FLAG: Factor-Lag Relational Context for Adaptive Group Communication in Multivariate Time Series Forecasting

Anonymous authors
Paper under double-blind review

Abstract

Effectively modeling and using cross-variable dependencies is an important route to better multivariate time series forecasting. Prior work has explored several mechanisms for interaction among variables, but it remains challenging to represent lagged relations compactly while adjusting how those relations are used according to the current input. In particular, cross-variable dependencies can recur across observation windows, yet their contribution to a forecast depends on the current state. If this difference is not taken into account, information aggregation can mix cross-variable information that is useful for the current input with information that is not. To address this problem, we propose **FLAG**, a unified framework that jointly models lagged relations and performs input-adaptive information exchange within the same representation. FLAG first combines shared low-rank latent factors with lag-specific factor interactions, incorporating other variables' histories into each variable's representation and forming a relational context. This design retains lag-specific interaction detail without constructing dense variable-pair matrices. Group communication then produces input-dependent aggregation weights and messages from this context and updates the same representation, so that the shared relations take part in the forecast according to the current state. With the other model and temporal dimensions held fixed, both the number of parameters and the computation grow linearly with the number of variables. Experiments on eight benchmarks show that FLAG achieves competitive forecasting performance. Code is available at `https://anonymous.4open.science/r/FLAG-DE06`.

1 Introduction

Multivariate time series forecasting is widely used in transportation, energy management, and weather monitoring (Jiang & Luo, 2022; Yin et al., 2023; Lam et al., 2023). Besides learning each variable's own temporal behavior, effectively modeling and using cross-variable dependencies is an important route to better forecasting performance (Lai et al., 2018). Prior work has explored several ways to model this problem. PatchTST learns temporal representations with a channel-independent shared model (Nie et al., 2023). Crossformer and iTransformer exchange information through routers and inter-variable attention, respectively (Zhang & Yan, 2023; Liu et al., 2024). SOFTS aggregates variable information into a shared core and redistributes it (Han et al., 2024). These methods have advanced cross-variable modeling and shift the question toward how to retain useful relational information within a compact exchange mechanism, and how to use that information according to the current input.

This question has two related aspects. First, cross-variable dependencies can occur at different temporal offsets (Zhao & Shen, 2024; Ma et al., 2025). Explicitly representing variable-pair relations at each lag brings a representation cost that grows quadratically with the number of variables. Second, even when some relations recur across observation windows, their contribution to the current forecast need not be the same (Hu et al., 2025). If this difference is not taken into account, information aggregation can mix information that is useful for the current input with information that is not (Qiu et al., 2025; Hu et al., 2025). Compact relational representations therefore need to be combined with input-adaptive use of those relations. Figure 1 shows related observations on the training data. In Traffic, the mean association strength of a fixed set of sensor pairs varies across windows. In Weather, event averages and the overall correlation profile show a temporal offset between pressure

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.

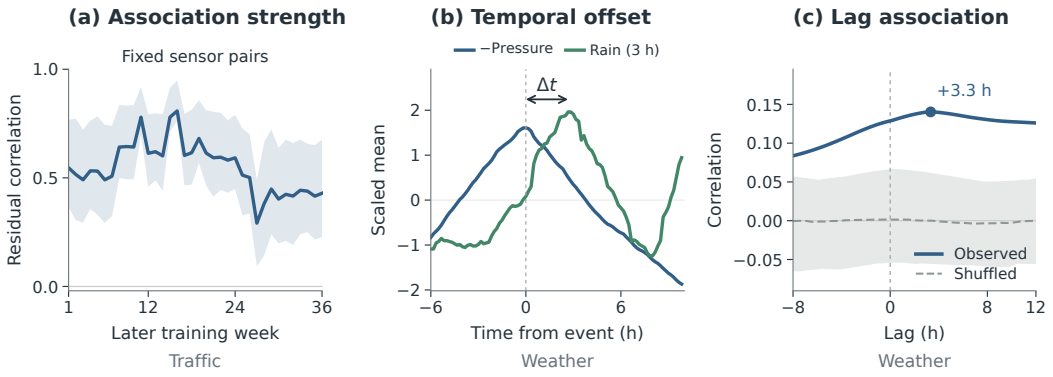


Figure 1: Cross-window and lag associations in the training data. (a) Traffic: weekly mean residual correlations, over the later training half, of highly correlated sensor pairs selected on the earlier half; shading shows the 25%–75% range across pairs. (b) Weather: mean negative pressure and trailing three-hour cumulative rainfall over 47 pressure events; the two mean curves are standardized separately and aligned at pressure minima. (c) Weather: Pearson correlation by lag over the full training set; positive lags place rainfall later. The gray line and band show the median and pointwise 95% range of 200 day-block rainfall shuffles. Hours are based on nominal sampling intervals.

and trailing three-hour rainfall. These observations suggest a design intuition: lag relations shared across windows can provide reusable structure, while their use should adapt to the current state.

Motivated by this view, we propose **FLAG**, a unified framework that combines lag-relation modeling and input-adaptive information exchange in the same representation. The model first obtains local temporal representations through a shared patch encoder, and then constructs a cross-variable relational context in a latent factor space. The factor-lag module connects variables to latent factors through shared low-rank read and write bases, and learns an independent factor-to-factor transition for each temporal offset. Shared factors let relations at different lags reuse the same representational basis, while the full factor transitions allow interaction between source and receiving factors. The module thus retains lag-specific relational information without explicitly constructing dense variable-pair matrices. With the structural hyperparameters and the input length held fixed, both the number of parameters and the computation of this module grow linearly with the number of variables. Because useful cross-variable information may come from the current patch or from earlier history, the model retains both same-patch interaction at lag zero and historical interaction at positive lags, without assuming that every relation requires a nonzero delay. The aggregated cross-variable messages are combined with each variable's own representation, forming a relational context that keeps both local information and cross-variable history.

To adapt these shared relations to the current input, FLAG further uses context-adaptive group communication. The module summarizes information through multiple soft groups and allows the same variable to participate in different groups, providing several compact summaries of cross-variable information. Group memberships are shared across observation windows, while each variable's aggregation weights and message content are generated from the current relational context, so that its contribution can change with the input. Group summaries are then redistributed to the variables and combined with the receiving context, further refining the representation through a residual update. Because the aggregation weights, the message content, and the representation being updated all come from the same relational context, the lag relations constructed in the previous step take part directly in the subsequent information exchange. A shared prediction head then maps the updated historical representation to the future forecast.

We evaluate FLAG on eight benchmarks and observe competitive forecasting performance. In component ablations on five datasets, the full model improves on the shared channel-independent backbone, illustrating the value of combining relational context with group communication. Further lag comparisons examine synchronous and delayed interaction, and show that the benefit of nonzero lags depends on the dataset and the prediction head.

Our main contributions are as follows:

- We introduce a compact factor-lag operator that combines shared low-rank bases with full lag-specific factor transitions to construct relational context from other variables' histories, capturing synchronous and delayed interactions without constructing variable-pair matrices.
- We propose a unified framework for relation modeling and input-adaptive relation use through context-adaptive group communication. The same relational context determines aggregation weights, supplies messages, and receives updates, connecting shared temporal structure with its adaptive use for forecasting.
- We conduct extensive experiments on eight multivariate time series forecasting benchmarks, demonstrating competitive forecasting performance. Component ablations on five datasets and matched lag studies further examine the contributions of relational context and group communication, as well as the roles of synchronous and delayed interactions.

2 Related Work

Time series forecasting. Classical forecasting methods such as ARIMA represent temporal dependence through explicit statistical models (Box et al., 2015). Neural approaches learn representations from collections of series; DeepAR, for example, trains a shared autoregressive recurrent network for probabilistic forecasting (Salinas et al., 2020). For long-horizon prediction, Transformer architectures introduced different ways to organize temporal information (Zhou et al., 2021; Wu et al., 2021): FEDformer combines seasonal–trend decomposition with frequency-domain processing (Zhou et al., 2022), and PatchTST uses patch tokens with a channel-independent encoder (Nie et al., 2023). Alongside this development, DLinear demonstrates the strength of simple linear prediction (Zeng et al., 2023), while TimeMixer combines decomposition with MLP-based mixing across temporal scales (Wang et al., 2024). TimeKAN decomposes a series into frequency bands and represents each band with Kolmogorov–Arnold networks (Huang et al., 2025; Liu et al., 2025). Recent designs further explore compact representations through signal-processing tokenization in SimpleTM and learned temporal queries in TQNet (Chen et al., 2025; Lin et al., 2025). This progression highlights the importance of how temporal information is represented and combined (Wu et al., 2023; Luo & Wang, 2024). Our focus is on organizing information across variables and time within an input-adaptive forecasting model.

Cross-variable relation modeling. Crossformer exchanges information through cross-dimension attention and routers (Zhang & Yan, 2023), while iTransformer attends over variable-history tokens (Liu et al., 2024). SOFTS aggregates series into a shared core and redistributes it through STAR (Han et al., 2024). FACT models fine-grained interactions with across-variable convolution (Wang et al., 2026), TimePro builds variable- and time-aware hyper-states so that different variables can influence a target at different delays (Ma et al., 2025), and GTR retrieves a global-cycle embedding for an MLP host (Cao et al., 2026). These approaches differ in the granularity and organization of cross-variable exchange. A complementary line of work imposes structure on temporal relations: dynamic factor models represent shared variation through latent processes (Stock & Watson, 2016), while tensor-structured vector autoregression restricts response, predictor, and lag dimensions (Wang et al., 2022). FLAG connects structured temporal relations with adaptive exchange in one representation. Shared low-rank read and write bases, together with full lag-specific factor transitions, form a patch-level relational context with same-variable contributions removed. Group communication computes both source weights and messages from this context and updates it through multiple latent groups. Thus, the factor-lag representation directly guides input-dependent communication under a single forecasting objective.

3 Method

3.1 Problem Definition

Multivariate time series forecasting aims to use the historical observations of multiple variables to predict their future changes. Given an observation window $\mathbf{X} \in \mathbb{R}^{T \times N}$ containing N variables and having length T, the model needs to generate predictions $\widehat{\mathbf{Y}} \in \mathbb{R}^{H \times N}$ for the next H time steps to estimate the corresponding ground-truth observations $\mathbf{Y} \in \mathbb{R}^{H \times N}$.

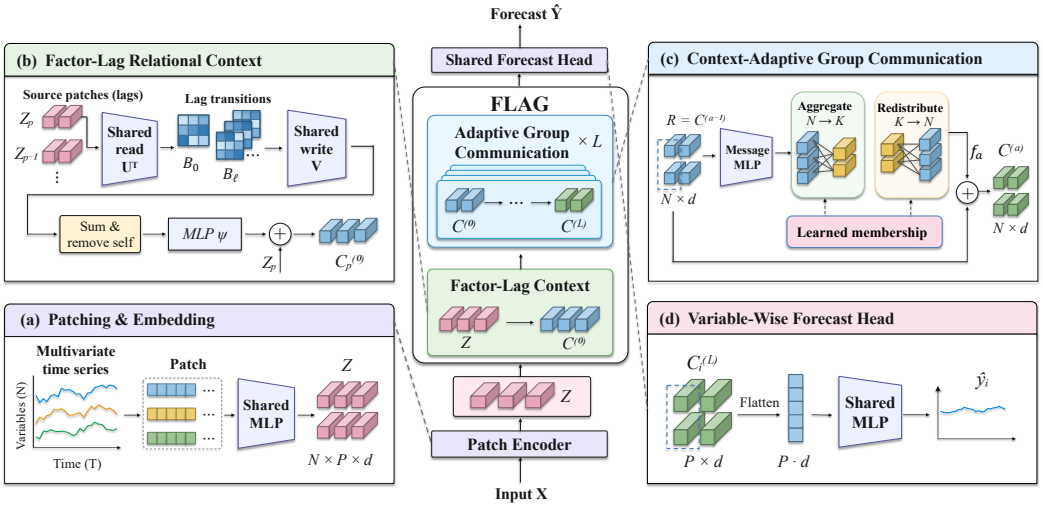


Figure 2: Overview of FLAG. (a) Shared patch encoding produces variable-wise tokens. (b) Factor-lag interactions construct relational context. (c) Context-adaptive group communication refines this context. (d) A shared head predicts each variable's future.

In addition to each variable's own temporal patterns, other variables may also provide useful predictive information. This information may come from the same temporal position or be associated with the target variable through different temporal offsets. At the same time, even if a cross-variable relation recurs across multiple observation windows, the role of the information it provides in the current forecast may not be the same. Therefore, we focus on two related questions: how to compactly represent cross-variable interactions at different temporal offsets, and how to organize the aggregation and use of this information according to the current observations.

3.2 Overall Architecture

We propose FLAG, whose overall architecture is shown in Figure 2. The model first constructs a relational context that contains cross-variable lag information, and then performs input-adaptive information exchange based on this context.

A shared patch encoder serves as the backbone for temporal representation. After the input window is normalized, each variable is divided into $P = T/s$ non-overlapping patches of length s. The shared encoder $\mathcal{E}$ applies a two-layer MLP and layer normalization to each patch and produces $\mathbf{Z} \in \mathbb{R}^{N\times P\times d}$, where d is the feature dimension. We use $p \in \{0, \dots, P-1\}$ to denote the patch position and $\mathbf{Z}_p \in \mathbb{R}^{N\times d}$ to denote the representations of all variables at that position.

The factor-lag module $\mathcal{D}$ thereby obtains the initial context $\mathbf{C}^{(0)} = \mathcal{D}(\mathbf{Z})$. The a-th group-communication layer is $\mathbf{C}^{(a)} = \mathcal{G}_a(\mathbf{C}^{(a-1)})$. With L layers, this yields $\mathbf{C}^{(L)}$. $\mathcal{D}$ is applied only once, and the parameters of each $\mathcal{G}_a$ are independent. Intermediate representations retain the shape $N \times P \times d$. A shared prediction head maps each variable's final historical representation to the future prediction and reverses the window normalization (Kim et al., 2022).

3.3 Factor-Lag Relational Context

Cross-variable information may lie in the same patch or in earlier history, so the module keeps both zero-lag and positive-lag interactions. A separate variable-pair matrix for each offset would use $L_c N^2$ parameters and would not share representations across offsets. We instead connect variables by shared low-rank read and write factors and learn a transition for each offset in that factor space.

Concretely, let $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{N\times r}$ denote the read and write bases, respectively, where $r \le N$ is the number of latent factors. For an offset $\ell \in \{0, \dots, L_c - 1\}$, a full matrix $\mathbf{B}_\ell \in \mathbb{R}^{r\times r}$ describes the transition from read factors to write factors, with rows corresponding to receiving factors and

columns corresponding to source factors. Let $\overline{\mathbf{Z}} = \mathrm{LN}(\mathbf{Z})$, where layer normalization acts along the feature dimension. For a source patch $p-\ell$, $\mathbf{U}^\top \overline{\mathbf{Z}}_{p-\ell}$ first summarizes the variable representations into read factors, $\mathbf{B}_\ell$ then passes information between factors, and $\mathbf{V}$ finally maps the result back to the receiving variables. Shared read and write bases let different lags reuse the same variable-to-factor maps, while each full transition $\mathbf{B}_\ell$ mixes source factors into receiving factors.

We keep each variable's own representation on a residual path and remove its same-variable contribution at every lag. The above factor mappings induce the variable interaction matrix $\mathbf{A}_\ell$, and removing its diagonal terms gives $\overline{\mathbf{A}}_\ell$:

$$\begin{aligned} \mathbf{A}_\ell &= \mathbf{V}\mathbf{B}_\ell\mathbf{U}^\top, \\ \overline{\mathbf{A}}_\ell &= \mathbf{A}_\ell - \mathrm{Diag}(\mathrm{diag}(\mathbf{A}_\ell)) \,. \end{aligned} \tag{1}$$

Here, diag extracts the diagonal of a matrix, and Diag constructs a diagonal matrix from a vector. The rows of $\mathbf{A}_\ell$ correspond to receiving variables and its columns to source variables, with $\mathrm{rank}(\mathbf{A}_\ell) \le r$; this rank constraint applies to the matrix before the diagonal terms are removed. What is removed here is variable-level self-interaction, which does not require the diagonal of the factor transition matrix $\mathbf{B}_\ell$ to be zero. The actual computation is performed along the factor dimension, and same-variable contributions are computed directly from the factors, without constructing $N \times N$ matrices such as $\mathbf{A}_\ell$ or $\overline{\mathbf{A}}_\ell$.

One offset unit corresponds to s sampling steps. For a receiving position p, the set of available offsets is $\mathcal{L}_p = \{\ell : 0 \le \ell < L_c,\ \ell \le p\}$. After aggregating cross-variable messages at these offsets, the relational context is

$$\mathbf{C}_p^{(0)} = \mathbf{Z}_p + \psi\left(\mathrm{LN}\left(\sum_{\ell\in\mathcal{L}_p} \overline{\mathbf{A}}_\ell \overline{\mathbf{Z}}_{p-\ell}\right)\right), \tag{2}$$

where ψ is a shared two-layer MLP acting along the feature dimension, used to transform the aggregated message and add it to the original representation. The read and write bases and lag transitions are shared across observation windows, while the generated context varies with the current input. Therefore, $\mathbf{C}^{(0)}$ contains both each variable's own local representation and cross-variable information incorporated through lag relations, providing a relational context for subsequent adaptive communication.

3.4 Context-Adaptive Group Communication

The shared factor-lag parameters do not change across windows, while the information needed for the current forecast still depends on the input. Context-adaptive group communication therefore forms compact summaries with K learnable soft groups, and one variable may belong to several groups.

Concretely, for layer $a \in \{1, \dots, L\}$, denote the input context by $\mathbf{R} = \mathbf{C}^{(a-1)}$. The message encoder ϕ_a maps it to $\mathbf{E} = \phi_a(\mathbf{R}) \in \mathbb{R}^{N\times P\times d_m}$, where d_m is the message dimension and $e_{i,p,q}$ denotes the value of variable i at patch p and message channel q. Learnable parameters $\mathbf{\Gamma} \in \mathbb{R}^{N\times K}$ define the soft membership of variable i in group k through $\pi_{i,k} = \mathrm{softmax}_k(\Gamma_{i,k})$. All group communication quantities below refer to the current layer.

At each patch and message channel, group k computes aggregation weights according to the variables' memberships and current messages, and forms a group summary:

$$\alpha_{i,p,k,q} = \frac{\pi_{i,k}\exp(e_{i,p,q})}{\sum_{j=1}^{N}\pi_{j,k}\exp(e_{j,p,q})}, \qquad g_{p,k,q} = \sum_{i=1}^{N}\alpha_{i,p,k,q}e_{i,p,q}. \tag{3}$$

Here, $i, j \in \{1, \dots, N\}$ are variable indices, $k \in \{1, \dots, K\}$ is the group index, and $q \in \{1, \dots, d_m\}$ is the message channel index. Group memberships π provide a grouping structure shared across windows, while the aggregation weights α are recomputed for each input, patch, and message channel. Therefore, even when group memberships remain unchanged, each variable's aggregation contribution can still be adjusted according to the current context.

The group summaries are then redistributed to the variables according to the receiving variables' soft memberships:

$$m_{j,p,q} = \sum_{k=1}^{K} \pi_{j,k} g_{p,k,q}, \qquad \mathbf{M} \in \mathbb{R}^{N \times P \times d_m}. \tag{4}$$

This path performs $N \to K \to N$ information exchange at the same patch position, and its input already contains the historical information incorporated by the factor-lag module. To combine the receiving variable's own context with the redistributed message, we concatenate them along the feature dimension and generate a residual update through a two-layer MLP f_a:

$$\mathbf{C}^{(a)} = \mathbf{R} + f_a([\mathbf{R}, \mathbf{M}]). \tag{5}$$

Here, $[\cdot, \cdot]$ denotes concatenation, and f_a maps the $d + d_m$-dimensional input back to d dimensions. Both the message encoder ϕ_a and the update map f_a act along the feature dimension. The specific normalization and initialization settings are given in Appendix A.

3.5 Prediction and Complexity

For variable i, the final context $\mathbf{C}_i^{(L)} \in \mathbb{R}^{P \times d}$ contains representations of all historical patches. The shared prediction head $\mathcal{P}$ flattens it and generates a prediction of length H through a two-layer MLP:

$$\widetilde{\mathbf{y}}_i = \mathcal{P}\Big(\mathrm{vec}(\mathbf{C}_i^{(L)})\Big) \in \mathbb{R}^H, \tag{6}$$

where vec denotes flattening, and $\widetilde{\mathbf{y}}_i$ is the prediction on the normalized scale. After inverse normalization using the statistics retained from the input window, we obtain $\widehat{\mathbf{y}}_i$. Combining the predictions of all variables as columns gives $\widehat{\mathbf{Y}} \in \mathbb{R}^{H \times N}$. The prediction head parameters are shared across variables, while cross-variable information has already been incorporated into its input through relational context construction and group communication.

The factor-lag module performs interactions through latent factors, and group communication performs aggregation and redistribution through group summaries. Neither requires constructing dense variable-pair matrices. The factor relations contain $2Nr + L_c r^2$ parameters, and L layers of group communication contain at most LNK group membership parameters. All other learnable mappings are shared across variables. When r, K, L, L_c, the input and prediction lengths, and the feature widths of all layers are fixed, the model's parameter count and computational cost both grow linearly with the number of variables N.

4 Experiments

We conduct extensive experiments to evaluate the forecasting performance of FLAG and analyze the roles of relational context construction and input-adaptive communication through component ablations, lag comparisons, and factor transition visualizations.

Datasets. For long-term multivariate time series forecasting, we conduct experiments on eight widely used benchmarks: *Weather*, *Traffic*, *Electricity*, *Solar-Energy*, and four *ETT* datasets, namely ETTh1, ETTh2, ETTm1, and ETTm2. These datasets cover weather, transportation, and energy scenarios and contain variable sets of different sizes. The ETT datasets are divided into training, validation, and test sets at a ratio of 6:2:2, while the other datasets use a 7:1:2 split. Detailed dataset statistics are provided in Appendix A.

Baselines. We compare FLAG with eleven representative methods, including DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024), TimeMixer (Wang et al., 2024), SOFTS (Han et al., 2024), TimeKAN (Huang et al., 2025), SimpleTM (Chen et al., 2025), TQNet (Lin et al., 2025), TimePro (Ma et al., 2025), GTR (Cao et al., 2026), and FACT (Wang et al., 2026). These methods cover different approaches, including channel-independent forecasting, inter-variable attention, compact information aggregation, and time-frequency representation modeling, to evaluate the forecasting performance of FLAG relative to different modeling strategies. Descriptions of the baselines are provided in Appendix B.

Experimental settings. All experiments are implemented in PyTorch (Paszke et al., 2019) and run on a single NVIDIA A100 GPU. To ensure comparability of the input settings, the main

comparison uses a historical window of length $T = 96$ for all methods, with prediction lengths $H \in \{96, 192, 336, 720\}$. We use mean squared error (MSE) and mean absolute error (MAE) to evaluate the predictions. Both metrics are computed on standardized targets, with lower values indicating more accurate predictions. Model configurations, training objectives, and other implementation details are provided in Appendix A.

4.1 Main Results

Table 1: Average long-term forecasting performance on eight benchmarks with an input length of 96. Each entry averages the results for prediction horizons $H \in \{96, 192, 336, 720\}$. Lower MSE and MAE indicate better prediction accuracy. Details are in Appendix E.

Model	Weather		Solar		ECL		Traffic		ETTh1		ETTh2		ETTm1		ETTm2	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DLinear (2023)	0.265	0.315	0.330	0.401	0.225	0.319	0.625	0.383	0.461	0.458	0.563	0.519	0.404	0.408	0.354	0.402
PatchTST (2023)	0.265	0.286	0.287	0.333	0.216	0.318	0.529	0.341	0.507	0.472	0.391	0.412	0.402	0.406	0.290	0.334
iTransformer (2024)	0.258	0.278	0.233	0.262	0.178	0.270	0.428	0.282	0.454	0.447	0.383	0.407	0.410	0.410	0.288	0.332
TimeMixer (2024)	0.240	0.272	0.216	0.280	0.182	0.273	0.485	0.298	0.447	0.440	0.365	0.395	0.381	0.396	0.275	0.323
SOFTS (2024)	0.259	0.280	0.230	0.256	0.176	0.267	**0.415**	**0.271**	0.449	0.444	0.379	0.405	0.399	0.407	0.287	0.329
TimeKAN (2025)	0.243	0.272	0.276	0.310	0.197	0.286	0.572	0.372	0.426	0.431	0.391	0.409	0.386	0.398	0.277	0.322
SimpleTM (2025)	0.242	0.271	**0.187**	0.248	0.172	0.265	0.446	0.286	**0.423**	0.429	**0.355**	0.391	0.383	0.397	0.278	0.324
TQNet (2025)	0.243	0.269	0.198	0.258	0.165	0.259	0.446	0.276	0.441	0.433	0.381	0.404	0.377	0.393	0.278	0.323
TimePro (2025)	0.251	0.276	0.232	0.266	0.169	0.262	0.439	0.286	0.438	0.438	0.377	0.403	0.391	0.400	0.279	0.326
GTR (2026)	0.241	0.269	0.209	0.261	0.167	0.260	0.465	0.282	0.447	0.438	0.378	0.403	**0.370**	0.390	**0.270**	**0.316**
FACT (2026)	0.246	0.273	0.198	0.260	**0.160**	**0.256**	0.426	0.277	0.484	0.457	0.379	0.402	0.390	0.403	0.278	0.323
FLAG (ours)	**0.237**	**0.263**	0.190	**0.209**	0.167	0.262	0.424	0.278	0.433	**0.425**	0.358	**0.385**	0.374	**0.380**	0.274	0.317

As shown in Table 1, we report the average results over four prediction lengths on eight datasets, with the best results highlighted in red and the second-best underlined. FLAG achieves the best or second-best average MSE on six datasets and the lowest average MAE on *Weather*, *Solar-Energy*, *ETTh1*, *ETTh2*, and *ETTm1*, showing competitive overall forecasting performance. Complete results for each prediction length are provided in Appendix E.

In particular, on *Weather*, FLAG achieves the lowest average MSE and MAE simultaneously. On *Solar-Energy*, FLAG ranks second in average MSE, while its average MAE decreases from 0.248 for the second-best method, SimpleTM, to 0.209, a reduction of approximately 15.7%. These results show that the framework can maintain good performance under different prediction error metrics.

Compared with the channel-independent PatchTST and iTransformer, which uses inter-variable attention, FLAG has lower average MSE on all eight datasets. Compared with SOFTS, which exchanges information through a shared core, FLAG has lower average MSE on seven datasets. This shows that combining structured lag modeling with context-adaptive communication provides an effective modeling approach for cross-variable forecasting. At the same time, the best method still varies across datasets: FACT and SOFTS achieve lower average MSE on *Electricity* and *Traffic*, respectively, reflecting differences in the advantages of interaction mechanisms on specific datasets.

4.2 Component Ablation

To examine the roles of relational context construction, group communication, and their combination, we conduct component ablations on *ETTm1*, *Weather*, *Solar-Energy*, *Electricity*, and *Traffic*. These datasets cover different numbers of variables and weather, transportation, and energy scenarios, allowing us to examine how component contributions vary with different data characteristics.

With $T = H = 96$, we compare four variants. **Base** generates predictions directly from the channel-independent representation $\mathbf{Z}$. **Factor-lag** uses the relational context $\mathbf{C}^{(0)} = \mathcal{D}(\mathbf{Z})$ without group communication. **Group communication** performs group communication directly on $\mathbf{Z}$. The complete **FLAG** uses the relational context as both the message source and the receiving representation for group communication. Within each dataset, these variants use the same encoder, prediction head, and corresponding training settings.

As shown in Figure 3, the complete model has lower test MSE than Base on all five datasets. It is the lowest of the four variants on *Solar-Energy*, *Electricity*, and *Traffic*. On *ETTm1* the four test MSEs

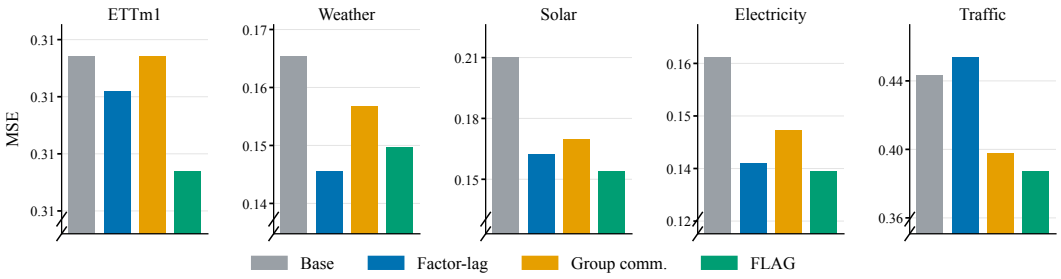


Figure 3: Component ablation results on ETTm1, Weather, Solar-Energy, Electricity, and Traffic, with an input length of 96 and a prediction horizon of 96.

are similar. This dataset has only seven variables, so the additional cross-variable signal is limited and the component differences remain small. Compared with performing group communication directly on $\mathbf{Z}$, using the factor-lag context lowers MSE on all five datasets. Adding group communication after the factor-lag context further lowers MSE on four of them.

Traffic shows why the two components should be read together. Factor-lag alone raises MSE from 0.444 for Base to 0.454. Placing the same module before group communication lowers MSE from 0.398 to 0.387. The relational context is useful here when adaptive aggregation uses it, even though it does not improve the backbone by itself.

Weather is the exception: factor-lag alone reaches 0.146, and adding group communication raises MSE to 0.150. On *Electricity*, factor-lag already accounts for most of the reduction, from 0.167 to 0.137, and the complete model reaches 0.134. The combined model improves on the channel-independent backbone throughout, while further exchange after the relational context depends on the dataset.

4.3 Effect of Lagged Interactions

To further examine the role of nonzero lags in cross-variable interactions, we compare three lag settings on *Weather* and *Traffic*: **Full** retains the configured zero-lag and positive-lag interactions, **Sync** retains only interactions within the same patch, i.e., $\ell = 0$, and **Lag-only** retains only positive-lag interactions, i.e., $\ell \geq 1$. These two datasets cover weather series and high-dimensional traffic series, respectively, and also correspond to the scenarios used in Figure 1 to illustrate cross-window variation and temporal offsets.

To examine whether the way the prediction head accesses historical representations affects the benefits of lags, we further compare two prediction heads. The standard **Flatten** head reads all historical output tokens, while the **Last-1** head reads only the final output token. Each variant is trained separately, with the other structural and training settings held fixed within the same dataset and prediction head. Table 2 reports the test MSE.

Table 2: Lag-bank ablation at $T = H = 96$. Red bold marks the lowest test MSE in each row. Full uses every patch in the input window, so both datasets use offsets $\{0, \ldots, 11\}$. Flatten reads all output tokens; last-1 reads only the final token.

Dataset	Head	Full	Sync ($\ell = 0$)	Lag-only ($\ell \geq 1$)
Weather	Flatten	**0.150**	0.151	**0.150**
Weather	Last-1	0.159	0.224	**0.156**
Traffic	Flatten	**0.387**	0.398	0.413
Traffic	Last-1	**0.545**	0.576	0.566

On *Traffic*, Full has the lowest MSE under both prediction heads. With Flatten it is lower than Sync (0.387 vs. 0.398) and Lag-only (0.413). With Last-1 it is lower than Sync (0.545 vs. 0.576) and Lag-only (0.566). Nonzero lags are useful on this dataset, and keeping synchronous interaction together with them is better than either support alone.

Weather depends on the prediction head. With Flatten the three settings are close, at 0.150, 0.151, and 0.150. With Last-1, Sync rises to 0.224, while Lag-only (0.156) is slightly lower than Full (0.159). Last-1 changes both the input range and the capacity of the prediction head, so the comparison is between lag settings within the same head.

These results show that the benefits of nonzero lags depend on the dataset and the historical representations that the prediction head can access directly. Retaining both zero-lag and positive-lag interactions allows the model to represent cross-variable information from both the current patch and earlier history, without assuming that all predictive relations must have a nonzero delay.

4.4 Visualizing Lag-Specific Factor Interactions

To provide an intuitive understanding of how factor-lag represents interactions at different temporal offsets in a shared factor space, we visualize the factor transition matrices learned on *Electricity*. We choose this dataset because it contains a relatively large number of variables and the component ablation shows that relational context construction brings a substantial improvement in forecasting. Under this experimental setting, the model connects 321 variables through $r = 16$ read and write factors.

Figure 4 shows the transition matrices $\mathbf{B}_\ell$ corresponding to offsets $\ell = 1, 3, 6, 11$. All panels keep the same factor order and color range. Rows represent receiving factors, columns represent source factors, and red and blue indicate positive and negative coefficients, respectively. The four matrices have similar Frobenius norms, approximately 0.096–0.100, but the locations and signs of the larger coefficients differ clearly. For example, the coefficient corresponding to receiving factor 10 and source factor 15 is strongly negative in $\mathbf{B}_1$ but positive in $\mathbf{B}_3$ and $\mathbf{B}_6$. Therefore, similar overall norms do not imply the same interaction structure; different offsets form different factor combinations under the shared read and write bases.

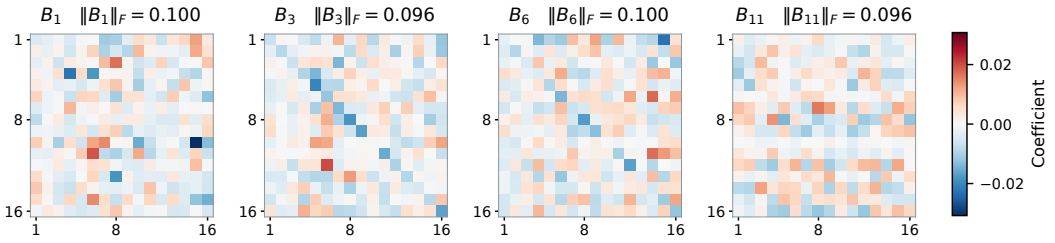


Figure 4: Lag-specific factor transitions on Electricity ($T = H = 96$, $r = 16$, seed 2021). Rows and columns index receiving and source factors, respectively. Four offsets span the nonzero support; one lag unit is one patch stride of 8 observations. Red and blue indicate positive and negative coefficients. All panels share the scale of the full nonzero bank, with no per-matrix rescaling.

In addition, the larger coefficients are not concentrated only on the diagonal, reflecting combinations of different source and receiving factors through full factor transitions. The matrices shown here are factor-to-factor matrices $\mathbf{B}_\ell$, whose diagonals do not correspond to variables' self-interactions. This is therefore consistent with the removal of same-variable contributions at the variable level in the method. The complete lag matrices and their extraction procedure are provided in Appendix C.

5 Conclusion

FLAG unifies shared temporal structure and input-adaptive cross-variable communication. Shared low-rank bases and lag-specific factor transitions construct the context used by adaptive group communication, with cost linear in the number of variables for fixed model dimensions. The complete model improves on the shared backbone across five ablation datasets, but the contributions vary: Traffic benefits from their combination, whereas Weather favors factor-lag context alone. Matched lag ablations show consistent gains from delayed exchange on Traffic and head-dependent gains on Weather. Learned interaction profiles further characterize how the model distributes cross-variable relations across temporal offsets.

AI use statement

In this work, we used generative AI tools to expand the design of the ablation experiments and to assist with translation. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for proofs, write proofs, propose or refine hypotheses, implement the method, clean or reformat datasets, perform qualitative or thematic analysis, or interpret experimental results. Additionally, we used generative AI tools to polish the manuscript, search for related work, and analyze the retrieved literature. We have reviewed all AI-assisted work, including the expanded ablation designs, the translations, the polished text, and the related-work suggestions, before including them. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

The code is available at `https://anonymous.4open.science/r/FLAG-DE06`. All datasets used in the experiments are publicly available.

References

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control, 5th Edition*. John Wiley & Sons, Hoboken, NJ, USA, 2015.

Fanpu Cao, Lu Dai, Jindong Han, and Hui Xiong. Enhancing multivariate time series forecasting with global temporal retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=QUJBPSfyui`.

Hui Chen, Viet Luong, Lopamudra Mukherjee, and Vikas Singh. Simpletm: A simple baseline for multivariate time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://proceedings.iclr.cc/paper_files/paper/2025/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html`.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. SOFTS: Efficient multivariate time series forecasting with series-core fusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/754612bde73a8b65ad8743f1f6d8ddf6-Abstract-Conference.html`.

Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. TimeFilter: Patch-specific spatial-temporal graph filtration for time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24893–24911, 2025. URL `https://proceedings.mlr.press/v267/hu25ac.html`.

Songtao Huang, Zhen Zhao, Can Li, and LEI BAI. TimeKAN: KAN-based frequency decomposition learning architecture for long-term time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=wTLc79YNbh`.

Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022. doi: 10.1016/j.eswa.2022.117921. URL `https://www.sciencedirect.com/science/article/pii/S0957417422011654`.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *The Tenth International Conference on Learning Representations*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=cGDAkQo1C0p`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*, 2015. URL `https://arxiv.org/abs/1412.6980`.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677): 1416–1421, 2023. doi: 10.1126/science.adi2336. URL https://www.science.org/doi/abs/10.1126/science.adi2336.

Shengsheng Lin, Haojun Chen, Haijie Wu, Chunyun Qiu, and Weiwei Lin. Temporal query network for efficient multivariate time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37797–37814, 2025. URL https://proceedings.mlr.press/v267/lin25e.html.

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 *The Twelfth International Conference on Learning Representations*. OpenReview.net, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/afaed89642ea100935e39d39a4da602c-Abstract-Conference.html.

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*. OpenReview.net, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

Xiaowen Ma, Zhen-Liang Ni, Shuai Xiao, and Xinghao Chen. TimePro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 42096–42111, 2025. URL https://proceedings.mlr.press/v267/ma25p.html.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, 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. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1185–1196, 2025. doi: 10.1145/3690624.3709325. URL https://arxiv.org/abs/2412.10859.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

James H. Stock and Mark W. Watson. Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. In *Handbook of Macroeconomics*, volume 2A, pp. 415–525. Elsevier, 2016. doi: 10.1016/bs.hesmac.2016.04.002. URL https://www.princeton.edu/~mwatson/papers/Stock_Watson_HOM_Vol2.

Di Wang, Yao Zheng, Heng Lian, and Guodong Li. High-dimensional vector autoregressive time series modeling via tensor decomposition. *Journal of the American Statistical Association*, 117 (539):1338–1356, 2022. doi: 10.1080/01621459.2020.1855183. URL https://arxiv.org/abs/1909.06624.

Huiqiang Wang, Jieming Shi, and Qing Li. FACT: Fine-grained across-variable convolution for multivariate time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/090d636f951ac0bb18fd5610c95c6a76-Abstract-Conference.html.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7oLshfEIC2.

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. URL https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

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*. OpenReview.net, 2023. URL https://openreview.net/forum?id=ju_Uqw3840q.

Linfei Yin, Xinghui Cao, and Dongduan Liu. Weighted fully-connected regression networks for one-day-ahead hourly photovoltaic power forecasting. *Applied Energy*, 332:120527, 2023. doi: 10.1016/j.apenergy.2022.120527. URL https://www.sciencedirect.com/science/article/pii/S0306261922017846.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

Lifan Zhao and Yanyan Shen. Rethinking channel dependence for multivariate time series forecasting: Learning from leading indicators. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JiTVtCUOpS.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/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*, pp. 27268–27286. PMLR, 2022.

A Implementation Details

Dataset Details. We evaluate long-term multivariate forecasting on eight public benchmarks. The ETT collection contains measurements from two electricity transformers with 7 variables, recorded hourly (ETTh1, ETTh2) and at 15-minute intervals (ETTm1, ETTm2). Electricity contains the hourly power consumption of 321 clients. Weather comprises 21 meteorological indicators recorded every 10 minutes. Traffic documents hourly road-occupancy rates from 862 sensors. Solar-Energy records solar power generation from 137 plants at 10-minute intervals. Each dataset is split chronologically into training, validation, and test sets, using a 6:2:2 ratio for the ETT datasets and 7:1:2 for the others. Each variable is standardized with statistics fitted on the training segment only. Table 3 summarizes these datasets.

Table 3: Statistics of the benchmark datasets.

Dataset	Dim	Prediction Lengths	Dataset Size	Frequency	Information
ETTh1	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Hourly	Electricity
ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Hourly	Electricity
ETTm1	7	{96, 192, 336, 720}	(34465, 11521, 11521)	15 min	Electricity
ETTm2	7	{96, 192, 336, 720}	(34465, 11521, 11521)	15 min	Electricity
Electricity	321	{96, 192, 336, 720}	(18317, 2633, 5261)	Hourly	Electricity
Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	10 min	Weather
Traffic	862	{96, 192, 336, 720}	(12185, 1757, 3509)	Hourly	Transportation
Solar-Energy	137	{96, 192, 336, 720}	(36601, 5161, 10417)	10 min	Energy

Experiment Details. All experiments are implemented in PyTorch and conducted on a single NVIDIA A100 40GB GPU. We use the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 1×10^{-4} and a batch size of 32. Training minimizes mean squared error for at most 100 epochs and keeps the checkpoint with the best validation loss; early stopping uses patience 3. The look-back window is fixed at $T = 96$, and the prediction horizons are $H \in \{96, 192, 336, 720\}$. Each configuration is trained with three random seeds, and the reported results are averages over these runs. MSE and MAE are the evaluation metrics.

Inputs are divided into non-overlapping patches of length and stride $s = 8$, giving $P = 12$ tokens per variable. The patch encoder is a two-layer MLP followed by layer normalization, and the prediction head is a two-layer MLP applied to the flattened $P \cdot d$-dimensional representation of each variable. Group communication uses deterministic context-weighted pooling, and the factor-lag context is both the source and the receiver.

The lag-bank size L_c counts offsets $\ell = 0, \ldots, L_c - 1$. Thus $L_c = 12$ covers all available patch offsets and $L_c = 1$ permits only same-patch cross-variable interaction.

The read and write bases $\mathbf{U}$ and $\mathbf{V}$ are obtained by reduced QR normalization of learned matrices, giving orthonormal columns. For each receiving factor, the transition bank is divided by its absolute sum over source factors and lags, with denominator floor 10^{-6}. The initial transitions are $\mathbf{B}_0 = \mathbf{I}$ and $\mathbf{B}_{\ell>0} = \mathbf{0}$. These factors parameterize predictive relations without an assumption of physical identifiability. Same-variable terms are computed exactly as $(\mathbf{A}_\ell)_{ii} = \mathbf{v}_i^\top \mathbf{B}_\ell \mathbf{u}_i$, where $\mathbf{u}_i^\top$ and $\mathbf{v}_i^\top$ are the corresponding rows of the read and write bases. Subtracting this diagonal need not preserve the rank of $\mathbf{A}_\ell$.

Both layer normalizations in the factor-lag branch act on the feature dimension without affine parameters. The map ψ uses two bias-free linear projections with GELU and dropout; the output projection is zero-initialized. The group message encoder ϕ_a uses widths $d \to d \to d_m$ with GELU between projections. The residual map f_a has widths $(d + d_m) \to d \to d$ with GELU and a zero-initialized final weight and bias. Both relational updates are therefore initially zero, so the model starts from the channel-independent backbone.

Group aggregation uses the weighted expectation in Equation 3 during both training and inference. For $K = 1$, $\pi_{i,1} = 1$ and no membership parameters are allocated; aggregation reduces to one context-weighted global group. Training uses the forecasting objective only, without an auxiliary reconstruction or routing loss.

B Compared Baselines

The long-term comparison includes eleven published baselines, trained under the protocol in Section 4. We summarize their main modeling ideas below in descending order of publication year.

- **FACT** (Wang et al., 2026): models fine-grained cross-variable interactions in both the time and frequency domains. It arranges variables on a two-dimensional grid and applies depthwise convolutions with channel-specific kernels. Progressively increasing dilation rates expand the receptive field to capture global variable interactions.
- **GTR** (Cao et al., 2026): augments a forecasting backbone with global periodic information beyond its input window. It maintains a learnable embedding of an entire cycle and retrieves segments aligned with the current input. Two-dimensional convolution and residual fusion combine the retrieved global context with local observations.
- **TimePro** (Ma et al., 2025): addresses the multi-delay problem, where different variables influence a target over different time intervals, using a Mamba-based architecture. It preserves fine-grained temporal features within each variable token and adaptively selects relevant time points to refine the state. The resulting hyper-state jointly represents variable relationships and salient temporal information.
- **TQNet** (Lin et al., 2025): combines periodically shifted learnable queries with input-derived keys and values to model cross-variable dependencies. The queries encode global inter-variable patterns, while the keys and values capture sample-specific correlations. A single attention layer and a lightweight MLP produce the forecasts.
- **SimpleTM** (Chen et al., 2025): uses a learnable stationary wavelet transform to construct time-frequency tokens at multiple scales. It models token interactions with geometric-product attention, which augments dot-product similarity with a wedge-product term. An inverse wavelet transform reconstructs the temporal representation for prediction, yielding a compact architecture specialized for multivariate forecasting.
- **TimeKAN** (Huang et al., 2025): uses cascaded frequency decomposition to separate frequency bands and learns their representations with multi-order Kolmogorov–Arnold networks. Higher-frequency bands use higher polynomial orders to accommodate more complex temporal patterns, and frequency-mixing blocks recombine the learned representations. Each variable is processed independently.
- **SOFTS** (Han et al., 2024): is an MLP-based forecaster that models channel interactions through its STar Aggregate-Redistribute (STAR) module. STAR pools all series into a shared core representation, then redistributes and fuses that core with each individual series. This centralized exchange allows variables to share information with complexity that grows linearly with the number of channels.
- **TimeMixer** (Wang et al., 2024): is an MLP-based architecture that exploits temporal patterns at multiple sampling scales. Its Past-Decomposable-Mixing block separates seasonal and trend components, mixing seasonality from fine to coarse scales and trends in the reverse direction. Its Future-Multipredictor-Mixing block combines forecasts from multiple scales to exploit their complementary predictive information.
- **iTransformer** (Liu et al., 2024): inverts the usual temporal-token formulation by embedding the history of each variable into a separate token. Self-attention captures dependencies across variable tokens, while a feed-forward network learns nonlinear representations for each token.
- **PatchTST** (Nie et al., 2023): segments each univariate series into temporal patches and treats them as Transformer tokens. It processes channels independently while sharing embedding and Transformer weights across channels. Patching preserves local temporal structure and reduces the number of attention tokens, enabling longer histories to be modeled with lower attention cost.
- **DLinear** (Zeng et al., 2023): separates each series into a moving-average trend and a residual seasonal component. Two linear projections map these components directly to the forecast horizon, and their outputs are summed to obtain the prediction. Variables are predicted independently without cross-variable interactions.

C Visualization of Learned Factor Transitions

Figure 4 uses the Electricity component-ablation configuration. Factor-lag accounts for most of the gain over Base on this dataset, and the 16×16 matrices permit inspection of interactions among multiple factors. The four main-text offsets, $1, 3, 6, 11$, sample short, intermediate, and long delays; Figure 5 shows every configured offset.

We extract the effective transition bank used in the forward pass. Writing the unconstrained checkpoint parameters as $\mathbf{\Theta}$, the joint normalization is

$$B_\ell[a, b] = \frac{\Theta[a, b, \ell]}{\max\left(\sum_{j,k} |\Theta[a, j, k]|, 10^{-6}\right)}. \tag{7}$$

Rows a correspond to write factors and columns b to read factors in $\mathbf{A}_\ell = \mathbf{V}\mathbf{B}_\ell\mathbf{U}^\top$. Their original checkpoint order is retained across all lags. Read and write factors use separate bases, so diagonal entries of $\mathbf{B}_\ell$ do not represent variable-level self-interactions. The raw bank is initialized with the identity at offset 0 and zeros at positive offsets; the figures show trained, normalized coefficients.

The main-text panels share one signed linear scale over all nonzero offsets, and no matrix is independently rescaled. $\mathbf{B}_0$ has a substantially larger magnitude, so Figure 5 uses a separate scale that includes it.

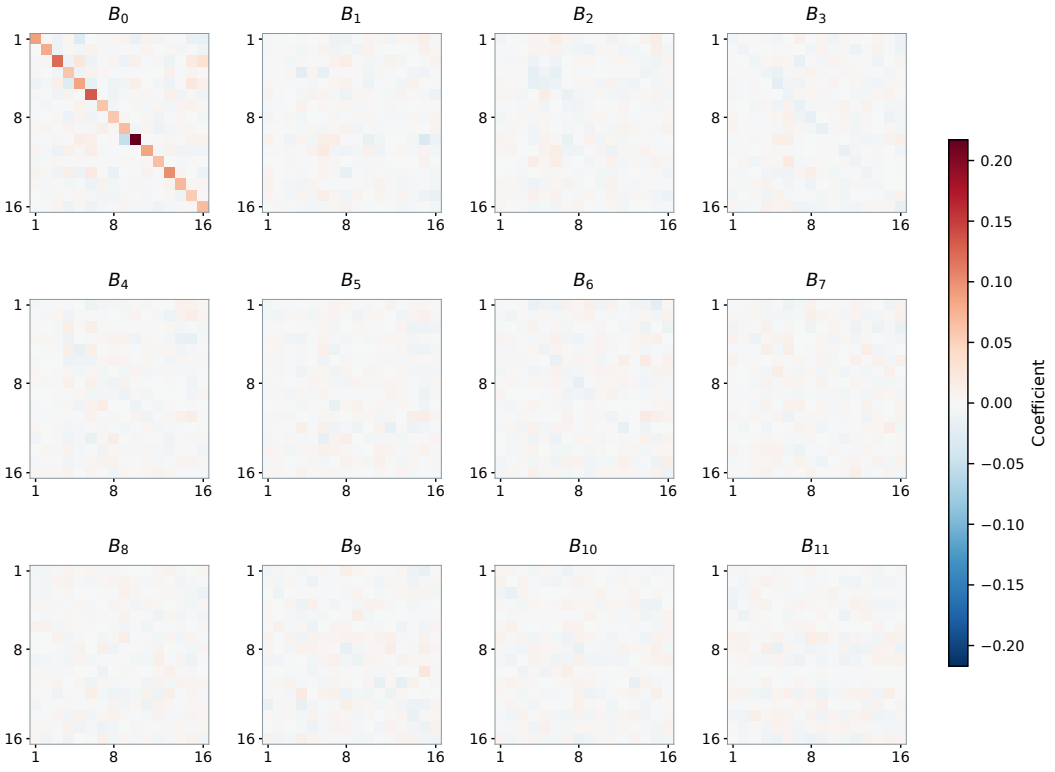


Figure 5: Complete Electricity factor-transition bank (seed 2021). Rows and columns index receiving and source factors, respectively. All twelve matrices share one signed linear scale, including $\mathbf{B}_0$. The larger synchronous coefficients make nonzero entries paler on this scale; Figure 4 resolves their structure using the shared range of the nonzero bank.

D Effect of Look-back Window Size

A model that uses historical context is expected to benefit from a longer look-back window (Zeng et al., 2023; Nie et al., 2023). To examine how FLAG uses additional history, we vary the input

length on ETTm1, Weather, and Solar-Energy. The forecast horizon is fixed at $H = 96$, and the input length takes values $T \in \{48, 96, 144, 192, 240, 336\}$. Structural hyperparameters stay at the $H = 96$ configuration of each dataset, and the factor-lag bank covers every patch in the current window. We compare FLAG with SOFTS, TQNet, and FACT. Results are reported in Figure 6.

The four models drop sharply when T increases from 48 to 96, but longer windows do not help uniformly. On Weather, FLAG reaches 0.146 at $T = 144$ and then stays between 0.148 and 0.151. It has the lowest MSE at every length except $T = 48$, where TQNet is 0.174 against 0.176. On Solar-Energy, FLAG is 0.154 at $T = 96$ and 0.155 at $T = 144$, then rises to 0.166 at $T = 192$. FACT is lower at $T = 48$ (0.291 vs. 0.300), while FLAG is lowest at the other five lengths. On ETTm1, the error shows an overall decrease up to $T = 336$: FLAG moves from 0.412 at $T = 48$ to 0.287 at $T = 336$. TQNet is slightly lower at $T = 144$ (0.294 vs. 0.298) and $T = 192$ (0.288 vs. 0.289), and FACT is lower at $T = 48$ (0.383 vs. 0.412). A moderate window is therefore enough on Weather and Solar-Energy, whereas ETTm1 still gains from the longest input. FLAG is competitive once $T \geq 96$, without being the lowest curve at every length.

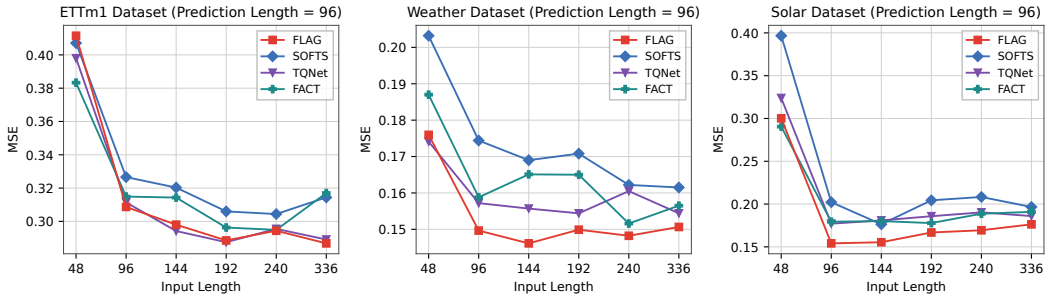


Figure 6: The effect of look-back window size on forecasting performance (MSE). We compare FLAG, SOFTS, TQNet, and FACT on ETTm1, Weather, and Solar-Energy with a fixed prediction horizon of $H = 96$.

E Full Results

Table 4 reports MSE and MAE at horizons $\{96, 192, 336, 720\}$ and their average for every compared model. Highlights are computed over the filled entries, including ties. Gray dashes are unfilled and do not enter the highlights; black dashes mark unavailable results.

Table 4: Performance comparison on the long-term forecasting task for prediction horizons of $\{96, 192, 336, 720\}$ and their average. The input sequence length is fixed to 96. The best and second-best results on each dataset in each metric are highlighted in **bold red** and underlined blue fonts, respectively. Gray dashes are unfilled entries and are excluded from the highlights; black dashes mark unavailable results.

Models		**FLAG (Ours)**		FACT (2026)		GTR (2026)		TimePro (2025)		TQNet (2025)		SimpleTM (2025)		TimeKAN (2025)		SOFTS (2024)		TimeMixer (2024)		iTransformer (2024)		PatchTST (2023)		DLinear (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.152**	**0.190**	0.159	0.202	0.156	0.201	0.166	0.207	0.157	0.201	0.157	0.203	0.162	0.208	0.174	0.214	0.163	0.209	0.174	0.214	0.186	0.227	0.195	0.252
	192	**0.204**	**0.241**	0.208	0.250	0.205	0.245	0.216	0.254	0.206	0.245	0.207	0.250	0.207	0.249	0.220	0.256	0.208	0.250	0.221	0.254	0.234	0.265	0.237	0.295
	336	0.263	0.288	0.271	0.294	0.261	0.288	0.273	0.296	0.263	0.288	0.262	0.289	0.263	0.290	0.281	0.300	**0.251**	**0.287**	0.278	0.296	0.284	0.301	0.282	0.331
	720	**0.328**	**0.333**	0.344	0.344	0.341	0.341	0.351	0.346	0.344	0.342	0.340	0.341	0.338	0.340	0.362	0.351	0.339	0.341	0.358	0.347	0.356	0.349	0.345	0.382
	Avg	**0.237**	**0.263**	0.246	0.273	0.241	0.269	0.251	0.276	0.243	0.269	0.242	0.271	0.243	0.272	0.259	0.280	0.240	0.272	0.258	0.278	0.265	0.286	0.265	0.315
Solar-Energy	96	**0.154**	**0.187**	0.179	0.242	0.199	0.255	0.196	0.237	0.177	0.238	0.167	0.229	0.234	0.290	0.202	0.229	0.189	0.259	0.203	0.237	0.265	0.323	0.290	0.378
	192	**0.179**	**0.201**	0.198	0.261	0.195	0.254	0.231	0.263	0.201	0.261	0.184	0.250	0.274	0.309	0.228	0.253	0.222	0.283	0.233	0.261	0.288	0.332	0.320	0.398
	336	0.205	**0.223**	0.209	0.269	0.210	0.259	0.250	0.281	0.205	0.261	**0.192**	0.257	0.299	0.324	0.241	0.268	0.231	0.292	0.248	0.273	0.301	0.339	0.353	0.415
	720	0.222	**0.226**	0.207	0.268	0.231	0.277	0.253	0.285	0.208	0.270	**0.204**	0.256	0.295	0.318	0.247	0.274	0.223	0.285	0.249	0.275	0.295	0.336	0.357	0.413
	Avg	0.190	**0.209**	0.198	0.260	0.209	0.261	0.232	0.266	0.198	0.258	**0.187**	0.248	0.276	0.310	0.230	0.256	0.216	0.280	0.233	0.262	0.287	0.333	0.330	0.401
Electricity	96	0.138	0.235	**0.133**	**0.229**	0.135	0.230	0.139	0.234	0.134	0.230	0.141	0.235	0.174	0.266	0.144	0.236	0.153	0.247	0.148	0.240	0.190	0.296	0.210	0.302
	192	0.154	0.250	**0.151**	**0.245**	0.152	**0.245**	0.156	0.249	0.154	0.247	**0.151**	0.247	0.182	0.273	0.164	0.255	0.166	0.256	0.162	0.253	0.199	0.304	0.210	0.305
	336	0.171	0.268	**0.165**	**0.261**	0.170	0.264	0.172	0.267	0.170	0.265	0.173	0.267	0.197	0.286	0.180	0.272	0.185	0.277	0.178	0.269	0.217	0.319	0.223	0.319
	720	0.204	0.293	**0.190**	**0.287**	0.212	0.302	0.209	0.299	0.202	0.295	0.222	0.310	0.236	0.320	0.216	0.304	0.225	0.310	0.225	0.317	0.258	0.352	0.258	0.350
	Avg	0.167	0.262	**0.160**	**0.256**	0.167	0.260	0.169	0.262	0.165	0.259	0.172	0.265	0.197	0.286	0.176	0.267	0.182	0.273	0.178	0.270	0.216	0.318	0.225	0.319
Traffic	96	0.385	0.261	0.392	0.262	0.438	0.265	0.407	0.269	0.413	0.261	0.417	0.272	0.580	0.379	**0.375**	**0.250**	0.462	0.285	0.395	0.268	0.526	0.347	0.650	0.396
	192	0.408	0.270	0.414	0.271	0.450	0.279	0.424	0.276	0.432	0.272	0.435	0.277	0.550	0.363	**0.407**	**0.268**	0.473	0.296	0.417	0.276	0.522	0.332	0.598	0.370
	336	0.434	0.280	0.431	0.278	0.467	0.283	0.443	0.286	0.451	0.277	0.447	0.284	0.559	0.363	**0.418**	**0.271**	0.498	0.296	0.433	0.283	0.517	0.334	0.605	0.373
	720	0.467	0.304	0.465	0.297	0.504	0.301	0.483	0.312	0.486	**0.295**	0.486	0.309	0.600	0.381	**0.459**	**0.295**	0.506	0.313	0.467	0.302	0.552	0.352	0.645	0.394
	Avg	0.424	0.278	0.426	0.277	0.465	0.282	0.439	0.286	0.446	0.276	0.446	0.286	0.572	0.372	**0.415**	**0.271**	0.485	0.298	0.428	0.282	0.529	0.341	0.625	0.383
ETTh1	96	0.378	**0.388**	0.393	0.404	0.369	0.392	0.375	0.398	0.372	0.393	**0.365**	0.393	0.374	0.397	0.382	0.401	0.375	0.400	0.386	0.405	0.460	0.447	0.397	0.412
	192	0.429	**0.418**	0.442	0.435	0.427	0.423	0.427	0.429	0.427	0.423	0.422	0.422	**0.416**	0.422	0.436	0.433	0.429	0.421	0.441	0.436	0.477	0.429	0.446	0.441
	336	0.463	**0.436**	0.487	0.458	0.479	0.448	0.472	0.450	0.475	0.446	**0.439**	0.439	0.451	0.443	0.479	0.452	0.484	0.458	0.487	0.458	0.546	0.496	0.489	0.467
	720	**0.462**	**0.460**	0.612	0.530	0.513	0.487	0.476	0.474	0.488	0.470	0.464	0.462	0.463	0.463	0.499	0.488	0.498	0.482	0.503	0.491	0.544	0.517	0.513	0.510
	Avg	0.433	**0.425**	0.484	0.457	0.447	0.438	0.438	0.438	0.441	0.433	**0.423**	0.429	0.426	0.431	0.449	0.444	0.447	0.440	0.454	0.447	0.507	0.472	0.461	0.458
ETTh2	96	**0.277**	**0.327**	0.299	0.346	0.297	0.346	0.293	0.345	0.297	0.346	0.279	0.338	0.290	0.340	0.297	0.348	0.289	0.341	0.297	0.349	0.308	0.355	0.340	0.394
	192	**0.356**	**0.376**	0.369	0.391	0.372	0.393	0.367	0.394	0.366	0.394	0.358	0.388	0.379	0.396	0.374	0.394	0.372	0.392	0.380	0.400	0.393	0.405	0.482	0.479
	336	0.400	0.412	0.419	0.425	0.420	0.431	0.419	0.431	0.417	0.427	**0.368**	**0.403**	0.423	0.435	0.419	0.431	0.386	0.414	0.428	0.432	0.427	0.436	0.591	0.541
	720	**0.400**	**0.425**	0.428	0.445	0.424	0.442	0.427	0.445	0.442	0.450	0.413	0.435	0.473	0.465	0.427	0.445	0.412	0.434	0.427	0.445	0.436	0.450	0.839	0.661
	Avg	0.358	**0.385**	0.379	0.402	0.378	0.403	0.377	0.403	0.381	0.404	**0.355**	0.391	0.391	0.409	0.379	0.405	0.365	0.395	0.383	0.407	0.391	0.412	0.563	0.519
ETTm1	96	0.309	**0.341**	0.315	0.360	**0.307**	0.352	0.326	0.364	0.311	0.353	0.326	0.364	0.329	0.366	0.327	0.363	0.320	0.357	0.334	0.368	0.352	0.374	0.346	0.374
	192	0.357	**0.368**	0.361	0.383	**0.355**	0.378	0.367	0.383	0.356	0.378	0.361	0.381	0.363	0.380	0.376	0.391	0.361	0.381	0.390	0.393	0.374	0.387	0.382	0.391
	336	0.384	**0.389**	0.397	0.410	**0.382**	0.398	0.402	0.409	0.391	0.402	0.390	0.404	0.390	0.404	0.414	0.418	0.390	0.404	0.426	0.420	0.421	0.414	0.415	0.415
	720	0.445	**0.424**	0.486	0.458	**0.434**	0.433	0.469	0.446	0.450	0.439	0.454	0.438	0.460	0.443	0.478	0.454	0.454	0.441	0.491	0.459	0.462	0.449	0.473	0.451
	Avg	0.374	**0.380**	0.390	0.403	**0.370**	0.390	0.391	0.400	0.377	0.393	0.383	0.397	0.386	0.398	0.399	0.407	0.381	0.396	0.410	0.410	0.402	0.406	0.404	0.408
ETTm2	96	0.171	**0.249**	0.172	0.254	**0.168**	0.251	0.171	0.260	0.173	0.255	0.175	0.258	0.174	0.255	0.182	0.261	0.175	0.258	0.180	0.264	0.183	0.270	0.193	0.293
	192	0.235	**0.293**	0.236	0.298	**0.233**	**0.293**	0.242	0.303	0.241	0.300	0.238	0.299	0.239	0.299	0.246	0.306	0.237	0.299	0.250	0.309	0.255	0.314	0.284	0.361
	336	0.296	0.333	0.300	0.340	**0.289**	**0.330**	0.303	0.342	0.297	0.337	0.297	0.338	0.301	0.340	0.310	0.346	0.298	0.340	0.311	0.348	0.309	0.347	0.382	0.429
	720	0.393	0.392	0.403	0.399	**0.390**	**0.390**	0.400	0.399	0.399	0.398	0.403	0.400	0.395	0.396	0.408	0.403	0.391	0.396	0.412	0.407	0.412	0.404	0.558	0.525
	Avg	0.274	0.317	0.278	0.323	**0.270**	**0.316**	0.279	0.326	0.278	0.323	0.278	0.324	0.277	0.322	0.287	0.329	0.275	0.323	0.288	0.332	0.290	0.334	0.354	0.402