Deployability of Deep Reinforcement Learning in Portfolio Management

Ishwar Govind*
ishwar.govind@gmail.com
Indian Institute of Technology Palakkad

Jerry Thomas*
jerryjohnthomastvm@gmail.com
Indian Institute of Technology Palakkad

Chandrashekar Lakshminarayanan
chandrashekar@cse.iitm.ac.in
Indian Institute of Technology Madras

Abstract

Portfolio management is a classical problem that has been studied in the financial literature in a wide variety of settings. In recent years, deep reinforcement learning (DRL) agents have been proposed as a potential solution for portfolio management. In this paper, we look at two fundamental challenges. (i) There is no common baseline to compare the various DRL agents, and (ii) there is no straightforward way to check if the DRL agents are indeed learning the near-optimal policy. To address these two challenges, we study the performance of DRL agents of varying complexity on the synthetic market based on a simplified version of the Baba-Engle-Kraft-Kroner (BEKK) model. We empirically demonstrate the following: (i) not all DRL agents perform well in the synthetic market; (ii) DRL agents that perform well in the synthetic market learn policies that align with the optimal policy obtained from a maximum Sharpe Ratio portfolio.

CCS Concepts

• **Computing methodologies** → **Reinforcement learning**; • **Applied computing** → **Economics**.

ACM Reference Format:
Ishwar Govind, Jerry Thomas, and Chandrashekar Lakshminarayanan. 2024. Deployability of Deep Reinforcement Learning in Portfolio Management. In *8th International Conference on Data Science and Management of Data (12th ACM IKDD CODS and 30th COMAD) (CODS-COMAD Dec '24), December 18–21, 2024, Jodhpur, India.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3703323.3703333

1 Introduction

Portfolio management is the sequential reallocation of wealth across various assets in the market. The value of the assets changes with time, and the goal of portfolio management is to maximize the total asset value with an acceptable amount of risk. Financial markets are often regarded as complex and hard to predict, posing considerable challenges even for experienced traders. In the past, quantitative traders came up with strategies which were tailored to perform well only in specific market conditions. Most of these strategies have assumptions that prevent them from being used in different markets without ad-hoc modifications[22].

Mathematical solutions to portfolio management date back to the Markowitz model[12], which emphasizes portfolio diversification to maximize return for a given level of risk and provides a foundation for these evaluations. Investors can select a portfolio on the efficient frontier based on their risk tolerance and return expectations. The model also introduces three variants: the minimum variance portfolio, the tangency portfolio, and the equal-weighted portfolio. The tangency portfolio is particularly notable, as it is the portfolio that has the highest Sharpe ratio, which represents the best risk-adjusted return.

Some of the earlier work using supervised learning and reinforcement learning to solve portfolio optimisation can be found respectively in [8, 23] and[14].

Recently, deep learning (DL) has become the leading approach across a variety of application domains[20]. In particular, the success of deep reinforcement learning (DRL) in various complex tasks, such as strategy games, have led to its adoption in portfolio management, yielding promising results [3][10][9].

A desirable aspect of the Markowitz theory is that the optimisation problem and the resulting strategies are appealing to human intuition which resulted in its adoption and deployability. In contrast, DRL agents are directly trained by maximizing a portfolio objective and achieving favorable returns on real market data using a stack of black-box DL models[22]. Thus, DRL agents do not fully address the interpretability of the resulting strategies, thereby limiting their deployability in practice.

Our paper focuses on evaluating the deployability of DRL agents for portfolio management. We seek to understand whether the DRL policies learned can align with well-established financial principles, such as buying low and selling high, and aim to infer other novel market patterns. Financial intuition and explainability are crucial for the adoption and deployment of these models, ensuring that strategies are effective and understandable by practitioners. Thus, in this paper deployability refers to the model's interpretability. Towards this end, we compare the performance of DRL-based portfolios with that of the traditional Tangency portfolio, which is known for maximizing the Sharpe ratio. This comparison helps to provide insights into the efficacy of DRL in real-world scenarios, particularly in terms of generating actions and allocating weights within portfolios. We study the performance of DRL agents on the synthetic market based on the Baba-Engle-Kraft-Kroner (BEKK) model. To achieve our goal, we investigate the following: (i) Do DRL agents perform well in the BEKK market? (ii) For the DRL agents that perform well in the BEKK market, do we learn policies that align with the optimal policy obtained from Tangency portfolio?

*Equal Contribution

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CODS-COMAD Dec '24, December 18–21, 2024, Jodhpur, India
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1124-4/24/12
https://doi.org/10.1145/3703323.3703333

2 Background

Portfolio management is a sequential decision making problem which involves reallocation of wealth across various assets. In this paper, we consider portfolio management of n assets whose price at time t are denoted by $p_t = (p_t^{(1)}, \ldots, p_t^{(n)})$. The weight or current holding of a particular asset is denoted by $w_t = (w_t^{(1)}, \ldots, w_t^{(n)})$. The asset price data is used to extract features such as MACD (Moving Average Convergence/Divergence) , RSI (Relative Strength Index) and ADX (Average Directional Index). The k-dimensional asset features at time t are denoted by:

$$f_t = (f_{(1,t)}^{(1)}, \ldots f_{(k,t)}^{(1)}), \ldots, (f_{(1,t)}^{(n)} \ldots f_{(k,t)}^{(n)})$$

for k features. The portfolio management problem can be formulated as a Markov Decision Process (MDP), whose state, action, reward and probability transition is described below:

2.0.1 State: At time t with time window l, the state is given by: $s_t = ((p_{t-l}, f_{t-l}, w_{t-l}), \ldots, (p_t, f_t, w_t))$.

2.0.2 Action: The action at time t is the agent's attempt at reallocating the weights, given by $a_t^{(1)}, \ldots, a_t^{(n)}$. Negative action denotes a short position, whereas positive action indicates a long position. The range of the actions could be [0,1] or [-1,1] depending on the shorting constraints. The action is then converted into market relevant action for trading as given in section 2.1.

2.0.3 Transition (BEKK Market Model): The state transition depends on portfolio action and market dynamics. Market dynamics is nothing but the evolution of asset prices p_t typically expressed in terms of log returns r_t, given by

$$\log \frac{p_{t+1}^{(i)}}{p_t^{(i)}} = r_t^{(i)} \tag{1}$$

In this paper, we use a simplified variant of the BEKK model [4] to describe the market dynamics. Most works in mathematical finance assume that prices follow a log-normal geometric random walk or geometric Brownian motion [5][17]. However, such assumptions may not accurately capture the correlations across assets, as they often involve sampling log returns independently for each asset. In our basic BEKK model, we generate the mean μ_{assets} and covariance Σ_{assets} of the log-returns of the assets. The log-return r is then sampled using multivariate normal distribution $r_t \sim \mathcal{N}(\mu_{\text{assets}}, \Sigma_{\text{assets}})$, where $r_t = (r_t^1, \ldots, r_t^n)$. The log-returns are then used to generate the asset price data p_t as given in equation 1 .

2.0.4 Reward: The reward is determined by the Differential Sharpe Ratio (DSR) function[14]. DSR is obtained by considering the exponential moving averages and standard deviation of the returns.

The reinforcement learning agent's goal is to learn a policy that maximizes the cumulative expected reward.

2.1 Converting Agent Actions to Market Actions

The agent's actions a_t from section 2.0.2 are converted to market relevant actions, such as taking *long* or *short positions*. Initially, the trader starts with a capital C_0 that can be used to trade assets. Given W_t, W_t^+ and W_t^- are the pool of long and short positions respectively, where

$$W_t^+ = \{w_{i,t}^+ \mid w_{i,t}^+ = a_t^{(i)} \text{ for all } i \in \{i \mid a_t^{(i)} > 0\}\},\ |W^+| = m1$$

$$W_t^- = \{w_{i,t}^- \mid w_{i,t}^- = -a_i \text{ for all } i \in \{i \mid a_t^{(i)} < 0\}\},\ |W^-| = m2$$

The sum of the weights is used to determine the ratio of the capital to be used for short and long positions in the market. Dropping the subscript t, we have: $w^+ = \sum_{i=1}^{m1} w_i^+$, $w^- = \sum_{i=1}^{m2} w_i^-$. The weight allocated to cash, $w_0 = \max(1 - w^+ - w^-, 0)$, is calculated to ensure that the total allocation does not exceed the available capital. The asset weights $(w_1, \ldots, w_n)$ are calculated by applying *softmax* function[1] on the long and short pool of weights separately, then using their weight sum ratio to determine the capital to invest in each asset.

$$C_t^+ = C_t^T \left(\frac{w^+}{w^+ + w^-}\right) \quad , \quad C_t^- = C_t^T \left(\frac{w^-}{w^+ + w^-}\right)$$

where C_t^T is the total tradable capital at time t. C_t^+ and C_t^- are the total capital to be allocated for long and short positions respectively. The simple return $R_t = \frac{p_{t+1} - p_t}{p_t}$, representing the percentage gain or loss of an asset relative to its last price, is used to compute the subsequent total capital. We assume the return of cash to be zero.

$$C_{t+1}^T = C_t^T (\mathbf{1} \cdot w_0) + C_t^+ ((\mathbf{1} + R_t^+) \cdot w_t^+) + C_t^- ((\mathbf{1} + R_t^-) \cdot (-w_t^-))$$

2.2 Max-Sharpe Baseline

Markowitz's theory of Mean-Variance Optimization (MVO) gives us a mathematical framework for choosing the most optimal portfolio that maximizes the expected return for a chosen amount of risk. It introduces the concept of an efficient frontier which is a Pareto optimal curve of efficient portfolios obtained on solving a convex optimization problem[12].

While the MVO gives more flexibility to the trader in choosing the strength of risk, the maximum Sharpe portfolio, also called the tangency portfolio (tangent to the efficient frontier) gives us the most optimal return and risk balanced option. The Sharpe ratio objective is described as the reward-to-volatility ratio[21].

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \frac{\mu^T w - r_f}{(w^T \Sigma w)^{1/2}} \\ \text{subject to} \quad & \mathbf{1}^T w = 1 \\ & w_i \geq 0, \text{ if non-shorting} \end{aligned}$$

where r_f is the risk-free rate of the market, described as the theoretical return of a risk-less asset, w is the vector of portfolio weights, and μ is the expected returns vector. We create a policy by solving the convex optimization problem derived from the Sharpe ratio objective given above by making variable substitutions[2][13].

3 Our Work

3.1 Reinforcement Learning Agent

To evaluate the effectiveness of RL in portfolio optimization we implement proximal policy optimization (PPO)[19] based agents with different neural network architectures. To deal with continuous action space weights and ensure stability we use proximal policy optimization (PPO) algorithm for training. The architecture is extended with a feature extractor model to give the actor and critic high-level knowledge from the asset price data as seen in Figure

1. We evaluate multiple feature-extractor models for analysing the capability of deep learning architectures[1].

3.2 Feature Extractor Models

NN and NN-R: The base agent with a fully connected neural network serving as the feature extractor [18]. In *NN-R* we use simple return as a reward instead of DSR.

DA-RNN : We employ dual-stage attention-based recurrent neural network (DA-RNN), due to its capability to extract long term dependencies and selectively focus on relevant features.[16]

PatchTST: An efficient transformer-based model for time-series forecasting [15]. It demonstrated superior performance in time series forecasting tasks.

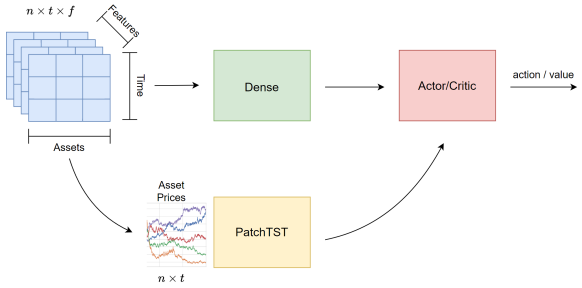


Figure 1: PatchTST + PPO agent architecture

4 Experiment

4.1 Experimental Setup

4.1.1 Dataset: Initially, we generate the synthetic data for five assets of length 5000 days (an equivalent to a standard market dataset of size 20 trading years)[11] using our simplified BEKK market model described in Section 2.0.3 and visualised in Figure 2. Subsequently, the data is split into a 9:1 ratio for training and testing, respectively.

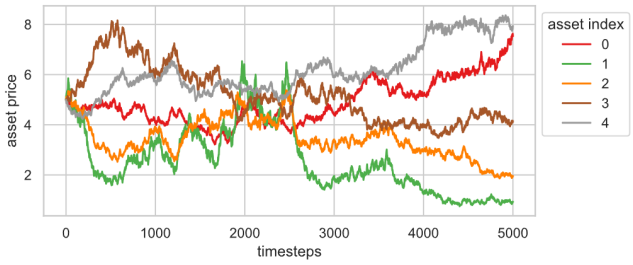


Figure 2: Synthetic data for 5 assets, with asset prices plotted on the Y-axis and timesteps (in days) on the X-axis.

4.1.2 Baseline: We compare the performance of max sharpe optimization with shorting and non-shorting constraints. We solve the optimization problem on the past t days to predict the weights of $(t+1)^{th}$ day.

[1]Our RL agents are developed utilizing the PPO implementation provided by CleanRL [7]

4.1.3 Evaluation Metrics: We compare the max-sharpe baseline with the agents having different feature extractor models. Four well-established metrics [9] are used to evaluate the different policies used in our experiments. The following four metrics are:

(1) *Cumulative Return (CR)* is the total return rate of the strategy for the whole duration of the trading period. CR is used to show the overall profitability of a strategy.
(2) *Sharpe Ratio (SR)* is the risk-adjusted return for the entire trading period. It reflects how efficiently the strategy balances risk and return.
(3) *Maximum DrawDown (MDD)* is the maximum observed loss from a peak to a trough of a portfolio, before a new peak is attained [6]. MDD measures the strategy's risk of significant loss.
(4) *Final Balance (Bal)* is the final capital of the agent on the test dataset on starting with an initial balance of $100. To assess the variance of the RL agent's results on the test dataset, the minimum and maximum balances at the end of the trading session are evaluated as well.

The findings of the RL agents are summarized after multiple runs on the test dataset.

4.2 Performance on Synthetic Data

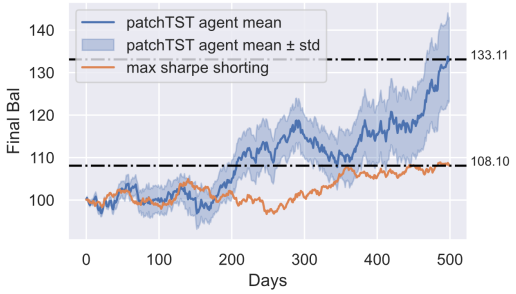


Figure 3: Test Performance comparison of the PatchTST agent and Baseline, with Balance (Y-axis) and Time, in days (X-axis).

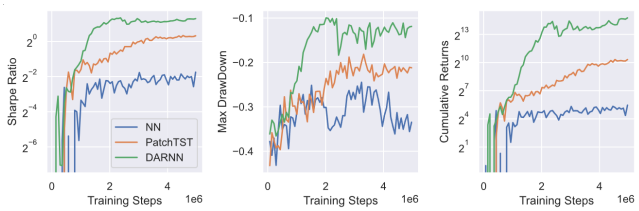
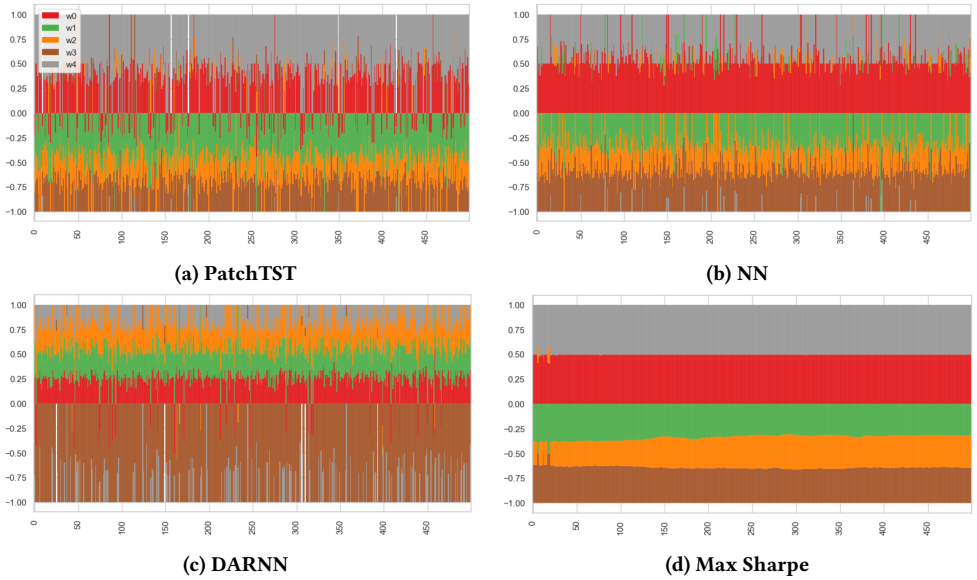


Figure 4: Training performance of various models on different metrics

We report the results of different policies in Table 1. The portfolio weights over time allocated on test data by different agents are shown in Figure 5. The results are summaries as follows:

PatchTST Model's Performance: The PatchTST-based agent achieved the best results in both shorting and non-shorting trading



(a) PatchTST (b) NN

(c) DARNN (d) Max Sharpe

Figure 5: The average weight distribution of policies on the shorting task is illustrated. The plots display the weight distribution in relation to the timestep for the assets, which are represented by the same color as in Figure 2.

Models	Non Shorting						Shorting					
	Bal (Min)	Bal (Avg)	Bal (Max)	CR%	SR	MDD	Bal (Min)	Bal (Avg)	Bal (Max)	CR%	SR	MDD
Max Sharpe	**107.37**	107.37	107.37	7.37	0.531	**-0.085**	108.1	108.1	108.1	8.095	0.67	-0.078
NN-*R*	99.07	102.95	110.05	2.95	0.243	-0.089	99.41	105.95	113.12	5.95	0.302	-0.118
NN	100.23	104.62	111.56	4.62	0.312	-0.101	109.68	111.82	117.66	11.82	0.856	**-0.075**
DARNN	91.81	96.12	101.33	-3.88	-0.065	-0.142	92.86	100.29	105.37	0.29	0.071	-0.121
PatchTST	102.67	**108.85**	**114.20**	**8.85**	**0.608**	-0.086	**116.35**	**133.11**	**144.24**	**33.11**	**1.275**	-0.098

Table 1: Performance metrics for various models with and without shorting. Bold text indicates the best result, while underlined text denotes the second-best result.

tasks, demonstrating effective risk-return balancing, particularly in shorting scenarios as seen in Table 1 and Figure 3 .

PatchTST Outperforms DARNN in Testing: DARNN surpasses PatchTST during training (Figure 4), likely due to overfitting, as evidenced by the weak correlation between its training and test performance. In testing, DARNN underperformed significantly compared to other models, attributed to its suboptimal and unique weight allocation strategy.

Impact of DSR Reward: The NN-based agent outperformed DARNN in terms of both return and risk metrics. Using the DSR reward improved performance over the simple return reward.

Weight Allocation Dynamics: The NN and PatchTST models had smoother weight changes compared to DARNN, contributing to better performance. The PatchTST model displayed a higher variance in weight allocation, suggesting more dynamic adjustments.

Superiority in Shorting Tasks: RL strategies, including the PatchTST-based agent, outperformed their non-shorting counterparts when short positions were allowed. This suggests the capability of RL agents to leverage various strategies to maximize rewards across diverse market conditions.

5 Conclusion

In this work, we examine the deployability of DRL agents in portfolio management with a data agnostic approach. We analyze the allocation of weights done by deep learning architectures of varying complexity on a randomly generated dataset by using PPO algorithm and comparing it with a deterministic baseline. We demonstrate superior performance in portfolio management by introducing an actor-critic model with a PatchTST backbone trained on DSR reward. Further investigation reveals a similar pattern of weight allocation by different models that can be explained by the baseline. While the RL agents were able to display superior performance in shorting tasks, the high variance in the results caused by the stochasticity of the PPO algorithm could lead to unsafe investments in the market. Additionally, the synthetic dataset does not take into account volatility clustering, causality, or other real-market circumstances. Even on synthetic data, issues with variance and interpretability suggest greater challenges in real markets, highlighting the need for further investigation before deployment. Thus, the deployability of DRL agents in finance demands careful evaluation. Future work could involve enhancing spatial correlation models with techniques such as graph neural networks and evaluating DRL agents on more complex, generalizable market models.

References

[1] John Bridle. 1989. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems* 2 (1989).

[2] Gerard Cornuejols and Reha Tutuncu. 2005. Optimization methods in finance. *Camegie Mellon University, Pittsburgh* (2005).

[3] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28, 3 (2016), 653–664.

[4] Robert F Engle and Kenneth F Kroner. 1995. Multivariate simultaneous generalized ARCH. *Econometric theory* 11, 1 (1995), 122–150.

[5] Yiyong Feng, Daniel P Palomar, et al. 2016. A signal processing perspective on financial engineering. *Foundations and Trends® in Signal Processing* 9, 1–2 (2016), 1–231.

[6] Adam Hayes. 2024. Maximum Drawdown (MDD) Defined, With Formula for Calculation — investopedia.com. https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp.

[7] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. 2022. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research* 23, 274 (2022), 1–18. http://jmlr.org/papers/v23/21-1342.html

[8] Kei-keung Hung, Yiu-ming Cheung, and Lei Xu. 2003. An extended ASLD trading system to enhance portfolio management. *IEEE Transactions on Neural Networks* 14, 2 (2003), 413–425.

[9] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. 2017. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059* (2017).

[10] Olivier Jin and Hamza El-Saawy. 2016. Portfolio management using reinforcement learning. *Stanford University* (2016).

[11] Hiransha M, Gopalakrishnan E.A., Vijay Krishna Menon, and Soman K.P. 2018. NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Computer Science* 132 (2018), 1351–1362. https://doi.org/10.1016/j.procs.2018.05.050 International Conference on Computational Intelligence and Data Science.

[12] Harry M Markowitz and Harry M Markowitz. 1967. *Portfolio selection: efficient diversification of investments*. J. Wiley.

[13] Robert Andrew Martin. 2021. PyPortfolioOpt: portfolio optimization in Python. *Journal of Open Source Software* 6, 61 (2021), 3066. https://doi.org/10.21105/joss.03066

[14] John Moody and Matthew Saffell. 1998. Reinforcement learning for trading. *Advances in Neural Information Processing Systems* 11 (1998).

[15] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[16] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971* (2017).

[17] David Ruppert and David S Matteson. 2011. *Statistics and data analysis for financial engineering*. Vol. 13. Springer.

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

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

[20] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing* 90 (2020), 106181.

[21] William F Sharpe. 1994. The sharpe ratio. *Journal of portfolio management* 21, 1 (1994), 49–58.

[22] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. 2021. DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 1 (May 2021), 643–650. https://doi.org/10.1609/aaai.v35i1.16144

[23] Lei Xu, Chi Chiu Cheung, H Hua Yang, and Shun-Ichi Amari. 1997. Independent component analysis by the information-theoretic approach with mixture of densities. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, Vol. 3. IEEE, 1821–1826.