PKTIME:LIGHTWEIGHT TIME SERIES FORECASTING VIA KRONECKER-STRUCTURED MODELING WITH ADAPTIVE PERIOD SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Lightweight time-series forecasting requires compact models that capture temporal dependencies effectively. We propose PKTime, a lightweight forecasting model combining Kronecker-structured modeling with adaptive period selection. PKTime selects periods from training data and organizes historical observations into intra-period and inter-period dimensions. Each branch uses a diagonal-plus-low-rank mapping for intra-period interactions and a dense mapping for inter-period dependencies, yielding a compact Kronecker-separable predictor. Multiple branches and their fusion weights are jointly learned to capture different periodic structures, with parameters shared across variables. Period selection occurs only at initialization, and forecasting operates entirely in the time domain. Experiments on eight benchmark datasets demonstrate state-of-the-art forecasting performance. Source code is available at `https://anonymous.4open.science/r/PKTime-CE08`.

1 INTRODUCTION

Time-series analysis supports decision-making in applications such as energy scheduling (Zhu et al., 2023; Wang et al., 2022), traffic management (Bai et al., 2020; Li et al., 2018), industrial monitoring (Lai et al., 2023; Xu et al., 2022), and financial analysis (He et al., 2026; Koa et al., 2025). As a central task in time-series analysis, forecasting estimates future values from historical observations. In resource-constrained deployment settings, forecasting models must maintain high predictive accuracy while satisfying limits on storage, computation, and response latency. The overhead of complex architectures can hinder deployment on edge devices and increase the operating costs of large-scale forecasting services. Existing lightweight forecasting methods (Ma et al., 2026; Lin et al., 2024; Ekambaram et al., 2023) demonstrate that compact models with appropriate structural constraints can achieve competitive predictive performance. A central question is therefore how to exploit the structure of time series to retain effective forecasting capabilities with limited parameters and computation.

First, capturing both within-period interactions and cross-period dependencies with a dense forecasting mapping is parameter-intensive. For a series with recurring patterns, relationships among positions within a cycle and dependencies across successive cycles serve different modeling roles (Wu et al., 2023; Dai et al., 2024). For example, the relationship between observations near a peak and those near a trough describes within-period interactions, whereas the relationship between the current cycle and historical cycles describes cross-period dependencies. A dense linear predictor can assign an independent coefficient to every historical–future position pair, but its parameter count grows with the product of the input and output lengths. DLinear (Zeng et al., 2023) simplifies forecasting through series decomposition and linear mappings, while MixLinear (Ma et al., 2026) further reduces parameter overhead through segmented linear transformations. For periodic sequences, an important question is how to combine parameter compression with explicit modeling of these two types of temporal relationship. Kronecker structure (Edalati et al., 2022; Martens et al., 2018) offers a compact parameterization by constructing the overall forecasting mapping from the Kronecker product of two smaller matrices, rather than learning all mapping coefficients independently. As illustrated in Fig. 1(a), the challenge is to organize the history according to its periodic structure and assign the two factors to within-period and cross-period mappings, with appropriate parameter constraints that preserve their modeling capabilities while keeping the operator compact.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

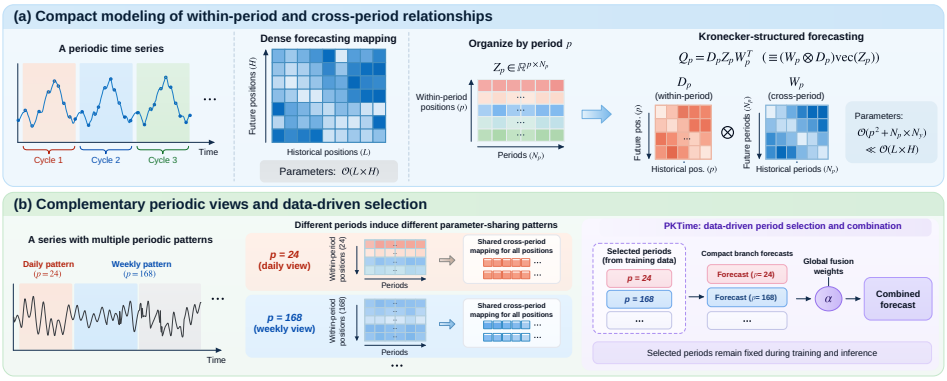

Figure 1: Motivation for PKTime. (a) Compact within- and cross-period modeling via Kronecker structure. (b) Data-selected periodic views combined through learned global fusion. Periods remain fixed after selection.

Second, a compact single-period mapping imposes a fixed parameter-sharing pattern, while additional periodic views increase model overhead. When a predictor uses separate factors for within-period and cross-period relationships, the same cross-period mapping is shared across all within-period positions. Consequently, these positions cannot independently adjust their dependence on historical periods, even when the cross-period matrix itself is fully trainable. For example, positions near peaks and troughs may require different weightings of historical cycles, which are constrained by a shared cross-period mapping. This restriction arises from the mapping structure, rather than from a loss of historical information under a single periodic partition. When multiple recurring patterns coexist (De Livera et al., 2011), organizing the same history according to different periods introduces alternative parameter-sharing patterns that may complement one another. SparseTSF (Lin et al., 2024) uses a specified period for cross-period forecasting, whereas TimesNet (Wu et al., 2023) extracts and aggregates features from multiple periodic views. For lightweight forecasting, however, relying on one predefined period may restrict the available forecasting structures, while indiscriminately adding period branches increases model size. As illustrated in Fig. 1(b), the challenge is therefore to determine which periods and how many branches to use from the training data, and to combine their forecasting mappings while keeping each branch compact.

To address these challenges, we propose PKTime, a lightweight time-series forecasting model combining **P**eriod-aligned **K**ronecker modeling with data-driven period selection and forecast combination. To address the first challenge, PKTime organizes the historical sequence according to a selected period p, yielding two axes corresponding to within-period positions and successive periods. A within-period mapping D_p and a cross-period mapping W_p act along these axes, respectively. For the folded history Z_p, the forecast matrix is computed as $Q_p = D_p Z_p W_p^\top$, which is equivalent to applying the operator $W_p \otimes D_p$ under column-wise vectorization, where $\otimes$ denotes the Kronecker product. This formulation unifies parameter compression and the modeling of both temporal relationships within a single operator, without explicitly constructing the full forecasting matrix. We further parameterize D_p as a diagonal component plus a low-rank correction, while keeping W_p fully trainable, thereby imposing different parameter constraints along the two temporal axes.

To address the second challenge, PKTime determines the period set and branch count from periodicity evidence in the training data and constructs a compact Kronecker forecasting operator for each selected period. The selection procedure can retain a single period or introduce additional branches when supported by the selection criteria. Each branch processes a different period-based arrangement of the same complete history. Branch parameters and global fusion weights are jointly learned under the forecasting objective, allowing the predictor to combine different period-induced parameter-sharing patterns. Within each branch, parameters are shared across variables. Period selection is performed only once, during the initialization stage of model fitting, and the selected periods remain fixed throughout subsequent training and inference. Fusion weights are global learnable parameters rather than input-dependent gates, and all subsequent forecasting operations take place in the time domain.

Our main contributions are summarized as follows:

1. **Period-aligned Kronecker forecasting.** We introduce a compact Kronecker parameterization of the historical-to-future mapping, using two factors to separately model within-period interactions and cross-period dependencies. A diagonal-plus-low-rank within-period factor and a fully trainable cross-period factor impose distinct parameter constraints along the two temporal axes, unifying parameter compression and temporal modeling within a single forecasting operator.
2. **Data-driven period selection and multi-period forecast combination.** We determine the period set and branch count from periodicity evidence in the training data and jointly learn the corresponding compact forecasting operators and their global fusion weights. Different branches organize the same history according to different periods, allowing the predictor to combine complementary parameter-sharing patterns rather than relying solely on a predefined periodic partition.
3. **Empirical evaluation of predictive performance and computational efficiency.** Experiments on eight benchmark datasets demonstrate competitive forecasting accuracy with a compact parameter budget. We evaluate efficiency through parameter counts, computational costs, and inference latency, and conduct ablation studies to assess the contributions of within-period interactions, cross-period parameterization, and period diversity.

2 Related Work

2.1 Lightweight Time-Series Forecasting

Lightweight forecasting models balance forecasting accuracy with computational and parameter efficiency. DLinear (Zeng et al., 2023) applies separate linear projections to trend and seasonal components, providing a simple baseline whose parameter count scales with input and prediction lengths. SparseTSF (Lin et al., 2024) reduces model size through shared cross-period prediction over phase-aligned subsequences, making period selection an important design choice. FITS (Xu et al., 2024) uses complex-valued frequency interpolation and low-pass filtering to learn compact spectral mappings. MixLinear (Ma et al., 2026) combines factorized temporal mappings with a low-dimensional frequency branch for further parameter reduction. These methods exploit decomposition, periodic sparsity, and compact representations. Our work instead examines how to allocate structural constraints between intraperiod and interperiod mappings within a multi-period linear framework.

2.2 Kronecker Products and Separable Linear Mappings.

Kronecker-structured parameterizations have been widely explored as a means of representing large operators through smaller factors. In neural network optimization, Kronecker-factored approximations have been used to obtain tractable approximations of large curvature matrices (Martens & Grosse, 2015; Martens et al., 2018). Kronecker decompositions have also been employed for parameter-efficient neural networks, including convolutional architectures (Hameed et al., 2022) and large language models (Edalati et al., 2022; Tahaei et al., 2022). These studies primarily use Kronecker structure for optimization or generic model compression. In contrast, PKTime assigns explicit temporal semantics to the two factors, associating them with intra-period and inter-period dependencies.

3 Preliminaries

Definition 1 (Time Series)**.** A time series is a temporally ordered sequence of observations, denoted by $\{\mathbf{x}_t\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^C$ represents the observation at time index t, T denotes the sequence length, and C denotes the number of observed variables. The time series is *univariate* if $C = 1$ and *multivariate* if $C > 1$. For convenience, we define $\mathbf{X}_{1:T} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t]^T \in \mathbb{R}^{T\times C}$, in which row line corresponds to a timestamp and a column corresponds to a variable.

Problem Formulation (Time Series Forecasting)**.** Given historical observation time series of length L,$\mathbf{X}_t = [\mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_t]^T \in \mathbb{R}^{L\times C}$ at forecast origin t and corresponding target sequence over the next H time stamps $\mathbf{Y}_t = [\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+H}]^\top \in \mathbb{R}^{H\times C}$. Time-series forecasting aims to learn a

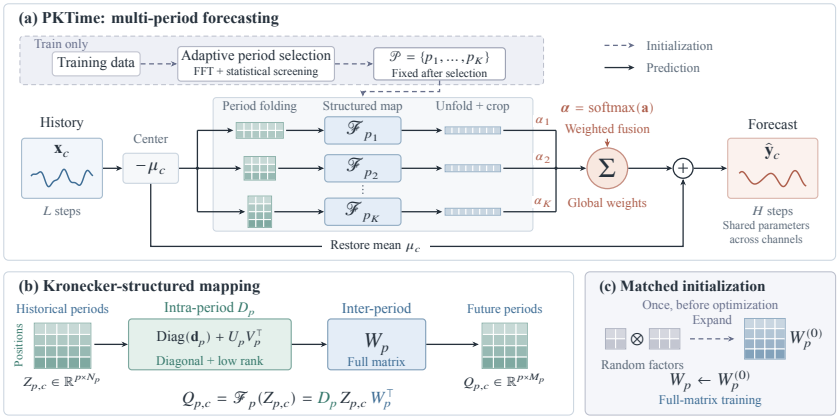

Figure 2: Overview of PKTime. (a) Periods are selected from training data and remain fixed; branch forecasts are combined using learned global weights. (b) Each branch computes $Q_{p,c} = D_p Z_{p,c} W_p^\top$, with a diagonal-plus-low-rank D_p and a fully trainable W_p. (c) Matched initialization expands a randomly initialized Kronecker-factorized inter-period operator into an equivalent full matrix before training.

parameterized mapping

$$f_\theta : \mathbb{R}^{L \times C} \to \mathbb{R}^{H \times C}, \qquad \widehat{\mathbf{Y}}_t = f_\theta(\mathbf{X}_t), \tag{1}$$

where θ denotes the model parameters, L is the historical window length, and H is the forecasting horizon. The model learns its parameters by minimizing a loss between the predicted and ground-truth sequences and forecasts future values using only information available at the forecast origin.

Kronecker Product of matrices. Given matrices $A = [a_{ij}] \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{u \times v}$, the Kronecker (Van Loan & Pitsianis, 1993; Van Loan, 2000) of A and B is defined as:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mu \times nv}, \tag{2}$$

which represents linear transformations acting along two distinct dimensions as a single operator. Specifically, for an input matrix $X \in \mathbb{R}^{n \times v}$ and transformation matrices $P \in \mathbb{R}^{m \times n}$ and $Q \in \mathbb{R}^{u \times v}$, the two-sided linear mapping satisfies

$$Y = PXQ^T, \qquad \mathrm{vec}(Y) = (Q \otimes P)\,\mathrm{vec}(X), \tag{3}$$

where $\mathrm{vec}(\cdot)$ denotes column-wise vectorization. Thus, applying separate linear transformations along the two axes of a matrix is equivalent to applying a Kronecker-structured operator to its vectorized representation. An unrestricted operator of the same dimensions requires $(mu)(nv)$ parameters, whereas its two-factor representation requires $mn + uv$ parameters.

4 Methodology

4.1 Model Architecture

PKTime organizes temporal dependencies using periodic structures in training data. It first selects periods from the training data, models intra-period and inter-period relationships within each period-specific representation, and finally fuses the predictions from different branches. The method consists of three components: adaptive period selection, Kronecker-structured modeling, and multi-period fusion and output restoration. Figure 2 illustrates the overall architecture of PKTime, including adaptive period selection, Kronecker-structured forecasting, and multi-period fusion.

Adaptive Period Selection. Using a single fixed period restricts modeling to one period-based view, whereas introducing numerous candidates increases the number of branches. PKTime therefore adopts training-data-driven period selection to determine the period set $\mathcal{P} = \{p_1, \ldots, p_K\}$ before constructing the predictor.

We first standardize each variable in the training sequence used for period detection and remove its linear trend, obtaining $Z^{\mathrm{det}} \in \mathbb{R}^{T \times C_s}$, where T and C_s denote the detection length and the number of selected variables, respectively. We then compute the average power spectrum across variables:

$$S(k) = \frac{1}{C_s} \sum_{c=1}^{C_s} \left| \mathrm{RFFT}\big(Z^{\mathrm{det}}_{:,c}\big)_k \right|^2 . \tag{4}$$

For each nonzero frequency bin k within the candidate range, the corresponding period is obtained by rounding T/k to the nearest integer. Using power spectra generated by random temporal permutations as a reference, we iteratively compare the strongest remaining frequency bin in the original spectrum against the maxima of the permuted spectra over the same candidate set, retaining candidates that satisfy the threshold criterion. We subsequently merge nearby periods and select branches based on relative spectral power and a proxy cost budget, retaining at most six periods. This budget uses the estimated parameter count of the initial factorized branches and does not bound the actual parameter count of the final model. If no valid candidate remains, we fall back to $\mathcal{P} = \{1\}$.

Adaptivity here refers to determining the periods and branch count from training data, rather than routing dynamically for individual samples. Period detection uses neither validation nor test data, and the selected periods remain fixed throughout subsequent training and inference. Spectral analysis determines the model structure only; all branches perform forecasting in the time domain. The complete selection algorithm, detection range, and thresholds are provided in the appendix.

Kronecker-Structured Modeling. Given the selected periods, we organize one-dimensional temporal dependencies along two axes: intra-period positions and inter-period sequences. For variable c, we compute the window mean and center the input:

$$\mu_c = \frac{1}{L} \sum_{t=0}^{L-1} X_{t,c}, \qquad \widetilde{x}_c = X_{:,c} - \mu_c \mathbf{1}_L . \tag{5}$$

For each period p, let $N_p = \lceil L/p \rceil$ and $M_p = \lceil H/p \rceil$. We right-pad $\widetilde{x}_c$ with zeros and fold it into $Z_{p,c} \in \mathbb{R}^{p \times N_p}$. Rows represent relative positions within a period, while columns represent consecutive historical period segments, placing observations at the same relative position in the same row. Different branches process different arrangements of the same history, rather than separate frequency components.

We impose different parameter constraints along the two axes and generate two-dimensional predictions through a two-sided mapping:

$$\begin{aligned} D_p &= \mathrm{Diag}(d_p) + U_p V_p^\top , \\ Q_{p,c} &= D_p Z_{p,c} W_p^\top \in \mathbb{R}^{p \times M_p} . \end{aligned} \tag{6}$$

Here, $d_p \in \mathbb{R}^p$ and $U_p, V_p \in \mathbb{R}^{p \times r}$. The diagonal term independently adjusts the response at each position. The low-rank term projects phase information into an r-dimensional intermediate representation and maps it back to the original phase space, enabling interactions across positions. The hyperparameter r controls the latent dimension of the low-rank correction, satisfying $\mathrm{rank}(U_p V_p^\top) \le \min(p, r)$, and is set to 2 by default. It does not constrain the rank of the entire matrix D_p. This parameterization uses $p + 2pr$ intra-period parameters.

For inter-period relationships, we use a dense trainable matrix $W_p \in \mathbb{R}^{M_p \times N_p}$, allowing the coefficient from each historical period segment to each future period segment to be updated independently. D_p is shared across period segments, whereas W_p is shared across intra-period positions. Consequently, the mapping coefficient from historical position (j, n) to future position (i, m) is $(D_p)_{i,j}(W_p)_{m,n}$, capturing phase relationships and period-segment relationships, respectively. Under column-major vectorization,

$$\mathrm{vec}(Q_{p,c}) = (W_p \otimes D_p)\, \mathrm{vec}(Z_{p,c}). \tag{7}$$

This equivalence shows that each branch retains a Kronecker-separable structure in its overall temporal mapping. Computation requires only two matrix multiplications, without explicitly constructing $W_p \otimes D_p$. Each period branch has its own parameters, which are shared across variables.

To provide a structured starting point for training, W_p is obtained through an equivalent dense expansion of an initialized two-factor Kronecker operator. The original factors are then removed,

and the complete matrix is optimized directly. The outer separable structure is retained throughout training, while the internal factorization of W_p is no longer enforced. This conversion occurs only during initialization and does not involve two-stage training.

Multi-Period Fusion and Output Restoration Different periods provide distinct structured forecasting views of the same historical sequence. Their fusion weights are jointly optimized through the forecasting objective rather than determined directly by input spectral power. We first unfold each branch output $Q_{p,c}$ column by column and truncate it to length H, denoting this operation by $\mathcal{U}_{p,H}$. We then introduce a trainable logit a_p for each branch, normalize the logits using softmax, and restore the window mean after fusion:

$$\alpha_p = \frac{\exp(a_p)}{\sum_{q\in\mathcal{P}} \exp(a_q)}, \qquad \widehat{y}_c = \mu_c \mathbf{1}_H + \sum_{p\in\mathcal{P}} \alpha_p \mathcal{U}_{p,H}(Q_{p,c}). \tag{8}$$

All logits a_p are initialized to zero, giving every branch the same initial fusion weight. These weights are global parameters within a fitted model, shared across samples, variables, and forecast positions, without an additional input-dependent routing network. The model generates all H future timestamps simultaneously, without autoregressive iteration.

After restoring the window mean, predictions remain in the standardized space. We subsequently apply inverse standardization using training-set statistics to recover the original scale. All branch mappings and fusion weights are jointly trained in a single stage using mean squared error in the standardized space, with checkpoints selected according to validation loss. For a fixed period set and trained parameters, the forecasting backbone remains linear with respect to the standardized input.

4.2 Parameter and Computational Cost Analysis

PKTime shares all trainable parameters across variables with total number of trainable parameters

$$P_{\text{total}} = K + \sum_{p\in\mathcal{P}} \left[(1+2r)p + M_p N_p\right].$$

Here, r denotes the latent dimension of the intra-period low-rank correction and provides an upper bound on its rank; we use $r = 2$ by default. $K = |\mathcal{P}|$ denotes the number of period branches and the number of fusion logits. The terms $(1+2r)p$ and $M_p N_p$ correspond to the parameter counts of the intra-period and inter-period mappings, respectively. Given the look-back length L, prediction horizon H, and period set $\mathcal{P}$, the trainable parameter count does not explicitly depend on the number of variables C.

PKTime explicitly constructs D_p and then performs two dense matrix multiplications. For a batch of size B with C variables, the leading-order forward-pass computational complexity is

$$O\left(\sum_{p\in\mathcal{P}} \left[p^2 r + BC\left(p^2 N_p + p N_p M_p\right)\right] + BCKH + BCL\right).$$

Here, $p^2 r$ accounts for the explicit construction of the low-rank correction matrix $U_p V_p^\top$. The terms $BCp^2 N_p$ and $BCpN_pM_p$ correspond to the intra-period and inter-period mappings, respectively, while $BCKH$ and BCL account for multi-period output fusion and window-mean computation and centering.

5 Experiments

5.1 Experimental Setup

To comprehensively and accurately evaluate PKTime, we conduct experiments on eight real-world datasets: ETT (ETTh1, ETTh2, ETTm1, and ETTm2), Exchange, Traffic, Electricity, and Solar. Our baselines include state-of-the-art lightweight and general-purpose time series forecasting models, including MixLinear (Ma et al., 2026), SparseTSF (Lin et al., 2024), FITS (Xu et al., 2024), DLinear (Zeng et al., 2023), FEDformer (Zhou et al., 2022b), PatchMLP (Tang & Zhang, 2025), FiLM (Zhou et al., 2022a), Non-stationary Transformer (Liu et al.), and TimesNet (Wu et al., 2023).

Table 1: Multivariate forecasting results (MSE/MAE; lower is better). Best results are **bold**; second-best results are underlined.

Model	ETTh1	ETTh2	ETTm1	ETTm2	Electricity	Traffic	Solar	Exchange
Forecasting horizon $F = 96$								
PKTime	**0.354/0.386**	0.275/0.335	0.308/0.350	0.165/0.257	**0.138/0.232**	**0.392/0.267**	0.195/**0.232**	0.081/**0.199**
MixLinear	0.365/0.387	0.291/0.343	0.332/0.361	0.183/0.271	0.220/0.291	0.707/0.433	0.509/0.476	0.088/0.209
SparseTSF	0.360/**0.386**	**0.273/0.334**	0.305/0.346	0.165/0.255	**0.138**/0.234	0.398/0.274	0.202/0.249	0.085/0.202
FITS	0.379/0.403	0.300/0.364	**0.300**/0.345	**0.164**/0.255	0.140/0.237	0.395/0.275	0.199/0.265	**0.080**/0.202
DLinear	0.376/0.396	0.277/0.345	0.303/0.345	0.165/**0.254**	0.139/0.237	0.407/0.290	0.208/0.255	0.082/**0.199**
PatchMLP	0.396/0.417	0.302/0.351	0.310/0.358	0.173/0.264	0.140/0.240	0.412/0.306	**0.179**/0.242	0.090/0.214
TimesNet	0.370/0.394	0.282/0.346	0.301/**0.343**	0.165/**0.254**	0.154/0.246	0.412/0.284	0.214/0.259	0.087/0.210
FiLM	0.389/0.412	0.334/0.370	0.340/0.378	0.189/0.265	0.169/0.271	0.595/0.312	0.198/0.270	0.112/0.242
Stationary	0.591/0.524	0.347/0.387	0.415/0.410	0.210/0.294	0.171/0.274	0.603/0.330	0.365/0.390	0.121/0.247
FEDformer	0.379/0.419	0.337/0.380	0.463/0.463	0.216/0.309	0.191/0.305	0.593/0.365	0.485/0.570	0.138/0.268
Forecasting horizon $F = 192$								
PKTime	0.395/0.408	0.334/**0.374**	0.337/0.368	0.220/0.294	**0.151/0.244**	**0.401/0.270**	0.213/**0.234**	**0.169/0.294**
MixLinear	0.397/**0.406**	0.344/0.380	0.346/0.370	0.247/0.316	0.216/0.291	0.648/0.401	0.555/0.515	0.178/0.301
SparseTSF	**0.393/0.406**	0.334/0.376	0.340/0.367	0.221/0.292	**0.151**/0.246	0.409/0.275	0.224/0.258	0.172/0.296
FITS	0.408/0.419	0.387/0.423	**0.336**/0.366	0.224/0.304	0.154/0.251	0.407/0.280	0.220/0.282	0.182/0.321
DLinear	0.400/0.418	**0.331**/0.379	0.337/**0.365**	**0.219/0.291**	0.154/0.250	0.418/0.294	0.229/0.267	0.173/0.295
PatchMLP	0.430/0.441	0.381/0.415	0.355/0.387	0.232/0.302	0.160/0.258	0.432/0.315	**0.194**/0.257	0.187/0.311
TimesNet	0.405/0.416	0.358/0.401	0.339/**0.365**	0.220/**0.291**	0.168/0.261	0.415/0.285	0.226/0.257	0.182/0.308
FiLM	0.440/0.443	0.404/0.413	0.392/0.404	0.254/0.310	0.180/0.280	0.613/0.322	0.206/0.276	0.209/0.334
Stationary	0.615/0.540	0.379/0.418	0.494/0.451	0.338/0.373	0.180/0.283	0.611/0.338	0.400/0.386	0.220/0.337
FEDformer	0.420/0.444	0.415/0.428	0.575/0.516	0.297/0.360	0.203/0.316	0.614/0.381	0.415/0.477	0.273/0.379
Forecasting horizon $F = 336$								
PKTime	0.423/0.424	0.354/**0.394**	**0.366/0.384**	**0.272**/0.329	**0.166/0.260**	**0.414/0.277**	0.229/**0.249**	**0.309/0.402**
MixLinear	0.418/0.421	0.365/0.400	0.389/0.392	0.290/0.338	0.230/0.307	0.668/0.407	0.581/0.544	0.319/0.410
SparseTSF	**0.413/0.420**	0.358/0.396	0.369/0.386	0.274/0.327	0.167/0.261	0.417/0.280	0.237/0.267	0.311/0.403
FITS	0.440/0.440	0.490/0.487	0.367/0.386	0.277/0.337	0.169/0.268	0.417/0.286	0.234/0.295	0.327/0.434
DLinear	0.419/0.435	**0.350**/0.396	0.368/**0.384**	**0.272/0.326**	0.170/0.268	0.433/0.308	0.241/0.273	0.317/0.406
PatchMLP	0.456/0.457	0.416/0.440	0.393/0.409	0.294/0.345	0.179/0.277	0.441/0.316	**0.198**/0.259	0.337/0.425
TimesNet	0.434/0.435	0.372/0.425	0.374/0.385	0.277/0.329	0.189/0.284	0.430/0.299	0.241/0.265	0.318/0.409
FiLM	0.523/0.487	0.389/0.435	0.423/0.426	0.313/0.345	0.204/0.304	0.626/0.332	0.208/0.284	0.358/0.435
Stationary	0.632/0.551	0.358/0.413	0.577/0.490	0.432/0.416	0.204/0.305	0.628/0.342	0.414/0.394	0.352/0.437
FEDformer	0.458/0.466	0.389/0.457	0.618/0.544	0.366/0.400	0.221/0.333	0.627/0.389	1.008/0.839	0.437/0.485
Forecasting horizon $F = 720$								
PKTime	0.419/0.443	**0.378/0.421**	**0.414/0.411**	**0.354/0.381**	0.205/**0.293**	**0.452/0.298**	0.234/**0.249**	0.855/0.693
MixLinear	**0.416/0.437**	0.389/0.423	0.430/0.418	0.383/0.394	0.274/0.340	0.727/0.431	0.558/0.505	0.833/0.687
SparseTSF	**0.416**/0.440	0.379/0.422	0.423/0.415	0.360/0.382	0.206/0.295	0.453/0.299	0.245/0.270	0.824/0.683
FITS	0.471/0.493	0.704/0.597	0.419/0.416	0.371/0.401	**0.204**/0.301	0.454/0.308	0.243/0.301	**0.578/0.605**
DLinear	0.435/0.458	0.382/0.425	0.420/0.413	0.359/**0.381**	0.212/0.304	0.486/0.347	0.248/0.277	0.825/0.684
PatchMLP	0.496/0.490	0.433/0.451	0.450/0.439	0.378/0.401	0.214/0.312	0.480/0.338	**0.206**/0.265	0.893/0.711
TimesNet	0.463/0.474	0.425/0.455	0.423/0.414	0.363/0.386	0.249/0.340	0.525/0.371	0.247/0.268	0.815/0.681
FiLM	0.521/0.495	0.434/0.448	0.475/0.453	0.413/0.402	0.205/0.304	0.635/0.340	0.232/0.294	0.944/0.736
Stationary	0.828/0.658	0.422/0.457	0.636/0.535	0.554/0.476	0.221/0.319	0.646/0.350	0.379/0.377	0.725/0.656
FEDformer	0.474/0.488	0.483/0.488	0.612/0.551	0.459/0.450	0.259/0.364	0.646/0.394	0.655/0.627	1.158/0.828

Model years: PKTime (Ours); MixLinear (2026); SparseTSF and FITS (2024); DLinear and TimesNet (2023); PatchMLP (2025); FiLM, Stationary, and FEDformer (2022).
First-place counts (MSE / MAE): PKTime (15 / 22); MixLinear (1 / 2); SparseTSF (6 / 4); FITS (6 / 1); DLinear (4 / 7); PatchMLP (4 / 0); TimesNet (0 / 4); FiLM (0 / 0); Stationary (0 / 0); FEDformer (0 / 0).

Following prior works, we use mean squared error (MSE) and mean absolute error (MAE) as evaluation metrics. We use the TFB benchmark repository for unified evaluation, with all baseline results reproduced using TFB for fair comparison. Please refer to the appendix for more experiment details.

5.2 Main Results

As shown Tab.1, PKTime achieves the best MSE on 15 tasks and the best MAE on 22 tasks, across the 32 tasks formed by eight datasets and four forecasting horizons, leading all compared methods in the number of best results for both metrics. On Traffic, PKTime achieves the lowest MSE and MAE at every forecasting horizon. Averaged over the four horizons, these metrics improve by 0.837% and 1.418% over the respective strongest baselines, FITS and SparseTSF, demonstrating consistent gains across forecasting horizons. On Electricity and Solar, PKTime also achieves the lowest MAE at every horizon. In particular, its average MAE on Solar is 5.767% lower than that of the strongest baseline, PatchMLP, indicating its effectiveness in reducing absolute forecasting errors on these tasks. As the forecasting horizon increases, PKTime maintains strong performance on ETTm1 and ETTm2, achieving the best MSE at horizons of both 336 and 720. At a horizon of 720, its MSE values are 0.414 and 0.354, respectively, improving upon the corresponding strongest baselines, FITS

Table 2: Ablation configurations.

Variant	Intra-period	Inter-period	Periods	Fusion
`full`	DLR	Full	$\mathcal{P}$	Learned
`diag_full`	Diag.	Full	$\mathcal{P}$	Learned
`diaglowrank_factorized`	DLR	Fact.	$\mathcal{P}$	Learned
`diag_factorized`	Diag.	Fact.	$\mathcal{P}$	Learned
`single_period`	DLR	Full	$\{p_1\}$	1
`same_period`	DLR	Full	$p_k = p_1$	Learned
`uniform_fusion`	DLR	Full	$\mathcal{P}$	$1/K$

Notes. DLR denotes $D_p = \mathrm{Diag}(d_p) + U_p V_p^\top$; Diag. removes $U_p V_p^\top$. Full optimizes W_p directly after matched initialization, whereas Fact. retains its internal factorization throughout training. All variants retain the outer structure $W_p \otimes D_p$. The same-period variant preserves K independently trained branches.

Table 3: Structural ablations (MSE/MAE).

Dataset	Full	Diag-Full	DLR-Fact.	Diag-Fact.
ETTh1	0.401/0.417	0.415/0.419	**0.398**/0.420	0.405/**0.414**
ETTh2	**0.338/0.382**	0.346/0.389	0.348/0.389	0.349/0.389
ETTm1	0.360/**0.382**	0.371/0.388	**0.358**/0.383	0.368/0.388
ETTm2	**0.266/0.328**	0.313/0.366	0.269/0.330	0.332/0.380

Table 4: Period and fusion ablations on ETTm1.

Variant	MSE	MAE	ΔMSE (%)	ΔMAE (%)
Full	**0.360**	**0.382**	0.000	0.000
Single period	0.446	0.439	+23.829	+15.009
Same period	0.388	0.407	+7.603	+6.642
Uniform fusion	0.361	0.382	+0.070	+0.050

Notes. Results are averaged over four horizons. Left: variant abbreviations follow the intra-/inter-period definitions in Table 2; Full denotes the complete model. Right: percentage changes are relative to Full. Boldface marks the best value for each metric within each table and dataset; both boldface and percentage changes use unrounded results.

and DLinear, by 1.193% and 1.393%. These results support its effectiveness over longer forecasting horizons. The overall results indicates that PKTime's advantages extend across multiple datasets and forecasting horizons.

5.3 Ablation Studies

We evaluate intra-period interactions, inter-period parameterization, period diversity, and fusion weights on four ETT datasets with forecasting horizons of 96, 192, 336, and 720. Results are averaged over three random seeds and the four horizons. Table 2 summarizes the variants in terms of D_p, W_p, and the fusion components $(\mathcal{P}, \alpha_p)$.

Structural ablations. Table 3 shows that removing $U_p V_p^\top$ increases MSE and MAE on all sixteen tasks, with cross-task averages rising by 5.868% and 3.417%, respectively. Retaining this term also improves MSE on thirteen tasks under factorized inter-period mappings, supporting the value of cross-phase interactions. With D_p unchanged, factorizing W_p increases average MSE and MAE by 0.648% and 0.848%. Removing both design choices increases the two errors by 6.540% and 4.019%. Thus, low-rank intra-period interactions provide more consistent benefits with relatively smaller gains from unrestricted inter-period mappings.

Period diversity and fusion. On ETTm1 (Table 4), retaining only one period increases average MSE and MAE by 23.829% and 15.009%. Repeating the same period while preserving the branch count still increases these errors by 7.603% and 6.642%, supporting complementary forecasting views from different folding periods beyond the benefit of additional branches alone. Uniform fusion performs close to learned fusion, with increases of only 0.070% and 0.050%; period diversity therefore shows a larger observed benefit than learned fusion weights in this setting. On ETTh1, ETTh2, and ETTm2, only one period is selected, so these variants reduce to the same single-period form and produce identical results.

5.4 Runtime Efficiency Analysis

As shown in Fig.3, PKTime combines a compact parameter size with efficient batched inference for multivariate forecasting. On Electricity, its parameter count increases from only 0.241K to 1.021K as the forecast horizon increases from 96 to 720. At the longest horizon, PKTime reduces the parameter count by 99.902%, 99.503%, and 96.849% compared with DLinear, FITS, and SparseTSF, respectively. With a batch size of 128, PKTime achieves the lowest inference time across all eight

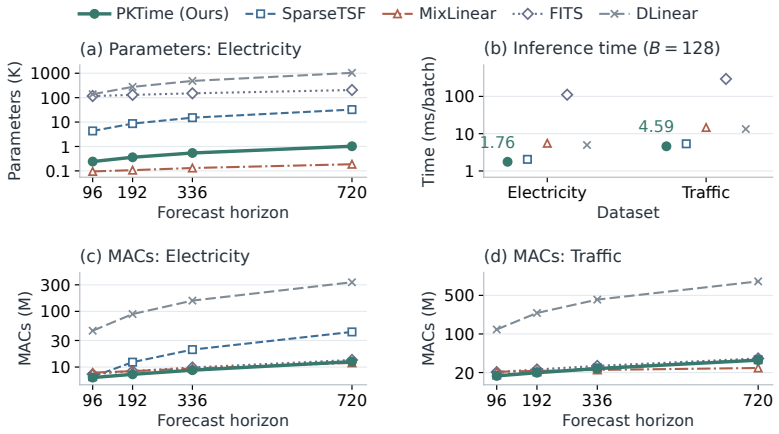

Figure 3: Efficiency comparison of PKTime and four baselines with $L = 720$.

forecasting settings on Electricity and Traffic, averaging 1.762 and 4.591 ms/batch, respectively. These results correspond to speedups of 3.140× and 3.187× over MixLinear and latency reductions of 14.061% and 14.317% relative to the second-fastest method, SparseTSF. Furthermore, when the forecast horizon increases by a factor of 7.500, inference time on the two datasets increases to only 1.688× and 1.710× its original value, respectively. These results demonstrate that PKTime combines compact parameterization with low inference latency in the evaluated high-dimensional batched forecasting settings.

Fig.3 also compares the multiply–accumulate operations (MACs) of PKTime and four baselines on Electricity and Traffic. PKTime achieves the lowest MACs at three forecasting horizons (96, 192, and 336) on Electricity and two horizons (96 and 192) on Traffic. On Electricity, its MAC reduction relative to SparseTSF increases from 3.4% to 70.8% as the forecasting horizon grows. On Traffic, PKTime reduces MACs by 15.8% and 7.3% relative to MixLinear at the two shorter horizons, respectively. Compared with DLinear, PKTime reduces MACs by approximately 85.6%–96.3% across all evaluated horizons on both datasets. As the forecasting horizon increases from 96 to 720, PKTime's MACs grow to only approximately 1.93× their initial values on both datasets, indicating modest growth in computational cost.

5.5 Parameter Sensitivity Analysis

We evaluate $r \in \{0, 1, 2, 4, 8\}$ on the four ETT datasets. Increasing r from 0 to 2 reduces the average MSE and MAE by 5.543% and 3.304%, demonstrating the benefit of low-rank intra-period interactions. Increasing r further to 8 raises the average parameter count by 141.061%, while reducing MSE and MAE by only 0.711% and 0.809%, with gains mainly concentrated on ETTm2. We therefore adopt $r = 2$ as the default, balancing forecasting accuracy and model compactness.

Table 5: Sensitivity to the intra-period rank r on four ETT datasets. Results are averaged over four forecast horizons. Params denotes the average parameter count in thousands.

Rank r	ETTh1			ETTh2			ETTm1			ETTm2		
	MSE	MAE	Params (K)	MSE	MAE	Params (K)	MSE	MAE	Params (K)	MSE	MAE	Params (K)
0	0.415	0.419	**0.445**	0.346	0.389	**0.445**	0.371	0.388	**0.281**	0.313	0.366	**0.127**
1	**0.401**	**0.417**	0.493	**0.338**	0.383	0.493	0.361	0.383	0.569	0.274	0.337	0.319
2 (default)	**0.401**	**0.417**	0.541	**0.338**	**0.382**	0.541	**0.360**	0.382	0.857	0.266	0.328	0.511
4	**0.401**	**0.417**	0.637	0.339	0.383	0.637	0.361	0.382	1.433	0.255	0.317	0.895
8	0.402	0.418	0.829	0.339	0.383	0.829	**0.360**	**0.381**	2.585	**0.253**	**0.315**	1.663

6 Conclusion

We presented PKTime, a lightweight forecasting model combining period-aligned Kronecker operators with data-driven period selection. Compact factors separately model within-period interactions and cross-period dependencies, while learned global weights combine branch forecasts. Experiments on eight datasets demonstrate competitive accuracy with compact parameter budgets, alongside efficient batched inference on Electricity and Traffic. Ablations support the benefits of low-rank within-period interactions and period diversity.

7 AI USE STATEMENT

ChatGPT was used as a general-purpose writing assistance tool to improve the grammar and clarity of writing during the preparation of this paper. LLMs did not contribute to the formulation of the research idea, the experimental design, the data analysis, or the formulation of scientific conclusions. The authors assume full responsibility for all contents presented in this paper.

REFERENCES

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, 2020.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *The Twelfth International Conference on Learning Representations, ICLR*. OpenReview.net, 2024.

Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527, 2011.

Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Nia, James Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for gpt compression. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 219–226, 2022.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 459–469, 2023.

Marawan Gamal Abdel Hameed, Marzieh S Tahaei, Ali Mosleh, and Vahid Partovi Nia. Convolutional neural network compression through generalized kronecker product decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 771–779, 2022.

Cheng He, Zhenyu Guan, Xijie Liang, Defu Lian, Jiajia Li, Enhong Chen, Patrick P. C. Lee, Geng Hu, and Zehao Chen. RAVEN: A regime-aware variable-context expert network for financial time series forecasting. *CoRR*, abs/2606.24062, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Kelvin J. L. Koa, Jan Chen, Yunshan Ma, Huanhuan Zheng, and Tat-Seng Chua. Reasoning on time-series for financial technical analysis. *CoRR*, abs/2511.08616, 2025.

Chih-Yu Lai, Fan-Keng Sun, Zhengqi Gao, Jeffrey H. Lang, and Duane S. Boning. Nominality score conditioned time series anomaly detection by point/sequential reconstruction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023*, 2023.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. SparseTSF: Modeling long-term time series forecasting with 1k parameters. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*.

Aitian Ma, Dongsheng Luo, and Mo Sha. MixLinear: Extreme low resource multivariate time series forecasting with 0.1k parameters. In *International Conference on Learning Representations*, 2026.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kroneckerbert: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2116–2127, 2022.

Peiwang Tang and Weitai Zhang. Unlocking the power of patch: Patch-based MLP for long-term time series forecasting. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025*, pp. 12640–12648. AAAI Press, 2025.

Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000.

Charles F Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear algebra for large scale and real-time applications*, pp. 293–314. Springer, 1993.

Zhiyuan Wang, Xovee Xu, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fan Zhou. Pref: Probabilistic electricity forecasting via copula-augmented state space model. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pp. 12200–12207. AAAI Press, 2022.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with 10k parameters. In *International Conference on Learning Representations*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2023. doi: 10.1609/aaai.v37i9.26317.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Xue Wang, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, and Rong Jin. Film: Frequency improved legendre memory model for long-term time series forecasting. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*, 2022a.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 2022b.

Zhaoyang Zhu, Weiqi Chen, Rui Xia, Tian Zhou, Peisong Niu, Bingqing Peng, Wenwei Wang, Hengbo Liu, Ziqing Ma, Qingsong Wen, and Liang Sun. eforecaster: Unifying electricity forecasting with robust, flexible, and explainable machine learning algorithms. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*, pp. 15630–15638. AAAI Press, 2023.

A Implementation Details

A.1 Experimental Settings

We implement PKTime in PyTorch and train it on a single NVIDIA A100 GPU using the Adam optimizer (Kingma & Ba, 2015) with a constant learning rate of 0.03. Training runs for a maximum of 50 epochs, with an early-stopping patience of 10 epochs to mitigate overfitting. Across all datasets, we use a batch size of 128 and a look-back window of 720 timestamps. We strictly follow the dataset splits provided by TFB.

A.2 Evaluation Metrics and Loss Function

We evaluate forecasting performance using mean squared error (MSE) and mean absolute error (MAE), both computed in the standardized space using training-set statistics:

$$\begin{aligned} \mathrm{MSE} &= \frac{1}{NHC}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{c=1}^{C}\left(\widehat{Y}_{n,h,c} - Y_{n,h,c}\right)^2, \\ \mathrm{MAE} &= \frac{1}{NHC}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{c=1}^{C}\left|\widehat{Y}_{n,h,c} - Y_{n,h,c}\right|. \end{aligned} \tag{9}$$

Here, N, H, and C denote the number of forecasting windows, forecast horizon, and number of variables, respectively. $\widehat{Y}$ and Y represent the predictions and ground truth. MSE penalizes larger errors more heavily, whereas MAE measures the average absolute deviation. Lower values indicate better performance for both metrics. During training, we minimize the mini-batch MSE of mini-batch size 32 , $\mathcal{L}_{\text{train}} = \mathrm{MSE}(\widehat{Y}, Y)$, to jointly optimize all branch parameters and fusion weights without additional auxiliary losses.

A.3 Baseline Settings

MixLinear (Ma et al., 2026) is a lightweight forecasting model that combines complementary time-domain and frequency-domain pathways. Its temporal pathway uses factorized linear transformations to capture intra-segment and inter-segment relationships, while its spectral pathway compresses global patterns through low-rank frequency transformations. The two pathways jointly support forecasting with a compact parameter footprint.

SparseTSF (Lin et al., 2024) introduces cross-period sparse forecasting for efficient long-term prediction. It downsamples the input sequence at periodic intervals and applies compact mappings to model dependencies across periods. By exploiting recurring temporal structure, the model reduces the dimensionality of the forecasting problem and achieves competitive performance with a small number of parameters.

FITS (Xu et al., 2024) formulates time-series forecasting as interpolation in the complex frequency domain. It retains low-frequency components and applies a learnable complex-valued linear transformation before reconstructing the extended sequence through an inverse Fourier transform. This design captures dominant temporal patterns with a compact architecture while filtering out high-frequency variations.

DLinear (Zeng et al., 2023) combines seasonal–trend decomposition with simple linear forecasting. A moving-average operation separates the input into a trend component and a residual component, which are independently projected onto the forecasting horizon. Their predictions are summed to obtain the final output, providing a simple baseline for evaluating the effectiveness of more complex forecasting architectures.

PatchMLP (Tang & Zhang, 2025) combines patch-based representations with multilayer perceptrons for long-term multivariate forecasting. It uses moving averages to separate smooth components from noise-containing residuals, enabling different processing strategies for the two components. Channel mixing captures cross-variable relationships in the smooth representations, while channel-independent processing handles residual variations.

TimesNet (Wu et al., 2023) transforms one-dimensional time series into multiple two-dimensional representations according to detected periods. Each representation organizes intra-period and inter-period variations along separate axes, allowing convolutional kernels to capture both types of temporal structure. Features extracted from different periodic views are subsequently aggregated to model complex temporal variations.

FiLM (Zhou et al., 2022a) combines Legendre polynomial projections with frequency-domain processing for long-term forecasting. The Legendre representation preserves historical information in a compact form, while Fourier-based filtering suppresses noise and extracts useful temporal patterns. Low-rank approximation further reduces computational overhead, supporting efficient modeling of long-range dependencies.

Non-stationary Transformer (Liu et al.) addresses distribution changes through Series Stationarization and De-stationary Attention. The former normalizes input statistics and restores them in the predictions, while the latter reintroduces non-stationary information into attention computation. Together, these modules improve predictability while preserving temporal dependencies that could otherwise be lost through normalization.

FEDformer (Zhou et al., 2022b) combines seasonal–trend decomposition with frequency-enhanced Transformer modules for long-term forecasting. Decomposition captures the overall temporal profile, while frequency-domain representations support efficient modeling of global dependencies. By exploiting sparse spectral representations, the model reduces the computational cost associated with standard attention while retaining informative temporal structure.

A.4 Dataset Details

In this section, we provide detailed descriptions of the eight datasets in our experiments. The table summarizes their key characteristics, including the number of channels, sampling frequency, and total number of timestamps.

Table 6: Details of datasets in our experiment

Dataset	Channels	Sampling frequency	Total #timestamps
Traffic	862	1 hour	17544
Electricity	321	1 hour	26304
Solar	137	10 min	52560
Exchange	8	1 day	7588
ETTh1	7	1 hour	17420
ETTh2	7	1 hour	17420
ETTm1	7	15 min	69680
ETTm2	7	15 min	69680

Traffic[1]: This dataset contains hourly road occupancy measurements collected by 862 sensors on freeways in the San Francisco Bay Area. It is used to evaluate multivariate traffic forecasting models.

Electricity[2]: The preprocessed benchmark contains hourly electricity consumption records for 321 customers from 2012 to 2014. Each variable represents an individual customer's electricity consumption.

Solar[3]: This dataset contains simulated solar power generation time series for 137 photovoltaic plants in Alabama, United States, based on weather conditions from 2006. The benchmark version comprises 137 variables sampled at 10-minute intervals.

Exchange[4]: This dataset contains daily exchange rate records for eight countries—Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore—from 1990 to 2016. It comprises eight variables and is used for multivariate exchange rate forecasting.

[1] https://pems.dot.ca.gov/

[2] https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

[3] https://www.nlr.gov/grid/solar-power-data

[4] https://github.com/laiguokun/multivariate-time-series-data

ETT Datasets[5]: The Electricity Transformer Temperature (ETT) (Zhou et al., 2021) dataset contains approximately two years of operational records from electricity transformers at two stations in China. It comprises four subsets: ETTh1, ETTh2, ETTm1, and ETTm2. ETTh1 and ETTh2 contain hourly observations, whereas ETTm1 and ETTm2 contain observations sampled every 15 minutes; the indices 1 and 2 distinguish the two stations. Each subset includes seven variables: transformer oil temperature and six power load measurements.

B Implementation Details of Adaptive Period Selection

B.1 Detection Data and Preprocessing

Period selection is performed during initialization for each model fit, using only the training observations used for gradient-based optimization. Neither validation nor test data are used. Let T_{tv} denote the combined length of the training and validation portions, and let ρ be the fraction allocated to optimization training. Period detection is restricted to the first $\lfloor \rho T_{\mathrm{tv}} \rfloor$ observations. If this prefix contains more than $20{,}000$ time steps, only its last $20{,}000$ observations are retained. If the number of variables exceeds 32, we select 32 variables at approximately equally spaced indices. We use T and C_s to denote the resulting detection length and number of selected variables, respectively.

Non-finite values in the detection segment are replaced with the median of the finite observations of the corresponding variable. If a variable contains no finite observations, its replacement value is zero. This procedure applies specifically to the period-detection module and does not imply that the forecasting training pipeline uses the same missing-value handling rule.

Each selected variable is centered and divided by its standard deviation. A standard deviation no greater than 10^{-8} is replaced with 1. We then remove a least-squares linear trend along the temporal axis. Specifically, let $\bar{Z}$ denote the standardized detection sequence and let τ denote the centered and scaled time coordinates. The detrended sequence is computed as

$$\begin{aligned} \beta_c &= \frac{\sum_{t=0}^{T-1} \tau_t \bar{Z}_{t,c}}{\sum_{t=0}^{T-1} \tau_t^2 + \epsilon}, \\ Z^{\mathrm{det}}_{t,c} &= \bar{Z}_{t,c} - \tau_t \beta_c, \qquad \epsilon = 10^{-12}. \end{aligned} \tag{10}$$

No additional variance normalization is applied after detrending. Detrending is used only for period identification. The forecasting backbone still operates on historical sequences processed by training-set standardization and window centering; it does not separately remove a linear trend from its forecasting input.

B.2 Spectral Candidates and Permutation Reference

We compute the real-valued fast Fourier transform of the detection sequence and average the power across the selected variables:

$$S(k) = \frac{1}{C_s} \sum_{c=1}^{C_s} \left| \mathrm{RFFT}\left(Z^{\mathrm{det}}_{:,c}\right)_k \right|^2. \tag{11}$$

Given the look-back length L, the maximum candidate period is

$$p_{\max} = \min\left(\left\lfloor \frac{L}{2} \right\rfloor, \left\lfloor \frac{T}{2} \right\rfloor \right). \tag{12}$$

The candidate frequency-bin set is

$$\Omega_0 = \left\{ k \in \mathbb{Z} : 1 \le k \le \left\lfloor \frac{T}{2} \right\rfloor,\ 2 \le \frac{T}{k} \le p_{\max} \right\}. \tag{13}$$

The integer period associated with bin k is obtained by rounding T/k. Candidate-range membership is determined using the unrounded value T/k; the rounded period is subsequently clipped to $[2, p_{\max}]$.

[5] https://github.com/zhouhaoyi/ETDataset

The power spectrum is not additionally doubled for one-sided spectral scaling or normalized by total spectral energy.

To construct a reference distribution, we generate $R = 99$ random permutations of the time indices. Within each permutation, the same row ordering is applied to all selected variables. This disrupts temporal order while preserving the correspondence between variables within each time row. For each permuted version of Z^{det}, we directly compute the mean power spectrum $S^{(b)}$, without repeating detrending. The period-detection random seed is fixed to 0 and is configured separately from the model-training seed.

At step j, we identify the strongest observed spectral bin within the remaining candidate set Ω_j:

$$k_j = \arg\max_{k \in \Omega_j} S(k). \tag{14}$$

Its power is compared with the maximum power of each permutation spectrum over the same remaining set:

$$\widehat{q}_j = \frac{1 + \sum_{b=1}^{R} \mathbb{I}\left[\max_{k \in \Omega_j} S^{(b)}(k) \geq S(k_j)\right]}{R+1}. \tag{15}$$

Here, $\mathbb{I}[\cdot]$ denotes the indicator function.

If $\widehat{q}_j \leq 0.05$, the candidate is retained, only k_j is removed from the remaining set, and the procedure continues. The procedure terminates when the threshold is first exceeded. The implementation records this first rejected candidate for diagnostic purposes, but excludes it from subsequent selection. Neighboring and harmonic frequency bins are not simultaneously removed at this stage.

B.3 Nearby-Period Merging, Power Filtering, and Proxy Budget

Candidates passing the permutation threshold are sorted in descending order of spectral power. A candidate period p is discarded if any previously retained, stronger period q satisfies

$$\frac{|p-q|}{\min(p,q)} \leq \delta, \qquad \delta = 0.1. \tag{16}$$

Thus, merging retains the stronger candidate rather than averaging nearby period values.

We subsequently retain only candidates whose power is at least $\eta = 0.2$ times that of the strongest remaining candidate. This threshold is relative to the maximum candidate power; it is not a cumulative spectral-energy coverage threshold.

Finally, candidates are considered in descending order of power, and at most $K_{\max} = 6$ branches are selected under a proxy parameter budget. Let $\widetilde{P}(p)$ denote the estimated branch parameter count before the inter-period operator is expanded into a dense matrix. For the strongest candidate $p_\star$, the budget is

$$\mathcal{B} = 3\,\widetilde{P}(p_\star). \tag{17}$$

If adding a candidate would exceed the cumulative proxy budget, that candidate is skipped and subsequent candidates are still considered. Selection stops when the maximum branch count is reached. If no valid candidate remains, the model falls back to $\mathcal{P} = \{1\}$. The same fallback is used when the detection sequence is too short, the candidate range is empty, or no spectral peak passes the screening procedure.

In the final implementation, let $N_p = \lceil L/p \rceil$ and $M_p = \lceil H/p \rceil$. The two internal factors used for initialization have input dimensions n_1, n_2 and output dimensions m_1, m_2. With a single Kronecker term as the initial inter-period operator, the proxy cost is

$$\widetilde{P}(p) = (1+2r)p + n_1 m_1 + n_2 m_2, \tag{18}$$

where r is the latent dimension of the intra-period low-rank correction.

This estimate excludes the global fusion logits and uses the factorized initialization cost rather than the expanded inter-period matrix size $N_p M_p$. Consequently, $\mathcal{B}$ is a proxy budget for structural selection, not a hard upper bound on the final model's trainable parameter count. Because the proxy

Algorithm 1 Training-Data-Driven Period Selection

Require: Optimization-training sequence X^{tr}; look-back length L; forecast horizon H; correction dimension r
Ensure: Selected period set $\mathcal{P}$

Detection data and reference spectra
Retain the last at most 20,000 training observations
Select at most 32 variables at approximately equally spaced indices
Replace non-finite values, standardize, and detrend to obtain $Z^{\mathrm{det}} \in \mathbb{R}^{T \times C_s}$
Compute $p_{\max}$ and Ω_0 using Eq.12 and Eq.13
if $T < 4$ or $\Omega_0 = \varnothing$ **then**
    return $\{1\}$
end if
Compute the observed spectrum S using Eq.11
Generate 99 shared time-row permutations with seed 0
Compute the corresponding spectra $S^{(1)}, \ldots, S^{(99)}$
Sequential spectral screening
$\Omega \leftarrow \Omega_0$; $\mathcal{C} \leftarrow [\,]$
while $\Omega \neq \varnothing$ **do**
    $k^\star \leftarrow \arg\max_{k \in \Omega} S(k)$
    Compute $\widehat{q}$ using Eq.15 with Ω and $k^\star$
    $p \leftarrow \mathrm{clip}\big(\mathrm{round}(T/k^\star), 2, p_{\max}\big)$
    Append $(p, S(k^\star), \widehat{q})$ to $\mathcal{C}$
    $\Omega \leftarrow \Omega \setminus \{k^\star\}$
    if $\widehat{q} > 0.05$ **then**
        break
    end if
end while
Multi-period selection
Keep candidates with $\widehat{q} \leq 0.05$ and sort them by decreasing power
Greedily discard candidates within relative distance 0.1 of a retained stronger period
Discard candidates with power below 0.2 times the strongest remaining power
if $\mathcal{C} = \varnothing$ **then**
    return $\{1\}$
end if
$\mathcal{B} \leftarrow 3\,\widetilde{P}(p_\star)$, where $p_\star$ is the strongest remaining candidate
$\mathcal{P} \leftarrow [\,]$; $c \leftarrow 0$
for each remaining candidate p in decreasing order of power **do**
    if $|\mathcal{P}| = 6$ **then**
        break
    end if
    if $c + \widetilde{P}(p) \leq \mathcal{B}$ **then**
        Append p to $\mathcal{P}$
        $c \leftarrow c + \widetilde{P}(p)$
    end if
end for
if $\mathcal{P} = \varnothing$ **then**
    $\mathcal{P} \leftarrow \{1\}$
end if
return $\mathcal{P}$

cost depends on the forecast horizon H, different horizons may yield different selected period sets even for the same training sequence.

Once selected, the periods and branch count remain fixed throughout the current training run and subsequent inference. They are neither learned through backpropagation nor dynamically changed for individual forecasting windows. The branch mappings and fusion weights are the trainable components.

Algorithm 1 summarizes the procedure. The factor-dimension selection rule used in Eq.18 is described in Appendix B.2. Validation and test forecasting errors are not used to adjust the selected period set.

B.4 Results of Adaptive Period Selection

Table 7 summarizes the number and lengths of the selected periods for each dataset. Six of the eight datasets retain a single period, while ETTm1 and Exchange retain two and three periods, respectively. Accounting for the sampling frequencies, the selected periods in ETTh1, ETTh2, ETTm2, Solar, Electricity, and Traffic correspond to daily cycles. ETTm1 additionally retains a half-day period, yielding two harmonically related temporal scales. Exchange selects three longer periods of 312, 280, and 253 time steps. These results illustrate how the selection procedure produces dataset-specific period configurations, retaining a compact single-branch structure for most datasets while allowing multiple branches when additional candidates satisfy the selection criteria.

Table 7: Number of selected periods and corresponding period length of each dataset.

Dataset	ETTh1	ETTh2	ETTm1	ETTm2	Solar	Electricity	Traffic	Exchange
#Selected Periods	1	1	2	1	1	1	1	3
Period Length (timestamps)	24	24	48, 96	96	144	24	24	312, 280, 253

C Boundary Handling and Forward Computation

C.1 Window Centering, Padding, and Period Folding

Let $x_c \in \mathbb{R}^L$ denote the input window for variable c in the space standardized using training-set statistics. We center each window as

$$\mu_c = \frac{1}{L}\sum_{t=0}^{L-1} x_{t,c}, \qquad \widetilde{x}_c = x_c - \mu_c \mathbf{1}_L. \tag{19}$$

For each selected period p, define

$$N_p = \left\lceil \frac{L}{p} \right\rceil, \qquad M_p = \left\lceil \frac{H}{p} \right\rceil. \tag{20}$$

We right-pad the centered sequence with zeros to length pN_p and fold it into $Z_{p,c} \in \mathbb{R}^{p \times N_p}$:

$$(Z_{p,c})_{i,n} = \begin{cases} \widetilde{x}_{np+i,c}, & np+i < L, \\ 0, & np+i \geq L, \end{cases} \quad 0 \leq i < p, \quad 0 \leq n < N_p. \tag{21}$$

Rows represent relative positions within a period, while columns represent consecutive period segments. Phase is defined relative to the beginning of the current input window, without additional calendar alignment. No additional phase rotation is applied when L is not divisible by p. Padding zeros are defined in the window-centered space, rather than in the original measurement units.

C.2 Branch Mapping, Unfolding, and Output Cropping

Each branch produces

$$Q_{p,c} = D_p Z_{p,c} W_p^\top, \qquad D_p = \mathrm{Diag}(d_p) + U_p V_p^\top, \tag{22}$$

where $d_p \in \mathbb{R}^p$, $U_p, V_p \in \mathbb{R}^{p \times r}$, and $W_p \in \mathbb{R}^{M_p \times N_p}$. The output matrix is unfolded column by column and cropped to its first H entries:

$$\left[\mathcal{U}_{p,H}(Q_{p,c})\right]_h = (Q_{p,c})_{h \bmod p,\, \lfloor h/p \rfloor}, \qquad 0 \leq h < H. \tag{23}$$

All branches process the same complete history under different period-based arrangements. They do not correspond to disjoint frequency components obtained through spectral decomposition.

C.3 Forward Computation

Algorithm 2 summarizes the forward pass. Its inputs are standardized using training-set statistics, and its output remains in the standardized space.

Algorithm 2 Forward forecasting with PKTime

Input: Standardized history $X \in \mathbb{R}^{B\times L\times C}$, fixed periods $\mathcal{P}$, horizon H, parameters θ
Output: Standardized forecast $\widehat{Y} \in \mathbb{R}^{B\times H\times C}$
Window centering
$\mu \leftarrow \mathrm{Mean}_t(X) \in \mathbb{R}^{B\times 1\times C}$
$X^{\mathrm{cen}} \leftarrow X - \mu$
Kronecker-structured modeling
for $p \in \mathcal{P}$ **do**
  $N_p \leftarrow \lceil L/p \rceil$; $M_p \leftarrow \lceil H/p \rceil$
  Right-pad X^{cen} to length pN_p
  Fold the padded input into $Z_p \in \mathbb{R}^{B\times C\times p\times N_p}$
  $D_p \leftarrow \mathrm{Diag}(d_p) + U_p V_p^\top$
  $Q_{p,b,c} \leftarrow D_p Z_{p,b,c} W_p^\top$ for all b, c
  $(Y_p)_{b,:,c} \leftarrow \mathcal{U}_{p,H}(Q_{p,b,c})$ for all b, c
end for
Multi-period fusion and output restoration
$\alpha \leftarrow \mathrm{softmax}(a)$
$\widehat{Y} \leftarrow \sum_{p\in\mathcal{P}} \alpha_p Y_p + \mu$
return $\widehat{Y}$

Here, $\mathcal{U}_{p,H}$ unfolds each $p \times M_p$ output column by column and retains its first H entries. Each Y_p has shape $[B, H, C]$, and μ is broadcast over the forecast horizon.

Both mapping matrices are shared across the batch and variable dimensions. The forward pass does not mix predictive information across variables. The external prediction interface subsequently applies inverse standardization using training-set statistics to restore the original measurement units. Period selection and matched initialization are not repeated during the forward pass, which requires neither FFT, inverse FFT, nor autoregressive iteration.

D Structured Matched Initialization

D.1 Selecting the Factor Dimensions

Matched initialization first constructs a two-factor inter-period mapping. Two integer factors are selected separately for N_p and M_p, allowing their product to slightly exceed the target dimension. For a target dimension $s > 1$, the preferred search range is

$$2 \le a \le b, \qquad s \le ab \le \lfloor 1.25s \rfloor. \tag{24}$$

Within this range, the selected pair minimizes

$$\phi_s(a, b) = 100\frac{ab - s}{s} + \sqrt{\frac{(\ell_a - \bar{\ell})^2 + (\ell_b - \bar{\ell})^2}{2}}, \tag{25}$$

where

$$\ell_a = \log(a + 10^{-12}), \qquad \ell_b = \log(b + 10^{-12}), \qquad \bar{\ell} = \frac{\ell_a + \ell_b}{2}. \tag{26}$$

This criterion prioritizes reduced padding while favoring balanced factor dimensions. Ties are resolved deterministically by comparing, in order, the padding ratio, logarithmic dispersion, product, and factor tuple.

If no nondegenerate pair exists in the preferred range, the product upper bound is expanded to at least $2\max(2, \lceil s/2 \rceil)$ to avoid a unit factor. A warning is recorded when the resulting padding exceeds the preferred range. Thus, 25% is a preferred search limit rather than an absolute padding bound for all dimensions. Pairs containing a unit factor are allowed only if no feasible nondegenerate pair can be found. For $s = 1$, the factor pair is $(1, 1)$.

The resulting input and output factor pairs are (n_1, n_2) and (m_1, m_2), respectively. We construct

$$A_{p,1} \in \mathbb{R}^{m_1\times n_1}, \qquad A_{p,2} \in \mathbb{R}^{m_2\times n_2}, \tag{27}$$

and initialize each factor independently using Xavier uniform initialization. The original factorized mapping right-pads its input, applies two linear transformations following its internal axis order, and unfolds and crops the output to length M_p. We denote this complete linear mapping by $\mathcal{K}_p : \mathbb{R}^{N_p} \to \mathbb{R}^{M_p}$. The two factors forming a single Kronecker term here are used to initialize the inter-period matrix. They are distinct from the intra-period correction dimension r.

D.2 Converting the Initial Operator into a Full Matrix

To preserve the structured initial function while allowing each inter-period coefficient to be optimized independently, we expand the factorized mapping into a full matrix:

$$(W_p^{(0)})_{:,j} = \mathcal{K}_p(e_j), \qquad j = 0, \dots, N_p - 1, \tag{28}$$

where $e_j \in \mathbb{R}^{N_p}$ is a standard basis vector. This definition includes the original operator's internal padding, axis ordering, and cropping. It therefore does not identify the effective mapping with an uncropped Kronecker matrix that omits these boundary operations.

The implementation processes at most 64 basis vectors at a time and checks output equivalence using three deterministic probes: constant, linearly varying, and cosine inputs. The relative and absolute tolerances are 2×10^{-5} and 5×10^{-6}, respectively. Nonfinite matrix entries or failed equivalence checks raise an error before training begins.

All branches are randomly initialized in their original order before any matrix expansion is performed. Expansion and verification consume no additional random numbers. For the intra-period mapping, we set $d_p = \mathbf{1}_p$ and initialize the entries of U_p and V_p independently from $\mathcal{N}(0, 0.02^2)$. Because neither low-rank factor is initialized to zero, D_p is initially close to, but not exactly equal to, the identity matrix. All fusion logits are initialized to zero.

After expansion, the original factor parameters are replaced by the trainable full matrix W_p and removed from the model. The optimizer is constructed only after this conversion. Training no longer constrains the internal factorization of W_p, whereas the outer Kronecker-separable structure $W_p \otimes D_p$ is retained throughout training. This is a parameterization conversion before optimization, not a two-stage training procedure.

D.3 Matched Initialization Procedure

Algorithm 3 gives the initialization procedure. All random initializations are completed before any matrix expansion begins, and the optimizer is constructed only after the conversion, using the returned parameter set.

E Mathematical Properties and Computational Complexity

E.1 Kronecker Equivalence

Under column-major vectorization, the identity for two-sided matrix multiplication gives

$$\mathrm{vec}(D_p Z_{p,c} W_p^\top) = (W_p \otimes D_p)\,\mathrm{vec}(Z_{p,c}). \tag{29}$$

The coefficient mapping historical position (j, n) to output position (i, m) is therefore $(D_p)_{i,j}(W_p)_{m,n}$. The intra-period matrix D_p is shared across period segments, while W_p is shared across within-period positions. The implementation realizes this mapping using two matrix multiplications without explicitly constructing the outer Kronecker matrix.

The low-rank correction satisfies

$$\mathrm{rank}(U_p V_p^\top) \le \min(p, r). \tag{30}$$

Here, r denotes the latent dimension of the correction.

E.2 Linearity under a Fixed Structure

Let $J_L = I_L - \mathbf{1}_L \mathbf{1}_L^\top / L$ be the centering matrix. Let $E_{p,L}$ be the linear embedding that pads the input and produces the column-major vectorization of its folded representation, and let $R_{p,H}$ crop the

Algorithm 3 Structured matched initialization of PKTime

Input: Periods $\mathcal{P}$, history length L, horizon H, correction dimension r, model random seed
Output: Trainable parameters $\theta = \{d_p, U_p, V_p, W_p, a_p\}_{p\in\mathcal{P}}$

Random initialization of all branches
for $p \in \mathcal{P}$ in selection order **do**
  $N_p \leftarrow \lceil L/p \rceil$; $M_p \leftarrow \lceil H/p \rceil$
  Select two-factor dimensions for N_p and M_p
  Construct $\mathcal{K}_p$, including its padding and cropping operations
  Initialize both factors of $\mathcal{K}_p$ independently using Xavier uniform
  $d_p \leftarrow \mathbf{1}_p$
  Initialize entries of $U_p, V_p \in \mathbb{R}^{p\times r}$ independently from $\mathcal{N}(0, 0.02^2)$
end for
$a \leftarrow \mathbf{0}_{|\mathcal{P}|}$
Deterministic expansion without additional random draws
for $p \in \mathcal{P}$ **do**
  for each index block $\mathcal{J} \subseteq \{0, \dots, N_p - 1\}$ with $|\mathcal{J}| \le 64$ **do**
    $(W_p^{(0)})_{:,j} \leftarrow \mathcal{K}_p(e_j)$ for all $j \in \mathcal{J}$
  end for
  Compare $W_p^{(0)} v$ and $\mathcal{K}_p(v)$ using constant, linear, and cosine probes
  Use relative tolerance 2×10^{-5} and absolute tolerance 5×10^{-6}
  if $W_p^{(0)}$ contains nonfinite entries or an equivalence check fails **then**
    Raise an error and terminate initialization
  end if
  Register $W_p \leftarrow W_p^{(0)}$ as a trainable full matrix
  Remove the original factor parameters of $\mathcal{K}_p$ from the model
end for
return θ

unfolded output to length H. For a fixed period set and fixed parameters, each variable is predicted as

$$\widehat{y}_c = A_\theta x_c, \tag{31}$$

where

$$A_\theta = \frac{1}{L}\mathbf{1}_H\mathbf{1}_L^\top + \sum_{p\in\mathcal{P}} \alpha_p R_{p,H}(W_p \otimes D_p)E_{p,L}J_L. \tag{32}$$

Although the fusion weights are parameterized by softmax, they do not depend on the input once the model is fixed. The forecasting backbone therefore remains linear in the standardized input.

E.3 Constant Preservation and Additive-Shift Equivariance

Since $J_L\mathbf{1}_L = 0$, we have

$$A_\theta\mathbf{1}_L = \mathbf{1}_H. \tag{33}$$

Consequently, a constant history with value b produces a constant forecast with the same value. For any history x_c and constant offset b,

$$f_\theta(x_c + b\mathbf{1}_L) = f_\theta(x_c) + b\mathbf{1}_H. \tag{34}$$

E.4 Functional Equivalence at Initialization

The original factorized mapping $\mathcal{K}_p$ is linear. Thus, for any $v = \sum_j v_j e_j$,

$$W_p^{(0)} v = \sum_j v_j \mathcal{K}_p(e_j) = \mathcal{K}_p(v). \tag{35}$$

E.5 Trainable Parameter Count

Each period branch contains p diagonal parameters, $2pr$ low-rank factor parameters, and N_pM_p entries in its full inter-period matrix. There are also $K = |\mathcal{P}|$ fusion logits. Hence,

$$P_{\text{total}} = K + \sum_{p\in\mathcal{P}} \left[(1 + 2r)p + N_pM_p\right]. \tag{36}$$

When p divides both L and H, the inter-period matrix of a single branch contains LH/p^2 parameters. For example, with $L = 720$, $H = 96$, $p = 24$, and $r = 2$, we obtain $N_p = 30$ and $M_p = 4$. The branch mappings contain $120 + 120 = 240$ parameters, excluding the fusion logit.

If the period selection falls back to $\mathcal{P} = \{1\}$, then $N_1 = L$ and $M_1 = H$. With $r = 2$, the current implementation stores $LH + 6$ trainable parameter elements.

E.6 Per-Batch MACs and Arithmetic Complexity

PKTime explicitly constructs D_p before applying the intra-period and inter-period mappings. For batch size B and C variables, the dominant matrix multiply-accumulate operation counts (MACs) are

$$\begin{aligned} \mathrm{MACs}_{D_p} &= p^2 r, \\ \mathrm{MACs}_{\mathrm{intra},p} &= BCp^2 N_p, \\ \mathrm{MACs}_{\mathrm{inter},p} &= BCpN_pM_p. \end{aligned} \tag{37}$$

The construction cost of D_p is not multiplied by BC, because the resulting matrix is shared across all samples and variables in a batch. Including output fusion, window centering, and mean restoration, the leading arithmetic complexity is

$$O\left(\sum_{p\in\mathcal{P}} \left[p^2 r + BC\left(p^2 N_p + pN_pM_p\right)\right] + BCKH + BCL\right). \tag{38}$$

The $BCKH$ term accounts for multi-period fusion and also covers mean restoration, which has equal or lower asymptotic order. The BCL term accounts for window-mean computation and centering. The implementation computes the full padded output before cropping; the preceding matrix-multiplication counts therefore cannot be reduced using the cropped output size.

E.7 One-Time Preparation Costs versus Forecasting Costs

Period identification and matched initialization are performed only once at the beginning of each model fit. The spectral procedure requires one transform of the observed sequence and R transforms of permuted sequences, resulting in FFT work of $O((R + 1)C_sT \log T)$. Here, T is the detection length, C_s is the number of selected variables, and R is the number of permutations.

F Limitations

PKTime keeps the selected periods fixed after initialization, so its periodic structure does not adapt to changes during inference. Its linear, channel-independent forecasting backbone does not explicitly model nonlinear or cross-variable dependencies. Extending the model to evolving temporal structures and richer dependencies while preserving compactness remains future work.