Neural Computing and Applications (2020) 32:17229–17244
https://doi.org/10.1007/s00521-020-05359-8

S.I. : EMERGING APPLICATIONS OF DEEP LEARNING AND SPIKING ANN

A deep Q-learning portfolio management framework for the cryptocurrency market

Giorgio Lucarelli[1] · Matteo Borrotti[1,2]

Received: 21 November 2019 / Accepted: 9 September 2020 / Published online: 20 September 2020

Abstract
Deep reinforcement learning is gaining popularity in many different fields. An interesting sector is related to the definition of dynamic decision-making systems. A possible example is dynamic portfolio optimization, where an agent has to continuously reallocate an amount of fund into a number of different financial assets with the final goal of maximizing return and minimizing risk. In this work, a novel deep Q-learning portfolio management framework is proposed. The framework is composed by two elements: a set of local agents that learn assets behaviours and a global agent that describes the global reward function. The framework is tested on a crypto portfolio composed by four cryptocurrencies. Based on our results, the deep reinforcement portfolio management framework has proven to be a promising approach for dynamic portfolio optimization.

Keywords Deep reinforcement learning · Q-learning · Portfolio management · Dueling double deep Q-networks

1 Introduction

Nowadays, new developments in Machine Learning (ML) and advancements in neuroscience together with an increasing amount of data and a new generation of computers are ushering in a new age of Artificial Intelligence (AI). Currently, AI researchers possess a rising interest in a collection of powerful techniques that fall under the umbrella of deep Reinforcement Learning (RL) [4].

The success of deep RL is related to the fact that both biological and artificial agents must achieve goals to survive and be useful. This goal-oriented behaviour is the milestone of RL. Such behaviour is based on learning actions that maximize rewards and minimize punishments or losses. RL relies on interactions between an agent and its environment. The agent must choose actions based on a set of inputs, where the inputs define the states of the environment. The agent tries to optimize the outcomes of these actions over time, which can be either rewards or punishments. This formulation is natural in biological systems, but it has also proven to be highly useful for artificial agents [22]. In fact, the combination of representation learning with goal-oriented behaviour gives deep RL an inherent interest for many different applications.

Deep RL approaches have been successfully applied to a range of fields that vary from image understanding [6] to natural language processing [32]. For example, deep RL has been widely proposed to tackle cyber attacks against Internet-connected systems. The complexity and dynamics of cyber attacks require protective mechanisms to be responsive, adaptive, and effective in large scale. It has been proven that deep RL is highly capable of solving complex, dynamic, and especially high-dimensional cyber defense problems [23]. Furthermore, deep RL algorithms exceeded human performance at the game Go [10] and at numerous Atari video games [21].

Additionally, deep RL has been applied to the definition of intelligent agents for automated financial trading. The process of trading is represented as an online decision making problem that concerns two critical steps of market condition summarization and optimal action execution.

✉ Matteo Borrotti
matteo.borrotti@unimib.it

[1] Department of Economics, Management and Statistics, University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, 20126 Milan, Italy

[2] Institute for Applied Mathematics and Information Technologies, National Research Council, Via Alfonso Corti, 12, 20133 Milan, Italy

Dynamic decision making requires the agent to explore an unknown environment independently from human interactions and, at the same time, perform correct online activities [7]. In this context, dynamic portfolio optimization is an attractive problem for deep RL applications [26]. In portfolio management, an agent or a set of agents should continuously reallocate an amount of fund into a number of different financial assets with the final aim of maximizing the return and minimizing the risk [12].

In this work, we extend the approach presented by Lucarelli et al. [18] by incorporating the idea of multi-agent employment by a framework that performs dynamic management of cryptocurrency portfolio (as presented by Patel [24]). With dynamic management, we intend a management able to define the best set of actions at each trading time. The final aim is to maximize portfolio returns. The management framework learns from past behaviours in order to adapt and evolve its way to act on the market.

More precisely, each financial asset in the portfolio is represented by a *local agent*. In each local agent, a specialized deep Q-learning technique competes in the definition of a *global reward function* used to dynamically update the information of each local agent. Global reward function is managed by a *global agent*. Hence, actions are weighted and applied to each asset in the portfolio. Differently from Patel [24], a set of local agents are considered, one for each asset in the portfolio. Each agent concurs in the description of the global environment, defined by hourly price movements. In Patel [24], the author considers two RL agents (one for tick data and one for order book data) for managing only one asset.

Three different deep RL approaches are tested as local agents: Deep Q-Networks (DQN), Double Deep Q-Networks (D-DQN) and Dueling Double Deep Q-Networks (DD-DQN). Each local agent is then evaluated in combination with two global reward functions: sum of nominal net returns and a linear combination of Sharpe ratio and portfolio net return. The proposed Q-learning portfolio management framework is tested on a cryptocurrency portfolio composed by four crypto assets: Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) and Riple (XRP).

Cryptocurrencies can be described by the opening price at the start of a certain period, the highest and lowest price within that period and the closing price at the end of the period (*i.e. candlesticks* or *Open-High-Low-Close* information). In this work, only the closing prices will be considered. However, two aspects of cryptocurrencies are worth mentioning. Cryptocurrencies are based on a decentralized structure and the so-called *openness*. Openness refers to the availability of all information to the public. Furthermore, without a central regulating party, anyone can participate in cryptocurrency trading with low entrance requirements. Consequently, there is an abundance of small-volume currencies that are affected by smaller amount of investment, compared to traditional markets. This helps deep RL techniques take advantage of the consequences of their own market actions. In addition, most cryptocurrency exchanges are open 24/7 without restricting the number of trades. These non-stop markets are ideal for deep RL approaches to learn how to act in shorter time-frames [13].

However, the complexity of cryptocurrencies is yet to be fully explored and this work goes on the direction of empirically understanding how deep RL approaches can deal with such a complex market. Furthermore, the Q-learning portfolio management framework is compared with two alternative techniques: an equally-weighted portfolio approach [20] and a portfolio management technique based on genetic algorithm optimization [8].

The paper is organized as follows. Section 2 briefly describes deep RL. Section 3 defines the portfolio management problem. The cryptocurrency portfolio management framework based on deep RL is described in Sect. 4. The results are shown in Sect. 5, whereas Sect. 6 provides conclusions and future work.

1.1 Related works

Nowadays, RL and deep RL have been largely applied to dynamic portfolio optimization. An increasing number of published works can be found in the literature. For a review of RL and deep RL approaches please see Sato [26]. The author provides a brief survey by examining existing methods of both value-based and policy-based model-free RL for the portfolio optimization problem identifying some key unresolved questions and difficulties facing today's portfolio managers. Some works are shortly described below.

In Liang et al. [16], authors adapted three versions of RL algorithm based on Deep Determinist Policy Gradient (DDPG), Proximal Policy Optimization (PPO) and Policy Gradient (PG) for portfolio management. For this purpose, authors proposed the so-called Adversarial Training in order to reach a more robust deep RL and to consider possible risks in optimizing the portfolio more carefully. They conducted extensive experiments on China's stock market. Similarly, Yu et al. [33] proposed a novel RL architecture consisting in an infused prediction module (IPM), a generative adversarial data augmentation module (DAM) and a behaviour cloning module (BCM). In both works, the proposed approaches have proven to be profitable.

Wang et al. [29] proposed the *AplhaStock* approach, a novel reinforcement learning (RL)-based investment strategy enhanced by interpretable deep attention networks. One of their main contributions is related to a sensitivity

analysis used to unveil how the model selects an asset to be invested according to its multi-aspect features.

Alessandretti et al. [1] and Jiang et al. [13] applied Artificial Intelligent (AI) approaches on portfolio management. In [1] the authors applied a gradient boosting decision tree (*i.e.* XGBoost) and Long Short Term Memory (LSTM) network on a cryptocurrency portfolio. Performances were evaluated considering Sharpe ratio [27] and geometric mean return. All the proposed strategies produced profit over the entire test period. Jiang et al. [13] applied a deterministic policy gradient using a direct reward function (average logarithmic return) for solving the portfolio management problem. The approach proved to outperform classical management techniques except against a Passive Aggressive Mean Reversion technique in terms of cumulative return.

Alternative approaches can be found in the literature. In Maillard et al. [20], the authors proposed an approach based on minimum variance and equally weighted portfolios. Other possible solutions are based on optimization algorithms [8]. Among the most effective optimization algorithms, are metaheuristic methods, which have proved to be very successful in portfolio optimization.

2 Deep reinforcement learning: a short description of main concepts

As described by François-Lavet et al. [9], the general RL problem is formalized as a discrete time stochastic control process where an agent interacts with its environment in the following way: the agent starts, in a given state within its environment $s_0 \in \mathcal{S}$, by gathering an initial observation $x_0 \in \mathbf{X}$. At each time step t, the agent has to take an action $a_t \in \mathcal{A}$. Actions are selected based on a policy, π. The policy is a description of the behaviour of the agent and tells the agent which actions should be selected for each possible state. As a consequence of each action, the agent obtains a reward $r_t \in \mathcal{R}$, the state transitions is updated to $s_{t+1} \in \mathcal{S}$, an observation $x_{t+1} \in \mathbf{X}$ is observed by the agent [2].

At this point, it is possible to define the *transition probability* of each possible next state s_{t+1} as $P(s_{t+1}|s_t, a_t)$, with $s_{t+1}, s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. Similarly, a *reward probability* of each possible reward r_t is defined as $P(r_t|s_t, a_t)$ where $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$. Hence, the expected scalar reward, r_t, received by executing action a in current state s is calculated based on $E_{P(r_t|s_t,a_t)}(r_t|s_t = s, a_t = a)$. This framework can be seen as a finite Markov Decision Process (MDP) [28]. The final aim of an agent is to learn an optimal policy π^*, which defines the probability of selecting action a in state s in order to maximize the sum of the discounted rewards.

The expected discounted return R at time t is defined as follows:

$$R_t = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right], \quad (1)$$

where $E[.]$ is the expectation with respect to the reward distribution and $0 < \gamma < 1$ is called the discount factor. At this point, a Q-value function, $Q^\pi(s, a)$, can be defined as follows:

$$Q^\pi(s, a) = E_\pi[r_t|s_t = s, a_t = a] = E_\pi\left[\sum_{k=0}^{\infty} r_{t+k}|s_t = s, a_t = a\right]. \quad (2)$$

The Q-value, $A_\pi(s, a)$, for an agent is the expected return achievable by starting from state $s \in \mathcal{S}$ and performing action $a \in \mathcal{A}$ following policy π. Equation 2 satisfies a recursive property, so that an iterative update procedure can be used for the estimation of Q-value function:

$$Q^\pi_{i+1}(s, a) = E_\pi\left[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right] = \\ = E_\pi[r_t + \gamma Q^\pi_i(s_{t+1} = s', a_{t+1} = a')|s_t = s, a_t = a], \quad (3)$$

for all $s, s' \in \mathcal{S}$ and $a, a' \in \mathcal{A}$.

The reinforcement learning agent aims at finding the policy which achieves the greatest outcome. Hence, it must learn an optimal policy π^* with the expected value greater than or equal to all other policies, and lead to an optimal Q-value $Q^*(s, a)$. In particular, the iterative update procedure for estimating the optimal Q-value function can be defined as in Eq. 4.

$$Q_{i+1}(s, a) = E_\pi[r_t + \gamma \max_{a'} Q_i(s', a')|s, a]. \quad (4)$$

The iteration procedure converges to the optimal Q-value, Q^*, as $i \to \infty$ and is called *value iteration algorithm*. One of the most popular value-based algorithms is the *Q-learning* algorithm [31].

The basic version of Q-learning algorithm makes use of the Bellman equation for the Q-value function [3] whose unique solution is $Q^*(s, a)$:

$$Q^*(s, a) = (\mathcal{B}Q^*)(s, a), \quad (5)$$

where $\mathcal{B}$ is the Bellman operator mapping any function $K : S \times A \to R$ into another function $S \times A \to R$ and is defined as follows:

$$(\mathcal{B}K)(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma \max_{a' \in A} K(s',a')], \tag{6}$$

where T is the function for calculating the transaction value to go from s to s' given action a. One general proof of convergence to the optimal value function is available in Walkins et al. [31] under the conditions that: (i) the state-action pairs are represented discretely, and (ii) all actions are repeatedly sampled in all states (which ensures sufficient exploration, hence not requiring access to the transition model).

In that context, a parametric value function $Q(s,a;\theta)$ is needed, where θ refers to the model parameters used for approximating the value function. For example, if neural networks are used, θ corresponds to the network weights. In what follows, there is a short description of deep Q-learning approaches used in this work.

2.1 Double deep Q-networks

Double Deep Q-Networks (D-DQNs) are deep RL methods based on Deep Q-Networks (DQNs). DQNs were introduced by Mnih et al. [21]. DQNs stabilize the training of action value function approximation with Convolutional Neural Networks (CNNs) [5] by using experience replay [17] and target network. In fact, DQNs use CNNs to approximate the optimal action value function:

$$Q^*(s,a) = \max_{\pi} E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1} | s_t = s, a_t = a, \pi\right]. \tag{7}$$

In standard Q-learning, as well as in DQN, the parameter θ in $Q(s,a;\theta)$ is updated as follows:

$$\theta_{t+1} = \theta_t + \alpha(y_t^Q - Q(s_t,a_t;\theta_t))\nabla_{Q_t}(s_t,a_t;\theta_t), \tag{8}$$

where α is the learning rate and $y_t^Q = r_{t+1} + \gamma \max_a Q(s_{t+1},a;\theta_t)$.

A limitation of DQN is related to over-estimation. To overcome this limitation, in D-DQN a greedy policy is evaluated in accordance with the online network and a target network is used to estimate its value. This can be achieved by replacing y_t^Q with:

$$y_t^{D-DQN} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1},a;\theta_t); \theta_t^-), \tag{9}$$

where θ_t is the parameter for online network and θ_t^- is the parameter for target network. At this point, y_t^Q can be written as:

$$y_t^Q = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1},a;\theta_t); \theta_t). \tag{10}$$

2.2 Dueling double deep Q-networks

Dueling Double Deep Q-Network (DD-DQN) [30] is based on a dueling network architecture to estimate value function

$$V(s) = E\left[\sum_{k=0}^{\infty} r_{t+k} | s_t = s, \pi\right], \tag{11}$$

and the associated advantage function

$$A(s,a) = Q(s,a) - V(s). \tag{12}$$

The two functions are then combined together in order to estimate $Q(s, a)$ and to converge faster than Q-learning. In DQN, a CNN layer is followed by a Fully Connected (FC) layer. In dueling architecture, a CNN layer is followed by two streams of FC layers, used to estimate the value function and the advantage function separately; then the two streams are combined to estimate the action value function. Usually Eq. 13 is used to combine $V(s)$ and $A(s, a)$.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \max_{a'} A(s,a';\theta,\alpha)). \tag{13}$$

In Eq. 13, δ and β are parameters of the two streams of FC layers. In DD-DQN, Wang et al. [30] propose to replace the max operator with an average action value (Eq. 14).

$$Q(s,a;\theta,\delta,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\delta) - \frac{a}{|\mathcal{A}|} A(s,a';\theta,\delta)). \tag{14}$$

3 Portfolio management problem definition

Portfolio management is a continuous decision process based on an adaptive capital allocation into a set of financial assets. Periodically, a portfolio agent takes investment actions based on specific management policies and information from the past.

A portfolio is generally composed by N financial assets usually described by a set of price information: the opening price at the start of a certain instant t, the highest and lowest price within t and the closing price at the end of t. Financial data can be sampled at different rates. For simplicity purposes, we consider only the closing price from now on.

Considering a period T composed by t trading instants and each instant characterized by N closing prices p, we can construct the portfolio price matrix $\mathbf{P}$ as in Eq. 15.

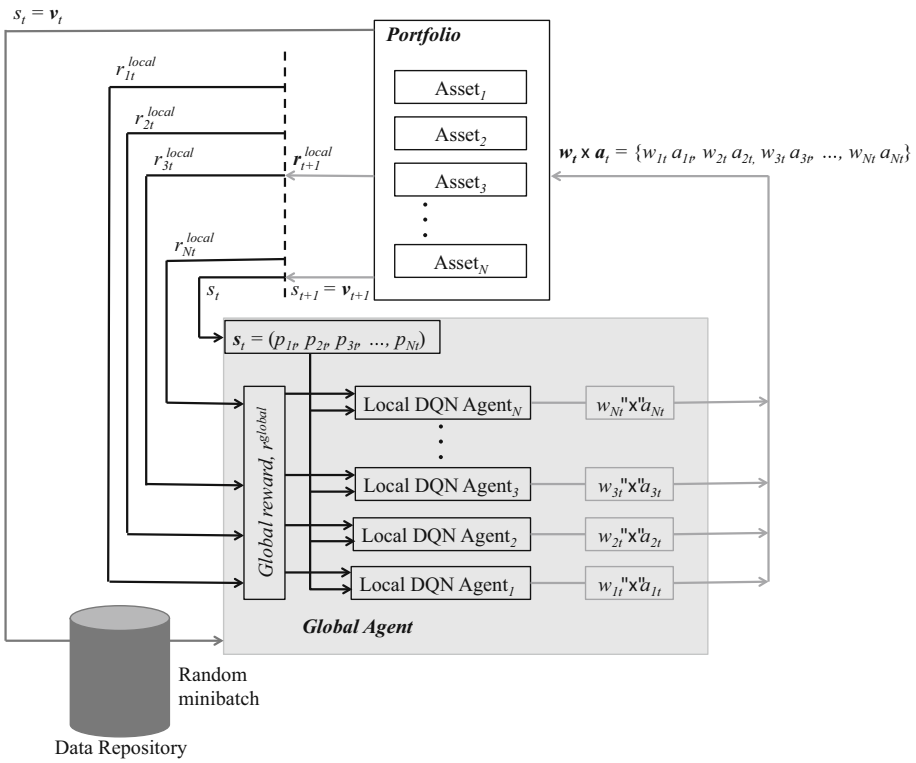


Fig. 1 Deep Q-learning portfolio management framework

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1t} & \cdots & p_{1T} \\ p_{21} & p_{22} & \cdots & p_{2t} & \cdots & p_{2T} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nt} & \cdots & p_{nT} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{Nt} & \cdots & p_{NT} \end{bmatrix} \tag{15}$$

where p_{nt} is the closing price at time t of a financial asset n. The t-th column is the price vector of the t-th trading instant, denoted by $\mathbf{v}_t$. The n-th row is the price time sequence of an asset. The first row is reserved for the risk-less asset. At this point, considering $\mathbf{v}_{t+1}$ e $\mathbf{v}_t$, the price change vector can be defined as follows:

$$\mathbf{y}_t = \mathbf{v}_{t+1} - \mathbf{v}_t = \{p_{1(t+1)} - p_{1t}, p_{2(t+1)} - p_{2t}, p_{3(t+1)} - p_{3t}, \ldots, p_{N(t+1)} - p_{Nt}\} \tag{16}$$

At time t, an agent invests on the market by a portfolio weights vector $\mathbf{w}_t = \{w_{1t}, w_{2t}, w_{3t}, \ldots, w_{Nt}\}$. w_{nt} represents the proportion of financial exposure on a trading period t for asset n.

Now, the portfolio management problem can be described by the MDP framework. Then, we need the following elements: $\mathcal{S}$, $\mathcal{A}$, transition probability $P(s_{t+1}|s_t, a_t)$ with $s_{t+1}, s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ and the reward function r.

In our example, $\mathcal{S}$ defines the closing price at time t of the n assets. $\mathcal{A}$ is the set of possible actions doable on the assets, defined as in Eq. 17 for n-th asset.

$$Action = \begin{cases} \text{Buy, if } E[p_{n(t+1)}] > p_{nt}, \\ \text{Hold, if } E[p_{n(t+1)}] = p_{nt}, \\ \text{Sell, if } E[p_{n(t+1)}] < p_{nt}. \end{cases} \tag{17}$$

where $E[p_{n(t+1)}]$ is expected price at time $t+1$ of the n-th asset.

The usual reward function r at trading time t for asset n can be defined as a simple net profit function, $g_{nt}^{profit} = (p_{nt} - p_{n(t-1)})$ (*i.e.* nominal return) or as Sharpe ratio [27], $s_{p_{nt}} = \frac{(\varrho_{p_{nt}} - \varrho_f)}{\sigma_{p_{nt}}}$, where $\varrho_{p_{nt}}$ is the return of the portfolio or merely the return of the asset, ϱ_f is the risk-free rate, $\sigma_{p_{nt}}$ is the standard deviation of portfolio's return [18].

4 The Q-learning portfolio management framework

The deep Q-learning portfolio management framework is basically based on two main principles: (1) *problem decomposition* and (2) on financial interactions identification (between micro- and macro-levels). In Fig. 1, a graphical representation of the proposed approach is provided. A multi-agent approach is used: *local agents* (one for each asset) are given with hourly data and makes the

decision to buy, sell or hold the specific asset, whereas a *global agent* is defined as a reward function that describes the global portfolio environment. Differently from other approaches [13, 33], N deep RL approaches are used to manage the portfolio instead of only one RL approach.

More precisely, the problem decomposition principle is translated by defining a *local agent* for each asset (*i.e.* currency) in the portfolio. A local agent is a deep Q-learning technique trained in a specific asset. We considered three different deep Q-learning techniques: Deep Q-Networks (DQNs), Double Deep Q-Networks (D-DQNs) and Dueling Double Deep Q-Networks (DD-DQNs). The connection between micro-levels (local agents) and macro-levels (portfolio) is designed by the *global reward function*. The global reward function is a mathematical combination of the single reward obtained by each local agent. The global reward value is then used to feed the local agents before selecting a new action. The portfolio weights, $\mathbf{w}$, are updated at the micro-level at each step. The weight represents the proportion of financial exposure for each asset at trading time t and is used to scale the single trading action. Algorithm 1 describes the deep RL portfolio management framework. In this specific case, each local agent is represented by a deep Q-network.

Algorithm 1 Deep Q-learning Portfolio Management Framework

procedure PORTFOLIO MANAGEMENT(
)**Input:**
Raw price $p_1, \ldots, p_T$ for the n assets in an online manner
r^{global}, definition of *global reward function*
r^{local}, definition of *local reward function*
c, learning rate
γ, discount rate
ϵ, random action probability
Initialize replay memory $\mathcal{D}_n$, $n = 1, \ldots, N$ (*local agent*)
Initialize action-value function Q_n with random weights
sl and tp, stop loss and take profit value
Initialize portfolio weights vector, $\mathbf{w}$
  for episode $= 1$ to M **do**
    Initialize sequence $s_1 = \mathbf{v}_1$
    for $t = 1$ to T **do**
      For each *local agent* n (with $n = 1, \ldots, N$):
      Following ϵ-greedy policy, select a_{nt} in $\mathcal{A} = \{(w_{nt}, buy), hold, (w_{nt}, sell)\}$,
      where w_{nt} is the financial exposure, as

$$a_{nt} = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s_{nt}, a_n; \theta_n) & \text{otherwise} \end{cases}$$

      Execute action a_{nt} and observe reward r_{nt}^{local} and state $s_{n(t+1)}$
      Compute *global reward*, r^{global}
      Set $s_{n(t+1)} = s_{nt}$ and a_{nt}
      Store transition $(s_{nt}, a_{nt}, r_t^{global}, s_{n(t+1)})$ in $\mathcal{D}$
      Sample random minibatch of transitions $(s_{nj}, a_{nj}, r_j^{global}, s_{n(j+1)})$ from $\mathcal{D}_n$
      Set

$$y_{nj} = \begin{cases} r_j^{global} & \text{for terminal } s_{n(t+1)} \\ r_j^{global} + \max_a Q(s_{n(j+1)}, a_n'; \theta_n) & \text{for non-terminal } s_{n(t+1)} \end{cases}$$

      Perform a gradient descent step on $(y_{nj} - Q(s_{nj}, a_{nj}; \theta_n))^2$ [21]

A local agent n takes the global reward value, r_t^{global}, and the closing price at time t, p_{nt}, as input. The global reward value, r_t^{global}, is the same for each agent and it is defined at macro-level. The output of each local agent is represented by an action, a_{nt}, in $\mathcal{A} = \{(w_{nt}, buy), hold, (w_{nt}, sell)\}$. w_{nt} is the related financial exposure. w_{nt} value ranges between 0.83 USD dollars and 25 USD dollars. For simplicity purposes, the interval is discretized in 30 values. Therefore, the cardinality of $\mathcal{A}$ is 61.

Similarly to Lucarelli et al. [18], a stop-loss (sl = -2.3%) and a take-profit (tp = $+6.5\%$) are also applied to the portfolio. The exploration—exploitation dilemma is considered by using an ϵ-greedy technique, where the agent can take a random action with probability, ϵ, and follows the policy that is believed to be optimal with probability, $1 - \epsilon$. As for the first observations, ϵ is set to $\epsilon_{initial} = 1$. Then, ϵ is decreased to 0.13 during the training phase. For each action on the market, a trade cost transition equal to 0.2% is applied (both buy and sell actions).

Locally, a reward function (*local reward*) is used for each local agent. The results are then combined in the global reward value. Two possible functions are considered as global reward: simple sum of nominal net returns and a linear combination of Sharpe ratio [27] and portfolio net return. In the first case, the global return is computed as the sum of the local reward, $r_t^{global,1} = \sum_{n=1}^{N} r_{nt}^{\text{local}}$. In the second case, after computing a portfolio Sharpe ratio, s_{p_t}, and a portfolio net return, $g_t^{\text{portfolio}}$, a weighted sum is applied as follows: $r_t^{global,2} = \eta_1 \times s_{p_t} + \eta_2 \times g_t^{\text{portfolio}}$, where $\eta_1 = \eta$ and $\eta_2 = (1 - \eta)$.

5 Experimental settings and results

5.1 Cryptocurrency data

The deep Q-learning portfolio management framework is tested on a portfolio composed by four cryptocurrencies: Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) and Riple (XRP). Cryptocurrencies are decentralized currencies based on blockchain-based platforms and are not governed by any central authority. Although Bitcoin is one of the most established and discussed cryptocurrency available today, there are more than 200 available tradable cryptocurrencies.

For each cryptocurrency we collect the main technical aspects, namely price movement (opening price, highest and lowest price and closing price). However, only one technical aspect is used as input of the deep Q-learning portfolio management framework, the closing price. The selected sample rate is hourly. Data goes from 01 July 2017 to 25 December 2018. The final dataset is composed by roughly 13,000 observations and one feature. All cryptocurrencies are in USD dollars. In Fig. 2 the USD close price movements are shown. Price movements are indexed using the first available data point (01 July 2017). BTC, LTC and ETH are gathered from www.coinbase.com and XRP from www.bitstamp.net.

Percentage historical daily returns and volatility, skewness and kurtosis are reported in Table 1. All statistical indicators are computed over the entire period. As pointed out in [25], cryptocurrencies exhibit long memory, leverage, stochastic volatility and heavy tailedness. Given that,

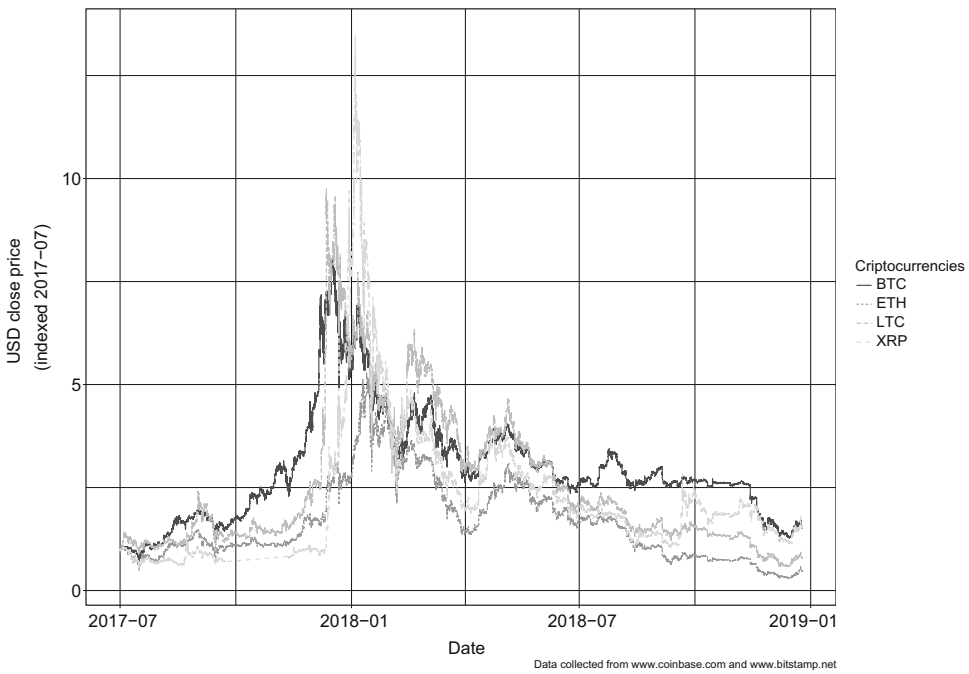


Fig. 2 USD close price movements of Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) and Riple (XRP) time series. All time series are indexed at the first available data point (01 July 2017)

Table 1 Historical daily return (%), historical daily volatility (%), skewness and kurtosis for each cryptocurrency computed over the entire period

Cryptocurrency	Return	Volatility	Skewness	Kurtosis
BTC	0.05	5.29	0.44	16.41
LTC	0.05	7.20	1.37	25.73
ETH	0.00	6.52	0.76	19.00
XRP	0.10	8.97	1.57	25.74

Table 2 List of hyper-parameters

Hyper-parameter	Domain
Learning rate, c	$\{0.001, 0.003, 0.005, 0.01\}$
Discount rate, γ	$\{0.85, 0.90, 0.95\}$
Number of epochs	$\{55, 65, 75\}$
Size of the training period (months)	$\{3, 6, 9\}$
(Only for $r^{\text{global},2}$) function weight, η	$\{0.3, 0.5, 0.7\}$
Total number of settings	108 ($r^{\text{global},1}$), 324 ($r^{\text{global},2}$)

cryptocurrencies have the potential to generate massive amounts of returns but there is also a high risk of losing a significant amount of capital due to the high volatility [14].

5.2 Hyper-parameter tuning

Reinforcement learning and deep learning algorithms frequently require careful tuning of model hyper-parameters, regularization terms and optimization parameters. The deep Q-learning portfolio management framework is based on three different deep Q-learning techniques as local agent. Deep Q-learning techniques have different hyper-parameters that can be tuned in order to increase performance and better fit the problem.

Three different validation periods are randomly sampled from the data and therefore the training set is built. For both reward functions, the cumulative average net return (%) over the three validations is considered as the objective for optimization purpose.

Two separate optimizations are performed. The first is related to $r^{global,1}$ and 4 hyper-parameters are considered for optimization leading to a total number of settings equal to 108. The second optimization concerns $r^{global,2}$ and 324 different settings are tested since 5 hyper-parameters are optimized. Table 2 summarizes selected hyper-parameters and related domains.

This task requires a large amount of computational time. For this reason, the neural network architecture is not involved in the optimization. Furthermore, DQN is considered as the only deep Q-learning technique used for hyper-parameter tuning. At a later stage, the best settings are then applied also to D-DQN and DD-DQN. Eventually, the best settings are applied to all the local agents in the considered deep Q-learning portfolio management framework.

After the optimization, the following best hyper-parameters are selected:

- $r^{\text{global},1}$: $c = 0.001$, $\gamma = 0.90$, epoch = 65 and training size = 6.
- $r^{\text{global},2}$: $c = 0.001$, $\gamma = 0.85$, epoch = 55, training size = 9 and $\eta = 0.30$.

Following [18], the DQN is composed by 3 CNN layers followed by a Fully Connected (FC) layer with 150 neurons. The batch size is set to 50. The network optimization is made by the ADAM algorithm [15]. The loss function is the Mean Squared Error, MSE $= \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$. The activation function of all the CNN layers is set as the Leaky Rectified Linear Units (Leaky ReLU) function [19]. Figure 3 shows the proposed architecture. The input of the network is a 1×24 vector, which represents the prices of a single cryptocurrency in last 24 trading periods. The output layer has 61 neurons. Each one represents a possible combination of action and related financial exposure. The output activation function is a softmax function. D-DQN has similar settings. DD-DQN varies only in the network architecture. It is composed by 3 CNN layers followed by a FC layer with 150 neurons. The FC layer is followed by two streams of FC layers: the first with 75 neurons dedicated to estimate the value function (Eq. 11) and the second with 75 neurons to estimate the advantage function (Eq. 12).

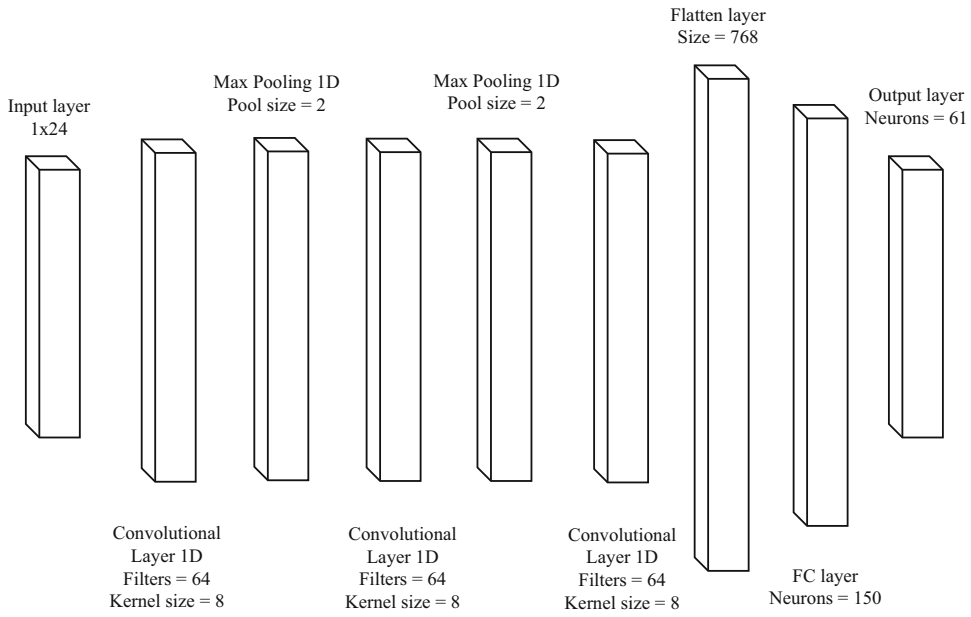
5.3 Algorithm configurations

Three different deep reinforcement learning techniques are used to characterize the local agents: Deep Q-Networks (DQNs), Double Deep Q-Networks (D-DQNs) and Dueling Double Deep Q-Networks (DD-DQNs). Furthermore, two different global reward functions are considered: $r_t^{\text{global},1}$ and $r_t^{\text{global},2}$. The first is the sum of the local reward (*i.e.* nominal net return). The second is the weighted sum of portfolio Sharpe ratio and portfolio net return.

A total of six different algorithm configurations are tested for the optimization of the crypto portfolio. More precisely, the deep RL portfolio management frameworks are set as follows:

(1) DQN-RF1: DQNs as local agents and $r_t^{\text{global},1}$ as global reward functions

Fig. 3 DQN architecture



(2) DQN-RF2: DQNs as local agents and $r_t^{\text{global},2}$ as global reward functions
(3) DDQN-RF1: D-DQNs as local agents and $r_t^{\text{global},1}$ as global reward functions
(4) DDQN-RF2: D-DQNs as local agents and $r_t^{\text{global},2}$ as global reward functions
(5) DDDQN-RF1: DD-DQNs as local agents $r_t^{\text{global},1}$ as global reward functions
(6) DDDQN-RF2: DD-DQNs as local agent and $r_t^{\text{global},2}$ as global reward functions

5.4 Results

All deep Q-learning portfolio management frameworks are tested and compared by sampling 10 consecutive test periods, from 25 July 2018 to 25 December 2018. Each framework is run 5 times for each test period. Each test period is composed by 15 out-of-sample trading days for each cryptocurrency. Given the start date of each test period, the training test is built accordingly. In accordance with the results obtained in Sect. 5.2, the deep Q-learning portfolio management frameworks based on $r_t^{\text{global},1}$ have a training size of 6 months and the ones based on $r_t^{\text{global},2}$ a training size of 9 months. As an example and by considering a training period of 6 months, if the test period starts from 25 July 2018 to 10 August 2018 then the training period starts from 25 January 2018 to 24 July 2018. All portfolio sizes are set to 10,000 USD dollars.

Table 3 shows the daily volatility (%) of each cryptocurrency on the 10 test periods. On average, the period with the lowest volatility (2.47%) goes from 25 October 2018 to 10 November 2018 (Test 7) and the highest volatility (7.34%) is reached in the period from 25 November 2018 to 10 December 2018 (Test 9).

In Fig. 4, the cumulative average portfolio net profit over the 10 test sets is reported. 95% confidence intervals around the mean are also included. A trade cost transition equal to 0.2% is applied (both buy and sell actions). All frameworks seem to be profitable on average.

In Table 4, the daily returns (%) for each framework computed on each test period are reported. Furthermore, Table 4 reports also the average daily returns computed over the four cryptocurrencies (BTC/LTC/ETH/XRP). In 80% of the cases the proposed frameworks have a higher daily returns with respect to the cryptocurrencies. No frameworks reach positive daily returns in all test periods. However, two frameworks (DQN-RF1 and DQN-RF2) out of six get positive daily returns on 90% of the test periods.

Table 3 Daily volatility (%) of the 10 test periods

Cryptocurrency	Test 1	Test 2	Test 3	Test 4	Test 5
BTC	3.33	3.82	2.74	2.25	1.57
LTC	3.77	5.78	4.46	4.90	3.97
ETH	3.28	6.17	5.05	6.07	3.67
XRP	4.07	6.91	4.16	13.37	6.91
Cryptocurrency	Test 6	Test 7	Test 8	Test 9	Test 10
BTC	2.11	0.98	5.68	6.32	4.95
LTC	3.28	2.74	7.25	8.13	6.96
ETH	4.02	2.30	6.47	7.99	7.64
XRP	5.09	3.87	7.05	6.91	6.76

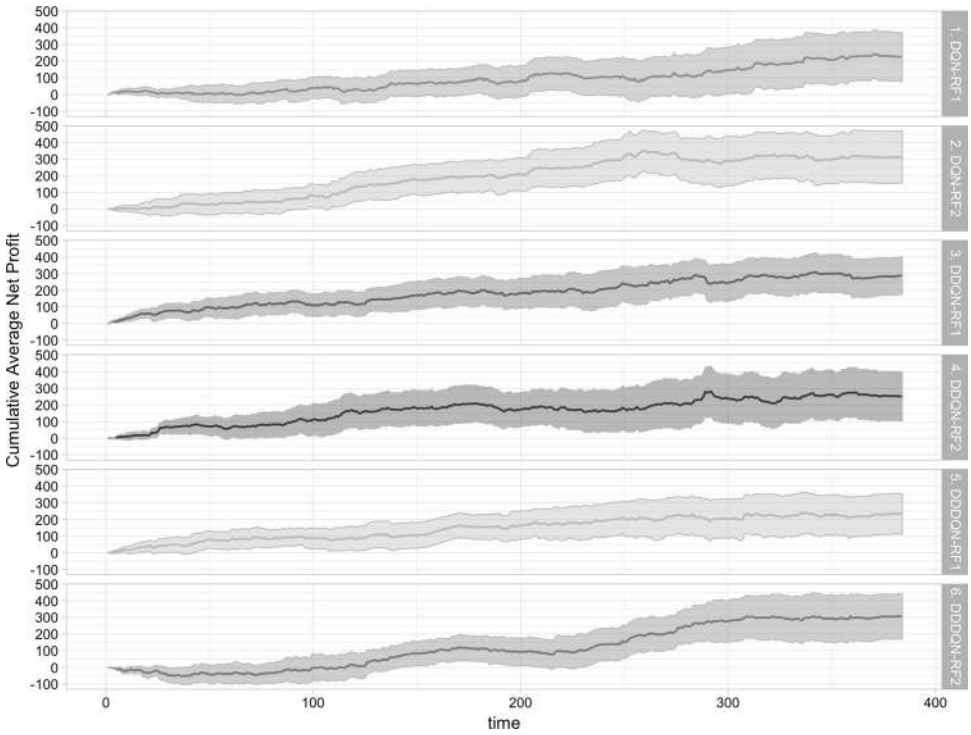


Fig. 4 Average cumulative portfolio net profit over the 10 test sets, *i.e.* different combinations of start and end dates for the trading activity

Table 4 Daily returns (%)

Framework	Test 1	Test 2	Test 3	Test 4	Test 5
DQN-RF1	4.443	4.190	1.819	9.031	7.240
DQN-RF2	0.781	3.893	2.691	3.073	2.586
DDQN-RF1	2.955	7.547	3.136	− 2.666	5.374
DDQN-RF2	3.987	− 1.425	0.044	− 4.650	4.623
DDDQN-RF1	4.138	6.947	2.858	− 1.694	7.437
DDDQN-RF2	6.005	1.888	− 2.283	5.101	4.657
BTC/LTC/ETH/XRP	− 0.353	− 0.092	− 0.190	0.310	0.031
Framework	Test 6	Test 7	Test 8	Test 9	Test 10
DQN-RF1	0.262	1.872	− 6.715	3.552	5.621
DQN-RF2	2.214	1.675	24.781	7.993	− 3.005
DDQN-RF1	5.365	1.837	− 0.811	2.979	− 8.516
DDQN-RF2	0.298	2.948	20.882	6.099	− 0.130
DDDQN-RF1	5.816	2.876	− 3.279	3.604	5.003
DDDQN-RF2	− 0.147	0.912	− 12.052	1.501	6.559
BTC/LTC/ETH/XRP	− 0.086	0.045	− 0.663	0.111	0.311

On test period 8, DQN-RF2 gets an exceptional result obtaining daily returns equal to 24.781%. On the same test period, DDDQN-RF2 obtains the worst result (−12.052%). Daily returns values are positive for all frameworks on 40% of the test periods. On the remaining 60%, at least one framework gets negative performance. However, all frameworks obtain positive daily returns values on average (see Table 5).

Table 5 reports average daily returns (%), Return Of Investments (ROI) and ROI standard deviation. At the end of the test periods (last trading time), ROI (on average) is always above the 2% considering an initial investment of 10,000 USD dollars. However, ROI standard deviation values underline a high variability around the ROI average values. DQN-RF2 reaches the highest value of average daily return, more precisely 4.67%. The worst results in

Table 5 Average daily returns (%), Return Of Investments (ROI) with standard deviation computed over the 10 trading periods. Best values are indicated in bold

Framework	Daily return (%)	ROI (%)	SD (ROI)
DQN-RF1	3.13	2.07	4.61
DQN-RF2	**4.67**	2.59	4.69
DDQN-RF1	1.72	2.56	**3.56**
DDQN-RF2	3.27	2.15	4.28
DDDQN-RF1	3.37	2.36	4.39
DDDQN-RF2	1.21	**2.68**	4.05

Table 6 Average maximum loss and gain with the corresponding range of variation computed over the 10 trading periods. Best values are indicated in bold

Framework	Avg. max loss (%)	Avg. max gain (%)	Range
DQN-RF1	− 2.71	4.48	7.19
DQN-RF2	− 1.97	**5.04**	7.01
DDQN-RF1	− **1.76**	4.56	**6.32**
DDQN-RF2	− 2.27	5.02	7.29
DDDQN-RF1	− 1.96	4.51	6.47
DDDQN-RF2	− 2.52	4.88	7.40

terms of average daily returns (%) are reached with frameworks DDQN-RF2 and DDDQN-RF2. All the other frameworks have an average daily return (%) higher than 3.00. The highest ROI is achieved by framework DDDQN-RF2 followed by framework DQN-RF2.

Table 6 shows the average maximum loss and the average maximum gain obtained in the 10 test periods and the range of variations in-between. The ranges of variation confirm a high fluctuation of the performance during the different test periods leading to a risky portfolio.

Given these results, framework DQN-RF2 is the most profitable solution with the considered portfolio. In Fig. 5, the average cumulative portfolio net profit (%) of framework DQN-RF2 for two different test periods is reported. Period 1 corresponds to the time windows from 25 October 2018 to 10 November 2018. Data from 25 January 2018 to 24 October 2018 are used as training set. Period 2 corresponds to time windows from 25 November 2018 to 10 December 2018. Data from 25 February 2018 to 24 November 2018 are used as training set. Period 1 has average volatility of 2.47% and Period 2 of 7.34% (see Table 3). Respectively, the lowest and highest average volatility in the 10 test periods considered.

Figure 6 (Period 1) and Fig. 7 (Period 2) show the contribution of each local agent in terms of cumulative portfolio net profit.

In both cases the contribution of the local agent related to trade BTC negatively impact the cumulative average net profit. Instead, local agents that are specialized on ETH and XRP have positive cumulative average net profit at the end of both periods. In Period 1 the local agent related to LTC does not perform well by reaching negative cumulative average net profit.

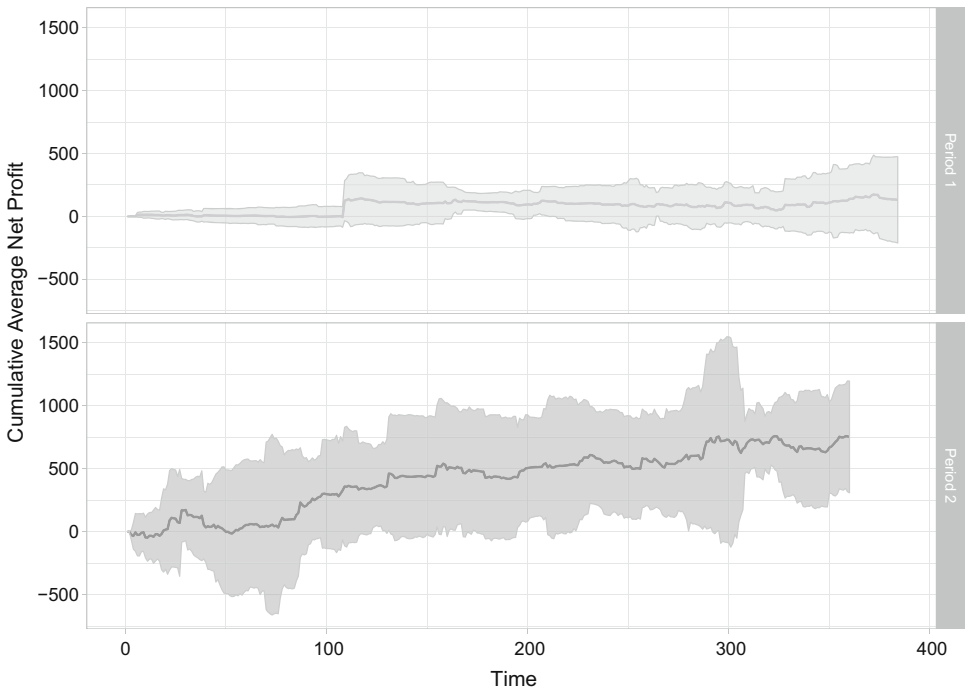


Fig. 5 Average cumulative portfolio net profit of framework DQN-RF2 for test period 7 (Period 1) and test period 9 (Period 2). Period 1: test set from 25 October 2018 to 10 November 2018, volatility 2.47%. Period 2: from 25 November 2018 to 10 December 2018, volatility 7.34%

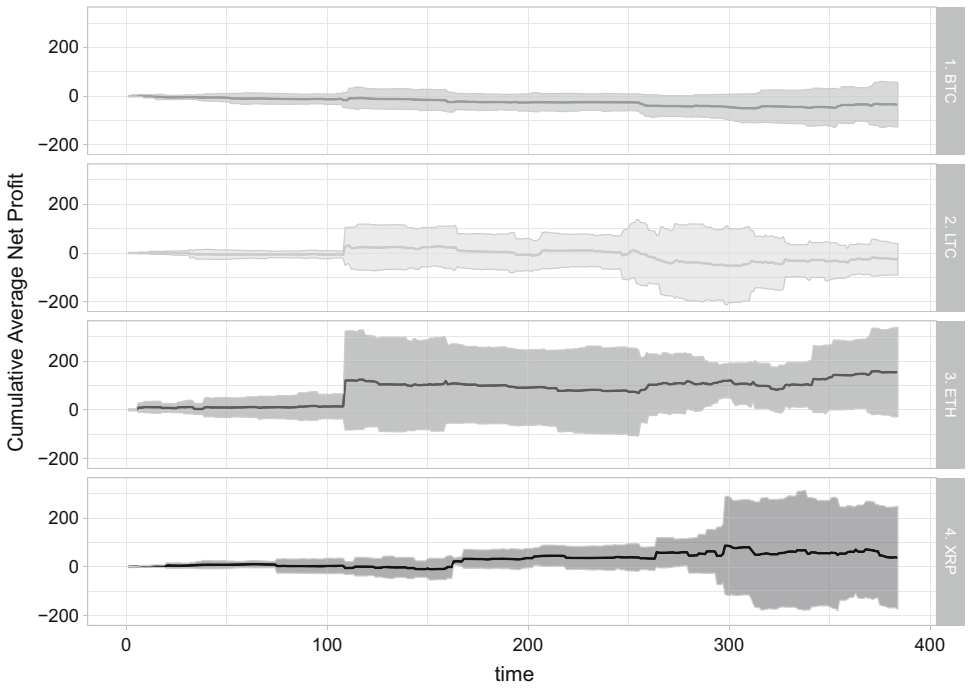


Fig. 6 Average cumulative portfolio net profit of each local agent of DQN-RF2 for Period 1

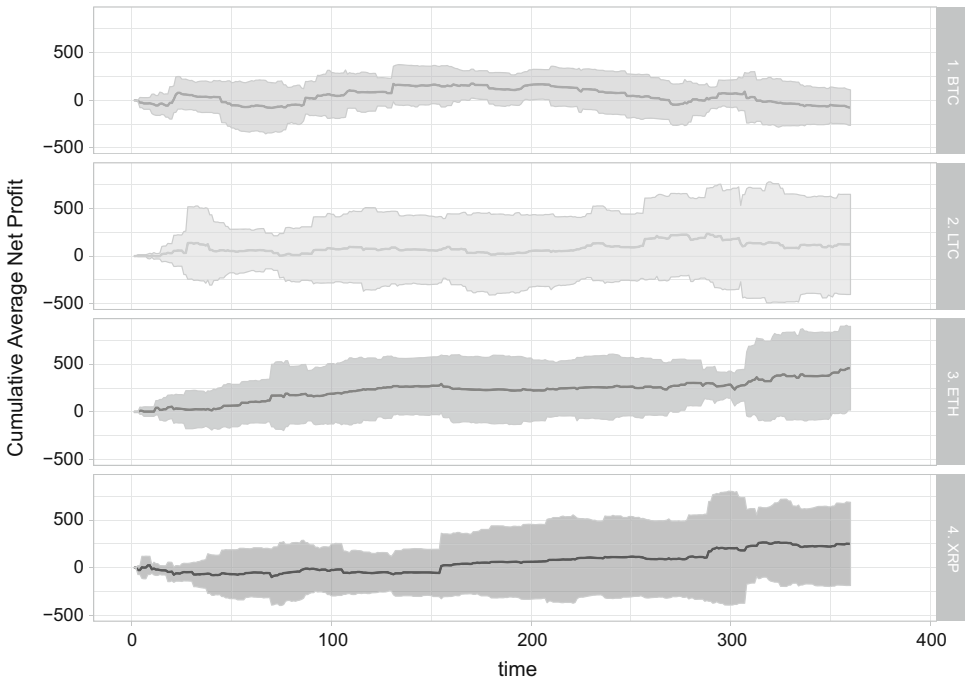


Fig. 7 Average cumulative portfolio net profit of each local agent of DQN-RF2 for Period 2

As a result, even if framework DQN-RF2 shows promising results, a further investigation of risk assessment should be done to improve performance over different periods. On average, framework DQN-RF2 is able to reach positive results in both periods, even though they differ in terms of magnitude. In Period 2, the ROI is around 7.5% and average daily return is almost 8%.

In general, different volatility values strongly influence the performance of the deep Q-learning portfolio management frameworks. Based on the results obtained by all frameworks in Period 1 (low volatility) and Period 2 (high volatility), Table 7 suggests which combination of local agent and global reward function is the most suitable with respect to the expected volatility of the portfolio. While considering a D-DQN as local agent, the most

Table 7 Comparison of the deep Q-learning portfolio management frameworks with respect to the type of expected volatility (*low* or *high*)

	Low	High
DQN	$r^{\text{global},1}$	$r^{\text{global},2}$
D-DQN	$r^{\text{global},2}$	$r^{\text{global},2}$
DD-DQN	$r^{\text{global},1}$	$r^{\text{global},1}$

suitable global reward function is $r^{\text{global},2}$, which considers also the Sharpe ratio. If a more complicated local agent is to be used, then a simpler global reward function ($r^{\text{global},1}$) is advisable, regardless of the type of volatility. A more carefully selection should be done if DQN is considered. In case of low volatility, then a reward function based on the sum of the returns ($r^{\text{global},1}$) is more suitable. If a high volatility is supposed, then a $r^{\text{global},2}$ should be implemented.

The introduction of the Sharpe ratio in $r^{\text{global},2}$ does not lead to more stable results. In fact, $r^{\text{global},2}$ shows higher variability of the performance, regardless of the type of deep RL. In general, $r^{\text{global},1}$ seems to be more stable.

Given these results, increase the complexity of the deep RL does not help improving the overall performance of the proposed framework. In this study, DQN represents the best trade-off between complexity and performance. The results suggest that the introduction of a greedy policy for limiting over-estimation (as in D-DQN) does not increase the performance while trading cryptocurrencies. The use of a dueling architecture to estimate $V(s)$ (Eq. 11) and $A(s, a)$ (Eq. 12) allows the deep RL to reach satisfactory results, on average. However, a DD-DQN does not always guarantee positive performance demonstrating a certain level of uncertainty. More stable results can be obtained by using a simple global reward function, such as $r^{\text{global},1}$.

5.5 Comparison

The deep Q-learning portfolio management framework based on DQN as local agent and with $r_t^{\text{global},2}$ as global reward function (DQN-RF2) is compared with two techniques: an equally weighted portfolio technique [20] (EW-P) and a portfolio selection using genetic algorithm technique [8] (PS-GA). All the approaches are compared in terms of Sharpe ratio.

The equally weighted portfolio technique [20] is a simple approach where all investments are uniformly distributed between the N financial assets. Given the four cryptocurrencies ($N = 4$), the following weight vector is considered, $\mathbf{w} = \{\frac{1}{N}, \frac{1}{N}, \frac{1}{N}, \frac{1}{N}\} = \{0.25, 0.25, 0.25, 0.25\}$ at each trading instant.

A Genetic Algorithm (GA) [11] is a metaheuristic search method used in many different optimization problems. At each iteration of the algorithm, GA works with a population of individuals each representing a possible solution to a given problem. Each individual in the population is assigned a *fitness score* (*i.e.* objective function value) that determines how good a solution is. The highly fit individuals are identified (*selection*) and are given the chance to reproduce by crossbreeding with other individuals (*crossover*). With a certain probability, some random modifications are possible (*mutation*). A whole new population of possible solutions is thus produced. In this way, over many generations (*i.e.* iterations), good characteristics are spread throughout the population in order to explore the most promising areas of the search space. In the context of portfolio management, GAs are used to find an optimal weight vector for financial investments. In general, metaheuristic-based algorithms cannot prove the optimality of the returned solution, but they are usually very efficient in finding (near-)optimal solutions.

For this reason, a real-valued GA is implemented. In contrast to the binary GA, the real-valued GA codifies the features of each individual in the population as real values. More precisely, an individual is represented as a vector, $\mathbf{w}$, of real values between 0 and 1. The length of the vector is equal to the number of cryptocurrencies in the portfolio ($N = 4$). Basically, each vector is a candidate portfolio weights vector. A population is then composed by a set of candidate portfolio weights vectors ($Pop_{size} = 50$) and each individual is evaluated based on an objective function (*Sharpe ratio*). At each iteration, the Sharpe ratio should be maximized. The real-valued GA is also set with crossover probability equal to 0.8 and mutation probability equal to 0.1. The *average single-point* procedure is selected as crossover operator and a *roulette wheel* technique as an example of proportional selection operator. Algorithm 2 describes the procedure.

Algorithm 2 Portfolio Selection using Genetic Algorithm

procedure WEIGHTS OPTIMIZATION( )**Input:**
TrainMatrix, Training period (matrix of size $T \times N$)
p_{cross}, crossover probability
p_{mut}, mutation probability
Pop_{size}, population size
Randomly generate the initial population of candidate individuals (portfolio weights vectors)
  for iteration = 1 to $MaxInteration$ **do**
    for each individual ∈ population **do**
      Apply weights vector to **TrainMatrix**
      Compute returns
      Compute Sharpe ratio (*i.e.* objective function)
    Select individuals with a proportional selection operator
    Apply average single-point crossover with probability p_{cross}
    Apply mutation with probability p_{mut}
    Update population for next iteration

The equally weighted portfolio technique is directly tested on the 10 test periods. The portfolio management technique based on GA optimization is based on the following steps: (1) weights vector is optimized on the training period and (2) performance is evaluated on the test period. GA is a stochastic optimization algorithm than the approach is run 5 times for each training and test period. Only the size of the training period which is equal to 9 months is considered.

In Table 8, the average Sharpe ratio for each approach is reported. None of them shows a remarkable Sharpe ratio. PS-GA has a negative value. In this case, this is due to the portfolio's return is negative. EW-P has a Sharpe ratio almost equal to zero due to an investment's excess return value near zero. DQN-RF2 has a Sharpe ratio that reaches a value of 0.202. This value highlights the fact that the standard deviation around the average daily return is quite high. A high standard deviation value can be expected while trading on an hourly basis. However, this result suggests that the DQN-RF2 approach needs to be improved by reducing the standard deviation. For instance, this can be done by selecting cryptocurrencies that are less correlated.

Now, we compare the three approaches on a specific scenario. The scenario coincides with Period 2. The test Period 2 corresponds to time windows from 25 November 2018 to 10 December 2018. Data from 25 February 2018 to 24 November 2018 are used as training set. This scenario is characterized by high daily volatility (see Table 3). Figure 8 shows how the approaches perform on the 15 out-of-sample trading days. The solid line represents the performance of the DQN-RF2 approach. The dashed line represents the EW-P technique and the dash-dotted line corresponds to the PS-GA. On the first trading days, DQN-RF2 and EW-P have similar behaviour. After 8 days, EW-P has a sharp reduction in terms of cumulative average net profit. PS-GA is not able to get any profit in the 15 out-of-sample trading days. In this scenario, DQN-RF2 shows higher ability to manage the entire portfolio.

Table 8 Sharpe ratio

	DQN-RF2	EW-P	PS-GA
Sharpe ratio	0.202	0.043	− 0.190

6 Conclusions and discussion

Portfolio management is an interesting field for the application of deep RL approaches. In this work, a first definition of a deep RL portfolio management framework is proposed. Differently from traditional works, *N* deep RL approaches are used to manage the portfolio instead of only one RL approach. The proposed general framework is composed by two main ingredients: (i) a set of *local agents* that are responsible for trading the single asset in the portfolio and (ii) a *global agent* that rewards each local agent based on the same reward function (global reward).

The local agent was based on three different deep RL approaches: deep Q-learning, double deep Q-learning and dueling double deep Q-learning. Furthermore, the global reward takes the following two possible forms: a sum of the local nominal returns ($r^{\text{global},1}$) and a weighted sum of portfolio Sharpe ratio and portfolio net return ($r^{\text{global},1}$).

All frameworks were tested on a crypto portfolio composed by four assets (currency): Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) and Riple (XRP). Data were

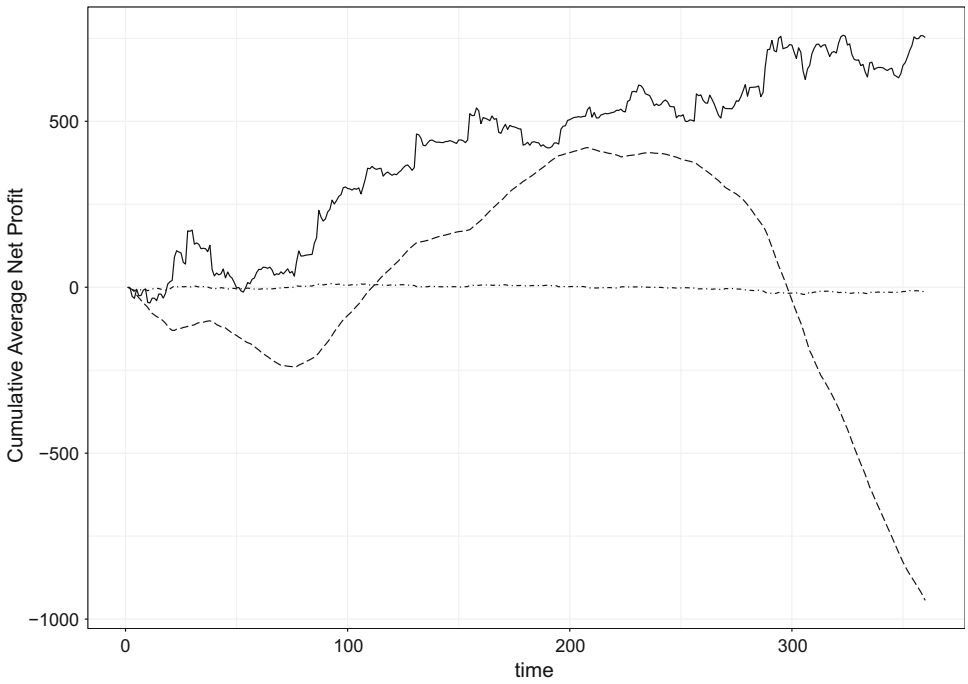


Fig. 8 Average cumulative portfolio net profit of DQN-RF2 (solid line), EW-P (dashed line) and PS-GA (dash-dotted line)

collected from 01 July 2017 to 25 December 2018. All cryptocurrencies were in USD dollars.

All frameworks produced positive average daily returns (%) for a set of shorter trading periods (different combinations of start and end dates for the trading activity). Furthermore, ROI (on average) is always above the 2% considering an initial investment of 10,000 USD dollars. As suggested by the empirical study, while considering a D-DQN as local agent, the most suitable global reward function is a linear combination of Sharpe ratio and portfolio net return. If a more complicated local agent is to be used, then a simpler global reward function (such as sum of nominal net returns) is advisable, regardless of the type of volatility. A more carefully selection should be done if DQN is considered. In presence of low volatility, then a reward function based on the sum of nominal net returns is more suitable. If a high volatility is supposed, then a reward function that takes Sharpe ratio into consideration should be implemented.

The portfolio management framework based on deep Q-network as local agents and $r^{\text{global},2}$ as global reward (DQN-RF2) achieved the best value in terms of average daily returns (4.67%). The DQN-RF2 was then compared with other two alternative approaches: an equally weighted portfolio technique (EW-P) and portfolio management technique based on GA optimization (PS-GA). The three approaches were compared in terms of Sharpe ratio. None of them showed remarkable results; however, DQN-RF2 reached the highest value of Sharpe ratio (0.20).

These initial results show that the proposed deep RL portfolio management framework is a promising approach that should be further investigated to improve overall performance.

Given that, some considerations should be done with respect to possible improvements. The proposed approach should be compared with more traditional techniques for portfolio management to demonstrate its potential superiority. Additionally, $r^{\text{global},2}$ demonstrated not to consider the risk of an investment as expected. A more in-depth study should be carried out to better understand the interaction between local agents through the global reward. Furthermore, each local agent used the same deep RL approach. The development of a network selection procedure could improve the performance on each single asset. Moreover, a study on the cryptocurrency portfolio composition should be performed to identify a set of cryptocurrencies that are less correlated and more suitable for portfolio management purpose. In this regard, considering works such as [25] or volatility studies on the cryptocurrency market [14] could help improve the deep RL framework for portfolio management. Furthermore, the deep RL framework should be tested on different markets in order to assess the generalization power of the proposed approach.

From a computational point of view, hyper-parameter tuning is a really intensive task. The use of parallel computing techniques and GPU processors could reduce the time needed to optimize the deep RL framework parameters and, furthermore, speed up the training phase.

As such, considering these elements for future work is surely a good prospect to improve the deep Q-learning portfolio management framework.

Acknowledgements We greatly acknowledge the DEMS Data Science Lab for supporting this work by providing computational resources. Furthermore, the authors gratefully acknowledge the anonymous reviewers and the editor for their helpful comments.

Funding Open access funding provided by Università degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alessandretti L, ElBahrawy A, Aiello LM, Baronchetti A (2018) Anticipating cryptocurrency prices using machine learning. Complexity 2018:1–16
2. Barto AG, Sutton RS, Anderson CW (1983) Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans Syst Man Cybern 13(5):834–846
3. Bellman RE, Dreyfus SE (1962) Applied dynamic programming. RAND Corporation, Santa Monica
4. Botvinick M, Ritter S, Wang JX, Kurth-Nelson Z, Blundell C, Hassabis D (2019) Reinforcement learning, fast and slow. Trends Cognit Sci 23(5):408–422
5. Buduma N (2017) Fundamentals of deep learning: designing next-generation artificial intelligence algorithms. O'Reilly Media, Newton
6. Caicedo JC, Lazebnik S (2015) Active object localization with deep reinforcement learning. In: IEEE international conference on computer vision (ICCV'15), pp 2488–2496
7. Deng Y, Bao F, Kong Y, Ren Z, Dai Q (2017) Deep direct reinforcement learning for financial signal representation and trading. IEEE Trans Neural Netw Learn Syst 28(3):653–664
8. di Tollo G, Roli A (2008) Metaheuristics for the portfolio selection problem. Int J Oper Res 5(1):13–35
9. François-Lavet V, Henderson P, Islam R, Pineau J, Bellemare MC (2018) An introduction to deep reinforcement learning. Found Trends Mach Learn 11(3–4):219–354
10. Fu MC, Chang HS, Hu J, Marcus SI (2016) Google DeepMind's AlphaGo. ORMS Today 43(5):1–14
11. Goldberg GE (1989) Genetic algorithms in search optimization and machine learning. Addison Wesley, Boston
12. Haugen RA (1986) Modern investment theory. Prentice Hall, Upper Saddle River
13. Jiang Z, Liang J (2017) Cryptocurrency portfolio management with deep reinforcement learning. In: Proceedings of the intelligent systems conference (IntelliSys'17), pp 905–913
14. Katsiampa P (2017) Volatility estimation for Bitcoin: a comparison of GARCH models. Econ Lett 158:3–6
15. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the 3rd international conference of learning representations, pp 1–15
16. Liang Z, Chen H, Zhu J, Jiang K, Li Y (2018) Adversarial deep reinforcement learning in portfolio management. arXiv:1808.09940:1-11
17. Lin LJ (1991) Programming robots using reinforcement learning and teaching. In: Proceedings of the ninth national conference on artificial intelligence (AAAI'91). AAAI Press
18. Lucarelli G, Borrotti M (2019) A deep reinforcement learning approach for automated cryptocurrency trading. In: IFIP advances in information and communication technology. Springer, Berlin
19. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th international conference on machine learning, pp 1–6
20. Maillard S, Roncalli T, Teïletche J (2010) The properties of equally weighted risk contribution portfolios. J Portf Manag Summer 36(4):60–70
21. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. arXiv:1312.5602:1-9
22. Neftci EO, Averbeck BB (2019) Reinforcement learning in artificial and biological systems. Nature Mach Learn 1:133–143
23. Nguyen TT, Reddi VJ (2019) Deep reinforcement learning for cyber security. arXiv:1906.05799v2:1-16
24. Patel Y (2018) Optimizing market making using multi-agent reinforcement learning. arXiv:1812.10252v1:1-10
25. Phillip A, Chan JSK, Peiris S (2018) A new look at cryptocurrencies. Econ Lett 163:6–9
26. Sato Y (2019) Model-free reinforcement learning for financial portfolios: a brief survey. arXiv:1904.04973:1:20
27. Sharpe WF (1994) The Sharpe ration. J Portf Manag 21:49–58
28. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. The MIT Press, Cambridge
29. Wang J, Zhang Y, Tang K, Wu J, Xiong Z (2019) AlphaStock: a buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. arXiv:1908.02646:1:9
30. Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N (2016) Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33 international conference on machine learning (ICML'16). PLMR
31. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8:279–292
32. Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75
33. Yu P, Lee JS, Kulyatin I, Shi Z, Dasgupta S (2019) Model-based deep reinforcement learning for dynamic portfolio optimization. arXiv:1901.08740:1-21