SELECT, DON'T COMPRESS: SPARSE ATTENTION FOR HIGH-DIMENSIONAL TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

In high-dimensional multivariate time series, forecasting accuracy may depend on specific interactions between individual channels. Dense cross-channel attention can capture them, but its cost grows quadratically with the number of channels. Traditional alternatives either compress these interactions or remove them altogether, potentially losing the fine-grained dependencies that matter. We introduce Native Sparse Time Attention (NSTA), which preserves direct interactions between individual channels while scaling linearly in their number for a fixed sparsity budget. On both evaluated high-dimensional datasets, NSTA shifts the accuracy–efficiency Pareto frontier toward lower forecasting error.

1 INTRODUCTION

Recent multivariate forecasting architectures represent temporal patches from different channels as distinct tokens, forming a joint patch–channel grid (Liu et al., 2025b;a). Dense attention over this grid enables direct interactions across both time and channels, but its cost grows rapidly with the channel dimension. This cost becomes concrete on a high-dimensional dataset such as Traffic (Wu et al., 2021), where 84 patches and 862 channels produce 72,408 tokens. Dense attention over this grid evaluates up to 5.24 billion attention scores per head in every layer. This quadratic scaling makes cross-channel attention costly in high-dimensional multivariate settings.

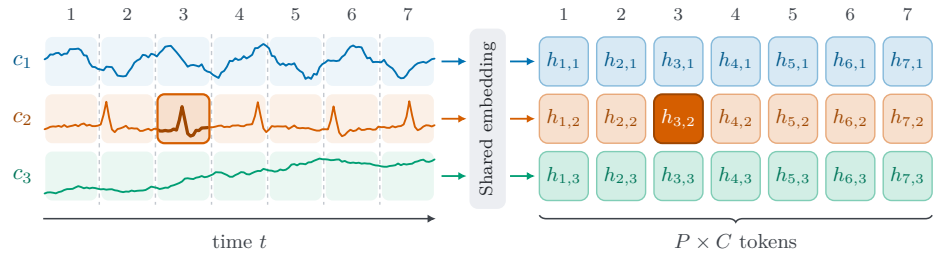


Figure 1: Multivariate patch-based approach: channels are split into P temporal patches, and a shared embedding is applied to obtain tokens $h_{p,c}$, indexed by patch p and channel c.

Eliminating cross-channel interactions entirely avoids this cost, but also removes potentially useful dependencies among series. Yet channel-independent models remain competitive with elaborate multivariate architectures (Zeng et al., 2023; Nie et al., 2023; Xu et al., 2024). This suggests that exhaustive interaction between all pairs of channels is often unnecessary. Several multivariate architectures therefore avoid dense pairwise attention by communicating through compressed or intermediate representations (Zhang & Yan, 2023; Liu et al., 2025a). A remaining design challenge is to retain linear channel scaling while preserving direct access to individual channel representations.

Our key idea is to make the relationship type explicit in the attention operator. We decompose the visible history into three complementary views (cf. Figure 3): the *univariate branch* provides exhaustive same-channel temporal history, the *compression branch* supplies broad cross-sectional context, while the *selection branch* retrieves a selective set for direct attention. Crucially, the selection branch separates shared candidate routing from query-specific attention: candidate tokens are

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.

selected once for each query patch and shared across its channel queries. This amortizes routing across channels and enables linear scaling in the channel dimension under fixed selection budgets, following the principle of Native Sparse Attention (Yuan et al., 2025).

Our main contributions are:

- We introduce *Native Sparse Time Attention* (NSTA), a three-path attention operator that distinguishes same-channel and cross-channel relationships without channel positional embeddings or channel-indexed parameters (Section 3).
- We show that NSTA's attention complexity scales linearly with the number of channels for fixed temporal and channel-selection budgets (Section 3.6), and validate this scaling empirically in both FLOPs and wall-clock latency (Section 4.1).
- On high-channel-count multivariate datasets, NSTA defines the lower-error region of the accuracy–latency Pareto frontier, providing Pareto-optimal configurations beyond the fastest operating points on both Traffic and GTWSF-Temp (Section 4.3).

We discuss related approaches and their relationship to Native Sparse Time Attention in Section 5.

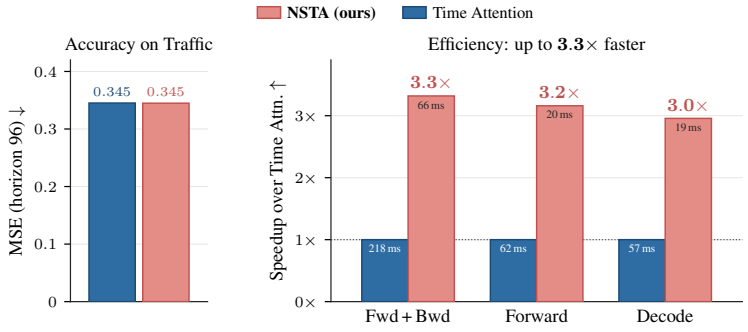


Figure 2: Matched accuracy at lower cost on Traffic for lookback window $L = 672$ and sparsity budget $k_p = 2$, $k_c = 256$. *Left:* forecasting MSE at horizon 96. NSTA matches Time Attention. *Right:* NSTA speeds up forward + backward by $3.3\times$, forward by $3.2\times$ and decoding by $3.0\times$.

2 Rethinking Relationship Modeling

Multivariate time series exhibit different types of dependencies. At the univariate level, the future evolution of each channel may depend on specific past observations through temporal structures such as autocorrelation or seasonality (Box et al., 2015). Capturing these dependencies naturally requires direct access to individual historical observations.

Cross-channel dependencies can take more heterogeneous forms. In particular, multiple variables may co-move in response to common underlying factors (Fama & French, 1993; Stock & Watson, 2002). When such shared structure is present, a substantial part of the relevant information can be summarized through a compact representation of the overall cross-sectional state.

Table 1: Cross-channel attention design space.

Cross-channel strategy	**Direct access**	**Channel complexity**
Independent	✗	$\mathcal{O}(C)$
Compressed	✗	$\mathcal{O}(C)$
Dense	✓	$\mathcal{O}(C^2)$
Sparse	✓	$\mathcal{O}(\mathbf{C})$

Other dependencies, however, are more localized. A disturbance often originates in a small number of channels before propagating to many others. In road networks, congestion triggered by an incident or bottleneck at a few sensors travels upstream as a shockwave and later affects a large number

of measurements (Lighthill & Whitham, 1955; Richards, 1956). Such dependencies are sparse and time-varying, but largely shared across targets. This motivates complementing a compressed summary with direct access to a small set of informative channels, identified at each patch and shared across queries. We call a cross-channel interaction *direct* when a query can attend directly to channel representations rather than only to a pooled or intermediate summary. Dense attention over the full grid provides this access, but requires $\mathcal{O}(P^2C^2)$ attention scores.

Existing efficient designs reduce this cost by restricting cross-channel interactions. Channel-independent models process each channel separately (Zeng et al., 2023; Nie et al., 2023; Xu et al., 2024), yielding linear complexity in C but eliminating direct cross-channel attention. Other approaches mediate channel interactions through compressed or intermediate representations (Zhang & Yan, 2023; Liu et al., 2025a), reducing computation at the cost of direct access to individual channel representations. This work targets the remaining regime: direct access to selected channel-specific representations with linear complexity in C for fixed sparsity budgets.

3 Architecture

3.1 Overview

Native Sparse Time Attention operates through three branches, each defining a different query-key-value representation and attention pattern. Following Liu et al. (2025b), RoPE is applied along the temporal axis in each attention branch. The outputs of the three branches are combined through learned gates, as described in Section 3.5.

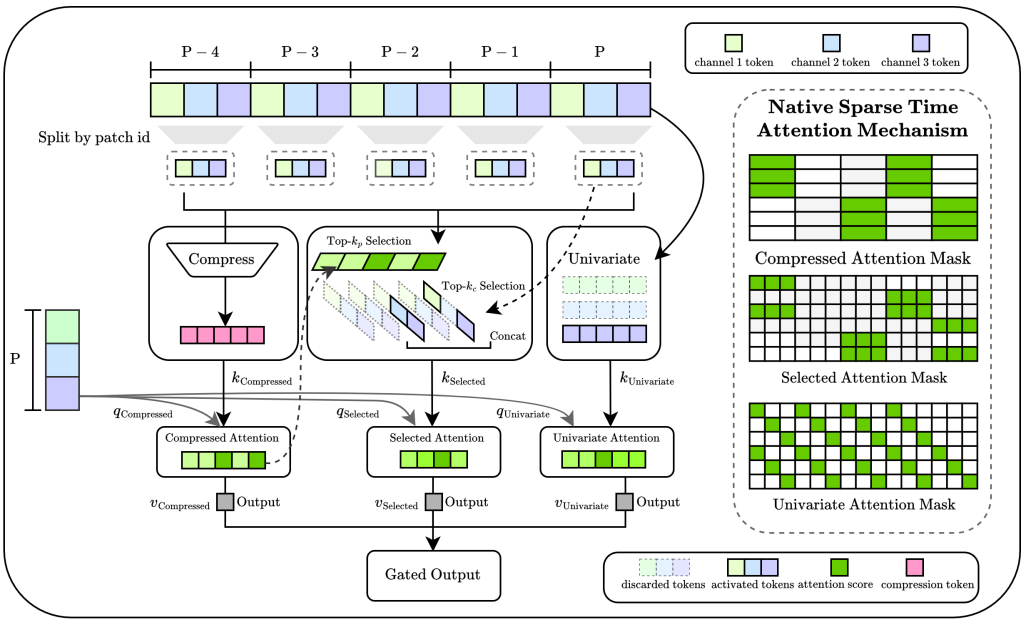


Figure 3: Overview of the Native Sparse Time Attention mechanism. *Left*: the three parallel attention branches. *Right*: corresponding attention patterns, with green indicating computed attention scores and white indicating skipped interactions. Visually inspired by Yuan et al. (2025).

The mechanism supports both causal and non-causal attention by applying the corresponding visibility constraints consistently across all three branches. The mechanism is applied independently across attention heads. For clarity, we describe a single head below and omit the head index from the notation.

3.2 UNIVARIATE BRANCH

Let $\mathbf{h}_{p,c} \in \mathbb{R}^{d_{\text{model}}}$ denote the embedded token at patch p and channel c. For each channel independently, the univariate branch applies standard self-attention over the sequence $(\mathbf{h}_{p,c})_{p=1}^{P}$ along the patch axis. Queries attend only to observations from the same channel, preserving exact temporal access at full resolution without introducing cross-channel interactions; the channel axis contributes only a linear factor to the cost, since no query attends across channels. This branch supplies the purely temporal component of the representation, while cross-channel structure is provided by the compression and selection branches (Sections 3.3 and 3.4).

Each channel token is projected with branch-specific matrices $\mathbf{W}_Q^{\text{uni}}, \mathbf{W}_K^{\text{uni}}, \mathbf{W}_V^{\text{uni}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ into univariate queries, keys, and values. Each query attends over the visible patches of its own channel, $\mathcal{P}_p = \{1, \ldots, p\}$.

3.3 COMPRESSION BRANCH

At each patch, the compression branch summarizes the C channel representations into a single cross-sectional summary token. This branch provides global cross-sectional context, while direct cross-channel interactions are delegated to the selection branch (Section 3.4).

Let $\mathbf{H}_p = [\mathbf{h}_{p,1}, \ldots, \mathbf{h}_{p,C}]^\top \in \mathbb{R}^{C \times d_{\text{model}}}$ collect the channel representations at patch p. A learned scorer $\mathbf{u}_{\text{pool}} \in \mathbb{R}^{d_{\text{model}}}$ assigns pooling weights $\mathbf{w}_p$ over channels, which are used to form the compressed representation $\bar{\mathbf{h}}_p \in \mathbb{R}^{d_{\text{model}}}$:

$$\mathbf{w}_p = \text{softmax}\left(\frac{\mathbf{H}_p\, \mathbf{u}_{\text{pool}}}{\sqrt{d_{\text{model}}}}\right), \qquad \bar{\mathbf{h}}_p = \mathbf{H}_p^\top \mathbf{w}_p = \sum_{c=1}^{C} w_{p,c}\, \mathbf{h}_{p,c} \tag{1}$$

The pooled representation is projected with branch-specific matrices $\mathbf{W}_K^{\text{comp}}, \mathbf{W}_V^{\text{comp}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ into compression keys and values, while channel tokens are projected into branch-specific queries with $\mathbf{W}_Q^{\text{comp}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. Each query then applies standard attention over the compressed representations of the visible patches $\mathcal{P}_p = \{1, \ldots, p\}$.

$$\alpha_{p,c}^{\text{comp}} = \underset{p' \in \mathcal{P}_p}{\text{softmax}}\left(\frac{\langle \mathbf{q}_{p,c}^{\text{comp}},\, \mathbf{k}_{p'}^{\text{comp}} \rangle}{\sqrt{d_{\text{model}}}}\right) \tag{2}$$

3.4 SELECTION BRANCH

The selection branch performs hierarchical sparse retrieval across both time and channels. It reuses quantities produced by the compression branch: $\alpha_{p,c}^{\text{comp}}$ (Eq. 2) for patch-level routing and the pooling weights $w_{r,c}$ (Eq. 1) for channel-level routing. As illustrated in Figure 4, it first selects k_p historical patches and then k_c channels within each retained patch.

Patch selection. For query patch p, we rank visible historical patches by the compression attention score (Eq. 2) averaged across channel queries:

$$s_{p,r} = \frac{1}{C} \sum_{c=1}^{C} \alpha_{p,c,r}^{\text{comp}}, \qquad \mathcal{R}_p = \underset{r \in \mathcal{P}_p}{\text{top-k}}\left(s_{p,r};\, k_p\right). \tag{3}$$

Channel selection. For each retained patch $r \in \mathcal{R}_p$, the k_c channels with largest pooling weights are retained, yielding $k_p k_c$ candidates for direct interactions:

$$\mathcal{C}_r = \underset{c \in \{1,\ldots,C\}}{\text{top-k}}\left(w_{r,c};\, k_c\right), \qquad \mathcal{S}_p = \left\{(r,c) \;:\; r \in \mathcal{R}_p,\; c \in \mathcal{C}_r\right\}. \tag{4}$$

Both routing stages are shared across channel queries: $\mathcal{R}_p$ and $\mathcal{C}_r$ are common to all queries in patch p. This query-shared routing is what preserves linear scaling in C. Although routing is shared, each query computes its own attention weights over the selected candidates. The selected original tokens

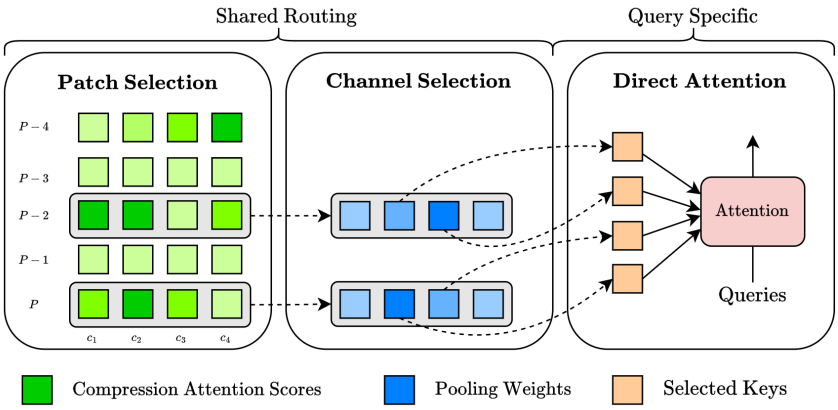


Figure 4: Overview of the hierarchical patch and channel selection mechanism. Hierarchical selection first ranks visible patches using shared compression-attention scores, then selects channels within retained patches; the resulting candidate set is shared across channel queries, while final attention weights are query-specific.

$\mathbf{h}_{r,c}$ are projected with branch-specific matrices $\mathbf{W}_Q^{\text{select}}, \mathbf{W}_K^{\text{select}}, \mathbf{W}_V^{\text{select}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. Each channel query attends over $\mathcal{S}_p$, with logits biased by the pooling weight of each retained token:

$$\alpha_{p,c}^{\text{select}} = \underset{(r,c') \in \mathcal{S}_p}{\text{softmax}} \left(\frac{\langle \mathbf{q}_{p,c}^{\text{select}}, \mathbf{k}_{r,c'}^{\text{select}} \rangle}{\sqrt{d_{\text{model}}}} + \lambda \log w_{r,c'} \right) \tag{5}$$

Routing prior and gradient flow. The term $\lambda \log w_{r,c'}$ acts as a learned prior over retained tokens and provides a differentiable path to the channel router despite the non-differentiable top-k operations. λ is a learnable routing-prior coefficient. Thus the pooling mechanism is trained not only through the compression branch but also through the downstream use of selected tokens.

3.5 Gating Mechanism

Learned gates combine the three branch outputs, following Yuan et al. (2025). Let $\mathcal{B} = \{\text{uni}, \text{comp}, \text{select}\}$ index the branches. For each branch, an independent sigmoid gate $g_{p,c}^{\text{branch}}$ is computed from the embedded token. The gated branch outputs are then summed:

$$\mathbf{o}_{p,c} = \sum_{\text{branch} \in \mathcal{B}} g_{p,c}^{\text{branch}} \, \mathbf{o}_{p,c}^{\text{branch}}. \tag{6}$$

3.6 Complexity Analysis

We characterize the attention cost of NSTA. The following proposition gives the resulting per-layer complexity in terms of the number of temporal patches, channels, attention heads, and selection budgets. The full derivation is provided in Appendix A.

Proposition 1 (Channel complexity of NSTA)**.** *Let P be the number of temporal patches, C the number of channels, and H the number of attention heads. Define the effective selection budgets as*

$$\tilde{k}_p = \min\{k_p, P\}, \qquad \tilde{k}_c = \min\{k_c, C\}.$$

Then the total number of attention scores computed by NSTA satisfies

$$S_{\text{NSTA}} = O\left(HC\left(P^2 + P\tilde{k}_p\tilde{k}_c\right)\right). \tag{7}$$

Table 2: Attention-score complexity of representative interaction strategies. Projection and feed-forward costs are omitted; NSTA assumes selection budgets fixed independently of C.

Method	Cross-channel access	Complexity	Scaling in C
PatchTST	None	$O(HCP^2)$	Linear
Timer-XL with Time Attention	Dense	$O(HC^2P^2)$	Quadratic
Timer-XL with NSTA	Sparse	$O(HCP(P+\tilde{k}_p\tilde{k}_c))$	Linear

4 Experiments

We evaluate NSTA along two axes, computational cost and forecasting accuracy. Our experiments address three questions:

Q1 Do NSTA's theoretical FLOP savings translate into wall-clock speedups? (Section 4.1)
Q2 At a fixed lookback and retrieval budget, how much accuracy is lost, if any? (Section 4.2)
Q3 Do NSTA's savings translate into better accuracy–latency operating points? (Section 4.3)

Common protocol. In all experiments, NSTA replaces the Time Attention operator of Timer-XL (Liu et al., 2025b), while the decoder-only backbone, optimizer, and training schedule are held fixed, so that the attention operator is the primary architectural change under comparison. We use standard multivariate benchmarks spanning 7 to 862 channels (ETT, Weather, Solar-Energy, ECL, Traffic), following Wu et al. (2023a), and the GTWSF-Temp dataset (Wu et al., 2023b), with $C = 3850$ channels, to probe the high-dimensional regime. We report MSE and MAE averaged over three random seeds. Dataset statistics are given in Appendix B.1; hyperparameters in Appendix B.2 and the efficiency measurement protocol in Appendix B.4.

4.1 Empirical Scaling with the Number of Channels

Setup. To test whether NSTA's theoretical savings translate into wall-clock gains, we time forward passes while scaling the number of channels from $C = 256$ to 4096 and holding everything else fixed. The dense baseline is Timer-XL's Time Attention (implementation description is provided in Appendix B.4). NSTA shares all other components and differs only in the attention mechanism. Synthetic inputs are used on an H100. Full details are given in Appendix B.4.

Results. Figure 5 shows that NSTA's latency grows substantially more slowly with C than that of Time Attention. At $C = 256$, where $k_c = C$ and NSTA performs no channel pruning, it is slightly slower due to selection overhead; it becomes faster from $C = 512$ onward and reaches a $23\times$ forward-latency speedup at $C = 4096$. The measured speedup tracks the FLOP reduction.

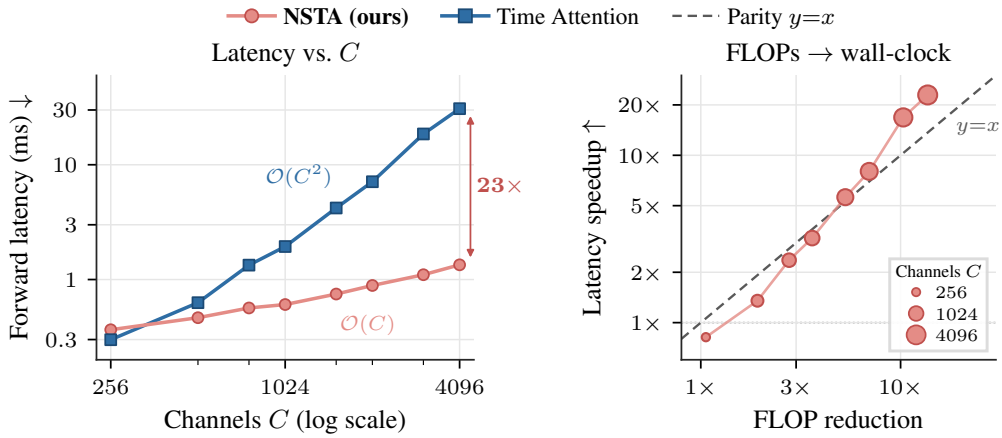


Figure 5: Empirical scaling with the number of input channels C. *Left*: forward latency of Time Attention and NSTA ($23\times$ faster at $C = 4096$). *Right*: latency speedup vs. FLOP reduction.

4.2 Forecasting Accuracy under a Fixed Sparsity Budget

Setup. To answer Q2, the Timer-XL one-for-all protocol is adopted: one model per dataset is trained with lookback 672 (7 patches per channel) and rolled out to horizons $96, 192, 336, 720$. Dense Time Attention is retrained under identical settings. A single untuned budget ($k_p = 4$, $k_c = 32$) is used.

Table 3: Channel retention induced by the fixed budget $k_c = 32$ in Section 4.2.

Dataset	ETTh1	Weather	Solar-Energy	ECL	Traffic
Retained $\tilde{k}_c/C$	100%	100%	23.4%	10.0%	3.7%

Results. Table 4 quantifies the fixed-context cost of this single untuned budget. With $k_c = 32$, NSTA matches or improves upon the MSE of dense Time Attention on ETTh1, Weather, and Solar-Energy. On the higher-dimensional ECL and Traffic datasets, NSTA incurs moderate increases of approximately $+4.5\%$ and $+1.6\%$, while retaining only 10.0% and 3.7% of the channels.

Table 4: Multivariate long-term forecasting under the Timer-XL one-for-all protocol (Liu et al., 2025b). MSE and MAE are averaged over prediction horizons with input length 672. NSTA uses $k_p = 4$ and $k_c = 32$ for all datasets. Full horizon-specific results are reported in Table 10.

Models	**Timer-XL (with NSTA)**		Timer-XL (2025b)		Timer* (2024b)		UniTST* (2025a)		iTransformer* (2024a)		DLinear* (2023)		PatchTST* (2023)		TimesNet* (2023a)		Stationary* (2022)		Autoformer* (2021)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	0.163	0.254	**0.156**	**0.249**	0.161	0.251	0.163	0.257	0.164	0.258	0.165	0.265	0.169	0.268	0.201	0.303	0.265	0.358	0.289	0.379
ETTh1	**0.407**	**0.431**	0.415	0.434	0.418	0.436	0.429	0.447	0.421	0.445	0.426	0.444	0.412	0.435	0.495	0.491	0.505	0.513	0.517	0.528
Traffic	0.386	0.261	**0.380**	**0.255**	0.384	0.259	0.385	0.265	0.384	0.274	0.423	0.298	0.391	0.275	0.602	0.322	0.630	0.347	0.684	0.433
Weather	0.234	0.272	0.242	0.276	0.232	0.270	0.231	0.272	0.266	0.291	0.239	0.291	**0.226**	**0.268**	0.264	0.293	0.308	0.329	0.435	0.455
Solar-Energy	0.220	0.259	0.230	0.256	0.233	**0.249**	0.241	0.275	0.213	0.291	0.222	0.283	**0.202**	0.269	0.213	0.295	0.254	0.315	0.265	0.325

* Results from Liu et al. (2025b).

4.3 Accuracy–Latency Trade-off across Lookback Windows

Setup. To answer Q3, the strategies in Table 1 are compared jointly in terms of accuracy and forward+backward latency across lookback windows. PatchTST, UniTST, and Timer-XL serve as baselines, with all models evaluated using the same protocol (Appendix B.2). The latency measurement protocol is reported in Appendix B.4.

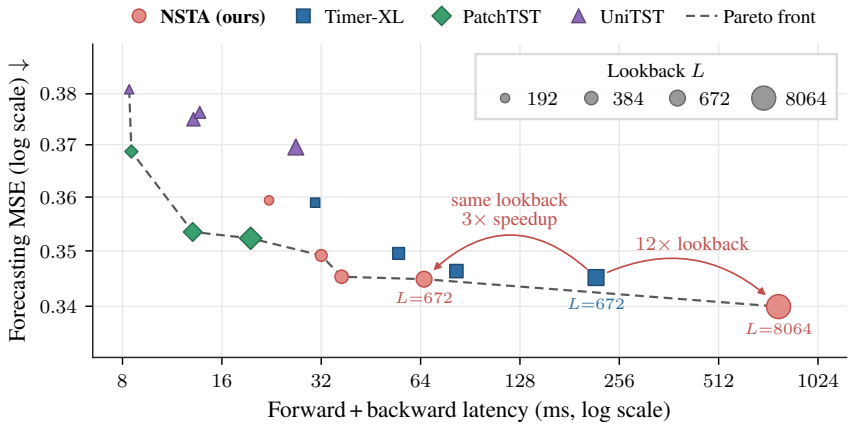


Figure 6: Accuracy–latency trade-off on Traffic (horizon 96). At $L = 672$, NSTA matches dense Time Attention's accuracy with approximately $3\times$ lower forward–backward latency; extending context to $L = 8064$ further improves accuracy. Marker size indicates lookback. NSTA uses $k_p = 2$, $k_c = 256$ throughout. Lower-left is better.

NSTA configurations. NSTA uses a fixed selection budget $k_p = 2, k_c = 256$ for every lookback in the Pareto experiments on both datasets. On GTWSF-Temp, we also evaluate one higher-compute configuration with $k_p = 2, k_c = 1024$ to characterize the accuracy–latency trade-off.

Results. NSTA translates computational savings into improved accuracy–latency trade-offs on both high-dimensional datasets. On Traffic (Figure 6), NSTA matches dense Time Attention's forecasting accuracy at the same lookback ($L = 672$), while reducing forward+backward latency by approximately $3\times$. Extending NSTA's lookback to $L = 8064$, a $12\times$ increase in context, reduces MSE to 0.339, the lowest among the evaluated configurations. On GTWSF-Temp (Figure 7), NSTA matches dense Time Attention's accuracy at approximately $2\times$ lower latency while using twice the context ($L = 96$ versus $L = 48$). Increasing the selection budget to $k_c = 1024$ further improves MSE to 6.887, with $k_p = 2$, while remaining within the dense baseline's latency budget. Thus, NSTA supports both faster forecasting at matched accuracy and more accurate forecasting with longer context.

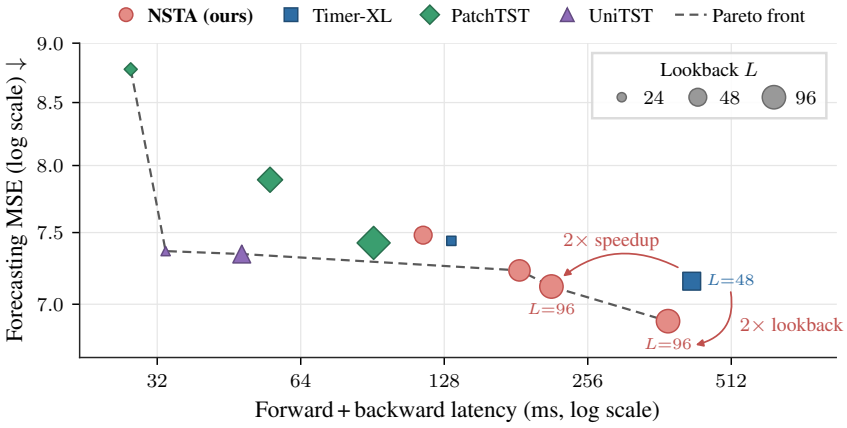


Figure 7: Accuracy–latency trade-off on GTWSF-Temp (horizon 24). NSTA matches dense Time Attention's accuracy with approximately $2\times$ lower forward–backward latency and twice the context. Marker size indicates lookback. NSTA uses $k_p = 2$, $k_c = 256$; the lowest-error point uses $k_c = 1024$. Lower-left is better.

4.4 Analysis

We investigate how NSTA's selection behavior varies with dataset characteristics.

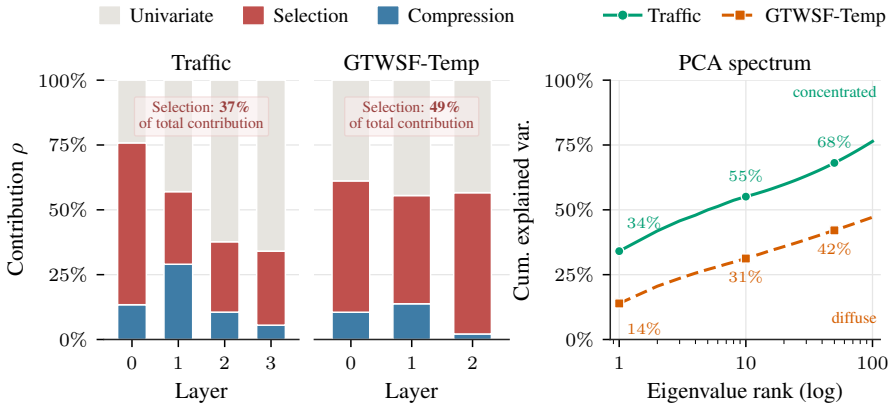


Figure 8: *Left*: Stacked shares of NSTA's three gated branches across depth. GTWSF-Temp has a larger selection contribution (49% vs. 37%). *Right*: Cumulative variance explained by principal components of standardized first differences. Traffic exhibits a substantially more concentrated spectrum than GTWSF-Temp.

Dataset structure and selection behavior. The selection branch contributes more on GTWSF-Temp than on Traffic (49% vs. 37%), alongside a more diffuse PCA spectrum of standardized first differences. The selection analysis in Appendix C.3 reveals a related distinction: channel selections overlap strongly across many deeper-layer heads on Traffic, whereas GTWSF-Temp exhibits distinct selected subsets. The budget sensitivity results in Appendix C.4 complement these observations: increasing k_c consistently improves forecasting accuracy on GTWSF-Temp, while Traffic exhibits a non-monotonic response. Together, these results suggest that retaining more channels is beneficial when informative cross-channel variation is more distributed, whereas a smaller selected subset can suffice when that variation is more concentrated.

5 Related Work

Temporal and cross-channel structure. PatchTST removes direct cross-channel interaction (Nie et al., 2023), whereas iTransformer attends across channels only after compressing each channel's history into a single token (Liu et al., 2024a). Crossformer, UniTST, and Timer-XL retain both axes by attending over patch–channel tokens (Zhang & Yan, 2023; Liu et al., 2025a;b). NSTA keeps this patch–channel formulation but resolves the two axes differently: same-channel temporal dependencies remain exact, while cross-channel information combines compressed global context with sparse direct access to selected patch–channel tokens. This retains the Timer-XL forecasting setup at lower interaction cost.

Efficient cross-channel interaction. Several multivariate architectures reduce cross-channel cost through intermediate representations: learned router vectors in Crossformer (Zhang & Yan, 2023), a dispatcher in UniTST (Liu et al., 2025a), a global series core in SOFTS (Han et al., 2024), and frequency-domain soft clustering in DUET (Qiu et al., 2025). DUET, however, still builds pairwise channel-relation matrices whose cost is quadratic in the number of channels. NSTA instead combines compressed global communication, which captures broad cross-sectional structure cheaply, with sparse direct access to the original patch–channel tokens, which preserves direct interactions for informative patches and channels.

Sparse and efficient attention. Efficient-attention methods reduce self-attention cost through low-rank projections (Wang et al., 2020), kernel feature maps (Choromanski et al., 2021), or sparse query selection (Zhou et al., 2021), but generally operate over one-dimensional token sequences. Content-based sparse routing has also been explored in Routing Transformer (Roy et al., 2021). NSTA inherits the three-branch decomposition of Native Sparse Attention (Yuan et al., 2025), which combines compressed, selected, and local attention paths, but it applies this decomposition with a different notion of what should be shared. In NSA, selected blocks are shared across the query heads of a GQA group, so that all heads in the group load the same key–value blocks. This sharing serves hardware efficiency by reducing memory access at a given sequence length. In NSTA, the selected candidate set is instead shared across all channel queries of the same patch, and this sharing is what makes the attention cost linear in the number of channels. The choice reflects a structural difference between the two settings (Section 2).

6 Conclusion and Future Work

In this work, we introduced Native Sparse Time Attention (NSTA), a sparse attention operator for multivariate time-series forecasting. NSTA combines exact same-channel temporal attention, a compressed cross-channel summary, and direct attention to a selected set of tokens. Because this selection is shared across the channel queries of each patch, the attention cost scales linearly with the number of channels for fixed selection budgets. NSTA therefore offers two benefits: it matches dense attention at a fixed context at lower cost, and it frees computation for larger channel sets and longer lookback windows. On Traffic and GTWSF-Temp, NSTA matches the accuracy of dense Time Attention at roughly $3\times$ lower forward–backward latency, and extending its lookback further reduces forecasting error. These results suggest that tunable sparsity is a practical route to channel and context scales that are costly for dense attention. Future work will extend NSTA to transfer learning and foundation-model settings, particularly across datasets with heterogeneous variable sets.

7 AI USE STATEMENT

Large language models were used to edit and polish the manuscript and to support the search for related work. The method, experiments, results and claims were developed by the authors. All LLM-suggested edits and references were checked by the authors, who take full responsibility for the content of the paper.

8 ETHICS STATEMENT

This work focuses on time-series forecasting. We do not identify any direct ethical risks specific to the proposed method.

9 REPRODUCIBILITY STATEMENT

The appendix provides the implementation details, including dataset descriptions, metrics, model configurations, and experimental protocols.

10 ACKNOWLEDGMENTS

Acknowledgments omitted for double-blind review.

REFERENCES

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5th edition, 2015. ISBN 9781118675021.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021.

Eugene F. Fama and Kenneth R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993. ISSN 0304-405X. doi: 10.1016/0304-405X(93)90023-5.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. SOFTS: Efficient multivariate time series forecasting with series-core fusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955. doi: 10.1098/rspa.1955.0089.

Juncheng Liu, Chenghao Liu, Gerald Woo, Yiwei Wang, Bryan Hooi, Caiming Xiong, and Doyen Sahoo. UniTST: Effectively modeling inter-series and intra-series dependencies for multivariate time series forecasting. *Transactions on Machine Learning Research*, 2025a.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 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 *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 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 (ICLR)*, 2025b. arXiv:2410.04803.

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

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

Paul I. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956. doi: 10.1287/opre.4.1.42.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353.

James H. Stock and Mark W. Watson. Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, 97(460):1167–1179, 2002. doi: 10.1198/016214502388618960.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

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 (ICLR)*, 2023a.

Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence*, 5(6):602–611, 2023b. URL https://www.nature.com/articles/s42256-023-00667-9.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with $10k$ parameters. In *International Conference on Learning Representations (ICLR)*, 2024.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. arXiv:2502.11089.

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.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2023.

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

A Complexity Analysis

A.1 Theoretical Proof

We provide the detailed derivation of Proposition 1.

Proof. Each attention head processes PC patch–channel queries. We count the number of attention scores computed by each branch.

For the *univariate branch*, a query attends only to keys from the same channel, of which there are at most P. Hence, across all PC queries and H heads, this branch computes

$$O(H\,PC \cdot P) = O\big(HCP^2\big)$$

attention scores.

For the *compression branch*, each query attends to at most one compressed key per patch, and therefore to at most P keys in total. Its cost is consequently also

$$O\big(HCP^2\big)\,.$$

For the *selection branch*, a query associated with patch p attends to the candidate set $\mathcal{S}_p$ defined in Eq. (4). By construction,

$$|\mathcal{S}_p| = |\mathcal{R}_p|\,\tilde{k}_c \le \tilde{k}_p\tilde{k}_c.$$

Thus, over all PC queries and H heads, the selection branch requires

$$O\Big(HCP\,\tilde{k}_p\tilde{k}_c\Big)$$

attention scores.

Summing the three contributions yields

$$O\Big(HCP^2 + HCP^2 + HCP\,\tilde{k}_p\tilde{k}_c\Big)\,,$$

which gives equation 7. The number of attention scores that must be materialized during the forward pass is bounded by the same quantity, yielding the stated complexity. □

A.2 Empirical Evaluation

We measure the attention FLOP counts with PyTorch's `FlopCounterMode` using the corresponding attention modules and configurations. The count includes the Q/K/V/output projections and attention matrix multiplications, with each multiply–accumulate counted as two floating-point operations. Elementwise operations such as softmax, masking, gathers, and top-k selection are not counted. For dense Time Attention, the full $PC \times PC$ score grid is charged even for masked entries, since the dense operator still executes this work; NSTA is likewise charged for the matrix multiplications actually executed by its sparse branches. The reported GFLOPs therefore measure executed attention work and exclude the remaining backbone, so end-to-end computational cost is higher for all configurations.

B Experimental Details

B.1 Datasets

We evaluate on widely used multivariate forecasting benchmarks together with a high-dimensional benchmark used to probe the regime NSTA targets.

- **ETT** (Zhou et al., 2021) records seven variables associated with electricity transformers from July 2016 to July 2018. ETTh1 and ETTh2 are sampled hourly; ETTm1 and ETTm2 every 15 minutes.
- **ECL** (Wu et al., 2021) records the hourly electricity consumption of 321 clients.
- **Traffic** (Wu et al., 2021) contains hourly road occupancy rates from 862 sensors on San Francisco Bay Area highways between January 2015 and December 2016.
- **Weather** (Wu et al., 2021) contains 21 meteorological factors recorded every 10 minutes at the Max Planck Institute for Biogeochemistry weather station during 2020.
- **Solar-Energy** records the power production of 137 photovoltaic plants in 2006, sampled every 10 minutes.
- **GTWSF** (Wu et al., 2023b) is drawn from the National Centers for Environmental Information and contains hourly average temperature and wind speed measurements from 3850 stations worldwide spanning 2019 to 2021. We use the temperature subset, denoted GTWSF-Temp, which provides $C = 3850$ channels and is the highest-dimensional benchmark in our evaluation.

We follow the data processing and train/validation/test protocol of Wu et al. (2023a), splitting chronologically to prevent leakage. Table 5 summarizes the dimensions, split sizes, and forecasting settings.

Table 5: Dataset descriptions. *Dim.* is the number of channels C. *Length* gives the number of time points in the (train, validation, test) splits.

Dataset	Domain (frequency)	Dim.	Length (train, val, test)
ETTh1, ETTh2	Electricity (hourly)	7	(8545, 2881, 2881)
ETTm1, ETTm2	Electricity (15 min)	7	(34465, 11521, 11521)
ECL	Electricity (hourly)	321	(18317, 2633, 5261)
Traffic	Transportation (hourly)	862	(12185, 1757, 3509)
Weather	Climate (10 min)	21	(36792, 5271, 10540)
Solar-Energy	Energy (10 min)	137	(36601, 5161, 10417)
GTWSF-Temp	Climate (hourly)	3850	(12280, 1755, 3509)

B.2 Baselines

Our primary comparison is against Time Attention, obtained by running the Timer-XL architecture with its original attention operator under the same backbone, data pipeline, and training schedule; this isolates the effect of the attention operator. PatchTST (Nie et al., 2023) provides a channel-independent reference point without cross-channel interactions.

Results marked with $*$ in Tables 10 and 4 for Timer, UniTST, iTransformer, DLinear, PatchTST, TimesNet, Non-stationary Transformer, and Autoformer are reported by Liu et al. (2025b) under the same one-for-all protocol and input length 672, and are reproduced here without modification. We did not rerun these baselines.

All experiments are implemented in PyTorch. We use the Adam optimizer (Kingma & Ba, 2015) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and the MSE training objective. Accuracy experiments are run on NVIDIA H100 GPUs. Because NSTA is evaluated as a drop-in replacement for Time Attention, we adopt the per-dataset model and optimization configuration of Timer-XL without modification. These configurations are listed in Table 6.

Table 6: Model and optimization configurations. N_{layer} is the number of Transformer blocks, d_{model} the residual width, d_k the per-head dimension, H the number of heads, P_{size} the patch size, and LR the initial learning rate. These follow Liu et al. (2025b) and are shared by all architectures.

Dataset	N_{layer}	d_{model}	d_k	H	P_{size}	LR	Batch size
ETTh1, ETTh2	1	1024	128	8	96	0.0001	32
ETTm1, ETTm2	1	1024	128	8	96	0.0001	32
ECL	5	512	64	8	96	0.0005	4
Traffic	4	512	64	8	96	0.0005	4
Weather	4	512	64	8	96	0.0005	32
Solar-Energy	6	512	64	8	96	0.0001	16
GTWSF-Temp	3	1024	128	8	24	0.0001	8

B.3 Metrics

We report mean squared error and mean absolute error, following prior work. For a univariate series $X \in \mathbb{R}^T$ with prediction $\widehat{X}$,

$$\text{MSE} = \frac{1}{T}\sum_{i=1}^{T}\left(X_i - \widehat{X}_i\right)^2, \qquad \text{MAE} = \frac{1}{T}\sum_{i=1}^{T}\left|X_i - \widehat{X}_i\right|. \tag{8}$$

For multivariate series both metrics are additionally averaged over the channel dimension. Table 4 reports the mean over the four horizons $\{96, 192, 336, 720\}$; the accuracy–latency results in Section 4.3 use the mean MSE over three seeds at horizons 96 and 24.

B.4 Efficiency Measurement

Setting. The scaling benchmark of Section 4.1 times a single attention layer as the number of channels varies, $C \in \{256, 512, 768, 1024, 1536, 2048, 3072, 4096\}$, at $P = 7$ patches per channel. The token sequence therefore has length PC, reaching 28,672 at $C = 4096$. The layer has $d_{\text{model}} = 512$ and eight heads. It includes the query, key, value and output projections, the rotary positional encoding and the attention itself. Its input is a random token grid of shape $[1, PC, d_{\text{model}}]$ drawn from a fixed seed. The grid is channel-major for Timer-XL and patch-major for NSTA, the order each model uses. NSTA uses the fixed budgets $k_p = 2$ and $k_c = 256$ for every C; at $C = 256$, k_c therefore covers all channels. Parameters are kept in FP32 and computation runs under BF16 autocast. The Traffic and GTWSF measurements on the H100 instead time each trained forecaster at its own configuration, on real, complete training windows. A training step is one forward pass in `train()` mode, the training MSE loss and the backward pass, with gradients reset before each call and no optimizer step. Every model is compiled, and the compilation is verified after warm-up. Computation runs in FP32 with TF32 enabled for matmul and cuDNN. After 10 warm-up calls, we time 50 synchronized calls cycling over three cached batches and report the median.

Compilation. Each timed layer is compiled with `torch.compile`, so every call replays a CUDA graph with no per-kernel launch overhead. Before timing, we verify that the graph has no breaks and that CUDA graphs were not skipped. After timing, we verify that no recompilation occurred. In NSTA, the routing gathers run on an FP32 copy of their BF16 inputs, which leaves the forward values unchanged.

Timing. Each configuration runs in its own process on one NVIDIA H100 80GB (PyTorch 2.6.0, CUDA 12.4, cuDNN 9.2). We time inference: `eval()` mode under `torch.inference_mode()`. We discard 20 warm-up calls, which include compilation and graph capture, then time 300 calls in three blocks of 100. Each call is bracketed by CUDA events and synchronized. We report the median latency.

Dense baseline implementation. Timer-XL's Time Attention is implemented with PyTorch's fused `scaled_dot_product_attention` kernel. It keeps Timer-XL's own channel-major token order, runs in the same precision as NSTA, and is compiled and replayed as a CUDA graph under the same protocol.

B.5 Branch contribution

The gate $g^b_{p,c}$ alone does not measure how much the branch b contributes to the layer: it is a scalar applied to $\mathbf{o}^b_{p,c}$. We measure the contribution after the output projection $\mathbf{W}^O$, which distributes over Eq. 6:

$$\mathbf{y}_{p,c} \;=\; \mathbf{W}^O \mathbf{o}_{p,c} \;=\; \sum_{b\in\mathcal{B}} \underbrace{\mathbf{W}^O\big(g^b_{p,c}\,\mathbf{o}^b_{p,c}\big)}_{\mathbf{y}^b_{p,c}} \tag{9}$$

We define the contribution of branch b as the projection of its term onto that vector:

$$\rho^b_{p,c} \;=\; \frac{\big\langle \mathbf{y}^b_{p,c},\, \mathbf{y}_{p,c}\big\rangle}{\big\|\mathbf{y}_{p,c}\big\|^2}, \qquad \sum_{b\in\mathcal{B}} \rho^b_{p,c} = 1 \tag{10}$$

We report ρ^b averaged over every token (p, c) and over three seeds.

C Supplementary Results

C.1 Ablation Studies

Branch ablation. The compression and selection branches are progressively added to the univariate path (Table 7), holding the backbone, selection budgets, and training schedule fixed. The full model achieves the lowest MSE on Weather, Traffic, and GTWSF-Temp. Compression alone does not consistently improve performance, whereas adding selection recovers or improves accuracy across all datasets, consistent with complementary roles for coarse cross-channel context and the selected context. Each configuration is trained with three random seeds; we report the mean and standard deviation. Because selection consumes the pooling weights produced by compression (Eq. 1), selection cannot be enabled without compression, and the univariate-plus-selection configuration is therefore not evaluated.

Table 7: Ablation of the NSTA branches. Results are mean $\pm$ standard deviation over three seeds; lower is better. Lookback window is 672 for Weather and Traffic and 48 for GTWSF-Temp. Selection branches use $k_p = 4$, $k_c = 32$ on Traffic and Weather datasets and $k_p = 2$ and $k_c = 256$ for GTWSF-Temp.

Dataset	Branches			Metrics	
	Uni.	Comp.	Select.	MSE ↓	MAE ↓
Weather	✓			0.245 ± 0.007	0.279 ± 0.005
	✓	✓		0.245 ± 0.004	0.277 ± 0.001
	✓	✓	✓	$\mathbf{0.234 \pm 0.003}$	$\mathbf{0.272 \pm 0.001}$
Traffic	✓			0.391 ± 0.002	0.265 ± 0.003
	✓	✓		0.396 ± 0.008	0.269 ± 0.004
	✓	✓	✓	$\mathbf{0.386 \pm 0.002}$	$\mathbf{0.261 \pm 0.002}$
GTWSF-Temp	✓			7.828 ± 0.003	1.874 ± 0.001
	✓	✓		7.612 ± 0.015	1.849 ± 0.002
	✓	✓	✓	$\mathbf{7.481 \pm 0.009}$	$\mathbf{1.831 \pm 0.001}$

Selection strategy. To test whether the ranking itself carries signal rather than merely acting as a regularizer, ranked selection is compared with uniform random selection of the same number of direct interactions. The random subset is resampled at every forward pass, with all other components, hyperparameters, and compute budgets held fixed (Table 8). Ranked selection performs better on all three datasets.

The selection-branch routing prior is further ablated on Traffic across five values of k_c, with $k_p = 4$. Enabling the prior improves MSE at every budget, with a mean reduction of 2.1% across matched runs (Table 9).

Table 8: Ablation study on the selection strategy. Results are reported as mean ± standard deviation over three seeds. Lower is better. Lookback is 672 for Traffic and ECL, 96 for GTWSF-Temp.

Dataset	k_p	k_c	**Selection Strategy**	MSE ↓	MAE ↓
Traffic	2	32	Top-ranked	**0.386 ± 0.003**	**0.262 ± 0.002**
			Random	0.413 ± 0.010	0.274 ± 0.008
ECL	2	32	Top-ranked	**0.160 ± 0.001**	**0.251 ± 0.002**
			Random	0.163 ± 0.001	0.254 ± 0.003
GTWSF-Temp	2	256	Top-ranked	**7.12 ± 0.048**	**1.767 ± 0.004**
			Random	7.39 ± 0.124	1.778 ± 0.016

Table 9: Effect of the selection-branch routing prior on the Traffic dataset. Each entry is the mean over the $\{96, 192, 336, 720\}$ rolling horizons, averaged over three seeds. Δ is the paired MSE improvement from enabling the prior. All configurations use $k_p = 4$ and lookback 672.

	No prior		Prior		Δ	
k_c	MSE	MAE	MSE	MAE	MSE	%
32	0.3922 ± 0.0039	0.2637 ± 0.0030	**0.3857** ± 0.0024	**0.2576** ± 0.0013	0.0065	1.7
64	0.3903 ± 0.0034	0.2618 ± 0.0012	**0.3852** ± 0.0020	**0.2607** ± 0.0038	0.0051	1.3
128	0.3897 ± 0.0041	0.2611 ± 0.0016	**0.3817** ± 0.0013	**0.2564** ± 0.0003	0.0080	2.1
256	0.3923 ± 0.0045	0.2618 ± 0.0029	**0.3808** ± 0.0023	**0.2565** ± 0.0010	0.0115	3.0
512	0.3891 ± 0.0025	0.2620 ± 0.0031	**0.3822** ± 0.0009	**0.2561** ± 0.0006	0.0069	1.8
Avg.	0.3907	0.2621	**0.3831**	**0.2575**	0.0076	1.95

C.2 Multivariate Forecasting Results

Table 10: Full multivariate forecasting results. We perform rolling forecasts with a single model trained on each dataset, using a lookback window of 672 and prediction horizons in $\{96, 192, 336, 720\}$.

Models		**Timer-XL (with NSTA)**		Timer-XL (2025b)		Timer* (2024b)		UniTST* (2025a)		iTransformer* (2024a)		DLinear* (2023)		PatchTST* (2023)		TimesNet* (2023a)		Stationary* (2022)		Autoformer* (2021)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.371	0.404	**0.369**	**0.400**	0.371	0.404	0.379	0.415	0.387	0.418	**0.369**	**0.400**	0.373	0.403	0.452	0.463	0.452	0.478	0.467	0.499
	192	**0.401**	0.424	0.411	0.427	0.407	0.429	0.415	0.438	0.416	0.437	0.405	**0.422**	0.405	0.425	0.474	0.477	0.484	0.510	0.492	0.523
	336	**0.415**	**0.436**	0.433	0.442	0.434	0.445	0.440	0.454	0.434	0.450	0.435	0.445	0.423	0.440	0.493	0.489	0.511	0.522	0.519	0.531
	720	**0.441**	**0.461**	0.449	0.466	0.461	0.466	0.482	0.482	0.447	0.473	0.493	0.508	0.445	0.471	0.560	0.534	0.571	0.543	0.589	0.560
	Avg	**0.407**	**0.431**	0.415	0.434	0.418	0.436	0.429	0.447	0.421	0.445	0.426	0.444	0.412	0.435	0.495	0.491	0.505	0.513	0.517	0.528
ETTh2	96	0.297	0.351	0.298	0.354	**0.285**	**0.344**	0.343	0.398	0.304	0.362	0.305	0.371	0.289	0.347	0.340	0.374	0.348	0.403	0.358	0.397
	192	**0.356**	**0.391**	0.367	0.400	0.365	0.400	0.376	0.420	0.372	0.407	0.412	0.439	0.360	0.393	0.402	0.414	0.408	0.448	0.435	0.451
	336	**0.370**	**0.410**	0.399	0.429	0.412	0.440	0.399	0.435	0.418	0.440	0.527	0.508	0.389	0.420	0.452	0.452	0.424	0.457	0.454	0.475
	720	0.432	0.460	0.447	0.470	0.468	0.487	0.419	0.457	0.463	0.476	0.830	0.653	**0.398**	**0.440**	0.462	0.468	0.448	0.476	0.479	0.492
	Avg	0.364	0.403	0.378	0.413	0.382	0.418	0.384	0.428	0.389	0.421	0.518	0.493	**0.359**	**0.400**	0.414	0.427	0.407	0.446	0.431	0.454
ETTm1	96	0.302	0.347	0.299	0.349	**0.281**	**0.338**	0.289	0.348	0.311	0.365	0.307	0.350	0.285	0.346	0.338	0.375	0.414	0.414	0.466	0.466
	192	0.354	0.376	0.354	0.380	0.330	**0.368**	0.332	0.375	0.353	0.390	0.337	**0.368**	**0.329**	0.372	0.371	0.387	0.524	0.482	0.504	0.496
	336	0.390	0.399	0.394	0.404	0.367	0.393	0.365	0.397	0.387	0.411	0.366	**0.387**	**0.363**	0.394	0.410	0.411	0.541	0.497	0.574	0.530
	720	0.449	0.435	0.463	0.443	0.432	0.433	0.421	0.431	0.452	0.445	**0.419**	**0.419**	0.421	0.426	0.478	0.450	0.578	0.509	0.596	0.558
	Avg	0.374	0.389	0.378	0.394	0.352	0.383	0.352	0.388	0.376	0.403	0.357	**0.381**	**0.349**	0.385	0.399	0.406	0.514	0.475	0.535	0.512
ETTm2	96	0.182	0.264	0.182	0.265	0.175	**0.257**	0.171	0.260	0.183	0.272	**0.167**	0.263	0.172	0.259	0.187	0.267	0.237	0.306	0.255	0.339
	192	0.249	0.309	0.242	0.306	0.239	0.301	**0.228**	**0.230**	0.250	0.315	0.230	0.311	0.233	0.299	0.249	0.309	0.330	0.387	0.279	0.335
	336	0.301	0.343	0.296	0.342	0.293	0.342	0.282	0.336	0.311	0.356	0.298	0.361	**0.280**	**0.331**	0.321	0.351	0.404	0.424	0.331	0.374
	720	0.389	0.400	0.382	0.395	0.392	0.407	0.380	0.398	0.417	0.419	0.432	0.446	**0.357**	**0.382**	0.497	0.403	0.525	0.486	0.413	0.450
	Avg	0.280	0.329	0.275	0.327	0.275	0.327	0.265	**0.306**	0.290	0.340	0.282	0.345	**0.261**	0.318	0.314	0.333	0.374	0.401	0.320	0.374
ECL	96	0.128	0.221	**0.127**	**0.220**	0.129	0.221	0.130	0.225	0.133	0.229	0.138	0.238	0.132	0.232	0.184	0.288	0.185	0.287	0.256	0.357
	192	0.149	0.241	**0.144**	**0.237**	0.148	0.239	0.150	0.244	0.158	0.258	0.152	0.251	0.151	0.250	0.192	0.295	0.282	0.368	0.291	0.376
	336	0.167	0.259	**0.159**	**0.253**	0.164	0.256	0.166	0.262	0.168	0.262	0.167	0.268	0.171	0.272	0.200	0.303	0.289	0.377	0.290	0.379
	720	0.207	0.295	**0.193**	**0.284**	0.201	0.289	0.206	0.297	0.205	0.294	0.203	0.302	0.222	0.318	0.228	0.325	0.305	0.399	0.320	0.403
	Avg	0.163	0.254	**0.156**	**0.249**	0.161	0.251	0.163	0.257	0.164	0.258	0.165	0.265	0.169	0.268	0.201	0.303	0.265	0.358	0.289	0.379
Traffic	96	0.347	0.240	**0.345**	**0.237**	0.348	0.240	0.359	0.250	0.353	0.259	0.399	0.285	0.359	0.255	0.593	0.315	0.610	0.322	0.675	0.412
	192	0.369	0.251	**0.366**	**0.247**	0.369	0.250	0.373	0.257	0.373	0.267	0.409	0.290	0.377	0.265	0.596	0.317	0.626	0.346	0.679	0.423
	336	0.387	0.262	**0.383**	**0.256**	0.388	0.260	0.386	0.265	0.386	0.275	0.422	0.297	0.393	0.276	0.600	0.319	0.633	0.352	0.688	0.440
	720	0.439	0.292	0.426	**0.281**	0.431	0.285	**0.421**	0.286	0.425	0.296	0.461	0.319	0.436	0.305	0.619	0.335	0.651	0.366	0.693	0.457
	Avg	0.386	0.261	**0.380**	**0.255**	0.384	0.259	0.385	0.265	0.384	0.274	0.423	0.298	0.391	0.275	0.602	0.322	0.630	0.347	0.684	0.433
Weather	96	0.155	0.206	0.160	0.210	0.151	**0.202**	0.152	0.206	0.174	0.225	0.169	0.229	**0.149**	**0.202**	0.169	0.228	0.185	0.241	0.355	0.409
	192	0.201	0.250	0.208	0.254	0.196	**0.245**	0.198	0.249	0.227	0.268	0.211	0.268	**0.194**	**0.245**	0.222	0.269	0.286	0.325	0.421	0.450
	336	0.254	0.290	0.262	0.295	0.249	0.288	0.251	0.291	0.290	0.309	0.258	0.306	**0.244**	**0.285**	0.290	0.310	0.323	0.347	0.452	0.465
	720	0.328	0.342	0.338	0.347	0.330	0.344	0.322	0.340	0.374	0.360	0.320	0.362	**0.317**	**0.338**	0.376	0.364	0.436	0.401	0.513	0.496
	Avg	0.234	0.272	0.242	0.276	0.232	0.270	0.231	0.272	0.266	0.291	0.239	0.291	**0.226**	**0.268**	0.264	0.293	0.308	0.329	0.435	0.455
Solar-Energy	96	0.186	0.230	0.182	**0.224**	0.212	0.230	0.190	0.240	0.183	0.265	0.193	0.258	**0.168**	0.237	0.180	0.272	0.199	0.290	0.206	0.296
	192	0.212	0.251	0.214	**0.246**	0.232	**0.246**	0.223	0.264	0.205	0.283	0.214	0.274	**0.189**	0.257	0.199	0.286	0.243	0.307	0.254	0.328
	336	0.226	0.266	0.239	0.264	0.237	**0.253**	0.250	0.283	0.224	0.299	0.233	0.291	**0.212**	0.277	0.220	0.301	0.264	0.322	0.272	0.330
	720	0.257	0.290	0.287	0.292	0.252	**0.266**	0.292	0.311	**0.239**	0.316	0.246	0.307	0.240	0.305	0.251	0.321	0.310	0.339	0.326	0.347
	Avg	0.220	0.259	0.230	0.256	0.233	**0.249**	0.241	0.275	0.213	0.291	0.222	0.283	**0.202**	0.269	0.213	0.295	0.254	0.315	0.265	0.325
First-place count		6	5	10	**13**	2	10	2	2	1	0	3	6	**17**	9	0	0	0	0	0	0

* Results taken from the Timer-XL paper (Liu et al., 2025b).

C.3 Analysis

Channel routing. Figures 9 and 10 compare the channel sets retained on each dataset through their pairwise Jaccard overlap. The two models organize their selection differently. On Traffic, most selections of layers 2 to 4 retain largely the same channels. On GTWSF-Temp, the selections split into about three groups, with high overlap within each group. On Traffic, the few dominant directions make the same sensors informative for most selections, which converge on them. On GTWSF-Temp, no single subset of stations summarizes the cross-section, and different selections retain different, complementary subsets, which is consistent with the larger share of output carried by the selection branch. Figure 11 maps the stations retained by individual heads to examine whether this specialization also has a spatial counterpart.

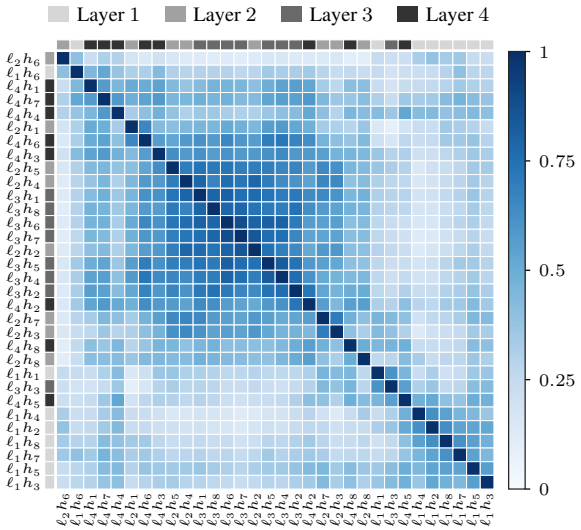


Figure 9: Channel-selection overlap on Traffic ($k_c = 256$, $k_p = 2$). Lookback window is 672. Each cell is the Jaccard index between the top-k_c channel sets of two selections ($\ell_i h_j$: head j of layer i), computed over all test windows. Rows and columns are ordered by hierarchical clustering, side bars indicate the layer.

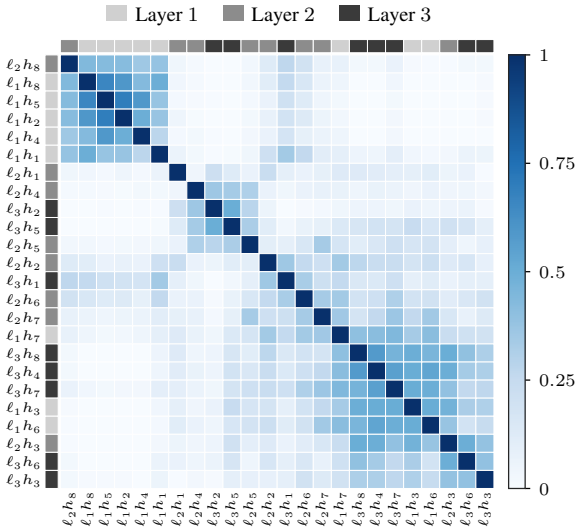


Figure 10: Channel-selection overlap on GTWSF-Temp ($k_c = 256$, $k_p = 2$). Lookback window is 96. Same construction as Figure 9. Selections form about three groups with high internal overlap. Each group mixes selections from different layers.

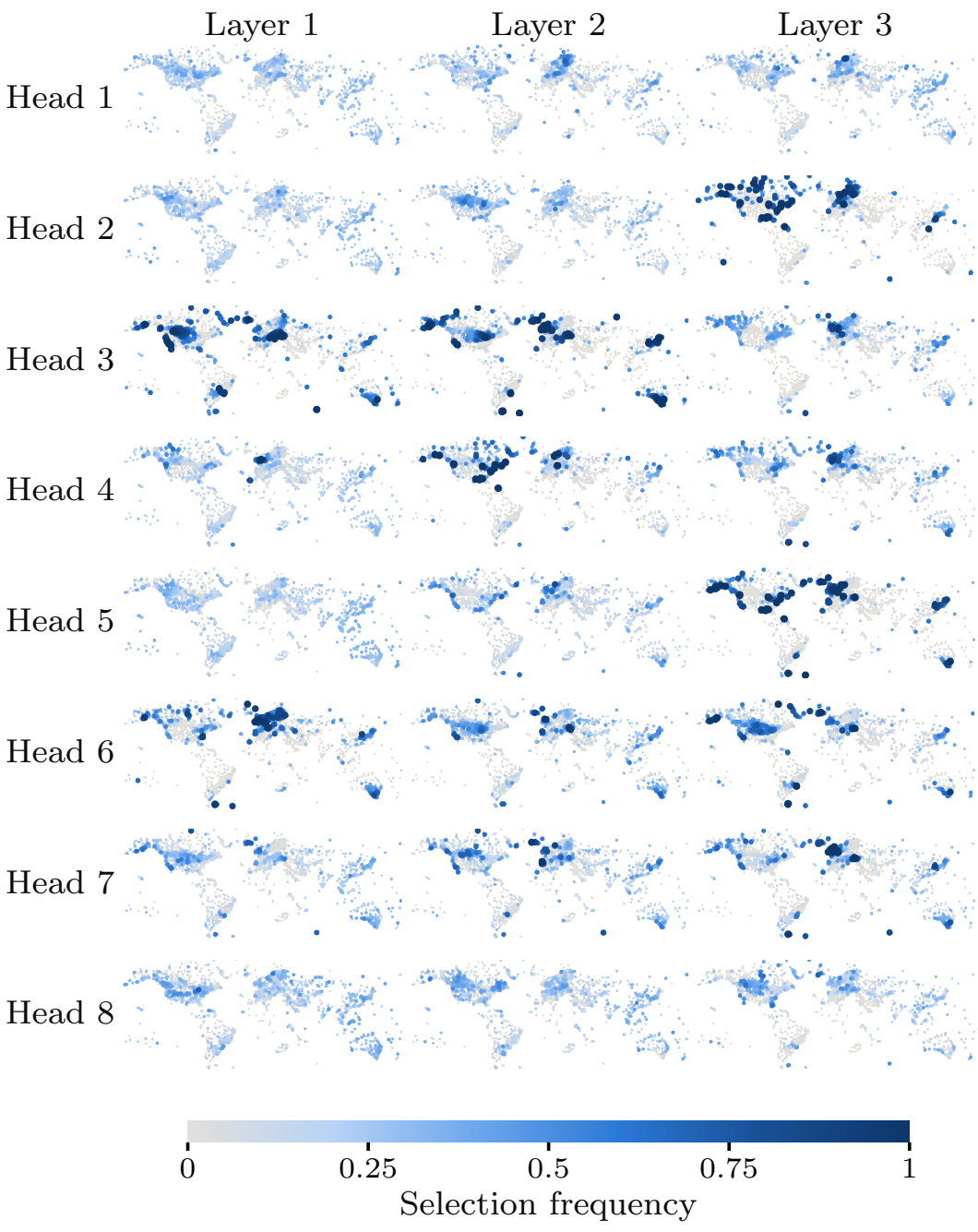


Figure 11: Selection frequency of each station per head: the fraction of the 24 hourly forecast windows ending 2020-09-08 18:00 UTC in which the head's channel selection selected that station. Gray dots are stations never selected.

Patch routing. Figures 12 and 13 report, for the forecast query, the fraction of test windows in which each selection keeps each key patch ($k_c = 256$, $k_p = 2$). These maps record which past patches reach the query. The two models again organize their selections differently. On Traffic ($L = 8064$, $P = 84$), selections concentrate on the most recent patches but are not confined to them. The selection branch thus retrieves tokens from across the long context rather than only from recent history. On GTWSF-Temp ($L = 96$, $P = 4$), the most recent patch is kept in about 80% of windows at every depth, and the second choice moves forward in time with depth.

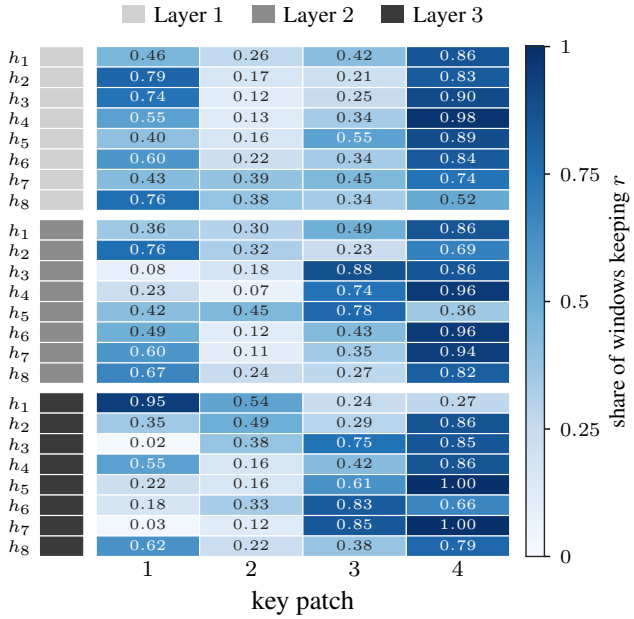


Figure 12: Patch-selection frequencies on GTWSF-Temp ($L = 96$, $k_c = 256$, $k_p = 2$). Each row represents a head within a layer, each column is a key patch. Values indicate the fraction of test windows in which the forecast query selects that patch.

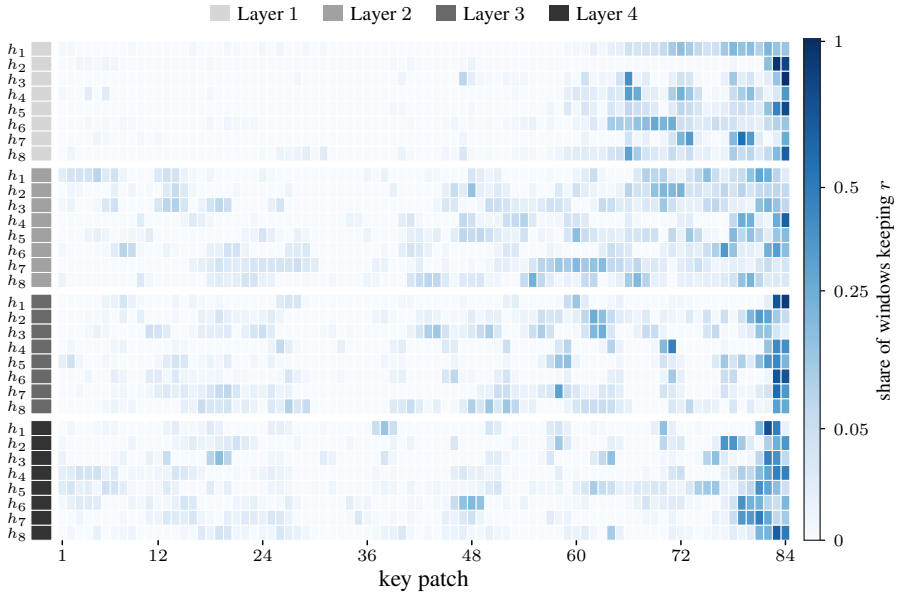


Figure 13: Patch-selection frequencies on Traffic ($L = 8064$, $k_c = 256$, $k_p = 2$). Rows represent heads grouped by layer, columns correspond to the 84 key patches. Selections favor recent patches while retaining access to earlier context.

C.4 SPARSITY SENSITIVITY

We examine how forecasting accuracy depends on the selection budget k_c and k_p. Figures 14 and 15 report MSE as k_c varies for $k_p \in \{1, 2\}$, with lookback windows fixed at $L = 48$ on GTWSF-Temp and $L = 672$ on Traffic. The datasets exhibit different responses: increasing k_c generally improves accuracy on GTWSF-Temp, whereas Traffic achieves its lowest error at an intermediate budget. The effect of k_p is comparatively modest over the tested configurations.

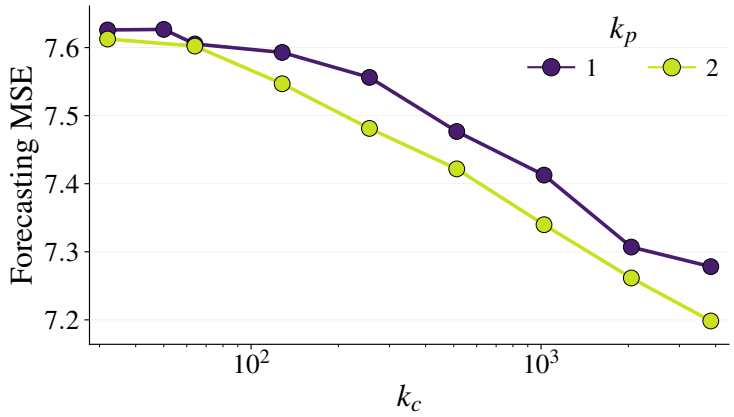


Figure 14: Forecasting MSE (horizon 24) on GTWSF-Temp as a function of the channel selection budget k_c for $k_p \in \{1, 2\}$. Increasing k_c consistently improves accuracy, while the effect of k_p is comparatively small.

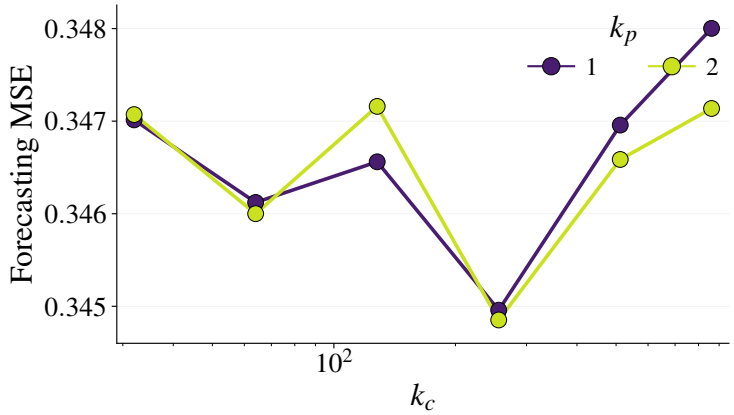


Figure 15: Forecasting MSE (horizon 96) on Traffic as a function of the channel selection budget k_c across $k_p \in \{1, 2\}$. Performance depends only weakly on k_p and exhibits a non-monotonic trade-off with k_c.