LATENT PROPAGATION FOR CONTINUAL-CONTEXT TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series forecasting is a fundamental task in machine learning, aiming to predict future horizons from historical observations. Prior methods have largely been developed under a *fixed-length formulation*, where a continuous time series is divided into input-output pairs of fixed-length input and prediction windows. This formulation, however, treats each forecasting instance independently, relying only on the observations within the fixed-length input window without maintaining context beyond it. As a result, it cannot leverage useful information accumulated from past observations over time. In this paper, we consider a *continual-context formulation*, where historical context can be accumulated over time beyond the fixed-length input window. Based on this formulation, we propose COLP, which constructs **Co**ntext-carrying **L**atents for each instance and **P**ropagates them along the instance stream, enabling context to accumulate over time. Specifically, our framework treats queries corresponding to future horizons as latent contexts that aggregate prediction-relevant historical information and carry it forward to later instances. Through extensive experiments, we empirically demonstrate the superiority of our framework over baselines across various benchmarks, highlighting the potential of the continual-context formulation.

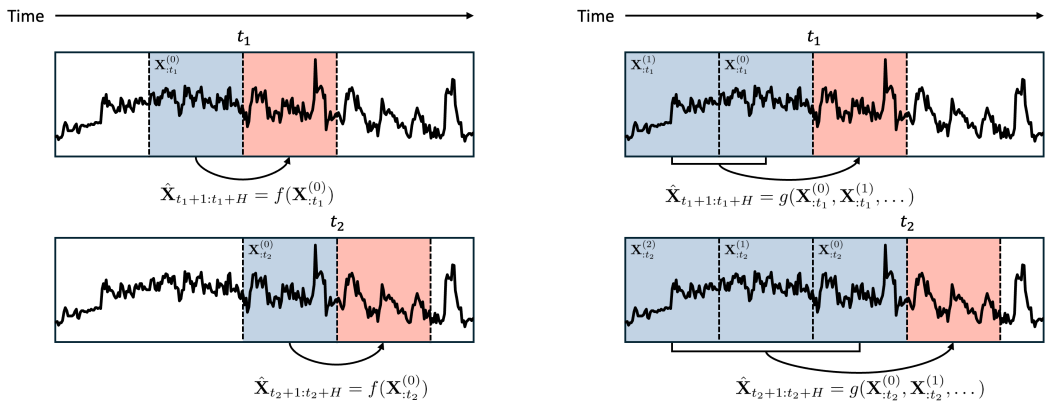
1 INTRODUCTION

Time series forecasting is an important task that aims to capture temporal dependencies from past observations to predict future values. It plays a crucial role in many real-world applications such as energy consumption forecasting, traffic flow prediction, and financial analysis (Lai et al., 2018; Zhou et al., 2021). A common way to formulate this task for deep learning is to divide a continuous time series into forecasting instances (*i.e.*, input-output pairs), each mapping a fixed-length past window to a fixed-length future horizon.

Under this fixed-length formulation, recent studies have proposed a wide range of deep forecasting architectures for modeling fixed-length input windows. Transformer-based approaches (Xue et al., 2024; Liu et al., 2024b; Nie et al., 2023; Lee & Lee, 2025; Kim et al., 2024) leverage attention mechanisms to capture temporal dependencies and variable interactions, while patch-based methods (Nie et al., 2023; Lee & Lee, 2025; Kim et al., 2024; Lin et al., 2026; Wu et al., 2025) represent time series as sequences of local segments to improve representation learning. Decomposition-based architectures (Zeng et al., 2023; Wang et al., 2024a) explicitly separate trend, seasonal, or residual components, and MLP-based models (Wu et al., 2025; Wang et al., 2024a; Das et al., 2023) provide simple yet competitive alternatives for time series forecasting. More recently, recurrent and state-space-based architectures (Lin et al., 2026; Wang et al., 2024b) have been explored to improve sequential modeling efficiency and scalability.

Despite this broad progress in forecasting architectures, forecasting has largely remained tied to the *fixed-length formulation* that treats each input-output pair as a self-contained prediction instance. This formulation overlooks a distinctive property of time series data. Unlike other domains (*e.g.*, image or tabular data), time series observations arrive sequentially over time, and information acquired from earlier instances can remain useful for subsequent predictions. In principle, forecasting models could continuously accumulate historical context as new observations arrive, rather than reconstructing all predictive information from a bounded input window for every instance. Although recent architectures with recurrent (Lin et al., 2026), state-space (Wang et al., 2024b), or memory-

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.



(a) Fixed-length formulation (conventional)

(b) Continual-context formulation (ours)

Figure 1: (a) Conventional *fixed-length formulation* uses only information within each instance, whereas (b) the *continual-context formulation* leverages accumulated observations over time.

like (Behrouz et al., 2025) mechanisms appear well suited for such continual information processing, they are still commonly evaluated by forming each prediction from an independently constructed fixed-length input window. As a result, their potential to maintain and reuse historical context across forecasting instances remains largely unexplored.

Contribution. In this paper, we consider the *continual-context formulation* to explicitly study the ability of forecasting models to accumulate and reuse historical context across sequential forecasting instances. Unlike the conventional *fixed-length formulation*, where each prediction is made from an independently constructed input-output pair, the formulation processes forecasting instances in chronological order and maintains a context state updated from previously observed inputs. Under this formulation, each prediction can condition on both the current input window and the context carried over from preceding instances. As a result, the historical context available for forecasting is no longer limited to the information contained in the current instance, enabling the model to continuously accumulate, update, and reuse useful context over time (see Figure 1).

Based on this formulation, we propose CoLP (**Co**ntext-carrying **L**atent **P**ropagation), a simple yet effective attention-based framework that maintains contextual latents for individual instances and carries them through the instance stream, allowing context to accumulate over time. CoLP operates through three key steps: (*i*) deriving latent tokens from the current input window, (*ii*) propagating information from the previous latent state to the current one, and (*iii*) mapping the refined latent state to future horizons. Specifically, to make the latent tokens predictive of future dynamics, we formulate them as learnable horizon-specific queries and refine them via cross-attention over the current input window. After the queries aggregate information from this window, they receive propagated information from the prior latent state that has accumulated previous observations. The refined latent tokens thus serve as latent states that accumulate observations up to the current instance and are propagated to the next instance. Each token is also projected onto the prediction horizon it represents within the current instance, encouraging horizon-specific information to be distributed across tokens and allowing them to predict distinct future positions.

Through extensive experiments, we demonstrate the strong empirical performance of our framework through comparisons with recent baselines (Liu et al., 2024b; Nie et al., 2023; Lee & Lee, 2025; Kim et al., 2024; Lin et al., 2026; Wu et al., 2025; Wang et al., 2024a;b; Cai et al., 2026) that adopt fixed-length formulation across diverse datasets (Zhou et al., 2021; Lai et al., 2018). Specifically, CoLP achieves **13** best and 2 second-best results among the 16 dataset-metric pairs summarized from 64 horizon-level results per method. We also provide component-wise ablation studies of our framework (Section 4.3), further analyzing the applicability of the continual-context formulation (Section 4.4) and the effectiveness of our training and inference strategies (Section 4.5).

More broadly, our results suggest that time-series forecasting can benefit from considering settings in which historical observations accumulate over time, rather than restricting each prediction to a fixed-length context. We hope this perspective encourages further research on forecasting methods that can effectively retain and leverage such accumulated information.

2 CONTINUAL-CONTEXT FORMULATION FOR TIME SERIES FORECASTING

We begin by establishing notations and the fundamental objective of time-series forecasting. Let $\mathbf{x}_t \in \mathbb{R}^C$ denote the multivariate observation at time step t, where C is the number of variables, and let $\mathbf{X}_{a:b} = [\mathbf{x}_a, \dots, \mathbf{x}_b] \in \mathbb{R}^{C\times(b-a+1)}$ denote a contiguous segment of the series. At its core, time-series forecasting aims to predict the future values $\mathbf{X}_{t+1:t+H}$ over a horizon of length H given all observations $\mathbf{X}_{1:t}$ available up to the current time t:

$$\hat{\mathbf{X}}_{t+1:t+H} = \mathcal{F}(\mathbf{X}_{1:t}), \tag{1}$$

where $\mathcal{F}$ is a forecasting model. While this captures the fundamental nature of the problem, directly modeling the conditional dependence on an unbounded history is challenging in practice.

Fixed-length formulation. A simplest way to approximately model the dependence is to restrict the predictive context to a historical input window of fixed length L:

$$\hat{\mathbf{X}}_{t+1:t+H} = f(\mathbf{X}_{t-L+1:t}). \tag{2}$$

Under this *fixed-length formulation*, a forecasting instance at time step t is constructed as the pair $(\mathbf{X}_{t-L+1:t}, \mathbf{X}_{t+1:t+H})$ of an input window and its corresponding prediction target, the model f is trained and evaluated over a collection of such instances. This formulation has been widely adopted in recent forecasting models (Kim et al., 2025) due to its simplicity, but it treats each forecasting instance as a self-contained prediction problem, even though instances are generated sequentially from the same underlying time series and earlier instances have already been observed by the time later predictions are made.

Continual-context formulation. To overcome this limitation, we consider a *continual-context formulation* that returns to the fundamental objective Eq. (1), where the prediction at time t is conditioned on the entire observed history through the sequence of preceding input windows. To make this explicit, we index past input windows by $n \geq 0$ via $\mathbf{X}_{:t}^{(n)} := \mathbf{X}_{t-(n+1)L+1:t-nL}$, so that $\mathbf{X}_{:t}^{(0)}$ is the current input window of length L used in the fixed-length formulation, $\mathbf{X}_{:t}^{(1)}$ is the immediately preceding window, and so on. Under the continual-context formulation, the prediction is made by

$$\hat{\mathbf{X}}_{t+1:t+H} = g(\mathbf{X}_{:t}^{(0)}, \mathbf{X}_{:t}^{(1)}, \dots), \tag{3}$$

where g is a continual-context model that aggregates information across an arbitrary number of past windows. From this viewpoint, the fixed-length formulation corresponds to the special case in which g retains only $\mathbf{X}_{:t}^{(0)}$ and discards all preceding windows $\{\mathbf{X}_{:t}^{(n)}\}_{n\geq 1}$. Instead, the continual-context formulation allows g to carry context forward across forecasting instances, so that observations from earlier instances remain available when later predictions are made.

A natural way to realize this formulation is through recurrent mechanisms that summarize past observations into hidden states. However, recurrent models in existing forecasting literature (Lin et al., 2026; Wang et al., 2024b) operate within the fixed-length formulation, accumulating information only at the time-step or patch level inside a single input window $\mathbf{X}_{:t}^{(0)}$. Their effective predictive context therefore remains bounded by L, and observations in $\mathbf{X}_{:t}^{(n)}$ for $n \geq 1$ are not carried forward across instances. Moreover, simply extending these recurrent models to the continual-context formulation does not lead to consistent improvements, as empirically examined in Section 4.4. Effectively realizing the continual-context formulation therefore calls for a mechanism specifically designed to propagate context *across* forecasting instances, which we describe in the next section.

3 METHODOLOGY

In this section, we present the proposed method, whose overall framework is illustrated in Figure 2. It realizes the *continual-context formulation* by propagating historical information across forecasting instances. Recall from Eq. (3) that the continual-context model g conditions on the sequence of past input windows $\{\mathbf{X}_{:t}^{(0)}, \mathbf{X}_{:t}^{(1)}, \dots\}$. Rather than re-encoding all preceding windows at every step, we implement g recurrently by maintaining a compact latent state across forecasting instances.

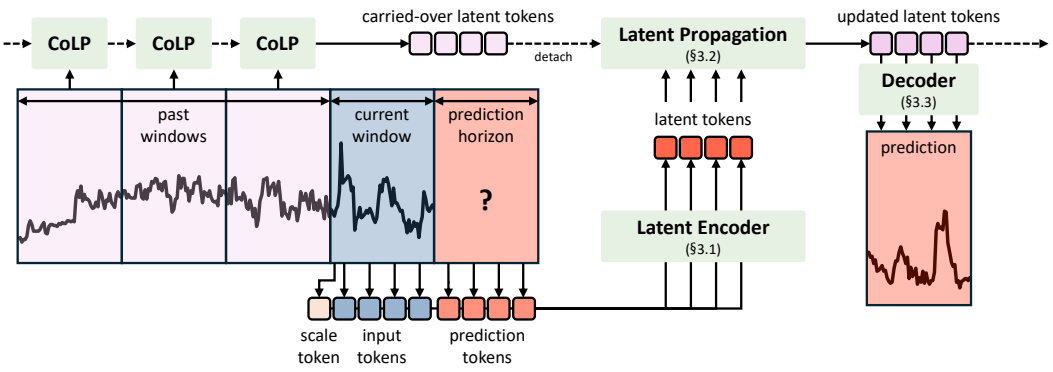


Figure 2: Overview of the proposed CoLP framework.

In analogy with the input windows, we denote by $\mathbf{Z}_{:t}^{(n)}$ the latent state computed at the n-th preceding forecasting instance from time t, so that $\mathbf{Z}_{:t}^{(0)}$ is produced at the current step, $\mathbf{Z}_{:t}^{(1)}$ is the one produced at the previous step, and so on. By construction, the latent state from n instances ago coincides with the current latent state of that earlier instance, i.e., $\mathbf{Z}_{:t}^{(n)} = \mathbf{Z}_{:t-nL}^{(0)}$, where L is the input window size.

At time t, given the current window $\mathbf{X}_{:t}^{(0)}$ and the latent state $\mathbf{Z}_{:t}^{(1)}$ carried over from the previous instance, our continual-context model g produces the forecast in three steps: (*i*) **contextual latent encoder** ($g_{\texttt{enc}}$) encodes the current window into latent tokens, (*ii*) **latent propagator** ($g_{\texttt{prop}}$) propagates previous latent tokens to the current ones, and (*iii*) **decoder** ($g_{\texttt{dec}}$) predicts the future horizon from the updated latent state, formally written as follows:

$$\tilde{\mathbf{Z}}_{:t}^{(0)} = g_{\texttt{enc}}\left(\mathbf{X}_{:t}^{(0)}\right), \quad \mathbf{Z}_{:t}^{(0)} = g_{\texttt{prop}}\left(\tilde{\mathbf{Z}}_{:t}^{(0)}, \mathbf{Z}_{:t}^{(1)}\right), \quad \hat{\mathbf{X}}_{t+1:t+H} = g_{\texttt{dec}}\left(\mathbf{Z}_{:t}^{(0)}\right). \tag{4}$$

The details of $g_{\texttt{enc}}$, $g_{\texttt{prop}}$, and $g_{\texttt{dec}}$ are described in Section 3.1, 3.2, and 3.3, respectively. The training and inference protocols are also provided in Section 3.4. For notation simplicity, in the following subsections, we omit the subscript $:t$ and the superscript (0) when they are clear from context.

3.1 Contextual Latent Aggregation

We first describe the module $g_{\texttt{enc}}$, which constructs context-carrying latent tokens from the current input window. Following recent patch-based forecasting models (Nie et al., 2023; Lee & Lee, 2025; Kudrat et al., 2025), we partition the current input window $\mathbf{X} \in \mathbb{R}^{C\times L}$ into $M = L/P$ non-overlapping patches along the temporal dimension for each channel $c = 1, \ldots, C$, where P is the patch length. Each patch $\mathbf{x}_{c,m} \in \mathbb{R}^P$ is then linearly projected and augmented with learnable temporal and channel positional embeddings, as follows:

$$\mathbf{H}_{c,m} = \mathbf{x}_{c,m}\mathbf{W}_p + \mathbf{E}_m^{\text{temporal}} + \mathbf{E}_c^{\text{channel}}, \quad \mathbf{H} \in \mathbb{R}^{C\times M\times D}, \tag{5}$$

where $\mathbf{W}_p \in \mathbb{R}^{P\times D}$ is a learnable patch projection matrix, and $\mathbf{E}^{\text{temporal}} \in \mathbb{R}^{M\times D}$ and $\mathbf{E}^{\text{channel}} \in \mathbb{R}^{C\times D}$ are learnable temporal and channel positional embeddings, respectively.

Although the patch-token representation above captures local temporal patterns, it does not explicitly retain the scale information removed by reversible instance normalization (RevIN) (Kim et al., 2022), which we apply to each input window before patch construction. This is particularly important in the *continual-context formulation*: since latent states are propagated in the normalized space and predictions are denormalized only at the output, windows with different scales can yield similar latent states when their normalized temporal patterns are similar. To preserve scale-dependent information, we append a scale token to the input patch-token sequence. Let σ_c denote the standard deviation of channel c in the current input window. Given the input patch tokens $\mathbf{H} \in \mathbb{R}^{C\times M\times D}$ in Eq. (5), we construct

$$\mathbf{s}_c = \sigma_c\mathbf{W}_s \in \mathbb{R}^{1\times D}, \quad \tilde{\mathbf{H}}_c = [\mathbf{s}_c; \mathbf{H}_c] \in \mathbb{R}^{(M+1)\times D}, \quad c = 1, \ldots, C, \tag{6}$$

where $\mathbf{W}_s \in \mathbb{R}^{1\times D}$ is a learnable projection matrix. This enables attention to both temporal patch tokens and the scale token, allowing the propagated latent state to retain scale-dependent context in addition to normalized temporal patterns.

Given the scale-aware input token sequence $\tilde{\mathbf{H}}$, we employ prediction query tokens to aggregate information for the prediction horizon $\mathbf{X}_{t+1:t+H}$. The initial prediction queries are constructed from learnable temporal and channel embeddings, $\mathbf{Q}^{\text{temporal}} \in \mathbb{R}^{K\times D}$ and $\mathbf{Q}^{\text{channel}} \in \mathbb{R}^{C\times D}$, which encode the relative position within the prediction horizon and the channel identity, respectively:

$$\mathbf{Q}^{[0]}_{c,k} = \mathbf{Q}^{\text{temporal}}_{k} + \mathbf{Q}^{\text{channel}}_{c}, \quad \mathbf{Q}^{[0]} \in \mathbb{R}^{C\times K\times D}, \tag{7}$$

where $K = H/P$ is the number of patches in the prediction horizon.

The prediction query tokens are then refined by attending to the scale-aware input tokens through R attention blocks:

$$\mathbf{Q}^{[r]}_{c} = \texttt{AttnBlock}^{(r)}\left(\mathbf{Q}^{[r-1]}_{c}, \tilde{\mathbf{H}}_{c}\right), \quad r = 1, \ldots, R, \tag{8}$$

where `AttnBlock` denotes a transformer-style attention block (Vaswani et al., 2017) that consists of multi-head attention and feedforward network with residual connections, in which the first and second arguments are query and key/value tokens, respectively, and the bracketed superscript $[r]$ denotes the refinement depth. Through this refinement, each query retrieves prediction-relevant information from the current input window while also having access to its scale information. The final refined query tokens form the latent tokens produced by $g_{\texttt{enc}}$, i.e., $\tilde{\mathbf{Z}} = g_{\texttt{enc}}(\mathbf{X}) = \mathbf{Q}^{[R]} \in \mathbb{R}^{C\times K\times D}$. Since each latent token is associated with a corresponding prediction-horizon patch, different tokens can capture prediction-relevant information at different future locations.

3.2 Latent Context Propagation

Given the current latent tokens $\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^{(0)}$ constructed from the current input window, and the carried-over latent tokens $\mathbf{Z}^{(1)}$ from the previous forecasting instance, the latent propagation module $g_{\texttt{prop}}$ updates the current latent tokens as follows:

$$\mathbf{Z}^{(0)}_{c} = \texttt{AttnBlock}\left(\tilde{\mathbf{Z}}^{(0)}_{c}, \mathbf{Z}^{(1)}_{c}\right), \quad \mathbf{Z}^{(0)} \in \mathbb{R}^{C\times K\times D}. \tag{9}$$

In this attention operation, $\tilde{\mathbf{Z}}^{(0)}$ serves as the query sequence, while $\mathbf{Z}^{(1)}$ serves as the key and value sequences. This cross-attention allows the current latent tokens to retrieve relevant historical information from the carried-over latent state, enabling context propagation across forecasting instances without re-encoding previous input windows.

3.3 Prediction Decoding

After latent propagation, the decoder $g_{\texttt{dec}}$ predicts the prediction horizon $\mathbf{X}_{t+1:t+H} \in \mathbb{R}^{C\times H}$ from the updated latent state $\mathbf{Z}^{(0)}_{:t} \in \mathbb{R}^{C\times K\times D}$ via a linear projection as follows:

$$\hat{\mathbf{x}}_{c,k} = \mathbf{z}^{(0)}_{:t,c,k}\mathbf{W}_o \in \mathbb{R}^{P}, \quad k = 1, \ldots, K, \tag{10}$$

$$\hat{\mathbf{X}}_{t+1:t+H} = [\hat{\mathbf{x}}_{:,1}, \ldots, \hat{\mathbf{x}}_{:,K}] \in \mathbb{R}^{C\times H}, \tag{11}$$

where $\mathbf{z}^{(0)}_{:t,c,k} \in \mathbb{R}^{D}$ is the k-th latent token of channel c, and $\mathbf{W}_o \in \mathbb{R}^{D\times P}$ is a learnable output projection matrix. Note that the updated latent state $\mathbf{Z}^{(0)}_{:t}$ is not discarded after decoding, but is carried over to the next forecasting instance as its prior latent state. This recurrent propagation allows the prediction at time t to depend on both the current input window and the latent context accumulated from preceding forecasting instances.

3.4 Training and Inference Protocols

Training (Algorithm 1). During training, we construct a finite segment of G consecutive forecasting instances ending at the reference time t:

$$\mathcal{S}_{:t} = \left(\mathcal{D}^{(G-1)}_{:t}, \ldots, \mathcal{D}^{(1)}_{:t}, \mathcal{D}^{(0)}_{:t}\right), \quad \mathcal{D}^{(n)}_{:t} = \left(\mathbf{X}^{(n)}_{:t}, \mathbf{X}_{t-nL+1:t-nL+H}\right), \tag{12}$$

Algorithm 1 Training with Continual Context

Require: $\mathcal{S}_{:t}, g$

$\mathbf{Z}_{:t}^{(G)} \leftarrow \mathbf{0}$
for $n = G-1, \ldots, 1$ **do**
  $(\cdot, \mathbf{Z}_{:t}^{(n)}) \leftarrow g(\mathbf{X}_{:t}^{(n)}, \mathbf{Z}_{:t}^{(n+1)})$
  $\mathbf{Z}_{:t}^{(n)} \leftarrow \mathrm{detach}(\mathbf{Z}_{:t}^{(n)})$
end for
$(\hat{\mathbf{X}}_{t+1:t+H}, \mathbf{Z}_{:t}^{(0)}) \leftarrow g(\mathbf{X}_{:t}^{(0)}, \mathbf{Z}_{:t}^{(1)})$
$\theta \leftarrow \theta - \eta \nabla_\theta \|\hat{\mathbf{X}}_{t+1:t+H} - \mathbf{X}_{t+1:t+H}\|_2^2$

Algorithm 2 Inference with Continual Context

Require: $\mathcal{X}_{1:\infty}, g$

$\mathbf{Z}_{\text{prev}} \leftarrow \mathbf{0}$
for each $\mathbf{X}_{:t_i}^{(0)} \in \mathcal{X}_{1:\infty}$ **do**
  $(\hat{\mathbf{X}}_{t_i+1:t_i+H}, \mathbf{Z}_{:t_i}^{(0)}) \leftarrow g(\mathbf{X}_{:t_i}^{(0)}, \mathbf{Z}_{\text{prev}})$
  Output $\hat{\mathbf{X}}_{t_i+1:t_i+H}$
  $\mathbf{Z}_{\text{prev}} \leftarrow \mathbf{Z}_{:t_i}^{(0)}$
  Move to the next input window
end for

where $n = 0$ denotes the final forecasting instance in the segment, and larger n indicates earlier instances. The first $G-1$ instances, $\mathcal{D}_{:t}^{(G-1)}, \ldots, \mathcal{D}_{:t}^{(1)}$, are used to build the latent context state, while supervision is applied only to the final instance $\mathcal{D}_{:t}^{(0)}$. Specifically, the model processes the instances in chronological order by propagating the latent state forward across forecasting instances, detaching it after each context-building step so that it serves as a fixed prior context for the next instance. This prevents the latent state from being optimized solely for the final target, aligns training with the sequential nature of inference, and avoids backpropagating through the entire context-building segment. After the prior context is propagated to the last instance, we compute the training loss using its final prediction and the corresponding target.

Inference (Algorithm 2). At inference time, forecasting instances are processed sequentially as input windows arrive. Let $\mathcal{X}_{1:\infty} = \{\mathbf{X}_{:t_i}^{(0)}\}_{i=1}^{\infty}$ with $t_i = iL$ denote the ordered input stream, where i denotes the arrival order and L denotes the length of each input window. At time t_i, the model generates the prediction $\hat{\mathbf{X}}_{t_i+1:t_i+H}$ using the latent state carried over from the previous forecasting instance, and the updated latent state $\mathbf{Z}_{:t_i}^{(0)}$ is then propagated to the next instance.

4 Experiments

In this section, we design experiments to examine the following questions:

- Does our continual-context forecasting framework improve both forecasting performance and efficiency over fixed-length forecasting baselines? (§4.2)
- How does each component of our framework contribute to overall performance? (§4.3)
- How should latent states be designed for continual-context forecasting? (§4.4)
- How should the model be trained to accumulate context during continual inference? (§4.5)

4.1 Experimental Setup

Datasets. We conduct experiments on 8 widely used multivariate time-series forecasting benchmarks: ETT (ETTh1, ETTh2, ETTm1, ETTm2), Weather, Electricity, and Traffic (Zhou et al., 2021), and Solar (Lai et al., 2018). These datasets exhibit diverse temporal dependencies, periodic patterns, and non-stationary dynamics, allowing us to evaluate forecasting performance across a broad range of scenarios.

Baselines. To demonstrate the effectiveness of our proposed framework, we compare it with recent state-of-the-art forecasting models from diverse architectural families. Specifically, we consider Transformer-based models (Liu et al., 2024b; Nie et al., 2023; Kim et al., 2024; Lee & Lee, 2025), state-space models (Wang et al., 2024b; Cai et al., 2026), a recurrent model (Lin et al., 2026), and MLP-based forecasting models (Wu et al., 2025; Wang et al., 2024a).

Setting. Since our method follows the *continual-context formulation*, it uses both the current input window and the latent context propagated from previous instances, whereas baselines following the *fixed-length formulation* rely only on the current input window. To avoid favoring our method, we evaluate all methods over multiple input lengths L and report the best result for each method.

Table 1: Comparison on multivariate long-term forecasting. **Bold** indicates the best result and underlined denotes the second best. For each method, we select the input window length with the lowest MSE and report its corresponding MAE. All results are averaged over prediction lengths $H \in \{96, 192, 336, 720\}$. Full results are provided in Appendix A.3.

Models	**CoLP**		TimePerceiver		SRSNet		CATS		SegRNN		TimeMixer		PatchTST		iTransformer		S-Mamba		MambaTS	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**0.215**	**0.257**	0.221	0.263	0.227	0.269	0.220	0.263	0.224	0.270	0.225	0.265	0.224	0.263	0.234	0.273	0.235	0.271	0.226	0.265
Solar	0.182	**0.225**	**0.179**	0.232	0.185	0.242	0.185	0.236	0.186	0.249	0.205	0.264	0.212	0.294	0.198	0.263	0.201	0.264	0.185	0.245
Electricity	**0.154**	**0.248**	0.157	0.249	0.162	0.255	0.158	0.250	0.162	0.261	0.166	0.259	0.162	0.261	0.172	0.271	0.163	0.259	0.159	0.252
Traffic	0.397	**0.263**	0.396	0.274	0.390	0.273	0.389	0.264	0.387	0.270	0.387	0.270	0.399	0.277	0.417	0.315	**0.371**	0.265	0.375	0.267
ETTh1	**0.394**	**0.417**	0.401	0.420	0.406	0.423	0.419	0.435	0.405	0.427	0.423	0.434	0.418	0.433	0.444	0.452	0.442	0.450	0.450	0.451
ETTh2	0.337	**0.385**	**0.334**	0.386	0.340	0.387	0.348	0.393	0.350	0.396	0.359	0.399	0.340	0.386	0.373	0.406	0.368	0.403	0.361	0.402
ETTm1	**0.332**	**0.367**	0.338	0.371	0.352	0.385	0.340	0.377	0.348	0.392	0.350	0.384	0.351	0.386	0.370	0.398	0.365	0.397	0.363	0.389
ETTm2	**0.246**	**0.304**	0.252	0.311	0.252	0.317	0.260	0.324	0.260	0.320	0.262	0.320	0.261	0.321	0.273	0.333	0.264	0.327	0.263	0.327

Specifically, our method is evaluated with $L \in \{96, 384\}$, while fixed-length baselines are evaluated with $L \in \{96, 384, 768\}$, allowing them to use raw input windows up to twice as long as ours; empirically, these baselines largely saturate beyond this range while incurring substantially higher cost. We evaluate prediction lengths $H \in \{96, 192, 336, 720\}$ to assess robustness across diverse forecasting horizons. Ablation studies, such as component analysis, are conducted with $L = 96$; unless otherwise specified, their results are averaged over all four prediction lengths.

Further details on the experimental setup are provided in Appendix A.1.

4.2 Main Results Compare to Baselines

We evaluate forecasting performance using two standard metrics, MSE and MAE, where lower values indicate better performance. For each method, we report the best result selected from its candidate input window lengths, with results averaged over four prediction horizons. As shown in Table 1, among the 16 dataset-metric aggregates obtained from 64 horizon-level results per method, our method achieves **13** best and 2 second-best results, ranking within the top two in 15 out of 16 cases and establishing strong state-of-the-art performance.

Notably, these gains do not come from extending the current input window; instead, our method preserves past information through propagated latent context while keeping the current raw input window shorter than those of fixed-length baselines. As shown in Figure 3, we further examine a representative case on the Weather dataset, comparing our method with two patch-based Transformer baselines: PatchTST (Nie et al., 2023), an early and influential patch-based model, and TimePerceiver (Lee & Lee, 2025), a more recent strong baseline. Our method achieves a lower MSE than both, improving over them by 4.0% and 2.7%, respectively. Importantly, the gain obtained by moving from the more recent fixed-length baseline to our method is larger than the improvement made from the early patch-based baseline to the more recent strong baseline. This suggests that our advantage is not simply due to incremental progress within fixed-length modeling, but stems from our continual-context formulation, which carries useful historical information through propagated latent context rather than relying solely on the current raw input window.

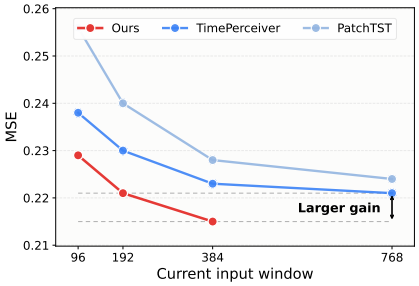


Figure 3: Performance vs. input window on Weather dataset.

4.3 Component Study of Latent Propagation

In this section, we analyze the contribution of the latent propagation mechanism. As shown in Table 2, latent propagation consistently improves performance across all datasets. Specifically, latent propagation reduces the average MSE and MAE by 4.9% and 3.3%, respectively, indicating that the propagated latent state effectively retains historical context beyond the current input window. The scale-aware context token further strengthens this effect, improving the average MSE and MAE by 4.7% and 1.4% over propagation alone by preserving scale information accumulated in the latent state.

Table 2: Component ablation study on latent-state propagation and scale-aware context tokens. **Bold** indicates the best result for each dataset. Full results are provided in Table 11.

Propagation	Scale-aware	ETTh1		ETTm2		Weather		Solar	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
✗	✗	0.423	0.425	0.277	0.321	0.242	0.272	0.265	0.285
✓	✗	0.420	0.423	0.273	0.317	**0.229**	**0.263**	0.226	0.257
✓	✓	**0.398**	**0.417**	**0.270**	**0.314**	**0.229**	0.269	**0.197**	**0.242**

Specifically, in ETTh1, propagation alone provides a modest improvement, while the scale-aware token further amplifies the gain, suggesting that preserving the scale of recurring temporal patterns is important for accurate forecasting. In ETTm2, propagation itself provides a stronger gain, likely because its finer temporal resolution offers dense local continuity and makes longer evolving trends easier to accumulate through the latent state. Weather shows a similar trend-oriented behavior, where propagation is the main source of improvement and the full configuration still remains clearly better than the non-propagation baseline. Notably, Figure 4 provides a clear case where temporal context benefits from retaining scale information. Without the scale-aware context token, observations from different scales can collapse into indistinguishable representations, leading to smoothed or mis-scaled predictions. This case highlights the benefit of combining both components: latent propagation captures recurring generation patterns, while the scale-aware context token distinguishes their amplitude variations.

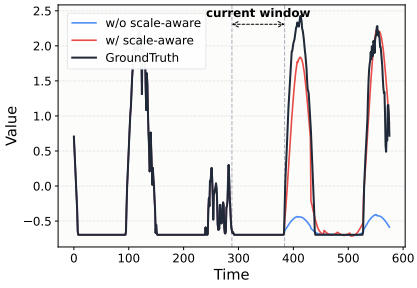


Figure 4: Case study on Solar dataset.

4.4 Latent State Design for Continual-Context Forecasting

Direct adaptation of recurrent models. We examine whether recurrent forecasting architectures can be directly adapted to the continual-context formulation. Although recurrent models naturally carry information over time, they are typically designed for fixed-length forecasting, where the state summarizes the current input window rather than preserving context across forecasting instances. We therefore adapt SegRNN (Lin et al., 2026) and MambaTS (Cai et al., 2026) by constructing chronologically consecutive, non-overlapping instances and propagating the recurrent state across them. As shown in Table 3, this direct state propagation does not improve over fixed-length inference and can even degrade performance, suggesting that simply carrying a recurrent state is insufficient for reliable context accumulation. We attribute this difference to how past information is incorporated. CoLP selectively accesses relevant information from previous states via cross-attention, whereas conventional recurrent-state forecasters propagate past context directly through the recurrent state.

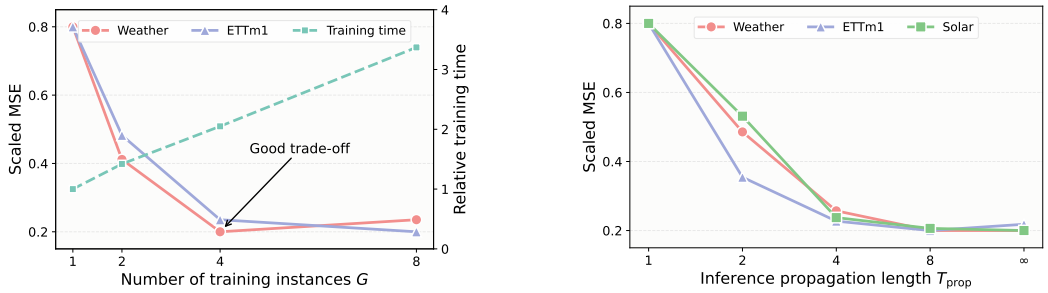
Table 3: Evaluation of recurrent architectures under the continual-context forecasting setting on Weather.

Inference Protocol	**Ours**		SegRNN		MambaTS	
	MSE	MAE	MSE	MAE	MSE	MAE
Fixed-length	0.245	0.277	**0.252**	**0.297**	**0.250**	**0.279**
Continual-context	**0.226**	**0.260**	0.333	0.377	0.257	0.283

Effect of prediction-aligned latent design. CoLP can be viewed as a recurrent latent-state forecaster, and our distinction from conventional recurrent approaches lies in how the recurrent state is represented and learned. Specifically, CoLP constructs prediction-aligned latent tokens as $\mathbf{Z}_t(\mathbf{Q}) = f(\mathbf{X}_t, \mathbf{Z}_{t-1}, \mathbf{Q})$, where the learnable prediction queries $\mathbf{Q}$ explicitly condition the state on future forecasting targets. In contrast, conventional recurrent approaches typically update their state as $\mathbf{Z}_t = \Phi(\mathbf{X}_t, \mathbf{Z}_{t-1})$, where the recurrent state is optimized to summarize the current input and previous memory without explicit alignment to the prediction targets. To examine whether this architectural difference matters empirically, we compare the two update mechanisms while keeping the encoder, state size, effective context, and training setup unchanged. As shown in Table 4, the query-based latent update consistently outperforms the conventional recurrent latent update. These results suggest that prediction-aligned state construction provides a more effective recurrent memory for preserving forecasting-relevant historical context.

Table 4: Latent design comparison.

Latent Design	ETTh1		Weather	
	MSE	MAE	MSE	MAE
Query-based (Ours)	**0.398**	**0.417**	**0.229**	**0.263**
Conventional	0.408	0.428	0.238	0.273



(a) Effect of training segment length

(b) Effect of inference propagation length

Figure 5: Training and inference protocol analysis for continual-context inference.

4.5 Training Strategy for Continual-Context Inference

In this section, we evaluate the training and inference protocol described in Section 3.4, focusing on how the training strategy affects continual-context inference. Since our model is expected to reuse the propagated latent state during inference, training should expose the model to context accumulation while keeping the computational cost manageable. As shown in Figure 5a, increasing the number of consecutive training instances G improves forecasting performance by allowing the model to experience longer latent propagation during training. However, the training cost increases almost linearly as more instances are processed for context construction. In practice, the performance gain rapidly improves at first and then begins to saturate, while the training cost continues to grow. Based on this tradeoff, we use $G = 4$ as the default training setting.

We further examine whether the model trained with this finite context length can still benefit from longer propagation during inference. As shown in Figure 5b, we vary the inference propagation length T_{prop}, the number of consecutive inference instances over which the latent state is carried before reset. When the latent state is reset frequently, the model has limited access to previously accumulated context. As the propagation length increases, forecasting performance generally improves and remains stable near the strongest performance range when the latent state is carried throughout the entire inference stream. This indicates that the model learns to accumulate reusable latent context during training and can effectively leverage longer context when applied to continual-context inference.

5 Related Works

Recent works (Kim et al., 2025) in time series forecasting have focused on leveraging historical information to improve future prediction. Among them, query-based frameworks have emerged as strong baselines. For example, CATS (Kim et al., 2024) replaces decoders based on simple linear projections (Liu et al., 2024b; Nie et al., 2023) with a query-based decoder, demonstrating that strong performance can be achieved without an encoder. Building on this, TimePerceiver (Lee & Lee, 2025) further advances this line of work by enabling a generalized formulation through a query-driven structure, allowing more flexible and effective modeling of temporal dependencies. In addition, another line of research has focused on leveraging richer historical information within a fixed-length formulation, aiming to efficiently process longer input windows through selective patch representations, or compact latent structures (Lee & Lee, 2025; Wu et al., 2025; Liu et al., 2024a; Zhang et al., 2025). In parallel, recurrent and state-space-based approaches (Lin et al., 2026; Wang et al., 2024b) maintain and propagate hidden states within each instance, thereby summarizing historical information. Despite their effectiveness, these approaches still operate under a fixed-length formulation, limiting their ability to explicitly model dependencies across forecasting instances.

6 Conclusion

In this paper, we tackle the limitation of the fixed-length formulation commonly used in prior baselines, where each fixed-length instance is treated as a self-contained prediction problem. To address this limitation, we consider a *continual-context formulation* that enables forecasting models to fully leverage accumulated historical observations. We further design CoLP as a framework for this formulation, achieving strong empirical performance. We hope that this continual-context perspective will motivate further research on time-series forecasting.

AI USE STATEMENT

We use LLMs as auxiliary tools for reproducibility checks, particularly to identify potential inconsistencies in baseline configurations, and to improve the presentation of the manuscript by correcting typographical errors and refining unnatural expressions.

ETHICS STATEMENT

This work focuses on a general methodology for time-series forecasting and does not involve human subjects, private data, or sensitive applications. We therefore do not identify any direct ethical concerns.

REPRODUCIBILITY STATEMENT

The code is provided in the supplementary materials, together with the scripts. Please refer to the accompanying README file for usage instructions. All experiments were conducted using PyTorch (Paszke et al., 2019) on NVIDIA RTX 4090 GPUs.

REFERENCES

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xiuding Cai, Xueyao Wang, Yaoyao Zhu, and Yu Yao. Mambats: improved selective state space models for long-term time series forecasting. *Pattern Recognition*, 2026.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

Dongbin Kim, Jinseong Park, Jaewook Lee, and Hoki Kim. Are self-attentions effective for time series forecasting? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jongseon Kim, Hyungjoon Kim, HyunGi Kim, Dongjun Lee, and Sungroh Yoon. A comprehensive survey of deep learning for time series forecasting: architectural diversity and open challenges. *Artificial Intelligence Review*, 2025.

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 *International Conference on Learning Representations (ICLR)*, 2022.

Dilfira Kudrat, Zongxia Xie, Yanru Sun, Tianyu Jia, and Qinghua Hu. Patch-wise structural loss for time series forecasting. In *International Conference on Machine Learning (ICML)*, 2025.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2018.

Jaebin Lee and Hankook Lee. Timeperceiver: An encoder-decoder framework for generalized time-series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *IEEE Internet of Things Journal*, 2026.

Juncheng Liu, Chenghao Liu, Gerald Woo, Yiwei Wang, Bryan Hooi, Caiming Xiong, and Doyen Sahoo. Unitst: Effectively modeling inter-series and intra-series dependencies for multivariate time series forecasting, 2024a. URL https://arxiv.org/abs/2406.04975.

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 (ICLR)*, 2024b.

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 (ICLR)*, 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, Zach 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 (NeurIPS)*, 2019.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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 *International Conference on Learning Representations (ICLR)*, 2024a.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is mamba effective for time series forecasting?, 2024b. URL https://arxiv.org/abs/2403.11144.

Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. Enhancing time series forecasting through selective representation spaces: A patch perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Wang Xue, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. Card: Channel aligned robust blend transformer for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *The Association for the Advancement of Artificial Intelligence (AAAI)*, 2023.

Jiahui Zhang, Zhengyang Zhou, Wenjie Du, and Yang Wang. Enhancing the maximum effective window for long-term time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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 *The Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

A Appendix

A.1 Implementation Details

Datasets. We evaluate our method on eight widely adopted benchmark datasets for time series forecasting: ETT (ETTh1, ETTh2, ETTm1, and ETTm2), Weather, Electricity, and Traffic (Zhou et al., 2021), Solar (Lai et al., 2018). These datasets cover diverse domains and temporal patterns, with different numbers of variables and time steps.

Table 5: Summary of benchmark datasets used in our experiments.

Dataset	Vars.	Freq.	Steps	Split	Description
ETTh1	7	1 hour	17,420	6:2:2	Hourly ETT subset from two electricity transformer sensors, covering temperature and load-related variables.
ETTh2	7	1 hour	17,420	6:2:2	
ETTm1	7	15 min	69,680	6:2:2	Minute-level ETT subset from two electricity transformer sensors, covering temperature and load-related variables.
ETTm2	7	15 min	69,680	6:2:2	
Weather	21	10 min	52,696	7:1:2	Meteorological measurements collected from a weather station in Germany.
Solar	137	10 min	52,560	7:1:2	Solar power generation records from photovoltaic plants in Alabama.
Electricity	321	1 hour	26,304	7:1:2	Hourly electricity consumption records from multiple clients.
Traffic	862	1 hour	17,544	7:1:2	Hourly road occupancy rates measured by freeway sensors in the San Francisco Bay Area.

Following the standard protocol widely adopted in time series forecasting research (Zhou et al., 2021), we split each dataset chronologically into training, validation, and test sets. For the ETT datasets, we use a 6:2:2 split, while for Weather, Solar, Electricity, and Traffic, we use a 7:1:2 split. During evaluation, we do not discard the last mini-batch, i.e., we set `drop_last=False`, so that all available evaluation samples are included in the validation and test results (Qiu et al., 2024).

Baselines. To validate the effectiveness of the proposed framework, we compare it against recent state-of-the-art forecasting models spanning diverse architectural families. These include Transformer-based methods, such as TimePerceiver (Lee & Lee, 2025), CATS (Kim et al., 2024), PatchTST (Nie et al., 2023), and iTransformer (Liu et al., 2024b), which leverage attention-based representations for time-series forecasting. We further include S-Mamba (Wang et al., 2024b), and MambaTS (Cai et al., 2026) as a state-space model and SegRNN (Lin et al., 2026) as a recurrent model tailored to sequential time-series modeling. Finally, we compare against MLP-based forecasting models, including SRSNet (Wu et al., 2025), and TimeMixer (Wang et al., 2024a), which achieve competitive performance using feed-forward architectures.

Generalized formulation. In relation to the training procedure in Section 3.4, we apply the generalized formulation to the last instance G during training. This formulation includes standard forecasting from past to future as a special case, while further allowing interpolation and extrapolation. Thus, the model can learn diverse representations suited to temporal characteristics. We train all datasets with the same generalized formulation; further details can be found in (Lee & Lee, 2025).

Hyperparameters. All reported results are averaged over five random seeds. We use AdamW optimizer with a weight decay of 0.01. The learning rate is selected from $\{1 \times 10^{-4}, 5 \times 10^{-4}\}$. We train for 100 epochs on Weather, Traffic, and Electricity, and 50 epochs on the ETT-series and Solar datasets. The batch size is generally set to 128, while smaller batch sizes of {16, 32, 64} are used for datasets with many channels, such as Electricity and Traffic.

The embedding dimension D is selected from $\{256, 512\}$, and the feed-forward dimension d_{ff} is set to twice the embedding dimension. The patch size is selected from $\{12, 16, 24, 48\}$. For multi-head attention, we use 4 or 8 attention heads. The use of the scale-token component is selected from $\{0, 1\}$ based on validation performance for each dataset. For the generalized formulation described above, we use a separation ratio from $\{0, 0.5, 1\}$, where the ratio controls how input and target patches are separated within each instance. Specifically, 0 corresponds to fully disjoint target patches, while 1 corresponds to fully continuous target patches. Please refer to (Lee & Lee, 2025) for further details.

Finally, the current input length L is not treated as a single hyperparameter to be selected across all settings. Instead, we tune the model separately for each L and report the corresponding performance with its own selected hyperparameters. Full results across input lengths are provided in Appendix A.3.

A.2 Error Bar

Table 6: Standard deviation of performance results with prediction lengths $H \in \{96, 192, 336, 720\}$ on five random seeds.

Dataset	Weather		Solar		ECL		Traffic	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
96	0.141±0.000	0.193±0.001	0.164±0.001	0.211±0.002	0.123±0.001	0.217±0.001	0.370±0.001	0.251±0.003
192	0.183±0.002	0.234±0.001	0.177±0.001	0.224±0.002	0.143±0.003	0.236±0.002	0.385±0.001	0.256±0.001
336	0.232±0.001	0.274±0.000	0.188±0.002	0.229±0.004	0.158±0.002	0.252±0.001	0.397±0.003	0.262±0.005
720	0.303±0.001	0.327±0.002	0.200±0.001	0.234±0.003	0.193±0.000	0.286±0.002	0.435±0.002	0.283±0.004
Dataset	ETTh1		ETTh2		ETTm1		ETTm2	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
96	0.355±0.002	0.386±0.001	0.274±0.003	0.336±0.002	0.276±0.003	0.328±0.003	0.156±0.001	0.242±0.002
192	0.393±0.001	0.415±0.001	0.336±0.004	0.380±0.003	0.315±0.002	0.356±0.002	0.210±0.002	0.280±0.002
336	0.407±0.003	0.423±0.004	0.344±0.001	0.390±0.002	0.345±0.002	0.377±0.003	0.264±0.004	0.317±0.001
720	0.420±0.003	0.442±0.002	0.394±0.005	0.435±0.004	0.391±0.004	0.408±0.003	0.355±0.001	0.378±0.002

We report the horizon-wise standard deviation of CoLP over five random seeds. As shown in Table 6, 55 of 64 results have a standard deviation of 0.003 or lower, demonstrating that CoLP yields stable performance across five random seeds.

A.3 FULL RESULTS OF LONG-TERM FORECASTING

In this section, we provide the full experimental results corresponding to Table 1. As described in Section 4.1, our method uses candidate input window lengths of $\{96, 384\}$, while all baseline methods use $\{96, 384, 768\}$.

Table 7: Comparison on multivariate long-term forecasting. **Bold** indicates the best result and underlined denotes the second best. For each method, we report the best performance selected from its candidate input window lengths and prediction lengths $H \in \{96, 192, 336, 720\}$

Models		**CoLP**		TimePerceiver		SRSNet		CATS		SegRNN		TimeMixer		PatchTST		iTransformer		S-Mamba		MambaTS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.141**	**0.193**	0.144	0.195	0.150	0.201	0.143	0.197	0.144	0.201	0.148	0.200	0.147	0.201	0.160	0.210	0.157	0.208	0.147	0.199
	192	**0.183**	**0.234**	0.186	0.237	0.194	0.244	0.188	0.240	0.189	0.247	0.192	0.243	0.193	0.241	0.203	0.251	0.203	0.250	0.194	0.244
	336	**0.232**	**0.274**	0.239	0.283	0.245	0.294	0.239	0.282	0.242	0.285	0.242	0.283	0.244	0.282	0.252	0.287	0.256	0.290	0.248	0.281
	720	**0.303**	**0.327**	0.315	0.336	0.317	0.336	0.309	0.333	0.319	0.347	0.319	0.333	0.310	0.328	0.321	0.342	0.322	0.336	0.315	0.335
	Avg	**0.215**	**0.257**	0.221	0.263	0.227	0.269	0.220	0.263	0.224	0.270	0.225	0.265	0.224	0.263	0.234	0.273	0.235	0.271	0.226	0.265
Solar	96	0.164	**0.211**	**0.158**	**0.211**	0.165	0.221	0.163	0.222	0.172	0.237	0.173	0.231	0.191	0.273	0.174	0.242	0.174	0.238	0.162	0.228
	192	0.177	**0.224**	**0.176**	0.228	0.181	0.241	0.189	0.229	0.181	0.247	0.210	0.273	0.211	0.282	0.194	0.259	0.198	0.264	0.182	0.240
	336	0.188	**0.229**	**0.185**	0.246	0.190	0.250	0.189	0.247	0.192	0.254	0.206	0.265	0.221	0.293	0.207	0.272	0.213	0.275	0.193	0.252
	720	0.200	**0.234**	**0.197**	0.244	0.204	0.256	0.199	0.245	0.200	0.257	0.230	0.287	0.225	0.329	0.217	0.279	0.217	0.277	0.201	0.260
	Avg	0.182	**0.225**	**0.179**	0.232	0.185	0.242	0.185	0.236	0.186	0.249	0.205	0.264	0.212	0.294	0.198	0.263	0.201	0.264	0.185	0.245
Electricity	96	**0.123**	**0.217**	0.125	0.219	0.132	0.226	0.126	0.222	0.131	0.229	0.131	0.225	0.130	0.235	0.140	0.240	0.134	0.230	0.129	0.223
	192	0.143	0.236	**0.142**	**0.235**	0.149	0.241	0.147	0.237	0.151	0.248	0.157	0.255	0.148	0.246	0.158	0.258	0.154	0.250	0.146	0.240
	336	**0.158**	**0.252**	0.161	0.254	0.165	0.260	0.162	0.255	0.167	0.267	0.169	0.260	0.168	0.269	0.175	0.275	0.169	0.268	0.161	0.257
	720	**0.193**	**0.286**	0.200	0.288	0.200	0.292	0.196	0.287	0.200	0.299	0.208	0.296	0.202	0.295	0.215	0.310	0.194	0.289	0.198	0.289
	Avg	**0.154**	**0.248**	0.157	0.249	0.162	0.255	0.158	0.250	0.162	0.261	0.166	0.259	0.162	0.261	0.172	0.271	0.163	0.259	0.159	0.252
Traffic	96	0.370	0.251	0.370	0.265	0.358	0.256	0.357	**0.248**	0.352	0.250	0.360	0.259	0.373	0.267	0.380	0.291	**0.342**	0.250	**0.342**	0.253
	192	0.385	**0.256**	0.382	0.268	0.374	0.266	0.379	0.258	0.375	0.263	0.373	0.263	0.384	0.269	0.400	0.306	**0.355**	0.257	0.358	0.259
	336	0.397	**0.262**	0.397	0.272	0.393	0.276	0.393	0.263	0.392	0.274	0.387	0.267	0.399	0.275	0.420	0.317	**0.363**	0.265	0.371	0.270
	720	0.435	**0.283**	0.433	0.289	0.434	0.294	0.426	0.285	0.430	0.292	0.427	0.291	0.439	0.295	0.466	0.344	**0.423**	0.289	0.427	0.285
	Avg	0.397	**0.263**	0.396	0.274	0.390	0.273	0.389	0.264	0.387	0.270	0.387	0.270	0.399	0.277	0.417	0.315	**0.371**	0.265	0.375	0.267
ETTh1	96	0.355	**0.386**	0.355	0.390	0.368	0.393	0.373	0.407	**0.354**	0.393	0.384	0.400	0.376	0.401	0.386	0.405	0.386	0.405	0.381	0.401
	192	0.393	0.415	**0.390**	0.415	0.401	**0.413**	0.416	0.433	0.394	0.417	0.419	0.431	0.410	0.421	0.433	0.452	0.432	0.451	0.429	0.436
	336	**0.407**	0.423	0.413	**0.422**	0.427	0.431	0.435	0.440	0.414	0.430	0.428	0.433	0.434	0.439	0.455	0.458	0.448	0.455	0.450	0.452
	720	**0.420**	**0.442**	0.445	0.453	0.428	0.456	0.451	0.460	0.458	0.467	0.459	0.470	0.451	0.471	0.503	0.491	0.502	0.489	0.540	0.513
	Avg	**0.394**	**0.417**	0.401	0.420	0.406	0.423	0.419	0.435	0.405	0.427	0.423	0.434	0.418	0.433	0.444	0.452	0.442	0.450	0.450	0.451
ETTh2	96	0.274	0.336	**0.272**	**0.333**	0.276	0.337	0.277	0.339	0.285	0.342	0.290	0.342	0.275	0.337	0.297	0.349	0.296	0.348	0.296	0.348
	192	0.336	0.380	**0.333**	0.381	0.334	**0.378**	0.337	0.383	0.334	0.380	0.365	0.401	0.336	**0.378**	0.366	0.402	0.367	0.399	0.362	0.401
	336	**0.344**	**0.390**	0.346	0.399	0.352	0.398	0.370	0.407	0.367	0.408	0.377	0.410	0.352	0.391	0.400	0.427	0.392	0.420	0.383	0.424
	720	0.394	0.435	**0.386**	**0.431**	0.396	0.433	0.408	0.443	0.413	0.452	0.405	0.441	0.397	0.436	0.427	0.445	0.416	0.445	0.401	0.435
	Avg	0.337	**0.385**	**0.334**	0.386	0.340	0.387	0.348	0.393	0.350	0.396	0.359	0.399	0.340	0.386	0.373	0.406	0.368	0.403	0.361	0.402
ETTm1	96	**0.276**	**0.328**	0.283	0.334	0.287	0.350	0.284	0.339	0.294	0.356	0.288	0.351	0.292	0.344	0.314	0.366	0.308	0.360	0.296	0.348
	192	**0.315**	**0.356**	0.321	0.359	0.331	0.371	0.318	0.368	0.329	0.380	0.330	0.371	0.329	0.373	0.347	0.387	0.344	0.382	0.343	0.378
	336	**0.345**	**0.377**	0.348	0.379	0.364	0.392	0.353	0.384	0.357	0.399	0.364	0.393	0.364	0.395	0.383	0.405	0.377	0.406	0.378	0.398
	720	**0.391**	**0.408**	0.399	0.410	0.425	0.426	0.404	0.415	0.412	0.433	0.419	0.421	0.420	0.432	0.437	0.435	0.430	0.440	0.434	0.432
	Avg	**0.332**	**0.367**	0.338	0.371	0.352	0.385	0.340	0.377	0.348	0.392	0.350	0.384	0.351	0.386	0.370	0.398	0.365	0.397	0.363	0.389
ETTm2	96	**0.156**	**0.242**	0.161	0.257	0.163	0.254	0.168	0.262	0.163	0.251	0.174	0.256	0.166	0.255	0.179	0.271	0.178	0.271	0.172	0.263
	192	**0.210**	**0.280**	0.215	0.285	0.219	0.298	0.230	0.307	0.229	0.299	0.230	0.304	0.228	0.299	0.239	0.317	0.233	0.308	0.235	0.309
	336	**0.264**	**0.317**	0.273	0.325	0.275	0.329	0.278	0.335	0.282	0.338	0.272	0.326	0.280	0.331	0.295	0.345	0.281	0.338	0.283	0.341
	720	0.355	0.378	0.357	**0.375**	**0.352**	0.387	0.363	0.391	0.366	0.390	0.370	0.394	0.371	0.398	0.377	0.397	0.362	0.390	0.361	0.394
	Avg	**0.246**	**0.304**	0.252	0.311	0.252	0.317	0.260	0.324	0.260	0.320	0.262	0.320	0.261	0.321	0.273	0.333	0.264	0.327	0.263	0.327

Table 8: Comparison on multivariate long-term forecasting. **Bold** indicates the best result and underlined denotes the second best. The results are reported using input window length $L = 96$ and prediction lengths $H \in \{96, 192, 336, 720\}$.

Models		**CoLP**		TimePerceiver		SRSNet		CATS		SegRNN		TimeMixer		PatchTST		iTransformer		S-Mamba		MambaTS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.151**	**0.198**	0.153	0.202	0.189	0.230	0.162	0.207	0.166	0.228	0.160	0.208	0.175	0.215	0.174	0.214	0.165	0.210	0.175	0.216
	192	**0.193**	**0.238**	0.202	0.247	0.231	0.265	0.207	0.248	0.212	0.274	0.207	0.251	0.219	0.254	0.221	0.254	0.214	0.252	0.224	0.259
	336	**0.244**	**0.280**	0.258	0.288	0.282	0.300	0.264	0.290	0.270	0.318	0.265	0.296	0.277	0.296	0.278	0.296	0.274	0.297	0.278	0.297
	720	**0.326**	**0.335**	0.338	0.340	0.355	0.347	0.343	0.341	0.358	0.377	0.344	0.345	0.352	0.345	0.358	0.347	0.350	0.345	0.355	0.348
	Avg	**0.229**	**0.263**	0.238	0.269	0.264	0.286	0.244	0.272	0.252	0.299	0.244	0.275	0.256	0.278	0.258	0.278	0.251	0.276	0.258	0.280
Solar	96	**0.181**	**0.229**	0.199	0.238	0.231	0.267	0.216	0.247	0.215	0.272	0.224	0.273	0.234	0.286	0.203	0.237	0.205	0.244	0.214	0.262
	192	**0.194**	**0.243**	0.228	0.259	0.270	0.291	0.247	0.267	0.235	0.286	0.238	0.284	0.267	0.310	0.233	0.261	0.237	0.270	0.252	0.290
	336	**0.203**	**0.251**	0.247	0.273	0.296	0.307	0.267	0.277	0.247	0.287	0.250	0.290	0.290	0.315	0.248	0.273	0.258	0.288	0.268	0.301
	720	**0.208**	**0.244**	0.257	0.281	0.341	0.333	0.296	0.293	0.246	0.285	0.246	0.291	0.289	0.317	0.249	0.275	0.260	0.288	0.273	0.304
	Avg	**0.197**	**0.242**	0.233	0.263	0.285	0.300	0.257	0.271	0.236	0.283	0.240	0.284	0.270	0.307	0.233	0.262	0.240	0.273	0.252	0.289
Electricity	96	0.134	0.229	**0.131**	**0.225**	0.176	0.262	0.148	0.236	0.151	0.246	0.156	0.247	0.181	0.270	0.148	0.240	0.139	0.235	0.158	0.249
	192	**0.149**	**0.244**	0.150	**0.244**	0.184	0.269	0.166	0.250	0.165	0.259	0.170	0.261	0.188	0.274	0.162	0.253	0.159	0.255	0.171	0.261
	336	**0.167**	**0.262**	0.168	0.263	0.202	0.286	0.179	0.267	0.181	0.277	0.188	0.278	0.204	0.293	0.178	0.269	0.176	0.272	0.187	0.277
	720	0.207	**0.298**	0.209	**0.298**	0.242	0.319	0.221	0.301	0.217	0.312	0.229	0.313	0.246	0.324	0.225	0.317	**0.204**	**0.298**	0.225	0.308
	Avg	**0.164**	**0.258**	0.165	**0.258**	0.201	0.284	0.179	0.264	0.179	0.274	0.186	0.275	0.205	0.290	0.178	0.270	0.170	0.265	0.185	0.274
Traffic	96	0.408	0.265	0.400	0.263	0.484	0.311	0.421	0.270	0.419	0.269	0.465	0.293	0.462	0.295	0.395	0.268	**0.382**	0.261	0.388	**0.260**
	192	0.418	0.267	0.416	**0.265**	0.482	0.310	0.436	0.276	0.435	0.277	0.505	0.301	0.466	0.296	0.417	0.276	**0.396**	0.267	0.410	0.269
	336	0.434	0.278	0.430	**0.276**	0.506	0.325	0.452	0.283	0.450	0.284	0.524	0.309	0.482	0.304	0.433	0.283	**0.417**	**0.276**	0.426	**0.276**
	720	0.463	**0.292**	0.467	0.297	0.544	0.346	0.484	0.304	0.482	0.302	0.560	0.324	0.514	0.322	0.467	0.302	**0.460**	0.300	**0.460**	0.296
	Avg	0.431	0.276	0.428	**0.275**	0.504	0.323	0.448	0.283	0.447	0.283	0.514	0.307	0.481	0.304	0.428	0.282	**0.414**	0.276	0.421	**0.275**
ETTh1	96	**0.359**	**0.387**	0.372	0.394	0.384	0.396	0.380	0.399	0.363	0.393	0.384	0.400	0.414	0.419	0.386	0.405	0.386	0.405	0.381	0.401
	192	**0.399**	**0.413**	0.422	0.425	0.435	0.424	0.426	0.425	0.411	0.421	0.447	0.434	0.460	0.445	0.441	0.436	0.443	0.437	0.434	0.431
	336	**0.413**	**0.425**	0.444	0.434	0.476	0.445	0.474	0.446	0.447	0.442	0.496	0.459	0.501	0.466	0.487	0.458	0.489	0.468	0.472	0.452
	720	**0.420**	**0.442**	0.451	0.453	0.479	0.475	0.469	0.460	0.458	0.467	0.474	0.469	0.500	0.488	0.503	0.491	0.502	0.489	0.540	0.513
	Avg	**0.398**	**0.417**	0.422	0.427	0.444	0.435	0.437	0.432	0.420	0.431	0.450	0.441	0.469	0.454	0.454	0.447	0.455	0.450	0.457	0.449
ETTh2	96	**0.279**	**0.333**	0.285	0.338	0.288	0.338	0.297	0.347	0.285	0.342	0.290	0.342	0.302	0.348	0.297	0.349	0.296	0.348	0.296	0.348
	192	**0.361**	**0.385**	0.365	0.388	0.375	0.392	0.375	0.396	0.368	0.393	0.378	0.396	0.388	0.400	0.380	0.400	0.376	0.396	0.370	0.393
	336	**0.404**	**0.417**	0.413	0.421	0.414	0.426	0.422	0.431	0.424	0.437	0.432	0.434	0.426	0.433	0.428	0.432	0.424	0.431	0.424	0.434
	720	**0.410**	**0.437**	0.425	0.443	0.427	0.444	0.428	0.444	0.432	0.456	0.468	0.468	0.431	0.446	0.427	0.445	0.426	0.444	0.433	0.446
	Avg	**0.364**	**0.393**	0.372	0.398	0.376	0.400	0.381	0.405	0.377	0.407	0.392	0.410	0.387	0.407	0.383	0.407	0.381	0.405	0.381	0.405
ETTm1	96	**0.299**	**0.346**	0.306	0.347	0.329	0.365	0.315	0.354	0.331	0.370	0.319	0.359	0.329	0.367	0.334	0.368	0.333	0.368	0.330	0.367
	192	**0.336**	**0.368**	0.343	0.369	0.371	0.384	0.356	0.377	0.370	0.392	0.366	0.387	0.367	0.385	0.377	0.391	0.376	0.390	0.377	0.391
	336	**0.371**	0.397	**0.371**	**0.391**	0.399	0.406	0.386	0.399	0.397	0.412	0.387	0.401	0.399	0.410	0.426	0.420	0.408	0.413	0.413	0.414
	720	**0.427**	0.430	0.428	**0.426**	0.473	0.437	0.447	0.436	0.454	0.443	0.449	0.439	0.454	0.439	0.491	0.459	0.475	0.448	0.476	0.449
	Avg	**0.359**	0.385	0.362	**0.383**	0.393	0.398	0.376	0.392	0.388	0.404	0.380	0.396	0.387	0.400	0.407	0.410	0.398	0.405	0.399	0.405
ETTm2	96	**0.168**	**0.247**	0.170	0.249	0.182	0.268	0.180	0.261	0.173	0.256	0.174	0.256	0.175	0.259	0.180	0.264	0.179	0.263	0.179	0.265
	192	**0.230**	**0.288**	0.236	0.293	0.243	0.307	0.243	0.304	0.237	0.298	0.238	0.300	0.241	0.302	0.250	0.309	0.250	0.309	0.247	0.308
	336	**0.293**	**0.330**	0.298	**0.330**	0.306	0.346	0.305	0.344	0.296	0.336	0.297	0.340	0.305	0.343	0.311	0.348	0.312	0.349	0.311	0.349
	720	**0.388**	**0.390**	0.399	0.392	0.407	0.402	0.407	0.402	0.389	0.407	0.396	0.398	0.402	0.400	0.412	0.407	0.411	0.406	0.408	0.404
	Avg	**0.270**	**0.314**	0.276	0.316	0.285	0.331	0.284	0.328	0.274	0.324	0.276	0.324	0.281	0.326	0.288	0.332	0.288	0.332	0.286	0.332

Table 9: Comparison on multivariate long-term forecasting. **Bold** indicates the best result and underlined denotes the second best. The results are reported using input window length $L = 384$ and prediction lengths $H \in \{96, 192, 336, 720\}$.

Models		**CoLP**		TimePerceiver		SRSNet		CATS		SegRNN		TimeMixer		PatchTST		iTransformer		S-Mamba		MambaTS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.141**	**0.193**	0.144	0.195	0.150	0.201	0.147	0.199	0.146	0.202	0.148	0.200	0.150	0.200	0.160	0.210	0.157	0.208	0.151	0.201
	192	**0.183**	**0.234**	0.190	0.243	0.194	0.244	0.191	0.242	0.190	0.244	0.192	0.243	0.194	0.241	0.203	0.251	0.203	0.250	0.194	0.244
	336	**0.232**	**0.274**	0.242	0.279	0.245	0.294	0.244	0.280	0.242	0.285	0.242	0.283	0.245	0.283	0.252	0.287	0.256	0.290	0.248	0.281
	720	**0.303**	**0.327**	0.315	0.336	0.317	0.336	0.314	0.331	0.324	0.343	0.319	0.333	0.323	0.333	0.325	0.339	0.322	0.336	0.330	0.339
	Avg	**0.215**	**0.257**	0.223	0.263	0.227	0.269	0.224	0.263	0.225	0.269	0.225	0.265	0.228	0.264	0.235	0.272	0.235	0.272	0.231	0.266
Solar	96	0.164	**0.211**	**0.163**	0.213	0.180	0.227	0.172	0.216	0.172	0.237	0.173	0.231	0.197	0.291	0.187	0.246	0.194	0.243	0.186	0.245
	192	0.177	**0.224**	**0.176**	0.228	0.193	0.243	0.189	0.229	0.181	0.247	0.210	0.273	0.214	0.319	0.209	0.269	0.221	0.271	0.197	0.256
	336	0.188	**0.229**	**0.185**	0.246	0.197	0.246	0.197	0.236	0.192	0.254	0.206	0.265	0.235	0.328	0.222	0.281	0.217	0.273	0.208	0.265
	720	0.200	**0.234**	**0.197**	0.244	0.206	0.256	0.203	0.245	0.200	0.257	0.230	0.287	0.225	0.329	0.231	0.291	0.230	0.286	0.206	0.263
	Avg	0.182	**0.225**	**0.180**	0.233	0.194	0.243	0.190	0.232	0.186	0.249	0.205	0.264	0.218	0.317	0.212	0.272	0.216	0.268	0.199	0.257
Electricity	96	**0.123**	**0.217**	0.125	0.219	0.132	0.226	0.128	0.224	0.132	0.229	0.131	0.225	0.133	0.232	0.140	0.240	0.134	0.230	0.130	0.224
	192	0.143	0.236	**0.142**	**0.235**	0.149	0.241	0.147	0.237	0.152	0.247	0.161	0.258	0.148	0.246	0.158	0.258	0.154	0.250	0.147	0.240
	336	**0.158**	**0.252**	0.161	0.254	0.166	0.259	0.162	0.255	0.170	0.266	0.169	0.260	0.169	0.265	0.175	0.275	0.174	0.271	0.162	0.257
	720	**0.193**	**0.286**	0.200	0.288	0.202	0.292	0.198	**0.286**	0.207	0.299	0.208	0.296	0.202	0.295	0.215	0.310	0.197	0.292	0.198	0.289
	Avg	**0.154**	**0.248**	0.157	0.249	0.162	0.255	0.159	0.251	0.165	0.260	0.167	0.260	0.163	0.260	0.172	0.271	0.165	0.261	0.159	0.253
Traffic	96	0.370	0.251	0.373	0.262	0.368	0.258	0.358	**0.248**	0.360	0.251	0.376	0.258	0.385	0.277	0.393	0.298	**0.346**	0.250	0.348	0.253
	192	0.385	**0.256**	0.383	0.263	0.390	0.271	0.386	0.266	0.382	0.262	0.395	0.267	0.401	0.283	0.414	0.312	**0.355**	0.257	0.372	0.258
	336	0.397	**0.262**	0.399	0.270	0.402	0.274	0.393	0.263	0.396	0.270	0.410	0.276	0.410	0.287	0.432	0.322	**0.363**	0.265	0.383	0.264
	720	0.435	**0.283**	0.433	0.289	0.438	0.293	0.435	0.295	0.430	0.293	0.453	0.288	0.443	0.304	0.472	0.347	**0.423**	0.289	0.427	0.285
	Avg	0.397	**0.263**	0.397	0.271	0.400	0.274	0.393	0.268	0.392	0.269	0.408	0.272	0.410	0.288	0.428	0.320	**0.372**	0.265	0.383	0.265
ETTh1	96	**0.355**	**0.386**	0.366	0.393	0.368	0.393	0.381	0.405	**0.355**	0.393	0.387	0.413	0.376	0.401	0.413	0.427	0.398	0.419	0.401	0.420
	192	**0.393**	0.415	0.394	**0.411**	0.401	0.413	0.416	0.433	0.395	0.418	0.419	0.431	0.410	0.421	0.449	0.450	0.436	0.444	0.429	0.436
	336	**0.407**	0.423	0.413	**0.422**	0.427	0.431	0.435	0.440	0.414	0.430	0.428	0.433	0.434	0.439	0.455	0.458	0.448	0.455	0.450	0.452
	720	0.432	**0.450**	0.445	0.453	**0.428**	0.456	0.451	0.460	0.467	0.472	0.472	0.473	0.451	0.471	0.528	0.521	0.508	0.508	0.601	0.538
	Avg	**0.397**	**0.419**	0.404	0.420	0.406	0.423	0.421	0.435	0.408	0.428	0.426	0.438	0.418	0.433	0.461	0.464	0.448	0.457	0.470	0.461
ETTh2	96	0.274	0.336	**0.272**	0.333	0.276	0.337	0.283	0.346	0.294	0.344	0.293	0.350	0.275	0.337	0.306	**0.326**	0.300	0.357	0.303	0.358
	192	0.336	0.380	**0.333**	0.381	0.334	0.378	0.345	0.384	0.359	0.391	0.368	0.401	0.338	**0.375**	0.366	0.402	0.367	0.399	0.364	0.395
	336	**0.344**	**0.390**	0.346	0.399	0.352	0.398	0.372	0.406	0.375	0.407	0.377	0.410	0.352	0.391	0.400	0.427	0.392	0.420	0.386	0.415
	720	0.394	0.435	**0.386**	**0.431**	0.396	0.433	0.408	0.443	0.413	0.452	0.444	0.464	0.398	0.435	0.435	0.454	0.416	0.445	0.401	0.435
	Avg	0.337	**0.385**	**0.334**	0.386	0.340	0.387	0.352	0.395	0.360	0.399	0.371	0.406	0.341	**0.385**	0.377	0.402	0.369	0.405	0.364	0.401
ETTm1	96	**0.276**	**0.328**	0.283	0.334	0.293	0.349	0.284	0.339	0.300	0.358	0.289	0.342	0.292	0.344	0.314	0.366	0.308	0.360	0.296	0.348
	192	**0.315**	**0.356**	0.321	0.359	0.331	0.371	0.323	0.363	0.337	0.382	0.330	0.371	0.340	0.373	0.353	0.389	0.344	0.382	0.343	0.378
	336	**0.345**	**0.377**	0.352	0.378	0.364	0.392	0.356	0.384	0.365	0.400	0.364	0.393	0.367	0.394	0.383	0.405	0.378	0.403	0.378	0.398
	720	**0.391**	**0.408**	0.399	0.410	0.433	0.424	0.411	0.421	0.417	0.431	0.429	0.420	0.422	0.427	0.437	0.435	0.439	0.441	0.434	0.432
	Avg	**0.332**	**0.367**	0.338	0.370	0.355	0.384	0.344	0.377	0.355	0.393	0.353	0.382	0.355	0.385	0.372	0.399	0.367	0.397	0.363	0.389
ETTm2	96	**0.156**	**0.242**	0.161	0.257	0.163	0.254	0.172	0.260	0.166	0.252	0.174	0.264	0.166	0.255	0.179	0.271	0.181	0.269	0.172	0.263
	192	**0.210**	**0.280**	0.215	0.285	0.223	0.297	0.234	0.306	0.232	0.299	0.246	0.306	0.228	0.299	0.243	0.313	0.233	0.308	0.243	0.311
	336	**0.264**	**0.317**	0.273	0.325	0.275	0.329	0.278	0.335	0.290	0.337	0.272	0.326	0.280	0.331	0.295	0.345	0.281	0.338	0.287	0.344
	720	**0.355**	0.378	0.357	**0.375**	0.373	0.393	0.363	0.391	0.366	0.390	0.381	0.400	0.373	0.389	0.377	0.397	0.368	0.391	0.366	0.390
	Avg	**0.246**	**0.304**	0.252	0.311	0.259	0.318	0.262	0.323	0.264	0.319	0.268	0.324	0.262	0.319	0.273	0.332	0.266	0.327	0.267	0.327

Table 10: Comparison on multivariate long-term forecasting. **Bold** indicates the best result and underlined denotes the second best. The results are reported using input window length $L = 768$ and prediction lengths $H \in \{96, 192, 336, 720\}$.

Models		**CoLP**		TimePerceiver		SRSNet		CATS		SegRNN		TimeMixer		PatchTST		iTransformer		S-Mamba		MambaTS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	-	-	0.144	0.200	0.150	0.206	**0.143**	**0.197**	0.144	0.201	0.155	0.211	0.147	0.201	0.165	0.217	0.165	0.219	0.147	0.199
	192	-	-	**0.186**	**0.237**	0.200	0.255	0.188	0.240	0.189	0.247	0.203	0.257	0.193	0.241	0.205	0.253	0.210	0.259	0.195	0.245
	336	-	-	**0.239**	0.283	0.251	0.295	**0.239**	**0.282**	0.244	0.294	0.253	0.294	0.244	**0.282**	0.259	0.296	0.267	0.301	0.252	0.289
	720	-	-	0.316	0.337	0.332	0.349	**0.309**	0.333	0.319	0.347	0.337	0.347	0.310	**0.328**	0.321	0.342	0.330	0.346	0.315	0.335
	Avg	-	-	0.221	0.264	0.233	0.276	**0.220**	**0.263**	0.224	0.273	0.237	0.277	0.224	**0.263**	0.238	0.277	0.243	0.281	0.227	0.267
Solar	96	-	-	**0.158**	**0.211**	0.165	0.221	0.163	0.222	0.175	0.244	0.191	0.256	0.191	0.273	0.174	0.242	0.174	0.238	0.162	0.228
	192	-	-	**0.180**	**0.228**	0.181	0.241	0.193	0.234	0.186	0.249	0.217	0.268	0.211	0.282	0.194	0.259	0.198	0.264	0.182	0.240
	336	-	-	**0.186**	**0.225**	0.190	0.250	0.189	0.247	0.196	0.259	0.229	0.279	0.221	0.293	0.207	0.272	0.213	0.275	0.193	0.252
	720	-	-	**0.197**	0.253	0.204	0.256	0.199	**0.245**	0.201	0.260	0.233	0.288	0.235	0.297	0.217	0.279	0.217	0.277	0.201	0.260
	Avg	-	-	**0.180**	**0.229**	0.185	0.242	0.186	0.237	0.189	0.253	0.218	0.273	0.215	0.286	0.198	0.263	0.201	0.264	0.185	0.245
Electricity	96	-	-	0.128	0.225	0.132	0.226	**0.126**	**0.222**	0.131	0.229	0.137	0.235	0.130	0.235	0.142	0.241	0.134	0.232	0.129	0.223
	192	-	-	0.147	0.241	0.150	0.243	0.149	0.243	0.151	0.248	0.157	0.255	0.153	0.249	0.160	0.258	0.157	0.253	**0.146**	**0.240**
	336	-	-	0.165	0.260	0.165	0.260	0.171	0.263	0.167	0.267	0.176	0.276	0.168	0.269	0.179	0.277	0.169	0.268	**0.161**	**0.257**
	720	-	-	0.202	0.293	0.200	0.292	0.196	**0.287**	0.200	0.299	0.215	0.310	0.205	0.298	0.221	0.312	**0.194**	0.289	0.201	0.293
	Avg	-	-	0.160	0.255	0.162	0.255	0.161	0.254	0.162	0.261	0.171	0.269	0.164	0.263	0.176	0.272	0.164	0.261	**0.159**	**0.253**
Traffic	96	-	-	0.370	0.265	0.358	0.256	0.357	**0.248**	0.352	0.250	0.360	0.259	0.373	0.267	0.380	0.291	**0.342**	0.250	**0.342**	0.253
	192	-	-	0.382	0.268	0.374	0.266	0.379	**0.258**	0.375	0.263	0.373	0.263	0.384	0.269	0.400	0.306	0.362	0.262	**0.358**	0.259
	336	-	-	0.397	0.272	0.393	0.276	0.393	**0.265**	0.392	0.274	0.387	0.267	0.399	0.275	0.420	0.317	**0.366**	0.268	0.371	0.270
	720	-	-	0.436	0.292	0.434	0.294	**0.426**	**0.285**	0.430	0.292	0.427	0.291	0.439	0.295	0.466	0.344	0.430	0.295	0.428	0.286
	Avg	-	-	0.396	0.274	0.390	0.273	0.389	**0.264**	0.387	0.270	0.387	0.270	0.399	0.277	0.417	0.315	**0.375**	0.269	**0.375**	0.267
ETTh1	96	-	-	0.355	**0.390**	0.377	0.405	0.373	0.407	**0.354**	0.393	0.389	0.418	0.379	0.410	0.401	0.428	0.393	0.425	0.388	0.418
	192	-	-	**0.390**	**0.415**	0.413	0.426	0.416	0.441	0.394	0.417	0.421	0.438	0.416	0.433	0.433	0.452	0.432	0.451	0.436	0.449
	336	-	-	**0.419**	0.436	0.442	0.444	0.456	0.464	0.425	**0.433**	0.486	0.480	0.440	0.451	0.475	0.481	0.474	0.479	0.467	0.470
	720	-	-	0.455	0.477	**0.439**	**0.467**	0.503	0.489	0.481	0.482	0.459	0.470	0.469	0.483	0.567	0.549	0.532	0.528	0.594	0.540
	Avg	-	-	**0.405**	**0.430**	0.418	0.436	0.437	0.450	0.413	0.431	0.439	0.452	0.426	0.444	0.469	0.478	0.458	0.471	0.471	0.469
ETTh2	96	-	-	0.278	**0.338**	0.276	0.342	0.277	0.339	0.288	0.345	0.292	0.350	**0.275**	0.339	0.307	0.363	0.298	0.356	0.296	0.356
	192	-	-	0.344	0.395	0.344	0.391	0.337	0.383	**0.334**	0.380	0.365	0.401	0.336	**0.378**	0.441	0.442	0.370	0.406	0.362	0.401
	336	-	-	**0.350**	0.409	0.354	**0.401**	0.370	0.407	0.367	0.408	0.418	0.441	0.357	**0.401**	0.452	0.456	0.394	0.425	0.383	0.424
	720	-	-	**0.395**	0.441	0.404	0.441	0.435	0.468	0.423	0.466	0.405	0.441	0.397	**0.436**	0.434	0.462	0.428	0.454	0.415	0.454
	Avg	-	-	0.342	0.396	0.345	0.394	0.355	0.399	0.353	0.400	0.370	0.408	**0.341**	**0.389**	0.409	0.431	0.373	0.410	0.364	0.409
ETTm1	96	-	-	**0.283**	**0.337**	0.287	0.350	0.285	0.340	0.294	0.356	0.288	0.351	0.295	0.351	0.316	0.370	0.310	0.367	0.301	0.354
	192	-	-	0.324	**0.366**	0.343	0.380	**0.318**	0.368	0.329	0.380	0.341	0.379	0.329	0.373	0.347	0.387	0.350	0.390	0.350	0.381
	336	-	-	**0.348**	**0.379**	0.377	0.407	0.353	0.384	0.357	0.399	0.376	0.405	0.364	0.395	0.388	0.412	0.377	0.406	0.383	0.403
	720	-	-	0.410	**0.409**	0.425	0.426	**0.404**	0.415	0.412	0.433	0.419	0.421	0.420	0.432	0.460	0.456	0.430	0.440	0.458	0.443
	Avg	-	-	0.342	**0.373**	0.358	0.391	**0.340**	0.377	0.348	0.392	0.356	0.389	0.352	0.388	0.378	0.406	0.367	0.401	0.373	0.395
ETTm2	96	-	-	0.166	0.257	0.167	0.259	0.168	0.262	**0.163**	**0.251**	0.175	0.265	0.182	0.272	0.185	0.278	0.178	0.271	0.172	0.264
	192	-	-	0.224	**0.293**	**0.219**	0.298	0.230	0.307	0.229	0.299	0.230	0.304	0.247	0.314	0.239	0.317	0.235	0.311	0.235	0.309
	336	-	-	**0.278**	**0.327**	0.285	0.339	0.283	0.339	0.282	0.338	0.284	0.341	0.303	0.354	0.298	0.355	0.290	0.347	0.283	0.341
	720	-	-	0.357	**0.384**	**0.352**	0.387	0.367	0.391	0.372	0.400	0.370	0.394	0.371	0.398	0.401	0.416	0.362	0.390	0.361	0.394
	Avg	-	-	**0.256**	**0.315**	**0.256**	0.321	0.262	0.325	0.262	0.322	0.265	0.326	0.276	0.335	0.281	0.341	0.266	0.330	0.263	0.327

A.4 Full Results of Ablation Study

Table 11: Full results of the component ablation study on latent-state propagation and scale-aware context tokens, extending Table 2. The best result for each dataset and horizon is highlighted in **bold**.

Propagation	Scale-aware	H	ETTh1		ETTm2		Weather		Solar	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
✗	✗	96	0.373	0.394	0.174	0.255	0.160	0.208	0.229	0.262
		192	0.416	0.420	0.239	0.298	0.205	0.249	0.260	0.284
		336	0.444	0.434	0.297	0.336	0.261	0.289	0.288	0.306
		720	0.460	0.453	0.398	0.395	0.342	0.342	0.282	0.289
		Avg	0.423	0.425	0.277	0.321	0.242	0.272	0.265	0.285
✓	✗	96	0.369	0.391	0.173	0.250	**0.151**	**0.198**	0.186	**0.228**
		192	0.417	0.418	0.235	0.292	**0.193**	**0.238**	0.219	0.257
		336	0.441	0.432	0.295	0.336	**0.244**	**0.280**	0.245	0.265
		720	0.456	0.451	0.390	0.391	0.326	**0.335**	0.252	0.276
		Avg	0.420	0.423	0.273	0.317	**0.229**	**0.263**	0.226	0.257
✓	✓	96	**0.359**	**0.387**	**0.168**	**0.247**	0.153	0.205	**0.181**	0.229
		192	**0.399**	**0.413**	**0.230**	**0.288**	**0.193**	0.244	**0.194**	**0.243**
		336	**0.413**	**0.425**	**0.293**	**0.330**	0.248	0.288	**0.203**	**0.251**
		720	**0.420**	**0.442**	**0.388**	**0.390**	**0.322**	0.339	**0.208**	**0.244**
		Avg	**0.398**	**0.417**	**0.270**	**0.314**	**0.229**	0.269	**0.197**	**0.242**

Table 11 provides the full results of the component ablation study corresponding to Table 2. Detailed discussion of the ablation setting and main findings is provided in Section 4.3.

Overall, the per-horizon results show trends consistent with the averaged results in Table 2. Latent-state propagation generally improves performance over the non-propagation baseline, indicating that accumulated historical context is useful across different prediction lengths. Adding scale-aware context tokens further strengthens the results in most cases, especially on datasets where scale variation plays an important role. These observations support that the two components provide complementary benefits and that the full configuration offers a robust design across datasets and forecasting horizons.

A.5 Additional Ablation Studies

Propagation as a plug-in module. The previous result shows that simply carrying the internal state of an existing fixed-length model is not sufficient for reliable continual-context inference. We next examine whether our latent propagation design can be used as a plug-in module for existing patch-based forecasting models. For example, in PatchTST (Nie et al., 2023), we retain the original patch-based encoder and attach our latent propagation module to the encoded patch representations. This enables the model to accumulate context across instances without redesigning its encoding architecture. As shown in Table 12, this simple integration improves performance, demonstrating that latent propagation can serve as a general module for extending patch-based models to the continual-context setting.

Table 12: Plug-in extension for continual-context forecasting with PatchTST (Nie et al., 2023) on ETTm2.

Model	**Ours**		PatchTST	
Metric	MSE	MAE	MSE	MAE
Fixed form.	0.277	0.321	0.281	0.326
Continual form.	**0.270**	**0.314**	**0.273**	**0.316**

Effect of the detach design. We also ablate the gradient detachment in latent propagation. In our design, the propagated latent state is detached and used as prior context for the next instance. While removing this detachment allows gradients to pass through the propagation path, it does not provide a clear advantage, since the query representation serves not only as a latent carrier but also as the direct output for forecasting the current instance. As shown in Table 13, removing detachment on Weather yields comparable or slightly degraded performance, but increases memory usage by 3.0× and

Table 13: Effect of gradient detachment.

Metric	w/ detach	w/o detach
Performance (MSE)	0.193	0.195
Memory (MB)	2608	7922
Time (s/iter)	0.0572	0.1153

training time by $2.0\times$ at $H = 192$. We therefore keep the propagated latent state detached, preserving both effectiveness and efficiency.