PRISM: LIGHTWEIGHT LONG-TERM TIME SERIES FORECASTING WITH PERIOD-ALIGNED SUMMARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term time series forecasting (LTSF) requires models that capture temporal dependencies while remaining computationally efficient. We propose PRISM, a lightweight forecasting architecture that exploits periodic structure as a computational prior. PRISM reorganizes the input sequence into a period-aligned representation with cross-period and intra-period axes, and assigns asymmetric computational roles to them. Sequence summarization compresses the cross-period dimension before forecasting, while lightweight temporal mixing refines the intra-period structure. This allows the forecasting head to operate on a compact representation rather than the full historical sequence. Experiments on seven LTSF benchmarks across four forecasting horizons show that PRISM maintains competitive forecasting accuracy with a small computational footprint. Efficiency measurements and structural ablations further confirm that the period-aligned design substantially reduces computational cost while preserving predictive performance. Overall, PRISM provides a simple and effective approach to balancing forecasting accuracy and computational efficiency in LTSF.

1 INTRODUCTION

Long-term time series forecasting (LTSF) supports planning and decision making in applications such as energy management, traffic systems, weather monitoring, and industrial operations. Forecasting over extended horizons requires models to capture informative temporal dependencies from historical observations. Modeling such dependencies while producing reliable long horizon forecasts remains a central challenge in LTSF.

To address this challenge, diverse deep architectures have been developed for LTSF, including recurrent, convolutional, and Transformer based models (Liu et al., 2022; Wu et al., 2023; Zhou et al., 2021; Nie et al., 2023). Transformer based models, in particular, have been extensively explored for long horizon forecasting, with numerous variants designed to improve temporal modeling and forecasting accuracy. However, many high capacity forecasting architectures require substantial computation and memory during training and inference. These requirements can limit their practical use in resource constrained environments.

To reduce this computational burden, we turn to the periodic structure present in many real world time series. Given a period length P, a one-dimensional input sequence can be reorganized into a period aligned representation with two temporal axes: progression across successive periods and positions within each period. Rather than using this organization solely as a predictive representation, we exploit it as a computational prior that exposes two complementary forms of temporal structure and enables both to be modeled compactly in different ways. Based on this principle, we propose PRISM, a lightweight LTSF model that applies axis specific computation to the cross period and intra period structures. Along the cross period axis, shared sequence summarization reduces redundant historical information and explicitly compresses the dimension from N to N' before forecasting. Along the intra period axis, a parameter shared temporal mixing operator captures within period dependencies while preserving the period length P. PRISM therefore replaces full resolution temporal modeling with compact, axis specific operations over a period aligned representation. The key design principle lies not in the use of convolution itself, but in using period alignment to expose

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.

two complementary temporal structures and modeling each with a lightweight operator suited to its role.

We evaluate PRISM on seven widely used LTSF benchmarks across four forecasting horizons using both MSE and MAE. In addition to forecasting accuracy, we evaluate computational efficiency through parameter count, MACs, memory usage, training time, and inference latency on both GPU and CPU. The results show that PRISM maintains competitive forecasting performance with a small computational footprint. Structural ablations further demonstrate the distinct roles of period reorganization, sequence summarization, and temporal mixing, while period sensitivity experiments examine the effect of the predefined period prior.

Our main contributions are summarized as follows:

- We propose PRISM, a lightweight LTSF architecture that uses periodic structure as a computational prior to expose two complementary temporal structures—cross-period and intra period—and models both with compact axis specific operators.
- We introduce an asymmetric dual axis design in which cross-period sequence summarization explicitly reduces the historical dimension from N to N', while parameter shared intra-period temporal mixing captures within period dependencies with only a small number of additional parameters. This enables forecasting to operate on a compact period aligned representation instead of directly modeling the original full resolution sequence.
- We evaluate PRISM across seven benchmarks and four forecasting horizons in terms of both forecasting accuracy and practical computational efficiency, complemented by structural ablations and period sensitivity analysis.

2 Related Work

2.1 Long-Term Time Series Forecasting

Long-term time series forecasting has been studied using recurrent, convolutional, and Transformer based architectures. Convolutional approaches such as SCINet, TimesNet, and ModernTCN model temporal structure through hierarchical or multi scale operations (Liu et al., 2022; Wu et al., 2023; Luo & Wang, 2024), while Transformer based methods including Informer, Autoformer, FEDformer, PatchTST, and iTransformer improve long-range modeling through efficient attention, decomposition, frequency domain processing, patching, or alternative tokenization (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022; Nie et al., 2023; Liu et al., 2024). These methods demonstrate the effectiveness of expressive temporal modeling, but many retain substantially greater computational requirements than lightweight forecasting approaches.

2.2 Lightweight Long-Term Time Series Forecasting

Alongside increasingly expressive forecasting architectures, a growing body of work has investigated lightweight alternatives for LTSF. DLinear demonstrates that simple linear mappings combined with trend–seasonal decomposition can provide competitive long term forecasting performance (Zeng et al., 2023). FITS further reduces model complexity by transforming time series into the frequency domain and performing interpolation with a compact complex valued linear layer, together with low pass filtering to retain dominant frequency components (Xu et al., 2024).

Recent lightweight approaches exploit different structural properties of time series to construct highly compact forecasting models. SparseTSF introduces Cross Period Sparse Forecasting, which reorganizes observations according to a period length and performs shared prediction over downsampled cross period subsequences (Lin et al., 2024). TimeBase leverages temporal pattern similarity and approximate low rank structure by extracting compact basis temporal components and shifting forecasting from point level to segment level prediction (Huang et al., 2025). MixLinear combines segment based processing in the time domain with adaptive low rank filtering in the frequency domain, using separate pathways to represent local and global temporal patterns with a highly compact parameterization (Ma et al., 2026).

These studies demonstrate that effective LTSF can be achieved through a variety of lightweight design principles, including frequency domain modeling, cross period sparse forecasting, basis based temporal modeling, and low rank spectral processing.

PRISM is related to these methods in exploiting temporal structure, but differs in how period alignment is used computationally. SparseTSF reorganizes a sequence according to periodicity and performs shared prediction over sparse cross period subsequences, primarily reducing the cross period forecasting problem. TimeBase instead extracts compact basis periods and performs forecasting through the learned basis representation. MixLinear separates local and global structures through time domain segmentation and frequency domain low rank filtering.

In contrast, PRISM explicitly exposes both cross period and intra period structures within a single period aligned representation and assigns a different compact operator to each axis. Cross period redundancy is reduced through sequence summarization, whereas intra period dependencies are modeled through a parameter shared lightweight mixing operator. PRISM therefore obtains efficiency without explicit basis extraction or frequency domain processing, by reorganizing full resolution temporal modeling into compact axis specific computation before the forecasting head.

3 METHOD

3.1 PRISM OVERVIEW

Given a multivariate input sequence $\mathbf{X} \in \mathbb{R}^{L \times C}$ with historical length L and C variables, PRISM predicts the subsequent H observations $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times C}$. Figure 1 illustrates the overall architecture of PRISM. After normalization, PRISM reorganizes the input according to a predefined period length P, producing a period aligned representation with two temporal axes. Let $N = L/P$ denote the number of historical periods and $M = H/P$ the number of periods to be forecast. The cross period axis represents observations at corresponding positions across successive periods, whereas the intra period axis represents temporal progression within each period.

PRISM processes the two temporal axes using different forms of compact modeling. Along the cross period axis, a shared convolutional operator summarizes redundant historical information and explicitly reduces the dimension from N to N' before forecasting. Along the intra period axis, a lightweight parameter shared convolution models dependencies among positions within each period while preserving the period length P. Finally, a shared linear forecasting head maps the compact cross period representation to M future periods, which are rearranged into the final forecast sequence of length H.

Thus, both temporal structures exposed by period alignment are modeled compactly, but through different mechanisms: the cross period axis undergoes explicit dimensional compression, whereas the intra period axis is represented through lightweight parameter sharing without reducing its length. This axis specific design avoids applying a large prediction operator directly to the original full resolution temporal sequence.

3.2 PERIOD-ALIGNED REPRESENTATION

PRISM adopts a channel independent forecasting strategy (Nie et al., 2023), where each variable in a multivariate time series is processed independently using the same model parameters. Accordingly, for notational simplicity, we omit the channel dimension in the following description and consider a univariate input sequence $\mathbf{X} \in \mathbb{R}^{L}$.

Before period based reorganization, we normalize each input sequence by subtracting its temporal mean. Given $\mathbf{X} = \{x_1, x_2, \ldots, x_L\}$, the mean is computed as

$$\mu = \frac{1}{L} \sum_{l=1}^{L} x_l, \tag{1}$$

and the normalized sequence is obtained as

$$\tilde{\mathbf{X}} = \mathbf{X} - \mu. \tag{2}$$

The removed mean μ is added back to the prediction after forecasting to restore the original scale.

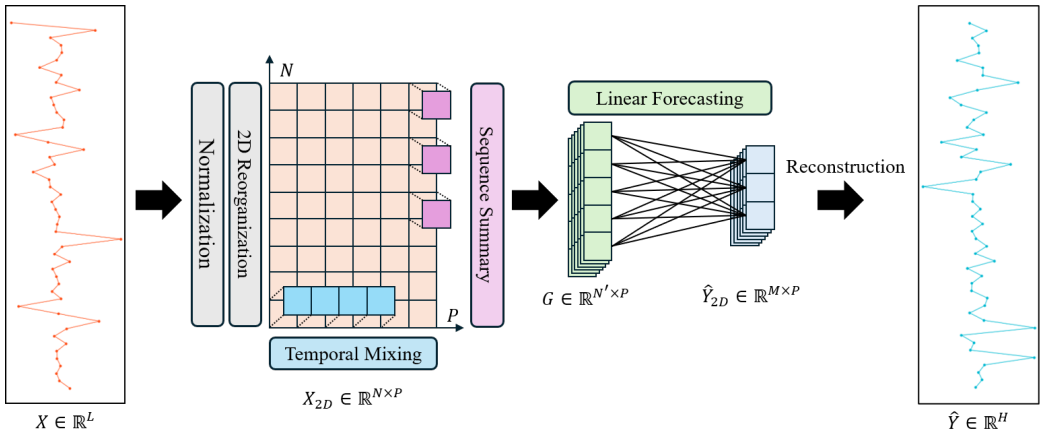

Figure 1: Overview of PRISM. The input sequence is first normalized and reorganized into a period aligned two dimensional representation that exposes cross period and intra period temporal structures. PRISM compactly models the two axes using different operations: sequence summarization compresses the cross period dimension, while parameter shared temporal mixing captures intra period dependencies without changing the period length. The resulting compact representation is mapped to future periods through a shared linear forecasting head and rearranged to produce the final forecast.

Given a predefined period length P, PRISM reorganizes the normalized one dimensional sequence into a period aligned two dimensional representation. We assume that P divides both the input length L and forecasting horizon H, and define

$$N = \frac{L}{P}, \qquad M = \frac{H}{P}, \tag{3}$$

where N and M denote the numbers of historical and future periods, respectively. The normalized input is then reshaped as

$$\mathbf{X}_{\text{2D}} \in \mathbb{R}^{N \times P}, \qquad \mathbf{X}_{\text{2D}}[n, p] = \tilde{x}_{(n-1)P+p}, \tag{4}$$

where $n \in \{1, \ldots, N\}$ indexes successive periods and $p \in \{1, \ldots, P\}$ indexes positions within each period.

This reorganization exposes two temporal axes with different structural interpretations. Along the cross period axis, observations at the same within period position are aligned across successive periods. Along the intra period axis, observations retain their temporal ordering within each period. PRISM subsequently assigns different computational operations to these two axes: sequence summarization is performed along the cross period axis, while temporal mixing is applied along the intra period axis. Importantly, this reorganization is not intended merely to expose periodicity; it reorganizes the original temporal structure into two complementary axes that admit different forms of compact modeling.

3.3 Sequence Summarization

After period based reorganization, PRISM summarizes the historical information along the cross period axis before forecasting. Given the period aligned representation $\mathbf{X}_{\text{2D}} \in \mathbb{R}^{N \times P}$, we apply a dilated one dimensional convolution independently to the cross period sequence associated with each intra period position. The same convolutional kernel is shared across all P positions.

Let k, s, and d denote the kernel size, stride, and dilation rate, respectively. The summarized representation is computed as

$$\mathbf{F}[n', p] = \sum_{i=0}^{k-1} w_i \cdot \mathbf{X}_{\text{2D}}[(n'-1) \cdot s + i \cdot d + 1, p], \tag{5}$$

where w_i denotes the i-th weight of the shared convolutional kernel. The resulting cross period length is

$$N' = \left\lfloor \frac{N - (k-1)d - 1}{s} \right\rfloor + 1, \tag{6}$$

and the summarized representation therefore has shape

$$\mathbf{F} \in \mathbb{R}^{N' \times P}. \tag{7}$$

Sequence summarization therefore reduces the cross-period dimension from N to N' before the forecasting stage. Since the same kernel is shared across intra period positions, this operation requires only k learnable kernel weights. The dilation rate d further expands the receptive field to

$$R = 1 + (k-1)d \tag{8}$$

cross period observations without increasing the number of kernel parameters. Together with the stride s, this provides a lightweight mechanism for controlling the degree of cross period summarization.

By performing this compression before forecasting, PRISM constructs a compact representation in which the subsequent forecasting head operates on N' cross period features rather than the original N observations.

3.4 Temporal Mixing and Forecasting

After sequence summarization, PRISM compactly models the intra period structure of the reduced representation. Given $\mathbf{F} \in \mathbb{R}^{N' \times P}$, we apply a one dimensional convolution independently to each summarized cross period feature while sharing the same kernel across all N' positions.

Let t denote the temporal mixing kernel size, which is chosen to be odd, and let $\mathbf{w}^{(t)} \in \mathbb{R}^t$ be the shared convolutional kernel. To preserve the intra-period length P, circular padding is applied along the period dimension.

$$\mathbf{G}[n', p] = \sum_{j=0}^{t-1} w_j^{(t)} \mathbf{F}\left[n', \left((p-1) + j - \left\lfloor \frac{t}{2} \right\rfloor\right) \bmod P + 1\right] \tag{9}$$

where $n' \in \{1, \ldots, N'\}$ and $p \in \{1, \ldots, P\}$. The resulting representation therefore preserves the same shape,

$$\mathbf{G} \in \mathbb{R}^{N' \times P}. \tag{10}$$

Temporal mixing allows neighboring positions within each period to exchange information after the cross period dimension has been compressed. Since the same kernel is shared across all N' summarized features, this stage introduces only t additional learnable parameters and does not increase the size of the representation passed to the forecasting head. Consequently, temporal mixing provides a parameter efficient representation of intra period dependencies while preserving the compact cross period structure obtained through sequence summarization. Although the intra period length P is preserved, parameter sharing allows this structure to be modeled with only t learnable kernel weights.

After temporal mixing, the compact representation $\mathbf{G} \in \mathbb{R}^{N' \times P}$ is mapped to $M = H/P$ future periods using a shared bias free linear layer. The final forecast is obtained as

$$\hat{\mathbf{y}} = \mathcal{R}(\mathbf{W}\mathbf{G}) + \mu, \qquad \mathbf{W} \in \mathbb{R}^{M \times N'}, \tag{11}$$

where $\mathbf{W}\mathbf{G} \in \mathbb{R}^{M \times P}$ represents the predicted future periods, $\mathcal{R}(\cdot)$ rearranges the period aligned representation into a sequence of length H, and μ is the input mean removed during normalization.

3.5 Parameter and Computational Complexity

PRISM contains k parameters for sequence summarization, t parameters for temporal mixing, and $N'M$ parameters for the shared forecasting head. Since the same parameters are shared across all C

variables, the parameter count is independent of C. For a C-variate input, the parameter count and computational cost are

$$\text{Params} = k + t + N'M, \qquad \text{MACs} = CN'P(k + t + M). \tag{12}$$

Thus, the forecasting head size and computational cost depend on the compressed cross period dimension N' rather than the original number of historical periods N.

For comparison, directly applying the same shared period level forecasting head to the uncompressed representation $X_{2\text{D}} \in \mathbb{R}^{N \times P}$ would require NM forecasting head parameters and $CPNM$ MACs for the forecasting projection. PRISM instead performs this projection after reducing the cross period dimension from N to N', requiring only $N'M$ forecasting head parameters and $CPN'M$ MACs for the forecasting projection. Therefore, sequence summarization reduces the forecasting head parameter and computational cost by a factor of approximately N/N', while requiring only k shared summarization parameters.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate PRISM on seven widely used LTSF benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity, and Traffic (Zhou et al., 2021), covering diverse temporal resolutions and dimensionalities; detailed dataset statistics and data splits are provided in Appendix A.4.

Baselines. We compare PRISM with six representative forecasting models: MixLinear (Ma et al., 2026), TimeBase (Huang et al., 2025), SparseTSF (Lin et al., 2024), FITS (Xu et al., 2024), PatchTST (Nie et al., 2023), and TimesNet (Wu et al., 2023). All baselines are evaluated under the same experimental framework using identical data splits and input sequence lengths. We follow the recommended configurations from the corresponding implementations or select hyperparameters based on validation performance when tuning is required.

Evaluation Metrics. Forecasting performance is evaluated using mean squared error (MSE) and mean absolute error (MAE), where lower values indicate better performance. Computational efficiency is evaluated using the number of trainable parameters, multiply accumulate operations (MACs), peak memory usage, training time per epoch, and inference latency on both GPU and CPU. All experiments are repeated three times independently, and the reported results are averaged over the three runs. Standard deviations for forecasting performance are provided in Appendix C.

Implementation Details. PRISM is implemented in PyTorch (Paszke et al., 2019), and all models are trained and evaluated under the same computational environment. The period length P is fixed as a dataset level structural prior as described in Appendix A.2. Model selection and the remaining hyperparameter configurations are performed using the validation set without reference to the test set. Complete PRISM configurations, optimization settings, and data split details are provided in Appendix A.

4.2 Forecasting Performance

Table 1 compares PRISM with six baselines across seven datasets and four forecasting horizons. PRISM performs particularly well on ETTh1 and remains competitive across the remaining benchmarks, including at the longest horizon $H = 720$.

Overall, these results show that PRISM preserves competitive forecasting accuracy across diverse datasets despite its highly compact architecture. Among the recent ultra lightweight baselines, PRISM compares particularly favorably with MixLinear and TimeBase. Across the 28 dataset horizon configurations in Table 1, PRISM achieves a lower MSE than MixLinear in 22 settings and the same MSE in one additional setting. Compared with TimeBase, PRISM obtains a lower MSE in 19 settings and matches it in three settings. These results indicate that the computational reductions of PRISM do not generally come at the expense of forecasting accuracy relative to recent lightweight alternatives. At the same time, we do not claim uniformly superior forecasting accuracy, as FITS

Table 1: Forecasting performance on seven benchmark datasets with input length $L = 720$ and forecasting horizons $H \in \{96, 192, 336, 720\}$. Results are averaged over three independent runs, and lower values indicate better performance. Standard deviations are reported in Appendix C.

Method		PRISM (Ours)		MixLinear		TimeBase		SparseTSF		FITS		PatchTST		TimesNet	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.352**	**0.382**	0.373	0.405	0.360	0.388	0.368	0.397	0.383	0.406	0.378	0.409	0.458	0.472
	192	**0.392**	**0.412**	0.397	0.418	0.403	**0.412**	0.403	**0.412**	0.418	0.427	0.416	0.434	0.481	0.486
	336	**0.409**	0.429	0.413	0.437	0.421	**0.422**	0.433	0.428	0.444	0.444	0.435	0.450	0.530	0.517
	720	**0.417**	**0.442**	0.424	0.454	0.449	0.454	0.425	0.450	0.435	0.458	0.461	0.478	0.620	0.568
ETTh2	96	0.281	0.342	0.284	0.342	0.295	0.353	0.282	0.342	**0.271**	**0.336**	0.275	0.340	0.383	0.418
	192	0.337	0.376	0.338	0.376	0.346	0.388	0.338	0.377	**0.331**	**0.375**	0.339	0.382	0.466	0.476
	336	0.360	0.399	0.357	0.396	0.382	0.417	0.358	0.397	0.354	0.396	**0.329**	**0.387**	0.429	0.453
	720	0.390	0.430	0.382	0.424	0.411	0.458	0.380	**0.423**	**0.378**	**0.423**	0.387	0.430	0.511	0.507
ETTm1	96	0.312	0.355	0.333	0.374	0.316	0.359	0.312	0.353	0.310	0.352	**0.297**	**0.350**	0.347	0.382
	192	0.341	0.370	0.353	0.382	0.345	0.376	0.338	0.369	0.338	**0.368**	**0.333**	0.372	0.510	0.458
	336	0.368	0.386	0.391	0.407	0.373	0.394	0.367	**0.385**	0.366	**0.385**	**0.363**	0.391	0.418	0.433
	720	0.416	0.413	0.435	0.431	**0.415**	0.416	**0.415**	**0.411**	**0.415**	0.412	0.416	0.419	0.465	0.462
ETTm2	96	**0.160**	0.257	0.171	0.262	0.166	0.257	0.165	0.255	0.162	**0.253**	0.163	0.254	0.203	0.289
	192	0.219	0.293	0.224	0.299	0.218	0.293	0.218	0.293	**0.217**	**0.291**	**0.217**	0.294	0.279	0.338
	336	0.271	0.329	0.276	0.333	0.271	0.328	0.270	0.328	0.268	**0.326**	**0.267**	0.327	0.341	0.380
	720	0.352	0.380	0.359	0.384	0.352	0.382	0.350	**0.378**	**0.349**	**0.378**	**0.349**	0.379	0.424	0.429
Weather	96	0.168	0.221	0.171	0.226	0.146	0.202	0.172	0.226	**0.143**	**0.193**	0.149	0.200	0.176	0.234
	192	0.213	0.261	0.215	0.264	0.187	0.242	0.213	0.261	**0.185**	**0.236**	0.194	0.241	0.219	0.270
	336	0.259	0.296	0.258	0.296	0.237	0.283	0.258	0.294	**0.236**	**0.276**	0.245	0.282	0.277	0.311
	720	0.320	0.339	0.322	0.342	0.308	0.333	0.321	0.339	**0.307**	**0.329**	0.314	0.334	0.344	0.356
Electricity	96	0.140	0.235	0.139	0.233	0.140	0.233	0.138	0.233	0.134	0.231	**0.129**	**0.222**	0.183	0.289
	192	0.152	0.245	0.154	0.248	0.156	0.248	0.151	0.244	0.149	0.244	**0.147**	**0.240**	0.218	0.322
	336	0.167	0.260	0.171	0.265	0.170	0.263	0.166	0.260	0.165	0.261	**0.163**	**0.259**	0.232	0.332
	720	0.205	0.293	0.205	0.297	0.209	0.295	0.205	0.293	0.204	0.293	**0.197**	**0.290**	0.299	0.375
Traffic	96	0.393	0.268	0.390	0.268	0.395	0.268	0.389	0.267	0.386	0.269	**0.360**	**0.249**	0.605	0.325
	192	0.401	0.271	0.406	0.278	0.405	0.274	0.399	0.271	0.397	0.272	**0.379**	**0.256**	0.627	0.340
	336	0.413	0.277	0.418	0.284	0.419	0.282	0.411	0.276	0.410	0.278	**0.392**	**0.264**	0.631	0.349
	720	0.451	0.297	0.452	0.301	0.457	0.301	0.449	0.297	0.449	0.299	**0.432**	**0.286**	0.700	0.371

and PatchTST achieve lower errors on several benchmarks. Rather than emphasizing forecasting accuracy in isolation, we consider these results together with the computational requirements of each model. The following section therefore examines whether this predictive performance is accompanied by corresponding gains in computational efficiency.

4.3 Efficiency Analysis

We further evaluate the practical efficiency of PRISM on Electricity with $L = H = 720$. We report the number of trainable parameters, MACs, peak memory usage, training time per epoch, and inference latency on both GPU and CPU. Parameters and MACs characterize architecture-level complexity, while memory usage and wall-clock measurements reflect practical execution behavior.

As shown in Table 2, PRISM requires only 2.70M MACs and 487.48 MB of peak memory, exhibiting a consistently small computational footprint across both theoretical and practical efficiency measures. It also maintains low inference latency on both GPU and CPU, although SparseTSF achieves lower GPU latency and TimeBase achieves slightly lower CPU latency.

The efficiency gains are most evident in computational cost and memory usage. Compared with TimeBase, the strongest baseline in terms of MACs and peak memory, PRISM reduces MACs by 11.5% and peak memory usage by 50.4%. PRISM also maintains competitive training and inference speed, although SparseTSF achieves lower GPU latency and TimeBase achieves slightly lower CPU latency. Overall, these results demonstrate that PRISM achieves a favorable efficiency profile

Table 2: Efficiency comparison on Electricity with $L = H = 720$. Lower values indicate better efficiency. The best result is shown in bold and the second-best result is underlined.

Method	Params	MACs	Peak Memory	Epoch Time	GPU Latency	CPU Latency
PRISM (Ours)	<u>279</u>	**2.70M**	**487.48**	<u>33.11</u>	<u>0.44</u>	<u>1.36</u>
MixLinear	**187**	9.80M	1143.65	35.22	1.66	5.36
TimeBase	396	<u>3.05M</u>	<u>982.51</u>	**31.41**	0.66	**1.34**
SparseTSF	925	12.71M	3183.89	33.29	**0.23**	1.55
FITS	205.44K	262.95M	2861.02	67.56	4.37	10.31
PatchTST	8.71M	16.08G	25470	1235.73	25.07	438
TimesNet	151.32M	1.08T	6226.58	14943.98	182.48	17364.23

Table 3: Ablation study on forecasting performance at $H = 720$. The best result is shown in bold and the second-best result is underlined.

Variant	ETTh1		Electricity		Traffic	
	MSE	MAE	MSE	MAE	MSE	MAE
PRISM	**0.417**	**0.442**	**0.205**	**0.293**	<u>0.451</u>	<u>0.297</u>
w/o Period Reorganization	0.514	0.512	**0.205**	<u>0.294</u>	<u>0.451</u>	0.300
w/o Sequence Summarization	0.425	0.458	<u>0.206</u>	<u>0.294</u>	**0.447**	**0.294**
w/o Temporal Mixing	<u>0.423</u>	<u>0.449</u>	0.208	0.295	0.454	<u>0.297</u>

Table 4: Ablation study on computational efficiency at $H = 720$. The best result is shown in bold and the second-best result is underlined.

Variant	ETTh1		Electricity		Traffic	
	Params	MACs	Params	MACs	Params	MACs
PRISM	<u>97</u>	<u>22.51K</u>	<u>279</u>	<u>2.70M</u>	<u>279</u>	<u>7.26M</u>
w/o Period Reorganization	32.47K	238.46K	503.29K	163.57M	503.29K	439.25M
w/o Sequence Summarization	923	267.12K	905	8.09M	905	21.72M
w/o Temporal Mixing	**74**	**14.78K**	**274**	**2.38M**	**274**	**6.33M**

without relying on a single metric. Notably, PRISM also matches the MSE of MixLinear under this setting while requiring substantially fewer MACs and lower memory usage.

4.4 Ablation Study

To examine the contribution of each architectural component, we conduct ablation studies on ETTh1, Electricity, and Traffic at the longest forecasting horizon, $H = 720$. We remove period reorganization, sequence summarization, and temporal mixing individually, and evaluate both forecasting performance and computational complexity. Tables 3 and 4 summarize the results.

The ablation results confirm the distinct roles of the three components. Removing period reorganization causes the largest increase in parameters and MACs without providing a consistent accuracy benefit, while removing sequence summarization similarly increases complexity because the forecasting head operates on the original cross period dimension N rather than N'. For example, on Traffic, sequence summarization reduces MACs from 21.72M to 7.26M, corresponding to a 66.6% reduction in computation. Temporal mixing contributes relatively little additional computation but generally improves or preserves forecasting accuracy. Overall, the results support PRISM's axis specific design: period reorganization enables axis specific computation, sequence summarization provides the primary dimensional reduction, and temporal mixing offers parameter efficient modeling of complementary intra-period structure.

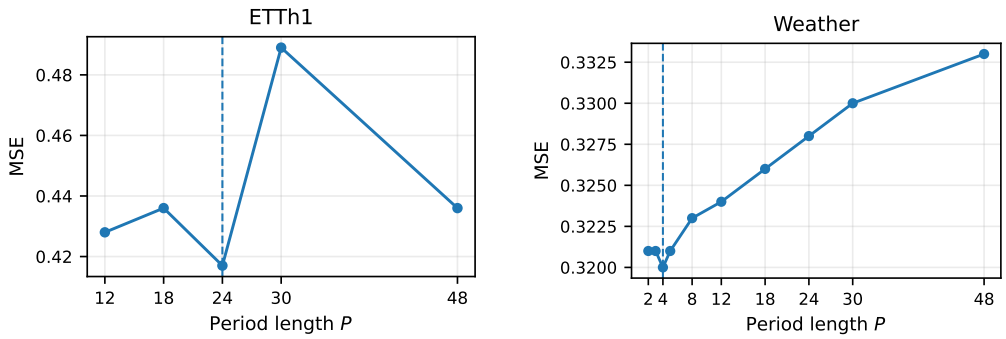

Figure 2: Sensitivity of PRISM to the period length P on ETTh1 (left) and Weather (right) with $H = 720$. The dashed vertical line indicates the default period used in the main experiments. Lower MSE indicates better forecasting performance.

4.5 Sensitivity to Period Selection

PRISM relies on a predefined period length P to construct the period aligned representation. Following prior period aware lightweight forecasting methods (Lin et al., 2024; Huang et al., 2025), we use the natural daily granularity for hourly sampled datasets ($P = 24$) and a smaller granularity for more densely sampled datasets ($P = 4$). These period settings are treated as dataset level structural priors and are fixed before test evaluation rather than selected according to test performance.

To examine the sensitivity of PRISM to this prior, we vary P while keeping the remaining configuration fixed and evaluate ETTh1 and Weather at $H = 720$. In this analysis, the remaining architectural parameters, including the temporal mixing kernel size t, are kept fixed so that the effect of changing P can be examined in isolation. The period length P determines the alignment granularity used to construct the period aligned representation and is not required to coincide exactly with a dominant physical or seasonal period. The sensitivity experiment therefore evaluates how deviations from the adopted structural prior affect forecasting performance rather than selecting P using test results. Figure 2 summarizes the results.

The effect of period selection is dataset dependent. ETTh1 is more sensitive to particular choices of P, although several alternative values still provide comparable performance to the default setting. In contrast, Weather exhibits a smoother response and remains relatively stable over a broader range of period values. These results indicate that PRISM does not exhibit the same degree of sensitivity to the alignment granularity across datasets, but an appropriate choice of P can still be important when forecasting performance is more sensitive to this structural prior.

5 Conclusion

We introduced PRISM, a lightweight LTSF architecture that uses period alignment to expose complementary cross period and intra period structures and models them with compact axis specific operators. Cross period summarization reduces the representation before forecasting, while parameter shared temporal mixing captures intra period dependencies with minimal additional complexity. Across seven benchmarks and four horizons, PRISM achieves competitive forecasting accuracy with a small computational footprint, demonstrating the effectiveness of period aligned compact modeling for efficient LTSF.

PRISM relies on a predefined period length P, and its sensitivity to this prior varies across datasets. Automatic or multi period extensions are therefore a natural direction for future work. In addition, practical efficiency is measured in detail under a representative Electricity setting with $L = H = 720$; broader evaluation across sequence lengths and hardware platforms would provide a more complete runtime characterization.

AI USE STATEMENT

Generative AI tools were used solely to assist with manuscript writing and editing, including language refinement, clarity improvement, and formatting. They were not used to generate research ideas, design experiments, conduct analyses, produce results, or make scientific conclusions.

REPRODUCIBILITY STATEMENT

The forecasting task, model operations, and analytical complexity are specified in Section 3, including all tensor shapes and the period divisibility assumption. Section 4 describes the datasets, baselines, evaluation metrics, and efficiency measurements. The appendix provides detailed training and data-split settings, dataset specific PRISM configurations, the PRISM forward algorithm, complete repeated run results, and additional analyses of dilation and temporal mixing. The anonymized implementation and experiment scripts are available at `https://anonymous.4open.science/r/PRISM-2DA2`.

REFERENCES

Qihe Huang, Zhengyang Zhou, Kuo Yang, Zhongchao Yi, Xu Wang, and Yang Wang. TimeBase: The power of minimalism in efficient long-term time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26227–26246. PMLR, 2025.

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

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*, pp. 30211–30226. PMLR, 2024.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5816–5828, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *International Conference on Learning Representations*, 2024.

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.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

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

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, pp. 11121–11128, 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. doi: 10.1609/aaai.v35i12.17325.

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*, pp. 27268–27286. PMLR, 2022.

A Detailed Experimental Settings

This section provides additional details on the training configuration, PRISM specific settings, baseline configurations, and dataset splits used in our experiments.

A.1 Training Configuration

PRISM is optimized using Adam (Kingma & Ba, 2015) with mean squared error (MSE) as the training objective. The learning rate is initialized at 0.02 and adjusted during training using the learning rate scheduler. The batch size is selected based on validation performance and is set to 32 for Weather, 256 for ETTh1 and ETTh2, and 64 for ETTm1, ETTm2, Electricity, and Traffic. PRISM is trained for up to 30 epochs with early stopping using a patience of 5 epochs. All forecasting experiments use an input sequence length of $L = 720$, and each experiment is repeated independently three times.

A.2 PRISM Configurations

PRISM uses the period length P as a dataset level structural prior, together with sequence summarization parameters k, s, and d, and temporal mixing kernel size t as architectural configurations. Following the general period setting principle adopted in prior period aware lightweight forecasting methods (Lin et al., 2024; Huang et al., 2025), P is fixed without reference to test performance rather than tuned as a conventional model hyperparameter.

For hourly sampled datasets with a clear daily structure, we use $P = 24$, corresponding to the natural daily granularity. For more densely sampled datasets, directly using the full daily period would produce a very small number of cross period observations $N = L/P$. Following the lightweight period based practice of using a smaller granularity for such long natural periods, we therefore use $P = 4$ for ETTm1, ETTm2, and Weather. Here, P should be interpreted as the alignment granularity used by PRISM and is not required to exactly match a physical seasonal period.

Accordingly, we set $P = 24$ for ETTh1, ETTh2, Electricity, and Traffic, and $P = 4$ for ETTm1, ETTm2, and Weather. The same P is fixed across all forecasting horizons of each dataset without reference to test set forecasting performance.

The remaining architectural configurations are selected using the validation set and are fixed for all forecasting horizons within each dataset. Specifically, (k, s, d, t) is set to $(14, 16, 1, 23)$ for ETTh1, $(20, 10, 1, 23)$ for ETTh2, $(3, 1, 7, 1)$ for ETTm1, $(3, 1, 12, 3)$ for ETTm2, $(8, 4, 1, 3)$ for Weather, $(4, 1, 7, 5)$ for Electricity, and $(4, 1, 7, 5)$ for Traffic.

A.3 Baseline Configurations

For the baseline models, we follow the configurations recommended in their corresponding papers and implementations. MixLinear, TimeBase, SparseTSF, FITS, PatchTST, and TimesNet are evaluated under the same forecasting setup using an input sequence length of $L = 720$ and the same dataset splits as PRISM. Test data are not used for model or hyperparameter selection.

A.4 Dataset Statistics and Splits

Table 5 summarizes the statistics of the seven benchmark datasets. All datasets are divided chronologically into training, validation, and test sets while preserving temporal order. For ETTh1, ETTh2, ETTm1, and ETTm2, we use a train–validation–test ratio of 6:2:2. For Weather, Electricity, and Traffic, the corresponding ratio is 7:1:2.

Table 5: Statistics of the benchmark datasets.

Dataset	Channels	Sampling	Time Steps
ETTh1	7	1 hour	17,420
ETTh2	7	1 hour	17,420
ETTm1	7	15 min	69,680
ETTm2	7	15 min	69,680
Weather	21	10 min	52,696
Electricity	321	1 hour	26,304
Traffic	862	1 hour	17,544

A.5 Efficiency Measurement Protocol

Peak memory is measured as the maximum allocated GPU memory observed during training. Training time per epoch is measured as the wall clock time required to process the complete training set for one epoch, excluding validation and test evaluation. Inference latency is measured with a batch size of one using 10 warm up forward passes followed by 100 timed forward passes. For GPU latency, CUDA synchronization is performed immediately before and after the timed iterations to account for asynchronous execution.

B PRISM Algorithm

Algorithm 1 summarizes the forward procedure of PRISM described in Section 3. Given an input sequence $\mathbf{X} \in \mathbb{R}^{B\times L\times C}$, PRISM first removes the temporal mean of each variable and reorganizes the normalized sequence according to the period length P. This produces $N = L/P$ historical periods, where observations at the same intra period position are aligned along the cross period dimension. The same operations are applied independently to all variables using shared parameters.

The period aligned representation is first compressed along the cross-period dimension using the sequence summarization operator with kernel size k, stride s, and dilation d. The resulting dimension is $N' = \left\lfloor \frac{N-d(k-1)-1}{s} \right\rfloor + 1$. Temporal mixing is then applied along the intra period dimension using circular padding, after which the shared linear forecasting head maps the compressed dimension from N' to $M = H/P$ future periods. Finally, the predicted periods are rearranged into a sequence of length H, and the input mean is restored to obtain the final forecast.

C Full Forecasting Results

Tables 6 and 7 report the complete forecasting results over three independent runs for all seven datasets and four prediction horizons. We present MSE and MAE as mean $\pm$ standard deviation, complementing the mean results reported in Table 1.

Algorithm 1 Forward Procedure of PRISM

Require: Input $\mathbf{X} \in \mathbb{R}^{B \times L \times C}$, forecast horizon H, period P, summarization parameters (k, s, d), temporal kernel size t

Ensure: Forecast $\hat{\mathbf{Y}} \in \mathbb{R}^{B \times H \times C}$

$N \leftarrow L/P$, $M \leftarrow H/P$

$\boldsymbol{\mu} \leftarrow \mathrm{Mean}(\mathbf{X}, \mathrm{dim} = 1)$

$\tilde{\mathbf{X}} \leftarrow \mathbf{X} - \boldsymbol{\mu}$ `// Normalization`

$\mathbf{X}_{2\mathrm{D}} \leftarrow \mathrm{PeriodReorganize}(\tilde{\mathbf{X}}, P)$ `// Period reorganization`

$N' \leftarrow \left\lfloor \dfrac{N - d(k-1) - 1}{s} \right\rfloor + 1$

$\mathbf{F} \leftarrow \mathrm{Conv1D}_{k,s,d}(\mathbf{X}_{2\mathrm{D}}, \text{cross-period})$ `// Sequence summarization`

$\mathbf{G} \leftarrow \mathrm{CircularConv1D}_{t}(\mathbf{F}, \text{intra-period})$ `// Temporal mixing`

$\mathbf{Y}_P \leftarrow \mathrm{Linear}_{N' \to M}(\mathbf{G})$ `// Forecast future periods`

$\hat{\mathbf{Y}} \leftarrow \mathrm{Reconstruct}(\mathbf{Y}_P)$ `// Sequence reconstruction`

$\hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}} + \boldsymbol{\mu}$ `// Denormalization`

return $\hat{\mathbf{Y}}$

Table 6: Full MSE results over three independent runs. Results are reported as mean ± standard deviation. Lower values indicate better performance. The best mean result is shown in bold and the second-best mean result is underlined.

Dataset	H	PRISM (Ours)	MixLinear	TimeBase	SparseTSF	FITS	PatchTST	TimesNet
ETTh1	96	**0.352** ± 0.002	0.373 ± 0.016	0.360 ± 0.000	0.368 ± 0.008	0.383 ± 0.000	0.378 ± 0.003	0.458 ± 0.009
	192	**0.392** ± 0.006	0.397 ± 0.002	0.403 ± 0.007	0.403 ± 0.001	0.418 ± 0.000	0.416 ± 0.003	0.481 ± 0.012
	336	**0.409** ± 0.005	0.413 ± 0.002	0.421 ± 0.009	0.433 ± 0.000	0.444 ± 0.001	0.435 ± 0.005	0.530 ± 0.025
	720	**0.417** ± 0.002	0.424 ± 0.001	0.449 ± 0.007	0.425 ± 0.000	0.435 ± 0.001	0.461 ± 0.002	0.620 ± 0.077
ETTh2	96	0.281 ± 0.001	0.284 ± 0.001	0.295 ± 0.002	0.282 ± 0.001	**0.271** ± 0.000	0.275 ± 0.000	0.383 ± 0.021
	192	0.337 ± 0.001	0.338 ± 0.002	0.346 ± 0.002	0.338 ± 0.000	**0.331** ± 0.000	0.339 ± 0.000	0.466 ± 0.023
	336	0.360 ± 0.002	0.357 ± 0.002	0.382 ± 0.005	0.358 ± 0.001	0.354 ± 0.000	**0.329** ± 0.000	0.429 ± 0.017
	720	0.390 ± 0.008	0.382 ± 0.002	0.411 ± 0.010	0.380 ± 0.000	**0.378** ± 0.000	0.387 ± 0.001	0.511 ± 0.050
ETTm1	96	0.312 ± 0.002	0.333 ± 0.001	0.316 ± 0.004	0.312 ± 0.000	0.310 ± 0.000	**0.297** ± 0.002	0.347 ± 0.012
	192	0.341 ± 0.003	0.353 ± 0.000	0.345 ± 0.008	0.338 ± 0.000	0.338 ± 0.000	**0.333** ± 0.001	0.510 ± 0.009
	336	0.368 ± 0.001	0.391 ± 0.002	0.373 ± 0.007	0.367 ± 0.001	0.366 ± 0.000	**0.363** ± 0.002	0.418 ± 0.013
	720	0.416 ± 0.001	0.435 ± 0.005	**0.415** ± 0.002	**0.415** ± 0.001	**0.415** ± 0.000	0.416 ± 0.003	0.465 ± 0.024
ETTm2	96	**0.160** ± 0.006	0.171 ± 0.000	0.166 ± 0.000	0.165 ± 0.001	0.162 ± 0.000	0.163 ± 0.001	0.203 ± 0.007
	192	0.219 ± 0.002	0.224 ± 0.001	0.218 ± 0.000	0.218 ± 0.001	**0.217** ± 0.000	**0.217** ± 0.000	0.279 ± 0.015
	336	0.271 ± 0.002	0.276 ± 0.000	0.271 ± 0.000	0.270 ± 0.000	0.268 ± 0.000	**0.267** ± 0.001	0.341 ± 0.013
	720	0.352 ± 0.001	0.359 ± 0.001	0.352 ± 0.000	0.350 ± 0.000	**0.349** ± 0.000	**0.349** ± 0.001	0.424 ± 0.020
Weather	96	0.168 ± 0.001	0.171 ± 0.001	0.146 ± 0.001	0.172 ± 0.001	**0.143** ± 0.000	0.149 ± 0.003	0.176 ± 0.005
	192	0.213 ± 0.002	0.215 ± 0.001	0.187 ± 0.002	0.213 ± 0.000	**0.185** ± 0.000	0.194 ± 0.003	0.219 ± 0.003
	336	0.259 ± 0.001	0.258 ± 0.002	0.237 ± 0.000	0.258 ± 0.001	**0.236** ± 0.000	0.245 ± 0.002	0.277 ± 0.007
	720	0.320 ± 0.001	0.322 ± 0.002	0.308 ± 0.000	0.321 ± 0.001	**0.307** ± 0.000	0.314 ± 0.002	0.344 ± 0.010
Electricity	96	0.140 ± 0.000	0.139 ± 0.000	0.140 ± 0.000	0.138 ± 0.000	0.134 ± 0.000	**0.129** ± 0.000	0.183 ± 0.003
	192	0.152 ± 0.000	0.154 ± 0.000	0.156 ± 0.001	0.151 ± 0.000	0.149 ± 0.000	**0.147** ± 0.000	0.218 ± 0.002
	336	0.167 ± 0.000	0.171 ± 0.000	0.170 ± 0.000	0.166 ± 0.000	0.165 ± 0.000	**0.163** ± 0.000	0.232 ± 0.003
	720	0.205 ± 0.000	0.205 ± 0.000	0.209 ± 0.000	0.205 ± 0.000	0.204 ± 0.000	**0.197** ± 0.000	0.299 ± 0.003
Traffic	96	0.393 ± 0.001	0.390 ± 0.000	0.395 ± 0.001	0.389 ± 0.000	0.386 ± 0.000	**0.360** ± 0.001	0.605 ± 0.007
	192	0.401 ± 0.000	0.406 ± 0.002	0.405 ± 0.002	0.399 ± 0.000	0.397 ± 0.000	**0.379** ± 0.000	0.627 ± 0.010
	336	0.413 ± 0.000	0.418 ± 0.002	0.419 ± 0.002	0.411 ± 0.000	0.410 ± 0.000	**0.392** ± 0.000	0.631 ± 0.005
	720	0.451 ± 0.001	0.452 ± 0.000	0.457 ± 0.000	0.449 ± 0.002	0.449 ± 0.000	**0.432** ± 0.000	0.700 ± 0.004

Table 7: Full MAE results over three independent runs. Results are reported as mean ± standard deviation. Lower values indicate better performance. The best mean result is shown in bold and the second-best mean result is underlined.

Dataset	H	PRISM (Ours)	MixLinear	TimeBase	SparseTSF	FITS	PatchTST	TimesNet
ETTh1	96	**0.382** ± 0.001	0.405 ± 0.016	0.388 ± 0.004	0.397 ± 0.012	0.406 ± 0.000	0.409 ± 0.002	0.472 ± 0.010
	192	**0.412** ± 0.012	0.418 ± 0.008	**0.412** ± 0.006	**0.412** ± 0.000	0.427 ± 0.000	0.434 ± 0.002	0.486 ± 0.006
	336	0.429 ± 0.007	0.437 ± 0.003	**0.422** ± 0.003	0.428 ± 0.000	0.444 ± 0.001	0.450 ± 0.003	0.517 ± 0.013
	720	**0.442** ± 0.006	0.454 ± 0.005	0.454 ± 0.006	0.450 ± 0.006	0.458 ± 0.000	0.478 ± 0.001	0.568 ± 0.038
ETTh2	96	0.342 ± 0.001	0.342 ± 0.002	0.353 ± 0.002	0.342 ± 0.000	**0.336** ± 0.000	0.340 ± 0.000	0.418 ± 0.010
	192	0.376 ± 0.001	0.376 ± 0.000	0.388 ± 0.001	0.377 ± 0.000	**0.375** ± 0.000	0.382 ± 0.000	0.476 ± 0.014
	336	0.399 ± 0.002	0.396 ± 0.001	0.417 ± 0.001	0.397 ± 0.000	0.396 ± 0.000	**0.387** ± 0.001	0.453 ± 0.012
	720	0.430 ± 0.007	0.424 ± 0.002	0.458 ± 0.009	**0.423** ± 0.001	**0.423** ± 0.000	0.430 ± 0.001	0.507 ± 0.028
ETTm1	96	0.355 ± 0.001	0.374 ± 0.001	0.359 ± 0.003	0.353 ± 0.000	0.352 ± 0.000	**0.350** ± 0.001	0.382 ± 0.005
	192	0.370 ± 0.001	0.382 ± 0.000	0.376 ± 0.007	0.369 ± 0.001	**0.368** ± 0.000	0.372 ± 0.001	0.458 ± 0.001
	336	0.386 ± 0.001	0.407 ± 0.002	0.394 ± 0.006	**0.385** ± 0.000	**0.385** ± 0.000	0.391 ± 0.000	0.433 ± 0.008
	720	0.413 ± 0.001	0.431 ± 0.005	0.416 ± 0.002	**0.411** ± 0.000	0.412 ± 0.000	0.419 ± 0.001	0.462 ± 0.017
ETTm2	96	0.257 ± 0.003	0.262 ± 0.000	0.257 ± 0.000	0.255 ± 0.000	**0.253** ± 0.000	0.254 ± 0.001	0.289 ± 0.004
	192	0.293 ± 0.002	0.299 ± 0.001	0.293 ± 0.000	0.293 ± 0.001	**0.291** ± 0.000	0.294 ± 0.001	0.338 ± 0.010
	336	0.329 ± 0.002	0.333 ± 0.000	0.328 ± 0.000	0.328 ± 0.000	**0.326** ± 0.000	0.327 ± 0.000	0.380 ± 0.010
	720	0.380 ± 0.001	0.384 ± 0.000	0.382 ± 0.001	**0.378** ± 0.000	**0.378** ± 0.000	0.379 ± 0.001	0.429 ± 0.011
Weather	96	0.221 ± 0.002	0.226 ± 0.001	0.202 ± 0.002	0.226 ± 0.002	**0.193** ± 0.000	0.200 ± 0.003	0.234 ± 0.004
	192	0.261 ± 0.003	0.264 ± 0.002	0.242 ± 0.002	0.261 ± 0.000	**0.236** ± 0.000	0.241 ± 0.002	0.270 ± 0.003
	336	0.296 ± 0.001	0.296 ± 0.003	0.283 ± 0.001	0.294 ± 0.000	**0.276** ± 0.000	0.282 ± 0.002	0.311 ± 0.005
	720	0.339 ± 0.001	0.342 ± 0.002	0.333 ± 0.000	0.339 ± 0.000	**0.329** ± 0.000	0.334 ± 0.001	0.356 ± 0.008
Electricity	96	0.235 ± 0.001	0.233 ± 0.000	0.233 ± 0.000	0.233 ± 0.000	0.231 ± 0.000	**0.222** ± 0.001	0.289 ± 0.003
	192	0.245 ± 0.000	0.248 ± 0.000	0.248 ± 0.001	0.244 ± 0.000	0.244 ± 0.000	**0.240** ± 0.001	0.322 ± 0.004
	336	0.260 ± 0.001	0.265 ± 0.001	0.263 ± 0.000	0.260 ± 0.000	0.261 ± 0.000	**0.259** ± 0.000	0.332 ± 0.003
	720	0.293 ± 0.000	0.297 ± 0.000	0.295 ± 0.000	0.293 ± 0.000	0.293 ± 0.000	**0.290** ± 0.001	0.375 ± 0.003
Traffic	96	0.268 ± 0.000	0.268 ± 0.000	0.268 ± 0.000	0.267 ± 0.000	0.269 ± 0.000	**0.249** ± 0.001	0.325 ± 0.007
	192	0.271 ± 0.001	0.278 ± 0.002	0.274 ± 0.000	0.271 ± 0.000	0.272 ± 0.000	**0.256** ± 0.000	0.340 ± 0.013
	336	0.277 ± 0.001	0.284 ± 0.003	0.282 ± 0.001	0.276 ± 0.000	0.278 ± 0.000	**0.264** ± 0.001	0.349 ± 0.004
	720	0.297 ± 0.000	0.301 ± 0.000	0.301 ± 0.001	0.297 ± 0.002	0.299 ± 0.000	**0.286** ± 0.001	0.371 ± 0.004

Across the evaluated settings, PRISM exhibits relatively small variation across independent runs. The standard deviation of MSE remains at or below 0.008, while that of MAE remains at or below 0.012 across all datasets and forecasting horizons. In particular, the variation is very small on large multivariate datasets such as Electricity and Traffic, where most standard deviations are close to zero. These results indicate that the forecasting performance of PRISM is generally consistent across repeated runs.

D Additional Analysis

This section provides additional analyses of two architectural hyperparameters in PRISM. We first examine how the dilation factor d in sequence summarization affects the trade off between forecasting accuracy and computational cost. We then investigate the effect of the temporal mixing kernel size t across datasets.

D.1 Effect of Dilation

Figure 3 examines the effect of the dilation factor d on Electricity at $H = 720$ in terms of both forecasting accuracy and computational cost. As d increases, the number of MACs decreases substantially, while the forecasting error remains relatively stable for $d \in \{1, 3, 5, 7\}$. In particular, increasing the dilation from 1 to 7 considerably reduces the computational cost without a corresponding degradation in MSE. However, further increasing the dilation to $d = 9$ results in a noticeable increase in forecasting error despite the additional reduction in MACs. These results show that dilation provides a direct mechanism for reducing the computational cost of sequence summarization, while an excessively large dilation may discard information required for accurate forecasting. A moderate dilation therefore provides a favorable balance between forecasting performance and computational efficiency.

D.2 Effect of Temporal Mixing Kernel Size

Figure 4 investigates the effect of the temporal mixing kernel size t on ETTh1 and Traffic at $H = 720$, and on Electricity at $H = 96$. Here, $t = 0$ denotes the configuration without temporal mixing. The effect of t is dataset dependent, but forecasting performance remains relatively stable over a broad range of kernel sizes in most settings. On ETTh1, MSE varies only slightly for most values of t, although particular kernel sizes produce larger deviations. Electricity exhibits a similarly stable response across a wide range of kernel sizes before showing a substantial degradation at $t = 23$. On Traffic, introducing temporal mixing reduces MSE compared with $t = 0$, and the performance becomes nearly unchanged once the kernel size reaches $t = 3$.

Overall, the results indicate that the temporal mixing module generally does not require a narrowly specified kernel size to maintain its effectiveness. Instead, a range of kernel sizes can provide comparable forecasting performance, although the effect of very large kernels depends on the dataset.

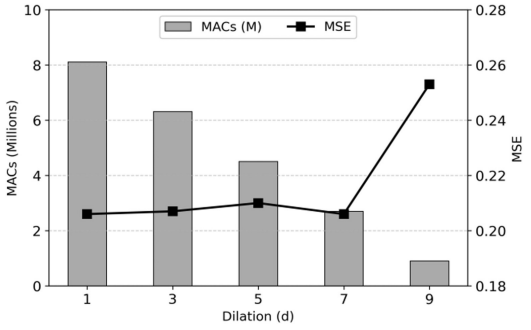

Figure 3: Effect of the dilation factor d on forecasting accuracy and computational cost. Bars indicate MACs, while the line indicates MSE. Increasing d reduces computational cost, whereas an excessively large dilation can degrade forecasting performance.

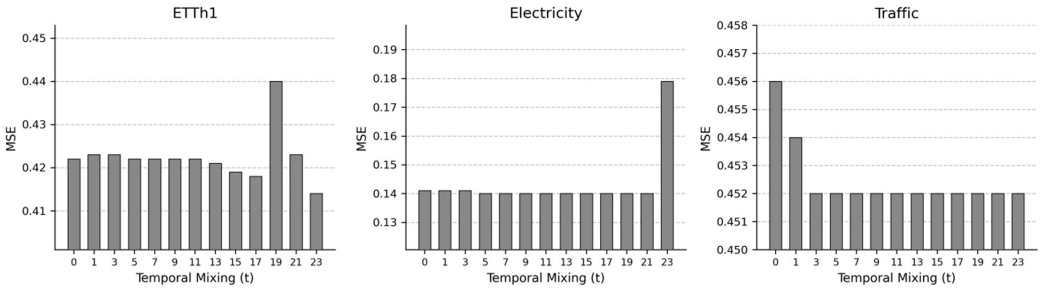

Figure 4: Effect of the temporal mixing kernel size t on ETTh1 and Traffic at $H = 720$, and on Electricity at $H = 96$. The case $t = 0$ denotes the configuration without temporal mixing. Lower MSE indicates better forecasting performance.

Together with the ablation results in Section 4.4, this analysis supports the role of temporal mixing as a lightweight refinement operation rather than the primary source of computational complexity in PRISM.