2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

IMPROVING DEEP REINFORCEMENT LEARNING FOR FINANCIAL TRADING USING NEURAL NETWORK DISTILLATION

Avraam Tsantekidis, Nikolaos Passalis, 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 known to be notoriously difficult and unstable, hindering the performance of financial trading agents. In this work, we propose a novel method for training deep RL agents, leading to better performing and more efficient RL agents. The proposed method works by first training a large and complex deep RL agent and then transferring the knowledge into a smaller and more efficient agent using neural network distillation. The ability of the proposed method to significantly improve deep RL for financial trading is demonstrated using experiments on a time series dataset consisting of Foreign Exchange (FOREX) trading pairs prices.

Index Terms— Deep Reinforcement Learning, Financial Markets, Trading

1. INTRODUCTION

Reinforcement Learning (RL) methods have been widely applied in many challenging fields with great success [1, 2, 3]. However, in many cases, they suffer from a significant degree of training instability, with multiple runs leading to vastly different results, increasing the training variability and, as a result, lowering the *reliability* of the models. In recent literature the resolution of this issue has been pursued through several methodological breakthroughs. Simpler heuristic approaches include the increase in the number of agents trained as well as the better utilization of the computational resources in training RL agents to ensure some of agents end up in good performing parameter regions [4]. More sophisticated approaches include methodological improvements, e.g., [5, 6, 7], which can improve the convergence reliability, but also reduce the sample efficiency in training.

Although these and other similar approaches can remedy, to some extent, the aforementioned issue, the problem still remains on many domains. In this work we explore this problem in the domain of financial trading where the inherent noise of the financial markets exacerbates the issue of reliability when training deep RL agents. It is worth noting that financial data presents a significant challenge for machine learning methods, including deep RL methods, since the intrinsically noisy nature of the data limits the amount of information that can be gained from observing them. Also, it is worth noting, that in many cases the outcome of similar observations can end up being vastly different. As a result, such excessive noise often overwhelms the useful information within the data, rendering standard training methods unable to properly fit the deep learning models to the data.

At the same time, deep RL also often employs large and especially complex models, increasing the computational requirements both for training and deploying the agents and reducing their flexibility. This can be a limiting factor in many applications, where fast inference, such as placing high frequency limit orders [8, 9], or fast re-training, such as online adaptation of the model to the current market conditions or domain adaptation [10], is required. In these cases, both the large variability of the training process, along with the computational power needed to deploy/train the model pose significant challenges.

The main contribution of this work is the proposal of a novel method that can overcome the aforementioned limitations, improving both the performance, in terms of acquired profit for the examined use case, and the efficiency of deep RL agents. The proposed method works by first training a pool of large, carefully tuned and regularised RL agents. Then, the best of these agents is used to guide the training process by transferring its knowledge into a smaller and faster model, which can fulfill the inference speed requirements. This is achieved by employing the well-known neural network distillation approach [11]. This allows for efficiently training the smaller *student* model, while also transferring the knowledge that has been encoded by the larger *teacher*, which is known to perform well on the target domain. This process has two benefits: a) it allows for training significantly smaller models that perform almost the same as the larger ones, and b) increases the reliability of the training process by mimicking a model that is known to perform well. Indeed, in this paper it is experimentally demonstrated that the proposed method can significantly improve the performance of the student model,

E-mail: avraamt@csd.auth.gr, passalis@csd.auth.gr, tefas@csd.auth.gr

978-1-7281-6662-9/20/$31.00

almost matching the performance of the teacher, while vastly reducing the complexity of the model. The proposed extension of the neural neural distillation approach provides a concrete framework for applying it in RL settings, paving the way for similar methods that can further improve the performance of deep RL, especially when used in noisy environments, such as those typically encountered in various financial settings.

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

Reinforcement learning methods have seen great success with the introduction of powerful neural network models into the internal estimations of the RL agents' environments and the potential return of their actions. In the case of Q-learning, the agent attempts to estimate the value of each potential action given the observation of the environment. Employing the Deep Q Network (DQN) extension to that approach has granted the ability for RL agents to solve very difficult tasks with exponentially large state spaces [1, 2, 3]. Although DQN through its numerous extensions, such as [12, 5], has managed great results, it still requires a large amount of training iterations and environment interactions to achieve them. Another approach for training an RL agent that achieves better sample efficiency is the Policy Gradient approach where the goal is to directly learn an optimal policy instead of implicitly learning a policy through the state-action values. This approach has some advantages over Q-learning, such as directly predicting a probability distribution over the available actions, providing a more straightforward way to tune exploration and exploitation. Also, policy gradient-based methods predict probability distributions over the action-space, allowing them to be directly applied on continuous action spaces, which is especially difficult for the Q-leaning approaches. Unfortunately, both families of RL approaches lead to a significant variability, in terms of their performance (e.g., mean reward), across multiple training trials.

Several methods have been proposed to reduce the variability and stabilize the training process of deep RL agents, e.g., using dueling and target networks [13], trust region optimization and its extensions [6, 14], stochastic averaging [15], and others. The proposed method is orthogonal to these approaches: it exploits a good snapshot of an already trained agent to ensure that the optimization will always be near a good region of the solution space. To the best of our knowledge, this is the first time neural network distillation is employed to this end and combined with the Proximal Policy Optimization method [6] on the challenging domain of financial trading. Note that neural network distillation has also been applied in deep RL, but mainly aiming to transfer the knowledge between agents trained on a different tasks [16, 17] and avoid forgetting older experiences [18].

Also, trading in the context of RL has been previously implemented in works such as [19, 20] where the profit or metrics relating to returns have been defined as the objective of an RL agent. Extending these approaches, [21], provides an auxiliary RL objective of following the price in order to better regularise the training process allowing for obtaining better results. It is worth noting again that the proposed method is generic and can be readily combined with any of these approaches, as demonstrated using a specific scenario in the experimental evaluation, and further improve their performance.

3. PROPOSED METHOD

This paper focuses on learning deep RL policies using Policy Gradient-based approaches, such as the Proximal Policy Optimization (PPO) [6]. This is without loss of generality, since the proposed method can also be directly applied for other methods, such as Q-learning-based approaches. Let $\pi(\alpha|s)$ denote the output of the deep RL model that observes the state s and returns the probability for selecting the action α. More specifically, let $\pi^{(t)}(\alpha|s)$ denote the 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.

First, a strong teacher model $\pi^{(t)}(\alpha|s)$ is trained. We propose 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. Note that 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 a smaller and faster model. This process allows 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.

Neural network distillation is employed in this work to transfer the knowledge from the teacher model into the student model [11]. 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 can act as a regulariser and provides additional knowledge to the teacher model, since the probability distributions estimated by the stronger teacher model contain additional infor-

mation regarding the similarity of training sample with 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:

$$q(\alpha|s) = \frac{\exp\left(\frac{y^{(t)}(\alpha|s)}{T}\right)}{\sum_a \exp\left(\frac{y^{(t)}(a|s)}{T}\right)}, \quad (1)$$

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:

$$p(\alpha|s) = \frac{\exp\left(\frac{y^{(s)}(\alpha|s)}{T}\right)}{\sum_a \exp\left(\frac{y^{(s)}(a|s)}{T}\right)}, \quad (2)$$

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

$$\mathcal{L}_D = -\frac{1}{N}\sum_{s\in\mathcal{S}}\sum_{i=0}^{n} q(\alpha_i|s)\log\left(p(\alpha_i|s)\right), \quad (3)$$

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

In this work the PPO method is used for training the models towards maximizing the expected reward. The loss function employed by the PPO method is denoted by $\mathcal{L}_{RL}$. The policy change ratio is clipped within a range of ϵ, where typically $\epsilon = 0.2$. The advantage is estimated using the Generalized Advantage Estimation (GAE) approach [22], which is derived from the Temporal Difference (TD) residual of each timestamp in the trading trajectories. Also, the critic network is trained by minimizing the Huber loss between the estimated values and the target values [23], with the cutoff point set to 1. Therefore, the final loss function for the proposed method is composed by combining the RL loss with the proposed distillation-based loss as:

$$\mathcal{L} = \mathcal{L}_{RL} + \beta\mathcal{L}_D, \quad (4)$$

where β defines the weight of the distillation loss (typically set to 1).

4. EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed method is provided in this Section. First, the employed dataset is briefly introduced in Subsection 4.1. Then, the developed training agent is presented in Subsection 4.2. Finally, the evaluation results are presented and discussed in Subsection 4.3.

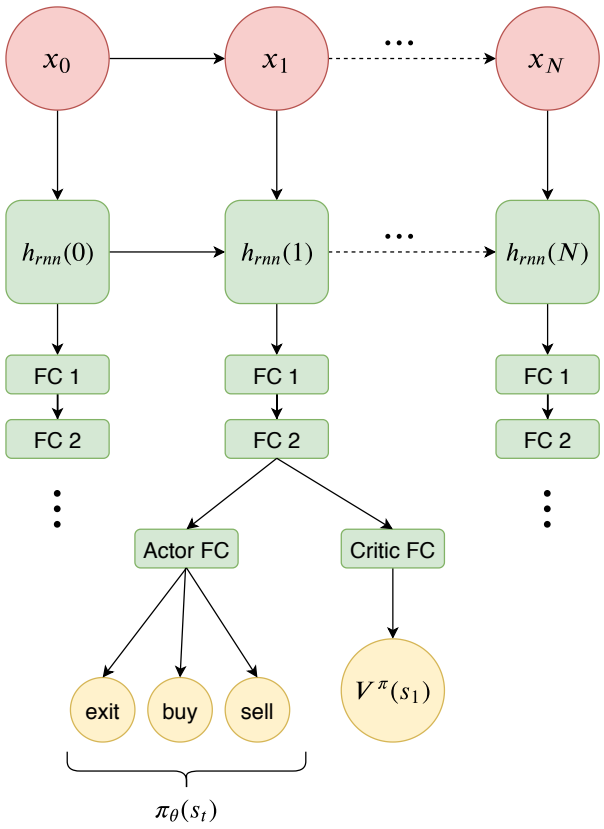

Fig. 1. Actor-Critic model architecture for an episode of size N. The final layers are depicted only for step 1, where the actor and critic parts of the agent branch out from the same hidden representation. However, this process is repeated for every step of the episode. $\pi_\theta(s_i)$ denotes the probability distribution predicted by the policy π_θ given state. $h_{rnn}(t)$ is the hidden state of the rrnn at time t s_i.

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 trading data are subsampled using the Open-High-Low-Close (OHLC) price level technique [24], which reduced the trading data into 4 values for a specified time 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 price that a trade took place during the interval. This subsampling technique gives a clear picture of the price movements across the time intervals. In this work we utilize price intervals of 1 hour as our simulation stepping interval.

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-

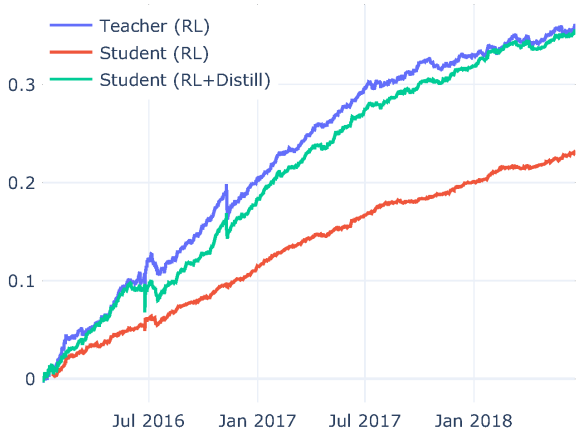

Fig. 2. Comparison of the mean Profit and Loss (PnL) between the evaluated deep RL agents

2016 for training 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 a well known FOREX IB.

The observation from the environment that is fed to the RL agent is a set of features extracted from the OHLC price data that represents the percentage returns across timesteps. These consist of the percentage change of the high, low and close prices across timesteps and the percentage difference between the current close price to the current high and low prices. Each episode that is simulated consists of a total of 500 timesteps.

4.2. Trading Agent

The employed neural network architecture consists of a Long Short-Term Memory (LSTM) Recurrent Neural Network [25] layer, followed by 2 fully connected layers that use Parametric Rectified Linear Units (PReLU) as activation function to preprocesses the input timeseries and produce a representation for each time step [26]. 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). A graphical representation of this architecture is shown in Fig. 1. The agents are optimized using RMSProp method [27].

We use two different network architectures, a powerful and more complex one for the teacher model and a more lightweight for the student model. The teacher is equipped with an LSTM with 1024 units and fully connected layers with 256 neurons each. We utilize various techniques to increase the performance of the model: dropout with $p = 0.5$ [28], weight decay [29] by reducing the scale of each weight in the network based proportionally to its value, as well as Stochastic Weight Averaging [15] by keeping an average of the weights of the network for the final 25% of the training process using the average as the final weights of the network. The student models are created using an LSTM layer with size of 128 and FC layers with 128 hidden units each. No extra regularization is employed for the student models.

4.3. Evaluation

To evaluate the agents' performance the Profit and Loss (PnL) metric is used which measures the percentage return of some initial investment. The performance of the teacher and student models is compared in Fig. 2. The best performing teacher, as selected through the conducted experiments, is used for the evaluation, while the "Student (RL)" and "Student (RL+Distill)" refer to the average of eight runs using the baseline students (using only the RL objective) and the students trained with the proposed method respectively. For all the conducted experiments, we set $T = 1$ following the suggestions of [30]. Note that the proposed method not only leads to significant improvements over the baseline student model, but it also almost matches the performance of the teacher model, despite being significantly less complex. This behavior demonstrates the ability of the proposed method to effectively transfer the knowledge encoded in a large teacher model into a smaller student model. It also suggests that even though training a large and complex teacher model might be necessary to better explore the solution space, the policy can effectively represented using a smaller model.

A total of 8 students are trained and their performance can be observed in Figure 3. The baseline students' performance ends up being worse than the more capable teacher as expected. We then train 8 more students from scratch but we also add a distillation cost to their RL objective (Figure 4) thus allowing the better trained teacher to guide each of the students by transferring some of its latent knowledge. Indeed, note that the 8 students that are trained using the distillation cost manage to achieve significantly higher returns, which can be easily verified by comparing Figures 3 and 4. Finally, the results in Figure 4 demonstrate that the top performing distilled students outperform even the teacher (Figure 2, leading us to the conclusion that the knowledge transfer from the teacher has the potential to benefit the individual student agents to even surpass the teacher, further strengthen-

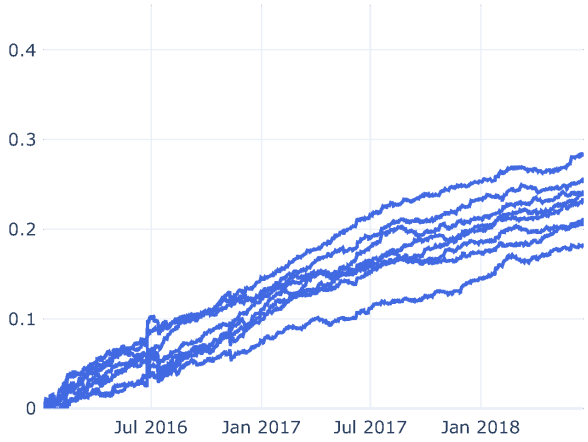

Fig. 3. PnL for individual baseline student agents

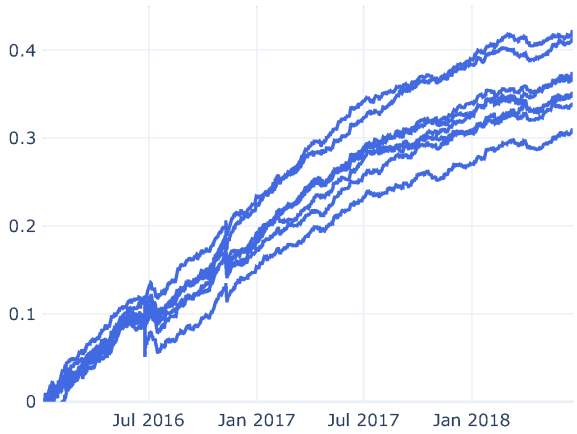

Fig. 4. PnL for individual distilled student agent

ing our claim that this method strongly regularises the fitting procedure of RL agents, allowing for improving the training stability and consequently boosting the performance of the final agent.

5. CONCLUSIONS

A novel method that allows for distilling the knowledge from one larger and more complex teacher RL agent to another smaller and more lightweight student is presented in this paper. It is experimentally demonstrated that when multiple smaller agents are trained on the same task, none ends up extracting sufficient knowledge alone to achieve anything close to the teacher agent's performance. However, employing the proposed method allows for significantly improving the performance of the student models, matching the performance of the teacher (or even exceeding it in some cases). Even though the variability of the training process is not fully minimized, due to optimizing a joint RL-distillation objective, the proposed method performs significantly better, demonstrating the effectiveness of the proposed approach.

6. REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the, International Conference on Machine Learning*, 2016, pp. 1928–1937.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Joseph Gonzalez, Ken Goldberg, and Ion Stoica, "Ray rllib: A composable and scalable reinforcement learning library," *arXiv preprint arXiv:1712.09381*, 2017.

[5] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz, "Trust region policy optimization," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 1889–1897.

[8] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proceedings of the IEEE Conference on Business Informatics*, 2017, vol. 1, pp. 7–12.

[9] Zihao Zhang, Stefan Zohren, and Stephen Roberts, "Deeplob: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019.

[10] Vahid Behbood, Jie Lu, and Guangquan Zhang, "Fuzzy refinement domain adaptation for long term prediction in banking ecosystem," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1637–1646, 2013.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[12] Oron Anschel, Nir Baram, and Nahum Shimkin, "Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 176–185.

[13] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[14] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 5279–5288.

[15] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[16] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu, "Distral: Robust multitask reinforcement learning," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 4496–4506.

[17] Glen Berseth, Cheng Xie, Paul Cernek, and Michiel Van de Panne, "Progressive reinforcement learning with distillation for multi-skilled motion control," *arXiv preprint arXiv:1802.04765*, 2018.

[18] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat, "Discorl: Continual reinforcement learning via policy distillation," *arXiv preprint arXiv:1907.05855*, 2019.

[19] John Moody and Matthew Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[20] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2016.

[21] Konstantinos Saitas Zarkias, Nikolaos Passalis, Avraam Tsantekidis, and Anastasios Tefas, "Deep reinforcement learning for financial trading using price trailing," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 3067–3071.

[22] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[23] Peter J Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*, pp. 492–518. 1992.

[24] Steve Nison, *Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the Far East*, Penguin, 2001.

[25] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international Conference on Computer Vision*, 2015, pp. 1026–1034.

[27] Tijmen Tieleman and Geoffrey Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] Ilya Loshchilov and Frank Hutter, "Fixing weight decay regularization in adam," *arXiv preprint arXiv:1711.05101*, 2017.

[30] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 742–751.