DECOMPOSED LOCAL-TO-GLOBAL RETRIEVAL FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series forecasting fundamentally requires capturing both short-term dynamics and long-term structural patterns. While parametric models often struggle to internalize these diverse dependencies from limited data, retrieval-based approaches offer a promising alternative by explicitly reusing informative historical patterns. However, existing methods typically perform retrieval in a monolithic feature space, overlooking that varying temporal structures (e.g., trend vs. seasonality) require distinct matching mechanisms. In this paper, we propose DLoG (Decomposed Local-to-Global retrieval), a framework that augments forecasting by leveraging decomposed historical patterns as inductive biases. DLoG specifically addresses the multi-scale nature of time series through two key designs: (1) Component-wise Retrieval, which decomposes series into complementary components to enable targeted pattern alignment; and (2) Dual-stage Retrieval, which first retrieves from a local window to preserve short-term consistency, followed by a global search to incorporate long-range structural regularities. By integrating these component-specific future references, DLoG reduces the burden on purely parametric forecasting by explicitly reusing informative historical patterns. Extensive experiments on nine benchmark datasets show that DLoG achieves best or second-best average MSE and MAE across all evaluated datasets, demonstrating strong and consistent performance against competitive forecasting baselines.

1 INTRODUCTION

Time series forecasting is a cornerstone of modern data-driven decision-making, underpinning critical applications ranging from grid load management and traffic flow optimization to financial market analysis (Mahalakshmi et al., 2016; Lim & Zohren, 2021). The fundamental challenge in this domain lies in the heterogeneous nature of temporal data: a forecasting model must simultaneously capture short-term dynamics (e.g., immediate fluctuations and local continuity) and long-term structural regularities (e.g., evolving trends and recurring seasonal periodicities) (Hewamalage et al., 2021b; Bai et al., 2018).

The dominant paradigm in recent years has shifted toward purely parametric deep learning models, such as Transformers and MLPs (Wu et al., 2021b; Zhou et al., 2022; Lin et al., 2024). These models attempt to internalize complex temporal dependencies within their weight parameters. While powerful, parametric approaches often face inherent limitations: they can be data-inefficient, prone to overfitting when data is sparse, and brittle when facing non-stationary environments where the distribution of future data drifts from the training set (Gu et al., 2021; Hewamalage et al., 2021b). Consequently, they often struggle to generalize distinct temporal structures from limited observations.

To mitigate the reliance on parametric memorization, retrieval-based forecasting has emerged as a promising alternative (Lewis et al., 2020; Guu et al., 2020; Shuster et al., 2021; Gao et al., 2023). By treating historical data as an external memory store, these methods explicitly reuse informative past patterns as evidence for future predictions. This transforms the forecasting problem from generation from scratch to conditional pattern reuse, effectively supplying the model with observed historical future references (Gao et al., 2023). However, existing retrieval-augmented frameworks typically perform retrieval in a monolithic feature space. They treat the time series look-back window as a single, unified vector, ignoring a critical property of temporal data: different underlying components,

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.

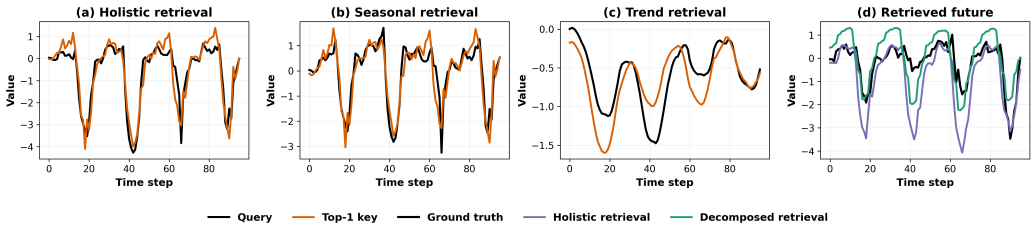


Figure 1: A representative retrieval example on the real-world ETTh1 dataset. (a) Holistic retrieval matches the raw entangled sequence and may retrieve a key with misleading overall similarity. (b) Seasonal retrieval aligns periodic patterns in the adaptive seasonal space. (c) Trend retrieval matches long-term evolution in the adaptive trend space with trend-aware similarity. (d) By aggregating component-specific future values, decomposed retrieval produces a future reference that better follows the ground-truth trajectory than holistic retrieval.

including trend and seasonality, possess fundamentally different semantic structures and matching logic.

Figure 1 illustrates this limitation through a concrete retrieval example. In the holistic setting, retrieval is performed directly on the raw sequence where trend and seasonal structures are entangled. As a result, the retrieved top-1 key may appear similar in the mixed space, but its underlying temporal semantics can still be mismatched, leading to an unreliable future reference. In contrast, decomposed retrieval separates the matching process into seasonal and trend spaces. The seasonal branch aligns periodic patterns with shift-tolerant correlation, reducing the interference from long-term level changes, while the trend branch matches the smooth trajectory using both level similarity and first-order difference similarity. By aggregating the retrieved seasonal future and trend future, the decomposed reference follows the ground truth more closely than the holistic retrieval result. This example suggests that effective retrieval for forecasting should not rely on a single monolithic similarity criterion, but should instead respect the distinct matching logic of different temporal components.

Motivated by this observation, we propose Decomposed Local-to-Global retrieval (DLoG), a framework that augments forecasting by leveraging decomposed historical patterns as structural inductive biases. DLoG introduces two key designs. First, Component-wise Retrieval decomposes each series into complementary trend and seasonal components and performs targeted retrieval in the corresponding component spaces, enabling the model to align trend trajectories and seasonal phases separately. Second, a Dual-stage Local-to-Global Strategy first retrieves locally consistent patterns from a short temporal window and then conducts global retrieval over historical memory conditioned on the local context. This design allows the model to preserve short-term continuity while still exploiting long-range recurring structures.

By integrating component-specific future references, DLoG reduces the burden on purely parametric prediction heads and improves long-term forecasting accuracy. Extensive experiments on nine benchmark datasets demonstrate that DLoG achieves consistently strong performance against competitive forecasting baselines, validating the effectiveness of structure-aware retrieval for time-series forecasting.

Our contributions are summarized as follows:

- We observe that monolithic retrieval may face a potential limitation in time series forecasting: matching entangled signals in a unified space can sometimes retrieve historical patterns that appear similar but differ in their underlying temporal structures.
- We propose DLoG, a decomposed retrieval framework that performs component-specific matching and future reference aggregation for trend and seasonal components.
- We design a local-to-global retrieval mechanism that combines short-range consistency with long-range historical evidence for robust forecasting.

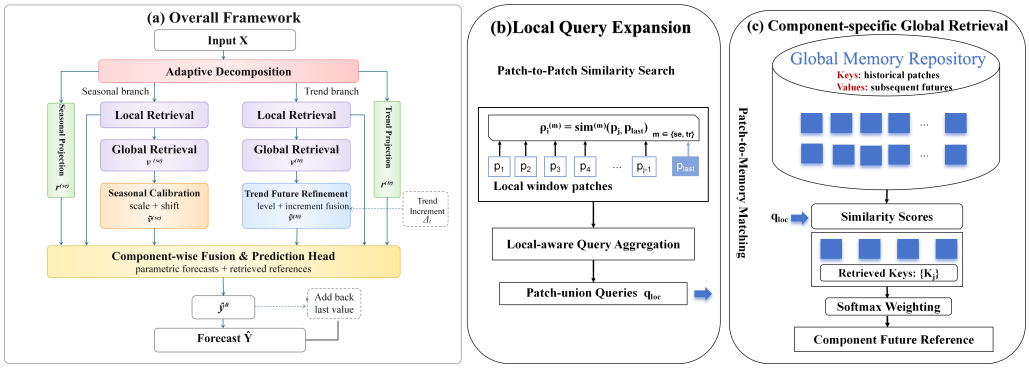


Figure 2: Overview of the proposed framework. (a) The input series is first adaptively decomposed into seasonal and trend components, which are processed by parallel projection and retrieval branches. The retrieved seasonal reference is calibrated by scale and shift, while the retrieved trend reference is refined with trend increment information. The two parametric forecasts, two local retrieval forecasts, and two component-specific global references are then fused to generate the final prediction. (b) Local retrieval selects informative patches around the anchor patch, aggregates them into local component forecasts, and constructs patch-union queries. (c) Component-specific global retrieval matches these queries against a memory repository and aggregates the retrieved future references with softmax weights.

2 Related Work

2.1 Time Series Forecasting

Time series forecasting aims to predict future values from historical observations. Early methods mainly rely on statistical models, while recent approaches are dominated by deep learning. Recurrent Neural Networks (RNNs) and their variants (Hewamalage et al., 2021a; Hochreiter & Schmidhuber, 1997; Wen et al., 2017) model sequential dependencies, but often struggle with long-range temporal patterns. Convolutional Neural Networks (CNNs) (Bai et al., 2018; Borovykh et al., 2017) capture local temporal correlations through causal or temporal convolutions, yet still face challenges in modeling long-term dependencies. Transformer-based models (Zhou et al., 2021; Li et al., 2019; Wu et al., 2021a; Zhou et al., 2022) improve long-range modeling and cross-variable interaction, but usually introduce higher computational cost.

Recently, lightweight MLP-based models have shown strong performance in long-term forecasting (Zeng et al., 2023a; Das et al., 2023; Chen et al., 2023). Among them, DLinear (Zeng et al., 2023a) adopts a simple decomposition strategy to separate the input series into trend and seasonal components, and then applies linear projections to each component for forecasting. Similar decomposition ideas have also been explored in Transformer-based models such as Autoformer (Wu et al., 2021a), where decomposition is mainly used inside the forecasting backbone to facilitate representation learning and prediction. Another related direction explores how to effectively use longer historical contexts. IRPA (Tong & Yuan, 2025) refines ultra-long look-back windows through input refinement and prediction auxiliary modules, extracting useful long-term information while avoiding directly feeding the entire long sequence into the forecasting model. Different from these forecasting-side decomposition or long-window refinement methods, our work uses decomposition to guide the retrieval process itself. Specifically, DLoG decomposes temporal patterns into trend and seasonal components before retrieval, applies component-specific matching rules, and reuses the corresponding historical future references. This makes decomposition a mechanism for structure-aware retrieval rather than only a forecasting-side representation or prediction module.

2.2 Retrieval-Augmented Models

Retrieval-augmented generation (RAG) has been widely studied in natural language processing (Lewis et al., 2020; Guu et al., 2020). It retrieves relevant instances from an external corpus and conditions generation on the retrieved evidence, improving factuality and reducing hallucina-

tion (Shuster et al., 2021; Borgeaud et al., 2022). Beyond NLP, retrieval-based ideas have also been explored in structured data through nearest-neighbor methods (Zhang, 2016), kernel-based retrieval (Nader et al., 2022), and prototype-based models (Arik & Pfister, 2020).

For time-series forecasting, early works retrieve information from related entities or historical instances to support prediction (Iwata & Kumagai, 2020; Yang et al., 2022). More recent methods explicitly study retrieval-augmented forecasting. RAFT (Han et al., 2025) retrieves historical windows similar to the current input and reuses their future values as auxiliary forecasting evidence. RAF (Tire et al., 2024) investigates retrieval augmentation for time-series foundation models, especially under zero-shot forecasting settings. TS-RAG (Ning et al., 2025) further retrieves semantically relevant time-series segments with pretrained encoders and fuses the retrieved information into foundation-model representations.

Unlike prior methods that retrieve holistic representations or target foundation-model adaptation, DLoG performs structure-aware retrieval for supervised long-term forecasting. It decomposes each look-back window into trend and seasonal components, applies component-specific matching, and aggregates their future references through local-to-global retrieval, enabling more targeted reuse of heterogeneous temporal patterns.

3 METHOD

In this section, we first formulate the forecasting problem and then present our adaptive decomposed retrieval framework. The key idea is that temporal patterns with different structural semantics should not only be decomposed into different spaces, but also retrieved and reused with component-specific matching and value designs. An overview of the proposed framework is shown in Figure 2.

3.1 PROBLEM FORMULATION

Given a multivariate time series $\mathbf{S} \in \mathbb{R}^{C\times T}$ with C variables and length T, time-series forecasting aims to predict the future window $\mathbf{y} \in \mathbb{R}^{C\times F}$ from a look-back window $\mathbf{x} \in \mathbb{R}^{C\times L}$. The ground-truth future is denoted as $\mathbf{y}_0 \in \mathbb{R}^{C\times F}$.

Following the implementation, we first apply last-value normalization to remove the local level shift:

$$\mathbf{x}^0 = \mathbf{x} - \mathbf{x}_{:,L}, \tag{1}$$

where $\mathbf{x}_{:,L} \in \mathbb{R}^{C\times 1}$ is broadcast along the temporal dimension. The final prediction is shifted back by the same offset.

Our framework decomposes $\mathbf{x}^0$ into adaptive seasonal and trend components, and then performs retrieval in a component-aware manner. Local retrieval first selects patches that are locally similar to the latest patch, where similarity is measured by Pearson correlation in each component space. The selected patches are aggregated into local component forecasts and also serve as expanded queries for matching against the global retrieval memory. In the global retrieval stage, seasonal and trend components use different matching rules: seasonal retrieval uses shift-tolerant correlation to handle phase variation, while trend retrieval combines level similarity and first-order difference similarity to preserve both trajectory and slope.

3.2 ADAPTIVE TIME-SERIES DECOMPOSITION

Given the normalized input $\mathbf{x}^0 \in \mathbb{R}^{C\times L}$, we decompose it into seasonal and trend components:

$$\mathbf{x}^0 = \mathbf{x}^{(\mathrm{se})} + \mathbf{x}^{(\mathrm{tr})}. \tag{2}$$

Instead of using a fixed moving-average kernel, we introduce a learnable multi-scale smoothing operator. Let $\mathcal{K} = \{\kappa_1, \ldots, \kappa_M\}$ denote a set of candidate kernel sizes. For each scale κ_m, a channel-wise learnable smoothing kernel is applied:

$$\mathbf{u}^{(m)}_{c,:} = \mathrm{Conv1D}\left(\mathrm{Pad}(\mathbf{x}^0_{c,:}); \mathrm{softmax}(\boldsymbol{\theta}^{(m)}_c)\right), \tag{3}$$

where $\boldsymbol{\theta}^{(m)}_c \in \mathbb{R}^{\kappa_m}$ is initialized as a moving average kernel. The softmax normalization makes the smoothing kernel convex and stable.

To select temporal scales adaptively, we compute channel-wise statistics and use a lightweight gating network:

$$\begin{aligned} \mathbf{r}_c &= \left[\mathrm{Mean}(\mathbf{x}^0_{c,:}), \mathrm{Std}(\mathbf{x}^0_{c,:})\right], \\ \alpha_{c,m} &= \mathrm{Softmax}_m\left(g(\mathbf{r}_c) + \mathbf{b}\right). \end{aligned} \tag{4}$$

The trend is obtained by a dynamic mixture of smoothed candidates, and the seasonal component is the residual:

$$\mathbf{x}^{(\mathrm{tr})}_{c,:} = \sum_{m=1}^{M} \alpha_{c,m} \mathbf{u}^{(m)}_{c,:}, \qquad \mathbf{x}^{(\mathrm{se})} = \mathbf{x}^0 - \mathbf{x}^{(\mathrm{tr})}. \tag{5}$$

3.3 Local-to-Global Component Retrieval

Retrieval database. For each training sample, we construct a key–value pair consisting of a historical context window $\mathbf{k}_i \in \mathbb{R}^{C\times L}$ and its subsequent future window $\mathbf{v}_i \in \mathbb{R}^{C\times H}$. To maintain consistency with the component-specific retrieval spaces, both the key and value are processed by the same adaptive decomposition module:

$$\mathbf{k}_i \mapsto \left\{\mathbf{k}_i^{(\mathrm{se})}, \mathbf{k}_i^{(\mathrm{tr})}\right\}, \qquad \mathbf{v}_i \mapsto \left\{\mathbf{v}_i^{(\mathrm{se})}, \mathbf{v}_i^{(\mathrm{tr})}\right\}.$$

The resulting repository therefore contains aligned seasonal and trend key–value pairs. Seasonal queries are matched only against seasonal keys and retrieve the corresponding seasonal futures, whereas trend queries are matched against trend keys and retrieve the corresponding trend futures.

Memory-bank update. Because the adaptive decomposition module is jointly optimized with the forecasting model, the component representations stored in the repository are refreshed throughout training. Specifically, at the beginning of each epoch e, we recompute all decomposed keys and values using the current decomposition parameters $\theta_{\mathrm{d}}^{(e)}$:

$$\mathcal{M}^{(e)} = \left\{\mathrm{sg}\left(\mathbf{k}_i^{(\mathrm{se})}, \mathbf{k}_i^{(\mathrm{tr})}, \mathbf{v}_i^{(\mathrm{se})}, \mathbf{v}_i^{(\mathrm{tr})}\right)\right\}_{i=1}^{N},$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation. The refreshed repository remains fixed during all mini-batch updates within the current epoch and is rebuilt before the next epoch using the updated decomposition module. This epoch-wise refresh avoids maintaining stale memory representations across the entire training process while eliminating the prohibitive cost of rebuilding the full repository after every optimization step.

Importantly, only the component-wise memory representations are cached. For every mini-batch, the query decomposition, local patch selection, component-specific similarity computation, top-K neighbor selection, retrieval weighting, and future-reference aggregation are recomputed online. Gradients are therefore propagated through the current query and forecasting branches, but not through the stored memory entries.

Leakage-free retrieval. During training, we exclude memory samples whose temporal intervals overlap with that of the current query. Let the current sample cover $[t, t+L+F-1]$, including both its look-back and prediction windows, and let a memory sample cover $[t_i, t_i+L+F-1]$. The memory sample is eligible for retrieval only if

$$t_i + L + F - 1 < t \qquad \text{or} \qquad t_i > t + L + F - 1.$$

The overlap mask is applied before the top-K operation, ensuring that excluded samples affect neither the retrieved neighbor set nor the subsequent softmax normalization. During validation and testing, the repository is reconstructed once using the final trained decomposition module and contains only samples from the training split.

Local patch retrieval and expansion. Before global retrieval, we perform local retrieval inside each decomposed component to obtain a complementary forecast and construct a more robust query set. Each component is divided into patches of length P with stride r, and the last patch $\mathbf{p}_{\mathrm{last}}^{(m)}$

is used as the anchor. For each historical patch $\mathbf{p}_j^{(m)}$ in component $m \in \{\mathrm{se}, \mathrm{tr}\}$, we compute channel-averaged Pearson correlation:

$$\rho_j^{(m)} = \frac{1}{C}\sum_{c=1}^{C} \frac{\left\langle \mathbf{p}_{j,c}^{(m)} - \bar{\mathbf{p}}_{j,c}^{(m)}, \mathbf{p}_{\mathrm{last},c}^{(m)} - \bar{\mathbf{p}}_{\mathrm{last},c}^{(m)} \right\rangle}{\|\mathbf{p}_{j,c}^{(m)} - \bar{\mathbf{p}}_{j,c}^{(m)}\|_2 \|\mathbf{p}_{\mathrm{last},c}^{(m)} - \bar{\mathbf{p}}_{\mathrm{last},c}^{(m)}\|_2}. \tag{6}$$

The top-R historical patches with the largest correlation are selected. Their temperature-scaled softmax weights $\beta_j^{(m)}$ are used to form a local component reference $\bar{\mathbf{p}}_{\mathrm{loc}}^{(m)} = \sum_j \beta_j^{(m)} \mathbf{p}_j^{(m)}$. Together with the anchor patch, the selected patches form the expanded query set $\mathcal{P}^{(m)}(\mathbf{x})$ for global retrieval. Hence, the same local search supplies an explicit component reference and improves the robustness of the subsequent global query.

Component-specific global retrieval. For each component-wise expanded patch query, we retrieve neighbors from the corresponding global memory separately.

Seasonal retrieval uses shift-tolerant normalized correlation:

$$s_{\mathrm{se}}(\mathbf{q}, \mathbf{k}) = \max_{\delta \in [-\Delta, \Delta]} \mathrm{Corr}\left(\mathrm{Shift}_\delta(\mathbf{q}), \mathbf{k}\right), \tag{7}$$

which allows small phase mismatch in periodic patterns.

Trend retrieval combines level similarity and first-order difference similarity:

$$s_{\mathrm{tr}}(\mathbf{q}, \mathbf{k}) = (1-\lambda)\mathrm{Corr}\left(\mathbf{q}, \mathbf{k}\right) + \lambda \mathrm{Corr}\left(\Delta_t \mathbf{q}, \Delta_t \mathbf{k}\right). \tag{8}$$

This makes trend matching sensitive to both the global trajectory and the local growth or decline direction.

For each component, every expanded query first retrieves its own top-K memory candidates. The candidate lists from all expanded queries are then merged by sample index. If the same memory sample is retrieved by multiple expanded queries, it is kept only once, and its accumulated score is the local-weighted sum of its matching scores from the queries that retrieved it. A temperature-scaled softmax is then applied over the merged candidate set to obtain the final retrieval weights. Thus, repeated retrieval by several locally similar patches increases a candidate's weight, while each memory sample contributes at most once to the retrieved future. The component-specific retrieved future is computed as

$$\bar{\mathbf{v}}^{(m)} = \sum_{i \in \mathcal{J}^{(m)}} w_i^{(m)} \mathbf{v}_i^{(m)}, \qquad m \in \{\mathrm{se}, \mathrm{tr}\}, \tag{9}$$

where $\mathcal{J}^{(m)}$ is the merged candidate set and $w_i^{(m)}$ is the softmax-normalized retrieval weight for component m.

3.4 Forecasting with Component-Specific Future References

After retrieval, we obtain seasonal and trend future references $\bar{\mathbf{v}}^{(\mathrm{se})}$ and $\bar{\mathbf{v}}^{(\mathrm{tr})}$. In parallel, the latest decomposed patches are projected to produce parametric component forecasts:

$$\mathbf{r}^{(\mathrm{se})} = p_{\mathrm{se}}\left(\mathbf{x}_{:,-P+1:L}^{(\mathrm{se})}\right), \qquad \mathbf{r}^{(\mathrm{tr})} = p_{\mathrm{tr}}\left(\mathbf{x}_{:,-P+1:L}^{(\mathrm{tr})}\right).$$

The projection heads p_{se} and p_{tr} are implemented as two independent lightweight linear heads that flatten the latest component patch along the temporal dimension and map it to the prediction horizon. They do not share parameters. This design keeps the parametric branch simple, so that the retrieved references remain an explicit source of forecasting information.

The locally retrieved component patches are mapped by the corresponding projection heads to obtain two additional forecasts:

$$\mathbf{l}^{(m)} = p_m\left(\bar{\mathbf{p}}_{\mathrm{loc}}^{(m)}\right), \qquad m \in \{\mathrm{se}, \mathrm{tr}\}. \tag{10}$$

For trend futures, the retrieved trend level and its temporal increment are used jointly:

$$\tilde{\mathbf{v}}^{(\mathrm{tr})} = f_{\mathrm{tr}}\left(\mathrm{Concat}\left[\mathbf{r}^{(\mathrm{tr})}, \bar{\mathbf{v}}^{(\mathrm{tr})}, d_{\mathrm{tr}}(\Delta_t \bar{\mathbf{v}}^{(\mathrm{tr})})\right]\right). \tag{11}$$

Here, d_{tr} maps the first-order temporal difference of the retrieved trend reference back to the prediction length, and f_{tr} is a channel-wise feed-forward projection applied to the concatenated trend features.

For seasonal futures, the retrieved seasonal pattern is calibrated by input-dependent amplitude and bias:

$$\tilde{\mathbf{v}}^{(\mathrm{se})} = \mathbf{a}^{(\mathrm{se})} \odot \bar{\mathbf{v}}^{(\mathrm{se})} + \mathbf{b}^{(\mathrm{se})}, \tag{12}$$

where $\mathbf{a}^{(\mathrm{se})}$ and $\mathbf{b}^{(\mathrm{se})}$ are estimated from the seasonal context. This calibration allows the retrieved periodic pattern to be reused under different amplitudes and local offsets.

Finally, the parametric forecasts, local retrieval forecasts, and refined global references are fused by a linear prediction head:

$$\hat{\mathbf{y}}^0 = h\left(\mathrm{Concat}\left[\mathbf{r}^{(\mathrm{se})}, \mathbf{l}^{(\mathrm{se})}, \tilde{\mathbf{v}}^{(\mathrm{se})}, \mathbf{r}^{(\mathrm{tr})}, \mathbf{l}^{(\mathrm{tr})}, \tilde{\mathbf{v}}^{(\mathrm{tr})}\right]\right). \tag{13}$$

The fusion head h maps the concatenated temporal features of length $6F$ to the prediction horizon F for each variable. All projection, calibration, refinement, and fusion heads are trained jointly with the decomposition module. No additional attention or recurrent blocks are used in these heads, which keeps the parametric forecasting capacity controlled and makes the contribution of retrieval easier to isolate.

The prediction is restored to the original scale by adding back the last observed value:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^0 + \mathbf{x}_{:,L}. \tag{14}$$

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate DLoG on nine widely used multivariate time-series forecasting benchmarks: ETTh1, ETTh2, ETTm1, ETTm2 (Zhou et al., 2021), Exchange (Lai et al., 2018), Solar-Energy (Liu et al., 2022), ECL (Trindade, 2015), Traffic, and Weather. Following prior work (Wu et al., 2021b), we use consistent data splits and normalization strategies across all benchmarks. Detailed dataset descriptions are provided in Section A.1 of the Appendix.

Baselines. We compare DLoG with seven representative forecasting methods: RAFT (Han et al., 2025), IRPA (Tong & Yuan, 2025), PatchTST (Nie et al., 2023), TimesNet (Wu et al., 2023), DLinear (Zeng et al., 2023b), SCINet (Liu et al., 2022), and Autoformer (Wu et al., 2021b). These methods cover retrieval-augmented, long-lookback refinement, patch-based, periodic, linear, multi-scale, and decomposition-based forecasting paradigms. Further details of the baseline methods are provided in Section A.2 of the Appendix.

Implementation Details. All experiments are implemented in PyTorch and conducted on NVIDIA H200 (140GB) and RTX 5090 (32GB) GPUs. Models are trained with Adam (Adam et al., 2014), an initial learning rate of 0.01, batch size 32, and 10 epochs. We fix the look-back length to $L = 96$ for all methods and evaluate prediction horizons $H \in \{96, 192, 336, 720\}$.

We follow the standard benchmark protocol and official baseline implementations released by Wang et al. (2025). All experiments are conducted under the multivariate setting and repeated with three random seeds, with results reported as mean$\pm$standard deviation. MSE and MAE are used as evaluation metrics. Additional experimental settings, hyperparameters, and standard deviations are provided in Section B of the Appendix.

4.2 Main Results

Overall performance. Table 1 reports the average results over four forecasting horizons. DLoG achieves the best MSE and MAE on ETTh1, ETTm1, ETTm2, Exchange, Solar-Energy, and ECL, as well as the best MAE on Traffic. It ranks second on both metrics for ETTh2 and Weather and obtains the second-best MSE on Traffic. These results demonstrate that DLoG delivers consistently strong performance across datasets with diverse temporal characteristics. Moreover, DLoG outperforms the retrieval-based RAFT baseline on every dataset, suggesting that the gains are attributable to the proposed structure-aware retrieval design rather than retrieval alone.

Table 1: Average multivariate long-term forecasting results over prediction lengths $H \in \{96, 192, 336, 720\}$ with a fixed lookback window $L = 96$. We report the averaged MSE and MAE across all horizons. Best results are in bold and second-best results are underlined. The complete results for each prediction horizon are provided in Section E of the Appendix.

Dataset	DLoG		RAFT		IRPA		PatchTST		TimesNet		DLinear		SCINet		Autoformer	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.398**	**0.417**	0.432	0.438	0.407	0.423	0.469	0.455	0.458	0.450	0.456	0.452	0.747	0.647	0.496	0.487
ETTh2	0.341	0.390	0.384	0.413	**0.340**	**0.386**	0.387	0.407	0.414	0.427	0.559	0.515	0.954	0.723	0.450	0.459
ETTm1	**0.341**	**0.376**	0.386	0.400	0.349	0.383	0.387	0.400	0.400	0.406	0.403	0.407	0.486	0.481	0.588	0.517
ETTm2	**0.241**	**0.310**	0.282	0.331	0.250	0.315	0.281	0.326	0.291	0.333	0.350	0.401	0.571	0.537	0.327	0.371
Exchange	**0.361**	**0.400**	0.396	0.415	0.397	0.416	0.367	0.404	0.401	0.430	0.367	0.421	0.750	0.626	0.613	0.539
Traffic	0.454	**0.294**	0.557	0.356	**0.443**	0.303	0.555	0.362	0.620	0.336	0.625	0.383	0.804	0.509	0.628	0.379
Solar-Energy	**0.203**	**0.273**	0.235	0.303	0.221	0.292	0.270	0.307	0.301	0.319	0.330	0.401	0.282	0.375	0.885	0.711
Weather	0.233	0.279	0.271	0.310	**0.220**	**0.271**	0.259	0.281	0.259	0.287	0.265	0.317	0.292	0.363	0.338	0.382
ECL	**0.166**	**0.258**	0.188	0.276	0.169	0.262	0.216	0.304	0.193	0.295	0.212	0.300	0.268	0.365	0.227	0.338

Comparison with retrieval and long-context baselines. RAFT retrieves historical windows in a unified representation space, whereas DLoG decomposes both queries and memories into trend and seasonal components and performs component-specific retrieval. This separation allows slowly varying dynamics and recurring local patterns to be matched under different temporal semantics, reducing interference between heterogeneous structures. Compared with IRPA, which benefits from ultra-long input contexts, DLoG performs better on most datasets, while IRPA remains competitive on ETTh2, Weather, and Traffic. The results indicate that structure-aware retrieval and long-context modeling capture complementary forms of historical dependency.

Analysis across temporal patterns. DLoG performs particularly well when historical trends and periodic patterns can be effectively reused. Its strong results on the ETT, Solar-Energy, and ECL datasets show that decomposition-aware retrieval can jointly exploit long-term evolution and local recurrence. The best performance on Exchange further suggests that useful historical analogies can still be identified in relatively irregular series. Overall, by combining component-specific retrieval with local-to-global matching, DLoG retrieves more structurally consistent future references and maintains robust performance across datasets and forecasting horizons.

4.3 Ablation Study

We conduct ablation studies to evaluate the key designs of DLoG, including decomposition-aware retrieval, the local-to-global retrieval mechanism, and adaptive decomposition. For each variant, we modify only the corresponding component while keeping the remaining settings unchanged, and evaluate the forecasting performance with prediction length $H = 96$.

Effectiveness of retrieval components. Table 2 shows that replacing component-specific retrieval with a matched holistic-retrieval variant consistently degrades performance on all five datasets. In particular, MSE increases from 0.353 to 0.361 on ETTh1, from 0.134 to 0.155 on ECL, and from 0.425 to 0.495 on Traffic. Since the holistic variant preserves the same retrieval architecture and training configuration while changing only the retrieval representation and candidate matching strategy, this comparison more directly supports the benefit of separately retrieving trend and seasonal patterns. Removing either Local or Global Retrieval causes further degradation, while retaining only the trend or seasonal pathway also performs worse, confirming the complementary roles of local-to-global retrieval and the two temporal components.

Effectiveness of adaptive decomposition. Replacing adaptive multi-scale decomposition with a fixed strategy degrades performance on all five datasets, with larger drops on ECL, Traffic, and Weather. This suggests that dynamically selecting decomposition scales produces more suitable trend and seasonal representations for component-wise retrieval. Together, these ablations show that the gains of DLoG arise from the combination of decomposition-aware retrieval, local-to-global matching, and adaptive temporal decomposition.

Table 2: Ablation study on the main components of DLoG with prediction length $H = 96$. Results are reported in MSE; lower values are better.

Method	ETTh1	Exchange	ECL	Traffic	Weather
DLoG	**0.353**	**0.082**	**0.134**	**0.425**	**0.153**
DLoG w/ Holistic Retrieval	0.361	0.084	0.155	0.495	0.168
w/o Local Retrieval	0.365	0.086	0.165	0.540	0.174
w/o Global Retrieval	0.381	0.091	0.173	0.547	0.180
Only Trend Retrieval	0.375	0.084	0.171	0.531	0.182
Only Seasonal Retrieval	0.362	0.083	0.182	0.537	0.183
w/o Adaptive Decomposition	0.363	0.084	0.161	0.526	0.171

4.4 Model Analysis

Local Retrieval vs. Longer Look-back Window. We examine whether DLoG benefits merely from accessing a longer historical context. As shown in Table 3, DLoG reduces MSE from 0.325 to 0.309 while using 29.2% fewer trainable parameters. Despite moderate retrieval overhead, this 4.9% error reduction indicates that selectively retrieving relevant historical patterns is more effective than directly encoding a substantially longer sequence. The gain therefore stems primarily from informative pattern selection rather than increased historical context alone.

Table 3: Local Retrieval versus a longer look-back window on Weather at $H = 720$.

Method	Look-back L	Local Window W_{local}	Parameters (M)	Time / epoch (s)	MSE ↓
DLoG without Local	3600	–	4.56	**15**	0.325
DLoG	96	3600	**3.23**	21	**0.309**

Comparison with Retrieval-Augmented Forecasting Methods. As shown in Table 4, DLoG achieves lower MSE than TS-RAG on four of five datasets, with relative reductions of 1.4%–4.8%. The largest improvement is observed on ETTm2, suggesting that separately retrieving trend and seasonal patterns provides more informative historical references than matching complete temporal segments. Although DLoG underperforms on Exchange, the overall results support the effectiveness of component-aware retrieval across diverse ETT benchmarks. Additional comparisons with RAF are provided in Section C.1 of the Appendix.

Table 4: Comparison with TS-RAG in terms of MSE (↓).

Method	ETTh1	ETTh2	ETTm1	ETTm2	Exchange
TS-RAG	0.355	0.245	0.290	0.146	**0.062**
DLoG	**0.350**	**0.238**	**0.285**	**0.139**	0.072

5 Conclusion

In this paper, we propose DLoG, a decomposed local-to-global retrieval framework for long-term time-series forecasting. Unlike conventional retrieval-augmented methods that match historical patterns in a monolithic space, DLoG separately retrieves trend and seasonal components using structure-specific criteria, reducing interference between heterogeneous temporal patterns. Its local-to-global strategy combines recent patch-level dynamics with long-range historical references for more reliable reuse of informative futures. Experiments on nine benchmark datasets demonstrate strong performance against both forecasting and retrieval-enhanced baselines.

AI USE STATEMENT

In preparing this submission, generative AI tools were used to assist with LaTeX template conversion and formatting. The authors reviewed the AI-assisted changes and verified that the scientific claims, equations, experimental results, tables, figures, and citations were preserved from the author-prepared manuscript. The authors take responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 provides the complete model formulation and retrieval procedure, while Section 4.1 specifies the evaluation protocol. Appendix A describes the benchmark datasets and baselines, Appendix B reports implementation details and hyperparameters, Appendix C provides additional component analyses, Appendix D reports computational-cost analyses, and Appendix E provides complete per-horizon forecasting results with standard deviations.

REFERENCES

Kingma DP Ba J Adam et al. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1412(6), 2014.

Sercan O Arik and Tomas Pfister. Protoattend: Attention-based prototypical learning. *Journal of Machine Learning Research*, 21(210):1–35, 2020.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O Arik, and Jinsung Yoon. Retrieval augmented time series forecasting, 2025. URL https://arxiv.org/abs/2505.04163.

H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021a.

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37 (1):388–427, 2021b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Tomoharu Iwata and Atsutoshi Kumagai. Few-shot learning for time-series forecasting. *arXiv preprint arXiv:2009.14379*, 2020.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. Sparsetsf: Modeling long-term time series forecasting with 1k parameters. *arXiv preprint arXiv:2405.00946*, 2024.

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

Ganapathy Mahalakshmi, S Sridevi, and Shyamsundar Rajaram. A survey on forecasting of time series data. In *2016 international conference on computing technologies and intelligent data engineering (ICCTIDE'16)*, pp. 1–8. IEEE, 2016.

Youssef Nader, Leon Sixt, and Tim Landgraf. Dnnr: Differential nearest neighbors regression. In *International Conference on Machine Learning*, pp. 16296–16317. PMLR, 2022.

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*, 2023.

Kanghui Ning, Zijie Pan, Yu Liu, Yushan Jiang, James Yiming Zhang, Kashif Rasul, Anderson Schneider, Lintao Ma, Yuriy Nevmyvaka, and Dongjin Song. Ts-rag: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster. *arXiv preprint arXiv:2503.07649*, 2025.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. *arXiv preprint arXiv:2411.08249*, 2024.

Suxin Tong and Jingling Yuan. Efficiently enhancing long-term series forecasting via ultra-long lookback windows. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20912–20920, 2025.

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark, 2025. URL `https://arxiv.org/abs/2407.13278`.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021a.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021b.

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*, 2023.

Sitan Yang, Carson Eisenach, and Dhruv Madeka. Mq-retcnn: Multi-horizon time series forecasting with retrieval-augmentation. In *8th SIGKDD International Workshop on Mining and Learning from Time Series – Deep Forecasting: Models, Interpretability, and Applications*, 2022.

A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023a. doi: 10.1609/aaai.v37i9.26317. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023b.

Zhongheng Zhang. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11), 2016.

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

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

A BENCHMARK AND BASELINE

A.1 BENCHMARK DATASETS

We provide detailed information about the benchmark datasets used in our experiments. Following common long-term time-series forecasting protocols, we evaluate DLoG on nine widely used multivariate datasets covering different domains, sampling frequencies, sequence lengths, and numbers of variables. These datasets include ETTh1, ETTh2, ETTm1, ETTm2 (Zhou et al., 2021), Exchange (Lai et al., 2018), Solar-Energy (Liu et al., 2022), ECL (Trindade, 2015), Traffic, and Weather.

The four ETT datasets record electricity transformer temperature and load-related factors. ETTh1 and ETTh2 are sampled hourly, whereas ETTm1 and ETTm2 are sampled every 15 minutes. The Exchange dataset contains daily exchange rates from multiple countries and is commonly used to evaluate forecasting under irregular and non-stationary temporal dynamics. Solar-Energy records solar power production from photovoltaic plants at 10-minute intervals, while ECL contains hourly electricity consumption measurements from multiple clients. Traffic records hourly road occupancy rates collected from freeway sensors, and Weather contains multiple meteorological indicators sampled every 10 minutes.

Following prior long-term forecasting studies such as Autoformer (Wu et al., 2021b), we use the standard train/validation/test partitions and apply consistent normalization strategies across all compared methods. The dataset sizes are reported as train/validation/test splits, and the corresponding statistics are summarized in Table 5. All datasets are evaluated under the multivariate forecasting setting, where both the input sequence and the prediction target contain multiple variables.

Table 5: Statistics of the benchmark datasets. Split sizes are reported as (train, validation, test).

Dataset	Vars.	Split Sizes	Freq.
ETTh1	7	(8545, 2881, 2881)	1 hour
ETTh2	7	(8545, 2881, 2881)	1 hour
ETTm1	7	(34465, 11521, 11521)	15 min
ETTm2	7	(34465, 11521, 11521)	15 min
Exchange	8	(5120, 665, 1422)	1 day
Solar-Energy	137	(36601, 5161, 10417)	10 min
ECL	321	(18317, 2633, 5261)	1 hour
Traffic	862	(12185, 1757, 3509)	1 hour
Weather	21	(36792, 5271, 10540)	10 min

A.2 BASELINES

We compare DLoG with seven representative and competitive long-term forecasting baselines covering retrieval-augmented, long-context, Transformer-based, convolution-based, linear, and decomposition-based forecasting paradigms.

RAFT (Han et al., 2025) is a retrieval-augmented forecasting framework that incorporates external historical patterns to improve predictions. It serves as an important comparison for evaluating whether the gains of DLoG arise solely from access to historical reference patterns.

IRPA (Tong & Yuan, 2025) is a recent long-context forecasting method designed to exploit ultra-long lookback windows. It introduces an Input Refinement Module and a Prediction Auxiliary Module to enhance information extracted from limited lookback inputs while incorporating auxiliary historical guidance.

PatchTST (Nie et al., 2023) is a Transformer-style forecasting model that divides time series into temporal patches and models dependencies at the patch level, enabling effective long-range representation learning.

TimesNet (Wu et al., 2023) transforms one-dimensional temporal variations into multi-period representations and captures complex temporal patterns through periodic modeling.

DLinear (Zeng et al., 2023b) is a lightweight forecasting model based primarily on linear projections and trend-seasonal decomposition, providing a strong and computationally efficient baseline.

SCINet (Liu et al., 2022) employs a recursive splitting and interaction mechanism to capture temporal dependencies at multiple scales.

Autoformer (Wu et al., 2021b) is a Transformer-based forecasting architecture that combines progressive series decomposition with an auto-correlation mechanism for long-term temporal dependency modeling.

For fair comparison, all baselines are evaluated using the same dataset partitions, input lengths, prediction horizons, and evaluation metrics. Where applicable, we follow the standard benchmark configurations and official implementations released by Wang et al. (2025) to minimize implementation-dependent differences.

B Implementation Details

All experiments are implemented in PyTorch and conducted on NVIDIA H200 GPUs with 140GB memory and NVIDIA RTX 5090 GPUs with 32GB memory. DLoG is optimized using Adam (Adam et al., 2014) with an initial learning rate of 0.01, a batch size of 32, and a training budget of 10 epochs.

Following the standard long-term forecasting protocol, the input look-back length is fixed to $L = 96$ for DLoG and all compared baselines, while the prediction horizons are set to

$$H \in \{96, 192, 336, 720\}.$$

All experiments are conducted under the multivariate-to-multivariate forecasting setting, where multiple variables are jointly provided as input and their corresponding future sequences are predicted simultaneously. Using identical input lengths and prediction horizons across all methods ensures a controlled comparison under the same amount of historical context.

We use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the primary evaluation metrics. Given ground-truth observations $\mathbf{Y}$ and predictions $\hat{\mathbf{Y}}$, they are computed as

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2,$$

and

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left| Y_i - \hat{Y}_i \right|.$$

Lower values indicate better forecasting performance.

To reduce the influence of stochastic optimization and initialization, each experiment is repeated with three random seeds. We report the mean performance across the three runs, together with the corresponding standard deviations where applicable. The same data preprocessing, dataset partitions, forecasting horizons, and evaluation procedures are adopted for DLoG and all baselines to ensure fair and reproducible comparisons.

Unless otherwise specified, we use the default hyperparameter settings summarized in Table 6.

In our implementation, R corresponds to `top_k`, whereas K_{tr} and K_{se} correspond to `topm_trend` and `topm_season`, respectively. We use H to denote the prediction horizon throughout the experiments.

For adaptive decomposition, `decomp_kernel_sizes` is instantiated as $[12, 25, 50]$ according to `moving_avg`$= 25$, i.e., $[\texttt{moving_avg}//2, \texttt{moving_avg}, \texttt{moving_avg} * 2]$.

Since the benchmark datasets have different sampling frequencies and temporal ranges, we use dataset-specific local candidate windows for Local Retrieval. The corresponding values of W_{local} are summarized in Table 7.

Table 6: Default hyperparameter settings used in DLoG.

Hyperparameter	Value	Description
`moving_avg`	25	Base smoothing window for decomposition
`decomp_kernel_sizes`	$[12, 25, 50]$	Multi-scale smoothing kernels
`decomp_gate_hidden`	16	Hidden dimension of the decomposition gate
`top_k`	3	Number of locally selected patches
`topm_trend`	20	Number of retrieved trend candidates
`topm_season`	20	Number of retrieved seasonal candidates
τ (`temperature`)	0.1	Softmax temperature for retrieval
`trend_diff_weight`	0.5	Weight of trend-difference similarity
`seasonal_shift`	3	Maximum shift for seasonal matching
`learning_rate`	0.01	Learning rate of the Adam optimizer

Table 7: Dataset-specific local candidate window sizes used in Local Retrieval.

Dataset	W_{local}
ETTh1	480
ETTh2	480
ETTm1	1920
ETTm2	1920
Exchange	480
Solar-Energy	1920
ECL	480
Traffic	480
Weather	3600

C Component Analysis

C.1 Model Analysis

Comparison with Retrieval-Augmented Forecasting Methods. We further compare DLoG with RAF under its original evaluation protocol. As shown in Table 8, DLoG yields particularly strong gains on Weather and Traffic across different forecasting horizons. On Weather, the average WQL decreases from 0.167 to 0.063, corresponding to an approximately 62.2% reduction, while MASE is also improved at $H = 15$ and $H = 20$. The improvement is even more pronounced on Traffic, where DLoG reduces the average WQL by 35.8% and the average MASE by 52.7%. These consistent gains across $H \in \{10, 15, 20\}$ indicate that component-aware retrieval remains effective as the forecasting horizon increases.

On ETTh1, the results are more mixed: DLoG achieves lower MASE at all three horizons, but improves WQL only at $H = 15$. This suggests that the advantage of decomposed retrieval may depend on the evaluation criterion and temporal characteristics of the dataset. Overall, the results show that separately retrieving trend and seasonal components can provide more useful historical references than retrieving complete temporal segments, with especially clear benefits on Weather and Traffic.

Table 8: Comparison with RAF under its original protocol. Results are reported as WQL / MASE (↓).

Dataset	H	RAF WQL / MASE	DLoG WQL / MASE
Weather	10	0.166 / **1.265**	**0.056** / 1.284
	15	0.165 / 1.562	**0.063** / **1.554**
	20	0.169 / 1.898	**0.070** / **1.789**
Traffic	10	0.225 / 1.524	**0.155** / **1.049**
	15	0.234 / 2.467	**0.157** / **1.066**
	20	0.292 / 2.911	**0.170** / **1.148**
ETTh1	10	**0.079** / 1.020	0.123 / **1.003**
	15	0.159 / 1.490	**0.145** / **1.381**
	20	**0.143** / 1.426	0.155 / **1.403**

Analysis of Retrieval Similarity Metrics. We further analyze the effect of different similarity metrics in the retrieval module. We compare Pearson correlation with three commonly used distance-based metrics: Euclidean distance, Manhattan distance, and Hellinger distance. All experiments use the same setting with prediction length $H = 96$, and only the retrieval similarity metric is varied.

As shown in Table 9, Pearson correlation achieves the lowest MSE on four of the five datasets. This suggests that correlation-based matching is generally more suitable for DLoG than direct distance-based matching. Pearson correlation emphasizes temporal shape similarity while being less sensitive to scale and level shifts, which is particularly useful when matching local patches and decomposed temporal components. We therefore adopt Pearson correlation as the default similarity metric in DLoG.

Table 9: Comparison of retrieval similarity metrics at prediction length $H = 96$. Results are reported in MSE; lower values are better.

Dataset	Pearson Correlation	Euclidean Distance	Manhattan Distance	Hellinger Distance
ETTh1	**0.353**	0.368	0.368	0.373
ETTm1	**0.294**	0.334	0.335	0.341
Exchange	0.082	**0.077**	0.079	0.084
ECL	**0.134**	0.155	0.158	0.164
Weather	**0.153**	0.169	0.168	0.178

C.2 Hyperparameter Analysis

We further analyze the sensitivity of several key retrieval hyperparameters in DLoG, including the number of selected local patches `top_k`, the number of retrieved global candidates `topm_trend` and `topm_season`, and the temperature parameter τ used in the softmax weighting over retrieval similarities. For compactness, when varying the number of global candidates, we set `topm_trend` and `topm_season` to the same value. All experiments are conducted with prediction length $H = 96$, while only the corresponding hyperparameter is varied.

As shown in Table 10, DLoG remains generally stable over reasonable hyperparameter variations. The default configuration, `top_k`$= 3$, `topm_trend`=`topm_season`$= 20$, and $\tau = 0.1$, provides consistently strong performance across datasets and is therefore used in the main experiments.

D Computational Complexity and Scalability Analysis

We analyze the computational characteristics of DLoG from four aspects: (1) asymptotic retrieval complexity, (2) repository refresh and online inference cost, (3) scaling with the repository size, and (4) the efficiency–accuracy trade-off induced by repository subsampling.

Table 10: Hyperparameter sensitivity at prediction length $H = 96$. For `topm`, `topm_trend` and `topm_season` are set to the same value. Default settings are marked with *, and the best result within each hyperparameter sweep is shown in bold. Results are reported in MSE (↓).

	`top_k`			`topm`			τ		
Dataset	1	3*	5	5	10	20*	0.01	0.1*	1
ETTh1	0.363	**0.353**	0.360	0.365	0.363	**0.353**	0.389	**0.353**	0.363
ETTh2	0.286	**0.265**	0.271	0.285	0.284	**0.265**	0.318	**0.265**	0.285
ETTm1	0.311	**0.294**	0.312	0.327	0.310	**0.294**	0.364	**0.294**	0.311
ETTm2	0.179	**0.160**	0.181	0.209	0.193	**0.160**	0.232	**0.160**	0.182
Exchange	0.095	**0.082**	0.093	0.094	0.096	**0.082**	0.093	**0.082**	0.096
ECL	**0.132**	0.134	0.163	0.222	0.171	**0.134**	0.202	**0.134**	0.161
Weather	0.179	**0.153**	0.184	0.196	0.192	**0.153**	0.308	**0.153**	0.185

An important distinction is that repository construction has different roles during training and inference. During training, the component-wise memory repository is refreshed at the beginning of each epoch using the current decomposition module and remains fixed during all mini-batch updates within that epoch. The stored memory representations are stop-gradient, while query decomposition, similarity computation, top-K selection, and reference aggregation are performed online. During validation and testing, the repository is constructed once from the training split using the final trained decomposition module.

D.1 Asymptotic Retrieval Complexity

Let N denote the number of entries in the global retrieval repository, C the number of variables, L the input look-back length, H the prediction horizon, P the patch length, and J the number of candidate patches considered by Local Retrieval. We further denote by R the number of selected local patches, by K the number of globally retrieved candidates, and by Δ the maximum seasonal shift used in shift-tolerant matching.

Repository Refresh. For each memory entry, DLoG decomposes both the historical key of length L and its corresponding future value of length H into seasonal and trend components. Ignoring the constant number of decomposition branches and smoothing kernels, a full repository refresh requires

$$T_{\text{refresh}} = \mathcal{O}(NC(L+H)) .$$

The component-wise keys and future values stored in the repository require

$$M_{\text{repo}} = \mathcal{O}(NC(L+H))$$

memory. Therefore, the current exact repository implementation scales linearly with the number of stored historical samples.

Local Retrieval. Local Retrieval compares the most recent component patch against J historical candidate patches. Computing channel-averaged Pearson correlations requires

$$T_{\text{local}} = \mathcal{O}(JCP),$$

while selecting the top-R local patches introduces an additional selection cost. Using a heap-style top-R bound, the total cost can be written as

$$T_{\text{local}} = \mathcal{O}(JCP + J \log R) .$$

Here, J is determined by the dataset-specific local candidate window, patch length, and patch stride.

Global Retrieval. The anchor patch together with the R selected local patches forms at most $Q = R + 1$ expanded queries for each temporal component. Let D denote the effective temporal dimension used for global matching, where $D \leq L$. Exact comparison against all N repository entries requires approximately

$$\mathcal{O}(QNCD)$$

operations for trend matching.

Seasonal retrieval evaluates correlations over at most $2\Delta + 1$ shifts, leading to the upper bound

$$\mathcal{O}(QNCD(2\Delta + 1)) .$$

Top-K candidate selection can additionally be bounded by

$$\mathcal{O}(QN \log K),$$

and aggregation of the retrieved future references requires

$$\mathcal{O}(QKCH).$$

Combining these terms, the online retrieval complexity can be summarized as

$$T_{\text{retrieval}} = \mathcal{O}(JCP + QNCD(2\Delta + 1) + QN \log K + QKCH) .$$

Since R, K, and Δ are small constants in our default configuration, the dominant cost of exact global retrieval scales approximately linearly with N, C, and the matching dimension. In the worst case where $D = \mathcal{O}(L)$, this term becomes $\mathcal{O}(NCL)$ up to constant factors.

D.2 Training and Inference Cost Breakdown

We next measure the practical wall-clock cost of the different stages of DLoG. All measurements use the same hardware, batch size, input length, and prediction horizon. GPU timing is performed after warm-up iterations with CUDA synchronization before and after each measurement.

Table 11 reports the results on ETTm1 with $L = 96$ and $H = 96$. A complete repository build takes 8.740 s. Because the default training procedure refreshes the repository once per epoch, this value also characterizes the cost of one full epoch-wise memory refresh. The subsequent optimization steps within an epoch take 11.820 s while the repository representations remain fixed. Under a sequential implementation, the resulting total cost is approximately 20.560 s per epoch.

At inference time, online retrieval requires 0.063 s per batch, while the forecasting forward pass given the retrieved references requires 0.019 s. The complete online inference cost is therefore 0.082 s per batch.

Table 11: Wall-clock cost breakdown of DLoG on ETTm1 with $L = 96$ and $H = 96$. The total training time per epoch includes one full repository refresh followed by optimization with fixed memory representations.

Stage	Time (s)
Full repository refresh / epoch	8.740
Optimization / epoch	11.820
Total training / epoch	20.560
Online retrieval / batch	0.063
Forecasting forward / batch	0.019
Total online inference / batch	0.082

In this setting, the full repository refresh accounts for approximately 42.5% of the sequential per-epoch training time. At inference time, retrieval accounts for approximately 76.8% of the measured online latency. These measurements indicate that exact retrieval constitutes a non-negligible computational overhead and motivate the repository-size reduction analysis below.

D.3 Scaling with Repository Size

The complexity analysis above predicts approximately linear dependence on the number of repository entries N. To empirically characterize this behavior, we vary the repository stride $s \in$

$\{1, 2, 4, 8\}$. A stride of s retains one candidate every s sliding windows, thereby reducing the effective repository size without changing the retrieval mechanism.

Table 12 shows that both repository refresh time and online retrieval latency decrease consistently with repository size.

Table 12: Computational scaling under different repository strides on ETTm1 with $L = 96$ and $H = 96$.

Stride s	1	2	4	8
Repository size	34,369	17,185	8,593	4,297
Relative size	$1.00\times$	$0.50\times$	$0.25\times$	$0.125\times$
Refresh time (s)	8.74	4.62	2.48	1.39
Retrieval / batch (s)	0.063	0.036	0.022	0.015

Reducing the repository from 34,369 to 4,297 entries decreases the refresh time from 8.74 s to 1.39 s and the retrieval latency from 0.063 s to 0.015 s. This corresponds to reductions of approximately 84.1% and 76.2%, respectively.

The observed trend is consistent with the linear dependence on N predicted by the exact-search complexity. Repository subsampling therefore provides a simple mechanism for reducing both epoch-wise refresh cost and online retrieval latency.

D.4 Efficiency–Accuracy Trade-off

Reducing the repository size lowers computational cost but may remove useful historical references. We therefore evaluate forecasting accuracy under the same repository strides. Table 13 reports MSE and MAE at $H = 96$.

Table 13: Forecasting performance under different repository strides. Lower values are better.

Dataset	Metric	$s = 1$	$s = 2$	$s = 4$	$s = 8$
ETTh1	MSE	0.353	0.359	0.363	0.371
	MAE	0.385	0.391	0.394	0.401
ETTh2	MSE	0.265	0.283	0.288	0.291
	MAE	0.330	0.339	0.341	0.347
ETTm1	MSE	0.294	0.308	0.312	0.320
	MAE	0.345	0.356	0.360	0.367
ETTm2	MSE	0.160	0.169	0.172	0.177
	MAE	0.252	0.264	0.267	0.272

As expected, increasing the repository stride leads to a gradual degradation in forecasting accuracy. However, small-to-moderate strides substantially reduce repository size and retrieval cost while retaining most of the forecasting performance.

For example, using $s = 2$ reduces the repository size by approximately 50% and the measured retrieval latency from 0.063 s to 0.036 s per batch, while the forecasting degradation remains relatively moderate across the evaluated datasets. This result illustrates a controllable efficiency–accuracy trade-off and provides a practical mechanism for scaling DLoG when the available retrieval repository becomes large.

D.5 Discussion on Large-Scale Memory Updates

The current implementation adopts synchronous epoch-wise repository refresh. Its main advantage is that all stored component representations within an epoch are generated by the same decomposition parameters, avoiding inconsistency between memory entries. However, because a complete

refresh has $\mathcal{O}(NC(L+H))$ complexity, its cost grows linearly with the repository size and may become non-negligible for substantially larger datasets.

Our current implementation does not employ asynchronous or incremental memory updates. Nevertheless, the repository entries are detached from gradient computation and remain fixed during mini-batch optimization, making less frequent, partial, or double-buffered asynchronous refresh compatible with the overall framework in principle. The repository-stride experiments demonstrate that reducing the effective memory size can substantially decrease both refresh and retrieval costs.

Investigating asynchronous or incremental refresh strategies for industrial-scale repositories is therefore an important direction for future work. It should also be noted that the experiments in this work are conducted on standard forecasting benchmarks rather than industrial-scale repositories. Therefore, while the measured scaling trends and complexity analysis characterize the computational behavior of DLoG, they should not be interpreted as direct evidence of industrial-scale deployment performance.

E FULL RESULTS

Table 14: Complete multivariate long-term forecasting results over forecast horizons $H \in \{96, 192, 336, 720\}$ with a fixed look-back window $L = 96$. DLoG results are reported as mean $\pm$ standard deviation over three random seeds. Best mean results are shown in bold, and second-best mean results are underlined.

Dataset	H	**DLoG**		**RAFT**		**IRPA**		**PatchTST**		**TimesNet**		**DLinear**		**SCINet**		**Autoformer**	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.353**±0.004	**0.385**±0.003	0.386	0.413	0.361	0.388	0.414	0.419	0.384	0.402	0.386	0.400	0.654	0.599	0.449	0.459
	192	**0.387**±0.005	**0.407**±0.004	0.422	0.430	0.393	0.410	0.460	0.445	0.436	0.429	0.437	0.432	0.719	0.631	0.500	0.482
	336	**0.413**±0.006	**0.424**±0.004	0.455	0.444	0.418	0.428	0.501	0.466	0.491	0.469	0.481	0.459	0.778	0.659	0.521	0.496
	720	**0.437**±0.007	**0.450**±0.005	0.463	0.463	0.454	0.464	0.500	0.488	0.521	0.500	0.519	0.516	0.836	0.699	0.514	0.512
ETTh2	96	**0.265**±0.004	**0.330**±0.003	0.295	0.349	0.270	0.334	0.302	0.348	0.340	0.374	0.333	0.387	0.707	0.621	0.346	0.388
	192	0.335±0.006	0.381±0.004	0.382	0.402	**0.331**	**0.374**	0.388	0.400	0.402	0.414	0.477	0.476	0.860	0.689	0.456	0.452
	336	**0.364**±0.005	0.404±0.004	0.425	0.442	0.365	**0.402**	0.426	0.433	0.452	0.452	0.594	0.541	1.000	0.744	0.482	0.486
	720	0.398±0.007	0.443±0.005	0.434	0.459	**0.393**	**0.432**	0.431	0.446	0.462	0.468	0.831	0.657	1.249	0.838	0.515	0.511
ETTm1	96	**0.294**±0.003	**0.345**±0.003	0.348	0.369	0.307	0.359	0.329	0.367	0.338	0.375	0.345	0.372	0.418	0.438	0.505	0.475
	192	**0.318**±0.004	**0.363**±0.003	0.362	0.387	0.329	0.369	0.367	0.385	0.374	0.387	0.380	0.389	0.439	0.450	0.553	0.496
	336	**0.351**±0.005	0.390±0.004	0.391	0.407	0.354	**0.384**	0.399	0.410	0.410	0.411	0.413	0.413	0.490	0.485	0.621	0.537
	720	**0.399**±0.006	**0.406**±0.004	0.444	0.438	0.405	0.418	0.454	0.439	0.478	0.450	0.474	0.453	0.595	0.550	0.671	0.561
ETTm2	96	**0.160**±0.002	**0.252**±0.002	0.176	0.264	0.164	0.255	0.175	0.259	0.187	0.267	0.193	0.292	0.286	0.377	0.255	0.339
	192	**0.215**±0.003	**0.292**±0.003	0.241	0.306	0.221	0.296	0.241	0.302	0.249	0.309	0.284	0.362	0.399	0.445	0.281	0.340
	336	**0.257**±0.004	**0.322**±0.003	0.303	0.346	0.264	0.330	0.305	0.343	0.321	0.351	0.369	0.427	0.637	0.591	0.339	0.372
	720	**0.331**±0.006	**0.372**±0.004	0.407	0.408	0.349	0.378	0.402	0.400	0.408	0.403	0.554	0.522	0.960	0.735	0.433	0.432
Exchange	96	**0.082**±0.003	**0.198**±0.003	0.084	0.199	0.089	0.208	0.088	0.205	0.107	0.234	0.088	0.218	0.267	0.396	0.197	0.323
	192	**0.172**±0.004	**0.293**±0.004	0.176	0.295	0.179	0.300	0.176	0.299	0.184	0.307	0.226	0.344	0.351	0.459	0.300	0.369
	336	0.337±0.007	0.417±0.005	0.352	0.422	0.341	0.420	**0.301**	**0.397**	0.349	0.431	0.313	0.427	1.324	0.853	0.509	0.524
	720	0.854±0.009	**0.690**±0.006	0.971	0.742	0.980	0.735	0.901	0.714	0.964	0.746	**0.839**	0.695	1.058	0.797	1.447	0.941
Solar-Energy	96	**0.179**±0.004	**0.267**±0.003	0.201	0.278	0.196	0.278	0.234	0.286	0.250	0.292	0.290	0.378	0.237	0.344	0.884	0.711
	192	**0.177**±0.004	**0.240**±0.003	0.222	0.298	0.216	0.292	0.267	0.310	0.296	0.318	0.320	0.398	0.280	0.380	0.834	0.692
	336	**0.218**±0.005	**0.287**±0.004	0.249	0.313	0.235	0.303	0.290	0.315	0.319	0.330	0.353	0.415	0.304	0.389	0.941	0.723
	720	**0.237**±0.006	0.298±0.004	0.266	0.324	**0.237**	**0.295**	0.289	0.317	0.338	0.337	0.356	0.413	0.308	0.388	0.882	0.717
ECL	96	**0.134**±0.003	**0.231**±0.002	0.159	0.259	0.143	0.240	0.195	0.285	0.168	0.272	0.197	0.282	0.247	0.345	0.201	0.317
	192	**0.148**±0.003	**0.244**±0.003	0.167	0.258	0.156	0.250	0.199	0.289	0.184	0.289	0.196	0.285	0.257	0.355	0.222	0.334
	336	**0.169**±0.004	**0.257**±0.003	0.193	0.273	0.171	0.265	0.215	0.305	0.198	0.300	0.209	0.301	0.269	0.369	0.231	0.338
	720	0.211±0.005	0.299±0.004	0.232	0.315	**0.205**	**0.293**	0.256	0.337	0.220	0.320	0.245	0.333	0.299	0.390	0.254	0.361
Traffic	96	**0.425**±0.004	**0.279**±0.003	0.534	0.348	0.429	0.296	0.544	0.359	0.593	0.321	0.650	0.396	0.788	0.499	0.613	0.388
	192	**0.435**±0.005	**0.281**±0.003	0.526	0.343	0.436	0.296	0.540	0.354	0.617	0.336	0.598	0.370	0.789	0.505	0.616	0.382
	336	**0.443**±0.005	**0.288**±0.003	0.574	0.361	0.447	0.308	0.551	0.358	0.629	0.336	0.605	0.373	0.797	0.508	0.622	0.337
	720	0.513±0.007	0.326±0.005	0.592	0.372	**0.459**	**0.311**	0.586	0.375	0.640	0.350	0.645	0.394	0.841	0.523	0.660	0.408
Weather	96	**0.153**±0.003	**0.210**±0.002	0.188	0.245	0.156	0.214	0.177	0.218	0.172	0.220	0.196	0.255	0.221	0.306	0.266	0.336
	192	0.203±0.004	0.255±0.003	0.238	0.288	**0.198**	**0.253**	0.225	0.259	0.219	0.261	0.237	0.296	0.261	0.340	0.307	0.367
	336	0.267±0.005	0.306±0.004	0.291	0.328	**0.235**	**0.287**	0.278	0.297	0.280	0.306	0.283	0.335	0.309	0.378	0.359	0.395
	720	0.309±0.006	0.343±0.004	0.367	0.379	**0.289**	**0.328**	0.354	0.348	0.365	0.359	0.345	0.381	0.377	0.427	0.419	0.428