NOT ALL HISTORY MATTERS EQUALLY: TOKENIZATION THROUGH INSTANCE-ADAPTIVE AGGREGATION WITH A TIME-DECAY PRIOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Tokenization determines how historical information is organized for time series forecasting, yet conventional patching assigns one token to every patch regardless of its predictive value. This uniform allocation leaves the forecasting backbone to resolve redundant patterns and suppress irrelevant or noisy context. We propose *PatchBasis*, an instance-adaptive tokenization method built on the principle that not all history matters equally. Its *BasisLearner* computes input-dependent aggregation weights that combine forecast-relevant patterns across non-contiguous historical regions into a compact token basis. Because recent observations provide an important starting point for forecasting and can restore details lost during aggregation, a *Time-Decay Prior* merges recent patches into the learned basis with weights that decrease with temporal distance from the forecast origin. The tokenizer and forecasting backbone are trained jointly under the forecasting objective, directing a fixed token budget toward predictive information. Experiments on nine long-term forecasting benchmarks demonstrate competitive performance. In a protocol-matched search-based comparison, PatchBasis achieves the lowest average MAE on all five datasets and the lowest average MSE on four. Aggregation and recent-candidate ablations further support the contributions of instance-adaptive aggregation and recent-context preservation, while sensitivity and efficiency analyses characterize the resulting trade-offs. Anonymous code is available at https://anonymous.4open.science/status/PatchBasis-0DB3.

1 INTRODUCTION

Tokenization determines how historical information is organized for time series forecasting. Its quality can determine the empirical performance ceiling of downstream models. CombinationTS (Wang et al., 2026) shows that even a parameter-free identity encoder can match or outperform complex backbones with well-designed tokenization. This finding motivates us to improve tokenization.

Patch-based tokenization (Nie et al., 2022; Huang et al., 2024; Hu et al., 2025; Tang & Zhang, 2025; Ding et al., 2026) has become a widely adopted approach to time series forecasting. PatchTST (Nie et al., 2022) groups consecutive observations into patches to preserve local temporal structure. It also enables the model to process longer histories. However, conventional patching assigns one token to every patch regardless of its forecast relevance (Figure 1). Low-relevance patches containing noise or anomalies receive the same token capacity as forecast-relevant motifs (Figure 1). Therefore, the downstream model must identify useful information among redundant and potentially distracting tokens.

Existing studies improve tokenization through local patching (Huang et al., 2024; Ding et al., 2026), aggregation (Huang et al., 2025; Lee & Lee, 2026), and selection (Wu et al., 2026). HDMixer (Huang et al., 2024) extends patch boundaries to preserve local patterns. TimeMosaic (Ding et al., 2026) adapts patch granularity to local information density. These methods still construct patches from contiguous temporal regions. TimeBase (Huang et al., 2025) observes that time series often exhibit repeated temporal patterns and approximate low-rank structure. It aggregates historical segments into a compact basis for forecasting. Its learned aggregation weights are shared across instances. SRSNet (Wu et al., 2026) selects and reorders informative patches for each input.

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.

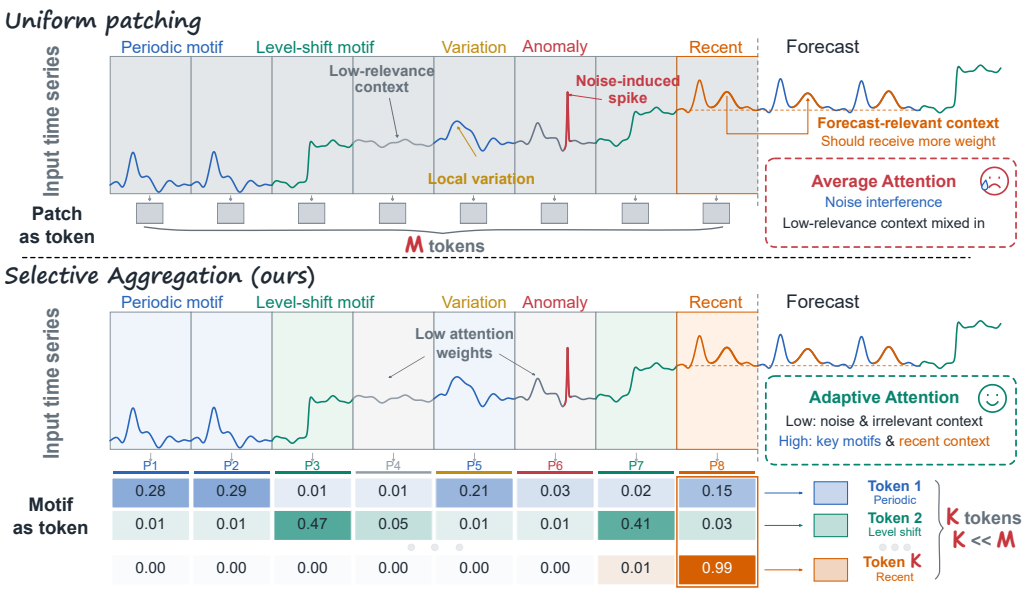

Figure 1: Uniform patching and selective aggregation. Uniform patching assigns a separate token to every historical patch. Selective aggregation combines recurring motifs and level shifts, downweights less informative regions and disturbances, and emphasizes recent context. The sequence and aggregation weights are illustrative.

Building on these insights, we argue that not all history matters equally. Selective aggregation can concentrate forecast-relevant information into a compact token basis and reduce redundancy (Figure 1). To this end, we introduce *BasisLearner* for instance-adaptive selective aggregation. It computes aggregation weights from the current input to reorganize information from non-contiguous historical patches into the token basis. We further observe that *BasisLearner* assigns high weights to recent patches in some tokens without an explicit recency prior (Figure 2). Recent patches provide a starting point for forecasting and supplement features lost during aggregation. Therefore, we design a *Time-Decay Prior* to inject recent patch information directly into the token basis. The injection weights decrease with temporal distance from the forecast origin. We combine *BasisLearner* and the *Time-Decay Prior* to form *PatchBasis* for time series forecasting.

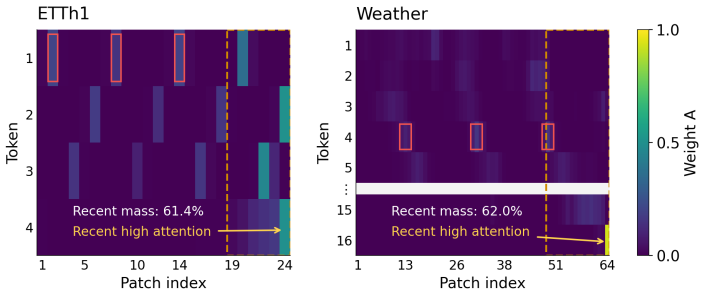

Figure 2: Learned aggregation weights without the time-decay branch on ETTh1 and Weather. Some tokens assign high weights to recent patches. Dashed boxes mark the recent region. Red boxes highlight informative motifs aggregated by BasisLearner.

Our contributions are summarized as follows:

- We explore tokenization from the perspective of instance-adaptive selective aggregation. It reorganizes forecast-relevant information across non-contiguous historical patches into a compact token basis.
- We propose *PatchBasis* to combine selective aggregation and recent patch preservation in a unified tokenization architecture. *BasisLearner* and the *Time-Decay Prior* jointly retain historical patterns and recent details within a fixed token budget.
- Experiments on five benchmarks demonstrate competitive forecasting performance. Ablations examine the contributions of selective aggregation and the time-decay prior. Further analyses characterize computational costs and sensitivity to tokenization settings.

2 Related Work

Local patching. PatchTST (Nie et al., 2022) constructs tokens from contiguous windows, whereas *PatchBasis* forms tokens from multiple historical regions. HDMixer (Huang et al., 2024) adapts patch offsets and extensions, while *BasisLearner* learns how much each patch contributes to a token. While TimeMosaic (Ding et al., 2026) and EntroPE (Abeywickrama et al., 2025) adapt local patch boundaries, *PatchBasis* adapts aggregation across the history.

Patch aggregation. Unlike TimeBase's instance-shared weights (Huang et al., 2025), *BasisLearner* generates aggregation weights from each input. TimePerceiver (Lee & Lee, 2026) aggregates temporal and channel information into latent tokens, while *PatchBasis* aggregates historical patches independently for each channel. These latent tokens update the original patch sequence before decoding, whereas *PatchBasis* passes compact tokens directly to the forecasting backbone. Beyond global aggregation, *PatchBasis* incorporates recent patches through a time-decay prior to supplement features lost during aggregation.

Patch selection. SRSNet (Wu et al., 2026) selects and reorders informative patches, whereas *BasisLearner* combines multiple patches within each token. Instead of replacing low-quality patches with series-level prototypes as in SEER (Qiu et al., 2026), *PatchBasis* aggregates patches from the current input. SEER preserves the token count during replacement, whereas *PatchBasis* aggregates the history into a fixed budget of K tokens.

3 Method

PatchBasis learns a compact set of tokens that adapts to each forecasting instance (Figure 3). Contextual patches provide the source representations. BasisLearner aggregates them into a compact set of tokens, and Time-Decay Prior merges recent patches with these tokens using fixed, temporally decreasing weights. The output contains a fixed number of tokens for the forecasting backbone.

Preliminaries: Problem Setup and Notations. Given a historical sequence $\mathbf{X} \in \mathbb{R}^{L\times C}$, multivariate time series forecasting aims to predict the next H observations $\mathbf{Y} \in \mathbb{R}^{H\times C}$, where L is the input length and C is the number of variables. A forecasting model f_θ produces predictions $\widehat{\mathbf{Y}} = f_\theta(\mathbf{X})$, with θ denoting its trainable parameters. Following PatchTST (Nie et al., 2022), we process channels independently with shared parameters. RevIN (Kim et al., 2021) normalizes each channel's input and restores its predictions to the original scale. We describe a single normalized history $\mathbf{x} \in \mathbb{R}^L$ and omit sample and channel indices. Let P denote the patch length, S the stride, and M the number of historical patches. The tokenizer produces $K \le M$ tokens, each with embedding dimension d. We denote the embedded patches with local context by $\mathbf{P} \in \mathbb{R}^{M\times d}$ and the final tokens supplied to the forecasting backbone by $\mathbf{Z} \in \mathbb{R}^{K\times d}$.

3.1 Rethinking: From Local Patches to Adaptive Aggregation

Local patching. PatchTST (Nie et al., 2022) assigns one token to each patch. TimeMosaic (Ding et al., 2026) adjusts local granularity. Each token still corresponds to a contiguous temporal region. We compare token construction using $\mathbf{A} \in \mathbb{R}^{K\times M}$ for the normalized patch-to-token aggregation map. Additive embeddings are omitted in this comparison.

$$\mathbf{T} = \underbrace{\mathbf{P}}_{\text{one token per patch}}, \qquad \mathbf{T} \in \mathbb{R}^{M\times d}.$$

Instance-shared aggregation. TimeBase (Huang et al., 2025) exploits approximate low-rank structure to aggregate historical segments into a compact token basis. This allows tokens to combine information across temporal regions. The aggregation weights remain shared across inputs:

$$\mathbf{T} = \underbrace{\mathbf{A}}_{\text{shared across inputs}}\mathbf{P}, \qquad \mathbf{T} \in \mathbb{R}^{K\times d}.$$

Instance-adaptive aggregation. *BasisLearner* computes aggregation weights from the current patches. Each token can selectively combine non-contiguous historical regions. The aggregation

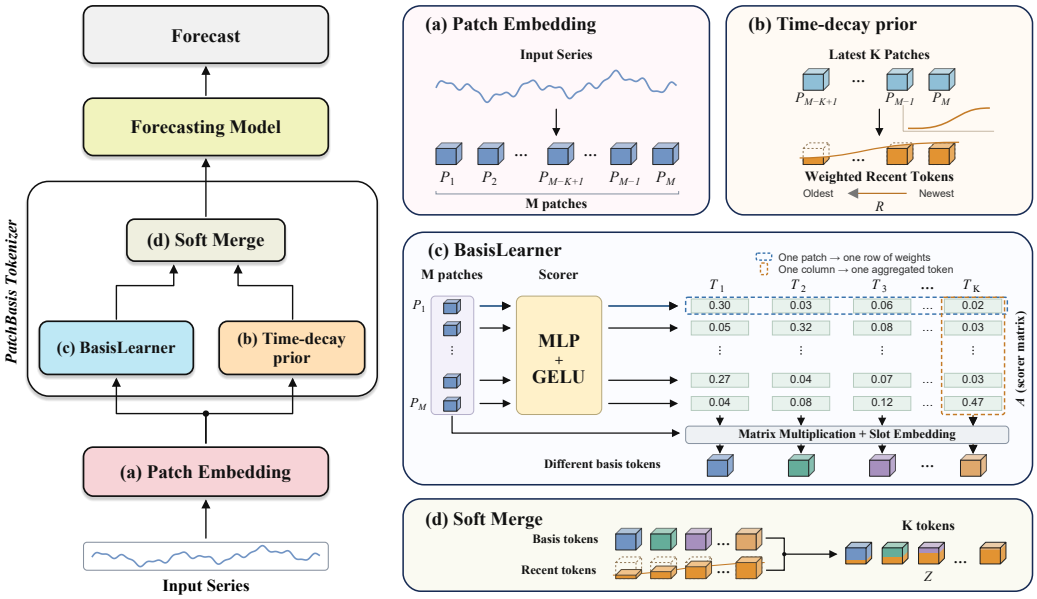

Figure 3: Overview of *PatchBasis*. (a) Patch Embedding maps historical patches into embeddings with local context. (b) *Time-Decay Prior* weights recent patches by their distance from the forecast origin. (c) *BasisLearner* computes instance-adaptive weights to aggregate historical patches into a compact token basis. (d) Soft Merge combines the token basis with weighted recent patches. The resulting tokens are passed to the forecasting model.

weights now depend on the input:

$$\mathbf{T} = \underbrace{\mathbf{A}(\mathbf{P})}_{\text{adapted to each input}} \mathbf{P}, \qquad \mathbf{A}(\mathbf{P}) \in \mathbb{R}^{K \times M}, \quad \mathbf{T} \in \mathbb{R}^{K \times d}.$$

The time-decay prior preserves recent patches. They provide a starting point for forecasting and supplement features lost during aggregation.

3.2 BasisLearner: Instance-Adaptive Patch Aggregation

We embed each patch through a shared linear projection and add positional embeddings. A residual temporal convolution incorporates neighboring patch information. Layer normalization yields $\mathbf{P} \in \mathbb{R}^{M \times d}$, with row $\mathbf{P}_m$ corresponding to patch m (Appendix A.2).

BasisLearner uses input-dependent weights to combine the historical regions relevant to the current instance. We apply a shared two-layer MLP to each contextual patch, producing K scores per patch, one for each output token. Denoting this scoring operation by Scorer, we first obtain the score matrix $\mathbf{S} \in \mathbb{R}^{M \times K}$ and then define the normalized patch-to-token aggregation map as

$$\mathbf{S} = \mathrm{Scorer}(\mathbf{P}), \qquad \mathbf{A}(\mathbf{P}) = \mathrm{softmax}(\mathbf{S})^{\top} \in \mathbb{R}^{K \times M}. \tag{1}$$

Here, $\mathrm{softmax}_M$ normalizes each score column over the M patches. Consequently, every row of $\mathbf{A}(\mathbf{P})$ sums to one and specifies how the historical patches contribute to one output token. Using the same aggregation notation as in Section 3.1, we aggregate the patches and add learnable slot embeddings:

$$\mathbf{T} = \mathbf{A}(\mathbf{P})\mathbf{P} + \mathbf{U}, \qquad \mathbf{T} \in \mathbb{R}^{K \times d}. \tag{2}$$

Here, $\mathbf{U} \in \mathbb{R}^{K \times d}$ distinguishes the output token slots.

Each token can combine non-contiguous positions, and a source patch can contribute to multiple tokens. Recomputing $\mathbf{A}$ from each input lets the same token slot draw on different historical regions across instances.

3.3 TIME-DECAY PRIOR

BasisLearner assigns high weights to recent patches in some tokens without an explicit prior (Figure 2). Recent patches provide a starting point for forecasting. Preserving them also supplements features lost during aggregation. The Time-Decay Prior explicitly incorporates this recent information. We collect the latest K rows of $\mathbf{P}$ in reverse temporal order to form the recent candidate matrix $\mathbf{R} \in \mathbb{R}^{K \times d}$, ordered from newest to oldest.

The vector $\boldsymbol{\mu}$ assigns each recent patch a fixed weight between zero and one. For recent candidates indexed from newest to oldest, we compute

$$\mu_k = \mathrm{sigmoid}\left(\frac{\rho - k + 1/2}{\tau_a}\right), \qquad k = 1, \ldots, K. \tag{3}$$

Here, ρ is a task-specific hyperparameter that controls the effective extent of recent history, and $\tau_a > 0$ is a temperature hyperparameter that controls the smoothness of the time decay. Both remain fixed during training and are shared across input sequences. A separate diagnostic extension replaces the fixed ρ with an input-dependent recent budget; its protocol and results are reported in Appendix F. Since k increases as a candidate patch moves farther from the forecast origin, μ_k decreases gradually from newer to older candidates. Writing $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^\top \in [0, 1]^{K \times 1}$, we merge the recent candidates with the aggregated tokens by weighting the two contributions and adding them:

$$\mathbf{Z} = \boldsymbol{\mu} \odot \mathbf{R} + (1 - \boldsymbol{\mu}) \odot \mathbf{T}. \tag{4}$$

Here, $\odot$ denotes element-wise multiplication, with the $K \times 1$ weights broadcast along the feature dimension. The resulting $\mathbf{Z} \in \mathbb{R}^{K \times d}$ retains recent context alongside the aggregated historical patterns while keeping the output token count at K.

3.4 FORECASTING AND OPTIMIZATION

The forecasting loss trains the tokenizer and backbone jointly. The backbone and prediction head map the K fused tokens to future values, which RevIN restores to the input scale. We minimize mean absolute error to train the patch representations, adaptive aggregation, and forecasting model jointly. The time-decay hyperparameters remain fixed during training. The tokenizer reduces M source patches to K output tokens, with compression ratio M/K.

4 EXPERIMENTS

We evaluate *PatchBasis* through forecasting comparisons and model analysis. The experiments examine selective aggregation, the time-decay prior, transfer across backbones, computational costs, and parameter sensitivity.

Datasets. We conduct long-term forecasting experiments on nine datasets: four ETT subsets, Weather, Electricity, Traffic, Exchange-Rate, and Solar-Energy. Dataset statistics, preprocessing, and chronological splits are provided in Appendix A.

Baselines. We compare *PatchBasis* with DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2022), iTransformer (Liu et al., 2024), TimeMixer (Wang et al., 2024), SimpleTM (Chen et al., 2025), TimeFilter (Hu et al., 2025), DUET (Qiu et al., 2025), TimeBase (Huang et al., 2025), SRSNet (Wu et al., 2026), TimePerceiver (Lee & Lee, 2026), TimeMosaic (Ding et al., 2026), and SEER (Qiu et al., 2026). These methods cover different forecasting architectures and tokenization strategies.

Unified Evaluation settings. We report MSE and MAE averaged over four prediction lengths: 96, 192, 336, and 720. Lower values indicate better performance. We use two forecasting comparisons. For the main-table comparison on five datasets, our experimental configurations and evaluation protocol are fully aligned with TimeMosaic (Ding et al., 2026). The second fixes the input length to 96 and uses shared dataset-specific configurations for baseline reruns on nine datasets. Detailed selection rules and baseline exceptions are provided in Appendix A.3.

Models	PatchBasis **Ours**		TimeMosaic (2026)		SimpleTM (2025)		TimeFilter (2025)		DUET (2025)		iTransformer (2024)		TimeMixer (2024)		PatchTST (2022)		DLinear (2023)	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.334**	**0.362**	0.342	0.367	0.346	0.378	0.353	0.377	0.354	0.377	0.356	0.385	0.349	0.392	0.348	0.380	0.356	0.378
ETTm2	**0.244**	**0.302**	0.250	0.305	0.257	0.316	0.253	0.317	0.253	0.313	0.267	0.325	0.256	0.315	0.256	0.318	0.262	0.326
ETTh1	**0.389**	**0.412**	0.397	0.417	0.416	0.434	0.414	0.429	0.406	0.425	0.425	0.438	0.416	0.428	0.410	0.430	0.419	0.437
ETTh2	0.338	**0.378**	0.348	0.383	0.358	0.396	0.347	0.394	**0.334**	0.384	0.362	0.398	0.349	0.392	0.359	0.399	0.412	0.433
Weather	**0.218**	**0.245**	0.223	0.251	0.220	0.259	0.220	0.261	0.240	0.277	0.230	0.269	0.229	0.265	0.223	0.260	0.240	0.291

Table 1: Forecasting errors averaged over $H \in \{96, 192, 336, 720\}$. Baseline results are taken from TimeMosaic (Ding et al., 2026); our main-table experiments use the same dataset-specific configurations and evaluation protocol. Rankings use three-decimal values.

Models	PatchBasis (**Ours**, CI)		TimeMosaic (2026, CI)		SEER (2026, CI)		TimeFilter (2025, CD)		DUET-CI (2025, CI)		TimeBase (2025, CI)		SRSNet (2026, CI)		TimePerceiver (2026, CD)		iTransformer (2024, CD)		PatchTST (2022, CI)		DLinear (2023, CI)	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.372**	**0.379**	0.387	0.385	0.390	0.390	0.404	0.394	0.403	0.391	0.430	0.411	0.393	0.389	0.392	0.386	0.406	0.398	0.382	0.384	0.397	0.391
ETTm2	0.273	0.315	0.278	0.318	0.280	0.318	0.272	0.316	0.286	0.323	0.314	0.349	0.284	0.324	**0.271**	**0.312**	0.281	0.320	0.280	0.319	0.287	0.330
ETTh1	**0.424**	**0.420**	0.438	0.431	0.460	0.440	0.473	0.441	0.443	0.432	0.459	0.445	0.448	0.432	0.645	0.511	0.451	0.441	0.446	0.434	0.492	0.477
ETTh2	**0.366**	**0.389**	**0.366**	0.390	0.374	0.393	0.374	0.396	0.376	0.396	0.457	0.466	0.374	0.395	0.371	0.392	0.372	0.396	0.371	0.392	0.438	0.440
Weather	0.244	**0.263**	0.259	0.273	0.250	0.269	**0.241**	**0.263**	0.276	0.284	0.303	0.301	0.245	0.266	0.243	**0.263**	0.258	0.274	0.255	0.269	0.272	0.293
Electricity	**0.184**	**0.273**	0.252	0.334	0.629	0.599	0.241	0.321	0.403	0.412	0.229	0.295	0.258	0.341	0.482	0.487	0.361	0.416	0.251	0.334	0.219	0.299
Traffic	0.484	0.292	0.480	0.282	**0.436**	**0.259**	0.455	0.273	0.643	0.367	0.742	0.387	0.497	0.301	0.457	0.266	0.438	0.270	0.486	0.284	0.661	0.352
Solar	**0.231**	0.244	0.316	0.300	0.615	0.451	0.248	0.270	0.347	0.317	0.497	0.453	0.302	0.297	0.263	**0.243**	0.305	0.294	0.297	0.314	0.335	0.319
Exchange	0.361	0.401	0.371	0.408	0.375	0.410	0.418	0.430	0.357	**0.400**	0.554	0.563	0.374	0.409	0.378	0.414	0.377	0.414	0.368	0.409	**0.330**	0.406

Table 2: Forecasting errors under the uniform baseline protocol ($L = 96$), averaged over $H \in \{96, 192, 336, 720\}$. Baselines use unified configurations; our entries use retained configurations. Rankings use three-decimal values. Each average uses all four horizons. Solar entries for TimeFilter (Hu et al., 2025) use its reported results, not a rerun.

Implementation details. Our runs use Adam and MAE loss. Checkpoints are selected by validation MAE. Due to training cost, we do not repeat the full forecasting performance comparisons across multiple random seeds. All other experiments are evaluated with three random seeds, and we report their mean results. Training configurations and search spaces are provided in Appendix A. A separate three-seed forecasting check is reported in Appendix A.4.

4.1 Main Results

Forecasting performance. Table 1 reports the search-based comparison. *PatchBasis* achieves the lowest average MAE on all five datasets and the lowest average MSE on four. Compared with TimeMosaic, the strongest baseline on ETTm1 and ETTm2, it reduces MSE by **2.3%** and **2.4%**, respectively. On Weather, it reduces MAE by **2.4%** relative to TimeMosaic, the best baseline for this metric. Full results for individual prediction lengths are provided in Appendix B.

Table 2 extends the comparison to nine datasets with an input length of 96. *PatchBasis* achieves the lowest MSE and MAE on ETTm1, ETTh1, and Electricity. The largest gains over the best baselines occur on Electricity. MSE decreases by **16.0%** relative to DLinear, and MAE decreases by **7.5%** relative to TimeBase. On Solar-Energy, MSE decreases by **6.9%** relative to TimeFilter, the strongest baseline for this metric. These reductions compare errors averaged over the four prediction lengths. Full results are provided in Appendix C.

Backbone	Dataset	Full patch		TimeBase		APE		TOTEM		BasisLearner	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Raw	ETTh1	0.371	0.389	**0.365**	**0.387**	–	–	0.375	0.395	0.369	0.388
	ETTh2	0.280	0.332	**0.278**	**0.331**	–	–	0.286	0.341	0.283	0.334
	ETTm1	0.290	0.336	0.296	0.338	–	–	0.302	0.345	**0.289**	**0.332**
	ETTm2	0.168	**0.248**	0.169	**0.248**	–	–	0.170	0.255	**0.167**	0.249
	Weather	0.154	0.194	0.176	0.215	–	–	0.172	0.215	**0.152**	**0.193**
	Avg.	0.253	0.300	0.257	0.304	–	–	0.261	0.310	**0.252**	**0.299**
Embedding	ETTh1	0.375	0.394	**0.361**	0.384	0.365	0.386	0.379	0.397	0.364	**0.383**
	ETTh2	0.272	**0.328**	0.277	0.331	**0.269**	**0.328**	0.277	0.334	0.277	0.330
	ETTm1	0.283	0.328	0.293	0.335	0.294	0.337	0.292	0.335	**0.279**	**0.325**
	ETTm2	0.165	0.246	0.165	0.248	0.165	0.248	0.166	0.249	**0.164**	**0.245**
	Weather	0.150	0.188	0.177	0.214	0.158	0.196	0.166	0.208	**0.148**	**0.184**
	Avg.	0.249	0.297	0.255	0.302	0.250	0.299	0.256	0.305	**0.246**	**0.293**
Transformer	ETTh1	0.363	0.387	**0.358**	0.383	0.365	0.389	0.384	0.400	0.360	**0.381**
	ETTh2	0.288	0.336	0.282	0.331	0.281	0.335	0.280	0.335	**0.279**	**0.328**
	ETTm1	0.294	0.332	0.278	0.326	0.292	0.334	0.293	0.335	**0.276**	**0.323**
	ETTm2	**0.163**	0.246	**0.163**	0.245	0.166	0.248	0.167	0.250	0.165	**0.242**
	Weather	0.148	**0.184**	0.148	0.187	0.149	0.188	0.154	0.191	**0.146**	0.185
	Avg.	0.251	0.297	0.246	0.294	0.251	0.299	0.256	0.302	**0.245**	**0.292**

Table 3: Tokenization across backbones ($L = 320$, $H = 96$). Configurations are selected by validation MAE within each method and backbone, across embedding widths where applicable. Entries report means over three random seeds after configuration selection. Avg. equally weights the five displayed dataset means. BasisLearner is evaluated without the recent branch. APE requires embeddings and is inapplicable to Raw (–). Rankings use three-decimal values, with ties sharing the same rank.

4.2 Model Analysis

BasisLearner ablation. We compare *BasisLearner* with Full patch, TimeBase, APE, and TOTEM under Raw, Embedding, and Transformer settings (Table 3). The time-decay branch is disabled to isolate selective aggregation. Configurations are selected by validation MAE. Each selected configuration is evaluated with three random seeds. The table reports their mean errors.

BasisLearner achieves the lowest average MSE and MAE under all three backbone settings. With Transformer, it reduces average MSE by **2.4%** and MAE by **1.7%** relative to Full patch. With Embedding on Weather, it reduces MSE by **16.4%** and MAE by **14.0%** relative to TimeBase. This comparison highlights the benefit of instance-adaptive aggregation over instance-shared weights in this setting. These gains occur without the time-decay branch. Appendix G further visualizes how aggregation weights respond to localized perturbations.

Time-decay prior ablation. We first examine whether *BasisLearner* favors recent patches without an explicit prior. Table 4 shows that the weight mass assigned to the latest patches is 1.43–7.18 times the uniform reference. Figure 2 provides examples of this learned preference for recent patches.

Dataset	Patches		Mass		Ratio
	M	K	Recent	K/M	
ETTm1	21	4	0.820	0.190	4.308×
ETTm2	32	8	0.530	0.250	2.122×
ETTh1	22	8	0.518	0.364	1.425×
ETTh2	128	8	0.449	0.063	7.180×
Weather	64	16	0.427	0.250	1.708×

Table 4: Recent attention with the branch disabled, averaged over three seeds. Mass covers the latest K of M patches; Ratio compares it with K/M, using unrounded values.

Figure 4 compares recent candidates, uniformly spaced candidates, and removal of the recent branch. Recent candidates achieve a lower Score than removing the branch on all five datasets. They also outperform uniform candidates on four datasets. The mean Score decreases by 1.53% and 2.26%, respectively. Uniform candidates perform better on ETTh2, indicating that broader historical coverage remains useful for this dataset. Detailed settings and numerical results are provided in Appendix A.5.

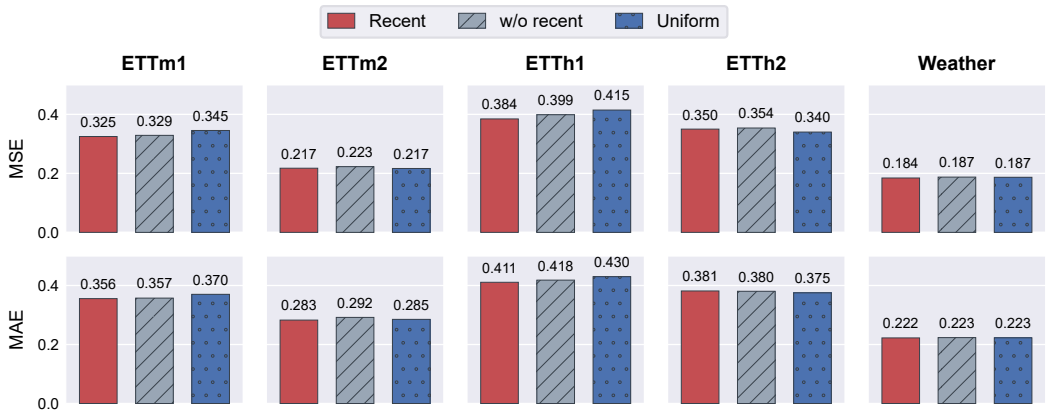

Figure 4: Recent-candidate ablation on five datasets ($L = 512$, $H = 192$). Rows show mean test MSE and MAE over three seeds. Recent uses the latest K patches; Uniform uses K evenly spaced patches; w/o recent removes the recent branch. Lower values are better.

Backbone	Native	+ plugin	Δ (%)
PatchTST	.277/.306	**.271/.302**	**+2.30/+1.45**
PatchTSMixer	**.273/.299**	.276/.303	-1.10/-1.29
PatchMixer	**.266/.301**	.272/.302	-2.22/-0.34
TimeXer	.272/.302	**.271/.302**	+0.26/+0.13

Table 5: Aggregation-plugin transfer averaged over three datasets. Entries are MSE/MAE; $\Delta > 0$ denotes improvement.

Tokenizer	MSE/MAE	Params	Infer.	Train
Full patch	.251/.297	321K	1.46 ms	5.96 ms
TimeBase	.246/.294	423K	1.22 ms	5.83 ms
BasisLearner	**.245/.292**	**229K**	1.88 ms	7.98 ms

Table 6: Transformer accuracy and efficiency averaged over five datasets. Time is per batch.

Transfer across backbones. We further evaluate selective aggregation as a plugin for existing forecasters (Table 5). The plugin improves both MSE and MAE for PatchTST on all three datasets. Its average improvements are 2.30% in MSE and 1.45% in MAE. Results are mixed for TimeXer. PatchTSMixer and PatchMixer show higher average errors after adding the plugin. These results indicate that plugin effectiveness depends on the forecasting architecture. Dataset-level results are provided in Appendix D.

Efficiency analysis. Table 6 reports computational costs using the configurations selected for Table 3. With Transformer, *BasisLearner* uses 229.37K parameters, compared with 321.03K for Full patch and 423.45K for TimeBase. Its batch inference time is 1.881 ms, compared with 1.456 ms and 1.218 ms, respectively. Selective aggregation reduces the parameter count in this comparison but introduces additional computation. Detailed measurements and profiling settings are provided in Appendix D.

Parameter sensitivity. Figure 5 examines compression ratio, patch length, and input length on ETTm1, Weather, and Electricity. Each point averages three random seeds. *PatchBasis* exhibits a smaller MSE range than TimeBase across all nine dataset–parameter combinations. For example, varying the compression ratio on ETTm1 produces an MSE range of 0.024 for *PatchBasis* and 0.341 for TimeBase. Longer histories do not consistently improve forecasting. The best evaluated input length for *PatchBasis* is 672 on ETTm1 and Weather. Additional MAE results and parameter grids are provided in Appendix E. Appendix H further evaluates the maximum compression ratio under a specified MSE tolerance relative to PatchTST.

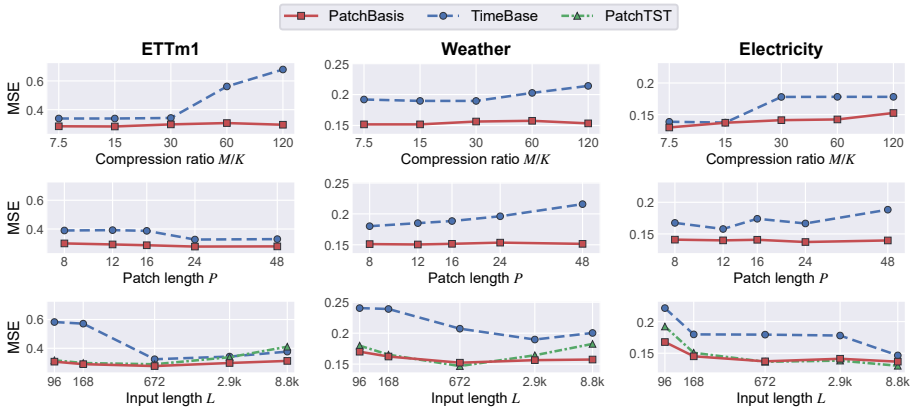

Figure 5: Parameter sensitivity on three datasets (columns). Rows vary compression ratio, patch length, and input length. Points show mean test MSE over three seeds with $H = 96$; horizontal axes are logarithmic. PatchTST is included in the input-length sweep.

5 Limitations and Future Work

The benefits of selective aggregation depend on the forecasting backbone. The plugin improves PatchTST but increases average errors for PatchTSMixer and PatchMixer (Table 5). Future work could adapt aggregation to the information needs of each backbone. Token compression also does not guarantee faster inference. With Transformer, *BasisLearner* uses fewer parameters but takes longer to run than Full patch and TimeBase (Table 6). More efficient scoring and aggregation could reduce this overhead. Finally, the time-decay prior uses fixed weights across inputs. Uniform candidates outperform recent candidates on ETTh2 (Figure 4), and an input-dependent budget allocator does not consistently improve over a validation-selected fixed budget (Appendix F). Learning how to adapt recent-history allocation reliably remains a direction for future work.

6 Conclusion

We introduced *PatchBasis*, an instance-adaptive tokenizer that compresses non-contiguous historical patterns into a compact token basis while preserving recent information through a time-decay prior. Across two long-term forecasting protocols, PatchBasis achieves competitive accuracy, including the lowest average MAE on all five datasets in the protocol-matched comparison. Ablations support both selective aggregation and recent-context preservation, while sensitivity, backbone-transfer, and efficiency analyses show that their benefits depend on the data and forecasting architecture. The component analyses clarify how the two mechanisms contribute. With the recent branch disabled, *BasisLearner* obtains the lowest average MSE and MAE under the Raw, Embedding, and Transformer settings, indicating that instance-adaptive aggregation can identify useful patterns beyond contiguous local patches. The time-decay prior complements this global aggregation by retaining a controlled amount of local context near the forecast origin. Recent candidates outperform removing this branch on all five evaluated datasets and outperform uniformly spaced candidates on four of them.

The broader analyses also identify the conditions under which this allocation is most useful. PatchBasis remains comparatively stable as the compression ratio, patch length, and input length vary, but longer histories do not consistently improve forecasting. Selective aggregation transfers effectively to PatchTST, whereas its gains are mixed for other backbones, showing that the tokenizer must match the information-processing pattern of the forecaster. Moreover, fewer tokenizer parameters do not automatically reduce inference time because input-dependent scoring and aggregation introduce computation. Tokenization is therefore a learnable resource-allocation problem: a fixed budget should combine forecast-relevant historical patterns with data-appropriate recent context. This view places token selection alongside the forecasting backbone as a central design choice for efficient forecasting.

AI USE STATEMENT

Generative AI tools were used to assist with language editing for grammar, clarity, and readability; drafting this AI use statement; designing and providing feedback on the research methodology and experiments; assisting with the implementation of the proposed method for the experiments; and translation. They were not used for any other task requiring disclosure under the ICLR 2027 policy. The authors critically reviewed, revised, and independently verified all AI-assisted suggestions, implementation details, translations, and text. All final scientific decisions were made by the authors, who take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the PatchBasis architecture, forecasting objective, datasets, baselines, evaluation metrics, configuration-selection procedures, and random-seed protocol. Appendix A provides the dataset sources and chronological splits, preprocessing and evaluation-window construction, training and checkpoint-selection details, tokenizer implementation, forecasting-comparison protocols, three-seed variability analysis, ablation settings, parameter-sensitivity grids, and backbone-transfer settings. Appendices B and C report the complete per-horizon forecasting results for the search-based and uniform-baseline comparisons, respectively. Appendix D documents the tokenizer and backbone configurations, search spaces, selected settings, and computational-cost measurement protocol. Anonymous source code is available at https://anonymous.4open.science/status/PatchBasis-0DB3.

REFERENCES

Sachith Abeywickrama, Emadeldeen Eldele, Min Wu, Xiaoli Li, and Chau Yuen. Entropy guided dynamic patch segmentation for time series transformers. *arXiv preprint arXiv:2509.26157*, 2025.

Hui Chen, Viet Luong, Lopamudra Mukherjee, and Vikas Singh. Simpletm: A simple baseline for multivariate time series forecasting. In *International Conference on Learning Representations*, volume 2025, pp. 15431–15456, 2025.

Kuiye Ding, Fanda Fan, Chunyi Hou, Zheya Wang, Lei Wang, Zhengxin Yang, and Jianfeng Zhan. Timemosaic: Temporal heterogeneity guided time series forecasting via adaptive granularity patch and segment-wise decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 20790–20798, 2026.

Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. Timefilter: Patch-specific spatial-temporal graph filtration for time series forecasting. *arXiv preprint arXiv:2501.13041*, 2025.

Qihe Huang, Lei Shen, Ruixin Zhang, Jiahuan Cheng, Shouhong Ding, Zhengyang Zhou, and Yang Wang. Hdmixer: Hierarchical dependency with extendable patch for multivariate time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 12608–12616, 2024.

Qihe Huang, Zhengyang Zhou, Kuo Yang, Zhongchao Yi, Xu Wang, and Yang Wang. Timebase: The power of minimalism in efficient long-term time series forecasting. In *ICML*, 2025.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

Jaebin Lee and Hankook Lee. Timeperceiver: An encoder-decoder framework for generalized time-series forecasting. *Advances in Neural Information Processing Systems*, 38:136136–136165, 2026.

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*, volume 2024, pp. 11116–11140, 2024.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. Duet: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD conference on knowledge discovery and data mining v. 1*, pp. 1185–1196, 2025.

Xiangfei Qiu, Xvyuan Liu, Tianen Shen, Xingjian Wu, Hanyin Cheng, Bin Yang, and Jilin Hu. Seer: Transformer-based robust time series forecasting via automated patch enhancement and replacement. *arXiv preprint arXiv:2602.00589*, 2026.

Peiwang Tang and Weitai Zhang. Unlocking the power of patch: patch-based mlp for long-term time series forecasting. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i12.33378. URL https://doi.org/10.1609/aaai.v39i12.33378.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 38626–38652, 2024.

Xiaorui Wang, Fanda Fan, Chenxi Wang, Yuxuan Yang, Rui Tang, Kuoyu Gao, Simiao Pang, Yuanfeng Shang, Zhipeng Liu, Wanling Gao, et al. Combinationts: A modular framework for understanding time-series forecasting models. *arXiv preprint arXiv:2605.01231*, 2026.

Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. Enhancing time series forecasting through selective representation spaces: A patch perspective. *Advances in Neural Information Processing Systems*, 38:23328–23354, 2026.

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.

A Experiment Details

A.1 Datasets

We evaluate PatchBasis on nine real-world multivariate time series datasets for long-term forecasting. They cover electricity, weather, traffic, exchange rates, and solar energy.

- **Electricity Transformer Temperature (ETT).** The four ETT subsets contain oil temperature and six power-load variables from two electricity transformers. ETTh1 and ETTh2 are sampled hourly, while ETTm1 and ETTm2 are sampled every 15 minutes.[1]
- **Weather.** This dataset contains 21 meteorological variables recorded every 10 minutes in 2020 at the Max Planck Institute for Biogeochemistry weather station in Jena, Germany.[2]
- **Electricity.** The processed benchmark records hourly electricity consumption from 321 clients.[3]
- **Traffic.** This dataset records hourly road occupancy rates from 862 sensors on freeways in the San Francisco Bay Area.
- **Exchange-Rate.** This dataset contains daily exchange rates for eight currencies.
- **Solar-Energy.** This dataset contains solar power production from 137 photovoltaic plants in Alabama, sampled every 10 minutes during 2006.

Table 7 summarizes the dimensions, sampling intervals, and chronological splits used in our experiments. ETT uses the standard 12/4/4-month training, validation, and test split, implemented with 30-day months. The other datasets use approximately 70%, 10%, and 20% of the observations for training, validation, and testing, respectively.

Dataset	Dim.	Train	Validation	Test	Frequency
ETTh1, ETTh2	7	8,640	2,880	2,880	Hourly
ETTm1, ETTm2	7	34,560	11,520	11,520	15 min
Weather	21	36,886	5,270	10,539	10 min
Electricity	321	18,412	2,632	5,260	Hourly
Traffic	862	12,280	1,756	3,508	Hourly
Exchange-Rate	8	5,311	760	1,517	Daily
Solar-Energy	137	36,792	5,256	10,512	10 min

Table 7: Dataset statistics. Dim. is the number of variables. Split sizes count time points before sliding-window construction and exclude history prepended from the preceding split. Counts in each ETT row apply to each listed subset.

A.2 Training and Evaluation

Forecasting tasks. All forecasting comparisons use multivariate inputs and predict all variables. We evaluate prediction lengths $H \in \{96, 192, 336, 720\}$ and report the arithmetic mean across these four horizons in the aggregate forecasting tables. The input lengths and settings for individual comparisons are specified below.

Preprocessing and evaluation windows. Each variable is standardized using the mean and standard deviation of the training split. The same transformation is applied to the validation and test splits. Input–target pairs are formed with sliding windows of stride one. Validation and test inputs may include preceding observations, while forecast targets remain within their respective splits. Training windows are shuffled, whereas validation and test windows retain chronological order. Our evaluation loader uses `drop_last=False`, so the final incomplete batch is included in the reported metrics.

[1] https://github.com/zhouhaoyi/ETDataset

[2] https://www.bgc-jena.mpg.de/wetter/

[3] Processed Electricity, Traffic, Exchange-Rate, and Solar-Energy datasets: https://github.com/laiguokun/multivariate-time-series-data

Training and metrics. Models are trained with Adam and MAE as the forecasting loss. Checkpoints are selected by validation MAE. We compute MSE and MAE on standardized targets, averaging over test windows, forecast steps, and variables. Score is defined as $S = (\mathrm{MSE} + \mathrm{MAE})/2$. Learning rates, batch sizes, training epochs, and learning-rate schedules follow the experiment-specific protocols below. Configuration selection is specified separately from checkpoint selection for each comparison.

Random-seed evaluation. The full forecasting comparisons in Tables 1 and 2 are not repeated across multiple random seeds due to training cost. All other experiments are re-evaluated with three random seeds under fixed configurations. We report means across these runs. Appendix A.4 provides a separate three-seed forecasting comparison at $L = 96$ and $H = 96$.

Tokenizer implementation. We extract M patches of length P with stride S, apply a shared linear projection to d dimensions, and add sinusoidal positional embeddings. A residual temporal convolution incorporates information from neighboring patches, yielding $\mathbf{P} \in \mathbb{R}^{M \times d}$, whose m-th row $\mathbf{P}_m$ represents the m-th contextual patch. Both adaptive patch aggregation and the recent branch use these representations.

The scoring MLP includes any fixed logit scaling before softmax, which can be absorbed into its final linear layer. The time-decay weights in Section 3.3 use the sigmoid function in Equation 3. Candidates are ordered from newest to oldest, and their weights decrease with temporal distance from the forecast origin. The task-specific hyperparameter ρ controls the effective extent of recent history, and τ_a controls the smoothness of the decay. Both remain fixed during training and are shared across inputs, while the patch representations and aggregation weights are learned jointly with the forecasting model.

A.3 Forecasting Comparison Protocols

Search-based forecasting comparison. The baseline entries in Table 1 follow the search-based comparison in Table 6 of TimeMosaic (Ding et al., 2026). To ensure direct comparability, our main-table experiments use the same dataset-specific configurations and evaluation protocol as TimeMosaic, including input lengths $L \in \{96, 192, 320, 512\}$. TimeMosaic's published protocol selected dataset-specific configurations using reported test performance; we inherit these configurations unchanged, while PatchBasis-specific choices and checkpoint selection use validation MAE. Table 11 gives the full results.

Uniform baseline comparison. Table 2 uses $L = 96$ and $H \in \{96, 192, 336, 720\}$. Table 8 lists the dataset-specific configurations for this protocol. Baseline reruns share dataset-specific training configurations with Adam, zero weight decay, cosine learning-rate scheduling, and early-stopping patience five. Model configurations retain the settings required by their respective architectures. Our entries use retained runs matching the fixed core configuration for each dataset and horizon. The uniform protocol refers to the baseline configurations, and is separate from the uniform candidate selection in the recent-branch ablation. DUET-CI is the evaluated DUET (Qiu et al., 2025) variant without its channel-clustering attention module. Appendix C provides completion details. The non-Solar comparison contains 320 completed baseline runs across eight datasets. For Solar-Energy at $H = 720$, TimePerceiver training was stopped after three completed epochs, during epoch four. Its reported result uses the checkpoint with the lowest validation MAE (epoch one), before the early-stopping patience was reached.

A.4 Variability Across Random Seeds

We repeat the PatchBasis and PatchTST experiments on seven datasets with seeds 2024, 2025, and 2026, using input length $L = 96$ and prediction length $H = 96$. Each method uses a fixed configuration for each dataset across the three runs, and checkpoints are selected by validation MAE. Table 9 reports the mean and two-sided 99% confidence interval for test MSE and MAE across these runs. Each interval is $\bar{x} \pm t_{0.995,2} s/\sqrt{3}$, where s is the sample standard deviation and $t_{0.995,2} \approx 9.925$. The intervals describe uncertainty in the mean across random seeds.

Dataset	d_{model}	d_{ff}	Layers	Heads	LR	P/S	Batch	Max. epochs
ETTm1	128	512	2	8	10^{-3}	12/12	32	50
ETTm2	64	256	1	4	10^{-3}	16/16	32	30
ETTh1	16	64	2	8	10^{-2}	2/2	32	50
ETTh2	512	2048	1	8	10^{-5}	12/12	32	50
Weather	128	512	1	8	10^{-3}	12/12	32	50
Electricity	512	2048	2	8	10^{-2}	24/24	16	50
Traffic	512	2048	1	8	10^{-4}	24/12	4	50
Solar	512	2048	3	8	10^{-3}	2/2	32	50
Exchange	512	2048	3	8	10^{-5}	2/2	32	50

Table 8: Dataset-specific configurations for the uniform baseline protocol in Table 2. Layers and Heads denote encoder depth and attention-head count; LR is the learning rate, and P/S is patch length/stride. Max. epochs is the training cap before early stopping.

Table 9: Forecasting results across three random seeds ($L = 96$, $H = 96$). Entries report the mean $\pm$ the half-width of its two-sided 99% Student-t confidence interval.

Dataset	PatchBasis		PatchTST		Confidence level
	MSE	MAE	MSE	MAE	
ETTh1	0.367 ± 0.004	0.383 ± 0.003	0.380 ± 0.046	0.393 ± 0.012	99%
ETTh2	0.290 ± 0.012	0.332 ± 0.008	0.284 ± 0.003	0.331 ± 0.001	99%
ETTm1	0.308 ± 0.003	0.338 ± 0.005	0.315 ± 0.015	0.342 ± 0.012	99%
ETTm2	0.172 ± 0.006	0.250 ± 0.002	0.175 ± 0.006	0.252 ± 0.003	99%
Weather	0.164 ± 0.008	0.199 ± 0.009	0.172 ± 0.005	0.205 ± 0.004	99%
Solar-Energy	0.202 ± 0.011	0.221 ± 0.023	0.247 ± 0.029	0.287 ± 0.050	99%
Electricity	0.161 ± 0.036	0.256 ± 0.018	0.239 ± 0.086	0.322 ± 0.090	99%

A.5 Ablation Protocols

Instance-adaptive patch aggregation. Table 3 uses $L = 320$ and $H = 96$ on ETTh1, ETTh2, ETTm1, ETTm2, and Weather. BasisLearner retains its local convolution, positional embeddings, and slot embeddings, with the time-decay branch disabled. TimeBase uses its native linear temporal basis without orthogonality loss. Each tokenizer feeds a flattening linear forecasting head, with the applicable embedding and Transformer components specified in Appendix D. Validation MAE selects both checkpoints and configurations. Each selected configuration is evaluated with three random seeds. Test errors and computational costs are averaged across these runs. Appendix D gives the method-specific search spaces, selected settings, and measurements.

Recent-candidate ablation. Figure 4 and Tables 10 and 4 use $L = 512$ and $H = 192$ on five datasets. We evaluate each of the 15 fixed configurations with three random seeds and report the mean results. Dataset-specific configurations are inherited from the corresponding search-based forecasting runs and then held fixed across all three variants. The recent and uniform variants use the same candidate count K and merge mechanism, differing in candidate positions. Removing the branch retains adaptive patch aggregation alone. The recent-weight ratio divides its mass on the latest K positions by the uniform reference K/M.

A.6 Parameter Sensitivity and Backbone Transfer

Parameter sensitivity. Sensitivity figures report means over seeds 2024, 2025, and 2026 with $H = 96$. Appendix E lists the compression, patch-length, and input-length grids and the MAE results.

Transfer across backbones. The backbone comparisons use $L = 320$ and $H = 96$ on ETTh1, ETTh2, ETTm1, ETTm2, and Weather. The time-decay branch is disabled throughout this comparison. Within each dataset–backbone–method group, configurations are selected by validation MAE over the applicable patch lengths, token counts, learning rates, and embedding widths. Each selected configuration is evaluated with three random seeds. The embedded and Transformer experiments

Dataset	Recent			w/o recent			Uniform		
	MSE	MAE	Score	MSE	MAE	Score	MSE	MAE	Score
ETTm1	**0.325**	**0.356**	**0.340**	0.329	0.357	0.343	0.345	0.370	0.358
ETTm2	**0.217**	**0.283**	**0.250**	0.223	0.292	0.257	**0.217**	0.285	0.251
ETTh1	**0.384**	**0.411**	**0.398**	0.399	0.418	0.408	0.415	0.430	0.422
ETTh2	0.350	0.381	0.366	0.354	0.380	0.367	**0.340**	**0.375**	**0.358**
Weather	**0.184**	**0.222**	**0.203**	0.187	0.223	0.205	0.187	0.223	0.205
Avg.	**0.292**	**0.331**	**0.311**	0.298	0.334	0.316	0.301	0.337	0.319

Table 10: Full recent-prior ablation results ($L = 512$, $H = 192$). Entries report means over three seeds. Recent and Uniform select the latest and evenly spaced K candidates; w/o recent disables the branch. Score is $(\mathrm{MSE} + \mathrm{MAE})/2$. Avg. equally weights datasets. Rankings use three-decimal values, with ties sharing the same rank.

search $D \in \{16, 128\}$ jointly with the other parameters. Appendix D provides backbone definitions, method-specific search constraints, selected configurations, and efficiency measurement settings.

Plugin transfer. Table 5 compares native forecasters and the aggregation plugin with $L = 192$ and $H = 96$ on ETTh1, ETTm1, and Weather. Configurations are selected by validation MAE and then re-evaluated with three random seeds. The table reports their mean results. Appendix D gives the configurations and computational costs.

B FULL FORECASTING RESULTS

Models		PatchBasis **Ours**		TimeMosaic (2026)		SimpleTM (2025)		TimeFilter (2025)		DUET (2025)		iTransformer (2024)		TimeMixer (2024)		PatchTST (2022)		DLinear (2023)	
	H	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.274**	**0.320**	0.282	0.328	0.280	0.337	0.290	0.342	0.300	0.345	0.299	0.348	0.280	0.340	0.289	0.343	0.299	0.343
	192	**0.315**	**0.350**	0.318	0.354	0.326	0.365	0.334	0.368	0.335	0.366	0.334	0.373	0.347	0.387	0.332	0.367	0.336	0.367
	336	**0.351**	**0.370**	0.356	0.374	0.362	0.387	0.362	0.384	0.364	0.383	0.370	0.394	0.366	0.405	0.362	0.389	0.367	0.385
	720	**0.396**	**0.406**	0.411	0.412	0.417	0.422	0.425	0.415	0.417	0.413	0.422	0.426	0.403	0.432	0.408	0.420	0.420	0.418
	Avg.	**0.334**	**0.362**	0.342	0.367	0.346	0.378	0.353	0.377	0.354	0.377	0.356	0.385	0.349	0.392	0.348	0.380	0.356	0.378
ETTm2	96	**0.157**	**0.241**	0.161	0.245	0.168	0.252	0.161	0.253	0.162	0.253	0.180	0.263	0.164	0.254	0.165	0.257	0.165	0.260
	192	**0.210**	**0.279**	0.216	0.284	0.223	0.297	0.221	0.296	0.217	0.290	0.236	0.307	0.223	0.295	0.223	0.294	0.219	0.297
	336	**0.265**	**0.317**	0.269	0.318	0.273	0.328	0.272	0.329	0.270	0.325	0.287	0.340	0.279	0.330	0.272	0.333	0.290	0.345
	720	**0.344**	**0.371**	0.354	0.373	0.364	0.386	0.359	0.391	0.361	0.384	0.364	0.391	0.359	0.383	0.363	0.386	0.375	0.403
	Avg.	**0.244**	**0.302**	0.250	0.305	0.257	0.316	0.253	0.317	0.253	0.313	0.267	0.325	0.256	0.315	0.256	0.318	0.262	0.326
ETTh1	96	**0.348**	**0.381**	0.359	0.384	0.375	0.394	0.377	0.397	0.364	0.392	0.380	0.401	0.375	0.398	0.370	0.397	0.368	0.393
	192	**0.381**	**0.407**	0.396	0.411	0.412	0.429	0.413	0.419	0.397	0.413	0.423	0.431	0.407	0.425	0.404	0.423	0.400	0.417
	336	**0.401**	**0.419**	0.413	0.428	0.429	0.443	0.425	0.438	0.422	0.432	0.437	0.450	0.431	0.435	0.423	0.437	0.430	0.441
	720	0.425	**0.441**	**0.419**	0.445	0.447	0.472	0.442	0.456	0.443	0.463	0.460	0.471	0.452	0.456	0.444	0.464	0.477	0.497
	Avg.	**0.389**	**0.412**	0.397	0.417	0.416	0.434	0.414	0.429	0.406	0.425	0.425	0.438	0.416	0.428	0.410	0.430	0.419	0.437
ETTh2	96	0.269	**0.328**	0.282	0.329	0.291	0.339	0.281	0.343	**0.265**	0.334	0.294	0.345	0.280	0.340	0.290	0.342	0.286	0.351
	192	0.335	**0.367**	0.342	0.376	0.349	0.390	0.333	0.383	**0.324**	0.373	0.358	0.395	0.348	0.388	0.354	0.397	0.352	0.394
	336	0.361	**0.393**	0.370	0.397	0.381	0.413	0.368	0.411	**0.352**	0.399	0.385	0.415	0.366	0.405	0.384	0.413	0.439	0.456
	720	**0.389**	**0.424**	0.400	0.428	0.410	0.442	0.407	0.440	0.393	0.432	0.412	0.436	0.404	0.433	0.408	0.442	0.570	0.530
	Avg.	0.338	**0.378**	0.348	0.383	0.358	0.396	0.347	0.394	**0.334**	0.384	0.362	0.398	0.349	0.392	0.359	0.399	0.412	0.433
Weather	96	0.144	**0.180**	0.147	0.184	0.143	0.195	**0.142**	0.195	0.168	0.221	0.155	0.206	0.147	0.197	0.145	0.194	0.170	0.229
	192	**0.186**	**0.222**	0.192	0.228	0.188	0.236	**0.186**	0.238	0.212	0.258	0.199	0.249	0.190	0.240	0.191	0.237	0.212	0.268
	336	**0.235**	**0.261**	0.240	0.270	0.238	0.276	0.237	0.278	0.258	0.292	0.247	0.284	0.253	0.285	0.243	0.279	0.258	0.307
	720	**0.308**	**0.316**	0.315	0.324	0.312	0.327	0.313	0.330	0.324	0.338	0.318	0.336	0.326	0.339	0.313	0.330	0.321	0.359
	Avg.	**0.218**	**0.245**	0.223	0.251	0.220	0.259	0.220	0.261	0.240	0.277	0.230	0.269	0.229	0.265	0.223	0.260	0.240	0.291
1st Count		**44**		1		0		2		4		0		0		0		0	

Table 11: Long-term forecasting results on five datasets with prediction horizons $H \in \{96, 192, 336, 720\}$. Avg. averages the four horizons. Rankings use three-decimal values.

C FULL RESULTS UNDER THE UNIFORM BASELINE PROTOCOL

Table 12 reports the horizon-level results corresponding to Table 2. Baselines use unified dataset-specific configurations. PatchBasis uses retained runs matching its fixed core configuration for each dataset and horizon, including finite results from resumed runs, and was not rerun with the baselines. Solar results include supplementary reruns, except for TimeFilter (Hu et al., 2025), whose Solar entries use its reported results. All model–dataset pairs have results for all four horizons. Each reported average uses these four results; the TimePerceiver training exception is described in Appendix A.

Models		PatchBasis (**Ours**, CI)		TimeMosaic (2026, CI)		SEER (2026, CI)		TimeFilter (2025, CD)		DUET-CI (2025, CI)		TimeBase (2025, CI)		SRSNet (2026, CI)		TimePerceiver (2026, CD)		iTransformer (2024, CD)		PatchTST (2022, CI)		DLinear (2023, CI)	
	H	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.307**	**0.337**	0.316	0.344	0.312	0.343	0.312	0.346	0.338	0.355	0.374	0.379	0.312	0.346	0.330	0.350	0.313	0.347	0.312	0.340	0.332	0.352
	192	**0.351**	**0.365**	0.368	0.372	0.372	0.374	0.424	0.395	0.382	0.378	0.410	0.397	0.384	0.379	0.375	0.374	0.380	0.382	0.370	0.370	0.378	0.377
	336	**0.384**	**0.388**	0.398	0.393	0.395	0.395	0.405	0.400	0.415	0.399	0.438	0.418	0.400	0.397	0.399	0.393	0.420	0.407	0.393	0.391	0.409	0.399
	720	**0.448**	**0.427**	0.465	0.431	0.482	0.447	0.474	0.435	0.476	0.433	0.500	0.449	0.477	0.434	0.463	0.428	0.510	0.455	0.455	0.434	0.471	0.437
	Avg.	**0.372**	**0.379**	0.387	0.385	0.390	0.390	0.404	0.394	0.403	0.391	0.430	0.411	0.393	0.389	0.392	0.386	0.406	0.398	0.382	0.384	0.397	0.391
ETTm2	96	0.171	0.249	0.176	0.252	0.175	0.251	0.169	0.250	0.183	0.259	0.220	0.295	0.180	0.260	**0.168**	**0.244**	0.175	0.252	0.176	0.252	0.184	0.263
	192	0.236	0.292	0.240	0.295	0.242	0.296	0.237	0.294	0.246	0.300	0.276	0.328	0.244	0.300	**0.235**	**0.289**	0.241	0.296	0.240	0.296	0.248	0.309
	336	0.294	0.331	0.299	0.333	0.299	0.333	**0.292**	0.331	0.307	0.337	0.332	0.361	0.306	0.339	0.293	**0.329**	0.302	0.335	0.299	0.334	0.308	0.341
	720	0.392	0.389	0.399	0.392	0.402	0.393	0.391	0.389	0.407	0.395	0.428	0.412	0.407	0.397	**0.388**	**0.387**	0.404	0.396	0.404	0.395	0.407	0.408
	Avg.	0.273	0.315	0.278	0.318	0.280	0.318	0.272	0.316	0.286	0.323	0.314	0.349	0.284	0.324	**0.271**	**0.312**	0.281	0.320	0.280	0.319	0.287	0.330
ETTh1	96	**0.378**	**0.387**	0.391	0.399	0.409	0.413	0.408	0.405	0.386	0.396	0.411	0.416	0.387	0.391	0.396	0.399	0.390	0.401	**0.378**	0.392	0.430	0.435
	192	**0.428**	**0.415**	0.430	0.422	0.442	0.432	0.466	0.432	0.435	0.424	0.450	0.437	0.435	0.419	0.743	0.536	0.438	0.430	0.441	0.424	0.474	0.460
	336	**0.451**	**0.433**	0.466	0.442	0.469	0.440	0.485	0.444	0.480	0.446	0.490	0.454	0.499	0.455	0.736	0.543	0.488	0.453	0.478	0.447	0.508	0.477
	720	**0.441**	**0.447**	0.468	0.460	0.521	0.475	0.532	0.483	0.472	0.461	0.484	0.471	0.472	0.464	0.706	0.566	0.488	0.479	0.486	0.473	0.555	0.535
	Avg.	**0.424**	**0.420**	0.438	0.431	0.460	0.440	0.473	0.441	0.443	0.432	0.459	0.445	0.448	0.432	0.645	0.511	0.451	0.441	0.446	0.434	0.492	0.477
ETTh2	96	**0.281**	**0.329**	0.284	0.330	0.296	0.335	0.286	0.335	0.288	0.334	0.388	0.422	0.284	0.332	**0.281**	**0.329**	0.287	0.335	0.284	0.331	0.299	0.349
	192	0.362	**0.380**	**0.360**	0.381	0.363	0.381	0.369	0.386	0.375	0.388	0.456	0.455	0.373	0.387	0.363	0.382	0.365	0.385	0.365	0.382	0.387	0.407
	336	**0.406**	**0.415**	**0.406**	0.416	0.408	0.418	0.410	0.422	0.419	0.425	0.493	0.491	0.418	0.423	0.414	0.420	0.418	0.424	0.414	0.421	0.454	0.456
	720	0.415	0.433	**0.413**	**0.432**	0.430	0.439	0.430	0.441	0.422	0.438	0.490	0.497	0.420	0.437	0.427	0.435	0.420	0.438	0.419	0.435	0.612	0.550
	Avg.	**0.366**	**0.389**	**0.366**	0.390	0.374	0.393	0.374	0.396	0.376	0.396	0.457	0.466	0.374	0.395	0.371	0.392	0.372	0.396	0.371	0.392	0.438	0.440
Weather	96	0.162	0.198	0.176	0.209	0.165	0.203	**0.153**	0.192	0.201	0.228	0.230	0.248	0.166	0.204	0.154	**0.189**	0.174	0.207	0.172	0.205	0.207	0.233
	192	0.208	0.241	0.225	0.252	0.215	0.245	**0.203**	**0.240**	0.244	0.263	0.271	0.280	0.208	0.242	0.210	0.243	0.223	0.251	0.219	0.247	0.245	0.270
	336	0.264	**0.282**	0.278	0.290	0.272	0.290	**0.263**	0.284	0.294	0.300	0.320	0.315	0.264	0.283	0.264	0.283	0.281	0.293	0.275	0.287	0.290	0.314
	720	**0.341**	**0.334**	0.357	0.341	0.349	0.340	0.344	0.338	0.365	0.346	0.390	0.360	0.343	0.335	0.346	0.336	0.356	0.344	0.353	0.339	0.347	0.357
	Avg.	0.244	**0.263**	0.259	0.273	0.250	0.269	**0.241**	**0.263**	0.276	0.284	0.303	0.301	0.245	0.266	0.243	**0.263**	0.258	0.274	0.255	0.269	0.272	0.293
Electricity	96	**0.156**	**0.248**	0.190	0.276	0.334	0.400	0.221	0.305	0.208	0.289	0.214	0.278	0.218	0.310	0.336	0.400	0.359	0.413	0.233	0.316	0.205	0.278
	192	**0.169**	**0.259**	0.262	0.349	0.413	0.460	0.201	0.286	0.206	0.290	0.212	0.280	0.250	0.336	0.380	0.442	0.329	0.393	0.234	0.319	0.201	0.283
	336	**0.186**	**0.275**	0.212	0.297	0.870	0.763	0.199	0.277	0.234	0.322	0.224	0.295	0.261	0.344	0.252	0.336	0.359	0.417	0.248	0.334	0.216	0.299
	720	**0.226**	**0.309**	0.344	0.415	0.899	0.773	0.344	0.416	0.963	0.748	0.266	0.327	0.305	0.375	0.960	0.770	0.397	0.443	0.290	0.366	0.254	0.335
	Avg.	**0.184**	**0.273**	0.252	0.334	0.629	0.599	0.241	0.321	0.403	0.412	0.229	0.295	0.258	0.341	0.482	0.487	0.361	0.416	0.251	0.334	0.219	0.299
Traffic	96	0.459	0.279	0.452	0.270	**0.396**	**0.243**	0.421	0.257	0.663	0.378	0.787	0.396	0.476	0.291	0.419	**0.243**	0.407	0.256	0.462	0.274	0.683	0.364
	192	0.469	0.288	0.465	0.275	**0.427**	**0.252**	0.443	0.267	0.618	0.356	0.713	0.375	0.481	0.292	0.434	0.255	**0.427**	0.264	0.470	0.278	0.643	0.340
	336	0.486	0.292	0.483	0.282	**0.444**	**0.261**	0.463	0.277	0.626	0.359	0.714	0.380	0.496	0.302	0.478	0.279	**0.444**	0.271	0.488	0.284	0.645	0.343
	720	0.524	0.309	0.519	0.301	0.478	**0.280**	0.494	0.294	0.664	0.377	0.754	0.397	0.535	0.318	0.497	0.289	**0.475**	0.289	0.523	0.302	0.673	0.362
	Avg.	0.484	0.292	0.480	0.282	**0.436**	**0.259**	0.455	0.273	0.643	0.367	0.742	0.387	0.497	0.301	0.457	0.266	0.438	0.270	0.486	0.284	0.661	0.352
Solar	96	**0.202**	0.224	0.271	0.275	0.985	0.583	0.222	0.253	0.296	0.294	0.437	0.427	0.246	0.268	0.213	**0.206**	0.278	0.285	0.253	0.296	0.286	0.300
	192	**0.227**	**0.241**	0.263	0.267	0.263	0.247	0.256	0.266	0.336	0.317	0.486	0.445	0.301	0.294	0.289	0.245	0.319	0.303	0.294	0.311	0.320	0.318
	336	**0.247**	0.257	0.381	0.350	0.458	0.400	0.254	0.283	0.377	0.330	0.525	0.468	0.330	0.312	0.262	**0.253**	0.343	0.320	0.292	0.311	0.361	0.330
	720	**0.248**	**0.254**	0.348	0.307	0.755	0.576	0.258	0.280	0.380	0.325	0.540	0.472	0.329	0.316	0.288	0.266	0.278	0.265	0.351	0.338	0.372	0.326
	Avg.	**0.231**	0.244	0.316	0.300	0.615	0.451	0.248	0.270	0.347	0.317	0.497	0.453	0.302	0.297	0.263	**0.243**	0.305	0.294	0.297	0.314	0.335	0.319
Exchange	96	**0.083**	**0.199**	0.085	0.202	0.086	0.204	0.089	0.209	**0.083**	**0.199**	0.267	0.407	0.084	0.201	0.086	0.203	0.088	0.209	0.085	0.203	0.090	0.216
	192	**0.174**	0.296	0.180	0.301	0.182	0.302	0.195	0.313	**0.174**	**0.295**	0.392	0.493	0.181	0.302	0.182	0.302	0.184	0.305	0.180	0.303	0.214	0.349
	336	0.328	0.413	0.334	0.418	0.346	0.426	0.370	0.437	0.325	**0.411**	0.504	0.549	0.346	0.423	0.351	0.429	0.351	0.429	0.364	0.438	**0.312**	0.421
	720	0.857	0.698	0.886	0.712	0.884	0.710	1.019	0.759	0.845	0.693	1.052	0.803	0.884	0.709	0.893	0.720	0.885	0.715	0.843	0.691	**0.703**	**0.641**
	Avg.	0.361	0.401	0.371	0.408	0.375	0.410	0.418	0.430	0.357	**0.400**	0.554	0.563	0.374	0.409	0.378	0.414	0.377	0.414	0.368	0.409	**0.330**	0.406

Table 12: Per-horizon results under the uniform baseline protocol ($L = 96$). Models are ordered from newest to oldest. Rankings use three-decimal values. Each average uses all four horizons. Solar entries for TimeFilter (Hu et al., 2025) use its reported results, not a rerun.

D TOKENIZATION ACROSS BACKBONES

Backbones and tokenizers. Table 3 compares Full patch, TimeBase, APE, TOTEM, and BasisLearner on ETTh1, ETTh2, ETTm1, ETTm2, and Weather. BasisLearner is evaluated without the time-decay branch. All configurations use a flattening linear forecasting head. Raw removes the additional patch-to-embedding projection and the external Transformer; TOTEM retains its native 64-dimensional latent tokens. Embedding uses learned representations without an external Transformer. Transformer adds an encoder with depth two on ETTh1, depth three on Weather, and depth one on the other datasets, fixed across tokenizers within each dataset. It uses eight attention heads, feed-forward width $4D$, GELU, and dropout 0.1.

Full patch retains all patches and uses the local convolutional encoder. TimeBase uses its native shared linear temporal basis, without its orthogonality loss. APE uses TimeMosaic's adaptive patch embedding with patch lengths $\{8, 16, 32\}$, 32-point regions, router temperature 0.5, and balancing-loss weight 0.001. TOTEM uses its frozen pretrained forecasting tokenizer, with compression factor four, code dimension 64, and 256 codebook entries. An adapter projects its tokens to width D for Embedding and Transformer. BasisLearner retains local convolution, positional embeddings, and slot embeddings. All methods use RevIN; TOTEM retains its non-affine normalization.

Search and selection. All runs use $L = 320$ and $H = 96$. Embedding width $D \in \{16, 128\}$ is selected jointly with the applicable tokenizer hyperparameters. Full patch searches patch-length/stride pairs $(8, 8)$, $(16, 16)$, $(24, 24)$, and $(48, 24)$. TimeBase searches the non-overlapping pairs among these and $K \in \{6, 12\}$. APE and TOTEM retain their native tokenizer settings. All methods search learning rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$. The method-specific search spaces therefore differ in size. Raw inherits the applicable grids without a width search; APE is inapplicable to Raw.

Training uses MAE loss, Adam, zero weight decay, batch size 32, cosine learning-rate decay, up to 50 epochs, and early-stopping patience five. Checkpoints and configurations are selected by validation MAE. For each dataset–backbone–tokenizer group, we first select the configuration and then repeat its evaluation with three random seeds. Table 3 reports the mean test MSE and MAE. Computational costs are also averaged across the three random seeds. No configuration is reselected for latency, memory, or parameter count.

Computational costs. Cost measurements are averaged across three random seeds for each selected configuration. Table 6 reports the equal-dataset means of the 70 configurations in Table 3. Total parameter counts include frozen parameters; trainable counts exclude them. Measurements use an NVIDIA Tesla V100S-PCIE-32GB in FP32 with batch size 32, five warm-up iterations, and 20 measured iterations. Inference time is per batch; training time includes the forward pass, backward pass, and Adam update. Data loading is excluded. Concurrent jobs may affect the recorded latency. Peak memory is the process-level peak CUDA allocation during inference, including the resident model and profiling copy, rather than an isolated model-only memory requirement. MACs and training peak memory were not recorded for these runs.

The selected settings can differ in patch length, token count, and width. These costs therefore describe the accuracy-selected configurations, rather than models constrained to equal parameter or token counts.

Plugin configurations and costs. Tables 15 and 16 report the settings and measurements from the same runs as Table 5. Dataset-level accuracy is reported in Table 14. Computational costs are averaged equally over ETTh1, ETTm1, and Weather. Each selected native and plugin configuration is re-evaluated with three random seeds. Accuracy and cost measurements are averaged across these runs.

Tokenizer	MSE	MAE	Params (K)	Trainable (K)	Infer. (ms)	Train step (ms)	Peak mem. (MiB)
			Raw				
Full patch	0.253	0.300	38.592	38.592	0.740	2.712	14.006
TimeBase	0.257	0.304	15.130	15.130	0.586	2.303	11.819
TOTEM	0.261	0.310	664.288	491.616	2.002	2.774	143.088
BasisLearner	0.252	0.299	32.646	32.646	1.285	4.237	19.139
			Embedding				
Full patch	0.249	0.297	291.543	291.543	0.783	3.029	33.019
TimeBase	0.255	0.302	94.176	94.176	0.565	2.276	17.071
APE	0.250	0.299	152.029	152.029	5.107	21.135	18.735
TOTEM	0.256	0.305	643.664	470.992	2.109	3.183	142.656
BasisLearner	0.246	0.293	172.957	172.957	1.316	4.118	41.973
			Transformer				
Full patch	0.251	0.297	321.031	321.031	1.456	5.964	37.995
TimeBase	0.246	0.294	423.448	423.448	1.218	5.833	28.431
APE	0.251	0.299	448.979	448.979	5.724	24.586	52.469
TOTEM	0.256	0.302	766.029	593.357	3.979	11.671	147.509
BasisLearner	0.245	0.292	229.374	229.374	1.881	7.981	33.162

Table 13: Complete accuracy and computational costs for the configurations selected for Table 3. Accuracy is averaged over five datasets, and costs over three seeds. Timings use a V100S GPU, FP32, and batch size 32.

Dataset	Native		+ plugin		Improv. (%)	
	MSE	MAE	MSE	MAE	MSE	MAE
			PatchTST			
ETTh1	0.373	0.388	**0.366**	**0.387**	+1.998	+0.412
ETTm1	0.300	0.337	**0.289**	**0.329**	+3.561	+2.506
Weather	0.158	0.194	**0.157**	**0.190**	+0.600	+1.690
Avg.	0.277	0.306	**0.271**	**0.302**	+2.297	+1.450
			PatchTSMixer			
ETTh1	0.373	**0.383**	**0.367**	0.384	+1.558	-0.294
ETTm1	**0.291**	**0.328**	0.302	0.334	-3.657	-1.860
Weather	**0.154**	**0.188**	0.158	0.192	-2.670	-2.336
Avg.	**0.273**	**0.299**	0.276	0.303	-1.095	-1.293
			PatchMixer			
ETTh1	**0.361**	**0.384**	0.370	0.387	-2.384	-0.764
ETTm1	**0.284**	**0.328**	0.291	0.330	-2.611	-0.650
Weather	**0.152**	0.190	0.154	**0.188**	-1.089	+1.078
Avg.	**0.266**	**0.301**	0.272	0.302	-2.217	-0.335
			TimeXer			
ETTh1	0.371	0.391	**0.369**	**0.386**	+0.541	+1.283
ETTm1	0.296	**0.329**	**0.295**	0.332	+0.340	-1.055
Weather	**0.149**	**0.187**	**0.149**	**0.187**	-0.588	-0.221
Avg.	0.272	**0.302**	**0.271**	**0.302**	+0.263	+0.125

Table 14: Dataset-level native and aggregation-plugin results ($L = 192$, $H = 96$). Entries report means over three seeds. Positive improvement indicates lower error. Bold marks the lower displayed error in each native/plugin pair.

Dataset	Backbone	Variant	P/S	D	Depth	LR	N	K	Val. MAE
ETTh1	PatchTST	Native	8/8	16	1	10^{-2}	25	25	0.538
		+ plugin	2/2	16	1	10^{-2}	97	12	0.538
	PatchTSMixer	Native	48/48	128	1	10^{-4}	4	4	0.536
		+ plugin	4/4	16	2	10^{-2}	48	16	0.535
	PatchMixer	Native	16/16	128	1	10^{-4}	13	13	0.537
		+ plugin	8/8	128	1	10^{-3}	25	4	0.536
	TimeXer	Native	8/8	16	2	10^{-2}	24	24	0.543
		+ plugin	4/4	16	1	10^{-2}	48	16	0.537
ETTm1	PatchTST	Native	8/8	16	2	10^{-2}	25	25	0.391
		+ plugin	24/12	16	2	10^{-2}	16	8	0.387
	PatchTSMixer	Native	8/8	16	2	10^{-3}	24	24	0.380
		+ plugin	4/4	128	1	10^{-3}	48	12	0.382
	PatchMixer	Native	16/16	16	1	10^{-3}	13	13	0.386
		+ plugin	2/2	128	1	10^{-4}	97	16	0.386
	TimeXer	Native	8/8	16	1	10^{-2}	24	24	0.382
		+ plugin	16/16	128	1	10^{-3}	12	8	0.383
Weather	PatchTST	Native	24/24	128	2	10^{-4}	9	9	0.261
		+ plugin	8/8	16	2	10^{-3}	25	12	0.257
	PatchTSMixer	Native	8/8	128	2	10^{-4}	24	24	0.253
		+ plugin	16/16	128	1	10^{-4}	12	6	0.255
	PatchMixer	Native	16/16	16	1	10^{-4}	13	13	0.249
		+ plugin	2/2	128	1	10^{-4}	97	4	0.246
	TimeXer	Native	16/16	128	1	10^{-4}	12	12	0.245
		+ plugin	8/8	128	2	10^{-4}	24	16	0.244

Table 15: Configurations used for every accuracy and cost entry in the plugin comparison. P/S: patch length/stride; D: embedding width; Depth: backbone depth; LR: learning rate; N and K: input and retained patch tokens. TimeXer additionally retains its global token, excluded from N/K.

Backbone	Variant	Params (K)	Trainable (K)	Infer. (ms)	Train step (ms)	Infer. mem. (MiB)	Train mem. (MiB)	Profiled FLOPs (M, partial)
PatchTST	Native	199.836	199.835	1.151	6.188	21.115	27.393	53.915
	+ plugin	27.181	27.180	2.071	9.883	19.585	64.417	5.430
PatchTSMixer	Native	287.133	287.133	1.010	5.151	25.200	44.781	126.993
	+ plugin	217.607	217.607	1.856	8.517	22.836	30.289	33.618
PatchMixer	Native	219.777	219.777	0.611	3.606	20.716	54.058	3.985
	+ plugin	359.280	359.280	1.566	7.013	34.724	85.613	41.943
TimeXer	Native	183.675	183.675	1.827	9.891	24.417	30.875	17.847
	+ plugin	433.460	433.460	2.733	13.430	31.587	48.921	52.850

Table 16: Computational costs of the same configurations as Table 5, averaged over three seeds and equally over ETTh1, ETTm1, and Weather. Timing uses an A100-SXM4-80GB, FP32, and batch size four. Memory is recorded peak CUDA allocation. Profiled FLOPs count supported matrix-multiplication and convolution operators only, not total model FLOPs. Settings are selected for validation MAE, not computational cost.

E Additional Parameter Sensitivity Results

Figure 6 reports MAE for the sweeps in Figure 5. Each point averages three seeds with prediction length $H = 96$. The compression sweep fixes $L = 2880$ and $P = S = 24$, varying $K \in \{16, 8, 4, 2, 1\}$ to obtain $M/K \in \{7.5, 15, 30, 60, 120\}$. The patch-length sweep fixes $L = 720$ and $K = 4$, varying $P = S \in \{8, 12, 16, 24, 48\}$. The input-length sweep fixes $P = S = 24$ and $K = 4$ for our method and TimeBase, with $L \in \{96, 168, 672, 2880, 8832\}$; PatchTST uses its patch length 16 and stride 8.

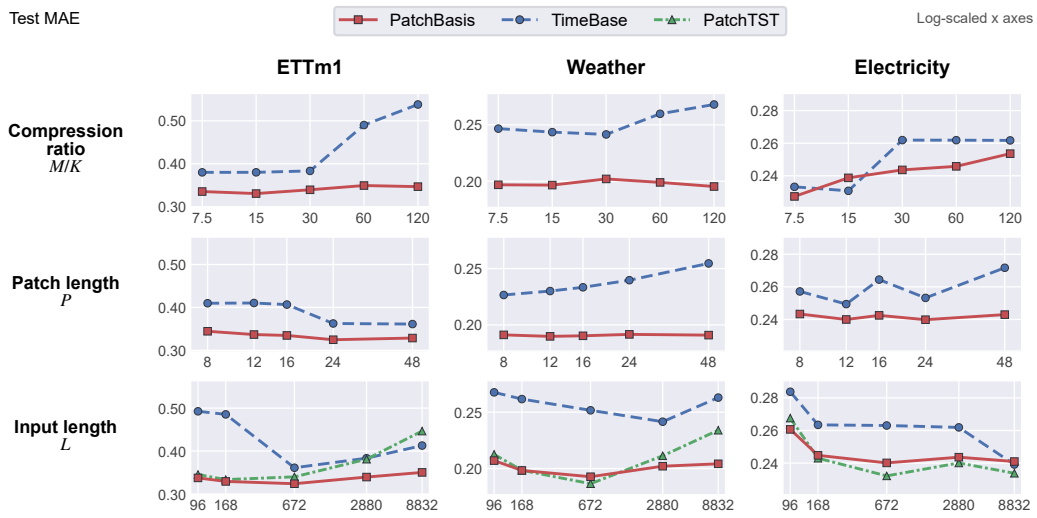

Figure 6: Mean test MAE over three seeds for the parameter sweeps in Figure 5, with $H = 96$ and logarithmic horizontal axes.

The MAE results confirm lower sensitivity to parameter choices. Our maximum-minus-minimum MAE range is smaller than TimeBase's in all nine dataset–parameter combinations and smaller than PatchTST's in all three input-length sweeps. Absolute MAE is also lower than TimeBase's except at two Electricity settings: compression ratio $M/K = 15$ and input length $L = 8832$. PatchTST attains lower errors at several input lengths on Weather and Electricity.

F Dynamic Recent-Budget Allocation

This appendix reports a diagnostic study of an input-dependent budget for the time-decay prior. The study is separate from the main forecasting evaluation and from the recent, uniform, and no-recent ablation in Figure 4. It asks whether predicting the effective amount of recent history for each input improves over a fixed budget, and whether any gain depends on matching the predicted budget to the input that produced it.

F.1 Budget Parameterization

The budget controls the center of the sigmoid decay rather than selecting a hard number of recent patches. For recent candidates indexed from newest to oldest, the weight of slot k under budget b is

$$\mu_k(b) = \mathrm{sigmoid}\left(\frac{b - k + 1/2}{\tau_a}\right), \qquad k = 1, \ldots, K, \tag{5}$$

where $\tau_a = 1$ in this study. A larger b assigns higher weights to more recent candidates, while the final operation remains the soft merge in Equation 4. The budget therefore provides a continuous approximation to the number of recent patches retained by the recent branch.

We compare three training modes. **No recent** removes the recent branch and retains only the aggregated BasisLearner tokens. **Fixed** uses one budget b_{fixed} for every input. For each dataset, five

candidate values are trained with seed 2025, and the value with the lowest validation MAE is selected. **Dynamic** predicts the budget from the current input:

$$b(\mathbf{x}) = b_{\min} + (b_{\max} - b_{\min}) \operatorname{sigmoid}(q(\mathbf{g}(\mathbf{x}))) , \tag{6}$$

where $\mathbf{g}(\mathbf{x})$ is a summary representation of the input and q is the budget-allocation MLP. The dynamic recent-slot weights are

$$\mu_k(\mathbf{x}) = \operatorname{sigmoid}\left(\frac{b(\mathbf{x}) - k + 1/2}{\tau_a} \right) . \tag{7}$$

The allocator operates channelwise. Channel indices are omitted above for clarity. This construction tests whether the effective extent of recent history should vary across forecasting instances.

Table 17: Candidate ranges for fixed-budget selection. Five values are trained with seed 2025 for each dataset, and validation MAE selects the fixed budget used in the three-seed evaluation.

Dataset	$[b_{\min}, b_{\max}]$	Fixed-budget candidates
ETTh1	$[2, 5]$	2, 2.75, 3.5, 4.25, 5
ETTh2	$[0.5, 1.5]$	0.5, 0.75, 1, 1.25, 1.5
ETTm1	$[1, 4]$	1, 1.75, 2.5, 3.25, 4
Weather	$[2, 5]$	2, 2.75, 3.5, 4.25, 5

F.2 Protocol and Inference Diagnostics

The study covers ETTh1, ETTh2, ETTm1, and Weather with input length $L = 192$ and prediction length $H = 96$. All tasks use multivariate input and multivariate prediction, MAE training loss, Adam with a cosine learning-rate schedule, at most 50 epochs, and early-stopping patience five. Checkpoints are selected by validation MAE. Final results average training seeds 2024, 2025, and 2026 and report MSE and MAE in normalized space.

Table 18 lists the inherited model configurations. Every configuration uses the local context encoder, token encoder, slot embeddings, RevIN, and a flattening prediction head.

Table 18: Model configurations for the dynamic-budget diagnostic. P/S denotes patch length and stride, M the input patch count, K the output token count, and d the embedding width.

Dataset	P/S	M	K	Transformer depth	d
ETTh1	$4/4$	48	8	1	128
ETTh2	$24/24$	8	2	1	128
ETTm1	$24/12$	15	6	2	128
Weather	$12/12$	16	8	2	16

Each dataset requires 13 training jobs. The seed-2025 search trains five fixed-budget candidates, one no-recent model, and one dynamic model. The selected fixed model, no-recent model, and dynamic model are then trained with seeds 2024 and 2026, adding six jobs. Across four datasets, the study therefore contains $4 \times 13 = 52$ training jobs.

The trained dynamic models are evaluated with three inference modes. In **own**, each test input uses its predicted channelwise budget vector. In **mean**, every test input uses the channelwise mean budget produced by the dynamic model on the training set. This comparison tests whether the allocator has only learned a strong global fixed budget. In **swap**, complete budget vectors are permuted across test inputs while preserving their channel structure. Results average permutation seeds 7101, 7102, and 7103. This comparison tests whether a predicted budget is useful for the specific input that produced it. Mean and swap change inference only and do not require additional training.

F.3 Results

Table 19 reports the three-seed averages. On ETTh1, the recent branch improves over no recent, but the selected fixed budget achieves the lowest MSE and MAE. Dynamic-own is only slightly better than dynamic-mean and dynamic-swap. On ETTh2, all modes are nearly indistinguishable, and

Table 19: Dynamic-budget diagnostic results averaged over training seeds 2024, 2025, and 2026. Entries are MSE/MAE in normalized space. Dynamic swap also averages permutation seeds 7101, 7102, and 7103. Lower is better.

Dataset	No recent	Fixed	Dynamic own	Dynamic mean	Dynamic swap
ETTh1	.3697/.3842	.3620/.3822	.3664/.3823	.3678/.3828	.3679/.3827
ETTh2	.2798/.3301	.2798/.3292	.2793/.3290	.2793/.3290	.2793/.3290
ETTm1	.2948/.3344	.2845/.3272	.2851/.3271	.2863/.3283	.2868/.3285
Weather	.1564/.1907	.1556/.1907	.1551/.1904	.1551/.1904	.1551/.1904

replacing each input's budget by a mean or permuted budget does not change the displayed dynamic result.

ETTm1 provides the clearest evidence that input-specific budgets carry information: dynamic-own outperforms dynamic-mean and dynamic-swap, although it remains close to the selected fixed budget. On Weather, dynamic inference slightly improves over fixed and no recent, but own, mean, and swap are identical at the reported precision. The improvement therefore cannot be attributed to matching a distinct budget to each test input.

Across datasets, the recent branch is useful on ETTh1, ETTm1, and Weather, and validation-selected fixed budgets form a strong baseline. The dynamic allocator does not consistently improve over fixed budgets. Only ETTm1 shows a clear, albeit small, advantage of own over mean and swap. On ETTh2 and Weather, averaging or permuting budgets has almost no effect, suggesting that the learned allocator behaves approximately as a fixed gate or that the predictor is insensitive to the observed budget variation. These results support the fixed time-decay prior used in the main method. The evaluated input-dependent extension does not provide stable gains.

The base architecture configurations are inherited unchanged from the main-table protocol described in Appendix A.3. Fixed-budget selection uses validation MAE. We therefore treat these results as a diagnostic comparison rather than a replacement for the main benchmark evaluation.

G Additional Noise-Response Cases

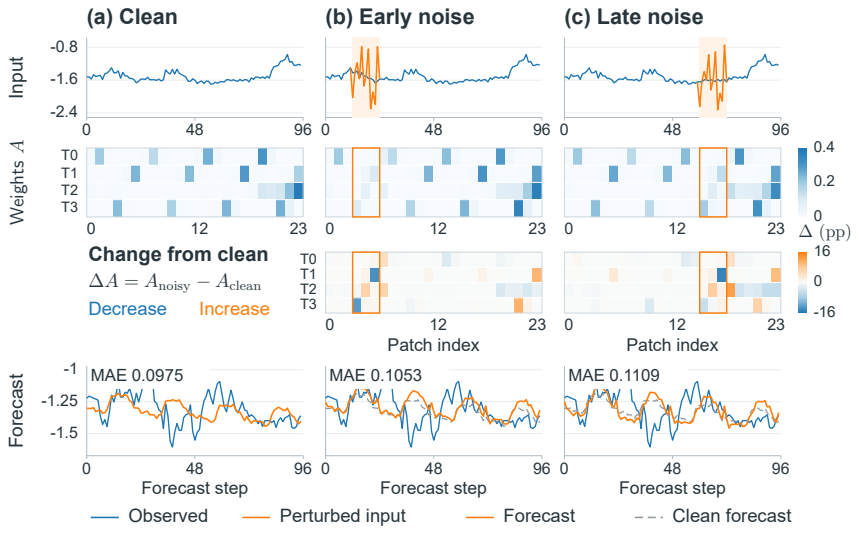

Figure 7: Input-dependent aggregation on ETTh1 (OT, $96 \rightarrow 96$). Columns: clean, early noise, and late noise. Rows: input, weights, weight changes (percentage points), and forecasts. Orange marks perturbed regions and overlapping patches. Token order is fixed; the time-decay branch is disabled.

We extend the localized-perturbation analysis in Figure 7 to ETTh2, ETTm1, ETTm2, Weather, and Electricity. Each model is trained without the time-decay branch. The configurations inherit the corresponding full-model search settings and remain fixed across the analysis. We repeat training and

noise-response analysis with three random seeds. The displayed quantities are averaged across these runs. Clean validation MAE selects each checkpoint. Table 20 lists the tokenization settings.

For each dataset, the analysis uses 200 fixed test windows and perturbs only the target column, named OT in the data files. The figures use window index 143, noise strength $\alpha = 0.5$, and the first noise repeat. A 12-point Gaussian segment is centered and normalized to unit RMS, then scaled by α times the target's training-set standard deviation. The same segment is added at an early or a late position: $[12, 24)$ and $[60, 72)$ for $L = 96$, or $[64, 76)$ and $[320, 332)$ for $L = 512$, using zero-based indices. Future targets remain unchanged.

All panels show values in standardized units. Token rows keep their native order across conditions, and color scales are shared within each dataset. Orange outlines identify patches overlapping the perturbed interval. The difference panels report $100(A_{\text{noisy}} - A_{\text{clean}})$ in percentage points. The forecast panels report the target-channel MAE for the displayed window.

Table 20: Settings for the additional noise-response cases. P and S denote patch length and stride; M and K are the numbers of input patches and output tokens. ETTm1 includes an end-aligned final patch.

Dataset	L	H	P	S	M	K
ETTh2	96	96	8	8	12	8
ETTm1	512	192	48	24	21	4
ETTm2	96	96	12	12	8	6
Weather	512	192	8	8	64	16
Electricity	96	96	12	12	8	2

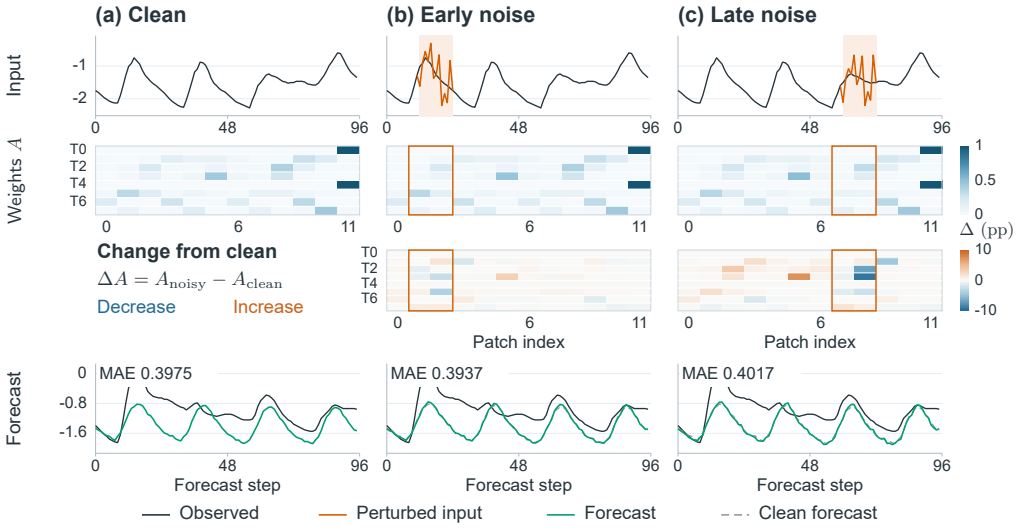

Figure 8: Localized-noise response on ETTh2 ($96 \to 96$), with 12 patches aggregated into eight tokens.

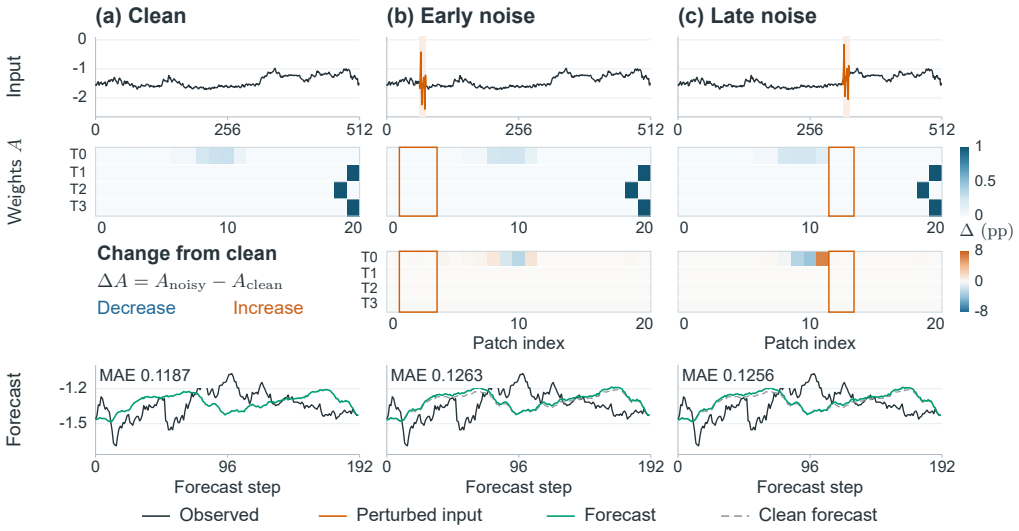

Figure 9: Localized-noise response on ETTm1 ($512 \rightarrow 192$), with 21 overlapping patches aggregated into four tokens.

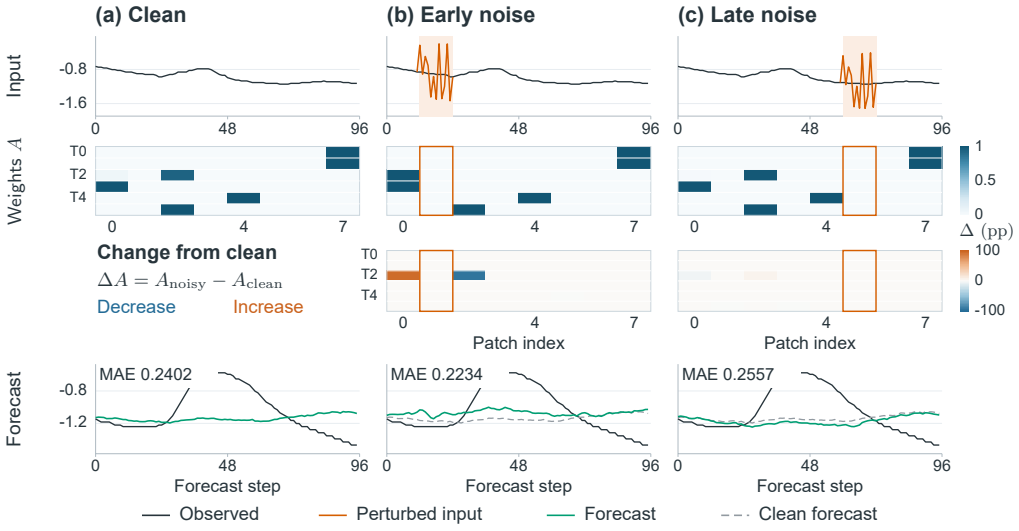

Figure 10: Localized-noise response on ETTm2 ($96 \rightarrow 96$), with eight patches aggregated into six tokens.

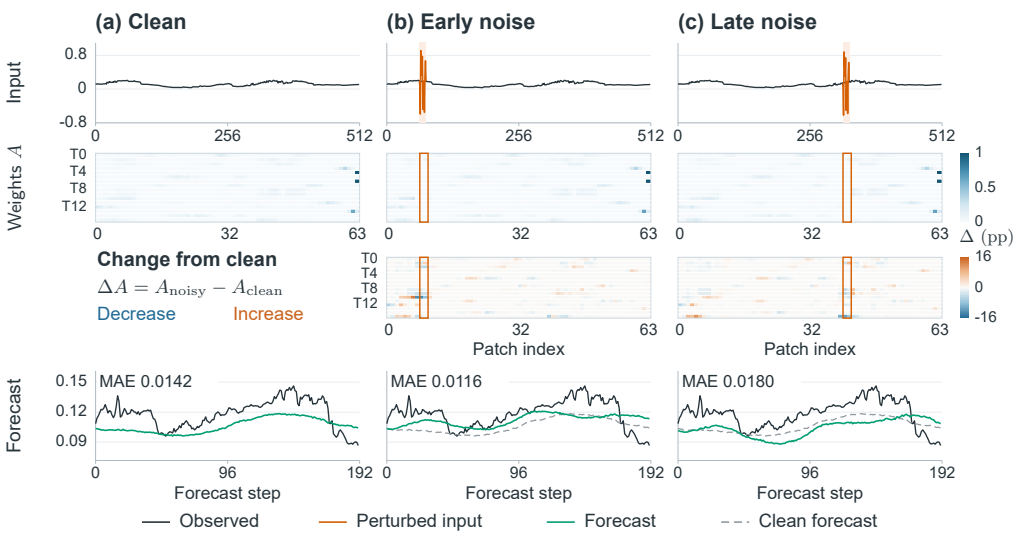

Figure 11: Localized-noise response on Weather ($512 \to 192$), with 64 patches aggregated into 16 tokens.

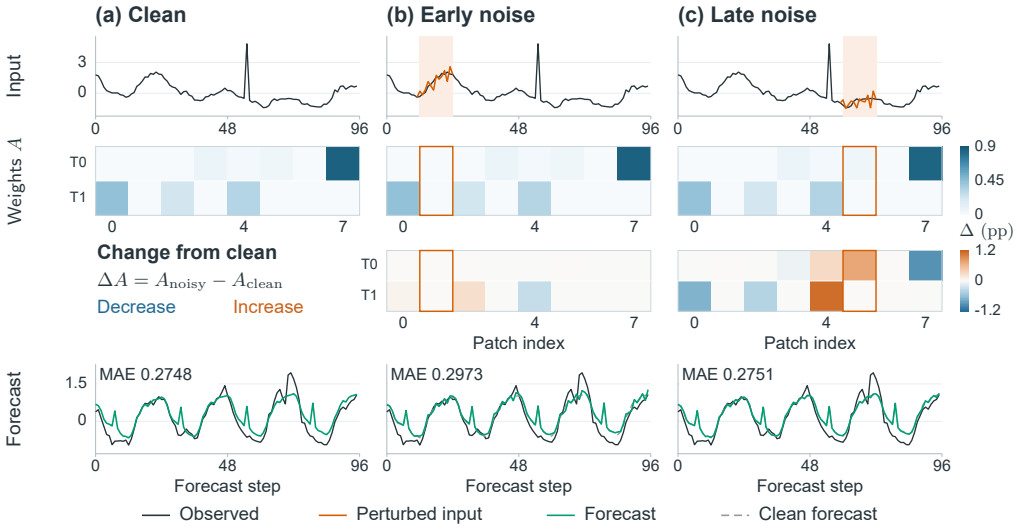

Figure 12: Localized-noise response on Electricity ($96 \to 96$), with eight patches aggregated into two tokens.

H MAXIMUM TOKEN COMPRESSION AT COMPARABLE MSE

We measure the largest token compression factor under a relative test-MSE tolerance ϵ against the full-token PatchTST reference. We use $\epsilon = 1\%$ on all datasets except ETTm2, where neither method meets this threshold and we relax it to 3%. For each dataset and method, we select the smallest evaluated K satisfying $\mathrm{MSE}(K) \le (1+\epsilon)\,\mathrm{MSE}_{\mathrm{PatchTST}}$ and report the compression factor M/K, where M is the number of input patches. Lower MSE also satisfies this condition. This comparison summarizes the completed test-MSE sweep; the reported maximum is over the evaluated token counts.

Experimental settings. The input and prediction lengths are $L = 512$ and $H = 336$. We compare PatchBasis and TimeBase against full-token PatchTST on ETTh1, ETTh2, ETTm1, ETTm2, and Weather. Patch length and stride are both 24 on ETT and 8 on Weather, giving $M = 22$ and $M = 64$ input patches for PatchBasis and TimeBase. We evaluate $K \in \{2, 4, 6, 8, 12, 16\}$ on ETT and additionally $K = 24, 32$ on Weather. PatchTST uses 22 tokens on ETT and 65 on Weather owing to its padding. The sweep covers 69 configurations, including the five PatchTST references. We evaluate each configuration with three random seeds and report the mean results.

All models use a one-layer Transformer with embedding dimension 128, feed-forward dimension 512, eight attention heads, dropout 0.1, and a flattening prediction head. TimeBase uses shared linear aggregation to construct its Transformer input tokens. Training uses batch size 32, Adam with learning rate 10^{-3} and no weight decay, a cosine learning-rate schedule, and MAE loss. We train for at most 50 epochs with early-stopping patience five and evaluate the checkpoint with the lowest validation MAE. Within each dataset, methods share chronological splits and forecasting windows. Standardization uses training statistics; test MSE averages all windows, forecast steps, and variables.

Dataset	ϵ (%)	M	PatchTST MSE	Method	K	M/K	MSE	Δ (%)
ETTh1	1	22	0.472	TimeBase	2	11.000×	0.428	−9.240
				PatchBasis	2	11.000×	0.411	−12.940
ETTh2	1	22	0.390	TimeBase	2	11.000×	0.380	−2.426
				PatchBasis	2	11.000×	0.379	−2.739
ETTm1	1	22	0.356	TimeBase	4	5.500×	0.354	−0.508
				PatchBasis	2	11.000×	0.357	+0.372
ETTm2	3	22	0.264	TimeBase	8	2.750×	0.271	+2.794
				PatchBasis	4	5.500×	0.269	+2.160
Weather	1	64	0.242	TimeBase	2	32.000×	0.243	+0.193
				PatchBasis	2	32.000×	0.242	+0.025

Table 21: Maximum observed token compression under the dataset-specific MSE tolerance ϵ relative to full-token PatchTST ($512 \to 336$). Errors are averaged over three seeds. The tolerance is 1%, relaxed to 3% on ETTm2 because neither method meets 1%. M denotes the input patch count for TimeBase and PatchBasis; K is the smallest evaluated token count satisfying the condition. $\Delta = 100(\mathrm{MSE}/\mathrm{MSE}_{\mathrm{PatchTST}} - 1)$. A dash indicates that no evaluated configuration satisfies the threshold. Selection and threshold checks use unrounded MSE values.

Results. PatchBasis retains two tokens on ETTh1, ETTh2, and ETTm1, giving an $11\times$ compression factor, and two tokens on Weather, giving $32\times$ compression. TimeBase reaches the same compression factors on ETTh1, ETTh2, and Weather, but requires four tokens on ETTm1 ($5.5\times$). Under the relaxed 3% tolerance on ETTm2, PatchBasis retains four tokens ($5.5\times$) with MSE 0.2693, a 2.16% increase over PatchTST. TimeBase requires eight tokens ($2.75\times$), with MSE 0.2710 and a 2.79% increase.