EXPLAINING TIME SERIES FORECASTING WITH HORIZON-RESOLVED ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in explaining time series (TS) models have produced methods that identify which past values a prediction depends on. However, most existing methods return a *single* importance vector, assuming that every predicted step depends on the *same* past values. In this paper, we show that this assumption does not hold, as *different forecast steps depend on different past values*. Motivated by this observation, we propose **H**orizon-**R**esolved e**X**planation (**HRX**), which adds a *horizon axis* to the explanation, so that every forecast step receives its *own* importance map. HRX is a simple yet effective plug-in framework with three components: 1) an **estimator** that reads these maps out of any differentiable forecaster without modifying the TS backbone, 2) an **evaluation protocol** that validates the horizon axis by measuring how much a single forecast step changes when the inputs an importance map ranks highest are removed, and 3) a **rank criterion** that predicts in advance whether the axis is worth resolving on a given TS. We further show that this stepwise dependence is *low-dimensional*, as the explanations of all steps are built from a few shared maps whose number grows far more slowly than the forecast length. Extensive experiments across various backbones and datasets show that the improvement comes from the horizon axis and holds for estimators of previous explanation methods. Code is available at `https://anonymous.4open.science/r/HRX2-BA0B`.

1 INTRODUCTION

Time series (TS) forecasting is widely used in various fields, including finance (Lee et al., 2026a;c) and traffic (Liu et al., 2024a). A range of TS forecasting methods have been developed based on different architectures, such as Transformers (Liu et al., 2024a), multi-layer perceptrons (MLPs) (Zeng et al., 2023; Lee et al., 2024), and convolutional networks (Wu et al., 2023). In parallel, TS explanation methods ask *why a model produces a particular output* (Queen et al., 2023).

However, existing TS explanation methods return a *single explanation vector* $e \in \mathbb{R}^L$ for the entire output, whether that output is a class label (classification) or an entire forecast horizon (forecasting). A forecast, unlike a class label, is a trajectory of H values, and a single vector collapses it into one summary. This implicitly assumes that every future value depends on the *same* past values.

In this paper, we argue that ***each forecast step needs its own explanation***, not one shared across all horizons, since different steps may rely on different past values. Figure 1 shows this on a synthetic series, where three forecast steps of the same forecast rely on three different parts of the past. To this end, we propose **H**orizon-**R**esolved e**X**planation (**HRX**), a simple yet effective plug-in framework that explains each forecast step separately instead of the entire forecast as a whole. Specifically, it replaces the explanation *vector* $e \in \mathbb{R}^L$, which is shared across all forecast steps, with a *matrix* $E \in \mathbb{R}^{H \times L}$, one row per forecast step, and applies to any differentiable forecaster without modifying the TS backbone. Note that our contribution is the *horizon axis rather than the estimator*

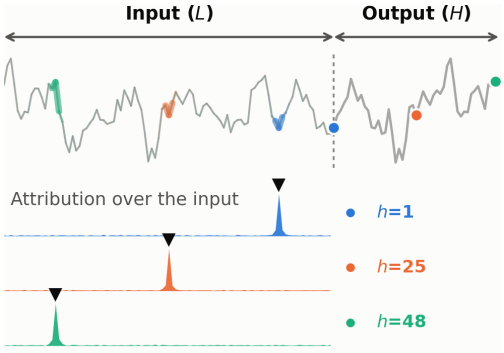


Figure 1: **Different steps, different evidence.** Each step needs its own explanation.

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.

Table 1: **Comparison with existing TS explanation methods.** Only ours resolves the horizon axis, returning one importance map per forecast step rather than one for the entire forecast, while remaining a plug-in that needs no change to the backbone and no auxiliary training.

Method	Resolved axes		Requirements		Explanation
	Time	Horizon	Plug-in	No train.	
Surrogate methods[1]	✓				$\mathbb{R}^L$
Learned mask methods[2]	✓		✓		$\mathbb{R}^L$
Gradient and Shapley methods[3]	✓		✓	✓	$\mathbb{R}^L$
HRX (Ours)	✓	✓	✓	✓	$\mathbb{R}^{H\times L}$

[1]TimeX++ (ICML'24).
[2]Dynamask (ICML'21), ExtremalMask (ICML'23), ContraLSP (ICLR'24).
[3]Saliency (ICLR-W'14), Integrated gradients (ICML'17), TSR (NeurIPS'20), TimeSHAP (KDD'21), TsSHAP (arXiv'23).

that produces each row of E, since the estimator of any existing explanation method can fill a row and we adopt an input gradient as our default estimator only because it is the simplest.

We further propose an evaluation protocol that scores one forecast step at a time, which no prior metric can do. We also propose a rank criterion, the rank of E, that says in advance whether the axis brings any gain over the shared vector on a given TS. The main contributions are summarized as:

- We show that TS forecasters read different parts of the lookback window for different forecast steps, and that the single explanation vector returned by prior works cannot express this structure.
- We propose **H**orizon-**R**esolved e**X**planation (**HRX**), a simple yet effective plug-in framework that resolves the explanation along the horizon axis, so that the explanation becomes a matrix with one row per forecast step. HRX is applicable to the estimator of any explanation method and to any differentiable forecaster, and needs no change to the TS backbone or auxiliary training.
- We propose an evaluation protocol, since established TS explanation metrics score one label per input and therefore cannot score a single forecast step. It measures how much one step degrades when we delete what its own row of E marks rather than what the shared vector marks.
- We show that the horizon axis is low-dimensional, as a handful of leading directions of E recovers most of the gain while a single direction recovers none of it. We turn the rank of E into a criterion that predicts the gain from the matrix alone, without running the protocol.
- We conduct extensive experiments over 56K runs across 26 backbones and 15 benchmarks, and show that the improvement comes from the horizon axis rather than any particular estimator. The axis also holds under the explanation metrics of prior works and against known ground truth.

2 Related Works

Three lines of related work (Appendix P.1). Work related to ours falls into three lines:

- TS forecasting models (Nie et al., 2023; Liu et al., 2024a).
- Gradient-based explanation (Simonyan et al., 2013; Sundararajan et al., 2017).
- Perturbation-based explanation (Crabbé & van der Schaar, 2021; Enguehard, 2023).

The first is what we attach to, since our framework treats any forecaster as a black box and changes nothing inside it. The second reads importance from the derivative of the output with respect to the input, which is what keeps it cheap, while the third changes the input and reads importance from how much the output moves, which is what makes it expensive. Details are discussed in Appendix P.1.

Positioning of our work. Every method above returns one explanation vector per instance, and that vector is attached *to the whole forecast rather than to any one step*. Table 1 compares TS explanation methods along the axes an explanation can resolve. Every method resolves the time axis, while ours is the only one that also resolves the *forecast step*. Additional related works are discussed in Appendix P.2, and details of the backbones we use are in Appendix C.

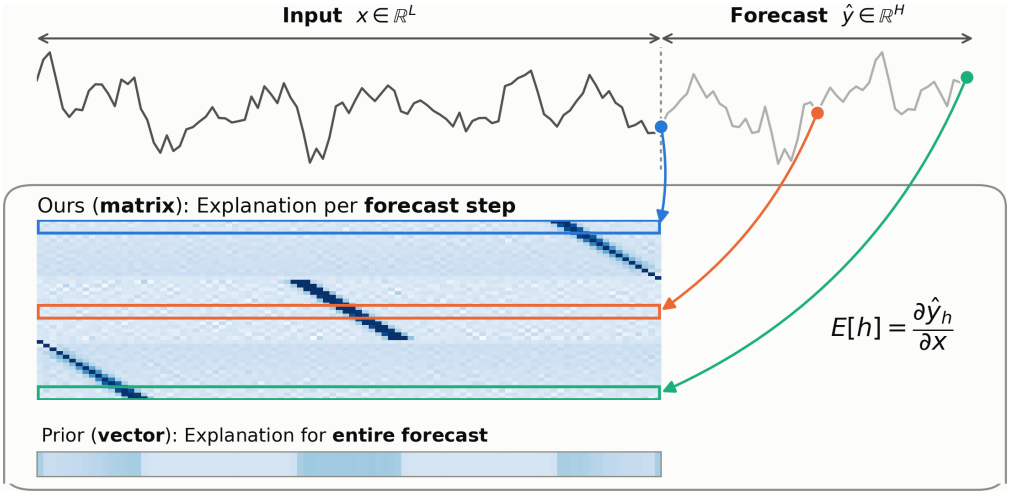


Figure 2: **Explanation matrix E.** A forecaster reads the input and outputs H steps, and HRX returns one row of E per step, where each colored row explains the step in the same color above.

3 Methodology

The framework has four parts: (Section 3.1) what an explanation should be, (Section 3.2) how we compute it, (Section 3.3) how we validate it, and (Section 3.4) how many directions it needs.

- **Explanation matrix.** From one shared vector to a matrix with one row per forecast step.
- **Estimator.** The input gradient of one forecast step fills a row, and any estimator can replace it.
- **Evaluation protocol.** A test of whether a row explains its own step better than another row.
- **Rank criterion.** The singular values of E say how many of the H rows are effectively distinct.

3.1 Explanation Matrix

From explanation vector to matrix. Let $f_\theta : \mathbb{R}^L \to \mathbb{R}^H$ be a trained forecaster that maps a lookback window x of length L to a forecast $\hat{y} = f_\theta(x)$ of length H. Existing TS explanation methods produce $e \in \mathbb{R}^L$, a single vector over the lookback that every forecast step has to share, which we call the *shared vector* throughout. As shown in Figure 2, we instead ask for

$$E \in \mathbb{R}^{H \times L}, \qquad E[h, t] = \text{importance of input } t \text{ for the forecast at step } h. \tag{1}$$

The shared vector of prior works is recovered as $e = \sum_h E[h]$, so E *strictly refines it rather than replacing it*. Note that E is only useful *when its rows actually differ*, since a matrix whose rows are copies of one vector carries no more than e does, and Section 3.4 makes that condition measurable. We take the univariate setting as our default, and the extension to multivariate TS follows below.

Extension to multivariate TS. A forecaster with C channels, $f_\theta : \mathbb{R}^{L \times C} \to \mathbb{R}^{H \times C}$, carries a second input axis, so the explanation of prior work becomes $e \in \mathbb{R}^{L \times C}$ and ours becomes

$$\tilde{E} \in \mathbb{R}^{H \times L \times C}, \qquad \tilde{E}[h, t, c] = \text{importance of input } (t, c) \text{ for the forecast at step } h. \tag{2}$$

Note that *nothing in the framework depends on the channel axis*, as a row belongs to one forecast step no matter how many input axes there are. We choose to report the time axis alone, taking $E[h, t] = \sum_c |\tilde{E}[h, t, c]|$ and keeping the $H \times L$ form throughout, for two reasons. First, which channels a forecaster reads does not change with the forecast step, so resolving that axis would store the same values once per step (Section 5). Second, the fixed $H \times L$ form makes the explanation the same shape on every dataset, so a rank or a gain is directly comparable across different C.

3.2 Estimator

Gradient of a single forecast step. For a differentiable f_θ, we take the row of E at step h to be the input gradient of that step alone, which is $E[h] = \partial \hat{y}[h] / \partial x$. For a multivariate forecaster we

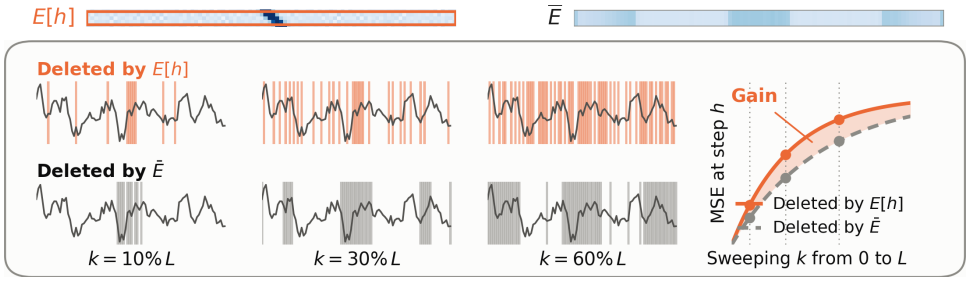


Figure 3: **Evaluation protocol.** For a forecast step h we delete the top-k lookback positions in two orders: (1) by its own row $E[h]$ and (2) by the shared vector $\bar{E}$, and sweep k from 0 to L. The gap between the two error curves, that is $\mathrm{AUC}(h; E[h]) - \mathrm{AUC}(h; \bar{E})$, is the gain.

Table 2: **Explanation quantities.** Each compares two AUCs with different importance vectors (own step $E[h]$ vs. other step $E[\pi(h)]$ vs. shared $\bar{E}$). Last column gives the ideal direction, where a step's own row should break that step more than the shared vector does and another step's row less.

Quantity	AUC comparison	Question it answers	Ideal dir.
g_{own}	$E[h]$ vs. $\bar{E}$	Does **own step's** row explain it better than the shared one?	> 0
$- g_{\text{shuffled}}$	$E[\pi(h)]$ vs. $\bar{E}$	Does **another step's** row explain it better than the shared one?	< 0
$= g_{\text{margin}}$	$E[h]$ vs. $E[\pi(h)]$	Does **own step's** row explain it better than **another step's** row?	$> g_{\text{own}}$

differentiate the channel sum $\sum_c \hat{y}[h, c]$ and then reduce it over input channels as above. Note that HRX is not tied to this gradient, since *any estimator conditioned on a single output coordinate* can fill a row of E. We adopt the gradient form as our default estimator because it is the simplest, needing no optimization of its own, and any of seven other estimators can take its place (Section 4.3).

Efficient implementation: Batched backward calls. A backward call propagates from a single scalar, so a naive implementation needs one pass per forecast step and H passes in total. To fill several rows in one call, we replicate the input, give each replica a different h, and differentiate the sum of the selected outputs, which keeps the rows separate because *each replica is its own leaf in the graph*. The cost of the whole matrix then stays within a small multiple of a forward pass.

3.3 Evaluation Protocol

An explanation is scored by *deletion* (Liu et al., 2024c;d), which *removes the positions the explanation ranks highest*[1] and measures how much the forecast changes (Figure 3). Since removing more positions changes the forecast further, sweeping the number removed traces a curve rather than a scalar, summarized by its area (AUC). As a forecast has H values, we read the error one step at a time and *compare the AUCs of different importance vectors*, giving the quantities of Table 2.

[1] Gain with own explanation (g_{own}). Write $\mathrm{AUC}(h; v)$ for the AUC at step h with importance vector v (i.e., the top-k lookback positions are deleted in the order v ranks them, sweeping k from 0 to L). The error is read at step h alone rather than over the whole forecast, no matter which v chose the deleted positions. Then, the gain with own explanation (g_{own}) compares 1) the row of E that belongs to step h ($E[h]$) against 2) the shared vector $\bar{E} = \frac{1}{H}\sum_{h'} E[h']$, so it asks whether step h is better explained by its own row than by one vector shared across the horizon,

$$g_{\text{own}}(h) \;=\; \mathrm{AUC}\big(h;\, E[h]\big) - \mathrm{AUC}\big(h;\, \bar{E}\big), \qquad g_{\text{own}} \;=\; \frac{1}{|\mathcal{H}|}\sum_{h \in \mathcal{H}} g_{\text{own}}(h). \tag{3}$$

A positive g_{own} says that deleting by a step's own row breaks that step more than deleting by the shared vector does, so the row points at what that step actually uses. A method without a horizon axis is pinned to $g_{\text{own}} = 0$, since the two terms then read the same vector.

[1] Removed positions are resampled from elsewhere in the same window rather than set to zero, which keeps the input in distribution (Hooker et al., 2019).

[2] Gain with other explanation (g_{shuffled}). A positive g_{own} is not enough, as a row of E is concentrated on fewer positions than $\bar{E}$ is, and deleting a few concentrated positions breaks the forecast more than deleting the same number of scattered ones, whether or not they are the right positions. Thus, g_{own} can be positive even when the row is not the right one for step h. To rule that out, we score step h with the row of another step, drawing a random permutation π of $\mathcal{H}$ and using $E[\pi(h)]$,

$$g_{\text{shuffled}}(h) \;=\; \text{AUC}\big(h;\, E[\pi(h)]\big) - \text{AUC}\big(h;\, \bar{E}\big), \qquad g_{\text{shuffled}} \;=\; \frac{1}{|\mathcal{H}|}\sum_{h\in\mathcal{H}} g_{\text{shuffled}}(h). \tag{4}$$

The only difference between g_{shuffled} and g_{own} is which row is used to explain step h. The shared vector keeps a share of every row, including the one that belongs to step h, whereas another step's row keeps none of it, *so the ideal value of g_{shuffled} is negative rather than merely zero*.

[1−2] Shuffle margin (g_{margin}). The decisive quantity of the protocol is their difference,

$$g_{\text{margin}} \;=\; g_{\text{own}} - g_{\text{shuffled}}, \tag{5}$$

which we average over three permutations of $\mathcal{H}$. The shared vector appears in both gains and cancels, leaving $\text{AUC}(h; E[h]) - \text{AUC}(h; E[\pi(h)])$, so the margin compares 1) the row of step h ($E[h]$) directly against 2) the row of another step ($E[\pi(h)]$). The margin therefore measures only *whether a row is used on the step it was computed for*, and is expected not only to be positive but to exceed g_{own}, since the ideal value of g_{shuffled} is negative.

Scale normalization. All three quantities are error differences, so they grow with the variance of the forecast. Without a correction, a dataset whose targets vary more would look more horizon-dependent than one whose targets are flat. We therefore divide each of them by $\text{Var}(\hat{y})$.

3.4 Rank of the Explanation Matrix

The matrix is worth its H rows only when those rows differ. We check it with the singular values of E, which say how many are effectively distinct, taking the effective rank to be the participation ratio (Litwin-Kumar et al., 2017)

$$r(E) \;=\; \big(\textstyle\sum_i \sigma_i^2\big)^2 \big/ \sum_i \sigma_i^4, \qquad \sigma_1 \geq \cdots \geq \sigma_{\min(H,L)} \geq 0, \tag{6}$$

which counts how many directions carry the magnitude of the matrix rather than how many are nonzero. Truncating E to the leading r of those directions writes every row as

$$E_r[h] \;=\; \textstyle\sum_{k=1}^{r} c_{h,k}\, v_k, \qquad c_{h,k} = \sigma_k u_k[h], \tag{7}$$

where u_k and v_k are the singular vectors of E, so that the r maps $v_1, \ldots, v_r \in \mathbb{R}^L$ are shared by every forecast step and only the weights $c_{h,k}$ change with h. At $r = 1$ every row is a positive multiple of v_1, whereas a larger r lets the rows rank the lookback differently. Note that $r(E) = 1$ is the setting of prior works, which pins every quantity of Table 2 to zero. Unlike the evaluation protocol in Section 3.3, $r(E)$ needs only the matrix itself and no deletion sweep, so it says before any measurement whether the axis is worth resolving on a given TS.

4 Experiments

Datasets. We evaluate on 15 benchmarks grouped into two domains. The first is a widely used general-domain suite, which is ETTh1, ETTh2, ETTm1, ETTm2, Electricity, Exchange, PEMS03, PEMS04, PEMS07, PEMS08, Solar, Traffic, and Weather (Wu et al., 2021). The second is a new financial benchmark we assemble from the daily closing prices of the 100 largest listed US companies between 2018 and 2025 (Aroussi, 2024). We add the financial domain, where decisions have to be justified to auditors and regulators (Yeo et al., 2025). Details are discussed in Appendix A.

Backbones. We build on the Time-Series-Library (Wang et al., 2024b) and evaluate 26 backbones spanning linear, convolutional, recurrent, and Transformer families. We add three deliberately minimal architectures, which are a single linear map, a small CNN, and a vanilla Transformer, so that the phenomenon can be separated from any modern design choice. Details are discussed in Appendix A.

Experimental setup. To test the horizon axis where it would have to hold in practice, we sweep the lookback length, the forecast length, and the depth over the ranges the forecasting literature uses, and we run every configuration with three random seeds. Details are discussed in Appendix A.

Table 3: **Explanation performance on the synthetic dataset.** On the ground-truth metrics the explanation matrix beats the shared vector. On the proposed metrics the shared vector has no per-step value, so every g is zero, while ours is ideally $g_{\text{own}} > 0$, $g_{\text{shuffled}} < 0$, $g_{\text{margin}} > g_{\text{own}}$. **Bold** marks the outcome the horizon axis predicts, here and in every table that follows.

Backbone	Proposed metrics[1]			Ground-truth metrics[2]							
				AUPRC		AUROC		AUP		AUR	
	g_{own}	g_{shuffled}	g_{margin}	Vector	Matrix	Vector	Matrix	Vector	Matrix	Vector	Matrix
Linear	**0.142**	**−0.215**	**0.357**	0.586	**0.983**	0.961	**0.998**	0.132	**0.154**	0.960	**0.985**
CNN	**0.161**	**−0.234**	**0.395**	0.578	**0.993**	0.962	**0.999**	0.132	**0.154**	0.963	**0.984**
Transformer	**0.121**	**−0.252**	**0.373**	0.577	**0.993**	0.960	**0.999**	0.130	**0.154**	0.963	**0.985**

[1]A method without a horizon axis has only the shared vector, so $E[h]$ and $\bar{E}$ are the same vector there and every g is *exactly zero*.
[2]AUPRC, AUROC, AUP and AUR compare an explanation against the positions that actually generated the target, which we know here because we built the generator. They are therefore *defined on this synthetic data and not on real data*.

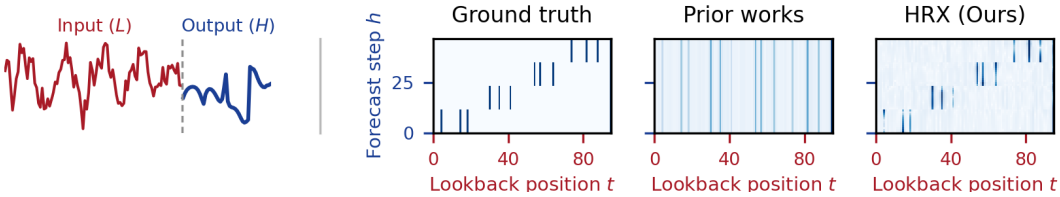


Figure 4: **Explanation on synthetic dataset.** [Left] TS forecasting visualization. [Right] 1) Ground truth, 2) shared vector that stands in for prior works, and 3) explanation matrix that HRX returns.

Metrics. We report our own protocol of Section 3.3, where g_{own} asks whether a step is better explained by its own row than by one shared vector and g_{margin} asks how much of that needs the rows matched to their steps. We also report metrics of prior works, which are 1) AUPRC, AUROC, AUP and AUR on the synthetic data, where we know the ground truth, and 2) Comprehensiveness, AOPC, Sufficiency and Insertion on the real-world data, with details discussed in Appendix F.3.

4.1 Sanity Check on Synthetic Dataset

Why this check comes first. On real data, we never know which past values a forecast step actually depends on. We therefore cannot tell whether our method reports the truth or only an artifact of the metric. To find out, we build six kinds of synthetic dataset in which we decide that dependence ourselves and check what our method returns, which we assess both qualitatively and quantitatively.

Synthetic datasets. The target of each generator is an explicit function of a few chosen lookback positions, so *the dependence is fixed by how we build the data* rather than asserted afterwards. Note that $\partial y_h / \partial x_t$ then follows from the definition and is exactly zero at every position we did not choose. Each generator takes the same form $y_h = w^{(h)} \cdot x_{S(h)}$, where $S(h)$ is the small set of lookback positions that step h reads and $w^{(h)}$ is a unit-norm weight vector that varies smoothly with h. They differ only in how $S(h)$ moves with h, and Appendix G defines all 6 of them.

Quantitative results. Table 3 reports the performance over the six synthetic datasets (generators) with three basic backbones, where every cell is a median over 54 runs, namely 6 generators, 3 window shapes, and 3 seeds. The explanation matrix beats the shared vector on all four ground-truth metrics and all three backbones. On our proposed metrics, all three backbones show the ideal direction, $g_{\text{own}} > 0$, $g_{\text{shuffled}} < 0$, and $g_{\text{margin}} > g_{\text{own}}$. Note that these three are exactly zero for prior works, since they score one forecast step at a time and the shared vector has no per-step value.

Qualitative results. To validate that the matrix recovers the true dependence, we compare the positions that actually generated each forecast step with what each method returns on the same trained backbone (Figure 4). Prior works repeat the shared vector down the horizon, which flattens the structure the generator planted into vertical stripes. The explanation matrix keeps one row per forecast step and reproduces that structure, and Appendix G shows the same for every generator.

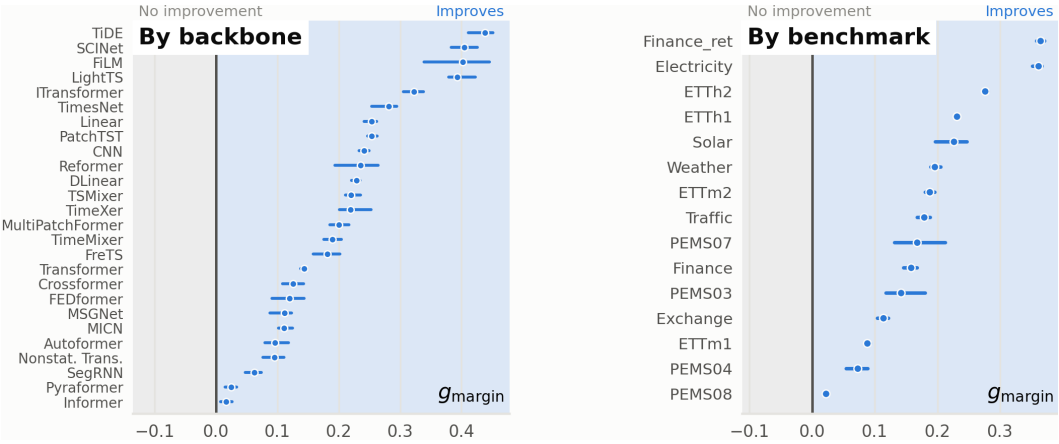


Figure 5: **Results under our own metric (g_{margin}).** [Left] By backbone. [Right] By benchmark. Points are medians and bars are bootstrap 95% confidence intervals, and a row right of zero means the step is explained better by its own row than by another step's row.

4.2 Main Experiments

Results under our own metric. To validate that the horizon axis holds beyond the synthetic setting, we aggregate every real-world run under the protocol of Section 3.3. As shown in Figure 5, g_{margin} is consistently positive across backbones and benchmarks, indicating that *each step is explained better by its own row* than by the row of another step. The gain g_{own} is consistently positive while g_{shuffled} falls below zero, which means *a row handed to the wrong step actively hurts*, and Appendix H has both. All quantities are normalized by the forecast variance, which never changes a run's sign.

Results under metrics of prior works. To validate the horizon axis under the metrics of prior works, which were defined for a model with a single output (e.g., a class label), we re-express four of them for forecasting and score the explanation of step h against the shared vector. Table 4 reports the median and the trimmed mean of that comparison for each metric, and all four metrics of prior works are positive under a sign test over 1,126 runs. The results indicate that *the horizon axis helps regardless of how faithfulness is measured*. The four metrics are re-expressed as follows, with details discussed in Appendix F.3.

Table 4: **Results under metrics of prior works.** Every row compares a step against the shared vector.

Metric	Median	Mean	Positive
Metrics of prior works			
Comprehensiveness	+.0038	+.0305	69.6%
AOPC	+.0030	+.0271	68.7%
Sufficiency	+.0022	+.0114	72.2%
Insertion	+.0320	+.0475	73.9%
Metrics of ours			
g_{own}	+.0888	+.1365	70.3%
g_{shuffled}	−.0530	−.1011	34.6%
g_{margin}	+.1418	+.2376	80.6%

- Comprehensiveness (DeYoung et al., 2020) deletes the highest-ranked positions.
- AOPC (Samek et al., 2017) deletes them and averages over how many are deleted.
- Sufficiency (DeYoung et al., 2020) keeps the highest-ranked positions instead.
- Insertion (Petsiuk et al., 2018) keeps them and reads how far the forecast recovers.

4.3 Application to Other Explanation Methods

Setup. To validate that the horizon axis is what improves the explanation rather than any particular estimator, we apply the axis to the estimators of seven other explanation methods[2] and score them under the identical protocol, on the synthetic data of Section 4.1 and on the real-world benchmarks.

[2]Methods published for classification need one change, since the quantity a mask or a gradient is asked to preserve becomes the forecast at step h instead of a class score. Details are provided in Appendix E.3.

Table 5: Application of HRX to various estimators (synthetic data).

Estimator	AUPRC		AUROC		AUP		AUR	
	Vector	Matrix	Vector	Matrix	Vector	Matrix	Vector	Matrix
Saliency	0.551	0.971	0.954	0.994	0.130	0.151	0.950	0.974
Gradient × input	0.547	0.968	0.952	0.994	0.130	0.151	0.947	0.974
Integrated gradients	0.549	0.968	0.953	0.994	0.130	0.151	0.947	0.974
Occlusion	0.125	0.150	0.688	0.838	0.068	0.084	0.694	0.822
Dynamask	0.455	0.756	0.886	0.947	0.115	0.134	0.882	0.925
ExtremalMask	0.460	0.705	0.853	0.903	0.106	0.118	0.853	0.847
ContraLSP	0.478	0.847	0.888	0.948	0.116	0.123	0.886	0.894
Input gradient	0.550	0.973	0.954	0.995	0.130	0.152	0.950	0.974

Table 6: Application of HRX to various estimators (real-world data).

Estimator	g_{own}		g_{shuffled}		g_{margin}	
	Median	95% CI	Median	95% CI	Median	95% CI
Saliency	0.098	[0.090, 0.108]	−0.167	[−0.179, −0.154]	0.265	[0.250, 0.280]
Gradient × input	0.086	[0.079, 0.095]	−0.142	[−0.150, −0.132]	0.228	[0.215, 0.239]
Integrated gradients	0.077	[0.071, 0.085]	−0.153	[−0.161, −0.145]	0.230	[0.221, 0.242]
Occlusion	0.029	[0.025, 0.036]	−0.044	[−0.049, −0.039]	0.073	[0.067, 0.080]
Dynamask	0.275	[0.254, 0.292]	−0.362	[−0.380, −0.344]	0.637	[0.606, 0.666]
ExtremalMask	0.168	[0.152, 0.187]	−0.193	[−0.208, −0.180]	0.361	[0.338, 0.387]
ContraLSP	0.158	[0.142, 0.177]	−0.229	[−0.242, −0.218]	0.387	[0.364, 0.413]
Input gradient	0.141	[0.129, 0.153]	−0.130	[−0.140, −0.117]	0.271	[0.252, 0.287]

Results on synthetic datasets. The synthetic data admits the ground-truth metrics, since the answer is known in advance. Each metric is read twice, once with the shared vector of prior works and once with our matrix, and each cell averages 26 backbones and 6 generators. As shown in Table 5, resolving the axis wins in almost every cell, indicating that the gain is robust across estimators.

Results on real-world datasets. Note that a method without the horizon axis uses one vector for every step, which makes all three quantities *exactly zero*, and every gain here is what the axis adds. As shown in Table 6, the gain is significantly positive for every estimator, indicating that *the improvement belongs to the horizon axis rather than to any particular estimator*. Every interval spans 26 backbones, 15 benchmarks, 4 window shapes and 3 random seeds.

5 Analysis

Two domains: General & Finance. To analyze the effect by the domain of a dataset, we report the general suite and the financial series apart, and split each by whether the model beats two naive baselines, which are predicting zero and repeating the last value. As shown in Table 7, runs that beat the baselines have a larger g_{own} than runs that do not, in all three groups, indicating that *the explanation gain grows with the forecasting skill of the model*. Note that every group is positive, which means forecasting skill sets the size of the gain rather than whether one exists.

Table 7: **Gain g_{own} by domain.** Each domain is split by whether the model beats both naive baselines.

Domain	g_{own}	95% CI
General	**+0.133**	[+0.129, +0.136]
Beats naive	**+0.155**	[+0.151, +0.159]
Loses to naive	**+0.060**	[+0.055, +0.064]
Financial	**+0.211**	[+0.199, +0.220]
Log price	**+0.061**	[+0.048, +0.072]
Beats naive	**+0.073**	[+0.056, +0.089]
Loses to naive	**+0.055**	[+0.039, +0.067]
Log return	**+0.340**	[+0.325, +0.356]
Beats naive	**+0.548**	[+0.390, +0.695]
Loses to naive	**+0.336**	[+0.322, +0.352]

Visualization of explanation matrices. To see what the matrix looks like in use, we vary the dataset in one group and the backbone in the other. As shown in Figure 6(a), a clear band appears on both the Solar series (for general domain) and the AAPL stock return (for financial domain). Furthermore, the three backbones of Figure 6(b) carry the same diagonal band on one benchmark, and they differ



Figure 6: **Visualization of explanation matrices.** Rows are forecast steps and columns are lookback positions, and the band tracks the step in (a) two datasets and (b) three backbones.

Table 9: **Rank of E.** Each truncation is scored under the protocol of Section 3.3, where rank 1 is pinned to zero.

Truncation	g_{own}	Fraction of full
$r = 1$	$+0.000$	0.00
$r = 2$	$\mathbf{+0.044}$	0.19
$r = 4$	$\mathbf{+0.110}$	0.51
$r = 8$	$\mathbf{+0.153}$	0.71
Full ($r = H$)	$\mathbf{+0.228}$	1.00

Table 10: **Rank predicts the gain.** Runs grouped by the effective rank of E ($r(E)$), where a rank near one means the shared vector of prior works loses nothing.

$r(E)$	Runs	g_{own}	95% CI	Positive
$[1.0, 1.1)$	2,103	$\mathbf{+0.015}$	$[+0.012, +0.018]$	**62%**
$[1.1, 1.25)$	3,977	$\mathbf{+0.074}$	$[+0.069, +0.079]$	**71%**
$[1.25, 1.5)$	5,492	$\mathbf{+0.109}$	$[+0.102, +0.119]$	**75%**
$[1.5, 2.0)$	6,675	$\mathbf{+0.146}$	$[+0.138, +0.150]$	**83%**
$[2.0, 3.0)$	4,751	$\mathbf{+0.250}$	$[+0.244, +0.258]$	**91%**
$[3.0, \infty)$	2,208	$\mathbf{+0.323}$	$[+0.316, +0.329]$	**98%**

only in how many parallel bands they hold. In every panel the important lookback positions shift as the forecast step advances, which a single shared vector cannot express. Appendix J visualizes all the panels at full size, with the naive-baseline verdict and the effective lookback of every row.

Time vs. channel axis. To ask whether the same dissociation appears across channels, we run the same deletion protocol on both input axes of multivariate forecasters. As shown in Table 8, *the time axis gives a clear gain* while *the channel axis gives almost none*, and its effective rank sits near 1, which means the channel rows are copies of one vector. In other words, a forecaster changes *when* it looks as the forecast step moves, but not *which* variable it looks at.

Table 8: **Horizon axis lives on time axis.** Effective rank of E is the participation ratio of its singular values.

Axis	Time	Channel
Gain g_{own}	$+0.026$	$+0.000$
Effective rank	1.72	1.0028

How many maps a forecast needs. To ask how many maps a forecast needs, we truncate E to its leading directions and score every truncation under the identical protocol. As shown in Table 9, a single direction recovers none of the gain, which is what Section 3.4 predicts, while a handful of further directions recover most of it. Table 10 reads the same structure the other way, where the gain rises with the rank and the runs closest to one direction are the ones where the shared vector loses almost nothing. Appendix M and Appendix N give the details of the rank measure and the distance.

Other analysis. The appendix reports three further analyses, and we summarize each one here.

- **Application to zero-shot TSFMs.** The gain is positive for almost every foundation model run zero-shot, so the axis holds even when the model has never seen the series (Appendix O).
- **Forecast length, lookback and depth.** The gain rises with both the forecast length and the lookback length, while depth moves in the opposite direction by far less (Appendix K.1).
- **Cost of each estimator.** The mask methods cost two orders of magnitude more than a gradient, since the axis makes each solve one problem per forecast step (Appendix K.2).

6 Conclusion

We showed that TS forecasters read *different parts of the lookback window for different forecast steps*, and we proposed horizon-resolved explanation, which recovers it as a matrix without modifying the TS backbone, together with a protocol that isolates the horizon axis. Two limitations remain, as no statistic of the raw series predicts the gain, and none of the downstream uses we tried gave a better forecast, which means HRX is a *measurement tool* for now. Future work is to find what makes a series reward the axis, and to turn the matrix into a signal a forecaster can use at training time.

AI USE STATEMENT

We used a generative AI tool to proofread this manuscript, where it checked grammar, wording, and typographical errors in text we had already written, and we received minor coding assistance from the same tool while implementing the experiment code. We also used it to help draft minor details in the appendix, such as clarifications of implementation settings and answers to anticipated questions, and to write scripts that summarize our stored runs into several appendix tables. The research question, the framing of the horizon axis as our contribution, the design of the evaluation protocol, and the main analysis of the results are our own. Every number reported in this paper comes from an experiment we executed, and no citation or reference was produced by a generative model. We reviewed every edit and every line of AI-assisted code we accepted, and checked each AI-assisted appendix statement against our code and stored results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how a trained forecaster uses its input, so it produces diagnostic information about models rather than new predictive capability. The general-domain benchmarks are standard public forecasting datasets, and they contain aggregate measurements from sensors and meters rather than records about individuals. The financial data are daily closing prices of publicly traded companies, retrieved from a public market data source, and they contain no personal or proprietary information. No human subjects were involved and no personally identifiable data were used at any stage. One risk follows from the nature of explanation methods, which is that an explanation map invites a causal reading it does not support. Our protocol addresses this directly, as it compares a perturbed forecast against the model's own unperturbed forecast and therefore measures what the model uses rather than what drives the underlying process. Furthermore, we report a forecasting baseline beside every result, so that an explanation of a model with no forecasting skill is not read as evidence about the world.

REPRODUCIBILITY STATEMENT

Appendix A gives the backbones, benchmarks, swept configurations, and training protocol, and Appendix F gives the deletion curve and the exact form of every metric we report. Appendix O gives the checkpoint, context length, and verification procedure for each pretrained forecaster. Appendix D reports the seed-to-seed variation of our main quantity, Appendix G reports the same conclusion against constructed ground truth, and Section 4.2 reports it under metrics of prior works. Note that every number quoted in the text is generated from the stored per-run records by a script rather than transcribed by hand, so the text and the tables cannot drift apart. We provide the full source code, which includes the estimators, the evaluation protocol, and the data preparation scripts, together with the code to reproduce the values reported in our tables. Appendix R answers the questions we expect about the protocol, the baselines and the statistics, and points to the evidence for each answer.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

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

Ran Aroussi. yfinance: Yahoo! Finance market data downloader. `https://github.com/ranaroussi/yfinance`, 2024.

João Bento, Pedro Saleiro, André F. Cruz, Mário A. T. Figueiredo, and Pedro Bizarro. TimeSHAP: Explaining recurrent models through sequence perturbations. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

Lennart Brocki and Neo Christopher Chung. Fidelity of interpretability methods and perturbation artifacts in neural networks. *arXiv preprint arXiv:2203.02928*, 2022.

Thea Brüsch, Kristoffer K. Wickstrøm, Mikkel N. Schmidt, Robert Jenssen, and Tommy S. Alstrøm. FLEXtime: Filterbank learning to explain time series. In *World Conference on eXplainable Artificial Intelligence*, 2025.

Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. MSGNet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In *AAAI Conference on Artificial Intelligence*, 2024.

Chi-Sheng Chen, Xinyu Zhang, En-Jui Kuo, Guan-Ying Chen, Qiuzhe Xie, and Fan Zhang. FreqLens: Interpretable frequency attribution for time series forecasting. *arXiv preprint arXiv:2602.08768*, 2026.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In *International Conference on Machine Learning*, 2025.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023.

Hyunseung Chung, Sumin Jo, Yeonsu Kwon, and Edward Choi. Time is not enough: Time-frequency based explanation for time-series black-box models. In *ACM International Conference on Information and Knowledge Management*, 2024.

Jonathan Crabbé and Mihaela van der Schaar. Explaining time series predictions with dynamic masks. In *International Conference on Machine Learning*, 2021.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

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

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

Joseph Enguehard. Learning perturbations to explain time series predictions. In *International Conference on Machine Learning*, 2023.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Aya Abdelsalam Ismail, Mohamed Gunady, Héctor Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. In *Advances in Neural Information Processing Systems*, 2020.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient Transformer. In *International Conference on Learning Representations*, 2020.

Seunghan Lee, Taeyoung Park, and Kibok Lee. Learning to embed time series patches independently. In *International Conference on Learning Representations*, 2024.

Seunghan Lee, Jaehoon Lee, Jun Seo, Junhyeok Kang, Sangjun Han, Sungdong Yoo, Minjae Kim, Tae Yoon Lim, Dongwan Kang, Hwanil Choi, Soonyoung Lee, and Wonbin Ahn. Beyond magnitude and shape: A direction-aware loss for time series forecasting. *arXiv preprint arXiv:2608.01857*, 2026a.

Seunghan Lee, Jaehoon Lee, Jun Seo, Tae Yoon Lim, Dongwan Kang, Hwanil Choi, Minjae Kim, Sungdong Yoo, Junhyeok Kang, Sangjun Han, et al. EXAONE Finance 1.0: An attention-free time series foundation model for financial time series. *arXiv preprint arXiv:2609.04239*, 2026b.

Seunghan Lee, Jun Seo, Jaehoon Lee, Sungdong Yoo, Minjae Kim, Tae Yoon Lim, Dongwan Kang, Hwanil Choi, Soonyoung Lee, and Wonbin Ahn. FinSTaR: Towards financial reasoning with time series reasoning models. In *Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2026c.

Kin Kwan Leung, Clayton Rooke, Jonathan Smith, Saba Zuberi, and Maksims Volkovs. Temporal dependencies in feature importance for time series prediction. In *International Conference on Learning Representations*, 2023.

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. SegRNN: Segment recurrent neural network for long-term time series forecasting. *arXiv preprint arXiv:2308.11200*, 2023.

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L F Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164, 2017.

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

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022b.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary Transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, 2022c.

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

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

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *International Conference on Machine Learning*, 2025.

Zichuan Liu, Tianchun Wang, Jimeng Shi, Xu Zheng, Zhuomin Chen, Lei Song, Wenqian Dong, Jayantha Obeysekera, Farhad Shirani, and Dongsheng Luo. TimeX++: Learning time-series explanations with information bottleneck. In *International Conference on Machine Learning*, 2024c.

Zichuan Liu, Yingying Zhang, Tianchun Wang, Zefan Wang, Dongsheng Luo, Mengnan Du, Min Wu, Yi Wang, Chunlin Chen, Lunting Fan, and Qingsong Wen. Explaining time series via contrastive and locally sparse perturbations. In *International Conference on Learning Representations*, 2024d.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

Vahid Naghashi, Mounir Boukadoum, and Abdoulaye Banire Diallo. A multiscale model for multivariate time series forecasting. *Scientific Reports*, 15(1):1565, 2025.

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

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference*, 2018.

Owen Queen, Thomas Hartvigsen, Teddy Koker, Huan He, Theodoros Tsiligkaridis, and Marinka Zitnik. Encoding time-series explanations through self-supervised model behavior consistency. In *Advances in Neural Information Processing Systems*, 2023.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference*, pp. 606–610, 2007.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

Sana Tonekaboni, Shalmali Joshi, Kieran Campbell, David Duvenaud, and Anna Goldenberg. What went wrong and when? Instance-wise feature importance for time-series black-box models. In *Advances in Neural Information Processing Systems*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*, 2023.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024a.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024b.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. TimeXer: Empowering Transformers for time series forecasting with exogenous variables. In *Advances in Neural Information Processing Systems*, 2024c.

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

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

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Wei Jie Yeo, Wihan van der Heever, Rui Mao, Erik Cambria, Ranjan Satapathy, and Gianmarco Mengaldo. A comprehensive review on financial explainable AI. *Artificial Intelligence Review*, 2025.

Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Defu Lian, Ning An, Longbing Cao, and Zhendong Niu. Frequency-domain MLPs are more effective learners in time series forecasting. In *Advances in Neural Information Processing Systems*, 2023.

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

Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented MLP structures. *arXiv preprint arXiv:2207.01186*, 2022.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 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 *AAAI Conference on Artificial Intelligence*, 2021.

Tian Zhou, Ziqing Ma, Xue Wang, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, and Rong Jin. FiLM: Frequency improved Legendre memory model for long-term time series forecasting. In *Advances in Neural Information Processing Systems*, 2022a.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting. In *International Conference on Machine Learning*, 2022b.

A Experimental Details

Swept configurations. All experiments are univariate except the channel-axis comparison of Section 5, and we sweep $L \in \{48, 96, 192, 252, 336, 512\}$, $H \in \{24, 48, 96, 192, 336, 720\}$, and depth in $\{1, 2, 3, 4, 6\}$. Five benchmarks hold hundreds of series each, which are Solar, the four PEMS sets, and the financial set. For these we sample series at even intervals rather than running all of them, since several hundred sensors on one road network carry nearly the same information.

Reporting. Note that every configuration is run with three random seeds and that we report the seed standard deviation separately from the across-configuration standard deviation, since pooling them would make the estimate look more stable than it is. Furthermore, we report an architecture only when it has at least 10 completed runs, so that a backbone with a handful of runs does not sit beside one with thousands under the same visual weight. We store metrics and figures only rather than model weights or predicted series.

Pretrained forecasters. The checkpoints are named in full in Table 21, since each family releases several sizes. TimesFM requires the lookback to be a multiple of its input patch length, so we keep the whole pretrained grid at $L \in \{96, 192, 320, 512\}$ and $H \in \{24, 48, 96, 192\}$ to compare the three models under identical conditions. GIFT-Eval is used at every dataset and sampling frequency whose test split admits a window at the smallest of those settings, and the configurations this excludes are the ones whose series are shorter than $L + H$.

B Datasets

General domain. The suite is ETTh1, ETTh2, ETTm1 and ETTm2 (Zhou et al., 2021), Electricity, Traffic, Weather, Exchange and Solar-Energy (Wu et al., 2021), and PEMS03, PEMS04, PEMS07 and PEMS08 (Liu et al., 2022a), which is the set the forecasting literature reports on. We take every one of them from the Time-Series-Library (Wang et al., 2024b) so that the split and the preprocessing are the ones the field already uses.

Financial domain. We assemble this benchmark ourselves, since the general suite is dominated by strongly periodic series and we want a domain where periodicity is nearly absent. We retrieve daily closing prices from 2018 to 2025 with `yfinance` (Aroussi, 2024), and keep the 100 largest United States companies by market capitalization whose history is complete over that window, so that no series is padded. Note that this selection excludes firms delisted inside the window, which makes the benchmark a survivor set. It changes which series a forecaster is trained on but not the question we ask of it, since the protocol compares two explanations of the same trained model on the same windows. Trading days are the grid and we do not interpolate onto calendar days, since a non-trading day is an absence of observation rather than a missing value. The same 100 companies enter in two forms, which are the log price and the log return, and Appendix I reports them apart, since the two differ in predictability.

The 100 companies. AAPL, ABBV, ABT, ACN, ADBE, ADI, ADP, AMD, AMGN, AMT, AMZN, ANET, AON, APH, AVGO, AXP, BAC, BKNG, BLK, BMY, BRK-B, BSX, C, CAT, CB, CI, CL, CME, COST, CRM, CSCO, CVX, DE, DHR, DIS, DUK, ELV, EMR, EOG, ETN, GE, GILD, GOOG, GOOGL, GS, HD, HON, IBM, ICE, INTC, INTU, ISRG, ITW, JNJ, JPM, KLAC, KO, LIN, LLY, LMT, LOW, MA, MCD, MCK, MDT, META, MO, MRK, MS, MSFT, MU, NEO, NFLX, NKE, NOC, NOW, NVDA, ORCL, PANW, PEP, PFE, PG, PGR, PLD, PM, PYPL, QCOM, REGN, RTX, SBUX, SCHW, SHW, SO, SPGI, SYK, T, TJX, TMO, TSLA, TXN.

C BACKBONES

We explain 26 backbones, grouped by family. Within each family they appear in the order every table of this appendix uses, and we state the one idea each rests on.

Basic architectures.

- **Linear.** A single map from the lookback window to the forecast.
- **CNN.** A stack of temporal convolutions over the lookback window.
- **Transformer** (Vaswani et al., 2017). Vanilla self-attention over time steps.

Linear family.

- **DLinear** (Zeng et al., 2023). Splits trend from seasonality and fits a linear map to each.
- **TSMixer** (Chen et al., 2023). Alternates mixing over time and over channels.
- **LightTS** (Zhang et al., 2022). Samples the series at two rates before mixing.
- **FiLM** (Zhou et al., 2022a). Projects the lookback onto Legendre polynomials.
- **FreTS** (Yi et al., 2023). Mixes in the frequency domain rather than in time.
- **TiDE** (Das et al., 2023). Encodes the lookback with a dense residual network.

Convolution and recurrence.

- **SCINet** (Liu et al., 2022a). Downsamples the series into branches that interact.
- **MICN** (Wang et al., 2023). Combines local convolution with a global view.
- **TimesNet** (Wu et al., 2023). Folds the series into two dimensions by period and applies a vision backbone.
- **SegRNN** (Lin et al., 2023). Runs a recurrent unit over segments rather than points.
- **TimeMixer** (Wang et al., 2024a). Mixes across several sampling scales.
- **MSGNet** (Cai et al., 2024). Learns a graph over those scales.

Transformer family.

- **Informer** (Zhou et al., 2021). Sparsifies attention by a query score.
- **Autoformer** (Wu et al., 2021). Replaces attention with autocorrelation.
- **FEDformer** (Zhou et al., 2022b). Attends in the frequency domain.
- **Pyraformer** (Liu et al., 2022b). Attends over a pyramid of resolutions.
- **Reformer** (Kitaev et al., 2020). Hashes queries into buckets.
- **Non-stationary Transformer** (Liu et al., 2022c). Normalizes the input and returns the statistics to the attention.
- **Crossformer** (Zhang & Yan, 2023). Attends across time and across channels in turn.
- **PatchTST** (Nie et al., 2023). Tokenizes contiguous patches and keeps channels independent.
- **iTransformer** (Liu et al., 2024a). Treats each channel as a token.
- **TimeXer** (Wang et al., 2024c). Adds exogenous variables to the patch tokens.
- **MultiPatchFormer** (Naghashi et al., 2025). Reads several patch sizes at once.

D SEED REPRODUCIBILITY

Table 11 folds the three random seeds within each configuration before aggregating, which separates two quantities that a pooled average would confuse. The seed standard deviation is the spread of the same configuration across seeds, and the configuration standard deviation is the spread across configurations. We fold before aggregating on purpose, since configurations differ in how many runs they have and a pooled average would weight them unequally. The seed spread is smaller than the configuration spread for every quantity, so the variation we report across architectures and benchmarks is a property of those settings rather than of the random draw.

Table 11: **Seed reproducibility.** Over the 345 configurations for which all three quantities completed on all three seeds, where each configuration is folded before aggregating.

Quantity	**Mean**	**Std (seed)**	**Std (configuration)**
g_{own}	$\mathbf{+0.179}$	0.1625	0.4138
g_{margin}	$\mathbf{+0.237}$	0.1227	0.2404
Effective rank of E	1.94	0.1326	1.0749

E Estimator Details

E.1 Filling the Matrix with Batched Backward Calls

Automatic differentiation propagates from a single scalar, so the naive way to fill E is to select $\hat{y}[h]$, run a backward pass, keep the resulting row, and repeat for every h. That costs H backward passes per window, which is 720 passes at our longest horizon.

We instead replicate the input k times and assign a different forecast step to each replica. Writing x^i for the i-th replica and h_i for the step assigned to it, we differentiate the single scalar $\sum_i \hat{y}^i[h_i]$. The rows do not mix, because each replica is a separate leaf of the computation graph and the terms belonging to the other replicas do not depend on x^i,

$$\frac{\partial}{\partial x^i} \sum_j \hat{y}^j[h_j] \;=\; \frac{\partial \hat{y}^i[h_i]}{\partial x^i}. \tag{8}$$

One backward call therefore returns k distinct rows rather than their sum, as shown in Figure 7. The replication is what makes this work, since differentiating the same sum through one shared input would collapse the terms into $\sum_h \partial \hat{y}[h]/\partial x$, which is the single vector of prior work and not a row of E. We use $k = 16$, so a horizon of 720 needs 45 backward calls instead of 720. The cost is paid in memory rather than time, as the batch grows by a factor of k, and k sets that trade-off.

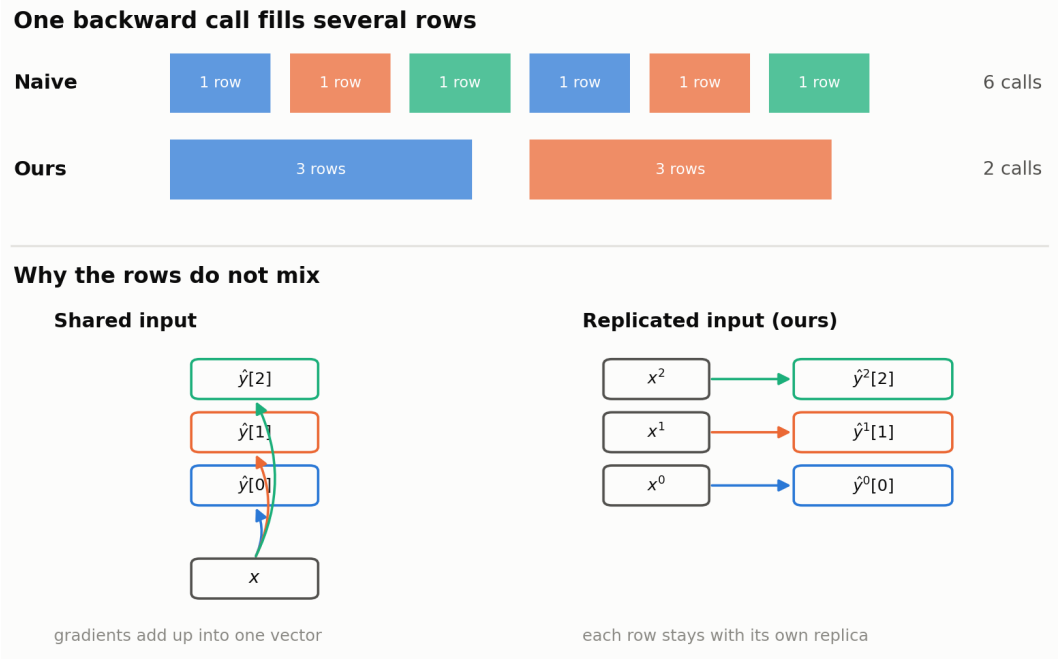


Figure 7: **Filling the explanation matrix with batched backward calls.** [Top] Backward calls for $H = 6$, where the naive way spends one per row and ours spends one per group of $k = 3$. [Bottom] Why grouping is safe. A shared input adds the gradients together and returns the shared vector, whereas one replica per step returns every row separately.

E.2 Implementation Details of the Estimator

Reading an input gradient out of a TS backbone meets two difficulties, and one fails *loudly* while the other fails *quietly*. The first is that many backbones normalize their input in place, as in `x /= stdev`, which autograd refuses to differentiate because the overwritten tensor is the one it saved for the backward pass. We therefore rewrite these operations out of place with a

`TorchFunctionMode`, and assert that the forward output is identical before and after, since code that relies on the side effect of an in-place call would otherwise become a no-op without any sign. The second is that a network can genuinely ignore its input, as we found one backbone whose decoder outputs a constant at initialization, which makes both the gradient and the deletion effect exactly zero. Note that such a zero is a fact about the untrained model rather than a finding about forecasting, so we validate the estimator on trained models only.

E.3 Porting the Estimators to Forecasting

Every estimator of Section 4.3 is reimplemented from the definition in its own paper rather than taken from a library, since this puts all of them on the same forward interface, the same perturbation operator, and the same protocol. The quantity each one is asked to preserve is the only thing we change, and it becomes $\hat{y}[h]$ in place of the scalar the original method reads.

Gradient methods. Saliency, gradient $\times$ input, and integrated gradients differentiate $\hat{y}[h]$ instead of a class score, and integrated gradients uses the mean of the lookback window as its baseline. Occlusion corrupts one window of the lookback at a time and reads the change at step h alone, rather than the change in a class score.

Mask methods. Dynamask, ExtremalMask, and ContraLSP were published for time series, but their objective reduces the output to a scalar. We keep their objective and let it preserve $\hat{y}[h]$ under the same budget on the mask, so the axis makes each of them solve one optimization problem per forecast step. Note that the same code with the axis removed preserves the whole forecast at once, which is the form closest to the original papers and the one scored in Appendix K.3.

Rows we compute. In the estimator comparison of Section 4.3, a row is estimated at six evenly spaced forecast steps rather than at all H of them, since the mask methods would otherwise need H separate optimizations per window. The protocol of Section 3.3 scores exactly the steps that were estimated, on 32 windows with seven deletion fractions and one permutation, which is lighter than the eight steps, nine fractions and three permutations of the main runs. Every estimator shares these settings, so the comparison between them is fair, while its values are not directly comparable to Figure 5. Note that the shared vector of the input gradient row averages all H rows, whereas the other estimators have only the six computed rows to average. This is why Saliency and Input gradient differ in g_{own} although they induce the same ordering at a step, while their g_{margin}, from which the shared vector cancels, nearly agrees (Table 6).

F METRIC DETAILS

F.1 DEFINITION OF THE DELETION CURVE

Section 3.3 writes $\mathrm{AUC}(h; v)$ for the area under a deletion curve and leaves the curve itself implicit, so we give it here. Let $v \in \mathbb{R}^L$ be an ordering vector over the lookback window, let $\rho = k/L \in [0, 1]$ be the deleted fraction, and let $S_\rho(v)$ be the k positions that v ranks highest. Writing $x \setminus S_\rho(v)$ for the input with those positions perturbed, the curve and its area are

$$\varepsilon(h, \rho; v) \;=\; \big\| f\big(x \setminus S_\rho(v)\big)[h] - \hat{y}[h] \big\|^2, \qquad \mathrm{AUC}(h; v) \;=\; \int_0^1 \varepsilon(h, \rho; v)\, d\rho. \tag{9}$$

Every quantity in Section 3.3 is this one function read at a different ordering vector, namely $v = E[h]$ for the row that belongs to the step, $v = \bar{E}$ for the single vector a prior method supplies, and $v = E[\pi(h)]$ for the row of another step under the shuffle control. Note that the error is taken against the model's own unperturbed forecast $\hat{y}$ rather than against the true future, so the curve reads faithfulness to the model and never forecast accuracy, and $\varepsilon(h, 0; v) = 0$ holds for every v. Perturbation resamples the selected positions from elsewhere in the same window instead of setting them to zero, which keeps the perturbed input in distribution (Hooker et al., 2019). We evaluate the integral by the trapezoidal rule on nine fractions evenly spaced in $[0, 1]$, except in the estimator comparison, which uses seven (Appendix E.3). The normalizer $\mathrm{Var}(\hat{y})$ of Section 3.3 is one scalar per run, the variance of the forecast pooled over the explained windows and all forecast steps, so it rescales a run without reweighting its steps. Note that the resampling is drawn afresh for each curve except in the sweeps of Appendix M and Appendix N, which means a single run carries Monte Carlo noise and our conclusions rest on medians over many runs.

F.2 REPORTING THE DELETION CURVE

A deletion curve can be swept in two orders, which are most-relevant-first (MoRF) and least-relevant-first (LeRF). Every quantity of Section 3.3 uses the MoRF area alone, which is $\mathrm{AUC}(h; v)$ of Eq. 9, and no LeRF curve enters it. The two orders appear together only in Table 19, which scores each estimator without the axis by the relative gap $(\mathrm{MoRF} - \mathrm{LeRF})/\mathrm{MoRF}$ over the whole forecast. We use the gap there because both curves end at the same point when every position has been deleted, so a MoRF area alone would compare methods partly by where the curve ends. Our own quantities do not need it, since they compare two orderings of the same step and the shared endpoint cancels in the difference.

F.3 DEFINITIONS OF THE BORROWED METRICS

Every metric in Section 4.2 was defined for classification, where the model outputs one label, so we restate each definition for forecasting below. We reuse the notation of Appendix F.1 and write $\varepsilon(h, \rho; v)$ for the deletion error and $\tilde{\varepsilon}(h, \rho; v) = \|f\big(x \mid S_\rho(v)\big)[h] - \hat{y}[h]\|^2$ for the retention error, in which only $S_\rho(v)$ is left intact and the rest of the window is perturbed. Averaging over the swept grid R of fractions gives

$$\begin{aligned}
\text{Comprehensiveness}(h) &= \tfrac{1}{|R|} \textstyle\sum_{\rho \in R} \varepsilon(h, \rho; v) && \text{higher is better} \\
\text{Sufficiency}(h) &= \tfrac{1}{|R|} \textstyle\sum_{\rho \in R} \tilde{\varepsilon}(h, \rho; v) && \text{lower is better} \\
\text{AOPC}(h) &= \textstyle\int \varepsilon(h, \rho; v)\, d\rho && \text{higher is better} \\
\text{Insertion}(h) &= \tfrac{1}{|R|} \textstyle\sum_{\rho \in R} \big[1 - \tilde{\varepsilon}(h, \rho; v)/\tilde{\varepsilon}(h, 0; v)\big] && \text{higher is better}
\end{aligned}$$

Here $\tilde{\varepsilon}(h, 0; v)$ is the retention error with nothing kept, which is the fully perturbed window that Insertion starts from. Note that Comprehensiveness and AOPC read the same deletion curve as Equation 9, and they differ from our gain only in how that curve becomes a number. The evidence independent of our own construction therefore comes from Sufficiency and Insertion, which perturb in the opposite direction. Each metric is evaluated twice, once at $v = E[h]$ and once at $v = \bar{E}$, and we sign the difference so that positive always means the horizon axis helped.

G SYNTHETIC DATA AND CALIBRATION

Constructed ground truth. Every generator writes the target as $y_h = w^{(h)} \cdot x_{S(h)}$ and differs only in how the set $S(h)$ of lookback positions moves with the forecast step h, and Table 12 lists the six. The weight vector $w^{(h)}$ has unit norm and varies smoothly with h, so that the target moves like a series instead of staying constant. Every $S(h)$ contains the last lookback position, so that the forecast starts from the value the series just took. The positions and weights are drawn once and reused for every window, since a map that changed every window would leave nothing to recover. We verified that a least-squares fit explains at least 98% of the target variance on held-out windows.

Table 12: **The six synthetic generators.** Each row states how the set $S(h)$ read by step h is placed. Only `sparsenull` makes every row a multiple of one vector, which is the negative control.

Generator	Placement of $S(h)$	Role
`sparsenull`	Fixed, with $w^{(h)} = a_h w$ for one fixed w	Negative control
`sparsesplit`	One set for $h < H/2$ and a disjoint one after	Two regimes
`sparseband`	One set per block of forecast steps	B regimes
`sparseshift`	Slid left in proportion to h	Gradual drift
`lagmix`	Placed at period-aligned lags	Seasonal structure
`plantrank`	Rows of a matrix A of planted rank, $y = Ax$	Known rank

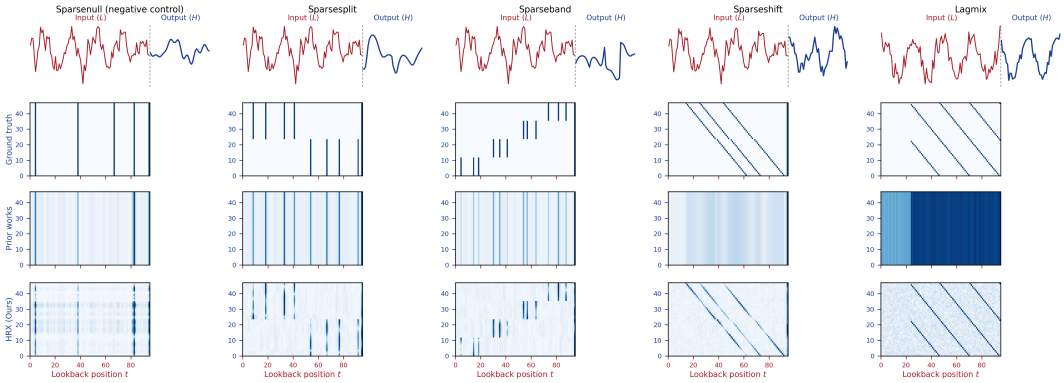


Figure 8: **What each method recovers.** The rows are one window, the answer the generator fixes, what a prior method returns, and our matrix. The last two come from the same trained backbone, and only `sparsenull` makes them agree.

Why only constructive labels. A label that only *asserts* which inputs should matter is not enough, since it can be wrong in a way the metric cannot detect. Our first attempt asserted that a sinusoidal generator depends on the points one period back. That assertion does not hold, since a sinusoid is fixed by any two points, which means a model reading two other points is equally correct. We discarded the conclusion it produced. Note that the answer belongs to the generator while the explanation belongs to the model. Agreement is therefore evidence that the measurement is sound, while disagreement can also mean that the model found an equal solution the generator did not name. Figure 8 shows what each method recovers on every generator, Table 13 repeats the body measurement for every backbone, and Table 14 splits the same measurement by backbone and generator.

The negative control. `sparsenull` sets $w^{(h)} = a_h w$ for one fixed w, so every forecast step truly depends on the same lookback positions and only the magnitude changes with h. The horizon axis has nothing to resolve there, and the effective rank of the answer is one, which is exactly the assumption prior works make. It is included because the protocol of Section 3.3 favors a concentrated explanation, so a positive gain could in principle come from the instrument rather than from the data. As shown in Table 14, the gap on `sparsenull` is not positive on any backbone, whereas it is positive on the other five generators. The gain reported in Section 4.1 therefore belongs to the horizon structure the generator planted rather than to the way it is measured.

Table 13: **Explanation performance on the synthetic dataset, all backbones.** The body reports the three minimal architectures. This is the same measurement for every backbone.

Backbone	Proposed metrics[1]			Ground-truth metrics[2]							
				AUPRC		AUROC		AUP		AUR	
	g_{own}	g_{shuffled}	g_{margin}	Vector	Matrix	Vector	Matrix	Vector	Matrix	Vector	Matrix
Linear	0.142	−0.215	0.357	0.586	0.983	0.961	0.998	0.132	0.154	0.960	0.985
CNN	0.161	−0.234	0.395	0.578	0.993	0.962	0.999	0.132	0.154	0.963	0.984
Transformer	0.121	−0.252	0.373	0.577	0.993	0.960	0.999	0.130	0.154	0.963	0.985
DLinear	0.153	−0.197	0.350	0.586	0.994	0.968	1.000	0.133	0.156	0.963	0.987
TSMixer	0.140	−0.210	0.350	0.585	1.000	0.967	1.000	0.133	0.156	0.963	0.987
LightTS	0.156	−0.210	0.366	0.573	0.991	0.963	1.000	0.133	0.156	0.961	0.987
FiLM	0.073	−0.194	0.267	0.569	0.934	0.958	0.979	0.133	0.151	0.955	0.973
FreTS	0.110	−0.149	0.259	0.583	1.000	0.967	1.000	0.133	0.156	0.963	0.987
TiDE	0.128	−0.265	0.393	0.566	0.933	0.960	0.973	0.131	0.150	**0.961**	0.961
SCINet	0.166	−0.197	0.364	0.579	0.981	0.960	0.998	0.133	0.154	0.961	0.983
MICN	0.058	−0.211	0.269	0.586	0.994	0.967	1.000	0.131	0.156	0.963	0.987
TimesNet	0.121	−0.234	0.355	0.576	0.966	0.960	0.993	0.136	0.154	0.963	0.982
SegRNN	0.060	−0.163	0.224	0.388	0.829	0.918	0.934	0.114	0.139	**0.917**	0.903
TimeMixer	0.126	−0.243	0.369	0.566	0.981	0.962	0.998	0.135	0.153	0.963	0.980
MSGNet	0.136	−0.187	0.323	0.571	0.993	0.960	1.000	0.133	0.156	0.961	0.986
Informer	0.081	−0.137	0.218	0.536	0.877	0.845	0.959	0.128	0.148	0.842	0.939
Autoformer	0.065	−0.102	0.168	**0.273**	0.184	**0.586**	0.571	**0.068**	0.068	**0.589**	0.576
FEDformer	0.037	−0.103	0.141	0.375	0.393	0.723	0.758	0.098	0.098	0.724	0.745
Pyraformer	0.061	−0.045	0.106	0.471	0.833	0.824	0.903	0.121	0.135	0.825	0.838
Reformer	0.021	−0.117	0.139	**0.369**	0.320	0.685	0.760	0.088	0.093	0.684	0.757
Nonstat. Trans.	0.021	−0.111	0.133	0.484	0.882	0.917	0.959	0.100	0.123	0.918	0.944
Crossformer	0.152	−0.163	0.315	0.582	1.000	0.956	1.000	0.133	0.156	0.952	0.986
PatchTST	0.149	−0.271	0.420	0.577	0.994	0.967	1.000	0.133	0.155	0.963	0.984
TimeXer	0.136	−0.246	0.382	0.579	1.000	0.967	1.000	0.133	0.156	0.963	0.984
MultiPatchFormer	0.147	−0.179	0.326	0.571	1.000	0.955	1.000	0.148	0.158	0.950	0.984
iTransformer	0.142	−0.197	0.339	0.579	0.980	0.960	0.998	0.133	0.156	0.962	0.984
Mean	0.110	−0.186	0.296	0.533	0.886	0.915	0.953	0.125	0.144	0.913	0.938

[1] A prior explanation cannot be measured at a single forecast step, so $E[h]$ and $\bar{E}$ are the same vector there and every g is exactly zero.

Table 14: **Explanation performance on the synthetic dataset, per backbone and generator.** Every cell is the median gap in AUPRC between our matrix and the vector prior works return, over 1,386 runs. Note that `sparsenull` is the negative control, so its gap should not be positive.

Backbone	lagmix	plantrank	sparseband	sparsenull	sparseshift	sparsesplit
Linear	+0.813	+0.556	+0.396	−0.089	+0.426	+0.262
CNN	+0.817	+0.552	+0.408	−0.125	+0.437	+0.276
Transformer	+0.818	+0.549	+0.403	−0.110	+0.435	+0.278
DLinear	+0.814	+0.556	+0.397	−0.062	+0.434	+0.275
TSMixer	+0.819	+0.543	+0.405	−0.109	+0.436	+0.281
LightTS	+0.818	+0.557	+0.397	−0.097	+0.435	+0.280
FiLM	+0.819	+0.189	+0.374	−0.150	+0.404	+0.271
FreTS	+0.813	+0.551	+0.403	−0.013	+0.435	+0.278
TiDE	+0.803	+0.318	+0.393	−0.141	+0.384	+0.207
SCINet	+0.812	+0.359	+0.402	−0.099	+0.422	+0.278
MICN	+0.820	+0.554	+0.397	−0.152	+0.440	+0.276
TimesNet	+0.811	+0.356	+0.399	−0.101	+0.413	+0.255
SegRNN	+0.812	+0.521	+0.525	−0.060	+0.437	+0.374
TimeMixer	+0.819	+0.311	+0.398	−0.141	+0.431	+0.246
MSGNet	+0.803	+0.373	+0.395	−0.084	+0.436	+0.274
Informer	+0.258	+0.542	+0.371	−0.137	+0.128	+0.233
Autoformer	−0.006	+0.004	−0.000	−0.084	−0.080	−0.097
FEDformer	+0.425	+0.044	−0.037	−0.165	+0.034	−0.061
Pyraformer	+0.810	+0.480	+0.286	−0.131	+0.310	+0.227
Reformer	+0.125	+0.511	+0.004	−0.216	−0.042	−0.074
Nonstat. Trans.	+0.416	+0.264	+0.361	−0.103	+0.380	+0.218
Crossformer	+0.819	+0.553	+0.409	−0.028	+0.446	+0.279
PatchTST	+0.816	+0.330	+0.407	−0.071	+0.439	+0.281
TimeXer	+0.814	+0.360	+0.403	−0.075	+0.439	+0.287
MultiPatchFormer	+0.809	+0.426	+0.408	−0.048	+0.430	+0.286
iTransformer	+0.816	+0.368	+0.400	−0.088	+0.420	+0.285
Mean	+0.704	+0.413	+0.350	−0.103	+0.358	+0.230

Instrument calibration. To ask whether the effective rank we report is a property of the matrix rather than of the estimator that fills it, we calibrate the measure against a known answer. We draw the target from a matrix A of planted effective rank with the `plantrank` generator of Table 12, where the true Jacobian is A itself and the answer is fixed before any measurement. The planted rank is swept from 1 to 16 over 28 backbones, which are the 26 of the main results and two further ones that we exclude elsewhere, and every trained model is then read with our own estimator. As shown in Figure 9, the measured rank rises with the planted one on every backbone, exactly on some of them and at a lower slope on the rest. The measure therefore reads the rank the matrix carries rather than one the estimator imposes, which is what Appendix M relies on when it reads the rank as a predictor of the gain.

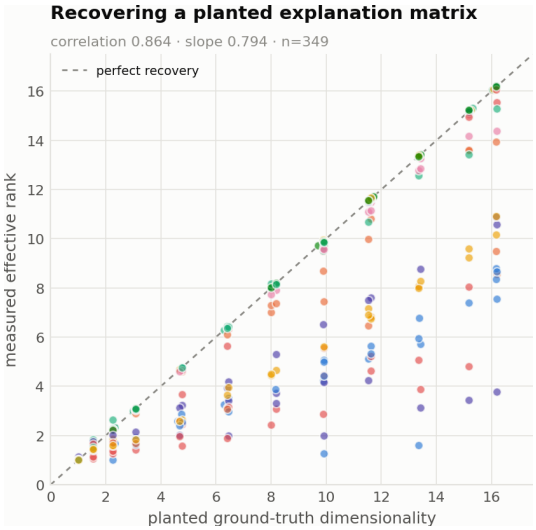


Figure 9: **Recovering a planted rank.** The measured effective rank against the planted one over 349 runs, where color marks the backbone. The measure tracks the plant on every backbone.

Checking the matrix against a known answer. As shown in Figure 10, we place a synthetic generator whose causal windows we control beside a real benchmark, and the recovered matrix reproduces the staircase on the synthetic side, which is a check real data cannot provide, since its causal windows are unknown.

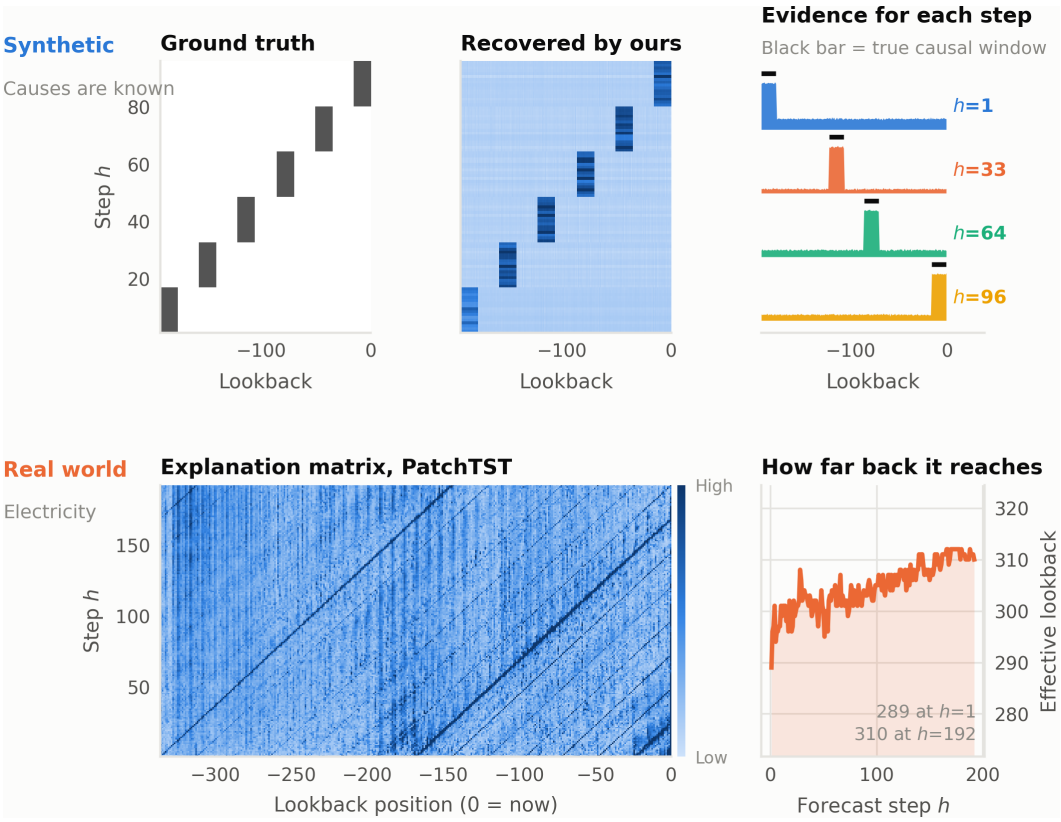


Figure 10: **HRX against a known answer.** [Top] On a constructed generator, the recovered matrix reproduces the ground-truth staircase, where black bars mark the true causal windows. [Bottom] The same procedure on Electricity with PatchTST.

H Results across Architectures and Benchmarks

Table 15 and Table 16 are the numbers behind the two panels of Figure 5, and they report all three quantities of Section 3.3 side by side, where g_{shuffled} is recovered from $g_{\text{margin}} = g_{\text{own}} - g_{\text{shuffled}}$. Note that the ideal direction holds on every benchmark except PEMS04, which does not reach significance, and that three backbones sit at or below zero. Pyraformer is the one whose interval falls entirely below zero, while FEDformer and SegRNN have intervals that contain it, and none of the three carries a gain of the size the other backbones report. Note also that a backbone is listed only when it has at least 10 completed runs, so that a handful does not sit beside thousands.

Table 15: **The three quantities of the protocol across architectures.** Aggregated over the 15 benchmarks and all swept settings. Every value is a median, and a check mark means the bootstrap 95% interval of g_{own} excludes zero.

Backbone	g_{own}	95% CI	g_{shuffled}	g_{margin}	n	Significant
Linear	**+0.199**	[+0.192, +0.206]	**−0.024**	**+0.223**	3,513	✓
CNN	**+0.134**	[+0.127, +0.140]	**−0.084**	**+0.218**	3,513	✓
Transformer	**+0.054**	[+0.050, +0.061]	**−0.071**	**+0.125**	3,667	✓
DLinear	**+0.172**	[+0.164, +0.181]	**−0.036**	**+0.208**	3,583	✓
TSMixer	**+0.169**	[+0.157, +0.179]	**−0.045**	**+0.214**	662	✓
LightTS	**+0.229**	[+0.213, +0.283]	**−0.109**	**+0.338**	284	✓
FiLM	**+0.256**	[+0.221, +0.280]	**−0.099**	**+0.355**	284	✓
FreTS	**+0.101**	[+0.086, +0.128]	**−0.054**	**+0.155**	284	✓
TiDE	**+0.264**	[+0.246, +0.283]	**−0.133**	**+0.397**	578	✓
SCINet	**+0.238**	[+0.216, +0.271]	**−0.153**	**+0.391**	578	✓
MICN	**+0.051**	[+0.031, +0.071]	**−0.037**	**+0.088**	284	✓
TimesNet	**+0.224**	[+0.197, +0.244]	**−0.074**	**+0.298**	654	✓
SegRNN	**+0.004**	[−0.019, +0.012]	**−0.076**	**+0.080**	284	–
TimeMixer	**+0.115**	[+0.102, +0.130]	**−0.064**	**+0.179**	284	✓
MSGNet	**+0.071**	[+0.052, +0.088]	**−0.034**	**+0.105**	282	✓
Informer	**+0.018**	[+0.011, +0.035]	+0.006	**+0.012**	272	✓
Autoformer	**+0.098**	[+0.070, +0.130]	**−0.011**	**+0.109**	284	✓
FEDformer	−0.000	[−0.018, +0.015]	**−0.120**	**+0.120**	283	–
Pyraformer	−0.003	[−0.011, −0.000]	**−0.048**	**+0.045**	284	–
Reformer	**+0.079**	[+0.056, +0.102]	**−0.137**	**+0.216**	284	✓
Nonstat. Trans.	**+0.057**	[+0.041, +0.078]	**−0.027**	**+0.084**	284	✓
Crossformer	**+0.055**	[+0.040, +0.073]	**−0.061**	**+0.116**	284	✓
PatchTST	**+0.204**	[+0.197, +0.212]	**−0.030**	**+0.234**	3,602	✓
ITransformer	**+0.279**	[+0.257, +0.294]	**−0.029**	**+0.308**	662	✓
TimeXer	**+0.183**	[+0.157, +0.223]	**−0.044**	**+0.227**	284	✓
MultiPatchFormer	**+0.178**	[+0.144, +0.204]	**−0.014**	**+0.192**	260	✓

Table 16: **The three quantities of the protocol across benchmarks.** Aggregated over the 26 backbones in the univariate setting. Every value is a median, and a check mark means the bootstrap 95% interval of g_{own} excludes zero. Note that the ranking is the normalized one, and that the raw gain would put the two highest-variance benchmarks on top for unrelated reasons.

Benchmark	g_{own}	95% CI	g_{shuffled}	g_{margin}	n	Significant
Finance_ret	**+0.340**	[+0.325, +0.356]	**−0.018**	**+0.358**	1,999	✓
Electricity	**+0.313**	[+0.305, +0.320]	**−0.079**	**+0.392**	2,681	✓
ETTh2	**+0.280**	[+0.273, +0.286]	**−0.010**	**+0.290**	2,278	✓
ETTh1	**+0.195**	[+0.191, +0.199]	**−0.047**	**+0.242**	2,736	✓
ETTm2	**+0.143**	[+0.126, +0.158]	**−0.018**	**+0.161**	2,274	✓
Traffic	**+0.138**	[+0.128, +0.151]	**−0.050**	**+0.188**	820	✓
Solar	**+0.112**	[+0.102, +0.125]	**−0.092**	**+0.204**	681	✓
ETTm1	**+0.079**	[+0.073, +0.084]	**−0.005**	**+0.084**	2,326	✓
Finance	**+0.061**	[+0.048, +0.072]	**−0.062**	**+0.123**	1,994	✓
Weather	**+0.056**	[+0.053, +0.059]	**−0.126**	**+0.182**	2,730	✓
PEMS03	**+0.055**	[+0.019, +0.100]	**−0.098**	**+0.153**	682	✓
Exchange	**+0.045**	[+0.039, +0.050]	**−0.076**	**+0.121**	2,278	✓
PEMS07	**+0.037**	[+0.004, +0.078]	**−0.074**	**+0.111**	679	✓
PEMS04	**+0.007**	[−0.004, +0.014]	**−0.061**	**+0.068**	679	–
PEMS08	**+0.005**	[+0.003, +0.008]	**−0.016**	**+0.021**	680	✓

I Results by Domain

The two naive baselines. The baselines are predicting the standardized target as zero and extending the last observation, and a run counts as skilled only when its test error beats both. Note that to our knowledge no prior TS explanation benchmark reports a forecasting baseline beside its faithfulness score, which is why Section 5 reports the split rather than a single number.

What actually differs between the domains. The explanation matrices are of similar complexity in both, with an effective rank of 1.84 in the general domain and 1.85 in the financial one. What changes is which backbone exploits the axis rather than the shape of the explanation, as the vanilla Transformer moves from one of the weakest in the general domain to one of the strongest in the financial one. Which backbone exploits the axis is therefore a property of the data as much as of the architecture, and our ranking should not be read as a general one.

J QUALITATIVE PANELS IN FULL

Figure 6 shows five panels in one row for space. Figure 11 gives the full domain comparison with the naive-baseline verdict under each panel, and Figure 12 gives the three backbones together with the effective lookback of every row.

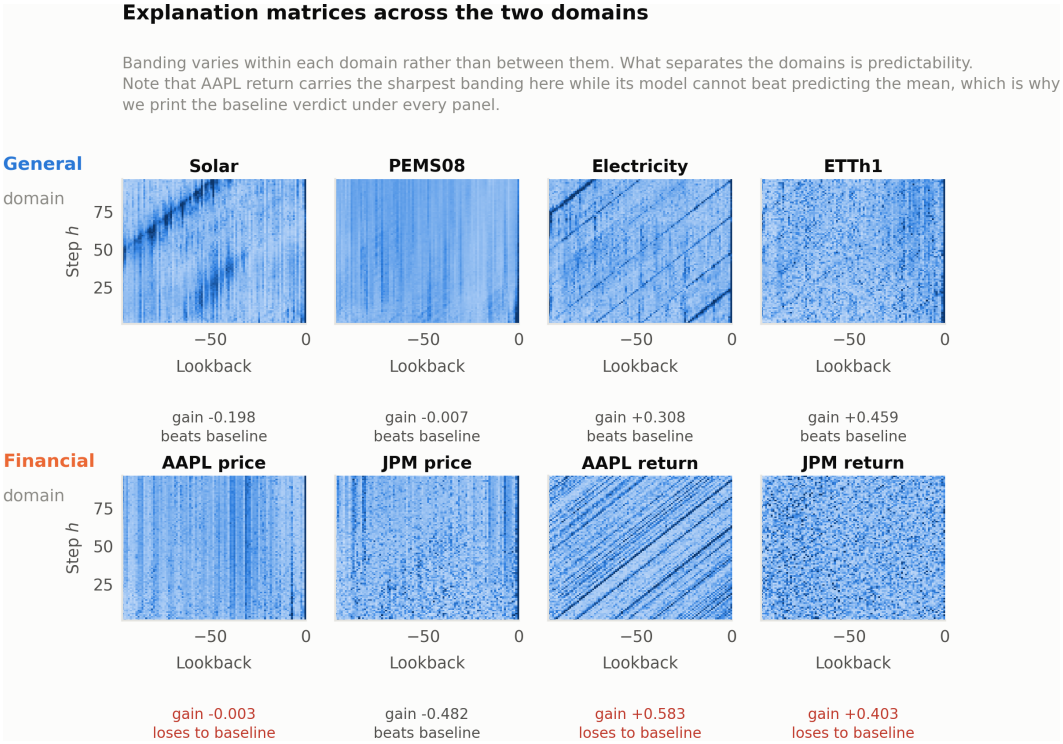


Figure 11: **Explanation matrices across the two domains.** The naive-baseline verdict is under each panel. Banding varies within each domain rather than between them, and the sharpest banding here comes from a model that cannot forecast.

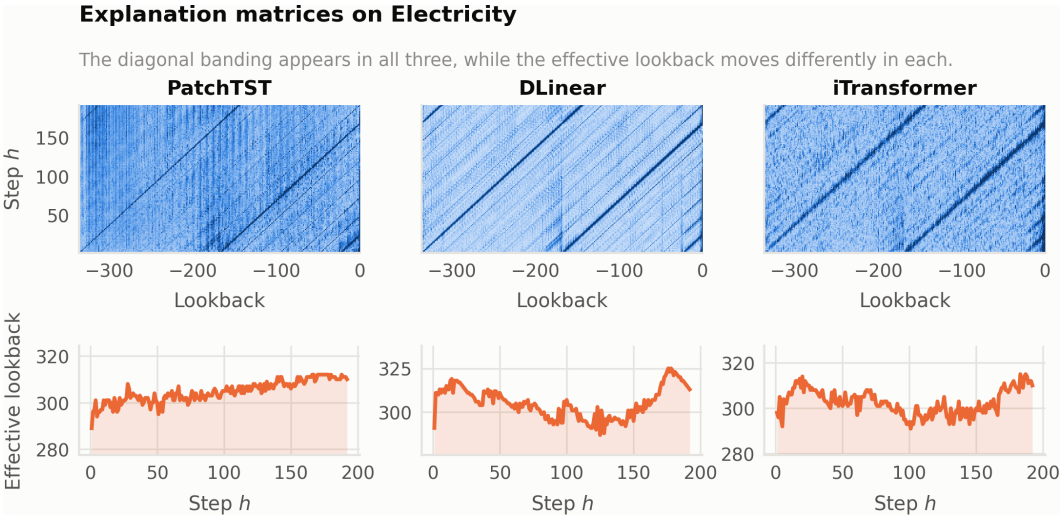


Figure 12: **Explanation matrices for three architectures on Electricity.** Underneath each is the effective lookback of every row, which is how many recent positions a row needs to reach 90% of its total. The banding appears in all three, while the effective lookback does not.

K Further Analyses

K.1 Swept Axes

To ask whether the horizon axis behaves the same at every setting, we contrast the endpoints of the three swept settings, holding backbone, dataset, seed, and the other two settings fixed so that the comparison is within-block. The pairing is necessary because the set of backbones that completed at the longest horizon differs from the set that completed at the shortest. As shown in Table 17, the gain rises with both the forecast length and the lookback length, while depth moves in the opposite direction by an order of magnitude less. We attribute this to the shape of the problem rather than to capacity, since a longer forecast spans more of the seasonal structure and a longer lookback offers more evidence to choose among, while adding layers offers neither. Note that only the endpoint contrasts are significant, so we read the pattern as *a trend rather than a law*.

Table 17: **Paired change in the gain along the three swept settings.** Each contrast holds backbone, dataset, seed and the other two axes fixed. A plain average per level would be misleading, since the backbones that completed at $H = 720$ differ from those at $H = 24$.

Setting	Range	Paired change	95% CI	Blocks
Forecast length H	$24 \to 720$	$\mathbf{+0.260}$	$[+0.118, +0.402]$	100
Lookback length L	$48 \to 512$	$\mathbf{+0.408}$	$[+0.017, +0.800]$	20
Depth (layers)	$1 \to 6$	-0.020	$[-0.039, -0.001]$	208

Figure 13 shows the same three contrasts as Table 17 at every intermediate level, which makes clear that the increase is not monotone step by step.

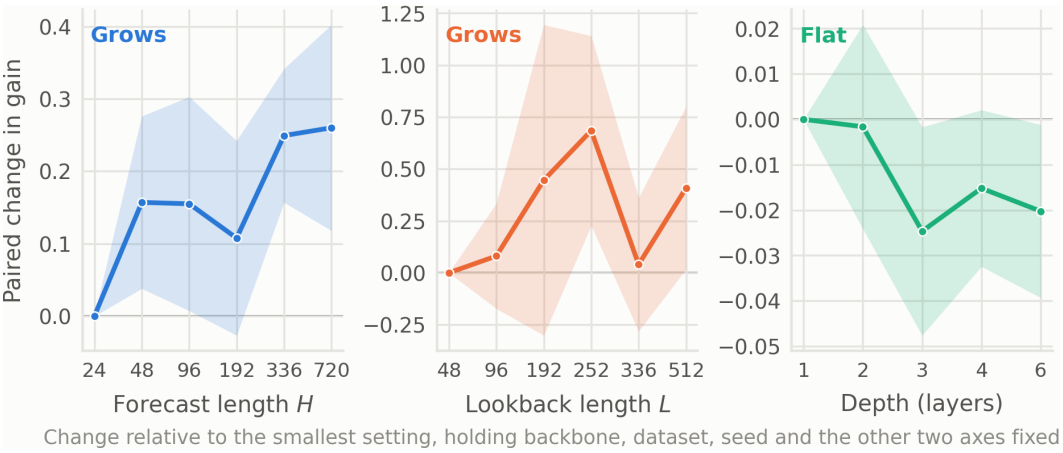


Figure 13: **Paired change in the gain along the three swept settings.** Each point is the average within-block difference against the smallest setting.

K.2 Cost of Each Estimator

Table 6 in the body reports what the horizon axis adds to each estimator and says nothing about what that costs. As shown in Table 18, the mask methods are two orders of magnitude more expensive than a plain gradient even in their published form, which solves one optimization for the whole forecast, and the axis multiplies that cost by the number of steps estimated. Every entry except Input gradient is measured in that published form. The Input gradient entry is the six rows of Appendix E.3 computed one backward pass at a time, which is roughly six times Saliency, and the batched implementation of Appendix E.1 reduces the cost of a full row set.

Table 18: **Cost of each estimator.** Wall-clock time relative to one forward pass of the same backbone, measured on the same hardware for every method.

Estimator	Cost
Saliency	3.1×
Gradient × input	3.1×
Integrated gradients	98×
Occlusion	14×
Dynamask	592×
ExtremalMask	643×
ContraLSP	675×
Input gradient	18×

K.3 The Estimators as Plain Attribution Methods

Table 6 in the body asks what the horizon axis adds to each estimator, which is a question about the axis rather than about the estimators themselves. To ask the complementary question, we score the same estimators without the axis, in the marginal form each one was published in, under the standard deletion gap between the most-relevant-first and least-relevant-first orders. As shown in Table 19, our default input gradient is competitive but not the best. This is the position we take throughout, since the contribution is the axis and the estimator that fills it is a free choice.

Table 19: **The same estimators without the horizon axis.** Each is scored in its published marginal form under the standard deletion gap, so the ordering here says nothing about the horizon axis.

Estimator	Relative deletion gap	95% CI	n
TSR	**+0.747**	[+0.745, +0.749]	15,898
Integrated gradients	**+0.694**	[+0.691, +0.697]	15,898
Dynamask	**+0.674**	[+0.672, +0.677]	15,898
Gradient × input	**+0.634**	[+0.628, +0.640]	15,898
Input gradient	**+0.630**	[+0.623, +0.635]	15,668
ContraLSP	**+0.588**	[+0.582, +0.594]	15,888
Saliency	**+0.576**	[+0.569, +0.586]	15,898
ExtremalMask	**+0.549**	[+0.542, +0.556]	15,889
Occlusion	**+0.458**	[+0.453, +0.464]	15,898

L LIMITATIONS IN DETAIL

We cannot predict the effect size from the series alone. The rank criterion of Section 3.4 predicts the gain (Table 10), but it is read from E and therefore needs the trained forecaster. A practitioner cannot tell before training how much the axis will buy on a new series. As statistics of the raw series, we checked spectral concentration, autocorrelation, forecast-target correlation across steps, and naive-forecast difficulty, and no candidate reached $|r| = 0.07$ on real data. An early analysis suggested that the number of stacked periods drives the gain, but that generator scaled the target variance with the number of periods. The correlation is -0.169 once the scale is fixed, so we report the failure rather than the artifact.

We found no downstream utility. None of the uses we tried turned the matrix into a better forecast or a better diagnosis. Neither explanation-guided input selection nor horizon-specific pruning gave a gain distinguishable from zero, which means HRX is a *measurement tool* and nothing more.

Two properties of our estimator. Our default estimator inherits the known weaknesses of input gradients, and our evidence on real data is faithfulness-based, which is why we anchor our method on constructed labels first. The effective rank reads $|E|$ and therefore measures how many components the magnitude map is built from, not how many the signed Jacobian is. Structure in the signs alone is invisible to it, which is why the calibration of Appendix G plants the rank in the magnitudes.

M THE RANK OF THE EXPLANATION MATRIX

What the effective rank counts. Eq. 6 is the participation ratio of the singular values of E, which counts how many directions carry the magnitude of the matrix rather than how many are nonzero. A matrix whose rows are positive multiples of one vector has a participation ratio of one, and every quantity of Table 2 is then pinned to zero, because the deletion order of Section 3.3 depends on a row only through the order it induces. We compute the ratio on $|E|$ for the same reason the protocol does, since the order a row induces does not see the sign. Structure in the signs alone is therefore invisible to the ratio, which is why the generators of Appendix G plant it in the magnitudes.

Why this measure and not the entropy-based one. Both Eq. 6 and the entropy-based effective rank (Roy & Vetterli, 2007) are exponentials of a Rényi entropy of a normalized singular value distribution, and they differ in two places. Theirs takes $p_k = \sigma_k / \|\sigma\|_1$ and the Shannon entropy, that is Rényi order one, giving $\exp(-\sum_k p_k \log p_k)$. Ours takes $p_k = \sigma_k^2 / \sum_j \sigma_j^2$ and Rényi order two, giving $1/\sum_k p_k^2$, which is Eq. 6 and the quantity reported as a dimension in prior work (Litwin-Kumar et al., 2017). We use order two because σ_k^2 is the magnitude a direction carries, so the criterion says how much of E a direction accounts for rather than how probable it is. Order two is also less sensitive to the long tail of small singular values, and Jensen's inequality makes it the smaller of the two, so the criterion never overstates how many directions are in use.

The truncation sweep. Table 9 truncates E to its leading directions and scores every truncation under the identical protocol, and the rank-one row is the analytic check just described. Note that the deletion in this sweep is paired, which means every truncation is scored against the same corruption pattern rather than a fresh draw, since an unpaired comparison adds the Monte Carlo spread of the corruption to the difference. Without the pairing the rank-one row misses its analytic value of zero by the scale of the spread. Only the distance sweep of Appendix N uses the same pairing.

Raw and normalized quantities. Every quantity in the paper is divided by the forecast variance $\mathrm{Var}(\hat{y})$. That variance is positive on every run, so the normalization cannot change the sign of a quantity and the fraction of runs in the ideal direction is identical with or without it. It does change how the benchmarks rank against each other, since a benchmark whose forecasts vary more carries a larger raw gain at the same normalized gain. We report the normalized reading throughout, since the forecast scale should not decide whether the horizon axis carries information. Table 16 orders the benchmarks under the normalized reading, and its caption names the two that the raw reading would move to the top.

N Distance Between Forecast Steps

Why a distance is needed. The protocol of Section 3.3 draws a permutation π uniformly over the horizon, which validates that a row belongs to its own step but does not say how far apart two steps have to be before their rows stop agreeing. A matrix whose rows vary smoothly answers that question differently at the two ends, since a uniform permutation lands far from a step far more often than it lands next to it. We therefore replace π with a fixed distance Δ and score step h with the row $E[h + \Delta]$, falling back to $E[h - \Delta]$ at the ends of the horizon and averaging the two where both exist. Table 20 sweeps Δ and puts the uniform permutation of the protocol on the same axis.

Table 20: **Margin g_{margin} by distance.** A step is scored with the row Δ away from it, where the protocol of Section 3.3 draws that distance at random and reads the lower rows of this table.

Distance	g_{margin}	Positive
$\Delta = 1$	$\mathbf{+0.118}$	**82%**
$\Delta = 2$	$\mathbf{+0.156}$	**84%**
$\Delta = 4$	$\mathbf{+0.214}$	**89%**
$\Delta = 8$	$\mathbf{+0.277}$	**89%**
$\Delta = 16$	$\mathbf{+0.349}$	**91%**
$\Delta = 32$	$\mathbf{+0.412}$	**93%**
Random π	$\mathbf{+0.293}$	**94%**

What the sweep says. The margin is positive at the smallest distance, which means two neighboring steps already read different lookback positions rather than sharing one row between them. It then grows with the distance and does not saturate inside the horizon, which is the reading a smoothly varying matrix predicts and which the truncation sweep of Appendix M predicts as well. The uniform permutation sits near the large-distance end of the sweep rather than at its average, which means the headline margin of Section 4.2 is the value for distant steps and not the value for neighboring ones. Note that this does not weaken the finding, since every distance carries a positive margin, and it says which part of the horizon axis the headline value reports.

O PRETRAINED FORECASTERS IN DETAIL

Application to zero-shot TSFMs. To validate that the horizon axis is not an artifact of training, we repeat the measurement on 8 time-series foundation models (TSFMs) run zero-shot. As shown in Table 21, we run every checkpoint on GIFT-Eval (Aksu et al., 2024), a benchmark held apart from the pretraining corpora, and the gain is positive for almost every model. The horizon axis therefore holds *even when the model has never seen the series*. The last two rows put the two settings side by side, where the sign is the same as for trained backbones and only the size differs. Note that every run is checked against the official API of its library.

Table 21: **Gain g_{own} for zero-shot TSFMs.** Zero-shot performance on GIFT-Eval.

Model	g_{own}	Positive
Chronos-2	**+0.070**	**66%**
Chronos-Bolt	**+0.027**	**59%**
TimesFM-2.5	**+0.047**	**66%**
Moirai-2.0	**+0.017**	**58%**
Timer	**+0.007**	**52%**
TimeMoE	**+0.015**	**58%**
TTM-r2	**+0.080**	**60%**
VisionTS	**+0.001**	**51%**
All zero-shot TSFMs	**+0.023**	**59%**
Trained backbones	**+0.143**	**80%**

Which models we could verify. Every run checks its own output against the official API of its library, which is necessary because the libraries wrap the inference path in a no-gradient block and some overwrite tensors in place. We therefore call the forward pass directly and replace each in-place step with an out-of-place one. The largest relative difference over 16,729 runs is 2e-6. The same check excludes four further models whose released inference path either leaves the forecasting head untrained or draws samples that carry no input gradient.

The two models that do not reach significance. Note that they are informative rather than contrary. Timer beats the naive baseline in the smallest fraction of runs of any model here, and Section 5 shows that runs which lose to that baseline read a smaller gain within the same series, although the gain remains positive. VisionTS was pretrained on images and has never seen a time series. Note that the naive baseline matters more here than anywhere else, since a zero-shot model has no guarantee of skill and runs that beat it read a larger gain.

Every checkpoint in full. Table 22 names the checkpoint of every model, since each family releases several sizes, and adds the naive-baseline verdict and the run count behind each row.

Table 22: **Zero-shot TSFMs in full.** The body table reports the gain and the positive fraction, and this one adds the checkpoint, the naive-baseline verdict and the run count.

Model	Checkpoint	Beats naive	g_{own}	Positive	n
Chronos-2	amazon/chronos-2	72%	**+0.070**	**66%**	2,285
Chronos-Bolt	amazon/chronos-bolt-base	68%	**+0.027**	**59%**	2,249
TimesFM-2.5	google/timesfm-2.5-200m-pytorch	72%	**+0.047**	**66%**	2,290
Moirai-2.0	Salesforce/moirai-2.0-R-small	72%	**+0.017**	**58%**	2,292
Timer	thuml/timer-base-84m	32%	**+0.007**	**52%**	2,239
TimeMoE	Maple728/TimeMoE-200M	71%	**+0.015**	**58%**	1,896
TTM-r2	ibm-granite/granite-timeseries-ttm-r2	73%	**+0.080**	**60%**	1,219
VisionTS	VisionTS (mae_base)	55%	**+0.001**	**51%**	2,249
All zero-shot TSFMs		64%	+0.023	59%	16,719
Trained backbones		72%	**+0.143**	**80%**	25,506

The sweeps and the borrowed metrics. Table 23 repeats the pretrained result under the four metrics we did not design, and all six agree in sign over 16,729 runs. Table 24 sweeps the horizon and the lookback, where the gain rises with the forecast length as it does for trained backbones, while g_{margin} is flat and stays near +0.118 at every setting. Table 25 splits GIFT-Eval by the field its series come from, and Table 26 reports every dataset so that no favorable subset is doing the work. Of 66 datasets, 32 are significantly positive and 15 are significantly negative.

Table 23: **Zero-shot TSFMs under metrics of prior works.** All are signed so that positive means the horizon axis helps, and we judge by median and sign test, since the values are heavy-tailed.

Metric	Median	Positive	Sign test	n
Comprehensiveness	**+0.0011**	**59%**	$< 10^{-12}$	16,719
Sufficiency	**+0.0010**	**62%**	$< 10^{-12}$	16,720
AOPC	**+0.0008**	**58%**	$< 10^{-12}$	16,719
Insertion	**+0.0122**	**58%**	$< 10^{-12}$	16,720
g_{own}	**+0.0227**	**59%**	$< 10^{-12}$	16,719
g_{margin}	**+0.1184**	**82%**	$< 10^{-12}$	16,717

Table 24: **Swept axes for zero-shot TSFMs.** The gain grows with the forecast length, whereas the margin is stable across every setting.

Setting	g_{own}	g_{margin}	Positive	n
Horizon H				
24	−0.016	**+0.101**	44%	2,982
48	**+0.001**	**+0.124**	**50%**	2,732
96	**+0.035**	**+0.117**	**64%**	9,288
192	**+0.057**	**+0.140**	**69%**	1,717
Lookback L				
96	**+0.036**	**+0.116**	**63%**	2,412
192	**+0.034**	**+0.119**	**64%**	2,453
320	**+0.034**	**+0.117**	**62%**	2,454
512	**+0.014**	**+0.119**	**56%**	9,400

Table 25: **Zero-shot TSFMs by field.** The second column counts the distinct GIFT-Eval datasets in each field, and the rest read as in Table 21.

Domain	Datasets	g_{own}	Positive	n
Nature	3	**+0.222**	**87%**	325
Transport	7	**+0.067**	**70%**	2,089
Energy	13	**+0.049**	**70%**	3,818
Cloud	6	**+0.022**	**54%**	1,924
Web	2	**+0.017**	**64%**	554
Healthcare	4	**+0.011**	**52%**	338
Climate	7	−0.006	46%	1,977
Econ	6	−0.011	46%	1,785
Sales	3	−0.063	43%	686

Table 26: **Zero-shot TSFMs by dataset.** Sorted by gain, where a name carries the GIFT-Eval sampling frequency as a suffix when the benchmark provides several.

Dataset	Gain	Pos.	n	Dataset	Gain	Pos.	n
Saugeenday_M	**+0.279**	**95%**	100	PEMS08	**+0.008**	**52%**	336
Saugeenday_D	**+0.209**	**82%**	111	Jena_weather	**+0.004**	**54%**	334
Traffic	**+0.192**	**94%**	111	Jena_weather_10T	**+0.003**	**55%**	332
Saugeenday_W	**+0.187**	**86%**	114	ETTm1	**+0.003**	**50%**	111
LOOP_SEATTLE_H	**+0.174**	**93%**	333	LOOP_SEATTLE_5T	**+0.001**	**51%**	336
Bitbrains_rnd_5T	**+0.160**	**75%**	331	ETTh1	**+0.001**	**50%**	111
Solar_H	**+0.143**	**92%**	336	Ett1_D	**+0.001**	**50%**	299
Electricity_H	**+0.142**	**86%**	335	PEMS07	−0.010	46%	331
Electricity	**+0.128**	**89%**	111	M_DENSE_D	−0.011	48%	300
Bitbrains_fast_storage_5T	**+0.126**	**65%**	333	Jena_weather_H	−0.011	43%	334
SZ_TAXI_H	**+0.124**	**70%**	303	Solar_D	−0.013	49%	150
LOOP_SEATTLE_D	**+0.118**	**78%**	152	Finance_ret	−0.017	47%	334
M_DENSE_H	**+0.091**	**86%**	328	Finance	−0.019	40%	331
Electricity_15T	**+0.086**	**75%**	333	Bitbrains_fast_storage_H	−0.022	48%	301
Ett1_H	**+0.084**	**92%**	331	M4_weekly	−0.027	39%	332
Us_births_M	**+0.080**	**68%**	50	Ett2_D	−0.028	43%	297
ETTh2	**+0.071**	**92%**	111	Hierarchical_sales_W	−0.034	43%	151
M4_hourly	**+0.064**	**72%**	336	M4_daily	−0.037	35%	333
SZ_TAXI_15T	**+0.060**	**69%**	337	Kdd_cup_2018_with_missing_H	−0.042	39%	333
Ett2_H	**+0.058**	**68%**	336	Restaurant	−0.046	44%	204
Ett1_15T	**+0.046**	**83%**	334	M4_quarterly	−0.065	29%	298
Ett2_15T	**+0.044**	**80%**	338	Us_births_W	−0.080	41%	114
PEMS04	**+0.040**	**68%**	334	PEMS03	−0.085	22%	334
Us_births_D	**+0.039**	**63%**	111	Exchange	−0.098	33%	113
ETTm2	**+0.036**	**80%**	110	Bitbrains_rnd_H	−0.107	47%	298
Solar	**+0.028**	**78%**	333	Kdd_cup_2018_with_missing_D	−0.136	33%	190
Solar_10T	**+0.028**	**78%**	334	Hierarchical_sales_D	−0.146	42%	331
Weather	**+0.022**	**71%**	112	Jena_weather_D	−0.165	27%	150
Bizitobs_l2c_H	**+0.020**	**58%**	330	Covid_deaths	−0.172	41%	63
Bizitobs_application	**+0.018**	**62%**	221	Bizitobs_l2c_5T	−0.185	30%	331
Bizitobs_service	**+0.017**	**65%**	333	M4_yearly	−0.206	42%	152
M4_monthly	**+0.015**	**56%**	334	Electricity_W	−0.215	29%	63
Temperature_rain_with_missing	**+0.013**	**54%**	304	Electricity_D	−0.267	33%	332

P Related Works

P.1 Lines This Work Sits Beside

Section 2 names the three lines this work sits beside, and this section gives each of them in full.

TS forecasting models. Architectures for long-horizon TS forecasting have settled into a few families. Transformer variants reshape attention along the sequence axis (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022b; Nie et al., 2023; Liu et al., 2024a), linear and MLP designs argue that most of the accuracy follows from trend and seasonality structure rather than from attention (Zeng et al., 2023; Das et al., 2023; Wang et al., 2024a), and convolutional designs fold the series along its periods so that a 2D kernel can read them (Wu et al., 2023). Our framework treats any of them as a black box it attaches to, and details of the 26 backbones we explain are deferred to Appendix C.

Gradient-based methods. These methods read importance from the *derivative of the output with respect to the input*, so a single backward pass scores every input value. Saliency (Simonyan et al., 2013) uses the magnitude of the gradient and gradient $\times$ input multiplies it by the input value at each position, while integrated gradients (Sundararajan et al., 2017) integrates it along a path from a baseline and SmoothGrad (Smilkov et al., 2017) averages it over noisy copies of the input. Note that none of them is optimized for a given input, which is what keeps this family cheap. These methods transfer directly to TS, where naive transfer is shown to conflate the time and feature axes and a two-stage rescaling is proposed (Ismail et al., 2020).

Perturbation-based methods. These methods *change the input* and read importance from *how much the output moves*. Dynamask (Crabbé & van der Schaar, 2021) learns a mask over the input and keeps it sparse, while ExtremalMask (Enguehard, 2023) learns the perturbation as well as the mask and ContraLSP (Liu et al., 2024d) perturbs contrastively so that the result stays in distribution. TimeX (Queen et al., 2023) and TimeX++ (Liu et al., 2024c) instead train a separate model to imitate the forecaster and read the explanation from it. Shapley variants (Lundberg & Lee, 2017; Bento et al., 2021) and window-based methods (Tonekaboni et al., 2020; Leung et al., 2023) replace subsets of the input and divide the resulting change among them. Note that all of them solve an optimization problem for every input, which is what makes them expensive.

P.2 Lines Adjacent to This Work

These three lines are adjacent to our setting rather than in it, so we place them here.

Time series foundation models (TSFMs). A separate line of work pretrains one forecaster on a large corpus and applies it to unseen series without any further training (Goswami et al., 2024; Liu et al., 2025; Lee et al., 2026b). The widely used families are Chronos (Ansari et al., 2024), TimesFM (Das et al., 2024), Moirai (Woo et al., 2024), Timer (Liu et al., 2024b), TimeMoE (Shi et al., 2025), and TinyTimeMixer (Ekambaram et al., 2024). VisionTS (Chen et al., 2025) sits apart from all of them, since it forecasts with a masked autoencoder pretrained on images rather than on any time series. GIFT-Eval (Aksu et al., 2024) collects the benchmarks these models are compared on and keeps them apart from the pretraining corpus. None of this line reports what a released model reads for a given forecast step, and the libraries that ship these models place their inference path behind a no-gradient block, so the question has not been asked of them.

Interpretable-by-design forecasters. N-BEATS (Oreshkin et al., 2020) decomposes a forecast into trend and seasonality stacks, the Temporal Fusion Transformer (Lim et al., 2021) exposes variable-selection weights and interpretable attention, and DLinear (Zeng et al., 2023) makes its $H \times L$ weight matrix directly readable. The DLinear weight matrix is in fact horizon-resolved, but it is a property of one linear architecture rather than a general tool, and it describes the learned map instead of the evidence used on a given input.

Frequency-based explanation. A smaller line of work explains along the frequency axis instead of the time axis, reporting which spectral bands a prediction relies on. SpectralX (Chung et al., 2024) perturbs features in the time-frequency plane and inverts the transform so that the original model can score the result, FLEXtime (Brüsch et al., 2025) optimizes a mask over the frequency bands of a filterbank, and FreqLens (Chen et al., 2026) learns the frequency bases from data for forecasting.

We leave all three out of Table 1, since a spectral explanation resolves neither the time axis nor the forecast step and therefore answers a different question from the one the table compares. Note that the horizon axis is independent of this choice, since a spectral explanation could be resolved per forecast step in the same way.

Evaluation of explanations. Deletion curves (Samek et al., 2017) are the standard faithfulness probe, and several works document their failure modes, including perturbation artifacts (Brocki & Chung, 2022), retraining confounds (Hooker et al., 2019), and explanations that pass visual inspection while being independent of the model (Adebayo et al., 2018). Every established metric assumes a model that outputs one label, so it scores a whole prediction and not one forecast step.

Q Full Results by Benchmark

Table 6 in the body reports one number per estimator, pooled over every backbone and every benchmark. The tables below unfold that number, with one table per benchmark, one row per backbone, and one column per estimator. Every cell is the median normalized gain g_{own} over the swept settings, since the distribution is heavy-tailed for the reason given in Section 4.2. Table 27 to Table 41 give one benchmark each, and the benchmarks themselves are described in Appendix B.

Table 27: **Gain g_{own} on ETTh1.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.186**	**+0.103**	**+0.123**	**+0.079**	**+0.209**	**+0.129**	**+0.148**	**+0.226**
CNN	**+0.132**	**+0.050**	**+0.139**	**+0.113**	**+0.210**	−0.016	**+0.016**	**+0.160**
Transformer	**+0.142**	**+0.069**	**+0.127**	**+0.053**	**+0.109**	**+0.025**	**+0.028**	**+0.181**
DLinear	**+0.180**	**+0.120**	**+0.116**	**+0.065**	**+0.187**	**+0.149**	**+0.173**	**+0.230**
TSMixer	**+0.194**	**+0.226**	**+0.153**	**+0.076**	**+1.076**	**+0.289**	**+0.177**	**+0.227**
LightTS	**+0.427**	**+0.351**	**+0.251**	**+0.086**	**+1.077**	**+0.470**	**+0.290**	**+0.544**
FiLM	**+0.219**	**+0.198**	**+0.228**	**+0.201**	**+0.264**	**+0.970**	**+0.254**	**+0.101**
FreTS	**+0.080**	**+0.052**	**+0.038**	**+0.025**	**+0.152**	**+0.003**	**+0.003**	**+0.064**
TiDE	**+0.228**	**+0.524**	**+0.231**	**+0.043**	**+0.524**	**+0.988**	**+0.099**	**+0.395**
SCINet	**+0.295**	**+0.407**	**+0.334**	**+0.107**	**+0.439**	**+0.235**	**+0.188**	**+0.514**
MICN	**+0.133**	**+0.154**	**+0.122**	**+0.041**	**+0.411**	**+0.154**	**+0.150**	**+0.205**
TimesNet	**+0.227**	**+0.200**	**+0.072**	**+0.063**	**+0.449**	**+0.080**	−0.021	**+0.325**
SegRNN	−0.116	−0.075	−0.104	−0.013	**+0.138**	**+0.351**	**+0.326**	−0.172
TimeMixer	**+0.223**	**+0.278**	**+0.068**	**+0.051**	**+0.324**	**+0.072**	**+0.070**	**+0.247**
MSGNet	**+0.093**	**+0.220**	−0.004	**+0.094**	**+0.290**	**+0.048**	**+0.056**	**+0.091**
Informer	−0.008	−0.007	−0.000	−0.000	−0.012	−0.082	−0.087	−0.026
Autoformer	**+0.152**	**+0.018**	**+0.112**	**+0.078**	**+0.009**	**+0.030**	**+0.155**	**+0.360**
FEDformer	−0.000	−0.105	−0.071	**+0.043**	**+0.303**	**+0.038**	**+0.174**	−0.058
Pyraformer	**+0.021**	−0.020	**+0.028**	−0.012	**+0.110**	−0.031	**+0.031**	**+0.005**
Reformer	−0.173	−0.044	**+0.351**	−0.182	−0.143	−0.261	−0.213	−0.071
Nonstat. Trans.	−0.172	**+0.045**	**+0.058**	−0.036	−0.289	−0.309	−0.299	**+0.054**
Crossformer	**+0.045**	**+0.071**	**+0.022**	**+0.011**	**+0.027**	−0.012	**+0.022**	**+0.057**
PatchTST	**+0.226**	**+0.155**	**+0.187**	**+0.091**	**+0.244**	−0.031	−0.027	**+0.324**
TimeXer	**+0.236**	**+0.311**	**+0.222**	**+0.076**	**+0.624**	**+0.089**	−0.031	**+0.355**
MultiPatchFormer	**+0.213**	**+0.276**	**+0.008**	**+0.032**	**+0.021**	**+0.155**	−0.014	**+0.252**
iTransformer	**+0.480**	**+0.585**	**+0.352**	**+0.150**	**+0.362**	−0.026	**+0.016**	**+0.506**

Table 28: **Gain g_{own} on ETTh2.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.268**	**+0.181**	**+0.158**	**+0.040**	**+0.429**	**+0.216**	**+0.233**	**+0.288**
CNN	**+0.115**	**+0.114**	**+0.158**	**+0.138**	**+0.053**	**+0.120**	**+0.415**	**+0.130**
Transformer	**+0.091**	**+0.071**	**+0.058**	**+0.024**	**+0.188**	**+0.107**	**+0.157**	**+0.094**
DLinear	**+0.256**	**+0.177**	**+0.155**	**+0.035**	**+0.368**	**+0.178**	**+0.219**	**+0.268**
TSMixer	**+0.290**	**+0.180**	**+0.160**	−0.027	**+0.669**	**+0.216**	**+0.317**	**+0.382**
LightTS	**+0.274**	**+0.166**	**+0.133**	**+0.035**	**+1.256**	**+0.453**	**+0.515**	**+0.428**
FiLM	**+0.192**	**+0.145**	**+0.136**	**+0.131**	**+0.634**	**+0.792**	**+0.447**	**+0.238**
FreTS	**+0.309**	**+0.201**	**+0.184**	−0.027	**+0.307**	**+0.108**	**+0.137**	**+0.277**
TiDE	**+0.205**	**+0.174**	**+0.125**	−0.014	**+0.756**	**+0.933**	**+0.526**	**+0.325**
SCINet	**+0.170**	**+0.354**	**+0.180**	**+0.039**	**+1.294**	**+2.395**	**+1.155**	**+0.337**
MICN	**+0.095**	**+0.001**	**+0.038**	−0.011	**+0.361**	**+0.145**	**+0.124**	**+0.079**
TimesNet	**+0.248**	**+0.144**	**+0.101**	−0.025	**+0.400**	**+0.343**	**+0.181**	**+0.261**
SegRNN	−0.003	−0.108	−0.014	−0.017	−0.135	**+0.043**	**+0.069**	−0.029
TimeMixer	**+0.177**	**+0.132**	**+0.113**	**+0.011**	**+0.672**	**+0.380**	**+0.309**	**+0.172**
MSGNet	**+0.068**	−0.004	−0.025	−0.051	**+0.243**	**+0.160**	**+0.132**	**+0.112**
Informer	−0.235	−0.026	**+0.365**	**+0.436**	−0.499	**+0.298**	−0.319	−0.185
Autoformer	**+0.077**	**+0.040**	**+0.093**	**+0.088**	−0.026	**+0.118**	**+0.133**	**+0.199**
FEDformer	−0.053	**+0.012**	−0.074	−0.053	**+0.263**	**+0.058**	**+0.142**	−0.003
Pyraformer	−0.010	**+0.015**	−0.002	**+0.007**	**+0.102**	−0.054	−0.002	**+0.073**
Reformer	−0.059	**+0.005**	**+0.015**	**+0.012**	−0.059	**+0.007**	**+0.158**	**+0.082**
Nonstat. Trans.	**+0.020**	**+0.005**	**+0.066**	**+0.043**	−0.037	−0.073	−0.056	**+0.080**
Crossformer	**+0.095**	**+0.069**	**+0.079**	−0.007	**+0.153**	**+0.132**	**+0.086**	**+0.100**
PatchTST	**+0.170**	**+0.081**	**+0.094**	**+0.015**	**+0.335**	**+0.273**	**+0.222**	**+0.234**
TimeXer	**+0.233**	**+0.124**	**+0.103**	−0.001	**+0.608**	**+0.349**	**+0.320**	**+0.318**
MultiPatchFormer	**+0.185**	**+0.028**	−0.037	−0.009	**+0.191**	**+0.047**	**+0.176**	**+0.247**
iTransformer	**+0.422**	**+0.210**	**+0.198**	−0.030	**+0.740**	**+0.323**	**+0.180**	**+0.453**

Table 29: **Gain g_{own} on ETTm1.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.049**	**+0.063**	**+0.041**	−0.041	**+0.129**	**+0.067**	**+0.026**	**+0.049**
CNN	**+0.025**	**+0.154**	**+0.180**	**+0.003**	−0.057	**+0.247**	−0.026	−0.022
Transformer	−0.016	−0.015	−0.004	−0.014	**+0.083**	−0.019	−0.028	−0.001
DLinear	**+0.029**	**+0.046**	**+0.026**	−0.023	**+0.121**	**+0.050**	**+0.013**	**+0.028**
TSMixer	**+0.119**	**+0.103**	**+0.081**	**+0.007**	**+0.593**	**+0.043**	−0.023	**+0.287**
LightTS	**+0.229**	**+0.186**	**+0.154**	**+0.047**	**+0.729**	**+0.351**	**+0.059**	**+0.291**
FiLM	**+0.169**	**+0.136**	**+0.212**	**+0.031**	**+0.347**	**+0.625**	−0.013	**+0.247**
FreTS	**+0.096**	**+0.070**	**+0.066**	**+0.002**	**+0.407**	−0.019	−0.027	**+0.084**
TiDE	**+0.157**	**+0.142**	**+0.139**	**+0.019**	**+0.602**	**+0.578**	−0.143	**+0.183**
SCINet	−0.040	**+0.084**	**+0.287**	**+0.021**	**+0.371**	**+0.064**	−0.048	−0.031
MICN	−0.012	−0.061	−0.030	**+0.021**	**+0.507**	**+0.057**	−0.023	−0.018
TimesNet	**+0.089**	**+0.017**	**+0.038**	**+0.025**	**+0.286**	**+0.066**	−0.116	**+0.118**
SegRNN	−0.106	−0.096	−0.054	**+0.031**	**+0.019**	−0.252	−0.259	−0.083
TimeMixer	**+0.155**	**+0.111**	**+0.090**	−0.039	**+0.461**	−0.043	−0.000	**+0.206**
MSGNet	**+0.069**	**+0.062**	**+0.027**	**+0.018**	**+0.219**	−0.063	**+0.012**	**+0.138**
Informer	**+0.020**	−0.014	−0.009	−0.017	**+0.056**	−0.075	−0.215	**+0.067**
Autoformer	**+0.139**	−0.018	**+0.181**	−0.171	**+0.008**	**+0.212**	**+0.441**	**+0.304**
FEDformer	−0.023	−0.017	**+0.006**	−0.029	−0.008	−0.035	−0.075	**+0.032**
Pyraformer	**+0.024**	**+0.027**	−0.022	−0.048	**+0.095**	−0.006	−0.005	**+0.051**
Reformer	−0.043	**+0.048**	−0.025	−0.109	−0.091	−0.076	**+0.010**	−0.016
Nonstat. Trans.	**+0.029**	**+0.110**	−0.039	−0.014	−0.065	−0.080	**+0.025**	−0.066
Crossformer	−0.006	**+0.000**	**+0.019**	**+0.022**	**+0.135**	−0.013	−0.048	**+0.005**
PatchTST	**+0.064**	**+0.063**	**+0.045**	−0.009	**+0.145**	−0.129	−0.039	**+0.045**
TimeXer	**+0.136**	**+0.111**	**+0.109**	−0.020	**+0.458**	−0.047	−0.182	**+0.167**
MultiPatchFormer	−0.007	**+0.083**	**+0.002**	−0.062	**+0.233**	−0.223	−0.044	**+0.158**
iTransformer	**+0.331**	**+0.268**	**+0.160**	**+0.007**	**+0.497**	**+0.031**	−0.064	**+0.544**

Table 30: **Gain g_{own} on ETTm2.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.182**	**+0.119**	**+0.128**	**+0.048**	**+0.558**	**+0.031**	**+0.126**	**+0.204**
CNN	**+1.036**	**+2.017**	**+2.192**	**+1.085**	**+2.109**	**+3.003**	**+1.710**	**+1.197**
Transformer	**+0.010**	−0.030	−0.008	**+0.028**	**+0.335**	**+0.008**	**+0.048**	**+0.015**
DLinear	**+0.149**	**+0.123**	**+0.118**	**+0.074**	**+0.646**	**+0.150**	**+0.210**	**+0.113**
TSMixer	**+0.189**	**+0.097**	**+0.130**	**+0.010**	**+0.622**	**+0.198**	**+0.282**	**+0.212**
LightTS	**+0.211**	**+0.120**	**+0.091**	**+0.027**	**+1.040**	**+0.526**	**+0.538**	**+0.310**
FiLM	**+0.204**	**+0.203**	**+0.288**	**+0.210**	**+1.036**	**+5.248**	**+2.084**	**+0.324**
FreTS	**+0.099**	**+0.009**	**+0.075**	**+0.015**	**+0.493**	**+0.107**	**+0.082**	**+0.071**
TiDE	**+0.241**	**+0.172**	**+0.132**	**+0.029**	**+0.777**	**+0.981**	**+0.832**	**+0.343**
SCINet	**+0.079**	**+0.109**	**+0.198**	−0.009	**+1.052**	**+2.305**	**+2.116**	**+0.153**
MICN	**+0.012**	−0.018	−0.048	**+0.039**	**+0.181**	**+0.120**	**+0.251**	−0.019
TimesNet	**+0.157**	**+0.109**	**+0.193**	**+0.039**	**+0.761**	**+0.459**	**+0.715**	−0.043
SegRNN	−0.063	−0.151	**+0.047**	**+0.009**	−0.125	−0.122	−0.154	−0.075
TimeMixer	**+0.255**	**+0.243**	**+0.092**	−0.009	**+0.884**	**+2.893**	**+2.253**	**+0.174**
MSGNet	**+0.054**	−0.008	**+0.027**	**+0.032**	**+0.577**	**+0.141**	**+0.112**	**+0.052**
Informer	**+0.264**	**+0.363**	**+0.371**	**+0.061**	**+0.296**	−0.001	−0.095	**+0.277**
Autoformer	−0.138	−0.094	**+0.043**	**+0.043**	**+0.622**	**+0.921**	**+1.077**	**+0.176**
FEDformer	−0.007	−0.185	**+0.026**	−0.085	**+0.919**	**+0.961**	**+1.085**	−0.164
Pyraformer	**+0.011**	−0.006	**+0.011**	−0.003	**+0.257**	**+0.015**	−0.016	−0.025
Reformer	**+0.141**	**+0.120**	**+0.129**	**+0.030**	**+0.281**	**+0.072**	**+0.100**	**+0.203**
Nonstat. Trans.	**+0.052**	−0.017	**+0.080**	**+0.034**	−0.031	−0.063	−0.038	**+0.047**
Crossformer	**+0.034**	**+0.025**	**+0.053**	**+0.048**	**+0.343**	**+0.054**	−0.026	**+0.013**
PatchTST	−0.024	**+0.114**	**+0.353**	**+0.166**	**+0.506**	**+0.456**	**+0.654**	−0.024
TimeXer	**+0.019**	−0.019	**+0.024**	**+0.032**	**+0.615**	**+0.582**	**+0.374**	**+0.017**
MultiPatchFormer	**+0.231**	**+0.214**	**+0.028**	**+0.177**	**+0.444**	**+0.298**	**+0.154**	**+0.040**
iTransformer	**+0.290**	**+0.156**	**+0.176**	**+0.029**	**+0.566**	**+0.292**	**+0.129**	**+0.375**

Table 31: **Gain g_{own} on PEMS03.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	+0.060	−0.416	+0.010	+0.167	+0.255	−0.312	−0.527	+0.186
CNN	−0.012	+0.507	+1.120	+0.078	+0.261	+2.605	+1.567	+0.950
Transformer	−0.029	−0.026	−0.061	−0.063	+0.094	+0.067	+0.101	−0.197
DLinear	+0.013	−0.049	+0.001	+0.163	+0.807	+1.541	+1.321	+0.245
TSMixer	+0.022	−0.039	+0.054	−0.022	+0.038	−0.041	−0.163	+0.004
LightTS	+0.010	+0.076	+0.089	+0.059	+0.474	+0.390	+0.462	+0.119
FiLM	+0.099	+0.075	+0.141	+0.179	−0.074	−0.346	−1.287	+0.372
FreTS	−0.101	−0.117	+0.014	+0.033	+0.140	+0.001	−0.086	−0.211
TiDE	+0.232	+0.056	+0.168	+0.107	+0.045	+0.818	−2.204	+0.184
SCINet	+0.135	+0.043	+0.118	+0.085	+0.561	+2.223	+1.271	−0.068
MICN	+0.070	+0.022	−0.028	+0.133	+0.462	+0.546	+0.336	+0.005
TimesNet	+0.546	+0.220	+0.421	−0.016	−0.083	+0.852	+0.528	+0.568
SegRNN	−0.138	−0.102	−0.130	−0.041	−0.038	−0.165	−0.115	−0.123
TimeMixer	+0.098	+0.036	−0.050	+0.115	−0.363	+0.427	+0.457	+0.045
MSGNet	+0.060	+0.009	−0.023	+0.121	−0.163	+0.214	+0.181	+0.057
Informer	−0.387	−0.117	+0.025	−0.026	−0.003	+0.025	−0.028	−0.330
Autoformer	+0.705	+0.959	+0.814	−0.177	+1.129	+0.913	+1.266	+0.973
FEDformer	−0.133	−0.067	−0.181	−0.257	+0.136	+0.141	+0.150	−0.093
Pyraformer	−0.373	−0.377	−0.042	−0.133	+0.047	−0.296	−0.343	−0.399
Reformer	−0.022	+0.003	−0.003	+0.024	+0.123	−0.071	+0.017	−0.034
Nonstat. Trans.	−0.035	−0.020	−0.045	−0.030	−0.042	+0.021	+0.005	−0.027
Crossformer	+0.037	+0.018	+0.045	−0.014	+0.165	−0.049	−0.005	+0.015
PatchTST	+0.256	+0.074	+0.064	+0.036	+0.332	+0.428	+0.383	+0.248
TimeXer	+0.139	+0.062	+0.084	+0.050	+0.231	+0.381	+0.432	+0.189
MultiPatchFormer	+1.735	+0.826	+1.501	+0.087	+0.438	+1.542	+3.167	+1.751
iTransformer	+0.130	−0.032	−0.048	−0.010	+0.091	+0.384	+0.402	+0.222

Table 32: **Gain g_{own} on PEMS04.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	+0.041	+0.138	+0.043	+0.154	+0.572	+0.214	+0.120	−0.025
CNN	−0.080	+0.990	+0.713	−0.008	+0.222	+1.577	+1.332	+0.179
Transformer	−0.089	−0.090	−0.091	−0.040	+0.098	+0.338	+0.155	−0.106
DLinear	+0.091	+0.114	+0.073	−0.013	+0.967	+0.149	+0.146	+0.010
TSMixer	−0.080	−0.001	+0.173	+0.093	+0.297	+0.152	+0.176	−0.119
LightTS	+0.118	+0.177	+0.224	+0.115	+0.372	+0.137	+0.226	+0.109
FiLM	+0.135	+0.346	+0.181	+0.080	+1.143	+1.428	+0.715	+0.248
FreTS	+0.183	+0.111	+0.217	+0.062	+0.351	+0.559	+0.532	+0.132
TiDE	+0.050	+0.291	+0.170	+0.136	+0.728	+1.824	+0.623	+0.241
SCINet	+0.204	+0.295	+0.064	+0.113	+0.735	+1.518	+0.619	+0.065
MICN	+0.089	+0.294	+0.314	+0.109	+0.363	+0.577	+0.463	+0.170
TimesNet	+0.306	+0.520	+0.135	+0.083	+0.119	+0.215	+0.612	+0.127
SegRNN	−0.097	+0.001	−0.143	+0.054	−0.380	+0.213	+0.248	−0.074
TimeMixer	+0.293	+0.191	+0.165	+0.111	+0.515	+0.914	+0.793	+0.369
MSGNet	+0.153	+0.134	+0.097	+0.001	+0.284	+0.134	+0.381	+0.107
Informer	+0.069	−0.022	−0.055	+0.056	−0.007	−0.021	+0.105	+0.121
Autoformer	+0.327	+0.236	+1.429	−0.834	−0.077	−1.353	+0.002	+0.534
FEDformer	+0.051	+0.167	+0.119	−0.082	+0.116	+0.166	+0.261	+0.178
Pyraformer	−0.073	−0.052	−0.001	−0.029	+0.090	−0.057	−0.045	+0.011
Reformer	+0.089	+0.096	+0.021	−0.068	+0.274	−0.005	−0.056	−0.085
Nonstat. Trans.	−0.039	−0.060	−0.160	−0.023	−0.229	+0.178	+0.002	−0.013
Crossformer	−0.014	+0.016	+0.022	+0.000	−0.185	+0.026	−0.007	−0.033
PatchTST	+0.164	+0.283	+0.020	+0.075	+0.372	+0.529	+0.567	+0.059
TimeXer	+0.179	+0.261	+0.129	−0.076	+0.693	+0.761	+0.576	+0.016
MultiPatchFormer	−0.142	+0.122	+0.107	+0.019	+0.071	+0.582	+0.861	+0.148
iTransformer	+0.423	+0.405	+0.361	+0.034	+0.360	+1.536	+1.195	+0.489

Table 33: **Gain g_{own} on PEMS07.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	+0.160	+0.087	+0.132	+0.153	+0.678	+0.095	+0.206	+0.346
CNN	+0.070	+11.854	+11.320	+11.543	+0.418	+3.187	+5.804	+0.807
Transformer	−0.123	−0.091	−0.073	+0.059	+0.101	−1.456	+0.007	−2.686
DLinear	+0.222	+0.066	+0.100	+0.112	+0.993	+0.615	+0.555	+0.306
TSMixer	+0.003	−1.898	+0.042	+0.871	+0.075	−0.013	−0.035	+0.004
LightTS	+0.226	+0.032	+0.140	+0.181	+0.415	+0.140	+0.304	+0.193
FiLM	+0.247	+0.193	+0.132	+0.120	+0.399	+1.426	+0.448	+0.514
FreTS	−0.001	−2.332	+0.016	+0.105	−0.061	−1.895	−3.479	+0.042
TiDE	+0.276	+0.157	+0.115	+0.077	+0.555	+1.525	+0.111	+0.436
SCINet	−6.268	−0.011	−0.074	+0.197	+0.258	+3.094	+5.148	−0.370
MICN	+0.126	+0.059	+0.051	+0.086	+0.746	+0.292	+0.188	+0.153
TimesNet	+0.142	+20.767	+19.165	+0.904	−0.137	+0.161	+0.477	+0.272
SegRNN	−0.283	−0.501	+0.287	+2.340	−0.411	−0.647	−2.218	−0.259
TimeMixer	−0.018	+8.778	+7.924	+2.655	+0.012	−0.076	+0.473	−0.088
MSGNet	+0.131	+0.211	+3.180	+2.578	−2.978	+0.039	+0.078	+7.876
Informer	+0.137	−0.080	−0.153	−0.002	+0.058	+0.057	−0.490	+0.425
Autoformer	+0.161	+0.451	+0.151	+0.023	+0.131	+1.035	+1.269	−0.113
FEDformer	+0.189	−0.052	−0.847	−0.316	−0.052	+0.034	+0.322	+0.538
Pyraformer	−0.068	+0.017	−0.040	+0.051	−0.077	+0.033	+0.066	+0.005
Reformer	+0.083	−0.049	+0.096	−0.007	+0.264	+0.085	−0.273	−0.060
Nonstat. Trans.	+0.411	−0.021	−0.026	+0.169	+0.079	−0.021	+0.044	+0.098
Crossformer	−1.015	−1.484	+0.002	+0.083	+0.082	−0.017	−0.067	−0.204
PatchTST	+6.784	+34.104	+46.251	+0.656	+0.024	+0.144	+0.090	+3.257
TimeXer	+0.027	+60.511	+80.018	+30.634	+0.083	−2.510	+0.186	+65.936
MultiPatchFormer	+69.260	+70.343	+96.816	+51.424	−25.601	+49.034	+120.490	+73.253
iTransformer	+0.264	+0.318	+0.384	+0.081	+0.008	+0.105	+0.001	+0.339

Table 34: **Gain g_{own} on PEMS08.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	+0.129	+0.082	+0.066	+0.163	+0.610	+0.235	+0.277	+0.056
CNN	+0.165	+0.215	+0.280	+0.068	+0.258	+0.494	+0.322	+0.198
Transformer	+0.020	−0.006	−0.003	+0.021	+0.144	+0.028	−0.002	+0.032
DLinear	+0.030	+0.047	+0.031	+0.009	+0.652	+0.194	+0.307	+0.049
TSMixer	+0.038	−0.010	+0.010	+0.041	+0.034	+0.016	+0.015	+0.058
LightTS	+0.128	+0.037	+0.108	+0.110	+0.520	+0.349	+0.393	+0.131
FiLM	+0.158	+0.116	+0.134	+0.122	+0.471	+1.146	+0.522	+0.147
FreTS	+0.027	+0.014	+0.044	+0.012	+0.095	+0.014	+0.009	+0.024
TiDE	+0.207	+0.143	+0.154	+0.071	+0.765	+2.387	+1.017	+0.175
SCINet	−0.040	+0.009	−0.010	+0.141	+0.184	+1.475	+1.056	−0.102
MICN	+0.006	+0.020	−0.002	+0.032	−0.006	+0.063	+0.197	−0.016
TimesNet	+0.296	+0.138	+0.147	+0.046	−0.001	+0.248	+0.197	+0.094
SegRNN	+0.017	+0.002	+0.006	+0.012	−0.042	−0.004	−0.031	+0.079
TimeMixer	+0.094	+0.017	−0.015	+0.038	+0.076	+0.150	+0.115	+0.030
MSGNet	−0.000	−0.005	−0.008	+0.061	−0.013	+0.059	+0.090	+0.039
Informer	+0.010	−0.016	−0.006	+0.014	+0.028	−0.050	−0.053	+0.011
Autoformer	+0.175	+0.402	+0.066	−0.010	+0.072	+0.328	+0.738	+0.288
FEDformer	−0.006	+0.002	+0.021	+0.028	−0.013	−0.003	+0.137	+0.002
Pyraformer	−0.006	−0.019	−0.015	−0.008	−0.016	−0.023	−0.013	−0.008
Reformer	+0.137	+0.004	+0.091	−0.004	+0.065	+0.020	−0.034	−0.017
Nonstat. Trans.	−0.007	−0.003	−0.032	+0.028	−0.054	−0.004	+0.042	+0.011
Crossformer	+0.030	−0.005	+0.007	+0.031	+0.008	+0.004	−0.001	+0.028
PatchTST	+0.015	−0.011	+0.029	+0.011	−0.004	−0.013	−0.013	−0.003
TimeXer	+0.026	+0.006	+0.008	+0.044	+0.007	−0.014	+0.121	+0.013
MultiPatchFormer	+0.052	+0.003	+0.037	+0.042	−0.034	−0.118	−0.032	+0.034
iTransformer	+0.091	+0.046	+0.012	+0.054	−0.009	+0.033	+0.114	+0.265

Table 35: **Gain g_{own} on Electricity.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.201**	**+0.319**	**+0.117**	**+0.055**	**+0.553**	**+0.483**	**+0.440**	**+0.301**
CNN	**+0.269**	**+0.614**	**+0.368**	**+0.245**	**+0.364**	**+0.631**	**+0.724**	**+0.395**
Transformer	**+0.189**	**+0.198**	**+0.084**	**+0.075**	**+0.246**	**+0.382**	**+0.377**	**+0.308**
DLinear	**+0.191**	**+0.282**	**+0.119**	**+0.114**	**+0.510**	**+0.413**	**+0.393**	**+0.276**
TSMixer	**+0.215**	**+0.194**	**+0.162**	**+0.055**	**+0.571**	**+0.400**	**+0.466**	**+0.334**
LightTS	**+0.223**	**+0.232**	**+0.199**	**+0.134**	**+0.491**	**+0.642**	**+0.680**	**+0.439**
FiLM	**+0.235**	**+0.260**	**+0.161**	**+0.186**	**+0.616**	**+0.925**	**+0.626**	**+0.449**
FreTS	**+0.210**	**+0.160**	**+0.129**	**+0.038**	**+0.427**	**+0.325**	**+0.425**	**+0.205**
TiDE	**+0.240**	**+0.272**	**+0.207**	**+0.138**	**+0.731**	**+1.030**	**+0.691**	**+0.505**
SCINet	**+0.280**	**+0.287**	**+0.215**	**+0.104**	**+0.655**	**+1.160**	**+0.685**	**+0.375**
MICN	**+0.148**	**+0.124**	**+0.058**	**+0.140**	**+0.355**	**+0.322**	**+0.391**	**+0.313**
TimesNet	**+0.267**	**+0.170**	**+0.173**	**+0.039**	**+0.333**	**+0.332**	**+0.332**	**+0.311**
SegRNN	**+0.072**	**+0.052**	**+0.038**	**+0.048**	**+0.150**	**+0.173**	**+0.164**	**+0.073**
TimeMixer	**+0.247**	**+0.156**	**+0.127**	**+0.067**	**+0.665**	**+0.507**	**+0.396**	**+0.340**
MSGNet	**+0.197**	**+0.172**	**+0.152**	**+0.049**	**+0.577**	**+0.406**	**+0.474**	**+0.342**
Informer	**+0.042**	**+0.048**	**+0.041**	**+0.013**	**+0.097**	**+0.074**	**+0.070**	**+0.056**
Autoformer	−0.020	−0.048	**+0.008**	**+0.004**	**+0.081**	**+0.051**	**+0.098**	**+0.091**
FEDformer	−0.014	**+0.079**	−0.100	−0.040	**+0.374**	**+0.157**	**+0.229**	**+0.003**
Pyraformer	−0.032	**+0.039**	**+0.013**	−0.018	**+0.128**	−0.003	**+0.024**	**+0.046**
Reformer	**+0.075**	**+0.071**	**+0.032**	**+0.057**	**+0.211**	**+0.064**	**+0.189**	**+0.180**
Nonstat. Trans.	**+0.111**	**+0.090**	**+0.089**	**+0.088**	**+0.089**	**+0.027**	**+0.025**	**+0.135**
Crossformer	**+0.157**	**+0.192**	**+0.115**	**+0.194**	**+0.249**	**+0.276**	**+0.244**	**+0.302**
PatchTST	**+0.218**	**+0.298**	**+0.104**	**+0.100**	**+0.342**	**+0.368**	**+0.272**	**+0.325**
TimeXer	**+0.299**	**+0.247**	**+0.173**	**+0.108**	**+0.704**	**+0.646**	**+0.437**	**+0.406**
MultiPatchFormer	**+0.343**	**+0.322**	**+0.220**	**+0.119**	**+0.395**	**+0.329**	**+0.153**	**+0.514**
iTransformer	**+0.401**	**+0.304**	**+0.249**	**+0.040**	**+0.551**	**+0.814**	**+0.672**	**+0.571**

Table 36: **Gain g_{own} on Exchange.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.131**	**+0.179**	**+0.078**	**+0.104**	**+0.247**	**+0.087**	−0.162	**+0.091**
CNN	**+0.079**	**+0.058**	**+0.174**	**+0.021**	**+0.183**	**+0.090**	−0.282	**+0.124**
Transformer	−0.033	−0.034	−0.068	−0.001	**+0.063**	−0.084	−0.167	−0.038
DLinear	**+0.034**	**+0.040**	**+0.029**	**+0.017**	**+0.029**	**+0.074**	−0.039	**+0.044**
TSMixer	**+0.074**	**+0.107**	**+0.076**	**+0.006**	**+0.344**	**+0.027**	**+0.010**	**+0.084**
LightTS	**+0.143**	**+0.142**	**+0.083**	**+0.019**	**+0.795**	**+0.176**	**+0.048**	**+0.159**
FiLM	**+1.452**	**+0.881**	**+1.162**	**+0.260**	**+4.966**	**+4.708**	−0.461	**+0.913**
FreTS	**+0.008**	**+0.009**	**+0.005**	−0.002	**+0.031**	**+0.007**	**+0.002**	−0.001
TiDE	**+0.681**	**+0.807**	**+0.496**	−0.077	**+2.017**	**+3.776**	−0.659	**+0.810**
SCINet	**+0.500**	**+0.597**	**+0.557**	**+0.043**	**+3.323**	−0.340	−0.167	**+0.817**
MICN	−0.006	**+0.008**	**+0.019**	**+0.010**	**+0.029**	**+0.020**	**+0.004**	**+0.011**
TimesNet	**+0.142**	**+0.163**	**+0.133**	**+0.069**	**+0.601**	−0.105	−0.338	**+0.070**
SegRNN	−0.734	−0.663	−0.385	**+0.044**	**+0.524**	**+0.440**	−0.469	−0.016
TimeMixer	**+0.183**	**+0.317**	**+0.290**	**+0.185**	**+2.096**	−0.519	−0.533	**+0.146**
MSGNet	−0.080	−0.271	**+0.056**	**+0.150**	**+0.131**	−0.025	−0.254	−0.016
Informer	−0.000	−0.000	**+0.000**	**+0.000**	**+0.001**	−0.000	−0.005	**+0.000**
Autoformer	**+0.146**	**+0.043**	**+0.251**	**+0.040**	**+0.074**	**+0.163**	**+0.353**	**+0.156**
FEDformer	−0.124	−0.059	−0.076	**+0.014**	−0.488	−0.004	**+0.012**	−0.122
Pyraformer	−0.000	−0.000	−0.001	−0.001	**+0.000**	**+0.000**	−0.000	−0.001
Reformer	**+0.019**	−0.054	−0.085	**+0.027**	**+0.117**	−0.220	−0.082	**+0.085**
Nonstat. Trans.	**+0.040**	−0.045	−0.076	−0.018	−0.055	−0.137	−0.260	−0.148
Crossformer	−0.001	**+0.004**	**+0.000**	−0.006	−0.009	**+0.002**	−0.009	−0.004
PatchTST	**+0.236**	**+0.316**	**+0.259**	**+0.004**	**+0.185**	−0.210	−0.189	**+0.482**
TimeXer	**+0.436**	**+0.199**	**+0.268**	**+0.091**	**+1.301**	−0.116	−0.501	**+0.445**
MultiPatchFormer	**+0.040**	**+0.103**	−0.085	−0.083	**+0.034**	**+0.055**	**+0.009**	−0.132
iTransformer	**+1.480**	**+1.159**	**+1.558**	**+0.760**	**+2.555**	−1.391	−0.118	**+0.697**

Table 37: **Gain g_{own} on Finance.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.004**	**+0.010**	−0.014	**+0.042**	**+0.018**	−0.022	−0.207	**+0.017**
CNN	**+0.036**	−0.056	**+0.046**	−0.132	**+0.192**	−0.067	−0.216	−0.104
Transformer	**+0.285**	**+0.332**	**+0.438**	**+0.067**	**+0.406**	−0.065	−0.932	**+0.371**
DLinear	**+0.068**	−0.006	**+0.029**	−0.002	**+0.055**	**+0.057**	−0.155	**+0.005**
TSMixer	**+0.052**	**+0.039**	**+0.023**	**+0.000**	**+0.089**	**+0.031**	**+0.015**	**+0.114**
LightTS	**+0.155**	**+0.114**	**+0.057**	−0.015	**+0.155**	**+0.116**	**+0.058**	**+0.134**
FiLM	**+0.101**	−0.280	**+0.072**	**+0.032**	**+1.356**	**+0.752**	−0.395	**+1.674**
FreTS	−0.027	−0.054	**+0.006**	−0.100	**+0.116**	−0.020	−0.052	−0.136
TiDE	**+0.257**	**+0.244**	**+0.122**	−0.020	**+0.881**	**+0.853**	**+0.007**	**+0.317**
SCINet	**+1.242**	**+1.109**	**+0.494**	−0.124	**+2.030**	−0.062	−0.049	**+1.421**
MICN	−0.060	**+0.006**	−0.003	−0.051	−0.028	**+0.088**	−0.003	−0.095
TimesNet	**+0.288**	**+0.234**	**+0.141**	**+0.006**	−0.017	**+0.072**	−0.103	**+0.103**
SegRNN	−0.927	−0.769	−0.174	**+0.021**	−0.010	−0.527	−0.255	−0.343
TimeMixer	**+0.149**	**+0.121**	**+0.055**	**+0.001**	**+0.330**	**+0.023**	−0.016	**+0.095**
MSGNet	−0.003	−0.015	**+0.040**	−0.028	**+0.062**	**+0.007**	−0.151	−0.015
Informer	−0.000	−0.001	−0.001	−0.002	−0.001	−0.000	−0.005	−0.000
Autoformer	−0.016	**+0.080**	**+0.077**	**+0.120**	−0.058	**+0.183**	**+0.496**	**+0.311**
FEDformer	−0.013	−0.049	−0.008	−0.001	**+0.145**	**+0.216**	**+0.080**	−0.035
Pyraformer	−0.000	**+0.000**	−0.003	−0.007	−0.011	−0.009	−0.004	−0.012
Reformer	−0.043	**+0.073**	−0.050	−0.008	−0.124	−0.026	−0.059	**+0.216**
Nonstat. Trans.	−0.043	**+0.018**	**+0.009**	−0.064	−0.012	−0.175	−0.023	−0.085
Crossformer	**+0.000**	−0.002	**+0.000**	**+0.000**	−0.000	−0.000	−0.001	**+0.000**
PatchTST	**+0.139**	**+0.190**	**+0.070**	**+0.034**	**+0.228**	−0.187	−0.153	**+0.368**
TimeXer	**+0.450**	**+0.658**	**+0.304**	−0.050	**+0.233**	−0.150	−0.172	**+1.070**
MultiPatchFormer	**+0.073**	**+0.063**	**+0.068**	**+0.008**	−0.006	**+0.039**	−0.087	**+0.073**
iTransformer	**+0.335**	**+0.318**	**+0.237**	−0.026	**+0.156**	−0.001	−0.078	**+0.273**

Table 38: **Gain g_{own} on Finance_ret.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.535**	**+0.271**	**+0.233**	−0.007	**+0.923**	**+0.698**	**+0.337**	**+0.591**
CNN	**+0.149**	**+0.079**	**+0.057**	**+0.019**	**+0.761**	**+0.148**	**+0.403**	**+0.141**
Transformer	**+0.232**	**+0.201**	**+0.203**	**+0.119**	**+0.569**	**+0.242**	**+0.484**	**+0.297**
DLinear	**+0.144**	**+0.113**	**+0.152**	**+0.009**	**+1.040**	**+0.039**	**+0.079**	**+0.349**
TSMixer	**+0.229**	**+0.179**	**+0.105**	**+0.011**	**+0.774**	−0.014	**+0.102**	**+0.305**
LightTS	**+0.316**	**+0.133**	**+0.199**	**+0.064**	**+1.361**	−0.021	**+0.205**	**+0.614**
FiLM	**+0.156**	**+0.106**	**+0.424**	**+0.019**	**+1.606**	**+1.414**	**+0.173**	**+0.899**
FreTS	**+0.313**	**+0.067**	**+0.155**	**+0.071**	**+0.350**	**+0.090**	**+0.369**	**+0.452**
TiDE	**+0.268**	**+0.184**	**+0.221**	**+0.054**	**+1.359**	**+0.433**	**+0.309**	**+0.578**
SCINet	**+0.272**	**+0.191**	**+0.166**	**+0.024**	**+1.618**	**+0.541**	**+0.219**	**+0.414**
MICN	**+0.185**	**+0.072**	**+0.064**	**+0.007**	**+0.621**	**+0.296**	**+0.157**	**+0.207**
TimesNet	**+0.123**	**+0.116**	**+0.078**	**+0.004**	**+0.426**	**+0.191**	**+0.302**	**+0.207**
SegRNN	−0.083	**+0.012**	−0.102	−0.094	−0.245	−0.032	−0.077	−0.156
TimeMixer	**+0.086**	**+0.134**	**+0.167**	**+0.044**	**+0.741**	**+0.460**	**+0.213**	−0.022
MSGNet	**+0.005**	**+0.071**	**+0.163**	**+0.051**	**+0.159**	**+0.155**	**+0.258**	**+0.023**
Informer	**+0.185**	−0.035	**+0.066**	−0.032	**+0.071**	−0.002	**+0.040**	**+0.176**
Autoformer	**+0.096**	**+0.030**	**+0.067**	**+0.180**	**+0.158**	**+0.160**	**+0.283**	**+0.355**
FEDformer	**+0.022**	**+0.027**	**+0.094**	−0.025	**+0.786**	**+0.097**	**+0.186**	**+0.072**
Pyraformer	**+0.023**	−0.014	−0.001	**+0.006**	−0.016	**+0.003**	−0.088	−0.001
Reformer	**+0.056**	−0.027	**+0.063**	**+0.170**	**+0.161**	**+0.479**	**+0.113**	−0.159
Nonstat. Trans.	**+0.329**	**+0.138**	**+0.010**	−0.058	**+0.024**	**+0.647**	**+0.528**	**+0.441**
Crossformer	**+0.119**	**+0.093**	**+0.087**	**+0.024**	**+0.333**	−0.004	−0.003	**+0.214**
PatchTST	**+0.302**	**+0.171**	**+0.175**	**+0.074**	**+0.823**	**+0.325**	**+0.292**	**+0.574**
TimeXer	**+0.217**	**+0.152**	**+0.142**	**+0.038**	**+0.414**	**+0.255**	**+0.199**	**+0.257**
MultiPatchFormer	**+0.100**	**+0.212**	**+0.290**	**+0.004**	**+0.153**	**+0.387**	**+0.318**	−0.096
iTransformer	**+0.296**	**+0.181**	**+0.141**	**+0.037**	**+0.978**	**+0.211**	**+0.181**	**+0.455**

Table 39: **Gain g_{own} on Solar.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.123**	**+0.182**	**+0.076**	**+0.047**	**+0.256**	**+0.241**	**+0.284**	**+0.185**
CNN	**+0.151**	**+0.344**	**+0.169**	**+0.062**	**+0.298**	**+0.629**	**+1.278**	**+0.211**
Transformer	**+0.100**	**+0.081**	**+0.035**	**+0.011**	**+0.086**	**+0.180**	**+0.203**	**+0.127**
DLinear	**+0.186**	**+0.137**	**+0.088**	**+0.045**	**+0.414**	**+0.468**	**+0.740**	**+0.174**
TSMixer	**+0.100**	**+0.085**	**+0.097**	**+0.013**	**+0.339**	**+0.209**	**+0.214**	**+0.143**
LightTS	**+0.144**	**+0.123**	**+0.134**	**+0.024**	**+0.413**	**+0.610**	**+0.690**	**+0.208**
FiLM	**+0.250**	**+0.207**	**+0.140**	**+0.025**	**+0.286**	**+0.913**	**+0.451**	**+0.217**
FreTS	**+0.120**	**+0.109**	**+0.120**	**+0.011**	**+0.325**	**+0.174**	**+0.131**	**+0.154**
TiDE	**+0.236**	**+0.237**	**+0.142**	**+0.025**	**+0.351**	**+1.402**	**+0.809**	**+0.265**
SCINet	**+0.035**	**+0.059**	**+0.098**	**+0.020**	**+0.273**	**+1.259**	**+1.144**	**+0.112**
MICN	**+0.152**	**+0.101**	**+0.065**	**+0.017**	**+0.047**	**+0.319**	**+0.260**	**+0.196**
TimesNet	**+0.190**	**+0.270**	**+0.390**	−0.014	**+0.311**	**+0.163**	**+0.461**	**+0.308**
SegRNN	**+0.043**	**+0.054**	**+0.084**	**+0.003**	**+0.200**	**+0.061**	**+0.129**	**+0.037**
TimeMixer	**+0.169**	**+0.149**	**+0.187**	**+0.004**	**+0.422**	**+0.528**	**+0.574**	**+0.136**
MSGNet	**+0.125**	**+0.078**	**+0.113**	**+0.014**	**+0.254**	**+0.279**	**+0.300**	**+0.137**
Informer	**+0.128**	**+0.118**	**+0.190**	**+0.073**	**+0.198**	**+0.037**	−0.035	**+0.177**
Autoformer	**+0.086**	**+0.068**	**+0.040**	**+0.010**	**+0.103**	**+0.172**	**+0.390**	**+0.194**
FEDformer	−0.032	**+0.012**	−0.026	**+0.012**	**+0.185**	**+0.082**	**+0.112**	**+0.004**
Pyraformer	**+0.024**	−0.004	**+0.023**	−0.003	**+0.070**	**+0.011**	−0.024	**+0.005**
Reformer	**+0.126**	**+0.081**	**+0.106**	**+0.020**	**+0.197**	**+0.073**	**+0.060**	**+0.099**
Nonstat. Trans.	**+0.040**	**+0.049**	**+0.043**	**+0.024**	**+0.125**	**+0.029**	**+0.034**	**+0.139**
Crossformer	**+0.128**	**+0.119**	**+0.147**	−0.008	**+0.323**	**+0.235**	**+0.181**	**+0.093**
PatchTST	−0.052	**+0.038**	**+0.057**	**+0.025**	**+0.172**	**+1.193**	**+0.666**	−0.064
TimeXer	**+0.079**	**+0.128**	**+0.125**	**+0.011**	**+0.265**	**+0.566**	**+0.518**	**+0.101**
MultiPatchFormer	**+0.182**	**+0.237**	**+0.380**	−0.015	**+0.333**	**+1.007**	**+0.412**	**+0.251**
iTransformer	**+0.205**	**+0.165**	**+0.164**	**+0.031**	**+0.313**	**+0.573**	**+0.414**	**+0.254**

Table 40: **Gain g_{own} on Traffic.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.087**	**+0.110**	**+0.074**	**+0.206**	**+0.474**	**+0.324**	**+0.417**	**+0.378**
CNN	**+0.028**	**+0.008**	**+0.009**	**+0.115**	**+0.183**	**+0.333**	**+0.486**	−0.027
Transformer	**+0.113**	**+0.093**	**+0.048**	**+0.067**	**+0.262**	**+0.230**	**+0.293**	**+0.177**
DLinear	**+0.076**	**+0.090**	**+0.079**	**+0.188**	**+0.467**	**+0.268**	**+0.378**	**+0.377**
TSMixer	**+0.123**	**+0.053**	**+0.078**	**+0.051**	**+0.391**	**+0.146**	**+0.205**	**+0.174**
LightTS	**+0.134**	**+0.130**	**+0.127**	**+0.154**	**+0.562**	**+0.676**	**+0.926**	**+0.439**
FiLM	**+0.111**	**+0.126**	**+0.088**	**+0.183**	**+0.535**	**+0.704**	**+0.452**	**+0.469**
FreTS	**+0.158**	**+0.110**	**+0.107**	**+0.067**	**+0.288**	**+0.238**	**+0.292**	**+0.197**
TiDE	**+0.203**	**+0.166**	**+0.160**	**+0.135**	**+0.540**	**+1.167**	**+0.767**	**+0.437**
SCINet	**+0.162**	**+0.131**	**+0.128**	**+0.129**	**+0.503**	**+1.176**	**+0.934**	**+0.410**
MICN	**+0.079**	**+0.024**	−0.020	**+0.065**	**+0.134**	**+0.145**	**+0.152**	**+0.132**
TimesNet	**+0.205**	**+0.132**	**+0.103**	**+0.039**	**+0.035**	**+0.221**	**+0.202**	**+0.236**
SegRNN	−0.020	−0.029	−0.015	−0.016	**+0.026**	**+0.046**	**+0.024**	−0.037
TimeMixer	**+0.124**	**+0.078**	**+0.054**	**+0.046**	**+0.150**	**+0.222**	**+0.164**	**+0.160**
MSGNet	**+0.108**	**+0.027**	**+0.024**	**+0.011**	**+0.069**	**+0.126**	**+0.131**	**+0.138**
Informer	−0.000	−0.015	**+0.017**	**+0.038**	**+0.097**	**+0.041**	−0.004	**+0.009**
Autoformer	**+0.003**	−0.024	−0.017	**+0.020**	−0.002	**+0.033**	**+0.073**	−0.011
FEDformer	−0.103	−0.036	−0.100	−0.074	**+0.146**	**+0.112**	**+0.201**	−0.094
Pyraformer	−0.067	−0.029	**+0.007**	−0.036	**+0.022**	−0.008	−0.022	−0.039
Reformer	**+0.131**	**+0.101**	**+0.112**	−0.001	**+0.080**	**+0.074**	**+0.010**	**+0.105**
Nonstat. Trans.	**+0.026**	**+0.022**	**+0.024**	**+0.016**	**+0.051**	−0.000	**+0.032**	**+0.061**
Crossformer	**+0.106**	**+0.044**	**+0.050**	**+0.086**	**+0.110**	**+0.093**	**+0.114**	**+0.203**
PatchTST	**+0.122**	**+0.085**	**+0.065**	**+0.075**	**+0.230**	**+0.336**	**+0.290**	**+0.208**
TimeXer	**+0.168**	**+0.089**	**+0.083**	**+0.071**	**+0.265**	**+0.319**	**+0.250**	**+0.263**
MultiPatchFormer	**+0.169**	**+0.100**	**+0.078**	**+0.064**	**+0.080**	**+0.198**	**+0.213**	**+0.218**
iTransformer	**+0.254**	**+0.189**	**+0.143**	**+0.005**	**+0.258**	**+0.501**	**+0.344**	**+0.340**

Table 41: **Gain g_{own} on Weather.** Every backbone against every estimator, where a cell is the median over the swept settings.

Backbone	Sal.	G×I	IG	Occ.	Dyna.	ExtrM.	CLSP	Grad
Linear	**+0.098**	**+0.072**	**+0.063**	**+0.001**	**+0.108**	**+0.048**	**+0.064**	**+0.081**
CNN	**+0.182**	**+0.168**	**+0.165**	**+0.095**	**+0.043**	**+0.732**	**+0.860**	**+0.125**
Transformer	**+0.018**	−0.007	−0.007	**+0.011**	**+0.077**	**+0.027**	**+0.044**	**+0.011**
DLinear	**+0.099**	**+0.079**	**+0.074**	**+0.012**	**+0.165**	**+0.085**	**+0.066**	**+0.079**
TSMixer	**+0.212**	**+0.067**	**+0.077**	−0.008	**+0.348**	**+0.109**	**+0.225**	**+0.283**
LightTS	**+0.142**	**+0.051**	**+0.075**	−0.007	**+0.475**	**+0.507**	**+0.803**	**+0.194**
FiLM	**+0.248**	**+0.179**	**+0.141**	**+0.043**	**+0.545**	**+1.467**	**+1.028**	**+0.239**
FreTS	**+0.065**	**+0.007**	**+0.009**	**+0.009**	**+0.154**	**+0.020**	**+0.021**	**+0.060**
TiDE	**+0.240**	**+0.151**	**+0.127**	−0.037	**+0.437**	**+1.290**	**+0.723**	**+0.342**
SCINet	**+0.206**	**+0.110**	**+0.154**	**+0.018**	**+0.458**	**+1.355**	**+1.406**	**+0.120**
MICN	**+0.096**	−0.009	−0.033	−0.004	**+0.277**	**+0.118**	**+0.154**	**+0.129**
TimesNet	**+0.098**	**+0.000**	**+0.004**	−0.051	**+0.179**	**+0.208**	**+0.068**	**+0.085**
SegRNN	−0.042	−0.066	−0.022	−0.023	−0.016	−0.037	**+0.024**	−0.043
TimeMixer	**+0.079**	**+0.031**	**+0.006**	**+0.010**	**+0.285**	**+0.312**	**+0.249**	**+0.111**
MSGNet	**+0.032**	−0.009	**+0.001**	−0.011	**+0.091**	**+0.040**	**+0.070**	**+0.046**
Informer	**+0.067**	**+0.042**	**+0.012**	**+0.042**	−0.071	−0.048	−0.013	**+0.064**
Autoformer	**+0.349**	**+0.392**	**+0.510**	**+0.093**	**+0.436**	**+0.384**	**+0.767**	**+1.154**
FEDformer	**+0.064**	**+0.208**	**+0.025**	−0.068	**+0.320**	**+0.034**	**+0.578**	**+0.026**
Pyraformer	−0.026	**+0.002**	−0.007	**+0.014**	**+0.112**	**+0.010**	−0.040	−0.008
Reformer	**+0.062**	**+0.006**	**+0.041**	**+0.012**	**+0.076**	**+0.013**	**+0.059**	**+0.084**
Nonstat. Trans.	−0.013	−0.004	−0.003	**+0.001**	−0.078	−0.009	**+0.000**	−0.010
Crossformer	**+0.065**	**+0.004**	**+0.016**	**+0.058**	**+0.144**	**+0.014**	**+0.003**	**+0.074**
PatchTST	**+0.082**	**+0.033**	**+0.029**	**+0.044**	**+0.085**	**+0.140**	**+0.096**	**+0.072**
TimeXer	**+0.080**	**+0.041**	−0.002	**+0.008**	**+0.243**	**+0.214**	**+0.130**	**+0.082**
MultiPatchFormer	**+0.071**	**+0.003**	**+0.011**	**+0.075**	**+0.091**	**+0.079**	**+0.036**	**+0.054**
iTransformer	**+0.144**	**+0.056**	**+0.044**	**+0.000**	**+0.137**	**+0.191**	**+0.122**	**+0.138**

R ANTICIPATED QUESTIONS

Why a row of E is more than a per-output attribution. A row is what an attribution library returns when its target is set to $\hat{y}[h]$, and we keep it that way on purpose. This is what lets HRX attach to any differentiable forecaster without a new estimator and without modifying the TS backbone. What prior works lack is the evidence that the rows differ in a way that matters, a protocol that scores one forecast step, and a criterion for when the rows are worth keeping. As shown in Table 1, every TS explanation method reports and evaluates only the shared vector, and no established metric scores a single step (Appendix P.2). Our contribution is therefore the axis together with the means to validate it, which is why Section 4.3 applies the axis to seven other estimators instead of proposing one.

Why the band is more than a seasonal lag. A model that copies the value one period back has a diagonal band by definition, and the `lagmix` generator of Table 12 plants exactly this case. Our finding is that the rows differ across 26 backbones, including on the financial log returns, where periodicity is nearly absent and the gain is the largest of any group (Table 7). Furthermore, a band is not needed for a gain, since two neighboring steps already read different positions (Table 20). We therefore read the band on periodic data as one instance of the horizon axis, not as its evidence.

Why the gradient does not grade itself. The default row is a local derivative, whereas deletion reads the model far from where that derivative was taken, and three results do not involve the gradient at all. First, the mask methods of Table 6 optimize their own objective under a perturbation, and each shows a gain at least as large as the gradient. Second, Sufficiency and Insertion keep the ranked positions instead of deleting them, which perturbs in the opposite direction to our gain (Appendix F.3). Third, the ground-truth metrics of Table 3 compare an explanation against the positions that generated the target and involve no deletion. Note that Comprehensiveness and AOPC read the same deletion curve as our gain, and we therefore do not count them as independent evidence.

Why the baseline is the shared vector. The shared vector $\bar{E}$ is the same estimator averaged over the horizon, which means the comparison changes the axis and nothing else. A published whole-forecast output would change the objective of the estimator at the same time, and a gain could then come from either change. For the gradient, that output is the derivative of the summed forecast, which differs from $\bar{E}$ in that signed rows can cancel before the magnitude is taken. We did not score this output at a single step, while Table 19 scores every estimator in its published form over the whole forecast. Our comparison therefore isolates the axis for a fixed estimator, and it should not be read as a ranking against published outputs.

Why the synthetic generators are linear. We build the generators as linear maps, since only a linear target fixes without ambiguity which positions a step depends on. A nonlinear target can often be computed from several different sets of positions, which is the failure we met with the sinusoid of Appendix G. The backbones still learn the map from data, and recovery is not automatic, as it is weaker on several Transformer variants and on Autoformer the matrix does not beat the shared vector (Table 13). We therefore read the synthetic check as validating the instrument, and the real-data results as the evidence that forecasters carry such structure.

Why the sharpest band can come from a model that cannot forecast. Table 7 compares skill within one series, where the runs that beat the naive baselines always read the larger gain. The log returns are close to unpredictable, which means almost all of their runs lose to the baseline, and they still read the largest gain of any series (Figure 11). An explanation describes what the model uses rather than what drives the market, and a model can read different positions for different steps without being right about any of them. This is why we report a forecasting baseline beside every result, which keeps such a band from being read as evidence about the series.

Whether the low rank comes from reading $|E|$. A nonnegative matrix carries a dominant direction close to its mean row, which pulls the participation ratio toward one. The low values we report partly reflect this, and we therefore do not read the rank as an absolute count of maps. We read it as a relative criterion instead, and it works as one, since the gain rises with it (Table 10) and it tracks a planted rank on every backbone (Appendix G). Note that the rank-one row of Table 9 is an analytic identity rather than an experimental result, and we report it only as a check of the pipeline.

How the number of maps grows with the forecast length. The number of maps grows with the forecast length, but far more slowly than the forecast length itself. As shown in Table 42, the last column pairs each run with the run at $H = 96$ that shares every other setting, since the backbones

differ between rows. The paired change stays well below one direction while H grows thirtyfold, which is what we mean by low-dimensional. Furthermore, the gain is larger at long horizons than at short ones, in line with Appendix K.1.

Table 42: **Effective rank by forecast length.** Medians over the main runs at each H. The last column is the median paired change against the run at $H = 96$ with every other setting held fixed.

H	Runs	Backbones	$r(E)$	g_{own}	Paired $\Delta r(E)$ vs. H=96
24	801	25	1.37	**+0.095**	−0.39 (560 pairs)
48	606	10	1.60	**+0.188**	−0.17 (519 pairs)
96	22,320	26	1.53	**+0.137**	reference
192	650	10	2.16	**+0.272**	+0.11 (452 pairs)
336	647	25	2.31	**+0.239**	+0.30 (437 pairs)
720	182	10	2.03	**+0.228**	+0.31 (87 pairs)

Whether the intervals are too narrow. Runs that share a backbone and a benchmark are correlated, which means an interval that resamples runs understates the uncertainty. Table 43 therefore resamples whole (backbone, benchmark) clusters instead. The interval widens as expected, while it still excludes zero for both quantities. Furthermore, most clusters have a positive median on their own, which does not depend on any resampling scheme, and Appendix H names the backbones and benchmarks whose interval includes zero.

Table 43: **Run-level against cluster-level intervals.** A cluster is one (backbone, benchmark) pair. The cluster interval resamples clusters with replacement and takes the pooled median.

Quantity	Runs	Clusters	Median	95% CI (run)	95% CI (cluster)	Clusters > 0
g_{own}	25,517	390	**+0.143**	[+0.139, +0.147]	[+0.106, +0.183]	**78%**
g_{margin}	25,517	390	**+0.217**	[+0.214, +0.220]	[+0.184, +0.252]	**91%**

What the protocol reports on the negative control. On `sparsenull` every step depends on the same positions, which makes the ideal value of every quantity zero. As shown in Table 44, g_{own} sits at zero there, while it is clearly positive on the five generators that plant a horizon structure. The margin is not exactly zero on the control, since two rows that differ only in estimation noise still rank the lookback slightly differently. It stays more than an order of magnitude below the margin on the planted generators, which bounds the margin the instrument alone can produce.

Table 44: **The protocol on the negative control.** Medians over synthetic runs, with the fraction of positive runs in brackets.

Generators	Runs	g_{own}	g_{shuffled}	g_{margin}
Five with a horizon structure	1,125	+0.153 (81%)	−0.208 (2%)	+0.399 (99%)
`sparsenull` (negative control)	228	+0.004 (54%)	−0.029 (20%)	+0.014 (66%)

Why the mask methods are estimated at a few steps. Each mask method solves one optimization per forecast step, and a full row set at the longest horizon would therefore cost hundreds of solves per window (Appendix K.2). Note that every estimator of Table 6, including the gradient, is scored at the same six steps, which means no estimator receives a finer resolution than another (Appendix E.3). We leave out Shapley estimators for the same reason, since each row would need its own coalition sampling. Occlusion is the perturbation estimator closest to them among those we include.

Whether the channel result comes from channel-independent backbones. The channel result partly reflects channel-independent backbones, and we report it as such. The multivariate runs of Table 8 include backbones that process every channel separately, and on these the channel rows are copies of one vector by construction. Without the four backbones whose channel rank is exactly one, the channel gain still stays more than an order of magnitude below the time gain. Note that

the channel protocol deletes a whole channel by replacing it with its window mean rather than by resampling positions, and it reads the error over all output channels at step h.

What a measurement tool offers in practice. HRX tells a user which past values each forecast step rests on, and the rank criterion tells them whether this per-step view adds anything to the shared vector. For a reviewer of a forecast, such as an auditor in the financial domain, the second part is the more useful one. As shown in Table 10, a rank near one licenses the familiar single vector, whereas a larger rank indicates that the vector hides a difference between steps. We found no downstream use that yields a better forecast (Appendix L), and we therefore report HRX as a measurement tool.