Meta-Adaptive Stock Movement Prediction with Two-Stage Representation Learning

Donglin Zhan[*]
Columbia University

Yusheng Dai[*]
University of Science and Technology of China

Yiwei Dong[*]
Renmin University of China

Jinghai He
University of California, Berkeley

Zhenyi Wang
University at Buffalo

James Anderson
Columbia University

Abstract

Stock movement prediction has always been a challenging but attractive task for researchers in machine learning and data mining. Generally speaking, two challenges for stock time series prediction remain not well-explored. One is the overfitting of deep learning models due to the data shortage and the other one is the potential domain shift that may happen during the evolution of stock time series. In this paper, we present ***M**eta-**A**daptive **S**tock movement prediction with two-Stag**E** **R**epresentation learning (**MASSER**)*, a novel framework for stock movement prediction based on self-supervised learning and meta-learning. Specifically, we first build up a two-stage representation learning framework; the first stage of representation learning aims for unified embedding learning for the data. And the second stage of learning, which is based on the first stage, is used for temporal domain shift detection via self-supervised learning. Then, we formalize the problem of stock movement prediction into a standard meta-learning setting. Inspired by importance sampling, we estimate sampling probability for tasks to balance the domain discrepancy caused by evolving temporal domains. Extensive experiment results on two open source datasets show that our framework with two simple but classical architectures (GRU and ResNet) as a model achieves improvements of 5% - 9.5% on average accuracy, compared to state-of-the-art baselines.

1 Introduction

One challenge for applying deep learning models to stock movement prediction is that there are always limited data and the models trained on small datasets are susceptible to overfitting [4]. Due to this difficulty, a common solution is applying pre-trained models [15]. Instead of training a large deep learning model from scratch, a model trained on relevant data in advance improves the model's performance in time series classification [11]. However, the conventional transfer learning setting may be inappropriate for stock prediction because of significant heterogeneity among different items. The ideal framework should have strong generalization with flexibility that can be fast adapted for all the stock with this issue of limited data.

Apart from the limited data, another issue is the domain shift in temporal patterns. In real-world scenarios, when researchers are making predictions on the stock market, an important concern is non-stationarity [28]. Due to the fact that the environment evolves with time, streaming data distribution

[*]Equal Contribution

Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

may change in unpredictable ways. In other words, the temporal distribution may shift within a large time scale [20, 34]. More precisely, three main problems lie in this issue, (i) how to detect the domain shift within time scales, (ii) how to perform well towards all the domain shifts since domain shifts could happen in both the training data and testing data, (iii) is there any importance or quantification for domain shifts when considering the shifting degree between different domains. A model aiming to train well on the training set can possibly have poor performance on testing sets because of the potential domain shifts in testing data. Thus, the model we desire here should have a strong generalization ability towards non-IID tasks from different distributions rather than improving performance on a specific target domain or distribution.

Before introducing our framework, we first identify two essential definitions of our problem setup.

Stock Movement Prediction Given a stock and its sequential features $X = \{x_1, \cdots, x_T\} \in \mathbb{R}^{D \times T}$ within T timestamps, where D denotes the dimension of the feature at each timestamp. The goal is to learn a prediction function $Y(X) = f(X; \theta)$, which maps the stock from its sequential features X to the label space, where the function f with parameters θ aims to predict the movement of stock s at the next timestamp either rise $(Y(X) = 1)$ or fall $(Y(X) = 0)$

Temporal Domain Shift Suppose $X = \{X_1, \cdots, X_n\}$ is the sequence of stock and movement label sequence $Y = \{Y_1, \cdots, Y_n\}$ corresponds to X. X_i and Y_i are drawn from their corresponding distribution $X_i \sim \mathbb{P}_i(X)$ and $Y_i \sim \mathbb{P}_i(Y)$. The temporal domain shift can be defined as $\exists\ k, l$: $\mathbb{P}_k(X) \neq \mathbb{P}_l(X)$ or $\mathbb{P}_k(X, Y) \neq \mathbb{P}_l(X, Y)$ or $\mathbb{P}_k(Y \mid X) \neq \mathbb{P}_l(Y \mid X)$.

To address the two mentioned issues, we propose ***M**eta-**A**daptive **S**tock movement prediction with two-Stag**E** **R**epresentation learning (**MASSER**)*, a stock prediction framework that combines supervised learning, self-supervised learning, and meta-learning. Extensive experiments show that MASSER outperforms the strong baseline models on several benchmarks in offline settings and online settings. The framework pipeline is shown in Figure 1. Furthermore, MASSER's performance is better than a SOTA [33] *even when MASSER has no access to the social media information in ACL18 dataset* [43]. We summarize our contributions as follows,

- We propose a two-stage encoder for stock price representation learning. The first stage aims at learning unified representations of data. The second stage is used to detect temporal domain shifts via self-supervised learning. To enhance the generalization ability, we formulate a meta-adaptive learning paradigm for stock movement prediction. The meta-learning framework efficiently learns knowledge across large temporal scales and different stocks and agilely adapts to unseen domains.
- Extensive experiments on two datasets demonstrate the effectiveness of the proposed method for improving prediction accuracy and increasing the generalization ability to temporal domain shift. We extend the offline setting of stock movement prediction to an online paradigm, admitting temporal domain shift could happen in testing streaming data, which is closer to the real day-trading scenarios.

2 Two-Stage Representation Learning for Encoders

2.1 First Stage: Macro Representation Learning

Due to the magnitude of the time scale of the datasets, forecasting decisions are often based on subsequence [43, 12, 33]. In our framework, we segment the raw data into several subsequences and feed them into the encoder. As it has been shown that temporal convolutional networks (TCN) can often produce superior prediction performance with sequential data [17] and are generally easier to train, we construct our encoder based on TCN. The first-stage encoder θ_1 aims to learn unified representation on the dataset level of the subsequences of raw data via (i) extracting useful features for prediction and (ii) matching the mapping between the inner structure of learned embeddings and their corresponding prediction labels. The loss function of the first stage is a convex combination of two terms, which match goals i) and (ii). Suppose $X = \{X_1, \cdots, X_n\}$ denotes the segmented subsequences input for the encoder and $Y' = \{Y'_1, \cdots, Y'_n\}$ as the next day rate of change (ROC) of each corresponding subsequence. The first term is mean squared error (MSE), which can be defined as $\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(Y'_i - \hat{Y}'_i \right)^2$, where $\hat{Y_i}'$ denotes the first-stage encoder's prediction of X_i. MSE is used to train the model to precisely predict the input subsequence. In order to achieve goal (ii), the second term is the Frobenius norm of the difference between the normalized embedding distance matrix and labels distance matrix within a mini-batch. Let $X_{i+1}, \cdots, X_{i+m}$ denote a mini-batch of subsequences, $E^{\theta_1}_{i+1}, \cdots, E^{\theta_1}_{i+m}$ the embeddings generated by the TCN θ_1, and $Y'_{i+1}, \cdots, Y'_{i+m}$ the ROC, where m denotes the batch size. Then we pair every two items l and k inside the same batch and compute the pairwise MSE loss on $(E^{\theta_1}_l, E^{\theta_1}_k)$ and (Y'_l, Y'_k). All the pairwise MSEs can be put into two matrices $E^{\theta_1}_b$ and Y'_b, and b represents the b-th mini-batch. These two matrices share the same size as $m \times m$. We take the Frobenius norm of $E^{\theta_1}_b - Y'_b$ and name it LabelSim: $\mathcal{L}_{LabelSim} = \| E^{\theta_1}_b - Y'_b \|^2_F$. The loss function term above forces the model to learn the alignment of intra-batch embeddings $E^{\theta_1}_b$ and ROC prediction label Y'_b. Through this objective, the subsequences with a similar ROC could be mapped near each other in the latent space. Thus, the encoder for macro representation learning $\theta_1$can be described as $\theta_1 = \arg\min_\theta \mathbb{E}_{X \in X} a * L_{MSE} + (1-a) * \mathcal{L}_{LabelSim}$, where a is a factor to control the importance of two terms.

2.2 Second Stage: Micro Representation Learning

The macro representation learning neglects the temporal order among individual subsequences so that the encoder hardly captures the evolution continuously. Therefore, the macro representation learning may be fragile towards the domain shifts. In the second stage, we modify the encoder to be more sensitive to the temporal domain shift between the continuous subsequences. We name the second stage micro representation learning. Similar to TS-CP2[9], we apply contrastive learning for temporal domain detection. We set two paired consecutive subsequences as dual windows, the embedding $E^{\theta_2}_i$ and its dual $E^{\theta_2}_{(d),i}$, to train the second-stage encoder θ_2 and make it adapt to domain shift detection. Specifically, InfoNCE [30] is used for the loss function to maximize the mutual information of dual windows. Suppose there is a mini-batch of embeddings $E^{\theta_2}_{i+1}, \cdots, E^{\theta_2}_{i+m}$ with batch size m, let $(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$ denotes the dual windows of X_i and $(E^{\theta_2}_k, E^{\theta_2}_{(d),l})_{k \neq l}$ represents the random pairs within the mini-batch. We take $(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$ as positive pair and $(E^{\theta_2}_i, E^{\theta_2}_{(d),l})_{i \neq l}$ as its corresponding negative pairs. The predicting probability q_i of positive pair $(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$ in each batch is as follows,

$q_i = \frac{\exp\left(d_{\cos}\left(E^{\theta_2}_i, E^{\theta_2}_{(d),i}\right)/\tau\right)}{\sum_{j=1}^m \exp\left(d_{\cos}\left(E^{\theta_2}_i, E^{\theta_2}_{(d),j}\right)/\tau\right)}$ where τ is a temperature hyper-parameter for scaling and $d_{\cos}$ is cosine similarity between each pair embeddings. The second-stage learning objective can be described as the minimization of binary cross entropy of the probabilities of all m positive pairs within the mini-batch. $\theta_2 = \arg\min_\theta -\mathbb{E}_{X \in X} \sum_{i,j} \mathbb{I}_{i=j} \log(q_i) + \mathbb{I}_{i \neq j} \log(1 - q_i)$ where $\mathbb{I}_{i=j}$ and $\mathbb{I}_{i \neq j}$ are indicator functions. We put more details in **Appendix** B.1.

With the encoder θ_2 in hand, we compute the similarity of dual windows $d_{\text{sim}}(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$ for all the input, where $E^{\theta_2}_i$ is the embedding for X_i from encoder θ_2. A threshold η is set for inferring whether the embedding contains a shift or not. For a specific embedding $E^{\theta_2}_i$, we infer the the subsequence X_i is under a temporal domain shift if $\eta \geq d_{\cos}(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$. Specific algorithms can be found in **Appendix** B.2.

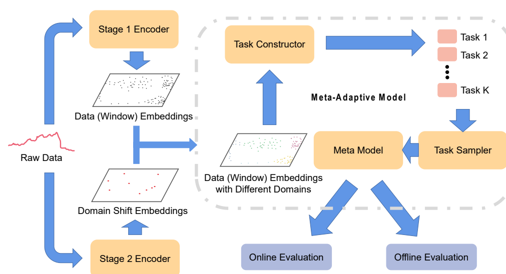

Figure 1: The figure demonstrates the whole process of the model prototype. First, we derive the embedding of raw time series data through the first stage encoder. Meta-learning tasks are constructed for each domain that is separated based on the results of the second stage encoder. We use a task sampler to select proper tasks for meta-training. Finally, the performance of the well-trained meta-model is evaluated in both offline and online experiment settings.

3 Meta-Adaptive Stock Movement Prediction

We elaborate on the meta-learning part of MASSER. Details about (i) the specific process of how to utilize the learned two-stage encoder for constructing meta-learning tasks and (ii) how to make the meta-model adaptive to the domain shifts can be found in **Appendix** C.

With meta-learning, we construct tasks for the meta-learning model. For classification, each task is further split into support set $\mathcal{S}$ and query set $\mathcal{Q}$ with items with balanced classes. Note that we don't allow any constructed task contains temporal domain shift (detected by stage 2 representation learning). All the constructed tasks are evaluated by specific criteria of (i) alignment of support set and query set, (ii) temporal adjacency, and (iii) representation adjacency. Each task will be given an appropriate probability for sampling to the training process of the meta-learning model after the task evaluation process. Tasks with good quality are more likely to be sampled for the training. Details can be found in **Appendix** C.3.

4 Experiment Result

We choose two open-source stock datasets for evaluation, **ACL18 dataset**[43][2] and **KDD17**[47][3]. To evaluate MASSER's performance, we compare it with the strong baselines for stock movement prediction: Momentum(MOM), LSTM[18], GRU[8], ALSTM[31], StockNet[43], Adv-ALSTM[12], and MAN-SF[33]. We evaluate the prediction performance by two metrics, Accuracy (Acc) and Matthews Correlation Coefficient (MCC)[43, 12, 33]. Note that better performance is evidenced by the higher value of the metrics.

We use a 4-layers TCN as our two-stage representation learning encoder, and the window size for the TCN encoder to generate embedding is 25. Furthermore, we apply ResNet for time series classification [40] and GRU[8] as the meta-model architectures with MAML[13] as meta-algorithm for updating parameters. Specifically, We use the Adam[21] for the MAML outer loop training and stochastic gradient descent (SGD) for the inner loop training. Utilizing the proposed task constructor, we formulate the meta tasks as 2-way 5-shot. After the meta-training process of MASSER, we first test the performance of the well-trained meta-model on testing data by directly using it to make predictions and then take an adaptation method to make the meta-model more specific to different stocks. Concretely, we randomly select data with the latest timestamps in the training set within the individual stock and slightly update the model parameters based on these selected data to help MASSER better capture the feature of each stock. We report the mean of best testing performance over six different random seeds. Additional description about the experiment setup and results of online setting, backtesting, and ablation study can be found in **Appendix** D.

4.1 Offline Experiment Settings

The offline setting here means the model is frozen after the entire learning process and output prediction on all the testing data simultaneously. Table 1 compares the Acc and MCC of our model and baselines for stock movement prediction in **ACL18** and **KDD17**. MASSER-ResNet* (* denotes adaptation) achieves the best accuracy and MCC. Compared to the baselines, MASSER-ResNet* exhibits an improvement of **9.1%** and **64.9%** (**2.3%** and **50.0%**) on the **ACL18** (**KDD17**) dataset regarding Acc and MCC, respectively. It is worth noting that the Acc (MCC) of MASSER is **2.6%** (**25.1%**) higher than MAN-SF, even though MAN-SF utilizes social media information. Another observation worth mentioning here is that the MASSER-GRU without adaptation averagely increases 1% Acc on Adv-ALSTM. This justifies the effectiveness of two-stage representation learning and meta-adaptive training, which may be caused by well-learned embedding and meta-adaptive learning on good-quality tasks. With two simple architectures as the meta-model, MASSER shows its strong generalization ability for unseen data and good ability to learn representations, which beats complex architectures with attention module[27] and transformer[39].

[2]https://github.com/yumoxu/stocknet-dataset

[3]https://github.com/z331565360/State-Frequency-Memory-stock-prediction/tree/master/dataset

	ACL18		KDD17	
Model	Acc	MCC	Acc	MCC
MOM	0.470	-0.064	0.498	-0.013
LSTM	0.532	0.067	0.516	0.018
ALSTM	0.549	0.104	0.519	0.026
StockNet	0.550	0.017	0.499	0.499
Adv-ALSTM	0.572	0.148	0.531	0.052
MAN-SF	0.608	0.195	$--$	$--$
MASSER-ResNet	0.552	0.099	0.535	0.074
MASSER-GRU	0.579	0.141	$\mathbf{0.543}$	0.073
MASSER-ResNet*	$\mathbf{0.624}$	$\mathbf{0.244}$	0.542	$\mathbf{0.078}$
MASSER-GRU*	0.581	0.162	$\mathbf{0.543}$	0.047

Table 1: Offline Setting Acc and MCC on **ACL18** and **KDD17** (* means adaptation)

5 Conclusion

In this paper, We introduce MASSER, a novel framework for stock movement prediction based on self-supervised learning and meta-learning. Extensive experiments on two datasets show that MASSER outperforms baselines in both offline and online settings, which justifies MASSER's strong generalization ability. We propose two-stage representation learning to train encoders for downstream meta-framework and detect domain shifts. We construct tasks, evaluate their quality, and then sample the task adaptively to feed the meta-framework. For future work, we would like to explore the causality over MASSER via a graph neural network.

References

[1] Sathyanarayanan N Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1197–1206, 2019.

[2] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[3] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[4] Yujin Baek and Ha Young Kim. Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. *Expert Systems with Applications*, 113:457–480, 2018.

[5] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. In *International Conference on Artificial Intelligence and Statistics*, pages 3080–3090. PMLR, 2022.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] Yizhou Chen, Shizhuo Zhang, and Bryan Kian Hsiang Low. Near-optimal task selection for meta-learning with mutual information and online variational bayesian unlearning. In *International Conference on Artificial Intelligence and Statistics*, pages 9091–9113. PMLR, 2022.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[9] Shohreh Deldari, Daniel V Smith, Hao Xue, and Flora D Salim. Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021*, pages 3124–3135, 2021.

[10] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 402–411, 2021.

[11] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE international conference on big data (Big Data)*, pages 1367–1376. IEEE, 2018.

[12] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. Enhancing stock movement prediction with adversarial training. *arXiv preprint arXiv:1810.09936*, 2018.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[14] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proceedings of International Conference on Machine Learning*, 2019.

[15] Elizabeth Fons, Paula Dawson, Xiao-jun Zeng, John Keane, and Alexandros Iosifidis. Augmenting transferred representations for stock classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3915–3919. IEEE, 2021.

[16] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[17] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Jean Kaddour, Steindór Sæmundsson, et al. Probabilistic active meta-learning. *Advances in Neural Information Processing Systems*, 33:20813–20822, 2020.

[20] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. Stock market prediction system with modular neural networks. In *1990 IJCNN international joint conference on neural networks*, pages 1–6. IEEE, 1990.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[22] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] Wendi Li, Xiao Yang, Weiqing Liu, Yingce Xia, and Jiang Bian. Ddg-da: Data distribution generation for predictable concept drift adaptation. *arXiv preprint arXiv:2201.04038*, 2022.

[24] Chenghao Liu, Zhihao Wang, Doyen Sahoo, Yuan Fang, Kun Zhang, and Steven CH Hoi. Adaptive task sampling for meta-learning. In *European Conference on Computer Vision*, pages 752–769. Springer, 2020.

[25] Ricardo Luna Gutierrez and Matteo Leonetti. Information-theoretic task selection for meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 33:20532–20542, 2020.

[26] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26, 2013.

[27] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[28] Guy P Nason. Stationary and non-stationary time series. *Statistics in volcanology*, 60, 2006.

[29] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. IEEE, 2017.

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[31] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[32] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[33] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. Deep attentive learning for stock movement prediction from social media text and company correlations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8415–8426, 2020.

[34] Robert P Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19, 2009.

[35] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[36] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.

[37] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[38] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[41] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[42] Hanwei Wu, Ather Gattami, and Markus Flierl. Conditional mutual information-based contrastive loss for financial time series forecasting. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–7, 2020.

[43] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, 2018.

[44] Huaxiu Yao, Yu Wang, Ying Wei, Peilin Zhao, Mehrdad Mahdavi, Defu Lian, and Chelsea Finn. Meta-learning with an adaptive task scheduler. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] Jaemin Yoo, Yejun Soun, Yong-chan Park, and U Kang. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2037–2045, 2021.

[46] Xiaoyu You, Mi Zhang, Daizong Ding, Fuli Feng, and Yuanmin Huang. Learning to learn the future: Modeling concept drifts in time series prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2434–2443, 2021.

[47] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2141–2149, 2017.

A Related Work

Stock Movement and Trading Modeling The previous work of stock movement can be separated into two categories. Individual stock modeling, which captures complex repetitive patterns from the historical prices of each individual stock instead of finding the correlations between multiple stocks [29, 43, 12, 42]. And taking the correlation between different stocks into consideration because correlations between multiple stocks may help to make robust and consistent predictions [31, 33, 45]. MASSER extends the view of correlation to meta-learn unified representations on the dataset level. In MASSER, not only is domain shift detection is considered, but also the importance of different domains, which could make our framework more consistent with the general pattern of historical data. Furthermore, our framework is flexible with online adaptation and offline scenarios. All the related works mentioned above are inappropriate with online settings, and those models haven't taken domain shifts within testing data into consideration.

Time Series Modeling with Domain Shifts Deep learning-based methods for time series modeling gained popularity over the past years [35, 36]. Recently, domain shift and concept drift modeling for time series have become a popular topic in the community of data mining [10, 46]. And more recently, Li et al. [23] construct a resampling strategy for data distribution generation for predictable concept drift adaptation. In MASSER, we aim to solve the temporal domain shift issue as well. The high stochasticity of stock data is a significant property. Therefore, we design a two-stage encoder learning process to handle this issue. The first stage encoder can learn the general features and denoise the data for the second stage domain shift detection. The second stage follows the first one by taking the encoder as a pre-trained model and learning to detect domain shifts via self-supervised learning.

Meta-learning By using the information gained from earlier assignments, meta-learning has emerged as an efficient model for learning with minimal data [13, 37, 22, 14, 5]. In common scenarios for meta-learning, the assumption that all tasks are equally important is held, therefore each iteration samples tasks at random. In [32, 19, 25], the probability of sampling a task from existing meta-training tasks is proportional to the amount of information it provides. Recently, some work [24, 44, 7] focused on task selection via reinforcement learning [41] and submodular maximization [3]. From the perspective of meta-learning, MASSER focuses on the same issue of task sampling but additionally takes domain shifts into consideration. We do not assume that tasks are equally significant. MASSER adaptively samples tasks for the meta-model based on learned temporal patterns via self-supervised representations to enhance the learning of general patterns against out-of-distribution (OOD) data.

Self-Supervised Learning Self-supervised learning has been utilized to capture informative and compact representations of video [1], image [6], and time series [9]. Specifically, we consider contrastive learning for our framework. Contrastive learning uses a collection of training examples made up of positive and negative sample pairs to establish what makes the samples in a dataset similar or dissimilar. N-Paired loss [38] and InfoNCE based on Noise Contrastive Estimation [16, 26] are examples of recent multiple negative learning loss functions that examine numerous negative sample pairs at the same time. Auto-regressive models are used in Contrastive Predictive Coding (CPC) [30, 17] to learn representations within a latent embedding space. CPC aims to learn inside a global, abstract representation of the signal rather than a high-dimensional, lower-level representation. Recently, TSCP2 [9] uses CPC to learn latent representations and find dissimilarities in time series data.

B Detailed Two-stage Representation Learning

Figure 2 shows the pipeline and illustration of our two-stage representation learning.

B.1 The InfoNCE Loss

The InfoNCE loss was first proposed by Oord et al. [30]. It has been proven that by minimizing the InfoNCE loss, we are actually maximizing the mutual information between dual windows. Take the same annotations in main context, $X = \{X_1, \cdots, X_n\}$ denotes the segmented subsequences input for the encoder, $E^{\theta_2}_{i+1}, \cdots, E^{\theta_2}_{i+m}$ is a mini-batch of embeddings with batch size m, let $(E^{\theta_2}_i, E^{\theta_2}_{(d),i})$

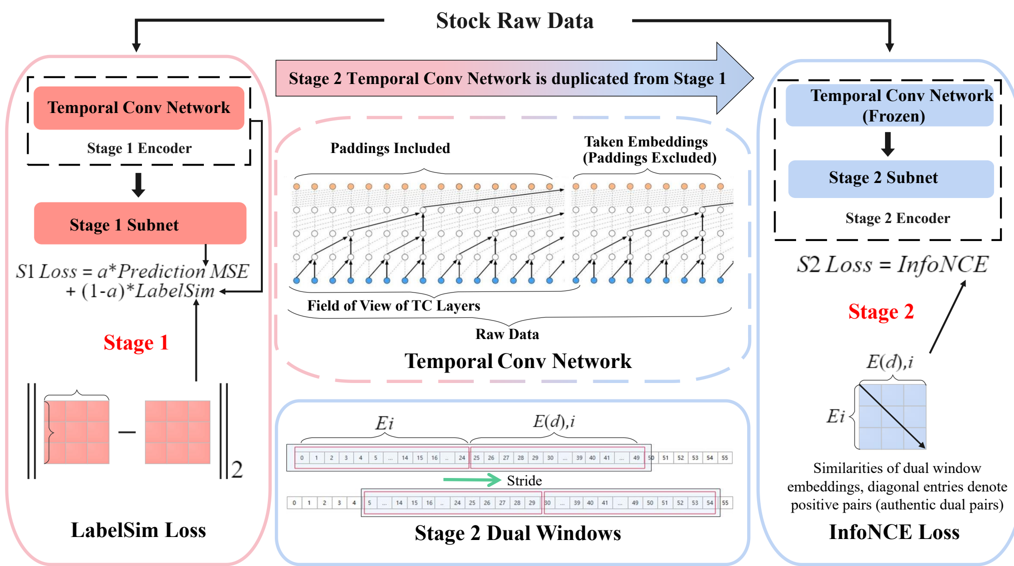

Figure 2: The two-stage encoder training framework pipeline. On the left side of the figure, the first-stage encoder is trained with a weighted sum of Label similarity loss and prediction loss to learn unified representations. Then on the right side, the second-stage encoder is trained based on the Temporal Conv encoder duplicated from the first stage to detect temporal domain shift via self-supervised learning of dual windows. The specific network structure of the Temporal Conv encoder is shown on the top of the middle.

denotes the dual windows of X_i and is taken as positive pair, $(E_k^{\theta_2}, E_{(d),l}^{\theta_2})_{k \neq l}$ represents the random pairs within the mini-batch and they are all negative pairs. By the original definition of the InfoNCE loss, we are about to minimize:

$$\mathcal{L}_{\mathrm{N}} = -\mathbb{E}\left[\log \frac{\exp\left(d_{\cos}\left(E_i^{\theta_2}, E_{(d),i}^{\theta_2}\right)/\tau\right)}{\sum_{j=1}^{m} \exp\left(d_{\cos}\left(E_i^{\theta_2}, E_{(d),j}^{\theta_2}\right)/\tau\right)}\right] \tag{1}$$

Armed with the knowledge of multi-class classification, it is easy to show that Equation 1 is the cross-entropy loss of classifying the positive pair correctly, and the predicting probability q_i of positive pair $(E_i^{\theta_2}, E_{(d),i}^{\theta_2})$ in each mini-batch is exactly the interior part of $\mathcal{L}_{\mathrm{N}}$:

$$q_i = \frac{\exp\left(d_{\cos}\left(E_i^{\theta_2}, E_{(d),i}^{\theta_2}\right)/\tau\right)}{\sum_{j=1}^{m} \exp\left(d_{\cos}\left(E_i^{\theta_2}, E_{(d),j}^{\theta_2}\right)/\tau\right)}$$

Finally, the optimal parameter θ_2 can be obtained by:

$$\theta_2 = \arg\min_{\theta} -\mathbb{E}_{X \in X} \sum_{i,j} \mathbb{I}_{i=j} \log(q_i) + \mathbb{I}_{i \neq j} \log(1 - q_i)$$

B.2 Details about Detecting Temporal Domain Shift

As shown in Figure 3, we slide dual windows in the time domain and apply the second-stage encoder to transform them into embedding pairs $(E_i^{\theta_2}, E_{(d),i}^{\theta_2})$. Next we compute the similarity of adjacent embedding from the same pair and filter the pairs whose similarity is below the threshold η, which can be computed as:

$$\eta = \varepsilon * (\delta * \eta_{pos} + (1 - \delta) * \eta_{neg})$$

where η_{pos} is the average of positive pairs' similarities in the training set, and η_{neg} represents the negative ones'. The adjustability coefficient ε and δ are set to 0.35 and 0.66 empirically. For each

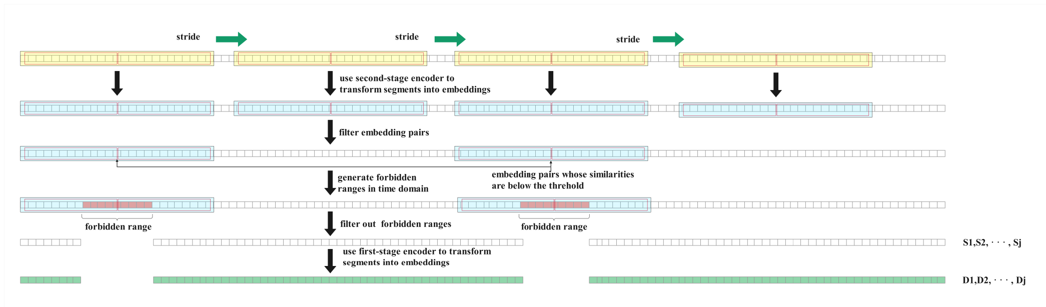

Figure 3: The specific process of how to utilize the learned two-stage encoder for constructing meta-learning tasks.

filtered embedding pair, we take the time point t_i which is corresponded to the junction of $E_i^{\theta_2}$ and $E_{(d),i}^{\theta_2}$ as the midpoint and extent it to a forbidden time range $(t_i - \tau, t_i + \tau)$, which possibly contains domain shift. Let set S be the union of all the timestamps of the raw stock data. After dropping the timestamps in forbidden ranges, stock data in the time domain is split into the consecutive segments $S_1, \cdots, S_j$. Then, generate all their embeddings using the first-stage encoder consecutively for constructing meta-learning tasks.

C Detailed Meta-Adaptive Stock Movement Prediction

In this section, we elaborate on the meta-learning part of MASSER, (i) the specific process of how to utilize the learned two-stage encoder for constructing meta-learning tasks, and (ii) the details of how to make the meta-model adaptive to the domain shifts. The motivation for (i) is that tasks containing domain shifts are not desired for the construction because of the assumption that the data inside the support set and query set is roughly IID in a meta-learning setting. When facing multiple domains following different distributions, the model needs to focus more on the general patterns within the whole time scale and prioritize the tasks with good quality. To demonstrate (ii), we first introduce how to evaluate task quality and then show how to make the meta-model adaptive.

C.1 Meta-learning Task Construction

We compute the similarity of dual windows $\{d_{\cos}(E_i^{\theta_2}, E_{(d),i}^{\theta_2})\}_{i=1,\cdots,n}$ and search all the dual windows whose similarity is below the threshold η. Let set **S** be the union of all the timestamps that are covered in inferred embedding entries $\{E_i^{\theta_2}\}_{i=1,\cdots,n}$ that is below the threshold and take the complement of **S** within all the timestamps as the feasible set for the following task construction. Let $S_1, \cdots, S_j$ be the consecutive segments within the feasible set. Then, generate all their embeddings from first-stage encoder θ_1 consecutively and we consider the embedding group of each segment S_i as a domain $\mathcal{D}_i$, which doesn't contain shift inside.

With meta-learning, we construct tasks for the meta-learning model. For classification, each task is further split into support set $\mathcal{S}$ and query set $\mathcal{Q}$ with items with balanced classes. For task $\mathcal{T}_{ij}$, all the data (subsequences) is selected from $\mathcal{D}_i$ and ij denotes the j-th task constructed inside domain $\mathcal{D}_i$. Suppose there are $n*$ classes in the whole dataset, and we intend to build a n way k shot task $\mathcal{T}_{ij}$ ($n* \geq n$). We first build the support set. As for the time series data, we set the data in the support set ahead of the data in the query set. We build a candidate pool for each class and scan the domain $\mathcal{D}_i$ by temporal order to divide data into the class candidate pools. The dividing process stops by checking all the class candidate pools have a number of data no less than k. For the class candidate pools that have a number of data larger than k, we use random sampling to draw k samples into the support set $\mathcal{S}_{ij}$. Then we start to build to query set $\mathcal{Q}_{ij}$ for task $\mathcal{T}_{ij}$ by the same process. Tasks in $\mathcal{D}_i$ are constructed in order, and we allow different tasks' support sets have overlapped.

We show the detailed procedures of task construction in **Algorithm** 1.

Algorithm 1 Task Construction

Require: Domains $\mathcal{D}_1, \cdots, \mathcal{D}_m$, n way k shot, k' for query set , window size l, retreated ratio α

1: **for** $t = 1, 2, \ldots m$ **do**
2: Construct class candidate pools $C_1, \cdots, C_{n*}$ for all the classes
3: **//Stage I: Support Set Construction**
4: Sample n classes for current task $\mathcal{T}_{tj}$
5: **while** not all the item numbers for sampled class candidate pools $|C_i| \geq$ k **do**
6: Add data into corresponding class candidate pool C_i
7: **end while**
8: **for** sampled class candidate pools $|C_i| >$ k **do**
9: Random draw k samples to keep in the class candidate pool and delete the rest
10: **end for**
11: Merge the current sampled class candidate pool C_i and save as task $\mathcal{T}_{tj}$'s support set $\mathcal{S}_{tj}$
12: **//Stage II: Query Set Construction**
13: Clean all the sampled class candidate pools C_i
14: **while** not all the item numbers for sampled class candidate pools $|C_i| \geq k'$ **do**
15: Add data into corresponding class candidate pool C_i
16: **end while**
17: Merge the current sampled class candidate pool C_i and save as task $\mathcal{T}_{tj}$'s query set $\mathcal{Q}_{tj}$
18: Initialize a new starting time stamp for the new task $\mathcal{T}_{t(j+1)}$ with the retreated ratio α and last task $\mathcal{T}_{tj}$'s starting point
19: Check whether the data in $\mathcal{D}_t$ after the starting point is enough for constructing a new task or not. If enough, return to line 4 for $\mathcal{T}_{t(j+1)}$ construction. If not, shift to $\mathcal{D}_{t+1}$
20: **end for**

C.2 Task Evaluation

We present detailed criteria for evaluating task quality after task construction according to the following three factors.

- **Alignment of Support Sets and Query Sets** Task $\mathcal{T}_{ij}$ similarity between the support and query set embeddings $E^{\theta_1}_{\mathcal{S}_{ij}}$, $E^{\theta_1}_{\mathcal{Q}_{ij}}$ with respect to the first-stage encoder, i.e.,

$$S_C(E^{\theta_1}_{\mathcal{S}_{ij}}, E^{\theta_1}_{\mathcal{Q}_{ij}}) := \frac{E^{\theta_1}_{\mathcal{S}_{ij}} \cdot E^{\theta_1}_{\mathcal{Q}_{ij}}}{\|E^{\theta_1}_{\mathcal{S}_{ij}}\|\|E^{\theta_1}_{\mathcal{Q}_{ij}}\|} \tag{2}$$

  Here, we use cosine similarity as metric (2). Specifically, the embedding similarity signifies the generalization gap from the support set to the query set since segmentation results from the second-stage encoder could not strictly guarantee the unification and stationary of data in the support set and query set. Ideally, good quality tasks we desire should have close embeddings for support sets and query sets.

- **Temporal Adjacency** When facing a large temporal scale time series, the data with later timestamps may be more important than the former ones, particularly for those that have domain shifts inside the time series but no shift within the latest parts. In the general case, any monotonically increasing function can be used here. For simplicity, we apply a linear monotonically increasing function, i.e., $S_{TA}(\mathcal{T}_{ij}) = k * t_{\mathcal{T}_{ij}}$, where k ($k > 0$) is the slope of the linear function and t is the average timestamp of $\mathcal{T}_{ij}$.

- **Representation Adjacency** Since focusing on general patterns at the dataset level, we want to degrade the importance of OOD data. For simplicity, we consider this issue from the inter-domain level rather than the intra-domain (task) level. We propose to compute how close a single domain is to the others. In terms of efficiency, assuming a large number of domains inside the dataset, it is computationally costive to compute all the distances between a specific domain and the rest. Thus, we intend to partition domains randomly into several groups and compute the sum of the distance inside each group. Suppose for each randomly partitioned group whose size is m, and then we can input all the pairwise domain embedding distance into a symmetric square matrix with the size of $m * m$. To sum up all the columns or rows, the distance between a specific domain and all the rest is acquired. Let $S_{RA}(\mathcal{T}_{ij})$ denote the distance sum of task $\mathcal{T}_{ij}$ within its random partitioned group.

We elaborate the steps of task evaluation in **Algorithm** 2

Algorithm 2 Task Evaluation

Require: Tasks $\mathcal{T}_{11}, \cdots, \mathcal{T}_{1j_1}, \mathcal{T}_{21}, \cdots, \mathcal{T}_{mj_m}$, Domains $\mathcal{D}_1, \cdots, \mathcal{D}_m$, Starting Timestamps of Tasks $t_{\mathcal{T}_{11}}, \cdots, t_{\mathcal{T}_{1j_1}}, t_{\mathcal{T}_{21}}, \cdots, t_{\mathcal{T}_{mj_m}}$, Temporal Adjacency Slope k, Representation Adjacency Group Size m', Factor Rate $\beta_1, \beta_2, \beta_3$

//Alignment of Support and Query and Temporal Adjacency

for $\mathcal{T}_{11}, \cdots, \mathcal{T}_{1j_1}, \mathcal{T}_{21}, \cdots, \mathcal{T}_{mj_m}$ **do**

Get support embedding $E^{\theta_1}_{\mathcal{S}_{ij}}$ and query embedding $E^{\theta_1}_{\mathcal{Q}_{ij}}$ for task $\mathcal{T}_{ij}$ through encoder θ_1

Compute alignment of support set and query set:

$$S_C(E^{\theta_1}_{\mathcal{S}_{ij}}, E^{\theta_1}_{\mathcal{Q}_{ij}}) := \frac{E^{\theta_1}_{\mathcal{S}_{ij}} \cdot E^{\theta_1}_{\mathcal{Q}_{ij}}}{\|E^{\theta_1}_{\mathcal{S}_{ij}}\| \|E^{\theta_1}_{\mathcal{Q}_{ij}}\|} \tag{3}$$

Compute temporal adjacency:

$$S_{TA}(\mathcal{T}_{ij}) = k * t_{\mathcal{T}_{ij}} \tag{4}$$

end for

//Representation Adjacency

Random partition tasks $\mathcal{T}_{11}, \cdots, \mathcal{T}_{1j_1}, \mathcal{T}_{21}, \cdots, \mathcal{T}_{mj_m}$ into different groups with the group size m' (assume there are l groups)

for $k = 1, \cdots, l$ **do**

Compute pairwise task similarity $S_C(E^{\theta_1}_{\mathcal{T}_{ij}}, E^{\theta_1}_{\mathcal{T}_{i'j'}})$ and fill the similarity into the pairwise similarity matrix RA_k

Compute the representation adjacency $S_{RA}(\mathcal{T}_{ij})$ by sum up the similarity of itself with the other tasks within group k (sum up the corresponding row/column of RA_k)

end for

for $\mathcal{T}_{11}, \cdots, \mathcal{T}_{1j_1}, \mathcal{T}_{21}, \cdots, \mathcal{T}_{mj_m}$ **do**

Compute the sampling probability of task $\mathcal{T}_{ij}$ with normalization by:

$$w_{ij} = \beta_1 S_C(E^{\theta_1}_{\mathcal{S}_{ij}}, E^{\theta_1}_{\mathcal{Q}_{ij}}) + \beta_2 S_{TA}(\mathcal{T}_{ij}) + \beta_3 S_{RA}(\mathcal{T}_{ij}) \tag{5}$$

end for

C.3 Training Meta-Models Adaptively

Considering three factors simultaneously, we define the sampling probability w_{ij} of each task $\mathcal{T}_{ij}$ as

$$w_{ij} = \beta_1 S_C(E^{\theta_1}_{\mathcal{S}_{ij}}, E^{\theta_1}_{\mathcal{Q}_{ij}}) + \beta_2 S_{TA}(\mathcal{T}_{ij}) + \beta_3 S_{RA}(\mathcal{T}_{ij}) \tag{6}$$

where β_1, β_2, β_3 are parameters controlling its corresponding factor.

We use sampling probability weight w_{ij} to sample B tasks from the task pool for current meta-training iteration, where a larger value of w_{ij} represents higher probability.

We demonstrate our pipeline of Meta-Adaptive Framework in **Algorithm** 3 with MAML[13] as the meta-learning algorithm. The two-stage representation learning process is not included.

Algorithm 3 Meta-Adaptive Pipeline

Require: Domains $\mathcal{D}_1, \cdots, \mathcal{D}_m$, Input Parameters of Task Construction ϕ_1, Input Parameters of Task Evaluation ϕ_2, Task Sampling Size $m_{\prime\prime}$

Get tasks $\mathcal{T}_{11}, \cdots, \mathcal{T}_{1j_1}, \mathcal{T}_{21}, \cdots, \mathcal{T}_{mj_m}$ by Task Construction (**Algorithm** 1) with input parameters ϕ_1

Get task sampling probability $w_{11}, \cdots, w_{1j_1}, w_{21}, \cdots, w_{mj_m}$ by Task Evaluation (**Algorithm** 2) with input parameters ϕ_2

Sample $m_{\prime\prime}$ tasks according to task sampling probability $w_{11}, \cdots, w_{1j_1}, w_{21}, \cdots, w_{mj_m}$

Update Meta-Adaptive model with MAML

Features	Calculation
z_{open}	z_{open} = open $_t$/ close $_t - 1$
z_{high}	z_{high} = high $_t$/ close $_t - 1$
z_{low}	z_{low} = low $_t$/ close $_t - 1$
z_{close}	z_{close} = close $_t$/ close $_{t-1} - 1$
z_{adj_close}	z_{adj_close} = adj_close $_t$/ adj_close $_{t-1} - 1$
z_{volume}	/

Table 2: Features of ACL18 and KDD17 Dataset

D Additional Experiment Results

D.1 Dataset Description

We choose two open-source stock datasets for evaluation, **ACL18 dataset**[43][4] and **KDD17**[47][5]. **ACL18** contains historical price and social media information from Jan-01-2014 to Jan-01-2016 of 88 high-trade-volume stocks in NYSE and NASDAQ markets. **KDD17** includes longer history ranging from Jan-01-2007 to Jan-01-2016 of 50 stocks in U.S. markets. Six features are included in this dataset. Three technical indicators $z_{open}, z_{high}, z_{low}$ are inter-day-based features. Two technical indicators $_{close}$ and z_{adj_close} are intra-day-based features, where adj_close denotes the adjusted closing price, which reflects the stock's value after accounting for any corporate actions, such as stock splits, dividend distribution, and rights offerings. For more details, please reach Table 2. The labels are assigned to each time window based on the next day's movement percent. Movement percent $\geq 0.55\%$ and $\leq -0.5\%$ are labeled as positive and negative examples. In the offline setting experiment of **ACL18**, we use the time period Jan-01-2014 to Aug-01-2015 as the training dataset and Oct-01-2015 to Jan-01-2016 as the testing dataset. And for **KDD17**, we split the date from Jan-04-2007 to Dec-31-2014 as the training set, Jan-02-2015 to Jan-04-2016 as a validation set, and Jan-04-2016 to Dec-30-2016 as the testing set. The setting above is the same as [43] and [12]. Furthermore, for our proposed online setting, we set Oct-01-2015 to Sep-01-2016 as the online testing set.

D.2 Baselines

We compare MASSER with the following baselines for stock movement prediction.

- **MOM** Momentum (MOM) is a technical indicator that predicts negative or positive for each example with the trend in the last 10 days
- **LSTM** [18] is a baseline for stock movement prediction, which uses a neural network with an LSTM layer and a prediction layer.
- **ALSTM** [31] represents attention LSTM that correlates multiple time steps using a temporal attention mechanism.
- **StockNet** [43] uses variational autoencoders to encode stock movements as latent probabilistic vectors.
- **Adv-ALSTM** [12] is the previous SOTA (with only stock price) that trains ALSTM with adversarial data to improve the robustness of stock movement prediction.
- **MAN-SF** [33] is the previous SOTA (with social media information) that uses graph neural network to blend chaotic temporal signals, social media, and inter-stock relationship.

D.3 Implementation Details

We set the learning rate for the meta-adaptive model as 0.004- 0.006 and batch size as 16, 32 and 64. The learning rate for online settings is 0.001. Grid search is used to find the appropriate learning rates and batch size of the model.

[4]https://github.com/yumoxu/stocknet-dataset

[5]https://github.com/z331565360/State-Frequency-Memory-stock-prediction/tree/master/dataset

D.4 Online Experiment Setting

We further demonstrate MASSER can deal with streaming data through an online experiment setting as well. Under the online setting, future data flows in batches according to temporal order. When the model gets access to a new batch of future data, we first use the model updated with the last batch of data to make movement predictions and then leverage the information of this new batch of data to update the model once again. When all the future data has been tested, we can calculate the Acc and MCC of the online setting. To cope with domain shift, we additionally introduce Bayesian Online Changepoint Detection (BOCPD) [2] to assist MASSER in this online setting as an advanced detector. If there is no domain shift within a batch of new data according to the BOCPD result, we use the former updated model to make predictions on this batch of data and update the model as usual. However, When a domain shift occurs within a batch of new data, we use the previously updated model to make predictions on the data before the domain shift point and use the meta-model without any update to predict the data after the time domain shift occurs, and re-update the meta-model from scratch.

Table 3 compares the Acc and MCC of our model and baselines for stock movement prediction on **ACL18** in the online setting. MASSER-ResNet outperforms baselines, with an average improvement of **18.6%** on Acc. Furthermore, BOCPD helps MASSER-ResNet to be more agile toward potential domain shifts in the streaming data. From the experiment results of the online setting, we can tell that meta-adaptive training precisely capture the general pattern of training data and make the model a powerful generalization ability.

	ACL18	
Model	Acc	MCC
LSTM	0.516	0.045
GRU	0.509	0.041
ALSTM	0.487	0.012
MASSER-ResNet	0.612	0.216
MASSER-ResNet-BOCPD	$\mathbf{0.625}$	$\mathbf{0.251}$

Table 3: Online Setting Acc and MCC on **ACL18**

D.4.1 Backtesting

To illustrate how accurate stock movement prediction contributes to real investment tasks, we design a straightforward strategy by making long-short decisions only based on the signals from the model. Considering to trade on one type of stock, this strategy sells all holding shares for cash if the model predicts the stock price to fall and purchases the maximum affordable shares if the model provides rising signals. We construct the portfolio based on this strategy with predictions from MASSER-ResNet and three baseline predictive models (LSTM, GRU, and ALSTM). On day t, our strategy reacts directly based on the movement prediction $\hat{y}_t$. We assume that at the start of the backtesting, the trader has 1 unit of asset, and on day t, the market value of his/her asset is w_t. If the utilized model predicts the price of the stock to rise from day t to $t+1$, say $\hat{y}_t = 1$, the trader will keep all his/her wealth as stock. The action 'keep' indicates that if the trader holds stocks on day t, he/she will not take action for the position, and if the trader holds cash, he/she will purchase stocks with all cash on day t. Otherwise ($\hat{y}_t = 0$), he/she will keep all his/her wealth as cash. Following this strategy, if one model can predict all rises and falls accurately, we may take advantage of all the rises and avoid all the falls, and the maximum final accumulative return will be $\prod_{t=0}^{T-1}(1 + r_t^+)$, where $r_t^+ = \max(r_t, 0)$ and r_t represents the daily return from day t to $t+1$. Intuitively, the more accurate one movement prediction model is, the better investment return we can obtain through this signaling strategy.

We list and compare the performance of our algorithm and baselines in terms of the average achieved return rate of all 88 considered stocks within the 205-day testing period in Table 4. MASSER outperforms all baselines, achieving a 29.52 % return rate, beating LSTM by 20.71%, GRU by 20.97% and ALSTM by 16.38 %.

Models	Achieved return rate
Stock	5.59%
LSTM	8.81%
GRU	8.55%
ALSTM	13.14%
MASSER-ResNet	**29.52** %

Table 4: Average Online Return Rate on **ACL18**

D.5 Ablation Study

We conduct extensive ablation experiments on the overall proposed framework, including whether to use macro representation learning (1st encoder) to derive good representations for downstream meta-learning, whether to perform micro representation learning (2nd encoder), whether to use the task sampler to adaptively meta-train the model (sampler), and whether to use a domain shift detector in the online setting (BOCPD). The experiment results are shown in Table 5. It is worth noting that the complete MASSER framework yields the best result among all the combinations in both offline and online settings. Models with macro representation learning already achieve better performance compared with other baseline models. After the second-stage self-supervised learning, potential domain shift points are detected and left out, the overall accuracy further improves by 1% to 2 %. In general, the task sampler contributes to the improvement of Acc according to the experiment results. It is also evident that BOCPD indeed helps MASSER get more accurate online movement prediction results, which can be attributed to BOCPD's ability to detect potential domain shifts. The complete MASSER framework also gets the best MCC compared with the framework without self-supervised learning, the framework without task sampler, and the framework with only first-stage representation learning.

	ACL18	
Model	Acc	MCC
MASSER-ResNet(w/o encoders)	0.544	0.089
MASSER-GRU(w/o encoders)	0.548	0.094
MASSER-ResNet(w/o 2nd encoder)	0.551	0.088
MASSER-GRU(w/o 2nd encoder)	0.573	0.130
MASSER-ResNet(w/o sampler)	0.550	0.097
MASSER-GRU(w/o sampler)	0.574	0.133
MASSER-ResNet	0.552	0.099
MASSER-GRU	**0.579**	**0.141**
MASSER-ResNet*(w/o 2nd encoder)	0.607	0.210
MASSER-GRU*(w/o 2nd encoder)	0.571	0.142
MASSER-ResNet*(w/o sampler)	0.619	0.234
MASSER-GRU*(w/o sampler)	0.580	0.160
MASSER-ResNet*	**0.624**	**0.244**
MASSER-GRU*	0.581	0.162
MASSER-ResNet(w/o 2nd encoder)	0.610	0.212
MASSER-ResNet	0.612	0.216
MASSER-ResNet-BOCPD(w/o 2nd)	0.605	0.207
MASSER-ResNet-BOCPD	**0.625**	**0.251**

Table 5: Ablation Study Acc and MCC on **ACL18** (* means adaptation)