HiRAF: Hidden-State Retrieval-Augmented Forecasting for Time Series Foundation Models

Anonymous authors
Paper under double-blind review

Abstract

Time series foundation models (TSFMs) deliver high-quality zero-shot forecasts on a target dataset. Unlike a conventional dataset-specific model, however, a TSFM can use only the part of the target series' history that fits in its input window, while the history outside the window is rich in periodic structure and statistical regularities. Retrieval-augmented forecasting can exploit that history without training the model, but existing methods compute similarity on raw series or with a separate embedding model: they measure how closely a historical segment's context matches in shape, whereas what enters the forecast is the segment's continuation, so the quantity the key measures is misaligned with the quantity the forecast needs. We argue that a retrieval key should measure predictive similarity, and that the hidden state of a frozen TSFM has exactly this property: used as the key, it retrieves continuations whose error is 14% lower than with the raw series and 17% lower than with a separate embedding model. Building on this, we propose HiRAF (Hidden-state Retrieval-Augmented Forecasting), which reads the hidden state from the forward pass that produces the forecast, retrieves the most similar segments from the target series' history with it, and uses their continuations to correct the backbone's forecast. No model is trained; the key is a by-product of the forward pass that produces the forecast, so no separate embedding model is needed, and with a bucketed index each retrieval scans only a small fraction of the history. On six benchmark datasets HiRAF achieves better zero-shot accuracy than existing TSFMs and 7% lower MSE than the retrieval-augmented method TS-RAG; with an index of five million entries, its prediction time is $1.05\times$ that of the backbone alone, against $8.17\times$ for TS-RAG.

1 Introduction

Time series forecasting has important applications across many domains, including energy (Hong & Fan, 2016), weather (Lam et al., 2023), traffic (Li et al., 2018) and finance (Sezer et al., 2020). In recent years the field has shifted from dataset-specific models to time series foundation models (TSFMs). A dataset-specific model is trained on a single dataset and serves only that dataset (Zhou et al., 2021; Wu et al., 2021; Nie et al., 2023; Zeng et al., 2023); a TSFM is pretrained on large-scale, general-purpose time series data and then forecasts each target dataset zero-shot, without retraining on it (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024; Liu et al., 2024b; 2026).

TSFMs have been widely adopted for their strong zero-shot accuracy, but the shift also introduces a new problem. Unlike a dataset-specific model, which is trained on the dataset it forecasts, a TSFM can use only the history contained in its input context window, and that window is usually a small fraction of the history the dataset holds: on Weather, a 512-step context covers 1.4% of the training split. The history outside the window is rich in periodic structure, recurring patterns and statistical regularities. A dataset-specific model learns this information into its parameters during training; a TSFM cannot reach it at all (Fig. 1). The missing information is one reason a TSFM's forecast sometimes deviates substantially from the true series.

Several direct remedies exist. One is to enlarge the context window so that the model sees more history. This costs efficiency, admits redundant information, and reaches its limit quickly: Ahuja

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.

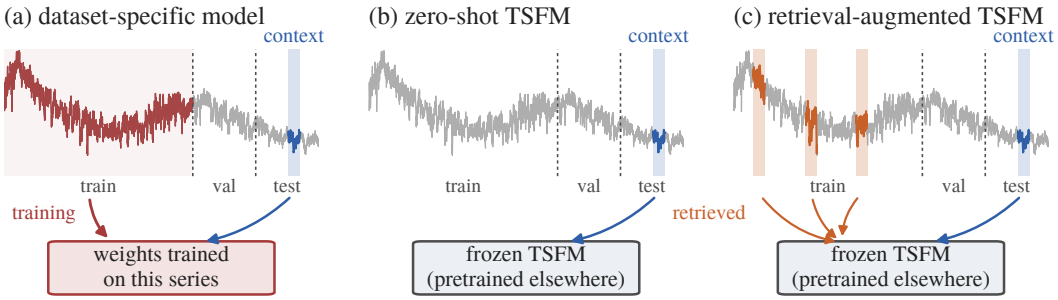


Figure 1: (a) A dataset-specific model absorbs the target series' history into its weights through training. (b) A zero-shot TSFM sees only the context window. (c) A retrieval-augmented TSFM reopens the history at inference.

et al. (2026) report that the error of a strong forecaster rises by 68% as its window grows from 720 to 3,000 steps, because attention does not screen out distant, irrelevant fluctuations. A second is to fine-tune the model on the target dataset. This requires a training run on every dataset, forfeits the zero-shot advantage that makes a TSFM attractive, places a demand on the amount of in-domain data, and yields gains that vary with the model, the dataset and the horizon (Laglil et al., 2026; Rahimikia et al., 2025). A third is to attach a memory module trained on the target dataset (Lyu et al., 2026). This also requires training; moreover, compressing the history into the module's parameters is lossy, and at prediction time one cannot tell which parts of the history the model has used.

In natural language processing, large language models face an analogous problem, and retrieval-augmented generation (RAG) is an established solution (Lewis et al., 2020; Khandelwal et al., 2020; Borgeaud et al., 2022). The same idea applies to forecasting: retrieve from the target series' own history the segments that resemble the input, take their continuations, and use them as a reference for the forecast. Several works follow this route, scoring candidates by a distance on the raw series or by an embedding from a separate time series encoder (Tire et al., 2024; Han et al., 2025; Ning et al., 2025). Retrieval for forecasting, however, differs from RAG in one essential respect. In RAG the retrieved object is the object that will be used: the similarity between a query q and a document d measures how useful d is. A forecasting database instead stores history segments together with their continuations, (x_r, y_r); similarity can be computed only between the query and x_r, whereas what enters the forecast is y_r. What retrieval needs is that y_r resemble the query's future, and this is not always aligned with how much x_r resembles the query. A frequency component of small amplitude, for instance, may foreshadow a sharp change in the series: it barely affects a similarity computed against x_r, yet it strongly affects the future (Fig. 2). Computing similarity on raw series or on their embeddings weights every position and component by its amplitude, and therefore cannot give more weight to the features that matter most for the forecast.

The hidden state of a TSFM is different: it carries the information the model has extracted in order to forecast. First, the hidden state is the model's processed representation of the input, so a feature of small amplitude that matters for the forecast is extracted and retained rather than drowned out by amplitude. Second, as depth increases, what the hidden state describes shifts from the input series itself toward the forecast the model is about to produce; the closer to the output layer, the more two similar hidden states imply two similar futures. The hidden state of a TSFM can therefore serve directly as the retrieval key. Compared with a dedicated embedding model, this has a further advantage: the hidden state is produced by the forward pass that makes the forecast, so no separate embedding model has to be run. Retrieving with a model's own hidden states has precedent in natural language processing, where the internal representations of a language model have been found to be effective retrieval representations in their own right (Skean et al., 2025; Jiang, 2026).

Building on this, we propose HiRAF (Hidden-state Retrieval-Augmented Forecasting). While the backbone model forecasts the input series, HiRAF takes its hidden state as the retrieval key, compares it against an index built in advance over the target series' history, selects the most similar historical segments, and uses their continuations to correct the backbone's forecast. The procedure trains no model and introduces no second model beside the backbone. To keep retrieval cost from growing with the length of the history, the hidden states in the index are grouped into buckets by nearest centroid, and a query scans only a few of them. On six benchmark datasets HiRAF outperforms a broad set

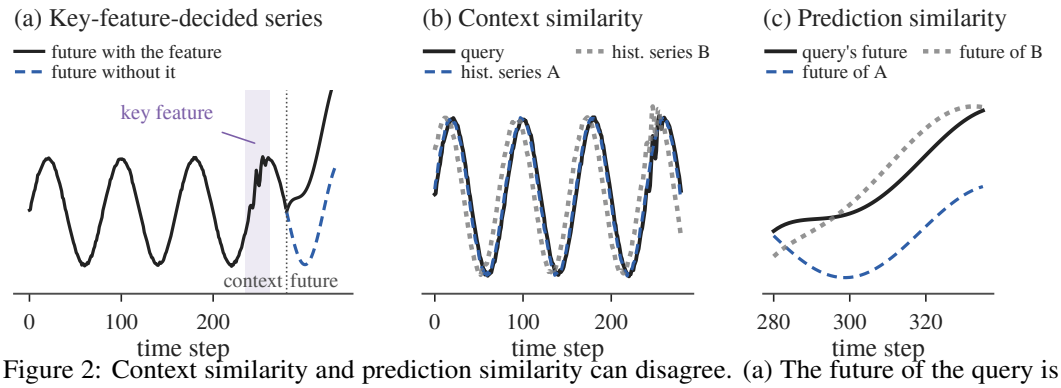


Figure 2: Context similarity and prediction similarity can disagree. (a) The future of the query is decided by a small feature. (b) Historical series A is more similar to the query in context, (c) whereas historical series B, which carries the feature, is more similar to the query in future.

of TSFMs and reduces MSE by 7% relative to TS-RAG, a retrieval-augmented forecasting method. Owing to bucketed retrieval and to the absence of a separate embedding model, on an index of five million entries its prediction time is $1.05\times$ that of the backbone alone, against $8.17\times$ for TS-RAG.

Our contributions are as follows.

- We identify a mismatch specific to retrieval for forecasting: the key measures similarity on a segment's context, while what enters the forecast is its continuation. A retrieval key should therefore measure predictive similarity rather than shape similarity.
- We show that the hidden state of a frozen TSFM is a better retrieval key than the raw series or a separate embedding model: on its own, it retrieves continuations with 14–17% lower error. Building on this, we propose HiRAF, which requires no model training and uses the hidden state directly as the retrieval key. In its self-retrieval configuration one model is both retriever and forecaster, the key is a by-product of the forward pass that produces the forecast, and no embedding model is needed; with a bucketed index each retrieval scans only a small fraction of the history. The result is retrieval-augmented forecasting that is both efficient and effective.
- Experiments show that HiRAF attains better zero-shot accuracy than existing TSFMs on all six benchmark datasets, with 7% lower MSE than TS-RAG, the strongest retrieval-augmented method; with an index of five million entries it adds only 5% to the backbone's prediction time, where TS-RAG needs more than $8\times$. Experiments further show that the gain comes from the hidden-state key itself.

2 Related Work

Time series foundation models and the target series' history. Long-horizon time series forecasting was for a long time a supervised problem solved one dataset at a time. The architectures that defined it—efficient attention (Zhou et al., 2021), series decomposition (Wu et al., 2021), patching (Nie et al., 2023), inverted attention (Liu et al., 2024a) and multi-periodicity modelling (Wu et al., 2023)—were each trained and evaluated on the same dataset, and a simple linear model was shown to be a strong competitor (Zeng et al., 2023). What TSFMs change is not only the model scale and the pretraining data but the way a model is used: a pretrained model is applied to unseen series without any training. Chronos tokenises scaled values and trains a language model over them (Ansari et al., 2024); TimesFM and Timer are decoder-only patch forecasters (Das et al., 2024; Liu et al., 2024b); Moirai handles arbitrary numbers of variates and sampling frequencies (Woo et al., 2024); Timer-XL and Timer-S1 extend context length and model scale (Liu et al., 2025; 2026). What this shift gives up is the use of the target series' own history, which a dataset-specific model learns into its parameters. The usual ways of recovering it reintroduce training: fine-tuning needs in-domain data and yields gains that vary with the model, the dataset and the horizon (Laglil et al., 2026; Rahimikia et al., 2025), and a trained memory module (Lyu et al., 2026) compresses the history into a fixed store before any query is seen. Without retrieval this private history goes unused; with retrieval, the part of it relevant to the current query can be brought back without training and without touching the weights.

Retrieval-augmented forecasting. Methods in this line share one construction: retrieve from the target series' own history the segments that resemble the query, and let their continuations take part in the forecast. RAF scores candidates by Euclidean distance between raw patches and prepends the retrieved segments to the input of Chronos, leaving the combination to in-context learning (Tire et al., 2024). RAFT matches shapes after subtracting the last value of each patch and repeats the same module at several downsampling rates to cover multiple periods (Han et al., 2025). TS-RAG obtains its key from a second pretrained model and trains an adaptive retrieval mixer to fuse the retrieved horizons with the forecast (Ning et al., 2025). TS-Memory replaces the retrieval database itself with a trained adapter distilled from nearest neighbours (Lyu et al., 2026). The remaining work differs mainly in how the retrieved segments are fused: through cross-attention in the feature space (Wang & Cui, 2024; Lee et al., 2026), as prompts injected into the backbone (Zhang et al., 2025), or rewritten as text for a large language model (Yang et al., 2025). In the choice of the retrieval key, however, these methods agree: all of them measure the similarity between the query and a historical segment's context—as a distance on the raw series or as the output of an embedding model trained for a different objective—while what enters the forecast is the segment's continuation. None of them designs the key around the question retrieval actually has to answer, namely whether the candidate's future resembles the query's future.

3 METHOD

3.1 PROBLEM FORMULATION

We consider univariate forecasting applied channel by channel. A channel is a series $s \in \mathbb{R}^T$, split contiguously into training, validation and test spans and standardised with statistics of the training span. A window is a pair (x, y) of a context $x \in \mathbb{R}^L$ and a future $y \in \mathbb{R}^H$. The retrieval database

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \tag{1}$$

consists of all stride-one windows of the training span, with both context and future inside that span, so that no information after the forecast time enters the database. A query window is written (x_q, y_q), and the subscript r is reserved for a retrieved entry. The backbone is a frozen TSFM m that maps a context to a point forecast $m(x) \in \mathbb{R}^H$; neither m nor the model from which the retrieval key is read receives any gradient update.

Retrieval-augmented forecasting has the general form

$$\hat{y}_q = F\Big(m(x_q),\ \{(x_r, y_r)\}_{r \in \mathcal{N}_k(q)}\Big), \qquad \mathcal{N}_k(q) = \operatorname*{top\text{-}k}_i\ \mathrm{sim}\big(\varphi(x_q), \varphi(x_i)\big), \tag{2}$$

where the retrieval key $\varphi : \mathbb{R}^L \to \mathbb{R}^d$ maps a context to a d-dimensional vector by which candidates are scored, and F combines the k retrieved windows with the backbone's forecast. Existing methods take φ to be a distance on the raw series (Tire et al., 2024; Han et al., 2025) or the output of a separately trained embedding model (Ning et al., 2025). The retrieval set that minimises the forecast error is

$$\mathcal{N}_k^\star(q) = \operatorname*{arg\,min}_{|S|=k} \sum_{r \in S} \|y_r - y_q\|^2, \tag{3}$$

which cannot be computed at prediction time because y_q has not been observed. Any retrieval key is therefore a surrogate for Eq. (3) built from x alone, and what is asked of it is not to preserve the information in the context but to make proximity under φ imply proximity of futures.

3.2 HIDDEN-STATE RETRIEVAL KEY

We take the retrieval key to be the backbone's own representation. Writing $h_\ell(x) \in \mathbb{R}^d$ for the hidden state of layer ℓ at the last context position,

$$\varphi(x) = h_\ell(x) \,/\, \|h_\ell(x)\|_2, \qquad \mathrm{sim}\big(\varphi(x_q), \varphi(x_i)\big) = \varphi(x_q)^\top \varphi(x_i), \tag{4}$$

so that candidates are ranked by the cosine similarity between last-position hidden states. The context is instance-normalised before the key is computed, which keeps it in the same frame as the aggregation of Section 3.3. Equation (4) does not by itself fix the layer ℓ; following experience and the analysis in Section 5, we use the last layer.

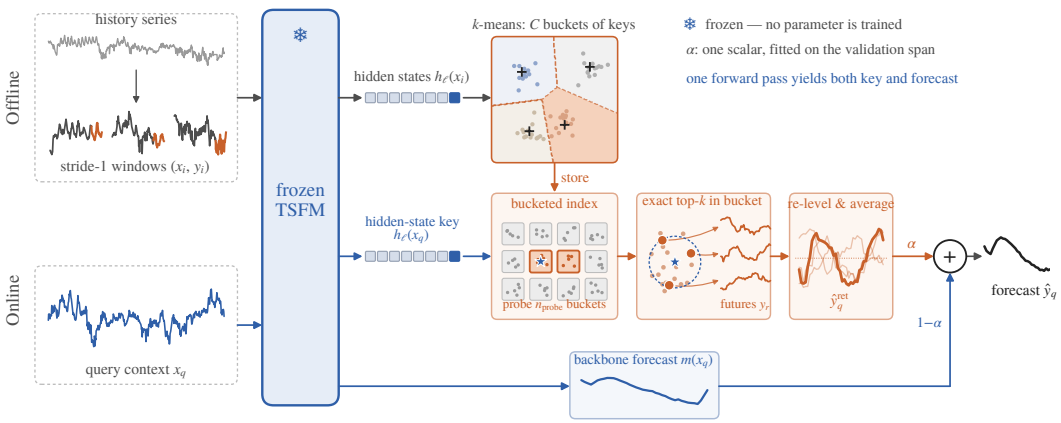


Figure 3: Overview of HiRAF. Offline: every stride-1 window of the training span is passed through the frozen TSFM, the hidden state at the last context position is taken as the key, and the keys are partitioned into buckets by k-means, each stored with its future. Online: one forward pass of the same frozen model yields both the backbone forecast $m(x_q)$ and the key; the key probes the nearest buckets, the top-k neighbours are found inside them, their futures are re-levelled and averaged (Eq. (5)), and the result is blended with the backbone forecast by a single scalar α (Eq. (6)).

3.3 Aggregation and fusion

The retrieved windows resemble the query in shape but sit at their own level and scale, so averaging their futures directly would mix a shape estimate with level noise. Let μ_r and σ_r be the mean and standard deviation of the retrieved context x_r, and μ_q, σ_q those of the query context. We average in the re-levelled frame and restore the query's statistics:

$$\hat{y}_q^{\text{ret}} = \mu_q + \sigma_q \cdot \frac{1}{k} \sum_{r \in \mathcal{N}_k(q)} \frac{y_r - \mu_r}{\sigma_r}. \tag{5}$$

RAFT subtracts the last value of each patch for the same purpose (Han et al., 2025); the statistic differs, the intent does not. Equation (5) uses the same statistics under which the key is computed, so retrieval and aggregation operate in one frame. The retrieved forecast is then combined with the backbone forecast by a single convex weight,

$$\hat{y}_q = \alpha\, \hat{y}_q^{\text{ret}} + (1 - \alpha)\, m(x_q), \tag{6}$$

where α is the least-squares solution on the validation span, and the test span is evaluated once at that value. TS-RAG fuses with an adaptive retrieval mixer that has to be trained (Ning et al., 2025); we use a single scalar so that the improvement in accuracy comes, as far as possible, from retrieval itself. Appendix E reports similarity-weighted, trimmed-mean and single-neighbour variants of Eq. (5).

3.4 Self-retrieval with a bucketed index

When φ is read from the backbone, the whole system consists of one model. The hidden state the key needs is produced by the forward pass that produces the forecast, so a query costs nothing beyond that forecast; the database is embedded once, with N forward passes whose results are cached and reused across horizons and across settings of k.

Exact retrieval is then one product with an $N \times d$ matrix. This is affordable for a single channel, but it grows linearly with the length of the history and with any pooling of channels into one database (Section 5). We therefore bucket the index: the N normalised keys are partitioned by k-means into C buckets, and each key is stored, together with its future y_i, in the bucket of its nearest centroid. A query is first compared with the C centroids, and the n_{probe} nearest buckets are scanned exactly under Eq. (4). The cost per query falls from $O(Nd)$ to $O(Cd + n_{\text{probe}}Nd/C)$, about $O(\sqrt{N}\,d)$ at $C \approx \sqrt{N}$. This is a standard inverted-file index (Johnson et al., 2021); we claim no novelty in the index itself and note only that it is directly compatible with hidden-state keys, since cosine similarity is an inner product of normalised vectors and k-means partitions the same space. Its effect

Algorithm 1 HiRAF in the self-retrieval configuration

Require: frozen TSFM m with hidden states h_ℓ; training-span history s; $L, H, k, C, n_{\text{probe}}$

Offline, once per series

$\mathcal{D} \leftarrow$ all stride-1 windows (x_i, y_i) of the training span, contexts instance-normalised

$\varphi_i \leftarrow h_\ell(x_i)/\|h_\ell(x_i)\|$ for every i ▷ one forward pass per window, cached

$\{c_1, \dots, c_C\} \leftarrow k$-means over $\{\varphi_i\}$; assign each $(\varphi_i, y_i, \mu_i, \sigma_i)$ to its nearest bucket

$\alpha \leftarrow$ least squares of Eq. (6) on the validation span

Online, per query x_q

run m once on x_q: obtain $m(x_q)$ and $\varphi_q \leftarrow h_\ell(x_q)/\|h_\ell(x_q)\|$

$B \leftarrow$ the n_{probe} buckets whose centroids are nearest to φ_q

$\mathcal{N}_k(q) \leftarrow$ top-k of $\varphi_q^\top \varphi_i$ over $i \in B$

$\hat{y}_q^{\text{ret}} \leftarrow \mu_q + \sigma_q \cdot \frac{1}{k} \sum_{r \in \mathcal{N}_k(q)} (y_r - \mu_r)/\sigma_r$ ▷ Eq. (5)

return $\alpha\, \hat{y}_q^{\text{ret}} + (1 - \alpha)\, m(x_q)$ ▷ Eq. (6)

on accuracy is measured in Section 5. For horizons beyond H, retrieval is performed once from the L-step context and the neighbours' true continuations are read out to the required length; model output is never fed back into a key.

4 Experiments

4.1 Setup

Protocol. Following TS-RAG (Ning et al., 2025): context length $L = 512$, horizon $H = 64$, channel-independent forecasting, per-channel standardisation with statistics of the training span, and MSE in the standardised space averaged over all test windows.

Datasets. Six benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Weather and Electricity, with the standard 12/4/4-month split for ETT and 7:1:2 for the others. The distributed Weather and Electricity files contain sentinel values (-9999); all results use repaired files, detailed in Appendix B.

Models. The backbone and the retrieval key are each taken from Chronos-Bolt (Ansari et al., 2024) or Timer-S1 (Liu et al., 2026). The key is the hidden state of the last layer at the last context position. Throughout, HiRAF retrieves $k = 50$ neighbours, aggregates them with the unweighted mean of Eq. (5) and probes $n_{\text{probe}} = 2$ buckets; index details are given in Appendix A.

Baselines. Six zero-shot TSFMs—Chronos-Bolt, Timer-S1, Moirai (Woo et al., 2024), TimesFM 1.0 and 2.5 (Das et al., 2024), and Toto 2.0 (Khwaja et al., 2026)—and TS-RAG (Ning et al., 2025), the retrieval-augmented method closest to ours. When a model's documentation lists a dataset in its pretraining corpus, that cell is excluded from ranking and from the mean.

4.2 Results

Main results. Table 1 compares HiRAF with the two foundation models it is built on, four further zero-shot TSFMs, and TS-RAG. HiRAF has the lowest MSE on all six benchmarks, with the largest margin on the most periodic datasets (11% below the best TSFM on ETTm1). Relative to its own backbone, HiRAF lowers mean MSE from .2309 to .2151 on Chronos-Bolt and from .2208 to .2071 on Timer-S1 without any training, and it does so on every dataset. Relative to TS-RAG, which shares its backbone with our Chronos-Bolt configuration, HiRAF lowers mean MSE from .2232 to .2151 with the same backbone and to .2071 with Timer-S1, while dispensing with the second model and the trained mixer that TS-RAG requires: our key comes from the same forward pass as the forecast, and the fusion is a single scalar.

Effect of the retrieval key. The comparison in Table 1 is between complete systems and says nothing about the key by itself. Table 2 isolates it: with the backbone, the library, the aggregation and the fusion held fixed, the raw-series key and the Chronos-T5 embedding used by TS-RAG are

Table 1: Zero-shot forecasting results, $L = 512$, $H = 64$. Best per column in bold, second best underlined; "–" marks a dataset listed in the model's pretraining corpus, excluded from ranking and mean.

Method	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Electricity	Mean
Chronos-Bolt	.3615	.2516	.3107	.1486	.2006	.1126	.2309
Timer-S1	<u>.3242</u>	.2483	.2948	.1609	.1799	.1170	.2208
Moirai	.3763	.2638	.4633	.1840	.2441	.1485	.2800
TimesFM 1.0	.3914	.2775	.3324	.1700	–	–	–
TimesFM 2.5	.3643	.2634	.2893	.1523	.1817	.1211	.2287
Toto 2.0	.3728	.2539	.2963	.1480	.1860	.1227	.2299
TS-RAG	.3556	<u>.2451</u>	.2904	.1465	.1905	<u>.1114</u>	.2232
HiRAF (Chronos-Bolt)	.3450	.2465	<u>.2739</u>	**.1405**	<u>.1743</u>	**.1104**	<u>.2151</u>
HiRAF (Timer-S1)	**.3162**	**.2385**	**.2561**	<u>.1457</u>	**.1715**	.1145	**.2071**

Table 2: Effect of the retrieval key. Timer-S1 backbone in the setting of Table 1, with only the key varied: the raw-series key is the instance-normalised context, and the T5 embedding is the Chronos-T5 encoder embedding used by TS-RAG. Left, the retrieved forecast alone; right, the fused forecast. MSE;

		Retrieved forecast only, Eq. (5)			Fused forecast, Eq. (6)		
Dataset	Backbone	Hidden state	Raw series	T5 embedding	Hidden state	Raw series	T5 embedding
ETTh1	.3242	**.3771**	.4019	.4215	**.3162**	.3194	.3201
ETTh2	.2483	**.2585**	.3409	.2697	**.2385**	.2479	.2413
ETTm1	.2948	**.2797**	.3168	.4083	**.2561**	.2762	.2803
ETTm2	.1609	**.1575**	.1939	.1883	**.1456**	.1523	.1549
Weather	.1799	**.2008**	.2440	.2161	**.1717**	.1736	.1754
Electricity	.1170	**.1445**	.1561	.1963	**.1145**	.1149	.1162
Mean	.2208	**.2364**	.2756	.2834	**.2071**	.2140	.2147

substituted for the hidden state. The retrieved forecast alone is the quantity that reflects the key, since nothing else has entered it, and here the hidden state is clearly the better key: its mean MSE is 14% below that of the raw-series key and 17% below that of the T5 embedding, on every dataset. A retrieved forecast is nonetheless not a forecaster in its own right. Even with the best key it remains behind the backbone, and its value lies in correcting the backbone rather than replacing it. Fusion therefore compresses the differences between keys, but does not remove them: after fusion the hidden state still gives the lowest error on every dataset, reducing the mean MSE of the backbone by 6.2% where the raw-series key reduces it by 3.0% and the T5 embedding by 2.8%.

5 Analysis

Case study. Figure 4 shows three queries on which the raw-series key and the hidden-state key retrieve different histories; more are given in Appendix C. Four failure modes of the raw-series key recur. (i) It weights all positions of the context equally, so it rewards a match over the early and middle part of the window, whereas the future depends most on the part closest to the forecast origin: in (a) and (b) its retrieved histories follow the query closely over most of the window and depart from it at the end. (ii) It is dominated by the segments of largest deviation, and attends to them whether or not they bear on the future: in (a) the match rests on the deep troughs. (iii) When no history matches the query in shape, it falls back on histories that share the query's trend or level, which score well on similarity but carry little information: in (c), where the context is nearly constant, this is what it retrieves. (iv) It recognises signals on a short time scale poorly, such as a narrow peak, since they contribute little to the distance (Appendix C). The hidden-state key matches the query where it matters for the forecast, and in all three cases its retrieved continuations follow the truth.

Depth of the key. The argument of Section 1 rests on one assumption: the deeper the hidden state, the more its proximity means proximity of futures. Figure 5 tests it directly. For each layer we correlate the key's similarity score with two orderings of the candidates, one by the distance between contexts and one by the error of using the candidate's future as the forecast. The two curves

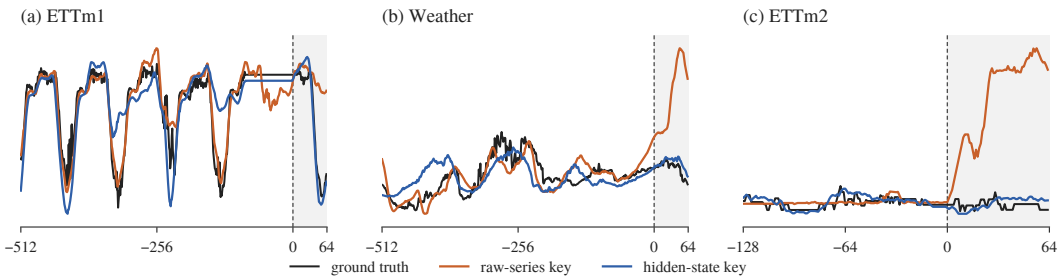


Figure 4: Histories retrieved by the raw-series key and by the hidden-state key. (c) shows the last 128 context steps.

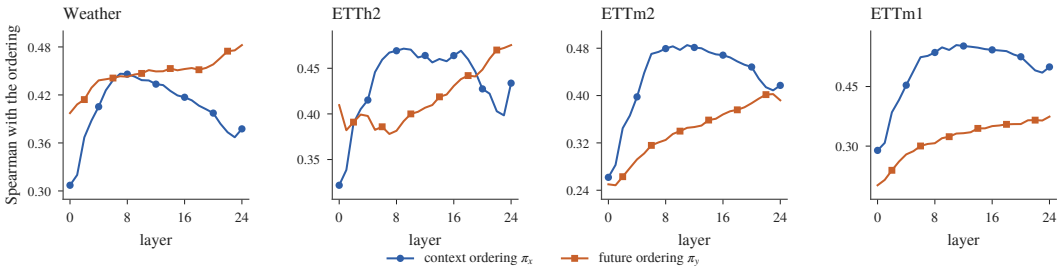


Figure 5: Spearman correlation of the layer-ℓ key's similarity score with two candidate orderings: by context distance (π_x) and by the error of using the candidate's future as the forecast (π_y).

rise together over the first third of the network and then separate: the agreement with the context ordering peaks in the middle layers and declines, whereas the agreement with the future ordering keeps rising and is highest at the last layer on all four datasets. Depth trades faithfulness to the context for faithfulness to the forecast.

Retrieval cost. The cost of the bucketed index is set by the number of entries it scans, so its speed-up over exact search follows from the scanned fraction alone and is independent of hardware. Figure 6a uses this theoretical speed-up to measure what approximate retrieval costs in accuracy. Probing a single bucket scans 0.9% of the database on average, a speed-up of $114\times$, and raises mean MSE by 0.17% relative to exact search. The individual datasets scatter around this mean: at $n_{\text{probe}} = 1$ their speed-up is 65–$164\times$ and their change in MSE lies between -0.85% and $+0.79\%$, and at $n_{\text{probe}} = 2$ the spread has already halved. From $n_{\text{probe}} = 2$ on, the mean is within noise of exact search, and several settings are marginally better than it.

The theoretical speed-up leaves out the forward pass and the computation of the key, so we also measure the prediction time on a database of five million entries (Fig. 6b). For HiRAF the key is a by-product of the forward pass that produces the forecast, and computing it takes no measurable time. At this scale exact search already brings the total to $2.28\times$ the backbone alone; the bucketed index reduces it to $1.05\times$ at $n_{\text{probe}} = 1$ and $1.59\times$ at $n_{\text{probe}} = 16$. TS-RAG runs a separate embedding model to obtain its key, and this step alone costs $5.7\times$ the backbone's forward pass; with exact search on top, its total is $8.17\times$ the backbone.

Impact of the retrieval library. We evaluate five compositions of the retrieval library (Fig. 7a): (1) own channel, the library is the training span of the target channel itself, the default in this paper; (2) pooled dataset, the training spans of all channels of the same dataset are merged into one library; (3) related dataset, the library is the training span of a closely related dataset with a different distribution (e.g. ETTm2 when evaluating on ETTm1); (4) other domain, a dataset from a different domain; (5) all other datasets, the training spans of the five remaining datasets merged. Under every composition HiRAF improves over the backbone, with the exception of the all-other-datasets library on Weather and Electricity, where it is on par with it; the own-channel library has the lowest error or is within 0.5 points of it, and the other-domain and all-other-datasets libraries bring the smallest gains. Retrieval is thus helpful in general, while the composition of the library decides how large the gain is, and the target series' own history is the most effective library.

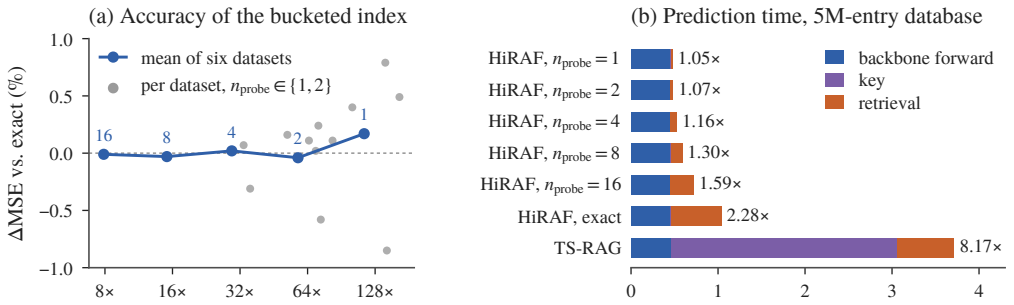


Figure 6: Retrieval cost. (a) Change in test MSE relative to exact search against the theoretical speed-up of the bucketed index; the line is the mean over the six datasets with n_{probe} as label, the grey points are the individual datasets at $n_{\text{probe}} \in \{1, 2\}$. (b) Prediction time on a database of five million entries in seconds per batch of 256 queries, Chronos-Bolt backbone, split into the backbone's forward pass, the computation of the key, and retrieval. Appendix D tabulates both panels.

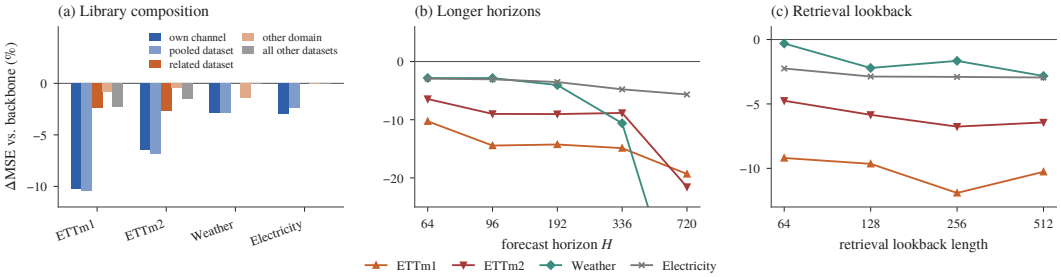


Figure 7: (a) Retrieval library: the channel's own history, all channels of the same dataset pooled, a related dataset (ETTm1↔ETTm2; none for Weather and Electricity), a dataset from another domain, and all other datasets pooled. (b) Forecast horizon H, with retrieval performed once from the 512-step context. (c) Length of the context used for retrieval; the forecast still uses 512 steps.

Longer horizons. The gain of retrieval grows with the horizon (Fig. 7b): on ETTm1 from -10.3% at $H = 64$ to -19.3% at $H = 720$, on ETTm2 from -6.4% to -21.6%—the backbone loses more accuracy over long horizons than the retrieved true continuations do. One cell needs comment: Weather at $H = 720$ leaves the plotted range (-46.6%), but the backbone's own error there is $2.5\times$ its error at $H = 336$, so the number reflects the backbone's failure more than the method's strength.

Retrieval lookback length. The forecast still uses the 512-step context, while the key is computed from only the last 64, 128, 256 or 512 steps (Fig. 7c). The best lookback varies with the dataset: 256 steps on ETTm1 and ETTm2, and 512 on Weather and Electricity, with the gain on Weather almost vanishing at 64 steps. The differences stay within a few points, and we use the full 512 steps by default.

6 Conclusion

In this paper we presented HiRAF, a retrieval-augmented forecasting method that uses the hidden state of a frozen time series foundation model as the retrieval key. We observed that existing methods compute similarity on the context of a historical segment while the forecast uses its continuation, and argued that the hidden state, which encodes what the model needs for prediction, is a better key for this purpose; used on its own, it retrieves continuations with 14–17% lower error than the raw series or a separate embedding model. HiRAF requires no training and no additional embedding model, and a bucketed index keeps the retrieval cost small as the history grows. Experiments on six benchmark datasets show that HiRAF improves the zero-shot accuracy of both backbones, outperforms existing TSFMs and the retrieval-augmented method TS-RAG, and on an index of five million entries increases the prediction time of the backbone by only 5%, compared with more than $8\times$ for TS-RAG. Our analysis further shows that the gain comes from the hidden-state key itself and increases with the forecast horizon. In future work we plan to extend the retrieval library beyond the target series' own history.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing and to organise the experimental results into tables and figures.

REPRODUCIBILITY STATEMENT

The method is specified in Section 3 and Algorithm 1; datasets, splits and the evaluation protocol in Section 4.1; index parameters, hyper-parameter selection and hardware in Appendix A; the repair of the benchmark files in Appendix B. Source code and configurations are provided in the supplementary material.

REFERENCES

Rishi Ahuja, Kumar Prateek, Simranjit Singh, and Vijay Kumar. Retrieval mechanisms surpass long-context scaling in time series forecasting. *arXiv preprint arXiv:2605.08217*, 2026.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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*, 2022.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, 2024.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan Ö. Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *International Conference on Machine Learning*, 2025.

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

Bo Jiang. One model is enough: Native retrieval embeddings from LLM agent hidden states. *arXiv preprint arXiv:2603.08429*, 2026.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era. *arXiv preprint arXiv:2605.20119*, 2026.

Morad Laglil, Bertrand Pracca, Emilie Devijver, and Eric Gaussier. Foundation models and fine-tuning: Toward a new generation of models for time series forecasting. *arXiv preprint arXiv:2607.23146*, 2026.

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

Seunghan Lee, Jaehoon Lee, Jun Seo, Sungdong Yoo, Minjae Kim, Tae Yoon Lim, Dongwan Kang, Hwanil Choi, SoonYoung Lee, and Wonbin Ahn. Not all retrievals are useful: Cross-attention for input-aware RAG in time series forecasting. *arXiv preprint arXiv:2603.14709*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

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*, 2024a.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *International Conference on Machine Learning*, 2024b.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. In *International Conference on Learning Representations*, 2025.

Yong Liu, Xingjian Su, Shiyu Wang, Haoran Zhang, Haixuan Liu, Yuxuan Wang, Zhou Ye, Yang Xiang, Jianmin Wang, and Mingsheng Long. Timer-S1: A billion-scale time series foundation model with serial scaling. *arXiv preprint arXiv:2603.04791*, 2026.

Sisuo Lyu, Siru Zhong, Tiegang Chen, Weilin Ruan, Qingxiang Liu, Taiqiang Lv, Qingsong Wen, Raymond Chi-Wing Wong, and Yuxuan Liang. Ts-memory: Plug-and-play memory for time series foundation models. *arXiv preprint arXiv:2602.11550*, 2026.

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

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*, 2025.

Eghbal Rahimikia, Hao Ni, and Weiguan Wang. Re(visiting) time series foundation models in finance. *arXiv preprint arXiv:2511.18578*, 2025.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *International Conference on Machine Learning*, 2025.

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

Tianfeng Wang and Gaojie Cui. RATSF: Empowering customer service volume management through retrieval-augmented time-series forecasting. *arXiv preprint arXiv:2403.04180*, 2024.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning*, 2024.

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*, 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.

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

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2023.

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

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.