Expert Systems With Applications 140 (2020) 112872

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading

Kai Lei[a,b], Bing Zhang[a], Yu Li[a], Min Yang[c], Ying Shen[a,*]

[a] Shenzhen Key Lab for Information Centric Networking & Blockchain Technology (ICNLAB), School of Electronics and Computer Engineering (SECE), Peking University, Shenzhen 518055, China
[b] PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
[c] Shenzhen Institutes of Advanced Technology, University of the Chinese Academy of Sciences, China

A R T I C L E I N F O

Article history:
Received 24 January 2019
Revised 12 August 2019
Accepted 13 August 2019
Available online 14 August 2019

Keywords:
Algorithmic trading
Deep reinforcement learning
Gate
Temporal attention

A B S T R A C T

Algorithmic trading is a continuous perception and decision making problem, where environment perception requires to learn feature representation from highly nonstationary and noisy financial time series, and decision making requires the algorithm to explore the environment and simultaneously make correct decisions in an online manner without any supervised information. To address these two problems, we propose a time-driven feature-aware jointly deep reinforcement learning model (TFJ-DRL) that integrates deep learning model and reinforcement learning model to improve the financial signal representation learning and action decision making in algorithmic trading. Concretely, we learn the environmental representation by adaptively selecting and reweighting various features of financial signals and summarize the attention values between historical information and changing trend depending on the current state. Besides, the supervised deep learning and reinforcement learning are jointly and iteratively trained to make full use of the supervised signals in the training data, and obtain more update information and stricter loss function constraints, thereby increasing investment returns. TFJ-DRL is evaluated on real-world financial data with different price trends (rising, falling and no obvious direction). A series of analysis show the robust superiority and the extensive applicability of the proposed method.

1. Introduction

In recent years, algorithmic trading has received increasing attention in the field of artificial intelligence and finance. Algorithmic trading is a branch of quantitative investment, refers to the use of quantitative analysis method to execute actions automatically according to the algorithm model and achieve asset trading without manual intervention (Treleaven, Galas, & Lalchand, 2013). The difficulty of algorithmic trading comes from two aspects, i.e., continuously perceiving environment and dynamically making action decision. Environment perceiving requires to obtain accurate and robust feature representation from highly nonstationary financial time series with a large amount of noise, jump, and movement. Decision making requires the algorithm to explore the environment itself and simultaneously make correct decisions in an online manner, without any supervised information.

In this paper, we design an improved deep reinforcement learning (DRL) method to solve the aforementioned algorithmic trading problem. As is known, reinforcement learning (RL) (Thrun & Littman, 2005) is a machine learning method that obtains policy improvement through trial and error and interaction with the environment. Its advantage is the ability of self-learning, online learning and continuous decision-making, but it lacks perception ability of the environment (Mnih et al., 2013). Deep learning (DL) combines features through multi-layer network structure and nonlinear transformation to form a distributed feature representation, which has strong perception and representation ability, but it is difficult to make continuous decision (Lecun, Bengio, & Hinton, 2015). Deep reinforcement learning combines the perception ability of DL and the decision making ability of RL, thus could provide solutions for the perception and decision-making problem of complex systems (Mnih et al., 2015).

Despite the effectiveness of deep reinforcement learning, algorithmic trading remains a challenge in real-world applications for three reasons. (1) Limited performance due to few or underutilized features. Manual selection of feature is time-consuming and is subject to ignoring certain important features such as technical indica-

* Corresponding author.
E-mail addresses: leik@pkusz.edu.cn (K. Lei), zhang_bing@pku.edu.cn (B. Zhang), liyu7u@pku.edu.cn (Y. Li), min.yang@siat.ac.cn (M. Yang), shenying@pkusz.edu.cn (Y. Shen).

https://doi.org/10.1016/j.eswa.2019.112872
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

tors. In addition, traditional trading experience is unbefitting to apply to emerging markets, making the model difficult to generalize (Moody & Saffell, 2001). (2) Financial signals are time-series signals containing fluctuation and noise. Existing models cannot capture long-term dependencies, and cannot adaptively select important information and trends from noisy historical sequences and summarize them into the current environmental state representation, so the environment could not be well represented (Deng, Bao, Kong, Ren, & Dai, 2017). (3) The deep reinforcement learning model has complex structure and a great deal of parameters. Simultaneous environment state representation learning and strategy learning only through the feedback of each step will cause the model to be unstable and difficult to converge, resulting in problems such as slow training and inaccurate decision-making.

Aiming at the above problems, we propose a time-driven feature-aware jointly deep reinforcement learning model (TFJ-DRL), which is divided into deep learning perception part and reinforcement learning decision making part for algorithmic trading. Specifically, we first learn the environmental states representation through "feature" and "temporal" aspects in the deep learning perception part. In term of feature aspect, we consider various features of financial signals, such as the current financial product features, related financial products features, technical indicators (Neely, Rapach, Tu, & Zhou, 2014), etc. These features are adaptively selected and assigned weights by neural network "gate" structure (Hochreiter & Schmidhuber, 1997). In term of temporal aspect, we utilize gated recurrent unit (GRU) (Chung, Gulcehre, Cho, & Bengio, 2014) to capture long-term dependency and adopt temporal attention mechanism (Pei, Baltrušaitis, Tax, & Morency, 2017) to take the preorder state with different weights into consideration according to the current state. Accordingly, the accuracy, adaptability and interpretability of the deep learning perception model are improved via the consideration of "feature" and "temporal" aspects. Afterwards, we designed a joint framework to jointly construct and iteratively train the supervised deep learning model and the reinforcement learning model. In this context, our proposed model can make full use of the supervised signals in the training data and obtain more updated information and stricter loss function constraints, thereby increasing investment returns.

The proposed model is tested on the real financial market for stock trading. In detail, we select stocks with different wave patterns and utilize real market data to compare the performance of our TFJ-DRL model and other deep or reinforcement learning models. The experimental results show that our model is much robust to different market conditions and could make reliable profits on various markets. Further, we conducted isolation experiments to verify the effectiveness and interpretability of the sub-modules.

This study not only has research significance and application meaning in financial signal representation and algorithmic trading, but also provides reference value for other continuous perception and decision making problems. The main contributions of this paper can be summarized as follows:

(1) We propose a novel deep reinforcement learning model, TFJ-DRL, which is a jointly learning neural framework that combines supervised deep learning and reinforcement learning to improve the financial signal representation learning and action decision making in algorithmic trading.
(2) To overcome the underutilized and noisy feature problem, we adaptively select and reweight various features of financial signals; To solve the time series modeling problem, we capture the long-term dependency of time series via neural network and summarize the attention values between historical information and changing trend depending on the current state. These contribute to well representing the environmental state.
(3) The experimental results show that TFJ-DRL consistently outperforms the state-of-the-art methods. A series of experiments are conducted to demonstrate the accuracy, adaptability and interpretability of the proposed method from various perspectives.

The remaining parts of this paper are organized as follows. Section 2 generally reviews some related works about algorithmic trading. Section 3 introduces the our TFJ-DRL model include time-driven feature-aware environmental states representation module and jointly deep reinforcement learning decision making module. Section 4 is the experimental part where we will verify the performances of our method and compare it with other algorithms. Section 5 concludes this paper and indicates some future directions.

2. Related work

A lot of effective work has been developed in terms on algorithmic trading, including methods based on prior knowledge and methods based on machine learning (Treleaven et al., 2013). The prior knowledge based method (Lee & Lee, 2012) is to design trading strategies based on financial research or trading experience, resulting in poor portability. In contrast, the machine learning method learns trading strategies directly from historical data, and can find profit patterns that people don't know yet without requirement of professional financial knowledge, thus received the focus of research in recent years. The machine learning approach for algorithmic trading can be further divided into the supervised learning approach and the reinforcement learning approach.

The supervised learning method has attempted to predict the stock price or price trend of the next time point (Chong, Han, & Park, 2017). For example, Ding, Zhang, Liu, and Duan (2015) and Akita, Yoshihara, Matsubara, and Uehara (2016) use DL method to predict the stock market through historical text or data. However, the proposed method is difficult to achieve online learning and cannot quickly adapt to new market conditions because of the high cost of retraining. Therefore, it cannot well process the fluctuations in the stock market itself, which is caused not only by short-term noise (such as accidents), but also by the inherent evolution of market fundamentals (such as economic growth, technological development, demographics, etc). Zhang, Aggarwal, and Qi (2017) consider stock predicting as a regression problem and have achieved the state-of-the-art performance by state frequency memory (SFM) recurrent network to capture the multi-frequency trading patterns. Nevertheless, the predicted price or trends of supervised learning method cannot be directly mapped to the trading action. Therefore, after predict the price, prior knowledge of financial field is required to choose trading actions. The accuracy of predict will affect the decision-making process, which is based on the predict result, and resulting in the second error propagation. In addition, prediction-based supervised method does not take into account environmental factors such as transaction cost (TC) and cannot solve the temporal credit assignment problem, resulting in its poor performance in algorithmic trading that has the characteristic of delayed return. Delayed return indicates that the action at a certain moment can determine the current return, may also affect the regression of the next moment, and even affect the return of all moments thereafter (Sutton, 1984).

Different from deep learning method which is difficult to finish continuous decision-making process, reinforcement learning is more suitable for continuous decision-making in algorithmic trading scenario and can be easily extended into an online learning version, Existing methods for algorithmic trading based on reinforcement learning can be divided into value-based methods and policy-based methods.

Gao and Chan (2000) and Pendharkar and Cusatis (2018) utilized value- based reinforcement method Q-Learning to build algorithmic trading model, which iteratively estimates the value function of the state (or state-action pair) and then selects greedy trading action based on the value function. On this basis, Wang et al. (2017) built Deep Q Network (DQN) and utilized the advantages of deep learning to learn distributed features from the original financial signals layer by layer, and serves Q learning network to improve the accuracy of the trading model. However, the disadvantage of value-based method is that the state value is expressed in tabular form and can only deal with discrete state and action space problems. It is easy to cause dimensional explosion and makes the model redundant and complicated. When the problem has continuous state and discretize the state is not allowed, one can only use the function approximation to represent the value function, which cannot guarantee convergence (Sutton, McAllester, Singh, & Mansour, 2000). The sensitivity of the strategy derived from the value function is also high, minor changes of the value function will lead to major changes in the strategy.

Since algorithmic trading is a continuous state scenario, value-based approach has no advantage. Compared with value function based model, the policy-based model is more stable (Li, 2017) and has such advantages: (1) Simplifies problem representation and does not require the estimating of value function, directly gives the action to make the model simpler and more intuitive. (2) Avoids the curse of dimensionality and can make decisions in continuous action and state space. (3) Avoids the sensitivity of strategies derived from value functions from value-based methods. Experimental results in Moody and Saffell (2001) and Dempster and Leemans (2006) also show that the profit of policy gradient method is better than value function based method.

Recurrent Reinforcement Learning (RRL) (Moody & Saffell, 2001) is a policy-based algorithmic trading model, which provides the previous time step trading action along with the current environmental state to Direct Reinforcement (DR) model and directly gives the trading action. The main drawback of this model is the direct input of all environmental features to reinforcement learning without perception and representation. Deng, Kong, Bao, and Dai (2015) introduced the sparse coding model as a feature extractor for financial analysis and achieved more reliable performance than RRL method. However, sparse coding is essentially a shallow data representation strategy whose performance is not comparable with the state-of-the-art DL in a wide range tests (Dahl, Yu, Deng, & Acero, 2012; Lee, Grosse, Ranganath, & Ng, 2009). Deep direct reinforcement (DDR) (Deng et al., 2017) enhanced RRL with Deep Neural Network (DNN) to learn distributed feature representation (Bengio et al., 2009), increasing the depth of the model and improving the accuracy of the environment representation.

However, DNN module cannot perform feature selection and cannot capture the long-term dependence of the timing signal (Hochreiter & Schmidhuber, 1997), and is even more difficult to dynamically consider the past state and the changing trend, either. Tsantekidis et al. (2017) combined spatial and temporal features in a matrix form, while Gudelek, Boluk, and Ozbayoglu (2017) designed 2-D representation for the deep neural network structures (e.g., CNN and LSTM) and successfully developed financial forecasting and algorithmic trading models. These previous studies provide us a reference for the format of input data and the feature representation of the DL module in our DRL algorithmic trading model. Further more, "gate" structure can be used to select features but there is no previous work in algorithmic trading to do so. Pei et al. (2017) has developed temporal attention for supervised classification problems to improve temporal modeling performance. However, there is no literature to use the temporal attention mechanism in the reinforcement learning framework or algorithmic trading field. Further, the variance of policy gradient estimation in existing policy gradient method is usually large, so the speed of convergence becomes very slow. If a deep learning perception module is added to the policy gradient method, the model will be especially difficult to converge because of the complex structure and abundant parameters. These are significant problems for policy gradient method to be widely applied.

In summary, the existing environment perception module is not accurate enough for environmental state perception and representation, the adaptability of model needs to be strengthened, too. The convergence and stability of the existing decision model are difficult to guarantee, and the decision accuracy and total profit need to be improved.

3. Methodology

Our model consists of two components. First, the time-driven feature-aware deep learning perception module is proposed to percept and represent the environmental states of the market. Second, the jointly deep reinforcement learning decision-making module is employed to make trading decisions and take actions. In the rest of this section, we elaborate these two components in details.

3.1. Time-driven feature-aware deep learning perception module

3.1.1. Data acquisition and preprocessing

This study intends to learn the environmental state representation and trading strategy through the historical trading data of stocks. The adopted features of existing models include stock price and trading volume. However, these features are more external representations of market state than internal laws of the market. In order to better model the changing rule of the market, this paper introduces more features in the financial field as the input of the deep neural network, aiming at provide more market information for the model. The features selected in this paper include:

(1) Historical trading data features
The historical trading data of financial products such as stocks include the opening price, closing price, maximum price, minimum price and trading volume of each time interval.

(2) Features of relevant stocks obtained by cointegration test
Cointegration means that the linear combination of two or more non-stationary time series may be stationary or lower order monotonic. Through the cointegration test, we can find stocks that are linearly stable with the current stock price fluctuations, and utilize their price trend features to indirectly represent the current stock price trends.
The lower the cointegration value, the stronger the cointegration relationship. When the value is lower than 0.05, the cointegration relationship is very strong. Fig. 1 is the heat map of calculating the cointegration value for 50 stocks in the S&P 500 index. It shows 1 minus the cointegration value, thus the redder color indicates the stronger cointegration.

(3) Technical analysis indicators
Technical analysis indicators refer to the data collection of stock prices calculated by mathematical formulas in the technical analysis of the stock market. The types and number of technical indicators selected in this study are shown in Table 1, which are calculated with *talib*[1]. Technical indicators are descriptions of markets from different perspectives. For example, exponential moving average(EMA) is computed as a weighted average of the price time series from the current price to the price of the *n* steps in the past, and relative strength index(RSI) shows the degree of recent rise over the recent price change.

[1] http://ta-lib.org/.

Table 1
Categories and numbers of technical analysis indicators.

Indicator category	Indicator name	Indicator number	Output number
Overlap studies	BBANDS, DEMA, EMA, SAREXT, SMA, TEMA, WMA	7	9
Momentum indicators	ADXR, APO, AROON, CCI, CMO, MFI, MACD, MOM, PLUS_DI, PPO, ROC, ROCP, RSI, STOCH, STOCHF, TRIX, ULTOSC, WILLR	18	22
Volume indicators	AD, OBV	2	2
Volatility indicators	ATR, NATR	2	2
Cycle indicators	HT_DCPERIOD, HT_SINE, HT_DCPHASE, HT_PHASOR	4	6
Total	_	33	41

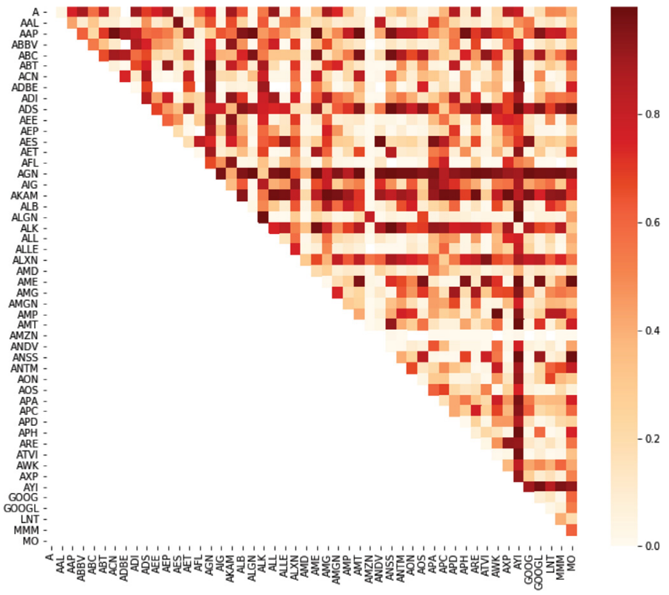


Fig. 1. Cointegration value heat map of some stocks in S&P 500.

To sum up, the input data of the model is a matrix consists of temporal dimension and feature dimension. In other words, the input data is a time series and there are three categories of features on each time step. The input data matrix and feature vector is like Fig. 2.

To facilitate subsequent calculations, we developed normalization and missing value processing for the aforementioned features.

When multiple features exist at the same time, if the range of feature values is very different, the feature with a larger value range is more likely to have a greater impact on the model, which is often not realistic. Normalization is to process the data by some algorithm and then limit it to a certain range. By summarizing the statistical distribution of unified samples, this study normalizes multiple features into the same order of magnitude, making the features comparable and computable. This study then corrects the contribution of each feature to improve the accuracy of the model.

Specifically, this study normalizes the values of each feature separately in the temporal dimension. The aforementioned features are normalized to a data set with a mean of 0 and a variance of 1 using a z-score standardization method. The normalization formula is as follow:

$$z = \frac{x-\mu}{\sigma} \tag{1}$$

Among which, μ and σ are the mean and variance of the original data set respectively. Due to the large fluctuations in the stock market data and the low correlation between the current and remote historical data, it is more reasonable to normalize the features by a certain length of sliding window, rather than normalize the global features data.

In addition, missing data in the dataset is flagged as illegal numbers (Not A Numbers, NANs) if there is a lack of data due to reasons such as trading day closures. Since the input of the neural network must be real, and the probability of data loss is extremely low, we simply replace these NANs with 0.

3.1.2. "Gate" based feature selection and weighting

Although most of the technical analysis methods are summarized in the practice of stock market for more than ten years or even decades, any technical indicators only reflect different angles and focuses on the different degrees of the internal laws of the stock market, thus has limitation and unknowable variables. Therefore, we will study how to adaptively seek one or a set of optimal feature combinations from the original indicators and technical indicators that are most suitable for the current market, and better represent the environmental state of the financial market.

In order to adaptively select and weight the features mentioned in the previous section, we introduce a "gate" structure. The "gate" structure is used in both the LSTM and GRU models, which is consist of a sigmoid neural network and an element-wise multiplication operation. The Sigmoid function is given by:

$$S(x) = \frac{1}{1-e^{-x}} \tag{2}$$

Sigmoid function guarantees that the value of the "gate" structure corresponding to each feature is in the range [0, 1]. When the gate is open (the sigmoid neural network layer output is 1), the feature is completely selected and all information can be passed; when the door is closed (the sigmoid neural network layer output is 0), the feature is discarded and no information can pass; or the feature is weighted selected (the sigmoid neural network layer output between 0 and 1).

The specific structure of the module is shown in Fig. 3.

For each layer in the deep neural network, the value of the "gate" structure g^l is automatically learned from the value of the previous layer f^l, and the formula is as follows:

$$g^l = \sigma(W^l \cdot f^l + b^l) \tag{3}$$

$$f^{l+1} = g^l \odot f^l \tag{4}$$

where σ denotes sigmoid function, g^l and f^l are vectors, and $\odot$ denotes the element-wise multiplication of vectors.

The "gate" based module could be trained in an automatically end-to-end manner and the convergence could be proved (Srivastava, Greff, & Schmidhuber, 2015). During the learning process, there are no additional supervised information and manually extracted features needed because the value of "gate" structure is adaptively derived from end-to-end training. Through such a feature weighting and selection module, our proposed model can

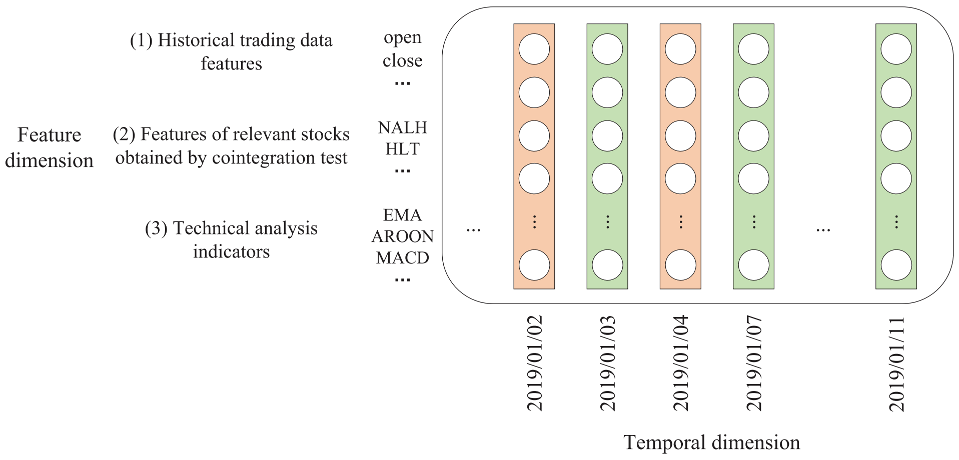

(1) Historical trading data features
open
close
...
Feature dimension
(2) Features of relevant stocks obtained by cointegration test
NALH
HLT
...
(3) Technical analysis indicators
EMA
AROON
MACD
...
2019/01/02
2019/01/03
2019/01/04
2019/01/07
2019/01/11
Temporal dimension

Fig. 2. Input data matrix.

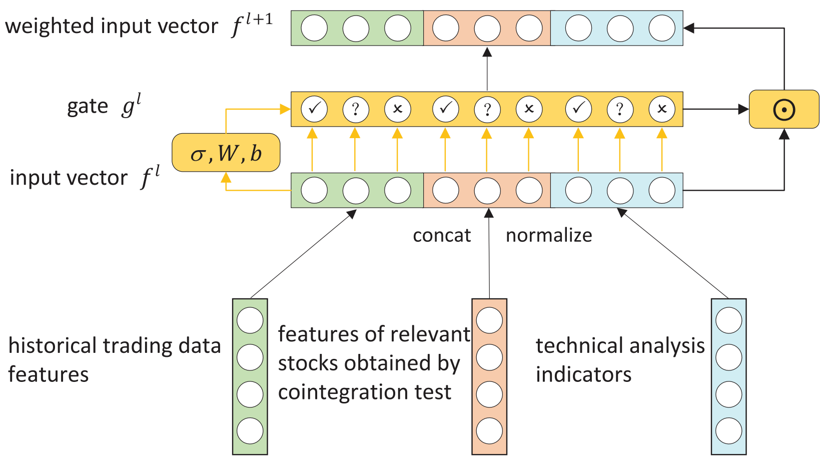

weighted input vector f^{l+1}
gate g^l
σ, W, b
input vector f^l
concat
normalize
historical trading data features
features of relevant stocks obtained by cointegration test
technical analysis indicators

Fig. 3. "Gate" based feature selection and weighting module structure.

adaptively select effective features according to the needs of different types of markets, and improving the adaptability and accuracy of the model representation of the environmental state. For example, suppose that the original vector need to pass the "gate" structure is [1, 2, 3], if the learned value of the "gate" is [0.1, 0.2, 0.3], the final vector is element-wise multiplication of the original vector and the "gate" structure, ie [1, 2, 3] $\odot$ [0.1, 0.2, 0.3] = [0.1, 0.4, 0.9]; if the value of the "gate" is [0.5, 0.5, 0.5], the final vector is [1, 2, 3] $\odot$ [0.5, 0.5, 0.5] = [0.5, 1.0, 1.5]. There are great differences between the above two results, showing the effectiveness of the "gate" structure for controlling information.

3.1.3. *Temporal attention based time series modeling*

In the financial field, when making trading decisions, we not only consider the market state of a single time step, but also need to model the temporal period to capture market trends. This study utilizes GRU model to capture long-term dependencies and adopts temporal attention mechanism to weight the time series.

Fig. 4 illustrates the architecture of temporal attention module. Traditional time series model such as LSTM or GRU considers that the contribution of each time step of the sequence to be the same and implicitly transmits information inside the GRU cell structure, and only outputs the state of the last cell for subsequent calculation. In this study, we adopt temporal attention mechanism to explicitly consider the hidden states of each GRU unit in the previous $T-1$ step with different contributions.

When calculating the Tth time step, different attention is assigned to the hidden state of the previous $T-1$ step according to the current state. The calculation formula of the attention vector is as follow:

$$a_t = softmax(tanh((h_T)^{\mathrm{T}} h_t)) = \frac{exp(tanh((h_T)^{\mathrm{T}} h_t))}{\sum_{t=1}^{T-1} exp(tanh((h_T)^{\mathrm{T}} h_t))}, \quad (1 \le t < T) \tag{5}$$

The state of the previous $T-1$ time steps is weighted and summed into a historical state vector, and the formula is as follow:

$$v_{T-1} = \sum_{t=1}^{T-1} a_t h_t \tag{6}$$

The final environmental state representation vector is computed by:

$$h'_t = tanh(W[h_T; v_{T-1}] + b) \tag{7}$$

The temporal attention mechanism allows the GRU model to adaptively focus on the environmental change information having seen in the previous time step based on the state of the current decision point. It can be seen as assigning weights to past states based on current demands and incorporating them into the environment representation. In this way, the model can pay more attention to market trends in time level. On one hand, it can capture the overall trends in the historical sequence, filter the noise caused by market fluctuations; on the other hand, it can capture and learn from the historical pattern similar to the current market pattern and improve the learning ability of the model. From the

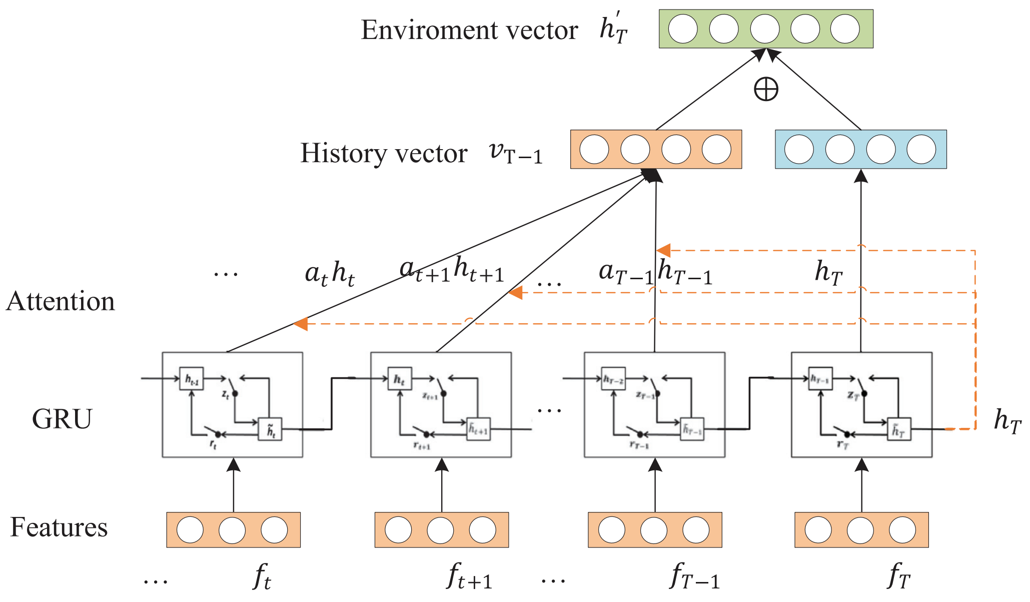


Fig. 4. Temporal attention mechanism module structure.

overall point of view, it can also adaptively select the length of the time window that the model should pay attention to.

3.2. Action decision making module based on jointly deep reinforcement learning

3.2.1. Policy gradient based reinforcement learning

The reinforcement learning decision making part of this paper is modeled by policy gradient method. The input of reinforcement learning model is the environmental state representation vector obtained in the previous section, which is a time series recorded as $h'_1, h'_2, \ldots, h'_t, \ldots$.

At each time step, the benefit is:

$$z_t = p_{t+1} - p_t \tag{8}$$

where p_t is the closing price feature organized in a time series manner $p_1, p_2, \ldots, p_t, \ldots$.

The trading action $\delta_t \in$ {long, neutral, short}={1, 0, -1} is made at each time step. A long position is initiated by purchasing some quantity of a security and makes profits when the subsequent market price goes higher. A short position involves borrowing shares and then selling the borrowed shares to a third party, profit is made when the subsequent market price goes lower and the shorted shares are repurchased at a lower price. Taking account of transaction cost, net income is defined as follow:

$$R_t = \delta_t z_t - c|\delta_t - \delta_{t-1}| \tag{9}$$

where c is the transaction fee paid to the trading company and is payable only if $\delta_t \neq \delta_{t-1}$. The above formula defines the benefit of a single time step, and the total income for the time period of length T is:

$$U_T = \sum_{t=1}^{T} R_t \tag{10}$$

The total income U_T can be regarded as the reward function of the reinforcement learning model.

In order to avoid excessive transaction cost caused by too frequent changes in trading positions (long or short), this study takes the last time step action and the environment state vector together as the input of reinforcement learning model, and obtains the action δ:

$$\delta_t = \pi_\theta(h'_t; \delta_{t-1}) = tanh(< w, h'_t > +b + u\delta_{t-1}) \tag{11}$$

where π is the learned policy, θ is the parameters of model, including the parameters of the deep learning part and the reinforcement learning part. A nonlinear function is adopted to approximate the trading action at each time point to learn the trading policy directly. Tanh maps the function into the range of $(-1, 1)$ during learning and the outputs is discretized when trading. The ultimate goal of the whole model is to obtain the optimal parameter θ to maximize the global reward function (Formula 10).

3.2.2. Jointly supervised deep and reinforcement learning

Due to the uncertainty of reinforcement learning method, it is difficult to guarantee convergence to global optimum. In addition, the deep learning perception model is more complicated and has many parameters. If only relying on the environmental feedback to update the parameters of reinforcement learning decision making module and parameters of deep learning perception module simultaneously, the whole model will converge slowly or even cannot converge, and it will be difficult to guarantee the benefits.

To alleviate these limitations, this study makes full use of the supervised information in the training data, and design a joint model for supervised deep learning and reinforcement learning to improve the training mode.

(1) Framework design
The framework of the joint model is shown in Fig. 5. The blue part is the unfolded deep learning module and the orange part is the reinforcement learning module. f_t is the feature vector, p_t is the closing price, h'_t is the output vector of the attentive GRU network, p'_{t+1} is the closing price of the next day predicted by the deep learning model, π is the reinforcement learning policy, *reward* is the feedback of the reinforcement learning model.
In this paper, the idea of auto-encoder is used to introduce supervised learning signal. The model predicts the closing price of the next day according to the current market situation, and the real data of the next day are used as feedback signals to update parameters of deep learning. Here, the price of the next day predicted by the deep learning

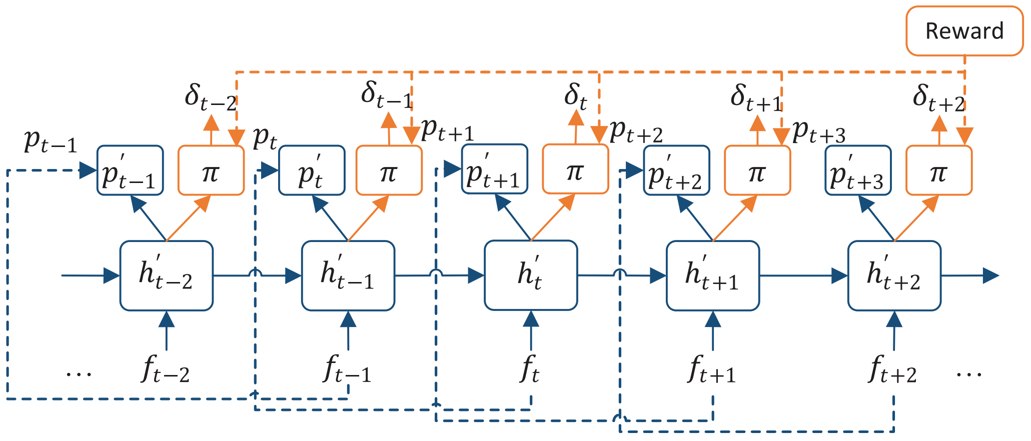


Fig. 5. Jointly construction and training of supervised deep learning and reinforcement learning.

model does not participate in the reinforcement learning decision making of the next action, but serves as the decoding part of the encoder together with the real data of the next day to provide update information for the auto-encoder. Since the predictability of the stock market has always been controversial (Fama, 1995), this study skips the prediction of the stock market price, and takes the environment representation vector as the input of reinforcement learning, which is the intermediate product of the deep learning module. Thus the model can make decisions directly according to the state of the environment, avoiding the second propagation of errors caused by inaccurate prediction. Feedback from reinforcement learning also participates in back propagation and parameter updating.

(2) Loss function design
The loss function of the whole model is designed as follows:

$$loss \ = -\log(\pi_\theta(h'_t, \delta_{t-1})) * U_t + mse_\theta(p_{t+1}, p'_{t+1}) \tag{12}$$

Concerning this formula, its first part is the loss function of reinforcement learning module. $\pi_\theta(h'_t, \delta_{t-1})$ denotes the probability of selecting action δ_{t-1} in state h'_t. When the probability value is smaller, $-\log(\pi_\theta(h'_t, \delta_{t-1}))$ is larger. If the probability of action is very small but a large return U_t is obtained, then loss function $-\log(\pi_\theta(h'_t, \delta_{t-1})) * U_t$ is larger. In other words, if an unusual action leads to a good return, then parameters should be modified by a big margin. The latter part of the formula is the mean square error (mse) loss function commonly used in regression problems, designed to constrain and update the network parameters of the deep learning module.
Adding the two loss functions together to get the overall loss function requires not only the accurate decisions made through reinforcement learning, but also the precise representation learned from deep learning. The parameter update of the deep learning module is affected by two aspects, i.e., the environmental feedback of the reinforcement learning and the supervision information of the original data set. Thus model can get more updated information and stricter constraints to accelerate the convergence speed and improve the overall benefits.

(3) Algorithm design
Based on the framework and loss function design, the pseudocode algorithm of jointly construction and training framework of supervised deep learning and reinforcement learning is described in Algorithm 1.

Algorithm 1: Jointly construction and training framework.

```
   Input: Feature vector $f_1$, $f_2$, ..., $f_t$,...
1  Initialize model parameters $\theta$;
2  for episode 1 to N do
3      while num steps < max length do
4          The DL perception module represents the environment
           state by vector $h'_t$;
5          Predict the price of the next time step $p'_{t+1}$ according
           to the representation vector $h'_t$;
6          The RL decision-making module decides trading action
           $\delta_t$ according to environment state $h'_t$;
7          Calculate the reward and prediction error at the next
           time step;
8          Update $\theta$ according to (Formula 12) based on Adam;
9      end
10 end
   Output: Model parameters $\theta$
```

As shown in Algorithm 1, the DL perception module and RL decision-making module are not separately trained, but are trained and learnt jointly and interactively. Finally, a set of parameters θ, which could maximize the profit, is obtained, which can be used in the test data and real market to improve the accuracy of representation and decision-making.

3.2.3. Overall structure of the TFJ-DRL model

The above two subsections describe the design ideas of each part in DL perception module and RL decision-making module, an overall description of the TFJ-DRL model is given in this section. Fig. 6 shows the overall structure of the whole model.

Fig. 6 clearly illustrates the position and role of each part in the overall model.From the perspective of model design, the whole model is divided into financial signal and market state perception and representation module based on DL and trading action decision-making module based on RL. From the perspective of model training, it is divided into forward propagation, as shown by the solid line; and back propagation, as shown by the dotted line. Among them, the green dotted line with indicates the environmental feedback information from reinforcement learning, and the red dotted line indicates back propagation update information from supervised deep learning.

In summary, such a design could improve the accuracy, adaptability and interpretability of the model, the profit could be improved and the risk could be reduced. Experiments and evaluations of the designed model will be described in the next section.

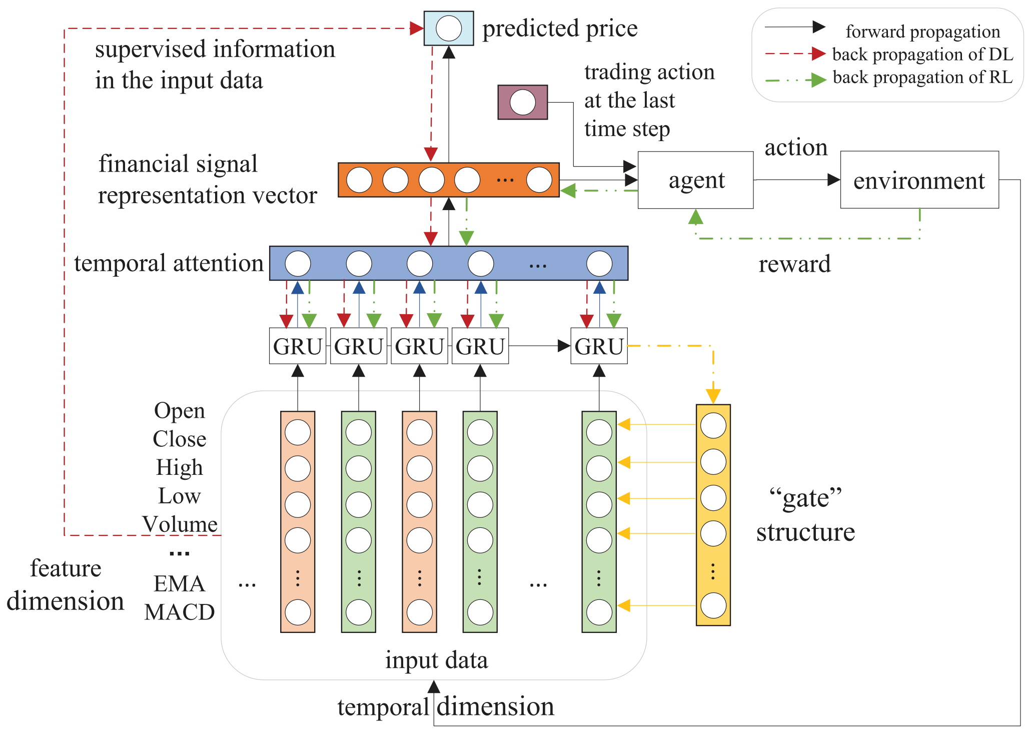


Fig. 6. Overall structure of the TFJ-DRL model.

Table 2
Format of historical stock data.

Time	Open	Close	High	Low	Volume
...	...	...	...	...	...
2018/1/2	2683.73	2695.89	2682.36	2695.81	3367250000
2018/1/3	2697.85	2714.37	2697.77	2713.06	3538660000
2018/1/4	2719.31	2729.29	2719.07	2723.99	3695260000
2018/1/5	2731.33	2743.45	2727.92	2743.15	3236620000
2018/1/8	2742.67	2748.51	2737.60	2747.717	3242650000
...	...	...	...	...	...

4. Experiment and result

Corresponding to the content of our study, the experiment of this paper mainly evaluate three aspects. Firstly, the profit and the effectiveness and generalization ability of the whole model. Secondly, the effectiveness, adaptability and interpretability of the "gate" based feature selection and weighting module for environmental state representation. Thirdly, it verifies the ability and interpretability of time series modeling based on attention mechanism. The data set and comparison experiment will be introduced in detail in this section.

4.1. Datasets

We test our algorithmic trading model on the real-world financial data. We obtain the day-level real transaction data of U.S. stocks from *yahoo finance*[2] and the original input data include the open, close, high, low price and the trading volume. The specific format is shown in Table 2.

We utilize two stock data sets of different sizes to conduct experiment. The smaller data set is used to compare our model with the existing models, and the performance details of the models are displayed. Then the proposed model is extended to the larger data set to verify its generalization ability and reliability.

(1) Small scale dataset containing three different price trends of stocks
In order to ensure the diversity of market patterns, we choose the S&P 500 index or stocks in S&P 500 of different price trends (rising, falling and no obvious direction) and obtain their day-level real transaction data for about 20 years, about 5000 days. Among them, the first 2000 days are used for model training and the last 3000 days are used for model testing. All these stocks allow both long and short operations. The price trends of the stocks we selected are shown in Fig. 7.
(2) Relatively large scale dataset The relatively large scale dataset construction method is similar to the dataset construction method in Zhang et al. (2017). In 11 different sections of the S&P 500, 5 stocks are randomly selected from each section as experimental data, totaling 55 stocks. The training and testing set division is same with the small scale dataset, first 2000 days are used for training and the next 3000 days are used for testing. The specific selected sectors and stocks are shown in Table 4.

4.2. Evaluation metrics

The experimental evaluation metrics in this paper include profit curves, total profits (TP), annualized rate of return (AR), annualized sharpe ratio (SR) and transaction times (TT). The details are as follows.

(1) Profit curve is a qualitative metric to measure the profitability of the model, refers to the change of profit over time, which can reflect the profit of the model at each moment.
(2) TP refers to the profit obtained by the algorithmic trading model using a certain starting capital after a period of trading, and is a quantitative indicator for measuring the profitability of the model.

[2] https://finance.yahoo.com/.

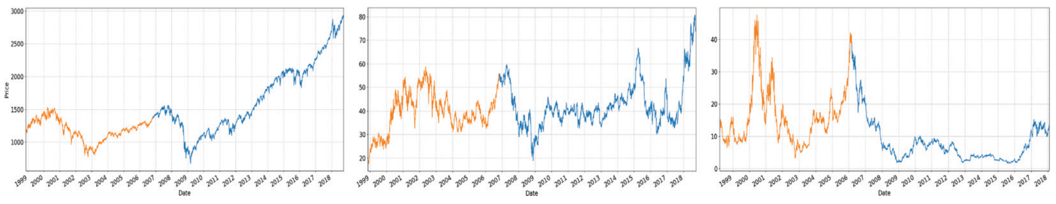

(a) Stock S&P 500. (b) Stock KSS. (c) Stock AMD.

Fig. 7. Prices of the three tested stocks with different pattern (orange parts: model training, blue parts: out-of-sample tests).

(3) AR is the ratio of the return to the principal of the investment period of one year, which is an indicator of the average profitability of the model during the trading period. The formula is defined as follows:

$$AR = \frac{Total\ Profits}{Principal} * \frac{Trading\ Days\ in\ one\ year}{Trading\ Days} * 100 \quad (13)$$

(4) SR (Sharpe, 1994) is a standardized and comprehensive indicator of fund performance evaluation, which can simultaneously consider the benefits and risks, so as to eliminate the adverse effects of risk factors on performance evaluation. The greater the SR, the higher the risk-adjusted rate of return. The SR formula is as follows:

$$sharpe = \frac{E(R) - R_f}{\sigma(R)} \quad (14)$$

where $E(R)$ is the expected return, $\sigma(R)$ is the standard deviation of the return, and R_f is the risk free rate.

(5) TT refers to the number of times the trading action is changed during a period of algorithmic trading, which is measured to prevent too frequent transaction resulting in excessive transaction fees.

4.3. Baseline methods

In this section, we evaluate our TFJ-DRL model on practical trading data. The model is compared with other algorithmic trading models.

- **Buy and Hold (B&H)** is a traditional trading strategy, indicating that after buying an asset, the investment status will not change during the holding period. Thus its profit directly reflects the market trends. It has the advantage of lower transaction cost and management cost, but it abandons the possibility of profitting from changes in the market.
- **SFM** (Zhang et al., 2017) is a prediction-based DL model, which predict the exact price of the next time step rather than the trend and obtain state-of-the-art result. After predicting the price, trading action needs to be selected according to prior knowledge of stock trading. In this paper, we choose long (respectively, short) position when the predicted price higher (respectively, lower) than the current price. The hidden dimensions is set to 50 and frequent dimensions is set to 10 according to the original paper.
- **RRL** (Moody & Saffell, 2001) is a classical RL model for algorithmic trading, which is a policy-based RL model without DL perception module. It is also known as Direct Reinforcement Learning (DRL) in the original literature. To avoid confusion with Deep Reinforcement Learning (DRL), we refer to it as RRL in this paper. The training epochs is set to 100 according to the original paper.
- **DDR** (Deng et al., 2017) is a DRL model, which enhance RRL with DL perception and representation module and obtain state-of-the-art result in algorithmic trading. It is the first paper to implement DL when designing a real trading system for financial signal representation and self-taught reinforcement trading. We conduct four DNN layers with 128, 128, 128, and 20 hidden nodes per layer follow the original paper.
- **GRU based DRL (GRU-RL)** is our basic model to enhance DDR. The DL module in DDR is DNN and cannot capture the long-term dependence of the time signal, thus our basic model replaces DDR with GRU to model time serious signal. We set the hidden layer sizes of GRU as 128.

4.4. Implementation details

Our TFJ-DRL model consists of DL perception module and RL decision-making module. In DL perception module, we set the hidden layer sizes of both GRU as 128 and use ReLU as activation function. The sliding window length of temporal attention is set to 30. In RL decision-making module, policy gradient model is utilized and passed through three dense layer with 128, 64 and 3 outputs dimensions per layer. Then a softmax layer is connected to predict the final action. The loss function is defined in Formula 12 in the previous chapter.

We conduct mini-batch training with batch size 32. Adam algorithm is used to optimize the proposed model with learning rate 0.0001. L_2 regularization (with strength = 0.01) and dropout (with dropout rate = 0.3) are applied to the output layer to avoid overfitting. Besides, the sliding window length of z-score normalization of features is 10. The transaction cost is fixed to 0.1% of the stock value.

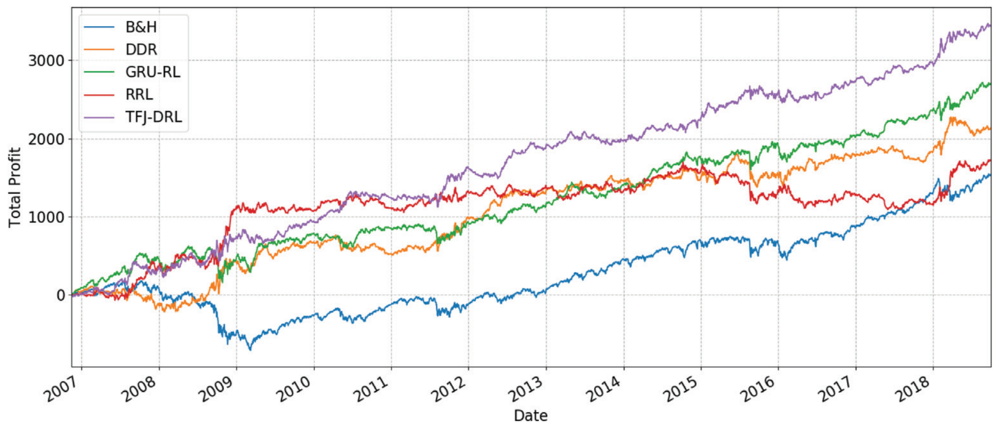
4.5. Experimental results

4.5.1. General evaluations

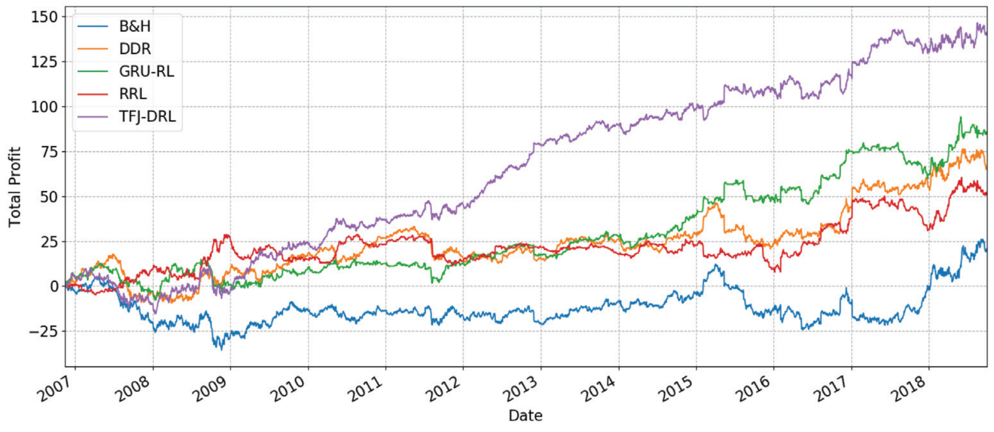
General evaluations of different methods are conducted on stocks of different price trends. Fig. 8 illustrates the details of the profit curves, while Table 3 summarizes the quantitative evaluations from different aspects. From the experimental results, three observations can be found as follows.

First, all the RL methods can make profits in the markets with different trends, but in general, the DRL models (include DDR, GRU-RL and TFJ-DRL) perform better than simple RL model (RRL) that without DL perception and representation module. This observation verifies that the DL feature representation module does contribute to the algorithmic trading. Further, since we have allowed short operation in trading, the trader can also make money in the downward market. For example, the price of AMD goes down in a long period but the models also be profitable.

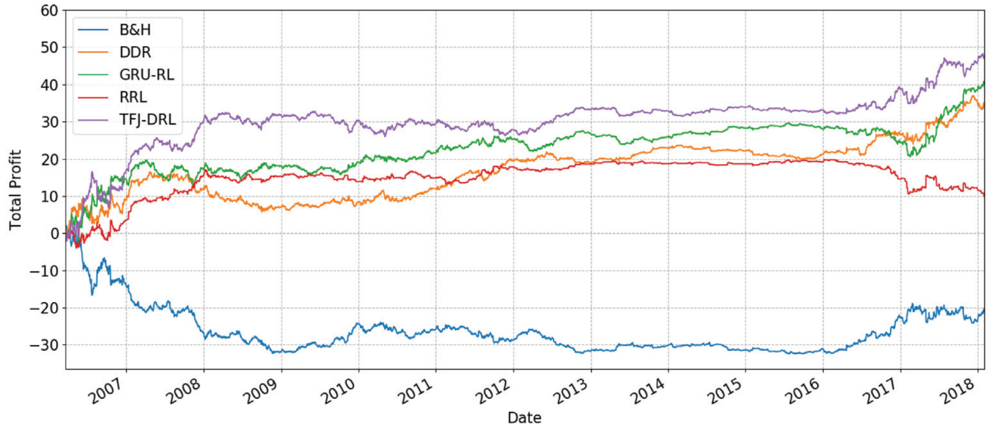
Second, our TFJ-DRL model generally outperform other competitors on all kinds of markets in both TP and SR, indicating that TFJ-DRL model can make good profits within acceptable risk. It is also observed that the profit curves suffer a drawback during the transition period from the increasing (respectively, decreasing) trend to the decreasing (respectively, increasing) trend but the drawback of TFJ-DRL model is smallest, which verifies the framework we proposed indeed contributes in improving the performance in algorithmic trading. In addition, GRU-RL is superior to



(a) Stock S&P 500.



(b) Stock KSS.



(c) Stock AMD.

Fig. 8. Profit curves of different trading methods on different markets.

Table 3
Performances of different trading methods on different markets.

Method	S&P 500 (rising)				KSS (no obvious direction)				AMD (falling)			
	TP	AR(%)	SR	TT	TP	AR(%)	SR	TT	TP	AR(%)	SR	TT
B&H	1546.65	9.43	0.49	_	21.92	3.44	0.13	_	−26.14	−5.57	−0.42	_
SFM	1625.61	9.91	0.52	1094	−12.86	−2.02	−0.07	2118	−0.70	−0.15	−0.01	1208
RRL	1718.21	10.47	0.55	2847	50.77	7.99	0.39	1929	9.89	2.11	0.24	921
DDR	2171.47	13.24	0.64	1872	69.38	10.91	0.44	2042	33.68	7.18	0.56	3076
GRU-RL	2709.68	16.52	0.86	645	89.28	14.04	0.63	778	40.45	8.62	0.67	765
TFJ-DRL	**3431.87**	**20.92**	**1.09**	**609**	**140.81**	**22.15**	**0.84**	**559**	**47.41**	**10.11**	**0.79**	**313**

Table 4
Experimental result on large scale stock market.

		AR(%)			SR			TT	
Sectors	Stocks	B&H	DDR	TFJ-DRL	B&H	DDR	TFJ-DRL	DDR	TFJ-DRL
Communication Services	T	12.26	17.04	**25.64**	0.36	0.70	**0.89**	2180	**601**
	CTL	-0.01	11.46	**16.56**	-0.03	0.41	**0.57**	3198	**859**
	CMCSA	25.64	27.92	**30.43**	0.46	0.56	**0.77**	1792	**520**
	EA	9.89	20.64	**34.38**	0.34	0.84	**1.20**	3054	**941**
	OMC	9.24	**23.39**	21.87	0.30	**0.80**	0.76	1318	**519**
Consumer Discretionary	BBY	4.54	11.59	**20.47**	0.16	0.55	**0.90**	2746	**715**
	HRB	5.33	13.16	**19.96**	0.13	0.41	**0.62**	1474	**611**
	CCL	7.48	20.23	**26.35**	0.25	0.52	**0.79**	1629	**597**
	GPC	16.14	19.02	**21.95**	0.40	0.49	**0.61**	2047	**651**
	WHR	10.33	20.47	**25.64**	0.21	0.44	**0.75**	1012	**529**
Consumer Staple	MO	47.56	47.71	**49.72**	0.93	0.98	**1.12**	802	**620**
	HSY	12.35	21.37	**30.13**	0.47	0.73	**1.02**	1296	**796**
	K	8.41	26.54	**37.84**	0.42	0.91	**1.35**	1984	**887**
	PG	7.61	13.28	**17.76**	0.41	0.61	**0.78**	2294	**1172**
	SYY	15.02	16.37	**21.41**	0.51	0.59	**0.71**	**116**	269
Health Care	ABT	26.34	28.38	**30.25**	0.69	0.71	**0.87**	1278	**529**
	COO	27.48	31.95	**34.59**	0.53	0.60	**0.64**	1017	**417**
	XRAY	6.86	11.61	**30.57**	0.33	0.41	**0.87**	1968	**622**
	VAR	11.15	21.28	**31.23**	0.39	0.73	**1.12**	2146	**796**
	VRTX	29.22	49.18	**57.07**	0.48	0.62	**0.75**	1668	**776**
Energy	APA	-3.79	15.15	**20.95**	-0.09	0.38	**0.56**	2958	**870**
	COP	5.70	16.77	**24.26**	0.30	0.48	**0.64**	2956	**697**
	SLB	1.18	7.83	**12.63**	0.21	0.37	**0.59**	1836	**572**
	XOM	4.80	18.76	**26.84**	0.28	0.52	**0.79**	1763	**682**
	OXY	6.77	16.93	**23.87**	0.32	0.48	**0.67**	1895	**574**
Financials	AMG	6.80	21.26	**27.75**	0.33	0.61	**0.72**	2063	**798**
	COF	2.46	22.38	**28.87**	0.29	0.64	**0.82**	1748	**559**
	STI	-1.91	11.55	**18.98**	-0.09	0.43	**0.59**	2736	**792**
	STT	7.04	20.46	**33.44**	0.36	0.72	**1.14**	1938	**871**
	NTRS	10.15	14.28	**21.82**	0.36	0.47	**0.67**	2234	**913**
Industrials	ARNC	-5.59	-1.67	**-0.12**	-0.24	-0.14	**-0.03**	1847	**756**
	EXPD	4.71	11.75	**17.82**	0.27	0.49	**0.72**	2036	**632**
	GE	-3.37	15.15	**18.52**	-0.16	0.42	**0.61**	2796	**785**
	JCI	11.35	19.82	**28.29**	0.40	0.71	**1.10**	2235	**782**
	UTX	12.86	18.95	**22.83**	0.45	0.77	**1.02**	2104	**652**
Information Technology	AMD	-5.57	4.18	**9.11**	-0.42	0.46	**0.79**	3076	**313**
	IBM	11.68	13.59	**17.57**	0.45	0.48	**0.60**	2386	**746**
	CSCO	12.70	16.77	**20.21**	0.41	0.58	**0.79**	3014	**1204**
	ORCL	20.38	21.42	**24.47**	0.51	0.63	**0.82**	2853	**678**
	AAPL	153.29	160.53	**171.99**	0.94	0.99	**1.13**	**121**	334
Materials	APD	17.65	18.63	**20.53**	0.48	0.54	**0.72**	1273	**489**
	FCX	-2.32	16.23	**27.10**	-0.04	0.42	**0.57**	2826	**1258**
	NEM	-1.65	18.86	**24.74**	-0.06	0.35	**0.54**	2730	**932**
	NUE	4.91	12.72	**19.89**	0.30	0.46	**0.58**	1930	**749**
	VMC	4.16	11.95	**20.03**	0.28	0.50	**0.59**	2047	**846**
Real Estate	HST	2.48	15.73	**22.34**	0.28	0.57	**0.76**	2658	**791**
	DRE	3.01	9.53	**14.26**	0.31	0.52	**0.65**	1758	**672**
	EQR	11.65	12.93	**16.03**	0.38	**0.47**	0.41	1525	**259**
	IRM	9.23	18.68	**25.83**	0.36	0.71	**0.86**	2341	**585**
	VNO	1.21	11.36	**20.06**	0.23	0.40	**0.67**	1160	**996**
Utilities	AES	-1.25	7.62	**12.00**	-0.04	0.39	**0.55**	1596	**341**
	DUK	14.05	16.86	**20.38**	0.52	0.61	**0.78**	2038	**863**
	ETR	6.96	10.98	**16.48**	0.34	0.49	**0.62**	1749	**468**
	FE	1.53	3.52	**9.45**	0.18	0.24	**0.46**	**166**	344
	PPL	5.95	13.43	**18.35**	0.31	0.46	**0.74**	1673	**907**
Average		11.31	19.95	**26.21**	0.29	0.55	**0.75**	1947	**693**

DDR in demonstrating the effectiveness of capturing the long-term dependence of the time serious signal.

Third, the TT of TFJ-DRL model is significantly less than other models, verifying the validity of actions performed by TFJ-DRL, which avoids heavy TCs and still guarantees profits when the TC rises sharply. It is shown in Table 3 that the SFM, RRL and DDR model may change trading positions quite frequent, whose pitfalls will become apparent and may be unacceptable when taking the practical transaction cost into consideration.

In conclusion, the TFJ-DRL framework achieve state-of-the-art results through experimental evaluations on practical stock data of different trend patterns and could make reliable profits.

4.5.2. Relatively large scale market validation

In this subsection, a relatively large scale data set is used to verify the generalization ability and reliability of the proposed model.

This experiment compares our TFJ-DRLmodel with the DDR model, which obtained the best experimental result in the current algorithmic trading research, and with the B&H method, which reflects the stock's own profitability. The experimental results are shown in Table 4.

The experimental results on relatively large scale dataset consisting of 55 stocks randomly selected from 11 sectors in S&P 500 show that the TFJ-DRL model could achieve an average AR of 26.21%, which is 14.9% higher than the stock itself (i.e. B&H

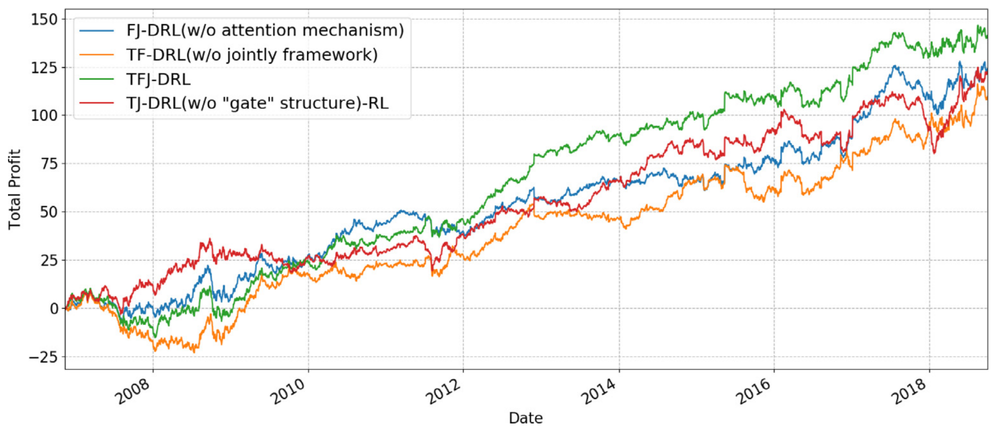


Fig. 9. Profit curves of ablation experiments on stock KSS.

Table 5
Ablation experiments of three parts.

Method	KSS		
	TP	SR(%)	TT
TFJ-DRL	**140.81**	**5.27**	**559**
FJ-DRL(w/o attention mechanism)	124.13	4.64	683
TJ-DRL(w/o "gate" structure)	122.60	4.58	603
TF-DRL(w/o jointly framework)	109.55	4.10	383

method), and 6.26% higher than the algorithm trading model achieving best experimental results currently(i.e. DDR model). The average SR of the model is 0.75, which is 0.46 higher than B&H method and 0.20 higher than the DDR model. The average TT of the TFJ-DRL model is 1254 times less than DDR model.

In conclusion, the TFJ-DRL model can obtain the highest TP and AR on datasets of different scale, which shows that the model has a good profitability and generalization ability; SR is the highest, indicating that the risk of the TFJ-DRL model is more controllable; and the least TT indicating that the transaction action is more effective and the transaction fee is lower. The TFJ-DRL framework achieves state-of-the-art results through experimental evaluations.

Further more, although the profitability of the TFJ-DRL model on different stocks is slightly different, most of them can ensure that the profits are higher than the existing algorithmic trading models. If the proposed TFJ-DRL model can be used after the stock selection or asset allocation technology, or combined with other timing and risk control strategies, the profitability and application value of the model will be further improved.

4.5.3. Ablation experiments

In this section, we conduct ablation experiments to illustrate the effect of "gate" structure, attention mechanism and jointly construction and training framework in the proposed method. We exclude these three parts one by one and report the results in Fig. 9 and Table 5. All ablation experiments are performed using Stock KSS dataset.

Generally, all adopted factors contribute, and it makes larger performance boosting to integrate jointly construction and training framework, which demonstrates the effectiveness of incorporating supervised DL into RL to improve the model convergence and total profits. In addition, it is within our expectation that "gate" structure can improve the overall performance by reweighting the selected features, while the temporal attention mechanism further enhances the representational learning of environmental state.

4.5.4. Visualization of "gate" and attention

"Gate" structure and attention mechanism provide an intuitive way to inspect the importance of each feature and time point respectively by visualizing the weights. We demonstrate the effectiveness and interpretability of the "gate" structure and attention mechanism through case study and visualization.

Due to the limited space, we randomly choose a trading day (2018/09/25) from the stock KSS and visualize the "gate" weights in Fig. 10. We observe that features which are considered to be the most valuable, such as the closing price, the price of relevant stock HST, and the technical analysis indicators TEMA and SLOWD, all derive high weight from the "gate" structure. Conversely, some unimportant features, such as price of VIAB and SIG stock as well as the technical analysis indicators ROC, are assigned with low weights and cannot play an important role in the model. The results show that the "gate" structure can adaptively select features and assign weights for financial signal representation.

We also choose the same trading day to visualize the attention weights (red) and stock price (blue) in Fig. 11. From the results, two observations can be found as follows. First, from a macro perspective, the closer to the current decision-making point, the higher the attention weight the time point obtained, and attention weights of the last ten time points are relatively higher, which indicate that at the time level, signals closer to the current decision point are paid more attention. Second, from a micro perspective, because the current trading day (2018/09/25) is in a downward trend, the previous time points which are in downward trend get higher attention (such as 2018/08/29) and those are in upward trend get lower attention (such as 2018/08/24, 2018/09/04). This shows that at the trend level, historical time points whose price trends are similar to the price trend of the current decision point are receiving more attention. In conclusion, when the model with attention mechanism making a decision, it not only consider the environmental state at the current time, but also consider the environmental states of the previous T time points, and can adaptively weight and incorporate the previous states into the model according to the current state.

5. Conclusions

In this paper, we study deep reinforcement learning approach to solve financial signal representation and algorithmic trading problem. We propose a novel time-driven feature-aware jointly framework, which can (1) adaptively conduct feature selection and temporal weighting, thus improve the accuracy, adaptability and interpretability of the financial signal representation; (2) jointly

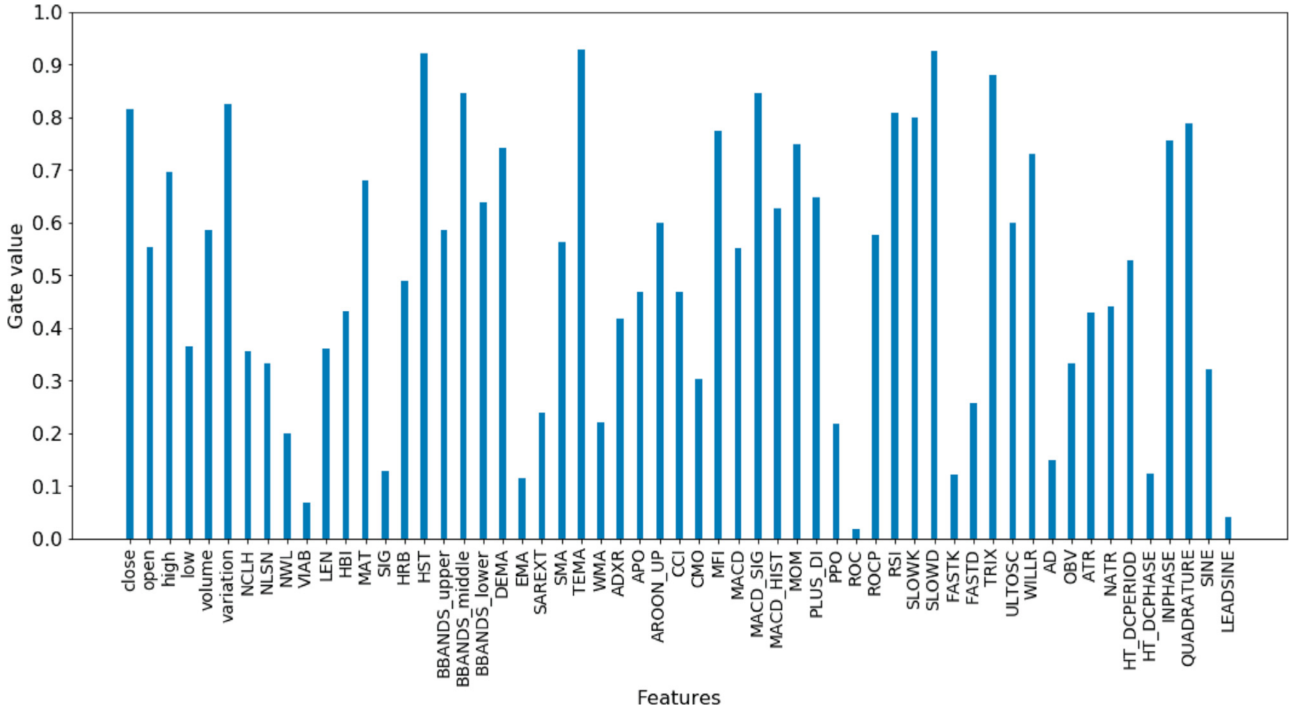


Fig. 10. Weights of the "gate" structure for features.

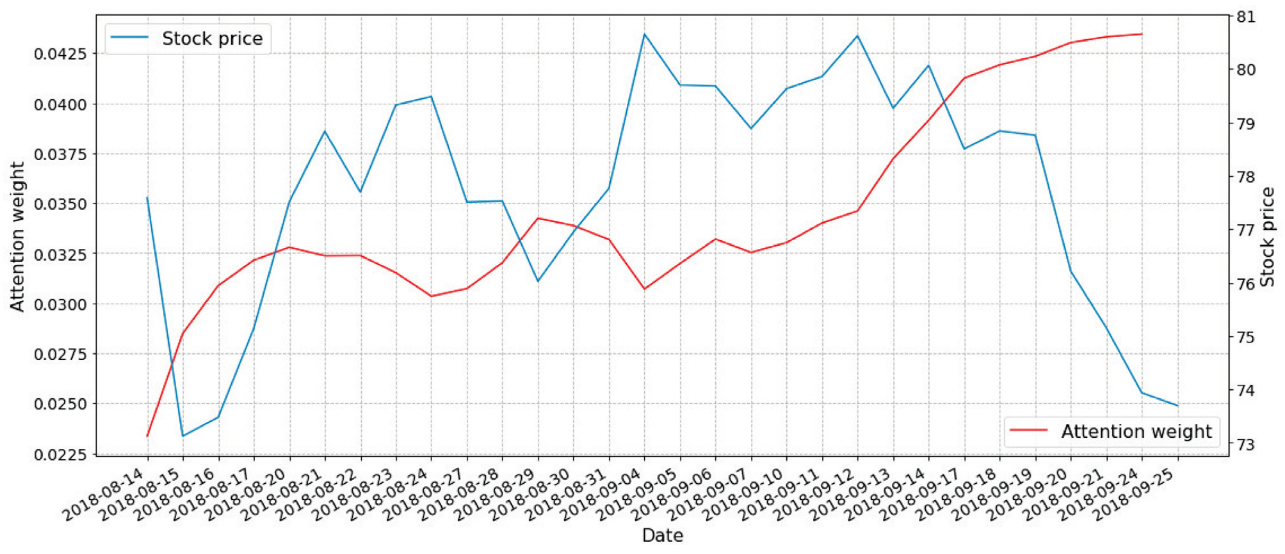


Fig. 11. Weights of attention mechanism for previous trading days.

construct and train the supervised deep learning and reinforcement learning, thus obtain more update information and stricter loss function constraints and improve the accuracy of environment representation and action decision in algorithmic trading. The effectiveness of model is verified on the real stock markets with different patterns. Experimental results demonstrate that our TFJ-DRL model outperforms other competitors with higher total profits and sharpe ratio, lower transaction times and more reliable profit curves.

This study can be further improved in future research in several ways. First, we will conduct experiments on fine-grained data such as intra-day trading data or even high frequency trading to verify the effectiveness of the model and further optimize the model. Also, we will further apply the model to emerging markets such as cryptocurrency. Second, the model proposed in this paper can only trade one asset at a time for the time being, it can be improved to manage the portfolio consists of a number of assets in the future. Last but not least, some basic financial skills could be introduced to enhance the model, for example, the manipulation such as end-of-day dislocation (Aitken, Cumming, & Zhan, 2015) or insider trading (Dai, Fu, Kang, & Lee, 2016) could provide the information for the machine learning model to take advantage of, the stock selection technology and the timing and risk control strategies could further improve the profitability and application value of the model.

Author contributions

Kai Lei conceived of the presented idea. Bing Zhang developed the theory and performed the computations. Ying Shen and Min Yang verified the analytical methods. Bing Zhang and Yu Li Yu Li

Yu Li carried out the experiment. All authors discussed the results and contributed to the final manuscript.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Kai Lei: Data curation, Formal analysis. **Bing Zhang:** Methodology, Writing - original draft. **Yu Li:** Data curation. **Min Yang:** Writing - review & editing. **Ying Shen:** Project administration.

Acknowledgement

This work was financially supported by National Natural Science Foundation of China (No. 61602013), Natural Science Foundation of Guangdong (No. 2018A030313017), Shenzhen Fundamental Research Project (Nos. JCYJ20170818091546869 and JCYJ20170412150946024), Shenzhen Key Laboratory Project (ZDSYS201802051831427) and the project "PCL Future Regional Network Facilities for Large-scale Experiments and Applications". Min Yang was sponsored by CCF-Tencent Open Research Fund.

References

Aitken, M., Cumming, D., & Zhan, F. (2015). High frequency trading and end-of-day price dislocation. *Journal of Banking &amp; Finance, 59*, 330–349.
Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *Proceedings of the 15th international conference on computer and information science (ICIS), IEEE/ACIS* (pp. 1–6).
Bengio, Y., et al. (2009). Learning Deep Architectures for AI. *Foundations and trends® in Machine Learning, 2*(1), 1–127.
Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. *Expert Systems with Applications, 83*, 187–205.
Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.
Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing, 20*(1), 30–42.
Dai, L., Fu, R., Kang, J.-K., & Lee, I. (2016). Corporate governance and the profitability of insider trading. *Journal of Corporate Finance, 40*, 235–253.
Dempster, M. A., & Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications, 30*(3), 543–552.
Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems, 28*(3), 653–664.
Deng, Y., Kong, Y., Bao, F., & Dai, Q. (2015). Sparse coding-inspired optimal trading system for HFT industry. *IEEE Transactions on Industrial Informatics, 11*(2), 467–475.
Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep learning for event-driven stock prediction.. In *Proceedings of the IJCAI* (pp. 2327–2333).
Fama, E. F. (1995). Random walks in stock market prices. *Financial Analysts Journal, 51*(1), 75–80.
Gao, X., & Laiwan, C. (2000). An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization. In *Proceedings of the international conference on neural information processing* (pp. 832–837).
Gudelek, M. U., Boluk, S. A., & Ozbayoglu, A. M. (2017). A deep learning based stock trading model with 2-d CNN trend detection. In *Proceedings of the IEEE symposium series on computational intelligence (SSCI)* (pp. 1–8). IEEE.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.
Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning.. *Nature, 521*(7553), 436.
Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616).
Lee, K. C., & Lee, S. (2012). A causal knowledge-based expert system for planning an internet-based stock trading system. *Expert Systems with Applications, 39*(10), 8626–8635.
Li, Y. (2017). Deep reinforcement learning: an overview. arXiv:1701.07274.
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv:1312.5602.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks, 12*(4), 875–889.
Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science, 60*(7), 1772–1791.
Pei, W., Baltrušaitis, T., Tax, D. M., & Morency, L. P. (2017). Temporal attention-gated model for robust sequence classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 820–829).
Pendharkar, P. C., & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications, 103*, 1–13.
Sharpe, W. F. (1994). The sharpe ratio. *Journal of Portfolio Management, 21*(1), 49–58.
Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. arXiv:1505.00387.
Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning*. University of Massachusetts Ph.D. thesis. 34 (5), 601–616
Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Proceeding of the Advances in neural information processing systems* (pp. 1057–1063).
Thrun, S., & Littman, M. L. (2005). Reinforcement learning: an introduction. *IEEE Transactions on Neural Networks, 16*(1), 285–286.
Treleaven, P., Galas, M., & Lalchand, V. (2013). Algorithmic trading review. *Communications of the ACM, 56*(11), 76–85.
Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *Proceedings of the IEEE 19th conference on business informatics (CBI): 1* (pp. 7–12). IEEE.
Wang, Y., Wang, D., Zhang, S., Feng, Y., Li, S., & Zhou, Q. (2017). *Deep q-trading*. cslt. riit. tsinghua. edu. cn.
Zhang, L., Aggarwal, C., & Qi, G.-J. (2017). Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2141–2149).