A Smart Trader for Portfolio Management based on Normalizing Flows

Mengyuan Yang[1], **Xiaolin Zheng**[1], **Qianqiao Liang**[1], **Bing Han**[2], **Mengying Zhu**[1*]
[1]Zhejiang University, Hangzhou, China
[2]MYbank, Ant Group, Hangzhou, China
{yangmy412, xlzheng, liangqq, mengyingzhu}@zju.edu.cn, hanbing.hanbing@antgroup.com

Abstract

In this paper, we study a new kind of portfolio problem, named trading point aware portfolio optimization (TPPO), which aims to obtain excess intraday profit by deciding the portfolio weights and their trading points simultaneously based on microscopic information. However, a strategy for the TPPO problem faces two challenging problems, i.e., *modeling the ever-changing and irregular microscopic stock price time series* and *deciding the scattering candidate trading points*. To address these problems, we propose a novel TPPO strategy named STrader based on normalizing flows. STrader is not only promising in reversibly transforming the geometric Brownian motion process to the unobservable and complicated stochastic process of the microscopic stock price time series for modeling such series, but also has the ability to earn excess intraday profit by capturing the appropriate trading points of the portfolio. Extensive experiments conducted on three public datasets demonstrate STrader's superiority over the state-of-the-art portfolio strategies.

1 Introduction

In financial markets, a stock is often represented as time series consisting of daily open, high, low, and close prices (OHLC) (*macroscopic price time series*), which is essentially an aggregation of the minute-level OHLC price time series (*microscopic price time series*). In actual trading practice, such microscopic information is valuable for a portfolio strategy because it provides feasibility to capture trading opportunities, i.e., the near-optimal trading points (TPs), for each stock. Note that a trading point refers to a tradable minute. However, most portfolio strategies [Jiang *et al.*, 2017; Xu *et al.*, 2020; Zhang *et al.*, 2022] pay no attention to such advantage of exploiting microscopic information in deciding the TPs of the daily portfolio, leading to their inabilities in obtaining excess intraday returns. In this paper, we propose a novel problem named *trading points aware portfolio optimization* (TPPO) and aim to design an effective portfolio strategy for this problem to improve the investment's performance.

A TPPO strategy aims to obtain excess intraday profit by deciding the portfolio and its profitable TPs on the target trading day. The decision of TPs relates to the considerations from two aspects, i.e., the trading direction of whether to buy or sell and the minute-level buying and selling points, for each stock. Although the first aspect has been extensively studied [Ding *et al.*, 2020; Shi *et al.*, 2019], the second aspect remains very challenging, which is determined by the microscopic intraday price time series. To be specific, the buying (selling) points of a stock within a day are the microscopic minutes when the stock approximates its lowest (highest) price, which is very difficult to be captured.

Motivating Example. To better investigate the characteristics of TPs, in Figure 1, we plot the microscopic intraday price time series of two stocks in the China market, i.e., Shanghai International Airport Co., Ltd. and China Shenhua Energy Co., Ltd., and their price distributions within each TP. We can make two observations. Firstly, the near-optimal TPs, i.e., appropriate buying or selling points of a stock, are not continuous and may be scattered as depicted in Figure 1(a). In actual practice, there are possibilities that a trading order fails due to accidents such as preemption of transactions or communication failures. This observation indicates that a practical strategy is supposed to find a set containing multiple candidate TPs of each stock. Secondly, as depicted in Figure 1(b), the mean of the minute-level price relative to the daily opening price is constantly shifting and the variance is constantly increasing with the time farther from the daily opening time. Based on this observation, the microscopic price time series is very likely to be governed by a continuous-time process with shifting mean and increasing variance.

Motivated by the above observations, a practical way for portfolio construction with microscopic information is to first model the continuous-time microscopic price time series and then obtain candidate TPs sets based on the modeled series. However, such an idea faces two challenging problems corresponding to the two steps. Firstly, *how to model microscopic price time series?* (**CH1**) The distribution that governs the stock microscopic price time series is unobservable, complicated, and ever-changing. What's worst, the discontinuity and irregularity of the price time series hinder the generation of a continuous-time series. Therefore, it is challenging to en-

[*]Corresponding Author

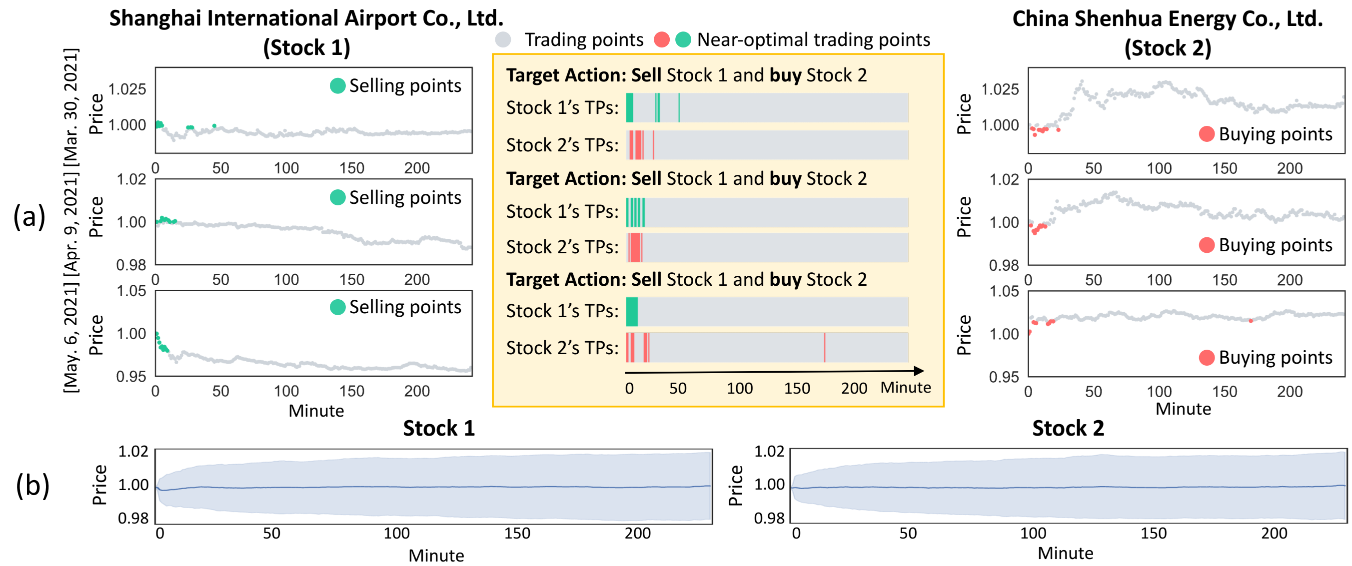

Figure 1: A motivation example that contains (a) the microscopic price time series and the candidate TPs sets, and (b) the price distributions, i.e., mean and variance, of each TP, of two China stocks. Note that each price is normalized by daily open price.

code the series' underlying dynamics and perform continuous extrapolation at arbitrary timestamps. Secondly, *how to obtain candidate TPs from the microscopic price time series and construct the portfolio?* (**CH2**) A straightforward idea is to first predict the optimal TP then choose the points close to the optimal point as candidate TPs. However, as illustrated in Figure 1(a), the sub-optimal TPs may not be close, i.e., continuous, to the optimal TP, which makes this naive solution ineffective. Instead, it is necessary to design a ranking mechanism for the construction of candidate TPs sets.

In order to address the above two challenging problems, we propose a novel TPPO strategy, named *STrader*, derived from normalizing flows, which inherits the superiority of normalizing flows in modeling complex distributions. Overall, our STrader is within the reinforcement learning (RL) framework to model a sequential decision-making process with a state observation process and an action generation process. In the state observation process, we propose a *Micro-Encoder* and a *Micro-Decoder* for each stock. Micro-Encoder utilizes our proposed reversible stochastic process flows (SPF) architecture to model a latent geometric Brownian motion (GBM) process, a stochastic process widely used in financial tasks [Yor, 2001; Oksendal, 2013], which governs the latent variables corresponding to the microscopic price time series. Micro-Decoder utilizes the reverse of SPF and the GBM process based on a learnable stochastic differential equation to model the unobservable, complicated, and ever-changing distribution of the price time series and ensure the continuity of the modeled series (*for addressing CH1*). In the action generation process, our proposed portfolio decision making module computes the portfolio weight and candidate TPs of each stock based on a ranking mechanism, which ranks each TP according to its possibility of being the optimal TP (*for addressing CH2*). In addition, we design a brand new reward function with TPs for the RL optimization.

Our main contributions are as follows: (1) *Problem:* We propose an innovative portfolio problem named TPPO, which focuses on deciding both the portfolio weights and trading points for excess intraday returns; (2) *Model:* We propose a novel TPPO strategy named STrader, which models the microscopic price time series based on normalizing flows and decides the portfolio weights and trading points within the RL framework to address the two challenges. To the best of our knowledge, this is the first attempt to incorporate normalizing flows into stock modeling tasks; (3) *Experiment:* We have conducted extensive experiments on three real-world datasets, which demonstrate the superiority of STrader over the state-of-the-art portfolio strategies.

2 Related Work

2.1 Portfolio Management

Portfolio management has attracted extensive research focus from the AI community [Li and Hoi, 2014], and it can be categorized into three kinds. The first kind utilizes macroscopic price time series intuitively [Jiang *et al.*, 2017; Xu *et al.*, 2020; Ye *et al.*, 2020; Zhang *et al.*, 2022] without considering the fine-grained microscopic information. The second category exploits both macroscopic and microscopic price time series [Shi *et al.*, 2019; Liu *et al.*, 2020; Ding *et al.*, 2020] to enhance the representation ability, which, however, only focuses on daily portfolio weight allocation and pays no attention to earning the excess intraday profit. The third category incorporates side information, such as market information [Wang *et al.*, 2021], news [Ye *et al.*, 2020] and factor indicators [Imajo *et al.*, 2021] to assist the characterization of sequential features. These strategies need additional information, which is not the focus of our study, and we will study this in our future work.

2.2 Deep Generative Time Series Modeling

The deep generative model has been proven to be effective for time series modeling [Fraccaro *et al.*, 2016; Luo *et al.*, 2018] such as financial series. It can be categorized into three kinds. The first kind is based on variational inference [Luo *et al.*, 2018], which focuses on modeling the distribution of discrete time series without considering the underlying continuous dynamics. The second kind is differential equations based models, which incorporate neural ordinary differential equations [Chen *et al.*, 2018; Rubanova *et al.*, 2019] or latent stochastic process [Li *et al.*, 2020] in the time series modeling framework to ensure the continuity of latent trajectory. However, this kind of method can only handle relatively simple distributions. The third kind is reversible generative models [Rezende and Mohamed, 2015; Kingma and Dhariwal, 2018], which uses normalizing flows to model a base distribution within each discrete inter-arrival time point and shows promise at capturing complex distributions. In conclusion, none of these three kinds of studies can handle the complexity of distribution and the continuity of time series simultaneously, which hinders their application in modeling stock price time series.

3 Preliminaries and Problem Setting

3.1 Preliminaries

Our portfolio strategy is built upon the GBM process, a continuous-time stochastic process for modeling stock prices in most financial models [Yor, 2001; Marathe and Ryan, 2005]. The generation of GBM relates to the Wiener process and stochastic differential equation (SDE), which we will introduce as follows.

Definition 1 (Wiener Process) *A D-dimensional Wiener process $\boldsymbol{W}$ has three properties: (1) $\boldsymbol{W}_0 = 0$; (2) $\boldsymbol{W}_{t_2} - \boldsymbol{W}_{t_1} \sim \mathcal{N}(0, (t_2 - t_1)\boldsymbol{I}_D)$ for $t_1 \leq t_2$, and $\boldsymbol{W}_{t_2} - \boldsymbol{W}_{t_1}$ is independent of past values of $\boldsymbol{W}_{t'}$ for all $t' \leq t_1$; (3) when $\boldsymbol{W}_{t_1} = \boldsymbol{w}_{t_1}$ and $\boldsymbol{W}_{t_2} = \boldsymbol{w}_{t_2}$, the conditional distribution for $t_1 \leq t \leq t_2$ is $p_{\boldsymbol{W}_t|\boldsymbol{W}_{t_1},\boldsymbol{W}_{t_2}}(\boldsymbol{w}_t \mid \boldsymbol{w}_{t_1}, \boldsymbol{w}_{t_2}) = \mathcal{N}\left(\boldsymbol{w}_t; \boldsymbol{w}_{t_1} + \frac{t-t_1}{t_2-t_1}(\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}), \frac{(t_2-t)(t-t_1)}{t_2-t_1}\boldsymbol{I}_d\right)$.*

Definition 2 (SDE) *An SDE describing the stochastic dynamics of $\boldsymbol{Z}$ usually takes the form $\mathrm{d}\boldsymbol{Z}_t = \boldsymbol{\mu}(\boldsymbol{Z}_t, t)\,\mathrm{d}t + \boldsymbol{\sigma}(\boldsymbol{Z}_t, t)\,\mathrm{d}\boldsymbol{W}_t$, where $\boldsymbol{Z}$ is a variable which continuously evolves with time, $\boldsymbol{\mu}(\cdot)$ is a drift function, $\boldsymbol{\sigma}(\cdot)$ is a diffusion function, and $\boldsymbol{W}_t$ is a Wiener process. For each sample trajectory $\boldsymbol{\omega} \sim \boldsymbol{W}_t$, the stochastic process $\boldsymbol{Z}_t$ maps $\boldsymbol{\omega}$ to a different trajectory $\boldsymbol{Z}_t(\boldsymbol{\omega}) = \boldsymbol{Z}_0 + \int_0^t \mathrm{d}\boldsymbol{Z}_s$ with initial $\boldsymbol{Z}_0$.*

Definition 3 (GBM) *Stochastic differential equation (SDE) is often used to model the continuous-time stochastic GBM process because of its ability to add instantaneous noise to the dynamics. The GBM SDE takes the form $\mathrm{d}\boldsymbol{Z}_t = \mu \boldsymbol{Z}_t \mathrm{d}t + \sigma \boldsymbol{Z}_t \mathrm{d}\boldsymbol{W}_t$, where μ and σ are the drift and variance terms.*

3.2 TPPO Problem Setting

We consider TPPO with N stocks during T trading days, and each day has M minute-level trading points. Each stock has $D = 4$ kinds of prices, i.e., open, high, low, and close prices.

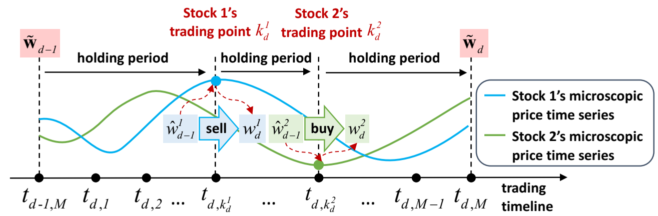

Figure 2: TPPO problem setting.

We denote the microscopic intraday prices of stock i in the m-th minute on trading day d as $\mathbf{x}^i_{d,m} \in \mathbb{R}^D$ and denote the prices of all assets on trading day d as $\mathbf{x}_d \in \mathbb{R}^{N\times M\times D}$. We denote the stocks' microscopic price time series during the previous τ days as $\mathbf{x}_{d-\tau:d-1} = \{\mathbf{x}_{d-\tau}, ..., \mathbf{x}_{d-1}\}$. For reward calculation, we use the minute-level closing prices, which are defined as $p^i_{d,m} \in \mathbb{R}$ and $\mathbf{p}_d \in \mathbb{R}^{N\times M}$ similarly.

The TPPO process in each trading day d is described in Figure 2, which is formulated within RL framework because of its superior nature in modeling sequential decision making process without annotated data. An Markov Decision Process (MDP) for the RL problem can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ denotes a finite state space, $\mathcal{A}$ denotes a finite set of actions, $\mathcal{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is a state transition function that defines the next state $\mathbf{s}'$ given the current state $\mathbf{s}$ and action $\mathbf{a}$, and $\mathcal{R}(\mathbf{s}, \mathbf{a})$ is a reward function. Moreover, a policy $\pi(\mathbf{a}|\mathbf{s})$ determines an action $\mathbf{a}$ given the current state $\mathbf{s}$.

Definition 4 (TPPO formulated as MDP) *In each trading day d, a TPPO strategy observes the state $\mathbf{s}_d$ constructed based on $\mathbf{x}_{d-\tau:d-1}$. Based on the state, the strategy determines a portfolio decision $\pi_d = (\mathbf{w}_d, \mathbf{k}_d)$ as an action, where $\mathbf{w}_d = (w^0_d, w^1_d, w^2_d, ..., w^N_d)$ is a weight vector with cash weight w^0_d and $\mathbf{k}_d = (k^1_d, k^2_d, ..., k^N_d)$ is its corresponding trading point for each stock satisfying the constraints that $\mathbf{w} \in \Delta_N = \{\mathbf{w}|0 \leq \mathbf{w} \leq 1, \mathbf{w}^\top \mathbf{1} = 1\}$ and $\mathbf{k}_d \in [1, M]^N$. Then, the strategy receives a reward $r_d \in \mathcal{R}$, which is used for portfolio optimization on the next target trading day, and the next state is reached based on $\mathbf{s}_d \sim \mathcal{T}(\mathbf{s}_{d-1}, \mathbf{a}_{d-1})$.*

Definition 5 (TPPO reward function) *A portfolio decision $\pi_d = (\mathbf{w}_d, \mathbf{k}_d)$ in trading day d can be evaluated as*

$$r_d = ((\underbrace{\log(\tilde{\mathbf{w}}^\top_{d-1}\frac{\mathbf{p}_{d,k_d}}{\mathbf{p}_{d-1,M}})}_{\textit{return before trading}} + \underbrace{\log(\mathbf{p}^\top \frac{\mathbf{p}_{d,M}}{\mathbf{p}_{d,k_d}})}_{\textit{return after trading}})(1 - c_t) - \lambda \underbrace{\|\hat{\mathbf{w}}_{d-1} - \mathbf{w}\|_1}_{\textit{transaction cost penalty}}), \quad (1)$$

where c_t is the transaction cost rate, $\lambda \geq 0$ is a trade-off hyperparameter of transaction cost, $\tilde{\mathbf{w}}_{d-1} = \frac{\mathbf{w}_{d-1} \circ \mathbf{r}^{after}_{d-1}}{\mathbf{w}^\top_{d-1}\mathbf{r}^{after}_{d-1}}$ and $\hat{\mathbf{w}}_{d-1} = \frac{\tilde{\mathbf{w}}_{d-1} \circ \mathbf{r}^{before}_{d}}{\tilde{\mathbf{w}}^\top_{d-1}\mathbf{r}^{before}_{d}}$ are the normalized portfolio weights as depicted in Figure 2, and the symbol $\circ$ denotes the Hadamard product. Note that $\mathbf{r}^{before}_d = \frac{\mathbf{p}_{d,k_d}}{\mathbf{p}_{d-1,M}}$ and $\mathbf{r}^{after}_{d-1} = \frac{\mathbf{p}_{d-1,M}}{\mathbf{p}_{d-1,k_d}}$ are the price relative vector before and after trading. According to [Zhang et al.*, 2022], the transaction cost penalty in Eq.(1) can be used to control the transaction costs.*

In this paper, we aim to devise a TPPO strategy to maximize the cumulative reward. Besides, we follow some widely adopted assumptions in portfolio management [Zhu *et al.*, 2020; Liang *et al.*, 2021; Zhang *et al.*, 2022], namely, no margin/short, unlimited market liquidity, and zero market impact. The main notations are summarized in Table 1.

Notation	Description
N	total number of stocks
T	total number of trading days
M	total number of trading points in each day
D	total number of prices types
$\mathbf{x}_{d,m}^i$	prices of stock i on the m-th trading point on trading day d
$\mathbf{x}_d$	prices of all stocks on trading day d
$\mathbf{x}_{d-\tau:d-1}$	prices series during days $d-\tau$ to day $d-1$
$\hat{\mathbf{x}}$	predictive prices time series of all assets
$\mathbf{p}_d$	closing prices of all assets on trading day d
$t_{d,m}$	time index of the m-th trading point on trading day d
$\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}$	state, action, transition, and reward space
$\mathbf{w}_d$	portfolio weight vector on trading day d
$\mathbf{k}_d$	portfolio trading point vector on trading day d
π_d	portfolio strategy $(\mathbf{w}_d, \mathbf{k}_d)$ on trading day d
$\mathbf{r}_d^i$	correlation representation for stock i on day d
r_d	reward of $\pi_d = (\mathbf{w}_d, \mathbf{k}_d)$ on trading day d
$\tilde{\mathbf{w}}_d, \hat{\mathbf{w}}_d$	normalized weight vectors on trading day d
$\mathbf{W}, \mathbf{X}$	Wienie process and observation process
$\mathbf{z}, \mathbf{Z}$	latent variable and latent process

Table 1: Notations in the paper.

4 Methodology

4.1 Overall

Our proposed STrader consists of three key modules, i.e., *Microscopic Price Time Series Encoder* (Micro-Encoder), *Microscopic Price Time Series Decoder* (Micro-Decoder), and *Portfolio Decision Making Module* (PDM). STrader's architecture is shown in Figure 3. Specifically, each stock has one Micro-Encoder and one Micro-Decoder. A Micro-Encoder learns a reversible transformation function between the stock price's stochastic process and its latent GBM process. A Micro-Decoder utilizes the inverse of its process transformation function to generate the stock's microscopic price time series in the target trading day d. Based on the generated microscopic price time series, PDM computes the portfolio weight and candidate TPs of each stock. We elaborate the three modules on each day d as follows.

4.2 Microscopic Price Time Series Encoder (Micro-Encoder)

Inherently, an observed microscopic stock price time series is governed by a stochastic process (*data-level process*), which is, however, unobservable and too complicated to learn. This hinders the modeling of the microscopic series. Inspired by normalizing flows [Rezende and Mohamed, 2015], which is a technique to reversibly transform a simple base distribution into a complex target distribution, STrader aims to find a function for transforming stochastic process in each Micro-Encoder. Specifically, STrader regards the GBM process as a base process and learns a reversible process transformation function that transforms the distributions between the data-level process and the GBM process based on our proposed stochastic process flows, a novel architecture derived from continuous normalizing flows [Grathwohl *et al.*, 2018]. The advantage of such an idea is that the data-level process can inherit many of the appealing properties of the GBM process.

In each Micro-Encoder of stock i, the microscopic price time series $\mathbf{x}_{d-\tau:d}^i$ is an incomplete realization of the data-level process $\{\mathbf{X}_t\}_{t\in[t_{d-\tau,1}, t_{d,M}]}$. For simplicity, we ignore the superscript i of the notations for stock i and formulate $[t_{d-\tau,1}, t_{d,M}]$ into $[t_0, ..., t_\beta, t_{\beta+1}, ..., t_{\beta+M}]$, where $t_{\beta+1}$ and $t_{\beta+M}$ are the first and last trading points on the target trading day d. The reversible transformation function $F_{\boldsymbol{\theta}}(\mathbf{Z}_t; t)$ is parametrized by the learnable parameters $\boldsymbol{\theta}$ for every trading point t, where $\mathbf{Z}_t$ is a random variable of the GBM process at time t. Our goal is to model $\{\mathbf{X}_t = F_{\boldsymbol{\theta}}(\mathbf{Z}_t; t)\}_{t\in[t_0, t_{\beta+M}]}$ such that the log-likelihood of the observations $\mathcal{L}_{\text{joint}}^d = \log p_{\mathbf{X}_0,...,\mathbf{X}_{\beta+M}}(\mathbf{x}_0, ..., \mathbf{x}_{\beta+M})$ is maximized. Such $F_{\boldsymbol{\theta}}(\mathbf{Z}_t; t)$ is approximated as follows.

Stochastic Process Flows Architecture. We define the mapping from observed prices to latent variables using neural ODE based on the instantaneous change of variables formula of Theorem 1 in [Chen *et al.*, 2018]. Given a price vector $\mathbf{x}_t$, we compute the latent variable $\mathbf{z}_t$ that generates $\mathbf{x}_t$, as well as $\log p(\mathbf{x}_t)$ by solving the initial value problem in Eq. (2):

$$\underbrace{\begin{bmatrix} \mathbf{z}_t \\ \log p(\mathbf{x}_t) - \log p_{z_t^{\kappa_0}}(\mathbf{z}_t) \end{bmatrix}}_{\text{solutions}} = \underbrace{\begin{bmatrix} \mathbf{x}_t \\ 0 \end{bmatrix}}_{\text{initial values}} + \underbrace{\int_{\kappa_1}^{\kappa_0} \begin{bmatrix} f_{\boldsymbol{\theta}}(\mathbf{z}_t^\kappa, a_t^\kappa, \kappa) \\ \mathrm{Tr}(\frac{\partial f}{\partial \mathbf{z}_t^\kappa}) \end{bmatrix} d\kappa}_{\text{dynamics}}, \quad (2)$$

where $z_t^{\kappa_0}$ is the base distribution governed by the latent GBM process in t, a_t^κ is a series of time feature vector elaborated in Section 4.3, and f is a Lipschitz continuous function to learn the parameter $\boldsymbol{\theta}$ of $F_{\boldsymbol{\theta}}$. Thus we can compute $\log p(\mathbf{x}_t)$ by adding the solutions with $logp_{z_t^{\kappa_0}}(\mathbf{z}_t)$. Furthermore, we use the unbiased estimate of the log-density introduced in [Grathwohl *et al.*, 2018] to accelerate the solving process.

Training Goal. The training goal of each Micro-Encoder i in each trading day d is to maximize Eq. (3), which measures the model's ability to fit the microscopic price time series.

$$\begin{aligned}
\mathcal{L}_{\text{joint}}^{d,i} &= \log p_{\mathbf{X}_0,...,\mathbf{X}_{\beta+M}}(\mathbf{x}_0, ..., \mathbf{x}_{\beta+M}) = \sum_{t=1}^{\beta+M} \log p_{\mathbf{X}_t|\mathbf{X}_{t-1}}(\mathbf{x}_t|\mathbf{x}_{t-1})) \\
&= \sum_{t=1}^{\beta+M} [\log p_{\mathbf{Z}_t|\mathbf{Z}_{t-1}}(\mathbf{z}_t|\mathbf{z}_{t-1})) - \log|\det \frac{\partial F_{\boldsymbol{\theta}}}{\partial \mathbf{z}_t}|] \\
&= \sum_{t=1}^{\beta+M} [\log p_{\mathbf{Z}_t|\mathbf{Z}_{t-1}}(\mathbf{z}_t|\mathbf{z}_{t-1})) - \int_{\kappa_0}^{\kappa_1} \mathrm{Tr}(\frac{\partial f_{\boldsymbol{\theta}}}{\partial \mathbf{z}_t^\kappa}) d\kappa].
\end{aligned} \quad (3)$$

4.3 Microscopic Price Time Series Decoder (Micro-Decoder)

In order to generate the microscopic price time series governed by the data-level process, which inherits the properties

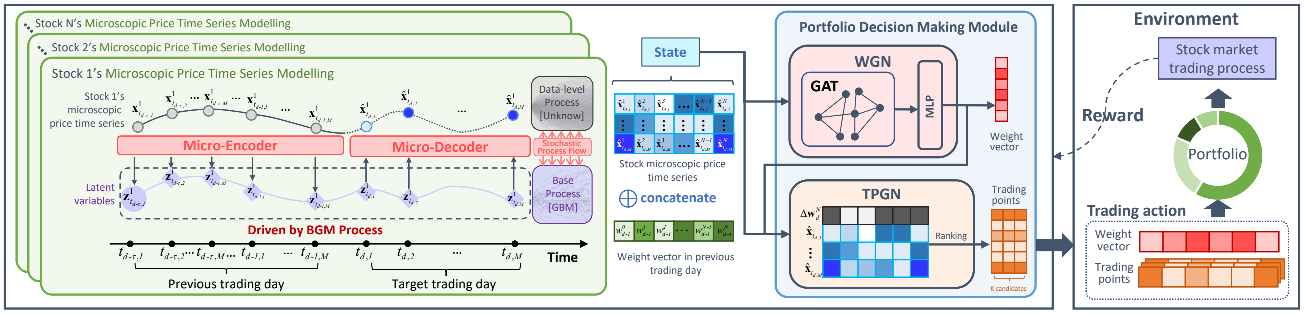

Figure 3: STrader's framework. The left green part is the state observation process, and the right blue part is the action generation process.

of the GBM process, we can follow three steps: (1) constructs the GBM SDE; (2) solves the GBM SDE to sample multiple trajectories; (3) transforms the trajectories to the stock price time series. The generated series covers not only the previous trading days (*interpolation*) for handling the irregularity of the time series but also the target trading day (*extrapolation*) for portfolio strategy construction.

Firstly, the GBM SDE is constructed based on

$$\boldsymbol{Z}_t = \boldsymbol{Z}_0 + \int_0^t (\mu \boldsymbol{Z}_s ds + \sigma \boldsymbol{Z}_s dW_s), \tag{4}$$

where μ and σ are learnable parameters.

Secondly, to sample one trajectory, STrader samples $\mathbf{z}_0 \sim \mathcal{N}(0,1)$ as the sample at the first trading point t_0, i.e., the initial value in Eq. (4), then solves the GBM SDE to obtain the samples on all subsequent trading points.

Thirdly, STrader transforms the samples on each trajectory to those on the stock price time series by solving Eq. (5) based on the learned θ of $F_{\boldsymbol{\theta}}$.

$$\underbrace{\begin{bmatrix}\hat{\mathbf{x}}_t \\ a_t^{\kappa_1}\end{bmatrix}}_{\text{solutions}} = \underbrace{\begin{bmatrix}\mathbf{z}_t \\ t\end{bmatrix}}_{\text{initial values}} + \underbrace{\int_{\kappa_0}^{\kappa_1} \begin{bmatrix} f_{\boldsymbol{\theta}}(\mathbf{z}_t^{\kappa}, a_t^{\kappa}, \kappa) \\ g_{\boldsymbol{\phi}}(a_t^{\kappa}, \kappa)\end{bmatrix} d\kappa}_{\text{dynamics}}, \tag{5}$$

where g is a time encoding function, $a_t^{\kappa_1}$ is used in Eq. (2), and $\hat{\mathbf{x}}_t$ is the transformed sample at time t.

Multiple generated stock price time series are averaged as the final generated series of one stock.

Training Goal. In each trading day d, the training goal of each Micro-Decoder i aims to minimize $\mathcal{L}_{\text{MSE}}^{d,i} = \sum_{t=\beta+1}^{\beta+M} (\hat{\mathbf{x}}_t - \mathbf{x}_t)^2 / M$.

4.4 Portfolio Decision Making Module (PDM)

PDM consists of two networks, i.e., *weight generating network* (WGN) and *trading points generating network* (TPGN). WGN outputs the target portfolio weight and the portfolio direction. TPGM outputs the candidate TPs of each stock.

Weight Generating Network (WGN). To better guide the portfolio decision making, STrader models the interrelations between stocks by the graph attention network (GAT) [Veličković *et al.*, 2017]. Note that the adjacency matrix in GAT can be characterized by any structural information between stocks. In this paper, we apply the covariance matrix of the price series as the adjacency matrix. Therefore, we feed the generated series and the adjacency matrix into GAT to obtain each stock i's representation $\mathbf{r}_d^i$. Finally, WGN feeds the generated microscopic price time series, stock correlation representations, and previous weight vector into an MLP followed by a *softmax* activation function to get portfolio weight vector $\mathbf{w}_d$ for target trading day d. The portfolio direction vector is $\Delta \mathbf{w}_d = \mathbf{w}_d - \tilde{\mathbf{w}}_{d-1}$.

Trading Points Generating Network (TPGN). Because we need to find a group of TPs for each stock, we design a ranking method to obtain the ranking score of each TP. Considering that the TPs of each stock are related to its trading direction, we feed both the $\Delta \mathbf{w}_d$ and the microscopic price time series to an MLP for score calculation. Then we construct the candidate set with the top K scoring TPs for each stock. During each trading day, we try to trade each stock with its weight as long as the instantaneous time matches those in its candidate TPs set, and if the trading fails, we will not trade until the next candidate TP arrives. The additional money that has not been traded successfully will be held as cash.

Training Goal. PDM's training goal is to minimize the ranking loss in Eq.(6) and maximize portfolio's reward r_d in Definition 5.

$$P_s = \prod_{j=1}^{K} \frac{\exp(\text{score}_{s,j})}{\sum_{l=j}^{M} \exp(\text{score}_{s,l})}, \quad \mathcal{L}_{\text{rank}}^{d} = -\sum_{s=1}^{N} \bar{P}_s \log P_s, \tag{6}$$

where $\text{score}_{s,j}$ is the score of the j-th predicted TP of the stock s, $\bar{P}$ and P are predicted and ground-truth probabilities.

In addition, we adopt a classical policy gradient algorithm [Moody and Saffell, 2001] in RL to train the policy network by maximizing the portfolio reward r_d in Eq.(1) earned by adjusting portfolio weights while controlling transaction costs.

4.5 Learning and Optimization of STrader

STrader's learning contains the goals from four aspects, i.e., fitting the distribution of microscopic price time series, generating microscopic price time series, capturing TPs, and earning wealth. We apply a dynamic weight average method [Liu *et al.*, 2019] for overall optimization with the four aspects. Therefore, the total loss function is given as:

$$\mathcal{L} = \frac{1}{T} \sum_{d=1}^{T} \left(-\zeta_1 * \sum_{i=1}^{N} \mathcal{L}_{\text{joint}}^{d,i} + \zeta_2 * \sum_{i=1}^{N} \mathcal{L}_{\text{MSE}}^{d,i} + \zeta_3 * r_d - \zeta_4 * \mathcal{L}_{\text{rank}}^{d}\right), \tag{7}$$

where $\boldsymbol{\zeta}$ is a learnable weight vector of the four terms.

Dataset	Training Data Range	Validation Data Range	Test Data Range
DJIA	01/02/2020 - 03/31/2020	04/01/2020 - 04/30/2020	05/01/2020 - 07/31/2020
SSE	01/02/2020 - 11/30/2020	12/01/2020 - 12/31/2020	01/01/2021 - 06/30/2021
COIN	01/02/2020 - 11/30/2020	12/01/2020 - 12/31/2020	01/01/2021 - 06/30/2021

Table 2: Datasets descriptions

5 Experiments

We conduct extensive experiments to answer the following questions: **Q1:** How does STrader perform on profitability over the trading rounds? **Q2:** How does STrader perform on reducing risk? **Q3:** How does STrader perform in addressing the **CH1** and **CH2**? **Q4:** Whether STrader finds the appropriate TPs for different stocks in a portfolio?

5.1 Experimental Settings

Data Collection. We have collected three up-to-date datasets. **DJIA** is a U.S. stock market dataset consisting 30 stocks from Dow Jones Industrial Average Index (DJIA). We collected daily and 1-minute intraday historical price data from Alpha Vantage[1]. **SSE** is a China stock market dataset covering 50 A-share stocks from the SSE 50 index. The historical daily and intraday price data were collected from iFinD[2]. **COIN** is a cryptocurrencies market dataset containing 12 different cryptocurrencies, each of which has daily and 1-minute intraday historical price data from coinbase[3]. All three datasets were divided into non-overlapping training/validation/test sets as described in Table 2. Results on all experiments are averaged over 10 runs.

Baselines. Six baselines classified into two groups and two ablation strategies are compared in our experiments:

(1) *Macro information based portfolio strategies*: **EIIE** [Jiang *et al.*, 2017] is a conventional RL portfolio strategy utilizing stock historical prices. **RAT** [Xu *et al.*, 2020] is a state-of-the-art RL strategy with attention on stock price series. **PPN** [Zhang *et al.*, 2022] is a state-of-the-art RL portfolio strategy focusing on reducing transaction cost.

(2) *Micro information based portfolio strategies*: **EI3** [Shi *et al.*, 2019] is a state-of-the-art portfolio strategy to integrate multi-scale price as states of RL. **MTDNN** [Liu *et al.*, 2020] is a state-of-the-art wavelet-based deep learning method classifying stock price movements, and we modify this method by trading the stocks where prices are expected to rise. **HMG-TF** [Ding *et al.*, 2020] is a state-of-the-art portfolio strategy to learn hierarchical features of high-frequency finance data.

(3) *Simplified versions of STrader*: **STrader-w/o-SPF** is STrader without the SPF architecture, which means that it regards the realization of the GBM process as the predicted stock price process. **STrader-w/o-TP** is STrader without considering TPs, which means that every stock is traded at the opening time of the target trading day.

Metrics. We use six standard metrics [Liang *et al.*, 2021] to measure portfolios' performance. Cumulative wealth (CW) and Annualized Percentage Yield (APY) measure the portfolios' returns. Maximum drawdown (MD) and Annualized Volatility (AVO) measure the portfolios' risks. Annualized Sharpe Ratio (ASR) and Calmar ratio (CR) measure the risk-adjusted returns. Overall, higher CW, APY, ASR, and CR values indicate better performance, while lower MD and AVO values indicate better performance. Note that if the APY is negative, the ASR and CR cannot be used as comparable metrics and thus are not reported in our following experiments.

Implement Details. To model real-world markets, we set the transaction cost rate to be 0.25% [Zhang *et al.*, 2022]. We set $\tau = 1$ as the time window size for STrader and $\tau = 5$ for other baselines that are not applicable to intraday trading. We set $b = 32$ as the batch size. We implement all experiments by using PyTorch and conduct the experiments on an NVIDIA RTX 3090 GPU. We adopt Adam optimizer [Kingma and Ba, 2014] and vary the learning rate of the network in $\{10^{-2}, 10^{-3}, 10^{-4}\}$ by using an exponential learning rate scheduler [Li and Arora, 2019] with a decay rate of 0.01. We vary the size of latent variable $h_z \in \{32, 64, 128\}$, We use two MLP model as $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\phi}}$ respectively. We vary the hidden layer in $f_{\boldsymbol{\theta}}$ network as $l_f \in \{2, 3, 4\}$ with the hidden size $h_f \in \{32, 64, 128\}$, and the hidden layer in $g_{\boldsymbol{\phi}}$ network as $l_g \in \{2, 3, 4\}$ with the hidden size $h_g \in \{32, 64, 128\}$. We vary the number of attention heads in GAT as $l_{\text{GAT}} \in \{2, 3, 4\}$, and the dimension of the feature space in GAT as $h_{\text{GAT}} \in \{32, 64, 128\}$. We vary the size of top-K in time-aware rank loss $K \in \{10, 20, 30\}$. Besides, we set the parameters of the baselines to be the optimal values reported in their publications because the optimal parameters of each baseline do not distinguish between datasets according to their publications. We tune parameters based on the validation dataset and evaluate performance on the test dataset.

5.2 Performance Comparison and Analysis

Result 1: Performance on Profitability. To answer question **Q1**, we compare the performance of STrader and the six baselines on CW and APY (cf. Table 3). STrader achieves the highest CW and APY values with average improvements of 7.541% and 72.529%, respectively, over the best-performing baselines. The results demonstrate the effectiveness of STrader from two aspects. First, STrader outperforms all macro information based strategies because it additionally models the microscopic price time series based on the novel stochastic process flow architecture to assist portfolio decisions. Second, STrader outperforms all micro information based strategies because they overlook excess intraday profit, whereas STrader leverages the microscopic information for appropriate TPs decisions and earns more profit.

Result 2: Performance on Risk. To answer **Q2**, we present the risk, i.e., MD and AVO, achieved by STrader and the six baselines in Table 3. STrader is among the best three out of all seven strategies. It is comparable to all baselines from the following three perspectives. First, STrader achieves the lowest MD, demonstrating that STrader can resist drawdown risks. Second, although STrader has a larger AVO than the best-performing baselines, such results are insignificant with their p-values > 0.05 on DJIA and COIN datasets, indicating that this not-so-good result is accidental. Such results can be tolerated for a risk-insensitive method. Third, high risk comes

[1]https://www.alphavantage.co/

[2]http://quantapi.10jqka.com.cn/

[3]https://developers.coinbase.com/api/

Categories	Strategies	DJIA CW	DJIA APY	DJIA ASR	SSE CW	SSE APY	SSE ASR	COIN CW	COIN APY	COIN ASR
Market	Index[4]	1.086	0.375	1.423	0.993	-0.015	-	7.523*	56.225*	30.546
Macro information based portfolio strategies	EIIE	1.070 ± 0.002	0.300 ± 0.007	1.253 ± 0.005	0.982 ± 0.001	-0.036 ± 0.001	-	5.753 ± 3.312	33.880 ± 48.302	21.613 ± 14.704
	RAT	1.062 ± 0.009	0.263 ± 0.042	1.122 ± 0.083	1.033 ± 0.076	0.071 ± 0.167	0.312 ± 0.733*	5.699 ± 0.718	10.146 ± 1.973	7.675 ± 1.064
	PPN	1.098 ± 0.026*	0.438 ± 0.128*	2.137 ± 0.379*	1.035 ± 0.045*	0.074 ± 0.097*	0.212 ± 0.171	7.295 ± 0.036	52.800 ± 0.535	33.409 ± 0.456*
Micro information based portfolio strategies	EI3	1.070 ± 0	0.299 ± 0	1.253 ± 0	0.982 ± 0	-0.037 ± 0	-	4.503 ± 8.010	29.311 ± 128.438	9.785 ± 25.710
	MTDNN	1.076 ± 0	0.327 ± 0	1.258 ± 0	0.981 ± 0	-0.039 ± 0	-	4.956 ± 0	23.784 ± 0	17.298 ± 0
	HMG-TF	1.094 ± 0.011	0.416 ± 0.053	1.543 ± 0.128	1.019 ± 0.011	0.040 ± 0.022	0.170 ± 0.095	6.261 ± 1.118	38.922 ± 13.234	27.418 ± 9.234
STrader (Our)	**STrader**	**1.117 ± 0.001**	**0.533 ± 0.006**	**2.411 ± 0.027**	**1.088 ± 0.005**	**0.191 ± 0.012**	**0.973 ± 0.065**	**8.711 ± 3.561**	**77.917 ± 73.771**	**55.796 ± 40.403**
	STrader-w/o-SPF	1.110 ± 0.002	0.498 ± 0.013	2.283 ± 0.059	1.008 ± 0.001	0.017 ± 0.002	0.095 ± 0.011	7.685 ± 1.345	59.127 ± 22.412	44.467 ± 13.393
	STrader-w/o-TP	1.073 ± 0.001	0.314 ± 0.006	1.250 ± 0.023	0.981 ± 0.001	-0.039 ± 0.001	-	4.503 ± 0.119	19.446 ± 1.073	15.306 ± 0.778
Improvement[5]		1.699	21.690	12.822	5.137	157.317	212.008	15.787	38.581	67.011
p-value[6]		0.004	0.004	0.002	0.000	0.000	0.000	0.024	0.034	0.001

Categories	Strategies	DJIA CR	DJIA MD	DJIA AVO	SSE CR	SSE MD	SSE AVO	COIN CR	COIN MD	COIN AVO
Market	Index[4]	4.040	0.093	0.263	-	0.156	0.206	84.771	0.663	1.841
Macro information based portfolio strategies	EIIE	3.545 ± 0.011	0.085 ± 0.002	0.239 ± 0.006	-	0.127 ± 0.002*	**0.177 ± 0.003***	54.836 ± 76.933	0.616 ± 0.013	1.502 ± 0.761
	RAT	3.133 ± 0.272	0.084 ± 0.007	0.234 ± 0.022	0.527 ± 1.144*	0.147 ± 0.050	0.225 ± 0.059	16.403 ± 3.183	0.618 ± 0.004	**1.319 ± 0.068***
	PPN	5.204 ± 1.376*	0.084 ± 0.003	**0.203 ± 0.027***	0.290 ± 0.217	0.245 ± 0.093	0.338 ± 0.148	98.254 ± 0.664*	0.537 ± 0.002*	1.580 ± 0.008
Micro information based portfolio strategies	EI3	3.547 ± 0	0.084 ± 0*	0.238 ± 0	-	0.128 ± 0	0.178 ± 0	45.005 ± 183.354	0.630 ± 0.100	3.166 ± 2.046
	MTDNN	3.564 ± 0	0.092 ± 0	0.260 ± 0	-	0.134 ± 0	0.188 ± 0	38.495 ± 0	0.618 ± 0	1.375 ± 0
	HMG-TF	4.935 ± 1.061	0.085 ± 0.01	0.270 ± 0.026	0.227 ± 0.127	0.176 ± 0.015	0.234 ± 0.015	63.666 ± 23.186	0.612 ± 0.028	1.418 ± 0.074
STrader (Our)	**STrader**	**6.576 ± 0.079**	0.081 ± 0.001	0.221 ± 0.001	**1.857 ± 0.148**	**0.103 ± 0.003**	0.196 ± 0.001	**150.563 ± 141.275**	**0.517 ± 0.014**	1.373 ± 0.204
	STrader-w/o-SPF	6.202 ± 0.168	**0.080 ± 0.001**	0.218 ± 0.001	0.144 ± 0.016	0.121 ± 0.001	0.184 ± 0.001	112.113 ± 41.560	0.527 ± 0.013	1.325 ± 0.084
	STrader-w/o-TP	3.538 ± 0.074	0.089 ± 0.001	0.251 ± 0.001	-	0.132 ± 0.001	0.184 ± 0.001	33.291 ± 1.848	0.584 ± 0.002	1.270 ± 0.006
Improvement[5]		26.367	3.335	-8.917	252.251	19.196	-10.516	53.239	3.759	-4.050
p-value[6]		0.000	0.000	0.474	0.000	0.000	0.000	0.000	0.000	0.100

Table 3: Results of all strategies on six profitable metrics (mean ± range, computed across 10 runs).

with high returns. If we take the return and risk into consideration simultaneously, i.e., estimating ASR and CR (cf. Table 3), STrader performs the best, indicating its outstanding ability in balancing return and risk.

Result 3: Ablation Study. To answer **Q3**, we conduct ablation experiments with two simplified versions of STrader (cf. Table 3). Firstly, STrader outperforms STrader-w/o-SPF on CW with an average improvement of 7.279%, which indicates STrader's superiority to model microscopic price time series with a complex stochastic process to address the **CH1**. Secondly, STrader outperforms STrader-w/o-TP on CW with an average improvement of 36.136%. Ignoring the TPs makes STrader unable to obtain the excess intraday return, which is consistent with the analysis of **CH2**.

Result 4: Portfolio Qualitative Analysis. To answer **Q4**, we conduct a qualitative analysis of the two stocks that we have described in our motivating example. Figure 4 depicts the candidate TPs sets decided by STrader and their comparison to the ground truth ones. The results indicate that STrader can indeed find the appropriate buying points and selling points for the stocks expected to buy and sell respectively because STrader models their microscopic price time series precisely. In addition, to validate how well STrader approximates the exact appropriate TPs, we calculate the recall on deciding TPs of our strategy normalized by that of the random sample strategy. Recall of each strategy is $recall = \frac{1}{NT}\sum_{i\in[N],d\in[T]}\frac{|\mathcal{T}_{i,d}^{10}|\cap|\mathcal{T}_{i,d}|}{|\mathcal{T}_{i,d}|}$, where $\mathcal{T}_{i,d}^{10}$ is the candidate TPs set of size 10 predicted by a strategy and $\mathcal{T}_{i,d}$ is the top 10 daily optimal TPs set of stock i on day d. The results of the normalized recall are 2.68, 1.12, and 1.34 on DJIA, SSE, and COIN, respectively. Combining the qualitative analysis and the ablation study results, we can conclude that STrader has outstanding ability in deciding appropriate trading points for earning intraday profit.

[4]The index is DJIA (^DJI) in DJIA dataset, and SSE 50 (000016.SH) in SSE dataset. The index in COIN dataset is the average return on 12 cryptocurrencies.

[5]Improvement of STrader over the best-performing baselines.

[6]Statistically not different from the best-performing baselines if p-value < 0.05 (p-value with paired t-test).

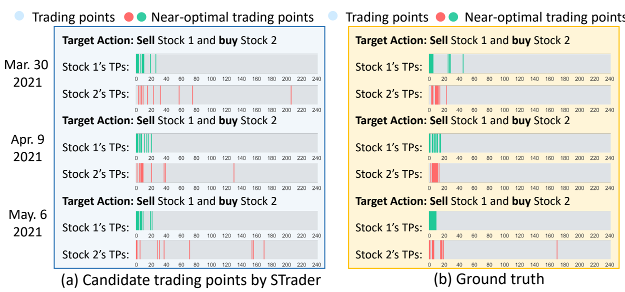

Figure 4: Case study.

6 Conclusions and Future Work

In this paper, we first propose an innovative portfolio problem named TPPO, then propose a novel strategy named STrader to address two challenging problems in modeling the microscopic price time series and deciding the trading points. Empirical studies demonstrate the effectiveness of STrader. In the future, we plan to extend our work in two potential directions. The first direction is to further incorporate side information to improve the ability to generate microscopic price time series for STrader. The second direction is to model multi-day microscopic time series, which are discontinuous due to price jumps between days.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.62172362), Leading Expert of "Ten Thousands Talent Program" of Zhejiang Province (No.2021R52001), and the cooperation project of MYbank, Ant Group.

References

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proc. of NeurIPS*, pages 6572–6583, 2018.

[Ding *et al.*, 2020] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. Hierarchical multi-scale gaussian transformer for stock movement prediction. In *Proc. of IJCAI*, pages 4640–4646, 2020.

[Fraccaro *et al.*, 2016] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Proc. of NeurIPS*, pages 2199–2207, 2016.

[Grathwohl *et al.*, 2018] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *Proc. of ICLR*, 2018.

[Imajo *et al.*, 2021] Kentaro Imajo, Kentaro Minami, Katsuya Ito, and Kei Nakagawa. Deep portfolio optimization via distributional prediction of residual factors. In *Proc. of AAAI*, pages 213–222, 2021.

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

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kingma and Dhariwal, 2018] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. of NeurIPS*, pages 10236–10245, 2018.

[Li and Arora, 2019] Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019.

[Li and Hoi, 2014] Bin Li and Steven C. H. Hoi. Online portfolio selection: a survey. *ACM Computing Surveys*, 87(2):1–36, 2014.

[Li *et al.*, 2020] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *Proc. of AISTATS*, pages 3870–3882, 2020.

[Liang *et al.*, 2021] Qianqiao Liang, Mengying Zhu, Xiaolin Zheng, and Yan Wang. An adaptive news-driven method for cvar-sensitive online portfolio selection in non-stationary financial markets. In *Proc. of IJCAI*, pages 2708–2715, 2021.

[Liu *et al.*, 2019] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proc. of CVPR*, pages 1871–1880, 2019.

[Liu *et al.*, 2020] Guang Liu, Yuzhao Mao, Qi Sun, Hailong Huang, Weiguo Gao, Xuan Li, Jianping Shen, Ruifan Li, and Xiaojie Wang. Multi-scale two-way deep neural network for stock trend prediction. In *Proc. of IJCAI*, pages 4555–4561, 2020.

[Luo *et al.*, 2018] Rui Luo, Weinan Zhang, Xiaojun Xu, and Jun Wang. A neural stochastic volatility model. In *Proc. of AAAI*, pages 6401–6408, 2018.

[Marathe and Ryan, 2005] Rahul R Marathe and Sarah M Ryan. On the validity of the geometric brownian motion assumption. *The Engineering Economist*, 50(2):159–192, 2005.

[Moody and Saffell, 2001] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, 2001.

[Oksendal, 2013] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[Rezende and Mohamed, 2015] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. of ICML*, pages 1530–1538, 2015.

[Rubanova *et al.*, 2019] Yulia Rubanova, Tian Qi Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Proc. of NeurIPS*, pages 5321–5331, 2019.

[Shi *et al.*, 2019] Si Shi, Jianjun Li, Guohui Li, and Peng Pan. A multi-scale temporal feature aggregation convolutional neural network for portfolio management. In *Proc. of CIKM*, pages 1613–1622, 2019.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang *et al.*, 2021] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In *Proc. of AAAI*, pages 643–650, 2021.

[Xu *et al.*, 2020] Ke Xu, Yifan Zhang, Deheng Ye, Peilin Zhao, and Mingkui Tan. Relation-aware transformer for portfolio policy learning. In *Proc. of IJCAI*, pages 4647–4653, 2020.

[Ye *et al.*, 2020] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proc. of AAAI*, pages 1112–1119, 2020.

[Yor, 2001] Marc Yor. *Exponential functionals of Brownian motion and related processes*, volume 112. Springer, 2001.

[Zhang *et al.*, 2022] Yifan Zhang, Peilin Zhao, Qingyao Wu, Bin Li, Junzhou Huang, and Mingkui Tan. Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):236–248, 2022.

[Zhu *et al.*, 2020] Mengying Zhu, Xiaolin Zheng, Yan Wang, Qianqiao Liang, and Wenfang Zhang. Online portfolio selection with cardinality constraint and transaction costs based on contextual bandit. In *Proc. of IJCAI*, pages 4682–4689, 2020.