Diversity-driven Knowledge Distillation for Financial Trading using Deep Reinforcement Learning ☆

Avraam Tsantekidis, Nikolaos Passalis, and Anastasios Tefas

School of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece.

Abstract

Deep Reinforcement Learning (RL) is increasingly used for developing financial trading agents for a wide range of tasks. However, optimizing deep RL agents is notoriously difficult and unstable, especially in noisy financial environments, significantly hindering the performance of trading agents. In this work, we present a novel method that improves the training reliability of DRL trading agents building upon the well-known approach of neural network distillation. In the proposed approach, teacher agents are trained in different subsets of RL environment, thus diversifying the policies they learn. Then student agents are trained using distillation from the trained teachers to guide the training process, allowing for better exploring the solution space, while "mimicking" an existing policy/trading strategy provided by the teacher model. The boost in effectiveness of the proposed method comes from the use of diversified ensembles of teachers trained to perform trading for different currencies. This enables us to transfer the *common view* regarding the most profitable policy to the student, further improving the training stability in noisy financial environments. In the conducted experiments we find that when applying distillation, constraining the teacher models to be diversified can significantly improve their performance of the final student agents. We demonstrate this by providing an extensive evaluation on various financial trading tasks. Furthermore, we also provide additional experiments in the separate domain of control in games using the Procgen environments in order to demonstrate the generality of the proposed method.

☆This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T2EDK-02094). Avraam Tsantekidis was solely funded by a scholarship from the State Scholarship Foundation (IKY) according to the "Strengthening Human Research Resources through the Pursuit of Doctoral Research" act, with resources from the "Human Resources Development, Education and Lifelong Learning 2014-2020" program. We kindly thank Speedlab AG for providing their expertise on the matter of FOREX trading and the comprehensive dataset of 28 FOREX currency pairs.

Email address: avraamt@csd.auth.gr, passalis@csd.auth.gr, tefas@csd.auth.gr. (Avraam Tsantekidis, Nikolaos Passalis, and Anastasios Tefas)

Keywords: Deep Reinforcement Learning, Financial Markets, Trading

1. Introduction

In recent years, financial trading has seen an increase in participation from automated trading agents [1]. This growth has been driven by financial trading firms using automated strategies that they apply to all suitable markets. As similar trading strategies are deployed by an increasing number of trading firms, their individual efficacy and ability to outperform the market decays and at some point stops altogether. In turn, new strategies are being developed that take advantage of different aspects of the available data to produce profitable signals and are constantly monitored to ensure their continued effectiveness in the market. This has evolved into a race of ever-evolving complex strategies that combine a multitude of information sources to outperform competing strategies, aiming to make the most profitable trades.

The recent advancements in the field of Deep Learning [2] has provided information analysis tools for developing even more sophisticated automated trading agents [3, 4], which are able to effectively exploit the vast amount of data that are collected from financial markets, as well as from other related sources, such as news articles and social media [5, 6]. Deep Reinforcement Learning (DRL) specifically has seen major improvements [7, 8, 9], providing powerful agents trained to directly maximize the obtained profit [10, 11], instead of approximating the task at hand through handcrafted proxy problems. This can be better understood by examining the way traditional supervised DL is used to develop trading agents. Such supervised agents are trained using handcrafted labels, which require for their creation an arbitrary profit margin compared to commission and other costs. However, the actual profit is not always directly related to the aforementioned handcrafted labels, since various other parameters, such as the volatility of the market and the confidence of the agent can affect the resulting profit or loss. In contrast, when DRL is used, the trading profits can be tightly integrated in the agent's reward, along with every other market cost, since simulated trading environments can be developed and used for training the DRL agents. This removes the complexity of handcrafted label selection and allows the DRL optimization to determine which position is worth taking and have predictable outcomes based on the received reward. In this way, DRL agents are able to directly optimize the metrics related to the task at hand, i.e., the obtained profit, in simulated trading environments.

Even though being able to accurately model the profits as rewards seems enough to allow the utilization of DRL training algorithms to produce a profitable trading agent, this is far from the truth. The training instability of most DRL methods which results in different training runs of the same model leading to models with vastly different trading behavior. The variability across models trained in the same way significantly lowers the reliability of the obtained profits and, as a result, our trust in the resulting models. Recent works attempt to resolve these issues by improving the utilization of computational resources [12] to train multiple instances of RL agents in parallel. Multiple methodological improvements have also been presented, such as the Trust Region and Proximal Policy Optimization [13, 14], and a combination of previous works in Deep

Q-learning with significant improvements [15]. However, even with the latest improvements in DRL, the results can still be unreliable, even more so for the noisy and sometimes unpredictable financial markets. To alleviate these problems, specialized training procedures are required to improve training stability, when such RL methods are employed for the task of financial trading [10, 3].

In this work, we present a novel distillation method for DRL agents, to take advantage of the experience diversity present across subsets of the available data, by building upon the well-known neural network distillation approach [16]. Our method significantly improves the resulting performance of agents, which we attribute to the diversification constraint we apply. The main contribution of our work is the proposed method of appropriately distributing the available data in constraint chunks to different teacher models thus ensuring a "diverse curriculum" when distilling knowledge to student agents. Student agents are trained on a single subset of data and have knowledge distilled from multiple teachers. To ensure diversification in teachers, each of them is previously trained on separate subsets of data. These students significantly outperform similar students, whose teachers are all trained on a single constrained chunk of data, thus lacking diversity. We compare this type of "in-domain" distillation with our diversified approach and with the baseline performance of no distillation at all.

To further improve the effectiveness of the proposed method, we also introduce a diversified ensemble of teacher models trained to perform trading for different currencies. In this way, the *common view* regarding the most profitable policy is transferred to the student, allowing for further improving the training stability in noisy financial environments, where even small changes in the trading policy can lead to significant changes in the obtained profits. This provides an effective way to identify and generalize the *trading knowledge*, as encoded by multiple trading agents. At the same time, this process also alleviates the need of having direct data access to large or private datasets. This is because it is possible to utilize an already trained agent on such a dataset in lieu of the whole dataset itself. Thus, the proposed method, apart from providing an effective framework for training DRL agents in highly stochastic environments, such as financial markets, it also provides a way to reduce the reliance on direct access to private datasets, while still being able to take advantage of the potential knowledge available within them.

The rest of the paper is structured as follows. First, the related work is briefly presented and discussed in Section 2. Then, the proposed method is introduced and analytically derived in Section 3. The experimental setup, along with the experimental evaluation, are presented in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

Deep learning using neural networks has had many breakthroughs in the way they can process time series [17] and improve their training regimes [18]. Their application in financial forecasting has seen a lot of published works in

recent years [19, 20, 21]. Reinforcement Learning techniques such as the Q-learning approach combined with Deep Neural Networks have also seen many improvements recently. In Q-learning, the role of the Q-value table is taken by an estimator neural network which determines the potential benefits of each action available in an environment by assigning a value to each one. The training objective is to improve the estimation given the environment state. The improvements proposed in recent works succeeded in solving challenging tasks with large state spaces [7, 8, 9]. Although DQN is an important method for solving problems in the realm of DRL and has managed many impressive results [15], it is not sample efficient and requires longer time to train, relative to methods using the Policy Gradient (PG) approach. Training RL agents using PG is more efficient focusing on directly learning a policy instead of implicitly creating a policy where the agent greedily selects the action with the highest estimated Q-value.

Policy Gradients have also had major advancements in their potential due to works such as the A3C [8] and the Deep Deterministic Policy Gradient [9]. One of the most impactful works on Policy Gradients is the Trust Region Policy Optimization (TRPO) [14] in which the agent is not allowed to modify the predicted distribution of its learnt policy outside certain bounds, ensuring that trajectories with relative large advantages do not affect it as much as trajectories with negative advantages. This bounding allows the agent to smoothly explore policy improvements and yields much better results. One simplification of this work comes from Proximal Policy Optimization [13], where some costly computation for the TRPO bounding was removed and replaced with an addition to the training objective which loosely attempts to enforce the policy update bounds. This improvement retains the RL performance gains of TRPO while trimming the implementation and computational costs.

In financial trading utilizing direct price forecasts or handcrafted labels by experts to train supervised models is argued to be suboptimal in [22] while RL is more likely to yield better results by directly optimizing the performance. This is attributed to the large difference of transaction costs incurred by the labels used for the supervised learning compared to the transaction costs of the predictions by the models. This transaction cost consideration is solved in the context of RL applied to financial trading, and works such as [23, 10], explore directly using the profit or metrics relating to returns as the objective of an RL agent. Extending these approaches, [24, 11], provides an auxiliary RL objective of following the price to better regularise the training process allowing for obtaining better results.

Even with these considerable improvements to DRL, both Q-learning and PG approaches suffer from a great variability in the resulting performance in terms of mean rewards across multiple training trials with different initialization. At the same time, appropriately tuning the exploration-exploitation trade-off can be especially challenging, but also critical for achieving good generalization performance, especially for noisy tasks, such as for financial trading. To the best of our knowledge, this is the first work where a diversity constraint is used in conjunction with knowledge distillation to improve the performance of

DRL for both simple games and financial trading simulations, while providing an orthogonal approach to trust-region-based optimization, which can further improve the performance of such algorithms. It is worth noting that after an teacher agent is trained on a dataset, a student agent does not require direct access or training to that specific dataset. This is important because it alleviates concerns about sharing data, while allowing the exchange of latent information pertinent to the task at hand. Even though that neural network distillation has also been applied in deep RL, mainly aiming to transfer the knowledge between agents trained on different tasks [25, 26] and avoid forgetting older experiences [27], this is the first work that aims at improving the reliability of the training, as well as improving the trust on the resulting DRL agents.

3. Proposed Method

This paper focuses on learning DRL policies using Policy Gradient-based approaches, such as the Proximal Policy Optimization (PPO) [13]. This is without loss of generality, since the proposed method can also be directly applied for other methods, such as Q-learning-based approaches. First, the employed Reinforcement Learning methodology is briefly presented. Then, the proposed distillation-based method is analytically derived and discussed, along with the proposed diversity-driven distillation method.

3.1. Reinforcement Learning Methodology

Let $\pi(\alpha|s)$ denote the output of a Deep Reinforcement Learning (DRL) model that observes the state s and returns the probability for selecting the action α. In this paper, the Proximal Policy Optimization (PPO) method is used for training the deep RL method. PPO has managed state-of-the-art results in agents that have been trained using it in works such as [14, 13]. Initially, it was proposed as the Trust Region Policy Optimization (TRPO) [14], which enforced a constraint on the policy parameter steps. The used constraint dictates that the Kullback Leibler (KL) divergence between policy update steps remains within a hyperparameter ϵ:

$$D_{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta) \leq \epsilon, \tag{1}$$

where $\pi_\theta(\cdot)$ is a policy with parameters θ. The previous step parameters are denoted as θ_{old}. This method achieved impressive performance, but upholding the constraint for every update step of the policy is impractical. One heuristic that was proposed, which also provides similar results, is the Proximal Policy Optimization (PPO) that utilizes the action probability ratio between the policy parametrization across steps:

$$r_t(\theta) = \frac{\pi_\theta(\alpha|s_t)}{\pi_{\theta_{\text{old}}}(\alpha|s_t)}, \tag{2}$$

where $\pi_\theta(\alpha|s_t)$ is the probability that policy π will select action α when the agent observes environment state s_t at time step t. By utilizing a clipped version of

the ratio $r_t(\theta)$ around the value of 1 within ϵ, the policy exploration can be constrained to the close vicinity in the parameter space. We define the clipped policy ratio as:

$$r_t^{\text{clip}}(\theta) = \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon), \tag{3}$$

where ϵ is the constraint range of policy update. The final objective function is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min(r_t(\theta)A, r_t^{\text{clip}}(\theta)A)\Big], \tag{4}$$

where A_t is the advantage or in this case the Generalized Advantage Estimation (GAE) [28] of a step within the trajectory. Taking the minimum between $r_t(\theta)A_t$ and $r_t^{\text{clip}}(\theta)A_t$ allows for large negative advantages that adversely impact the performance to also greatly affect the objective, but limits the effect of positive advantages on the objective. This constraint limits the changes in the policy based on actions that yield excessive positive rewards but allows for larger changes that avoid excessive negative rewards. This results in a conservative exploration of the positively rewarded action paths without rushed policy updates towards the most obvious reward incline.

To calculate the GAE, we utilize the Temporal Difference (TD) residual for each time step t as follows:

$$\delta_t = R_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t), \tag{5}$$

where R_t is the reward the agent receives at time step t and $V^{\pi}(s_t)$ is the value estimation predicted by the critic of policy π for the current state s_t. Then the advantage A_t is defined as:

$$A_t = \sum_{i=0}^{n-t} (\gamma \cdot \lambda)^i \cdot \delta_{t+i} \tag{6}$$

where n is the total number of steps within a rollout and t is the time step.

To accumulate the required rollouts to train the RL agents with the objectives described above, an experience replay memory [29, 30] is utilized. The experience replay stores the pertinent information about every trajectory that was simulated by the agents, such as the reward, the predicted state values, and action probabilities. This information stored in the experience replay memory is then sampled during the optimization steps to train the neural networks that make up the actor and critic components of the agent.

3.2. Knowledge Distillation for Improving Deep Reinforcement Learning for Financial Trading

Let $\pi^{(t)}(\alpha|s)$ denote the action probability distribution output of the teacher model, while $\pi^{(s)}(\alpha|s)$ denote the output of the student model. In this work, we assume that a softmax activation function is used for transforming the raw output of the employed model into a probability distribution, while the notation $y(\alpha|s)$ is used to refer to the *logits*, i.e., the raw output of the network. The

teacher can be trained with any RL method, however, the method used for training the teacher should be compatible with the method used for training the student models, e.g., both should policy-based or value-based. The teacher model can be then used to transfer its knowledge into the student model. This process may also allow for training smaller and more efficient agents, while also ensuring their good performance, since they will be trained to jointly optimize the employed RL objective, as well as get as much knowledge as possible from the teacher model. Using a good performing teacher can be crucial, so we suggest a) employing strong regularization techniques to ensure the good performance of the teacher model, while also b) running several training trials and selecting the best performing model using a validation set.

Neural network distillation is employed in this work to transfer the knowledge from the teacher model into the student model [16]. Initially proposed for classification tasks, neural network distillation proposes to generate soft-labels using the teacher model, by raising the temperature of the softmax activation, to train the student model. This process acts as a regulariser and provides additional knowledge to the student model, since the probability distributions estimated by the stronger teacher model contain additional information regarding the similarity of training samples in each class.

We propose using neural network distillation to transfer the knowledge from the teacher model by producing a softer version of the probability distribution over the available actions, as estimated by the teacher:

$$q(\alpha|s) = \frac{\exp\left(\frac{y^{(t)}(\alpha|s)}{T}\right)}{\sum_a \exp\left(\frac{y^{(t)}(a|s)}{T}\right)}, \tag{7}$$

where $y^{(t)}(\alpha|s)$ are the logits of the teacher model. Similarly, a soft version of the output of the student model is calculated as:

$$p(\alpha|s) = \frac{\exp\left(\frac{y^{(s)}(\alpha|s)}{T}\right)}{\sum_a \exp\left(\frac{y^{(s)}(a|s)}{T}\right)}, \tag{8}$$

where $y^{(s)}(\alpha|s)$ are the logits of the student model. Then, the loss function we optimise to transfer the latent knowledge of the teacher to the student is defined as:

$$\mathcal{L}_D = -\frac{1}{N}\sum_{s\in\mathcal{S}}\sum_{i=0}^{k} q(\alpha_i|s)\log\left(p(\alpha_i|s)\right), \tag{9}$$

where $\mathcal{S}$ is a set of N states used for the optimization, as sampled from the experience replay memory, k is the number of available actions, and α_i is the i-th available action.

The final loss function for the proposed method is obtained by combining the RL loss $\mathcal{L}_{RL}$ with the proposed distillation-based loss as:

$$\mathcal{L} = \mathcal{L}_{RL} + \beta\mathcal{L}_D, \tag{10}$$

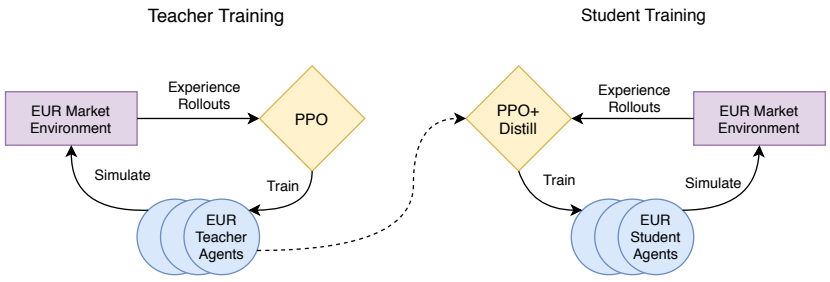


Figure 1: Training of teacher agents that specialize in a single currency domain, which then is used to train student agents.

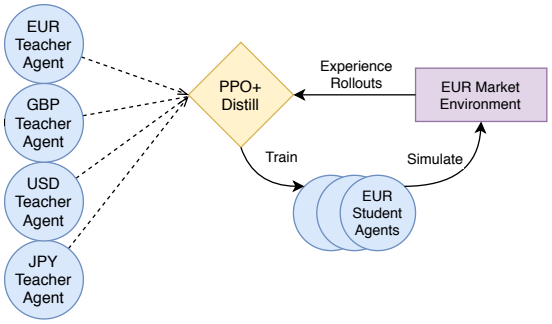


Figure 2: Training a student agent from a diverse set of trained teachers. (One teachers from each currency group)

where β defines the weight of the distillation loss (typically set to 1). To better understand why the proposed distillation-based approach allows for improving the stability of DRL optimization, we should consider that $\mathcal{L}_D$ always provide gradients towards an established policy that is trusted (as provided by the teacher model), while the DRL loss $\mathcal{L}_{RL}$ provides gradients based on the current noisy reward that has been obtained during the optimization. Therefore, the proposed method allows for providing consistent and noise-free descent directions during the critical early stages of the optimization, while also reducing the effect of suboptimal exploration policies. It is worth noting that viewed under the prism of (2), the proposed method ensures that the optimization will proceed towards a trusted region of the optimization space.

3.3. Robust Knowledge Distillation using Diversity-driven Teacher Ensembles

Training a robust teacher model is not straightforward, since it is equivalent to the problem at hand. To overcome this limitation, while still allowing for exploiting the benefits provided by the proposed method, we suggest training a diversified ensemble of teacher models that will encode the common knowledge

about trading in different financial instruments. The employed pool will be more robust to overfitting phenomena that often occur when training DRL agents for solving noisy trading tasks.

A brief description of FOREX pairs is the following: Most currencies have trading pairs between them allowing for the exchange of one currency for the other. An example of pairs are the EUR/USD pair allowing to exchange euros for US dollars and vice versa. Each exchange pair price is different and evolves based on the underlying "value" of each currency and to the speculative trades made by investors. The exchange pairs are grouped based on the currencies they consist of. The EUR/USD pair is included in two different groups, the EUR group and the USD group. Similarly, the USD/JPY pair is included in the USD and JPY groups. This allows us to train teachers in a diverse manner by constraining their access to the data of a single group. Since each pair denotes the exchange between two currencies, there is a minor overlap in the groups. Using the data of each limited group, agents are independently trained as shown in Figure 1. The diversified teachers ensemble is then used to train student agents. The individual teacher agents are trained using the PPO as described in Section 3.1.

Utilizing the developed pool of diversified teachers, a new agent can be trained combining the PPO objective and the distillation objective, as already described in (10). The student agents start sampling rollouts (or experiences) from the simulation environment, backed by a specific currency data group (e.g., EUR currency pairs). Every time a certain number of new rollouts are sampled, the student is optimized based on the current set of rollouts. During the optimization process, the teacher and student policy distributions are calculated. The distillation objective is determined using the softer version of the policy distribution (defined in (8), in conjunction with the PPO objective. The objectives are combined as a single loss which the model is trained to minimize using an optimizer based on stochastic gradient descent. The aim of the distillation objective is for the student model to have a more consistent objective that affects the optimization process less from the variability inherent in RL training. To simultaneously transfer the knowledge encoded by multiple teachers, the average of the distillation loss imposed by each teacher is used:

$$\mathcal{L}_D = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{Di}, \tag{11}$$

where N is the number of teachers and $\mathcal{L}_{Di}$ is the distillation loss of teacher i.

We explore two different approaches to distillation in RL for financial trading. First, we examine whether distillation is beneficial on its own, and then we demonstrate the benefits of using diversified teachers for RL students. The difference between using teachers in a specific domain and teachers from multiple domains is shown in Figure 2. Although the teachers from different domains/currency groups may have never "seen" some of the data in the current student's domain, the knowledge that is transferred through distillation can still be useful and improve the student's performance as we show later in the

conducted experiments.

4. Experimental Evaluation

In this section, the dataset and the appropriate feature extraction pipeline are first presented. The trading agent architecture is introduced followed by the distillation regimens that are tested. In Section 4.3 we independently show that knowledge distillation is beneficial to student RL agents and its improvements increase with more teachers and higher distillation factor. Then in Section 4.4, teacher agents are trained in separate subsets of the datasets. Then teacher pools are formed with either a diverse set of teachers that were trained on different subsets of the dataset and in-domain teacher pools that were trained on the same subset of the dataset. It is shown that students having knowledge distilled by the diverse teacher pools outperform those that were distilled by the in-domain teacher pools. An important detail is that for both distillation approaches, the students are trained on the same subsets of the dataset, and the difference being the subsets of the dataset where their teachers were trained.

4.1. Dataset

The dataset used to drive the simulation that interacts with the RL agents consists of FOREX trading data of multiple currency pairs, such as EUR/USD, EUR/GBP, and GBP/JPY. The trade data are subsampled using the Open-High-Low-Close (OHLC) price level technique, which reduces the trading data into 4 values for every time interval specified (e.g., 1 hour interval). Namely, these values are the open price or the first traded price of the set interval, the highest and lowest traded prices within the interval and finally the last (close) price that a trade occurred during the interval. This subsampling technique gives a clear picture of the price movements across time intervals. We utilize price intervals of 1 hour as our simulation stepping interval.

The observation provided to the agents consists of a window of past price candles, which has been preprocessed so that it does not exhibit massively different scales of values between asset pairs. This allows us to train the agents across multiple currency pairs and gain insight on the agent's ability to learn useful patterns from all of them. The preprocessed features are constructed as follows:

1. $\frac{p_h(t)}{p_c(t)} - 1$
2. $\frac{p_l(t)}{p_c(t)} - 1$
3. $\frac{p_c(t)}{p_c(t-1)} - 1$
4. $\frac{p_h(t)}{p_h(t-1)} - 1$
5. $\frac{p_l(t)}{p_l(t-1)} - 1$

The high, low, and close price are denoted as $p_h(t), p_l(t), p_c(t)$ respectively, where t denotes the time step. The high and low prices are the maximum and

minimum prices of trades that occured during the interval at time t while the close price is the price of the last trade that occured during said interval. All features aim to represent percentage distances between the sampled price values within the same interval or across intervals. Features 1 and 2 represent the price range within a single candle in terms of its fraction to the close price of that candle. Features 3, 4 and 5 represent the change of the close, high, and low price between each consecutive candle. Features 1 - 5 are concatenated into a vector and defined as $\mathbf{x}_f(t)$ for each time step t. The observation also contains the position of the agent in the market at that time. This is represented by a one-hot vector of size 3, where $[1, 0, 0]$ means that no position is currently taken, $[0, 1, 0]$ means that a long position is currently active and $[0, 0, 1]$ means a short position is currently active. The position one-hot vector is defined as $\mathbf{x}_p(t)$ for each time step t.

The total number of available currency pairs is 28 and their available trading data spans from about 2008 to the middle of 2018. In our experiments we utilize the range 2008-2016 for training and validation and 2016-2018 to test the resulting agents. In total, the dataset contains about 1.7 million data points, each representing one subsampled hour of trading data. The developed simulation environment provides three available actions, which consist of investing by buying the observed asset (long), selling the asset (short) or exiting the current position and staying out of the market (exit). The reward the agent receives is the profit or loss that is incurred by the position currently held. When the agent changes the current position held, a commission is incurred to make the change, which is set to be similar to the FOREX brokerage IB. Each episode that is simulated consists of a total of 500 time steps.

The reward and performance of a trading agent is measured using the Profit and Loss (PnL) metric. Its value represents the percentage of gains or losses achieved based on the trading decisions taken. This means that if an agent achieves a PnL of 0.05 it is translated to profit of 5%, whereas a PnL of -0.1 is a loss of 10%. This means that an investment of 100$ yield a 0.05 PnL would become 105$ in simple terms. In actuality, the size of the investment would matter for many aspects of the PnL since commissions and slippage are affected by it. We choose a sensible commission and slippage for all trades to simplify the simulation process.

The PnL is measured using the price time series of the asset along with the position an agent takes while stepping through it. We considered a static lot size for every position opened, thus allowing us to deal only with the percentage changes of our theoretical initial investment. This means that profits or losses are not added to the lot invested at each time step thus making the backtest less biased to past profits or losses from the agent. To create the final performance curve, the cumulative sum PnL in every time step is calculated.

4.2. Trading Agent

The employed neural network architecture consists of a Long Short-Term Memory (LSTM) Recurrent Neural Network [31] layer, followed by 2 fully connected layers that use Parametric Rectified Linear Units (PReLU) as activation

function to preprocesses the input time series and produce a representation for each time step. This representation is then given as input to two branches composed of fully connected (FC) layers. The first branch estimates the value of the current state (critic) based on the hidden representation provided, while the second outputs a probability distribution over the 3 available actions (exit, buy, sell). The architecture was chosen for its simplicity so that it can reliably train agents without excess overfitting as determined by preliminary experiments. The method we present is not constrained by the architecture itself and can be applied to different architectures. A graphical representation of this architecture is shown in Fig. 3. The agents are optimized using RMSProp method.

The PPO method is used for training the models towards maximizing the expected reward as described in Section 3.1. The loss function employed by the PPO method is denoted by $\mathcal{L}_{RL}$. The advantage is estimated using the Generalized Advantage Estimation (GAE) approach [28], which is derived from the Temporal Difference (TD) residual of each timestamp in the trading trajectories. The critic network is trained by minimizing the Huber loss [32] between the estimated value and the target values, with the cutoff point set to 1.

Both teachers and students have the same architecture parameters, with their LSTM consisting of 64 hidden units and their fully connected layers with 16 neurons each. During training, each agent's policy is evaluated for a total of 20k rollouts which are sequentially added to a replay memory holding a maximum size of 3k rollouts. When the replay memory is full, the oldest saved rollouts are ejected as new rollouts are inserted to maintain the 3k maximum rollout memory capacity. For every 4 rollout simulations completed and appended to the replay memory, the model is trained for 128 iterations on batches of 4 randomly selected rollouts from the replay memory, using the PPO training methodology described in Section 3.1. For the value ϵ defined in Eq. 3 0.2 is used. For the gradient descent momentum optimizer, RMSProp is utilized with a learning rate of 0.002, which is decayed linearly until the end by a factor of 10, i.e. 0.0002.

4.3. Market-wide Distillation

In this section we present the effects of using distillation when all the market data is available to every agent trained. By using distillation from a pool of trained agents, increased performance is observed for agents that have more teachers (Fig. 5) and larger distillation factors (Fig. 4). This improvement is attributed to the better training stability provided by the teachers as well as the smoother exploration afforded by the distillation process and the PPO algorithm.

4.3.1. Distillation factor

Initially, the teacher agents' policies are trained on the dataset described in Section 4.1 without any distillation. In Equation 10 the distillation component of the loss $\mathcal{L}_D$ is weighted using term β. We first conduct a set of experiments testing the benefit of the distillation cost as its weight β is increased across

Table 1: Backtest PnL for different currency groups with the examined distillation methods

Distillation	None	In-domain Distill.	Proposed
JPY	0.18 ± 0.07	0.22 ± 0.05	$\mathbf{0.31 \pm 0.09}$
USD	0.21 ± 0.03	0.31 ± 0.03	$\mathbf{0.34 \pm 0.06}$
EUR	0.29 ± 0.02	0.30 ± 0.01	$\mathbf{0.34 \pm 0.03}$
GBP	0.42 ± 0.05	0.45 ± 0.09	$\mathbf{0.47 \pm 0.07}$

experiments. In Figure 4 it is clear that as the distillation cost weight increases, so does the average PnL evaluated in test data. To address potential issues with RL instability, we have trained 4 different initializations for each β value and plotted the average value across them.

The performance of the agents is gradually improved as β increases from 0.0 to 1.0, at which point it shows no clear improvement as it increases further. We can conclude that using a weight of 1.0 for the distillation cost should be sufficient for augmenting the training of an agent, by distilling and transferring the knowledge from other already trained agents.

4.3.2. Number of teachers

Further improvement of the distillation process could be yielded by increasing the diversity of teachers used to distill knowledge to a student agent. By using multiple separately trained teacher agents to distill a single student agent, the performance should increase because the student gets more consistent gradients due to the distillation. We examine this statement experimentally by training multiple teacher agents. The experimental setup is identical to the one in Section 4.3.1.

To incorporate multiple teachers in the training process, the distillation cost $\mathcal{L}_D$ is averaged across them according to Eq. 11. In the experiments presented in Figure 5, the number of teachers tested ranges from 0 to 8. Each experiment is conducted 10 times with different initialisation for the student and the final test PnL is averaged across the experiments.

The results presented in Figure 5, indicate that having more teachers contributing to the averaged distillation loss increases the final performance of the student agent. Similarly to the distillation factor experiment, in Section 4.3.1, increasing the number of teachers yields diminishing returns after a certain point.

4.4. Market Subset distillation

In this section, the benefit of having a diversely trained teacher pool is demonstrated. When distilling knowledge to student agents, even though only a limited subset of the dataset is used, having a diverse teacher pool trained

can improve the results significantly more that a in-domain teacher pool. Although the diverse teacher agents may have never been optimized on the data used to distill knowledge to the student agents, they can still impart important information that leads to better performance by the students.

This approach offers the unique benefit in cases where market data is not available, or only available under expensive licensing that disallow their direct usage by third parties. A model trained in a restricted dataset by the copyright holder may be utilized by other parties without access to the said dataset. Using the trained model, third parties can distill their own models needing only the datasets under their ownership. In this experiment, we aim to show that distilling knowledge from teacher models that have been trained in diverse and potentially unavailable data sources can provide a bigger benefit than simply distilling only from teachers trained with a single limited source of data.

The data were split into 4 groups. Each group contains FOREX pairs including a specific currency, i.e., EUR, GBP, JPY, and USD. This means that the "EUR group" contains only the currency pairs EUR/USD, EUR/AUD, EUR/CAD, EUR/GBP, EUR/JPY, EUR/NZD and, EUR/CHF. Each group has a total of 7 currency pairs. For each currency group, four agents are trained with different initializations, using the same setup as the experiment in Section 4.3.1. In this manner, we obtain four teacher agents for each of the currency groups.

For each currency group, the in-domain teacher pool distills the knowledge to 8 separate and differently initialized student agents for a total of 32 students across all groups. The resulting performance of the students that were trained by the in-domain pools are labeled as "in-domain distilled", since they were distilled from teachers trained within the same domain, using the same limited data subset that trained their teachers.

To create diverse teacher pools, we pick a teacher from each in-domain pool and form 4 groups of 4 teachers each, where each teacher was trained in a different currency group. Thus 4 diversified teacher pools are created. Using the new diversified pools, 8 students are trained with distillation using the same exact process as mentioned before for the in-domain distillation. The difference is that the diversified teacher pools are now distilling their policies into the students' policies. The dataset used for this purpose is again the limited subset of each currency group to preserve the fairness of the comparison. Even though most of the data in the subset have never been seen by some of the teachers in the diversified pool, the knowledge that is distilled improves the final performance of the student models, which can be seen in Fig. 6 and 7.

In the results shown in Table 1, the proposed distillation outperforms, in every case, the distillation baseline (denoted as in-domain). Both the proposed and baseline distillation approaches outperform the agents trained without the use of distillation. The detailed mean PnL plots across the experiments can be observed in Figure 6, where it is clear that broad distillation outperforms the other methods in all cases, some with large margins and a few with a smaller one. Finally, we also backtest the same agents on all available data, including the currency pairs that they were not trained on. In the all-currency backtest,

Hypothesis	t-statistic	p value
ND $\not\equiv$ IDD	2.649	0.008
IDD$\not\equiv$ DD	3.081	0.002
ND $\not\equiv$ DD	3.462	0.0005

Table 2: Wilcoxon rank sum tests statistics and significance values for each hypothesis tested. Samples are No Distilation (ND), In-Domain Distillation (IDD) and Diversified Distillation (DD).

shown in Figure 7, the broad distillation offers the best mean PnL result. The better performance of the proposed distillation can be attributed to the more diverse set of teachers improving the student ability to generalize through the combined distillation from each of the teachers trained in different groups of currency pairs.

To ensure the statistical significance between the resulting performances, we conduct a series of hypothesis tests. For the first test the null hypothesis states the performance of non-distilled agents is the same as in-domain distilled agents. The second null hypothesis states that in-domain distillation performance is the same as the proposed diversified distillation.

The Wilcoxon ranked sum test is applied and supplemented. The paired values used are the PnL performance of each asset from each of the compared experiments. The null hypothesis is rejected in both cases with a confidence of $p = 0.008$ for the comparison of no distillation to in-domain distillation and $p = 002$ for the comparison of in-domain to proposed diversified distillation. The results are also shown in Table 2.

4.5. Diversifying on Procgen

The proposed diversification method is also examined in a domain other financial trading. The Procgen Benchmark [33] is utilized using the same model as the one proposed in the original work, namely, an Impala CNN with 3 residual blocks. All hyper-parameters used for the model are the same as the original work, except the total environment observations. A total of 30M observations are made (instead of the original 200M) for each training trial run, to allow for multiple trials in a reasonable time frame.

The previously described training regime is applied to Procgen, i.e., 4 teachers are trained for each "group" of data (500 levels of Procgen), with a total of 4 different groups, totaling 16 teachers. For the In-Domain distillation, students are trained on a group of 500 levels with the addition of the distillation loss of the 4 teachers previously trained on the same group of levels. For the diversified distillation, students are again trained on a group of 500 levels, but the distillation loss is derived from 4 teachers, each of which has been trained on different groups of 500 levels.

We utilize the starpilot and fruitbot environments for this test and show that the students trained with diversified teachers outperform the other group

Distillation	In-Domain	Proposed
Starpilot	7.73 ± 0.2	$\mathbf{8.24 \pm 0.4}$
Fruitbot	2.39 ± 2.4	$\mathbf{4.83 \pm 2.1}$

Table 3: Results comparison between In-Domain and diversified distillation on the Procgen benchmark with 30M observations used to train each agent. The reported performance is the average score of 250k steps in unlimited number of levels including unseen levels.

of students. The diversified approach to distillation outperforms the simpler approach as shown in Table 3.

The resulting performance in both environments tested shows that using diversified teachers for distillation is beneficial, although the levels used to train the students are the same in both cases.

5. Conclusions

A novel reinforcement learning distillation regime to transfer the knowledge from teacher trading agents to student agents is presented in this paper. First, it is experimentally demonstrated that distilling knowledge from multiple trained teacher agents to a student agent improves the final trading performance of the student agent even in simple scenarios. Increasing the number of teacher models improves the performance of the student models, with diminishing returns after 4 teacher models. Secondly, we show that this effect is also enhanced as we increase the distillation factor. Finally, we experimentally validate that training teachers in a constraint subset of currency pairs available, thus enforcing diversity between teachers, significantly improves the performance of the students. The improvements span the entire dataset even if the student distillation only utilized a constraint subset of it. The results show that this diversification of teacher agents can improve the student performance both in terms of the mean PnL backtested within the constraint dataset group but also the mean PnL the entire dataset.

In the future, an interesting area to explore is the interaction of diversified agents across different types of assets such as stocks and bonds in conjunction with the currency approach presented here. Such groups of assets exhibit vastly different behaviour from each other, which may prove useful as a diversification strategy aimed towards better distillation.

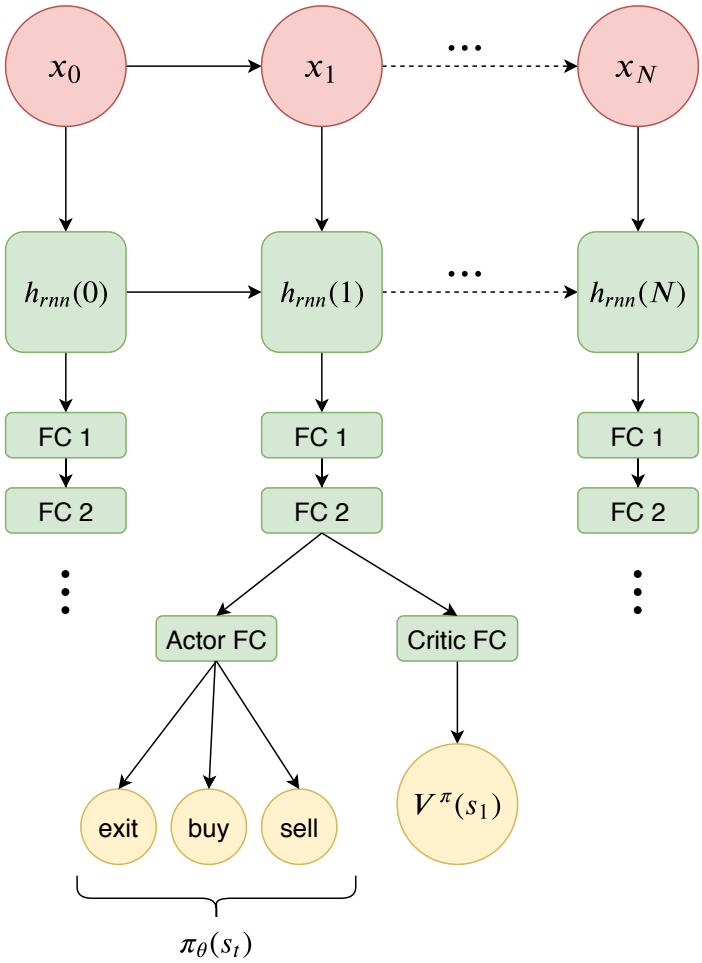


Figure 3: Actor-Critic model architecture for an episode of size N. The final layers are depicted only for the first step, where the actor and critic parts of the agent branch out from the same hidden representation. However, this process is repeated for every step of the episode.

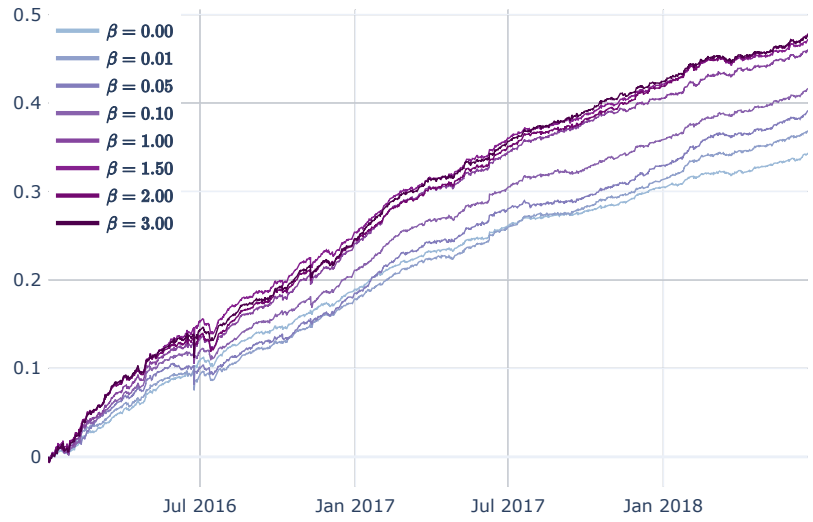


Figure 4: Average PnL of agents trained using different β values as defined in Equation 10. Each PnL represents the average of 4 separate runs with different initial random seeds.

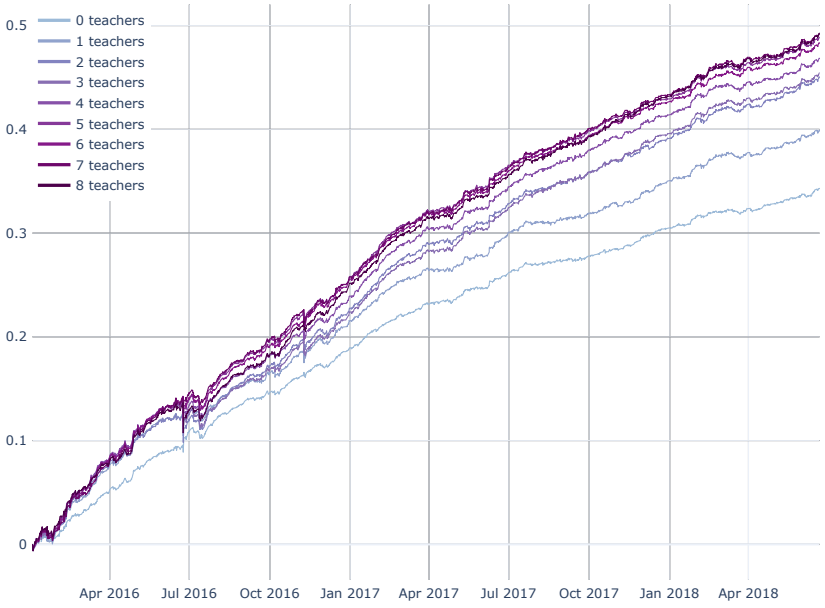
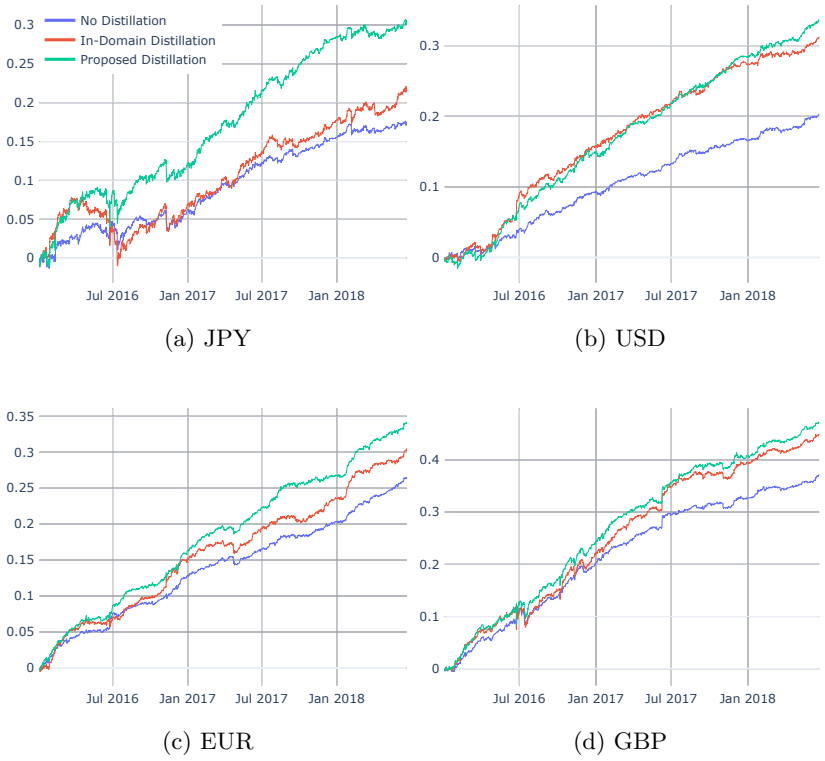


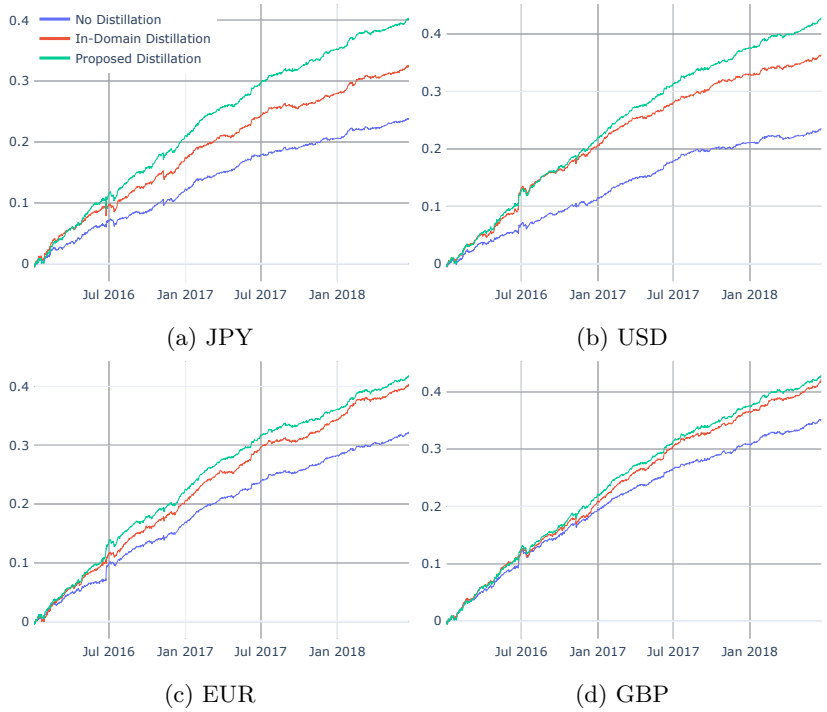
Figure 5: Average test PnL of agents trained using different number of teachers.



(a) JPY (b) USD

(c) EUR (d) GBP

Figure 6: Comparison of mean PnL between agents with no distillation, in-domain distillation and broad distillation. The PnL is obtained from backtesting in of currency pairs within the limited group, the agents where trained in.



(a) JPY (b) USD

(c) EUR (d) GBP

Figure 7: Comparison of mean PnL between agents with no distillation, in-domain distillation and the proposed distillation. The PnL is obtained from backtesting in all available currency pair to show the generalizations of each training method.

References

[1] R. Haynes, J. S. Roberts, Automated trading in futures markets, CFTC White Paper.

[2] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.

[3] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Forecasting stock prices from the limit order book using convolutional neural networks, in: Proc. IEEE Conf. on Business Informatics, Vol. 1, 2017, pp. 7–12.

[4] Z. Zhang, S. Zohren, S. Roberts, Deeplob: Deep convolutional neural networks for limit order books, IEEE Transactions on Signal Processing 67 (11) (2019) 3001–3012.

[5] T. H. Nguyen, K. Shirai, J. Velcin, Sentiment analysis on social media for stock movement prediction, Expert Systems with Applications 42 (24) (2015) 9603–9611.

[6] N. Oliveira, P. Cortez, N. Areal, The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices, Expert Systems with Applications 73 (2017) 125–144.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: Proc., Int. Conf. on Machine Learning, 2016, pp. 1928–1937.

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

[10] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Trans. on Neural Networks and Learning Systems 28 (3) (2016) 653–664.

[11] A. Tsantekidis, N. Passalis, A.-S. Toufa, K. Saitas-Zarkias, S. Chairistanidis, A. Tefas, Price trailing for financial trading using deep reinforcement learning, IEEE transactions on neural networks and learning systems PP. doi:10.1109/tnnls.2020.2997523.
URL https://doi.org/10.1109/TNNLS.2020.2997523

[12] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, I. Stoica, Ray rllib: A composable and scalable reinforcement learning library, arXiv preprint arXiv:1712.09381.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: Int. Conf. on Machine Learning, 2015, pp. 1889–1897.

[15] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, in: Proc. AAAI Conf. on Artificial Intelligence, 2018.

[16] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531.

[17] N. Mammone, C. Ieracitano, F. C. Morabito, A deep cnn approach to decode motor preparation of upper limbs from time–frequency maps of eeg signals at source level, Neural Networks 124 (2020) 357–372.

[18] D. Roy, P. Panda, K. Roy, Tree-cnn: a hierarchical deep convolutional neural network for incremental learning, Neural Networks 121 (2020) 148–160.

[19] J. Wang, J. Wang, Forecasting stochastic neural network based on financial empirical mode decomposition, Neural Networks 90 (2017) 8–20.

[20] A. Grigorievskiy, Y. Miche, A.-M. Ventelä, E. Séverin, A. Lendasse, Long-term time series prediction using op-elm, Neural Networks 51 (2014) 50–56.

[21] R. d. A. Araújo, Hybrid intelligent methodology to design translation invariant morphological operators for brazilian stock market prediction, Neural Networks 23 (10) (2010) 1238–1251.

[22] J. Moody, L. Wu, Y. Liao, M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, Journal of Forecasting 17 (5-6) (1998) 441–470.

[23] J. Moody, M. Saffell, Learning to trade via direct reinforcement, IEEE Trans. on Neural Networks 12 (4) (2001) 875–889.

[24] K. S. Zarkias, N. Passalis, A. Tsantekidis, A. Tefas, Deep reinforcement learning for financial trading using price trailing, in: Proc. Int. Conf. on Acoustics, Speech and Signal Processing, 2019, pp. 3067–3071.

[25] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, R. Pascanu, Distral: Robust multitask reinforcement learning, in: Proc. Advances in Neural Information Processing Systems, 2017, pp. 4496–4506.

[26] G. Berseth, C. Xie, P. Cernek, M. Van de Panne, Progressive reinforcement learning with distillation for multi-skilled motion control, arXiv preprint arXiv:1802.04765.

[27] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, G. Cai, N. Díaz-Rodríguez, D. Filliat, Discorl: Continual reinforcement learning via policy distillation, arXiv preprint arXiv:1907.05855.

[28] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438.

[29] P. Wawrzyński, Real-time reinforcement learning by sequential actor–critics and experience replay, Neural Networks 22 (10) (2009) 1484–1497.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (7540) (2015) 529–533.

[31] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[32] P. J. Huber, Robust estimation of a location parameter, in: Breakthroughs in statistics, 1992, pp. 492–518.

[33] K. Cobbe, C. Hesse, J. Hilton, J. Schulman, Leveraging procedural generation to benchmark reinforcement learning, arXiv preprint arXiv:1912.01588.