Cost-Sensitive Portfolio Selection via Deep Reinforcement Learning

Yifan Zhang, Peilin Zhao[†], Bin Li, Junzhou Huang, Qingyao Wu and Mingkui Tan[†]

Abstract—Portfolio Selection is an important real-world financial task and has attracted extensive attention in artificial intelligence communities. This task, however, has two main difficulties: (i) the non-stationary price series and complex asset correlations make the learning of feature representation very hard; (ii) the practicality principle in financial markets requires controlling both transaction and risk costs. Most existing methods adopt handcraft features and/or consider no constraints for the costs, which may make them perform unsatisfactorily and fail to control both costs in practice. In this paper, we propose a cost-sensitive portfolio selection method with deep reinforcement learning. Specifically, a novel two-stream portfolio policy network is devised to extract both price series patterns and asset correlations, while a new cost-sensitive reward function is developed to maximize the accumulated return and constrain both costs via reinforcement learning. We theoretically analyze the near-optimality of the proposed reward, which shows that the growth rate of the policy regarding this reward function can approach the theoretical optimum. We also empirically evaluate the proposed method on real-world datasets. Promising results demonstrate the effectiveness and superiority of the proposed method in terms of profitability, cost-sensitivity and representation abilities.

Index Terms—Portfolio Selection; Reinforcement Learning; Deep Learning; Transaction Cost.

1 Introduction

PORTFOLIO Selection [40] aims at dynamically allocating the wealth among a set of assets to maximize the long-term return. This task, however, is difficult for many individual investors, since even an expert has to spend a lot of time and efforts to deal with each asset with professional knowledge. Recently, many intelligent portfolio selection methods have been proposed and have shown remarkable improvement in performance [9], [13], [15], [27], [28], [53], [64]. However, these methods can be limited in practice, due to two main challenges brought by the complex nature of portfolio selection as follows.

One of the key challenges in portfolio selection is how to represent the non-stationary price series, since the asset price sequences often contain a large number of noises, jumps and oscillations. Most existing methods use handcraft features, such as moving average [34] and stochastic technical indicators [45], which, however, perform unsatisfactorily because of poor representation abilities [14]. In recent years, deep neural networks (DNNs) have shown strong representation abilities in modeling sequence data [57] and often lead to better performance [31]. However, it is non-trivial for existing DNNs to directly extract price sequential patterns and asset correlations simultaneously. Nevertheless, both kinds of information significantly affect the decision-making for portfolio selection. More critically, the dynamic nature of portfolio selection and the lack of well-labeled data make DNNs hard to train.

- *Y. Zhang, Q. Wu and M. Tan are with South China University of Technology, China. E-mail: sezyifan@mail.scut.edu.cn; {qyw, mingkuitan}@scut.edu.cn.*
- *P. Zhao and J. Huang are with Tencent AI Lab, China. Email: peilinzhao@hotmail.com; joehhuang@tencent.com.*
- *B. Li is with Wuhan University, China. E-mail: binli.whu@whu.edu.cn.*
- *[†] P. Zhao is the co-first author. M. Tan is the corresponding author.*

Another key challenge of portfolio selection is how to control costs in decision-making, since the transaction cost and the risk cost highly affect the practicality of algorithms. The transaction cost (*e.g.*, tax and commission) is common in decision-making [36], [47]. Ignoring this cost may lead to aggressive trading [12] and bring biases into the estimation of returns [49]. The risk cost is incurred by the fluctuation of returns and is an important concern in financial investment [46]. Neglecting this cost may lead to a disastrous consequence in practice [21]. Most existing methods consider either one of them but do not constrain both costs simultaneously, which may limit their practical performance.

In this paper, considering the challenges of portfolio selection and its dynamic nature, we formulate portfolio selection as a Markov Decision Process (MDP), and propose a cost-sensitive portfolio policy network (PPN) to address it via reinforcement learning. Our main contributions are summarized as follows.

- To extract meaningful features, we devise a novel two-stream network architecture to capture both price sequential information and asset correlation information. With such information, PPN makes more profitable decisions.
- To control both transaction and risk costs, we develop a new cost-sensitive reward function. By exploiting reinforcement learning to optimize this reward function, the proposed PPN is able to maximize the accumulated return while controlling both costs.
- We theoretically analyze the near-optimality of the proposed reward. That is, the wealth growth rate regarding this reward function can be close to the theoretical optimum.
- Extensive experiments on real-world datasets demonstrate the effectiveness and superiority of the proposed method in terms of profitability, cost-sensitivity and representation abilities.

2 Related Work

Following Kelly investment principle [29], many kinds of portfolio selection methods have been proposed, including online learning and reinforcement learning based methods.

Online learning based methods maximize the expected log-return with sequential decision-making. The pioneering studies include Constant Rebalanced Portfolios (CRP) [10], [11], Universal Portfolios (UP) [11], Exponential Gradient (EG) [23], Anti-correlation (Anticor) [6] and Online Netwon Step (ONS) [2]. Recently, several methods exploit the mean reversion property to select the portfolio, e.g., Confidence Weighted Mean Reversion (CWMR) [32], Passive Aggressive Mean Reversion (PAMR) [33], Online Moving Average Reversion (OLMAR) [34], Robust Median Reversion (RMR) [26] and Weighted Moving Average Mean Reversion (WMAMR) [17]. In addition, the work [55] proposes an ensemble learning method for Kelly growth optimal portfolio.

However, all the above methods ignore the learning of sequential features and only use some handcraft features, such as moving average and stochastic technical indicators. As a result, they may perform unsatisfactorily due to poor representation abilities [14]. More critically, many of the above methods assume no transaction cost. Such a cost will bring biases into the estimation of accumulative returns [49], and thus affects the practical performance of these methods. In contrast, our proposed method not only learn good feature representation based on the proposed two-stream architecture, but is also sensitive to both costs.

On the other hand, reinforcement learning based methods use reinforcement learning algorithms to optimize specific utility functions and make comprehensive policies [42], [43], [44], [46], [47], [48]. However, all these methods ignore the feature representation on portfolios. Very recently, some studies apply deep reinforcement learning to portfolio selection, where they use neural networks to extract features [19], [28]. Specifically, the state-of-the-art one is the ensemble of identical independent evaluations (EIIE) [28]. However, both methods [19], [28] ignore the asset correlation and do not control costs during optimization, leading to limited representation abilities and performance. In contrast, our method can control both kinds of costs relying on the new proposed cost-sensitive reward.

Beyond that, there are also some theoretical studies on the optimal portfolio. To be specific, a theoretical optimal policy can be obtained by maximizing the expected log-return [3]. Based on this, a mean-variance portfolio selection is studied [50]. However, both studies assume no transaction cost, making them less practical. When considering the transaction cost, a theoretical optimal strategy can be achieved by optimizing the expected rebalanced log-return [20]. This work, however, ignores the risk cost. Instead, in this paper, we provide theoretical analyses for the proposed reward in the presence of both costs.

3 Problem Settings

Consider a portfolio selection task over a financial market during n periods with $m{+}1$ assets, including one cash asset and m risk assets. On the t-th period, we denote the prices of all assets as $p_t{\in}\mathbb{R}_+^{(m+1)\times d}$, where each row $p_{t,i}{\in}\mathbb{R}_+^d$ indicates the feature of asset i, and d denotes the number of prices. Specifically, we set $d{=}4$ in this paper. That is, we consider four kinds of prices, namely the opening, highest, lowest and closing prices. One can generalize it to more prices to obtain more information. The price series is represented by $P_t{=}\{p_{t-k}, .., p_{t-1}\}$, where k is the length of the price series.

The price change on the t-th period is specified by a *price relative vector* $x_t{=}\frac{p_t^c}{p_{t-1}^c} \in \mathbb{R}_+^{m+1}$, where p_t^c is the closing price of assets. Typically, $x_{t,0}$ represents the price change of the cash asset. Assuming there is no inflation or deflation, the cash is risk-free with invariant price, *i.e.*, $\{\forall t | x_{t,0}{=}1\}$, and it has little influence on the learning process. We thus exclude the cash asset in the input, *i.e.*, $P_t{\in}\mathbb{R}^{m\times k\times 4}$. When making decisions, the investment decision is specified by a *portfolio vector* $a_t{=}[a_{t,0}, a_{t,1}, a_{t,2}, \dots, a_{t,m}] \in \mathbb{R}^{m+1}$, where $a_{t,i}{\geq}0$ is the proportion of asset i, and $\sum_{i=0}^{m+1} a_{t,i}{=}1$. Here, the portfolio decision contains the proportion of all assets, including the cash $a_{t,0}$. We initialize the *portfolio vector* as $a_0{=}[1, 0, .., 0]$ and initialize the gross wealth as $S_0{=}1$. After n periods, the accumulated wealth, if ignoring the transaction cost c_t, is $S_n{=}S_0 \prod_{t=1}^n a_t^\top x_t$; otherwise, $S_n{=}S_0 \prod_{t=1}^n a_t^\top x_t(1-c_t)$.

There are two general assumptions [34], [59] in this task: (i) perfect liquidity: each investment can be carried out immediately; (ii) zero-market-impact: the investment by the agent has no influence on the financial market, *i.e.*, the environment.

3.1 Markov Decision Process for Portfolio Selection

We formulate the investment process as a generalized Markov Decision Process by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Specifically, as shown in Fig. 1, on the t-th period, the agent observes a state $s_t{=}P_t{\in}\mathcal{S}$, and takes an action $a_t{=}\pi(s_t, a_{t-1}){\in}\mathcal{A}$, which determines the reward $r_t{=}a_t^\top x_t{\in}\mathcal{R}$, while the next state is a stochastic transition $s_{t+1}{\sim}\mathcal{P}(s_t)$. Specifically, $\pi(s_t, a_{t-1})$ is a *portfolio policy*, where a_{t-1} is the action of last period. When considering the transaction cost, the reward will be adjusted as $r_t^c{:=}r_t{*}(1{-}c_t)$, where c_t is the proportion of transaction costs. In Fig. 1, portfolio policy network serves as an agent which aims at maximizing the accumulated return while controlling both the transaction and risk costs.

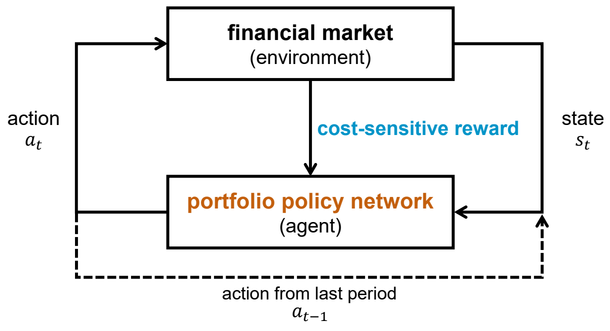


Fig. 1: Markov decision process for portfolio selection (Better viewed in color)

Remark 1. When trading volumes in the financial market are high enough, both general assumptions are near to reality. Moreover, the assumption (ii) indicates that the action $\mathcal{A}$ will not affect the state transaction $\mathcal{P}$. That is, the state transaction only depends on the environment.

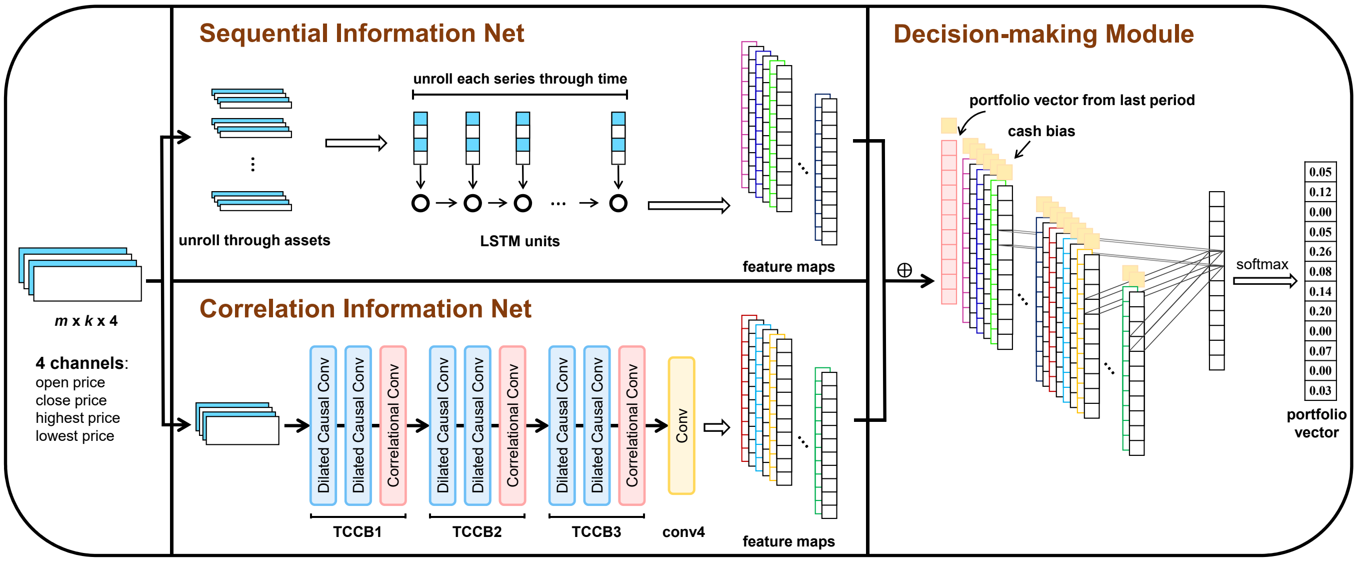


Fig. 2: The scheme of the proposed portfolio policy network, where Correlation Information Net consists of three temporal correlational convolution blocks and $\oplus$ denotes the concatenation operation. More detailed architecture information can be found in Section 6.1.4 (Better viewed in color)

4 Portfolio Policy Network

4.1 General Architecture

In practice, both the price sequential pattern and the asset correlation are significant for the decision-making in portfolio selection tasks. Specifically, the price sequential pattern reflects the price changes of each asset; while the asset correlation reveals the macro market trend and the relationship among assets. Therefore, it is necessary to capture both types of information in the learning process. To this end, we develop a two-stream architecture for portfolio policy network (PPN) to extract portfolio features. As shown in Fig. 2, PPN consists of three major components, namely the sequential information net which is to extract price sequential patterns, the correlation information net which is to extract asset correlations, and the decision-making module. Specifically, we will detail these components in the following subsections.

4.2 Sequential Information Net

It is non-trivial to extract the price sequential pattern of portfolio series due to the non-stationary property of asset prices. To solve this issue, we propose a sequential information net, based on LSTM [24], to extract the sequential pattern of portfolios. This is inspired by the strong ability of LSTM in modeling non-stationary and noisy sequential data [18]. Concretely, as shown in Fig. 2 (top), the sequential information net processes each asset separately, and concatenates the feature of each asset along the height dimension as a whole feature map. We empirically show that the sequential information net is able to extract good sequential features and helps to gain more profits when only considering the price sequential information (See results in Section 6.3).

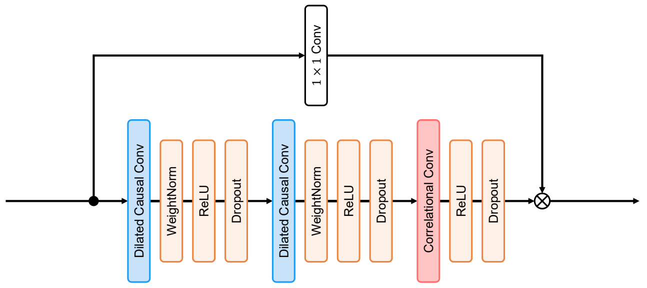
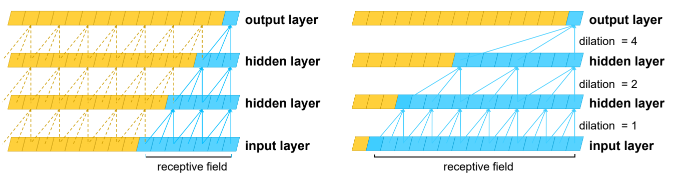


Fig. 3: Illustration of the temporal correlational convolution block, where $\otimes$ denotes the ReLU activation operation (Better viewed in color)

4.3 Correlation Information Net

Although recurrent networks can model the price sequential information, they can hardly extract asset correlations, since they process the price series of each asset separately. Instead, we propose a correlation information net to capture the asset correlation information based on fully convolution operations [30], [39], [60]. Specifically, we devise a new temporal correlational convolution block (TCCB) and use it to construct the correlation information net, as shown in Fig. 2 (bottom).

The proposed TCCB is motivated by the complex nature of portfolio selection. To be specific, we need to extract the asset correlation and model the price series simultaneously. To this end, we exploit the dilated causal convolution operation to model the portfolio time-series variation, and devise a new correlational convolution operation to capture the asset correlation information. To make it clear, we summarize the detailed structure of TCCB in Fig. 3, and describe its two main components as follows.



(a) Causal convolutions (b) Dilated causal convolutions

Fig. 4: Superiority of dilated causal convolutions compared to casual convolutions (Better viewed in color)

4.3.1 *Dilated causal convolutions*

Inspired by [60], we use the causal convolution operation, built upon 1D convolutions, to extract the sequential information. Specifically, it can keep the sequence order invariant and guarantee no information leakage from the future to the past by using padding and filter shifting. A simple example is presented in Fig. 4 (a), which depicts a stack of causal convolutions with kernel size 3×1. However, the causal convolution usually requires very large kernel sizes or too many layers to increase the receptive field, leading to a large number of parameters.

To overcome this, inspired by [5], we use the dilated operation to improve the causal convolution, since it can guarantee exponentially large receptive fields [63]. To be specific, the dilation operation is equivalent to introducing a fixed step between every two adjacent filter taps [5]. A simple example is provided in Fig. 4 (b), which depicts a stack of dilated causal convolutions with kernel size 3×1. One can find that the receptive field of the dilated causal convolution is much larger than the causal convolution. Specifically, the gap of receptive fields between the two convolutions increases exponentially with the increase of the network depth.

4.3.2 *Correlational convolutions*

Note that existing fully convolution networks [5], [39], [60], *e.g.*, dilated causal convolutions, can hardly extract asset correlations, since they process the price of each asset separately by using 1D convolutions. To address this, we devise a correlational convolution operation, which seeks to combine the price information from different assets, by fusing the features of all assets at every time step. Specifically, we apply padding operations to keep the structure of feature maps invariant. With this operation, the correlation information net can construct a multi-block architecture as shown in Fig. 2 (bottom), and asymptotically extract the asset correlation without changing the structure of asset features.

In addition, we denote a degenerate variant of TCCB as TCB which does not use the correlational convolution operation. Concretely, TCB only extracts the price sequential information by using dilated causal convolutions. We empirically show that the correlation information net with TCCB can extract good asset correlations and helps to gain more profits, compared to TCB (See results in Section 6.3). This result further demonstrates the significance of the asset correlation in portfolio selection and confirms the effectiveness of the correlation information net.

4.4 Decision-making Module

Based on all extracted feature maps, PPN makes the final portfolio decision. To avoid heavy transaction costs, we adopt a recursive mechanism [42] in the decision-making. That is, the decision-making requires considering the action from last period, which helps to discourage huge changes between portfolios and thus constrain aggressive trading.

In practice, we directly concatenate the portfolio vector from last period into feature maps. Here, the recursive portfolio vector $a_{t-1}\in\mathbb{R}^m$ also excludes the cash term, since it is risk-free and has little influence on the learning process. We then add a fixed cash bias into all feature maps in order to construct complete portfolios, and decide the final portfolio $a_t\in\mathbb{R}^{m+1}$ with a convolution operation via softmax function. We highlight that the final convolution operation is analogous to making decisions by voting all feature vectors.

Remark 2. The recursion mechanism of PPN makes the optimal portfolio policy time-variant, i.e., it is a non-stationary portfolio selection process [20]. More critically, the well-labeled data in portfolio selection is very scarce. These challenges make PPN hard to train with supervised learning.

5 Reinforcement Learning

Considering the complexity of portfolio selection, instead of supervised learning, we adopt reinforcement learning to optimize PPN and develop a new cost-sensitive reward function to constrain both transaction and risk costs during the optimization.

5.1 Direct Policy Gradient Algorithm

With the success of AlphaGo, many deep reinforcement learning algorithms have been proposed and achieve impressive performance [38], [41], [52]. Among these reinforcement learning algorithms, the most suitable one seems to be DDPG, since it can directly approximate the deterministic portfolio decision with DNNs. Nevertheless, DDPG needs to estimate the state-action values via a Q network, which is often hard to learn and usually fails to converge even in a simple MDP [42]. In our case, this issue is more serious since the decision process is non-stationary. Hence, the selection of reinforcement learning algorithms is non-trivial.

Fortunately, the sequential decision-making is an immediate reward process. That is, the reward of portfolio selection is immediately available. We can directly optimize the reward function, and use the policy gradient from rewards to train PPN. We highlight that this simple policy gradient method can guarantee at least a sub-optimal solution as follows.

Proposition 1. Let θ be the parameters of the policy network, e.g., PPN and R be the reward. If the policy network is updated approximately proportional to the gradient $\triangle\theta\approx\eta\frac{\partial R}{\partial\theta}$, where η is the learning rate, then θ can usually be assured to converge to a local optimal policy in the reward R [58].

We will further discuss the selection of reinforcement learning in Section 7.2.

5.2 Cost-sensitive Reward Function

To constrain both transaction and risk costs, we develop a new cost-sensitive reward function. To this end, we first devise a risk-sensitive reward regarding no transaction cost.

5.2.1 *Risk-sensitive reward*

Assuming there is no transaction cost, most existing methods use the log-return ($\log r_t$) as the reward, since it helps to guarantee a log-optimal strategy.

Proposition 2. If there is no transaction cost and the market is stationary ergodic, the portfolio policy that maximizes the **expected log-return** $\mathbb{E}\{\log r_t\}$ can achieve a **log-optimal strategy**, with the theoretical maximal growth rate $\bar{W}^*{=}\lim_{t\to\infty}\frac{1}{t}\log\bar{S}_t^*$, where $\bar{S}_t^*$ is the accumulated wealth [20].

In practice, we can use the empirical approximation of the expected log-return $\mathbb{E}\{\log r_t\}$ as the reward: $R = \frac{1}{T}\sum_{t=1}^{T}\hat{r}_t$, where $\hat{r}_t{:=}\log r_t$ is the log-return on the t-th period, and T is the total number of sampled portfolio data.

However, this reward ignores the risk cost, thus being less practical. To solve this issue, we define the empirical variance of log-return on sampled portfolio data as the risk penalty, *i.e.*, $\sigma^2(\hat{r}_t|t{=}1,..,T)$, shortly $\sigma^2(\hat{r}_t)$, and then develop a risk-sensitive reward function as:

$$R = \frac{1}{T}\sum_{t=1}^{T}\hat{r}_t - \lambda\sigma^2(\hat{r}_t),$$

where $\lambda \geq 0$ is a trade-off hyperparameter.

We next show the near-optimality of the risk-sensitive reward. It represents the relationship between the policy regarding this risk-sensitive reward and the log-optimal strategy in Prop. 2 which cannot constrain the risk cost.

Theorem 1. Let $\bar{W}^*$ be the growth rate of the log-optimal strategy and S_t^* be the wealth achieved by the optimal portfolio policy that maximizes $\mathbb{E}\{\log r_t\}{-}\lambda Var\{\log r_t\}$. Under the same condition as in Prop. 2, for any $\lambda{\geq}0$ and $\frac{1}{e}{\leq}r_t{\leq}e$, the maximal growth rate of this policy satisfies:

$$\bar{W}^* \geq \liminf_{t\to\infty}\frac{1}{t}\log S_t^* \geq \bar{W}^* - \frac{9}{4}\lambda.$$

See the supplementary for the proof. From Theorem 1, when λ is sufficiently small, the growth rate of the optimal strategy regarding this reward can approach the theoretical best one, *i.e.*, the log-optimal strategy.

5.2.2 *Cost-sensitive reward*

Despite having theoretical guarantees, the risk-sensitive reward assumes no transaction cost, thus being insufficient. To solve this issue, we improve it by considering the proportional transaction cost. In this setting, the log-return will be adjusted as $\hat{r}_t^c{:=}\log r_t^c{=}\log r_t{*}(1{-}c_t)$. Specifically, the expected rebalanced log-return can also guarantee the optimality when facing the transaction cost.

Proposition 3. If the market is stationary and the return process is a homogeneous first-order Markov process, the policy that maximizes the **expected rebalanced log-return** $\mathbb{E}\{\log r_t^c\}$ can be **optimal** when facing transaction costs, with the maximal growth rate $\tilde{W}^*{=}\lim_{t\to\infty}\frac{1}{t}\log\tilde{S}_t^*$, where $\tilde{S}_t^*$ is the wealth achieved by this optimal strategy [20].

However, optimizing this rebalanced log-return cannot control transaction costs well. To solve this, we further constrain the transaction cost proportion c_t. Let $\omega_t{:=}1{-}c_t$ be the proportion of net wealth, and let ψ_p and ψ_s be transaction cost rates for purchases and sales. On the t-th period, after making the decision a_t, we need to rebalance from the current portfolio $\hat{a}_{t-1}{=}\frac{a_{t-1}\odot x_{t-1}}{a_{t-1}^\top x_{t-1}}$ to a_t, where $\odot$ is the element-wise product. During rebalancing, the sales occur if $\hat{a}_{t-1,i}{-}a_{t,i}\omega_t{>}0$, while the purchases occur if $a_{t,i}\omega_t{-}\hat{a}_{t-1,i}{>}0$. Hence,

$$c_t = \psi_s\sum_{i=1}^{m}(\hat{a}_{t-1,i}{-}a_{t,i}\omega_t)^+{+}\psi_p\sum_{i=1}^{m}(a_{t,i}\omega_t{-}\hat{a}_{t-1,i})^+,$$

where $(x)^+{=}\max(x,0)$. Following [36], we set $\psi_p{=}\psi_s{=}\psi \in [0,1]$, and then obtain:

$$c_t = \psi\|a_t\omega_t - \hat{a}_{t-1}\|_1.$$

Getting rid of ω_t, we can bound c_t as follows.

Proposition 4. Let ψ be the transaction cost rate, $\hat{a}_{t-1}$ and a_t be the asset allocations before and after rebalancing. The cost proportion c_t on the t-th period is bounded:

$$\frac{\psi}{1+\psi}\|a_t{-}\hat{a}_{t-1}\|_1 \leq c_t \leq \frac{\psi}{1-\psi}\|a_t{-}\hat{a}_{t-1}\|_1,$$

where $\|a_t{-}\hat{a}_{t-1}\|_1 \in \Big(0, \frac{2(1-\psi)}{1+\psi}\Big]$.

See the supplementary for the proof. Prop. 4 shows that both upper/lower bounds of c_t are related to $\|a_t{-}\hat{a}_{t-1}\|_1$: the smaller the L1 norm, the smaller the upper/lower bounds and thus the c_t. By constraining this L1 norm, we derive the final cost-sensitive reward based on Theorem 1 and Prop. 4 as:

$$R = \underbrace{\frac{1}{T}\sum_{t=1}^{T}\hat{r}_t^c - \lambda\sigma^2(\hat{r}_t^c)}_{\text{risk-sensitive reward}} - \underbrace{\frac{\gamma}{T{-}1}\sum_{t=2}^{T}\|a_t{-}\hat{a}_{t-1}\|_1}_{\text{transaction cost constraint}}, \quad (1)$$

where $\gamma \geq 0$ is a trade-off hyperparameter.

We next show the near-optimality of the cost-sensitive reward, which reflects the relationship between the strategy regarding this cost-sensitive reward and the theoretical optimal strategy in Prop. 3 which cannot control both costs.

Theorem 2. Let $\tilde{W}^*$ be the growth rate of the theoretical optimal strategy that optimizes $\mathbb{E}\{\log r_t^c\}$, and S_t^* be the wealth achieved by the optimal policy that maximizes $\mathbb{E}\{\log r_t^c\}{-}\lambda Var\{\log r_t^c\}{-}\gamma\mathbb{E}\{\|a_t{-}\hat{a}_{t-1}\|_1\}$. Under the same condition as in Props. 3 and 4, for any $\lambda{\geq}0$, $\gamma{\geq}0$, $\psi{\in}[0,1]$ and $\frac{1}{e}{\leq}r_t^c{\leq}e$, the maximal growth rate of this policy satisfies:

$$\tilde{W}^* \geq \liminf_{t\to\infty}\frac{1}{t}\log S_t^* > \tilde{W}^* - \frac{9}{4}\lambda - \frac{2\gamma(1-\psi)}{1+\psi}.$$

See the supplementary for the proof. Specifically, when λ and γ are sufficiently small, the wealth growth rate of the strategy regarding the cost-sensitive reward can be close to the theoretical optimum.

We highlight that this reward can be helpful to design more effective portfolio selection methods with the near-optimality guarantee when facing both transaction and risk costs in practice tasks. Specifically, by optimizing this reward with the direct policy gradient method, the proposed PPN can learn at least a sub-optimal policy to effectively maximize accumulated returns while controlling both costs as shown in Prop. 1.

Remark 3. The denominator T in Eqn. (1) ensures that the rewards from different price sequences are equivalent. Moreover, the assumption (ii) makes the action and environment isolated, allowing us to use the same price segment to evaluate different actions. These enable us to train PPN with the online stochastic batch method [28], which helps to improve the data efficiency.

6 Experimental Results

We evaluate PPN in three main aspects: (1) the profitability on real-world datasets; (2) the feature extraction ability for portfolio series; (3) the cost-sensitivity to both transaction and risk costs. To this end, we first describe the baselines, metrics, datasets and implementation details in experiments.

6.1 Experimental Settings

6.1.1 Baselines

We compare PPN with several state-of-the-art methods, including Uniform Buy-And-Hold (UBAH), best strategy in hindsight (Best), CRP [11], UP [11], EG [23], Anticor [6], ONS [2], CWMR [32], PAMR [33], OLMAR [34], RMR [26], WMAMR [17] and EIIE [28]. In addition, to evaluate the effectiveness of the asset correlation, we also compare PPN with a degenerate variant PPN-I that only exploits independent price information by using TCB.

6.1.2 Metrics

Following [36], [54], we use three main metrics to evaluate the performance. The first is *accumulated portfolio value* (APV), which evaluates the profitability when considering the transaction cost.

$$\text{APV} = S_n = S_0 \prod_{t=1}^{n} a_t^\top x_t (1 - c_t),$$

where $S_0 = 1$ is the initialized wealth. In addition, a_t, x_t and c_t indicate the portfolio vector, the price relative vector and the transaction cost proportion on the t-th round, respectively.

A major drawback of APV is that it neglects the risk factor. That is, it only relies on the returns without considering the fluctuation of these returns. Thus, the second metric is *Sharpe Ratio* (SR), which evaluates the average return divided by the fluctuation, *i.e.*, the *standard deviation* (STD) of returns.

$$\text{SR} = \frac{\text{Average}(r_t^c)}{\text{Standard Deviation}(r_t^c)},$$

where r_t^c is the rebalanced log-return on the t-th round.

Although SR considers the volatility of portfolio values, it treats upward and downward movements equally. However, downward movements are usually more important, since it measures algorithmic stability in the market downturn. To highlight the downward deviation, we further use *Calmar Ratio* (CR), which measures the accumulated profit divided by *Maximum Drawdown* (MDD):

$$\text{CR} = \frac{S_n}{\text{MDD}},$$

where MDD denotes the biggest loss from a peak to a trough:

$$\text{MDD} = \max_{t:\tau>t} \frac{S_t - S_\tau}{S_t}.$$

Note that the higher APV, SR and CR values, the better profitability of algorithms; while the lower STD and MDD values, the higher stability of returns. We also evaluate average *turnover* (TO) when examining the influence of transaction costs, since it estimates the average trading volume.

$$\text{TO} = \frac{1}{2n} \sum_{t=1}^{n} \|\hat{a}_{t-1} - a_t \omega_t\|_1,$$

where $\hat{a}_{t-1}$ and ω_t indicate the current portfolio before rebalance and net wealth proportion on the t-th round.

6.1.3 Datasets and preprocessing

The globalization and the rapid growth of crypto-currency markets yield a large number of data in the finance industry. Hence, we evaluate PPN on several real-world crypto-currency datasets. Following the data selection method in [28], all datasets are accessed with Poloniex[1]. To be specific, we set the bitcoin as the risk-free cash and select risk assets according to the crypto-currencies with top month trading volumes in Poloniex. We summary statistics of the datasets in Table 1. All assets, except the cash asset, contain all 4 prices. The price window of each asset spans 30 trading periods, where each period is with 30-minute length.

Some crypto-currencies might appear very recently, containing some missing values in the early stage of data. To fill them, we use the flat fake price-movements method [28]. Moreover, the decision-making of portfolio selection relies on the relative price change rather than the absolute change [34]. We thus normalize the price series with the price of the last period. That is, the input price on the t-th period is normalized by $P_t = \frac{P_t}{P_{t,30}} \in \mathbb{R}^{m\times 30\times 4}$, where $P_{t,30} \in \mathbb{R}^{m\times 4}$ represents the prices of the last period.

TABLE 1: The detailed statistics information of the used crypto-currency datasets

Datasets	#Asset	Training Data		Testing Data	
		Data Range	Num.	Data Range	Num.
Crypto-A	12	2016-01 to 2017-11	32269	2017-11 to 2018-01	2796
Crypto-B	16	2015-06 to 2017-04	32249	2017-04 to 2017-06	2776
Crypto-C	21	2016-06 to 2018-04	32205	2018-04 to 2018-06	2772
Crypto-D	44	2016-08 to 2018-06	32205	2018-06 to 2018-08	2772

1. Poloniex's official API: https://poloniex.com/support/api/.

TABLE 2: Detailed network architecture of the proposed portfolio policy network, where we use the following abbreviations: CONV: convolution layer; N: the number of output channel; K: kernel size; S: stride size; P: padding size or operation name; DiR: dilation rate; DrR: dropout rate

Correlation Information Net		
Part	Input → Output shape	Layer information
TCCB1	$(m,k,4) \to (m,k,8)$	DCONV-(N8, K[1x3], S1, P2), DiR1, DrR0.2, ReLU
	$(m,k,8) \to (m,k,8)$	DCONV-(N8, K[1x3], S1, P2), DiR1, DrR0.2, ReLU
	$(m,k,8) \to (m,k,8)$	CCONV-(N8, K[mx1], S1, P-SAME), DrR0.2, ReLU
TCCB2	$(m,k,8) \to (m,k,16)$	DCONV-(N16, K[1x3], S1, P4), DiR2, DrR0.2, ReLU
	$(m,k,16) \to (m,k,16)$	DCONV-(N16, K[1x3], S1, P4), DiR2, DrR0.2, ReLU
	$(m,k,16) \to (m,k,16)$	CCONV-(N16, K[mx1], S1, P-SAME), DrR0.2, ReLU
TCCB3	$(m,k,16) \to (m,k,16)$	DCONV-(N16, K[1x3], S1, P8), DiR4, DrR0.2, ReLU
	$(m,k,16) \to (m,k,16)$	DCONV-(N16, K[1x3], S1, P8), DiR4, DrR0.2, ReLU
	$(m,k,16) \to (m,k,16)$	CCONV-(N16, K[mx1], S1, P-SAME), DrR0.2, ReLU
Conv4	$(m,k,16) \to (m,1,16)$	CONV-(N16, K[1xk], S1, P-VALID), ReLU
Sequential Information Net		
LSTM	$(m,k,4) \to (m,1,16)$	LSTM unit number:16
Decision-making Module		
Concatenation	$(m,16)\oplus(m,16)\oplus(m,1)\oplus(1,33) \to (m+1,33)$	Concatenation of extracted features and other information
Prediction	$(m+1,33) \to (m+1,1)$	CONV-(N1, K[1x1], S1, P-VALID), Softmax

6.1.4 Implementation details

As mentioned above, the overall network architecture of PPN is shown in Fig. 2. Specifically, there are three main components: Correlation Information Net, Sequential Information Net and Decision-making Module. To make it more clearer, we record the detailed architectures in Table 2.

To be specific, in Correlation Information Net, we adopt the temporal correlational convolutional block as the basic module. To be specific, it consists of two components, *i.e.*, the dilated causal convolution layer (DCONV) and the correlation convolution layer (CCONV).

Note that the concatenation operation in the decision-making module has two steps. First, we concatenate all extracted features and the portfolio vector from last period. Then, we concatenate the cash bias into all feature maps.

In addition, we implement the proposed portfolio policy network with Tensorflow [1]. Specifically, we use Adam optimizer with batch size 128 on a single NVIDIA TITAN X GPU. We set the learning rate to 0.001, and choose γ and λ from $10^{[-4:1:-1]}$ using cross-validations. Besides, the training step is 10^5, the cash bias is fixed to 0, and the transaction cost rate is 0.25%, which is the maximum rate at Poloniex. In addition, the training time of PPN is about 4, 5.5, 7.5 and 15 GPU hours on the Crypto-A, Crypto-B, Crypto-C and Crypto-D datasets, respectively. All results on crypto-currency datasets are averaged over 5 runs with random initialization seeds.

6.2 Evaluation on Profitability

We first evaluate the profitability of PPN, and record the detailed performance in Table 3.

From the results, EIIE and PPN-based methods perform better than all other baselines in terms of APV. Since the three methods adopt neural networks to learn policies via reinforcement learning, this observation demonstrates the effectiveness and superiority of deep reinforcement learning in portfolio selection.

Moreover, PPN-based methods perform better than EIIE. This finding implies that PPN-based methods can extract better sequential feature representation, which helps to learn more effective portfolio policies with better profitability.

In addition, PPN outperforms PPN-I in terms of APV. This observation confirms the effectiveness and significance of the asset correlation in portfolio selection.

Last, PPN also achieves the best or relatively good SR and CR performance. Since both metrics belong to risk-adjusted metrics, this finding implies that PPN is able to gain more stable profits than other baselines.

6.3 Evaluation on Representation Ability

We next evaluate the representation abilities of PPN with different extraction modules, when fixing all other parameters. Specifically, we compare PPN and PPN-I with the variants that only adopt one module, *i.e.*, LSTM, TCB or TCCB, namely PPN-LSTM, PPN-TCB and PPN-TCCB. To demonstrate the parallel structure, we also compare PPN and PPN-I with the variants that use the cascaded structure, namely PPN-TCB-LSTM and PPN-TCCB-LSTM. The only difference among these variants is that the extracted features are different. We present the results in Table 4 and Fig. 5, from which we draw several observations.

Firstly, we discuss the variants that only use one feature extraction module. Specifically, PPN-LSTM outperforms PPN-TCB, which means the proposed sequential information net extracts better price sequential patterns. Besides, PPN-TCCB outperforms PPN-LSTM and PPN-TCB, which verifies both TCCB and the correlation information net.

Secondly, all variants that consider asset correlations, *i.e.*, PPN, PPN-TCCB and PPN-TCCB-LSTM, outperform their independent variants, *i.e.*, PPN-I, PPN-TCB and PPN-TCB-LSTM. This observation confirms the significance and effectiveness of the asset correlation in portfolio selection.

Thirdly, all combined variants, *i.e.*, PPN, PPN-I and cascaded modules, outperform the variants that only adopt LSTM, TCB or TCCB. This means that combining both types of information helps to extract better features, which further confirms the effectiveness of the two-stream architecture.

Next, PPN outperforms all other variants, which confirms its strong representation ability. Note that PPN is

TABLE 3: Performance comparisons on different datasets

Algos	Crypto-A			Crypto-B			Crypto-C			Crypto-D		
	APV	SR(%)	CR	APV	SR(%)	CR	APV	SR(%)	CR	APV	SR(%)	CR
UBAH	2.59	3.87	3.39	1.63	2.57	2.43	1.32	3.00	1.61	0.63	0.20	-5.85
Best	6.65	4.59	13.67	3.20	2.95	3.61	2.97	3.15	3.96	1.04	0.64	0.63
CRP	2.40	3.95	3.24	1.90	3.77	3.81	1.30	3.30	1.77	0.66	0.20	-5.85
UP	2.43	3.95	3.28	1.89	3.70	3.71	1.30	3.29	1.76	0.66	0.20	-5.85
EG	2.42	3.96	3.27	1.89	3.71	3.75	1.30	3.30	1.76	0.67	0.21	-5.85
Anticor	2.17	2.96	2.23	21.80	9.92	103.68	0.75	-1.48	-0.91	3.14	6.81	66.28
ONS	1.28	1.40	0.53	1.71	3.15	4.00	1.14	2.33	1.95	1.00	0.19	0.01
CWMR	0.01	-8.21	-0.99	0.64	0.42	-0.54	0.01	-16.61	-0.99	0.38	-0.02	-5.19
PAMR	0.01	-7.17	-0.99	0.880	0.91	-0.20	0.01	-15.48	-0.99	0.82	0.08	-1.78
OLMAR	0.65	0.32	-0.47	774.47	10.91	2040.70	0.05	-7.56	-0.99	11.25	7.21	135.97
RMR	0.69	0.46	-0.42	842.26	11.62	3387.69	0.05	-7.72	-0.99	14.337	7.93	192.59
WMAMR	0.85	0.67	-0.22	87.85	8.25	245.23	0.26	-3.78	-0.98	7.72	6.62	227.16
EIIE	10.48	5.27	21.47	2866.15	13.42	8325.78	2.87	4.04	9.54	113.58	15.11	4670.91
PPN-I (ours)	25.76	6.75	57.05	7549.35	14.74	28915.43	3.93	5.12	15.04	238.93	16.07	8803.95
PPN (ours)	**32.04**	**6.85**	**79.87**	**9842.56**	**14.82**	**37700.03**	**4.81**	**5.89**	**16.11**	**538.22**	**17.82**	**15875.72**

TABLE 4: Evaluations of portfolio policy network with different feature extractors

Module	Crypto-A			Crypto-B			Crypto-C			Crypto-D		
	APV	SR(%)	CR	APV	SR(%)	CR	APV	SR(%)	CR	APV	SR(%)	CR
PPN-LSTM	14.48	5.62	38.19	3550.32	13.75	13297.32	2.85	3.99	6.69	159.54	15.16	6319.84
PPN-TCB	12.76	5.40	26.52	3178.42	13.63	11011.87	2.01	3.32	4.66	102.85	14.09	2972.63
PPN-TCCB	16.51	6.01	35.89	4181.17	13.85	15798.05	3.29	4.53	12.38	171.82	14.97	3945.99
PPN-TCB-LSTM	18.62	6.28	39.87	4485.89	14.18	15232.31	3.49	4.48	10.96	179.43	15.18	9150.33
PPN-TCCB-LSTM	21.03	6.12	52.75	5575.25	14.46	21353.23	3.69	4.72	10.50	224.41	15.99	8522.43
PPN-I	25.76	6.75	57.05	7549.35	14.74	28915.43	3.93	5.12	15.04	238.93	16.07	8803.95
PPN	**32.04**	**6.85**	**79.87**	**9842.56**	**14.82**	**37700.03**	**4.81**	**5.89**	**16.11**	**538.22**	**17.82**	**15875.72**

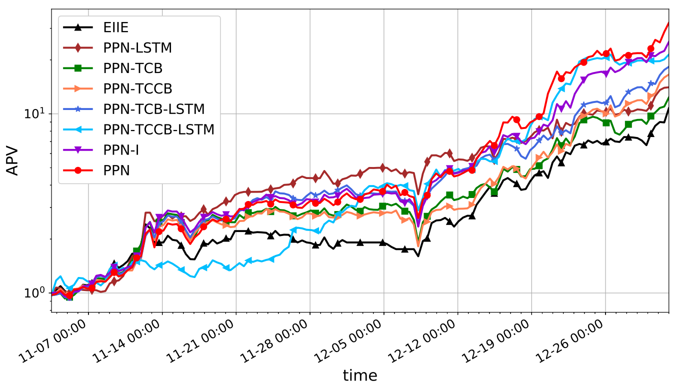


Fig. 5: The performance development of the proposed portfolio policy network with different feature extractors and EIIE on the Crypto-A dataset (Better viewed in color). A larger scale version of this figure can be found in Appx. D.1

not always the best throughout the backtest in Fig. 5. For example, in the early stage, many variants perform similarly. But in the late stage, PPN performs very well. Considering that the correlation between two price events decreases exponentially with their sequential distance [25], this result demonstrates better generalization abilities of PPN.

Lastly, as shown in Fig. 5, there are some periods that all methods (EIIE and PPN based methods) suffer from significant draw-down, like in the middle November and the earlier December. Since it is model-agnostic, such draw-down may result from the market factor instead of the methods themselves. Motivated by this, it is interesting to explore the market influence based on social text information for better portfolio selection in the future.

6.4 Evaluation on Cost-sensitivity

6.4.1 *Influences of transaction costs*

In previous experiments, we have demonstrated the effectiveness of PPN, where the transaction cost rate is 0.25%. However, the effect of the transaction cost rate has not been verified. We thus examine their influences on three dominant methods on Crypto-A.

From Table 5, PPN achieves the best APV performance across a wide range of transaction cost rates. This observation further confirms the profitability of PPN.

Compared to EIIE, PPN-based methods obtain relatively low TO, *i.e.*, lower transaction costs. Since EIIE optimizes only the rebalanced log-return, this finding indicates that our proposed reward controls the transaction cost better.

Moreover, when the transaction cost rate is very large, *e.g.*, $c{=}5\%$, PPN-based algorithms tend to stop trading and make nearly no gains or losses, while EIIE, however, loses most of the wealth with relatively high TO. This implies our proposed methods are more sensitive to the transaction cost.

6.4.2 *Cost-sensitivity to transaction costs*

We further evaluate the influences of γ in the cost-sensitive reward. From Table 6, we find that with the increase of γ, TO values of PPN decrease. This observation means that when introducing $\|a_t{-}\hat{a}_{t-1}\|_1$ into the reward, PPN can better control the trading volume, and thus better overcome the negative effect of the transaction cost.

This finding is also reflected in Fig. 6. With the increase of γ, there are more period intervals remaining unchanged. That is, when the transaction cost outweighs the benefit of trading, PPN will stop the meaningless trading.

Note that, PPN achieves the best APV performance when $\gamma{=}10^{-3}$ in Table 6. This observation is easy to un-

TABLE 5: Comparisons under different transaction cost rates on the Crypto-A dataset

Algos	c=0.01%		c=0.05%		c=0.1%		c=0.25%		c=1%		c=2%		c=5%	
	APV	TO	APV	TO	APV	TO	APV	TO	APV	TO	APV	TO	APV	TO
EIIE	871.18	1.232	254.73	1.076	77.79	0.859	10.48	0.471	1.07	0.247	0.81	0.021	**0.28**	0.020
PPN-I	1571.67	0.964	570.73	0.779	219.30	0.668	25.76	0.316	1.18	0.040	0.96	0.013	0.99	2e-7
PPN	**3741.13**	1.018	**754.57**	0.731	**242.27**	0.658	**32.04**	0.368	**1.61**	0.063	**1.09**	0.019	**1.00**	5e-8

TABLE 6: The performance of portfolio policy network under different γ

γ	Crypto-A		Crypto-B		Crypto-C		Crypto-D	
	APV	TO	APV	TO	APV	TO	APV	TO
10^{-4}	25.24	0.433	2080.69	0.950	4.65	0.667	268.63	1.104
10^{-3}	**32.04**	0.368	**9842.56**	0.888	**4.81**	0.304	**538.22**	0.839
10^{-2}	4.30	0.025	44.01	0.161	1.21	0.008	1.72	0.012
10^{-1}	1.01	2e-08	1.65	3e-03	1.00	1e-7	1.00	3e-7

derstand. If γ is too small, *e.g.*, 10^{-4}, PPN tends to trade aggressively, leading to a large number of transaction costs, thus affecting the profitability of PPN. If γ is large, *e.g.*, 10^{-2} and 10^{-1}, PPN tends to trade passively, thus limiting the model to seeking better profitability. As a result, when setting a more reasonable value, *e.g.*, $\gamma = 10^{-3}$, PPN can learn a better portfolio policy and achieve a better trade-off between the profitability and transaction costs.

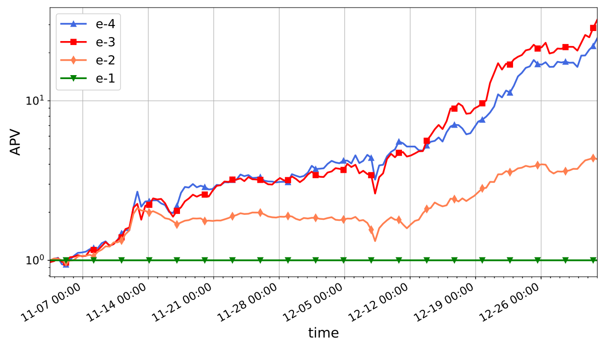


Fig. 6: Performance development of portfolio policy network under different γ on the Crypto-A dataset. A larger scale version of this figure can be found in Appx. D.1

6.4.3 *Cost-sensitivity to risk costs*

We also examine the influences of λ in the cost-sensitive reward, and report the results in Table 7. Specifically, with the increase of λ, the STD values of PPN asymptotically decrease on all datasets. Since λ controls the risk penalty $\sigma^2(\hat{r}_t)$, this result is consistent with the expectation and also demonstrates the effectiveness of PPN in controlling the risk cost. Moreover, with the increase of λ, the MDD results decrease on most datasets. Since MDD depends on the price volatility of the financial market, this result implies that constraining the volatility of returns is helpful to control the downward risk.

7 Discussion

In this section, we further discuss the architecture design of PPN, the selection of reinforcement learning algorithms for PPN and the generalization abilities of PPN.

7.1 Architecture Design of Portfolio Policy Network

As shown in Fig. 2, we propose a two-stream architecture for PPN. Concretely, the sequential information net is based on LSTM, and the correlation information net is built upon TCCB. Noting that the fully convolution networks (e.g., TCB and TCCB) can also extract price sequential patterns, one may ask why we still use LSTM.

To be specific, although TCB and TCCB can learn sequential information, they can hardly make full use of them. Specifically, the traditional convolution assumes time invariance, and uses time-invariant filters to combine convolutional features. This makes fully convolution networks hard to extract large-scale sequence order information [7].

As shown in Fig. 2, the last Conv4 layer of the correlation information net directly uses the time-invariant filter to combine features, and hence only extracts some local sequential information. This makes PPN-TCB perform inferior to PPN-LSTM in Table 4. We thus exploit LSTM to better extract the sequential representation.

On the other hand, we propose TCCB to effectively extract the asset correlation. Such information is beneficial to improve the profitability of PPN and makes PPN-TCCB outperform PPN-TCB and PPN-LSTM in Table 4.

In addition, note that combining both types of information can further strengthen the feature representation of portfolios and make more profitable decisions (See results in Section 6.3). Hence, we devise a two-stream network architecture for PPN to better learn the portfolio series.

7.2 Reinforcement Learning Algorithm Selection

We next discuss the selection of reinforcement learning algorithms. Since we use the direct policy gradient (DPG) method, we mainly discuss why not use Actor-Critic (AC) policy gradient methods.

AC requires learning a "critic" network to approximate the value function, which then generates the policy gradient to update the "actor" network. In AC, the key step is the accurate approximation of the value function. Typically, there are three kinds of value functions. (1) State value: measure the performance of the current state, *i.e.*, good or bad; (2) State-Action value (Q value): measure the performance of the determined action in the current state; (3) Advantage value: measure the advantage of the determined action than the average performance in the current state.

However, all of them are difficult to optimize PPN. First, the state value is unsuitable for our case, since the action of PPN does not affect the environment state due to the general assumption (ii). Thus, it cannot accurately measure

TABLE 7: The performance of portfolio policy network under different λ

λ	Crypto-A			Crypto-B			Crypto-C			Crypto-D		
	APV	STD(%)	MDD(%)	APV	STD(%)	MDD(%)	APV	STD(%)	MDD(%)	APV	STD(%)	MDD(%)
10^{-4}	32.04	2.16	38.86	9842.56	2.43	26.11	4.81	1.06	23.66	538.22	1.32	20.30
10^{-3}	25.56	1.99	37.40	8211.08	2.39	26.11	4.57	1.01	21.86	300.12	1.28	20.39
10^{-2}	25.38	1.95	37.26	4800.81	2.31	**26.10**	2.42	1.00	21.60	264.79	1.27	18.43
10^{-1}	9.81	**1.85**	**31.64**	3353.55	**2.30**	**26.10**	2.39	**0.97**	**20.12**	195.75	**1.17**	**16.54**

TABLE 8: Performance Comparisons on the S&P500 dataset

Algos	UBAH	Best	CRP	UP	EG	Anticor	ONS	CWMR	PAMR	OLMAR	RMR	WMAMR	EIIE	PPN-I	PPN
APV	1.21	89.63	1.21	1.22	1.22	1.09	1.41	0.99	1.34	17.81	17.77	1.02	99.35	129.27	**167.84**
SR(%)	12.03	10.37	11.69	11.70	11.73	5.85	36.34	-7.75	8.09	78.04	79.29	2.32	108.81	115.27	**148**
CR	2.39	1417.49	2.34	2.32	2.34	0.89	3.86	-0.98	3.68	208.32	207.84	0.16	1994.89	2135.87	**6792.51**
TO	0.021	0.022	0.033	0.034	0.033	0.099	0.291	0.011	0.115	1.972	1.970	0.119	1.925	1.918	1.920

the policy performance. Next, the Q value is also unsuitable, since the Q network is often hard to train regarding the non-stationary decision process [42]. Finally, the advantage value is still inappropriate, since its optimization relies on the accurate estimations of both state and Q values.

In conclusion, the value functions are inappropriate for PPN, due to the difficult approximation for portfolio selection. Hence, they may lead to biased policy gradients and worse performance of AC. On the contrary, DPG is guaranteed to obtain at least a sub-optimal solution as shown in Proposition 1, and helps to obtain better performance.

We next empirically evaluate AC algorithms on Crypto-A. We refer to the variant as PPN-AC, which is built upon Q values. Specifically, we adopt the DDPG algorithm [38] to optimize PPN-AC. The actor network in PPN-AC uses the same architecture as PPN, while the Q network and target Q network follow the network architecture in DDPG [38]. To better stabilize the training, we improve both Q networks with the dueling architecture mechanism [61].

We record the detailed results on the Crypto-A dataset in Table 9. To be specific, the performance of PPN-AC is far worse than PPN. This is because the Q network fails to approximate the Q value accurately, leading to biased policy gradients and worse performance. Although PPN-AC cannot achieve a satisfactory result, it still performs better than other baselines in Table 3. Such superiority mainly attributes to the strong representation ability of the actor network, *i.e.*, PPN. This further confirms the effectiveness of the two-stream architecture. In the future, we will continue to improve the task-specific deep reinforcement learning algorithm for portfolio selection.

TABLE 9: Evaluations of reinforcement learning algorithms for portfolio policy network on the Crypto-A dataset

Algos	APV	STD(%)	SR(%)	MMD(%)	CR
PPN-AC	11.72	2.73	4.60	60.25	17.79
PPN	32.04	2.16	6.85	38.86	79.87

7.3 Application to Stock Portfolio Selection

In previous experiments, we have demonstrated the effectiveness of the proposed methods on crypto-currency datasets. Here, we further evaluate our methods on the S&P500 stock dataset obtained from Kaggle[2], which is summarized in Table 10. All experimental settings and implementation details are the same as before, except that the results are averaged over 20 runs with random initialization seeds. The results in Table 8 further verify the effectiveness of the proposed method in terms of the superiority of reinforcement learning, and the importance of sequential feature learning and asset correlation extraction. Also, the results demonstrate the generalization ability of our method.

2. https://www.kaggle.com/camnugent/sandp500

TABLE 10: The statistics of the S&P500 dataset

Datasets	#Asset	Training Data		Testing Data	
		Data Range	Num.	Data Range	Num.
S&P500	506	2013-02 to 2017-08	1101	2017-08 to 2018-02	94

8 Conclusion

This paper has proposed a novel cost-sensitive portfolio policy network to solve the financial portfolio selection task. Specifically, by devising a new two-stream architecture, the proposed network is able to extract both price sequential patterns and asset correlations. Here, we show that both types of information are necessary for portfolio selection. In addition, to maximize the accumulated return while controlling both transaction and risk costs, we develop a new cost-sensitive reward function and adopt the direct policy gradient algorithm to optimize it. We theoretically analyze the near-optimality of the reward and show that the growth rate of the policy regarding this reward function can approach the theoretical optimum. We also empirically study the proposed method on real-world crypto-currency and stock datasets. Extensive experiments demonstrate its superiority in terms of profitability, cost-sensitivity and representation abilities. In the future, we will further discuss two general assumptions, and continue to improve the task-specific deep reinforcement learning method for better effectiveness, stability and interpretability, for example by exploring the correlation between social text information and price sequential information [62].

9 Acknowledgement

This work was partially supported by National Natural Science Foundation of China (NSFC) (61876208, 61502177 and 61602185), Guangdong Provincial Scientific and Technological Fund (2017B090901008, 2017A010101011, 2017B090910005, 2018B010107001), Pearl River S&T Nova Program of Guangzhou 201806010081, CCF-Tencent Open Research Fund RAGR20170105, Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183.

References

[1] M.Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, ane et al. Tensorflow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*, 2016.

[2] A. Agarwal, E. Hazan, S. Kale, R. E. Schapire. Algorithms for portfolio management based on the newton method. In *International Conference on Machine Learning*, 2006, pp. 9-16.

[3] P. H. Algoet, T. M. Cover. Asymptotic optimality and asymptotic equipartition properties of log-optimum investment, *The Annals of Probability*, 1988, vol. 16, no. 2, pp. 876-898.

[4] M. Babaioff, S. Dobzinski, S. Oren, A. Zohar. On bitcoin and red balloons. In *ACM Conference on Electronic Commerce*, 2012.

[5] S. Bai, J. Z. Kolter, V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[6] A. Borodin, R. El-Yaniv, V. Gogan. Can we learn to beat the best stock. In *Advances in Neural Information Processing Systems*, 2004.

[7] J. Bradbury, S. Merity, C. Xiong, R. Socher. Quasi-recurrent neural networks. *arXiv:1611.01576*, 2016.

[8] J. Cao, L. Mo, Y. Zhang, K. Jia, C. Shen, and M. Tan. Multi-marginal Wasserstein GAN. In *Advances in Neural Information Processing Systems*, 2019.

[9] W. Cao, C. Wang, L. Cao. Trading strategy based portfolio selection for actionable trading agents. In *International Workshop on Agents and Data Mining Interaction*, 2012, pp. 191-202.

[10] T. M. Cover, D. H. Gluss. Empirical Bayes stock market portfolios. *Advances in Applied Mathematics*, 1986.

[11] T. M. Cover, et al. Universal portfolios. *Mathematical Finance*, 1991.

[12] P. Das, N. Johnson, A. Banerjee. Online lazy updates for portfolio selection with transaction costs. In *AAAI*, 2013.

[13] P. Das, N. Johnson, A. Banerjee. Online portfolio selection with group sparsity. In *AAAI*, 2014, pp. 1185-1191.

[14] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[15] Y. Ding, W. Liu, J. Bian, D. Zhang, T.-Y. Liu. Investor-imitator: A framework for trading knowledge extraction. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[16] L. Fortnow, J. Kilian, D. M. Pennock, M. P. Wellman. Betting Boolean-style: a framework for trading in securities based on logical formulas. *Decision Support Systems*, 2005.

[17] L. Gao, W. Zhang. Weighted moving average passive aggressive algorithm for online portfolio selection. In *Intelligent Human-Machine Systems and Cybernetics*, 2013, pp. 327-330.

[18] C. L. Giles, S. Lawrence, A. C. Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning*, 2001, Vol. 44, No. 2, pp. 161-183.

[19] Y. Guo, X. Fu, M. Liu. Robust log-optimal strategy with reinforcement learning. *arXiv:1805.00205*, 2018.

[20] L. Györfi, I. Vajda. Growth optimal investment with transaction costs. In *International Conference on Algorithmic Learning Theory*, 2008, pp. 108-122.

[21] M. Heger. Consideration of risk in reinforcement learning. In *International Conference on Machine Learning*, 1994, pp. 105-111.

[22] H. Heidari, S. Lahaie, D. M. Pennock, J. W. Vaughan. Integrating market makers, limit orders, and continuous trade in prediction markets. *ACM Transactions on Economics and Computation*, 2018.

[23] D. P. Helmbold, R. E. Schapire, Y. Singer, M. K. Warmuth. On-Line portfolio selection using multiplicative updates. *Mathematical Finance*, 1998, Vol. 8, No. 4, pp. 325-347.

[24] S. Hochreiter, J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997, Vol. 9, No. 8, pp. 1735-1780.

[25] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 2004, Vol. 20, No. 1, pp. 5-10.

[26] D. Huang, Y. Zhu, B. Li, S. Zhou, S. C. Hoi. Robust median reversion strategy for on-Line portfolio selection. In *IEEE Transactions on Knowledge and Data Engineering*, 2013.

[27] D. Huang, J. Zhou, B. Li, S. C. Hoi, S. Zhou. Semi-universal portfolios with transaction costs. In *International Joint Conference on Artificial Intelligence*, 2015.

[28] Z. Jiang, D. Xu, J. Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv*, 2017.

[29] J. Kelly jr. A new interpretation of information rate. *Bell System Technical Journal*, 1956.

[30] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *Computer Vision and Pattern Recognition*, 2017.

[31] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. *Nature*, 2015.

[32] B. Li, S. C. Hoi, P. Zhao, V. Gopalkrishnan. Confidence weighted mean reversion strategy for online portfolio selection. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[33] B. Li, P. Zhao, S. C. Hoi, V. Gopalkrishnan. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 2012.

[34] B. Li, S. C. Hoi. On-line portfolio selection with moving average reversion. In *International Conference on Machine Learning*, 2012.

[35] B. Li, S. C. Hoi. Online portfolio selection: A survey. *ACM Computing Surveys*, 2014, Vol. 46, No. 3, pp. 35.

[36] B. Li, J. Wang, D. Huang, S. C. Hoi. Transaction cost optimization for online portfolio selection. *Quantitative Finance*, 2017.

[37] J. Li, K. Zhang, L. Chan. Independent factor reinforcement learning for portfolio management. In *International Conference on Intelligent Data Engineering and Automated Learning*, 2007, pp. 1020-1031.

[38] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, , ..., D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[39] J. Long, E. Shelhamer, T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, 2015.

[40] H. Markowitz. Portfolio selection. *The Journal of Finance*, 1952.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, ..., S. Petersen. Human-level control through deep reinforcement learning. *Nature*, 2015.

[42] J. Moody, M. Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 2001.

[43] J. Moody, L. Wu, Y. Liao, M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 1998, Vol. 17, pp. 441-470.

[44] J. Moody, M. Saffell, Y. Liao. Reinforcement learning for trading systems and portfolios. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 129-140.

[45] C. J. Neely, D. E. Rapach, J. Tu, G. Zhou. Forecasting the equity risk premium: the role of technical indicators. *Management Science*, 2014, Vol. 60, No. 7, pp. 1772-1791.

[46] R. Neuneier, O. Mihatsch. Risk sensitive reinforcement learning. In *Advances in Neural Information Processing Systems*, 1999.

[47] R. Neuneier. Optimal asset allocation using adaptive dynamic programming. In *Advances in Neural Information Processing Systems*, 1996, pp. 952-958.

[48] R. Neuneier. Enhancing Q-learning for optimal asset allocation. In *Advances in Neural Information Processing Systems*, 1998.

[49] M. Ormos, A. Urbán. Performance analysis of log-optimal portfolio strategies with transaction costs. *Quantitative Finance*, 2013.

[50] G. Ottucsák, I. Vajda. An analysis of the mean-variance portfolio selection. *Statistics and Decisions*, 2007.

[51] O. Schrijvers, J. Bonneau, D. Boneh, T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, 2016.

[52] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[53] W. Shen, J. Wang. Transaction costs-aware portfolio optimization via fast Lowner-John ellipsoid approximation. In *AAAI*, 2015.

[54] W. Shen, J. Wang. Portfolio Selection via Subset Resampling. In *AAAI*, 2017, pp. 1517-1523.

[55] W. Shen, B. Wang, B. Pu, J. Wang. The Kelly growth optimal portfolio with ensemble learning. In *AAAI*, 2019.

[56] S. Sikdar, S. Adali, L. Xia. Top-trading-cycles mechanisms with acceptable bundles. In *AAAI*, 2018.

[57] I. Sutskever, O. Vinyals, Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[58] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000, pp. 1057-1063.

[59] I. Vajda. Analysis of semi-log-optimal investment strategies. In *Prague Stochastics*, 2006, pp. 719-727.

[60] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, ..., K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *ISCA Speech Synthesis Workshop*, 2016.

[61] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas. Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning*, 2016.

[62] H. Wu, W. Zhang, W. Shen, J. Wang. Hybrid deep sequential modeling for social text-driven stock prediction. In *ACM International Conference on Information and Knowledge Management*, 2018.

[63] F. Yu, V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[64] H. Zhao, Q. Liu, G. Wang, Y. Ge, E. Chen. Portfolio selections in P2P lending: A multi-objective perspective. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[65] P. Zhao, Y. Zhang, M. Wu, S. C. Hoi, M. Tan, J. Huang. Adaptive cost-sensitive online classification. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[66] Y. Zhang, P. Zhao, J. Cao, W. Ma, J. Huang, Q. Wu, M. Tan. Online adaptive asymmetric active learning for budgeted imbalanced data. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018. pp. 2768-2777.

[67] Y. Zhang, Y. Wei, P. Zhao, S. Niu, Q. Wu, M. Tan, J. Huang. Collaborative unsupervised domain adaptation for medical image diagnosis. In *Medical Imaging meets NeurIPS*, 2019.

[68] Y. Zhang, H. Chen, Y. Wei, P. Zhao, J. Cao, and el al.. From whole slide imaging to microscopy: Deep microscopy adaptation network for histopathology cancer image classification. In *Medical Image Computing and Computer Assisted Intervention*, 2019.

[69] Y. Zhang, G. Shu, Y. Li. Strategy-updating depending on local environment enhances cooperation in prisoner's dilemma game. *Applied Mathematics and Computation*, 2017, Vol. 301, pp. 224-232.

[70] H. Zhong, C. Liu, J. Zhong, H. Xiong. Which startup to invest in: a personalized portfolio strategy. *Annals of Operations Research*, 2018.

Yifan Zhang is working toward the M.E. degree in the School of Software Engineering, South China University of Technology, China. He received the B.E. degree in electronic commerce from the Southwest University, China, in 2017. His research interests include machine learning, data mining, and their applications in data limited problems and decision-making tasks.

Peilin Zhao is currently a Principal Researcher at Tencent AI Lab, China. Previously, he has worked at Rutgers University, Institute for Infocomm Research (I2R), Ant Financial Services Group. His research interests include: Online Learning, Deep Learning, Recommendation System, Automatic Machine Learning, etc. He has published over 90 papers in top venues, including JMLR, ICML, KDD, etc. He has been invited as a PC member, reviewer or editor for many international conferences and journals, such as ICML, JMLR, etc. He received his bachelor degree from Zhejiang University, and his PHD degree from Nanyang Technological University.

Bin Li received the bachelor's degree in computer science from the Huazhong University of Science and Technology and the bachelor's degree in economics from Wuhan University in 2006, and the PhD degree from the School of Computer Engineering at Nanyang Technological University in 2013. He is currently an associate professor in the Department of Finance, Economics and Management School at Wuhan University, Wuhan, China. He was a postdoctoral research staff in the Nanyang Business School at Nanyang Technological University, Singapore. His research interests are quantitative investment, computational finance, and machine learning.

Junzhou Huang is an Associate Professor in the Computer Science and Engineering department at the University of Texas at Arlington. He received the B.E. degree from Huazhong University of Science and Technology, China, the M.S. degree from Chinese Academy of Sciences, China, and the Ph.D. degree in Rutgers university. His major research interests include machine learning, computer vision and imaging informatics. He was selected as one of the 10 emerging leaders in multimedia and signal processing by the IBM T.J. Watson Research Center in 2010. He received the NSF CAREER Award in 2016.

Qingyao Wu received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2013. He was a Post-Doctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore, from 2014 to 2015. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His current research interests include machine learning, data mining, big data research.

Mingkui Tan is currently a professor with the School of Software Engineering at South China University of Technology. He received his Bachelor Degree in Environmental Science and Engineering in 2006 and Master degree in Control Science and Engineering in 2009, both from Hunan University in Changsha, China. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore, in 2014. From 2014-2016, he worked as a Senior Research Associate on computer vision in the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning and large-scale optimization.