Deep Reinforcement Learning for Optimizing Finance Portfolio Management

Yuh-Jong Hu and Shang-Jen Lin

Dept. of Computer Science,
National Chengchi University, Taipei, Taiwan hu@cs.nccu.edu.tw 106971001@nccu.edu.tw

Abstract: Deep reinforcement learning (DRL) is an emerging artificial intelligence (AI) research field which combines deep learning (DL) for policy optimization and reinforcement learning (RL) for goal-oriented self-learning without human intervention. We address major research issues of policy optimization for finance portfolio management. First, we explore one of the deep recurrent neural network (RNN) models, GRUs, to decide the influences of earlier states and actions on policy optimization in non-Markov decision processes. Then, we craft for a viable risk-adjusted reward function to evaluate the expected total rewards for policy. Third, we empower the integration of RL and DL to leverage their respective capabilities to discover an optimal policy. Fourth, we investigate each type of RL approaches for integrating with the DL method while solving the policy optimization problem.

Keywords: Artificial intelligence (AI), deep reinforcement learning (DRL), deep learning (DL), neural nets (NNs) reinforcement learning (RL), policy gradient, value/policy iteration, Qlearning, deep RNNs, finance portfolio management

I. INTRODUCTION

The term 'artificial intelligence' (AI) was first coined in the 1950s and rose in popularity in the 1990s but then faded away. Two major branches of AI are computational logicbased deductive reasoning, such as the Semantic Web, and machine learning-oriented inductive reasoning, such as big data analytics. The Semantic Web proposed in the 2000s was an ultimate goal of the original Web developer, Tim BernersLee, but it turned out the Web was moving towards the Social Web. Therefore, the combination of ontology and rule-based Semantic Web did not really achieve its initial dream. On the contrary, AI, with its subfield machine learning, has been progressing rapidly. In recent years, big data analytics was significantly developed because data are divulged everywhere.

Deep reinforcement learning (DRL) is a type of machine learning algorithm that combines deep learning (DL) with reinforcement learning (RL) for development of various intelligent self-learning applications [1]. DRL has been demonstrated in several successful examples of real-world applications, such as robotic motion planning, AlphaZero, and self-driving cars, etc. For the last few decades, some studies have been proposed for optimizing finance portfolio management based on reinforcement learning (RL) or deep learning (DL) [2] [3]. However, only a few prominent examples have been proposed for optimizing portfolio management by applying the DRL in finance [4] [5].

Online portfolio selection is an important research area from a financial engineering viewpoint [6]. Several challenges still exist when applying DRL in optimizing finance portfolio management. First, it is impossible to obtain a real state space of the finance world, so we have to use partially observable Markov decision processes (POMDPs) to approximate its state space. In addition, we need to deal with the non-Markovian property with dependence on earlier states and actions to learn and estimate future expected rewards. So a viable deep recurrent neural network (RNN) should be used to weigh in the influences of earlier states and actions on searching an optimal policy. Moreover, a risk-adjusted reward function, such as the Sharpe ration, should be crafted for searching for an optimal policy. This is quite different from the other DRL applications, where a risk-adjusted reward function is not so complicated. Besides, transaction overheads, such as transaction fees and tax, should be considered when computing the risk-adjusted reward function to obtain total effective rewards.

Second, we have to empower the integration of the DL and RL algorithms to leverage their respective capabilities to discover an optimal finance planning policy. The ultimate goal is to derive a maximal total (or average) reward in an asset allocation process.

Third, two major RL algorithms have been used for evaluating an optimal policy, including value/policy iteration and policy gradient. Should we use a value/policy iteration method, such as off-policy Q-learning or on-policy TD learning, and combine it with a DL algorithm through dynamically updating a policy function to estimate the maximal expected shortterm immediate rewards? Or instead, should we use a policy gradient method and combine it with a DL algorithm to estimate its maximal long-term return for an asset allocation episode? Moreover, how can we use a hybrid actor-critic (AC) method and combine it with a DL algorithm while optimizing an optimal policy?

We discuss various DR and RL combinations, then we propose one of the DRL approaches and argue why this one is better

978-1-5386-9346-9/19/$31.00 ©2019 IEEE

than previous simple DL or RL methods for solving the optimizing finance problem.

A. Research Issues and Contributions

This study addresses the following major *research issues*:

1) How should a deep recurrent neural network (RNN) be used for weighing the influences of earlier states and actions in an evaluation of future expected rewards? Moreover, how can a risk-adjusted reward function be crafted for searching for an optimal policy?

2) How can we empower the integration of the DL and RL algorithms and leverage their respective capabilities to discover an optimal portfolio management policy?

3) Three types of RL methods are available for integrating with DL algorithms to become one of the DRL algorithms. Which DRL approach is the most appropriate to address the optimal asset allocation problem?

This study makes the following four contributions: First, we exploit one of the RNN models on learning the influences of earlier states and actions to estimate the total rewards. In addition, we discuss several risk-adjusted reward functions and craft for one of the approaches in the policy gradient method to evaluate the expected total rewards of finance planning policy. Then, one DRL algorithm is proposed to explicitly leverage its capabilities for optimizing portfolio management. Third, we apply one of the RL methods and combine with a DL algorithm to search for an optimal policy.

This study is organized as follows: In Section I, we give an introduction. Then, we provide some background knowledge in Section II. In Section III, we address related work. In Section IV, we investigate current RL and DL techniques for learning finance planning and argue why it is important to learn a policy function through DRL techniques. In Section V, we explore several types of RL and DL combinations. In Section VI, we point out how to empower incentives for integration of RL with DL in optimizing finance policy. Finally, in Section VII, we conclude this paper and show possible future research possibilities.

II. BACKGROUND

Machine learning algorithms have been developed for building a model to perform clustering, classification, and causeeffect analytics. Moreover, we use training and validation techniques to obtain an optimal model with balancing biasvariance for unknown labels classification or future values prediction. Recently, machine learning techniques have been extended to apply to the finance domain, such as FinTech. In fact, big data analaytics with its machine learning algorithm has been developed for numerous financial applications, such as asset-price prediction, risk analysis, and program trading, etc. Among them, financial machine learning is a type of emerging finance analytics service for portfolio management [7]. However, financial machine learning is not quite acceptable by traditional P- (or Q-) Quants finance experts. Moreover, it is not easy for an ordinary data scientist to deal with finance analytics for the following reasons.

First, we need a white-box but not a black-box when using machine learning inferencing in finance analytics because portfolio investors usually need a detailed picture to infer the important features (or states) and their weights while restructuring their portfolio ratio. Second, it is not easy to pick an appropriate machine learning algorithm with a proper objective function to ensure that financial machine learning is useful and effective. Third, it is usually too optimistic when we use a back-testing model to predict future asset prices because states input for a model derived from highdimensional features do not completely repeat themselves so learning from the past sometimes is not so effective for the future. This makes the goal of high accuracy of asset price prediction hard to achieve. Finally, financial time-series data are not independent identically distributed (iid). This excludes using randomized cross-validation (CV) in training and validation learning phases to obtain an optimal asset allocation model's parameters.

III. RELATED WORK

Financial supervised learning is used for classification (or regression) that only provides forecasting but does not directly have asset trading and allocation actions [7]. Therefore, we need to optimize two sets of parameters for different purposes, one is for forecasting and the other is for trading actions. Using two sets of parameters, a model evaluates profits or losses by executing a specific trading action after prediction. However, information bottleneck prevents effectively linking these two systems together [3]. Training a neural network (NN) model or deep NNs can be used for stock price prediction, trading, and asset allocation. It faces the same problem of not having the ability to trade assets directly [2] [8].

RL has been used for direct asset trading and portfolio management for almost two decades [9]. Direct recurrent policy learning applies the policy gradient method for non-Markovian decision processes (MDP) with sampling observations to approximate real financial world state space. Stochastic dynamic programming uses the value or policy iteration method, such as Q-learning for optimal asset allocation [10] [11].

When we face multiple data sources with high-dimensional features to define the state space of RL, conventional feature engineering with limited state and action space representation is not feasible for optimal policy learning. Integration of DL with RL, as DRL, is a novel approach to enhance standard feature engineering which has only limited state and action space. DRL has been used for solving a few asset allocation

problems [4] [5] [12]. However, they did not really address the research challenge issues of DRL that we propose.

IV. DRL FOR OPTIMIZING FINANCE

RL has been developed for several decades [13]. The structure of RL includes state space, action space, transition probability, reward function, policy function, and a world model. An agent observes the state of a model-free or modelbased world. Then, the agent uses this collected state in the world to search for an action with a maximal expected immediate reward or a long-term return. The agent has a policy function to decide which state to choose and which action can achieve this goal. A state will transit to another state with a transition probability. A standard RL is a Markov decision process, where an agent learns the best action in a policy function only depends only on the most recent one-period time of state and action to maximize its expected rewards.

In a portfolio management of finance, state space can be defined as a combination of stock market status, current asset portfolio ratio, and sentiment of the future finance market, etc. A full state space is impossible to obtain from unknown highdimensional features of the financial world, so it is usually approximated by using observable samples to simplify its state-space representation [14].

Action space is defined as a trading decision to reallocate each asset's ratio in an asset bucket through buy, sell, or hold action. A fixed amount of asset trading is designated as discrete action space and a variable amount of asset trading is designated as continuous action space. Transition probability is a probability of moving from the current state with an action into an immediate future state. The reward function is known and fixed with risk-adjusted profit. The policy function can be extended to have a non-Markovian property, which includes the most recent past n-period time of states and actions sequence to learn one or multiple future m-period time of expected rewards [15].

RL is self-learning, which is different from supervised learning for classification (or regression) and unsupervised learning for clustering. A supervised learning model learns each instance's classification (or regression) outputs from pre-selected input features with the smallest error rate. RL proceeds in selflearning by maximizing the expected rewards by searching for an optimal policy function with the selected action for each state. Thus, RL uses the environment's feedback-reward signal to decide whether the selected action is good or bad.

Standard RL is classified as one of two types: dynamic programming or direct policy optimization. These two approaches have been used in optimizing portfolio management for almost two decades [9] [10]. In this study, we explore each type of RL and aim at using the direct policy gradient method combines with deep DL, that is, deep RNNs technique, for optimizing asset allocation.

A. How can we use DL for Optimizing Finance?

Neural Nets (NNs) have been classified as convolutional NNs (CNNs) and recurrent NNs (RNNs). NNs were used for each stock price prediction and portfolio weight recomputation [2]. Two layers of NNs modules, a CNNs-based prediction module and a RNNs-based trading decision module, were proposed to predict each stock price and decide each asset reallocation action. First, the prediction module was used to predict each stock's price with a mean squared error (MSE) criterion. Then, MSE outputs were used to classify a stock asset into one of three categories: good, neutral, or bad. Following the prediction module was a trading-decision module that uses MSE outputs to decide each asset's reallocation weight in each decision epoch. The training criterion was chosen for a discrete sequence of asset weight in the portfolio by the trading-decision module, unfolded in time, to maximize rewards.

DL, an extended version of the original NNs, is further classified as deep CNNs and deep RNNs [16]. Recently, deep CNNs research is resurgent because of its success at image recognition in the ImageNet competition [17]. The features of a high-dimensional dataset can be directly fed into multi-layer deep CNNs, and optimizing the weight parameters among neurons through backpropagation optimizer. This DL backpropagation optimizer is a black-box, not a white-box, so it cannot explain the reasons behind the scene. Original RNNs do not have the capacity to consume high-dimensional features from financial structured and unstructured datasets. We need to apply deep RNNs to learn each model's optimal (hyper-) parameters in finance time-series data [8]. However, deep RNNs still do not have the self-learning capability to learn the optimal policy function for each state and action responding to a reward with respect to the financial signal so we need RL [12].

B. How can we use RL for Optimizing Finance?

Recurrent RL (RRL) and stochastic direct reinforcement (SDR) are two types of direct policy gradient methods to optimize a recurrent policy function and a stochastic function for the trading systems [9]. The authors have claimed that the best asset allocation strategy by using RRL and SDR can avoid the burden of computating numerical value problems shown in dynamic programming. Moreover, searching for optimal decision actions can be solved simultaneously for structural credit assignment of each asset ratio and temporal credit assignment of each time step. Therefore, the best asset allocation strategy can be found during the entire finance planning episode.

In the financial world, it is impossible to obtain full-state information to learn an optimal RRL model, so instead the POMDP is used for state-space approximation. Besides, a non-Markovian decision processes is more appropriate for learning an optimal policy function by using the most recent n-period

time of states and actions to predict future expected rewards [18].

1) RRL Optimization [9]: A finance portfolio agent learns a

recurrent policy function $\pi(\theta_t;P_{t-1},O_t)$ by varying θ_t, where O_t is the collected observation samples in POMDP. The goal of RRL is to maximize the reward (or utility) function $U_T = U(R_t,R_{t-1},...,R_1)$ through the deterministic gradient method with batch analytics shown as:

$$\frac{dU_T(\theta)}{d\theta} = \sum_{t=1}^{T} \frac{dU_T}{dR_t}\{\frac{dR_t}{d\pi_t}\frac{d\pi_t}{d\theta} + \frac{dR_t}{d\pi_{t-1}}\frac{d\pi_{t-1}}{d\theta}\}$$

with *recursion policy function* (see equation(1)):

$$\frac{d\pi_t}{d\theta} = \frac{d\pi_t}{d\theta} + \frac{d\pi_t}{d\pi_{t-1}}\frac{d\pi_{t-1}}{d\theta} \quad (1)$$

Learning parameter update with on-line learning rate ρ depends on the entire sequence of previous trades: $\delta\theta = \rho^{\frac{dU}{d\theta^{t(\theta t)}}}$ The stochastic gradient method with on-line analytics is shown as:

$$\frac{dU_t(\theta)}{d\theta_t} = \frac{dU_t}{dR_t}\{\frac{dR_t}{d\pi_t}\frac{d\pi_t}{d\theta_t} + \frac{dR_t}{d\pi_{t-1}}\frac{d\pi_{t-1}}{d\theta_{t-1}}\}$$

with *recursion policy function*(see equation(2)):

$$\frac{d\pi_t}{d\theta_t} = \frac{d\pi_t}{d\theta_t} + \frac{d\pi_t}{d\pi_{t-1}}\frac{d\pi_{t-1}}{d\theta_{t-1}} \quad (2)$$

The learning parameter update with on-line learning rate ρ depends on previous one-step trade:$\delta\theta_t = \rho\frac{dU_t(\theta_t)}{d\theta_t}$

2) SDR Optimization [9]: A finance portfolio agent learns stochastic policy function π with inputs based on the most recent n past *observations* o and m past *actions* a as:

$$\pi_\theta(a_t \mid H_{t-1}) = \pi_\theta(a_t; O_{t-1}^{(n)}, A_{t-1}^{(m)})$$

where$O_{t-1}^{(n)} = \{o_{t-1}, o_{t-2}, ..., o_{t-n}\}$ (standard memory)

$A_{t-1}^{(m)} = \{a_{t-1}, a_{t-2}, ..., a_{t-m}\}$ (recurrent memory)

The goal of SDR is to maximize the *expected total utility* of a future sequence of T horizon time step of actions:

$$\max_\theta U_T(\theta) = \max_\theta E_{H_T(\theta)} \sum_{t=1}^{T}\sum_{a_t} R_t(s_t,a_t)\pi_\theta(a_t \mid H_{t-1})\pi_\theta(H_{t-1}) \quad (3)$$

where $R_t(s_t,a_t)$ is a reward function, and $\pi_\theta(a_t \mid H_{t-1})$ is a policy function with conditional probability based on H_{t-1} history of a sequence of observations and actions at time step $t-1$, where $t = 1,...,T$, and via the direct gradient ascent optimization of utility function, we have:

$$\nabla\theta \propto \frac{dU_T(\theta)}{d\theta} = E_{H_T(\theta)} \sum_t \sum_{a_t} u_t(a_t)\frac{d}{d\theta}\pi_\theta(a_t \mid H_{t-1})\pi_\theta(H_{t-1}) \quad (4)$$

Evaluate $\frac{d}{d\theta}\pi_\theta(a_t) = E_{H_{t-1}}\frac{d}{d\theta}\pi_\theta(a_t \mid H_{t-1})\pi_\theta(H_{t-1})$ for a policy function$\pi_\theta(a_t \mid H_{t-1}) = \pi_\theta(a_t; O_{t-1}^{(n)}, A_{t-1}^{(m)})$, where (n,m) $H_t = (O_t,A_t)$ is a complete history, whereas $^{H_t} = (O_t^{(n)}, A_t^{(m)})$ is a partial history of length (n,m) observations and actions.

3) Risk-Adjusted Reward Functions: A finance portfolio agent needs to consider the risk factor in the reward function for each time-step's reward. From the early *Markowitz mean-variance* framework to the recent *Sharpe ratio*, $S_T = \frac{Average(R_t)}{SD(R_t)}$, where average return $Average(R_t)$ and standard deviation of return $SD(R_t)$ are estimated for periods $t = \{1,...,T\}$. In on-line learning, we consider using the *differential Sharpe ratio* [3]:

$$D_t = \frac{dS_t}{d\eta} \quad (5)$$

where η is the hyper-parameter which controls the magnitude of influence of the return R_t on the Sharpe ratio S_t by exponential moving averages of returns and standard deviation of returns. In addition, the *Sterling ratio* is not easy to compute, so the downside deviation ratio $DDR_T = \frac{Average(R_t)}{DD_T}$ is used to encourage upward increasing return. Differential DDR_T is defined as ∇D_t (see equation (6)):

$$\nabla D_t \equiv \frac{dDDR_t}{d\eta}, \quad DD_T = \sqrt{(\frac{1}{T}\sum_{t=1}^{T} min\{R_t, 0\}^2)} \quad (6)$$

C. Why should we use DRL for Optimizing Finance?

DRL is DL integrated with RL, where DL includes deep CNNs and deep RNNs [17]; RL includes value/policy iteration, policy gradient, and actor-critic methods [13]. DRL has been successfully applied to computer games, Go, chess, and self-driving cars. But DRL is still at an emerging stage in its application for finance, such as FinTech. Only a few studies have been conducted about how to apply DRL for optimizing finance, in particular, portfolio management [5].

When compared with existing DRL applications, we face several research challenges when applying DRL for optimizing portfolio management (see Section I).

Possible incentives for an agent to use DRL on resolving the above research challenge issues are (1) The agent leverages DL's capability to learn an optimal policy function $\pi_\theta(s,a)$'s parameter θ for input state created from high-dimensional features in data of the finance world. (2) The agent uses an optimal function $\pi_\theta(s,a)$, generated from DL, and takes a sequence of actions to the environment , that is, in the finance real world, to obtain an expected reward through RL's optimal action selection on planning an immediate time-step or an entire episode of asset allocation processes (see Figure 1).

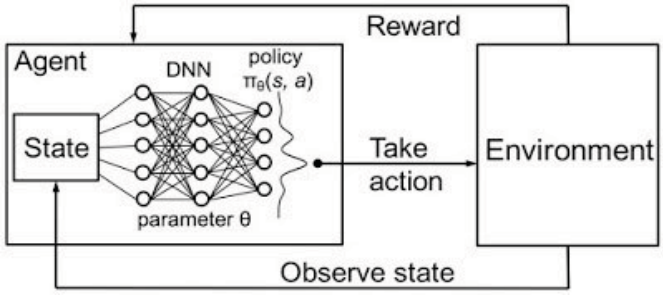

Fig. 1. An agent uses DRL to search for an optimal policy function $\pi_\theta(s,a)$ for each observable state s, and takes an action a to this environment to obtain a maximal expected reward.

V. TYPES OF DL+RL AS DRL FOR OPTIMIZING FINANCE

In this study, DL is selected from one of the deep RNNs algorithms, namely, gated recurrent unit (GRU) [16]. GRUs are an improved version of standard RNNs with update gates and reset gates to solve the gradient vanishing problem (see Figure 2). These two gate vectors decide what information should be passed to the output. In a training phase of backtesting, the deep RNN (GRU) algorithm automatically propagates the past n-step of states and actions on learning the influences to predict future one-step or m-step of an episode portfolio's expected rewards. When enforcing DL+RL as DRL, two types of policy optimization techniques are available in the RL landscape: dynamic programming and policy gradients.

A. Dynamic Programming vs. Policy Gradients

In dynamic programming, the value iteration of the Qlearning function $Q_\theta(s,a)$ is learned by using a tabular method to enumerate all of the possible states and actions of Qvalues to find the optimal Q-value for $Q_\theta(s,a)$. In reality, we cannot possibly learn every state and action combination, so a parametrized Q function, shown as a linear function in features, is used through sampling states and actions from observable state space.

In off-policy learning, such as Q-learning, two sampling processes are available for action a and state s. In action sampling, one is *exploration* by random action chosen strategy, and the other is *exploitation* by choosing an action that maximizes the Q-learning function. In state sampling, the next state chosen can be stochastic or deterministic through transition probability.

As for policy iteration, on the one hand, we have *policy evaluation* of the current policy π_k, which iterates until convergence. On the other hand, we have policy improvement to find the best action based on one-step look-ahead.

$TD(\lambda)$-learning, such as SARSA, is on-policy learning, where policy evaluation and policy improvement are enforced in the same policy [13]. This is different from off-policy learning, such as Q-learning, where policy evaluation and policy improvement are separately enforced in different policies.

In the policy gradients approach, we directly find the best policy through a derivative of a policy function for an episode [19]. Policy gradient is on-policy learning. It is more compatible with rich architecture, including the recurrence relation of policy function. This is different from the dynamic programming, which is more compatible with exploration with off-policy learning.

B. DL + Value/Policy Iteration of RL for Finance

Q-learning has been used for searching for an optimal asset allocation for two decades [11]. Recently, stochastic dynamic programming was used for dynamic asset allocation [10]. Optimal exact Q-values iteration should obey the Bellman equation shown as follows (see equation (7)):

$$Q^*(s,a) = E_{s'}\left[r + \gamma \max_{a'} Q(s',a')^* | s,a\right] \tag{7}$$

where $r + \gamma \max Q(s^0,a^0,\theta)$ is a target function with r as $a0$ the immediate reward. The Q-learning function minimizes its mean square error (MSE) by stochastic gradient descent:

$$MSE = \left(r + \gamma \max_{a'} Q(s',a',w) - Q(s,a,w)\right)^2 \tag{8}$$

Deep Q-network (DQN) learning represents the value iteration of the Q-learning function by a parametrized Q-network with weights θ between neurons in deep RNNs. The Qlearning function converges using table lookup, but it diverges using NNs because of the problems of correlations between samples and non-stationary target function[1]. The DQN agent replays sample experiences and can remove samples' correlations, and deal with non-stationary target functions by fixed parameters w in target function as in equation (8) .

C. DL + Policy Gradient of RL for Finance

Instead of a parametrized Q-learning value function with deep RNNs, we can directly parametrize the policy function with DL, and searching for total rewards of an episode[2]. Let τ

[1] *Deep Reinforcement Learning, D. Silver, Google DeepMind, 2015.*
[2] *Deep RL Bootcamp., 26-27 August, 2017, Berkeley CA, USA, 2017*

denotes an episode of state-action sequence $(s_0,a_0),\ldots,(s_N,a_N)$ with joint probability $P(\tau;\theta)$ and $R(\tau) = \sum_{t=0}^{N} R(s_t,a_t)$ denotes the total return of the reward function in the finance planning episode. The maximum return (or total rewards) for a policy's episode is shown as:

$$\max_{\theta} U(\theta) = \max_{\theta} E[\sum_{t=0}^{N} R(s_t,a_t);\pi_\theta] = \max_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_{\tau} P(\tau;\theta)R(\tau) = \sum_{\tau} \nabla_\theta P(\tau;\theta)R(\tau) \quad (9)$$

After decomposing the planning path into states and actions, we have equation (10), which can be optimized θ by the RNNs *backpropagation with time* for each path i of a policy function $\pi_\theta(a_t^{(i)}|s_t^{(i)})$. The dynamics model is ruled out in the equation because there is no weight θ for the transition probability. When we plug in equation (10) into equation (9), we have a maximal total return of an episode through policy gradient.

$$\nabla_\theta P(\tau^{(i)};\theta) = \nabla_\theta \log \Big[\Pi_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)})}_{\text{a finance world model}} \cdot \underbrace{\pi_\theta(a_t^{(i)} \mid s_t^{(i)})}_{\text{policy function}}\Big]$$

$$= \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \quad (10)$$

where a finance world model is not required because there is no θ parameter inside so backpropagation with time optimization is applied to the $\log\pi_\theta(\cdot)$ policy function only.

D. DL + Actor-Critic of RL for Finance

An actor-critic (AC) algorithm is a hybrid combination of value iteration and policy gradient. Value iteration plays the role of the critic and policy gradient plays the role of the actor. For example, we estimate the Q-value reward function by using $Q(s,a,w) = Q^{\pi}(s,a)$, as shown in equation (7).

This is followed by updating the policy function $\pi_\theta(a_t^{(i)}|s_t^{(i)})$ parameters θ through the stochastic gradient descent method for each time-step of s_t with its optimal action a_t.

VI. EMPOWERMENT OF DRL IN OPTIMIZING FINANCE

How to leverage the power of the DL and RL combination to optimize the learning model's parameters θ is a research challenge. In this study, the agent uses deep RNNs (GRUs) for training an optimal policy function $\pi_\theta(a_t|s_t)$. GRUs can consume a state denoted as an output created from very highdimensional features, including current portfolio ration of each asset, well-known stock trading indices, stocks price variation and trading volume, and positive/negative sentiment about future stock markets, etc.

When a real state space cannot be obtained, sampling observable data features as a state is an answer to approach this problem. Deep RNNs (GRUs) can automatically learn its parameter values θ based on the history of states and actions. This non-MDP with deep RNNs (GRUs) allows learning parameters in a model with variable length of the influences of past states and action [18]. This policy network direct optimization's capability through GRUs cannot be achieved by SDR alone with its fixed length of partial history of states and actions, as shown in equation (4).

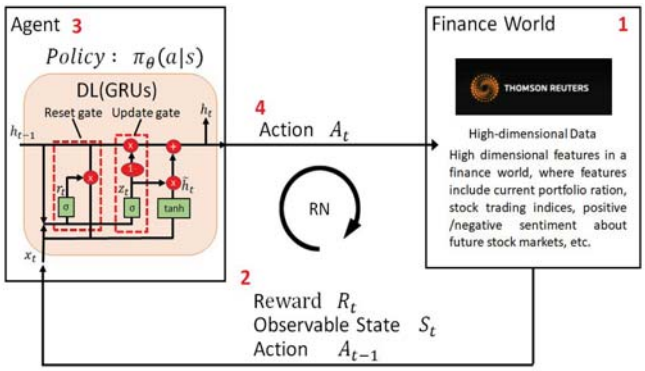

Fig. 2. An agent uses deep RNNs (GRUs) to search for an optimal policy function $\pi_\theta(a_t|s_t)$ of each observable state s_t, and takes an action a_t to this environment to obtain a maximal expected immediate reward.

The agent selects an action that has maximal expected longterm rewards in an episode of the finance planning horizon. The action space consists of *continuous actions* with *sell, buy*, and *hold* for a ration of each asset. This goal cannot be reached by SDR optimization in equation (4), which only has a discrete action space with a fixed ratio of asset trading. Finally, the risk-adjusted reward function is proposed in equations (5) and (6). These risk-adjusted reward functions are allowed to be embedded in the equation (9) for an entire episode.

VII. CONCLUSION AND FUTURE WORK

Original NNs and RL algorithms are still lack the capability on pre-processing states from high-dimensional features and trading portfolio assets with actions with respect to optimize RL's policy function. Therefore, we propose a type of DL and RL combination - known as DRL - for optimizing finance. On the one hand, deep RNNs in DL can automatically keep track of past states and actions history and learn their influences in a model on future expected rewards of an episode. On the other hand, value/policy iteration or policy gradient RL methods can be integrated with deep RNNs (GRUs) to deal with the optimal trading actions searching problem.

We propose using deep RNNs (GRUs) for DL and policy gradient for RL to search for the optimal policy function's parameters. Mathematical formulae have been explicitly demonstrated based on the results of previous and recent other research results. In our future work, we will investigate all types of DL and RL combinations, find the best one, and discover its incentives for finance planning. A scenario of optimizing portfolio management has been implemented in the well-known PyTorch and Open Gym platforms. Empirical results will be available in the near future.

REFERENCES

[1] Y. Li, "Deep reinforcement learning: an overview," no. arXiv:1701.07274v6 [cs.LG], Jan. 2017. [Online]. Available: https://arxiv.org/abs/1701.07274

[2] Y. Bengio, "Training a neural network with a financial criterion rather than a prediction criterion," in *Proc. of the 4th Int. Conf. on Neural Networks in the Capital Markets*, ser. Progress in Neural Processing. World Scientific, 1996, pp. 36–48.

[3] J. Moody *et al.*, "Performance functions and reinforcement learning for trading systems and portfolios," *Journal of Forecasting*, vol. 17, pp. 441–470, 1998.

[4] Z. Jiang, "A deep reinforcement learning framework for the financial portfolio management problem," no. arXiv:1706.10059v2 [q-fin.CP], July 2017. [Online]. Available: https://arxiv.org/abs/1706.10059

[5] Z. Liang *et al.*, "Deep reinforcement learning in portfolio management," no. arXiv:1808.09940v3 [q-fin.PM], Nov. 2018. [Online]. Available: https://arxiv.org/abs/1808.09940

[6] B. Li and C. H. S. Hoi, "Online portfolio selection: A survey," *ACM Computing Survey*, no. 3, pp. 35:1–35:36, 2014.

[7] D. L. M. Prado, *Advances in Financial Machine Learning*. Wiley, 2018. [Online]. Available: http://www.quantresearch.info/index.html

[8] B. J. Heaton *et al.*, "Deep learning in finance," no. arXiv:1602.06561v3 [cs.LG], Jan. 2018. [Online]. Available: https://arxiv.org/abs/1602.06561

[9] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. on Neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[10] G. Infanger, "Dynamic asset allocation strategies using a stochastic dynamic programming approach," in *Handbook of Asset and Liability Management*. Elsevier, 2008.

[11] R. Neuneier, "Enhancing q-learning for optimal asset allocation," in *Proc. of the 10th International Conference on Neural Information Processing Systems*, ser. NIPS'97. MIT Press, 1997, pp. 936–942.

[12] Y. Deng *et al.*, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.

[13] S. R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2017.

[14] T. Jaakkola, P. S. Singh, and I. M. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," in *Proc. of the 7th Int. Conf. on Neural Information Processing Systems*, ser. NIPS'94. MIT Press, 1994, pp. 345–352.

[15] J. Moody *et al.*, "Stochastic direct reinforcement: Applications to simple games with recurrence," in *Artificial Multiagent Learning*. AAAI Press, 2004.

[16] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016, http://www. deeplearningbook.org.

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

[18] J. Perez and T. Silander, "Non-markovian control with gated end-to-end memory policy networks," no. arXiv:1705.10993v1 [stat.ML], May 2017. [Online]. Available: https://arxiv.org/abs/1705.10993

[19] R. Sutton *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*, ser. NIPS'00. MIT Press, 2000, pp. 1057–1063.