Review

Deep Reinforcement Learning For Trading - A Critical Survey

Adrian Millea [1]

[1] Imperial College London; am2714@ic.ac.uk

Abstract: Deep reinforcement learning (DRL) has achieved significant results in many Machine Learning (ML) benchmarks. In this short survey we provide an overview of DRL applied to trading on financial markets, including a short meta-analysis using Google Scholar, with an emphasis on using *hierarchy* for dividing the problem space as well as using *model-based RL* to learn a world model of the trading environment which can be used for prediction. In addition, multiple *risk measures* are defined and discussed, which not only provide a way of quantifying the performance of various algorithms, but they can also act as (dense) reward-shaping mechanisms for the agent. We discuss in detail the various *state representations* used for financial markets, which we consider critical for the success and efficiency of such DRL agents. The market in focus for this survey is the cryptocurrency market.

Keywords: deep reinforcement learning, model-based RL, hierarchy, trading, cryptocurrency, foreign exchange, stock market, risk, prediction, reward shaping

1. Introduction

Predicting and analyzing financial indices has been of interest in the financial community for a long time, but recently, there has been a wide interest in the Machine Learning (ML) community to make use of and benefit from the more advanced techniques, like deep networks or deep reinforcement learning. In addition, the cryptocurrencies market has gained a lot of popularity worldwide, among which Bitcoin, makes the top news almost weekly now.

In this survey article we explore different aspects of using DRL for trading, by first using Google Scholar to get a glimpse of what the corpus of papers which use DRL for trading looks like, then going through the main concepts of interest at the intersection of trading and DRL, like using different measures of risk for reward shaping and embedding some form of forecasting or prediction into the agent's decision-making. We then explain the contributions of the first DRL paper in more detail [6] and continue by going through the specifics of applying DRL to trading markets. Then, we go deeper into the technologies which look most promising, hierarchical DRL and model-based RL and finish by stating some of the commonalities and methodological issues we found in the corpus explored.

We begin by referring to a few survey articles which are somehow related to this work:

- [1] looks at model-free methods and is thus limiting. Moreover, it discusses some aspects from a financial-economical point of view, of the portfolio management problem, the relation between the Kelly Criterion, Risk Parity and Mean Variance. We are interested in the RL perspective, and the need for risk measures is only to feed the agents with helpful reward functions. We also discuss the hierarchical approach which we consider to be quite promising, especially for applying a DRL agent to multiple markets.
- [2] looks only at the prediction problem using deep learning.
- [3] has a similar outlook as this work, but the works discussed are quite old, very few using deep learning techniques.
- [4] looks at more applications in economics, not just trading. Also, the deep reinforcement learning part is quite small in the whole article.

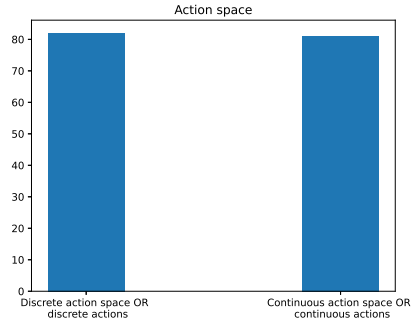
We focus more on the cryptocurrency markets, but without loss of generality, since trading is very similar among markets, with the only thing differing being the volatility of assets

Citation: Millea, A. Deep reinforcement learning for trading - a critical survey. *Preprints* **2021**, *1*, 0. https://doi.org/

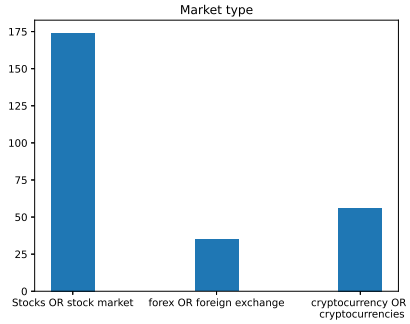
Received:
Accepted:
Published:

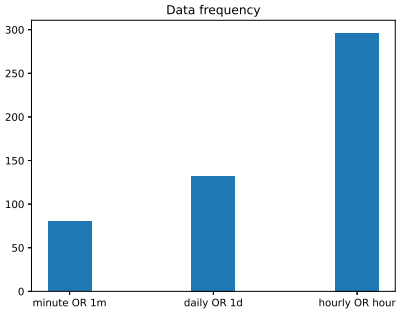
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

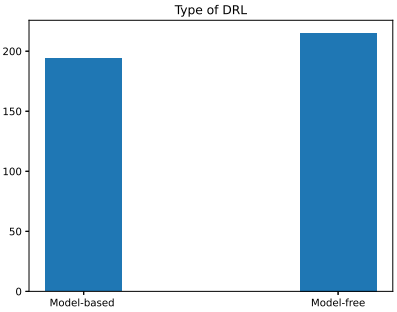
(a) Type of action space.



(b) Type of market.







(c) Frequency of data used.

(d) Model-free vs model-based DRL.

Figure 1. Using Google Scholar to explore 395 papers using DRL for trading.

in each market type, with the cryptocurrencies being the most volatile. We do advise the curious reader to take a look at the mentioned works, since they provide different perspectives and plenty of resources to further one's research. Following [5] we take a similar approach to our investigation. We use Google Scholar for meta-analysis, to search through the papers instead of downloading them and then individually searching through them, since this is a cumbersome process due to the fact that a large number of papers is behind a paywall. This is possible for two reasons:

- All papers using DRL for trading will most probably cite the first paper that introduced DQN [6].
- Because of the above reason we can use the feature in Scholar which *searches through the citing articles*.

The following data is obtained through this methodology. The total number of articles citing [6] is 16652, today (18 October 2021). By adding *trading market reward action state* as search words we narrow it down to 395. We have to mention here that we had to put every word between quotes, otherwise Google Scholar seemed a bit stochastic sometimes, in the sense that, by changing the position of *market* in the search query, we got a different number of results. We investigate the type of action space, state space, market type, input data and if the papers use some form of prediction or not by using representative keywords (e.g. *continuous actions* vs. *discrete actions*). Of course the following results should be taken with a grain of salt, in the sense that some keywords might appear in the paper, but as references or mentiones, rather than actual uses. We think these results provide a rough idea of the underlying structures but definitely not the precise ones. More details in Figures 1 and 2.

The cryptocurrency market has been of wide interest for economic agents for some time now. Everything started with the whitepaper on Bitcoin [7]. This is in part due to the fundamentally different principles on which it is functioning (i.e. coins are minted / mined, they are decentralised, etc.) [8] but also due to the extremely high volatility the market is experiencing [9]. This means there is room for significant profit for the smart

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

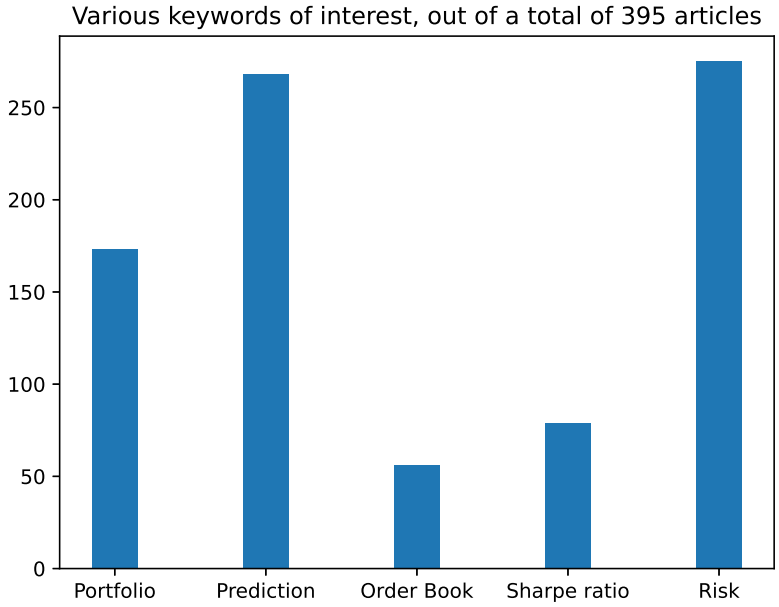


Figure 2. Various keywords presence.

agent but also the possibility of huge losses for the speculative trader. Moreover the market capitalisation is increasing almost continuously, which means new actors are entering and investing actively in the crypto markets. When looking at the problem of automatically trading on these markets, we can define a few, quite different problems one is facing and which have been tackled separately:

- The **prediction** problem, where we want to accurately predict some timesteps in the future based on the past and on the general dynamics of the time-series. What we mean by this is that, for example, gold or oil do not have such large fluctuations as the crypto markets. When one is predicting the next point for the price, it is advisable to take this information into account. One way to do this is to consider general a priori models, based on large historical samples, such that we embed this possible, historical dynamics into our prediction model. This would function as a type of general prior information, for example it is absurd to consider models where gold will rise 100 percent overnight, this has never happened and probably never will. Deep learning models have been successfully applied to such predictions [2,10–12] as well as many other ML techniques [13–15]. However, for the crypto markets this might not be that impossible and some of them did in fact rise thousands percent overnight, with many still rising significantly every day [1]. In the context of RL, prediction is usually done through model-based RL [16], where a simulation of the environment is created based on data seen so far. And then prediction amounts to running this simulation to see what comes next (at least according to the prediction model). More on model-based RL in section 9.1. For a comprehensive survey see [17].
- The **strategy** itself might be based on the prediction, but embed other information as well, like for example the certainty of the prediction (which should inform how much we would like to invest, assuming a rising or bull future market) or other external information sources, like social media sentiments [18](e.g. are people positive or negative about an asset?) or news [19], volume, or other sources of related information which might influence the decision. The strategy should be also customisable, in the sense that we might want a risk-averse strategy or risk-seeking one. This cannot

[1] https://coinmarketcap.com/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

be done from a point prediction only, but there are some ways of estimating risk for DRL [20]. For all these reasons we consider the strategy, or decision-making, a quite different problem than the prediction problem. And this is the general trend in the related literature [21,22].

- The **portfolio management** problem is encountered when dealing with multiple assets. Until now we've been talking only about a single asset, but if some amount of resources is being split among multiple concurrent assets then we have to decide how to split these resources. At first the crypto markets were highly correlated (if Bitcoin was going up then most other cryptocurrencies would go up, so this differentiation didn't matter too much), but more recently, it has been shown that the correlation among different pairs is significantly decreasing [2]. Thus, an investing strategy is needed to select among the most promising (for the next time slice) coins. The timeslice for which one does rebalancing is also a significant parameter of the investing strategy. Rebalancing, usually means, how to split the resources (evenly) among the most promising coins. Most of the work in trading using smart techniques like DRL is situated here. See for example [1,23–28]. Not all the related works we are citing are dealing with the cryptocurrency markets, but since all financial markets are quite similar, we consider these relevant for this article.
- The **risk** associated with some investment is inevitable. However, through smart techniques, a market actor might minimize the risk of investment while still achieving significant profits. Also, some actors might want to follow a risk-averse strategy while other might be risk-seeking. The risk of a strategy is generally measured by using the Sharpe ratio or Sortino ratio, which take into account a risk-free investment (e.g. U.S. treasury bond or cash), but even though we acknowledge the significance of these measures, they do have some drawbacks. We will discuss in more detail the two measures, giving the formal description and propose alternative measures of risk. Moreover, if one uses the prediction in the strategy, by looking at the uncertainty of prediction, one can also infuse our strategy with a measure of risk given by this certainty, i.e. if the prediction of some asset going up is very uncertain then the risk is high and the other way around, if prediction is certain, risk is lower.

We show an overview of the above mentioned financial concepts, the relations between them and their RL correspondents in Figure 3.

2. Prediction

Prediction or forecasting of a time-series is the process of estimating what will happen in the next time period based on what we have seen so far. Will the price go up or down, by how much and for how long? A comprehensive prediction provides a probability distribution (if it is not a point estimate) of these quantities. Many older works deal with the prediction problem in a financial market, assuming the strategy is clear once the prediction is accurate. And it is indeed so, but quite often the predictions are not correct and this can significantly impact the performance of the trading strategy. However, it has been shown that some techniques exist which can predict even quasi-periodic time-series, which is quite remarkable, considering the fundamentally chaotic nature of such time-series [29,30]. Depending on the nature of the time-series, the prediction interval is generally one time-step and for each new time-point prediction, the ground truth is fed back to the network. If the prediction is fed, such that multiple time-steps are predicted, the prediction degrades rapidly, due to accumulating errors. However, there are some exceptions, see for example [31]. One way to overcome this limitation is to look at data at a higher granularity (at coarser data). For example, if one looks at minute data (by this, we mean that the prediction machinery is fed with this type of data), to predict 30 minutes in the future, one needs to predict 30 time-points, whereas if the one looks at 30 minutes data (meaning each point is a summarization of a 30 minutes time-slice), then there is need only for one-step

[2] https://charts.coinmetrics.io/correlations/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

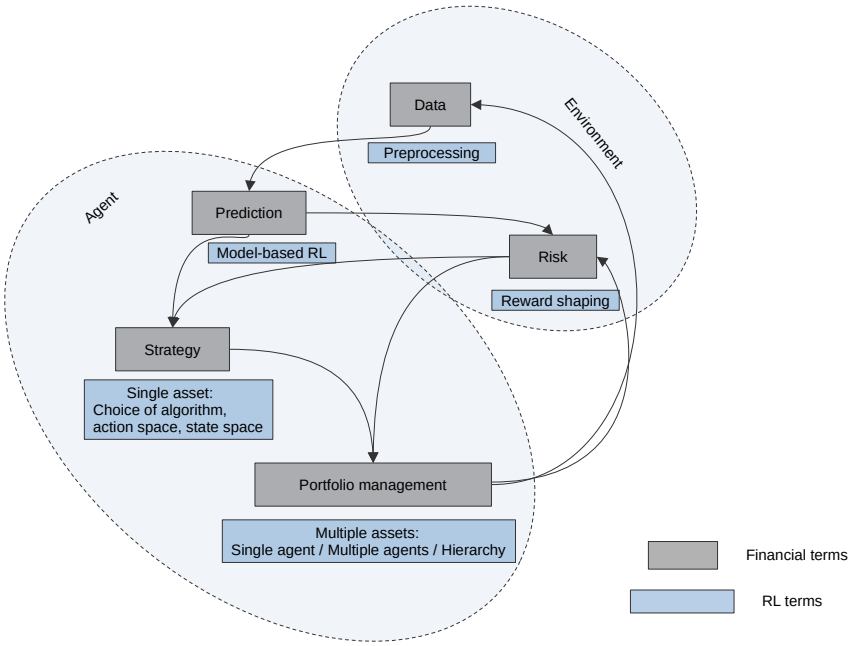


Figure 3. Overview of the main financial concepts discussed and their RL correspondents.

prediction. Any strategy built on top of a prediction module is critically dependent on this sampling period. By using multiple different time-scales, one can have a much better picture of the future, especially if the predictions at different levels are combined (for example using ensembles). Surprisingly, there are not many related works which use such a technique, but there are some [28].

3. Strategy for RL

The strategy for trading can vary widely, depending on its scope, the data sampling rate, the risk quantification choice and more. We distinguish a few ways to organise the type of strategy:

- Single or multi-asset strategy. This needs to take into account data for each of the assets, either time-based or price-based data. This can be done individually (e.g. have one strategy for each asset) or aggregated.
- The sampling rate and nature of the data. We can have strategies which make decision at second intervals or even less (High Frequency Trading or HFT [32]) or make a few decisions throughout a day (intraday) [33], day trading (at the end of each day all open orders are closed) [34] or higher intervals. The type of data fed is usually time-based but there are also price-based environments for RL ([35].
- The reward fed to the RL agent is completely governing its behaviour, so a wise choice of the reward shaping function is critical for good performance. There are quite a number of rewards one can choose from or combine, from risk-based measures, to profitability or cumulative return, number of trades per interval, etc. The RL framework accepts any sort of rewards, the denser the better. For example, in [35] the authors test 7 different reward functions based on various measures of risk or PnL (Profit and Loss).

There are also works which discover or learn algorithmic trading rules through various methods like evolutionary computation [36] or deep reinforcement learning [37].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

3.1. Benchmark strategies

When evaluating this type of work using backtests, one needs to have some benchmark strategies to compare with. We have a wide variety to choose from, but we restrict ourselves to just a few for brevity. We list them below in increasing order of complexity:

- Buy and Hold. This is the classical benchmark [38–40] when trading on the cryptocurrency markets since the markets are significantly increasing, thus simply buying an asset and holding it until the end of the testing period provides some informative baseline profit measure.
- Uniform constant rebalanced portfolio (UCRP) [41] which rebalances the assets every day keeping the same distribution among them.
- On-line moving average reversion (OLMAR) [42] is an extension to the moving average reversion (MAR) strategy [43] but overcomes some of its limitations. MAR tries to make profits by assuming the asset will return (revert) to the moving average value in the near future.
- A deep RL architecture [27] very well received by the community and with great results. It has also been used in other related works for comparison since the code is readily available [3].

4. Portfolio management

In a financial market one of the main issues is in which assets to invest, which coins to buy. There are many available assets, on Binance (the largest platform for cryptocurrencies by market cap) for example, there are hundreds of coins and each one can form a pair with a few major stable coins like USDT (Tether). Thus the choices are numerous and the decision is hard. The portfolio management problem deals with allocating the cash resources (this can be any stable currency which does not suffer from the fluctuations of the market, a baseline) into a number of assets m. Formally, this means finding a set of weights

$$\mathbf{w} = \{w_0, w_1, w_2, ..., w_m\} \text{ such that } \sum_{i=0}^{m} w_m = 1 \tag{1}$$

where at the beginning the weights are $\mathbf{w} = \{1, 0, 0, ..., 0\}$ which means we have only cash. The goal is to increase the value of the portfolio by selecting the assets which give the highest return. If at the time-step t the overall value of the porfolio is given by:

$$V_t = \sum_{i=1}^{m} p_i^t w_i^t \tag{2}$$

where p_i is the sell price of the asset i, then $V_{t+1} > V_t$. Prices and asset quantities change during trading periods and when switching back to cash (this is what the above equation does), this new amount should be larger than the previous cash value of the portfolio.

5. Risk

Risk has been quantified (measured) and/or managed (fed as reward to the DRL agent, or part of the reward) through various measures throughout the years. We state below what we consider to be some of the most important ones.

5.1. Sharpe ratio and Sortino ratio

One of the most ubiquitous is the Sharpe ratio [44] or the related Sortino ratio. These are performance measures of investments which take into account the risk by considering an extra term, which is a risk-free asset (originally was suggested that U.S. treasure bonds be used here). Formally, the Sharpe ratio is defined as:

[3] https://github.com/ZhengyaoJiang/PGPortfolio

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

$$S_a = \frac{E[R_a - R_b]}{\sqrt{var[R_a - R_b]}} \tag{3}$$

where R_a is the asset return and R_b is the risk-free return, with E being the expectation over the excess return of the asset return compared to the risk-free return. var is the variance of the respective excess, so in the denominator we have the standard deviation of the excess. To accurately measure the risk of an investment strategy, this should be taken over comprehensive periods, where all types of transactions happen, thus we need a representative sample for the Sharpe ratio to give an accurate estimate of the risk.

5.1.1. Sortino ratio

The Sortino ratio is a modified version of the Sharpe ratio where negative outcomes (failures) are penalized quadratically. This can be written as:

$$S = \frac{R - T}{DR} \tag{4}$$

where R is the average return of the asset of interest, T is the target rate of return (also called MAR - minimum acceptance return) and DR is called the downside risk, and is the target semi-deviation (square root of the target semi-variance) and is formally, it's given by:

$$DR = \sqrt{\int_{-\infty}^{T} (T - r)^2 f(r) dr} \tag{5}$$

where r is the random variable representing the return (this might be annual) and $f(r)$ is the distribution of returns.

5.1.2. Differential Sharpe ratio

In an online learning setting, as is the case for RL (if using this measure as a reward shaping mechanism), a different measure is needed, which does not require all the returns to get the expectation over the returns, which are obviously not available in an online setting. To overcome this limitation, [45] has derived a new measure given by:

$$D_t = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}} \tag{6}$$

where A_t and B_t are exponential moving estimates of the first and second moment of R_t, where R_t is the estimate for the return at time t (R_T is the usual return used in the standard Sharpe ratio, with T being the final step of the evaluated period), so small t usually refers to the current timestep. A_t and B_t are defined as:

$$\begin{aligned} A_t &= A_{t-1} + \eta \Delta A_t = A_{t-1} + \eta(R_t - A_{t-1}) \\ B_t &= B_{t-1} + \eta \Delta B_t = B_{t-1} + \eta(R_t^2 - B_{t-1} \end{aligned} \tag{7}$$

5.2. Value at risk

Another measure for risk is the so called value at risk (VaR) [23,37,46], which specifies with probability at most α that a loss greater then VaR will occur and with probability at least $1 - \alpha$ a loss smaller than VaR will occur. It is the smallest number y such that the probability $(p(Y = -X) \leq y) \geq 1 - \alpha$. Formally, the VaR is defined as:

$$VaR_\alpha(X) = -inf\{x \in \mathbf{R} : F_X(x) > \alpha\} = F_Y^{-1}(1 - \alpha) \tag{8}$$

where X is the random variable which describes the profit and losses, F_X is the cumulative distribution function. This can also be seen as the $(1 - \alpha)$-quantile of Y. However, it implies that we assume a parametric form for the distribution of X, which is quite limiting in practice.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

5.3. Conditional value at risk

This is also called expected shortfall [23,46] or average value at risk and is related to VaR but it is more sensitive to the shape of the tail of the loss distribution. It is a more conservative way of measuring risk, since for the wins, the measure ignores the highest profits less likely to happen and for the losses it focuses on the worst ones. Formally, CVaR is defined as:

$$\begin{aligned} CVaR_\alpha(X) = -\frac{1}{\alpha}(E[X\mathbf{1}_{X<x_\alpha}] + x_\alpha(\alpha - P[X < x_\alpha])) \\ \text{with } x_\alpha \text{ the lower } \alpha\text{-quantile given by :} \\ x_\alpha = \inf\{x \in \mathbf{R} : P(X \leq x) \geq \alpha\} \end{aligned} \tag{9}$$

and $\mathbf{1}$ being the indicator function. For more details and for other measures of risk see [4].

5.4. Turbulence

In [47] they use the measure of turbulence to indicate if the market is in some extreme conditions. If this value is above some threshold, then the algorithm is stopped and all assets are sold. This quantity, it time-dependent and is defined as:

$$turbulence_t = (\mathbf{x}_t - \mu)\mathbf{\Sigma}^{-1}(\mathbf{x}_t - \mu)^T \tag{10}$$

where $\mathbf{x}_t \in \mathbb{R}^D$ is the return of the asset which we're computing the turbulence for at timestep t and $\mu \in \mathbf{R}^D$ is the mean return of the asset. $\Sigma \in \mathbf{R}^{D\times D}$ is the covariance of the returns, with D the number of assets.

5.5. Maximum drawdown (MDD)

The maximum drawdown is a measure of downside risk, it represents the maximum loss from a peak to a trough in some period, and is given by:

$$MDD = \frac{TV - PV}{PV} \tag{11}$$

where TV is the through value (the lowest drop an asset reaches before recovering) and PV is the peak value of an asset in the period of interest. Obviously this says nothing about the frequency of losses or about the magnitude or frequency of wins, but it is considered very informative nonetheless (usually correlated with other complementary measures), with small MDD indicating a robust strategy. The Calmar ratio is another measure which uses MDD as its only quantification of risk. For a comprehensive overview of Calmar ration, MDD and related measures see [48].

6. Deep Reinforcement Learning

Deep reinforcement learning is a recent field of ML that combines two powerful techniques, the data-hungry deep learning and the older process oriented reinforcement learning (RL). Reinforcement learning took birth as a heuristic approach, descendant of dynamic programming. Sutton and Barto in their seminal work [49] put the basis for a whole new field, which would turn out to be highly influential throughout the years, both in neuroscience and ML. In general, reinforcement learning was employed where the data was little and the behaviour was complex. However, recently, because of the advent of deep networks, reinforcement learning has received increased horsepower to tackle more challenging problems. For a comprehensive overview see [50–52]. The general reinforcement learning problem is defined by an agent acting, or making decisions in an environment, where the process is modelled as a Markov Decision Process (MDP).

The MDP is represented by a state space $\mathcal{S}$ comprising the states the agent can be in, defined by the environment, the action space $\mathcal{A}$, which is the set of actions an agent can

[4] https://en.wikipedia.org/wiki/Coherent_risk_measure

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

take, a transition dynamics which gives the probabilities that the agent has of being in a state at time t, taking an action and being in another state, at time $t+1$ i.e. $p(s_{t+1}|s_t, a_t)$, and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$. An MDP dynamics is assumed to satisfy the Markov property $p(s_{t+1}|s_1, a_1, ..., s_t, a_t) = p(s_{t+1}|s_t, a_t)$, that is the next state is dependent only on the previous state and action for any state-action space trajectory. This is generally verbalised as: the future is independent of the past given the present. The general goal of the agent is to *find a policy* which maximizes the discounted future reward, given by $R = \sum_{k=t}^{T} \gamma^{k-t} r_{s_k, a_k}$, with γ a real number called discount factor, with $0 < \gamma < 1$. The policy is modelled to be stochastic in general and is parameterized by a set of parameters θ, i.e. $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on the action space $\mathcal{A}$. The reward is assumed to be given by the environment, but we will see later that auxiliary reward functions (not given by the environment) can significantly improve an agent's performance. We will state briefly the main paradigms used in finding a good policy and then we will delve directly into the acclaimed DQN, the first DRL agent introduced in the machine learning community.

We consider the partitioning of the RL algorithms following Silver [5]. The three approaches are: **value-based RL**, where the value in each state (and action) is a kind of prediction of the cumulative future reward. In short, each state has an associated real value number which defines how good it is to be in one state (or to take a specific action being in one state). This is given by the state-value function, usually denoted by $V^\pi(s)$, where π is the policy which governs the agent, and which has one identifier, namely the state, or the state-action value function given by $Q^\pi(s, a)$ which has two identifiers namely the state and action. We will see later that new flavours of the value function exist, where additional identifiers are used, for example the current goal of the agent (in the case where there are multiple goals). In DRL, the value functions are outputs of deep networks.

The second approach is to represent the actual policy as a deep network, and this is referred to as **policy-based RL**. In general, the policy networks can be directly optimized with stochastic gradient descent (SGD).

The third approach is **model-based RL**, where the agents specifically construct a transition model of the environment, which is generally modelled by a deep network. This is sometimes hard, depending on the complexity of the environment but offers some advantages over the model-free approaches, for example, the model is generative, meaning that the agent can generate samples from its model of the environment and thus can avoid actually interacting with the environment. This can be highly beneficial when this interaction is expensive or risky. Having stated the main approaches we proceed to the description of the first DRL agent, often referred to as DQN (Deep Q Network) [6]. We show in Figure 5 the much acclaimed architecture of the original DQN. In DQN the powerful deep networks were employed for approximating the optimal action-value function, i.e.:

$$Q^*(s, a) = \max_\pi \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi\Big] \tag{12}$$

where r_t, a_t are the reward and action at timestep t, each future reward is discounted by a factor of γ and π is the policy that governs the agent's behaviour. For the rest of this document, if we do not specify with respect to what variables the expectation operator is considered, then it means it is with respect to the state variable. The iterative version is then given by:

$$Q^*(s, a) = \mathbb{E}\Big[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a')|s_t = s, a_t = a\Big] \tag{13}$$

In general, in reinforcement learning, the optimal Bellman value or action-value function is denoted with an upper *. DQN was able to play a few tens of Atari games with no special tuning of hyperparameters or engineered features. Learning just from pixel data, the agent

[5] http://icml.cc/2016/tutorials/deep_rl_tutorial.pdf

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

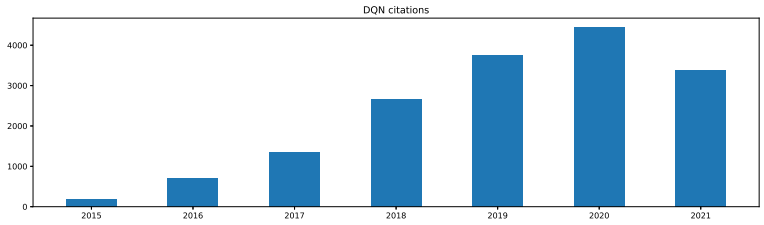


Figure 4. The number of articles citing the original DQN paper. Taken from Google Scholar.

was capable of learning to play all games with human or almost human performance. This is remarkable for an artificial agent. For the first time, an agent combined the representational power of deep nets with the RL's complex control, with two simple tricks.

The first one is *replay memory*, which is a type of memory that stores that last million frames (this number can vary as a function of the task, it is a hyperparameter of the model) from the experience of the agent and avoids the highly correlated consecutive samples arising from direct experience learning.

The second one was brought by replacing the supervised signal, i.e. the target values of the Q-network, or the optimal Q function from the Bellman equation with an approximation consisting of a previously saved version of the network, this is referred to as the *target network*. Using this approach defines a well-posed problem, avoiding the trouble of estimating the optimal value function. Then the loss function is given by (we use the notation used in the original DQN paper [6]):

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{14}$$

where $U(D)$ is the uniform distribution giving the sample from replay memory, γ is the usual discount factor, and θ_i^- is the set of parameters from a previous iteration (this is the second enhancement we mentioned above). This gives the following gradient for the loss:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right] \tag{15}$$

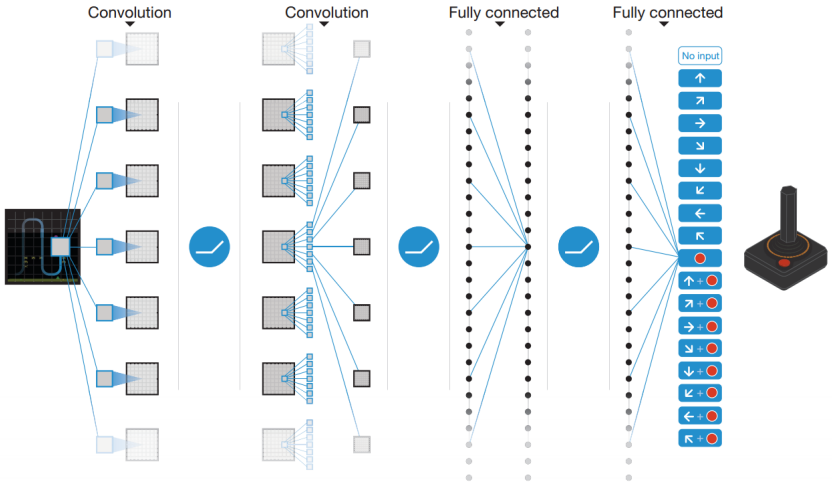
These two main additions enabled unprecedented performance and adaptability of the deep net, and enabled agent to deal with a relatively diverse set of task. Different games, but still the same domain, Atari 2600. The work that followed was immense, we show in Figure 4 the number of citations over the years.

We will talk about some of the ideas that extend the current approach, but functionally and theoretically. We will not be discussing practical concerns, even though we are aware of the critical importance of such works and the fact that they effectively enable these algorithms to run in an acceptable time. Deep learning is a relatively small subset of the whole machine learning community, even though it is progressing fast, at an unprecedented rate, due to the relative ease of use and ease of understanding but also due to the higher computational resources available. Being a visual task, CNNs perform best at playing Atari games, but other types of networks have been used, for example, for text based reinforcement learning as well [53,54]. More and more tasks will be amenable to DRL as the field embraces the new paradigm and the various existing approaches are adapted to fit these new understandings of what can be achieved. Trading has been of wide interest for a long time in the ML community and DRL has been viewed as a new set of powerful tools to be used for modelling and analysis on the financial markets.

We now proceed to describe different aspects of training a DRL agent to make profitable trades. The first critical problem is how we present the data for an agent, such that it gets all the information it needs in a compressed form for reduced dimensionality. This

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

Figure 5. The original architecture of the Deep Q Network that learnt to play Atari games from pixel data.



is refered to as the representation learning problem in RL [55] and recently it has been showed that this can be decoupled from policy learning [56].

7. Time-series encodings or state representation for RL

There are many ways to encode a time-series, from a simple moving average to high-rank tensors. We will discuss in detail the most widely used encodings for the financial markets. We are interested solely in trading based on price, and will not consider other alternatives, like for example, limit order books (LOB) [6]. In short, LOBs are lists of orders which actors on the market are willing to execute. They usually specify a price level and a quantity. Sometimes these are aggregated to give on single value for the price (for example first 10 buy orders and first 10 sell orders are weighted by their respective quantities to give a weighted average). We also skip the details of how the news / twitter feeds sentiments are encoded in the state.

7.1. Convolutions

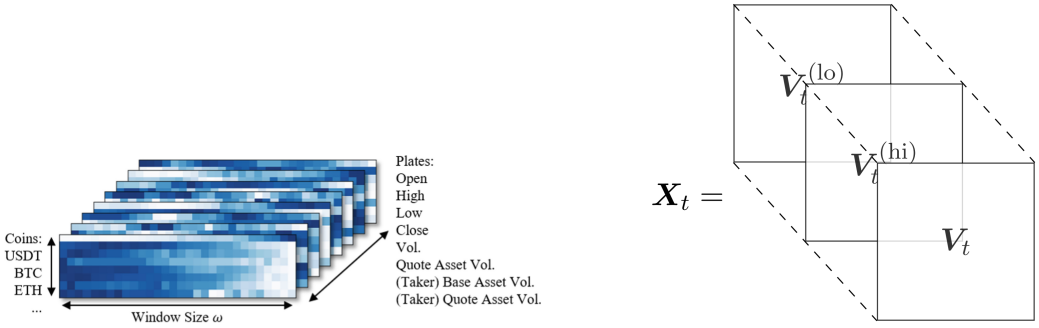
As Convolutional Neural Networks (CNNs) have been initially successfully used in vision [57], they gained popularity throughout the wider ML field, for example in Natural Language Processing (NLP) [58], anomaly detection [59], drug discovery [7] and many others. Among these, there is also the application to time-series encoding and forecasting [60]. As we are interested in RL, one way to do it for state representation of financial time-series is to concatenate the time-series of interest into a tensor, with one axis corresponding to time (usually there are multiple adjacent time-steps fed as a state, the last 5 or last 10 steps) and the other axis corresponding to the different quantities of interest (e.g. Open, Close, High, Low price of a candlestick). We show the graphical representation from [40] and [27] in Figure 6.

7.2. Raw-data OHLCV

Due to large variations in the price of different assets, we need to preprocess each one such that they have a standardized form (e.g. between 0 and 1). Using the return, v_t/v_{t-1} to normalize prices is often used in the financial market literature. The OHLCV means the Open, High, Low, Close and Volum of one time-point. Many works use this representation, concatenating the last few timesteps as a single state for each point. A variation on this is to obtain features from the raw data, which is often quite noisy, by preprocessing it with a

[6] https://www.investopedia.com/terms/l/limitorderbook.asp
[7] https://www.kqed.org/futureofyou/3461/startup-harnesses-supercomputers-to-seek-cures

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1



(a) Stacked data used for convolutional preprocessing. From [40]

(b) Another way of stacking prices for convolutions. From [27]

Figure 6. Different RL state representations for time-series built for being processed by CNNs.

Kalman filter and then extraction features with an Autoencoder and/or SVD for time-series [61].

7.3. Technical trading Indicators

These are analytical ways of transforming the time-series, developed by economists to summarize or highlight different properties of the time-series. A wide variety of technical indicators exist, see for example here [62] or some of their uses combined with DRL in [63], [64] or [65].

7.4. Correlated pairs

In one work [66] on the FX market, the authors use 3 related pairs to predict one. The same can be done for the crypto markets where we use inputs of correlated pairs of cryptocurrencies (e.g. for predicting BTCUSD we might used ETHUSD, BTCEUR, ETHBTC, etc.). This assumes there is a lot of information in the related pairs such that we get a more informative dynamics than in the predicted time-series alone.

7.5. Historical binning or price-events environment

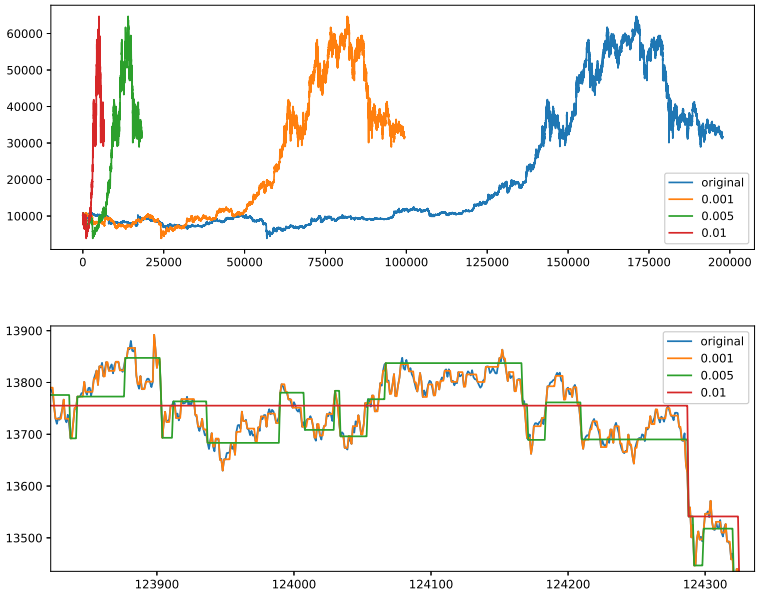
One interesting way to transform the continuos trading environment (i.e. price) is to discretize the price, transform the time-series such that it can take only a fixed number of values. If the price change is between the threshold values, than no effective change is seen by the DRL agent. This, in fact, compresses the time-series and transforms into a price-event based instead of time-event based [35]. It is a way of simplifying the time-series, making it more relevant for trading and removing some of the noise existent in the original data. Due to the shorter time-series produced in this way, agents using this transformation should train faster. We can also combine the two approaches, where we keep the time component but we still discretise the price value. We show both approaches in Figure 7. We embark on developing more types of environments (among them being also the one in Figure 7b, where we modify the incoming data on the fly, by for example artificially augmenting rises or drops. This should make agents more sensitive to one or the other and thus change their behaviour into being risk-seeking or risk-averse.

7.6. Adding various account information

Some of the authors add to the state of the RL agent the price or quantity of the last buy, or the current balance of the porftolio, or the specific stocks held and their respective quantities. There doesn't seem to be a general conclusion for what sort of information is useful in the state of the agent, as agents can be widely different and the optimal choice is specific to each. For more information see Table 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

(a) Preprocessing the price to compress it for a DRL agent.



(b) Preprocessing the price to compress it, but keeping the time component.

Figure 7. Different ways of preprocessing the price.

8. Action space

The action space is generally the simple discrete $\{buy, hold, sell\}$ but there are some which use more actions to denote a different number of shares or quantity to buy or sell, as in [68]. If dealing with multiple assets either a continuous multi-dimensional action space is used or the problem is split into selecting the number of shares / quantity (for example with a deep learning module [69,70]) and separately selecting the type of order (buy, sell).

9. Reward shaping

The reward is the most important part when dealing with a DRL trading agent. Most works use the direct profit from a transaction or a longer term profit (even after a full episode, but episodes can vary in length from a few hundred time-steps to thousands). The other group of reward functions are based on various measures of risk, for a comprehensive overview see [35]. Even the number of transactions can be used [71].

Table 1: Table summarising the main modelling choices from a small subset of the papers which use DRL for trading.

Article	Data	State Space	Action Space	Reward	Performance
Deep Reinforcement Learning Agent for S&P 500 stock selection [72]	28 S&P500 5283 days for 415 stocks	total return, closing price, volumes, quarterly earnings, the out-paid dividends, the declaration dates for dividends, the publication dates for the quarterly reports (d, n, f), where d is the length of the observation in days, n is the number of stocks in the environment, f is the number of features for a single stock	Continuous 20 dims	Differential Sharpe	1255 days Total return: 328% Sharpe=0.91
Diversity-driven knowledge distillation for financial trading using Deep Reinforcement Learning [73]	28 FX, 1h	normalized OHLCV, current open position as one-hot	Discrete (exit, buy, sell)	PnL	2 years PnL $\approx$ 0.35
A Deep Reinforcement Learning Approach for Automated Cryptocurrency Trading [74]	Bitcoin, 3200 hours	time stamp, OHLCV, USD volume weighted Bitcoin price	Discrete (buy, hold, sell)	Profit Sharpe	800h return $\approx$ 3%
An application of deep reinforcement learning to algorithmic trading [39]	30 stocks daily, 5 years	current trading position, OHLCV	Discrete (buy, sell)	Normalized price change	Sharpe = 0.4 AAPL annual return=0.32
Adaptive Stock Trading Strategies with Deep Reinforcement Learning Methods [63]	15 stocks daily, 8 years	OHLCV + Technical indicators (MA, EMA, MACD, BIAS, VR, and OBV)	Discrete (long, neutral, short)	Sortino ratio	3 years -6%-200%
Portfolio management system in equity market neutral using RL [75]	50 stocks daily, 2 years	normalized OHLC	Continuous 50 dims	Sharpe ratio	2 years profit $\approx$ 50%
Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy [47]	30 stocks daily, 7 years	balance, shares, price technical indicators (MACD, RSI, CCI, ADX)	Discrete $(2k+1)^{30}$	Portfolio value change Turbulence	Sharpe=1.3 Annual return=13%
Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism [76]	20 assets 30 minutes, 4 years	normalized HLC, shares	Continuous 20 dims	Log cumulative return	2 months Return = 22x
A framework of hierarchical deep q-network for portfolio management [26]	4 Chinese stocks 2 days, 2 years	OHLC, shares	Discretised $\times$ 4	Price change	1 year 44%
Attentive hierarchical reinforcement learning for stock order executions [22]	6 stocks 2 minutes, 2 months	OHLCV	Discrete (buy, sell)	Balance	1 month unclear%
Trader: Practical deep hierarchical reinforcement learning for trade execution [77]	35 stocks 1 minute, 1 year	OHLCV	Discrete (buy, hold, sell) and Continuous	Auxiliary: Surprise minimisation	unclear
Commission fee is not enough: A hierarchical reinforced framework for portfolio management [25]	46 stocks daily, 14 years	LOB, OHLCV	Discrete (buy, sell) and Continuous	Price change, Cumulative trading costs	Wealth $\approx$ 20
Model-based reinforcement learning for predictions and control for limit order books [78]	one stock 17s, 61 days	LOB, trade prints	Discrete 21 actions	Predicted price change	19 days cumulative PnL = 20
Model-based deep reinforcement learning for dynamic portfolio optimization [23]	7 stocks hourly, 14 years augmented	OHLC predicted next HLC market perf. index	Continuous	Predicted price change	2 years annual return = 8%

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

9.1. Model-based RL

In general, model-based methods are more data efficient, or have lower sample complexity, but are more computationally intensive. There are many approaches in the general RL literature, both parametric and non-parametric, most notably [79–81], but we are interested to describe in this work just the few approaches used in deep RL. Both model-free and model-based methods are known to exist in the brain [82], but the cognitive sciences differentiate between the type of task which employs each. Model-based decision making is shown to be used when the task is outcome-based, so one is interested in the unfolding in time of the sequence of states and actions (and sometimes rewards), while model-free methods are used when the decision making is mostly concerned with the value of an action (e.g. moral judgements, this action is good because it is moral). For DRL however, model-based approaches are quite scarce, probably due to the high complexity of models that need to be learned and the variety of models needed for the tasks that DRL is usually applied to. For example, TRPO deals with a number of classical control tasks and Atari games as well, it would be quite hard to derive a model building technique easily applicable to both of these domains. Even the Atari games are so different from one another, that a general model-building algorithm for all of them (the 50 games used more often in the DRL literature) would not be trivial.

Model-based RL in trading

As we mentioned earlier, model-based RL implies the creation of a model or simulation of the environment. This simulation is then run to produce different (if it's run multiple times and it is a stochastic model) possible prediction for the future. One such example is the work in [78], where they learn a model of the time-series which is a summary of a limit order book with an MDN after encoding (or extracting features from) the LOB with an AE [83]. They then use this model to learn from it with a RL agent and without requiring the interaction with the actual environment. In [23] they test two prediction models, the NDyBM and WaveNet adapted to multi-variate data. The predicted data is the Close, High, Low percentage change in the asset price value. With a couple of additional modules (data-augmenting module and behavioural cloning module) the overall architecture is applied on a small number of US stocks data successfully.

9.2. Policy-based RL

There are many approaches which use, a policy-based RL algorithms, combined with other techniques.

For example in [70] they use a supervised learning module (autoencoder) to predict the next time step of the price which was then used in signal representation for RL. This learns through a combined reward function both the prediction and the policy. They use also temporal attention and an interesting set of features in addition to the OHCLV price and a set of technical indicators, which is called cointegration and is a measure of the stationarity, or linear stability of a sum of two different stocks. There's also a preprocessing step of gate-based feature selection among all these, where a gate similar to the ones in GRU and LSTM, is present for each stock and is learned end-to-end.

There are many contributions and additions of this paper, training and testing on 2000 and 3000 steps respectively for 20 years with an annual rate of return of about 20% in bull and 10% in a bear market. A larger scale test (55 stocks) show 26% annual return. One of the most well-received papers is [27] where they apply their portfolio management DRL to 65 cryptocurrencies on Poloniex, on 30 minute data, concatenating the last 50 timesteps and all assets into one large tensor $(50, 65, 3)$ with 3 the number of features (high, low and the so-called return v_t/v_{t-1}, where the v_t is the open price for time t and v_{t-1} is the close price for time $t-1$. The algorithm used is DDPG [84] with the action-space $\in \mathbb{R}^{m+1}$ with m the number of coins (11 plus cash). The architecture is called EIIE (Ensembles of Identically Independent Evaluators) because to some degree each stream is independent and they only connect at the output, even though they are sharing the weights among them. They

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

also add the previous portfolio value before the softmax output as to minimise transaction costs. The results are remarkable, compared to the baseline strategies tested (up to almost $50x$ the initial investment). Interesting to note is the large training set (2 years) compared to the test set (50 days). They test 3 different topologies, a CNN, a simple RNN and an LSTM, with all three performing significantly better than any other strategy tested.

10. Hierarchical Reinforcement Learning

DRL works great on a wide class of problems, where the state space and action space are reasonably small (there are flavours of DRL, like Deterministic Policy Gradient [84] which can deal with continous action spaces as well, so in theory an infinite number of actions). However, there are certain problems where DRL is insufficient or takes too much time to train. A divide-and-conquer extension for DRL is Hierarchical Reinforcement Learning (HRL) where the overall problem is split into somewhat independent subproblems. Either the actions space is divided into smaller partitions or the state space or both. The most appealing aspect of HRL is that the agent higher in the hierarchy gives goal information to the level below and thus, rewards based on how well the agent is doing with respect to the goal being pursued. This means that auxiliary rewards (rewards which are given to the agent in addition to the environment rewards, usually defined by the programmer, see for example [85]) can be learned rather than hard-coded. This is a huge milestone in DRL since the rewards are fully governing an agent's behaviour. There are many different architectures for HRL, each inheriting some of the traits of the original flat RL, some with engineered dynamics between the agents, while others learn this as well. HRL holds the promise of scaling DRL to large sets of different problems, because agents in lower levels learn how to solve specific narrow tasks, or skills, whereas going up the hierarchy the agents learn more general behaviour and how to select between the agents in lower levels. Transfer learning also comes more naturally in HRL due to the advantages mentioned above.

HRL in trading

There is little research of HRL in trading, but some works exists, see for example [22,25,26,77]. HRL is used in different in ways in the mentioned works, we will discuss each in detail.

In one of the works, [26], the authors use HRL to manage a subset of the portfolio with an individual, low-level agent. Of course, the principle can be applied repeatedly, where a few flat agents (we call flat agents the lowest level agents which interact with the actual trading environment) are controlled by a higher level agent (level 1), then a group of agents at level 1 are controlled by an agent at the level 2 and so on. In this way, an arbitrary level hierachical agent can be formed.

This type of partitioning deals with the state space (since each DRL agent will receive only the prices for the stocks that is managing) and action space (will output only actions for specific stocks), but there is no shared dynamics between the levels of the hierarchy. However, there is information sharing in the form of the weight vector for each agent coming from the level above. Even though this approach is quite simplistic, it is also quite natural and partitions the overall problem into smaller subproblems which can be solved individually without loss of expressivity.

In a second line of work, [22] considers the problem of prediction and order execution as two separate problems and first uses one network to predict the price and another network which uses this prediction (concatenates it to the original state) to execute the orders. Even though this is not strictly speaking HRL, since the problem is not partitioned into smaller subproblems, and one cannot form an arbitrary level hierarchy (as in the previous work) based on these principle, there is no goal giving, etc. there is information sharing between the agents. We would classify this work in the the domain of multi-agent systems, where each agent is assigned one specific task and they communicate to reach the overall goal. Unrelated, but worth mentioning, is the fact that this work also applies a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

simple version of causal attention which seems to improve the profitability of the strategy on stocks.

A somehow related work (by the same principal author) uses HRL in [77] to now partition the trading problem into first estimating the quantity to be bought, held or sold and then actually choosing one of the three actions. This is somehow unintuitive since the operation to be performed seems more important than the actual quantity, so first choosing a quantity and the choosing what to do with it seems quite far-fetched. Again, there is no partitioning of the state-space (the quantity estimated is just concatenated to the original state) and no goal setting or reward giving by the high-level agent. However, there is an interesting contribution, the authors add a surprise term to the reward (coupled with the mellowmax operator) which makes the reward suprise agnostic, the authors state. One last work we found in this narrow *HRL for trading* subfield of RL is [25] which tackles a practical problem in trading, i.e. once we know what we want to invest in on the long run, how can we go about to minimise transaction cost in the short run. Even though this hierarchy only has two levels, it leverages the ability of HRL to deal with multiple time-scales to make decisions. The high-level agent decides which assets to hold for some larger period (they call it the holding period) and the low-level agent decides on how to buy these assets (and/or sell them to reach the desired portfolio configuration) such that transaction costs are minimised through a shorter period (the trading period). The authors use limit order books for states, which are higher dimensional and require more resources, see more details in section 7.

11. Conclusion

As a conclusion to this work we mention some of the commonalities between papers and some methodological issues which arise, and if solved could benefit the community significantly.

Common ground

- Many papers use convolutional neural networks to (at least) preprocess the data.
- A significant number of papers concatenate the last few prices as a state for the DRL agent (this can vary from 10 steps to hundreds).
- Many papers use the Sharpe ratio as a performance measure.
- Most of the papers deal with the portfolio management problem.
- Very few papers consider all three factors: transaction cost, slippage (when the market is moving so fast that the price is different at the time of transaction than at decision time) and spread (difference between buy price and sell price). The average slippage on Binance is around 0.1% [8] while the spread is around 1% with the maximum being even 8%, see Figure 8

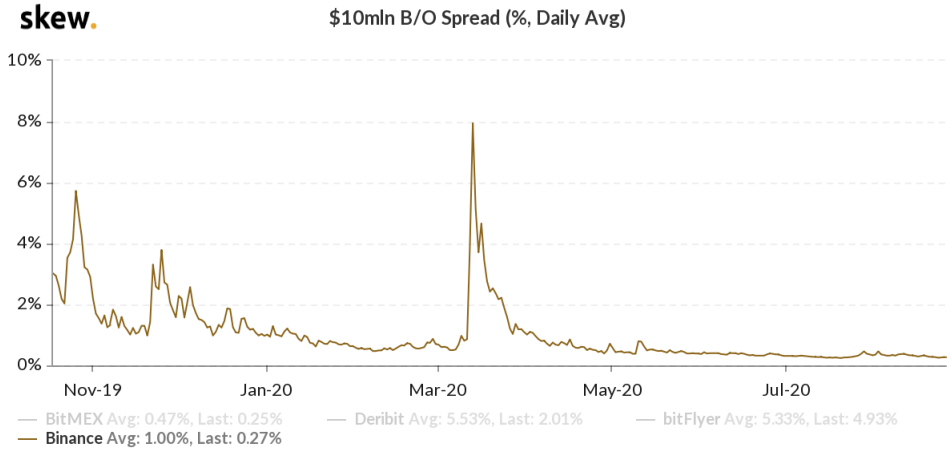
Methodological issues

- Inconsistency of data used. Almost all papers use different datasets, so it is hard to compare between them. Even if using the same assets (like Bitcoin), the periods used are different, but in general, the assets are significantly different.
- Different preprocessing techniques. Many papers preprocess the data differently and thus feed different inputs to the algorithms. Even if algorithms are the same, the results might be different due to the nature of the data used. One could argue that this is part of research and we agree, but the question remains: how can we know if the increased performance of one algorithm is due to the nature of the algorithm or the data fed?
- Very few works use multiple markets (Stock market, FX, Cryptocurrency markets, Gold, Oil). It would be interesting to see how some algorithm performs on different types of markets, if hyperparameters for optimal performance differ and if so, how

[8] https://blog.bybit.com/en-us/insights/a-comparison-of-slippage-in-different-exchanges/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2021 doi:10.20944/preprints202111.0044.v1

Figure 8. The Bitcoin spread on Binance.



they differ. Basically, nobody answers the question, how does this algorithm perceive and perform on different type of markets?

- Very few works have the code available online. In general, the code is proprietary and hard to reproduce due to missing details of the implementation, like hyperparameters or the deep neural network architecture used.

We showed in Table 1 a list of the most representative modelling choices with respect to the data used, its sampling frequency, the state space, the action space, the reward functions and the performance measures and values reported in the articles. This is not at all a comprehensive list, but a small set which seemed to the author to catch as much of the diversity in the whole corpus. The table is intentionally heterogeneous to reflect the difference in modelling as well as in reporting the different choices. Therefore, one conclusion which comes out from this study, is that there is need for more homogeneity in the methodology, data used, sampling frequency, preprocessing choice and reporting methods.

Funding: This research was funded by the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems at Imperial College London (HiPEDS, Grant Reference EP/L016796/1).

Conflicts of Interest: The author declares no conflict of interest.

Acknowledgments: The author thanks professor Abbas Edalat for helpful feedback.

References

1. Sato, Y. Model-free reinforcement learning for financial portfolios: a brief survey. *arXiv preprint arXiv:1904.04973* **2019**.
2. Hu, Z.; Zhao, Y.; Khushi, M. A survey of forex and stock price prediction using deep learning. *Applied System Innovation* **2021**, *4*, 9.
3. Fischer, T.G. Reinforcement learning in financial markets-a survey. Technical report, FAU Discussion Papers in Economics, 2018.
4. Mosavi, A.; Faghan, Y.; Ghamisi, P.; Duan, P.; Ardabili, S.F.; Salwana, E.; Band, S.S. Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics* **2020**, *8*, 1640.
5. Meng, T.L.; Khushi, M. Reinforcement learning in financial markets. *Data* **2019**, *4*, 110.
6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; others. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
7. Nakamoto, S.; Bitcoin, A. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* **2008**, *4*.
8. Islam, M.R.; Nor, R.M.; Al-Shaikhli, I.F.; Mohammad, K.S. Cryptocurrency vs. Fiat Currency: Architecture, Algorithm, Cashflow & Ledger Technology on Emerging Economy: The Influential Facts of Cryptocurrency and Fiat Currency. 2018 International Conference on Information and Communication Technology for the Muslim World (ICT4M). IEEE, 2018, pp. 69–73.
9. Tan, S.K.; Chan, J.S.K.; Ng, K.H. On the speculative nature of cryptocurrencies: A study on Garman and Klass volatility measure. *Finance Research Letters* **2020**, *32*, 101075.

10. Wang, J.; Sun, T.; Liu, B.; Cao, Y.; Wang, D. Financial markets prediction with deep learning. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2018, pp. 97–104.
11. Song, Y.G.; Zhou, Y.L.; Han, R.J. Neural networks for stock price prediction. *arXiv preprint arXiv:1805.11317* **2018**.
12. Selvin, S.; Vinayakumar, R.; Gopalakrishnan, E.; Menon, V.K.; Soman, K. Stock price prediction using LSTM, RNN and CNN-sliding window model. 2017 international conference on advances in computing, communications and informatics (icacci). IEEE, 2017, pp. 1643–1647.
13. Henrique, B.M.; Sobreiro, V.A.; Kimura, H. Stock price prediction using support vector regression on daily and up to the minute prices. *The Journal of finance and data science* **2018**, *4*, 183–201.
14. Vijh, M.; Chandola, D.; Tikkiwal, V.A.; Kumar, A. Stock closing price prediction using machine learning techniques. *Procedia Computer Science* **2020**, *167*, 599–606.
15. Rathan, K.; Sai, S.V.; Manikanta, T.S. Crypto-currency price prediction using decision tree and regression techniques. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, 2019, pp. 190–194.
16. Ke, N.R.; Singh, A.; Touati, A.; Goyal, A.; Bengio, Y.; Parikh, D.; Batra, D. Modeling the long term future in model-based reinforcement learning. International Conference on Learning Representations, 2018.
17. Moerland, T.M.; Broekens, J.; Jonker, C.M. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712* **2020**.
18. Pant, D.R.; Neupane, P.; Poudel, A.; Pokhrel, A.K.; Lama, B.K. Recurrent neural network based bitcoin price prediction by twitter sentiment analysis. 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS). IEEE, 2018, pp. 128–132.
19. Vo, A.D.; Nguyen, Q.P.; Ock, C.Y. Sentiment Analysis of News for Effective Cryptocurrency Price Prediction. *International Journal of Knowledge Engineering* **2019**, *5*, 47–52.
20. Clements, W.R.; Van Delft, B.; Robaglia, B.M.; Slaoui, R.B.; Toth, S. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638* **2019**.
21. Sebastião, H.; Godinho, P. Forecasting and trading cryptocurrencies with machine learning under changing market conditions. *Financial Innovation* **2021**, *7*, 1–30.
22. Suri, K.; Saurav, S. Attentive Hierarchical Reinforcement Learning for Stock Order Executions.
23. Yu, P.; Lee, J.S.; Kulyatin, I.; Shi, Z.; Dasgupta, S. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740* **2019**.
24. Lucarelli, G.; Borrotti, M. A deep Q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications* **2020**, *32*, 17229–17244.
25. Wang, R.; Wei, H.; An, B.; Feng, Z.; Yao, J. Commission Fee is not Enough: A Hierarchical Reinforced Framework for Portfolio Management. *arXiv preprint arXiv:2012.12620* **2020**.
26. Gao, Y.; Gao, Z.; Hu, Y.; Song, S.; Jiang, Z.; Su, J. A Framework of Hierarchical Deep Q-Network for Portfolio Management. ICAART (2), 2021, pp. 132–140.
27. Jiang, Z.; Xu, D.; Liang, J. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059* **2017**.
28. Shi, S.; Li, J.; Li, G.; Pan, P. A Multi-Scale Temporal Feature Aggregation Convolutional Neural Network for Portfolio Management. Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1613–1622.
29. Itoh, Y.; Adachi, M. Chaotic time series prediction by combining echo-state networks and radial basis function networks. 2010 IEEE International Workshop on Machine Learning for Signal Processing. IEEE, 2010, pp. 238–243.
30. Dubois, P.; Gomez, T.; Planckaert, L.; Perret, L. Data-driven predictions of the Lorenz system. *Physica D: Nonlinear Phenomena* **2020**, *408*, 132495.
31. Mehtab, S.; Sen, J. Stock price prediction using convolutional neural networks on a multivariate timeseries. *arXiv preprint arXiv:2001.09769* **2020**.
32. Briola, A.; Turiel, J.; Marcaccioli, R.; Aste, T. Deep Reinforcement Learning for Active High Frequency Trading. *arXiv preprint arXiv:2101.07107* **2021**.
33. Boukas, I.; Ernst, D.; Théate, T.; Bolland, A.; Huynen, A.; Buchwald, M.; Wynants, C.; Cornélusse, B. A deep reinforcement learning framework for continuous intraday market bidding. *arXiv preprint arXiv:2004.05940* **2020**.
34. Conegundes, L.; Pereira, A.C.M. Beating the Stock Market with a Deep Reinforcement Learning Day Trading System. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020, pp. 1–8.
35. Sadighian, J. Extending Deep Reinforcement Learning Frameworks in Cryptocurrency Market Making. *arXiv preprint arXiv:2004.06985* **2020**.
36. Hu, Y.; Liu, K.; Zhang, X.; Su, L.; Ngai, E.; Liu, M. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing* **2015**, *36*, 534–551.
37. Taghian, M.; Asadi, A.; Safabakhsh, R. Learning Financial Asset-Specific Trading Rules via Deep Reinforcement Learning. *arXiv preprint arXiv:2010.14194* **2020**.
38. Bisht, K.; Kumar, A. Deep Reinforcement Learning based Multi-Objective Systems for Financial Trading. 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE). IEEE, 2020, pp. 1–6.
39. Théate, T.; Ernst, D. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications* **2021**, *173*, 114632.

40. Bu, S.J.; Cho, S.B. Learning optimal Q-function using deep Boltzmann machine for reliable trading of cryptocurrency. International Conference on Intelligent Data Engineering and Automated Learning. Springer, 2018, pp. 468–480.
41. Cover, T.M. Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*; World Scientific, 2011; pp. 181–209.
42. Li, B.; Hoi, S.C. On-line portfolio selection with moving average reversion. *arXiv preprint arXiv:1206.4626* **2012**.
43. Moon, S.H.; Kim, Y.H.; Moon, B.R. Empirical investigation of state-of-the-art mean reversion strategies for equity markets. *arXiv preprint arXiv:1909.04327* **2019**.
44. Sharpe, W.F. Mutual fund performance. *The Journal of business* **1966**, *39*, 119–138.
45. Moody, J.; Wu, L. Optimization of trading systems and portfolios. Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), 1997, pp. 300–307. doi:10.1109/CIFER.1997.618952.
46. Gran, P.K.; Holm, A.J.K.; Søgård, S.G. A Deep Reinforcement Learning Approach to Stock Trading. Master's thesis, NTNU, 2019.
47. Yang, H.; Liu, X.Y.; Zhong, S.; Walid, A. Deep reinforcement learning for automated stock trading: An ensemble strategy. *Available at SSRN* **2020**.
48. Magdon-Ismail, M.; Atiya, A.F. An analysis of the maximum drawdown risk measure. *Citeseer* **2015**.
49. Sutton, R.S.; Barto, A.G. *Reinforcement learning: An introduction*; Vol. 1, MIT press Cambridge, 1998.
50. Li, Y. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* **2017**.
51. Mousavi, S.S.; Schukat, M.; Howley, E. Deep reinforcement learning: an overview. Proceedings of SAI Intelligent Systems Conference. Springer, 2016, pp. 426–440.
52. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* **2017**, *34*, 26–38.
53. Narasimhan, K.; Kulkarni, T.; Barzilay, R. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* **2015**.
54. Foerster, J.N.; Assael, Y.M.; de Freitas, N.; Whiteson, S. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *arXiv preprint arXiv:1602.02672* **2016**.
55. Heravi, J.R. *Learning representations in reinforcement learning*; University of California, Merced, 2019.
56. Stooke, A.; Lee, K.; Abbeel, P.; Laskin, M. Decoupling representation learning from reinforcement learning. International Conference on Machine Learning. PMLR, 2021, pp. 9870–9879.
57. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **2012**, *25*, 1097–1105.
58. Grefenstette, E.; Blunsom, P.; De Freitas, N.; Hermann, K.M. A deep architecture for semantic parsing. *arXiv preprint arXiv:1404.7296* **2014**.
59. Ren, H.; Xu, B.; Wang, Y.; Yi, C.; Huang, C.; Kou, X.; Xing, T.; Yang, M.; Tong, J.; Zhang, Q. Time-series anomaly detection service at microsoft. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 3009–3017.
60. Chen, Y.; Kang, Y.; Chen, Y.; Wang, Z. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* **2020**, *399*, 491–501.
61. Yashaswi, K. Deep Reinforcement Learning for Portfolio Optimization using Latent Feature State Space (LFSS) Module. *arXiv preprint arXiv:2102.06233* **2021**.
62. Technical indicators. https://www.tradingtechnologies.com/xtrader-help/x-study/technical-indicator-definitions/list-of-technical-indicators/. Accessed: 2021-06-21.
63. Wu, X.; Chen, H.; Wang, J.; Troiano, L.; Loia, V.; Fujita, H. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences* **2020**, *538*, 142–158.
64. Chakraborty, S. Capturing financial markets to apply deep reinforcement learning. *arXiv preprint arXiv:1907.04373* **2019**.
65. Jia, W.; Chen, W.; Xiong, L.; Hongyong, S. Quantitative trading on stock market based on deep reinforcement learning. 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019, pp. 1–8.
66. Rundo, F. Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems. *Applied Sciences* **2019**, *9*, 4460.
67. Millea, A. Deep Reinforcement Learning environments for trading. Work in progress.
68. Leem, J.; Kim, H.Y. Action-specialized expert ensemble trading system with extended discrete action space using deep reinforcement learning. *Plos one* **2020**, *15*, e0236178.
69. Jeong, G.; Kim, H.Y. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications* **2019**, *117*, 125–138.
70. Lei, K.; Zhang, B.; Li, Y.; Yang, M.; Shen, Y. Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. *Expert Systems with Applications* **2020**, *140*, 112872.
71. Hirchoua, B.; Ouhbi, B.; Frikh, B. Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. *Expert Systems with Applications* **2021**, *170*, 114553.
72. Huotari, T.; Savolainen, J.; Collan, M. Deep reinforcement learning agent for S&P 500 stock selection. *Axioms* **2020**, *9*, 130.
73. Tsantekidis, A.; Passalis, N.; Tefas, A. Diversity-driven knowledge distillation for financial trading using Deep Reinforcement Learning. *Neural Networks* **2021**, *140*, 193–202.

74. Lucarelli, G.; Borrotti, M. A deep reinforcement learning approach for automated cryptocurrency trading. IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, 2019, pp. 247–258.
75. Wu, M.E.; Syu, J.H.; Lin, J.C.W.; Ho, J.M. Portfolio management system in equity market neutral using reinforcement learning. *Applied Intelligence* **2021**, pp. 1–13.
76. Weng, L.; Sun, X.; Xia, M.; Liu, J.; Xu, Y. Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism. *Neurocomputing* **2020**, *402*, 171–182.
77. Suri, K.; Shi, X.Q.; Plataniotis, K.; Lawryshyn, Y. TradeR: Practical Deep Hierarchical Reinforcement Learning for Trade Execution. *arXiv preprint arXiv:2104.00620* **2021**.
78. Wei, H.; Wang, Y.; Mangu, L.; Decker, K. Model-based reinforcement learning for predictions and control for limit order books. *arXiv preprint arXiv:1910.03743* **2019**.
79. Deisenroth, M.; Rasmussen, C.E. PILCO: A model-based and data-efficient approach to policy search. Proceedings of the 28th International Conference on machine learning (ICML-11). Citeseer, 2011, pp. 465–472.
80. Abdolmaleki, A.; Lioutikov, R.; Peters, J.R.; Lau, N.; Pualo Reis, L.; Neumann, G. Model-based relative entropy stochastic search. *Advances in Neural Information Processing Systems* **2015**, *28*, 3537–3545.
81. Levine, S.; Koltun, V. Guided policy search. International conference on machine learning. PMLR, 2013, pp. 1–9.
82. Littman, M.L. Reinforcement learning improves behaviour from evaluative feedback. *Nature* **2015**, *521*, 445–451.
83. Hinton, G.E.; Zemel, R.S. Autoencoders, minimum description length, and Helmholtz free energy. *Advances in neural information processing systems* **1994**, *6*, 3–10.
84. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* **2015**.
85. Jaderberg, M.; Mnih, V.; Czarnecki, W.M.; Schaul, T.; Leibo, J.Z.; Silver, D.; Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* **2016**.