A Comparative Study of Deep Reinforcement Learning Algorithms for Dynamic Option Hedging

Andrei Neagu[1] **Frédéric Godin**[2] **Leila Kosseim**[1]
[1]Dept. of Computer Science and Software Engineering, Concordia University, Montréal, Canada
[2]Dept. of Mathematics & Statistics, Concordia University, Montréal, Canada
{andrei.neagu, frederic.godin, leila.kosseim}@concordia.ca

Abstract

Dynamic hedging involves periodically trading financial assets to offset the risk associated with a correlated liability. Deep Reinforcement Learning (DRL) algorithms have been used to find optimal solutions to dynamic hedging problems by framing them as sequential decision-making problems. However, most previous work assesses the performance of only one or two DRL algorithms, making an objective comparison across algorithms difficult. In this paper, we compare the performance of eight DRL algorithms in the context of dynamic hedging; Monte Carlo Policy Gradient (MCPG), Proximal Policy Optimization (PPO), four variants of Deep Q-Learning (DQL) and two variants of Deep Deterministic Policy Gradient (DDPG). Two of these variants represent a novel application to the task of dynamic hedging. In our experiments, we use the Black-Scholes delta hedge as a baseline and simulate the dataset using a GJR-GARCH(1,1) model. Results show that MCPG obtains the best performance in terms of the root semi-quadratic penalty. Moreover, MCPG is the only algorithm to outperform the Black-Scholes delta hedge baseline with the allotted computational budget, possibly due to the sparsity of rewards in our environment. All datasets and code are available here.

1 Introduction

Hedging is a financial practice in which one or multiple assets are traded to minimize the risk associated with a correlated financial liability. In dynamic hedging, the hedging portfolio is periodically rebalanced, unlike static hedging, which remains fixed until the expiry date of the liability. By allowing continuous adjustment to changing risk profiles, dynamic hedging can be more effective. Since dynamic hedging can be formulated as a sequential decision-making problem, DRL methods can be used to learn optimal hedging strategies, where states represent market conditions and actions specify the number of hedging asset shares to be included in the hedging portfolio. However, most of the previous related works only investigate the performance of one or two DRL algorithms for the task of dynamic hedging, making an objective comparison between algorithms difficult.

Contributions The main contributions of this paper are two-fold: **1)** We compare the performance and computational efficiency of widely used DRL algorithms for dynamic hedging. **2)** We evaluate previously unexplored variants of DRL algorithms in this context, namely: Dueling DQL and Dueling Double DQL.

2 Related Work

The first papers tackling dynamic hedging using DRL are (Buehler *et al.*, 2019) and (Halperin, 2020). The former seminal paper, along with some subsequent works, uses a Monte Carlo Policy Gradient (MCPG) algorithm (Buehler *et al.*, 2019; Carbonneau and Godin, 2021; Horvath *et al.*, 2021; Carbonneau and Godin, 2024; Neagu *et al.*, 2024). This approach directly learns the optimal policy, which is the mapping from states to actions that minimizes a given risk measure applied

to the terminal hedging loss. Since then, multiple papers have expanded upon this work by using different DRL algorithms. (Du *et al.*, 2020) have used *value-based* algorithms, such as Deep Q-Learning (DQL) (Mnih *et al.*, 2015), which learn the value of taking any action in a given state; the optimal policy is then derived by taking the action with the highest value in each state. Moreover, the Fitted Q-Iteration algorithm, which is an adaptation of Q-learning, is used in the QLBS model of (Halperin, 2020). The performance of the QLBS approach is investigated in (Stoiljkovic, 2025). Other papers used *actor-critic* algorithms which combine policy- and value-based approaches, see for instance (Cao *et al.*, 2023; Marzban *et al.*, 2023; Sharma *et al.*, 2024). Actor-critic algorithms considered include Proximal Policy Optimization (PPO) (Schulman *et al.*, 2017) used by (Du *et al.*, 2020), Deep Deterministic Policy Gradient (DDPG) (Lillicrap *et al.*, 2016) used by (Cao *et al.*, 2020) and Twin Delayed DDPG (TD3) (Fujimoto *et al.*, 2018) used by (Mikkilä and Kanniainen, 2023). However, all these works focus on the evaluation of a single algorithm, making standard comparisons difficult. To our knowledge, (Du *et al.*, 2020) is the only work that analyzes and compares the performance of multiple algorithms, but limits this analysis to only DQL and PPO.

3 BACKGROUND

This section provides financial background knowledge, describes the market environment model considered, and provides an overview of the different DRL algorithms that are compared. For the reader's convenience, all financial and DRL notations used in this paper are summarized in Table 1.

Financial Symbols					
$\mathbf{t}$	**time step**	X	hedging strategy (# stock shares)	K	strike price
T	expiry	p_0	option premium	ρ	risk measure
R	random loss variable	$\mathcal{P}_X$	profits	M	cash reserve
r_f	risk-free rate	e^{r_f}	accrual factor	c	transaction total
$\mathbf{S}$	**underlying stock price**	$\mathbf{V}$	**hedging portfolio value**	δ_t	time step length
DRL Symbols					
s	state	a	action	r	reward
$\pi(a\|s)$	stochastic policy	$\pi(s)$	deterministic policy	γ	discount factor
α	policy learning rate	β	value function learning rate	θ	policy parameters
ϕ	value function parameters	$Q(s,a)$	state-action value function		

Table 1: Symbols and their definition, with state variables in bold.

3.1 FINANCIAL BACKGROUND

An option is a financial contract which gives the right to purchase (for a *call* option) or to sell (for a *put* option) an asset, referred to as the underlying asset, at a given date T, called the *expiry* and at a predetermined price, called the *strike price* K. Movements in the price of the underlying asset cause either an increase or a decrease in the value of the option. This correlation can be leveraged by a financial institution which issued the option; the value of which is a liability to them. To offset the risk of a potential appreciation or depreciation in the value of the option, the financial institution can periodically purchase or sell shares of the underlying asset.

In our work, we hedge the sale of a call option, for which we receive a cash amount, called a *premium* p_0, and hedge the risk of the option value increasing by periodically adjusting the number of shares of the underlying stock held at each time step $t = 0, 1, \ldots, T$.

3.1.1 DYNAMIC HEDGING AS AN OPTIMIZATION PROBLEM

To offset potential losses related to the sale of the call option at the expiry T, a hedging portfolio consisting of cash and underlying stock shares is set up. The dynamic hedging problem consists of selecting a hedging strategy $X = \{X_t\}_{t=1}^{T}$ which minimizes the possible risk at time T, where X_t

corresponds to the number of stock shares in the portfolio during $(t-1, t]$. That is, we wish to solve

$$X^* = \arg\min_X \rho(R) \tag{1}$$

where ρ is a risk measure mapping a random loss variable R into a real number representing the risk, and where $R = -\mathcal{P}_X$ is the negative of the total profits $\mathcal{P}_X$ at time T under the hedging strategy X. Several risk measures ρ have been used in the literature (Carbonneau and Godin, 2024). In our work, we consider the *root semi-quadratic penalty (RSQP)* risk measure

$$\rho^{RSQP}(R) = \sqrt{\mathbb{E}\left[R^2 \mathbb{1}_{\{R>0\}}\right]}. \tag{2}$$

where $\mathbb{1}_{\{R>0\}}$ is an indicator variable taking value 1 if $R > 0$ and 0 otherwise. The popular *quadratic penalty* is not considered as a risk measure, since it has the downside of penalizing gains, unlike its *semi-quadratic* counterpart.

Let the quantity M_t denote the amount of cash in the portfolio at time t right before any transaction. Since all transactions of stock shares are financed through the portfolio cash reserve represented by process M, the hedging portfolio is said to be *self-financing*. At each time period, the cash reserve is set to accrue (increase) by a factor of e^{r_f} where r_f is the one-period risk-free rate. The cash amount in the portfolio can be found recursively through

$$M_t = \begin{cases} p_0 & \text{for } t = 0, \\ (M_{t-1} - c_t(X))e^{r_f} & \text{for } t = 1, \dots, T, \end{cases} \tag{3}$$

with the stock transaction total amount at time t being

$$c_t(X) = \begin{cases} 0 & \text{for } t = 0, \\ S_{t-1}(X_t - X_{t-1}) & \text{for } t = 1, \dots, T, \end{cases} \tag{4}$$

where S_t is the underlying stock price at time t and $X_0 = 0$. Consider an agent hedging the sale of a call option. If, at the expiry T, the underlying stock price is larger than the strike price K, the buyer of the option will choose to exercise their right to buy the underlying stock at the strike price K. After implementing the hedging strategy X, the total profit for the agent right after the expiry is:

$$\mathcal{P}_X = S_T X_T + M_T - \mathbb{1}_E(S_T - K), \tag{5}$$

with $\mathbb{1}_E$ being the indicator variable worth 1 if event $E \equiv \{S_T > K\}$ occurs, or 0 otherwise. An important value that we use as a state variable into the DRL algorithm is the portfolio value:

$$V_t = S_t X_t + M_t, \tag{6}$$

which is the sum of the value of the underlying stock shares currently held and the cash amount at time t.

3.2 Market Environment

We define an underlying stock price process $S = \{S_t\}_{t=0}^T$, where $S_t = S_{t-1}\exp(Y_t)$, and Y_t is the time-t log-return of the underlying stock. Log-returns are modeled with the GJR-GARCH(1,1) model (Glosten *et al.*, 1993):

$$\begin{aligned} Y_t &= \mu + \varepsilon_t \\ \varepsilon_t &= \sigma_t z_t, \qquad\qquad z_t \sim \text{i.i.d.}(0,1) \\ \sigma_t^2 &= \nu_0 + (\nu + \lambda I_{t-1})\varepsilon_{t-1}^2 + \xi\sigma_{t-1}^2 \end{aligned} \tag{7}$$

with

$$I_{t-1} = \begin{cases} 0 & \text{if } Y_{t-1} \geq \mu \\ 1 & \text{if } Y_{t-1} < \mu \end{cases}$$

where $\{\sigma_t^2\}_{t=1}^T$ are the conditional variances of log-returns, $\{\mu, \lambda\} \in \mathbb{R}$ and $\{\nu_0, \nu, \xi\} \in \mathbb{R}^+$ are the model parameters, and $\{\varepsilon_t\}_{t=1}^T$ is a series of standard normal random variables.

The GJR-GARCH(1,1) model expands upon the geometric Brownian motion from the classic Black-Scholes model (Black and Scholes, 1973) by assuming the presence of stochastic volatility, volatility clustering and the leverage effect; features that more closely resemble real stock prices behaviour.

3.3 DEEP REINFORCEMENT LEARNING ALGORITHMS

Our work compares the performance of eight DRL algorithms. Table 2 describes the attributes of each algorithm as well as showing which variants stem from which baseline algorithm (i.e. TD3 from DDPG; Double, Dueling, and Dueling Double (DD) DQL from DQL).

Algorithm	Type	Action Space	# of Hyperparameters
DQL	Value-based	Discrete	17
Double DQL	Value-based	Discrete	19
Dueling DQL*	Value-based	Discrete	17
DD DQL*	Value-based	Discrete	19
MCPG	Policy-based	Continuous	9
PPO	Actor-critic	Continuous	17
DDPG	Actor-critic	Continuous	22
TD3	Actor-critic	Continuous	27

Table 2: Characteristics of the DRL algorithms explored in this paper. The algorithms denoted by stars are the two variants not previously explored in the dynamic hedging literature.

3.3.1 DEEP Q-LEARNING (DQL)

Deep Q-Learning (DQL) (see Mnih *et al.* (2013)) has become a widely used DRL algorithm since 2013 when it was first used to surpass the performance of human players on a number of the suite of Atari games. DQL works by deriving the optimal policy from an action-value function which is derived from the *Bellman equation*. The optimal action-value function is the expected return (sum of rewards r) when taking a given action a in state s and following the optimal policy π^* subsequently: $Q^*(s,a) = \mathbb{E}\big[\sum_{u=t}^{\infty} \gamma^{u-t} r_u | s_t = s, a_t = a, (a_{u+1},)_{u=t}^{\infty} \sim \pi^*\big]$ with γ being a discount factor. In DQL, the optimal action-value function is represented with a neural network with parameter set $\phi = \{\phi_i\}_{i=0}^{\mathcal{I}-1}$ which is estimated iteratively, for a total of $\mathcal{I}$ iterations.

DQL is prone to overestimating state–action values. Double DQL mitigates this by using separate networks for action selection and evaluation (van Hasselt *et al.*, 2015). Another variant of DQL is Dueling DQL which decomposes the Q-function into state-value and advantage components, improving generalization when actions have similar effects (Wang *et al.*, 2016).

3.3.2 MONTE CARLO POLICY GRADIENT (MCPG)

Classic value-based RL methods tend to suffer from high computational complexity as we increase the dimension of the problem (Sutton *et al.*, 1999). An alternative approach to mitigate such problem is to directly parameterize the policy instead of deriving it from a value function (Sutton *et al.*, 1999).

A simple instance of such approach is the Monte Carlo Policy Gradient found in (Buehler *et al.*, 2019). To improve the policy, multiple instances of the hedging loss R are simulated based on the current policy. Its distribution is then used to assess the value of a given policy. In our work, we employ a deterministic policy $a \sim \pi(s;\theta)$ as in (Buehler *et al.*, 2019). The policy parameters are updated iteratively using stochastic gradient descent with batches of $R \equiv \{R_n\}_{n=1}^{N}$ of N i.i.d. loss variables simulated with the current policy parameters θ_i:

$$\theta_{i+1} \leftarrow \theta_i - \alpha \nabla_{\theta_i} \widehat{\rho}^{RSQP}(R) \tag{8}$$

where the gradient of the empirical estimate of the ***RSQP*** risk measure $\widehat{\rho}^{RSQP}(R)$ is computed in closed-form with automatic differentiation packages, where α is the learning rate of the procedure.

3.3.3 DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

Deep Deterministic Policy Gradient (DDPG) (Lillicrap *et al.*, 2016) is an actor–critic algorithm in which a policy network (actor) is directly learned and iteratively refined using a learned action-value function (critic), rather than being derived from the Q-function as in DQL. DDPG employs

a deterministic policy $\pi(s;\theta)$, and approximates the action-value function with a neural network $Q(s,a;\phi)$. The objective is to jointly learn the actor and critic parameters θ and ϕ.

A key advantage of DDPG over DQL is its ability to naturally handle continuous action spaces. In DQL, computing $\max_a Q^*(s,a)$ is tractable only for discrete action spaces, whereas DDPG avoids this issue by using a differentiable policy with continuous outputs.

However, DDPG also suffers from overestimation bias similar to DQL (Fujimoto *et al.*, 2018). Twin Delayed DDPG (TD3) (Fujimoto *et al.*, 2018) addresses this issue by learning two action-value functions and using the minimum of the two for target updates, while also delaying target network updates, which has been shown to improve performance.

3.3.4 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) (Schulman *et al.*, 2017) is an actor–critic algorithm designed to reduce policy update variance. It clips the probability ratio, limiting how far the new policy can deviate from the old one, thereby preventing large, potentially catastrophic updates.

4 Experimental Setting

This paper provides an objective comparison of the performance and computational complexity of eight DRL algorithms for dynamic hedging: DQL, Double DQL, Dueling DQL, Dueling Double DQL, MCPG, DDPG, TD3, and PPO. To our knowledge, Dueling DQL and Dueling Double DQL have not been studied in this context before.

4.1 Baseline

We evaluate our DRL model's optimal hedging strategies X^* against the Black–Scholes delta hedge (B-S DH) (Black and Scholes, 1973), a standard baseline in the literature:

$$\begin{aligned} X_{t+1} &= \Phi(d_1), \qquad \text{for } t = 0, \dots, T-1, \\ d_1 &= \frac{\log(S_t/K) + (r_f + \sigma^2/2)(T-t)\delta_t}{\sqrt{\sigma^2(T-t)\delta_t}} \end{aligned} \tag{9}$$

where Φ is the standard normal cumulative distribution function, and δ_t is the time elapsing (in years) between any two time points t. In the absence of transaction costs, in continuous time, and for market dynamics following a geometric Brownian motion, this procedure is shown to completely eliminate risk (Black and Scholes, 1973), which explains its popularity.

4.2 Financial Setting

In our experiments, we hedge the sale of a standard call option with strike price $K = 100$ and one-year expiry with monthly time steps $\left(T = 12,\ \delta_t = \frac{1}{12}\right)$ and employ the RSQP (Eq. 2) as a risk measure. The premium p_0 is set to the Black-Scholes option price and the risk-free rate to $r_f = 0$.

4.3 DRL Setting

4.3.1 Dataset

DRL algorithms are trained on simulated price paths following Eq. 7, with parameters estimated via maximum likelihood from monthly S&P 500 prices $(2000/11/15 - 2024/10/15)$: $\mu = 0.00533410$, $\nu_0 = 0.00018216$, $\nu = 0.00026564$, $\xi = 0.70611408$, $\lambda = 0.34275732$.

Hyperparameter tuning is performed over 200,000 updates on a validation set of 2^{17} paths; training is run for 500,000 updates on 2^{19} paths with early stopping. Testing is done on 10 different test sets of 2^{17} paths, using the average RSQP and its standard deviation as a performance measure. Policies and value functions used by the DRL algorithms are approximated using feed-forward neural networks.

States. The states input into the DRL algorithms at each time step t consist of three variables: the current normalized time step $\frac{t}{T} \in (0, 1)$, the current normalized underlying stock price $\frac{S_t}{S_0} \in \mathbb{R}^+$, and the current normalized hedging portfolio value $\frac{V_t}{V_0} \in \mathbb{R}$.

Actions. The actions at each time step t consist of the next number of the underlying shares to hold $X_{t+1} \in (0, 1)$. As DQL algorithms require discrete action spaces (see Table 2), their actions are discretized as $X_{t+1} \in \{0.00, 0.02, \ldots, 0.98, 1.00\}$, for a total of 51 possible actions.

Rewards. The only non-zero reward in an episode is given at time T. The expected reward at time T is directly estimated from a mini-batch of size N:

$$\mathbb{E}[r_T] = -\sqrt{\frac{1}{N}\sum_{n=1}^{N} R_n^2 \mathbb{1}_{\{R_n>0\}}}. \tag{10}$$

However, TD updates are performed using the reward $r_T = -R^2 \mathbb{1}_{\{R>0\}}$ whose expected value is

$$\mathbb{E}[r_T] = -\frac{1}{N}\sum_{n=1}^{N} R_n^2 \mathbb{1}_{\{R_n>0\}}. \tag{11}$$

The difference arises because TD learning requires additive rewards, whereas the square root in RSQP is non-additive. Therefore, TD updates use the negative squared loss as the reward, while Monte Carlo methods apply the square root only after expectation.

Rewards are null for all prior time steps, i.e. $r_t = 0$ for $t = 0, \ldots, T-1$. Rewards in this environment are therefore sparse, with a single reward being provided on each episode.

Episodes. Figure 1 shows a DRL dynamic hedging episode over $t = 0, \ldots, T$. At each time step, the state s_t is fed into the DRL algorithm, which outputs action X_{t+1} until the final hedging loss R is computed. For MCPG and PPO, which rely on Monte Carlo rollouts to compute policy gradients, losses are backpropagated through the episode, giving the policy network an implicit recurrent structure, while value networks are updated via Temporal-Difference (TD) learning.

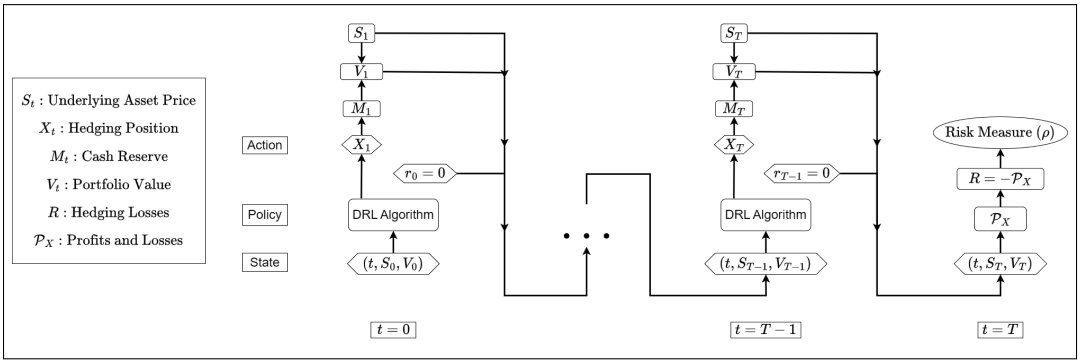

Figure 1: An episode of DRL dynamic hedging from $t = 0, \ldots, T$.

4.3.2 Early Stopping

Early stopping was implemented by training the DRL algorithms on the training set and testing them on the validation set every 1000 episodes. For any given algorithm, we train it until two conditions are met: 1) the last five validation RSQPs have not improved over the sixth-last and 2) these six validation RSQPs are lower than that of the B-S DH baseline. Additionally, the model with the lowest validation RSQP was saved for out-of-sample testing. Only MCPG triggered early stopping.

5 EXPERIMENTAL RESULTS

Table 3 shows the performance of the eight tested algorithms. The best performing DRL algorithm in terms of RSQP is MCPG (RSQP: 0.8248). The outperformance of MCPG over the baseline can be attributed to the fact that it directly learns the policy, with results showing that this approach leads to higher performance in the context of option hedging. Notably, its standard deviation across the 10 runs on the test sets is also the lowest among the other algorithms and the B-S DH, emphasizing its robustness. The other algorithms, such as the DQL family of algorithms, rely exclusively on *Temporal-Difference (TD) learning*, where an update is performed at each time step to learn a value function, while Actor-Critic algorithms, such as PPO, DDPG, and TD3, learn both the policy and a value function. The formulation of the option hedging problem leads to an environment in which rewards are sparse, where only a single reward is obtained from the environment at the final time step. Thus, early-iteration updates of TD learning algorithms are imprecise since the value function component of the target, which is its main driver when rewards are frequently null, is heavily biased at the onset (Kulkarni *et al.*, 2016). Ultimately, MCPG is the only algorithm that was capable of

Algorithm	Average RSQP	p-value	Runtime[1] (hh:mm)
MCPG	$\mathbf{0.8248}$ $(\mathbf{0.0010})$	< 0.01	$00:08$
B-S DH (Baseline)	0.9038 (0.0074)	< 0.01	$00:00$
TD3	0.9144 (0.0086)	< 0.01	$13:24$
PPO	0.9565 (0.0088)	0.25	$08:46$
Dueling DQL*	0.9593 (0.0084)	< 0.01	$08:35$
DDPG	1.0123 (0.0254)	0.15	$19:04$
DQL	1.0230 (0.0138)	< 0.01	$07:31$
DD DQL*	1.0446 (0.0142)	< 0.01	$07:40$
Double DQL	1.0686 (0.0208)		$06:33$

Table 3: Performance of DRL algorithms. Column 2: Average RSQP attained over 10 test sets (standard deviation in parenthesis). Column 3: P-values from t-tests assessing whether the mean RSQP of each algorithm is significantly lower than that of the algorithm listed below it. Column 4: Training time over the training set. The algorithms indicated by stars are the two variants not previously explored in the dynamic hedging literature.

significantly outperforming the B-S DH baseline in the allotted computational budget (0.8248 vs 0.9038). This improvement is statistically significant at the 1% confidence level as indicated by the p-value of < 0.01 from Table 3.

Another advantage of MCPG is the training time. As shown in Table 3, the MCPG algorithm was able to converge in just 8 minutes[1] after early stopping at 21,000 episodes while most other models ran for the whole 500,000 episodes and took between 6 and 19 hours to complete. The Black-Scholes delta hedge strategy remains the most efficient to compute, finishing virtually instantly since it has a closed-form solution (Eq. 9).

PPO. PPO ranks third among the studied algorithms with an RSQP of 0.9565, behind MCPG and TD3. This result is unexpected since, like MCPG, PPO relies on Monte Carlo rollouts. However, PPO uses bootstrapped value function estimates in its policy updates, which introduce bias and can degrade performance, particularly in sparse-reward environments.

DQL algorithms. As shown in Table 3, DQL does not outperform the B–S DH baseline (1.0230 vs. 0.9038). Dueling DQL performs better than DQL (0.9593), while Double DQL performs worse (1.0686). This suggests that identifying states in which actions have limited impact via Dueling DQL is beneficial, whereas addressing DQL's overestimation bias through Double DQL does not provide a clear benefit in the context of option hedging. Combining both approaches in Dueling Double DQL (DD DQL) yields an intermediate RSQP of 1.0446, between DQL and Double DQL. Overall, the performance of the DQL family of algorithms ranks near the bottom of the eight DRL algorithms considered, likely due to its reliance on solely value function estimates to derive its policy.

[1] All experiments are run on an Nvidia A100 GPU.

DDPG algorithms. DDPG ranks fifth in terms of performance with a RSQP of 1.0123, being able to outperform most variants of DQL. This is expected given its ability to natively handle continuous action spaces. In principle, this avoids action discretization and allows for more finely adjusted hedging strategies. Surprisingly, however, it was unable to outperform Dueling DQL, which could be attributed to the increased complexity of DDPG. Unlike DQL, DDPG simultaneously learns both a policy and a state–action value function using two separate neural networks, requiring the training of an additional network.

Twin Delayed DDPG (TD3) addresses the overestimation bias in DDPG by using the minimum of two state–action value estimates. This results in an improvement over DDPG (0.9144 vs. 1.0123), suggesting that overestimation is an issue for DDPG in our setting. TD3 is able to outperform all variants of DQL and its performance is close to that of the B-S DH benchmark (0.9144 vs 0.9038) but still falls short of outperforming it.

5.1 Hyperparameters

As described in Section 4.3, both policy and value functions are approximated using feed-forward neural networks. Hyperparameters were selected via grid search over learning rates $\{0.001, 0.0001, 0.00001\}$, batch sizes $\{64, 128, 256\}$, number of hidden layers in $\{2, 3, 4\}$, and hidden layer sizes in $\{64, 128, 256\}$, resulting in $3^4 = 81$ configurations.

During tuning, MCPG proved highly robust: 73 of the 81 configurations outperformed the baseline on the validation set, with RSQPs ranging from 0.8186 to 0.9733. In contrast, PPO surpassed the baseline for 27 out of the 81 configurations, with RSQPs ranging from 0.8644 to 3.4669. However, it was still unable to outperform the B-S DH benchmark on the test set using the best configuration, showing more sensitivity to hyperparameters and less stable training. Consequently, Table 4 reports the hyperparameter configurations yielding the lowest validation RSQP for each DRL algorithm, these configurations are also used to produce the performance results in Table 3. No other algorithm surpassed the baseline on the validation set within the given computational budget.

Algorithm	Learning Rate	Batch Size	# Hidden Layers	Hidden Size
MCPG	0.00010	256	2	256
TD3	0.00010	128	4	256
PPO	0.00010	64	2	256
Dueling DQL	0.00001	256	4	256
DDPG	0.00001	128	4	256
DQL	0.00001	256	4	128
DD DQL	0.00010	256	3	64
Double DQL	0.00010	128	3	256

Table 4: Optimal hyperparameters of the DRL algorithms neural networks chosen over 81 combinations.

5.2 Hedging Strategy

Figure 2 illustrates a single simulated underlying price path $\{S_t\}_{t=0}^{T}$ generated according to Eq. 7, together with the corresponding hedging positions $\{X_{t+1}\}_{t=0}^{T-1}$ produced by the best-performing DRL algorithms in Table 3 (MCPG, Dueling DQL, PPO, and TD3) and the B–S DH baseline. As shown, both the DRL strategies and the baseline respond to movements in the underlying price. The top-performing DRL methods closely track each other and remain near the B–S DH strategy. DQL exhibits less aligned positions, due to the discretization of the action space.

5.3 Limitations

A key limitation shared by most DRL algorithms, and present in our study, is their high computational cost and strong sensitivity to hyperparameters, which restricts the scope of hyperparameter search. Broader tuning could potentially improve performance, possibly enough to outperform the baseline, whereas most algorithms considered here do not achieve this under current constraints.

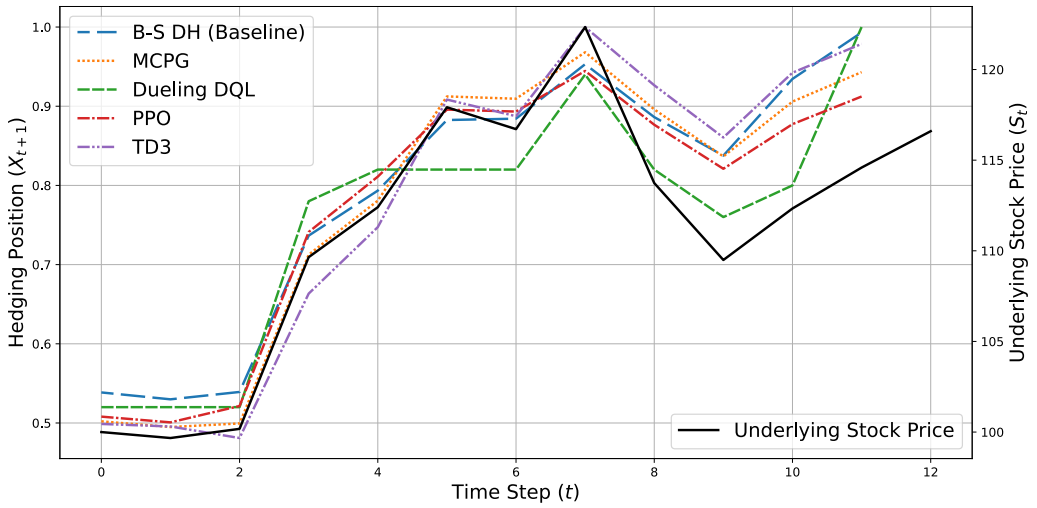

Figure 2: Hedging position $\{X_{t+1}\}_{t=0}^{T-1}$ and underlying stock price $\{S_t\}_{t=0}^{T}$ for one simulated episode.

Several algorithms include hyperparameters that we could not optimize due to computational limitations. Some of which include the target network learning rate $\bar{\beta}$ in DQL, the clipping parameter ϵ in PPO, and the separate learning rates α and β in PPO and DDPG, which could potentially benefiting from independent tuning. As shown in Table 2, MCPG requires tuning only nine hyperparameters, while DQL and PPO involve nearly twice as many. Access to greater computational resources would enable a more thorough exploration of these parameters, likely leading to improved performance.

6 Conclusion and Further Work

This paper advances the literature on DRL for dynamic option hedging in two main ways. First, it provides a comprehensive benchmark by comparing eight DRL algorithms for dynamic hedging, addressing the limited scope of prior studies that typically focus on only one or two methods. Second, it evaluates two algorithmic variants, Dueling DQL and Dueling Double DQL, that, to our knowledge, have not previously been applied to dynamic hedging.

Our results show that Monte Carlo Policy Gradient (MCPG) achieves the strongest risk reduction as measured by RSQP. Notably, MCPG is the only algorithm that outperforms the Black–Scholes delta hedge baseline, while PPO and all variants of DQL and DDPG fail to do so. In addition, MCPG exhibits substantially faster training, with a runtime nearly 50 times shorter than the next fastest algorithm, Double DQL. These findings indicate that MCPG, inspired by (Buehler *et al.*, 2019), is the most effective and practical approach for deep hedging in this setting.

The relatively weak performance of algorithms employing value function estimates can be partly attributed to the reward sparsity of the hedging problem. Future work could explore reward-shaping approaches, such as decomposing the terminal reward into intermediate components as in (Chong *et al.*, 2023), to assess whether denser rewards improve the effectiveness of value function estimates.

Moreover, the hedging environment considered here is low-dimensional, involving a single hedging instrument and a limited state space. Extending the analysis to higher-dimensional settings, such as hedging with multiple instruments or option contracts (Carbonneau, 2021; Carbonneau and Godin, 2024), would help determine whether MCPG's advantage persists in more complex environments.

Finally, implementing the different algorithms studied in this paper for other financial sequential decision-making tasks such as portfolio optimization and optimal execution would allow assessing whether alternative shapes of the value function associated with these tasks could impact the performance of value-based algorithms.

REFERENCES

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

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

Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep hedging of derivatives using reinforcement learning. *Journal of Financial Data Science*, 3(1):10–27, 2020.

Jay Cao, Jacky Chen, Soroush Farghadani, John Hull, Zissis Poulos, Zeyu Wang, and Jun Yuan. Gamma and vega hedging using deep distributional reinforcement learning. *Frontiers in Artificial Intelligence*, 6, 2023.

Alexandre Carbonneau and Frédéric Godin. Equal risk pricing of derivatives with deep hedging. *Quantitative Finance*, 21(4):593–608, 2021.

Alexandre Carbonneau and Frédéric Godin. Deep equal risk pricing of financial derivatives with multiple hedging instruments. *Journal of Computational Finance*, 28(3), 2024.

Alexandre Carbonneau. Deep hedging of long-term financial derivatives. *Insurance: Mathematics and Economics*, 99:327–340, 2021.

Wing Fung Chong, Haoen Cui, and Yuxuan Li. Pseudo-model-free hedging for variable annuities via deep reinforcement learning. *Annals of Actuarial Science*, 17(3):503–546, November 2023.

Jiayi Du, Muyang Jin, Petter N. Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, 2(4):44–57, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48(5):1779–1801, 1993.

Igor Halperin. QLBS: Q-learner in the Black-Scholes (-Merton) worlds. *The Journal of Derivatives*, 28(1):99–122, 2020.

Blanka Horvath, Josef Teichmann, and Žan Žurič. Deep hedging under rough volatility. *Risks*, 9(7):138, 2021.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Saeed Marzban, Erick Delage, and Jonathan Yu-Meng Li. Deep reinforcement learning for option pricing and hedging under dynamic expectile risk measures. *Quantitative Finance*, 23(10):1411–1430, 2023.

Oskari Mikkilä and Juho Kanniainen. Empirical deep hedging. *Quantitative Finance*, 23(1):111–122, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:484–489, 2015.

Andrei Neagu, Frédéric Godin, Clarence Simard, and Leila Kosseim. Deep hedging with market impact. In *The 37th Canadian Conference on Artificial Intelligence (Canadian AI 2024)*, Guelph, Canada, May 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arxiv:1707.06347*, 2017.

Anil Sharma, Freeman Chen, Jaesun Noh, Julio DeJesus, and Mario Schlener. Hedging beyond the mean: A distributional reinforcement learning perspective for hedging portfolios with structured products. *arXiv:2407.10903*, 2024.

Zoran Stoiljkovic. Advanced option pricing and hedging with Q-learning: Performance evaluation of the QLBS algorithm. *Journal of Derivatives*, 32(3), 2025.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. *arXiv:1509.06461*, 2015.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1995–2003. JMLR.org, 2016.