Thesis for the Master of *Science*

Automated stock trading using transformer and reinforcement learning

Nam Yeong Lee

Graduate School of Hanyang University

August 2023

Thesis for the Master of *Science*

Automated stock trading using transformer and reinforcement learning

Thesis Supervisor: *Jun Moon*

A Thesis submitted to the graduate school of Hanyang University in partial fulfillment of the requirements for the degree of *Master of Science*

Nam Yeong Lee

August 2023

Department of Artificial Intelligence
Graduate School of Hanyang University

This thesis, written by *Nam Yeong Lee*,
has been approved as a thesis for the degree of
Master of Science.

August 2023

Committee Chairman: Jun-Won Choi ( )

Committee Member: Jun Moon ( )

Committee Member: Yung-Kyun Noh ( )

Graduate School of Hanyang University

Contents

List of Figures

List of Tables

Abstract

Automated stock trading using transformer and reinforcement learning

Nam Yeong Lee

Department of Artificial Intelligence,

Graduate School of HANYANG UNIVERSITY

Directed by Professor Jun Moon

Recently, with the growing interest in trading in financial stock markets, several algorithms have been proposed to automatically allocate stocks and/or predict future stock values using machine learning methods, such as reinforcement learning (RL), LSTM, and transformers. Among them, RL has been used to allocate portfolio assets with a series of optimal actions. The most important thing in trading in stocks is the consideration of past stock price data. However, existing RL algorithms used to stock markets do not include previous stock data when taking optimal actions, as RL is formulated based on the Markov property. To resolve this problem, we propose Transformer Actor-Critic (TAC) using GPT to train the model with the relation of previous MDP elements using an attention mechanism. Additionally, a critic method is applied to improve the result by training the parameters based on the evaluation of an action. For an effective training

method, we train TAC applying an offline RL algorithm through suboptimal trajectories. To solve the problem of overestimating the value of actions and reduce training time, we train TAC through a regularization method with an additional behavior cloning term. The experimental performances using several stock market data show that TAC performs better than other recent papers and RL in terms of the Sharpe ratio and portfolio value.

CHAPTER 1 Introduction

Recently, several machine learning algorithms such as reinforcement learning (RL), long short-term memory (LSTM), and transformers have been investigated for trading financial stock markets. The main purpose of such research is to predict the next stock values and/or automatically invest stocks based on specified optimization criteria. The reader is referred to [1–9] and the references therein for several learning methods applied to research financial markets. In fact, such methods show good performance, as stock values exhibit patterns. In particular, the RL techniques [1, 2] show high profits and stable returns compared with the other methods [10, 11]. Furthermore, the transformer and LSTM [3, 6] show overall good results of stock value predictions by learning a series of previous stock prices, compared to MLP and CNN [12]. This finding shows the necessity of utilizing previous stock information for stock market analysis.

We choose the RL algorithm to automatically manage portfolio assets in various portfolio allocation fields by taking optimal actions on a daily basis [1, 2]. However, when a policy takes an action based on the current state using RL in [1, 2], previous information cannot be used due to the inherent Markov property of RL. Although the methods in [8, 9] are proposed as a combined model of LSTM and transformers with RL, only the historical state is utilized among MDP elements, and LSTM has a vanishing gradient limit. Therefore, there is a problem when the model learns a long series. In addition, the transformer is not applied as a decision model for the inference of actions.

We believe that RL would have good performances when past data could be utilized as an combined model with transformers. Our research solves this problem. Specifically, we take into account the RL algorithm, which can consider historical

stock data. To resolve this problem, the decision transformer [13] is a appropriate algorithm, which adopts the GPT (Generative Pre-Training) model [14], where GPT has shown good results in the domain of NLP (natural language processing), which uses the attention method to infer the next word. By incorporating the GPT algorithm with the RL method, the GPT can infer the current action considering past MDP elements with the attention method. Hence, when constructing based on a decision transformer in stock trading, this model is able to be learned using previous data.

The decision transformer trains the model through the mean squared error (MSE), which is a loss objective function. Employing only MSE, it is difficult to get higher result than suboptimal trajectories generated with training data. We verify this opinion with an ablation study. Therefore, in this paper, we propose TAC [1] (Transformer Actor-Critic), a new RL method, which uses the critic network to the GPT model to improve the overall result of RL considering previous MDP elements. We get inspiration using an actor-critic method (Deep Deterministic Policy Gradient, DDPG) [15]. DDPG shows good performance when the critic is applied compared with that when only the actor is applied. Furthermore, we train TAC offline using created suboptimal datasets for cloning good actions and decreasing learning time. we remove the necessity to gather extra data online. However, when only the critic is used, the value function is unstable and inaccurate, leading to overestimation of action values and decreasing overall performance [16], as the agent does not interact with the environment. To resolve this limitation, we adopt the state-of-the-art regularization technique [17] with an applied behavior cloning term to DDPG. The regularization algorithm prevents actions from being overestimated by the Q function.

TAC shows the highest profits compared with other state-of-the-art algorithms [1, 2, 7, 18–21] using several datasets from Dow Jones, US 50, HighTech, NDX, MDAX, and CSI. In addition, we contain the terminology of the trading strategy turbulence in [1, 22] for more stable profits. Our contributions are listed as follows:

[1] Our code is available at: `https://github.com/VarML/TACR`

- We construct Transformer Actor-Critic (TAC), which infers current action with the decision transformer and then evaluates it by applying the critic to improve the performance of stock allocation.
- TAC is effectively trained offline through created suboptimal trajectories. Additionally, we employ a regularization technique to prevent overvaluing the value of actions.
- We describe the good performance of TAC, compared to the other recent RL and transformer methods on stock investment.
- Through various experiments, we show that applying the critic network and regularization algorithm increases performance, which robustly leads to good performances on several datasets.

1.1 Background

In this Section, we cover the basics of reinforcement learning [23]. Furthermore, we design the MDP for the stock investment problem, where we also describe how portfolio assets are allocated in portfolio allocation. Then we explain the terminology of *investment strategy*, which is needed to alleviate the sharp drop in portfolio profits. In fact, the trading strategy can compensate for the defects of the learning model, leading to higher profits. Additionally, we introduce the transformer's attention method to train the correlation between MDP elements.

1.1.1 Markov property

The Markov decision process is represented by a tuple $M = (S, A, T, r, \gamma)$. Here, S denotes the set of states, where s belongs to S and can be discrete or continuous (represented as multi-dimensional vectors). A represents the set of actions, where a belongs to A and can also be discrete or continuous. The system's dynamics are characterized by T, which represents a conditional probability distribution denoted as $T(s_{t+1}|s_t, a_t)$. The reward function, denoted as $r : S \times A \to R$, assigns

real numbers from the set R to combinations of states and actions. Lastly, γ is a discount factor that is a scalar value between 0 and 1.

The state signal is the information provided to the agent from the environment, and it influences the agent's decisions. The state signal should contain relevant information for decision-making but is not expected to provide complete knowledge of the environment. The state can be a processed version of sensory input or a complex representation built over time. The state signal should summarize past sensations compactly while retaining all relevant information, which is referred to as the Markov property. The Markov property means that the current state contains all the necessary information, independent of the history of previous states. The environment's dynamics can be defined by

$$Pr\{R_{t+1} = r, S_{t+1} = s^{'} | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \tag{1.1}$$

$$p(s^{'}, r|s, a) = Pr\{R_{t+1} = r, S_{t+1} = s^{'} | S_t, A_t\} \tag{1.2}$$

If the state signal follows the Markov property, (1.1) and (1.2) are equivalent. Information prior to t can be ignored.

1.1.2 Value functions

The majority of reinforcement learning algorithms involve the estimation of value functions, which are functions that estimate the desirability of being in a specific state (or performing a particular action in a given state). The notion of "desirability" is determined based on the expected future rewards that can be obtained, also known as the expected return. Since future rewards depend on the actions taken by the agent, value functions are defined with respect to specific

policies.

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{1.3}$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t = s\right] \tag{1.4}$$

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{1.5}$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t = s, A_t = a\right] \tag{1.6}$$

A policy, denoted as π, is a mapping that assigns a probability $\pi(a|s)$ to each state s and action a, representing the likelihood of taking action a when in state s. In (1.4), the value of a state s under a policy π, denoted as $v_\pi(s)$, represents the expected return when starting in state s and following policy π from that point onwards. γ serves to discount future rewards. In (1.6), $q_\pi(s,a)$ is called the "action-value function". The function calculates the expected value including the action. These functions can be estimated, and the expected value approximated through sampling in the Monte Carlo method.

In reinforcement learning and dynamic programming, value functions possess a fundamental characteristic whereby they adhere to specific recursive relationships. Irrespective of the policy π and state s, there exists a consistency condition between the value of s and the value of its potential successor states.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[R_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}\middle|S_t = s\right] \\ &= \sum_a \pi(a|s)\sum_{s'}\sum_r p(s',r|s,a)\left[r + \gamma\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}\middle|S_{t+1} = s'\right]\right] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \end{aligned} \tag{1.7}$$

To get an expected value, we compute the probability of each triple, $\pi(a|s)p(s',r|s,a)$, and multiply it by the quantity inside the brackets. Finally, we sum up these weighted quantities over all possibilities. The Bellman equation for v_π, represented in (1.7), establishes a connection between the value of a state and the values of

its potential successor states. It can be visualized as a process of considering the future outcomes from a given state and examining its various successor states. By assigning appropriate weights based on their respective probabilities, the Bellman equation in (1.7) calculates the average of all possible outcomes. This equation asserts that the value of the initial state is equal to the discounted value of the anticipated next state, combined with the expected reward during the transition. The value function v_π is the sole solution to its corresponding Bellman equation.

1.1.3 Markov Decision Process for portfolio allocation

We construct the stock investment problem as a Markov decision process (MDP) [24]. The elements of MDP composed of state, action, and reward are defined to design the portfolio allocation problem [25–27].

State space $\mathcal{S}$: We define the state space of MDP including meaningful stock data such as opening/closing/high/low stock prices and technical indicators. Technical indicators mean heuristic or pattern-based signals generated by price, volume, and/or open interest of a security or contract used by traders following technical analysis.

Action space $\mathcal{A}$: The action space is the set of the investment weights $\boldsymbol{a}_t = \{a_{0,t}, a_{1,t}, \ldots, a_{J,t}\}^\top$ satisfying $\sum_{j=0}^{J} a_{j,t} = 1$. Here, $a_{j,t}$ denotes the volume of allocation weight to trade in the jth stock for the current period t.

Reward $\mathcal{R}$: The instant reward is scalar and is represented as the sum of the rate of daily profits for each stock:

$$\boldsymbol{\rho_t} = \frac{\boldsymbol{p_{t+1}}}{\boldsymbol{p_t}} = (\frac{p_{0,t+1}}{p_{0,t}}, \frac{p_{1,t+1}}{p_{1,t}}, ..., \frac{p_{J,t+1}}{p_{J,t}})^\top \tag{1.8}$$

$$r_{t+1} = r(s_t, a_t) = \boldsymbol{a}_t \cdot (\boldsymbol{\rho}_t - \mathbf{1}), \tag{1.9}$$

where $\boldsymbol{p_t} \in \mathbb{R}_+^J$ is the closing price of stocks for the current period t. Note that (1.8) computes the ratio of the next day's stock prices to present stock prices. Finally, the policy maximizes the cumulative reward while increasing the allocation weights of the positive high-ratio stocks in (1.9).

1.1.4 Investment strategy

Typically, RLs [1, 28] perform well in the stock trading field. However, in early 2020, the pandemic caused stock movements to drop abruptly, leading to poor returns. To solve this situation, an index that represents the volatility of stock movements is needed. When volatility is huge, the allocation weights have to be adjusted. We add a stock movements indicator to preserve assets. This indicator is called *turbulence*, which was applied in the NeurIPS Workshop 2020 [1] and was proposed by [22]:

$$\text{turbulence}_{\text{t}} = (\boldsymbol{p_t} - \boldsymbol{\mu})^{\top} \boldsymbol{\Sigma^{-1}} (\boldsymbol{p_t} - \boldsymbol{\mu}) \in \mathbb{R}, \tag{1.10}$$

where $\boldsymbol{p_t} \in \mathbb{R}_+^J$ denotes the stock prices for the current period t, and $\boldsymbol{\mu} \in \mathbb{R}_+^J$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{J \times J}$ represents the mean and the covariance of previous stock prices, respectively. This index is used as a constraint when taking allocation weights. We explain in more detail in Subsection 2.0.1.

1.1.5 Transformers

We explain how the attention technique is composed to compute the correlation for each token as follows:

$$\texttt{query} = \tilde{h}_{\tilde{l}} W_q,\ \texttt{key} = \tilde{h}_{\tilde{l}} W_k,\ \texttt{value} = \tilde{h}_{\tilde{l}} W_v \tag{1.11}$$

$$h_{\tilde{l}+1} = \texttt{Attention}(\texttt{query}, \texttt{key}, \texttt{value}) \tag{1.12}$$

$$= \texttt{Softmax}(\frac{\texttt{query} \cdot \texttt{key}}{\sqrt{d_k}})\texttt{value}$$

$$h_{\tilde{l}+2} = h_{\tilde{l}+1} + h_{\tilde{l}} \tag{1.13}$$

$$\texttt{FFN}(\tilde{h}_{\tilde{l}+2}) = \max(0, \tilde{h}_{\tilde{l}+2} W_{\tilde{l}} + b_{\tilde{l}}) W_{\tilde{l}+1} + b_{\tilde{l}+1} \tag{1.14}$$

$$h_{\tilde{l}+3} = h_{\tilde{l}+2} + \texttt{FFN}(\tilde{h}_{\tilde{l}+2}), \tag{1.15}$$

where $\tilde{l}$ is the layer of the hidden state in a particular layer. In (1.11), three weight matrices represent: `query` weights W_q, `key` weights W_k, and `value` weights W_v.

The hidden layer $\tilde{h}_{\bar{l}}$ with layer normalization [29] used is changed into matrices `query`, `key`, and `value` by multiplying each weight. In (1.12), the similarities between each `query` and `key` are calculated by applying a dot product. To alleviate the gradient from becoming zero in the Softmax layer, $\sqrt{d_k}$, where d_k is the dimension of `key`, is divided for scaling. In addition, the Softmax function is used to gain the weights for the `value`. In (1.14), the FFN (Feed Forward Neural Network) contains two linear layers, which include the ReLu activation function. Furthermore, the residual connection [30] in (1.13) and (1.15) prevents the problem of gradient diverging or vanishing, which also makes faster learning by keeping a stable gradient value.

1.2 Related work

We describe two applications in the stock field: stock prediction and stock investment. Various algorithms such as LSTM, transformer, and RL are employed with several datasets.

LSTM: Supervised learning is useful with a deep learning algorithm, as the stock datasets are easy to gain. LSTM [31] is able to train relatively longer series than RNN (Recurrent Neural Network) [12]. Therefore, LSTM has good performance in predicting the following stock price. It trains the model with time sequence data on past stock prices [3]. Another element to consider when trading stocks is news. Stock movement often relates to various news that affect the financial economy. [4, 5] show that including news or tweet data as features leads to better accuracy performance compared to using only LSTM. This model adopts VADER [32] to compute emotional scores based on the news.

Transformer: Transformer [33] has had a huge impact on the area of NLP (natural language processing). Additionally, Bert [34] and GPT [35] algorithms, which consist of the encoder and the decoder, respectively, show better accuracy as the scale of the parameters increases. The transformer is also adopted in the stock field to learn patterns of historical data of stocks [6]. Furthermore, [7] learns the correlation between each stock and a market index. As each stock often relates

to the market index, it provides useful information to more accurately predict the next stock prices.

Reinforcement learning (RL): RL is the appropriate algorithm for the stock investment, where sequential decision making is needed. Some representative RL algorithms (DDPG [15], TD3 [19], SAC [20], PPO [21], A2C [36]) are used to stock investment in [1]. Another stock investment research [8] is investigated by combining A3C [36] with LSTM. When defining the state features, RL improves the performance by including the state into significant features related to the stock movement using SARL [2]. In fact, SARL shows the good performance of RL by including the news embeddings vector as the augmented state. SARL inputs embedding into a hierarchical attention network [37] to train a binary classifier model to infer the stock movement.

Transformer + RL: To propose an ensemble model, the transformer and RL are adopted in a combined form. [9] uses the transformer's attention technique to the RL model to show the good performance of the stock investment. As mentioned in [7], the attention mechanism is also applied to train the correlation between stock prices. Our TAC is also a combined model of RL and transformer. However, TAC trains the interrelation with MDP elements as tokens through an offline RL algorithm. TAC is based on [13, 38], which shows good performance in physics simulation and has not been used in the stock trading field. The impact of TAC is that it can consider stocks with previous data via the transformer attention method. Furthermore, we experiment with various datasets to prove the wide range of applicability.

CHAPTER 2 Transformer Actor-Critic

Analyzing price patterns from historical data is crucial in stock investments as it enables investors to make more informed decisions. In the past, different investment strategies [39, 40] were developed using historical stock data to generate profitable returns without relying on machine learning techniques. However, in recent times, various methods [3, 7] that utilize machine learning algorithms such as LSTM and transformer to train correlation by leveraging past price information have been studied.

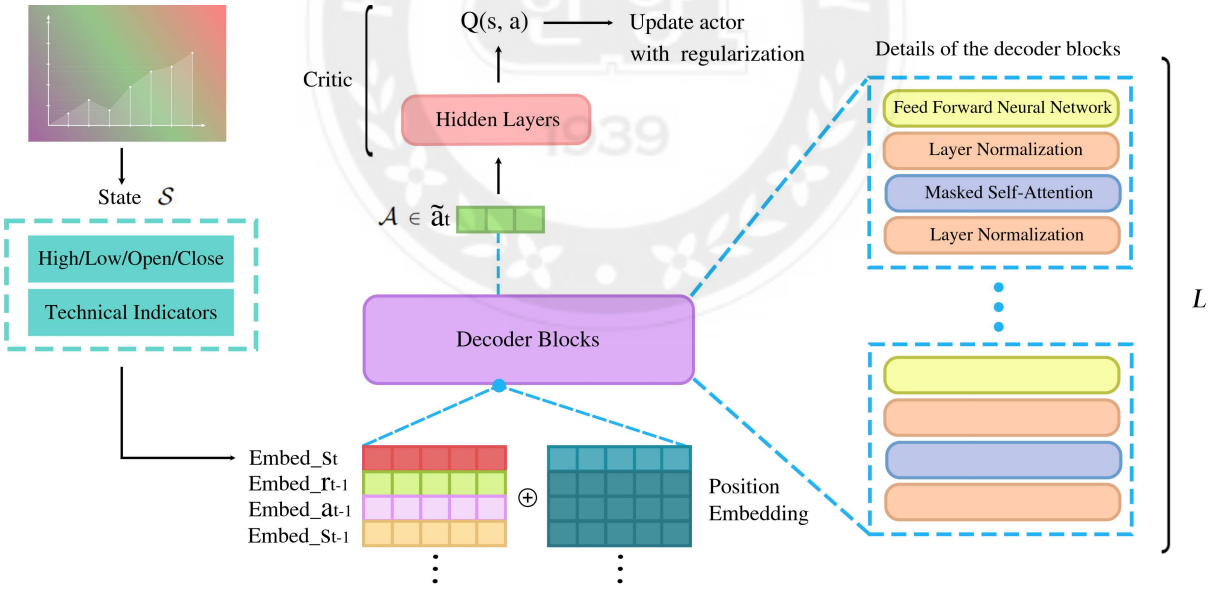

Figure 2.1 The TAC framework involves incorporating different stock prices and technical indicators to construct state features. The transformer then uses the embeddings of past MDP elements and the current state as inputs to predict the action $\tilde{a}_t$ using decoder blocks. Finally, the policy is learned via the estimated Q-function with the regularization technique.

In the case of RL, state information alone may not suffice to make optimal decisions, since RL is formulated based on the MDP framework. To overcome this limitation and enhance performance, we introduce TAC (Transformer Actor-Critic), which employs the decision transformer to leverage past stock information and MDP components. In TAC, the decision transformer functions as an actor by predicting an action, which is then assessed by a critic network that incorporates a regularization technique to improve performance. Specifically, we utilize GPT [14] as the actor network and the Q-function as the critic network in TAC. Figure 2.1 provides a visual representation of the complete TAC architecture. To construct state features, we include opening/closing/high/low stock prices and technical indicators such as MACD, BOLL, RSI, CCI, DX, and SMA. Then, we transform not only the current state but also previous states, actions, and rewards into embeddings of a specific dimension. TAC leverages the decoder blocks to learn the correlation of each MDP element and produce the current action as an output. The predicted action is then assessed by the critic, which uses hidden layers to approximate its value and update the parameters of the actor. For a more detailed understanding of TAC, we explain how the policy infers the action in Subsection 2.0.1 and how the policy updates the parameters based on the critic with regularization method in Subsection 2.0.2.

2.0.1 Transformer Actor

The algorithm of the transformer actor in TAC maps the previous MDP elements to the current action. The actor network π comprises multiple decoder blocks that employ the attention mechanism described in Subsection 1.1.5, along with hidden layers structured as follows:

$$h_0 = MW_e + W_p \tag{2.1}$$

$$h_l = \texttt{Decoder\_block}(h_{l-1}),\ l = 1, \ldots, L \tag{2.2}$$

$$\tilde{a} = \texttt{Softmax}(h_L W_L + b_L), \tag{2.3}$$

where (2.1) uses the matrix $M = (r_{-u}, s_{-u}, a_{-u}, \ldots, r_{-1}, s_{-1})$, which comprises token vectors of prior MDP elements with a length of u, indicating the sequence

length of the previous tokens. (2.1) multiplies the MDP elements with the weight embedding matrix W_e to represent the hidden state as an input. Then, the position embedding weights W_p are added to include information on temporal continuity. The correlation of the embedding inputs is learned through L decoder blocks via (2.2). The action $\tilde{a} = \{\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_J\}^\top$ is predicted using a linear transformation layer and a Softmax function in (2.3). The predicted action represents the allocation weight, with $\sum j = 0^J \tilde{a}_j = 1$, ensuring the sum of the weights equals 1.

As a constraint to prevent a sharp decline in the final step of action prediction at each time step, we implement the turbulence strategy described in Subsection 1.1.4 as follows:

$$\tilde{a}_t = \begin{cases} \mathbf{0}, & \text{if turbulence}_\text{t} > \text{threshold} \\ \pi(M), & \text{otherwise} \end{cases} \tag{2.4}$$

We use the strategy where all held shares are sold when the indices surpass a particular threshold, and the agent can take automatic action otherwise. This strategy is employed to meet the same conditions as those of the NeurIPS Workshop 2020.

2.0.2 Behavior Cloning and Critic with Regularization

Instead of the usual off-policy method used in stock trading, we train the model using an offline method. In this approach, the agent does not interact with the environment and learns through trial and error. Instead, it imitates prepared suboptimal actions with the expectation of better performance and reduced learning time. To train the model using the offline method, we need to create suboptimal trajectories by pairing the suboptimal actions corresponding to each state. Since past data is abundant, we can easily create suboptimal data as follows:

$$score_{j,t} = e^{\rho_{j,t} \cdot c} \tag{2.5}$$

$$\texttt{suboptimal\_action}_{j,t} = \frac{e^{score_{j,t}}}{\sum_j e^{score_{j,t}}}, \tag{2.6}$$

where (2.5) provides a score based on the rate of increase in the stock price, where $\rho_{j,t}$ is defined in (1.8) and c is a constant within the range of 1-5, and in (2.6). The sum of the actions equals 1 at each time step. The reward is then calculated as described in (1.9), and this process completes a set of states, actions, and rewards at each time step.

RL methods applied in stock trading often use off-policy algorithms, such as replay buffer, to improve data efficiency. However, this approach has a drawback that it requires additional time to interact with the environment in each episode to improve the policy. To address this issue, we adopt an offline training method for TAC, similar to the use of expert trajectories in physics simulations provided in D4rl [41]. By generating pre-generated suboptimal datasets, TAC can significantly reduce training time and improve performance with relatively few iterations, as it does not need to interact further with the environment. This differs from RL approaches used in the stock field, such as [1, 2], which utilize off-policy algorithms.

Furthermore, in offline RL algorithms, there is a limitation of not being able to interact with the environment while updating the policy, which can cause agents to inaccurately approximate the value of actions for unseen states. This can result in poor performance since the optimal action may not be taken. To address this problem and improve the accuracy of evaluating out-of-distribution actions, a regularization method [17] based on TD3 [19] is required.

We modify the policy update process by incorporating a regularization method using the critic as follows:

$$\begin{aligned} \pi &= \underset{\pi}{\operatorname{argmax}} \bar{J} \\ &= \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{(s,a,r)\sim D}\Big[\lambda Q(s, \pi(M)) - (\pi(M) - a)^2\Big], \end{aligned} \tag{2.7}$$

where the input of transformer actor π is represented by the sequence M, which stacks a certain number of MDP elements. To ensure that the transformer actor follows the distribution of actions in the suboptimal trajectories contained in the dataset, we incorporate a behavior cloning regularization term into DDPG.

Note that the hyperparameter λ serves as a control parameter that regulates both the maximization of Q and the minimization of the behavior cloning term. It is essential to emphasize that λ is sensitive to the scale of Q. In other words, λ is a scalar that measures the intensity of the regularizer. λ with randomly sampled by N transitions (s_i, a_i) is defined by

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|}, \tag{2.8}$$

where α is a hyperparameter that regulates the intensity of λ. Note that (2.8) is used to be normalization, where the denominator $\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|$ denotes the sample mean of $|Q(s_i, a_i)|$. In (2.8), as the Q is an undifferentiating scalar, it acts as a normalization of the learning rate by adjusting the gradient $\nabla_a Q(s, a)$.

2.0.3 Algorithm

Algorithm 1 outlines the TAC algorithm we propose for trading multiple stocks concurrently. To enable continuous actions, we adopt DDPG, an actor-critic algorithm that has demonstrated its good performance in various domains, including robotics, energy management, and recommendation systems [42–44]. We leverage DDPG as a foundation for building the TAC algorithm.

Firstly, we randomly initialize the weights of the critic and transformer actor. Next, we set the weights of the target networks. The hyperparameter u determines the length of the sequence of previous MDP elements to be fed as input to the transformer actor. To break the correlation between the data as suggested by [45], we randomly sample $n \times u$ transitions based on the minibatch size n and the sequence length u.

Through the attention network, the transformer actor learns the correlation among the stacked embedding MDP elements M and generates an output that represents the predicted action for the current state. The parameters of the critic network are updated by using the mean squared error method with the θ^Q. Here,

$Q(s_i, a_i|\theta^Q)$ represents an approximation of Q.

$$\text{Loss} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2. \tag{2.9}$$

Then we update the transformer actor parameter θ^π using the chain rule with respect to both the Q term and the behavior cloning term, in order to optimize the objective function $\bar{J}$ of DDPG with regularization given by (2.7). $\pi(M_i^e|\theta^\pi)$ represents an approximation of the policy π.

$$\nabla_{\theta^\pi}\bar{J} \approx \frac{1}{N}\sum_i \nabla_{\tilde{a}}\Big\{\lambda Q(s, \tilde{a}|\theta^Q) - (\tilde{a} - a)^2\Big\}|_{s=s_i, \tilde{a}=\pi(M_i|\theta^\pi)} \nabla_{\theta^\pi}\pi(M_i|\theta^\pi). \tag{2.10}$$

To ensure learning stability, we gradually update the weights of the target networks using the equation: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, where τ is hyperparameter. We set $\tau \ll 1$.

Algorithm 1 TAC algorithm

Randomly set critic weights θ^Q and transformer actor weights θ^π.
Set target weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\pi'} \leftarrow \theta^\pi$
Initialize the length of the sequence u and minibatch size n.
for each iteration **do**
  for each environment step **do**
    Sample a random minibatch of $n \times u$ transitions $(s_{i-u}, ..., s_i, a_{i-u}, \ldots, a_i, r_{i-u}, \ldots, r_i, s_{i+1-u}, \ldots, s_{i+1})$ from suboptimal trajectories.
    $M_i = (r_{i-u}, s_{i-u}, a_{i-u}, ..., r_i, s_i)$
    Output action $\tilde{a} = \pi(M_i|\theta^\pi)$ using transformer
    Set $y_i = r + \gamma Q'(s_{i+1}, \pi'(M_{i+1}|\theta^{\pi'})|\theta^{Q'})$
    Use mean squared error to train critic in (2.9):
$$\text{Loss} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2,$$
    where $N = n \times u$.
    Set hyperparameter α and $\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i,a_i)|}$ in (2.8)
    Train the transformer actor network in (2.10):
$$\nabla_{\theta^\pi} \bar{J} \approx \frac{1}{N}\sum_i \nabla_{\tilde{a}} \Big\{ \lambda Q(s, \tilde{a}|\theta^Q) \; -(\tilde{a} - a)^2 \Big\}|_{s=s_i, \tilde{a}=\pi(M_i)} \nabla_{\theta^\pi} \pi(M_i|\theta^\pi)$$
    Train the target weights:
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'}$$
  end for
end for

CHAPTER 3 Experiment

3.0.1 Datasets

To compare TAC with state-of-the-art methods, we utilize the same datasets. Additionally, we demonstrate the general applicability of TAC by presenting experimental results using various datasets:

KDD 2021 [7] selects 50 US stocks, which we select as the best performing dataset in KDD 21. [7] divides the dataset into training data from *2007-01-03* to *2015-12-31* and testing data from *2016-01-04* to *2016-12-30*.

AAAI 2020 [2] chooses nine stocks from the HighTech daily dataset [46]. [2] divides the dataset into training data from *2006-10-20* to *2012-11-15* and testing data from *2012-11-16* to *2013-11-20*.

NeurIPS Workshop 2020 [1] includes the Dow 30 daily dataset containing 30 prominent companies listed on stock exchanges in the United States. [1] divides the dataset into training data from *2009-01-01* to *2018-12-31* and testing data from *2019-01-01* to *2020-09-23*.

NDX, MDAX, CSI are the stock market indices of the US, Germany, and China, respectively. We choose 30 random stocks for each index and use them as a dataset. We divide the datasets into training data from *2009-01-01* to *2020-08-31* and testing data from *2020-09-01* to *2021-12-30*.

3.0.2 Performance

All datasets are based on daily transactions. We use the widely-used Portfolio Value (PV) and Sharpe Ratio (SR) [47] as evaluation metrics. PV represents the

model	TAC	AAAI 20	NeurIPS 20
HighTech	**2.70**	2.37	-
DOW	**0.86**	-	0.76

Table 3.1 We compare the Sharpe ratios of TAC with those of other papers using the HighTech and DOW datasets.

model	TAC	CQL	PPO	SAC	EW
NDX	**1.59**	1.29	1.29	1.29	1.31
MDAX	1.81	**1.96**	1.85	1.61	1.94
CSI	**1.00**	0.22	0.36	0.25	0.27

Table 3.2 We compare the Sharpe ratios of TAC, other RL algorithms, and a naive baseline using the NDX, MDAX, and CSI datasets.

final asset value, while SR measures the stability of the portfolio:

$$\mathbf{PV} = pv_0 \prod_{t=1}^{T} (r_{t+1} - tf \sum_j |a_{j,t} - a_{j,t-1}| + 1) \tag{3.1}$$

$$\mathbf{SR} = \frac{R_p - R_f}{\sigma_p}, \tag{3.2}$$

where pv_0 represents the initial amount of investment (set as 1 million dollars), tf is the transaction fee (set as 0.25%), and r_{t+1} is the daily return rate computed using (1.9). R_p and R_f denote the portfolio return rate and the risk-free rate, respectively, and σ_p denotes the standard deviation of the portfolio return. We set $R_f = 2\%$ to represent the annual bank interest rate in (3.2).

To evaluate the performance of our model in portfolio allocation, we compare it with other existing approaches, including offline RL, off-policy RL, on-policy RL, and classic methods as baselines:

- **KDD 21, AAAI 20, NeurIPS Workshop 20** [1, 2, 7]. These are the state-of-the-art papers showing good performance with portfolio allocation.

- **CQL.** Conservative Q-learning [18] is a representative model of offline RL. We train the model with suboptimal datasets introduced in (2.6).
- **SAC.** Soft actor-critic [20] is an off-policy method proposed to solve the problem of application in the real world domain.
- **PPO.** Proximal policy optimization [21] is an on-policy algorithm that has stability and reliability and compensates for the defect of policy gradient learning.
- **EW.** Equal weight is a naive baseline that allocates all stocks with the same weight in a way that does not use machine learning algorithm.

We compare TAC's performance with other baselines, including other model-free RL algorithms and classic methods, to demonstrate its superiority in portfolio allocation. Figure 3.1 clearly shows that TAC outperforms all baselines on various datasets. Note that for KDD 21 and AAAI 20, we have to compare our results with the figures presented in [7] and [2], respectively, since their codes and hyper-parameters were not available. In addition, our model achieves the highest Sharpe ratio on most datasets, as shown in Tables 3.1 and 3.2. This indicates that our model provides more stable returns compared to other algorithms. Table 3 presented in [2] compares our model's Sharpe ratio performance with that of AAAI 20 for the 6-month period. Unfortunately, the Sharpe ratio performance of KDD 21 cannot be included in Table 3.1 since information about the Sharpe ratio is not provided in that paper.

3.0.3 Ablation study

An ablation study is conducted to demonstrate the necessity of the critic and regularization method in the decision transformer of TAC. The results, presented in Table 3.3, indicate that the performance of TAC reduces the performances on various datasets (NDX, MDAX, CSI) when either of the modules is removed.

- TAC: Transformer Actor-Critic with Regularization;

	model	TAC	TAC w/o RG	TAC w/o RG, CT
NDX	PV	**2.13**	1.56	1.02
	SR	**1.59**	0.97	0.13
MDAX	PV	**6.07**	6.07	1.24
	SR	**1.81**	1.81	0.94
CSI	PV	**1.65**	0.85	0.31
	SR	**1.00**	-0.07	-3.20

Table 3.3 Ablation study

	u (length)	20	10	1
NDX	PV	**2.13**	1.56	1.56
	SR	**1.59**	0.97	0.97
MDAX	PV	**6.07**	6.07	6.07
	SR	**1.81**	1.81	1.81
CSI	PV	**1.65**	1.38	1.03
	SR	**1.00**	0.74	0.24

Table 3.4 Comparison of the sequence length

- TAC w/o RG: TAC without the regularization (RG);
- TAC w/o RG, CT: TAC without the regularization and critic (CT).

The notation "w/o RG, CT" denotes the absence of any module in TAC. In addition, the performance of "TAC w/o RG" is presented, where only the critic is included. As expected, the performance is the worst with "w/o RG, CT". However, "TAC w/o RG" exhibits better performance than "w/o RG, CT" by including the critic. By incorporating the regularization method, TAC shows better performance than "TAC w/o RG". The regularization method aids in preventing overestimation of action values and selecting better actions on new observations. Ultimately, TAC, which includes all modules (critic and regularization), shows the best performance. The ablation study proves the necessity of each module in TAC.

3.0.4 The length of the sequence

The hyperparameter u can adjust the length of the previous MDP sequence. To study its effect, we analyze the performance of TAC with increasing u on different datasets (NDX, MDAX, CSI) in Table 3.4. The range of u is from 1 to 20, with increments of 10. As shown in the table, most datasets exhibit better results with increasing u, which leads to higher profits and stability. This suggests that considering a longer history of MDP elements enables TAC to take actions closer to the optimal. It is possible that the price flow factor plays a role in determining the actions. By utilizing the attention network, TAC overcomes the limitations of RL due to limited observations and effectively learns the relationships among past information.

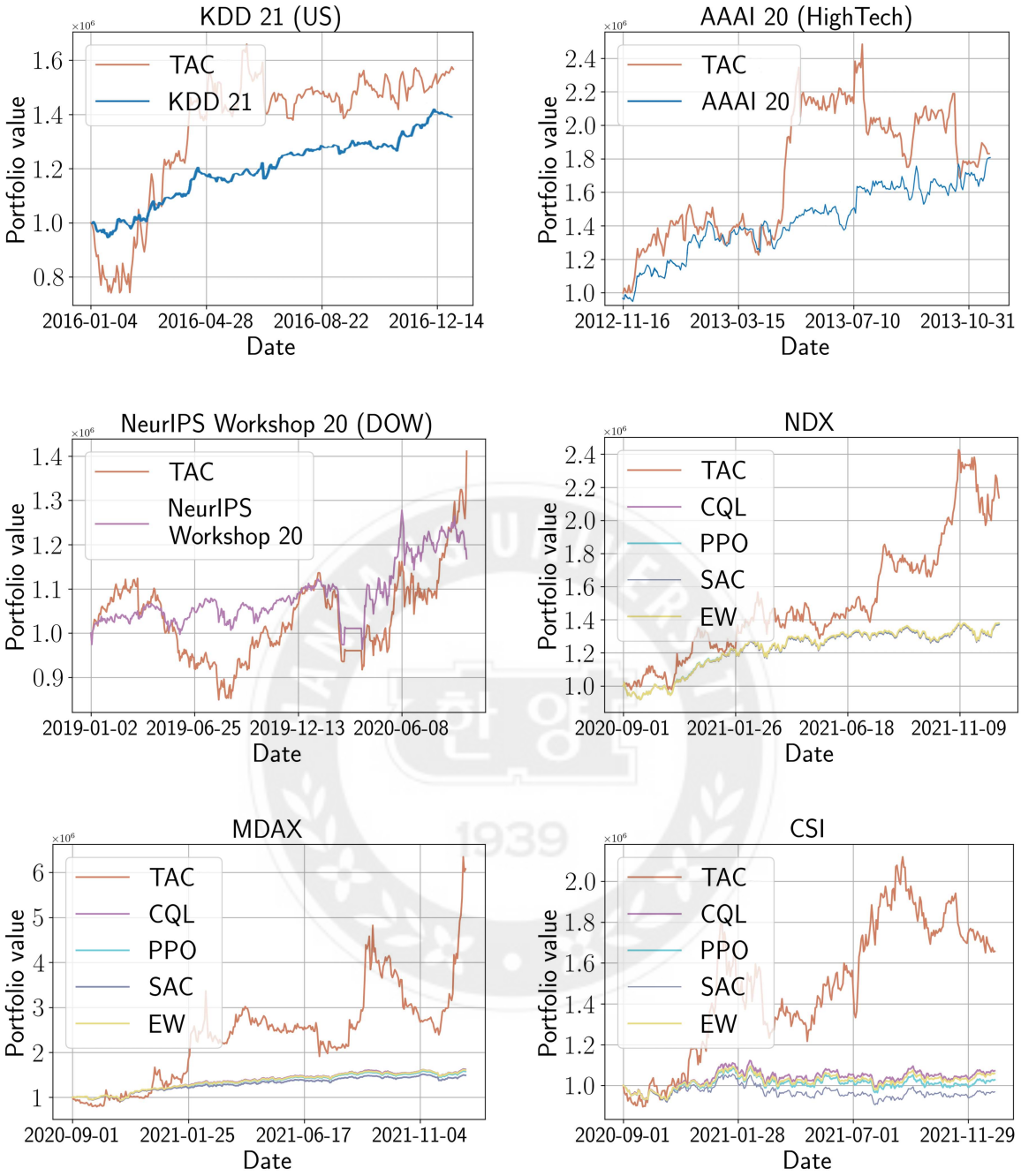

Figure 3.1 The performance of TAC is compared to other papers (KDD 21, AAAI 20, and NeurIPS Workshop 20), as well as other existing RL approaches (CQL, SAC, and PPO), and a naive baseline (EW) using six different datasets. TAC demonstrates significant improvements, with portfolio values that are 13% higher for KDD 21, 1.1% higher for AAAI 20, 20.7% higher for NeurIPS Workshop 20, 54.8% higher for NDX, 274.8% higher for MDAX, and 54.5% higher for CSI, as compared to the other algorithms.

CHAPTER 4 Conclusions

We present TAC, a model designed for analyzing stock markets using historical data in stock trading. TAC includes a critic network in the decision transformer and a regularization method, and it is trained using suboptimal trajectories offline, making it more efficient than other models. TAC allows the RL method to consider both past stock data and previous MDP elements to take optimal actions, which distinguishes it from standard RL techniques based on the MDP framework. Additionally, TAC prevents overestimation of action values. TAC shows good performance for various datasets compared to state-of-the-art methods. However, similar to other models, TAC's performance drops when stock prices fall sharply due to external factors such as pandemics, which cause the environment to lead out of distribution during testing data. Future research will focus on developing a model that can analyze external factors under unseen observations.

Bibliography

[1] X.-Y. Liu, H. Yang, Q. Chen, R. Zhang, L. Yang, B. Xiao, and C. D. Wang, "Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance," *arXiv preprint arXiv:2011.09607*, 2020.

[2] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1112–1119.

[3] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1419–1426.

[4] X. Li, Y. Li, H. Yang, L. Yang, and X.-Y. Liu, "DP-LSTM: Differential privacy-inspired LSTM for stock prediction using financial news," *arXiv preprint arXiv:1912.10806*, 2019.

[5] D. Chen, Y. Zou, K. Harimoto, R. Bao, X. Ren, and X. Sun, "Incorporating fine-grained events in stock movement prediction," *arXiv preprint arXiv:1910.05078*, 2019.

[6] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale gaussian transformer for stock movement prediction." in *IJCAI*, 2020, pp. 4640–4646.

[7] J. Yoo, Y. Soun, Y.-c. Park, and U. Kang, "Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts," in

Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2037–2045.

[8] E. Ponomarev, I. V. Oseledets, and A. Cichocki, "Using reinforcement learning in the algorithmic trading problem," *Journal of Communications Technology and Electronics*, vol. 64, no. 12, pp. 1450–1457, 2019.

[9] J. Wang, Y. Zhang, K. Tang, J. Wu, and Z. Xiong, "Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1900–1908.

[10] B. Li and S. C. Hoi, "On-line portfolio selection with moving average reversion," *arXiv preprint arXiv:1206.4626*, 2012.

[11] L. Gao and W. Zhang, "Weighted moving average passive aggressive algorithm for online portfolio selection," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1. IEEE, 2013, pp. 327–330.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, 2021.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[17] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[19] S. Dankwa and W. Zheng, "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–5.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

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

[22] M. Kritzman and Y. Li, "Skulls, financial turbulence, and risk management," *Financial Analysts Journal*, vol. 66, no. 5, pp. 30–41, 2010.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] ——, *Reinforcement learning: An introduction*. MIT press, 2018.

[25] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *arXiv preprint arXiv:1706.10059*, 2017.

[26] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," *arXiv preprint arXiv:1808.09940*, 2018.

[27] X. Li, Y. Li, Y. Zhan, and X.-Y. Liu, "Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation," *arXiv preprint arXiv:1907.01503*, 2019.

[28] 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*, 2018.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[32] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 216–225.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[35] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[37] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[38] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, 2021.

[39] L. C. MacLean, E. O. Thorp, and W. T. Ziemba, *The Kelly capital growth investment criterion: Theory and practice.* world scientific, 2011, vol. 3.

[40] A. Ang, "Mean-variance investing," *Available at SSRN 2131932*, 2012.

[41] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[42] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for list-wise recommendations," *arXiv preprint arXiv:1801.00209*, 2017.

[43] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 601–627, 2019.

[44] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577–8588, 2019.

[45] 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.

[46] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Using structured events to predict stock price movement: An empirical investigation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1415–1425.

[47] A. W. Lo, "The statistics of sharpe ratios," *Financial analysts journal*, vol. 58, no. 4, pp. 36–52, 2002.

APPENDIX A Experimental Details

Software:

- Linux Ubuntu 20.04.3
- Python 3.9
- Pytorch 1.10.1
- Transformers 4.5.1
- Gym 0.19.0
- Anaconda 4.12.0
- Numpy 1.21.2

Hardware:

- Intel Core i9-10980XE CPU at 3.00GHz.
- GPU RTX A5000

Hyperparameters. The default hyperparameters for the GPT transformer model are used, with a focus on the context length (u) and α, as defined in (2.8). By setting other hyperparameters to specific values and tuning only u and α, we achieve the best performance across all datasets. The detailed hyperparameters can be found in Table A.1. To test the impact of u, we vary its value from 1 to 60 with increments of 10, and find that performance metrics such as Sharpe ratio and profit improved as u increased, up to a maximum of 40. Additionally, we experiment with different values of α (ranging from 0.1 to 2.0 in increments of 0.1) and find that the highest Sharpe ratio and profit are achieved when α is set to 1.6 in the US, 1.4 in the DOW, 2.0 in the HighTech, 0.9 in the NDX, MDAX, and

CSI. The MDAX is found to be insensitive to changes in α. Our algorithm TAC is modeled using a combination of the transformer and regularization techniques. Note that while we are able to reproduce the NeurIPS Workshop 2020 from `https://github.com/AI4Finance-Foundation/FinRL`, we are unable to reproduce the results of [2, 7], as the code and hyperparameters are not provided in those papers. To construct our model, we refer to the GPT model and regularization method as below:

- `https://github.com/kzl/decision-transformer`
- `https://github.com/sfujim/TD3_BC`

Environment. To conduct our stock trading experiments using RL, we utilize the stock data, stock trading environment, and allocation experiments code provided by the NeurIPS Workshop 2020. In constructing the state for each stock, we include technical indicators such as MACD, BOLL, RSI, CCI, DX, and SMA, as well as prices (high, low, opening, and closing). The dimension of the state varies across datasets, with HighTech having a 108-dimensional state and the other datasets having a 360-dimensional state. The number of stocks invested in determines the dimension of the action. Moreover, we employ an investment strategy where the turbulence threshold is set to 100. This strategy is used exclusively for the NeurIPS Workshop 2020 dataset [1], whereas the other datasets achieve good performance without the investment strategy.

Hyperparameter	Value
Number of layers	5
Number of attention heads	1
Embedding dimension	128
Batch size	64
Context length	20 (NDX, MDAX, CSI), 40 (US, HighTech, DOW)
Activation function	ReLu
Dropout	0.1
Learning rate	10^{-4}
Grad norm clip	0.1
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training step
Critic hidden dim	(512 x 256), (256 x 1)
Critic hidden layers	2
Critic activation function	ReLu
Critic learning rate	10^{-4} (HighTech) 10^{-6} (Others)
seed	0
α	0.1 - 2.0

Table A.1 Transformer and critic hyperparameters of TAC for US 50, HighTech, DOW, NDX, MDAX, and CSI dataests

	Features 1 (Section) "Experiment")	Features 2 (Additional)	w/o TI
State	Opening	Opening	Opening
	Closing	Closing	Closing
	High	High	High
	Low	Low	Low
	MACD	TRIX_30	
	BOLL_UB	MVAR_30	
	BOLL_LB	EMA_30	
	RSI_30	RSI_20	
	CCI_30	WR_30	
	DX_30	ATR	
	SMA_30, 60	DMA	

Table A.2 State features for additional experiments

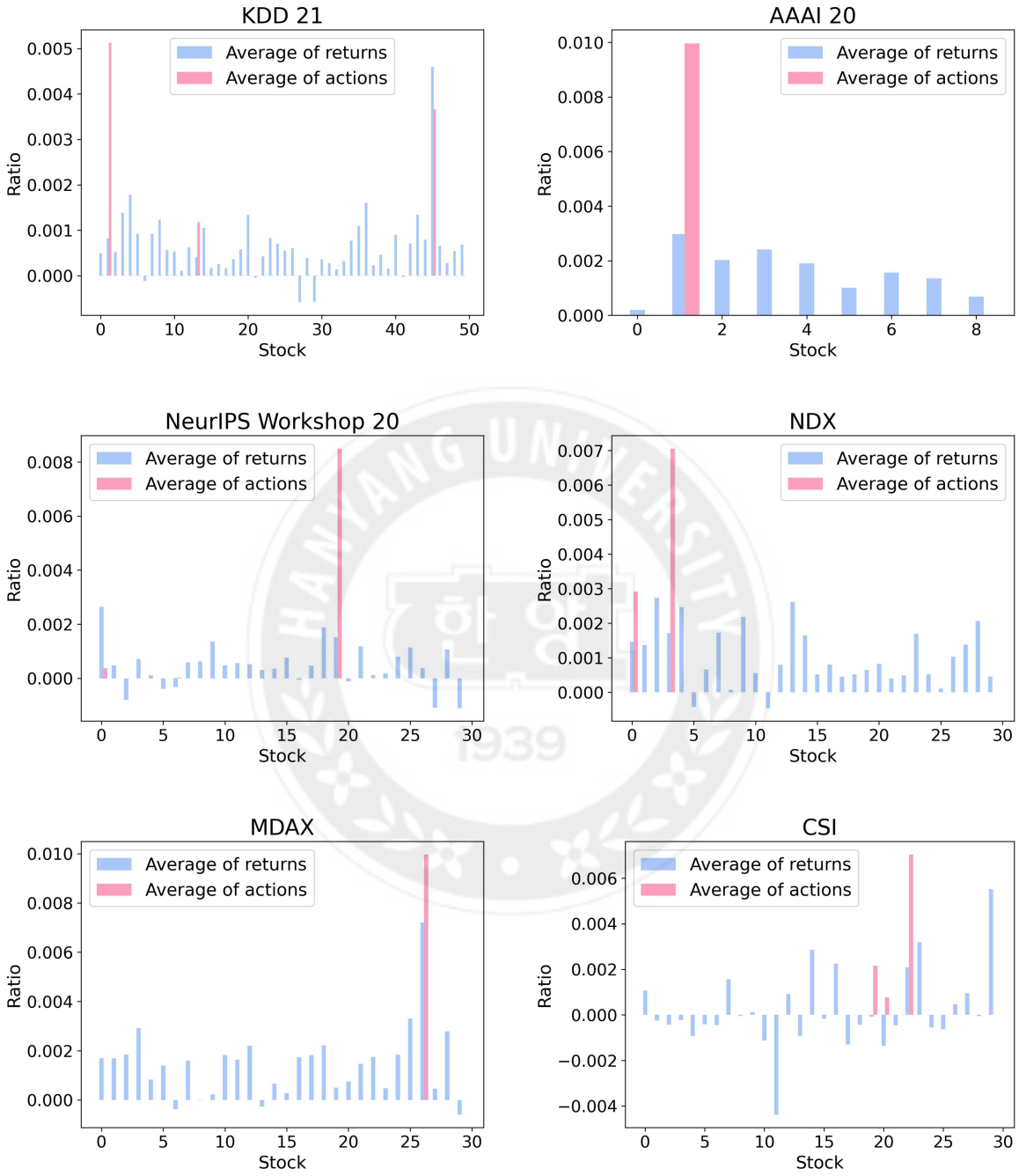

Figure A.1 The x-axis depicts each stock, while the y-axis shows the average of actions and returns. To facilitate the comparison between the two, the average of actions is normalized by dividing it by 100.

APPENDIX B Additional Experiments

The RL baselines CQL, PPO, and SAC usually allocate almost equal weights while taking actions, resulting in their performances being similar to that of the EW baseline. To demonstrate TAC's superiority over other algorithms in assigning actions, we present a graph as an initial supplementary experiment.

Constructing meaningful state information is crucial for enabling the agent to make high-reward decisions. Therefore, we enhance the state feature by including technical indicators and stock prices (opening, closing, high, low) in Figure 2.1. To evaluate the impact of technical indicators, we present the second set of experimental results for the HighTech, DOW, NDX, MDAX, and CSI datasets, both with and without different sets of technical indicators (refer to Table A.2). Note that we exclude the KDD 21 dataset since it does not contain technical indicators.

We present a categorization of the technical indicators used in the primary and supplementary experiments as follows:

- **Moving average**:
  - Moving Average Convergence Divergence (MACD)
  - Exponential Moving Average (EMA)
  - Different of Moving Average (DMA)
  - Directional Movement Index (DX)
  - Triple Exponential Moving Average (TRIX)
  - Simple Moving Average (SMA)
- **Overbought/Oversold**:

		Features 1	Features 2	w/o TI
Hightech	PV	**1.83**	1.22	1.83
	SR	**1.35**	0.65	1.35
DOW	PV	**1.41**	1.30	1.59
	SR	**0.86**	0.65	1.15
NDX	PV	**2.13**	1.44	1.56
	SR	**1.59**	0.77	0.97
MDAX	PV	**6.07**	6.07	6.07
	SR	**1.81**	1.81	1.81
CSI	PV	**1.65**	1.37	0.81
	SR	**1.00**	0.72	-0.43

Table B.1 Comparison of technical indicators

  - Relative Strength Index (RSI)
  - Williams Overbought/Oversold index (WR)
  - Commodity Channel Index (CCI)

- **Volatility**:
  - Moving Variance (MVAR)
  - Average True Range (ATR)
  - Bollinger (BOLL).

We classify the technical indicators into three groups based on their characteristics. Moving averages reflect the direction of stock prices, overbought/oversold indicators detect significant changes in stock volume, and volatility indicators gauge the stability of stocks by measuring price fluctuations. Note that for detailed information on technical indicators, refer to `https://www.investopedia.com`.

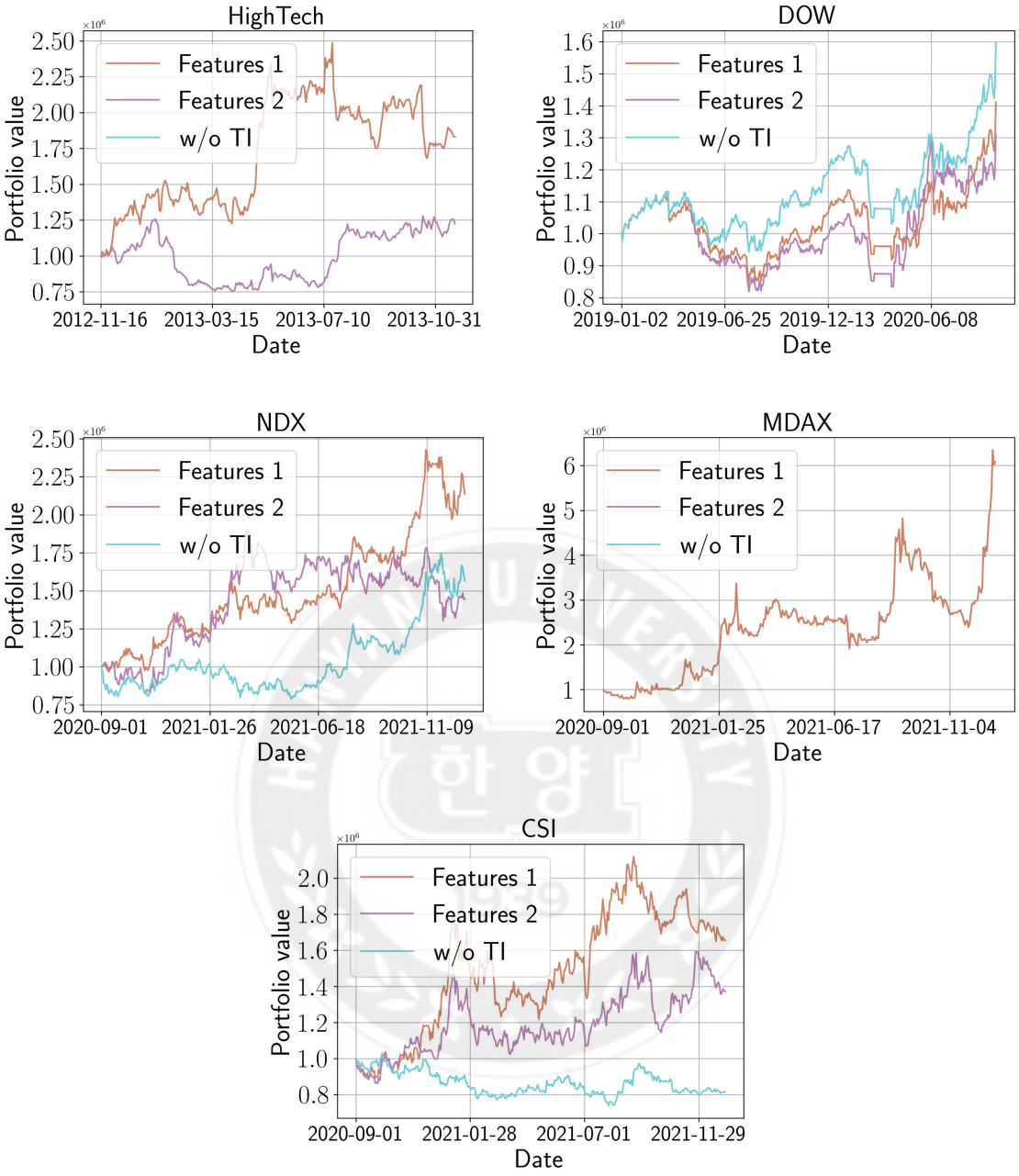

Figure B.1 Comparison of different sets of technical indicators for the HighTech, DOW, NDX, MDAX, and CSI datasets (see Features 1, Features 2, w/o TI in Table A.2).

B.0.1 Results

In the first additional experiment, the results in Figure A.1 demonstrate the average returns and actions assigned to each stock during the entire trading period. The graphs indicate that TAC concentrates the majority of actions on specific stocks rather than distributing them evenly across multiple stocks. For most

datasets, except CSI, TAC assigns actions to stocks that exhibit high average returns. Notably, the model is not provided with information on cumulative or average returns during training, yet it effectively identifies stocks with high growth rates.

Table A.2 presents detailed states of Features 1, Features 2, and the state without technical indicators (w/o TI). The technical indicators are calculated based on the closing price, and "_20, 30, 60" denotes the frequency of computing the indicators every 20, 30, and 60 days. In addition, "_LB" and "_UB" denote the lower and upper bounds, respectively. Note that we use Features 1 in the main experiment discussed in Section "Experiment," while Features 2 contains extra technical indicators for additional experiments.

In the second additional experiment, the performance of TAC with respect to different sets of technical indicators is analyzed. Figure B.1 and Table B.1 show the results obtained when training a model with Features 1 and Features 2 on various datasets. It is observed that most datasets perform well with Features 1, which provides meaningful information that can help the agent identify patterns in stock prices. However, for the DOW dataset, the model trained without any technical indicators performs the best, while using Features 2 leads to poor results in most datasets. This suggests that the choice of technical indicators can significantly impact the performance of TAC. Some technical indicators may even interfere with the analysis of stock price patterns. Hence, selecting appropriate technical indicators is crucial to improve the performance of TAC.

국문요지

Nam Yeong Lee
Department of Artificial Intelligence,
Graduate School of HANYANG UNIVERSITY
Directed by Professor Jun Moon

최근에 금융 주식 시장 투자에 대한 관심이 증가하면서 강화학습, LSTM, 트랜스포머 등의 알고리즘을 이용한 머신러닝 기법으로 주식을 자동으로 거래하거나 다음 주식 가격을 예측하는 알고리즘들이 개발되고 있다. 그중에 강화학습은 최적의 일련의 액션을 통해 포트폴리오 자산을 관리하는 데에 적용되고 있다. 주식을 투자하는데 가장 중요한 요소는 과거 주식 가격 데이터 활용이다. 그러나, 기존의 주식 도메인에 적용된 강화학습 알고리즘들은 마르코프 성질을 기반으로 형성되었기 때문에 최적의 액션을 취할 때 과거 주식 데이터를 고려하지 않는다. 이러한 한계를 해결하기 위해 우리는 과거 MDP 요소들을 attention 메커니즘을 통해 모델을 학습할 수 있는 decision transformer를 이용하여 Transformer-Actor-Crtic (TAC)를 제안한다. 추가적으로, critic 네트워크를 추가하고 액션을 평가하여 파라미터들을 업데이트하므로써 성능을 향상시킨다. 효율적인 학습 방법으로 우리는 강화학습 모델을 suboptimal 트레젝토리들을 통해 오프라인으로 모델을 학습시킨다. 액션의 가치를 과추정하는 것을 방지하기 위해 우리는 TAC를 behavior cloning term이 추가된 regularization 방법으로 학습시킨다. 다양한 주식 시장 데이터셋들을 사용하여 실험했을 때 TAC가 다른 최신 방법들보다 샤프지수와 수익의 결과가 더 좋다.

Declaration of Ethical Conduct in Research

I, as a graduate student of Hanyang University, hereby declare that I have abided by the following Code of Research Ethics while writing this dissertation thesis, during my degree program.

"First, I have strived to be honest in my conduct, to produce valid and reliable research conforming with the guidance of my thesis supervisor, and I affirm that my thesis contains honest, fair and reasonable conclusions based on my own careful research under the guidance of my thesis supervisor.

Second, I have not committed any acts that may discredit or damage the credibility of my research. These include, but are not limited to : falsification, distortion of research findings or plagiarism.

Third, I need to go through with Copykiller Program(Internet-based Plagiarism-prevention service) before submitting a thesis."

MAY 17, 2023

Degree : Master

Department : DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Thesis Supervisor : Jun Moon

Name : LEE NAM YEONG (Signature)

연구 윤리 서약서

본인은 한양대학교 대학원생으로서 이 학위논문 작성 과정에서 다음과 같이 연구 윤리의 기본 원칙을 준수하였음을 서약합니다.

첫째, 지도교수의 지도를 받아 정직하고 엄정한 연구를 수행하여 학위논문을 작성한다.

둘째, 논문 작성시 위조, 변조, 표절 등 학문적 진실성을 훼손하는 어떤 연구 부정행위도 하지 않는다.

셋째, 논문 작성시 논문유사도 검증시스템 "카피킬러"등을 거쳐야 한다.

2023년05월17일

학위명 : 석사

학과 : 인공지능학과

지도교수 : 문준

성명 : 이남영 (서명)

한 양 대 학 교 대 학 원 장 귀 하