56,58]. As is discussed in Section 5.5, the first experimental step for most papers is the training of a DRL agent using GBM stock paths. For testing, more GBM stock paths are then generated to perform a hedging performance comparison between the proposed RL agent and the BS Delta hedging strategy. If no market frictions are considered and the option is European, outperforming the hedging of the BS Delta method in the limit of infinite testing episodes is theoretically impossible. Therefore, comparing the DRL agent to a BS delta strategy under no market frictions ensures that the DRL agent learns a theoretically viable strategy.

As discussed in Section 1, the assumption of constant volatility in GBM processes is not reflective of a true market environment [8]. Ref. [70] introduces the SABR (stochastic alpha,

beta, rho) model, which accounts for non-constant volatility by modeling the volatility as a stochastic process. Ref. [1] uses the SABR model for training and testing data generation after completing its first experiment with a GBM data process for training and testing. Ref. [58] uses only SABR to generate data. Ref. [58] does not explain why it does not perform an experiment with an agent trained on GBM data. Ref. [71] introduces a Delta hedging strategy for the SABR model (Bartlett Delta hedging) by taking into account both the influence of an alteration in the asset price and the anticipated alteration in implied volatility. Thus, once Ref. [1] trains its agent on SABR data in its second experiment, it compares the agent's test performance to both the BS Delta and the Bartlett Delta strategies using testing data generated with an SABR model.

Another common stochastic volatility model is the Heston [72] model. In the Heston model, volatility undergoes a stochastic evolution influenced by various parameters, including the volatility of volatility, the ongoing estimation of asset price variance and a mean-reversion rate for this estimation [72]. To train DRL agents for an environment with stochastic volatility, the Heston model is used to generate training data in [49,54,57,59]. Note that, to formulate a BS-based hedging strategy for an option written on an underlying Heston model path, the BS Greeks can be computed at each time step by using the square root of the variance process as the implied volatility [56].

Although Refs. [49,54,57,59] train and test their agents on simulation data, each of these studies performs an additional sim-to-real test in which it tests the simulation-trained DRL agent on empirical data. Ref. [49] uses end-of-day S&P500 asset prices from 2019–2022, omitting 2020 due to the COVID-19 pandemic. Ref. [49] also collects option prices for each day on each of the underlying assets. The data in [49] are from ivolatility (https://www.ivolatility.com). For its sim-to-real test, Ref. [54] uses S&P500 index prices from 1 January 2016 to 31 December 2019 and uses Wharton Data Research Services (WRDS) to find the prices on a set of European call options written in the S&P500 index (Strike Price of USD 1800.00, various maturities). Ref. [59] uses WRDS to obtain S&P500 index data, uses this index as the underlying asset price and uses 59,550 options with different strikes and maturities. The dataset used in [59] was originally created in [73]. Ref. [57] uses S&P index intraday options from 3 January 2006 to 31 December 2013, totaling 2009 trading days. Ref. [57] writes that the P&L calculation required for the agent's reward function is computed using the mid-price of the option quote. Ref. [57] uses data obtained from the Chicago Board of Options Exchange (CBOE) LiveVol data shop. In addition to using empirical data for a sim-to-real test, [57] trains its DRL agent using the empirical dataset from LiveVol.

Others that train a DRL agent with empirical data are [51,53]. Ref. [53] uses stock price data from the Ho Chi Minh Stock Exchange (HSX) and uses the Ha Noi Stock Exchange (HNX) for option prices. The data used in [53] are from 25 September 2017 to 21 May 2020 (17 May 2019 to 21 May 2020 for testing). Ref. [51] uses S&P500, S&P100 and DJIA price paths from years 2004–2020, obtained from the CBOE data shop. Ref. [51] performs a random 70–30 train–test split on the data. A final data generation process is considered in [50]. Ref. [50] trains and tests the DRL agent using data generated with a time series GAN. A GAN is an ML technique designed to learn the distribution of a collection of training examples in order to reproduce the given distribution [74]. The key results of all 17 analyzed works are presented in the next section.

5.5. Comparison of Results

With the complete learning process for each RL study now outlined, the results of these works may be analyzed. Ref. [30] performs numerical experiments for prior work, [5]. Ref. [30] considers no market frictions, and the QLBS model is tested with a risk aversion parameter $\lambda = 0.001$ (used in mean-variance reward formulation). GBM stock price paths are generated for training, and actions are multiplied by a uniform random variable in the range of $[1-\eta, 1+\eta]$, for $\eta = [0.15, 0.25, 0.35, 0.5]$ [30]. When compared to the BS delta hedging strategy, Ref. [30] shows that most actions are optimal, even with noise. Ref. [30]

concludes that the agent can learn from random actions and potentially even learn the BS model itself.

The analysis put forth in [31] is the first to consider transaction costs in the hedging environments [31]; it models the cost of crossing the bid–ask spread for a given tick size, formulating the transaction cost, c, as $c = 0.1 \times (|n| + 0.01n^2)$, where 0.1 is the tick size and n is the number of purchased shares. Note that the market impact is also considered in [31], as it adds $0.01n^2$ to the cost function. After comparing a transaction-cost-free DRL model with the BS Delta using a GBM simulation to show that the agent can learn to effectively hedge, Ref. [31] shows that, in the presence of transaction costs, the transitions between hedging actions are smaller for the RL agent than the BS Delta strategy. The smoother path of DRL actions, i.e., not buying (or selling) many shares in one time step due to a large change in stock price, reflects the RL agent's cost consciousness [31]. Ref. [32] finds a similar result to [31] in that its DRL agent is much less aggressive than the BS Delta strategy when hedging in an environment with transaction costs. This result found in [32] is confirmed with kernel density plots of volatility, t-statistics and total cost. Recall that [32] also compares the performance of three algorithms, DQN, DQN with pop-art (clipped actions) and PPO. Ref. [32] shows that the policy-based PPO algorithm converges much faster in all tests, whereas DQN with pop-art displays a more stable convergence profile than DQN.

The results of [52] are similar to [31,32]. Ref. [52] shows that the policies learned by their TRVO DRL agent are robust across multiple risk aversion parameters. Ref. [56] also incorporates multiple risk aversion parameters into its model and finds similar results to [52]. Ref. [56] shows that, when risk aversion is low, the agent prefers to leave positions unhedged, leading to reduced transaction costs. When the risk aversion is high, Ref. [56] shows that the agent prefers to hedge often, but like [31], the DRL agent action path is smoother and has less large hedging position changes than the BS Delta path, reflecting the cost consciousness of the RL agent. Ref. [51] draws similar conclusions to [55]. Ref. [59] models the environment with proportional transaction costs and simulates the performance of both DDPG and DDPG-U DRL agents trained on using GBM price paths. Ref. [59] finds that, although both DRL agents outperform the BS Delta strategy, the DDPG-U method outperforms DDPG by achieving a lower expected cost and a similar variance.

Ref. [1] also finds that its trained DRL agent is cost-conscious, outperforming the BS Delta strategy with GBM training and testing data when transaction costs are considered. The key result of [1] is that, when the current holding is close to the new required BS holding, the RL agent action transition closely matches that of a BS Delta hedge. However, if the new BS holding is much higher than the current one, the DRL agent prefers to under-hedge relative to the BS Delta (and vice versa for over-hedging). Ref. [1] also shows that, as the rebalance frequency increases, the enhancement of the DRL agent over the BS Delta becomes more pronounced. The BS Delta method is not cost-conscious like the DRL agent, and transaction costs accumulate as the position is rebalanced more frequently [1]. Ref. [1] then performs a test with stochastic volatility using the SABR model, and the DRL agent trained on SABR data once again outperforms the BS Delta and Bartlett Delta strategies.

Ref. [49] conducts a comparison between DQN and DDPG and shows that employing discrete actions with DQN leads to reduced P&L variance but also introduces rounding errors, resulting in a lower mean P&L compared to DDPG. Ref. [49] then shows that the DDPG agent trained on GBM data outperforms the BS Delta strategy when transaction fees are considered. Ref. [49] repeats the DDPG training and testing process with Heston model data, using a Wilmott Delta hedging strategy as a benchmark. A Wilmott Delta strategy is akin to BS Delta hedging, except that the strategy defines a non-constant no-transaction region to account for transaction costs [75]. Ref. [49] shows that its DRL agent outperforms the Wilmott Delta strategy. To test the robustness of the GBM-trained DRL agent and the Heston-model-trained DRL agent, Ref. [49] performs sim-to-real tests to evaluate the DRL agents' performance on empirical test data. In this sim-to-real test, Ref. [49] finds that both DRL agents exhibit significantly worse performance on empirical test data

compared to when the DRL agents are tested using data that align with the training model. To improve the robustness of its agents, Ref. [49] incorporates a domain randomization technique. Domain randomization results in different training environments, and agents can learn more generalized policies [49]. However, Ref. [49] concludes that the DRL agents trained with domain randomization show no significant improvements when tested on empirical data.

Ref. [59] also performs a sim-to-real experiment, training a DRL with data from a GBM process and testing the agent on real data. Ref. [59] concludes that its DRL agent shows the desired robustness and matches its simulated results, wherein the DRL agent learns a cost-efficient policy and achieves the desired results of a low cost and low volatility for the results. Ref. [54] finds similar results to [59] in its sim-to-real and stochastic volatility tests. Ref. [57] also uses empirical data for its study, and it finds that its DRL agent (TD3) outperforms a BS Delta strategy in terms of expected cost and variance for both constant and stochastic volatility. Ref. [57] concludes that the DRL agent is superior to the BS Delta strategy in both a sim-to-real test and a test wherein the DRL agent is trained on empirical data. Note that, based on the information in the respective papers, it is difficult to reason why [54,57,59] obtain preferable results in sim-to-real tests, whereas [49] do not. In the field of RL for dynamic hedging, the absence of a universally acknowledged benchmark or standardized testing dataset complicates the analysis of variations in DRL agent performance across different papers. For example, in the field of image classification, the performance of separate AI algorithms can be easily compared using the MNIST dataset. There is no such comparison method in the field of dynamic hedging with RL, making for a difficult comparison of algorithms put forth by separate parties.

In addition to [49] showing that the hedging performance of a DQN agent is bested by a DDPG agent, Ref. [32] shows that PPO outperforms DQN. Further, Ref. [33] compares a CMAB method to DQN with GBM simulated data and shows that CMAB has faster convergence, higher data efficiency and better hedging results than DQN, both when transaction costs are absent and when they are incorporated. Ref. [60] compares the performance of DDPG and an NN-approximated DP solution to the optimal hedging strategy. Using GBM simulated data, Ref. [60] concludes that, as the time to maturity, volatility and risk aversion increase, the DRL solution outperforms the approximated DP. This is an important result, as approximate DP solutions are prevalent in the dynamic option hedging field, and those using model-free RL argue that model-based approximate DP solutions can lead to errors due to calibration in model parameters [54].

The results of [58], which trains and tests its D4PG-QR agent on data from an SABR model process, shows that, under the presence of increased transaction costs, the DRL agent learns to decrease the BS Gamma hedge. The DRL agent in [58] beats the BS Delta, the BS Delta–Gamma and the BS Delta–Vega strategies in both expected cost and VaR (95%) when volatility is both constant and stochastic. Note that Delta–Gamma hedging involves taking a position that makes the hedging portfolio both Delta- and Gamma-neutral [4]. Similarly, a Delta–Vega hedge aims to make the hedging portfolio Delta- and Vega-neutral [4].

Other works with interesting results are [51,53], which use empirical data for training. Ref. [51] shows that a DDPG agent outperforms a BS Delta strategy in an empirical testing setting. Ref. [51] also shows that the BS Delta is bested by the DDPG agent when hedging empirical American options. However, one should note that the BS Delta strategy is derived without the consideration of early exercise [4]. Ref. [53] shows that its DRL agent yields a profit higher than the market return on the empirical testing dataset. A final notable result comes from [50], which shows that an agent trained on data produced by GANs achieves a higher profit than a DRL agent trained on GBM data. This represents a potential future direction for the literature, as GANs may provide a closer representation to real data than GBM processes [50].

6. Conclusions

In conclusion, this paper provides a thorough review of 17 papers that attempt to solve the problem of dynamic option hedging in frictional markets by leveraging DRL methods. Fifteen of the analyzed papers were found using the "reinforcement learning" and "hedging" key words in the University of Toronto library and Google Scholar, whereas the other two papers were found in a reference list and a manual search, respectively. For each of the 17 papers, the DRL models, state and action spaces, reward formulations, data generation processes and results were analyzed. Consistent with the explanations of the RL algorithms in Section 2, this review displays that most authors implement policy-based methods such as DDPG for their analyses because this DRL method allows for continuous action spaces. In terms of state spaces, the analyzed papers have no consensus on which variables to include and exclude in the feature vector. Most authors do not consider the trade-off between efficiency and information provided to the agent. Therefore, the conduction of a thorough sensitivity analysis is recommended to determine which variables may be redundant in the option hedging problem, such as the inclusion of the BS Delta when the option and stock prices are already used as state inputs.

An analysis of reward formulations put forth in the examined works shows that most papers use some variant of the mean-variance metric to maximize returns while minimizing the variance of those returns. However, authors who do not use a mean-variance reward formulation achieve similar results with other reward functions, such as episodic return, VaR and CvaR. As GBM is the underlying process for the BS model and BS Delta hedging, it is used as the data generation process in almost every paper. In addition to GBM, several studies also train their DRL agents on stochastic volatility models such as the SABR and Heston models. Four papers perform sim-to-real tests, and only three papers use empirical data to train their DRL agents.

The consensus conclusion from the analyzed studies is that DRL agents that are trained to monitor transaction costs are more cost-conscious than the BS Delta method in frictional environments, leading to higher final P&Ls. Each of the analyzed articles report very similar results when training and testing with both constant and stochastic volatility. However, when testing RL agents on real data, Refs. [54,57,59] see desirable performance, whereas [49] does not. Note again that it is difficult to compare the results of different papers due to a lack of a standardized dataset or universal benchmark.

Another interesting result in this analysis stems from the work of [50], which uses GANs to generate training and testing data. Although papers such as [57] use empirical data for training and testing, large enough empirical datasets may be difficult to obtain. Therefore, a recommended future direction in the study of DRL for dynamic option hedging is an investigation of whether GANs or another data generation method could advance the literature. Additionally, it is once again noted that each paper except [51], which performs an experiment with empirical American options, restricts its analysis to considering European options. As pricing and hedging American options require complex numerical methods [4], investigating the efficiency and performance of a DRL-based American option hedging solution is a worthy future research direction.

Author Contributions: Both authors contributed to the design of the study and outline of the paper. Paper reviews and analysis were conducted by R.P. All drafts of the manuscript were written by R.P. and Y.L. commented on all drafts. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Center for Management of Technology & Entrepreneurship (CMTE).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: R.P. would like to acknowledge Y.L. for his support throughout both the writing and editing processes.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cao, J.; Chen, J.; Hull, J.; Poulos, Z. Deep Hedging of Derivatives Using Reinforcement Learning. *J. Financ. Data Sci.* **2021**, *3*, 10–27. [CrossRef]
2. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
3. Black, F.; Scholes, M. The Pricing of Options and Corporate Liabilities. *J. Polit. Econ.* **1973**, *81*, 637–654. [CrossRef]
4. Hull, J. *Options, Futures, and Other Derivatives*, 8th ed.; Prentice Hall: Boston, MA, USA, 2012.
5. Halperin, I. QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *J. Deriv.* **2017**, *28*, 99–122. [CrossRef]
6. Leland, H.E. Option Pricing and Replication with Transactions Costs. *J. Financ.* **1985**, *40*, 1283–1301. [CrossRef]
7. Rogers, L.C.G.; Singh, S. The Cost of Illiquidity and Its Effects on Hedging. *Math. Financ.* **2010**, *20*, 597–615. [CrossRef]
8. Daly, K. Financial Volatility: Issues and Measuring Techniques. *Phys. Stat. Mech. Its Appl.* **2008**, *387*, 2377–2393. [CrossRef]
9. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; Bradford Books: Bradford, PA, USA, 2018; ISBN 0-262-03924-9.
10. Zou, L. *Meta-Learning: Theory, Algorithms and Applications*; Academic Press: Cambridge, MA, USA, 2022; ISBN 978-0-323-90370-7.
11. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An Introduction to Deep Reinforcement Learning. *Found. Trends Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]
12. Hambly, B.; Xu, R.; Yang, H. Recent Advances in Reinforcement Learning in Finance. *Math. Financ.* **2023**, *33*, 437–503. [CrossRef]
13. Al Mahamid, F.; Grolinger, K. Reinforcement Learning Algorithms: An Overview and Classification. In Proceedings of the 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Online, 12–17 September 2021; pp. 1–7.
14. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, Cambridge University, Cambridge, UK, 1989.
15. Tesauro, G. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Comput.* **1994**, *6*, 215–219. [CrossRef]
16. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv* **2016**, arXiv:1609.04747.
17. Lin, L.-J. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.* **1992**, *8*, 293–321. [CrossRef]
18. Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Larochelle, H.; Rowland, M.; Dabney, W. Revisiting Fundamentals of Experience Replay. *arXiv* **2020**, arXiv:2007.06700.
19. Bellemare, M.G.; Dabney, W. A Distributional Perspective on Reinforcement Learning. *arXiv* **2017**, arXiv:1707.06887.
20. Lillicrap, T.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971.
21. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; Volume 32, pp. 387–395.
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
23. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; Abdeel, P. Trust Region Policy Optimization. *arXiv* **2015**, arXiv:arXiv:1502.05477.
24. Dayan, P.; Niv, Y. Reinforcement Learning: The Good, The Bad and The Ugly. *Cogn. Neurosci.* **2008**, *18*, 185–196. [CrossRef]
25. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
26. Mousavi, S.S.; Schukat, M.; Howley, E. Deep Reinforcement Learning: An Overview. In *Lecture Notes in Networks and Systems, Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, London, UK, 21–22 September 2016*; Bi, Y., Kapoor, S., Bhatia, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 426–440.
27. Wang, H.; Liu, N.; Zhang, Y.; Feng, D.; Huang, F.; Li, D.; Zhang, Y. Deep Reinforcement Learning: A Survey. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 1726–1744. [CrossRef]
28. Botvinick, M.; Ritter, S.; Wang, J.X.; Kurth-Nelson, Z.; Blundell, C.; Hassabis, D. Reinforcement Learning, Fast and Slow. *Trends Cogn. Sci.* **2019**, *23*, 408–422. [CrossRef] [PubMed]
29. Sivamayil, K.; Rajasekar, E.; Aljafari, B.; Nikolovski, S.; Vairavasundaram, S.; Vairavasundaram, I. A Systematic Study on Reinforcement Learning Based Applications. *Energies* **2023**, *16*, 1512. [CrossRef]
30. Halperin, I. The QLBS Q-Learner Goes NuQLear: Fitted Q Iteration, Inverse RL, and Option Portfolios. *Quant. Financ.* **2019**, *19*, 1543–1553. [CrossRef]
31. Kolm, P.N.; Ritter, G. Dynamic Replication and Hedging: A Reinforcement Learning Approach. *J. Financ. Data Sci.* **2019**, *1*, 159–171. [CrossRef]
32. Du, J.; Jin, M.; Kolm, P.N.; Ritter, G.; Wang, Y.; Zhang, B. Deep Reinforcement Learning for Option Replication and Hedging. *J. Financ. Data Sci.* **2020**, *2*, 44–57. [CrossRef]
33. Cannelli, L.; Nuti, G.; Sala, M.; Szehr, O. Hedging Using Reinforcement Learning: Contextual k-Armed Bandit versus Q-Learning. *J. Financ. Data Sci.* **2023**, *9*, 100101. [CrossRef]
34. Malibari, N.; Katib, I.; Mehmood, R. Systematic Review on Reinforcement Learning in the Field of Fintech. *arXiv* **2023**, arXiv:2305.07466.
35. Charpentier, A.; Élie, R.; Remlinger, C. Reinforcement Learning in Economics and Finance. *Comput. Econ.* **2023**, *62*, 425–462. [CrossRef]

36. Singh, V.; Chen, S.-S.; Singhania, M.; Nanavati, B.; Kar, A.K.; Gupta, A. How Are Reinforcement Learning and Deep Learning Algorithms Used for Big Data Based Decision Making in Financial Industries—A Review and Research Agenda. *Int. J. Inf. Manag. Data Insights* **2022**, *2*, 100094. [CrossRef]
37. Pricope, T.V. Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review. *arXiv* **2021**, arXiv:2106.00123.
38. Sun, S.; Wang, R.; An, B. Reinforcement Learning for Quantitative Trading. *Assoc. Comput. Mach.* **2023**, *14*, 1–29. [CrossRef]
39. Gašperov, B.; Begušić, S.; Posedel Šimović, P.; Kostanjčar, Z. Reinforcement Learning Approaches to Optimal Market Making. *Mathematics* **2021**, *9*, 2689. [CrossRef]
40. Atashbar, T.; Aruhan Shi, R. *Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects*; IMF Working Papers; International Monetary Fund: Washington, DC, USA, 2022; Volume 2022.
41. Mosavi, A.; Faghan, Y.; Ghamisi, P.; Duan, P.; Ardabili, S.F.; Salwana, E.; Band, S.S. Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics. *Mathematics* **2020**, *8*, 1640. [CrossRef]
42. Sato, Y. Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey. *arXiv* **2019**, arXiv:1904.04973.
43. Liu, P. A Review on Derivative Hedging Using Reinforcement Learning. *J. Financ. Data Sci.* **2023**, *5*, 136–145. [CrossRef]
44. Buehler, H.; Gonon, L.; Teichmann, J.; Wood, B. Deep Hedging. *Quant. Financ.* **2019**, *19*, 1271–1291. [CrossRef]
45. Buehler, H.; Gonon, L.; Teichmann, J.; Wood, B.; Mohan, B.; Kochems, J. *Deep Hedging: Hedging Derivatives Under Generic Market Frictions Using Reinforcement Learning*, Swiss Finance Institute Research Paper No. 19-80. SSRN 2020. *preprint*.
46. Chong, W.F.; Cui, H.; Li, Y. Pseudo-Model-Free Hedging for Variable Annuities via Deep Reinforcement Learning. *Ann. Actuar. Sci.* **2023**, *17*, 503–546. [CrossRef]
47. Mandelli, F.; Pinciroli, M.; Trapletti, M.; Vittori, E. Reinforcement Learning for Credit Index Option Hedging. *arXiv* **2023**, arXiv:2307.09844.
48. Carbonneau, A. Deep Hedging of Long-Term Financial Derivatives. *Insur. Math. Econ.* **2021**, *99*, 327–340. [CrossRef]
49. Giurca, B.; Borovkova, S. *Delta Hedging of Derivatives Using Deep Reinforcement Learning*, SSRN 2021. *preprint*.
50. Kim, H. Deep Hedging, Generative Adversarial Networks, and Beyond. *arXiv* **2021**, arXiv:2103.03913.
51. Xu, W.; Dai, B. Delta-Gamma–Like Hedging with Transaction Cost under Reinforcement Learning Technique. *J. Deriv.* **2022**, *29*, 60–82. [CrossRef]
52. Vittori, E.; Trapletti, M.; Restelli, M. Option Hedging with Risk Averse Reinforcement Learning. In Proceedings of the ICAIF' 20: Proceedings of the First ACM International Conference on AI in Finance, New York, NY, USA, 15–16 October 2020; Association for Computing Machinery: New York, NY, USA; 2021.
53. Pham, U.; Luu, Q.; Tran, H. Multi-Agent Reinforcement Learning Approach for Hedging Portfolio Problem. *Soft Comput.* **2021**, *25*, 7877–7885. [CrossRef] [PubMed]
54. Xiao, B.; Yao, W.; Zhou, X. Optimal Option Hedging with Policy Gradient. In Proceedings of the 2021 International Conference on Data Mining Workshops (ICDMW), Auckland, New Zealand, 7–10 December 2021; pp. 1112–1119.
55. Assa, H.; Kenyon, C.; Zhang, H. *Assessing Reinforcement Delta Hedging*, SSRN 2021. *preprint*.
56. Murray, P.; Wood, B.; Buehler, H.; Wiese, M.; Pakkanen, M. Deep Hedging: Continuous Reinforcement Learning for Hedging of General Portfolios across Multiple Risk Aversions. In Proceedings of the ICAIF' 22: Proceedings of the Third ACM International Conference on AI in Finance, New York, NY, USA, 2–4 November 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 361–368.
57. Mikkilä, O.; Kanniainen, J. Empirical Deep Hedging. *Quant. Financ.* **2023**, *23*, 111–122. [CrossRef]
58. Cao, J.; Chen, J.; Farghadani, S.; Hull, J.; Poulos, Z.; Wang, Z.; Yuan, J. Gamma and Vega Hedging Using Deep Distributional Reinforcement Learning. *Front. Artif. Intell.* **2023**, *6*, 1129370. [CrossRef] [PubMed]
59. Zheng, C.; He, J.; Yang, C. Option Dynamic Hedging Using Reinforcement Learning. *arXiv* **2023**, arXiv:2306.10743.
60. Fathi, A.; Hientzsch, B. A Comparison of Reinforcement Learning and Deep Trajectory Based Stochastic Control Agents for Stepwise Mean-Variance Hedging. *arXiv* **2023**, arXiv:2302.07996. [CrossRef]
61. Ashraf, N.M.; Mostafa, R.R.; Sakr, R.H.; Rashad, M.Z. Optimizing Hyperparameters of Deep Reinforcement Learning for Autonomous Driving Based on Whale Optimization Algorithm. *PLoS ONE* **2021**, *16*, e0252754. [CrossRef] [PubMed]
62. Wang, N.; Zhang, D.; Wang, Y. Learning to Navigate for Mobile Robot with Continual Reinforcement Learning. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 3701–3706.
63. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
64. Van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. *arXiv* **2015**, arXiv:1509.06461. [CrossRef]
65. Barth-Maron, G.; Hoffman, M.W.; Budden, D.; Dabney, W.; Horgan, D.; TB, D.; Lillicrap, T. Distributed Distributional Deterministic Policy Gradients. *arXiv* **2018**, arXiv:1804.08617.
66. Dabney, W.; Rowland, M.; Bellemare, M.G.; Munos, R. Distributional Reinforcement Learning with Quantile Regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
67. Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv* **2018**, arXiv:1802.01561.
68. Markowitz, H. Portfolio Selection. *J. Financ.* **1952**, *7*, 77–91. [CrossRef]

69. Rockafellar, R.T.; Uryasev, S. Conditional Value-at-Risk for General Loss Distributions. *J. Bank. Financ.* **2002**, *26*, 1443–1471. [CrossRef]
70. Hagan, P.; Kumar, D.; Lesniewski, A.; Woodward, D. Managing Smile Risk. *Wilmott Mag.* **2002**, *1*, 84–108.
71. Bartlett, B. Hedging under SABR Model. *Wilmott Mag.* **2006**, *4*, 2–4.
72. Heston, S.L. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *Rev. Financ. Stud.* **1993**, *6*, 327–343. [CrossRef]
73. Wachowicz, E. Wharton Research Data Services (WRDS). *J. Bus. Financ. Librariansh.* **2020**, *25*, 184–187. [CrossRef]
74. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
75. Whalley, A.E.; Wilmott, P. An Asymptotic Analysis of an Optimal Hedging Model for Option Pricing with Transaction Costs. *Math. Financ.* **1997**, *7*, 307–324. [CrossRef]