Quantitative Finance

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/rquf20

Empirical deep hedging

Oskari Mikkilä & Juho Kanniainen

To cite this article: Oskari Mikkilä & Juho Kanniainen (2022): Empirical deep hedging, Quantitative Finance, DOI: 10.1080/14697688.2022.2136037

To link to this article: https://doi.org/10.1080/14697688.2022.2136037

Quantitative Finance, 2022
https://doi.org/10.1080/14697688.2022.2136037

Empirical deep hedging

OSKARI MIKKILÄ and JUHO KANNIAINEN*

Group of Financial Computing and Data Analytics, Tampere University, Tampere, Finland

(*Received 1 December 2021; accepted 7 October 2022; published online 31 October 2022*)

Existing hedging strategies are typically based on specific financial models: either the strategies are directly based on a given option pricing model or stock price and volatility models are used indirectly by generating synthetic data on which an agent is trained with reinforcement learning. In this paper, we train an agent in a pure data-driven manner. Particularly, we do not need any specifications on volatility or jump dynamics but use large empirical intra-day data from actual stock and option markets. The agent is trained for the hedging of derivative securities using deep reinforcement learning (DRL) with continuous actions. The training data consists of intra-day option price observations on S&P500 index over 6 years, and top of that, we use other data periods for validation and testing. We have two important empirical results. First, a DRL agent trained using synthetic data generated from a calibrated stochastic volatility model outperforms the classic Black–Scholes delta hedging strategy. Second, and more importantly, we find that a DRL agent, which is empirically trained directly using actual intra-day stock and option prices without the prior specification of the underlying volatility or jump processes, has superior performance compared with the use of synthetic data. This implies that DRL can capture the dynamics of S&P500 from the actual intra-day data and to self-learn how to hedge actual options efficiently.

Keywords: Options; Hedging; Deep reinforcement learning; Stochastic volatility

JEL Classification: G13, C00

1. Introduction

Deep Reinforcement Learning (DRL) has shown its potential automated trading (Deng *et al.* 2016). In the recent literature, the application of DRL for the hedging of options has gained attention. This is not surprising, because hedging is about sequential decisions to maximize long-term rewards (wealth). So far, the research papers have mainly been training and testing DRL agents using synthetic data (Buehler *et al.* 2018, Halperin 2019, Kolm and Ritter 2019, 2020, Cao *et al.* 2021, Halperin 2020, Du *et al.* 2020, Giurca and Borovkova 2021).

Surprisingly, the extant literature on DRL for option hedging uses empirical data quite barely. In this regard, Buehler *et al.* (2018) provide important results by training a DRL model with simulated sample paths generated by an empirically estimated volatility (GARCH) model, achieving good performance. More recently, Giurca and Borovkova (2021) showed that an agent trained by synthetic data can be transferred to the real environment successfully. Both of papers used daily option data on S&P 500 for the estimation of a volatility model to be used for data generation. As in these papers, model-driven assumptions are required about volatility and/or jump dynamics because of the limited size of daily data (see Charpentier *et al.* 2021). Some papers exist for neural-network based market simulators (Wiese *et al.* 2019, 2020) that could be used to tackle the problem of limited size of (daily) data. Nevertheless, we lack research that not only tests but also trains DRL agents using *empirical intra-day option data* from the actual option markets in a data-driven without specifying nor using a volatility and/or jump model for data generation.

In this paper, we aim to address this research gap using the state-of-the-art DRL techniques with intra-day observations on S&P 500 index options. Our question is if DRL can find new data-driven hedging strategies without specifying a model on volatility or jumps. This is fundamentally important because, to our best knowledge, this is the first time in the academic literature the DRL agent is trained empirically without advice about the volatility or jump dynamics of the underlying asset. In that sense, the resulting DRL hedging procedure is free of assumptions on volatility dynamics. The only source of information used to train the agent is the actual option data coming from the markets and the reward function that we specify to be a linear

*Corresponding author. Email: juho.kanniainen@tuni.fi combination of the mean and standard deviation of the wealth.

Delta hedging can be understood as a sequential (and dynamic) optimization problem, where trader's expected cumulative utility is maximized. From this point of view, delta hedging is a very natural problem to be solved by Reinforcement Learning (RL). In general, the idea in RL is to identify a policy, which, by optimal actions in an environment, maximizes the expected discounted cumulative rewards. In the context of hedging, rewards can be determined by two components: return and risk. That is, a trader wants to have a hedging strategy that reduces the risk of her portfolio, but not at any cost. An important feature of RL is that it maximizes the *cumulative* rewards, which leads to a solution that is good not only in the short term but also in the long term.

In RL, the 'machine' employs trial and error to develop an optimal solution to the problem. Once the reward and other settings are determined by a user, the machine can iteratively teach itself based upon the rewards of its actions drawn from the policy. There are alternative algorithms and approaches, but perhaps the most important innovation in RL has been the combination of RL and deep neural networks to capture the optimal policies. It is the most useful in problems with high dimensional state space, which is the case in the context of this paper (for more information about deep reinforcement learning, see, e.g. Henderson *et al.* 2018, François-Lavet *et al.* 2018).

Fischer Black and Myron Scholes derived their option pricing model (Black and Scholes 1973) using argumentation on the delta hedging procedure. By that way, the option holder can remove directional risk related to price movements in the underlying asset. Their model assumes constant volatility and no jumps in returns. In the real world, there are more than one sources of risk, in which case single-instrument hedges remain partial (Naik 1993, Bakshi *et al.* 1997, Tankov and Voltchkova 2009). Moreover, Black–Scholes delta hedging requires continuous trading in the absence of transaction costs. Even if algorithms could rebalance the positions on high frequency almost continuously, transaction costs make it impossible to achieve continuous trading in practice. For these reasons, hedging error and hedging costs exist in the real-world situation and the question is, how to find a single-instrument hedging procedure that yields the best performance.[†]

To proceed systematically towards the main objective, we first test our DRL specification for hedging under Monte Carlo experiments. The benchmark is classic Black–Scholes delta hedging, which, among practitioners, is a popular procedure to take the time-varying volatility into account by using current implied volatility in the calculation of option's delta (see Hull and White 2017, Cao *et al.* 2021), and thus sometimes called as practitioners delta hedging. In the Monte Carlo experiments and in the empirical investigation, the proportional transaction costs are set to 1 BPS, which is relatively low. The motivation for this rather conservative choice is to have settings under which it is a non-trivial exercise to use DRL to beat the naive classic Black–Scholes delta hedging procedure. If the transaction costs were large, then the Black–Scholes delta hedging automatically fails because of frequent trading, while the DRL agent can perform relatively well, even if it was poorly specified and trained.

In the Monte Carlo experiment, we test our DRL agent against classic Black–Scholes delta hedging under the constant and stochastic volatility models. The constant volatility model is used to analyze how well the DRL agent performs when the assumptions are favorable for the classic Black–Scholes delta hedging procedure. This is analyzed by generating synthetic data under constant volatility, then train the DRL agent, and finally test the DRL agent against the delta hedging benchmark with the independent test data (generated by the constant volatility model with the same parameters). As the transaction costs are relatively low, we are happy to see that the performance of the DRL agent is almost as good as that of the delta hedging strategy. This means that the agent was able to capture the idea of delta hedging on its own. Second, to know if our DRL can beat the delta hedging strategy under more realistic settings, we generate synthetic data from the calibrated stochastic volatility under Heston (1993) model. Indeed, we find that the DRL gives us a cutting edge: under stochastic volatility, the DRL clearly outperforms classic Black–Scholes delta hedging, and this is robust across different weights between return and risk in the reward function.

In the empirical part of this paper, we train the DRL agent in two different ways. First, identically to the Monte Carlo experiment, we train the agent with a synthetic data generated using the Heston model, which is calibrated using actual option pricing observations on S&P500. This somewhat corresponds what Giurca and Borovkova (2021) do: training the agent with synthetic data and transferring the optimal policies identified by the synthetic data to a real market environment.[‡] This kind of procedure is called Sim-to-Real Transfer. We find that the agent trained by synthetic data generated from a calibrated Heston model outperforms the classic Black–Scholes delta hedging strategy, which is again robust across different weights between return and risk in the reward function.

To address the *main research objective*, we train the DRL agent using actual empirical observations on option prices written on S&P500 index. Importantly, we do not transfer any information from the agent trained by the synthetic data, but the training starts from scratch to make the agent not to

[†] Another question in derivative hedging is the use of the other derivatives to hedge one under stochastic volatility and other sources of randomness, but in this paper, we focus on hedging with a single instrument only.

[‡] The differences between our paper and other papers that use empirical data, Buehler *et al.* (2018) and Giurca and Borovkova (2021) are that (i) we formulate the calibration of the volatility model as a least square problem and (ii) our calibration data span 6 years of *cross-sectional intraday* option data. In contrast to this, Giurca and Borovkova (2021) estimate the model parameters using daily observations on VIX as a proxy for instantaneous volatility. The advantage of our approach is that it automatically takes the volatility risk-premium into account (if VIX corresponded instantaneous volatility, then volatility risk-premium would be zero). Moreover, Buehler *et al.* (2018) estimate the parameters of the volatility (GARCH) model with maximum likelihood estimation with return data. The main difference between our and their approach is that we have day-specific parameter estimates as no historical data is needed when the model is calibrated with the cross-sectional option data.

be influenced by assumptions on any volatility model. Moreover, the feature set does not include the Black–Scholes Delta, because we do not want to provide any hints about possible hedging strategies for the DRL agent. This means that the agent, fed with the reward function, must learn how to hedge actual options in a real environment properly on its own by trial-and-error. The agent is trained with 6 years of intra-day option data from 2006 to 2011 and then validated and tested using intra-day option price observations from 2012 and 2013, respectively. Importantly, no data used to train the agent were used to report its performance. The results show that the empirically trained DRL agent not only beats the Black–Scholes benchmark but also the Sim-to-Real agent trained by the data generated from the calibrated Heston model. The results are robust across different weights between return and risk in the reward function. This is a strong result: it shows that the DRL agent can autonomously learn to hedge options in without any prior knowledge about possible specifications of the volatility or jump processes. Second, it shows that by intra-day observations, one can collect a sufficient amount of cross-sectional option data for training the DRL agent for hedging purposes. These findings provide the main contribution of this paper.

2. Deep reinforcement learning with continuous action domain

In Reinforcement Learning (RL), by a trial-and-error method in an environment, the goal is to learn a policy that determines optimal actions. RL systems have two entities that interact with each other: an agent and an environment. The optimal action of an agent is defined as an action that maximizes the expected lifetime reward (Silver *et al.* 2014) in light of the observations on the current environment. Roughly speaking, the agent tries out different strategies and observes, which is the best in terms of the expected reward (Sutton and Barto 2018).

There are two types of actions in reinforcement learning problems. Discrete actions have a known, limited space of available actions. Continuous actions might have upper and lower limits, but any action in between can be chosen. For example, a discrete action could be if an agent should turn the car left or right, while a continuous action tells how much to steer the wheel. Therefore, continuous actions are in line with the assumption of frictionless in delta hedging, for which reason this paper is based on the continuous action space. At each time step, an agent observes the state of the environment, $s_t \in \mathcal{S}$, and then selects an action $a_t \in \mathcal{A}$ with respect to its deterministic policy $\mu : \mathcal{S} \to \mathcal{A}$ with parameter vector $\phi \in \mathbb{R}^n$. The problem is modeled as a Markov Decision Process (MDP) with a stationary transition dynamics distribution with conditional density $p(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$, that is, the distribution of future states depends only upon the present state and the present action and not on the past.

After taking an action at time t, the agent receives a reward $r_{t+1} \in \mathbb{R}$, determined by a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and observes a new state of the environment s_{t+1}. The new state is an outcome of an action and external variables that can be stochastic. Then based on s_{t+1}, an agent takes a new action a_{t+1}, receiving reward r_{t+2} and so on. As a result, we have a trajectory $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$ This continues until a terminal state is reached. The period between the start and terminal states is called an episode. An episode can end for different reasons depending on the problem being learned. In a video game, the terminal state is reached when the game is over if the time runs out. In the context of hedging, the episode ends no later than at the maturity of the option. Some problems do not have a natural ending for an episode and instead, an arbitrary limit for the maximum number of time steps can be set.

In RL, the goal is to learn a policy that maximizes the received reward over the *entire episode*. This gives an additional dimension when choosing a correct action: the optimal action in the current state might result in a negative reward immediately but lead to large positive rewards later. On the other hand, an action might lead to a large immediate reward but be disastrous in the long run. The balance between immediate and future rewards is achieved with a familiar method from the world of finance: by discounting the rewards. The total discounted reward from time-step t onwards is defined by

$$\tilde{r}_t = \sum_{i=0}^{\infty} r_{t+i}\gamma^i,$$

where $r_{t+i} = R(s_{t+i-1}, a_{t+i-1})$ is the immediate reward received at time $t+i$ and γ is the discount parameter with $0 < \gamma \leq 1$.

We denote the discount state distribution by

$$\rho^{\mu_\phi}(s'; t) := \int_{\mathcal{S}} \sum_{i=0}^{\infty} \gamma^{t+i} p_t(s) p(s \to s'; t+i, \mu_\phi) \mathrm{d}s,$$

where s' is a state after transitioning for $t+i$ time steps from state s with probability $p(s \to s'; t+i, \mu_\phi)$ and μ_ϕ is the policy function with parameters ϕ. Agent's goal is to maximize the expected cumulative reward, which can be written as

$$\begin{aligned} J(\mu_\phi) &= \int_{\mathcal{S}} \rho^{\mu_\phi}(s; t) r(s, a) \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^{\mu_\phi}} \left[\tilde{r}_t | \mu_\phi \right], \end{aligned}$$

where $a = \mu(s|\phi)$. In RL, the objective is to find the optimal policy μ, with parameters ϕ, which maximizes the expected return $J(\mu_\phi)$. For more information on the performance objective with both stochastic and deterministic policies, we refer to Silver *et al.* (2014).

This paper uses an advanced version of so-called Q-learning procedure, which is does not require a model of the environment. In particular, for Q-learning we use Twin Delayed Deep Deterministic Policy Gradients, TD3, introduced by Fujimoto *et al.* (2018). This method builds on the Temporal Difference (TD) methods. Particularly, on the first hand, it is based on Deterministic Policy Gradient algorithm (DPG) proposed by Silver *et al.* (2014) and, on the other hand, deep Q-learning (Mnih *et al.* 2013). Recently, DPG was used in Giurca and Borovkova (2021) for train the agent for option hedging with synthetic data. TD methods have several advantages as they combine the benefits of Monte Carlo (MC) methods and Dynamic Programming (DP): (i) Similarly to MC methods, TD methods can learn directly from raw experience without a model of the environment's dynamics, and (ii) like in the DP, in the TD, estimates are updated without waiting for a final outcome (they bootstrap). Compared to the DP methods, TDs do not require a model of the environment, which is a very important property to enabling the hedging of the derivatives in a data-driven manner. Moreover, compared to MC methods, TD methods are online and they are implemented in a fully incremental fashion. Thus prediction, action, and learning proceed continuously as the agent interacts with an unknown environment. At the same time, TD methods can avoid the main drawback of MC methods that one must wait until the end of an episode, because only then is the reward known. For that reason, compared to TD, MC methods could substantially delay the training of the model if the episodes are long. Importantly, like the MC methods, TD methods converge asymptotically to the correct predictions (for more information, see, e.g. Sutton and Barto 2018, chapter 6).†

The cornerstone of the modern reinforcement learning algorithms is the action value function. In the fashion of Q-learning (Watkins and Dayan 1992), we define the value of taking action a in state s under a policy μ_ϕ as the expected return starting from state s, taking the action a, and thereafter following policy μ_ϕ:

$$\begin{aligned} Q^{\mu_\phi}(s,a) &= \mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[\tilde{r}_t | s_t = s, a_t = a; \mu_\phi\right] \\ &= r(s_t,a_t) + \gamma\mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[Q^{\mu_\phi}\left(s_{t+1}, \mu(s_{t+1}|\phi)\right)\right], \end{aligned} \quad (1)$$

where a is determined by $\mu_\phi(s)$. By following Mnih *et al.* (2013), in deep Q-learning, for a large state space, the optimal action-value function

$$Q^*(s,a) = \max_\phi Q^{\mu_\phi}(s, \mu(s|\phi))$$

can be approximated by (deep) neural networks with parameters θ. We denote the function approximator with parameters θ by $Q(s,a|\theta)$. In this research, we parameterize an approximate value function using the feedforward neural network specified in section 3.

In actor methods, the policy structure is known as the actor (it is used to select actions) and the estimated value function is known as the critic (it criticizes the actions made by the actor). The policy can be updated through the deterministic policy gradient algorithm. Policy gradients are based on the idea that the policy parameters ϕ are adjusted by calculating the performance gradient. In the case of deterministic policy, the gradient is (see Silver *et al.* 2014, theorem 1)

$$\begin{aligned} \nabla_\phi J(\mu_\phi) &= \int_S \rho^{\mu_\phi}(s)\nabla_\phi\mu(s|\phi)\nabla_a Q_\theta(s,a)|_{a=\mu(s|\phi)}\mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[\nabla_\phi\mu(s|\phi)\nabla_a Q_\theta(s,a)|_{a=\mu(s|\phi)}\right]. \end{aligned} \quad (2)$$

Multiple algorithms have been introduced to estimate the action-value function $Q^{\mu_\phi}(s,a)$. In continuous action space research, perhaps the best known deterministic policy gradient algorithm is the Deep Deterministic Policy Gradients (DDPG) by Lillicrap *et al.* (2015). DDPG introduces similar improvements to deterministic policy gradients as Deep Q-learning (Mnih *et al.* 2013) did to Q-learning, and DDPG is known as the equivalent of Deep Q-Learning for continuous action spaces. DDPG is an off-policy actor-critic algorithm. It learns a Q-function, which is used as the critic. The method considers function approximator parameterized by θ, which is optimized by minimizing

$$L(\theta_i) = \mathbb{E}_{s_i,a_i,r_i,s_{i+1}}\left(Q(s_i,a_i|\theta_i) - y_i\right)^2, \quad (3)$$

where

$$y_i = r(s_i,a_i) + \gamma Q(s_{i+1}, a_i^*|\theta_i^*), \quad (4)$$

where a_i^* is the target action determined below.

A couple of important observations must be made at this point. First, we have a sequence of loss functions $L(\theta_i)$ that changes at each iteration i. Second, in deep-Q-learning (Mnih *et al.* 2013), the neural network is updated using temporal difference parameters θ_i^* and ϕ_i^* are used to compute the target at iteration $i+1$. That is, we create versions for the actor and critic neural network models, $Q(s,a|\theta^*)$ and $\mu(s,|\phi^*)$, to calculate the target values. The target network parameters θ_i^* and ϕ_i^* are held fixed between individual updates.

DDPG estimates the loss function as follows. First, at a given iteration i, noise sampled from a noise process ϵ is added to actor policy: $a_i = \mu(s_i|\phi_i) + \epsilon_i$, where $\epsilon \sim N(0,\sigma)$ with $\sigma > 0$. The target action in (4) is solved accordingly. Second, an action a_i is executed and reward r_i and new state s_{i+1} are observed. The transition tuple (s_i,a_i,r_i,s_{i+1}) is stored in $\mathcal{B}$. Third, a random minibatch of N transitions (s_j,a_j,r_j,s_{j+1}) is sampled from $\mathcal{B}$. The minibatch is used to set

$$y_j = r(s_j,a_j) + \gamma Q(s_{j+1}, a_{j+1}^*|\theta_i^*),$$

with $a_{j+1}^* = \mu(s_{j+1}|\phi_i^*) + \epsilon_j$. This, on the other hand, is used to update the critic by minimizing the loss:

$$L(i) = \frac{1}{N}\Sigma_j\left(Q(s_j,a_j|\theta_i) - y_j\right)^2. \quad (5)$$

In DDPG (Lillicrap *et al.* 2015), the parameters θ_i^* and ϕ_i^* of the target neural networks are updated by

$$\begin{aligned} \theta_{i+1}^* &\leftarrow \tau\theta_i^* + (1-\tau)\theta_i, \\ \phi_{i+1}^* &\leftarrow \tau\phi^* + (1-\tau)\phi_i, \end{aligned} \quad (6)$$

where the network weights are slowly copied into the target networks and the rate of copy is controlled by the parameter $\tau \in [0,1]$. In that way, we can combat drastic changes in the policy and action-value networks.

Even if DDPG gained a lot of attention and popularity for continuous action problems, it performs poorly in some environments and has some issues. The algorithm was unstable in some environments and hyper-parameter sensitive, like

† There is a trade-off between choosing the action that has worked in the past and exploring the action space in hopes of learning something even better. This problem is known as the exploration–exploitation dilemma. Without any exploration, policies can easily fall into local value maximums. For continuous action spaces, exploration can be done by adding a random number from for example the Gaussian distribution.

so many other reinforcement learning algorithms. Most of the problems found in DDPG arose from the estimation of Q-values in the critic network. This leads to the policy exploiting the overestimations (Fujimoto *et al.* 2018). To overcome these issues, an extended version of DDPG, so-called TD3, was introduced by Fujimoto *et al.* (2018) and named as Twin Delayed Deep Deterministic Policy Gradients. TD3 combines some of the greatest achievements introduced in other deep reinforcement learning (DRL) methods, such as using two independent critic networks. The idea for double critic comes from Double Q-learning (Hasselt 2010). The overestimation of Q-values is battled by taking the smaller Q-value from the two critic networks for the target value

$$y_j = r(s_j, a_j) + \gamma \min_{k=1,2} Q\left(s_{j+1}, a^*_{k,j+1}\right), \tag{7}$$

where

$$a^*_{k,j+1} = \mu(s_{j+1}|\theta^*_{k,j}) + \epsilon_{j+1}.$$

For each time step (iteration), we update the pair of critics towards the minimum target value. We use the smaller Q-value for the target, which helps fend off overestimation in the Q-function. Moreover, to smooth possible peaks, the action is clipped to lie in a given action range with $a_{\text{low}} \le a \le a_{\text{high}}$. Consequently, in TD3, the action is determined as

$$a_i = \text{clip}\left(\mu(s_i|\phi_i) + \epsilon_i, a_{\text{low}}, a_{\text{high}}\right),$$

where $\epsilon \sim \text{clip}(N(0,\sigma), -c, c)$ with $c > 0$. The target action is determined consistently. In this paper, we implement this procedure, TD3, which is the state-of-the-art method to train a model with deterministic policy with continuous actions.

3. Proposed model for deep hedging

The reinforcement learning method implemented in this paper is the TD3 described above. Because it is a continuous action space method, similarly to the Black–Scholes hedging, unlimited short selling is allowed. Additionally, the underlying asset can be sold or bought in any fraction as we are operating in the continuous action space, and the tick sizes of the underlying asset are not considered. This combined with a continuous action RL method allows the agent to hedge on any precision.

In this section, we describe the implementation details of the method and the general features of the hedging environment.

3.1. *Reinforcement learning approaches for hedging in the literature*

Among the first papers in the relatively young literature on deep hedging, Buehler *et al.* (2018) provide a framework in the presence of market frictions. They operate under convex risk measures subject to proportional transaction costs. Cao *et al.* (2021) and Giurca and Borovkova (2021) use DDPG to allow for continuous state and action space. The method implemented in this paper, TD3, can be seen as an extension to DDPG. In fact, we tested simpler methods, including vanilla actor critic methods and DDPG, but the complexity and stochasticity of the environment lead to learning problems. These alternative methods seemed to either not converge at all or started to learn and then later diverged. This can occur if the gradients are not behaving well the case of diverging weights.† At the same time, the stabilizing features introduced in TD3 seemed to make it the most viable method. Kolm and Ritter (2019) and Du *et al.* (2020) use methods with discrete action space. Additionally, Halperin (2020) considers both discrete-space and continuous-space versions but with (non-deep) Q-learning without the use of neural networks. The limited precision with discrete action space is not necessarily an issue if the hedged position is small. However, if the position to be hedged grows to thousands or hundreds of thousands of options, having perfect precision with a discrete action method becomes impossible. The continuous action method works for any precision, and assuming convergence of policies should result in more optimal hedging.‡

3.2. *Reward function*

To derive the reward function for this paper, we follow Cao *et al.* (2021) in that we aim to identify parameters of the deep neural network approximators maximize $\mathbb{E}(w_T) - \xi\mathbb{SD}(w_T)$ with $\xi > 0$, where w_T is the wealth at time T. Kolm and Ritter (2019) and Du *et al.* (2020) use almost similar objective function, but used variance instead of standard deviation. In this regard, we tested the version with the use of variance, but the results were clearly more stable and robust for the DRL agent with the use of standard deviation. The reward r_t should be a function of wealth increments (see also Kolm and Ritter 2019, Du *et al.* 2020), and based on that we choose reward such that $\mathbb{E}(r_t) = \mathbb{E}(\Delta w_t) - \xi\mathbb{SD}(\Delta w_T)$. At each step, this, on the other hand, is approximated by

$$r_t = \text{PnL}_t - \xi\,|\text{PnL}_t|, \tag{8}$$

where PnL_t (Profit-and-Loss) is the change in agent's wealth between time steps. More specifically, the PnL_t, the profit or loss from time $t-1$ to t, is

$$\begin{aligned}\text{PnL}_t &= H^O_t\,(C_t - C_{t-1}) + H^S_t\,(S_t - S_{t-1})\\ &\quad - c|S_t\left(H^S_t - H^S_{t-1}\right)|,\end{aligned} \tag{9}$$

where H^O_t is the option position, which equals $-1(+1)$ in the case of a short (long) call option position, C_t the price of the option, H^S_t the holdings in the underlying asset at time t, which is positive (negative) in the case of a short (long) call option position, and S_t the price of the underlying asset at time t.

The objective function essentially becomes a trade-off between transaction costs, captured by c, caused by frequent rebalancing (expected P&L) and risk caused by infrequent rebalancing (the standard deviation of the P&L). If the transactions costs are assumed to be large, then the DRL agent would outperform classic Black–Scholes delta hedging strategy 'too easily', because the DRL agent has the freedom to

† We than the anonymous referee for pointing this out.
‡ In Fujimoto *et al.* (2018), there is proof of convergence for a version of Clipped Double Q-learning, yet for a finite MDP setting.

avoid transaction costs. For that reason, the size of proportional transaction costs, c, is conservatively assumed to equal 1 BPS. P&L is largely assumed to come from transaction costs on the underlying stock. To quantify the hedging performance of the policy, we monitor it against the Black–Scholes delta hedging benchmark.

3.3. *State observation*

Determining the features to capture the state of the environment is critical, as the actions determined by them (with a neural network approximator). In this regard, Kolm and Ritter (2019), Cao *et al.* (2021), and Du *et al.* (2020) use information about the moneyness, time to maturity, and current number of shares held. In this paper, we use the same features, but on top of them, the fourth feature is Black–Scholes implied volatility. In fact, it is quite crucial to include the implied volatility to the state variables, because then the DRL algorithm is provided the same information we need for the calculation of Black–Scholes delta. As the neural networks are universal function approximators, the Black–Scholes delta hedging can be considered as a special case of our model. However, to learn a policy independent of any pricing model and to see if DRL can learn a hedging policy independently, we do not include delta and other Greeks. Instead, by having a sufficient amount of trading data, the agent should learn them from the input variables if needed. Additionally, the performance of previous steps does not matter in choosing the optimal action, which is a function of only the current state observation, and the agent's estimate on the future states.

3.4. *State featurization*

Featurization can be used to improve the ability of neural networks to learn the underlying patterns (Ekenel and Stiefelhagen 2006). By featurization, the input data are run through a featurization function engineered to a specific problem. Polynomial, interaction and normalization techniques were used in this paper to improve the hedging performance.

With non-linear models, a feature with higher order of magnitude might dominate other features, for which reason the input variables are normalized. Normalization has been shown to increase the performance of neural networks (Ekenel and Stiefelhagen 2006, Passalis *et al.* 2019), by improving the predicting accuracy or reducing learning times, often achieving both. In this paper, we implement z-score normalization with the skicit-learn library (Pedregosa *et al.* 2011).

3.5. *Hyperparameters*

In delta hedging, we set the discrete-time interval to 60 minutes, which equals seven re-balances a day. Episode length will be set to 35 time-steps, and as there are 7 steps in a day, the length is 5 trading days. In the existing literature, Kolm and Ritter (2019) use episode lengths of 10 days with 5 rebalances a day. In contrast, the rebalance frequency in Cao *et al.* (2021) is substantially longer, from daily to weekly.

The implementation of the TD3 algorithm is based on the paper that introduced TD3 (Fujimoto *et al.* 2018) (the original implementation is available at Fujimoto *et al.* (2020)). For TD3 hyper parameters, the implementation of Fujimoto *et al.* (2018) used a τ of 0.005, and a policy update frequency of 2. Tau controls how fast the target networks are updated and policy update frequency controls how delayed the policy update is. Implementation in this paper uses a tau of 0.001 for slower target network update and a policy update frequency of 2.

When using a continuous action RL method with a stochastic policy, noise should be added for action space exploration. TD3 has built-in source for noise, as it adds noise to the targets. On top of that, we add noise to the agent's action from a normal distribution with standard deviation between 0.2 and 0.7. The standard deviation is high in the beginning for fast exploration and decreases when the performance increases. The hyper parameters were selected empirically by assessing both the performance and the speed of convergence of different parameter sets, and by selecting the one that performed sufficiently.

3.6. *Actor and critic neural networks*

The original TD3 implementation uses relatively simple structures for the actor and critic networks. Both the actor and critic consist of two hidden layers of size 256 and the activation function used in the paper is Rectified Linear Unit ('ReLU'). For the critic, we add another hidden layer for a total of three hidden layers. Layer size is close to the original one, 250 for each hidden layer. The advantage of adding another layer seemed to provide a marginal improvement on the performance. For the activation functions of the hidden layers, we use Leaky ReLU. Leaky ReLU allows a small leakage for negative values (Maas *et al.* 2013). The amount of leakage is controlled by a hyper parameter α:

$$h(i) = \begin{cases} i & \text{if } i > 0 \\ \alpha \times i & \text{otherwise.} \end{cases} \tag{10}$$

When $\alpha = 0$ the function becomes ReLU. Common values for the leakiness parameter are between 0.05 and 0.10 (Maas *et al.* 2013). Leaky ReLU was chosen, as it has been shown to improve neural networks' performance (Xu *et al.* 2015). Empirical tests conducted suggested that Leaky ReLU works better than ReLU, and that 0.05 is suitable for the leakyness parameter.

The activation function for the actor's output layer is hyperbolic tangent. Hyperbolic tangent outputs a value from a range of $[-1, 1]$, which is the limited position the agent is allowed to take in the underlying asset (with a long and short positions to a range of $[0, 1]$ and $[-1, 0]$, respectively). This is then later scaled outside the neural network to a range of $[0, 1]$. For the critic network, the output layer outputs a single value with linear activation.

The learning rate in both the actor and critic network is relatively low, $1\mathrm{E}-04$. Together with a small learning rate and a high batch size of 10,000 a stabilizing effect is achieved. The batch size of 10,000 is larger than usually used in TD3 or DDPG implementations, where the usual batch size is between 64 and 256 (Lillicrap *et al.* 2015, Fujimoto *et al.* 2018). Batch sizes of this size are not unheard of for continuous action problems, and used regularly in algorithms like TRPO by Schulman *et al.* (2015). Adam is used as an optimizer.

4. Data and results

4.1. Data

In the empirical part of this paper, we use S&P 500 index intraday options data between January 3, 2006, and December 31, 2013 (2009 trading days), provided by CBOE Livevol. The raw option data is on minute-by-minute bases, but as the rebalance interval is 60 minutes, hourly observations are used. For each option quote, the following information is available: the value of the underlying index, strike, maturity date, call-put flag, bid price, bid quantity, ask price and ask quantity. The P&L in (9) is calculated from the mid-price of the option. The roots considered are with the codes used by CBOE in the calculation of VIX.† The risk-free rate used is the 1 year point of the U.S. Treasury Yield Curve, obtained from U.S. Department of the Treasury.‡

As a preliminary step, options with clearly misreported quotes are excluded and a few ambiguous records consisting of the same type of options sharing the same timestamp, strike, and maturity but having different quotes have been removed. On 3 days, there were issues related to the underlying price being equal to zero, in which case those observations were removed. Moreover, in a very few cases there were different underlying prices reported for the same time stamps, the median value was used. Table 1 shows an example of the cleaned data used in this paper.

Real-life applications could hedge S&P 500 options with any instruments that track the index, e.g. exchange-traded funds. On the other hand, in the existing literature, the index value has been an accepted proxy for option hedges (e.g. Bakshi *et al.* 1997), which we also use in this paper. In fact, this is a common assumption in the literature on the pricing of index options (that is, the pricing models assume that the index options can be hedged by the underlying asset). We assume that the underlying asset can be sold or bought in any fractions, and the tick sizes of the underlying asset are not considered. As mentioned earlier, the transaction costs are assumed to equal 1 BPS.

The data are divided between the following data sets, which are summarized in table 2:

- Training data that span from 2006 to 2011, the validation data. These data are used for the empirical training of the model.
- Validation data observed from 2012, used selecting the best model to be evaluated with the independent test data.
- Test data from 2013, by which we evaluate and report the performance on the DRL model (against the classic Black–Scholes delta hedging benchmark).

The training data include options with moneyness $0.83 \leq S/K \leq 1.17$. We do not use deep-out/in-the money options because delta of the option is known to move the most when the option is at-the-money, i.e. the gamma of the option is at its highest. The larger the movements in the delta between each rebalancing step is, the harder the option is to hedge in the market with frictions. In this regard, both Kolm and Ritter (2019) and Cao *et al.* (2021) consider at-the-money options, and recently Du *et al.* (2020) developed a method that can be used to hedge with a whole range of strikes. Regarding the maturity times in training data, we use options that will expire in 10–90 days, which corresponds to the maturity times used in Cao *et al.* (2021). In contrast, Kolm and Ritter (2019) consider options with expiry in 10 days and Du *et al.* (2020) use options 'close to maturity'. The properties of the options considered here are in no way constraints to the hedging ability of the agent: the algorithm can be used to hedge a short or long position in a call or put option. Moreover, in principle, the time to expiry can be longer or shorter than in this study, and the option can be as much out-of-money or in-the-money as one wants, yet the good performance level is not guaranteed without re-training the agent.

For empirical validation and testing, the target strikes are $S/K \in [0.85, 0.925, 1, 1.075, 1.15]$ and target maturity times are 10, 30, and 60 days. Options that are closest to target strikes and maturities (in percentage terms) are included. For that reason, the option which is being hedged can change every week. As the hedging period is 5 trading days, options always have expiry after the hedging period ends. This limit is chosen to ensure that none of the hedging periods is cut short due to the option maturing.

4.2. Volatility models for Monte Carlo experiments

In the Monte Carlo experiment, we generate synthetic data from both a constant volatility model and Heston model (Heston 1993), which we use to train the DRL agent. First, with constant volatility, we ensure that the DRL agent can achieve satisfactory performance compared to classic Black–Scholes delta hedging. Second, to generate realistic data for Monte Carlo experiments, the Heston model is calibrated using empirical option price observations. Particularly, synthetic stock price and option data is generated from the model, which is calibrated using 2012 empirical option data on daily bases to train the DRL model to hedge options under different circumstances. In the calibration, the loss function is the squared difference in model and market prices. Table 3 reports the summary statistics on parameter estimates. In the Monte Carlo experiment, we randomly select the set of Heston parameter estimates from the daily calibrations in 2012. Here the question is if the DRL agent can *outperform* the Black–Scholes hedge under stochastic volatility. Moreover, we also analyze the empirical performance of a DRL agent trained with synthetic data. In particular, we first train an agent with a synthetic data generated by the Heston model and

† SPB, SPQ, SPT, SPV, SPX, SPZ, SVP, SXB, SXM, SXY, SXZ, SYG, SYU, SYV, and SZP.
‡ https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yield.

Table 1. Snapshot of option data.

Time	Code	Underlying	Strike	Days to mat.	Rate	Call price	Call bid	Put price	Put bid
2006-Jan-03 09:31	SPX	1253.00	800	18	0.0438	454.45	454.2	0.1	0
2006-Jan-03 09:31	SPX	1253.00	825	18	0.0438	429.45	429.2	0.1	0
…			…						
2006-Jan-03 09:31	SXZ	1253.00	1400	18	0.0438	0.1	0	144.25	144
2006-Jan-03 09:31	SXM	1253.00	1500	18	0.0438	0.1	0	244.05	243.8
2006-Jan-03 09:31	SPX	1253.00	800	46	0.0456	455.05	454.8	0.1	0
2006-Jan-03 09:31	SPX	1253.00	850	46	0.0456	405.35	405.1	0.1	0
…				…					
2006-Jan-03 09:31	SPB	1253.00	1600	347	0.0478	0.65	0.55	297.85	297.6
2006-Jan-03 09:32	SPX	1253.72	800	18	0.0438	455.15	454.9	0.1	0
…									
2006-Jan-03 16:00	SPB	1268.80	1600	347	0.0478	0.75	0.5	284.4	283.4

Table 2. Summary statistics on the number of unique option contracts and 5-day paths for training, validation, and test data.

	Training data (2006–2011)	Validation data (2012)	Test data (2013)
Unique options	7228	1539	1683
Unique five-day paths	12,298	2703	2914

Table 3. Summary statistics on the estimated parameters obtained from daily calibrations of Heston (1993) model in 2012. There were totally 249 trading days in 2012. We use the following notation for the Heston model: $\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sqrt{v_t} S_t \mathrm{d}W_{1,t}$, $\mathrm{d}v_t = \kappa(v_t - \theta)\mathrm{d}t + \eta\sqrt{v_t}\mathrm{d}W_{2,t}$, where $\kappa > 0, \theta > 0, \eta > 0$, and $\mathrm{d}W_{1,t}\mathrm{d}W_{2,t} = \rho \mathrm{d}t$, $\rho \in [-1, 1]$.

	θ	κ	η	ρ	v_0
Min	0.02	0.07	0.04	−1.00	0.01
Max	0.25	9.49	1.00	−0.45	0.08
Median	0.04	3.08	0.56	−0.78	0.02
Average	0.04	3.41	0.62	−0.79	0.03
Standard deviation	0.02	1.60	0.31	0.16	0.01

then analyze how the agent performs with the real-world data, along with Giurca and Borovkova (2021). This is so-called Sim-to-Real reinforcement learning, where the methods of transferring the agent's policy from simulation to real life (Peng *et al.* 2017).

4.3. *Results from Monte Carlo experiments*

In this section, to test our deep hedging framework, we conduct Monte Carlo experiments under (i) geometric Brownian motion and (ii) (Heston 1993) stochastic volatility model. All tests displayed are out of sample simulations. In the next section, we will use a simplified version of transfer learning to train the agent in a Heston model-simulated environment and test with real empirical SPX option data, and, on top of that, we train and test the model with empirical data without any assumptions about volatility dynamics.

To assess the hedging performance of the agent and the benchmarks, we will consider four values from the tests: (i) the mean episode P&L is the average of the total P&L over each 5-day episode; (ii) the episode P&L standard deviation is the standard deviation of the episode total P&Ls over the 10,000 tests; (iii) mean episode transaction costs over each 5-day episode; (iv) average rewards accumulated, as determined in equation (8).

Table 4 reports the results from the Monte Carlo experiments with *constant volatility* of 25% for $\xi \in 1, 2, 3$ based on the use of 10,000 simulated stock price paths for each test. With constant volatility, the starting point is when the transaction costs are not large (which is the case in this paper), the Black–Scholes delta hedging should theoretically be close to optimal. Hence, not any hedging strategy, including the use of DRL agent, should outperform classic Black–Scholes delta hedging. Rather, the question is how well the DRL model can perform compared to the standard delta hedging procedure. In this regard, the results on constant volatility with $\xi = 1$ reported in the first row of Panel A show that in terms of the performance, the agent trained by DRL accomplishes the hedging of options surprisingly well without using any information about the dynamics of the underlying asset. The algorithm automatically learns the tradeoff of hedging costs and the variance, whether the perfect hedging is possible or not due to the existence of transaction costs. When larger values for the risk-return coefficient, ξ, are applied, a gap between the classic delta hedging and the DRL agent slightly increases, yet remain quite small. Overall, the Monte Carlo experiment with constant volatility shows that the DRL with continuous action space can yield quite consistent results with Black–Scholes delta hedging.

Next, table 5 compares the performance the DRL agent against the Black–Scholes hedging from Monte Carlo experiment with *stochastic volatility*. In the presence of stochastic volatility, in which case there are more than one source of risk, single-instrument hedges can only be partial (Bakshi *et al.* 1997). In this case, classic Black–Scholes delta hedging does not minimize the variance of changes in the value of a trader's position, not even theoretically when there is a non-zero correlation between returns and volatility of the underlying security (Hull and White 2017). Indeed, this can be observed from table 5. For example, whereas with $\xi = 2$, the reward for classical classic Black–Scholes delta hedging is 7.17 with constant volatility (see table 4, Panel B), it decreases to –9.19 with stochastic volatility (see table 5, Panel B). More importantly, *when volatility is stochastic, the DRL agent trained with the synthetic data from the Heston model*

Table 4. Deep reinforcement learning (DRL) agent's performance and hedging cost against classic Black–Scholes delta hedging benchmark under *constant volatility*. The hedge period is 5 days and the option is a call option expiring in 2 weeks. A hedge is rebalanced seven times a day. Results are reported on 10,000 out-of-sample simulations. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods. Panels A–C report the results for $\xi = 1, 2, 3$. Here, as described by (8), greater ξ gives more weight to minimize the risk (the standard deviation of wealth).

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black–Scholes Delta Hedging	−0.0119%	0.0426%	−0.0094%	**8.79**
DRL Agent	−0.0124%	0.0459%	−0.0094%	8.60
		Panel B: $\xi = 2$		
Black–Scholes delta hedging	−0.0114%	0.0436%	−0.0095%	**7.17**
DRL Agent	−0.0107%	0.0490%	−0.0094%	6.74
		Panel C: $\xi = 3$		
Black–Scholes delta hedging	−0.0108%	0.0428%	−0.0094%	**5.59**
DRL Agent	−0.0112%	0.0478%	−0.0094%	4.90

Table 5. Deep reinforcement learning (DRL) agent's performance and hedging cost against classic Black–Scholes delta hedging benchmark under *stochastic volatility*. The DRL agent is trained using synthetic data from the Heston model. The hedge period is 5 days and the option is a call option expiring in 2 weeks. Hedge is rebalanced seven times in a day. Results are reported on 10,000 out-of-sample simulations based on Heston (1993) model, which was calibrated with empirical option data. The parameter estimates for a given simulation were randomly selected from the daily calibrations in 2012. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods. Panels A–C report the results for $\xi = 1, 2, 3$. Here, as described by (8), greater ξ gives more weight to minimize the risk (the standard deviation of wealth).

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black–Scholes delta hedging	−0.0086%	0.2628%	−0.0084%	0.42
DRL Agent	−0.0072%	0.2343%	−0.0077%	**1.66**
		Panel B: $\xi = 2$		
Black–Scholes delta hedging	−0.0149%	0.2551%	−0.0084%	−9.19
DRL Agent	−0.0157%	0.2263%	−0.0080%	− **6.82**
		Panel C: $\xi = 3$		
Black–Scholes delta hedging	−0.0104%	0.2606%	−0.0084%	−19.44
DRL Agent	−0.0106%	0.2301%	−0.0079%	− **15.97**

clearly outperforms the classic Black–Scholes delta hedging. Particularly, the use of DRL yields a hedging model that leads to the clearly lower variance of hedger's wealth, which yields larger rewards for all the $\xi \in \{1, 2, 3\}$, yet the differences are substantial especially with $\xi = 3$, which is intuitive because then more weight is given to minimize the portfolio risk.

Overall, the Monte Carlo experiment shows that when volatility is constant and thus the most important assumptions of Black–Scholes hedging procedure are met, the DRL agent can accomplish the hedging relatively well compared. And more importantly, when stochastic volatility is introduced, then the DRL agent clearly outperforms the classical Black–Scholes hedging procedure.

4.4. *Empirical performance*

In this section, we evaluate our empirical performance of DRL models that are trained by two different ways:

(i) We train the DRL agent using synthetic data from the calibrated Heston model (Heston 1993) and test the trained agent with empirical option data (this model is used for Monte Carlo experiment, see table 5 in section 4.3).
(ii) We train and test the DRL agent using empirical stock and option price observations instead of the data generated from a model.

Here the latter represents an agent, which is trained with actual empirical data only without any assumptions about volatility dynamics.

There are advantages of using synthetic data simulated with stochastic volatility for training agents intended for real-life usage, especially because simulations can be ran as many times as needed. However, the policy an agent learns is specific to the simulated environment, and thus is dependent on the volatility model used. Therefore, a problem of how to transfer the learning to real life arises. Sim-to-Real reinforcement learning studies the methods of transferring the agent's policy from simulation to real life (Peng *et al.* 2017). In this paper, we will apply the most basic form of Sim-to-Real reinforcement learning: we empirically test the model trained with simulated. The test data are from 2013, in which there are 1683 options. From that population, we select options with the maturities of 5–70 days and strikes of 0.825–1.175. By that way, the final test sample consists of 419 options for which there are observations for the 5-day trajectories. Consequently, we have 419 options × 5 trading days/option × 7 steps/trading day = 14,665 steps

Second, we use empirical option and stock price data not only to test the performance but to train the agent independently on any volatility model. In that sense, our we train a completely data-driven DRL agent. The existing literature on DRL hedging has purely used simulated data and this paper, to our best knowledge, is the first research that trains and

Table 6. Empirical performance of DRL agent and classic Black–Scholes delta hedging, which serves as a benchmark. The hedge period is 5 days. Hedge is rebalanced seven times in a day. Results are reported in panels A, B, and C for different levels of the risk-return tradeoff parameter $\xi = 1, 2, 3$, respectively. The results are available for (i) classic Black–Scholes delta hedging, for (ii) an agent trained by synthetic data simulated from calibrated Heston models, and (iii) an agent trained by the actual empirical data on option and stock prices. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods.

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black–Scholes Delta Hedging	−0.0195%	0.1515%	−0.0076%	4.204
Sim-to-Real DRL Agent trained with Heston model	−0.0095%	0.1502%	−0.0068%	4.375
DRL Agent trained with empirical data	−0.0123%	0.1415%	−0.0072%	**4.466**
		Panel B: $\xi = 2$		
Black–Scholes Delta Hedging	−0.0195%	0.1515%	−0.0076%	−1.896
Sim-to-Real DRL Agent trained with Heston model	−0.0140%	0.1483%	−0.0072%	−1.423
DRL Agent trained with empirical data	−0.0112%	0.1378%	−0.0071%	− **1.361**
		Panel C: $\xi = 3$		
Black–Scholes Delta Hedging	−0.0195%	0.1515%	−0.0076%	−7.997
Sim-to-Real DRL Agent trained with Heston model	−0.0122%	0.1499%	−0.0075%	−7.386
DRL Agent trained with empirical data	−0.0141%	0.1402%	−0.0072%	− **7.272**

evaluates the DRL agents with actual option market data. To train the model, we use option data from 2006 to 2011 (see table 2). Particularly, option and stock price trajectories of 5 days are used to train the model. Totally, there are 12,298 unique 5-day periods for training (see section 4.1), which was achieved by using different strikes and expiry days for the same time period. This amount of data is found to be sufficient with TD3, which uses experience replay, i.e. a replay memory technique (Mnih *et al.* 2013). By that way, a tuple (s_i, a_i, r_i, s_{i+1}) can be used more than once by random sampling, thus improving data efficiency. Second, this method addresses the problem of a high correlation between consecutive steps in the environment making the neural network more robust.

Table 6 reports results about the empirical performance of agent-hedging and classic Black–Scholes delta hedging. The results are very favorable for the DRL models: the empirically trained DRL agent outperforms not only the classical classic Black–Scholes delta hedging but also the DRL agent trained by synthetic data generated by the calibrated Heston model. The result is robust with all the values of $\xi \in \{1, 2, 3\}$.

Second, the DRL agent trained with the synthetic Heston data always yields higher returns compared to the classic Black–Scholes delta hedging. This finding suggests that even if no option data is sufficiently available for the training of a DRL agent, a simple but effective solution is to transfer the agent's policy from simulation to real markets. The idea of the use of synthetic data in real-life financial applications has been recently developed further (see, e.g. Da Silva and Shi 2019, Kondratyev and Schwarz 2019, Wiese *et al.* 2020), and was recently applied in Giurca and Borovkova (2021) with the applications to the option hedging procedures. This is a very encouraging result, because it suggests that practitioners could start to implement RL-based techniques even if there is no sufficient amount of empirical data available to train the models.

Moreover, figure 1 illustrates a sample trajectory on the training of the DRL agent with the empirical data with $\xi = 1$. The performance (reward) is calculated with the validation data and it is plotted against the episodes. The figure uses log-scale for the episodes to zoom into the beginning of the training process. Totally, the training was performed over 100,000 episodes, but, in fact, DRL model reached the classic Black–Scholes delta hedging already at the 1400th episode. That is, the training procedure was very efficient and fast. The results are consistent with different values of ξ as well as with the synthetic data simulated with the Heston model.

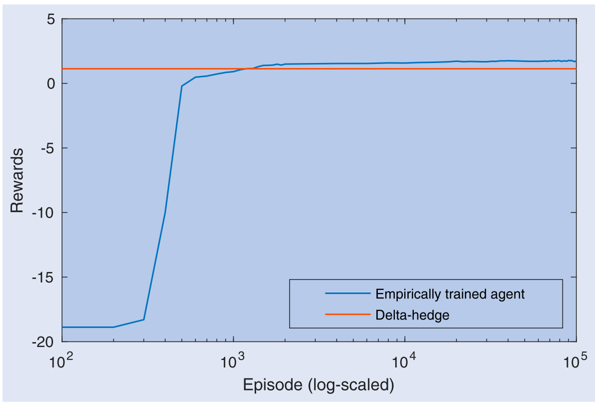

Figure 1. The performance (rewards) of empirically trained agent against the validation data over the episodes with $\xi = 1$. The performance is compared against the classic Black–Scholes delta hedging strategy over. The performance is plotted against the log-scaled episodes.

4.5. *Public data and codes*

We have made the synthetic data and codes publicly available at https://github.com/oskarimikkila/Empirical-Deep-Hedging. As the option data are available at CBOE's LiveVol with a paywall (see https://www.livevol.com/stock-options-analysis-data/), we are not allowed to publish the actual empirical data used to train the agent empirically. The published codes and synthetic data can be used to reproduce results for Monte Carlo experiment.

5. Conclusion

To our best knowledge, this is the first paper that uses intraday option data from actual markets to both train and test the self-learning hedging algorithm based on the Deep Reinforcement Learning (DRL). Our research provides two important contributions. First, the results confirmed that the DRL agent is transferable from a simulated environment to reality. That is, the agent was able to learn a policy from synthetic option data generated by empirically calibrated Heston model that work on both simulated and actual environments. This was verified with an extensive data set. This finding is consistent with the evidence provided by Giurca and Borovkova (2021), who conclude that the use of Reinforcement Learning is suitable for traders taking real-life hedging decisions when the agents are trained on synthetic data.

Second, and most importantly, to our best knowledge, this is the first paper showing that it is possible to successfully train an empirical DRL agent for derivative hedging with option data from actual markets, obtaining even better performance than training the agent with synthetic data. The agent is free of volatility models in the sense that no specifications on volatility or jumps were provided. The hedging based on the empirical agent we call *Empirical Deep Hedging*, and we found that it yields consistently better performance than the use of simulated data from stochastic volatility model and also clearly outperforms the classic Black–Scholes delta hedging. Importantly, this research provides strong evidence that the DRL agent can self-learn to capture the actual properties of stock price and volatility processed and to use them for hedging in a real-life environment.

The results of this paper are important not only academically but also in practice as we provide evidence that practitioners can develop their hedging strategies using empirical self-learning DRL agents in and achieve superior performance compared to the classic Black–Scholes delta hedging procedure. In our future research, we plan to develop DRL agents that exploit statistical arbitrage opportunities in option markets by identifying over/under priced options and deep-hedging the position until a point where the profits are the best to be realized.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Bakshi, G., Cao, C. and Chen, Z., Empirical performance of alternative option pricing models. *J. Finance*, 1997, **52**, 2003–2049.doi:10.1111/j.1540-6261.1997.tb02749.x

Black, F. and Scholes, M., The pricing of options and corporate liabilities. *J. Polit. Econ.*, 1973, **81**, 637–654.

Buehler, H., Gonon, L., Teichmann, J. and Wood, B., Deep hedging. *Quant. Finance*, 2018, **19**, 1271–1291.

Cao, J., Chen, J., Hull, J. and Poulos, Z., Deep hedging of derivatives using reinforcement learning. *J. Financ. Data Sci.*, 2021, **3**, 10–27.

Charpentier, A., Elie, R. and Remlinger, C., Reinforcement learning in economics and finance. *Comput. Econ.*, 2021, **32**, 1–38.

Da Silva, B. and Shi, S.S., Style transfer with time series: Generating synthetic financial data. arXiv preprint arXiv:1906.03232.2019.

Deng, Y., Bao, F., Kong, Y., Ren, Z. and Dai, Q., Deep direct reinforcement learning for financial signal representation and trading. *IEEE Trans. Neural Netw. Learn. Syst.*, 2016, **28**, 653–664.

Du, J., Jin, M., Kolm, P.N., Ritter, G., Wang, Y. and Zhang, B., Deep reinforcement learning for option replication and hedging. *J. Financ. Data Sci.*, 2020, **2**, 44–57.

Ekenel, H.K. and Stiefelhagen, R., Analysis of local appearance-based face recognition: Effects of feature selection and feature normalization. In: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pp. 34–34, 2006.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G. and Pineau, J., An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560, 2018.

Fujimoto, S., Hoof, H. and Meger, D., Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning, PMLR*, pp. 1587–1596, 2018.

Fujimoto, S., van Hoof, H. and Meger, D., Github: Addressing function approximation error in actor-critic methods. https://github.com/sfujim/TD3, 2020.

Giurca, A. and Borovkova, S., Delta hedging of derivatives using deep reinforcement learning. Available at SRRN 3847272. 2021.

Halperin, I., The QLBS Q-learner goes NuQLear: Fitted Q iteration, inverse RL, and option portfolios. *Quant. Finance*, 2019, **19**, 1543–1553.

Halperin, I., QLBS: Q-Learner in the Black–Scholes (–Merton) worlds. *J. Deriv.*, 2020, **28**, 99–122.

Hasselt, H., Double Q-learning. *Adv. Neural Inf. Process. Syst.*, 2010, **23**, 2613–2621.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D. and Meger, D., Deep reinforcement learning that matters. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Heston, L.S., A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Rev. Financ. Stud.*, 1993, **6**, 327–343.

Hull, J. and White, A., Optimal delta hedging for options. *J. Bank. Finance*, 2017, **82**, 180–190.

Kolm, P.N. and Ritter, G., Modern perspectives on reinforcement learning in finance. SSRN Working Paper, 2020.

Kolm, P.N. and Ritter, G., Dynamic replication and hedging: A reinforcement learning approach. *J. Financ. Data Sci.*, 2019, **1**, 159–171.

Kondratyev, A. and Schwarz, C., The market generator. Available at SSRN 3384948. 2019.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

Maas, A.L., Hannun, A.Y. and Ng, A.Y., Rectifier nonlinearities improve neural network acoustic models. In: *Proc. icml, Citeseer*, p. 3, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.A., Playing atari with deep reinforcement learning. CoRR http://arxiv.org/abs/1312.5602, 2013.

Naik, V., Option valuation and hedging strategies with jumps in the volatility of asset returns. *J. Finance*, 1993, **48**, 1969–1984.

Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Deep adaptive input normalization for time series forecasting. *IEEE. Trans. Neural Netw. Learn. Syst.*, 2019, **31**, 3760–3765.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.

Peng, X.B., Andrychowicz, M., Zaremba, W. and Abbeel, P., Sim-to-real transfer of robotic control with dynamics randomization. CoRR abs/1710.06537. http://arxiv.org/abs/1710.06537, arXiv:1710.06537, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P., Trust region policy optimization. In *International Conference on Machine Learning, PMLR*, pp. 1889–1897, 2015.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M., Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International*

Conference on Machine Learning – Volume 32, JMLR.org, pp. I–387–I–395, 2014.

Sutton, R.S. and Barto, A.G., *Reinforcement Learning: An Introduction*, 2018 (A Bradford Book).

Tankov, P. and Voltchkova, E., Asymptotic analysis of hedging errors in models with jumps. *Stoch. Process. Their Appl.*, 2009, **119**, 2004–2027.

Watkins, C.J. and Dayan, P., Q-learning. *Mach. Learn.*, 1992, **8**, 279–292.

Wiese, M., Bai, L., Wood, B. and Buehler, H., Deep hedging: Learning to simulate equity option markets. arXiv preprint arXiv:1911.01700, 2019.

Wiese, M., Knobloch, R., Korn, R. and Kretschmer, P., Quant gans: Deep generation of financial time series. *Quant. Finance*, 2020, **20**, 1419–1440.

Xu, B., Wang, N., Chen, T. and Li, M., Empirical evaluation of rectified activations in convolutional network. ArXiv abs/1505.00853, 2015.