BEYOND NUMERICAL SIMILARITY: RESOLUTION-VARIANT VISUAL RETRIEVER FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented time series forecasting relies on identifying historical segments that provide relevant predictive context. However, a single numerical similarity metric is constrained in capturing global and local temporal patterns. Viewing time series as visual trajectories makes trends, fluctuations, and turning points explicit, but exploiting this information requires both a comprehensive representation and a quantifiable similarity measurement for retrieval. In this work, we propose **ReViR**, a visual retrieval method that represents temporal patterns through triangle regions and measures their geometric agreement. Specifically, Visual Sampling selects visual anchors, and the triangle regions induced by these anchors form a region map for each time series. For retrieval similarity, we introduce Approximate Triangle-IoU, which measures region overlap on a shared time-value grid, providing an interpretable visual similarity without a pretrained vision encoder. Historical time series selected by the visual similarity are aggregated and fused with a lightweight forecaster. To control the overhead of visual retrieval, configurable sampling density and resolution balance representation content against cost, while packed binary masks enable efficient overlap computation. Experiments on seven real-world benchmarks demonstrate that ReViR achieves state-of-the-art forecasting performance, while substantially reducing retrieval-library storage in the evaluated settings.

1 INTRODUCTION

Time series forecasting is fundamental to decision-making in applications such as energy (Hong & Fan, 2016; Lago et al., 2021), transportation (Li et al., 2018; Yu et al., 2018), and finance (Fischer & Krauss, 2018; Zhang et al., 2019). Despite substantial progress, real-world temporal dynamics remain highly diverse and non-stationary, with patterns varying across domains, variables, and time. Large-scale pretraining has enabled forecasting models to capture generalizable temporal regularities from diverse datasets. However, global representations will overlook local characteristics that are crucial for accurately modeling future time-series evolution.

Inspired by retrieval-augmented generation (RAG) in large language models, recent studies address this challenge by retrieving relevant historical examples to provide local information beyond the pretrained model. However, effective retrieval requires a reliable measure of historical relevance. Existing methods commonly rely on numerical similarity measures such as DTW or Pearson correlation. Although each captures certain aspects of a time series, a single score cannot fully characterize its temporal structure, including trend persistence, fluctuation patterns, and the timing of turning points. Figure 1 illustrates this limitation: histories retrieved by Euclidean distance, cosine similarity and DTW have lower MSE than the time series retrieved by our method. However, their future trajectories differ more from the actual future. After aligning each segment to its best future point, they also have larger errors. This mismatch suggests that numerical proximity alone may miss temporal structures that are important for forecasting. This leads to our first challenge: ***how to represent time series in a way that preserves the temporal structures relevant to historical retrieval?***

Our opportunity comes from viewing a time series as a visual trajectory in the time-value plane. Recent vision-based time series studies motivate the use of visual representations for temporal mod-

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.

eling (Shen et al., 2025; Guan et al., 2026). In this view, trends, fluctuations, peaks, troughs, and turning points become explicit structures, making it possible to compare how patterns are organized over time. Iterative Largest Triangle Sampling (ILTS) (Rui et al., 2025) offers a useful visualizing method by selecting representative observations through triangle-area maximization to preserve salient temporal patterns. However, existing visual representations capture temporal patterns but provide no effective way to measure their similarity for retrieval. This leads to a second challenge: ***how to measure similarity between visual representations of time series for effective retrieval?***

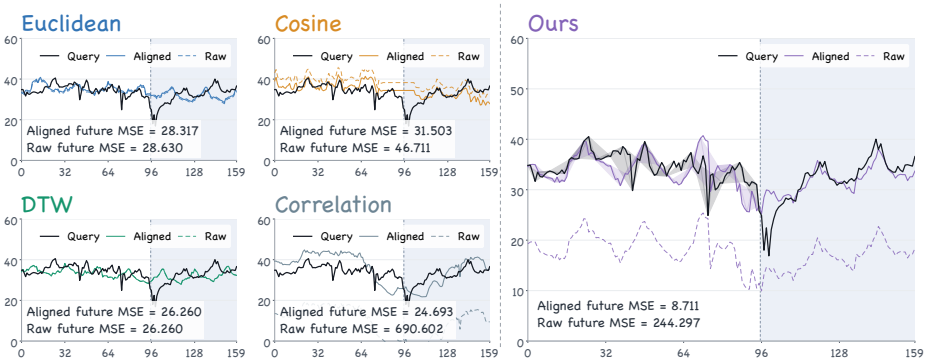


Figure 1: Retrieval examples on ETTh1 (OT channel) using Euclidean distance, cosine similarity, DTW, Pearson correlation and our method. Each panel compares the query with the corresponding Top-1 retrieved sequence. For each retrieved sequence, we select the alignment yielding the lowest future MSE among historical point alignments.

To address these challenges, we propose **ReViR** (**Re**solution-Variant **Vi**sual **R**etriever), a visual retrieval method for time series forecasting. First, we construct a region-area map through *Visual Sampling*. Second, we introduce *Approximate Triangle-IoU*, a resolution-variant similarity measure that compares the overlap of these regions on a shared time-value grid, capturing agreement in the shape and timing of salient changes in time series. ReViR uses this score to select historical time series, which are aggregated and fused with a lightweight forecaster. Configurable sampling density and raster resolution, packed binary masks, and candidate-restricted comparison are introduced to control the storage and computational cost of visual retrieval. Experiments on seven real-world datasets assess forecasting accuracy and the resulting efficiency trade-off.

Our contributions are as follows:

- We are the first to use visual representations of time series and direct similarity measurement between these representations for retrieval-augmented time series forecasting.
- We propose ReViR, a resolution-variant visual retrieval method for time series forecasting. ReViR connects visual matching to forecasting through retrieved historical continuations. Configurable sampling density and raster resolution, together with packed binary masks and candidate-restricted comparison, enable controllable storage and computational efficiency for visual retrieval.
- We achieve state-of-the-art forecasting performance across seven real-world benchmarks. ReViR obtains the best average MSE on five of seven datasets and the best average MAE on all seven.

2 Related Work

Time series forecasting has been studied through statistical models and learned predictors. Classical approaches explicitly model trends and seasonality, while early deep models such as LSTNet (Lai et al., 2018) combine convolutional and recurrent layers to capture short- and long-term patterns. Transformer-based methods extend the range of temporal dependencies: Informer (Zhou et al., 2021) reduces attention cost for long sequences, Autoformer (Wu et al., 2021) combines decomposition with autocorrelation-based dependencies, and FEDformer (Zhou et al., 2022) incorporates

frequency-domain modeling. Other architectures refine how temporal and variable relationships are represented. Crossformer (Zhang & Yan, 2023) explicitly models both cross-time and cross-variable dependencies, PatchTST (Nie et al., 2023) models each variable through temporal patches, and iTransformer (Liu et al., 2024) treats variables as tokens to learn their interactions. Simpler predictors and models with different temporal inductive biases have also proved effective. DLinear (Zeng et al., 2023) uses decomposition followed by linear extrapolation, while TiDE (Das et al., 2023) uses an MLP encoder-decoder to model nonlinear dependencies and covariates. TimesNet (Wu et al., 2023) organizes temporal variation around multiple periods; TimeMixer (Wang et al., 2024) combines information across temporal scales; and DUET (Qiu et al., 2025) uses dual clustering to adapt to changing temporal patterns and variable relationships.

Despite the diversity of forecasting architectures, most existing predictors primarily rely on patterns encoded within the input sequence and learn a direct mapping from historical observations to future values. However, many time series exhibit recurring or structurally similar temporal patterns across different periods, variables, or instances. Retrieving historically similar subsequences can therefore provide complementary contextual information that is not readily available from the current input alone. Many retrieval-augmented forecasting methods select historical examples using observed numerical sequences or representations learned from them. TimeRAG (Yang et al., 2025) uses dynamic time warping to compare temporal patterns, RAFT (Han et al., 2025) primarily uses Pearson correlation to retrieve input-continuation pairs, and SARAF (Zhou et al., 2026) incorporates temporal and stationarity-aware scores. Other methods exploit graph relations, pattern-specific similarity, or hierarchical organization (Jing et al., 2022; Wen et al., 2025; Ma et al., 2026). RAF and TS-RAG retrieve neighbors in pretrained time-series representation spaces (Tire et al., 2026; Ning et al., 2025); TimeRAF learns the retriever end to end (Zhang et al., 2025), while Cross-RAG models query-retrieval relevance through cross-attention (Lee et al., 2026). However, numerical or representation-level similarity alone is insufficient to characterize the non-stationary and evolving nature of time series, such that retrieved sequences may appear similar to the query historically but exhibit substantially different future dynamics, limiting their value as predictive references.

3 METHOD

Figure 2 provides an overview of **ReViR**, which consists of three parts. (1) The numerical and temporal recalling module identifies relevant historical windows. (2) The resolution-variant visual reranking selects candidates with similar structures of time series. (3) The forecaster combines their observed continuations with a base prediction to produce the final predictions.

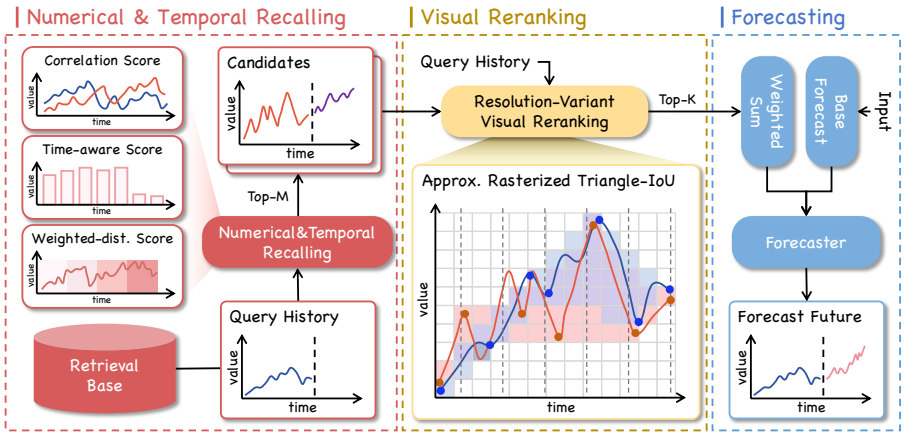


Figure 2: Overview of ReViR. Left: Numerical & Temporal Recalling selects the top-M historical windows using complementary numerical and time-aware scores. Middle: Resolution-Variant Visual Reranking measures overlap between Visual Sampling derived triangle regions through binary occupancy grids and selects the top-K candidates using the combined recall and visual scores. Right: the selected historical continuations are aggregated and fused with the base forecaster's prediction to produce the final forecast.

3.1 Problem Setup

In multivariate time series forecasting, given historical observations $Q = \{q_1, \ldots, q_L\} \in \mathbb{R}^{L\times C}$, where L is the number of observed time steps and C is the number of variables, our goal is to learn a forecasting function $f : \mathbb{R}^{L\times C} \to \mathbb{R}^{H\times C}$ that predicts the next H observations, i.e., $\hat{Y} = f(Q)$, where $Y = \{q_{L+1}, \ldots, q_{L+H}\} \in \mathbb{R}^{H\times C}$ denotes the ground-truth future observations. Standardization is detailed in Appendix A.3.

3.2 Resolution-variant Visual Retriever

3.2.1 Retrieval Knowledge Base

We construct the retrieval knowledge base from the standardized training sequence by sliding a window of length $L + H$ with stride one. This yields $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^{N}$, where candidate i is the i-th input-continuation pair: $X^{(i)} \in \mathbb{R}^{L\times C}$ contains the observed history and $Y^{(i)} \in \mathbb{R}^{H\times C}$ contains its following observations. We retain the timestamps associated with $X^{(i)}$ for time-aware matching. Candidate relevance is computed from $X^{(i)}$ and its observed timestamps; $Y^{(i)}$ is used only after retrieval to form the forecast. For training queries, candidates with overlapping input-continuation intervals are excluded, while validation and test queries search only the training library. The eligibility rules are detailed in Appendix A.3.

3.2.2 Numerical & Temporal Recalling

Numerical matching can identify multiple historical segments with relevance to the query, but a single criterion cannot capture all relevant temporal patterns. We therefore combine global correlation, recent numerical agreement and time-aware alignment into a recall score:

$$S^{(i)}_{\text{recall}} = \frac{S^{(i)}_{\text{corr}} + S^{(i)}_{\text{num}} + S^{(i)}_{\text{time}}}{3}, \tag{1}$$

where $S^{(i)}_{\text{corr}}$ represents global co-variation, $S^{(i)}_{\text{num}}$ represents recent numerical agreement, and $S^{(i)}_{\text{time}}$ represents recurring temporal similarity. Each term lies in $[0, 1]$ and evaluates a different relationship between the standardized, endpoint-aligned query $\widetilde{Q}$ and candidate $\widetilde{X}^{(i)}$. Detailed definitions are given in Appendix C.1. After applying the numerical recalling, we retain the M windows with the highest $S^{(i)}_{\text{recall}}$ as the candidate pool $\mathcal{M}(Q)$ for visual reranking.

3.2.3 Resolution-Variant Visual Reranking

Given the candidate pool $\mathcal{M}(Q)$, we further assess their structural relevance to the query. We design a resolution-variant visual reranking mechanism to refine their ranking through visual similarity. As illustrated in Figure 3, it includes four parts: (1) Temporal Bucketing, (2) Visual Sampling, (3) Region Rendering, and (4) Approximate Rasterizing.

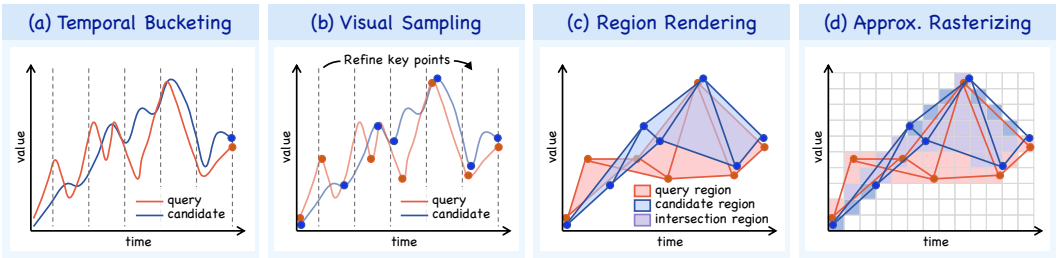


Figure 3: Details of visual reranking for a query (red) and a recalled candidate (blue): (a) Temporal Bucketing partitions each time series into contiguous buckets; (b) Visual Sampling iteratively selects representative points within these buckets; (c) Region Rendering forms triangle regions from the final sampling pass; and (d) Approximate Rasterizing maps these regions onto a shared grid to estimate Triangle-IoU from their intersection and union.

Temporal Bucketing. For each candidate $m \in \mathcal{M}(Q)$, we channel-independently apply the same temporal partition to the aligned query $\widetilde{q}_c$ and each recalled candidate $\widetilde{m}_c$, where $c \in \{1, \ldots, C\}$ indexes the c-th variable of the multivariate time series. We describe the procedure for the query using points $\widetilde{q}_c(t)$, where $t \in \{1, \ldots, L\}$ indexes time. We divide the first $L-1$ time indices into B contiguous buckets $\{\mathcal{B}_1, \ldots, \mathcal{B}_B\}$ and reserve $\widetilde{q}_c(L)$ as the terminal anchor (Figure 3(a)). This partition distributes the subsequent anchors across the history, while B controls sampling density.

Visual Sampling. Inspired by iterative largest-triangle sampling (Rui et al., 2025), we refine one visual anchor per bucket through repeated bucket-wise updates. By favoring observations that deviate from local reference trends, this procedure retains salient peaks, troughs and turns. At iteration $j \in \{1, \ldots, J\}$, buckets are processed from left to right. The left reference $u_b^{(j)}$, where $b \in \{1, \ldots, B\}$ indexes buckets, is $\widetilde{q}_c(1)$ for the first bucket and the newly selected point from the preceding bucket otherwise. The right reference $v_b^{(j)}$ is the next bucket's mean point initially, and its previous-iteration anchor thereafter. The final bucket always uses $\widetilde{q}_c(L)$. The channel index is omitted from $u_b^{(j)}$, $t_b^{(j)}$ and $v_b^{(j)}$ for clarity. Each bucket selects a time index $t_b^{(j)}$ and its corresponding point $p_b^{(j)}$ as follows:

$$t_b^{(j)} = \arg\max \Delta\big(u_b^{(j)}, \widetilde{q}_c(t), v_b^{(j)}\big), \quad t \in \left\{ \left\lceil \frac{(b-1)(L-1)}{B} \right\rceil + 1, \ldots, \left\lceil \frac{b(L-1)}{B} \right\rceil \right\}, \tag{2}$$

$$p_b^{(j)} = \widetilde{q}_c\big(t_b^{(j)}\big), \quad \Delta(u, p, v) = \frac{1}{2} \left| (v_x - u_x)(p_y - u_y) - (v_y - u_y)(p_x - u_x) \right|. \tag{3}$$

The function $\Delta(u, p, v)$ gives the area of the triangle formed by left reference u, trial point p, and right reference v. Subscripts x and y denote their time and value coordinates. Maximizing this area favors departures from the local trend, with ties resolved by the earliest observation. We perform J iterations to obtain the visual anchors shown in Figure 3(b). The iteration budget J controls the degree of refinement. Further implementation details are provided in Appendix C.2.

Region Rendering. As illustrated in Figure 3(c), we retain the filled triangles induced by the final Visual Sampling pass. For query channel $\widetilde{q}_c$, the triangle collection and its covered region are

$$\mathcal{A}(\widetilde{q}_c) = \bigcup_{b=1}^{B} \mathrm{conv}(u_b^{(J)}, p_b^{(J)}, v_b^{(J)}). \tag{4}$$

Here $\mathrm{conv}(\cdot)$ denotes the filled triangle determined by its three vertices. Applying the same construction to the independently sampled candidate channel $\widetilde{m}_c$ yields $\mathcal{A}(\widetilde{m}_c)$. Under shared time-value coordinates, similarly shaped time series are expected to have nearby visual anchors and overlapping triangles. Temporal shifts, amplitude differences and missing local structures can reduce this overlap. We therefore introduce **Triangle-IoU**, a visual similarity measure defined by the intersection-over-union of the resulting triangle regions. For channel c, it is

$$\text{Triangle-IoU}(\widetilde{q}_c, \widetilde{m}_c) = \frac{\mathrm{Area}(\mathcal{A}(\widetilde{q}_c) \cap \mathcal{A}(\widetilde{m}_c))}{\mathrm{Area}(\mathcal{A}(\widetilde{q}_c) \cup \mathcal{A}(\widetilde{m}_c))}. \tag{5}$$

Here $\mathrm{Area}(\cdot)$ measures the area of a region in the shared time-value plane. Taking triangle unions avoids double-counting overlapping triangles within a time series and yields a visual similarity interpretable through the regions themselves.

Approximate Rasterizing. Exact $\text{Triangle-IoU}(\widetilde{q}_c, \widetilde{m}_c)$ requires computing intersections and unions of polygonal regions for each query-candidate pair. We instead use a *rasterized approximation of Triangle-IoU* (Figure 3(d)), replacing continuous areas with binary occupancy counts on a uniform $G_x \times G_y$ grid over $[1, L] \times [\mathrm{low}_c, \mathrm{up}_c]$. Here G_x and G_y are the numbers of grid cells along the time and value axes. The lower and upper value bounds $\mathrm{low}_c, \mathrm{up}_c$ for channel c are determined exclusively from training statistics and shared across windows and splits; out-of-range vertices are clipped. Shared scaling preserves amplitude differences that independent rescaling could obscure. For grid indices $r \in \{1, \ldots, G_x\}$ and $s \in \{1, \ldots, G_y\}$, let $g_{r,s}^{(c)}$ be the cell center in channel c's grid. The query occupancy mask is $R(\widetilde{q}_c)[r, s] = \mathbf{1}[g_{r,s}^{(c)} \in \mathcal{A}(\widetilde{q}_c)]$, where the indicator equals one when the cell center lies in the region and zero otherwise. The candidate mask $R(\widetilde{m}_c)$ is defined identically using $\mathcal{A}(\widetilde{m}_c)$. For candidate m, the intersection count $I_c^{(i)}$ and union count $U_c^{(i)}$ are

$$I_c^{(i)} = \sum_{r=1}^{G_x} \sum_{s=1}^{G_y} \big(R(\widetilde{q}_c)[r, s] \wedge R(\widetilde{m}_c)[r, s]\big), \quad U_c^{(i)} = \sum_{r=1}^{G_x} \sum_{s=1}^{G_y} \big(R(\widetilde{q}_c)[r, s] \vee R(\widetilde{m}_c)[r, s]\big). \tag{6}$$

Here $\wedge$ and $\vee$ denote logical AND and OR. Because all cells have equal area, the ratio of these counts approximates $\mathrm{Triangle\text{-}IoU}(\widetilde{q}_c, \widetilde{m}_c)$. Averaging across channels gives the multivariate visual score for candidate $m = X^{(i)}$:

$$S_{\mathrm{IoU}}^{(i)} = \frac{1}{C} \sum_{c=1}^{C} \frac{I_c^{(i)}}{U_c^{(i)}}. \tag{7}$$

Empty unions receive zero similarity. Discretization affects boundary cells, and grid resolution controls the fidelity–cost trade-off. Historical masks are cached with eight occupancy values per byte, enabling intersection and union counts through bitwise AND/OR and population counts. Reranking thus uses fixed-size mask comparisons for only the recalled candidates.

Resolution-variant representation. We control representation granularity through the raster dimensions G_x, G_y and the bucket count B, which determines the number of selected anchors. A finer configuration combines more grid cells with denser sampling to retain more local time series detail. A coarser configuration reduces these budgets to limit storage and computation. These parameters are separately configurable: increasing raster resolution alone does not add Visual Sampling anchors. We select a configuration for each dataset and apply the same grid and coordinate mapping to its queries and candidates, ensuring that occupancy masks remain comparable. This flexibility is particularly useful when the historical library or channel count is large, where a uniformly fine representation would be expensive. Detailed computational costs are given in Appendix C.4.

Joint retrieval score. We equally weight the numerical-temporal and visual similarity scores for each recalled candidate, then select the K highest-scoring historical segments. Their continuations are aggregated into a retrieval-based forecast $\widehat{\widetilde{Y}}_{\mathrm{ret}} \in \mathbb{R}^{H \times C}$, which provides historical predictive context for the forecasting module. The exact scoring and selection expressions are given in Appendix C.3.

3.3 Forecasting

After selecting the continuations, we aggregate them with Gaussian-kernel weights based on their final retrieval scores. To isolate the contribution of retrieval, we use three lightweight affine maps, $f_{\mathrm{base}} : \mathbb{R}^L \to \mathbb{R}^H$ and $f_{\mathrm{ret}}, f_{\mathrm{out}} : \mathbb{R}^H \to \mathbb{R}^H$, applied along time and shared across channels. The final forecast is

$$\hat{Y} = f_{\mathrm{out}} \left(f_{\mathrm{base}}(\widetilde{Q}) + f_{\mathrm{ret}}(\widehat{\widetilde{Y}}_{\mathrm{ret}}) \right). \tag{8}$$

4 Experiments

We evaluate ReViR on seven real-world datasets to assess its forecasting accuracy and the contribution of the visual similarity. We compare against retrieval-based and non-retrieval forecasting methods, ablate the main retrieval components, and analyze sensitivity to retrieval budgets and Visual Sampling iterations. We further examine forecasting accuracy, storage requirements, retrieval latency, and throughput under compact visual configurations.

4.1 Experimental Setup

Datasets and protocol. We evaluate ReViR on seven widely used multivariate forecasting benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Exchange, Weather, and Electricity. We use chronological training, validation, and test splits. Dataset descriptions and leakage-prevention details are provided in Appendix A.

Baselines. Retrieval-augmented baselines are SARAF (Zhou et al., 2026), RAFT (Han et al., 2025), GTR (Cao et al., 2026) and TS-RAG (Ning et al., 2025); non-retrieval baselines are DUET (Qiu et al., 2025), PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024), DLinear (Zeng et al., 2023) and TimeMixer (Wang et al., 2024).

Evaluation. We use an input length of $L = 720$ and prediction horizons $H \in \{96, 192, 336, 720\}$ in the main experiments. We report mean squared error (MSE) and mean absolute error (MAE), where lower values indicate better performance. Unless otherwise stated, methods in the main comparison use the same dataset splits and forecasting horizons. Metric definitions are given in Appendix A.6.

Implementation details. We implement ReViR in PyTorch and run the experiments on an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory. The forecasting model is optimized with Adam. Training and retrieval configuration details are provided in Appendix A.4.

Table 1: Multivariate forecasting results on seven datasets at prediction horizons of {96, 192, 336, 720} steps. Avg denotes the arithmetic mean across the four horizons. For each row and metric, the best and second-best distinct values are shown in **bold** with red shading and underlined with blue shading, respectively. The 1st count includes both horizon-specific and Avg rows across both metrics, counting tied best values for each method.

Dataset	Horizon	Retrieval-Augmented Models								Non-Retrieval-Augmented Models									
		ReViR		SARAF		GTR		RAFT		DUET		TimeMixer		PatchTST		DLinear		iTransformer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.362**	**0.385**	0.373	0.398	0.383	0.413	0.379	0.412	0.382	0.415	0.410	0.437	0.426	0.439	0.369	0.399	0.392	0.424
	192	**0.403**	**0.411**	0.404	0.418	0.417	0.433	0.409	0.429	0.412	0.434	0.440	0.453	0.466	0.469	0.406	0.422	0.437	0.455
	336	0.434	**0.429**	0.432	0.441	0.460	0.462	0.438	0.446	**0.430**	0.448	0.880	0.668	0.531	0.512	0.440	0.452	0.457	0.470
	720	**0.442**	**0.457**	0.449	0.470	0.492	0.489	0.467	0.477	0.457	0.480	0.715	0.608	0.525	0.515	0.488	0.507	0.553	0.537
	Avg	**0.410**	**0.420**	0.415	0.432	0.438	0.449	0.423	0.441	0.420	0.444	0.611	0.542	0.487	0.484	0.426	0.445	0.460	0.472
ETTh2	96	**0.270**	**0.330**	0.275	0.338	0.289	0.351	0.281	0.348	0.276	0.340	0.288	0.352	0.339	0.383	0.304	0.370	0.303	0.364
	192	0.336	**0.376**	**0.332**	0.379	0.359	0.398	0.349	0.394	0.333	**0.376**	0.384	0.422	0.399	0.420	0.386	0.423	0.410	0.423
	336	0.371	**0.406**	**0.355**	**0.406**	0.383	0.419	0.373	0.421	0.371	0.411	0.385	0.423	0.504	0.472	0.539	0.512	0.440	0.450
	720	0.411	0.446	**0.400**	0.448	0.408	0.444	0.432	0.472	0.404	**0.440**	1.108	0.761	0.538	0.515	0.864	0.658	0.439	0.469
	Avg	0.347	**0.390**	**0.341**	0.393	0.360	0.403	0.359	0.409	0.346	0.392	0.541	0.490	0.445	0.448	0.523	0.491	0.398	0.427
ETTm1	96	0.290	**0.338**	0.298	0.349	0.296	0.349	0.300	0.351	**0.288**	0.347	0.347	0.382	0.294	0.353	0.306	0.349	0.318	0.366
	192	**0.324**	**0.357**	0.330	0.369	0.337	0.375	0.329	0.368	0.330	0.374	0.406	0.419	0.354	0.397	0.342	0.374	0.348	0.388
	336	**0.353**	**0.376**	0.367	0.393	0.371	0.393	0.355	0.384	0.357	0.392	0.451	0.448	0.371	0.405	0.374	0.398	0.386	0.413
	720	0.404	**0.406**	**0.402**	0.411	0.426	0.425	0.405	0.411	0.413	0.427	0.566	0.503	0.431	0.444	0.419	0.419	0.442	0.443
	Avg	**0.343**	**0.369**	0.349	0.381	0.358	0.386	0.347	0.379	0.347	0.385	0.443	0.438	0.363	0.400	0.360	0.385	0.374	0.403
ETTm2	96	**0.160**	**0.246**	**0.160**	0.252	0.173	0.259	0.165	0.257	0.164	0.253	0.186	0.272	0.197	0.284	0.162	0.256	0.178	0.272
	192	**0.216**	**0.286**	**0.216**	0.292	0.231	0.302	0.219	0.296	0.224	0.293	0.242	0.310	0.243	0.321	0.221	0.300	0.245	0.317
	336	**0.267**	**0.321**	0.268	0.329	0.287	0.340	0.275	0.335	0.276	0.329	0.292	0.344	0.294	0.347	0.282	0.345	0.299	0.352
	720	**0.348**	**0.375**	0.350	0.383	0.359	0.391	0.369	0.398	0.357	0.384	0.394	0.406	0.432	0.429	0.402	0.427	0.382	0.404
	Avg	**0.248**	**0.307**	0.249	0.314	0.263	0.323	0.257	0.322	0.255	0.315	0.279	0.333	0.292	0.345	0.267	0.332	0.276	0.336
Exchange	96	**0.083**	**0.200**	0.086	0.203	0.095	0.218	0.089	0.206	0.147	0.277	0.167	0.306	0.114	0.241	0.119	0.261	0.108	0.239
	192	**0.169**	**0.291**	0.190	0.307	0.200	0.320	0.192	0.309	0.258	0.372	0.188	0.311	0.250	0.370	0.255	0.385	0.243	0.358
	336	**0.307**	**0.400**	0.365	0.436	0.351	0.435	0.370	0.442	0.440	0.491	0.374	0.447	0.308	0.402	0.452	0.519	0.483	0.515
	720	**0.752**	**0.648**	0.985	0.742	1.072	0.772	1.133	0.807	0.871	0.699	1.115	0.787	1.244	0.814	0.968	0.746	0.934	0.736
	Avg	**0.328**	**0.385**	0.407	0.422	0.430	0.436	0.446	0.441	0.429	0.460	0.461	0.463	0.479	0.457	0.449	0.478	0.442	0.462
Weather	96	0.162	0.205	0.164	0.225	**0.148**	0.202	0.165	0.222	0.150	**0.198**	0.173	0.230	0.154	0.210	0.169	0.228	0.173	0.226
	192	0.202	0.245	0.208	0.263	**0.194**	0.245	0.211	0.264	0.200	**0.244**	0.229	0.285	0.203	0.255	0.211	0.267	0.216	0.261
	336	0.247	**0.282**	0.259	0.305	0.248	0.286	0.260	0.302	0.253	0.284	**0.246**	0.284	0.257	0.300	0.256	0.304	0.287	0.312
	720	**0.315**	**0.333**	0.324	0.354	0.324	0.340	0.327	0.355	0.323	0.337	0.331	0.341	0.344	0.351	0.316	0.357	0.355	0.361
	Avg	0.232	**0.266**	0.239	0.287	**0.229**	0.268	0.241	0.286	0.232	**0.266**	0.245	0.285	0.240	0.279	0.238	0.289	0.258	0.290
Electricity	96	0.130	0.226	0.130	0.228	**0.126**	**0.222**	0.131	0.229	0.131	0.224	0.132	0.227	0.131	0.225	0.173	0.226	0.133	0.230
	192	**0.142**	**0.238**	0.143	0.241	0.146	0.240	0.147	0.243	0.154	0.245	0.155	0.249	0.148	0.241	0.216	0.261	0.153	0.249
	336	**0.151**	**0.249**	0.160	0.260	0.159	0.256	0.160	0.257	0.167	0.261	0.167	0.264	0.168	0.266	0.287	0.312	0.171	0.267
	720	**0.167**	**0.264**	0.173	0.274	0.193	0.289	0.187	0.282	0.195	0.285	0.212	0.300	0.199	0.291	0.355	0.361	0.192	0.286
	Avg	**0.148**	**0.244**	0.152	0.251	0.156	0.252	0.156	0.253	0.162	0.254	0.167	0.260	0.162	0.256	0.258	0.290	0.162	0.258
1st count		54		8		5		0		7		1		0		0		0	

4.2 Forecasting Performance

Table 1 shows that ReViR achieves the lowest average MSE on five of the seven datasets and the lowest average MAE on all seven, tying with DUET on Weather, with the best result in 54 of the 70 dataset-horizon-metric entries. Overall, ReViR outperforms the strongest retrieval baseline in 12 of 14 dataset-level metric comparisons and leads in 11 of 14 comparisons at the longest horizon.

4.3 Effectiveness of Components in ReViR

Table 2 evaluates the contribution of historical retrieval and visual reranking on four datasets at prediction horizons of 96 and 336 steps. ReViR consistently outperforms *w/o Retr* in all eight dataset-

Table 2: Component ablation on four datasets.

Dataset	H	ReViR		w/o Retr		w/o RVVR	
		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.362**	**0.385**	0.367	0.391	0.363	0.389
	336	**0.434**	**0.429**	0.451	0.455	0.438	0.448
ETTm1	96	**0.290**	**0.338**	0.298	0.340	0.292	0.339
	336	0.353	**0.376**	0.362	0.378	**0.352**	**0.376**
Exchange	96	**0.083**	**0.200**	0.086	0.207	**0.083**	0.201
	336	**0.307**	**0.400**	0.319	0.405	0.309	0.401
Weather	96	**0.162**	**0.205**	0.170	0.209	0.164	0.209
	336	**0.247**	**0.282**	0.258	0.288	0.253	0.288

Table 3: Backbone performance comparison.

Backbone	Dataset	H	Original		+ Retrieval	
			MSE	MAE	MSE	MAE
DUET	ETTh1	96	0.382	0.415	**0.368**	**0.397**
		336	**0.430**	0.448	0.435	**0.445**
	ETTm1	96	0.288	0.347	**0.287**	**0.339**
		336	0.357	0.392	**0.351**	**0.378**
PatchTST	ETTh1	96	0.426	0.439	**0.401**	**0.426**
		336	0.531	0.512	**0.458**	**0.461**
	ETTm1	96	0.294	0.353	**0.290**	**0.351**
		336	0.371	0.405	**0.370**	**0.403**

horizon settings for both MSE and MAE, confirming that access to retrieved historical continuations provides useful predictive information beyond the lightweight base forecaster. Adding resolution-variant visual reranking on top of numerical and temporal retrieval provides further improvements in most settings. Compared with *w/o RVVR*, the complete ReViR achieves lower MSE in six of eight settings, while achieving lower MAE in seven settings.

4.4 Parameter Sensitivity Analysis

Figure 4 evaluates parameter sensitivity on ETTm1 with a prediction horizon of 96 steps. Starting from $(M, K, J) = (100, 20, 2)$, we vary the recall pool size M, the number of retrieved candidates K, and Visual Sampling iteration count J one at a time while holding the other two fixed. All configurations satisfy $M \geq K$. Each panel reports the percentage change in MSE and MAE relative to the reference; a positive value indicates higher error.

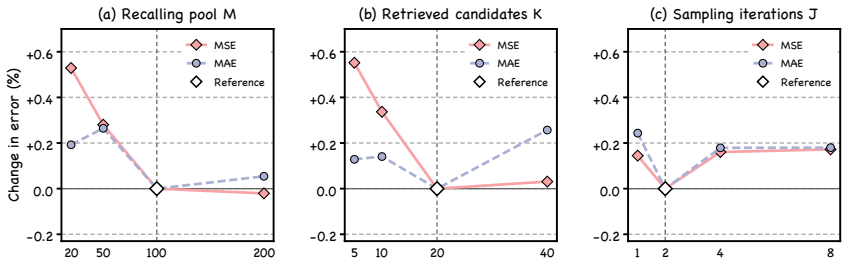


Figure 4: Parameter sensitivity on ETTm1 with a prediction horizon of 96 steps. Panels vary the recall pool size M, the number of retrieved candidates K, and the Visual Sampling iteration count J separately; each shows relative changes in both MSE and MAE. Hollow diamonds and vertical dashed lines mark the reference $(M, K, J) = (100, 20, 2)$.

Retrieval budgets. Figure 4(a) shows that increasing M from 20 or 50 to 100 generally lowers MSE. Figure 4(b) shows a similar trend when K increases from 5 or 10 to 20. These results suggest that very small recall and retrieval pools can omit useful historical context. Thus, expanding the retrieval budget beyond the reference offers benefit in this setting. $(M, K) = (100, 20)$ is a balanced choice for this experiment, not a universal optimum.

Visual Sampling iterations. Figure 4(c) shows that moving from one to two refinement passes slightly lowers both errors, whereas four and eight passes give nearly identical results and no further improvement. It suggests that the visually salient anchors stabilize quickly for this configuration, so extra iterations bring little forecasting benefit. We therefore use $J = 2$ as a practical iteration budget and the observed plateau does not imply that Visual Sampling has found a global optimum.

4.5 Retrieval Efficiency

We compare retrieval efficiency on ETTm2 and Weather at prediction horizons of 96 and 336 steps, using RAFT as the primary reference. ReViR uses the compact configurations `g16-b24` on ETTm2 and `g16-b8` on Weather. Here, `g` denotes the raster resolution along each axis, and `b` denotes the number of temporal buckets used for visual sampling. Thus, `g16-b24` uses a 16×16 grid with $B = 24$ buckets, while `g16-b8` uses the same grid with $B = 8$ buckets. The complete measurements and profiling protocol are provided in Appendix B.2 (Table 7).

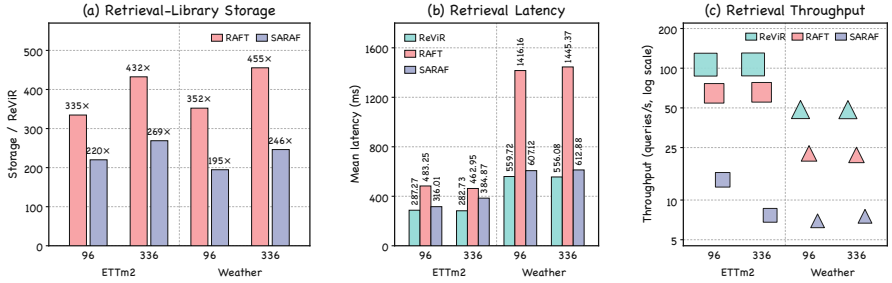


Figure 5: Retrieval efficiency on ETTm2 and Weather at prediction horizons of 96 and 336 steps. Dataset–horizon groups are shared across the horizontal axes. (a) Vertical bars show the retrieval-library storage ratios of RAFT and SARAF to ReViR. (b) Vertical bars show mean query-level retrieval latency in milliseconds. (c) Scatter points show batched retrieval throughput and larger markers indicate higher throughput. Latency and throughput are averaged over three independent profiling runs.

As shown in Figure 5(a), ReViR reduces retrieval-library storage by 99.70–99.78% relative to RAFT across the evaluated settings. The rasterized and bit-packed visual representation therefore cuts storage by more than two orders of magnitude. Figure 5(b) shows that ReViR lowers mean query-level retrieval latency in all four settings, reducing latency by 39–62% relative to RAFT. At $H = 336$, it achieves a $1.64\times$ speedup on ETTm2 and a $2.60\times$ speedup on Weather. Figure 5(c) shows consistent throughput gains under batched query processing. Relative to RAFT, ReViR increases throughput by 65% and 63% on ETTm2 and by 115% and 121% on Weather at horizons 96 and 336, respectively.

4.6 Forecasting with Alternative Forecasters

We replace ReViR's default lightweight forecaster with DUET or PatchTST while retaining the same retriever. Original denotes each standalone forecaster; + Retrieval adds ReViR's retrieval module. Table 3 compares each resulting model with its standalone forecaster. Retrieval reduces MAE in all eight settings and MSE in seven, yielding MSE/MAE reductions of 1.1%/2.7% for DUET and 5.3%/3.5% for PatchTST. These results demonstrate that our retrieval module can serve as a plug-and-play component to improve the forecasting performance of different backbones.

5 Conclusion

We introduced ReViR, a resolution-variant visual retriever that measures the visual representations' similarity for time series forecasting. By extracting visual anchors with Visual Sampling and comparing their triangle regions' area through Approximate Triangle-IoU, ReViR makes salient temporal structure explicit in historical matching. Configurable sampling density and raster resolution, together with packed binary masks, enable compact visual representations and efficient similarity computation. A lightweight forecasting head aggregates the retrieved continuations and combines them with a direct prediction from the query. Experiments on seven real-world datasets demonstrate strong forecasting performance, while ablation studies support the value of visual similarity and efficiency evaluations show substantial storage savings and improved retrieval throughput. These results establish resolution-variant visual similarity as an effective means of enriching historical context for retrieval-augmented time series forecasting.

AI USE STATEMENT

We used generative AI tools to assist with writing including improving grammar, clarity and the presentation of technical descriptions. We also used these tools to assist with editing plotting code and scientific figures. We reviewed and revised the AI-assisted content and verified reported numerical comparisons against the experimental records. We take responsibility for the final content of this work, including all text, claims, code, and figures produced with the aid of generative AI.

ETHICS STATEMENT

Our study solely targets the retrieval-augmented time series forecasting task, and therefore carries no underlying ethical risks.

REFERENCES

Fanpu Cao, Lu Dai, Jindong Han, and Hui Xiong. Enhancing multivariate time series forecasting with global temporal retrieval. In *International Conference on Learning Representations*, 2026.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=pCbC3aQB5w.

Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. doi: 10.1016/j.ejor.2017.11.054.

Tong Guan, Sheng Pan, Johan Barthelemy, Zhao Li, Yujun Cai, Cesare Alippi, Ming Jin, and Shirui Pan. TimeOmni-VL: Unified models for time series understanding and generation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 21774–21797, 2025.

Tao Hong and Shu Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016. doi: 10.1016/j.ijforecast.2015.11.011.

Baoyu Jing, Si Zhang, Yada Zhu, Bin Peng, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. Retrieval based time series forecasting, 2022.

Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafał Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293:116983, 2021. doi: 10.1016/j.apenergy.2021.116983.

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

Seunghan Lee, Jaehoon Lee, Jun Seo, Sungdong Yoo, Minjae Kim, Tae Yoon Lim, Dongwan Kang, Hwanil Choi, SoonYoung Lee, and Wonbin Ahn. Cross-RAG: Zero-shot retrieval-augmented time series forecasting via cross-attention, 2026.

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

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

Shichao Ma, Zhengyang Zhou, Qihe Huang, Binwu Wang, and Yang Wang. QuiZSF: A retrieval-augmented framework for zero-shot time series forecasting. In *Proceedings of the ACM Web Conference 2026*, pp. 1947–1958, 2026. doi: 10.1145/3774904.3792141.

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

Kanghui Ning, Zijie Pan, Yu Liu, Yushan Jiang, James Y. 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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5448.

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

Lei Rui, Xiangdong Huang, Shaoxu Song, Chen Wang, Jianmin Wang, and Zhao Cao. Largest triangle sampling for visualizing time series in database. *Proceedings of the ACM on Management of Data*, 3(1):1–26, 2025. doi: 10.1145/3709699.

Lefei Shen, Mouxiang Chen, Xu Liu, Han Fu, Xiaoxue Ren, Jianling Sun, Zhuo Li, and Chenghao Liu. VisionTS++: Cross-modal time series foundation model with continual pre-trained visual backbones, 2025.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300, pp. 4294–4302, 2026.

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*, 2024.

Zimo Wen, Hanwen Hu, Nan Fang, Shiyou Qian, and Jian Cao. DANet: A RAG-inspired dual attention model for few-shot time series prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 3302–3311, 2025. doi: 10.1145/3746252.3761012.

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

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Silin Yang, Dong Wang, Haoqi Zheng, and Ruochun Jin. TimeRAG: Boosting LLM time series forecasting via retrieval-augmented generation. In *2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025. doi: 10.1109/ICASSP49660.2025.10889933.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3634–3640, 2018. doi: 10.24963/ijcai.2018/505.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317.

Huanyu Zhang, Chang Xu, Yi-Fan Zhang, Zhang Zhang, Liang Wang, and Jiang Bian. TimeRAF: Retrieval-augmented foundation model for zero-shot time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 37(9):5654–5665, 2025. doi: 10.1109/TKDE.2025.3579137.

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

Zihao Zhang, Stefan Zohren, and Stephen Roberts. DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019. doi: 10.1109/TSP.2019.2907260.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325.

Shiqiao Zhou, Holger Schöner, Zipeng Wu, Edouard Fouché, IAG Wilson, and Shuo Wang. Stationarity-aware retrieval-augmented time series forecasting. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. doi: 10.1145/3770855.3817813.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 27268–27286, 2022. URL https://proceedings.mlr.press/v162/zhou22g.html.

A Implementation Details

A.1 Notation

Table 4 summarizes the principal symbols used in the method and appendix. Superscript (i) indexes a training-library candidate; subscripts t, c, and b index time, variable, and temporal bucket, respectively.

Table 4: Principal notation used in the main text and appendix.

Symbol	Meaning
L, H, C, N	Input length, prediction horizon, number of variables, and number of training-library pairs.
$Q, Y, \hat{Y}$	Query history, ground-truth future, and final forecast.
$\mathcal{D}, (X^{(i)}, Y^{(i)})$	Retrieval library and its i-th history–continuation pair.
$\widetilde{Q}, \widetilde{X}^{(i)}, \widetilde{Y}^{(i)}$	Endpoint-aligned query, candidate history, and continuation.
$S^{(i)}_{\mathrm{corr}}, S^{(i)}_{\mathrm{num}}, S^{(i)}_{\mathrm{time}}$	Correlation, recent-weighted numerical, and time-aware recall scores.
$S^{(i)}_{\mathrm{recall}}, \mathcal{M}(Q), M$	Combined recall score, recalled candidate pool, and pool size.
$\ell, P, \tau_{\mathrm{num}}$	Numerical patch length, number of patches, and calibrated distance scale.
κ, D	Calendar cycle length and number of observations per day in the time-aware score.
$\widetilde{q}_c(t), \widetilde{m}_c(t)$	Time-value points of the aligned query and recalled candidate $m = X^{(i)}$.
$\mathcal{B}_b, B, j, J$	Bucket b, number of buckets, sampling iteration, and iteration budget.
$u_b^{(j)}, v_b^{(j)}, t_b^{(j)}, p_b^{(j)}$	Left and right reference points, selected time index, and selected point.
$\Delta, \mathcal{A}(\cdot)$	Triangle-area function and rendered triangle region.
$G_x, G_y, low_c, up_c, R(\cdot)$	Raster dimensions, shared value-axis bounds, and binary occupancy mask.
$I_c^{(i)}, U_c^{(i)}, S^{(i)}_{\mathrm{IoU}}$	Per-channel intersection and union counts and multivariate visual score.
$S^{(i)}_{\mathrm{rank}}, K, \mathcal{K}(Q)$	Final retrieval score, selection count, and indices of selected candidates.
$\sigma, \widehat{\widetilde{Y}}_{\mathrm{ret}}$	Gaussian-kernel bandwidth and aggregated aligned retrieval forecast.

A.2 Dataset Details

We evaluate on seven multivariate time series datasets covering transformer monitoring, exchange rates, weather, and electricity consumption. Table 5 summarizes their dimensions, split sizes, and sampling frequencies. The dimension counts observed variables and excludes the timestamp column; the reported split sizes count time steps used in the experiments before constructing sliding windows.

ETT. The Electricity Transformer Temperature (ETT) benchmark, introduced by Informer (Zhou et al., 2021), contains records from two transformer stations. Each record includes seven variables: oil temperature and six power-load measurements. The benchmark comprises four datasets: ETTh1 and ETTh2 provide hourly observations from the two stations, respectively, with 17,420 time steps each; ETTm1 and ETTm2 provide the corresponding 15-minute series, with 69,680 time steps each. Together, these datasets support evaluation across different stations and temporal granularities.

Exchange. The Exchange dataset, used in LSTNet (Lai et al., 2018), contains daily exchange-rate series for eight countries: Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore. The benchmark contains 7,588 observations with one variable per exchange-rate series.

Weather. Weather contains measurements from the weather station of the Max Planck Institute for Biogeochemistry in Jena, Germany. We use the 2020 benchmark version evaluated by Autoformer (Wu et al., 2021), comprising 21 meteorological variables, including air temperature, atmospheric pressure, and humidity. The data are sampled every 10 minutes and contain 52,696 time steps. The original measurements are available from the station's public archive.

Electricity. Electricity contains hourly electricity-consumption series for 321 clients, with 26,304 time steps. We use the processed forecasting benchmark described in LSTNet (Lai et al., 2018) and subsequently used by Autoformer (Wu et al., 2021). It is derived from the UCI Electricity-LoadDiagrams20112014 dataset, whose original 15-minute measurements are aggregated to hourly observations.

Table 5: Dataset statistics. Dimension excludes the timestamp column. Dataset size is reported as training : validation : test time steps, excluding reused input context; these counts are not numbers of forecasting windows. Timestamp frequency denotes the interval between consecutive benchmark observations.

Dataset	Dimension	Dataset size (Train : Val : Test)	Timestamp frequency
ETTh1	7	8,640 : 2,880 : 2,880	1 hour
ETTh2	7	8,640 : 2,880 : 2,880	1 hour
ETTm1	7	34,560 : 11,520 : 11,520	15 minutes
ETTm2	7	34,560 : 11,520 : 11,520	15 minutes
Exchange	8	5,311 : 760 : 1,517	1 day
Weather	21	36,887 : 5,270 : 10,539	10 minutes
Electricity	321	18,412 : 2,632 : 5,260	1 hour

Data splits and preprocessing. Following the standard ETT loader, we use consecutive training, validation, and test intervals of 12, 4, and 4 months, respectively, with each month implemented as 30 days. Exchange, Weather, and Electricity use chronological splits of approximately 70%, 10%, and 20%. Validation and test windows use preceding observations as input context, while their forecast targets remain within the corresponding split. Each variable is standardized using the training-set mean and standard deviation, which are reused unchanged for validation and testing. All retrieval-related calibration parameters are likewise estimated exclusively from the training split. All seven datasets are evaluated in the multivariate setting, predicting every observed variable.

A.3 Endpoint Alignment and Retrieval Protocol

We align each history to its own endpoint element-wise: $\widetilde{Q}_{t,c} = Q_{t,c} - Q_{L,c}$ and $\widetilde{X}^{(i)}_{t,c} = X^{(i)}_{t,c} - X^{(i)}_{L,c}$. Each candidate continuation is expressed as $\widetilde{Y}^{(i)}_{h,c} = Y^{(i)}_{h,c} - X^{(i)}_{L,c}$, where $t \in \{1, \dots, L\}$, $h \in \{1, \dots, H\}$ and $c \in \{1, \dots, C\}$. This removes window-level offsets while preserving relative changes in amplitude. Alignment is applied once per window and shared by numerical and visual comparisons; the query endpoint $Q_{L,:}$ is added back to the model's final relative forecast. Candidates are complete multivariate windows, with a common ranking across channels. Training queries exclude candidates whose input–continuation intervals overlap the query's input–target interval. Validation and test queries retrieve only from the training library.

A.4 Training and Retrieval Configuration

We implement ReViR in PyTorch and run experiments on an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory. Trainable parameters are optimized with Adam. We train for at most 100 epochs and select the checkpoint with the lowest validation loss, stopping after five epochs without improvement. Learning rates and batch sizes vary across datasets and prediction horizons; the random seed is 2021.

The retrieval library is constructed from the training split. We cache the rasterized Visual Sampling representations of historical candidates as packed binary occupancy masks and reuse them across queries. The main retrieval hyperparameters are the recall pool size M, the number of selected candidates K, the numerical patch length ℓ, the Visual Sampling bucket count B, the iteration budget

J, and the raster dimensions $G_x \times G_y$. Unless otherwise specified, the numerical-temporal and visual similarity scores receive equal weights. The default Visual Sampling configuration uses $B = 24$ and $J = 2$, with a 128×128 raster grid; dataset-specific settings can use a coarser grid to reduce memory and comparison cost.

A.5 Baseline Methods

We compare ReViR with retrieval-augmented methods and forecasters that operate without a retrieval library. The former test whether a different way of selecting and using historical examples improves prediction; the latter establish the performance of forecasting architectures without external historical continuations. All methods in the main comparison are evaluated on the same multivariate datasets, chronological splits, input length, and prediction horizons described in Section A.2 and the main experimental setup. TS-RAG is reported separately because its published evaluation uses a different input–output setting and a pretrained forecasting backbone (Appendix B.4).

Retrieval-augmented methods. **RAFT** (Han et al., 2025) retrieves training histories that resemble the query and supplies their observed continuations alongside the current input to a forecaster. It serves as a direct reference for using input–continuation pairs as external forecasting evidence. **SARAF** (Zhou et al., 2026) augments candidate recall with temporal alignment and accounts for non-stationary regimes when selecting and aggregating retrieved examples. Its retrieval design tests whether relevance and diversity among numerical histories suffice without a direct visual comparison. **GTR** (Cao et al., 2026) is a plug-in global temporal retriever that extends a forecaster's effective context by retrieving and aligning segments from a learned representation of longer cycles. It emphasizes long-range periodic information that may lie outside the immediate input window. **TS-RAG** (Ning et al., 2025) uses a pretrained time-series encoder to find semantically related histories and a learnable mixture-of-experts module to combine retrieved information with a time-series foundation model. We compare against its published results in Appendix B.4; that supplementary comparison should be interpreted in light of the different backbone and adaptation protocol.

Forecasters without retrieval. **DLinear** (Zeng et al., 2023) decomposes the input into trend and seasonal components and extrapolates them with linear layers, providing a compact linear reference. **PatchTST** (Nie et al., 2023) divides each variable's history into temporal patches and models relations among patches with a Transformer. **iTransformer** (Liu et al., 2024) inverts the usual token organization so that each variable is a token, allowing attention to model inter-variable relationships. **TimeMixer** (Wang et al., 2024) mixes information across multiple temporal scales after decomposition to capture patterns at different frequencies. **DUET** (Qiu et al., 2025) uses dual clustering to adapt to changing temporal patterns and variable relationships. These baselines cover linear, patch-based Transformer, variable-oriented Transformer, multiscale, and clustering-based forecasting designs. The alternative-forecaster experiment integrates our retriever with DUET and PatchTST, so their standalone results indicate the effect of adding retrieved continuations to these two forecasters.

A.6 Evaluation Metrics

We evaluate forecasting accuracy using mean absolute error (MAE) and mean squared error (MSE). Let N_{test} denote the number of test windows, H the prediction horizon, and C the number of variables. For window n, let $y_{n,h,c}$ and $\widehat{y}_{n,h,c}$ be the ground-truth value and prediction at forecast step h for variable c. Both metrics average errors over all test windows, forecast steps, and variables:

$$\text{MAE} = \frac{1}{N_{\text{test}} HC} \sum_{n=1}^{N_{\text{test}}} \sum_{h=1}^{H} \sum_{c=1}^{C} \left| \widehat{y}_{n,h,c} - y_{n,h,c} \right|, \tag{9}$$

$$\text{MSE} = \frac{1}{N_{\text{test}} HC} \sum_{n=1}^{N_{\text{test}}} \sum_{h=1}^{H} \sum_{c=1}^{C} \left(\widehat{y}_{n,h,c} - y_{n,h,c} \right)^2. \tag{10}$$

MAE measures the average magnitude of prediction errors, whereas MSE gives greater weight to large errors through squaring. Lower values indicate better performance for both metrics. We compute the reported metrics on standardized values using statistics fitted exclusively on the training split, consistent with the evaluation implementation. For each dataset, the *Avg* entries in the main

results table are arithmetic means of the corresponding metric over the four prediction horizons $H \in \{96, 192, 336, 720\}$.

B Additional Experimental Results

B.1 Ablation of Numerical and Temporal Similarity

Table 6: Ablation of numerical and temporal similarity on ETTh1 and Electricity. ReViR is the full model; *w/o TA*, *w/o ND*, and *w/o Corr.* remove the time-aware score, recent-weighted numerical distance, and correlation, respectively; *L2-only* and *Cos-only* use only Euclidean distance and cosine similarity for candidate recall, respectively.

Dataset	H	ReViR		w/o TA		w/o ND		w/o Corr.		L2-only		Cos-only	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.362**	**0.385**	0.363	0.390	0.364	0.390	0.364	0.390	0.363	0.390	0.364	0.390
	336	**0.434**	**0.429**	0.441	0.453	0.438	0.451	0.437	0.449	0.441	0.454	0.442	0.453
Electricity	96	**0.130**	**0.226**	**0.130**	0.227	**0.130**	0.227	**0.130**	0.227	**0.130**	0.227	**0.130**	0.227
	336	**0.151**	**0.249**	0.158	0.253	**0.151**	**0.249**	0.152	**0.249**	0.158	0.253	0.158	0.253

Table 6 examines the numerical and temporal similarity components used for candidate recall. Removing an individual similarity component generally degrades performance, especially at the longer horizon. Both *L2-only* and *Cos-only* yield higher MAE in every setting and higher MSE in three of the four settings, tying with ReViR on Electricity at $H = 96$ at the reported precision. Together, these results show that global correlation, recent numerical agreement, and temporal context provide distinct signals for candidate recall before visual reranking.

B.2 Detailed Retrieval Efficiency Results

Table 7 reports the updated ETTm2 and Weather measurements underlying Figure 5, together with the earlier ETTh2 results. The updated experiments use an input length of 720, seed 2021, and an NVIDIA GeForce RTX 4090 GPU with 24 GB memory. The retrieval libraries contain 33,745 and 33,505 training windows on ETTm2 and 36,072 and 35,832 on Weather at $H = 96$ and $H = 336$, respectively. Library storage is retained from the original storage profiling; build time, latency, and throughput are updated from the latest records. Library storage counts the distinct backing bytes of CPU/GPU tensors retained after constructing the training-split retrieval library, including histories, continuations, time markers, and retrieval representations; it excludes model weights, data loaders, Python objects, and cached forecasts. This is in-memory tensor storage rather than a serialized index-file size. Each build-time measurement times a CUDA-synchronized cold call to `prepare_dataset(train)` after data loading and excludes model training and forecast precomputation. Build time for ETTm2 and Weather is averaged over three independent runs; ETTh2 retains its earlier measurement.

For query-level latency, we select the same 32 evenly spaced validation windows for each method and use batch size 1. We report the mean and 95th-percentile latency after two warmup runs and five timed repetitions with CUDA synchronization. Timing includes query encoding, retrieval-library transfer to the device, candidate ranking and refinement, and retrieval aggregation, but excludes data loading, input transfer, and forecasting. Throughput uses the same queries with batch size 32 and the same warmup and repetition counts. For ETTm2 and Weather, mean latency, p95 latency, and throughput are averages of the corresponding statistics from three independent profiling runs. Thus, p95 is the average of the per-run 95th percentiles rather than a percentile pooled across runs. RAFT and SARAF retain their native retrieval and training settings, so this comparison measures complete methods rather than an isolated replacement of the retrieval representation.

B.3 Effect of Raster Resolution

We evaluate the sensitivity of ReViR to raster resolution on ETTh1 with input length $L = 720$ and prediction horizons $H \in \{96, 336\}$. We vary the square raster grid from 16×16 to 256×256, while

Table 7: Detailed retrieval-efficiency measurements. ETTm2 and Weather build time, latency, and throughput are averaged over three independent runs; library storage and the earlier ETTh2 results are retained from the original measurements. Mean and p95 refer to query-level latency; throughput uses batch size 32. H is the prediction horizon. Lower storage, build time, and latency are better, while higher throughput is better.

Dataset	H	Method	Config.	Library (MiB)	Build (s)	Mean (ms)	p95 (ms)	Throughput (q/s)
ETTh2	96	ReViR	`g16-b24`	2.03	30.27	161.12	167.30	197.048
		RAFT	–	682.01	0.57	112.94	116.73	274.650
		SARAF	–	426.97	0.61	135.16	141.25	14.839
	336	ReViR	`g16-b24`	1.98	29.48	186.49	190.18	171.143
		RAFT	–	855.53	1.33	156.14	259.35	251.503
		SARAF	–	511.10	0.85	199.11	202.06	7.755
ETTm2	96	ReViR	`g16-b24`	8.79	129.09	287.27	311.15	106.388
		RAFT	–	2941.15	1.14	483.25	532.68	64.489
		SARAF	–	1933.99	1.99	316.01	322.51	14.242
	336	ReViR	`g16-b24`	8.74	129.12	282.73	304.49	106.802
		RAFT	–	3779.13	1.29	462.95	479.67	65.645
		SARAF	–	2349.69	1.35	384.87	388.86	7.658
Weather	96	ReViR	`g16-b8`	26.78	201.53	559.72	567.92	48.590
		RAFT	–	9431.91	3.98	1416.16	1430.21	22.587
		SARAF	–	5211.33	4.16	607.12	611.08	6.955
	336	ReViR	`g16-b8`	26.62	200.20	556.08	564.08	48.489
		RAFT	–	12124.79	4.35	1445.37	1454.69	21.985
		SARAF	–	6554.47	4.01	612.88	617.03	7.530

fixing the Visual Sampling bucket count $B = 24$, iteration budget $J = 2$, recall pool size $M = 100$, selected candidate count $K = 20$, and numerical patch length $\ell = 24$. All runs use seed 2021 and the same similarity scores. The forecasting model is retrained for each resolution, including 128×128, under a common protocol within each horizon; the reference resolution therefore uses a fresh result from this experiment rather than the full-model result in Table 6. For $H = 96$, the learning rate is 5×10^{-4} and the training batch size is 32; for $H = 336$, they are 10^{-4} and 128, respectively. We use MAE loss, at most 100 training epochs, and early stopping after five epochs without improvement in validation MAE.

Table 8 reports test accuracy and retrieval costs. Library storage measures the distinct backing bytes of retained retrieval tensors, excluding model weights and cached forecasts, and is converted to MiB using 2^{20} bytes per MiB. Build time covers retrieval-library construction. Retrieval profiling uses 32 validation queries, two warmup rounds, and five timed repetitions with CUDA synchronization. Mean latency uses batch size 1, while throughput uses batch size 32; both cover retrieval and continuation aggregation and exclude forecasting, input transfer, and library construction. Each resolution has one training run and one profiling run; the timed repetitions do not represent independent training seeds.

Accuracy and storage. Forecasting accuracy changes little across the tested resolutions: all five configurations have identical MSE and MAE when rounded to three decimal places at each horizon. Relative to 128×128, a 32×32 grid reduces total library storage by approximately 93.4% at both horizons, with relative changes below 0.01% in either error metric. A 16×16 grid reduces storage by approximately 98.1%, with the largest relative MSE increase being 0.077% at $H = 336$. The visual cache scales quadratically with the grid side length, whereas the nonvisual tensor storage remains fixed. These results support using a coarse raster to reduce storage on ETTh1 under this protocol; increasing resolution does not consistently improve accuracy.

Retrieval cost. Mean query latency ranges from 181.66 to 186.92 ms at $H = 96$ and from 179.16 to 181.37 ms at $H = 336$. These differences are smaller than the recorded within-run latency standard deviations (19.27–26.59 ms), so the measurements do not establish a clear reduction in single-query latency with a coarser grid. Batch throughput varies nonmonotonically, although 256×256 has the lowest throughput at both horizons. Its build time is also approximately 74–75% higher than that of 128×128. Thus, the clearest benefit of reduced resolution is lower library storage,

Table 8: Raster-resolution ablation on ETTh1 with fixed $B = 24$, $J = 2$, $M = 100$, and $K = 20$. G denotes a $G \times G$ raster grid. Accuracy is reported to six decimal places to expose small differences. Storage refers to in-memory retrieval tensors; latency is measured per query, and throughput uses batch size 32.

H	G	MSE	MAE	Library (MiB)	Build (s)	Mean (ms)	Throughput (q/s)
96	16	0.363057	0.389366	2.03	31.25	181.66	166.094
	32	0.363036	0.389401	7.05	31.26	183.20	196.108
	64	0.363085	0.389411	27.11	32.29	184.94	167.315
	128	0.363041	0.389384	107.35	32.74	186.92	142.914
	256	0.363074	0.389418	428.29	57.44	183.77	94.908
336	16	0.436964	0.450392	1.98	30.51	180.26	169.496
	32	0.436601	0.450245	6.84	30.66	180.27	170.070
	64	0.436646	0.450268	26.29	30.59	179.16	181.170
	128	0.436629	0.450264	104.06	31.72	181.37	161.548
	256	0.436640	0.450269	415.17	55.23	181.20	99.358

while the finest grid incurs additional construction and batch-processing costs without a consistent accuracy gain. The accuracy findings are limited to the single-seed ETTh1 experiment reported here.

B.4 Comparison with TS-RAG

Table 9 shows that ReViR achieves lower MSE than TS-RAG on five datasets, ties on ETTh2 at the reported precision, and trails on Electricity. Its MSE decreases by 3.7% on ETTh1, 4.5% on ETTm1, 6.1% on ETTm2, 6.5% on Exchange, and 2.8% on Weather relative to TS-RAG; on Electricity, it is 8.9% higher.

Table 9: MSE comparison with TS-RAG across seven datasets. Bold denotes the best result for each dataset; the comparison protocol is described below.

Model	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Weather	Electricity
ReViR	**0.343**	**0.245**	**0.277**	**0.138**	**0.058**	**0.141**	0.122
TS-RAG	0.356	**0.245**	0.290	0.147	0.062	0.145	**0.112**

Comparison protocol. To match the setting reported in the original TS-RAG study, we evaluate ReViR with an input length of 512 and a prediction horizon of 64 rather than the 720-step input used in our main experiments. The TS-RAG results in Table 9 are taken from the original paper; its Chronos-Bolt backbone is frozen without target-dataset fine-tuning. Both methods construct their retrieval libraries from the training split of each target dataset. Because their forecasting backbones and adaptation protocols differ, this comparison is supplementary rather than a fully controlled reproduction.

C Theoretical Details

C.1 Detailed Similarity Scores

The recall stage combines three scores: global correlation, recent-weighted numerical similarity, and time-aware similarity. All numerical comparisons use the standardized, endpoint-aligned histories $\widetilde{Q}, \widetilde{X}^{(i)} \in \mathbb{R}^{L \times C}$ defined in Appendix A.3. Each score produces one scalar per query-candidate pair, yielding a common retrieval ranking across all channels.

C.1.1 Correlation Score

Global multivariate correlation. Flatten each aligned history jointly over time and channels into $q = \mathrm{vec}(\widetilde{Q})$ and $x^{(i)} = \mathrm{vec}(\widetilde{X}^{(i)})$ in $\mathbb{R}^{LC}$. Let $\bar{q}$ and $\bar{x}^{(i)}$ be their scalar means, and define the

centered vectors $q^\circ = q - \bar{q}\mathbf{1}$ and $x_\circ^{(i)} = x^{(i)} - \bar{x}^{(i)}\mathbf{1}$. The implementation computes the dot product of their normalized versions:

$$\rho^{(i)} = \left\langle \frac{q^\circ}{\max(\|q^\circ\|_2, \epsilon_{\text{corr}})}, \frac{x_\circ^{(i)}}{\max(\|x_\circ^{(i)}\|_2, \epsilon_{\text{corr}})} \right\rangle, \tag{11}$$

where ϵ_{corr} is the positive numerical floor used by vector normalization. Away from the numerical floor, this is Pearson correlation over all LC entries. It is not an average of independently computed channel-wise correlations. If either centered vector is identically zero, its normalized vector is zero and $\rho^{(i)} = 0$.

Mapping to a similarity score. To place correlation on the same numerical range as the other scores, we use

$$S_{\text{corr}}^{(i)} = \frac{\text{clip}(\rho^{(i)}, -1, 1) + 1}{2}. \tag{12}$$

Perfect positive and negative correlations map to 1 and 0, respectively, while zero correlation maps to 0.5. Correlation describes global co-variation but does not directly penalize differences in the magnitude of relative changes, motivating the numerical distance below.

C.1.2 Time-aware Similarity Score

The time-aware score compares calendar positions rather than elapsed time between windows. It is shared across channels and uses only the timestamps of the observed histories. We describe the exact calculation used by the retrieval implementation below.

Timestamp extraction. For each length-L window, we use the timestamp at position $t_\star = \lfloor L/2 \rfloor + 1$ under one-based indexing. The calendar marker contains year, month, day of month, weekday, and hour, with an optional sixth minute field. Let m_q, w_q, h_q, u_q denote the query's month, weekday, hour, and optional minute marker; the corresponding candidate fields carry superscript (i). Weekdays are indexed from 0 (Monday) to 6 (Sunday). Year and day of month are not directly scored. The implementation expects these calendar fields, rather than normalized time-feature embeddings.

Cyclic and weekly agreement. For two positions a, b in a cycle of length κ, define

$$d_\kappa(a, b) = \min\big(|a - b|, \kappa - |a - b|\big). \tag{13}$$

This treats, for example, hours 23 and 0 as one hour apart. Hour and month similarities use exponential decay, while weekly agreement rewards both an exact weekday match and a shared weekday/weekend category:

$$\begin{aligned} k_h^{(i)} &= \exp\big(-d_{24}(h_q, h^{(i)})/\tau_h\big), \\ k_m^{(i)} &= \exp\big(-d_{12}(m_q, m^{(i)})/\tau_m\big), \\ k_w^{(i)} &= 0.2\,\mathbf{1}[w_q = w^{(i)}] + 0.1\,\mathbf{1}[\mathbf{1}[w_q \geq 5] = \mathbf{1}[w^{(i)} \geq 5]]. \end{aligned} \tag{14}$$

Thus, the weekly component equals 0.3 for the same weekday, 0.1 for different days in the same weekday/weekend category, and 0 otherwise.

Frequency-dependent weights. Let D be the configured number of observations per day: $D = 24$ for hourly data, 96 for 15-minute data, 144 for 10-minute data, and 1 for daily data. The hour and month weights and decay scales are

$$(\alpha_h, \tau_h, \alpha_m, \tau_m) = \begin{cases} (0.6, 2, 0.2, 3), & L < D, \\ (0.3, 4, 0.4, 2), & L \geq D. \end{cases} \tag{15}$$

Shorter-than-daily histories therefore receive a stronger, narrower hour-matching term. For windows spanning at least one day, the month term receives more weight. When a sixth marker is present, a minute term is enabled only if $L < \lfloor D/4 \rfloor$:

$$k_u^{(i)} = \begin{cases} 0.1 \exp\big(-d_{60}(u_q, u^{(i)})/15\big), & \text{minute marker available and } L < \lfloor D/4 \rfloor, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

An unavailable minute term is defined as zero. In the current ETT-minute loader, the sixth field stores the quarter-hour index $\lfloor \mathrm{minute}/15 \rfloor$, and the scoring routine consumes that field directly. This distinction has no effect on the reported $L = 720$ experiments: $L \geq D$ for all evaluated frequencies, so the minute term is inactive. Their unnormalized score is therefore

$$b_q^{(i)} = 0.3 \exp\bigl(-d_{24}(h_q, h^{(i)})/4\bigr) + k_w^{(i)} + 0.4 \exp\bigl(-d_{12}(m_q, m^{(i)})/2\bigr). \tag{17}$$

More generally, $b_q^{(i)} = \alpha_h k_h^{(i)} + k_w^{(i)} + \alpha_m k_m^{(i)} + k_u^{(i)}$. The hour, weekday, and month terms are computed when at least five calendar fields are available; otherwise, these terms are zero.

Normalization. For the current query batch $\mathcal{Q}_{\mathrm{batch}}$ and the complete training retrieval library of N candidates, the implementation computes

$$S_{\mathrm{time}}^{(i)} = \frac{b_q^{(i)}}{\max\left(10^{-8}, \max_{q' \in \mathcal{Q}_{\mathrm{batch}},\, 1 \leq k \leq N} b_{q'}^{(k)}\right)}. \tag{18}$$

This normalization yields scores in $[0, 1]$ and leaves an all-zero score matrix unchanged. The denominator is shared by the entire query-candidate matrix, rather than computed separately per query or candidate chunk. It is calculated before candidate eligibility masking; consequently, the normalized scale can depend on the query batch.

C.1.3 Recent-weighted Numerical Similarity Score

Backward temporal partition. Given a patch length ℓ, define $P = \lceil L/\ell \rceil$. Number patches from oldest to newest and set

$$s_p = \max\bigl(1, L - (P - p + 1)\ell + 1\bigr), \qquad e_p = L - (P - p)\ell, \qquad L_p = e_p - s_p + 1. \tag{19}$$

Patch p contains observations $s_p, \ldots, e_p$. Partitioning backward ensures that recent patches contain exactly ℓ observations; only the oldest patch can be shorter. A positive configured patch length takes precedence. Otherwise, the implementation uses 24 steps for hourly data and 96 for 15-minute data. For daily data, it uses the rounded median number of training observations per observed calendar week when available, with fallback lengths of seven for daily and five for business-day data. Other frequencies require an explicit patch length.

Patch distance and recency weighting. For each patch, compute the root mean squared difference jointly over its time steps and channels:

$$d_p^{(i)} = \sqrt{\frac{1}{L_p C} \sum_{t=s_p}^{e_p} \sum_{c=1}^{C} (\widetilde{Q}_{t,c} - \widetilde{X}_{t,c}^{(i)})^2}, \qquad \omega_p = \frac{2p}{P(P+1)}. \tag{20}$$

The weights sum to one and increase linearly toward the forecasting origin. The resulting distance and similarity are

$$D_{\mathrm{num}}^{(i)} = \sum_{p=1}^{P} \omega_p d_p^{(i)}, \qquad S_{\mathrm{num}}^{(i)} = \exp(-D_{\mathrm{num}}^{(i)}/\tau_{\mathrm{num}}). \tag{21}$$

The square root is taken within each patch before weighted aggregation; this differs from taking a square root after averaging squared patch errors. The score equals 1 for zero distance and decreases toward zero as distance increases. In the batched implementation, squared distances are obtained from vector norms and matrix products, with small negative values caused by floating-point cancellation clamped to zero before taking the square root.

Training-only temperature calibration. The scale τ_{num} is fitted once from training-window pairs, using the same endpoint alignment, patches, and weights as retrieval. The default sampler requests 4,096 pairs with random seed 2021 and retains pairs whose start indices differ by at least $L + H$, so their input-continuation intervals do not overlap when the library permits this separation. Precisely, for N training windows, the implementation uses separation $\min(L+H, N-1)$; a smaller

library therefore relaxes the non-overlap condition. Sampling can retain repeated pairs and return fewer pairs under its finite attempt budget.

Let $\mathcal{T}_+$ contain the finite, strictly positive sampled distances. Then

$$\tau_{\text{num}} = \begin{cases} \max\big(\text{median}_{\text{lower}}(\mathcal{T}_+), 10^{-6}\big), & \mathcal{T}_+ \neq \varnothing, \\ 10^{-6}, & \mathcal{T}_+ = \varnothing. \end{cases} \tag{22}$$

Here, $\text{median}_{\text{lower}}$ selects the lower middle element when the number of distances is even, matching the implementation. Validation and test data do not enter this calibration.

C.1.4 Combined Recall Score

The three scores describe global co-variation, recent numerical agreement, and calendar compatibility, respectively. We clip each score to $[0, 1]$ for numerical safety and combine them with equal weights:

$$S^{(i)}_{\text{recall}} = \frac{S^{(i)}_{\text{corr}} + S^{(i)}_{\text{num}} + S^{(i)}_{\text{time}}}{3}. \tag{23}$$

After applying the candidate eligibility mask, we retain the top-M candidates for visual comparison. Thus, this combined score defines the recall pool; Approximate Triangle-IoU subsequently contributes to the final ranking within that pool.

C.2 Visual Sampling Procedure

Our Visual Sampling procedure is designed based on iterative largest-triangle sampling (ILTS) (Rui et al., 2025). We apply it independently to each channel of the endpoint-aligned query $\widetilde{Q}$ and recalled candidate $\widetilde{X}^{(i)}$. The following definitions use the query's time-value points $\widetilde{q}_c(t) = (t, \widetilde{Q}_{t,c})$ for $t \in \{1, \ldots, L\}$; replacing them with $\widetilde{m}_c(t) = (t, \widetilde{X}^{(i)}_{t,c})$ gives the identical candidate-side procedure for $m = X^{(i)}$. The implementation uses zero-based time coordinates, which differ from the notation here by a common shift of one and therefore do not change triangle areas or selected indices.

Temporal Bucketing. As in the main text, the first $L-1$ time indices form B contiguous buckets $\{\mathcal{B}_1, \ldots, \mathcal{B}_B\}$, while $\widetilde{q}_c(L)$ is reserved as the terminal anchor. The admissible time indices in each bucket are specified in the selection formula below. Bucket sizes differ by at most one, and the first observation belongs to $\mathcal{B}_1$ as well as serving as its initial left reference. The right reference of a nonterminal bucket initially uses the two-dimensional mean point of the next bucket, defined by

$$\mu_b = \frac{1}{|\mathcal{B}_b|} \sum_{t \in \mathcal{B}_b} \widetilde{q}_c(t). \tag{24}$$

The channel index on μ_b and the references below is suppressed for readability.

Visual Sampling. For each iteration $j \in \{1, \ldots, J\}$, process buckets from left to right, as in the main text. Its left and right references are computed as

$$u^{(j)}_b = \begin{cases} \widetilde{q}_c(1), & b = 1, \\ p^{(j)}_{b-1}, & b > 1, \end{cases} \qquad v^{(j)}_b = \begin{cases} \widetilde{q}_c(L), & b = B, \\ \mu_{b+1}, & b < B,\ j = 1, \\ p^{(j-1)}_{b+1}, & b < B,\ j > 1. \end{cases} \tag{25}$$

Thus, $u^{(j)}_b$ uses the preceding bucket's selection from the current iteration, while $v^{(j)}_b$ uses the next bucket's mean initially and its selection from the preceding iteration thereafter. These dependencies require a sequential bucket sweep. We then use the same selection and triangle-area formulas as in the main text:

$$t^{(j)}_b = \arg\max \Delta\big(u^{(j)}_b, \widetilde{q}_c(t), v^{(j)}_b\big), \quad t \in \left\{ \left\lceil \frac{(b-1)(L-1)}{B} \right\rceil + 1, \ldots, \left\lceil \frac{b(L-1)}{B} \right\rceil \right\}, \tag{26}$$

$$p^{(j)}_b = \widetilde{q}_c\big(t^{(j)}_b\big), \quad \Delta(u, p, v) = \frac{1}{2} \left| (v_x - u_x)(p_y - u_y) - (v_y - u_y)(p_x - u_x) \right|. \tag{27}$$

The range in Eq. equation 26 is exactly $\mathcal{B}_b$. Ties are resolved by the earliest observation, as stated in the main text. For a fixed nonvertical reference line, triangle area is proportional to the trial point's absolute vertical deviation from that line.

Region Rendering. After J iterations, the sampled sequence contains the first observation, one selected point per bucket, and the terminal observation; the first point can occur twice if selected from $\mathcal{B}_1$. Region Rendering retains the triangles constructed during the final sweep, exactly as in the main text:

$$\mathcal{A}(\widetilde{q}_c) = \bigcup_{b=1}^{B} \mathrm{conv}(u_b^{(J)}, p_b^{(J)}, v_b^{(J)}). \tag{28}$$

For $J > 1$, the right reference of a final-pass triangle still comes from the next bucket's selection at iteration $J-1$; for $J = 1$, it remains that bucket's mean. It is not replaced by the next bucket's final selection afterward. Sampling uses a fixed iteration budget rather than an early-stopping test, and local area maximization does not imply a globally optimal representation.

Batched implementation and cost. Windows and channels are flattened into a batch of one-dimensional trajectories, and the trial points within each bucket are scored in parallel. Unequal bucket lengths are padded to a common width; padding is excluded from bucket means and assigned score $-\infty$ before maximization. Bucket sweeps and sampling iterations remain sequential, using 32-bit floating-point values. Each pass evaluates $L-1$ valid trial points per trajectory, so sampling costs $O(JL)$ per channel and $O(JLC)$ per multivariate window, excluding rasterization. Only the final-pass triangles are retained for visual comparison.

C.3 Joint Retrieval and Forecast Aggregation

For each candidate $X^{(i)} \in \mathcal{M}(Q)$ in the numerical-temporal recall pool, we combine its recall score and visual similarity with equal weights, then select the indices of the top K candidates:

$$S_{\mathrm{rank}}^{(i)} = \frac{1}{2} S_{\mathrm{recall}}^{(i)} + \frac{1}{2} S_{\mathrm{IoU}}^{(i)}, \qquad \mathcal{K}(Q) = \mathrm{TopK}_{i:\, X^{(i)} \in \mathcal{M}(Q)} S_{\mathrm{rank}}^{(i)}. \tag{29}$$

The selected endpoint-aligned continuations $\widetilde{Y}^{(i)}$ are aggregated using normalized Gaussian-kernel weights with bandwidth $\sigma > 0$:

$$w^{(i)} = \frac{\exp\left[-\frac{(1-S_{\mathrm{rank}}^{(i)})^2}{2\sigma^2}\right]}{\sum_{j \in \mathcal{K}(Q)} \exp\left[-\frac{(1-S_{\mathrm{rank}}^{(j)})^2}{2\sigma^2}\right]}, \qquad \widehat{\widetilde{Y}}_{\mathrm{ret}} = \sum_{i \in \mathcal{K}(Q)} w^{(i)} \widetilde{Y}^{(i)}. \tag{30}$$

We compute the normalized weights with a numerically stable softmax.

C.4 Computational Cost

Let $G = G_x G_y$ denote the number of grid cells per channel. Query-side Visual Sampling and rasterization cost $O(CJL + CBG)$, and comparing the packed masks of the M recalled candidates costs $O(MCG)$ bit-level operations, followed by top-K selection and $O(KHC)$ continuation gathering. The historical masks require approximately $NCG/8$ bytes, excluding metadata. Halving both grid dimensions quarters mask storage and comparison work, while reducing B lowers rasterization cost. Numerical-temporal recall still requires $O(NLC)$ operations plus top-M selection; visual comparison scales with M rather than the full library size N.