Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 12, No. 1, 2025

Faster and More Robust GA-Lion and MM-MLP for High-Frequency Trading

Rongsen Zhang, Ailing Gu *, Yu Hao

School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou Guangdong, 510000, China
*** Corresponding author:** Ailing Gu

Abstract: The key challenges in applying deep neural networks (DNNs) for stock high-frequency trading prediction lie in accelerating the training of DNNs as well as improving the robustness of deep learning (DL) models. High-frequency trading is a race-to-the-bottom task, which places high demands on the efficiency of model training. To accelerate the training of DNNs, we propose an improved Evolved Sign Momentum (Lion) algorithm, namely the Gauss Simulated Annealing Lion (GA-Lion), to significantly improve the training speed so that the model can adapt to the market dynamics more quickly and meet the real-time demand of high-frequency trading. Meanwhile, in order to enhance the robustness of DNN, we design a novel feature extraction module based on multilayer perceptron (MLP) called Masked Moving-Average MLP (MM-MLP), which is able to effectively capture the key features in the price series and reduce the noise interference through the moving-average mechanism, thus improving the stability and generalization ability of the model in the complex market environment. This improves the stability and generalization ability of the model in complex market environments. Combined with the gated recurrent unit (GRU) and optimized by GA-Lion and MM-MLP, our model achieves faster loss convergence on high-frequency trading data, while demonstrating greater robustness in highly volatile market environments.

Keywords: Deep Learning (DL); Gated Recurrent Unit (GRU); Evolved Sign Momentum (Lion); Simulated Annealing (SA); Multilayer Perceptron (MLP).

1. Introduction

High-Frequency Trading (HFT) is a form of algorithmic trading that relies on high performance computers to execute a large number of orders in a very short period of time. [1] However, as competition increases, the short-term dynamics of the market become more chaotic, raising the level of noise and nonlinearity in high-frequency data.[2]With the development of Artificial Intelligence (AI) and Deep learning (DL), studies[3] have proposed the use of Deep Neural Networks (DNNs) to mine potential profit opportunities from high-frequency market data However ,DL requires considerable computational resources[4] making the training of deep neural networks (DNNs) time-consuming , especially for the platform with limited computational resources[5].This issue is compounded in stock forecasting by the high volatility of stock price data[6],which hinders DNNs from promptly identifying stock price patterns before market changes occur in a short period of time[7-9],Secondly, DNNs are sensitive to input noise, leading to a lack of robustness[10],noise learning often manifests as over-fitting [11-12], This over-fitting problem is particularly pronounced in stock due to the noisy nature of stock price data[13].Faced with noise and volatility[14-15],a crucial challenge in modeling the stock market[16]lies in developing training methods that enable faster convergence of loss and enhancing robustness (i.e.generalization ability) of model [17-19].

Training acceleration and robustness enhancing techniques can be categorized into five main perspectives [20]:(1)*optimization-centric* , including the selection of learning rates ; (2) *data-centric* , including regularization (e.g. Dropout , averaging weight[21]) ; (3) *model-centric* ; (4) *budget-centric* ; (5) *system-centric* .

From an *optimization-centric* perspective, focused on adaptive learning rates method, Ajay et al. [22] concluded that learning rate optimizers can accelerate the training of DNNs. Traditional optimizers like AdaGrad, RMSprop, Adam, and Adam W adapted learning rates based on past gradients [23-26]. Recently, Chen et al. [27]introduced Lion (EvoLved Sign Momentum) algorithm, enabling faster training compared to traditional optimizers. Chen et al. [28]provided a theoretical framework for further improvements of Lion. In fact, the existing improvements to Lion's algorithm are limited and do not fully utilize its potential in high-frequency trading. Drawing inspiration from an enhanced Adam algorithm integrating Cosine-Annealing technique[29] , we propose the Gauss Simulated Annealing Lion (GA-Lion), which achieves faster convergence speed than Lion, and shows better adaptability and stability in high-frequency trading environments, so that the model can capture market dynamics more efficiently and improve the accuracy of trading decisions.

From an *data-centric* perspective, focused on regularization method, Wang et al. [30]introduced an Exponential Moving Average (EMA) method[31], which enhanced the robustness of DNNs by generalizing the weight [21] . Drawing inspiration from Dropout, Gong et al. [32] introduced a Local Feature Masking (LFM) technique that randomly masked features in the shallow layers of CNNs , allowing the model to more effectively handle variations in input data. However, the above techniques have not yet been applied to stock forecasting. Inspired by these works , we proposed a module called Masked Moving-Average MLP (MM-MLP) to enhance the model's noise adaptability .

The GRU model consists of update gate and reset gate, compared with LSTM, the structure of GRU is more simplified, so it is faster. Therefore it is more suitable for high-frequency trading tasks. It is a promising tool for stock forecast [33],enhanced by the GA-Lion and MM-MLP , our

model achieved faster loss convergence and demonstrated robustness, i.e. generalization ability [17-19,34].Our main contributions are as follows: Firstly, GA-Lion improves the Lion optimizer, enhancing optimization performance and significantly accelerating model training. This makes it more suitable for the fast-changing environment of high-frequency trading. Compared to the original Lion, GA-Lion demonstrates superior adaptability and stability in stock high-frequency trading forecasting, improving the timeliness of trading decisions. Secondary, MM-MLP enhances MLP by improving its ability to handle noisy data. By incorporating the Masked Moving-Average mechanism, MM-MLP effectively reduces market noise interference and enhances the model's generalization ability, enabling more stable predictions in complex market environments. Therefore, for modules with similar limitations applied to high-frequency trading, our method can provide valuable insights.

2. Design of GA-Lion Optimizer

2.1. SA, Lion

2.1.1. SA

SA (simulated annealing)[35] is used to find an approximate global optimum in a large search space. The algorithm is based on the Metropolis criterion, which allows for the acceptance of worse solutions probabilistically to avoid getting stuck in local optima. The probability of accepting a worse solution is given by the Metropolis criterion:

$$\mathrm{P(accept)} = \mathrm{e}^{-\frac{\Delta \mathrm{E}}{\mathrm{T}}}$$

where ΔE is the change in energy (objective function value in optimization), and T is the control parameter that decreases over time, known as the cooling schedule.

2.1.2. Lion

The update rule for the *Lion* (evo*L*ved s*i*gn mo*m*e*n*tum) is [27]:

$$\text{Lion:} = \begin{cases} u_t = sign(\beta_1 m_{t-1} + (1-\beta_1) g_t) + \lambda \theta_{t-1} \\ \theta_t = \theta_{t-1} - \eta u_t \\ m_t = \beta_2 m_{t-1} + (1-\beta_2) g_t \end{cases}$$

where η represents the learning rate of Lion, λ represents decoupled weight decay, $\mathrm{m_t}$ represents momentum, $g_t = \nabla_\theta L(\theta_{t-1})$ represents the gradient of the loss function. Compared with widely-used AdamW optimizer, Lion uses fewer parameters, requires less memory, removes computationally intensive operations. The sign(·) function, which adds noise to the updates, resembles signed gradient descent and signed momentum [36], normalizing coordinate-wise updates[37-39].

However, Lion does not consistently perform better than AdamW in terms of convergence rate[27] .From this, in high-frequency trading and other time-critical tasks, the advantage of Lion's small computational size is not well reflected. Therefore it is necessary to further improve the optimization speed of Lion.

2.2. Gauss Simulated Annealing Lion (GA-Lion).

Inspired by the idea of numerous evolutionary algorithms jumping out of local optima, GA-Lion extends the parameter search space of the Lion optimizer to adapt more efficiently to complex optimization problems and to reduce the effect of noise. However, simply extending the search space may lead to too large a search range and affect the convergence stability. Therefore, we further borrow the idea of simulated annealing algorithm (SA) and introduce the simulated annealing factor so that GA-Lion can dynamically adjust the search intensity during the optimization process and balance the relationship between global exploration and local convergence. This mechanism not only enhances GA-Lion's robustness to noise, but also improves the stability and convergence speed of the optimization, giving it better performance in highly dynamic environments such as high-frequency trading.

Startup Stage: At the beginning of training, GA-Lion uses a high initial learning rate to speed up the early global search. This process is shown below:

$$exp\_avg \leftarrow exp\_avg \times \beta_1 + g \times (1-\beta_1)$$
$$exp\_avg \leftarrow exp\_avg \times \beta_2 + g \times (1-\beta_2)$$
$$\theta \leftarrow \theta - \eta \times sign(exp\_avg)$$
$$\theta \leftarrow \theta \times (1 - \eta \times \lambda)$$

Exploration *Stage*: During the exploration stage, to broaden the optimizer's parameter search, GA-Lion introduces a update mechanism every 10 epochs. This adds random disturbances at each interval to prevent the optimizer from getting stuck in local minima. The mathematical expression is:

$$\theta \leftarrow \theta + \eta \times k \times N$$

Here, k is a predefined factor multiplied by a Gaussian random number N to create periodic disturbances.

Annealing Stage: In the annealing stage, to reduce computational load and maintain training efficiency, GA-Lion gradually decreases update factor k. Initially, a higher sparrow factor allows larger disturbances, helping to explore a wider range of parameters and escape local optima. As training progresses, k decreases, reducing disturbances and enabling finer parameter adjustments to find the optimal solution. The mathematical expression is:

$$k \leftarrow k \times (1 - ar \times gs)$$

Here, ar represents annealing rate, gs represents global steps, which is used to keep track of the number of training iterations. The whole GA-Lion algorithm is summarized in Algorithm 1.

3. Masked and Moving Average Multilayer Perceptron (MM-MLP)

MLP excels at extracting features from datasets, its feature is that each node in the linear layer is connected to all nodes in the previous layer, which means that each node must learn all the features output from the previous layer. This full connectivity can lead to over-learning of meaningless features, especially for noisy data like stock prices, which can lead to overfitting. To enhance the generalization ability of MLP in stock market forecasting, drawing inspiration from an enhanced Adam algorithm integrating Cosine-Annealing technique[29], we employed a random matrix to randomly mask certain weights in linear layer during training. This selective neglect of noise features reduces the learning intensity of the noise and converts the linear layer into a *sparse linear layer*. In addition, we perfected the generalization ability of these sparse layers by applying moving average techniques, and finally created MM-MLP. The core steps of a MM-MLP include:

Algorithm 1 Gauss Simulated Annealing Lion (GA-Lion)

Input: η, β_1,β_2, λ, k, *ar*, *update_interval*, Θ_0
Output: loss
1: defaults ← η, β_1,β_2, λ, k, ar, update_interval
2: Initialize parameter group: $\Theta \leftarrow \Theta_0$
3: Initialize global step: gs ← 0
4: **while** not converged **do**
5: gs ← gs + 1
6: **for all** θ **do**
7: $k \leftarrow k \times (1 - ar \times gs)$
8: **end for**
9: **for all** θ **do**
10: for all θ in Θ do
11: $\theta \leftarrow \theta \times (1 - \eta \times \lambda)$ { **Weight Decay** }
12: $update \leftarrow \beta_1 \times exp\_avg + (1 - \beta_1) \times grad$
13: $\theta \leftarrow \theta + sign(update) \times (-\eta)$ { **Lion Weight Update** }
14: $exp\_avg \leftarrow exp\_avg \times \beta_2 + grad \times (1 - \beta_2)$ { **Momentum Decay** }
15: **end for**
16: **if** *gs mod update_interval = 0* **then**
17: **for all** θ in Θ **do**
18: θ ← θ + k × N × η
19: **end for**
20: **end if**
21: **end for**
22: **return** *loss*
23: **end while**

Constructing a sparse linear layer: A random matrix of the same dimensions as the weight matrix is generated using a random number generator torch. rand() and is multiplied with the weight matrix, it results in a number of elements being zeroed out. The forward propagation formula for the sparse linear layer is as follows:

$$y = x \times (weight \odot mask)^T + b$$

The symbol $\odot$ denotes element-wise multiplication, and the mask is a matrix formed by a randomly generated sparsity mask.

Smoothing the change of parameters of model : During forward propagation, the weights in the sparse linear layer are adjusted. We applied exponential moving average (EMA) and Hull moving average (HMA) to the weight adjustment process. HMA first calculates the weighted moving average for half the window size n/2 , then calculate the weighted moving average of the complete window size n , and obtain the formula for HMA :

$$W_{Half} = \frac{1}{n/2}\sum_{i=1}^{n/2} w_i \times HP_t$$

$$W_{Full} = \frac{1}{n}\sum_{i=1}^{n} w_i \times HP_t$$

$$H_t = 2 \times W_{Half} - W_{Full}$$

where w_i is the weighting coefficient, and HP_t is the HMA parameter at time t. Finally, the parameters calculated are applied to update the EMA model parameters. The EMA update formula is as follows:

$$EP_t = decay \times EP_t + (1 - decay) \times HP_t$$

,where HP_t represents the current EMA parameter. This hybrid moving average strategy smooths the weight updates in the sparse linear layer after each backward propagation (i.e. model update) , making the change of weights smoother and reducing the risk of over-fitting to noise . The algorithmic process of MM-MLP integrated GRU model is as Fig1. Experimental results will be showed in the next chapter.

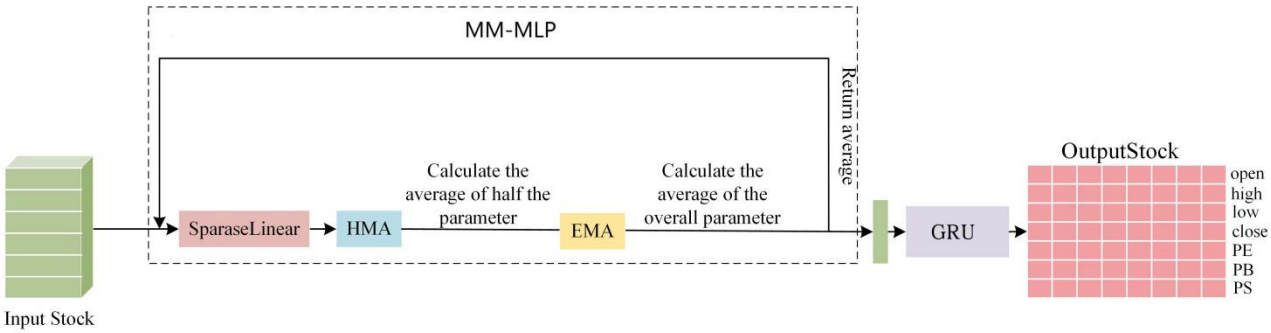

Fig 1. Applying MM-MLP to GRU

4. Experimental Design and Experimental Results

4.1. Data Acquisition

All experiments were conducted on a consistent setup with NVIDIA GeForce RTX 3080Ti GPU and 12v CPU Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz. Implemented in PyTorch 1.10.0 with 100 epochs, η=1e-2, (β_1, β_2) = (0.9, 0.99), k= 0.01, and ar = 0.0001, updated every 10 epochs.

The purpose of this study is to forecast the opening prices of typical stocks representing four different industries, involving China Vanke Co., Ltd. (000002), iFLYTEK Co., Ltd. (002230), BYD Company Limited (002594), Contemporary Amperex Technology Co., Limited (300750), Kweichow Moutai Co., Ltd. (600519), and Semiconductor Manufacturing International Corporation (688981). These stocks cover a wide range of sectors such as real estate, artificial intelligence, new energy, consumer goods (liquor), new energy batteries, biopharmaceuticals, etc., and have high market influence and attention. We use data from the Wind Economic Database, a platform that provides comprehensive, accurate and timely financial data services. Crossover frequency data for the daily opening of each stock from January 2, 2024 to November 7, 2024, totaling 48,960 records, were selected for this study. In HFT, the commonly used time series data include price data (e.g., opening price, closing price) and trading data (e.g., price-earnings ratio, price-sales ratio). Price data provides a direct picture of stock price fluctuations, while trading data provides additional market

information through the lens of stock activity and corporate profitability. These data are considered to be important references in predicting stock movements, covering a wide range of factors that affect stock prices. In this study, we extracted seven key factors for each stock, including opening price (Open), high price (High), low price (Low), closing price (Close), price-earnings ratio (PE), price-to-book ratio (PB), and price-to-sales ratio (PS). Among them, the opening price is our target variable, while the remaining six factors are analyzed as characteristic variables. The high, low, and closing prices reflect the price fluctuations of a stock during the day, corresponding to the maximum demand, minimum supply, and overall evaluation of the market, respectively, and these factors directly affect the market's expectations of a stock's upside potential, downside risk, and rate of return. In addition, price-to-earnings (PE), price-to-book (PB), and price-to-sales (PS) ratios measure a company's valuation from the perspectives of profitability, net worth, and sales capacity, which in turn affects investor perceptions of a company's future prospects, and consequently, a stock's opening price. Therefore, we chose these six factors as characteristic variables in order to more accurately predict the opening price of a stock.

We used MM-MLP integrated GRU to forecast stock price data and applied GA-Lion for hyper-parameter tuning, achieve the result below.

4.2. Performance of GA-Lion

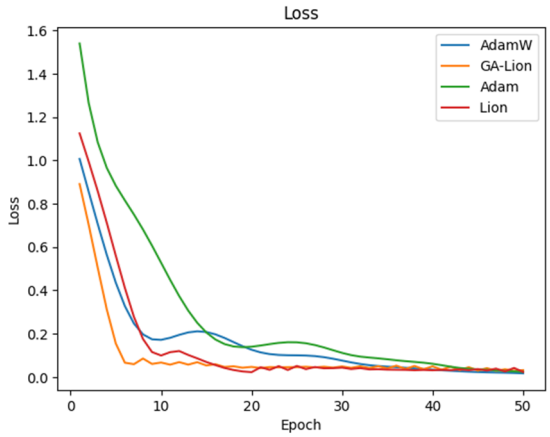

Fig 2. China Vanke Co., Ltd. (000002)

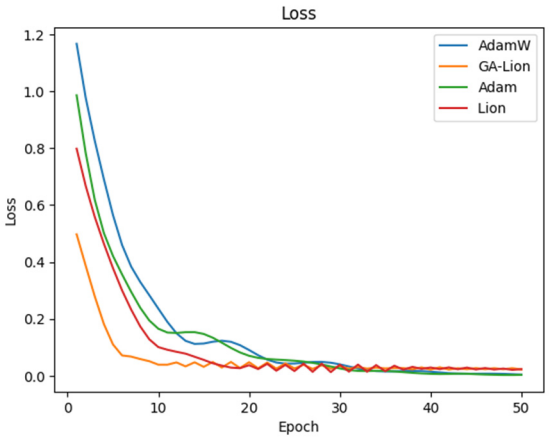

Fig 3. iFLYTEK Co., Ltd. (002230)

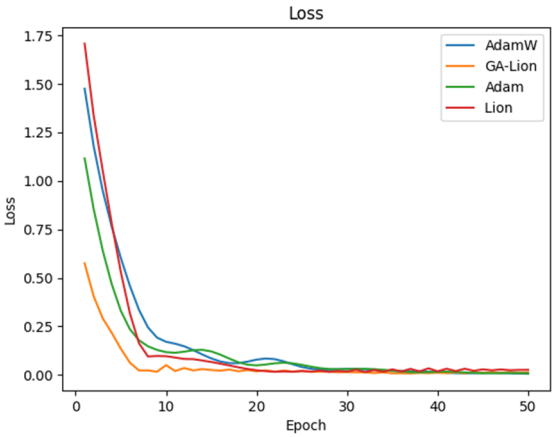

Fig 4. BYD Company Limited (002594)

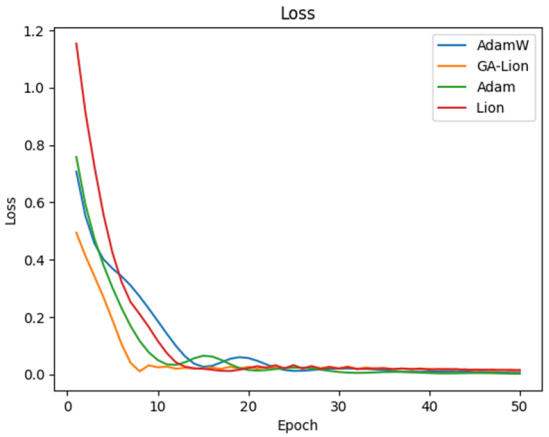

Fig 5. Contemporary Amperex Technology Co., Limited (300750)

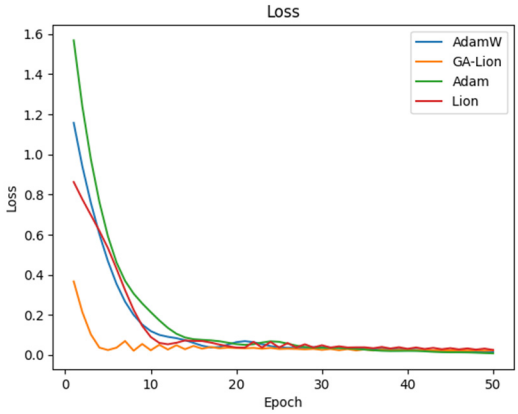

Fig 6. Kweichow Moutai Co., Ltd.(600519)

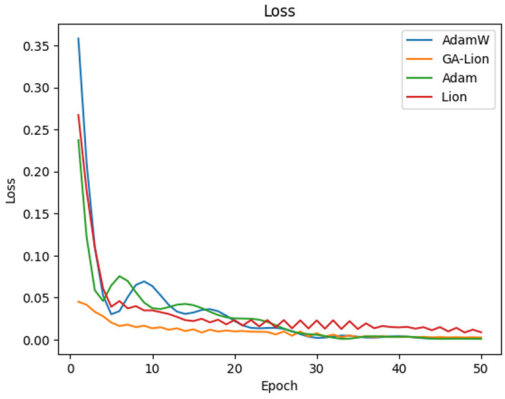

Fig 7. Semiconductor Manufacturing International Corporation(688981)

In this paper, the GA-Lion optimizer is used to optimize GRU to cope with the impact of time-sensitive data on the training process. Compared with other optimizers, the GA-Lion optimizer performs more superior in optimization speed, can adjust the GRU model parameters more quickly, improves the stability of network training, and improves the prediction accuracy. The GA-Lion optimizer is used to optimize the training process of the network when facing time-sensitive data, in order to adjust the parameters of the GRU more finely and make the network training faster. In order to evaluate the overall performance of GA-Lion, we compared, Adam[25], AdamW [26], Lion[28], GA-Lion performance, the experimental results and Loss curves are shown in Fig2,3,4,5,6 and7.

4.3. Performance of MM-MLP

To demonstrate the effectiveness and superiority of MM-MLP for GRU networks, we conducted experiments using the Adam optimizer, comparing with and without the addition of EMA and HMA strategies. The results are shown in table1.The results, shown in Table1, indicate that without EMA or HMA, the MLP provides minimal improvement to the network. However, when EMA and HMA are added separately, the network performance improves, and the best results are achieved when both are added (using MM-MLP). This demonstrates that EMA, by giving higher weight to recent data, and HMA, by further reducing lag through weighted and square root calculations, effectively enhance the network's ability to process and utilize information from stock sequence features.

Table 1. Comparison of Different Modules

Method	MSE	MAE	RMSE	R^2
Without EMA and HMA	0.01435	0.08233	0.11979	0.71119
With EMA	0.00605	0.04462	0.07781	0.87814
With HMA	0.01094	0.06051	0.10460	0.77989
MM-MLP	*0.04761*	*0.12939*	*0.21820*	*0.95239*

4.4. Performance of GRU Enhanced by GA-Lion and MM-MLP

4.4.1. Goodness-of-Fit Test

To further analyze our model's goodness-of-fit performance, we compared it with several classic and state-of-the-art time series forecasting methods under the same environment and dataset, with results as shown in the Table 2.

Table 2. Comparison of different models for six stocks

Stocks	MSE	MAE	RMSE	R^2	Method
China Vanke Co., Ltd.(000002)	0.10597	0.25236	0.32554	0.89373	FEDformer
	0.10379	0.25741	0.32216	0.89593	TCN
	0.06285	0.18857	0.25070	0.93715	GRU
	0.03309	*0.14846*	*0.18191*	*0.96691*	*Ours*
iFLYTEK Co., Ltd. (002230)	0.22195	0.33005	0.47111	0.77807	FEDformer
	0.16519	0.29031	0.40643	0.83483	TCN
	0.15652	0.22036	0.39563	0.84348	GRU
	0.11023	*0.20567*	*0.33201*	*0.88977*	*Ours*
BYD Company Limited (002594)	0.27414	0.33178	0.52362	0.72583	FEDformer
	0.30234	0.38628	0.54986	0.69766	TCN
	0.21168	0.25349	0.46008	0.78832	GRU
	0.14042	*0.20637*	*0.37473*	*0.85958*	*Ours*
Contemporary Amperex Technology Co., Limited (300750)	0.39230	0.33050	0.62634	0.60769	FEDformer
	0.10215	0.20547	0.31961	0.89785	TCN
	0.13187	0.22191	0.36314	0.86813	GRU
	0.05926	*0.14784*	*0.24344*	*0.94074*	*Ours*
Kweichow Moutai Co., Ltd.(600519)	0.14471	0.27379	0.38040	0.85529	FEDformer
	0.09736	0.25469	0.31203	0.90263	TCN
	0.09039	0.19260	0.30066	0.90961	GRU
	0.04857	*0.13863*	*0.22039*	*0.95143*	*Ours*
Semiconductor Manufacturing International Corporation(688981)	0.21891	0.22128	0.46788	0.78113	FEDformer
	0.37408	0.25816	0.61162	0.62599	TCN
	0.27940	0.18530	0.52858	0.72060	GRU
	0.06871	*0.11749*	*0.26212*	*0.93129*	*Ours*

4.4.2. Generalization Ability Test

In the HFT experiment, generalization ability is a crucial evaluation metric, reflecting whether the model's robustness is sufficiently high. To conduct generalization experiments, we tested our network on the four previously collected datasets, assessing its generalization performance with experimental parameters and hardware configurations consistent with the aforementioned setup. The following

Fig8,9,10,11 shows the results of the experiments on four publicly available datasets.

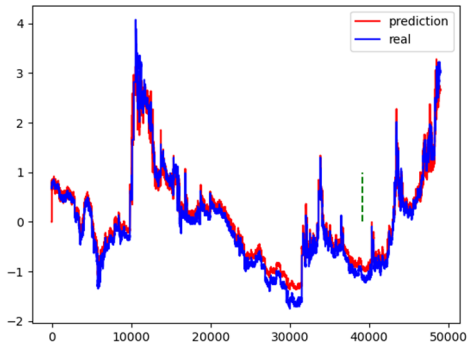

Fig 8. China High-Speed Railway Technology Co., Ltd.(000008)

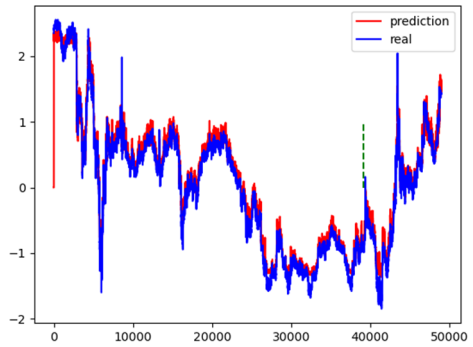

Fig 9. Shenzhen Cereals Holdings Co., Ltd. (000019)

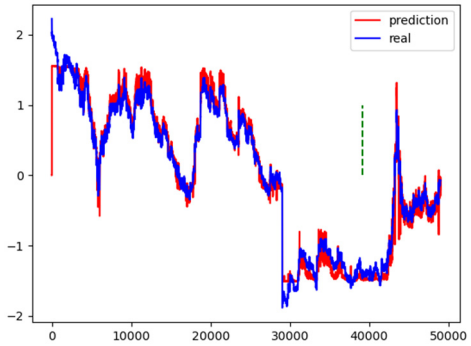

Fig 10. Topchoice Medical Co., Inc. (600763)

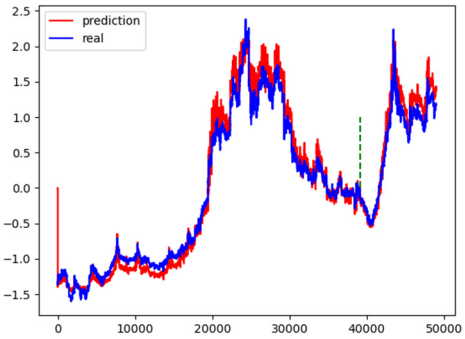

Fig 11. COSCO Shipping Holdings Co., Ltd.(601919)

It is clear from the data in the figure8,9,10,11 that our model has good prediction accuracy and goodness of fit in terms of prediction accuracy for the four publicly available datasets.

5. Conclusion

In this paper, we propose a novel stock prediction model tailored for high-frequency trading, addressing key challenges in training efficiency and robustness. To enhance the model's capability in handling high-noise stock data, we introduce Masked Moving-Average MLP (MM-MLP), which improves feature extraction by capturing complex patterns and reducing noise interference. This leads to more accurate stock price predictions.

Furthermore, we develop the Gauss Simulated Annealing Lion (GA-Lion) optimizer, an improved version of the Lion algorithm, designed to accelerate model training. GA-Lion significantly enhances convergence speed, making it particularly effective in high-frequency trading scenarios where rapid decision-making is crucial.

By integrating MM-MLP and GA-Lion with a Gated Recurrent Unit (GRU) model, our approach improves prediction accuracy and better adapts to the dynamic characteristics of financial data. Experimental results demonstrate that our model outperforms existing methods, achieving superior performance in terms of MSE and MAE, while also exhibiting strong generalization ability.

Overall, our work provides a robust and efficient deep learning framework for HFT, offering valuable insights for both individual and institutional investors. By enhancing predictive accuracy and responsiveness, our model aids in informed decision-making, ultimately reducing investment risks and improving financial performance in high-frequency trading environments.

References

[1] Securities U S, Exchange Commission. Part III: Concept release on equity market structure; Proposed Rule, 17 CFR Part 242[J]. Federal Register, 2010, 75(13): 3594-3614.

[2] Leal S J, Napoletano M. Market stability vs. market resilience: Regulatory policies experiments in an agent-based model with low-and high-frequency trading[J]. Journal of Economic Behavior & Organization, 2019, 157: 15-41.

[3] Arévalo A, Niño J, Hernández G, et al. High-frequency trading strategy based on deep neural networks[C]//International conference on intelligent computing. Cham: Springer International Publishing, 2016: 424-436.

[4] Thompson N C, Greenewald K, Lee K, et al. The computational limits of deep learning[J]. arxiv preprint arxiv:2007.05558, 2020, 10.

[5] Chen C, Zhang P, Zhang H, et al. Deep learning on computational-resource-limited platforms: A survey[J]. Mobile Information Systems, 2020, 2020(1): 8454327.

[6] Yang T, Zhou F, Du M, et al. Fluctuation in the global oil market, stock market volatility, and economic policy uncertainty: A study of the US and China[J]. The quarterly review of economics and finance, 2023, 87: 377-387.

[7] Bustos O, Pomares-Quimbaya A. Stock market movement forecast: A systematic review[J]. Expert Systems with Applications, 2020, 156: 113464.

[8] Zhang D, Lou S. The application research of neural network and BP algorithm in stock price pattern classification and prediction[J]. Future Generation Computer Systems, 2021, 115: 872-879.

[9] Kasman A. The impact of sudden changes on the persistence of volatility: Evidence from the BRIC countries[J]. Applied Economics Letters, 2009, 16(7): 759-764.

[10] Chun S, Oh S J, Yun S, et al. An empirical evaluation on robustness and uncertainty of regularization methods[J]. arxiv preprint arxiv:2003.03879, 2020.

[11] Ying X. An overview of overfitting and its solutions [C]//Journal of physics: Conference series. IOP Publishing, 2019, 1168: 022022.

[12] Xie Z, He F, Fu S, et al. Artificial neural variability for deep learning: On overfitting, noise memorization, and catastrophic forgetting[J]. Neural computation, 2021, 33(8): 2163-2192.

[13] Brogaard J, Nguyen T H, Putnins T J, et al. What moves stock prices? The roles of news, noise, and information[J]. The Review of Financial Studies, 2022, 35(9): 4341-4386.

[14] Verma R, Verma P. Noise trading and stock market volatility[J]. Journal of Multinational Financial Management, 2007, 17(3): 231-243.

[15] Orlitzky M. Corporate social responsibility, noise, and stock market volatility[J]. Academy of Management Perspectives, 2013, 27(3): 238-254.

[16] Sonkavde G, Dharrao D S, Bongale A M, et al. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications[J]. International Journal of Financial Studies, 2023, 11(3): 94.

[17] Karystinos G N, Pados D A. On overfitting, generalization, and randomly expanded training sets[J]. IEEE Transactions on Neural Networks, 2000, 11(5): 1050-1057.

[18] Xu H, Mannor S. Robustness and generalization[J]. Machine learning, 2012, 86: 391-423.

[19] Karolina Dziugaite G, Drouin A, Neal B, et al. In Search of Robust Measures of Generalization[J]. arxiv e-prints, 2020: arxiv: 2010.11924.

[20] Shen L, Sun Y, Yu Z, et al. On efficient training of large-scale deep learning models: A literature review[J]. arxiv preprint arxiv:2304.03589, 2023.

[21] Izmailov P, Podoprikhin D, Garipov T, et al. Averaging weights leads to wider optima and better generalization[J]. arxiv preprint arxiv:1803.05407, 2018.

[22] Shrestha A, Mahmood A. Review of deep learning algorithms and architectures[J]. IEEE access, 2019, 7: 53040-53065.

[23] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of machine learning research, 2011, 12(7).

[24] Tieleman T. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural networks for machine learning, 2012, 4(2): 26.

[25] Kingma D P, Ba J. Adam: A method for stochastic optimization [J]. arxiv preprint arxiv:1412.6980, 2014.

[26] Loshchilov I, Hutter F. Decoupled weight decay regularization [J]. arxiv preprint arxiv:1711.05101, 2017.

[27] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. Symbolic discovery of optimization algorithms. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23). Curran Associates Inc., Red Hook, NY, USA, Article 2140, 49205–49233.

[28] Chen L, Liu B, Liang K, et al. Lion secretly solves constrained optimization: As lyapunov predicts[J]. arxiv preprint arxiv: 2310.05898, 2023.

[29] Zhang C, Shao Y, Sun H, et al. The WuC-Adam algorithm based on joint improvement of Warmup and cosine annealing algorithms [J]. Math. Biosci. Eng, 2024, 21: 1270-1285.

[30] Wang J, Zhang S. An improved deep learning approach based on exponential moving average algorithm for atrial fibrillation signals identification[J]. Neurocomputing, 2022, 513: 127-136.

[31] Klinker F. Exponential moving average versus moving exponential average[J]. Mathematische Semesterberichte, 2011, 58: 97-107.

[32] Gong, Yunpeng et al. Beyond Dropout: Robust Convolutional Neural Networks Based on Local Feature Masking. 2024 International Joint Conference on Neural Networks (IJCNN) (2024): 1-8.

[33] Yadav S, Singh A, Singh S K, et al. Improving Market Efficiency and Profitability in High-Frequency Trading Using Neural Network-Based Deep Learning Techniques[C]//2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2024: 1-6.

[34] Kawaguchi K, Deng Z, Luh K, et al. Robustness implies generalization via data-dependent generalization bounds[C]// International conference on machine learning. PMLR, 2022: 10866-10894.

[35] Bertsimas D, Tsitsiklis J. Simulated annealing[J]. Statistical science, 1993, 8(1): 10-15.

[36] Crawshaw M, Liu M, Orabona F, et al. Robustness to unbounded smoothness of generalized signsgd[J]. Advances in neural information processing systems, 2022, 35: 9955-9968.

[37] Chen X, Hsieh C J, Gong B. When vision transformers outperform resnets without pre-training or strong data augmentations[J]. arXiv preprint arXiv:2106.01548, 2021.

[38] Foret P, Kleiner A, Mobahi H, et al. Sharpness-aware minimization for efficiently improving generalization[J]. arXiv preprint arXiv:2010.01412, 2020.

[39] Neelakantan A, Vilnis L, Le Q V, et al. Adding gradient noise improves learning for very deep networks[J]. arXiv preprint arXiv:1511.06807, 2015.