# Towards Lightweight Time Series Forecasting: a Patch-wise Transformer with Weak Data Enriching

Meng Wang[1,†], Jintao Yang[1], Bin Yang[2,*], Hui Li[3,4], Tongxin Gong[1], Bo Yang[1,†], Jiangtao Cui[3]
[1]*School of Computer Science, Xi'an Polytechnic University, China*
[2]*School of Data Science and Engineering, East China Normal University, China*
[3]*School of Computer Science and Technology, Xidian University, China*
[4]*Shanghai Yunxi Technology, China*
wangmeng@xpu.edu.cn, byang@dase.ecnu.edu.cn, {220721093, 220721091}@stu.xpu.edu.cn,
{hli, cuijt}@xidian.edu.cn, yangboo@stu.xjtu.edu.cn

***Abstract*—Patch-wise Transformer based time series forecasting achieves superior accuracy. However, this superiority relies heavily on intricate model design with massive parameters, rendering both training and inference expensive, thus preventing their deployments on edge devices with limited resources and low latency requirements. In addition, existing methods often work in an autoregressive manner, which take into account only historical values, but ignore valuable, easy-to-obtain context information, such as weather forecasts, date and time of day. To contend with the two limitations, we propose LiPFormer, a novel Lightweight Patch-wise Transformer with weak data enriching. First, to simplify the Transformer backbone, LiPFormer employs a novel lightweight cross-patch attention and a linear transformation-based attention to eliminate Layer Normalization and Feed Forward Network, two heavy components in existing Transformers. Second, we propose a lightweight, weak data enriching module to provide additional, valuable weak supervision to the training. It enhances forecasting accuracy without significantly increasing model complexity as it does not involve expensive, human-labeling but using easily accessible context information. This facilitates the weak data enriching to plug-and-play on existing models. Extensive experiments on nine benchmark time series datasets demonstrate that LiPFormer outperforms state-of-the-art methods in accuracy, while significantly reducing parameter scale, training duration, and GPU memory usage. Deployment on an edge device reveals that LiPFormer takes only 1/3 inference time compared to classic Transformers. In addition, we demonstrate that the weak data enriching can integrate seamlessly into various Transformer based models to enhance their accuracy, suggesting its generality.**

***Index Terms*—Time Series Data Forecasting, Weak Data Enriching, Lightweight, Patch-wise Transformer**

## I. Introduction

Time series constitutes a chronological sequence of data points that record successive states of an event. Forecasting is a fundamental task in time series data analysis, which aims to predict future values by tracking historical observations. Time series forecasting has received considerable research attention due to its crucial role in a spectrum of applications, such as finance [1], weather [2], energy [3], and traffic [4], [5], [6], [7], [8], [9], [10]. Accurate forecasts can provide reliable data support, facilitating sound decision-making. For instance, industrial embedded sensors [11] collect real-time operational data (e.g., temperature, pressure) from machinery to anticipate equipment health and failure risks. Power grids [12] utilize historical data from smart device (e.g., power loads, renewable energy production) to predict future power demand and supply, ensuring grid stability.

Time series forecasting has achieved remarkable advancement with a variety of architectures [13], [14], [15], [16], [17], [18], [19], including the recurrent neural networks (RNNs) [20], [21], [22] and Transformers [23], [24]. Compared to RNNs, Transformers have revolutionized in both natural language processing (NLP) [25] and computer vision (CV) [26] fields via their attention mechanism, enabling better understanding of global correlations. Benefiting from that, specialized Transformer-based architectures for time series forecasting emerged, e.g., Informer [27], Autoformer [28], FEDformer [29]. These works attempt to preserve order information among time series data elements by Positional Encoding (PE). However, unlike natural language, the lack of semantics in numerical data makes PE invalid to capture sequential dependencies. Fortunately, the *Patching* technique, inspired by a recent linear strategy (DLinear) [30], segments time series data into subseries-level patches and assists Transformer-based models [31], [32], [33] in perceiving the order information.

Despite the progress made by patch-wise Transformers in time series forecasting, they face two substantial challenges.

**Challenge 1: Intricate Models with Massive Parameters.** Heavyweight Transformer models, characterized by complicated modules and extensive parameters, incur prohibitive resource requirements and latency. The vanilla Transformer [23] exhibits $O(N^2)$ complexity ($N$ denotes time series length), rendering it unfriendly for training and deployment in resource-constrained scenarios. Rapid-response time series analysis tasks, increasingly prevalent in industrial [34] and networking [35] domains, are hindered by edge devices' limited computational power and memory to execute intricate algorithms, especially early and low-cost devices. Designing a tailored Transformer model that simultaneously achieves lightweight architecture and enhanced predictive performance is urgently required. The crucial issue lies in differentiating between components in Transformers effective for time series

* Corresponding author.
† Affiliated with The Shaanxi Key Laboratory of Clothing Intelligence, China.

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00100

analysis and those of lesser importance. Developing strategies to optimize the former and simplify the latter is nontrivial.

**Challenge 2: Autoregressive forecasting without considering contexts.** Transformer-based models typically work in an autoregressive manner and predict time series solely on historical data, neglecting an explicit inductive bias: Future value changes (especially sudden changes) are highly correlated with future apriori contexts, like weather forecasts, date, or time of day. For instance, photovoltaic generation [36] fluctuates due to weather variations, which are irrelevant to past electricity generation series data. Similarly, textual pollution severity labels (heavy, moderate, light) can provide weak supervision to aid in $PM_{2.5}$ prediction [37]. Thus, incorporating easily obtainable weak labels as prior knowledge can enhance understanding of future time series dynamics. However, a unified multimodal framework is lacking in extracting features from explicit weak labels, such as textual (weather, wind direction, air pollution level, etc.) and numerical (temperature, humidity, etc.) covariates. Additionally, not all scenarios possess available explicit future covariates. While implicit weak labels (time of day, peak/non-rush hours, etc.) can be augmented via temporal feature encoding [27], [31], the absence of a decoder in patch-wise Transformer models hinders the availability of implicit future covariates for guiding predictions.

In order to explore the untapped potential of lightweight and weak labels in the Transformer architecture, this paper proposes a novel **Lightweight Patch-wise TransFormer with weak data enriching** (**LiPFormer**) for time series data forecasting, which offers superior performance in both universal and resource-constrained environments. Specifically, an attention-based lightweight backbone network and a weakly supervised future covariate framework are devised to tackle the challenges above.

**Lightweight Patch-wise Transformer.** The lightweight backbone network, based on the multi-head self-attention mechanism, solves the first limitation through three key strategies as follows. 1) ***Integrated cross-patch attention***. We incorporate a newly designed cross-patch attention mechanism to the existing patching technique, simultaneously reducing complexity and improving predictive capability. Patching time series data dramatically reduces the number of input sequences to $O(N^2/pl^2)$ ($pl$ denotes patch length). The capability of patching mechanism [31] to perceive local dependencies was inspired by DLinear. Motivated by trend components, another principle of DLinear, we extend to capture the global trend correlations across patches. By extracting fixed-position data points in each patch to construct trend sequences spanning all patches (details in Section III-C), our patch-wise attention effectively substitutes Positional Encoding in the vanilla Transformer, perceiving both local and global sequential information. 2) ***Layer Normalization elimination***. Given the cross-patch's sampling manner to capture global trends, it partially supplants the generalization function of Layer Normalization (LN). Moreover, the uniform patch size of time series avoids the issue of varying token lengths characteristic of natural language, diminishing LN's utility [38]. Hence, its exclusion in our design is justified. 3) **FFNs-less (Feed Forward Networks-less)** linear attention. Recognizing the fundamental semantic disparities between textual and numerical sequences, we devise a novel FFNs-less attention scheme employing linear transformations. Transformers were originally tailored for language tasks and the nonlinear mappings of FFNs are more inclined towards learning semantic and syntactic structural information. DLinear's insights suggest that linear connections might suffice for time series representation. Thus, we employ a lightweight Multi-Layer Perceptron (MLP) instead of the two-layer ascending and descending FFNs, significantly reducing the parameter scale from $O(8\times hd^2)$ to $O(hd\times pl)$ ($hd$ denotes hidden feature dimension). Encouragingly, deployment trials on an edge device with only CPU (with 16GB RAM, 6 cores, and 12 threads) reveal that the inference time of the lightweight LiPFormer is less than 1/3 of that of Transformer models.

**Weak Data Enriching**. Typically, acquiring high-quality future covariate data are often economically pricey and labor-intensive. Fortunately, weakly supervised learning mitigates this issue by leveraging data augmentation to lower modeling barriers. Specifically, publicly accessible weather forecasts serve as expert annotations, while date-related contexts are augmented when explicit future features, like weather forecasts, are unavailable. The weak label enriching architecture branches into two policies depending on the availability of explicit future covariates. 1) ***For scenarios with explicit weak label***, there exist multimodal nature and cross-channel correlations. Textual and numerical future attributes are encoded into vectors with the same dimension, followed by an attention module to capture their dependencies. Similar to the backbone network, we employ a simplified MLP instead of the heavyweight Transformer for output, achieving a tradeoff between prediction accuracy and efficiency. 2) ***In the absence of explicit covariates***, temporal attributes (e.g., holidays, weekdays, rush hours) implicitly embraces meaningful semantic information. We encode these textual future covariates and embed them in a semantic space to maximize their correlation with the future ground truth time series (referred to as target sequences). We devise a contrastive learning framework for the augmented temporal features, adopting a dual encoder module—target sequences vs. future covariates—to effectively model their latent correlation. Notably, benefiting from the aligned dimension of target sequences and future covariates, the weak label enriching architecture can be seamlessly transplanted into existing time series forecasting frameworks and enhance their prediction capacities (detailed in Section III-C).

Our main contributions in this paper are outlined as follows.

- We propose LiPFormer, a novel patch-wise Transformer architecture with weak label enriching that lightweights the backbone network and integrates a weakly supervised architecture for future covariates, enhancing the predictive performance in time series forecasting.
- We present a novel weakly supervised architecture to learn the impact of future covariates. Based on data augmentation, a dual encoder contrastive learning framework is designed to uniformly model the correlation between

textual, numerical, and implicit temporal future contexts.
- We innovate by integrating a cross-patch attention mechanism, eliminating Layer Normalization and Positional Encoding, and devising an FFNs-less linear attention to successfully simplify the Transformer architecture.
- Extensive evaluations on benchmark datasets demonstrate that LiPFormer significantly outperforms state-of-the-art methods in training speed, inference time, parameter size and accuracy. Deployment trials on a CPU-only edge device further confirm LiPFormer's lightweight superiority in resource-constrained scenarios over Transformer.
- Experiments on two real-world datasets with explicit future covariates validate the superiority of our weakly supervised architecture. Extension tests reveal it can be seamlessly transplanted into diverse time series forecasting models, contributing to improved performance.

## II. RELATED WORK

In this section, we overview existing time series forecasting models classified into four categories: Transformer-based, MLP-based, Deep Learning models, and patch-wise models.

**Transformers**: RNNs were introduced to model temporal dependencies [39], [40], [41] for short-term series prediction. A recent study [37] followed the general autoregressive manner to predict $PM_{2.5}$, leveraging easy-to-obtain categorical information to supervise the prediction. However, RNNs were susceptible to the vanishing gradient when handling long-term series. Based on self-attention mechanism, Transformer models [23] avoided the recurrent structure and were superior in capturing long-term dependencies. Nonetheless, attention mechanism suffers from high time and space complexity $O(N^2)$ and insensitivity to local context. LogTrans [42] proposed sparse convolutional self-attention to reduce complexity to $O(N(\log N)^2)$ incorporating local context into attention. Reformer [43] employed a locality-sensitive hashing attention, taking into account attention vectors only within the same hash buckets. Reversible residuals reduced complexity to $O(N \log N)$. Zhou et al. [27] introduced ProbSparse self-attention, exploiting sparsity in attention parameters to decrease the complexity. Conformer [44] reduces the complexity of the attention mechanism by $O(N)$ using window attention and a novel RNN network. Triformer [45] leveraged a triangular structure and matrix factorization to achieve linear complexity. Inspired by stochastic processes, Wu et al. [28] devised an autocorrelation mechanism, substituting the traditional attention with a series-wise one. Instead of optimizing attention, PatchTST [31] and Crossformer [32] explored a different patching strategy to achieved efficiency gains. Recently, iTransformer [46] employed variate-wise attention to facilitate information exchange among variables. Despite the above advancements in tackling the local-agnostics issue from various attention mechanisms or patching techniques, they still inadequately capture global sequential trend information, which limited the performance of Transformer models [30].

**MLPs**: Transformers had been the prevailing method for time series forecasting until a linear alternative challenged their supremacy. Zeng et al. [30] proposed a direct multi-step DLinear method, essentially an MLP, outperforming conventional Transformers. The authors decomposited time series into trend and seasonal components, which inspired numerous linear studies [33], [47], [48], [49], [50]. MLP-Mixer [33] effectively replaced self-attention with an MLP and contained patch processing. Inspired by visual MLP mixers, Chen et al. [47] devised a two-stage framework with mixer layers and temporal projections. TiDE [48] excelled under channel independence assumptions using a residual structure. Recently, a LightTS [49] framework employed distillation and Pareto optimality techniques to substitute computationally intensive ensemble learning. TimeMixer [51] leveraged multi-resolution sequences in two Mixing modules for past and future feature extraction. While some of these models accounted for the impact of date-related implicit features on predictions, the lack of modeling future weak label (external covariates) limits their ability to exploit prior knowledge, which provides valuable supervision to enhance forecasting.

**Deep learning models**: FourierGNN [52] innovatively treats time series values as graph nodes and performs predictions on hypervariable graphs. Deng et al. [53] opted for an alternative policy, designing an SCNN network to individually model each component of the spatio-temporal patterns.

**Patching Models**: Most existing approaches directly utilized entire time series as model inputs, whereas PatchTST [31] and TSMixer [33] adopted a distinctive patching strategy. Patch-wise methods divide time series into subseries-level patches. Then it treats patches as input tokens to learn dependencies via attention mechanism. Data in a patch preserve local order information. However, as discussed earlier, patching attention lacks of global order awareness and the fixed patch size fails to accommodate different temporal scales, degrading model generalization.

Table I summarizes representative time series forecasting methods, categorized according to their lightweight nature and consideration of future weak label. Our proposed LiPFormer bridges the gap in this research field.

TABLE I
REPRESENTATIVE TIME SERIES FORECASTING MODELS.

| | Heavyweight models | Lightweight models |
|---|---|---|
| Not consider weak label | RNNs [39], [40], [41]<br>Transformer [23]<br>Autoformer [28]<br>LogTrans [42]<br>PatchTST [31]<br>Crossformer [32]<br>SCNN [53]<br>Informer [27] | LightTS [49]<br>Reformer [43]<br>Conformer [44]<br>Triformer [45]<br>DLinear [30]<br>TSMixer [33] |
| Consider weak label | CGF [37]<br>TiDE [48] | LiPFormer (Ours) |

## III. METHODOLOGY

In this section, we first define the notations and formally describe the time series forecasting task. Then we elaborate the training paradigm and our proposed lightweight Transformer architecture, following the order of data flow.

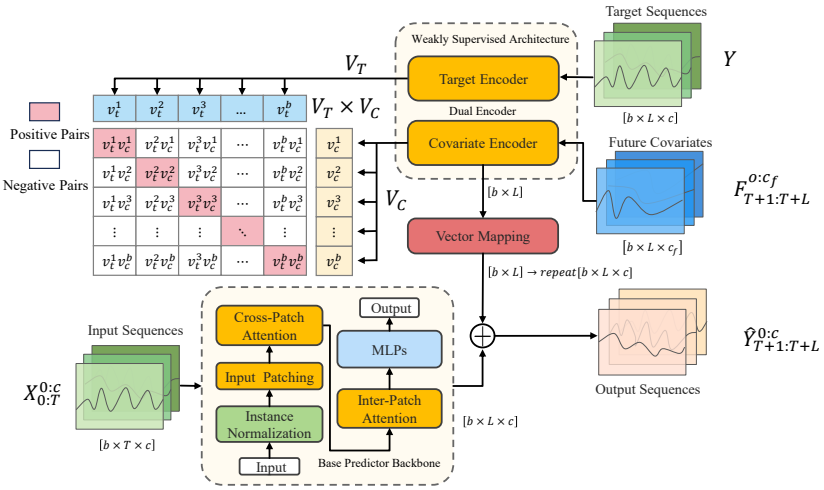


Fig. 1. The architecture of LiPFormer. The Base Predictor backbone network comprises two patch-wise attentions and simplified MLPs. The weakly supervised Dual Encoder is a contrastive learning architecture, consisting of a Covariate Encoder and a Target Encoder, to model the correlation between future attributes.

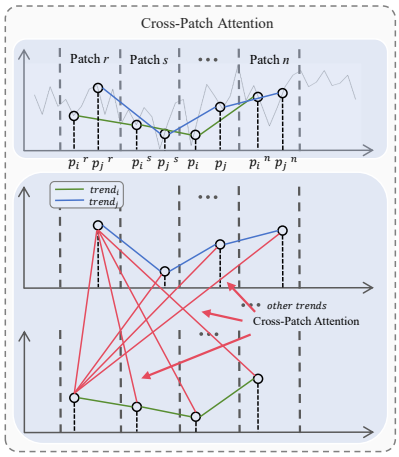


Fig. 2. The construction of trend sequences and Cross-Patch attention.

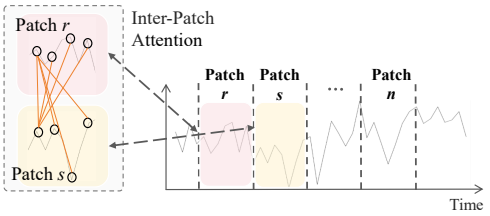


Fig. 3. Patch division and Inter-Patch attention.

### A. Notations

Frequently used notations in this paper are defined as follows. $c$ and $c_f$: channels (features) of a mutivarite time series and the corresponding future covariates, they possibly are different; $X_{0:T}^{0:c}$: an input time series (sequences) of length $T$ that consists of $T$ historical data points (a.k.a. time steps); $F_{T+1:T+L}^{0:c_f}$: future covariates of length $L$; $Y_{T+1:T+L}^{0:c}$ and $\hat{Y}_{T+1:T+L}^{0:c}$: the ground truth and forecast future sequence values of length $L$, respectively; $\hat{Y}_{base}$: intermediate prediction results of Base Predictor, independent of future covariates; $n$: number of patches; $pl$: patch length; $nt$: number of target patches, which is the ratio of predictive length divided by patch length; $hd$: hidden feature dimensions; $b$: batch size.

Given a historical time series $X_{0:T}^{0:c}$ and future covariates $F_{T+1:T+L}^{0:c_f}$, the multivariate forecasting task can be formally defined as the prediction of future values:

$$\hat{Y}_{T+1:T+L}^{0:c} = H(X_{0:T}^{0:c}, F_{T+1:T+L}^{0:c_f}),$$

where $H$ denotes the proposed forecasting architecture.

### B. Training Methodologies

LiPFormer involves two main training process, pre-training and prediction-oriented training. The weak label enriching part, depicted as Weakly Supervised Architecture in the top part of Figure 1, is a contrastive learning-based pre-training module with dual encoders that characterize target sequences (i.e., ground truth future time series) and weak labels, respectively. The backbone network, Base Predictor, performs prediction-oriented training over input sequences.

Departing from prior representation learning methods that focus on encoding input sequences, we aim to extract representation vectors for easily accessible weak labels. As discussed in Section I, weak label enriching can provide additional and valuable weak supervision to the prediction. We leverage both explicit (textual and numerical external factors, e.g., weather, temperature) and implicit (temporal attributes, e.g., date, time of day) weak labels as future covariates. We follow a paradigm to regard future covariates as expert annotations, and utilize temporal information to augment weak data. Notably, the pre-training of weak labels serves to guide the Base Predictor in making predictions, instead of functioning as encoders to embed themselves as extra features of input time series.

Since patch-wise Transformers do not possess any decoder, in the absence of explicit weak labels, they cannot leverage future temporal features for guiding the Base Predictor. To this end, we devise the contrastive learning-based pre-training architecture (detailed in Section III-C2) to attain the implicit future feature representations via a joint encoding of "covariate-target" pair. Given a batch of $b$ covariate-target pairs, the "covariate" (resp., "target") element corresponds to a representative vector $\mathcal{V}_c^{(j)}$ (resp., $\mathcal{V}_t^{(i)}$) of future covariates $\mathcal{V}_\mathcal{C} = \{\mathcal{V}_c^{(1)}, \cdots, \mathcal{V}_c^{(b)}\}$ (resp., target sequences $\mathcal{V}_\mathcal{T} = \{\mathcal{V}_t^{(1)}, \cdots, \mathcal{V}_t^{(b)}\}$), where $\mathcal{V}_c^{(j)}$ (resp., $\mathcal{V}_t^{(i)}$) is of length $L$. This pre-training process is conducted upon learning the two representation vectors $\mathcal{V}_\mathcal{C}, \mathcal{V}_\mathcal{T}$ to estimate which of the $b^2$ pairs actually occurred. Specifically, we train dual encoders, Covariate Encoder and Target Encoder, aiming at maximizing

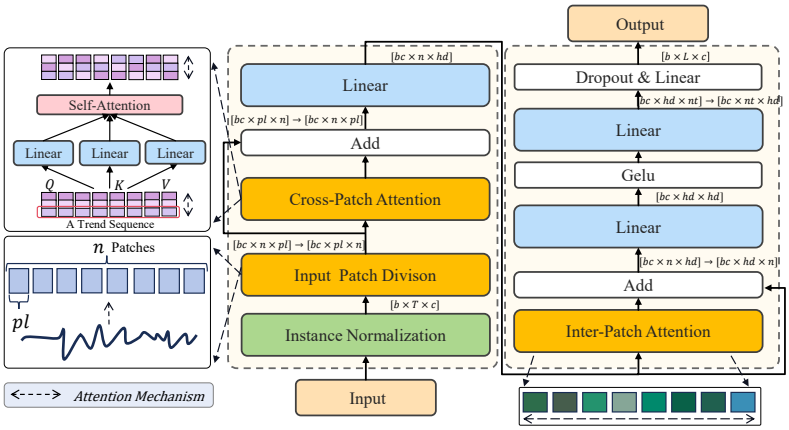


Fig. 4. The structure of Base Predictor block, which mainly comprises two patch-wise attentions and simplified MLPs.

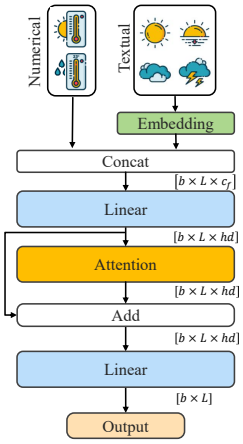


Fig. 5. The structure of Covariate Encoder, which uses Res-attention and linear layers to model numerical and textual weak label supervision.

the cosine similarity of the $b$ diagonal positive pairs while minimizing the embedding of the remaining $b^2 - b$ negative pairs. We use symmetric cross-entropy loss $\mathcal{L}_{sce}$ to optimize the cosine similarity score as follows:

$$\mathcal{L}_{sce} = \frac{1}{2}(\mathcal{L}_{ce}(logits, labels)^{(0)} + \mathcal{L}_{ce}(logits, labels))^{(1)}),$$

where $logits = (\mathcal{V}_{\mathcal{T}} \times \mathcal{V}_{\mathcal{C}}) \cdot e^t \in \mathbb{R}^{b \times b}$ and $labels = (1, \cdots, b)$ are conceptually consistent with those in CLIP [54]. $\mathcal{L}_{ce}(p, q) = -\sum_x p(x) \log(q(x))$ denotes cross-entropy loss, and the superscript $(0)$ and $(1)$ in $\mathcal{L}_{sce}$ means to calculate $\mathcal{L}_{ce}(p, q)$ separately by row and column.

In the Base Predictor training process, the input time series are normalized and patched, then fed into the lightweight backbone network. The prediction head outputs the base prediction $\hat{Y}_{base}$, which is guided by the aforementioned future covariate dual encoders to get the future sequence prediction $\hat{Y}$. To achieve a balance between accuracy, convergence, and robustness during the backbone network training, we adopt Smooth $\mathcal{L}1$ loss ($\mathcal{L}1_{Smooth}$) [55]. When the error between the predicted and true values is small, $\mathcal{L}1_{Smooth}$ employs $\mathcal{L}2$ loss (Mean Squared Error, MSE) to maintain high accuracy, with its differentiable nature aiding convergence during training. Conversely, for larger errors, $\mathcal{L}1_{Smooth}$ uses $\mathcal{L}1$ loss (Mean Absolute Error, MAE), which is less sensitive to outliers, thus reducing the risk of gradient explosion. The hyperparameter $\beta$ in $\mathcal{L}1_{Smooth}$ is used to adjust the error threshold, providing learning flexibility for diverse time series datasets.

$$\mathcal{L}1_{Smooth} = \begin{cases} \frac{1}{2\beta} \left\| Y - \hat{Y} \right\|^2 & if \left\| Y - \hat{Y} \right\| < \beta, \\ \left\| Y - \hat{Y} \right\| - \frac{\beta}{2} & otherwise. \end{cases}$$

### C. Model Components

In this subsection, the first part delves into the intricate architecture of the Base Predictor, outlining its constituent components and functional mechanisms. In the second part (Section III-C2), we elaborate the details of Weakly Supervised dual encoder framework, specifically tailored for the purpose of learning meaningful representations of future weak labels. The entire end-to-end workflow, inclusive of the modular structures and data processing steps for each mini-batch, is depicted in Figure 1.

*1)* ***Base Predictor:*** As illustrated in Figure 4, the backbone network of LiPFormer first eliminates LN, as the fixed patch size evades the issue with varying input sequence lengths. Secondly, adopting patching and channel independence preserves local order information and extends the receptive horizon to learn correlations between features and time series. Notably, existing patch-wise models commonly adhere to a fixed patch size, treating each patch as an individual token akin to text-oriented models. However, this rigidity in patch scales often hampers the capacity to effectively capture temporally varying periodic attributes inherent in time series. As a result, such models demonstrate limited adaptability to the task of time series prediction. This presents a significant challenge in holistically capturing information encapsulated within both individual patches and the broader sequence context, while maintaining a constant patch length. To overcome this issue, based on the patch division of input sequences, LiPFormer applies two novel patch-wise attention mechanisms, Cross-Patch and Inter-Patch, to comprehensively understand global and local sequential dependencies. Finally, in light of the inherent differences between numerical and textual sequences, the heavyweight FFNs module originally devised for understanding language information is replaced with two different single-layer MLPs, resulting in the base prediction $\hat{Y}_{base}$.

**Instance Normalization.** Time series may encounter distribution shifts. To mitigate this issue, we employ a straightforward normalization approach [30] without significantly increasing model complexity. The initial input time series $x^i_{0:T}$ are subtracted by the last value $x^i_T$ from each element, resulting in a new input sequences $x^{i\prime}_{0:T}$, and subsequently re-

adding it at the intermediate output $\hat{y}^{i\prime}_{T+1:T+L}$, denoted as:

$$x^{i\prime}_{0:T} = x^{i}_{0:T} - x^{i}_{T}, \quad \hat{y}^{i}_{T+1:T+L} = \hat{y}^{i\prime}_{T+1:T+L} + x^{i}_{T}.$$

**Channel Independence and Patching.** The use of multivariate time series data to simultaneously predict all variables may be an inductive bias for time series forecasting, as the interrelationships between variables can be captured. However, recent researches [31], [30], [48] have shown that this idea is not an optimal solution. The channel independent strategy maps only univariate sequences to future values without taking into account the mutual information of other variables. This design enables each univariate to independently process data while sharing a common weight space, enhancing local semantic capturing and extending the observation horizon.

Additionally, we divide each univariate sequence into patches of size $pl$. A mini-batch $X^{0:c}_{b\times T}$ of batch size $b$ will be reshaped into $X^{0:c}_{b\times n\times pl}$, where $n = \lceil T/pl \rceil$ denotes the number of patches. With the patching technique, the number of input tokens (i.e., data units) can be reduced by a factor of $pl$ and the complexity dramatically drops to $O(N^2/pl^2)$, which significantly improves the model performance compared to the standard point-wise Transformer [31]. On the other hand, data points in a patch are treated while preserving local coherence, thus exploiting local dependencies among them.

**Novel Patch-wise Attention.** The aforementioned patching manner logically treats each patch as a token. However, due to the variations in local granularity (e.g., periodicity) across diverse time series datasets, it is difficult to predetermine an appropriate patch size, limiting its generalization capability. Also, the patching mechanism does not intend to perceive the global order dependencies among tokens. To address these limitations, we introduce a novel **Cross-Patch** strategy.

We construct a *global trend sequence* by arranging data points from a fixed position within patches in chronological order, as the construction of $trend_i$ and $trend_j$ shown in Figure 2. According to the patch length, we can obtain a total of $pl$ trend sequences, each lagged by one time step. By applying attention across all these trend sequences, we can capture the mutual information at different positions of all the patches, perceiving sequential dependencies in the global trend. The lagging-like relationship among trend sequences is conducive to understanding the impact of historical information on future trends. Since data points in a trend sequence span beyond the scope of an individual patch, the generalization capability is less constrained by the fixed patch size, and multiple trend sequences collectively further enhance this generalization.

To explore correlations between patches, we map internal data points of patches onto an $hd$-dimensional latent feature vector using a MLP between them [33], [31], referred to as the **Inter-Patch** attention mechanism:

$$X^{0:c}_{b\times n\times hd} = MLP(X^{0:c}_{b\times n\times pl}).$$

Figure 3 illustrates how an input time series is divided into patches and the proposed Inter-Patch attention works. For a particular patch $r$, the Inter-Patch attention mechanism focuses on the relevance between its data points to all counterparts within other patches, such as patch $s$. Similar to the self-attention mechanism, the Inter-Patch attention captures correlations between patch tokens.

Inspired by the mixer technique in CV domain [47], the Cross-Patch attention performs feature mixing alongside the Inter-Patch attention with a MLP, enabling LiPFormer to learn correlations between local and trend features, thereby further alleviating degradation in generalization performance:

$$x^{0:c}_{b\times n\times hd} = MLP(Attn(X^{0:c}_{b\times n\times pl}) + X^{0:c}_{b\times n\times pl}), \quad (1)$$

where $Attn$ denotes the self-attention mechanism of Vanilla Transformer [23], mapping $X^{i}_{b\times n\times pl}$ to query matrices $Q^i = (x^i)^T\mathbf{W}^Q$, key matrices $K^i = (x^i)^T\mathbf{W}^K$, and value matrices $V^i = (x^i)^T\mathbf{W}^V$, with trainable parameter matrices $\mathbf{W}^{(\cdot)}$ and utilizing the $softmax$ function:

$$Attn(X^{i}_{b\times n\times pl}) = Softmax(\frac{Q^i {K^i}^T}{\sqrt{d_k}})\,V^i.$$

**Elimination of Positional Encoding.** The positional information that represents the order of data points in time series is of great importance. As the self-attention mechanism cannot effectively retain the order information, some types of Positional Encoding (PE) were utilized by classical Informer [27], Autoformer [28], and FEDformer [29]. For the sake of simplifying internal details, these PE approaches for enhancing the temporal order of time series inputs are uniformly represented by the trainable matrices $\mathbf{W}^{PE}$. Then the existing attention mechanism with Positional Encoding can be depicted as $Attn(x^{0:c}_{b\times n\times hd} + \mathbf{W}^{PE})$.

Unlike the traditional Transformers, by using the Inter-Patch attention, it is available to capture comprehensive positional information both at the point and patch levels without using any Positional Encoding method. As a result, we eliminate it between patches for direct inputs:

$$Attn_{Inter-Patch} = Attn(x^{0:c}_{b\times n\times hd}). \quad (2)$$

**Lightweight Architecture.** The aforementioned patching technique reduces the time complexity of Transformer's attention mechanism to $O(N^2/pl^2)$, while it is still computationally expensive. To this end, we explore the impact of Transformer's other auxiliary modules on time series modeling. At the beginning of this section, we clarified substituting Layer Normalization, mainly pertinent to token length variations in NLP tasks and with limited effect on time series forecasting [38], with Instance Normalization. Our experiments demonstrate that LN tends to have no improvement in the accuracy of time series prediction and over-deep layers can lead to overfitting due to the applied channel independent strategy.

In addition, Feed Forward Networks (FFNs) in the Transformer architecture, which are used to perform nonlinear mapping to learn semantic and syntactic structural information, is overly heavy and not effective in capturing time series features. Similar to LN, the FFNs module lacks specificity for numerical time series data and exhibits prohibitively expensive computational overhead. Hence, we devise two linear transformation-based single-layer MLPs to capture linear

dependencies. Specifically, we change the FFNs to directly predict $\hat{Y}_{base}$ instead of making residuals and then predicting through the linear layer. The specific shape changes are as follows (also shown in Figure 4), where the arrowheads denote reshaping operations, and $b \cdot c$ indicates the single dimension size into which the batch size $b$ and the number of channels $c$ are reshaped:

$$\hat{Y}_{base} = MLP(Attn_{wo/p}) \in \mathbb{R}^{b \cdot c \times n \times hd} \rightarrow \mathbb{R}^{b \cdot c \times hd \times nt} \rightarrow \mathbb{R}^{b \times L \times c}$$

In summary, the theoretical analyses in this subsection and empirical studies below (detailed in Section IV-E4) give us the confidence to eliminate PE, LN and FFNs layers from the standard and patch-wise Transformers.

*2) **Covariate Encoder:*** Existing time series forecasting models, designed predominantly on publicly accessible datasets, typically neglect the influence of future weak labels on prediction, despite the intuitive and strong correlations between them. Bridging this gap necessitates a unified framework to handle datasets with and without future covariates. However, linear models are inadequate for capturing nonlinear relationships between time series and future covariates, which can be textual and numerical attributes.

To this end, we propose a nonlinear module for weak label enriching. For datasets inclusive of explicit future covariates, we employ a co-trained encoder alongside the Base Predictor. Conversely, for datasets lacking such future weak label covariates, we augment weak data with temporal feature encodings [27] that align data dimension with future values in the latent space. Inspired by CLIP [54], we formulate data pairs in the form of $(F^{0:c_f}_{T+1:T+L}, Y^{0:c}_{T+1:T+L})$, where the two elements denote the feature encoding of future covariates and the target sequences, respectively.

In particular, we devise a simplified Transformer-based architecture, Covariate Encoder, for encoding future covariates (see Figure 5). The encoder classifies input weak labels into two categories: numerical (e.g., temperature, humidity) and textual (e.g., wind direction, date, day of the week). We first encode textual weak labels into embeddings and then concatenate them with numerical labels, denoted as follows:

$$F^{0:c_f}_{CatEmb} = Concat(Embed(F^{c_t}), F^{c_n}) \in \mathbb{R}^{b \times L \times c_f}, \quad (3)$$

where $Embed$ and $Concat$ are the corresponding operations, $c_n$ and $c_t$ are the numbers of numerical and textual channels, and $c_f = c_n + c_t$.

Next, the concatenated data are first fed into a linear MLP layer to map all channels to a hidden feature size of $hd$, followed by a fully flattened operation (denoted as $Flat$) over a residual self-attention mechanism. Finally, two encoded vectors (i.e., $\mathcal{V}_\mathcal{C}$ and $\mathcal{V}_\mathcal{T}$) of the same dimension are obtained by another MLP layer, and we make the dual encoders converge by the pre-training paradigm introduced in subsection III-B. The Covariate Encoder works as below:

$$F^{c_f}_{MLP} = MLP(F^{0:c_f}_{CatEmb}) \in \mathbb{R}^{b \times L \times hd}, \quad (4)$$

$$F^{c_f}_{FlatAttn} = Flat(Attn(F^{c_f}_{MLP}) + F^{c_f}_{MLP}) \in \mathbb{R}^{b \times L \cdot hd}, \quad (5)$$

$$F^{c_f}_{PreTrain} = MLP(F^{c_f}_{FlatAttn}) \in \mathbb{R}^{b \times L}. \quad (6)$$

Note that, the Target Encoder, designed for encoding target sequences, adopts an architecture akin to that of the Covariate Encoder. The Target Encoder dispenses with embedding and concatenation steps, necessitating merely the direct replacement of $F^{c_f}_{MLP}$ in Equation (5) with the following $F^{c}_{MLP}$:

$$F^{c}_{MLP} = MLP(Y^{0:c}_{T+1:T+L}) \in \mathbb{R}^{b \times L \times hd}. \quad (7)$$

During prediction, we freeze the parameters of the Covariate Encoder and map the representation vectors to the same dimensions as the output of the Base Predictor through a learnable linear layer. As illustrated in Figure 1, the Vector Mapping linear layer is trained alongside the Base Predictor, enabling it to learn the relative contributions of the Covariate Encoder's and backbone network's parameters. By weighting these contributions, the pre-trained knowledge captured from weak labels helps to compensate for biases in the target sequences, thereby guiding the final prediction results.

$$\hat{Y} = \hat{Y}_{base} + MLP(F^{c_f}_{PreTrain}) \in \mathbb{R}^{b \times L \times c} \quad (8)$$

TABLE II
THE STATISTICS OF DATASETS.

| Datasets | ETTh1 | ETTh2 | ETTm1 | ETTm2 | Weather | Electricity | Traffic | Electri-Price | Cycle |
|---|---|---|---|---|---|---|---|---|---|
| Variables | 7 | 7 | 7 | 7 | 21 | 321 | 862 | 40 | 22 |
| Timestamps | 17420 | 17420 | 69680 | 69680 | 52696 | 26304 | 17544 | 35808 | 21864 |
| Split Ratio | 6:2:2 | 6:2:2 | 6:2:2 | 6:2:2 | 7:1:2 | 7:1:2 | 7:1:2 | 7:1:2 | 7:1:2 |

## IV. EXPERIMENTS

This section presents experimental studies on the proposed LiPFormer compared with state-of-the-art (SOTA) models over a series of forecasting benchmarks. Performance comparisons between Transformer-based methods are conducted in terms of training time and running memory. Ablation studies on Layer Normalization and Feed Forward Network demonstrate the effectiveness of lightweight.

### A. Experimental Settings

*1) **Datasets:*** We evaluate the performance of the proposed LiPFormer model on seven multivariate time series datasets, including Weather[1], Traffic[2], Electricity[3], and 4 ETT datasets[4] (ETTh1, ETTh2, ETTm1, ETTm2). The statistics of these datasets are summarized in Table III. These datasets have been extensively utilized in the literature [30], [31], [27] for benchmarking time series forecasting models and are publically available in [30]. We follow the same data loading parameters (i.e., train:validate:test split ratio) in [30]. To evaluate the

[1]https://www.bgc-jena.mpg.de/wetter/
[2]https://pems.dot.ca.gov/
[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[4]https://github.com/zhouhaoyi/ETDataset

TABLE III

MULTIVARIATE LONG-TERM TIME SERIES FORECASTING RESULTS WITH LIPFORMER. ELECTRI-PRICE AND CYCLE ARE TWO DATASETS WITH FUTURE FEATURES. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD, AND THE SECOND-BEST RESULTS ARE UNDERLINED.

| Models | | **LiPFormer** | | | iTransformer | | | TimeMixer | | | FGNN | | | PatchTST | | | DLinear | | | TiDE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | Efficiency | MSE | MAE | Efficiency | MSE | MAE | Efficiency | MSE | MAE | Efficiency | MSE | MAE | Efficiency | MSE | MAE | Efficiency | MSE | MAE | Efficiency |
| ETTh1 | 96 | **0.359** | **0.379** | 0.75s | 0.392 | 0.423 | 1.17s | 0.385 | 0.417 | 2.87s | 0.647 | 0.561 | 0.58s | 0.375 | 0.399 | 2.11s | 0.375 | 0.399 | 0.28s | 0.375 | 0.398 | 7.84s |
| | 192 | **0.404** | **0.405** | 0.14s | 0.437 | 0.455 | 0.19s | 0.424 | 0.443 | 0.58s | 0.688 | 0.583 | 0.1s | 0.414 | 0.421 | 0.32s | 0.405 | 0.436 | 0.05s | 0.412 | 0.422 | 0.96s |
| | 336 | 0.444 | **0.424** | 0.27G | 0.456 | 0.469 | 18.02G | 0.456 | 0.459 | 1.42T | 0.704 | 0.601 | 1.05G | **0.431** | 0.436 | 17.09G | 0.439 | 0.443 | 4.15M | 0.435 | 0.433 | 1.13G |
| | 720 | 0.450 | **0.453** | 66K | 0.553 | 0.537 | 6.4M | 0.601 | 0.559 | 4.27M | 0.772 | 0.654 | 592K | **0.449** | 0.466 | 6.90M | 0.472 | 0.490 | 18.62K | 0.454 | 0.465 | 2.53M |
| ETTh2 | 96 | **0.265** | **0.327** | 0.80s | 0.303 | 0.364 | 1.04s | 0.296 | 0.354 | 2.42s | 0.479 | 0.496 | 0.6s | 0.274 | 0.336 | 2.07s | 0.289 | 0.353 | 0.29s | 0.27 | 0.336 | 7.83s |
| | 192 | 0.335 | **0.374** | 0.14s | 0.409 | 0.422 | 0.19s | 0.384 | 0.415 | 0.59s | 0.568 | 0.540 | 0.15s | 0.339 | 0.379 | 0.33s | 0.383 | 0.418 | 0.04s | **0.332** | 0.38 | 1.27s |
| | 336 | 0.364 | 0.395 | 0.27G | 0.440 | 0.450 | 18.02G | 0.383 | 0.423 | 1.42T | 0.692 | 0.604 | 1.05G | **0.331** | **0.380** | 17.09G | 0.480 | 0.465 | 4.15M | 0.36 | 0.407 | 1.13G |
| | 720 | 0.392 | 0.425 | 66K | 0.439 | 0.468 | 6.4M | 0.399 | 0.453 | 4.27M | 1.107 | 0.774 | 592K | **0.379** | **0.422** | 6.90M | 0.605 | 0.551 | 18.62K | 0.419 | 0.451 | 2.53M |
| ETTm1 | 96 | 0.296 | **0.338** | 3.1s | 0.318 | 0.366 | 3.56s | 0.305 | 0.358 | 10.62s | 0.403 | 0.427 | 4.49s | **0.290** | 0.342 | 8.55s | 0.299 | 0.343 | 1.12s | 0.306 | 0.349 | 32.51s |
| | 192 | 0.336 | **0.360** | 0.55 | 0.347 | 0.387 | 0.71s | 0.343 | 0.379 | 2.15s | 0.426 | 0.440 | 0.46s | **0.332** | 0.369 | 1.54s | 0.335 | 0.365 | 0.20s | 0.335 | 0.366 | 4.64s |
| | 336 | 0.365 | **0.379** | 0.27G | 0.380 | 0.405 | 18.02G | 0.371 | 0.394 | 1.42T | 0.450 | 0.454 | 1.05G | 0.366 | 0.453 | 17.09G | 0.369 | 0.386 | 4.15M | **0.364** | 0.384 | 1.13G |
| | 720 | **0.408** | **0.413** | 66K | 0.436 | 0.439 | 6.4M | 0.427 | 0.423 | 4.27M | 0.498 | 0.481 | 592K | 0.420 | 0.533 | 6.90M | 0.425 | 0.421 | 18.62K | 0.413 | **0.413** | 2.53M |
| ETTm2 | 96 | **0.160** | **0.244** | 3.24s | 0.180 | 0.273 | 2.89s | 0.181 | 0.270 | 9.45s | 0.225 | 0.322 | 2.74s | 0.165 | 0.255 | 7.68s | 0.167 | 0.260 | 1.21s | 0.161 | 0.251 | 32.01s |
| | 192 | 0.217 | **0.285** | 0.55 | 0.243 | 0.315 | 0.74s | 0.239 | 0.313 | 2.07s | 0.296 | 0.368 | 0.56s | 0.220 | 0.292 | 2.05s | 0.224 | 0.303 | 0.21s | **0.215** | 0.289 | 4.21s |
| | 336 | 0.273 | **0.322** | 0.27G | 0.299 | 0.352 | 18.02G | 0.289 | 0.340 | 1.42T | 0.423 | 0.460 | 1.05G | 0.278 | 0.329 | 17.09G | 0.281 | 0.342 | 4.15M | **0.267** | 0.326 | 1.13G |
| | 720 | **0.348** | **0.372** | 66K | 0.382 | 0.405 | 6.4M | 0.452 | 0.455 | 4.27M | 0.497 | 0.493 | 592K | 0.367 | 0.385 | 6.90M | 0.397 | 0.421 | 18.62K | 0.352 | 0.383 | 2.53M |
| Electricity | 96 | 0.131 | 0.224 | 5.12s | 0.147 | 0.249 | 4.24s | 0.134 | 0.231 | - | 0.211 | 0.319 | 3.63s | **0.129** | **0.222** | 16.19s | 0.140 | 0.237 | 2.61s | 0.132 | 0.229 | 1452.56s |
| | 192 | **0.147** | **0.238** | 1.01s | 0.169 | 0.271 | 0.97s | 0.339 | 0.414 | - | 0.226 | 0.331 | 0.78s | **0.147** | 0.24 | 2.20s | 0.153 | 0.249 | 0.46s | **0.147** | 0.243 | 255.30s |
| | 336 | 0.163 | **0.254** | 2.94G | 0.190 | 0.292 | 66.56G | 0.280 | 0.370 | 2.03T | 0.242 | 0.345 | 1.51G | 0.163 | 0.259 | 195.96G | 0.169 | 0.267 | 47.57M | **0.161** | 0.261 | 250.86G |
| | 720 | 0.199 | **0.285** | 66K | 0.236 | 0.329 | 6.4M | 0.557 | 0.933 | 4.27M | 0.274 | 0.370 | 592K | 0.197 | 0.29 | 6.90M | 0.203 | 0.301 | 18.62K | **0.196** | 0.294 | 34.10M |
| Traffic | 96 | 0.382 | **0.243** | 3.82s | 0.421 | 0.318 | 3.14s | 0.385 | 0.276 | | 0.721 | 0.478 | - | 0.367 | 0.251 | 9.84s | 0.410 | 0.282 | 1.58s | **0.336** | 0.253 | 2652.14s |
| | 192 | 0.397 | **0.255** | 1.02s | 0.455 | 0.340 | 0.63s | 0.394 | 0.281 | | 0.777 | 0.494 | - | 0.385 | 0.259 | 1.69s | 0.423 | 0.287 | 0.29s | **0.346** | 0.257 | 399.65s |
| | 336 | 0.411 | **0.260** | 7.91G | 0.487 | 0.359 | 177G | 0.413 | 0.290 | - | 0.813 | 0.503 | 4.06G | 0.398 | 0.265 | 526.22G | 0.436 | 0.296 | 127.76M | **0.355** | **0.26** | 1.77T |
| | 720 | 0.451 | 0.281 | 66K | 0.555 | 0.394 | 6.4M | 0.452 | 0.312 | | 0.856 | 0.517 | 592K | 0.434 | 0.287 | 6.90M | 0.466 | 0.315 | 18.62K | **0.386** | **0.273** | 88.49M |
| Weather | 96 | **0.146** | **0.186** | 2.89s | 0.159 | 0.211 | 2.26s | 0.151 | 0.200 | 7.17s | 0.166 | 0.236 | 1.60s | 0.152 | 0.199 | 4.01s | 0.176 | 0.237 | 0.49s | 0.166 | 0.222 | 35.62s |
| | 192 | **0.189** | **0.230** | 0.52s | 0.202 | 0.251 | 0.50s | 0.193 | 0.242 | 1.57s | 0.208 | 0.274 | 0.32s | 0.197 | 0.243 | 0.67s | 0.22 | 0.282 | 0.11s | 0.209 | 0.263 | 7.79s |
| | 336 | **0.244** | **0.277** | 0.78G | 0.256 | 0.291 | 5.12G | 0.242 | 0.281 | 532.70G | 0.255 | 0.311 | 0.39G | 0.249 | 0.283 | 51.27G | 0.265 | 0.319 | 12.45M | 0.254 | 0.301 | 6.17G |
| | 720 | **0.313** | **0.326** | 66K | 0.323 | 0.342 | 6.4M | 0.319 | 0.334 | 4.27M | 0.314 | 2.000 | 592K | 0.320 | 0.335 | 6.90M | 0.323 | 0.362 | 18.62K | **0.313** | 0.34 | 3.93M |
| Electricity Price | 96 | **0.486** | **0.424** | 2.54s | 0.677 | 0.549 | 1.63s | 0.621 | 0.531 | 4.52s | 0.703 | 0.552 | 1.31s | 0.635 | 0.537 | 2.74s | 0.572 | 0.480 | 0.38s | 0.585 | 0.480 | 3.28s |
| | 192 | **0.528** | **0.443** | 0.50s | 0.749 | 0.592 | 0.31s | 0.720 | 0.586 | 0.96s | 0.729 | 0.563 | 0.21s | 0.697 | 0.591 | 0.50s | 0.720 | 0.447 | 0.11s | 0.618 | 0.520 | 0.65s |
| | 336 | **0.459** | **0.446** | 8.31G | 0.747 | 0.591 | 1.22G | 0.704 | 0.575 | 12.68G | 0.723 | 0.557 | 11.79M | 0.773 | 0.615 | 4.88G | 0.651 | 0.533 | 1.18M | 0.643 | 0.542 | 0.22G |
| | 720 | **0.495** | **0.467** | 74.86K | 0.797 | 0.648 | 6.4M | 0.675 | 0.566 | 4.27M | 0.696 | 0.545 | 592K | 0.832 | 0.629 | 6.90M | 0.860 | 0.520 | 18.62K | 0.632 | 0.538 | 2.02M |
| Cycle | 96 | **0.136** | **0.221** | 1.09s | 0.182 | 0.278 | 1.05s | 0.169 | 0.265 | 2.79s | 0.441 | 0.464 | 1.01s | 0.160 | 0.248 | 1.60s | 0.174 | 0.254 | 0.22s | 0.150 | 0.236 | 1.70s |
| | 192 | **0.145** | **0.230** | 0.19s | 0.212 | 0.302 | 0.18s | 0.203 | 0.303 | 0.58s | 0.469 | 0.475 | 0.12s | 0.167 | 0.254 | 0.29s | 0.177 | 0.254 | 0.05s | 0.158 | 0.240 | 0.21s |
| | 336 | **0.152** | **0.235** | 0.84G | 0.232 | 0.318 | 1.02G | 0.146 | 0.240 | 6.34G | 0.469 | 0.472 | 18M | 0.179 | 0.263 | 2.44G | 0.184 | 0.256 | 0.59M | 0.167 | 0.246 | 0.10G |
| | 720 | **0.159** | **0.236** | 74K | 0.258 | 0.337 | 6.4M | 0.175 | 0.263 | 4.27M | 0.482 | 0.480 | 592K | 0.214 | 0.284 | 6.90M | 0.192 | 0.267 | 18.62K | 0.181 | 0.256 | 1.92M |
| Count | | **51**/12 | | | **0**/0 | | | **0**/1 | | | **0**/0 | | | **11**/23 | | | **0**/11 | | | **15**/29 | | |

For the *Efficiency* columns, values are exhibited, in order, as training time (seconds per epoch), inference time (seconds per inference), MACs and the number of model parameters ("-" indicates insufficient GPU memory under the current experiment environment). These values correspond to the forecast sequence length of 96 and exhibit quantitative similarity for other lengths. The best and second-best counts are in the last row.

impact of the dual encoder architecture for future covariates on prediction performance, this study employs two additional datasets with future features, namely Cycle and Electri-Price. The former dataset[5] records the bicycle counts per hour passing through the Seattle Fremont Bridge from October 2012 to March 2015, with future features mainly consisting of weather forecast information such as temperature, humidity, visibility, and wind strength. The latter dataset, proprietary business data[6], documents the real-time electricity market prices in a Chinese province every 15 minutes over one year from 2021 to 2022. Its future characteristics incorporate weather forecasts and features such as renewable energy generation and electricity load forecasts provided by the grid dispatching center. Table IV presents detailed features of the two datasets.

*2) **Data & Model Configuration:*** By default, we use the following data and model configurations: Input Sequence length $T = 720$ , Patch length $pl = 48$, Batch size $b = 256$ , Forecast sequence length $L = \{96, 192, 336, 720\}$, Hidden feature size $hd = 512$, and Dropout = 0.5. LiPFormer utilizes the AdamW optimizer[56]. Training is performed in a distributed fashion with 1 GPU, 1 CPU and 32 GB of memory. For ETT datasets, we use a lower hardware and model configuration with high dropout to avoid overfitting, as the dataset is relatively small. Training is performed with 10 epochs and used early stopping with 3 patients. We choose the final model based on the best validation score. For patching, we use a patch length $pl = 48$ that divides the input length $T$ equally and without overlap. Every experiment is executed with the same random seed and the mean scores are reported. We use $\text{MSE} = \frac{1}{T}\sum_{i=0}^{T}(\hat{X}_i - X_i)^2$ and $\text{MAE} = \frac{1}{T}\sum_{i=0}^{T}|\hat{X}_i - X_i|$ to evaluate accuracy.

In addition, we also examine the training/inference time, the number of parameters and MACs (Multiply–Accumulate Operations) of the models. We conduct the experiments using the same input and output lengths of 96. A batch size of 32 is utilized for all datasets. Given the substantial number of channels and data volume present in both the Traffic and Electricity datasets, we employ a batch size of 8 for experiments on the two datasets, in order to better adapt to the actual GPU memory size.

*3) **Baselines:*** We evaluated the following SOTA benchmarks: iTransformer [46], TimeMixer [51], FGNN [52], PatchTST [31], Dlinear [30], and TiDE [48].

[5]https://github.com/nkullman/CSE512_A3

[6]https://github.com/wangmeng-xpu/LiPFormer

### B. Non-covariate Prediction

*1)* ***Data Preprocess:*** Since these public datasets do not have future covariates, we can only construct features as the hour of the day, day of the week, day of the month, and month of the year as inputs for future covariates in a similar way to the time encoding in Informer.

*2)* ***Accuracy Improvements:*** In Table III, we compare the accuracy of LiPFormer to the SOTA benchmarks. Since similar relative patterns can be observed in both MSE and MAE, we use the MAE metrics to interpret all results in this paper. LiPFormer significantly outperforms the existing benchmarks (DLinear: 10.4%, iTransformer: 18%, TimeMixer: 21%, FGNN: 62%). PatchTST and TiDE are the two strongest baseline models, while LiPFormer outperforms them by a narrow margin of 8.3% and 5.4%.

LiPFormer demonstrates strong multivariate long-term time series prediction performance, achieving top-two rankings in 64 out of 72 metrics across various scenarios, with 51 instances of first place. This convincingly surpasses other baselines, including the SOTA Transformer-based model, PatchTST. LiPFormer excels particularly on the ETT and Weather datasets, consistently outperforming SOTA on nearly all metrics. On larger datasets like electricity and transportation, where channel counts exceed 300 and 800, respectively, LiPFormer's performance marginally declines. This can potentially be attributed to its relatively smaller model capacity. Nonetheless, even in these challenging scenarios, LiPFormer maintains competitiveness or demonstrates superiority relative to alternative models. We believe that this is due to the fact that Cross-Patch attention and Inter-Patch attention better capture the continuity and global nature of the time series. At the same time, since the extra component of transformer does not significantly improve the accuracy, it leads us to a win-win situation in terms of accuracy and efficiency. In particular, for more volatile datasets (ETTh1, Weather), we prefer to use LiPFormer as a predictive model.

### C. Forecasting via Future Covariate

In this part, we investigate the effectiveness of the Covariate Encoder framework using two datasets containing future weak labels, where the Electricity Price and Cycle datasets respectively include 61 and 22 covariates. The Electricity-Price dataset records the variations of electricity spot market from Shanxi province in China, and the Cycle dataset corresponds to the bicycle-riding scenario in Seattle, U.S. Their future weak labels are detailed in Table IV.

As shown in the last two datasets of Table III, the proposed LiPFormer surpasses all other comparative models in prediction accuracy, indicating that leveraging the weak supervision of future covariates through pre-training can effectively guide the predictions. It is worth noting that the TiDE model also took into account external factors for prediction, enabling it to outperform other counterparts, except for LiPFormer, on these two datasets. This further highlights the utility of weak labels, and also validates our claim that future value changes are highly correlated with apriori contexts. Compared to TiDE, LiPFormer's average MSE using future covariates is reduced by 20.6% and 18.6% on the two datasets. This is attributed to our model's ability to characterize the alignment of time series data achieved during the pre-training of covariates, demonstrating the superiority of the dual encoder strategy.

TABLE IV
DETAILS OF THE TWO DATASETS WITH FUTURE FEATURES.

| Datasets | Future Covariates | # of Fields | Data Field Type |
|---|---|---|---|
| Electricity Price | Unified Load Forecast (MW) | 1 | numerical |
| | Outgoing forecast (MW) | 1 | numerical |
| | Sum of wind and light projections | 1 | numerical |
| | Wind power projections | 1 | numerical |
| | Photovoltaic Forecast | 1 | numerical |
| | Max and Min temperature at a location | 22 | numerical |
| | Wind rating at a location | 11 | numerical |
| | The direction of the wind at a location | 11 | numerical |
| | Weather conditions at a location | 11 | categorical |
| | holiday | 1 | categorical |
| Cycle | Max,Min,Mean temperature | 3 | numerical |
| | Max,Min,Mean dewpointF | 3 | numerical |
| | Max,Min,Mean humidity | 3 | numerical |
| | Max,Min,Mean sea level pressure in | 3 | numerical |
| | Max,Min,Mean visibility miles | 3 | numerical |
| | Max,Mean wind speed MPH and wind direct degree | 3 | numerical |
| | Max gust speed MPH | 1 | numerical |
| | precipitation in | 1 | numerical |
| | cloud cover | 1 | numerical |
| | weekend | 1 | categorical |

Furthermore, for datasets that do not contain explicit weak labels, augmenting the weak data with implicit temporal features can still enrich data understanding and significantly enhance prediction performance. As illustrated in Table VI, we examine forecast results on four datasets (with prediction lengths of 96) that lack explicit future covariates. By comparing the prediction outcomes with and without leveraging implicit temporal features for weak data enriching, it is evident that adopting the pre-trained model yields substantially positive results.

### D. Univariate Forecasting

Table V showcases univariate long-term time series forecast outcomes for LiPFormer and competing baselines across the entire ETT benchmark datasets. Notably, among the 32 evaluation metrics, LiPFormer ranks within the top-two in 26 metrics and achieves the best performance in 16 of them. This outstanding performance highlights the superiority of LiPFormer over other methods. These findings further demonstrate the robustness of LiPFormer and reconfirm the capability of attention mechanism in time series forecasting, irrespective of multivariate or univariate contexts.

### E. More Analysis

*1)* ***Model Efficiency:*** We present a thorough comparison of our model's parameters count, training speed, inference time and memory consumption against existing time series forecasting models, using official configurations and identical batch sizes. Tables III illustrate the efficiency comparisons under multivariate datasets.

Both LiPFormer and PatchTST significantly outperform the Vanilla Transformer-based models, due to the patching technique. Compared to the state-of-the-art patch-wise PatchTST, LiPFormer still reduces training and inference time by 57%

TABLE V
UNIVARIATE LONG-TERM TIME SERIES FORECASTING RESULTS WITH LIPFORMER. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD, AND THE SECOND-BEST RESULTS ARE UNDERLINED.

| Models | | LiPFormer | | iTransformer | | TimeMixer | | FGNN | | PatchTST | | DLinear | | TiDE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 96 | 0.057 | **0.18** | 0.059 | 0.189 | 0.103 | 0.252 | 0.106 | 0.256 | 0.059 | 0.189 | **0.056** | **0.18** | 0.067 | 0.204 |
| | 192 | 0.070 | 0.209 | **0.068** | **0.204** | 0.113 | 0.270 | 0.112 | 0.265 | 0.074 | 0.215 | 0.071 | **0.204** | 0.081 | 0.225 |
| | 336 | **0.075** | **0.216** | 0.078 | 0.221 | 0.102 | 0.257 | 0.126 | 0.284 | 0.076 | 0.22 | 0.098 | 0.244 | 0.090 | 0.240 |
| | 720 | 0.102 | 0.252 | **0.080** | **0.230** | 0.122 | 0.288 | 0.252 | 0.423 | 0.087 | 0.236 | 0.189 | 0.359 | 0.104 | 0.254 |
| ETTh2 | 96 | 0.134 | 0.282 | 0.147 | 0.307 | 0.158 | 0.316 | 0.224 | 0.384 | **0.131** | 0.284 | **0.131** | **0.279** | 0.164 | 0.320 |
| | 192 | **0.157** | **0.314** | 0.155 | 0.320 | 0.205 | 0.357 | 0.248 | 0.407 | 0.171 | 0.329 | 0.176 | 0.329 | 0.186 | 0.345 |
| | 336 | 0.169 | 0.330 | **0.159** | **0.326** | 0.200 | 0.359 | 0.303 | 0.450 | 0.171 | 0.336 | 0.209 | 0.367 | 0.194 | 0.357 |
| | 720 | 0.217 | 0.370 | **0.173** | **0.342** | 0.207 | 0.362 | 0.320 | 0.458 | 0.223 | 0.38 | 0.276 | 0.426 | 0.234 | 0.388 |
| ETTm1 | 96 | 0.027 | **0.123** | 0.036 | 0.146 | 0.037 | 0.146 | 0.075 | 0.218 | **0.026** | **0.123** | 0.028 | **0.123** | 0.028 | 0.129 |
| | 192 | **0.039** | **0.151** | 0.060 | 0.185 | 0.053 | 0.180 | 0.192 | 0.376 | 0.04 | **0.151** | 0.045 | 0.156 | 0.04 | 0.154 |
| | 336 | **0.051** | 0.175 | 0.072 | 0.204 | 0.069 | 0.202 | 0.146 | 0.313 | 0.053 | **0.174** | 0.061 | 0.182 | 0.053 | 0.177 |
| | 720 | 0.086 | 0.226 | 0.081 | 0.220 | 0.089 | 0.235 | 0.087 | 0.230 | 0.073 | **0.206** | 0.08 | 0.21 | **0.072** | **0.206** |
| ETTm2 | 96 | **0.063** | 0.184 | 0.079 | 0.217 | 0.087 | 0.223 | 0.094 | 0.240 | 0.065 | 0.187 | **0.063** | **0.183** | 0.071 | 0.203 |
| | 192 | **0.087** | **0.220** | 0.124 | 0.274 | 0.135 | 0.283 | 0.122 | 0.276 | 0.093 | 0.231 | 0.092 | 0.227 | 0.098 | 0.240 |
| | 336 | **0.116** | **0.260** | 0.183 | 0.337 | 0.157 | 0.311 | 0.179 | 0.336 | 0.121 | 0.266 | 0.119 | 0.261 | 0.126 | 0.273 |
| | 720 | **0.169** | **0.314** | 0.188 | 0.345 | 0.194 | 0.351 | 0.205 | 0.354 | 0.172 | 0.322 | 0.175 | 0.32 | 0.180 | 0.332 |
| Count | | **16**/10 | | **8**/3 | | **0**/4 | | 0 | | **8**/14 | | **8**/8 | | **2**/3 | |

TABLE VI
COMPARISON OF FORECAST RESULTS WITH AND WITHOUT IMPLICIT TEMPORAL FEATURES.

| Datasets | Without Pre-train | | With Pre-train | |
|---|---|---|---|---|
| | MSE | MAE | MSE | MAE |
| ETTh1 | 0.368 | 0.386 | **0.359** | **0.379** |
| ETTh2 | 0.272 | 0.331 | **0.265** | **0.327** |
| ETTm1 | 0.310 | 0.351 | **0.296** | **0.338** |
| ETTm2 | 0.165 | 0.252 | **0.160** | **0.244** |

TABLE VII
COMPARISON OF INFERENCE TIME (SECONDS PER INFERENCE) IN A CPU-ONLY DEVICE VARYING INPUT SEQUENCE LENGTHS.

| Input Lengths | ETTh1 | | | | Weather | | | |
|---|---|---|---|---|---|---|---|---|
| | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 |
| Transformer | 1.47 | 2.86 | 5.82 | - | 1.84 | 3.88 | 6.08 | - |
| LiPFormer | 0.55 | 0.60 | 0.62 | 0.74 | 0.87 | 1.35 | 3.77 | 4.57 |

and 51% on average, while dramatically decreasing the amounts of computation and model parameters. This margin is especially evident in datasets Traffic and ETTh2, where LiPFormer requires only 2% MACs compared to PatchTST. The reason for the significant increase in efficiency is twofold. On one hand, we remove computationally intensive components, like LN and FFNs, of the transformer that do not substantially contribute to accuracy. On the other hand, the novel Inter-Patch and Cross-Patch attention mechanisms have not incurred a prohibitive increase in model complexity. Benefitting from its lightweight design, LiPFormer has even better efficiency values than the linear model TiDE. Although DLinear slightly leads in efficiency with its simple linear structure, the inferior prediction accuracy greatly downgrades its availability.

As in Table VII, we evaluate the compatibility of LiPFormer in resource-constrained environments. To simulate the numerous early-stage or low-budget edge devices already in operation, which typically lack GPUs, we configure a CPU-only edge device with 16GB RAM, 6 cores, and 12 threads. We deploy the trained LiPFormer and Transformer models on this device and conduct prediction inference on the ETTh1 and Weather datasets. Compared to Transformer, LiPFormer exhibit a significant drop in inference time, achieving nearly a tenfold increase in efficiency for an input length of 336 in ETTh1. For the Weather dataset with more channels (21), LiPFormer's efficiency improvement is reduced but still maintains nearly a two-fold enhancement. Notably, Transformer exceeds the memory limit for an input length of 720 on both datasets, whereas LiPFormer exhibits its lightweight superiority.

*2) **Impact of Patch Size:*** To verify the impact of patch length $pl$, we perform experiments with four patch lengths. Results are shown on Table VIII. Fixed patch length does not lead to its performance loss on different datasets, which proves that the mixing operation we adopted effectively improves the generalization performance. Specifically, We recommend using $pl = 48$ as a more suitable choice for most datasets.

*3) **Impact of Input Length:*** Since longer input sequences indicate that more historical information is available, models with strong ability to capture long-term time dependencies should perform better when the input length increases. As shown in Table IX, a comprehensive evaluation on the MSE metric is conducted using the ETT and Weather datasets with varying input lengths (96, 192, 336, 720). Overall, the performance of LiPFormer improves as input length increases. LiPFormer significantly outperforms SOTA benchmarks under most input lengths (15/20), indicating that our model can effectively extract useful information from histories and capture long-term dependencies.

*4) **Ablation of Lightweight Architecture:*** Time series data does not have a large number of data entries and the corresponding inductive bias as in the NLP domain, the Normalization approach used by the Transformer class of models and their deeper layers may capture greater noise in time series prediction, In order to validate the effectiveness of our deletion of the layer norm versus the feed forward layer, the we constructed four model model variants without removing these parts:

TABLE VIII
THE IMPACT OF PATCH SIZE. $pl = \{6, 12, 24, 48\}$ FOR ALL DATASETS.

| Datasets Forecasting Length | Metric | ETTh1 | | | | ETTh2 | | | | ETTm1 | | | | ETTm2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 |
| pl = 6 | MSE | 0.372 | 0.405 | **0.427** | **0.439** | 0.276 | **0.333** | 0.365 | 0.402 | 0.482 | **0.440** | 0.422 | 0.417 | 0.575 | 0.375 | 0.358 | **0.369** |
| | MAE | 0.397 | 0.413 | 0.430 | 0.452 | 0.337 | 0.374 | 0.400 | 0.435 | 0.440 | 0.426 | 0.412 | 0.411 | **0.390** | 0.382 | 0.373 | **0.378** |
| pl=12 | MSE | 0.375 | 0.417 | 0.431 | 0.468 | 0.268 | 0.335 | 0.364 | **0.390** | 0.480 | 0.442 | **0.420** | **0.414** | 0.402 | 0.370 | **0.356** | 0.376 |
| | MAE | 0.395 | 0.415 | 0.431 | 0.463 | 0.333 | 0.376 | 0.398 | 0.426 | 0.439 | **0.424** | **0.411** | **0.411** | 0.392 | 0.379 | **0.372** | 0.381 |
| pl = 24 | MSE | 0.369 | 0.411 | 0.423 | 0.450 | 0.274 | 0.336 | 0.366 | 0.391 | **0.479** | 0.446 | 0.426 | 0.507 | 0.403 | **0.370** | 0.358 | 0.370 |
| | MAE | 0.387 | 0.407 | 0.437 | **0.448** | 0.335 | **0.373** | 0.399 | 0.430 | **0.436** | 0.426 | 0.412 | 0.456 | 0.393 | **0.379** | 0.374 | 0.378 |
| pl=48 | MSE | **0.359** | **0.404** | 0.444 | 0.450 | **0.265** | 0.335 | **0.364** | 0.392 | 0.483 | 0.450 | 0.425 | 0.526 | **0.400** | 0.372 | 0.357 | 0.372 |
| | MAE | **0.379** | **0.405** | **0.425** | 0.453 | **0.327** | 0.374 | **0.395** | **0.425** | 0.439 | 0.429 | 0.412 | 0.473 | 0.391 | 0.380 | 0.374 | 0.380 |

TABLE IX
IMPACT OF INPUT SEQUENCE LENGTH.

| Datasets | Input length | **LiPFormer** | PatchTST | DLinear | TiDE | iTransformer | FGNN | TimeMixer |
|---|---|---|---|---|---|---|---|---|
| ETTh1 | 96 | **0.373** | 0.383 | 0.396 | 0.432 | 0.394 | 0.503 | 0.398 |
| | 192 | **0.368** | 0.380 | 0.386 | 0.424 | 0.396 | 0.530 | 0.454 |
| | 336 | 0.383 | **0.382** | 0.375 | 0.410 | 0.397 | 0.553 | 0.393 |
| | 720 | **0.359** | 0.375 | 0.375 | 0.375 | 0.392 | 0.647 | 0.385 |
| ETTh2 | 96 | **0.286** | 0.317 | 0.341 | 0.318 | 0.300 | 0.416 | 0.286 |
| | 192 | **0.287** | 0.311 | 0.323 | 0.311 | 0.302 | 0.409 | 0.294 |
| | 336 | **0.277** | 0.308 | 0.307 | 0.297 | 0.307 | 0.390 | 0.287 |
| | 720 | **0.265** | 0.274 | 0.289 | 0.270 | 0.303 | 0.479 | 0.296 |
| ETTm1 | 96 | 0.326 | 0.335 | 0.345 | 0.365 | 0.341 | 0.400 | **0.324** |
| | 192 | 0.306 | 0.310 | 0.306 | 0.319 | 0.303 | 0.368 | **0.302** |
| | 336 | **0.285** | 0.293 | 0.300 | 0.310 | 0.304 | 0.378 | 0.298 |
| | 720 | 0.296 | **0.290** | 0.299 | 0.306 | 0.318 | 0.403 | 0.305 |
| ETTm2 | 96 | **0.175** | 0.182 | 0.193 | 0.188 | 0.183 | 0.230 | 0.176 |
| | 192 | **0.167** | 0.175 | 0.179 | 0.176 | 0.185 | 0.231 | 0.171 |
| | 336 | **0.161** | 0.172 | 0.169 | 0.170 | 0.173 | 0.223 | 0.179 |
| | 720 | **0.160** | 0.165 | 0.161 | 0.167 | 0.180 | 0.225 | 0.181 |
| Weather | 96 | 0.179 | **0.177** | 0.196 | 0.177 | 0.184 | 0.170 | 0.184 |
| | 192 | 0.162 | **0.159** | 0.184 | 0.187 | 0.168 | 0.182 | 0.168 |
| | 336 | 0.154 | **0.15** | 0.174 | 0.172 | 0.158 | 0.173 | 0.158 |
| | 720 | **0.146** | 0.152 | 0.176 | 0.166 | 0.159 | 0.166 | 0.151 |

- **LiPFormer + LN:** LN after adding the Attention mechanism to the original model
- **LiPFormer + FFNs:** FeedForward network added to the Attention mechanism and then exported
- **LiPFormer + LN + FFNs:** Add all the above components to the LiPFormer model.

The comparison experiments are shown in Table X. The use of either FFNs or LN caused a decrease in the performance of LiPFormer, where the average MSE and MAE of LiPFormer increased by 15 % and 9%, respectively, after the addition of FFNs, while the addition of LN also had an impact on the performance of LiPFormer, with an increase in the MSE by 3.5% and in the MAE by 4%. In particular, after adding both FFNs and LN, LiPFormer's MSE improves by 24% on average and MAE improves by 13% on average, and this severe performance degradation in the ETTh1 dataset results in a 45% improvement in MSE and a 23% improvement in MAE. This experimental result is consistent with our assumption that FFNs and LN are not effective for time series modeling. In fact, removing these components tends to contribute to the accuracy improvement of the time series prediction.

*5) **Ablation of Patch-wise Attention:*** To verify the effectiveness of Cross- and Inter-Patch attention mechanisms, we constructed three model variants for comparison:

- **Without Cross-Patch attn.:** We remove the Cross-Patch attention and use a linear layer instead.

TABLE X
ABLATION STUDY OF ARCHITECTURE LIGHTWEIGHT, INCLUDING FEED FORWARD NETWORKS AND LAYER NORMALIZATION.

| Datasets | Metric | ETTh1 | | | | ETTm2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 |
| LiPFormer +FFNs | MSE | 0.389 | 0.439 | 0.514 | 0.523 | 0.184 | 0.217 | 0.282 | 0.521 |
| | MAE | 0.408 | 0.441 | 0.478 | 0.497 | 0.271 | 0.285 | 0.330 | 0.466 |
| LiPFormer +LN | MSE | 0.392 | 0.437 | 0.448 | 0.508 | 0.158 | 0.218 | 0.266 | 0.350 |
| | MAE | 0.412 | 0.435 | 0.453 | 0.498 | 0.243 | 0.286 | 0.317 | 0.376 |
| LiPFormer +FFNs+LN | MSE | 0.754 | 0.505 | 0.614 | 0.498 | 0.164 | 0.226 | 0.284 | 0.360 |
| | MAE | 0.557 | 0.465 | 0.524 | 0.497 | 0.252 | 0.292 | 0.332 | 0.381 |
| LiPFormer | MSE | 0.359 | 0.404 | 0.444 | 0.450 | 0.160 | 0.217 | 0.273 | 0.348 |
| | MAE | 0.379 | 0.405 | 0.425 | 0.453 | 0.244 | 0.285 | 0.322 | 0.372 |

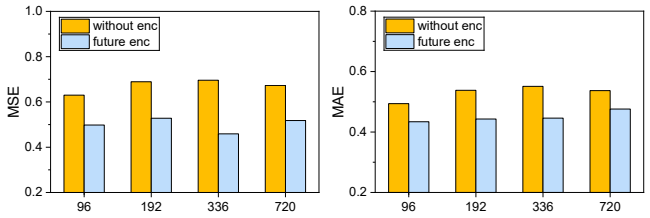


Fig. 6. The impact of incorporating or excluding the future Covariate Encoder (enc) on the MSE and MAE in the Electricity Price dataset.

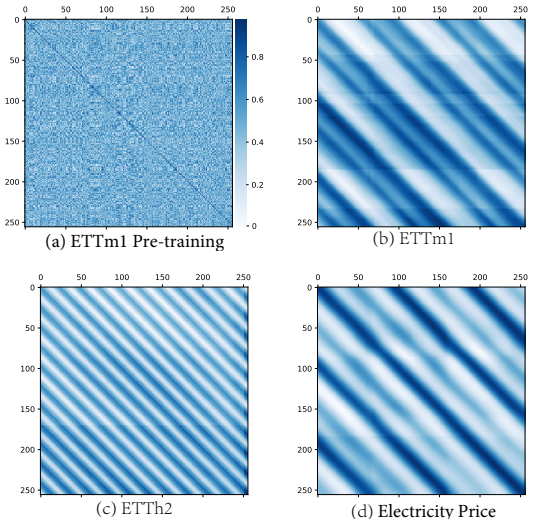


Fig. 7. Visualization of the logits matrices for Weakly Supervised Architecture: (a) Logits matrix pre-trained on ETTm1; (b), (c), (d) Logits matrices on the validation sets of ETTm1, ETTh2 and Electricity-Price, where $b = 256$.

TABLE XI
ABLATION STUDY OF PATCH-WISE ATTENTIONS, INCLUDING CROSS-PATCH ATTENTION AND INTER-PATCH ATTENTION.

| Datasets | Metric | ETTh1 | | | | ETTh2 | | | | ETTm1 | | | | ETTm2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 | 96 | 192 | 336 | 720 |
| Without Cross-Patch attn. | MSE | 0.366 | **0.401** | **0.441** | 0.461 | 0.270 | 0.332 | 0.370 | 0.394 | 0.327 | 0.348 | 0.386 | 0.434 | 0.167 | 0.225 | 0.280 | 0.364 |
| | MAE | 0.380 | 0.424 | 0.450 | 0.461 | 0.329 | 0.371 | 0.396 | 0.427 | 0.365 | 0.378 | 0.398 | 0.425 | 0.255 | 0.295 | 0.333 | 0.386 |
| Without Inter-Patch attn. | MSE | 0.364 | 0.413 | 0.458 | 0.472 | 0.271 | 0.335 | 0.383 | 0.586 | 0.309 | 0.342 | 0.370 | 0.430 | 0.164 | 0.220 | 0.275 | 0.353 |
| | MAE | 0.380 | 0.408 | 0.434 | 0.455 | 0.329 | 0.371 | 0.405 | 0.477 | 0.350 | 0.367 | 0.384 | 0.417 | 0.251 | 0.289 | 0.324 | 0.382 |
| Neither | MSE | 0.379 | 0.417 | 0.456 | 0.466 | 0.273 | 0.379 | 0.397 | 0.422 | 0.314 | 0.340 | 0.377 | 0.423 | 0.168 | 0.224 | 0.276 | 0.356 |
| | MAE | 0.392 | 0.412 | 0.432 | 0.453 | 0.332 | 0.388 | 0.427 | 0.441 | 0.356 | 0.371 | 0.393 | 0.419 | 0.255 | 0.293 | 0.330 | 0.379 |
| LiPFormer | MSE | **0.359** | 0.404 | 0.444 | **0.450** | **0.265** | **0.330** | **0.364** | **0.392** | **0.296** | **0.336** | **0.365** | **0.408** | **0.160** | **0.217** | **0.273** | **0.348** |
| | MAE | **0.379** | **0.405** | **0.425** | **0.453** | **0.327** | **0.369** | **0.395** | **0.425** | **0.338** | **0.360** | **0.379** | **0.413** | **0.244** | **0.285** | **0.322** | **0.372** |

TABLE XII
COVARIATE ENCODER TO OTHER MODELS.

| Models | | Covariate Encoder | | Without Covariate Encoder | |
|---|---|---|---|---|---|
| | | MSE | MAE | MSE | MAE |
| Informer | 96 | **0.699** | **0.525** | 0.736 | 0.569 |
| | 192 | **0.692** | **0.569** | 0.706 | 0.576 |
| Transformer | 96 | **0.691** | **0.530** | 0.723 | 0.571 |
| | 192 | **0.691** | **0.533** | 0.704 | 0.549 |
| Autoformer | 96 | **0.699** | **0.583** | 0.739 | 0.592 |
| | 192 | **0.685** | **0.536** | 0.744 | 0.580 |

- **Without Inter-Patch attn.:** We replace the Inter-Patch attention with a linear layer.
- **Neither:** Only traditional patching technique is used.

Table XI reports the impact of altering the patching mechanisms. Compared to using only the classical patching method, employing Cross-Patch alone outperforms using Inter-Patch alone. The former exceeds Neither in 72% of the metrics, while the latter shows only comparable results. This may be attributed to Cross-Patch's ability to mitigate fixed patch size limitation, which possibly makes Inter-Patch less effective on ETTm than on ETTh. It should be noted that Cross-Patch and Inter-Patch are designed to be complementary in their effects, serving to perceive global trends and local correlations, respectively. While using either mechanism alone results in only limited improvement, their combined use consistently demonstrates robust accuracy enhancements across all datasets, with MSE and MAE decreasing by 5% and 3%, respectively. This validates the essentiality of combining both mechanisms and their collective effectiveness.

*6) **Ablation of Covariate Encoder:*** We verify the effect of the future covariate encoder in a simple way: as shown in the Figure 6, by removing the covariate encoder, in the dataset with future covariates, the MSE of the LiPFormer decreases by 34%, and the MAE decreases by 17%, but it still outperforms the SOTA model in a number of cases, which is side by side a proof of the validity of our underlying predictor.

We further integrate the Covariate Encoder seamlessly into various models, including Transformer, Informer, and Autoformer. To substantiate the efficacy of this encoder, we conduct experiments on the Electricity-Price dataset. The experimental results reported in Table XII reveal that all the transformer-based models incorporating the Covariate Encoder outperform their original versions, achieving an average reduction of 4% in MSE and 5% in MAE. These improvements confirm both the validity of our proposed inductive bias about weak labels and the effectiveness of the Dual Encoder architecture.

*7) **Visiualization:*** Figure 7 visualizes the logits matrices across various datasets, revealing that our weakly supervised learning approach aligns with the latent vectors between predictions (X-axis) and future covariates (Y-axis). The $b$ diagonal values in Figure 7(a) highlight how contrastive learning optimizes similarity for true values. Given that validation sets are unshuffled, Figures 7(b) and (c) display periodic correlations in the logits matrices corresponding to the actual periods (ETTm1=96, ETTh2=24). For datasets featuring explicit covariates, as shown in Figure 7(d), we observe clear periodicity alongside irregularities at the edges of "stripes", including less pronounced "blurred stripes" within these periods. These patterns support the role of explicit weak labels in guiding predictions, consistent with our inductive bias that future covariates correlate with time series models. Due to space constraints, similar results from other datasets are not included.

## V. CONCLUSION

This paper proposes a Lightweight Patch-wise Transformer with weak label enriching (LiPFormer) for time series forecasting. To simplify the Transformer backbone, it integrates a novel Cross-Patch mechanism into existing patching attention and devises a linear transformation-based attention to eliminate Positional Encoding and two heavy components, Layer Normalization and Feed Forward Networks. A weak label enriching architecture is presented to leverage valuable context information for modeling multimodel future covariates via a dual encoder contrastive learning framework. These innovative mechanisms collectively make LiPFormer a significantly lightweight model, achieving outstanding prediction performance. Deployment on an CPU-only edge device and transplant trials of the weak label enriching module further demonstrate the scalability and versatility of LiPFormer.


## ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No. 62272369, 62372352), Natural Science Basic Research Program of Shaanxi (No. 2023-JC-YB-558, 2024JC-YBMS-473), Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 23JS028), Scientific and Technological Program of Xi'an (No. 24GXFW0016).

## REFERENCES


[1] C. Capistrán, C. Constandse, and M. Ramos-Francia, "Multi-horizon inflation forecasts using disaggregated data," *Economic Modelling*, vol. 27, no. 3, pp. 666–677, 2010.

[2] R. A. Angryk, P. C. Martens, B. Aydin, D. Kempton, S. S. Mahajan, S. Basodi, A. Ahmadzadeh, X. Cai, S. Filali Boubrahimi, S. M. Hamdi *et al.*, "Multivariate time series dataset for space weather data analytics," *Scientific data*, vol. 7, no. 1, p. 227, 2020.

[3] D. C. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, 1991.

[4] S. B. Yang, C. Guo, and B. Yang, "Context-aware path ranking in road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3153–3168, 2022.

[5] X. Ding, L. Chen, Y. Gao, C. S. Jensen, and H. Bao, "Ultraman: A unified platform for big trajectory data management and analytics," *Proceedings of the VLDB Endowment*, vol. 11, no. 7, pp. 787–799, 2018.

[6] L. Chen, Y. Gao, Z. Fang, X. Miao, C. S. Jensen, and C. Guo, "Real-time distributed co-movement pattern detection on streaming trajectories," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1208–1220, 2019.

[7] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system: mining loop detector data," *Transportation research record*, vol. 1748, no. 1, pp. 96–102, 2001.

[8] Z. Fang, L. Pan, L. Chen, Y. Du, and Y. Gao, "Mdtp: A multi-source deep traffic prediction framework over spatio-temporal trajectory data," *VLDB*, vol. 14, no. 8, pp. 1289–1297, 2021.

[9] S. B. Yang, C. Guo, J. Hu, J. Tang, and B. Yang, "Unsupervised path representation learning with curriculum negative sampling," in *IJCAI*, 2021, pp. 3286–3292.

[10] D. Kieu, T. Kieu, P. Han, B. Yang, C. S. Jensen, and B. Le, "Team: Topological evolution-aware framework for traffic forecasting," *Proc. VLDB Endow.*, vol. 18, 2024.

[11] X. Zhang, Y. Lei, H. Chen, L. Zhang, and Y. Zhou, "Multivariate time-series modeling for forecasting sintering temperature in rotary kilns using dcgnet," *IEEE Trans. Ind. Informatics*, vol. 17, no. 7, pp. 4635–4645, 2021.

[12] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, "An edge-ai based forecasting approach for improving smart microgrid efficiency," *IEEE Trans. Ind. Informatics*, vol. 18, no. 11, pp. 7946–7954, 2022.

[13] Y. Cheng, P. Chen, C. Guo, K. Zhao, Q. Wen, B. Yang, and C. S. Jensen, "Weakly guided adaptation for robust time series forecasting," *Proc. VLDB Endow.*, vol. 17, no. 4, pp. 766–779, 2023.

[14] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, "AutoCTS+: Joint neural architecture and hyperparameter search for correlated time series forecasting," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 97:1–97:26, 2023.

[15] X. Wu, X. Wu, B. Yang, L. Zhou, C. Guo, X. Qiu, J. Hu, Z. Sheng, and C. S. Jensen, "AutoCTS++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting," *VLDB J.*, vol. 33, no. 5, pp. 1743–1770, 2024.

[16] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, and B. Yang, "TFB: towards comprehensive and fair benchmarking of time series forecasting methods," *Proc. VLDB Endow.*, vol. 17, no. 9, pp. 2363–2377, 2024.

[17] D. Campos, B. Yang, T. Kieu, M. Zhang, C. Guo, and C. S. Jensen, "Qcore: Data-efficient, on-device continual calibration for quantized models," *Proc. VLDB Endow.*, vol. 17, no. 11, pp. 2708–2721, 2024.

[18] H. Miao, Z. Liu, Y. Zhao, C. Guo, B. Yang, K. Zheng, and C. S. Jensen, "Less is more: Efficient time series dataset condensation via two-fold modal matching," *Proc. VLDB Endow.*, vol. 18, 2024.

[19] K. Zhao, C. Guo, Y. Cheng, P. Han, M. Zhang, and B. Yang, "Multiple time series forecasting with dynamic graph modeling," *Proc. VLDB Endow.*, vol. 17, no. 4, pp. 753–765, 2023.

[20] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] V. Flunkert, D. Salinas, and J. Gasthaus, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *CoRR*, vol. abs/1704.04110, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems, NeurIPS*, 2017, pp. 5998–6008.

[24] P. Chen, Y. Zhang, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, and C. Guo, "Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting," in *ICLR*, 2024.

[25] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *International Conference on Computer Vision, ICCV*, 2021, pp. 9992–10 002.

[27] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Conference on Artificial Intelligence, AAAI*, 2021, pp. 11 106–11 115.

[28] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems, NeurIPS*, 2021, pp. 22 419–22 430.

[29] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning, ICML*, vol. 162, 2022, pp. 27 268–27 286.

[30] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Conference on Artificial Intelligence, AAAI*, 2023, pp. 11 121–11 128.

[31] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations, ICLR*, 2023.

[32] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representations, ICLR*, 2023.

[33] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, and J. Kalagnanam, "Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting," in *ACM SIGKDD*, 2023, pp. 459–469.

[34] R. Mishra and H. P. Gupta, "Designing and training of lightweight neural networks on edge devices using early halting in knowledge distillation," *IEEE Transactions on Mobile Computing*, pp. 1–12, 2023.

[35] X. Shi, S. Zhang, J. Wu, N. Chen, K. Cheng, Y. Liang, and S. Lu, "Adapyramid: Adaptive pyramid for accelerating high-resolution object detection on edge devices," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2023.

[36] T. Carriere, C. Vernay, S. Pitaval, and G. Kariniotakis, "A novel approach for seamless probabilistic photovoltaic power forecasting covering multiple time frames," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2281–2292, 2020.

[37] H. Yu, J. Hu, X. Zhou, C. Guo, B. Yang, and Q. Li, "CGF: A category guidance based pm$_{2.5}$ sequence forecasting training framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10 125–10 139, 2023.

[38] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *ACM SIGKDD*, 2021, pp. 2114–2124.

[39] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[40] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv:1711.11053*, 2017.

[41] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *International Joint Conference on Artificial Intelligence, IJCAI*, C. Sierra, Ed., 2017, pp. 2627–2633.

[42] R. Sen, H. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," in *Advances in Neural Information Processing Systems, NeurIPS*, 2019, pp. 4838–4847.

[43] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations, ICLR*, 2020.

[44] Y. Li, X. Lu, H. Xiong, J. Tang, J. Su, B. Jin, and D. Dou, "Towards long-term time-series forecasting: Feature, pattern, and distribution," in *International Conference on Data Engineering (ICDE)*, 2023, pp. 1611–1624.

[45] R. Cirstea, C. Guo, B. Yang, T. Kieu, X. Dong, and S. Pan, "Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting," in *International Joint Conference on Artificial Intelligence, IJCAI*, 2022, pp. 1994–2001.

[46] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in *International Conference on Learning Representations, ICLR*, 2024.

[47] S. Chen, C. Li, N. Yoder, S. Ö. Arik, and T. Pfister, "Tsmixer: An all-mlp architecture for time series forecasting," *CoRR*, vol. abs/2303.06053, 2023.

[48] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, and R. Yu, "Long-term forecasting with tide: Time-series dense encoder," *CoRR*, vol. abs/2304.08424, 2023.

[49] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, "Lightts: Lightweight time series classification with adaptive ensemble distillation," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 171:1–171:27, 2023.

[50] K. Yan, C. Long, H. Wu, and Z. Wen, "Multi-resolution expansion of analysis in time-frequency domain for time series forecasting," *IEEE Trans. Knowl. Data Eng.*, pp. 1–14, 2024.

[51] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. Zhou, "Timemixer: Decomposable multiscale mixing for time series forecasting," 2024.

[52] K. Yi, Q. Zhang, W. Fan, H. He, L. Hu, P. Wang, N. An, L. Cao, and Z. Niu, "FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective," no. arXiv:2311.06190, 2023.

[53] J. Deng, X. Chen, R. Jiang, D. Yin, Y. Yang, X. Song, and I. W. Tsang, "Disentangling structured components: Towards adaptive, interpretable and scalable time series forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 8, pp. 3783–3800, 2024.

[54] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning, ICML*, vol. 139, 2021, pp. 8748–8763.

[55] S. Lin, W. Lin, W. Wu, S. Wang, and Y. Wang, "Petformer: Long-term time series forecasting via placeholder-enhanced transformer," *arXiv preprint arXiv:2308.04791*, 2023.

[56] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations, ICLR*, 2019.