Integrating Contractive Loss in Proximal Policy Optimization for Robust Decision-Making

Michail Dadopoulos[1], Loukia Avramelou[2], Nikolaos Passalis[3], Anastasios Tefas[2]
[1]*School of Electrical and Computer Engineering*
[2]*School of Informatics*
[3]*School of Chemical Engineering*
Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mails: mdadopoul@ece.auth.gr, avramell@csd.auth.gr, passalis@auth.gr, tefas@csd.auth.gr

***Abstract*—Deep Reinforcement Learning (DRL) has found successful applications across various domains, including robotics, healthcare, finance, and autonomous systems. However, in real-world applications, DRL agents often struggle in noisy and volatile environments, leading to instability during training and inconsistent performance. To address these challenges, we propose a novel approach that enhances the robustness of DRL agents by incorporating contractive loss in their training procedure, inspired by Contractive Auto-Encoders. Our method penalizes the sensitivity of learned representations to small input perturbations, thereby rendering the extracted features more resilient to noise and minor fluctuations in the environment. We evaluate the performance of our approach on tasks, namely financial trading and CartPole, characterized by significant variability and uncertainty, demonstrating notable improvements over traditional DRL agents trained without contractive loss. Furthermore, we observe that integrating a contractive loss results in more stable policy behaviors, which is particularly beneficial in settings where erratic actions may be costly or harmful. Overall, our findings suggest that incorporating a contractive loss into DRL training can yield more robust and reliable agents, especially in noisy, real-world conditions.**

***Index Terms*—Deep Reinforcement Learning, Contractive Loss, Financial Trading**

I. Introduction

Deep Reinforcement Learning (DRL) has driven significant breakthroughs across a variety of domains, ranging from robotics and autonomous vehicles to healthcare and financial trading. Early milestones included Deep Q-Networks (DQN), which surpassed human-level performance in Atari games by combining convolutional neural networks with Q-learning [1], [2], paving the way for more advanced algorithms. Subsequent progress led to policy gradient-based approaches, such as Deep Deterministic Policy Gradient (DDPG) [3], Trust Region Policy Optimization (TRPO) [4], and Proximal Policy Optimization (PPO) [5], that enabled the application of reinforcement learning to continuous action environments, enhancing algorithm stability and efficiency.

Though these diverse applications highlight DRL's capacity for adaptability and real-time decision making, the method's potential extends even further. In the realm of financial trading, DRL reframes trading from mere price prediction to active decision making, enabling agents to learn sequential strategies that maximize cumulative profit rather than only predictive accuracy [6], [7], [8]. Algorithms like PPO have shown particular promise, as they can rapidly adapt to shifts in dynamic markets, such as those encountered in cryptocurrency trading and improve decision making [9], [10], [11]. As a result, DRL has gained traction in automated trading systems, offering enhanced flexibility and better alignment with the ultimate objective of profit under volatile, noise-prone conditions.

Despite the impressive progress of DRL in various fields, maintaining stability and robust performance in noisy or volatile environments remains a major hurdle. In contexts like financial trading, where abrupt price movements, volatile market dynamics, and sporadic external events are the norm, agents can struggle to distinguish transient fluctuations from truly meaningful signals. The inherent sensitivity of learned models to these input fluctuations can lead to overfitting or inconsistent behaviors, meaning that even an agent that performs well during backtesting may falter when confronted with real-world conditions. Addressing these concerns is imperative for unlocking the full potential of DRL, whether in finance or other data-intensive settings where noise and volatility pose significant risks. In financial trading, a novel online probabilistic knowledge distillation approach has been shown to improve training stability by transferring knowledge from an ensemble teacher to a single student in real time [12]. Many studies also target robustness through regularization and improved feature extraction. For instance, contrastive representation learning can help DRL agents better disentangle useful signals from noisy inputs [13], while denoising-based approaches aim to enhance networks' ability to generalize in volatile financial markets [14]. Such methods share the common goal of mitigating detrimental perturbations and maintaining stable performance under uncertain conditions.

A complementary perspective comes from Contractive Auto-Encoders (CAEs) [15], which introduce a penalty on the sensitivity of learned representations to input variations. Building on this concept, in this work, we propose an approach that integrates a contractive loss term into the Proximal Policy Optimization (PPO) algorithm equipped with Long Short-Term Memory (LSTM) based actor and critic networks for discrete action tasks. By penalizing excessive sensitivity to small input perturbations, an idea inspired by Contractive Auto-Encoders, this contractive loss encourages more stable latent representations, reducing overfitting and improving generalization in volatile settings. The resulting agents show fewer erratic shifts and more consistent performance under abrupt changes, showcasing resilience to noise and stabilizing training. We evaluate the effectiveness of our method in both financial trading, where volatile markets often undermine conventional DRL agents, and classic control tasks in the Gymnasium environment, showcasing how our loss term can yield robust, consistent policies across a range of noisy, uncertain domains.

The rest of the paper is structured as follows. The proposed method is introduced and analytically derived in Section II, breaking down the motivation behind our methodology. The experimental setup and the experimental evaluation are provided in Section III. Finally, the conclusions of the paper and the potential avenues for future research are discussed in Section IV.

II. Proposed Method

A. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [5] is a policy-gradient algorithm designed to improve stability and sample efficiency in on-policy settings. Let $\pi_\theta(a_t|s_t)$ denote the policy, parameterized by θ, which outputs the probability of selecting action a_t in state s_t at time step t. PPO seeks to maximize the expected advantage while constraining the updated policy to remain close to the old policy, using a clipped surrogate loss:

$$L^{CLIP}(\theta) = -\mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)A_t\right)\right], \tag{1}$$

where

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

Here, $A_t \in \mathbb{R}$ is the estimated advantage, typically computed via a value function or the generalized advantage estimator (GAE), and $\epsilon > 0$, typically set to 0.2 is a hyperparameter that controls the allowable deviation between the old and the new policy.

In addition to the surrogate loss L^{CLIP}, the objective of the PPO algorithm includes a value loss term L^{VF}, to train the critic, and an entropy term $H(\pi_\theta)$, that encourages exploration:

$$L^{\text{total}}(\theta) = L^{CLIP}(\theta) + c_1\, L^{\text{VF}}(\theta^V) - c_2\, H(\pi_\theta), \tag{3}$$

where $L^{\text{VF}}(\theta^V)$ is the loss term for training the critic and $H(\pi_\theta)$ is the policy's entropy in state s_t, which promotes stochasticity and exploration. The coefficients c_1 and c_2 are hyperparameters that weight the critic loss and the entropy term, respectively.

B. Recurrent Actor-Critic Architecture

A recurrent actor-critic architecture trained using PPO is used to address tasks with temporal dependencies or partial observability, commonly encountered in both financial trading and classic control tasks. The input feature vector $\mathbf{x}_t$ is fed into a recurrent neural network at time step t, typically an LSTM, which maintains hidden and cell states $(\mathbf{h}_t, \mathbf{c}_t)$ over time to capture sequential patterns. In the actor network, the LSTM output $\mathbf{z}_t$ proceeds to a fully connected layer (with softmax) that outputs a probability distribution over discrete actions. Simultaneously, the critic network follows a similar process, but its LSTM output is passed through a fully connected layer to produce a scalar value function V_ψ, parameterized by the model's parameters ψ. By maintaining separate parameters for the actor and critic, the method avoids potential interference from shared weights, allowing each network to specialize in its respective task.

C. Contractive Loss on LSTM Representations

The key contribution of our method is the introduction of a contractive term that penalizes the sensitivity of the LSTM-based representation to small input perturbations. Inspired by Contractive Auto-Encoders (CAEs) [15], which penalize the Jacobian of hidden-layer representations with respect to their inputs, we apply a similar penalty directly to the square of the LSTM output of the actor. Let $\mathbf{x}_t$ represent the input features at time t, and let $h(\mathbf{x}_t)$ denote the elementwise square of the LSTM output of the actor. We define the contractive loss as:

$$L_{\text{contractive}} = \|\nabla_{\mathbf{x}_t} h(\mathbf{x}_t)\|_F = \sqrt{\sum_{i,j}\left(\frac{\partial h_i(\mathbf{x}_t)}{\partial x_{tj}}\right)^2}, \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm, $h_i(\mathbf{x}_t)$ is the i-th component of $h(\mathbf{x}_t)$, and x_{tj} is the j-th component of the input feature vector $\mathbf{x}_t$. A hyperparameter λ scales this term, controlling the degree to which the model is encouraged to remain invariant to small perturbations in $\mathbf{x}_t$.

D. Overall Objective

We integrate the contractive loss into the PPO objective as an additional regularization factor:

$$J_{\text{total}} = L^{CLIP}(\theta) + c_1\, L^{\text{VF}}(\theta^V) - c_2\, H(\pi_\theta) + \lambda\, L_{\text{contractive}}. \tag{5}$$

where the agent ultimately aims to minimize J_{total}, effectively balancing policy improvement, value estimation, exploration, and robustness to input perturbations. By adjusting c_1, c_2, and λ, one can emphasize different facets of performance and representation stability.

III. Experimental Evaluation

In this section, we evaluate the performance of our proposed contractive loss integrated into the recurrent PPO algorithm across (i) a challenging financial trading environment, characterized by highly volatile market conditions and discrete trading decisions [16], and (ii) the classic CartPole control task [17], augmented with Gaussian noise in state observations to simulate instability. These domains highlight the method's capacity to stabilize training and improve policy robustness under noisy or rapidly changing conditions.

A. Financial Trading

1) Dataset and Preprocessing: The dataset used in the conducted experiments consists of the OHLCV (Open, High, Low, Close, and Volume) information for 14 cryptocurrency-to-stablecoin pairings at a minute-to-minute frequency. The specific pairs include XRP/USDT, XMR/USDT, ATOM/USDT, VET/USDT, BTC/USDT, BTCB/USD, ETH/USDT, NEO/USDT, EOS/USDT, ETC/USDT, ADA/USDT, WAVES/USDT, XLM/USDT, and TRX/USDT. For the conducted experiments, the data were sampled on an hourly basis starting from 5 A.M. on August 17th, 2017, and ending at 6 A.M. on February 12th, 2022. Although all the pairs in the dataset end at the same time, not all of them start at the same time. We split the dataset into train/test sets, where the testing period begins on March 15th, 2021. The testing period was selected to lead to a zero profit for a buy-and-hold trading strategy, avoiding evaluation in a constantly trending period and creating a more challenging evaluation setup.

The feature extraction process involves calculating the percentage changes between the high, low, and closing prices of consecutive time intervals, providing a concise representation of price movements. These features are further normalized using z-score normalization, trimmed to limit extreme values, and included alongside a one-hot encoded representation of the market agent's position (no position, buy, or sell) as part of the agent's observation at each time interval.

2) Model Architecture: We adopt a recurrent Proximal Policy Optimization (PPO) architecture with separate Actor and Critic networks. Each network is composed of an LSTM layer with 32 hidden units for feature processing and a linear position encoder with 10 neurons for positional inputs. The outputs of the LSTM and position encoder are concatenated and passed to a base MLP with 32 neurons, SiLU activation [18], and 0.2 dropout, followed by the final linear heads with dropout 0.2. The Actor 3-neuron head produces logits for the three possible trading actions (buy, hold, sell), while the 1-neuron Critic head generates a single value estimate.

We set the PPO clipping threshold ϵ in Eq. 1 to 0.2, while c_1 and c_2 in Eq. 5 were set to 1 and 0.02, respectively. The hyperparameter λ for the contractive loss was fixed at 0.3 during all experiments. The learning rate is 5×10^{-5}, the batch size is 32, the number of parallel environments is 128, and training is conducted for 2000 epochs. We train the agent with the RAdam optimizer [19].

3) Evaluation Metrics: We evaluated the agent's performance using the Profit and Loss (PnL) metric, a widely adopted measure in trading systems. A commission penalty of 2×10^{-5} is applied to reflect real-world trading scenarios and discourage overly frequent trades.

$$PnL = \sum_{t=1}^{N} \delta_t p_t - |\delta_t - \delta_{t-1}|c, \tag{6}$$

where N denotes the total duration of the back-testing period (number of time-steps), p_t is the return at time step t, c is the commission paid for realizing profits/losses and δ_t is an index variable used to indicate the current position, which is defined as:

$$\delta_t = \begin{cases} -1, \text{if agent holds a short position at time-step } t \\ 1, \text{if agent holds a long position at time-step } t \\ 0, \text{if the agent is not in the market at time-step } t \end{cases}. \tag{7}$$

As a baseline, we compare the performance of our proposed method against the standard PPO algorithm without contractive loss, highlighting the improvements in generalization and robustness. Additionally, the impact of noise and volatility on the agent's decisions and outcomes is analyzed to assess its resilience in challenging environments. In each experiment, the performance of each method was evaluated during the train and test period, and the Profit and Loss (PnL) for both was reported as the mean PnL across five runs executed with distinct random seeds to mitigate variance caused by initialization or stochasticity in the training procedure.

4) Results: The mean and standard deviation of cumulative PnL for both the training and testing periods is reported in Table I, comparing our proposed method against the baseline. While the baseline achieves a higher training PnL (84.24 ± 2.46) compared to the proposed method (29.05 ± 13.12), it struggles in testing conditions, achieving a significantly lower test PnL (14.40 ± 1.49). These findings indicate overfitting, as the baseline fails to generalize to unseen, noisy data. In contrast, the proposed method demonstrates a much higher test PnL (36.07 ± 14.11), highlighting its ability to generalize more effectively to unseen data and better navigate the noisy, volatile market conditions.

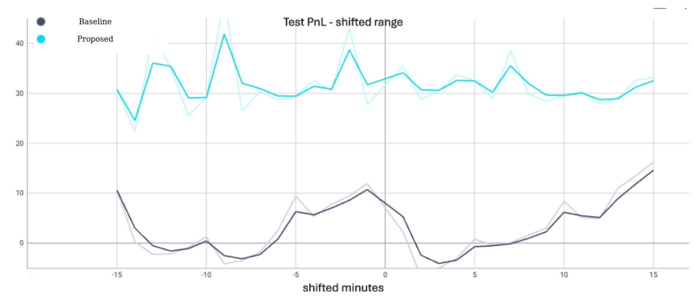
TABLE I: Mean and standard deviation of train and test PnL after training within the financial trading environment

Method	**Test PnL**	**Train PnL**
Baseline	14.40 ± 1.49	84.24 ± 2.46
Proposed Method	$\mathbf{36.07 \pm 14.11}$	29.05 ± 13.12

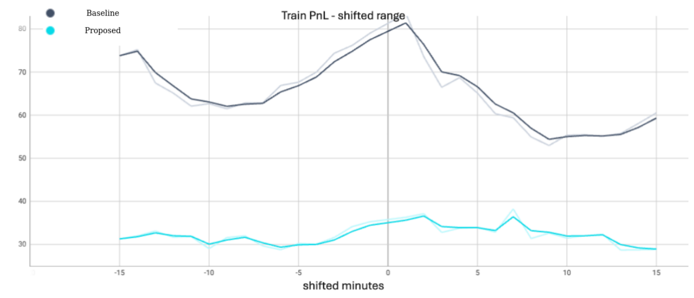
To further assess robustness, we introduce artificially augmented noise via temporal shifts in the input data before feature extraction, ranging from $[-15, \ +15]$ minutes. For each shift, we compute the mean cumulative PnL for the training and testing periods after the agents' training, as illustrated in Fig. 1a and 1b. Our proposed method exhibits notable stability in performance regardless of temporal shifts, maintaining a relatively consistent level of cumulative gains, as shown in Fig. 1b. In contrast, the baseline displays significant fluctuations when the inputs are shifted, suggesting greater sensitivity to noise during training. Similarly, the proposed method maintains superior stability and performance, with the PnL curve showing relatively higher resilience to temporal shifts compared to the baseline, as shown in Fig. 1a.

B. CartPole Environment

1) Environment: The CartPole environment was chosen for its simplicity, widespread use as a benchmark for reinforcement learning (RL) algorithms, and its ability to train fast.



(a) Mean PnL for testing period for different input data shifts.



(b) Mean PnL for training period for different input data shifts.

Fig. 1: Mean cumulative PnL for different temporal shifts in input data during both testing (a) and training (b) period. The x-axis represents the applied temporal shift in input data, ranging from $[-15, +15]$ minutes and the y-axis denotes the mean cumulative PnL.

The environment represents a classic control problem where a pole is attached to a cart moving along a frictionless track. The agent's objective is to balance the pole by applying left or right forces to the cart. The environment provides a 4-dimensional state space consisting of cart position (bounded between -4.8 and 4.8), cart velocity, pole angle (bounded between -24 and +24 degrees), and pole angular velocity. The action space is discrete with two possible actions: pushing the cart left (0) or right (1). An episode terminates if the pole angle exceeds ± 12 degrees, the cart position exceeds ± 2.4 units, or the episode length exceeds 500 steps. A reward of +1 is given for each timestep the pole remains balanced.

2) Model Architecture: We adopt a recurrent Proximal Policy Optimization (PPO) architecture with separate Actor and Critic networks. Each network begins with a feature extractor that flattens the 4-dimensional state input. The flattened features are processed through parallel LSTM layers, each with 64 hidden units, enabling separate temporal processing for the Actor and Critic. The LSTM outputs are fed into MLPs, each containing a hidden layer of 64 neurons with ReLU activation. The Actor network ends in a 2-neuron head producing logits for the two possible actions (left/right), while the Critic head outputs a single neuron for state-value estimation.

We configure the PPO algorithm with the optimized hyperparameters from RL-Baselines3-Zoo [20] with a clipping threshold ϵ of 0.2, an entropy coefficient c_2 of 0.0, a value function coefficient c_1 of 0.5 and a hyperparameter for the contractive loss λ to 0.3, according to the Eq 5. Training uses the Adam optimizer with a learning rate of 1×10^{-3} and the training process continues until reaching 100,000 total timesteps. Each update consists of a rollout collection phase of 32 steps across 8 parallel environments, followed by 20 epochs of policy optimization on the collected data with a batch size of 256.

3) Evaluation Metrics: Performance was measured using episodic rewards, which represent the cumulative rewards obtained by the agent during an episode. As a baseline, we compare the performance of our proposed method against the standard PPO algorithm without contractive loss, highlighting the improvements in generalization and robustness. To assess the robustness of the proposed method, the evaluation is conducted under both normal conditions and with added Gaussian noise to the state observations.

During training, the agent's performance was evaluated every 10,000 timesteps by running 10 episodes with the current policy. The reported scores represent the mean and standard deviation of these evaluation episodes, providing a consistent measure of the agent's performance throughout training. To ensure statistical reliability, each experiment is repeated five times with distinct random seeds. The reported results include the mean and standard deviation of episodic rewards across these runs, highlighting both the average performance and the stability of the learning process.

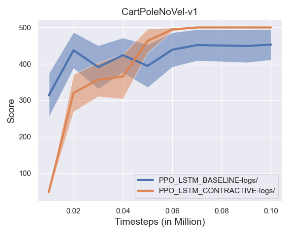
4) Results: To assess the robustness of the proposed method, we inject dimension-specific additive Gaussian noise into the observations at each step of the CartPole environment. Small perturbations ($\sigma = 0.01$) are used for the sensitive pole angle, while more moderate perturbations are injected for the unbounded velocities ($\sigma = 0.5$) and cart position ($\sigma = 0.1$, approximately 2% of the range). The noisy observations are clipped to respect the physical bounds of the environment for cart position $[-4.8, 4.8]$ and pole angle $[-0.418, 0.418]$ radians.

The performance of the proposed method and the baseline are summarized in Table II and Fig. 2a and 2b, which illustrate the episodic reward (score) over training timesteps under normal and noisy conditions, respectively. Under normal conditions, the proposed method (PPO-LSTM Contractive) achieves perfect performance, maintaining an episodic reward of 500 ± 0, while the baseline scores 453 ± 42. Although the baseline also converges to high rewards, it exhibits greater fluctuations during training, suggesting less consistent policy learning. This demonstrates that the contractive loss improves stability and convergence, resulting in consistent, optimal performance.

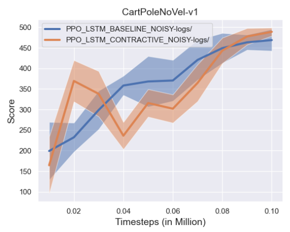
Interestingly, the baseline performs slightly better under noisy conditions (469 ± 26) than under normal conditions (453 ± 42). This counterintuitive result may indicate that the noise introduces a form of implicit regularization, preventing the baseline agent from overfitting to specific features of the normal environment. In essence, the noise may force the baseline to generalize better, leading to improved performance. In comparison, the proposed method (PPO-LSTM Contractive

TABLE II: Mean and standard deviation of episode rewards for different PPO variants

Method	**Noise**	**Episode Reward**
Baseline	No	453 ± 42
Proposed Method	No	$\mathbf{500 \pm 0}$
Baseline	Yes	469 ± 26
Proposed Method	Yes	$\mathbf{489 \pm 10}$



(a) Episodic reward over training timesteps under normal conditions.



(b) Episodic reward over training timesteps under noisy conditions.

Noisy) not only achieves a higher mean reward (489 ± 10) under noisy conditions but also exhibits significantly reduced variability. This demonstrates that the contractive loss enables the agent to benefit from the noise-induced regularization while maintaining stability and robust performance.

IV. Conclusion

In this paper, we introduced a contractive loss term into the recurrent PPO algorithm to improve the stability and robustness of DRL agents in noisy and volatile environments. Experimental results across financial trading and the CartPole environment demonstrated the efficacy of the proposed method, with significant improvements in generalization, reduced variability, and more consistent performance compared to the baseline. The contractive loss enabled agents to achieve higher test rewards and better handle noise, particularly in challenging scenarios. These findings suggest that the proposed approach provides a principled solution for enhancing the reliability of DRL agents, making it a promising avenue for future research and real-world applications. A promising direction for future research is the expansion of the proposed method to continuous action tasks, enabling evaluation in more complex control environments. Furthermore, the integration of the contractive loss into alternative DRL algorithms like Soft Actor-Critic (SAC) and TD3 could reveal its effectiveness across different training dynamics and policy architectures. Finally, applying the method to real-world problems characterized by higher levels of noise and uncertainty, such as robotic control or energy system optimization could further demonstrate its robustness and broaden its range of applications.

Acknowledgment

This work was supported by the research project ”Energy Efficient and Trustworthy Deep Learning - DeepLET” which is implemented in the framework of H.F.R.I call “Basic research Financing (Horizontal support of all Sciences)” under the National Recovery and Resilience Plan “Greece 2.0” funded by the European Union –NextGenerationEU (H.F.R.I. Project Number: 016762).

References

[1] V. Mnih, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[2] 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*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] T. Lillicrap, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[4] J. Schulman, “Trust region policy optimization,” *arXiv preprint arXiv:1502.05477*, 2015.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[6] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid, “Practical deep reinforcement learning approach for stock trading,” *arXiv preprint arXiv:1811.07522*, pp. 1–7, 2018.

[7] K. S. Zarkias, N. Passalis, A. Tsantekidis, and A. Tefas, “Deep reinforcement learning for financial trading using price trailing,” in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2019, pp. 3067–3071.

[8] Z. Zhang, S. Zohren, and S. Roberts, “Deep reinforcement learning for trading,” *arXiv preprint arXiv:1911.10107*, 2019.

[9] A. Tsantekidis, N. Passalis, and A. Tefas, “Diversity-driven knowledge distillation for financial trading using deep reinforcement learning,” *Neural Networks*, vol. 140, pp. 193–202, 2021.

[10] L. Avramelou, P. Nousi, N. Passalis, and A. Tefas, “Deep reinforcement learning for financial trading using multi-modal features,” *Expert Systems with Applications*, vol. 238, p. 121849, 2024.

[11] H. Zhang, Z. Jiang, and J. Su, “A deep deterministic policy gradient-based strategy for stocks portfolio management,” in *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*. IEEE, 2021, pp. 230–238.

[12] V. Moustakidis, N. Passalis, and A. Tefas, “Online probabilistic knowledge distillation on cryptocurrency trading using deep reinforcement learning,” *Pattern Recognition Letters*, vol. 186, pp. 243–249, 2024.

[13] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov, “Contrastive learning as goal-conditioned reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 603–35 620, 2022.

[14] Y. Li, W. Zheng, and Z. Zheng, “Deep robust reinforcement learning for practical algorithmic trading,” *IEEE Access*, vol. 7, pp. 108 014–108 022, 2019.

[15] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Proceedings of the 28th international conference on international conference on machine learning*, 2011, pp. 833–840.

[16] L. Avramelou, P. Nousi, N. Passalis, S. Doropoulos, and A. Tefas, “Cryptosentiment: A dataset and baseline for sentiment-aware deep reinforcement learning for financial trading,” in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. IEEE, 2023, pp. 1–5.

[17] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[18] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural networks*, vol. 107, pp. 3–11, 2018.

[19] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, “On the variance of the adaptive learning rate and beyond,” *arXiv preprint arXiv:1908.03265*, 2019.

[20] A. Raffin, “Rl baselines3 zoo,” https://github.com/DLR-RM/rl-baselines3-zoo, 2020.