Multi-period Learning for Financial Time Series Forecasting

Xu Zhang*
School of Computer Science
Fudan University
Shanghai, China
xuzhang22@m.fudan.edu.cn

Zhengang Huang
Ant Group
Shanghai, China
huangzhengang.hzg@antgroup.com

Yunzhi Wu
School of Computer Science
Fudan University
Shanghai, China
yzwu22@m.fudan.edu.cn

Xun Lu
Ant Group
Shanghai, China
hilber.lx@antgroup.com

Erpeng Qi
Ant Group
Shanghai, China
erpeng.qep@antgroup.com

Yunkai Chen
Ant Group
Shanghai, China
chenyunkai.cyk@antgroup.com

Zhongya Xue
Ant Group
Shanghai, China
zhongya.xzy@antgroup.com

Qitong Wang†
Universite Paris Cite
Paris, France
qitong.wang@u-paris.fr

Peng Wang
School of Computer Science
Fudan University
Shanghai, China
pengwang5@fudan.edu.cn

Wei Wang
School of Computer Science
Fudan University
Shanghai, China
weiwang1@fudan.edu.cn

Abstract

Time series forecasting is important in finance domain. Financial time series (TS) patterns are influenced by both short-term public opinions and medium-/long-term policy and market trends. Hence, processing multi-period inputs becomes crucial for accurate financial time series forecasting (TSF). However, current TSF models either use only single-period input, or lack customized designs for addressing multi-period characteristics. In this paper, we propose a Multi-period Learning Framework (MLF) to enhance financial TSF performance. MLF considers both TSF's accuracy and efficiency requirements. Specifically, we design three new modules to better integrate the multi-period inputs for improving accuracy: (i) Inter-period Redundancy Filtering (IRF), that removes the information redundancy between periods for accurate self-attention modeling, (ii) Learnable Weighted-average Integration (LWI), that effectively integrates multi-period forecasts, (iii) Multi-period self-Adaptive Patching (MAP), that mitigates the bias towards certain periods by setting the same number of patches across all periods. Furthermore, we propose a Patch Squeeze module to reduce the number of patches in self-attention modeling for maximized efficiency. MLF incorporates multiple inputs with varying lengths (periods) to achieve better accuracy and reduces the costs of selecting input lengths during training. The codes and datasets are available at https://github.com/Meteor-Stars/MLF.

CCS Concepts

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Artificial intelligence**.

Keywords

time series forecasting; deep learning; financial sales; multi-periods; multi-scales; spatio-temporal data mining

ACM Reference Format:
Xu Zhang, Zhengang Huang, Yunzhi Wu, Xun Lu, Erpeng Qi, Yunkai Chen, Zhongya Xue, Qitong Wang, Peng Wang, and Wei Wang. 2025. Multi-period Learning for Financial Time Series Forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709422

1 Introduction

Time series forecasting (TSF) is a critical task in the finance industry [16, 23, 27, 38]. For example, Alipay is a well-known App that offers convenient digital payment and investment services [3, 35, 39, 40]. Fund sales forecasting is used in financial services of Alipay App to manage the inventory of different fund products. Accurate inventory management is not only crucial for ensuring sufficient availability of various fund products on the Alipay App, but also for

*This work was conducted when Xu Zhang was interning at Ant Group. This work was supported by Ant Group Research Intern Program.
†Corresponding author.

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709422

Table 1: MSE (Mean Squared Error; lower is better) of a single-step TSF under various input lengths n (representing different periods) on the Alipay Fund dataset. $PTST_5$ refers to a PatchTST [20] model trained on inputs of length $n = 5$. $\kappa = mean((X_h - X_f)^2)$ quantifies the consistency between historical windows X^h (30 time steps) and the forecasted value X^f. A higher κ indicates sharper pattern fluctuations.

Data Subset Ratios	MSE			Distribution
	$PTST_5$	$PTST_{10}$	$PTST_{30}$	Consistency κ
33.53%, best by $PTST_5$	**49.28**	54.85	63.31	60.64
27.35%, best by $PTST_{10}$	12.34	**8.92**	13.57	22.46
39.12%, best by $PTST_{30}$	14.56	12.23	**7.34**	14.82

aiding fund institutions in investment preparation and risk management. Moreover, TSF models need to be daily retrained and inferred to accommodate new data for the most accurate fund sales forecasting. Therefore, not only accuracy, but also the efficiency of the TSF model are crucial for real-world financial applications.

One of the key characteristics of financial TS is that their temporal patterns are influenced by different information from multiple periods of historical data (i.e., TS with multiple lengths) [4, 11, 29, 37]. For instance, the sales of fund products are affected not only by short-term public opinions and the company's operational conditions, but also by medium-/long-term policies and market trends. We also found this phenomenon in our experiments. In the scenario of fund sales forecasting, Table 1 indicates that there is no single period input that can provide the best predictions. Specifically, 33.53% samples are best forecasted using the short-period input ($PTST_5$), 27.35% using medium-period input ($PTST_{10}$) and 39.12% using long-period input ($PTST_{30}$), respectively. These uniform ratios suggest that choosing a single fixed period will result in suboptimal forecasting accuracy. For this situation, a simple solution might be to select appropriate input lengths for different forecasting legnths during training, but this leads to high overhead and may not be accurate enough.

Further analyzing the temporal pattern, we found 33.53% samples, best predicted by short-period input, tend to have sharply fluctuating future values. This is evidenced by the highest κ=60.64. In contrast, the remaining 27.35% and 39.12% samples, best predicted by medium- and long-period inputs, have relatively smooth future values as they show lower κ=22.46 and 14.82, respectively. This can be interpreted in the financial domain that short-term hot public topics lead to urgent fund transactions, while medium-/long-term global policies and market trends have a bigger impact on the steady trend of fund sales. Hence, short-period inputs are tightly correlated with future fluctuations, while medium- and long-period inputs are more associated with future steady trends. This explains why different periods work best at predicting different samples in Table 1. In fact, future values can consist of both sharp fluctuations and smooth trends. In this case, even selecting appropriate input lengths in advance for different prediction lengths cannot address the issue. Hence, considering the multi-period inputs simultaneously is crucial for accurate financial TSF.

Recently, several architectures have been proposed for TSF task, including transformer-based Scaleformer [24], PatchTST [20] and

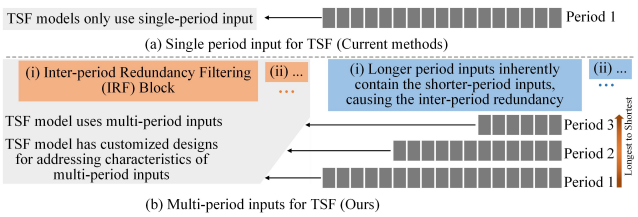


Figure 1: Current TSF scheme (a) and ours (b). The blue rectangle highlights the characteristics of the multi-period inputs.

Pathformer [5], as well as linear model-based NHits [2], TSmixer [6] and TiDE [9]. These models show promising accuracy in the TSF task, but their designs only consider the single-period input (TS with a fixed length), as shown in Figure 1(a). On one hand, this leads to the need to select appropriate input lengths for different prediction lengths, resulting in high training overhead. On the other hand, as analyzed above, future values may contain both sharp fluctuations and stable trends, which requires joint predictions using inputs of different lengths. Although FiLM [41] considers the multi-period inputs, it just linearly integrates multi-period outputs without any specific architecture designs for multi-period characteristics. We discuss one multi-period property in the right of Figure 1(b). As longer-period inputs inherently contain shorter-period inputs, there is information redundancy among different periods. If not well processed, this inter-period redundancy may have adverse effects on model training. For example, it can cause self-attention to overfocus on the information of repetitive parts among different periods, leaving the other parts underutilized. Hence, in our TSF scheme, TSF model not only takes multi-period inputs, but also embrace customized designs to address the multi-period characteristics, as shown in the left of Figure 1(b). To our knowledge, such multi-period based TSF model is rarely explored.

In this paper, we introduce MLF (Multi-period Learning Framework) for accurate financial TSF. MLF is designed to effectively utilize the diverse information across multi-period inputs. However, it is nontrivial to integrate multi-period inputs, which presents challenges in accuracy and efficiency. First, how to address the characteristics of multi-period inputs to effectively integrate multi-period information for forecasting is a key challenge (*integration challenge*). Second, compared to single-period inputs in Figure 1(a), multi-period inputs in Figure 1(b) will significantly increase the computational and memory usage of the model. Therefore, how to enhance the efficiency of multi-period based models while maintaining improved accuracy is another challenge (*efficiency challenge*). To address the *integration challenge*, we introduce three new designs into the transformer architecture: (i) Inter-period Redundancy Filtering (IRF) block to eliminate the inter-period information redundancy, (ii) Learnable Weighted-average Integration (LWI) module to effectively integrate the final multi-period forecasts, (iii) Multi-period self-Adaptive Patching (MAP) module to set the same number of patches for all periods. To tackle the *efficiency challenge*, we propose a Patch-Squeeze module to reduce input length by removing redundancy within each single period (i.e., intra-period redundancy). Specifically, the motivation of integration design (i) is that redundancy naturally exists between different periods. Due to similar segments having higher correlations, this redundancy may

cause the self-attention to excessively focus on repetitive segments and can't effectively utilize information in non-repetitive segments among periods for more accurate multi-period integration. We propose an Inter-period Redundancy Filtering (IRF) block to address this issue. For integration design (ii), before reaching the Learnable Weighted-average Integration (LWI) module, the model's output is still multi-period forecasts, so we need to make the final integration of them. Considering different future positions are best forecasted by different periods, simply averaging may lead to inaccurate integration. Hence, we propose LWI to adaptively assign weights to different period forecasts, giving higher weights to more accurate predictions and vice versa. Accordingly, LWI can improve the integration accuracy of multi-period forecasts. The design intuition for integration design (iii) is that the model may have a bias towards certain periods due to they have varying numbers of patches. This may prevent the model from fully utilizing information of all periods, thereby reducing integration accuracy. Hence, we propose Multi-period self-Adaptive Patching (MAP) to set the same number of patches for all periods to address this issue.

Finally, employing multi-period inputs will significantly increase time and memory costs, posing challenges for scaling to large datasets and deploying in a production system. To address the *efficiency challenge*, we design a Patch-Squeeze module to reduce the number of patches within each period. Patch-Squeeze is based on a key observation, originally from the TS self-supervised learning tasks, that a few numbers of patches are sufficient in reconstructing a much larger set of masked patches [25]. This can be interpreted as there exists intra-period information redundancy. Hence, reducing intra-period redundancy, i.e., reducing the number of patches within each period, can improve model efficiency without compromising accuracy.

In summary, compared to existing methods, MLF processes multiple inputs of different lengths to enhance prediction accuracy, considering both accuracy and efficiency. Our contributions are as follows:

(1) We introduce MLF, a new Multi-period Learning Framework for financial TSF. MLF incorporates multiple inputs with varying lengths (periods) to achieve better accuracy, and one benefit is reducing the costs of selecting input lengths for different prediction lengths during training.

(2) We propose three new architecture designs to better integrate the multi-period inputs, namely, Multi-period self-Adaptive Patching (MAP) modules, Inter-period Redundancy Filtering (IRF) block, and Learnable Weighted-average Integration (LWI).

(3) We further propose a simple but effective Patch-Squeeze module for time dimensionality reduction, which provides high efficiency and maintain expected TSF accuracy while addressing multi-period inputs with varying lengths.

(4) We introduce a new financial dataset of fund sales collected from Ant Fortune and the Alipay App, expanding the public dataset repository and enabling a more comprehensive evaluation of TSF models.

(5) Our experimental results indicate that MLF outperforms advanced TSF baselines in terms of accuracy and efficiency on both collected fund and public datasets. The deployment results on the Alipay App further confirms MLF's effectiveness.

2 Related Work and Preliminary

2.1 Time Series Forecasting (TSF)

2.1.1 Single-period based TSF. Numerous deep learning models have been proposed by using single-period. RNNs [8, 10, 12, 15] use recurrent structures to capture short-term temporal dependency. Meanwhile, temporal convolutional networks (TCN) [18, 22, 33, 34] employ causal and dilated convolutions for parallel computation, effectively capturing temporal dependencies in the TSF task.

The transformer [30] has been extensively adapted for long-term TSF task [32, 42]. Pyraformer [19] employs a multi-pass downsampling operation to capture temporal dependencies at various granularities. Scaleformer [24] further enhances Pyraformer by iteratively refining forecasts at finer scales. Instead of using multi-pass downsampling, Pathformer [5] utilizes different patch sizes to construct multiple time series and designs an adaptive pathway for improving the performance of TSF. However, linear models like DLinear [36], TSMixer [6], N-BEATS [21] and NHits [2] have emerged to challenge the effectiveness of transformer-based methods. PatchTST [20] addresses this by splitting input series into patches and learning self-attention at the patch level, achieving advanced performance in long-term TSF tasks. This illustrates that transformers retain significant potential for TSF when appropriately tailored to the task.

Recently, Large Language Models (LLMs) have shown effectiveness in TSF tasks, particularly in scenarios requiring few training examples (few-shot learning) [14, 43]. Nonetheless, researchers also express worry about using LLMs for TSF [26].

2.1.2 Multi-period based TSF. To our knowledge, few TSF models are proposed to utilize multi-period inputs simultaneously to obtain more accurate TSF. FiLM [41] is a multi-period based method, but it just linearly integrates multi-period outputs for TSF. Although it shows accuracy improvement for using multi-period inputs, it lacks customized designs to effectively handle their characteristics such as redundancy between periods, varying prediction effectiveness, and varying sequence lengths among periods, which suggests an opportunity for further research.

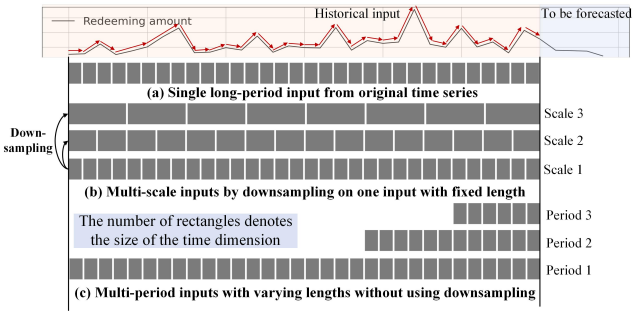
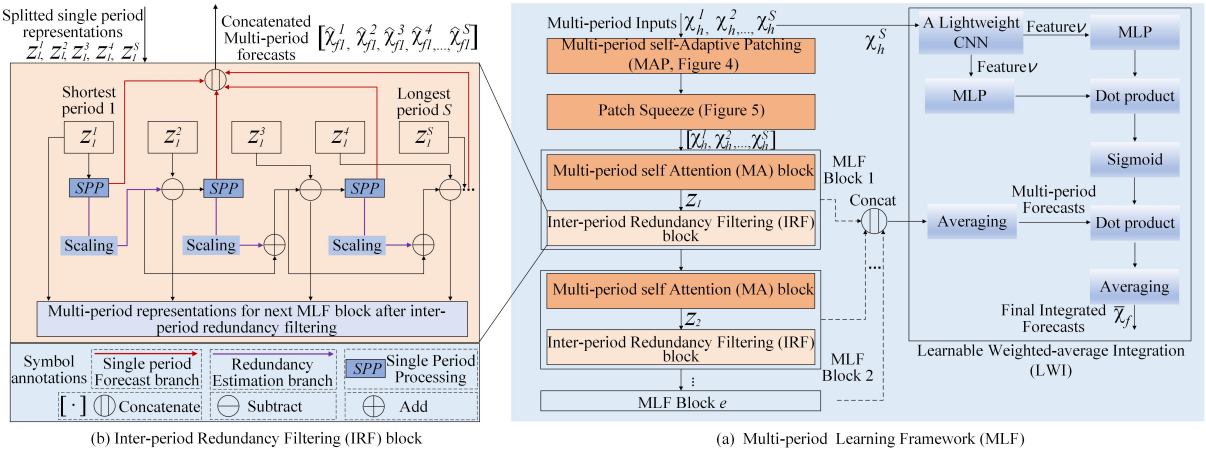


Figure 2: Comparison between multi-period inputs in subfigure (c) and multi-scale inputs in subfigure (b).

2.1.3 Difference between multi-period inputs and multi-scale inputs. In this paper, multi-period inputs refer to multiple original time series windows with varying input lengths, as shown in Figure 2(c). This is different from the multi-scale inputs in Pyraformer [19] and Scaleformer [24], which are obtained by downsampling from the



(b) Inter-period Redundancy Filtering (IRF) block (a) Multi-period Learning Framework (MLF)

Figure 3: MLF and corresponding components.

same fixed input length (Figure 2(b)). In extensive experiments, we observe that different input lengths have a significant impact on prediction accuracy. However, selecting appropriate input lengths is a crucial challenge affecting time series forecasting.

To address this, we propose MLF to extract the semantic information of short-medium-long-term individually using sequences with varying lengths, to avoid models failing to learn the different semantics under only long-term inputs, e.g., the prediction error of Pathformer and Scaleformer using long-term sequence inputs is higher than that of short-term ones. In contrast, by directly inputting multiple windows with varying lengths, MLF can achieve good prediction accuracy, reducing the costs of selecting input windows during training.

2.2 Problem Definition

Given a historical multivariate TS instance $\mathcal{X}_h = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times c}$ with length n, the TSF task aims to predict the future m steps $\mathcal{X}_f = [x_{n+1}, x_{n+2}, ..., x_{n+m}] \in \mathbb{R}^{m \times c}$ for all c variables (for simplicity, we will omit c in the following sections).

In this paper, we focus on designing a Multi-period Learning Framework for TSF. The multi-period inputs $\mathcal{X}_h^*$ are defined as $[\mathcal{X}_h^1, \mathcal{X}_h^2, \ldots, \mathcal{X}_h^S]$ where s $(1 \le s \le S)$ denotes the period index. The multi-period input consists of multiple time series with different lengths n_s. A larger s indicates a longer period, and n_S represents the length of the longest period.

3 Multi-period Learning Framework (MLF)

We illustrate MLF in Figure 3. First, the multi-period input passes through the Multi-period self-Adaptive Patching (MAP) module. The input of each period is converted into the same number of patches and then linearly embedded. Second, each period's patch embeddings are sent to the Patch Squeeze module. It reduces the number of patches by reducing the information redundancy within each period. Third, the squeezed period patch embeddings are fed into the Multi-period self-Attention (MA) and Inter-period Redundancy Filtering (IRF) blocks. IRF removes information redundancy between periods for more accurate MA modeling. MA and IRF blocks form a MLF block. By stacking MLF blocks, MLF progressively removes inter-period redundancy and achieves better modeling of multi-period input. Finally, the Learnable Weighted-average Integration (LWI) module integrates multi-period forecasts from all blocks to generate the final prediction. We will detail each design.

3.1 Multi-period self-Adaptive Patching (MAP)

Following the convention [20], we split the multi-period input $\mathcal{X}_h^*$ into patches instead of individual points as input units.

$$\mathcal{X}_p^s = Patch(\mathcal{X}_h^s, L, K), \tag{1}$$

$$N^s = \lfloor (n^s - L)/K \rfloor + 2, \tag{2}$$

where $Patch(\cdot)$ represents the patching function. L and K denote the patch length and stride of the $Patch(\cdot)$ function, respectively. The parameter K determines the non-overlapping region between consecutive patches. Following [20], we pad K repeated numbers of the last value to the end of the original sequence before patching. In Eq. 2, $\lfloor \cdot \rfloor$ denotes rounding down. $Patch(\cdot)$ transforms a time series into multiple subsequences (patches). $\mathcal{X}_p^s$ denotes the patched period $\mathcal{X}_h^s$. Each $\mathcal{X}_p^s \in \mathbb{R}^{L \times N^s}$, where N^s represents the number of patches in $\mathcal{X}_h^s$.

Eq. 1 to Eq. 2 depict standard multi-period patching, where different periods $\mathcal{X}_h^s$ are patched based on fixed values of L and K. As illustrated in Figure 4(a), this approach results in an increase in the number of patches from short-term to long-term periods. This disparity can lead to unequal treatment of different periods by the model, as shorter-term period inputs may not receive adequate utilization during optimization due to fewer patches.

To address this issue of unequal treatment, we introduce Multi-period self-Adaptive Patching (MAP) module. Illustrated in Figure 4(b), MAP ensures each period $\mathcal{X}_h^s$ contains an equal number of patches by self-adaptively adjusting patch lengths and strides. Specifically, we set $L^s = \alpha \cdot K^s$ (where α is fixed at 2 in this paper) and maintain a fixed number of patches $\widetilde{N}$ across all periods. Substituting this into Eq. 2 and expanding, we obtain:

$$\widetilde{N} = (n^s - L)/K + 2 \tag{3}$$

$$= (n^s - \alpha \cdot K)/K + 2 \tag{4}$$

After further expanding the formula, we can derive the self-adaptive L^s and K^s for different periods:

$$\widetilde{N} \cdot K - \alpha \cdot K = n^s - \alpha \cdot K \tag{5}$$

$$K^s = \lfloor n^s / \widetilde{N} \rfloor, \quad L^s = \alpha \cdot K^s \tag{6}$$

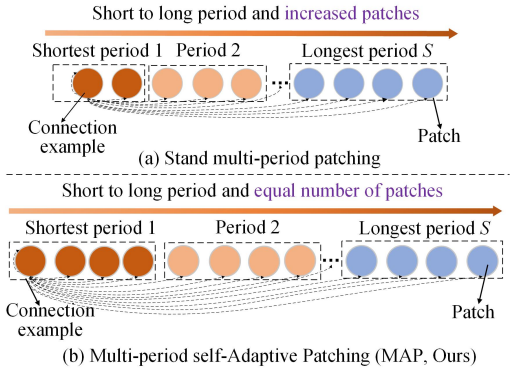


Figure 4: Illustration of the standard multi-period patching (using fixed patch lengths) and self-adaptive multi-period patching (using self-adaptively varied patch lengths).

Self-adaptive L^s and K^s ensure each period $\mathcal{X}_h^s$ contains an equal number of patches, allowing the model to evenly consider the temporal information across different periods. Substituting L^s and K^s into Eq. 1 yields the self-adaptive patched $\mathcal{X}_p^s$.

Before feeding the patches of each period into the transformer encoder, they are projected into a D-dimensional embedding space with a learnable additive position encoding [20, 30]. The embedded representation of $\mathcal{X}_p^s$ is denoted as $\mathcal{X}_d^s$ and can be formulated as:

$$\mathcal{X}_d^s = W_p^s \mathcal{X}_p^s + W_{pos}^s \tag{7}$$

where each $W_p^s \in \mathbb{R}^{D \times L^s}$ and $W_{pos}^s \in \mathbb{R}^{D \times N^s}$.

3.2 Patch Squeeze

Studies of self-supervised learning for time series [25] suggest that there is information redundancy within periods. That is, a small subset of patches can effectively capture the global temporal patterns of the entire period. Motivated by this insight, we propose a patch squeeze module to distill essential information from the original time series data into a reduced number of patches. This module not only substantially reduces both time complexity and memory usage, but also shows an insignificant impact on accuracy (empirically verified in Section 4.3 and Section 4.5.3).

The patch squeeze module, illustrated in Figure 5, consists of a lightweight encoder ($PatchEnc$, implemented with one linear layer) and decoder (composed of MLP_L^s and MLP_N^s). The outputs of the patch squeeze module, denoted as $\hat{\mathcal{X}}_p^s$ (used for reconstruction loss) and $\hat{\mathcal{X}}_d$ (sent to the transformer encoder), are formulated as follows:

$$\hat{\mathcal{X}}_d^s = PatchEnc(\mathcal{X}_d^s, r) \tag{8}$$

$$\hat{\mathcal{X}}_d = [\hat{\mathcal{X}}_d^0, \cdots, \hat{\mathcal{X}}_d^s] \tag{9}$$

$$\hat{\mathcal{X}}_p^s = MLP_L^s(MLP_N^s(\hat{\mathcal{X}}_d^s)) \tag{10}$$

where r represents the squeeze factor, which reduces the original patch number N^s to $\frac{N^s}{r}$. MLP_N^s and MLP_L^s are used to align the dimensions of patch number and patch length accordingly. $\hat{\mathcal{X}}_p^s$ represents the reconstructed original time series. This reconstruction is used to calculate the reconstruction loss, helping the reduced number of patches effectively capture the essential information from the original patches. $\hat{\mathcal{X}}_d$ denotes the concatenation of the squeezed multi-period patch embeddings, which are fed into the transformer encoder.

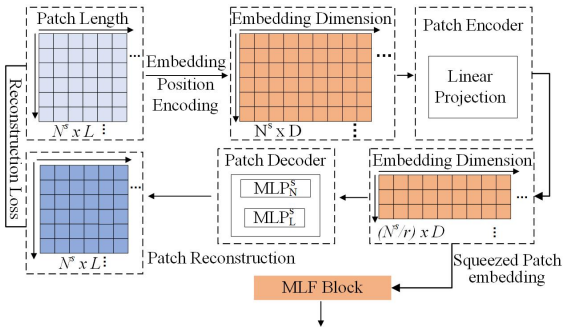


Figure 5: Illustration of patch squeeze module.

3.3 Vanilla Multi-period Attention

By directly feeding $\hat{\mathcal{X}}_d$ into the transformer encoder, we implement the vanilla multi-period self-attention. Specifically, each head $h = 1, 2, ..., H$ in the multi-head attention processes $\hat{\mathcal{X}}_d$ to generate H distinct query matrices $Q_h = (\hat{\mathcal{X}}_d)^T W_h^Q$, key matrices $K_h = (\hat{\mathcal{X}}_d)^T W_h^K$, and value matrices $V_h = (\hat{\mathcal{X}}_d)^T W_h^V$. T denotes matrix transposition. A scaled dot-product operation is then applied to compute the attention output $O_h \in \mathbb{R}^{D \times N}$:

$$O_h = Attention(Q_h, K_h, V_h) = \sigma(\frac{Q_h (K_h)^T}{\sqrt{d_k}}) V_h \tag{11}$$

here, σ denotes the Softmax function. The output $z_e \in \mathbb{R}^{D \times N}$ at the e-th encoder layer is computed using BatchNorm and a feed-forward network with residual connections [20, 30].

3.4 Inter-period Redundancy Filtering-based Multi-period Self-attention

The information redundancy between different periods can cause self-attention to excessively focus on repetitive segments due to higher correlations between similar parts. This can prevent self-attention from capturing all patterns distributed across short and medium-/long-term periods, thereby impairing TSF performance.

To mitigate this issue, we propose an Inter-period Redundancy Filtering (IRF) block, illustrated in Figure 3(b). IRF identifies redundant information in longer periods based on shorter ones, and then removes it in the embedding space. Consequently, IRF helps better self-attention modeling, allowing important information from all periods to be effectively utilized, thereby improving prediction accuracy. For implementation, IRF starts by splitting the concatenated period representations from z_e:

$$z_e^1, z_e^2, \cdots, z_e^s, \cdots, z_e^S = split(z_e) \tag{12}$$

Each $z_e^s \in \mathbb{R}^{D \times N^s}$ denotes the representation of the s-th period. Then, we devise a Single Period Processing (SPP, as shown in Figure 3(b)) module for each period representation z_e^s. SPP has two branches of linear layers. The forecast branch generates prediction of the current period, while the redundancy estimation branch estimates the redundancy to be removed from the longer period representation $z_e^{s'}$ $(s < s' \leq S)$. SPP outputs the forecasts $\hat{\mathcal{X}}_{fe}^s$ and the estimated redundancy ϵ_e^s for the s-th period at e-th block.

$$\hat{\mathcal{X}}_{fe}^s, \epsilon_e^s = SPP(z_e^s) \tag{13}$$

To reduce redundancy, we subtract the estimated redundancy of all shorter periods from the current period embedding z_e^s. The period representation after redundancy filtering is denoted as $\hat{z}_e^s$:

$$\hat{z}_e^s = z_e^s - \sum_{j=0}^{s-1} \frac{\epsilon_e^j}{\sqrt{d_k}} \tag{14}$$

$\sqrt{d_k}$ serves as a scaling factor designed to prevent numerical overflow and stabilize gradients. $\hat{z}_e^s$ are then output to the subsequent encoder layer.

Lastly, as illustrated in Figure 3(a), stacking the MLF block (one transformer block and one IRF block) enables progressive redundancy filtering, thereby improving the accuracy of redundancy estimates and self-attention modeling. Each MLF block has a forecast head [13, 17]. We aggregate the forecasts from all blocks by:

$$\overline{\mathcal{X}}_f^s = \frac{1}{E} \sum_{e=1}^{E} \mathcal{X}_{fe}^s \tag{15}$$

where $\overline{\mathcal{X}}_f^s$ is the average forecast at the block level for period s.

3.5 Learnable Weighted-average Integration (LWI)

After obtaining forecasts $\overline{\mathcal{X}}_f^s$ for each period, we now integrate multiple period forecasts to obtain the final prediction by learned weighted averaging. As different future positions are best forecasted by different periods, simply adding or averaging these predictions may lead to inaccurate forecasting. Hence, we propose a Learnable Weighted-average Integration (LWI) module to effectively integrate multi-period forecasts. LWI adaptively assigns the weights for different period forecasts, which increases the contribution of accurate predictions while reducing the impact of inaccurate ones.

To implement LWI, we start by extracting temporal features v from the longest-period TS $\mathcal{X}_h^S$ using a lightweight Convolutional Neural Network (CNN), which consists of a convolutional layer ($Conv(\cdot)$), padding function ($Pad(\cdot)$), BatchNorm ($BN(\cdot)$) and Max-pooling ($MaxPool(\cdot)$) layer. Then we employ two MLPs to derive query and key values, enabling a dot product operation to compute attention scores that indicate the model's effective focus on different periods. These attention scores are then passed through a sigmoid activation function $\varsigma(\cdot)$ to generate weights Att for each period. Finally, we average the multi-period forecasts based on the learned weights as the final prediction (Eq. 18). The entire process that depicted in right of Figure 3(a) can be formulated as follows:

$$v = MaxPool(BN(Conv(Pad(\mathcal{X}_h)))) \tag{16}$$

$$Att = \varsigma\left(\tanh\left(\Theta_1 v + \mathbf{b}_1\right) \odot \tanh\left(\Theta_2 v + \mathbf{b}_2\right)\right) \tag{17}$$

$$\overline{\mathcal{X}}_f = \frac{1}{S} \sum_{e=1}^{S} \mathcal{X}_f^s \odot Att^s \tag{18}$$

where Θ and $\mathbf{b}$ are learnable parameters.

3.6 System Deployment

We illustrate the online system architecture in Figure 6. It consists of three sub-systems: the Fund Inventory Management System (FIMS), Alipay APP Market Shelf, and the MaxCompute database. FIMS

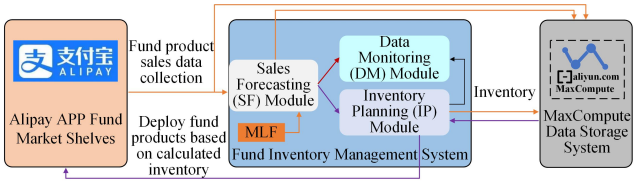


Figure 6: Alipay's fund management system architecture. MLF has been deployed in this production environment.

Table 2: Statistics of used popular public datasets.

Datasets	ETTh1/h2	ETTm1/m2	Weather	Exchange	Illness	Electricity
Features	7	7	21	8	7	321
Timesteps	17420	69680	52696	7588	966	26304

incorporates modules for Sales Forecasting (SF, with MLF in it), Inventory Planning (IP), and Data Monitoring (DM). SF predicts fund sales for the upcoming 5 days based on historical sales data. IP uses an inventory planning algorithm to determine product inventory according to the forecasted sales. DM monitors the reasonableness of forecasted sales and inventory plans.

The system operates as follows. The shelf subsystem continuously sends trading data to the FIMS subsystem, which updates the sales records of fund products in the MaxCompute database. At the end of each day, FIMS performs three sequential daily batch tasks: 1) retraining the MLF model; 2) using the updated MLF to forecast fund sales for the next 5 days; and 3) planning product inventory with the IP module. Subsequently, the inventory of all fund products on the shelf is adjusted accordingly. Additionally, both forecasted fund sales and planned inventory are persistently stored in the MaxCompute database for retrospective analysis.

Training. The final loss function for MLF is:

$$\mathcal{L}_{\text{MLF}} = \mathcal{L}_{\text{MSE}}(\overline{\mathcal{X}}_f, \mathcal{X}_f) + \frac{1}{S} \sum_{j=1}^{S} \mathcal{L}_{\text{MSE}}(\hat{\mathcal{X}}_p^j, \mathcal{X}_p^j) \tag{19}$$

here, the first term refers to the forecasting loss, while the second term represents the reconstruction loss.

4 Experiments and Results

4.1 Experimental Settings

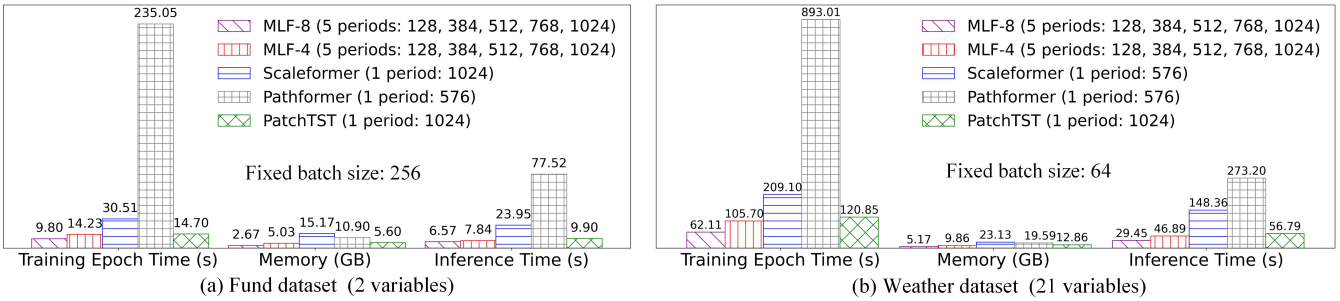
4.1.1 Datasets. **Fund sales dataset.** We collected sales data for different fund products from Ant Fortune, an online wealth management platform on the Alipay APP, consisting of daily user transactions for applying and redeeming funds. The dataset spans from January 2015 to January 2023 and is split into training, validation, and testing sets in a 7:1:2 ratio. For model training and evaluation, we merged the training, validation, and test sets across all fund datasets.

Public datasets. These datasets, as shown in Table 2, cover a range of time steps and variables and have been widely employed in the literature for multivariate forecasting tasks [20, 32, 42].

4.1.2 Evaluation Metrics. We evaluate multivariate TSF tasks using three metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Weighted Mean Absolute Percentage Error (WMAPE or WMA. for short). WMAPE is particularly relevant for the fund dataset on the Alipay APP, as it assesses the accuracy of predictions

Table 3: Comparing MLF with popular benchmarks in short-term multivariate forecasting. "-" denotes that the prediction length is too short to use Scaleformer. The best results are in bold and the second best are underlined.

		MLF		PatchTST		Patch-Ensemble		Patch-Concat		NHits		FiLM		Scaleformer		Pathformer	
		MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.
Fund	1	**33.85**	**75.84**	36.68	81.05	40.30	86.68	39.68	85.87	36.53	80.29	46.12	96.10	-	-	35.54	80.75
		±0.033	**±0.093**	±0.068	±0.288	±0.235	±0.431	±0.328	±0.266	±0.057	±0.108	±0.009	±0.009	-	-	±0.08	±0.06
	5	**38.28**	**80.37**	40.28	83.09	39.95	83.24	41.60	87.14	40.91	83.93	42.86	85.35	40.43	83.74	39.90	82.58
		±0.029	**±0.071**	±0.055	±0.143	±0.068	±0.087	±0.357	±0.643	±0.024	±0.090	±0.009	±0.001	±0.088	±2.47	±0.02	±0.06
	8	**41.94**	**86.06**	44.71	88.81	44.06	88.77	43.49	88.08	44.81	89.23	44.57	89.12	43.97	87.62	44.17	88.67
		±0.031	**±0.123**	±0.085	±0.164	±0.052	±0.094	±0.289	±0.552	±0.020	±0.062	±0.009	±0.001	±0.062	±2.22	±0.04	±0.11
	10	**44.42**	**88.66**	46.18	90.63	46.09	91.36	45.59	90.46	46.73	91.12	46.62	92.67	45.75	89.63	45.56	89.82
		±0.057	**±0.041**	±0.063	±0.160	±0.049	±0.070	±0.054	±0.101	±0.087	±0.097	±0.004	±0.009	±0.088	±2.23	±0.06	±0.13
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	1	**0.0472**	**0.1340**	0.0510	0.1410	0.053	0.144	0.0492	0.1382	0.0517	0.1430	0.0729	0.1750	-	-	0.0613	0.1590
ETTh1	1	**0.0873**	**0.1899**	0.0979	0.2033	0.115	0.2202	0.0958	0.1983	0.113	0.2268	0.1327	0.2395	-	-	0.1067	0.213
ETTm1	1	**0.0412**	**0.1245**	0.0461	0.13	0.0524	0.1457	0.0450	0.1298	0.0475	0.139	0.0998	0.1918	-	-	0.0470	0.1346
Illness	1	**0.1493**	**0.2188**	0.1782	0.2340	0.2918	0.2934	0.2476	0.2409	0.2515	0.2833	0.3104	0.3306	-	-	0.269	0.279
Exchange	1	**0.0029**	**0.0267**	0.0042	0.0363	0.0037	0.030	0.0047	0.0325	0.008	0.0559	0.0055	0.0445	-	-	0.0035	0.0293



(a) Fund dataset (2 variables) (b) Weather dataset (21 variables)

Figure 7: Efficiency analysis for best-performing TSF models was conducted on Ettm1, Weather, and Fund datasets.

on larger values. This metric aligns with the business goal of accurately forecasting larger transactions in fund products. We present the sum of WMAPE for the two variables in the fund dataset.

4.1.3 Baselines. We compare our Multi-period Learning Framework (MLF) against three types of baselines. **(i) single-period based models:** This category includes PtachTST [20], NHits [2], Scaleformer [24] and PathFormer [5]. Since Scaleformer serves as a general architecture, we combine it with Autoformer [32], NHits [2], and PatchTST [20] to create strong baseline.

(ii) Multi-period based models: This category includes FiLM [41] and two multi-period variants based on PatchTST, called Patch-Concat and Patch-Ensemble. Specifically, for Patch-Concat, we concatenate multi-period data for training PatchTST. For Patch-Ensemble, we train multiple instances of PatchTST using multiple periods with different TS lengths for ensemble learning. We compare these three baselines to demonstrate the necessity of the structure designed for multi-period forecasting in MLF.

4.1.4 Implementation Details. For short-term time series forecasting (TSF) tasks, the short-term to long-term multi-period lengths for MLF are 5, 10, 30, 60, 120, and 150 time steps. Prediction lengths vary across $m \in 1, 5, 8, 10$. For long-term TSF tasks, the used short-term to long-term multi-period lengths are 128,256,512,768, 1024 and 2048 time steps. Prediction lengths vary across $m \in 96, 192, 336, 720$. Each period within MLF employs a fixed number of 64 patches, with a squeeze factor $r \in 2,4,8$ to squeeze the long-sequence input. Both Patch-Concat and Patch-Ensemble models utilize the same multi-period configuration as MLF. The learning rate and batch size are set to 0.0001 and 128 respectively. All methods follow the same data loading parameters (e.g., train/val/test split ratio) as in [20]. Each method is trained for 30 epochs. All experiments in this paper are conducted using PyTorch on an NVIDIA GeForce RTX 3090 GPU.

4.2 Time Series Forecasting Accuracy

4.2.1 Comparison with single-period baselines. Extensive results show that MLF outperforms advanced single-period baselines, including PatchTST, NHits, Scaleformer, and PathFormer on both short-term (Table 3) and long-term tasks (Table 4). These baselines are well-designed for single-period time series and doesn't utilize multi-period information. However, we have observed that inputs of different lengths have a significant impact on prediction accuracy. However, these single-period-based methods can only process one input at a time, which prevents them from achieving better prediction accuracy. In contrast, MLF not only utilizes multi-period inputs, but also incorporates new designs to maximize its TSF potential, achieving better accuracy.

Table 4: Comparing MLF with popular benchmarks in long-term multivariate forecasting. The best results are indicated in bold, and the second-best results are underlined.

		MLF		PatchTST		Patch-Ensemble		Patch-Concat		NHits		FiLM		Scaleformer		Pathformer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.363**	**0.392**	0.370	0.400	0.370	0.401	0.369	0.400	0.427	0.439	0.371	0.394	0.379	0.409	0.393	0.406
	192	**0.399**	**0.416**	0.413	0.429	0.402	0.422	0.406	0.427	0.472	0.473	0.414	0.423	0.411	0.43	0.421	0.436
	336	**0.416**	**0.423**	0.422	0.440	0.424	0.432	0.427	0.433	0.525	0.498	0.442	0.445	0.43	0.443	0.451	0.451
	720	**0.438**	**0.454**	0.447	0.468	0.445	0.463	0.445	0.463	0.608	0.565	0.465	0.472	0.446	0.465	0.483	0.470
ETTh2	96	**0.267**	**0.334**	0.274	0.337	0.276	0.341	0.277	0.341	0.314	0.379	0.284	0.348	0.275	0.343	0.285	0.349
	192	**0.327**	**0.374**	0.341	0.382	0.337	0.379	0.336	0.379	0.401	0.434	0.357	0.4	0.337	0.384	0.331	0.385
	336	**0.350**	**0.396**	0.359	0.405	0.357	0.403	0.358	0.403	0.452	0.469	0.377	0.417	0.364	0.414	0.368	0.409
	720	**0.383**	**0.423**	0.388	0.427	0.387	0.429	0.388	0.430	0.545	0.527	0.439	0.456	0.397	0.438	0.389	0.427
ETTm1	96	**0.285**	**0.344**	0.293	0.346	0.296	0.347	0.290	0.345	0.32	0.367	0.302	0.345	0.293	0.347	0.301	0.352
	192	**0.324**	**0.366**	0.333	0.370	0.332	0.373	0.331	0.372	0.357	0.392	0.338	0.368	0.333	0.371	0.356	0.383
	336	**0.355**	**0.384**	0.360	0.392	0.366	0.395	0.365	0.396	0.392	0.417	0.365	0.385	0.364	0.391	0.387	0.405
	720	**0.401**	**0.410**	0.404	0.417	0.415	0.427	0.417	0.427	0.442	0.441	0.42	0.42	0.42	0.425	0.416	0.420
ETTm2	96	**0.164**	**0.253**	0.166	0.256	0.163	0.257	0.169	0.263	0.176	0.255	0.165	0.256	0.172	0.255	0.168	0.258
	192	**0.218**	**0.295**	0.223	0.296	0.226	0.304	0.227	0.306	0.245	0.305	0.222	0.296	0.231	0.298	0.227	0.296
	336	**0.270**	**0.330**	0.277	0.336	0.278	0.336	0.285	0.340	0.295	0.346	0.277	0.333	0.278	0.328	0.273	0.331
	720	**0.344**	**0.379**	0.357	0.381	0.360	0.393	0.360	0.393	0.401	0.413	0.371	0.389	0.361	0.383	0.366	0.392
Weather	96	**0.145**	**0.195**	0.148	0.200	0.149	0.201	0.147	0.199	0.158	0.212	0.199	0.262	0.152	0.208	0.155	0.208
	192	**0.191**	**0.238**	0.194	0.241	0.196	0.244	0.194	0.240	0.202	0.247	0.228	0.288	0.197	0.251	0.196	0.246
	336	**0.241**	**0.280**	0.245	0.282	0.247	0.284	0.248	0.284	0.257	0.3	0.267	0.323	0.253	0.296	0.25	0.286
	720	**0.296**	**0.328**	0.309	0.330	0.318	0.337	0.316	0.335	0.327	0.356	0.319	0.361	0.311	0.343	0.324	0.337

4.2.2 Comparison with multi-period methods. Extensive results also show that MLF outperforms advanced multi-period methods FiLM, Patch-Concat and Patch-Ensemble, as shown Table 3 and Table 4. This highlights that merely linearly integrating multi-period outputs (FiLM), concatenating multi-period inputs for training PatchTST (Patch-Concat) and using multiple instances of PatchTST for ensemble learning (Patch-Ensemble) fails to fulfill the forecasting potential of multi-period data. In contrast, MLF is customized to address the characteristics of multi-period data, such as inter-period redundancy and varying sequence lengths. New designs in MLF enable it to effectively utilize information across all periods, thereby achieving superior accuracy in TSF tasks.

4.3 Time Series Forecasting Efficiency

In MLF, we use squeeze factors of 8 (MLF-8) and 4 (MLF-4). For a fair comparison, we maintained consistency in shared hyper-parameters (such as hidden size and attention heads in the transformer) and settings (like batch size).

We select the best-performing models for an efficiency comparison. Results are depicted in Figure 7. Due to GPU memory constraints (24GB), baselines like Pathformer can handle a maximum sequence length of 576 across all datasets, while Scaleformer manages 576 for Weather datasets and 1024 for ETTm1 or Fund datasets. Despite MLF's significantly longer actual input sequence length (2816 compared to 576 or 1024 for PathFormer and Scaleformer), it achieves better efficiency. For instance, on the Fund dataset (Figure 7(a)), MLF is 3.6 times faster than Scaleformer and 11.8 times faster than PathFormer in terms of inference time. MLF's efficiency advantage is even bigger with larger datasets, as shown in the results of the Weather dataset (Figure 7(b)), while maintaining satisfactory accuracy (Table 7).

Table 5: Deployment results over consecutive five weeks.

Year (2024)	Week 1	Week 2	Week 3	Week 4	Week 5	Mean
IR_{WMAPE} (%)	6.84	4.25	5.21	4.92	4.62	5.16
IR_{GMV} (%)	1.14	0. 58	0.72	0.85	0.95	0.84

4.4 Deployment Results

MLF has been deployed in the FIMS of Alipay since late February 2024. We compared the WMAPE and Gross Merchandise Volume (GMV) of all fund products between MLF and the previously deployed model PatchTST. Table 5 shows the improvement ratio of WMAPE ($IR_{WMAPE} = \frac{PatchTST_{WMAPE} - MLF_{WMAPE}}{PatchTST_{WMAPE}}$) and the improvement ratio of GMV ($IR_{GMV} = \frac{MLF_{GMV} - PatchTST_{GMV}}{PatchTST_{GMV}}$) over five consecutive weeks. The improvement ratios of WMAPE and GMV over advanced PatchTST demonstrate that multi-period based MLF is effective in the fund sales forecasting scenario.

4.5 Ablation Study

4.5.1 Ablation of each component (Table 6). **(1) Effectiveness of Inter-period Redundancy Filtering (IRF).** "w/o IRF" denotes we remove redundancy subtract operation (without using Eq. 14) in IRF block. When removing IRF, we observe raised MSE. Moreover, compared to "w/o MA", "w/o (MA+IRF)" also shows further raised MSE. The raised MSE demonstrates that filtering redundancy between periods allows self-attention to avoid overly focusing on repetitive parts. This enables the utilization of information from all periods, resulting in better TSF accuracy.

Figures 8(a)-(d) further illustrate this. Specifically, we obtained self-attention heatmaps by averaging the attention score matrices of all test samples on the Fund dataset. Each index on the horizontal

Table 6: Ablation study on various datasets.

Methods/ Datasets	Illness MSE	Electricity MSE	ETTh1 MSE	Exchange MSE	Fund WMA.
MLF	**0.149**	**0.0472**	**0.087**	**0.0030**	**75.84**
w/o IRF	0.163	0.0500	0.091	0.0033	78.56
w/o (LWI)	0.155	0.0491	0.088	0.0032	77.32
w/o MA	0.236	0.0539	0.114	0.004	78.23
w/o (MA+IRF)	0.261	0.0559	0.130	0.004	79.85

axis represents a patch, with every consecutive three patches representing a period. The heatmap without the IRF operation shows an overemphasis on the overlapping parts among the multi-period inputs (vertical highlight areas in Figures 8(b) and (d)), while the heatmap with the IRF operation demonstrates that most regions (non-repetitive parts among periods) receive effective attention.

(2) Effectiveness of MA and LWI. The raised MSE observed in the "w/o LWI" and "w/o MA" highlights the importance of the Learnable Weighted-average Integration (LWI) and Multi-period self-Attention (MA) modules. Based on the learned adaptive period weights, LWI increases the contribution of accurate predictions while reducing the impact of inaccurate ones, thereby improving overall forecasting accuracy. MA effectively captures multi-period dependencies. Together, these modules are crucial for realizing the full potential of multi-period based time series forecasting.

4.5.2 Effectiveness of Multi-period self-Adaptive Patching (MAP). In the standard multi-period patching, the shorter periods have fewer patches while the longer one is the opposite. The results in Figure 9 show the raised MSE when MAP is removed (w/o MAP), indicating that shorter periods may be unfairly treated in standard multi-period patching due to the insufficient number of patches. MAP ensures each period X_h^s has an equal number of patches, allowing the model to equally pay attention to the time semantics of all periods and improving TSF performance.

4.5.3 Effectiveness of patch squeeze module. As shown in Table 7, MLF achieves satisfactory performance with squeeze factors of 2, 4, and 8. This demonstrates that the patch squeeze module in MLF can significantly reduce time complexity while maintaining good accuracy. Additionally, the reconstruction loss in the patch squeeze module is effective, as Figure 8(d) shows raised MSE when this term is removed ("w/o reconstruction loss").

Table 7: Effectiveness of the patch squeeze module: MLF's accuracy with squeeze factors 2 (MLF-2), 4 (MLF-4), and 8 (MLF-8). '-' indicates out of memory.

		PatchTST		MLF-2		MLF-4		MLF-8	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.293	0.346	**0.283**	**0.342**	0.285	**0.344**	0.287	0.345
	192	0.333	0.370	**0.322**	**0.366**	0.324	**0.366**	0.326	**0.366**
	336	0.360	0.392	0.357	0.387	0.356	0.384	**0.355**	**0.384**
	720	0.404	0.417	0.406	0.413	**0.400**	0.410	0.401	**0.410**
Weather	96	0.148	0.200	-	-	0.146	0.196	**0.145**	**0.195**
	192	0.194	0.241	-	-	**0.191**	**0.238**	0.191	0.239
	336	0.245	0.282	-	-	0.244	0.283	**0.241**	**0.280**
	720	0.309	0.330	-	-	0.304	0.333	**0.296**	**0.328**

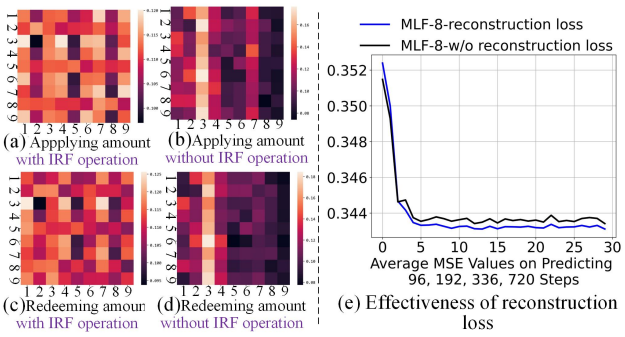


Figure 8: Extra ablation study: (a)-(d) Self-attention visualization with and without IRF. (e) Ablation of the reconstruction loss in the patch squeeze module on the ETTm1 dataset.

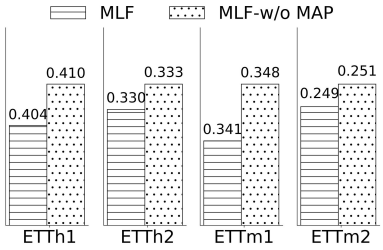


Figure 9: Ablation study of Multi-period self-Adaptive Patching (MAP) module. We report the average MSE values for predicting future 96, 192, 336, and 720 time steps.

5 Conclusion

This paper introduces Multi-period Learning Framework MLF, which incorporates multiple inputs with varying lengths to achieve better accuracy and reduces the costs of selecting input windows during training. MLF also demonstrates that using a simple encoder to squeeze the input time series (dimensionality reduction) can significantly improve model efficiency, reduce memory overhead, and achieve comparable or even better forecasting accuracy. MLF's other designs, including MAP, IRF, and LWI also help better address multi-period characteristics and enhance TSF performance. Nevertheless, there is still considerable room for exploration in model architectures that use historical inputs of different lengths simultaneously for prediction. Future research could explore novel designs aimed at better addressing the challenges caused by multi-period characteristics for multi-period forecasting, thereby further releasing the predictive potential of multi-period inputs. To facilitate research, we have open-sourced the code.

6 Acknowledgements

This work is supported by the Ministry of Science and Technology of China, National Key Research and Development Program (No. 2021YFB3300503).

References

[1] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[2] C Challu, KG Olivares, BN Oreshkin, F Garza, M Mergenthaler, and A Dubrawski. 2022. N-hits: Neural hierarchical interpolation for time series forecasting. arXiv. *arXiv preprint arXiv:2201.12886* (2022).

[3] Cen Chen, Xiaolu Zhang, Sheng Ju, Chilin Fu, Caizhi Tang, Jun Zhou, and Xiaolong Li. 2019. AntProphet: an Intention Mining System behind Alipay's Intelligent Customer Service Bot.. In *IJCAI*, Vol. 8. 6497–6499.

[4] Jou-Fan Chen, Wei-Lun Chen, Chun-Ping Huang, Szu-Hao Huang, and An-Pin Chen. 2016. Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks. In *7th International Conference on Cloud Computing and Big Data, CCBD 2016, Macau, China, November 16-18, 2016*. IEEE Computer Society, 87–92. https://doi.org/10.1109/CCBD.2016.027

[5] Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. 2024. Pathformer: Multi-scale transformers with Adaptive Pathways for Time Series Forecasting. *arXiv preprint arXiv:2402.05956* (2024).

[6] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. 2023. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053* (2023).

[7] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *Transactions on Machine Learning Research* (2023).

[10] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[11] Zheng Gu and Yuhua Xu. 2021. Chaotic Dynamics Analysis Based on Financial Time Series. *Complex.* 2021 (2021), 2373423:1–2373423:6. https://doi.org/10.1155/2021/2373423

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

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[14] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2023. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[15] Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. In *Advances in psychology*. Vol. 121. Elsevier, 471–495.

[16] Bjoern Krollner, Bruce J. Vanstone, and Gavin R. Finnie. 2010. Financial time series forecasting with machine learning techniques: a survey. In *18th European Symposium on Artificial Neural Networks, ESANN 2010, Bruges, Belgium, April 28-30, 2010, Proceedings*. https://www.esann.org/sites/default/files/proceedings/legacy/es2010-50.pdf

[17] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-Supervised Nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015 (JMLR Workshop and Conference Proceedings, Vol. 38)*, Guy Lebanon and S. V. N. Vishwanathan (Eds.). JMLR.org. http://proceedings.mlr.press/v38/lee15a.html

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJiHXGWAZ

[19] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[20] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[21] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

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

[23] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing* 90 (2020), 106181.

[24] Mohammad Amin Shabani, Amir H Abdi, Lili Meng, and Tristan Sylvain. 2022. Scaleformer: Iterative Multi-scale Refining Transformers for Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*.

[25] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1567–1577.

[26] Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. 2024. Are Language Models Actually Useful for Time Series Forecasting? *arXiv preprint arXiv:2406.16964* (2024).

[27] Yajiao Tang, Zhenyu Song, Yulin Zhu, Huaiyu Yuan, Maozhang Hou, Junkai Ji, Cheng Tang, and Jianqiang Li. 2022. A survey on machine learning models for financial time series forecasting. *Neurocomputing* 512 (2022), 363–380. https://doi.org/10.1016/J.NEUCOM.2022.09.003

[28] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[29] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. 2019. Temporal Attention-Augmented Bilinear Network for Financial Time-Series Data Analysis. *IEEE Trans. Neural Networks Learn. Syst.* 30, 5 (2019), 1407–1418. https://doi.org/10.1109/TNNLS.2018.2869225

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

[31] Xiaozhe Wang, Kate Smith, and Rob Hyndman. 2006. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery* 13 (2006), 335–364.

[32] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[33] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 753–763. https://doi.org/10.1145/3394486.3403118

[34] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 1907–1913. https://doi.org/10.24963/IJCAI.2019/264

[35] Xiaoling Zang, Binbin Hu, Jun Chu, Zhiqiang Zhang, Guannan Zhang, Jun Zhou, and Wenliang Zhong. 2023. Commonsense Knowledge Graph towards Super APP and Its Applications in Alipay. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5509–5519.

[36] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[37] Zhanggui Zeng. 2008. *Financial Time Series Analysis using Pattern Recognition Methods*. Ph. D. Dissertation. University of Sydney, Australia. https://hdl.handle.net/2123/3558

[38] Cheng Zhang, Nilam Nur Amir Sjarif, and Roslina Binti Ibrahim. 2024. Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020-2022. *WIREs Data. Mining. Knowl. Discov.* 14, 1 (2024). https://doi.org/10.1002/WIDM.1519

[39] Xu Zhang, Zhengang Huang, Yunzhi Wu, Xun Lu, Erpeng Qi, Yunkai Chen, Zhongya Xue, Peng Wang, and Wei Wang. 2024. Self-Adaptive Scale Handling for Forecasting Time Series with Scale Heterogeneity. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7485–7489.

[40] Zhiqiang Zhang, Chaochao Chen, Jun Zhou, and Xiaolong Li. 2018. An Industrial-Scale System for Heterogeneous Information Card Ranking in Alipay. In *Database Systems for Advanced Applications - 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 10828)*, Jian Pei, Yannis Manolopoulos, Shazia W. Sadiq, and Jianxin Li (Eds.). Springer, 713–724. https://doi.org/10.1007/978-3-319-91458-9_44

[41] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. 2022. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems* 35 (2022), 12677–12690.

[42] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. PMLR, 27268–27286.

[43] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems* 36 (2023), 43322–43355.

Table 8: Average trends, periodicity, and kurtosis comparison.

Datasets	Fund	Electricity	ETTh1	ETTm1	Exchange	Illness
Trend	0.74	0.84	0.86	0.86	1.0	0.71
Periodicity	0.62	0.92	0.72	0.74	0.31	0.81
Kurtosis	29.21	0.39	1.37	1.37	-0.53	5.56

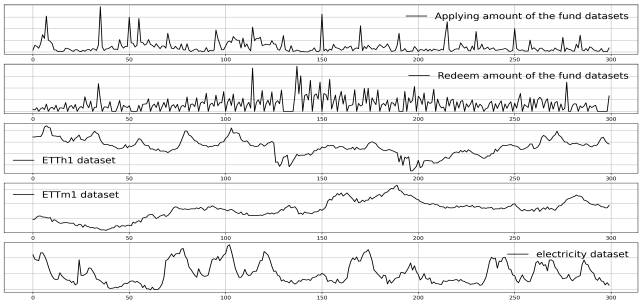


Figure 10: Time series visualization of Fund dataset (first two lines) and other public datasets.

A Appendix

A.1 More details about fund datasets

On the one hand, from the time series visualization in Figure 10, we can intuitively observe pattern differences between the Fund dataset and the public datasets. Influenced by long-term market conditions, short-term policy, and public opinion, the time series of fund sales exhibit steeper ascending and descending trends as well as more frequent fluctuations, containing complex semantics.

On the other hand, inspired by [31], in Table 8, we quantify the trend, periodicity, and kurtosis of the Fund dataset and other popular TSF public datasets. Fund dataset exhibits lower trend and periodicity along with higher kurtosis, indicating more complex TS [31] and bringing more challenges for TSF tasks.

By introducing the Fund datasets, we intend to expand the public dataset repository for more comprehensive evaluations of TSF algorithms. The features of each fund product are as follows:

(1) "product_pid" denotes the fund ID of the different fund products.
(2) "is_summarydate" denotes whether the transaction date of the fund product is a *summary date* (fund products do not trade on holidays and weekends, and the trading volume during these periods is aggregated to the next non-holiday or non-weekend, called *summary date*).
(3) "apply_amt" represents the applying transaction amount of the current fund product.
(4) "redeem_amt" represents the redemption transaction amount of the current fund product.
(5) "during_days" indicates the holding period of the current fund product (the number of days to hold the fund product before it can be traded).
(6) "is_trad" indicates whether the current day is a trading day.
(7) "is_weekend_delay" indicates whether it is a weekend before a trading day.
(8) "holiday_num" indicates how many statutory holidays occur before the trading day.

A.2 More details about public datasets

The following public datasets are extensively used for TSF algorithm evaluation, covering four practical applications: energy, economics, weather, and disease:

(1) The Electricity dataset collects the electricity consumption (kWh) every 15 minutes of 321 clients from 2012 to 2014.
(2) The ETT dataset comprises two sub-datasets, ETT1 and ETT2, collected from two separate counties. Each sub-dataset offers two versions with varying sampling resolutions (15 minutes and 1 hour). ETT dataset includes multiple time series of electrical loads and a single time sequence of oil temperature.
(3) The Weather dataset contains 21 meteorological indicators, such as air temperature, humidity, etc, recorded every 10 minutes for the entirety of 2020.
(4) Exchange dataset contains real-time exchange rates for eight different countries.
(5) The Illness dataset contains data on patients with influenza-like illnesses in the United States.

We have made the Fund and public datasets available at the link[1]. Our code link[2] provides the all data processing scripts.

Table 9: More comparison results of short-term TSF task on the collected Fund dataset.

		MLF		Autoformer		FEDformer		LSTM		RNN	
		MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.
Fund	1	**33.85**	**75.84**	36.5	82.17	36.85	80.31	51.41	85.91	36.2	79.82
		±0.033	**±0.093**	±0.027	±0.007	±0.002	±0.003	±0.18	±0.18	±0.08	±0.06
	5	**38.28**	**80.37**	45.45	93.79	47.92	97.4	62.98	101.88	40.42	83.52
		±0.029	**±0.071**	±0.038	±0.004	±0.038	±0.003	±0.18	±0.17	±0.02	±0.06
	8	**41.94**	**86.06**	51.05	99.9	45.2	90.3	69.94	109.04	44.69	89.74
		±0.031	**±0.123**	±0.144	±0.007	±0.001	±0.001	±0.1	±0.03	±0.04	±0.11
	10	**44.42**	**88.66**	52.72	100.35	47.1	91.92	71.68	109.78	46.67	91.55
		±0.057	**±0.041**	±0.104	±0.003	±0.037	±0.001	±0.11	±0.08	±0.06	±0.13

		DLinear		GRU		XGBoost		Prophet		ARIMA	
		MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.	MSE	WMA.
Fund	1	97.23	158.72	51.57	85.35	49.29	93.07	66.5	128.18	46.64	103.71
		±0.009	±0.002	±0.03	±0.01	-	-	-	-	-	-
	5	99.50	159.95	59.21	93.95	53.55	95.86	93.75	154.62	49.56	105.51
		±0.004	±0.003	±0.15	±0.23	-	-	-	-	-	-
	8	95.37	148.25	66.54	102.56	57.23	101.97	136.97	180.16	53.11	108.6
		±0.005	±0.004	±0.08	±0.11	-	-	-	-	-	-
	10	90.70	142.89	69.0	104.86	58.53	101.7	171.59	194.13	54.35	108.91
		±0.005	±0.004	±0.07	±0.11	-	-	-	-	-	-

A.3 Extra results for short-term TSF task

Here we compare MLF with more TSF baselines:

(1) Popular transformer models Autoformer [32] and FEDformer [42].
(2) Popular neural networks LSTM [12], RNN [10, 15], GRU [8] and DLinear [36].
(3) Popular machine learning models XGBoost [7], Prophet [28] and ARIMA [1].

[1]https://drive.google.com/drive/folders/1IecxNQqH6hYEgaZT70t273BFHV-WeIw1?usp=sharing
[2]https://github.com/Meteor-Stars/MLF

Table 10: Extra result of short-term TSF task on the public datasets.

	MLF		Autoformer		FEDformer		LSTM		DLinear		RNN		GRU	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.0873**	**0.1899**	0.2213	0.3429	0.2318	0.3495	0.1318	0.2389	0.2320	0.3345	0.1139	0.2224	0.1074	0.2146
ETTm1	**0.0412**	**0.1245**	0.0496	0.1382	0.0489	0.136	0.0458	0.1366	0.0659	0.1682	0.045	0.1357	0.0453	0.1349
Illness	**0.1493**	**0.2188**	0.3006	0.3367	0.3081	0.3409	0.852	0.5899	0.7842	0.6832	0.4092	0.3739	0.3688	0.3363
Excha.	**0.0029**	**0.0267**	0.0186	0.0959	0.0101	0.0685	0.015	0.0835	0.0080	0.0576	0.0094	0.0634	0.0079	0.0601

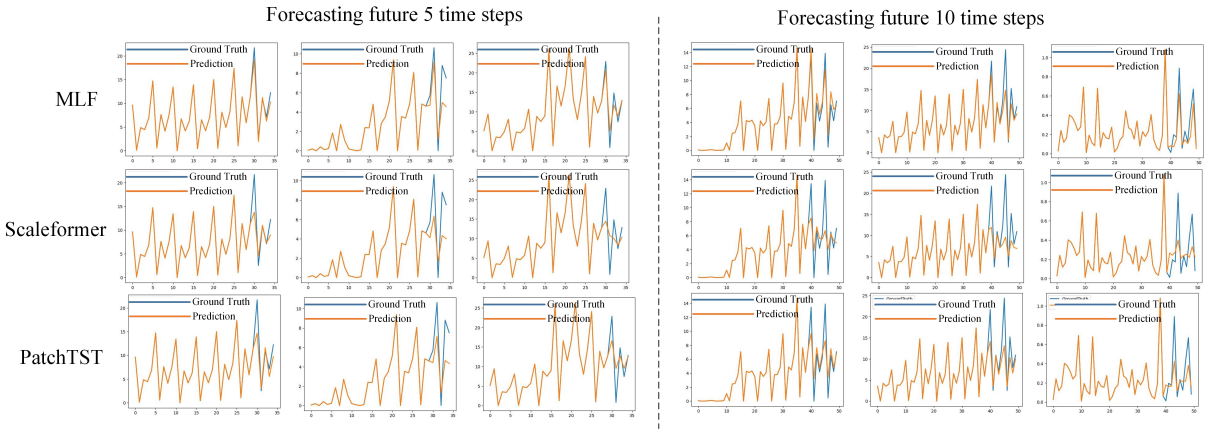


Figure 11: Visualizations of forecasting future 5 and 10 time steps on MLF and two strong baselines (Scaleformer and PtachTST) on Fund dataset.

Table 11: Extra results about the effectiveness of patch squeeze module: MLF's accuracy with squeeze factors 2 (MLF-2), 4 (MLF-4), and 8 (MLF-8).

		PatchTST		MLF-2		MLF-4		MLF-8	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	0.166	0.256	0.164	0.254	**0.163**	0.253	0.164	**0.252**
	192	0.223	0.296	0.220	0.295	**0.218**	**0.295**	0.223	0.297
	336	0.277	0.336	0.275	0.330	**0.269**	**0.329**	0.270	0.330
	720	0.357	0.381	**0.343**	0.380	0.346	**0.378**	0.344	0.379

For ARIMA, the auto-regressive order, differencing order, and moving average order are all set as 1 for better performance. **For Prophet,** the growth, changepoint range, and changepoint prior scale are set as "linear", 0.8 and 0.1 respectively. The seasonality prior scale and holidays prior scale are set as 10 for better performance. **For XGBoost,** the learning rate, max depth, and the number of estimators are set as 0.3, 12, and 1000 respectively. We use time and variable lag information to construct features for better performance. **For LSTM, RNN, and GRU**, the number of hidden layers is 3. The hidden size is set as 1024. For **Autoformer** and **FEDformer**, as with the baseline approach in the main experiments, we use the recommended hyperparameters in baseline codes.

The comparison results on Fund dataset and public datasets are shown in Table 9 and Table 10 respectively. Results show that the overall accuracy of MLF is the best and second is RNN. The traditional machine learning models have not gained an advantage in predicting fund sales. Moreover, we observed that models that perform well in long-term TSF task (e.g., DLinear, Autoformer, and FEDformer) do not perform as well in short-term TSF task, while MLF achieves desirable results in both short-term and long-term TSF tasks. This indicates the superiority of MLF.

A.4 Extra results for patch squeeze module

Table 11 shows the effectiveness of the patch squeeze module on the ETTm1 dataset. We also have applied Patch Squeeze to PatchTST with fixed input length 1536. Specifically, on weather dataset, the average error of original PatchTST is 0.2554, with training epoch time 248.87s. while setting squeeze factor as 2 or 4, corresponding results are 0.2550 or 0.260 and 83.50s or 44.26s, indicating remarkable improvements in efficiency. The same conclusion is reached on the ETTm datasets. This further demonstrates that the simple but effective Patch Squeeze module significantly improves efficiency while maintaining good accuracy.

A.5 Visualizations of forecasting on Fund dataset

As shown in Figure 11, we visualize the forecasting of MLF and two strong baselines (Scaleformer and PatchTST) on the Fund dataset. The visualization shows that MLF can better forecast the steeper ascending and descending trends, indicating the superiority of MLF, which utilizes customized designs for addressing and using multi-period inputs for better TSF effect.