Tokenization and Architecture Jointly Allocate Component Roles in Time Series Transformers

Mingyu Kim
Department of Artificial Intelligence
Inha University
Incheon, Republic of Korea
mingyu@inha.edu

Doguk Kim*
Department of Artificial Intelligence
Inha University
Incheon, Republic of Korea
dgkim@inha.ac.kr

Abstract

Time series transformers have achieved strong forecasting performance, yet it remains unclear which of their components, attention or the feed-forward network (FFN), constructs temporal features, and how tokenization such as patching changes this. We propose a temporal probing framework that uses 15 token-level diagnostic tasks to measure feature construction as the gain in linear decodability of temporal properties from one sub-layer to the next (a matched MLP probe already saturates at the embedding). It targets models whose probed representations are read by a linear forecasting head. Applied to 5 supervised models trained on 8 long-term forecasting benchmarks, it finds that the constructing component is not fixed but allocated by the interaction of tokenization and architecture. Under patch tokenization (PatchTST), the FFN's gain exceeds attention's in probing on all 8 datasets, and its 8-dataset mean is 3.2$\times$ attention's. Without attention, the FFN alone reaches nearly the same total gain, both when attention is removed from a trained model and when PatchTST is trained without it, and the retrained model forecasts within a few percent of the full model. In probing, attention dominates under variate tokenization (iTransformer); in CATS, cross-attention is the only input path. In a controlled manipulation of PatchTST, single-scalar tokens (P=1) nearly eliminate the FFN's construction, while one more point per token (P=2) restores it. Our forecasting-side interventions (activation patching, knockout, and training without attention) establish component-level reliance and sufficiency. These results offer a representational reading of prior findings on the gains from patching, the competitiveness of self-attention-free models (TSMixer, PatchMLP), and the distinct role of cross-attention in CATS.

https://github.com/mingyu-11/tokenization-component-roles

1 Introduction

Time series transformers have evolved rapidly, and the results reported along the way appear contradictory. Early forecasters such as Informer [Zhou et al., 2021], Autoformer [Wu et al., 2021], and FEDformer [Zhou et al., 2022] use time-step tokens, and a one-layer linear model, DLinear [Zeng et al., 2023], surpassed all of them. PatchTST [Nie et al., 2023] restored transformer dominance through patch tokenization and channel independence (variates modeled separately), and subsequent work questioned whether attention is needed: TSMixer [Ekambaram et al., 2023] keeps patching but replaces self-attention with MLP-Mixer blocks [Tolstikhin et al., 2021], and PatchMLP [Tang and Zhang, 2025] attributes the gains mainly to patching rather than attention. CATS [Kim et al., 2024] replaces self-attention with cross-attention from one learnable query per future patch, and

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

iTransformer [Liu et al., 2024] makes each variate's whole lookback window one token. These results may reflect an underlying pattern: transformers with time-step tokens have been reported to lose to linear models and those with patch or whole-variate tokens to win, and attention can be removed with little or no loss in some models while cross-attention alone suffices in others. The same name, "attention," may thus cover different information flows.

Telling these flows apart requires observing model internals rather than comparing final outputs. We bring linear probing [Alain and Bengio, 2016], well established in NLP (§2), to the sub-layers of supervised models and ask, along two axes, tokenization (point, patch, or variate) and attention type (self-attention, cross-attention, or none), how the relative roles of attention and the feed-forward network (FFN) shift and how the role of the same attention mechanism changes with tokenization. Because the models in our main analysis read their representations through a linear forecasting head, we treat a gain in linear decodability as feature construction and complement probing with forecasting-side interventions. Our contributions are:

- **Temporal probing framework.** We design 15 token-level diagnostic tasks and a controlled protocol that locates where temporal properties become linearly decodable, with sub-layer extraction, attention/FFN separation, and variants that remove attention or change the patch length or stride.
- **Role allocation by tokenization and architecture.** In probing, the dominant constructing component is allocated by the interaction of tokenization and architecture: the FFN under patching with self-attention, attention under variate-level tokenization, the cross-attention path in CATS, and attention under single-scalar tokens (PatchTST with patch length $P{=}1$), where the total gain is smaller. Under patching, the allocation recurs across patch lengths, strides, and horizons (each on 4 datasets) and three added probing targets. On the forecasting side, activation patching, training without attention, and knockout establish component-level reliance and sufficiency in the standard PatchTST, and knockout ranks the architectures by attention's share as probing does.
- **A representational reading of prior results.** We read reported results for DLinear, PatchTST, TSMixer, PatchMLP, and CATS qualitatively (§6) and suggest three design practices (§7).

2 Related Work

Analyses of supervised time series transformers. Liang et al. [2025] found that transformer blocks often degenerate into MLPs, with performance concentrated in the FFN, and traced this to embeddings that do not let attention operate in a well-structured latent space. They relied on forecasting error and attention weights, whereas we ask where temporal properties become linearly decodable. Chen et al. [2025b] compared point-wise, patch-wise, and variate-wise tokenization through mutual information, identifying intra-variate dependency as the primary contributor to benchmark performance, but did not separate the roles of attention and the FFN. Other analyses cover the geometry of layer representations [Kaufman and Azencot, 2024], neuron-level attribution [Wang et al., 2025], convergence failures [Ilbert et al., 2024], and concept bottlenecks [van Sprang et al., 2024]. Two concurrent works take other routes. Yıldırım [2026] found that the FFN of a single-layer PatchTST variant needs no superposition (packing more features than dimensions) and argued that benchmarks demand little nonlinear feature composition; this, however, does not rule out nonlinearly constructed features, and zeroing that FFN raised MSE in most of their settings. Wang et al. [2026] reported that a parameter-free identity encoder often matches learned encoders once the embedding is well designed.

Analyses of time series foundation models. Wiliński et al. [2025] probed the residual stream after each block and found concepts such as trend and periodicity linearly separable at specific layers. Bao et al. [2026] ablated MLPs and attention heads layer by layer, found that MLP ablation hurts decoder-only models much more than the encoder-decoder Chronos-Bolt, and asked how architecture determines what MLPs and attention heads each carry. Yu et al. [2026] showed that the numerical rank of patch embeddings is bounded by a quantity that grows with patch size rather than with the embedding dimension. Our work complements these studies by probing frozen supervised models, each trained on a single dataset, at sub-layer resolution, and §8 states when our framework covers foundation models.

Probing and component roles in NLP. Probing trains a predictor on frozen representations [Conneau et al., 2018, Tenney et al., 2019, Hewitt and Manning, 2019, Hewitt and Liang, 2019, Belinkov, 2022], in its simplest form a linear one [Alain and Bengio, 2016]. NLP work has separated component

roles: Geva et al. [2021, 2022] showed that FFN layers function as key-value memories that build predictions across layers. Elhage et al. [2021] and Olsson et al. [2022] identified attention's induction heads and their role in in-context learning, building on earlier attention-head analyses [Voita et al., 2019]. Dong et al. [2025] used MixiT, whose attention scores are random, to show that attention largely drives in-context reasoning while MLPs, together with attention, store knowledge, and Chen et al. [2025a] showed that feed-forward layers learn distributional associations. Probing shows what is linearly decodable rather than what a model uses, so we pair it with component-level interventions such as activation patching [Meng et al., 2022] (§8).

3 Method: Temporal Probing Framework

3.1 Probing Methodology

We freeze the weights of a trained transformer and extract intermediate representations at the embedding and after the two standard sub-layers [Vaswani et al., 2017] of each layer l (Figure 1): `embed` (token embedding output), l`a_attn` (after attention, residual connection, and normalization), and l`b_ffn` (after FFN, residual connection, and normalization), e.g., `1a_attn` and `1b_ffn` for Layer 1. On each frozen representation, we train a linear probe [Alain and Bengio, 2016] for each temporal property: its input is one token's representation $h \in \mathbb{R}^{d_{\text{model}}}$, and its output, $Wh + b$, has one logit per class of the property of the token's segment (one output for regression). Normalization is BatchNorm in PatchTST and its variants [Nie et al., 2023] and LayerNorm in iTransformer, CATS, and Informer. BatchNorm is affine at inference, so it does not change what a linear probe can decode; LayerNorm is a token-wise nonlinearity, so in these 3 models the attention delta defined below also reflects it.

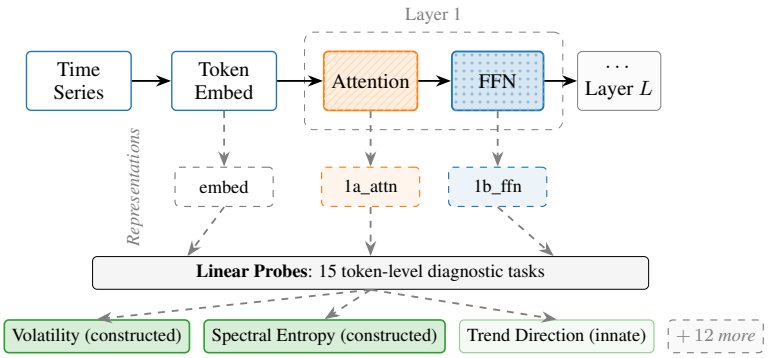


Figure 1: **Probing protocol with delta-based component attribution.** Linear probes on the frozen sub-layer outputs (`embed`, `1a_attn`, `1b_ffn`) predict 15 token-level temporal properties (§3.2); per-component deltas (Δ) attribute accuracy gains to sub-layers.

Construction as linear decodability. A sub-layer *constructs* a property when the property becomes linearly decodable from its output. This is a change in usable information (here, linearly extractable), not Shannon information [Xu et al., 2020]: PatchTST embeds each token injectively, so no sub-layer adds Shannon information about a property computed from the token's own values [Pimentel et al., 2020]. As a proxy for this change, not an estimate of it, we use the gain in probing accuracy over the preceding probing point, in the spirit of baselined probing [Hewitt et al., 2021]:

$$\Delta_{\text{attn}}(l) = \text{Acc}(l\text{a_attn}) - \text{Acc}(\text{prev}), \qquad \Delta_{\text{FFN}}(l) = \text{Acc}(l\text{b_ffn}) - \text{Acc}(l\text{a_attn})$$

where prev $=$ embed if $l = 1$, else $(l-1)$b_ffn. A delta near zero or below means no net gain in linear decodability, and the two deltas at a layer indicate which component constructs; a delta does not measure forecasting quality or whether the forecaster uses the property [Elazar et al., 2021]. Like the heads through which these forecasters read their representations, our probes are linear rather than maximally expressive [Pimentel et al., 2020]: a matched MLP probe already saturates at the embedding (Appendix E.2). On the forecasting side, three interventions complement probing: activation patching, which restores a component's clean output into a run on corrupted input, tests whether attention contributes at all (in PatchTST); training without attention tests whether the FFN suffices; and knockout, which zeroes a component's output in all layers of a frozen model, tests which component a trained model relies on.

Table 1: Tasks (left: mainly intra-token; right: inter-token, regression, and control; Cl.: classes).

Task	Cl.	Dom.	Computation	Task	Cl.	Dom.	Computation
volatility$^{\mathrm{C}}$	4	stat	var(patch) quartile	adjacent_similarity	2	stat	$\cos(p_i, p_{i+1}) >$ median
diff_energy$^{\mathrm{C}}$	4	diff	mean($(\Delta x)^2$) quartile	relative_magnitude	2	stat	mean(p_i) > mean(all)
spectral_entropy$^{\mathrm{C}}$	4	freq	$-\sum q_k \log q_k$ quartile	position_quarter	4	time	token-index quarter
trend_direction$^{\mathrm{I}}$	2	time	sign($x_P - x_1$)	trend_consistency	2	time	3-token trend alignment
slope_magnitude$^{\mathrm{B}}$	4	time	|linear_slope| quartile	reg_variance$^{\mathrm{C}}$	R^2	stat	var(patch) continuous
change_point$^{\mathrm{B}}$	2	time	$\lvert\bar{x}_{1\mathrm{st}} - \bar{x}_{2\mathrm{nd}}\rvert$ top 30%	reg_slope$^{\mathrm{I}}$	R^2	time	$x_P - x_1$ continuous
dominant_period$^{\mathrm{D}}$	4	freq	$\arg\max$ of 4 |rfft| band sums	reg_next_mean	R^2	pred	mean(next_patch)
local_global_trend	2	time	sign(local) = sign(global)	random_label	4	—	random label

3.2 Task Design

We design 15 token-level diagnostic tasks (Table 1), adapted to each model's tokenization (§3.3): eight **intra-token**, four **inter-token**, and three **regression** tasks, plus a random-label **control**. The suite is built to locate where temporal properties are constructed, not to inventory what forecasting needs; §5.3 tests what it may miss. Table 1 marks innate (I), constructed (C), boundary (B), and degenerate (D) tasks, defined below; unmarked tasks fall outside this classification.

Innate and constructed tasks. PatchTST's embedding is an affine map $W_P\tilde{x} + W_{\text{pos}}$ of the normalized patch $\tilde{x}$ from which intra-token labels are computed, so a linear probe on it can decode only labels that a linear read-out of the patch (and its position) determines. *Innate* tasks such as trend_direction (whether the patch rises) have such labels. *Constructed* tasks such as volatility, diff energy, and spectral entropy do not: the first two are quadratic forms and the third a scale-invariant function of the power spectrum, all functions of second-order statistics, so they can become linearly decodable only after a nonlinearity (the FFN's activation or attention's content-dependent weights).

Main tasks. The main analysis uses three constructed tasks: volatility, diff energy, and spectral entropy. Only constructed tasks can show which component constructs, since innate ones are linearly decodable at the embedding and leave nothing to attribute. Of the other constructed tasks, slope_magnitude and change_point only take the absolute value of a linear quantity the embedding already exposes, so we report them as boundary tasks, though on ETTm1 both also gain mainly at the Layer-1 FFN (Appendix B.2). The main tasks lack such a linear part and view temporal complexity from statistical, differential, and spectral perspectives (results for all tasks in Appendix B).

Supporting roles. The other tasks either check the classification or ask a different question: trend_direction shows what the embedding already exposes (§5.1), reg_slope and reg_variance repeat the innate-versus-constructed contrast with continuous labels (Appendix B.3), and the random-label control checks probe reliability. The inter-token tasks, reg_next_mean, and local_global_trend have labels that depend on other tokens, so they ask what reaches a token from elsewhere, not only what is constructed within it: routing between tokens (Appendix B.10), what a token carries about the next segment, and whether its trend matches that of the whole window, respectively. Finally, dominant_period, meant to capture the strongest frequency band, is degenerate: one band wins so often that its embedding accuracy matches the majority-class rate on every dataset (Appendix B.1).

Other temporal properties. Forecasting work often stresses four further properties: trend shifts, seasonality, multi-scale structure, and long-range dependence. Trend shifts are probed by change_point and local_global_trend. For the other three, at the granularity a 16-point token permits, we add three targets on PatchTST (results in §5.2). Seasonal (spectral) concentration and a multi-scale ratio are both constructed ratios of second-order statistics; long-range consistency compares a token's slope with that of the preceding eight tokens and so also needs routing.

3.3 Architecture-Specific Adaptations

For patch-based models (PatchTST, CATS), task labels are computed on P time points per token after the model's input normalization (for a CATS query, on the future patch it forecasts). For iTransformer, each token represents an entire variate's normalized lookback series, so task labels are computed per variate, before this normalization, which would make volatility nearly constant. As a token lacks its own series' mean and standard deviation, part of attention's gain can come from other tokens

(Appendix B.4). For PatchTST (P=1) and Informer, we apply a mean adapter: 16 consecutive representations (stride 8) are mean-pooled; Informer's pooled vector mixes all variates yet is probed against per-variate labels (Appendix B.9). Absolute accuracies are comparable only within the PatchTST family, which shares the same task granularity (16-point patches); for iTransformer, we compare the ratio of the two deltas.

3.4 Probe Training Protocol

All probes are single linear layers trained with Adam (lr $= 10^{-3}$), batch size 32,768, for 30k gradient steps on an 80/20 train/test split (Appendix A describes how samples are drawn, labeled, and split). We evaluate every 500 steps and report the best test accuracy over the entire training trajectory. All experiments use 3 model seeds $\times$ 3 probe seeds = 9 runs. While probing accuracy rises slightly beyond 30k (e.g., $+1.6$ percentage points (pp) at `1b_ffn` by 80k on volatility), the component that leads on each task is the same from 5k steps onward on all 4 datasets tested (Appendix E).

4 Experimental Setup

We arrange 5 models (3 architectures and 2 PatchTST variants) by tokenization and attention type (Table 4 in Appendix A): at the patch level, PatchTST [Nie et al., 2023] (self-attention), CATS [Kim et al., 2024] (cross-attention), and PatchTST-NoAttn (no attention); at the variate level, iTransformer [Liu et al., 2024]; and at the point level, PatchTST (P=1), with single-scalar tokens. Informer [Zhou et al., 2021], whose time-step tokens hold all variates, is an auxiliary case.

PatchTST-NoAttn sets the self-attention output to zero, reducing the sub-layer to its residual identity with all else unchanged. We do so either in a trained checkpoint with the FFN weights fixed (*ablated*; probing differences then reflect only the removal of attention) or from initialization on all 8 datasets (*from scratch*, as in RAM [Guo et al., 2024], which found that this costs little in forecasting).

PatchTST (P=1) changes only the patch length P and the stride to 1, so the model differs from PatchTST only in tokenization; its 16-point tasks are read from 16 consecutive tokens through an adapter (§3.3). PatchTST is also trained with $P \in \{2, 4, 8, 32\}$ (stride $P/2$) and with non-overlapping patches (P = stride = 16) on 4 datasets (ETTh1, ETTm1, Weather, Electricity). **Informer**, probed on the same 4 datasets, is not a point-level model: its tokens are time steps embedded by a three-step convolution over all variates (Appendix B.9).

PatchTST and its variants, iTransformer, and CATS use per-dataset hyperparameters from the original papers and official scripts and, for datasets they do not cover, those of the most similar dataset in scale (Appendix A). The input length seq_len is 336 for PatchTST and its variants and 96 for the other models. Inputs are instance-normalized with RevIN without affine parameters [Kim et al., 2022] in PatchTST and its variants and with built-in series normalization in iTransformer and CATS. All models use pred_len $= 96$ in the main analysis; PatchTST results for pred_len $\in \{192, 336, 720\}$ on the same 4 datasets are in Appendix B.13. Appendices G and H list the models, datasets, and seeds used for Layer-1 activation patching and component knockout.

Datasets. We use 8 long-term forecasting benchmarks with standard splits (Table 8 in Appendix A): ETTm1, ETTm2, ETTh1, and ETTh2 [Zhou et al., 2021], Weather [Wu et al., 2021], and Electricity, Exchange, and Traffic [Lai et al., 2018]. Solar-Energy is excluded from the main analysis for data quality reasons (Appendix C). Probe samples come from each dataset's test split.

5 Results

Unless stated otherwise, we report Layer-1 deltas and their sum (the total) in pp of probing accuracy averaged over the three constructed tasks of §3.2, as 8-dataset means of 9-run means. Each of these tasks labels a token by the quartile of one statistic of its time points (16 in PatchTST; chance 25%): *volatility* is their variance, *diff energy* the mean squared step between consecutive points (larger for rougher segments), and *spectral entropy* how evenly their power spreads over frequencies (low for a clean oscillation, high for noise). The other tasks play the supporting roles of §3.2 and are analyzed in Appendices B, B.10, and F. Deltas locate where a property becomes linearly decodable and do not measure forecasting quality; the forecasting-side interventions establish component-level reliance and sufficiency, not that a model builds and uses the individual probed features (§8).

We separate innate from constructed properties at the embedding (§5.1), show in probing that the FFN constructs in PatchTST and test with activation patching whether attention contributes (§5.2), remove attention from trained checkpoints and from training (§5.3), reduce tokens to single scalars and vary patch length and stride (§5.4), and compare architectures in probing and with component knockout (§5.5). Datasets and exceptions are given with each claim.

5.1 Innate vs. Constructed: Two Classes of Temporal Information

PatchTST's embedding shows the dichotomy of §3.2 (Figure 2): the linear patch projection keeps the **innate** trend direction at 94.9 to 99.9% accuracy, whereas the **constructed** properties stay at 27.7 to 54.9% (chance 25%) until a nonlinearity acts on them (§5.2). Regression shows the same split: reg_slope reaches $R^2 \geq 0.98$ on every dataset, while reg_variance stays at most 0.15 except on Traffic (0.50; Appendix B.3).

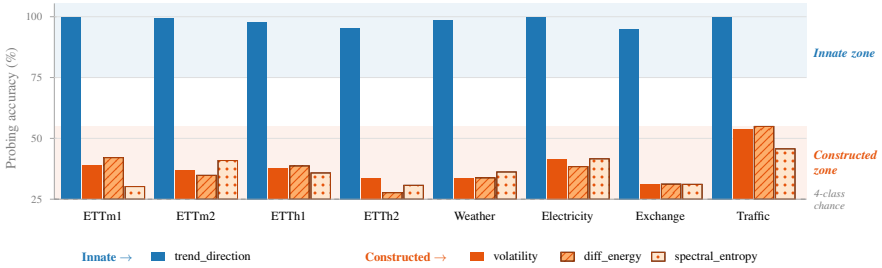


Figure 2: **Embedding-layer probing reveals two classes of temporal information.** PatchTST's patch embedding preserves the *innate* trend direction (94.9 to 99.9%), whereas *constructed* properties (volatility, diff energy, spectral entropy) reach only 27.7 to 54.9% (Appendix B.1).

Control validation. As a pipeline sanity check, a random-label probe stays at chance ($\leq 26.4\%$) at every layer in all 4 models of Appendix F, against PatchTST's Layer-1 constructed-task gain of $+26.7$ pp (Cohen's $d_z = +1.67$, $p < 10^{-6}$, paired t-test pooled over the 24 dataset-task pairs).

5.2 The FFN Dominates Feature Construction in PatchTST

Layer 1 raises the linear decodability of the constructed properties (volatility, diff energy, spectral entropy), and Table 2 splits this gain between its two sub-layers. The FFN delta exceeds the attention delta on all 8 datasets: on 6 by at least 2× or with a negligible attention delta, and on ETTm1 and Weather by 1.9× and 1.3× (the 3.2× in Table 2 is the ratio of the 8-dataset means). On ETTh1 the construction itself is weak (Attn $+0.0$, FFN $+3.8$). Per task, attention leads only on diff energy for ETTm1 and Weather (Appendix B.2). Across horizons, $\Delta_{\text{FFN}} \geq \Delta_{\text{attn}}$ holds in all 16 pairs of dataset and horizon (Appendix B.13; Weather at pred_len $= 720$ is a near tie). The three targets added in §3.2 agree: their FFN deltas exceed their attention deltas on 6 or 7 of 8 datasets ($+12.0$, $+9.5$, and $+5.8$ against $+4.8$, $+6.4$, and $+2.1$ on average). Weather is the exception on the two ratio targets, and Traffic and ETTh2 (a tie) on long-range consistency (Appendix B.11). A larger FFN delta on long-range consistency does not imply routing by the FFN.

Table 2: PatchTST Layer-1 constructed-task deltas (volatility, diff energy, spectral entropy; 9-run means). ± is the root-sum-square of the 9-run standard deviations at the two probing points of each delta, averaged over the three tasks; the mean averages the per-dataset values and their ± before rounding. "alone": negligible attention delta.

	ETTm1	ETTm2	ETTh1	ETTh2	Weather	Electricity	Exchange	Traffic	**Mean**
Attn Δ	$+13.5$±1.6	$+0.2$±0.5	$+0.0$±0.4	-0.6±0.9	$+16.7$±0.7	$+11.3$±0.5	$+0.1$±0.6	$+9.5$±0.4	$+6.4$±0.7
FFN Δ	$+\mathbf{26.1}$±1.7	$+\mathbf{43.0}$±0.7	$+\mathbf{3.8}$±1.8	$+\mathbf{11.1}$±2.2	$+\mathbf{21.5}$±1.8	$+\mathbf{23.7}$±0.8	$+\mathbf{12.8}$±2.4	$+\mathbf{20.8}$±0.6	$+\mathbf{20.4}$±1.5
Ratio	1.9×	alone	alone	alone	1.3×	2.1×	alone	2.2×	**3.2**×

Why the FFN. This allocation fits what each sub-layer computes on a patch token; the account is an argument, not a derivation (§8). The FFN, a GELU MLP applied to each token's own representation, can compute nonlinear statistics of the 16-point patch such as its variance, whereas attention mixes

linearly projected tokens with content-dependent weights and, for a fixed attention pattern, is linear in token content [Elhage et al., 2021].

Forecasting side (activation patching). We restore clean Layer-1 component outputs into a run whose input has its 16-step chunks shuffled in time and measure the forecasting MSE recovered [Meng et al., 2022], a denoising test of whether a component's clean output suffices to restore the forecast [Heimersheim and Nanda, 2024] (Appendix G). The test agrees with probing on whether attention contributes at all: restoring attention recovers at most 0.022 MSE on the 4 datasets where probing puts Δ_{attn} near zero (ETTh1, ETTh2, ETTm2, Exchange) and 0.036 to 0.229 on the other 4. Restoring the FFN recovers more on 7 of 8 datasets (all but Weather, where it leads only from Layer 2 on; Appendix G.3), but the instrument favors it, since a restored FFN output is computed from the clean attention output as well; only attention's recovery is informative.

5.3 The FFN Constructs Without Attention

We next ask whether the FFN needs attention for this construction, using the 2 NoAttn models of §4: attention is zeroed either in a trained checkpoint whose FFN weights are kept (*ablated*) or throughout training (*from scratch*). Table 3 summarizes both; per-dataset values are in Appendices B.6 and H.

Table 3: Removing attention from PatchTST: Layer-1 constructed-task deltas (pp) and forecasting MSE, 8-dataset means. Total $= \Delta_{\text{attn}} + \Delta_{\text{FFN}}$ before rounding; MSE change is relative to PatchTST's mean MSE. †Zero by construction (`1a_attn` is then an affine map of the embedding; per-dataset $|\Delta| \leq 0.7$): a sanity check of the probing pipeline, not a result.

Model	Attn Δ	FFN Δ	Total	MSE	MSE change
PatchTST	$+6.4$	$+20.4$	$+26.7$	0.2305	
NoAttn, ablated	$0^{\dagger}$	$+27.4$	$+27.4$	0.2636	$+14.4\%$
NoAttn, from scratch	$0^{\dagger}$	$+28.8$	$+28.8$	0.2352	$+2.0\%$

Both removals leave the probing total nearly unchanged ($+26.7$ with attention, $+27.4$ ablated, $+28.8$ from scratch): without attention, the FFN alone makes the constructed properties as linearly decodable as attention and the FFN do together. In the ablated model, the same FFN takes over attention's share: its delta barely changes where attention added almost nothing and rises by about attention's former delta elsewhere, so no dataset's total moves by more than 3 pp. Trained from scratch, the FFN alone reaches a similar total on every dataset (-0.8 to $+5.7$ pp relative to PatchTST), so the FFN does not need attention during training either.

Reliance is not construction. In forecasting, the two removals differ: ablation (attention knockout) raises MSE on all 8 datasets ($+14.4\%$ on the mean MSE), whereas the from-scratch model comes within 2.0% of PatchTST (at most 5.4% on any dataset). Because ablation leaves the probed construction intact, its cost shows that the frozen model relies on attention for something the suite does not probe: on ETTm2, attention adds almost nothing in probing ($\Delta_{\text{attn}} = +0.2$), yet ablating it costs $+11.4\%$ MSE. The cost is consistent with a train/inference shift, since the FFN and downstream layers were trained on inputs that included attention outputs (cf. Li and Janson, 2024).

Sufficiency, not necessity. Retraining removes most of this cost, whether by removing the shift or by recovering attention's contribution in other ways. At the component level, then, the FFN suffices: the retrained network reaches the PatchTST probing total and forecasts within a few percent of PatchTST, with the FFN as its only nonlinear transformation of token content (RevIN acts before patching, and BatchNorm is affine at inference) and with tokens interacting only through the linear head. Whether the FFN is necessary (training without it) is not tested here.

5.4 Under Single-Scalar Tokens, the FFN Adds Little

We now keep PatchTST's architecture and change its tokens. At $P{=}1$ (§4), each token is a linear projection of one scalar, and the 16-point tasks are read through the mean adapter of §3.3; per-dataset values are in Appendix B.7. The FFN's construction then nearly vanishes: the 8-dataset deltas shift from (Attn $+6.4$, FFN $+20.4$) at $P{=}16$ to (Attn $+10.9$, FFN $+2.3$) at $P{=}1$, a fall of 89% in the FFN delta, and attention leads on 6 of 8 datasets, the exceptions being ETTh2 (FFN slightly ahead) and Exchange (flat). Deeper layers do not recover it: summed over the three layers, the FFN delta is

+1.2 at P=1 against +21.0 at P=16. The total also halves on average (+26.7 to +13.2 pp), though it grows on ETTh1 and ETTh2.

Patch-length sweep. One more point per token restores the FFN's construction. The sweep of §4 computes the tasks on each token's own P points, so it asks whether the FFN constructs these statistics whenever a token holds the points they depend on, not how a fixed task changes with P (at P=2, volatility and diff energy both become quartiles of the boundary-type $|x_2 - x_1|$; Appendix B.12). At P=2 the FFN delta is already +13.0 to +31.4, against −0.9 to +3.4 for the 16-point tasks at P=1, and for $P \geq 2$ it exceeds the attention delta in 18 of the 20 pairs of dataset and patch length (including P=16); the exceptions are Weather at P=2 (a near tie) and at P=8. The strength varies by dataset rather than growing with P (on ETTh1 the FFN delta is only +3.8 at P=16 and +0.5 at P=32), and non-overlapping patches (P = stride = 16) reproduce the P=16 deltas within 3 pp.

What the tokenization explains. The FFN computes statistics of its own token (§5.2), and a token needs at least two points to have any, so at P=1 only attention can bring several time points into a token. It can do so: a sufficiently wide multi-head attention layer can in principle stack lagged values into each token [Wu et al., 2026, Lemma 3.2], and the FFN could then compute statistics of what attention gathered. The argument thus does not predict which component leads at P=1. Empirically, in PatchTST, attention leads and the FFN adds little, and this is not an artifact of averaging: concatenating the 16 token representations instead of averaging them gives the same pattern on the 4 sweep datasets (Appendix B.7).

Forecasting side (knockout and activation patching). The shift toward attention is representational: the frozen P=1 model still relies more on the FFN. Knocking out the FFN costs more than knocking out attention on all 4 datasets tested (ETTh1, ETTm1, Weather, Exchange; e.g., ETTh1 +41.4% vs. +5.8%), as it does at P=16 (Appendix H). Layer-1 activation patching on ETTh1 and ETTm1 also restores more through the FFN, although attention's share rises relative to P=16 (Appendix G.4). Activation patching is biased toward the FFN (§5.2), and knockout measures reliance, not construction (§5.3), so neither instrument can confirm or refute the shift seen in probing.

Informer (auxiliary) is flat (Attn +0.85, FFN +1.47), but its tokens are not single scalars (§4), so the result is confounded (Appendix B.9).

5.5 Component Roles Across Tokenizations and Architectures

Figure 3 compares PatchTST, iTransformer, CATS, and PatchTST (P=1). In iTransformer, whose tokens are whole variates, $\Delta_{\text{attn}} \geq \Delta_{\text{FFN}}$ on all 8 datasets (a 2.5× ratio of the 8-dataset means), with near ties on Exchange (+3.6 vs. +3.3) and Weather (+13.9 vs. +12.2; Appendix B.4). In CATS, cross-attention is the only path from the input to the learnable queries, so its dominance is expected by construction (Appendix B.5). Both are representational results: in iTransformer, attention's gain in linear decodability does not show that attention is necessary for forecasting.

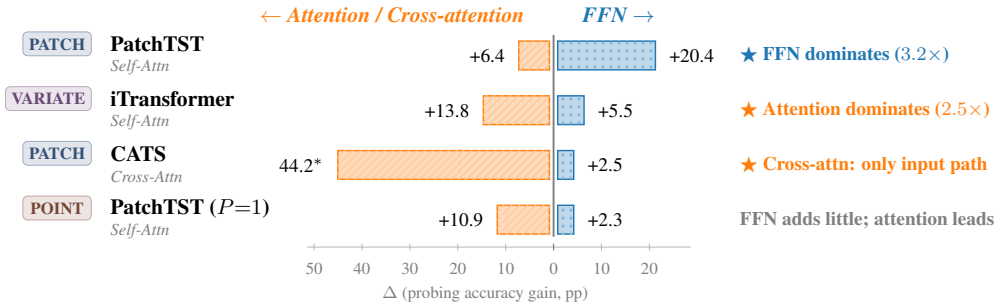


Figure 3: **Component dominance varies with tokenization and architecture.** Layer-1 constructed-task deltas (volatility, diff energy, spectral entropy; 8-dataset means, 9 runs) for 4 settings, tagged by tokenization. *CATS cross-attention is reported as absolute accuracy, not as a delta, because the learnable queries start without per-instance input information; its bar is thus not on the delta scale. Its labels are those of the future patches the queries forecast.

Forecasting side (component knockout). Activation patching restores mostly through the FFN in iTransformer as well, which we attribute to the instrument (Appendix G.5), so across architectures we

compare the MSE cost of component knockout on all 8 datasets (Appendix H). The ordering across architectures matches probing: knocking out the FFN raises MSE by 35.1% in PatchTST, 14.0% in iTransformer, and 6.1% in CATS, and the ratio of the mean attention cost to the mean FFN cost rises from 0.45 to 1.32 to 21.9; both orderings hold when any one dataset is left out. Since knocking out cross-attention removes CATS's only input path, the informative comparison is PatchTST against iTransformer (0.45 against 1.32). Per dataset the agreement is looser: PatchTST relies more on attention on 2 datasets and iTransformer more on the FFN on 4. Knockout measures what a frozen model relies on, not which component constructs (ETTm2 in §5.3), and within PatchTST it does not track the shift at $P{=}1$ (§5.4).

A common pattern. Patch tokens with self-attention place construction in the FFN, variate tokens place it in attention, the cross-attention path carries it in CATS, and under single-scalar tokens, attention (the first operator to combine time points) leads while the FFN adds little. We read this as a regularity across the evaluated models rather than a mechanism. Within PatchTST, tokenization can be varied separately from the presence of a token mixer (the sub-layer that mixes tokens, here self-attention), and their effects on the total differ. Changing only the tokenization halves the total ($P{=}16 \to 1$, which also changes the read-out), whereas removing only the mixer leaves it nearly unchanged (§5.3, §5.4); we did not vary the form of the mixer. Tokenization and what attention mixes cannot be separated, because switching from patch to variate tokens also switches attention from mixing time to mixing variates; nor is PatchTST versus iTransformer a controlled comparison (input length 336 vs. 96; channel independence vs. dependence, Han et al., 2024; scope in §8).

6 A Representational Reading of Prior Results

Our reading of the four findings of §1 is qualitative, not a causal account of their MSE orderings. **(1) DLinear $>$ early transformers** [Zeng et al., 2023]. Like our $P{=}1$ tokens, their tokens cover one time step each, and at $P{=}1$ the FFN adds little (§5.4). Yet their tokens also mix all variates and neighboring steps, as Informer's do (§4), so the $P{=}1$ result applies to them at most partially. **(2) PatchTST $>$ DLinear** [Nie et al., 2023]. A linear map of the raw window, such as DLinear, reads only linear functionals of it, whereas patching gives multi-point tokens to PatchTST's FFN, which makes within-patch statistics linearly decodable (§5.2). **(3) Self-attention-free patch models** [TSMixer, PatchMLP; Ekambaram et al., 2023, Tang and Zhang, 2025]. At $P{=}16$, attention is not needed for the probed construction, and retraining without it costs little (§5.3), a component-level result consistent with the forecasting accuracy of these models. Probing one such model, ModernTCN [Luo and Wang, 2024], gives the same allocation: summed over its three blocks on ETTm1, the ConvFFNs, not the convolution mixer, carry the construction (Appendix D). **(4) Cross-attention alone** [CATS; Kim et al., 2024]. CATS's learnable queries start without per-instance input information, so cross-attention, their only path from the input, supplies it by construction, and the FFN adds little (§5.5). Unlike the self-attention dropped in (3), which mixes tokens that already hold the input, this cross-attention is the input path itself. The same name, "attention," thus covers different information flows: a removable mixer under patch tokens (3), the only input path in CATS (4), and, in probing, the main constructing component in iTransformer, whose tokens are whole variates (§5.5).

7 Design Implications

Our results suggest three practices.

(i) Diagnose before re-engineering. Measure the Layer-1 attention and FFN deltas on the constructed tasks with a random-label control; Layer 1 suffices in PatchTST, whose deeper layers add little to the FFN delta (Table 18). A large FFN delta indicates that the tokens already hold the points the FFN needs, so attention is a candidate for cheaper mixing. A small one calls for trying another patch length on the same data: if construction appears there, the tokenization is the bottleneck.

(ii) Choose the token before the mixer. In PatchTST, shrinking tokens to single scalars halves the probing total, whereas removing the mixer leaves it nearly unchanged (§5.5). One extra point per token restores the FFN's construction, and longer patches need not add to it (§5.4). Under variate tokens, attention leads (iTransformer, §5.5), so the same mixer plays opposite roles.

(iii) Under patch tokenization, self-attention is replaceable. At $P{=}16$ even the lightest replacement, no mixer at all, suffices: PatchTST trained without attention keeps its probing total, with the

FFN taking over attention's share, and stays within 2.0% of the full model in mean MSE (Table 3). This gives a component-level basis for the forecasting accuracy of TSMixer and PatchMLP (§6) and suggests spending the design budget on the tokenization and the FFN rather than on attention variants. For variate tokens we only predict the opposite: since attention leads in probing (§5.5), the same replacement would remove the constructing component rather than a redundant mixer.

8 Limitations and Future Directions

Reach of the evidence. A delta locates where a property becomes linearly decodable, not whether the forecaster uses it [Elazar et al., 2021]. In PatchTST at $P{=}16$, Layer-1 activation patching, training without attention, and knockout establish component-level reliance and sufficiency on the forecasting side (§5.2, §5.3, Appendix H). At $P{=}1$, activation patching and knockout point to the FFN but can neither confirm nor refute the shift seen in probing (§5.4). Across architectures, knockout supports only the ordering, so the construction claims for iTransformer and CATS stay representational (§5.5); Informer is only probed (Appendix B.9). Below the component level, whether an individual probed property is used remains open: erasing one with iterative nullspace projection (INLP; Ravfogel et al., 2020) removed 33 to 94% of the 128 dimensions and was inconclusive against a random-label control (Appendix I).

Scope. Within PatchTST at $P{\geq}2$, the strength of construction varies with dataset and patch length (ETTh1 is weak at $P{=}16$ and 32), but the allocation does not: it holds across patch lengths, strides, horizons, training without attention, both per-dataset widths (d_{model} of 16 and 128), and three added targets, with Weather the recurring exception. Single-scalar and variate tokens lie outside this regime. These checks narrow but do not remove the generalization concern: we did not vary lookback, depth, or the form of the mixer, and a sweep with 16-point tasks fixed across P is future work.

Task suite. The tasks are hand-picked to make construction visible, and the main conclusions rest on three second-order statistics of a token's segment (16 points in PatchTST), so the allocation may differ for other properties. The suite omits higher-order moments, multi-resolution structure beyond a two-scale split, dependence beyond eight tokens, and cross-token multi-scale structure.

Where the framework applies. The framework requires (i) an input embedding that is an affine map of each token's values, so that no construction happens inside it, and (ii) a linear head that reads the probed representations. Informer is a boundary case on both: its embedding mixes neighboring steps and variates, and its head reads decoder tokens. Any model meeting both is covered, including pretrained time series foundation models that do; we leave these to future work.

No formal theory. We establish the phenomenon and its boundary conditions experimentally but do not prove why they hold; a formal account is the next step. It should explain why the allocation shifts between single-scalar and multi-point tokens, and why, under patch tokens, the FFN alone reaches the full total when trained without attention. So far we have only an argument for patch tokens (§5.2), a single-scalar analysis valid only before attention (§5.4), and nothing for variate tokens.

9 Conclusion

In time series transformers, the sub-layer that makes temporal properties linearly decodable is allocated by the interaction of tokenization and architecture: the FFN under patch tokens (PatchTST), attention under variate tokens (iTransformer, in probing), the cross-attention path in CATS, and attention under single-scalar tokens (in probing), where the FFN adds little. This reading accounts qualitatively for why patching helps and self-attention-free models compete, and suggests probing first, choosing the token before the mixer, and replacing self-attention only under patch tokens. Our forecasting-side interventions establish component-level reliance and sufficiency, not where individual features are causally constructed. Feature-level tests, pretrained foundation models, and a formal account are the next steps.

Acknowledgments and Disclosure of Funding

This work was supported by Inha University and by Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) under the Leading Generative AI Human Resources Development program (IITP-2026-RS-2024-00360227) and the Artificial Intelligence Innovation Human Resources Development program (IITP-2026-RS-2026-25548560).

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Anthony Bao, Venkata Hasith Vattikuti, Jeffrey Lai, and William Gilpin. Universal redundancies in time series foundation models. In *ICML*, 2026.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *NeurIPS*, 2023.

Lei Chen, Joan Bruna, and Alberto Bietti. Distributional associations vs in-context reasoning: A study of feed-forward and attention layers. In *ICLR*, 2025.

Yu Chen, Nathalia Céspedes, and Payam Barnaghi. A closer look at transformers for time series forecasting: Understanding why they work and where they struggle. In *ICML*, 2025.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*, pages 2126–2136, 2018.

Yihe Dong, Lorenzo Noci, Mikhail Khodak, and Mufan Li. Is random attention sufficient for sequence modeling? Disentangling trainable components in the transformer. *arXiv preprint arXiv:2506.01115*, 2025.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. TSMixer: Lightweight MLP-Mixer model for multivariate time series forecasting. In *KDD*, pages 459–469, 2023.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL `https://transformer-circuits.pub/2021/framework/index.html`.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP*, pages 5484–5495, 2021.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*, pages 30–45, 2022.

Suhan Guo, Jiahong Deng, Yi Wei, Hui Dou, Furao Shen, and Jian Zhao. RAM: Replace attention with MLP for efficient multivariate time series forecasting. *arXiv preprint arXiv:2410.24023*, 2024.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):7129–7142, 2024.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *EMNLP*, pages 2733–2743, 2019.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *NAACL-HLT*, pages 4129–4138, 2019.

John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher D. Manning. Conditional probing: Measuring usable information beyond a baseline. In *EMNLP*, pages 1626–1639, 2021.

Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. SAMformer: Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention. In *ICML*, 2024.

Ilya Kaufman and Omri Azencot. Analyzing deep transformer models for time series forecasting via manifold learning. *Transactions on Machine Learning Research*, 2024.

Dongbin Kim, Jinseong Park, Jaewook Lee, and Hoki Kim. Are self-attentions effective for time series forecasting? In *NeurIPS*, 2024.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *ICLR*, 2022.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR*, pages 95–104, 2018.

Maximilian Li and Lucas Janson. Optimal ablation for interpretability. In *NeurIPS*, 2024.

Zida Liang, Jiayi Zhu, and Weiqiang Sun. Why attention fails: The degeneration of transformers into MLPs in time series forecasting. *arXiv preprint arXiv:2509.20942*, 2025.

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 *ICLR*, 2024.

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *ICLR*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *NeurIPS*, 2022.

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

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL `https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html`.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In *ACL*, pages 4609–4622, 2020.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *ACL*, pages 7237–7256, 2020.

Peiwang Tang and Weitai Zhang. Unlocking the power of patch: Patch-based MLP for long-term time series forecasting. In *AAAI*, 2025.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *ACL*, pages 4593–4601, 2019.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In *NeurIPS*, 2021.

Angela van Sprang, Erman Acar, and Willem Zuidema. Interpretability for time series transformers using a concept bottleneck framework. *arXiv preprint arXiv:2410.06070*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, pages 5797–5808, 2019.

Chengxin Wang, Yiran Zhao, Shaofeng Cai, and Gary Tan. Investigating pattern neurons in urban time series forecasting. In *ICLR*, 2025.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *ICLR*, 2023.

Xiaorui Wang, Fanda Fan, Chenxi Wang, Yuxuan Yang, Rui Tang, Kuoyu Gao, Simiao Pang, Yuanfeng Shang, Zhipeng Liu, Wanling Gao, Lei Wang, and Jianfeng Zhan. CombinationTS: A modular framework for understanding time-series forecasting models. In *ICML*, 2026.

Michał Wiliński, Mononito Goswami, Willa Potosnak, Nina Żukowska, and Artur Dubrawski. Exploring representations and interventions in time series foundation models. In *ICML*, pages 66861–66886, 2025.

Dennis Wu, Yihan He, Yuan Cao, Jianqing Fan, and Han Liu. Understanding transformers for time series forecasting: A case study on Moirai. In *ICLR*, 2026.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *ICLR*, 2020.

Alper Yıldırım. Superposition is not necessary: A mechanistic interpretability analysis of transformer representations for time series forecasting. *arXiv preprint arXiv:2605.05151*, 2026.

Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Yuyang Wang. Understanding transformers for time series: Rank structure, flow-of-ranks, and compressibility. In *ICLR*, 2026.

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

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *ICLR*, 2024.

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*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

A Model Hyperparameters, Datasets, Probe Protocol, and Compute

Table 4: Model coverage matrix: each cell indicates the model analyzed for the corresponding (tokenization, attention type) combination. PatchTST-NoAttn sets the self-attention output to zero, in a trained checkpoint (ablated) or from initialization (from scratch), so that the sub-layer is a residual identity and everything else, including BatchNorm, the FFN, and the per-dataset hyperparameters, is unchanged. Informer (4 datasets only) is a separate auxiliary case: its time-step tokens are embedded by a three-step convolution over all variates, so it is not a point-level model, and its probe is confounded (Appendix B.9).

	Self-Attention	Cross-Attention	No Attention
Point (P=1)	PatchTST (P=1)	—	—
Time step (aux.)	Informer	—	—
Patch	PatchTST (P=16)	CATS	PatchTST-NoAttn (P=16)
Variate (whole series)	iTransformer	—	—

Table 5: Architecture-level configurations across the analyzed models (input length 336 for the PatchTST family and 96 for the other models).

Model	Tokenization	n_{layers}	patch_len	stride
PatchTST (P=16)	Patch	3	16	8
PatchTST (P=1)	Point	3	1	1
PatchTST-NoAttn	Patch (attn output $= 0$)	3	16	8
iTransformer	Variate (whole series)	varies (2–4)	—	—
CATS	Patch (cross-attn only)	0 enc / 3 dec	24 or 48	24 or 48
Informer	Time step (conv k=3; ProbSparse)	2 enc / 1 dec	—	—

PatchTST (and variants). PatchTST splits each univariate time series into 16-point patches with stride 8 and applies a vanilla 3-layer Transformer encoder per channel (channel-independent). Architectural width is determined by dataset scale (Table 6); for datasets without original-paper specifications (Exchange, Solar), we use the configuration of the most similar dataset in scale: Exchange is grouped with ETTh1/ETTh2 (small), and Solar is grouped with Weather/Electricity/Traffic (large). PatchTST (P=1) sets `patch_len` = `stride` = 1 with all other hyperparameters unchanged.

Table 6: PatchTST per-dataset architectural configurations. Two width settings (small vs. large) are used.

Datasets	d_{model}	d_{ff}	n_{heads}
ETTh1, ETTh2, Exchange	16	128	4
ETTm1, ETTm2, Weather, Electricity, Solar, Traffic	128	256	16

iTransformer. Each variate's whole lookback window becomes a single token; attention operates across variates. Width and depth follow the official per-dataset scripts (Table 7).

Table 7: iTransformer per-dataset architectural configurations. $n_{\text{heads}} = 8$ across all datasets; no patching.

Datasets	d_{model}	n_{layers}	d_{ff}
ETTh1	256	2	256
ETTh2, ETTm1, ETTm2, Exchange	128	2	128
Solar	512	2	512
Weather, Electricity	512	3	512
Traffic	512	4	512

CATS. The encoder is removed ($e_{\text{layers}} = 0$); only the cross-attention decoder is used. Width and depth are identical across datasets ($d_{\text{model}} = 256$, $n_{\text{heads}} = 32$, $d_{\text{layers}} = 3$, $d_{\text{ff}} = 512$), and `patch_len=stride=` 48 for ETTh1/ETTh2 and 24 for the remaining datasets. Following the official scripts, the query-independence option (one set of queries shared by all variates; on for ETTm2, ETTh2, and Traffic), the batch size, the number of epochs, and the learning-rate schedule also vary by dataset; Exchange, which the scripts do not cover, uses the ETTm1 settings.

Informer. A single architectural setting is used across all datasets: $d_{\text{model}} = 512$, $n_{\text{heads}} = 8$, $e_{\text{layers}} = 2$, $d_{\text{layers}} = 1$, $d_{\text{ff}} = 2048$, ProbSparse factor $= 3$; no patching. Training uses label_len $= 48$, batch size 128, learning rate 10^{-4}, and 10 epochs (2 for Weather).

Table 8: Dataset information. The splits follow the standard PatchTST protocol; Train/Val/Test sizes and Domain are as listed by Liu et al. [2024]. Solar-Energy is excluded from the main analysis (Appendix C).

Dataset	Variables	Train/Val/Test	Frequency	Domain
ETTm1	7	(34465, 11521, 11521)	15 min	Electricity
ETTm2	7	(34465, 11521, 11521)	15 min	Electricity
ETTh1	7	(8545, 2881, 2881)	1 hour	Electricity
ETTh2	7	(8545, 2881, 2881)	1 hour	Electricity
Weather	21	(36792, 5271, 10540)	10 min	Weather
Electricity	321	(18317, 2633, 5261)	1 hour	Electricity
Exchange	8	(5120, 665, 1422)	Daily	Economy
Traffic	862	(12185, 1757, 3509)	1 hour	Transportation
Solar-Energy	137	(36601, 5161, 10417)	10 min	Energy

Forecaster training. The models of Table 4 are trained with each model's official code and scripts, which standardize each dataset with the statistics of its training split; datasets the scripts do not cover use the settings of the most similar dataset, as above, and Informer is trained with the implementation in the PatchTST code base. Averaged over the 3 seeds, the test MSE of PatchTST [Nie et al., 2023], iTransformer [Liu et al., 2024], and CATS [Kim et al., 2024] is within 3.7% of the published result at the same input length on each of our 8 datasets that these papers report (PatchTST and CATS do not report Exchange).

Probe samples. Probes are trained and evaluated on frozen representations of each dataset's test split, on which the forecasters are not trained. The split is read in order from its start, as the standard sliding windows that each start one time step after the previous one. Each window gives one token per variate and position (one per variate in iTransformer), and the tokens of all windows, variates, and positions form one pooled sample set per dataset, model seed, and probing point. Collection stops at a per-model budget. The PatchTST family uses 5,000 univariate input windows, that is, 210,000 tokens at P=16 and 205,000 mean-adapter samples at P=1. iTransformer uses whole batches of up to 32 windows until at least 50,000 variate tokens are collected (or the whole split, if it holds fewer), excluding the time-feature tokens that iTransformer appends. CATS uses full test batches, and Informer uses up to 250,000 pooled samples (Appendix B.9).

Labels. A token's labels are computed with the formulas of Table 1 from its own time points, except that those of the inter-token tasks, reg_next_mean, and local_global_trend depend on its position or on other tokens of its window (§3.2); §3.3 gives, for each model, the time points from which a token's labels are computed. In spectral entropy, q_k is the share of frequency bin k in the power of the token's real FFT, zero frequency included. In PatchTST, dominant_period's four bands are equal groups of consecutive bins starting at zero frequency (two bins each at P=16, with the Nyquist bin left out). The lowest band thus holds the zero-frequency bin, and it is the majority class on each of the 8 datasets (84.93 to 98.55% of tokens, the majority column of Table 9). Class boundaries (the quartiles, the 70th percentile for change_point, and, for patch tokens, the median for adjacent_similarity) are computed once over the pooled set, so all probing points and both parts of the probe's 80/20 split (below) share them.

Probe training. One probe is trained for each dataset, model seed, probe seed, probing point, and task. It maps one token representation $h \in \mathbb{R}^{d_{\text{model}}}$ to C class logits, $Wh + b$ with $W \in \mathbb{R}^{C \times d_{\text{model}}}$ (one output for regression; $16\, d_{\text{model}}$ inputs with the concatenation adapter of Appendix B.7). The pooled set is split at random into 80% for training and 20% for evaluation. Each of the 30k Adam steps (lr $= 10^{-3}$, no weight decay) uses 32,768 training samples (or as many as the training part holds, if fewer), drawn uniformly with replacement, and minimizes the cross-entropy, or the mean squared error for regression. Every 500 steps the probe is scored on the evaluation part (accuracy, or R^2 for regression), and the best score over training is reported. The probe seed sets the split, the initialization, and the batch draws.

Compute setup. Experiments ran on NVIDIA A100 (40GB), A40, and RTX A6000 GPUs.

B Full Probing Results

Appendix tables abbreviate trend_direction as trend, dominant_period as dom_p, and volatility, diff_energy, and spectral_entropy as Vol, DE, and SE; the subscript $q4$ marks quartile labels. A probing accuracy's $\pm$ is the population standard deviation of its 9 runs, and the $\pm$ of a probing delta combines these standard deviations s_a and s_b of its two probing points as $\sqrt{s_a^2 + s_b^2}$, as in Table 2; for a mean over tasks or datasets, these $\pm$ values are averaged.

B.1 PatchTST Embed-Layer Probing (Numerical Values for Figure 2)

Table 9: PatchTST embed-layer probing accuracy on intra-token tasks (9-run mean). trend is innate, and Vol, DE, and SE are constructed; dom_p is degenerate (§3.2): its probing accuracy is within 0.12 pp of the majority rate of its labels (shown to two decimals), the accuracy of always predicting the most common band. Figure 2 plots trend, Vol, DE, and SE.

		dom_p				
Dataset	trend	probe	majority	Vol	DE	SE
ETTm1	99.6	96.2	96.25	38.9	42.1	30.2
ETTm2	99.3	90.0	89.88	36.9	34.8	40.9
ETTh1	97.9	92.3	92.36	37.6	38.7	35.8
ETTh2	95.3	84.9	84.93	33.7	27.7	30.7
Weather	98.5	97.8	97.82	33.6	33.8	36.2
Electricity	99.6	94.2	94.26	41.6	38.4	41.6
Exchange	94.9	98.5	98.55	31.1	31.2	31.1
Traffic	99.9	95.0	94.91	53.7	54.9	45.7

B.2 PatchTST Full Results

Table 10 gives the per-task Layer-1 deltas on all 8 datasets, and Table 11 gives all tasks at every probing point on ETTm1 as an example; the released result files hold these values for every dataset.

Table 10: PatchTST per-task Layer-1 delta (9-run mean $\pm$ std). Averaging the three tasks gives Table 2. The FFN delta exceeds the attention delta for every task on every dataset except diff_energy on ETTm1 and Weather.

Dataset	Vol Attn Δ	Vol FFN Δ	DE Attn Δ	DE FFN Δ	SE Attn Δ	SE FFN Δ
ETTm1	$+10.3 \pm 2.1$	$+34.5 \pm 2.3$	$+13.5 \pm 1.6$	$+10.6 \pm 1.9$	$+16.8 \pm 1.2$	$+33.2 \pm 0.9$
ETTm2	-0.5 ± 0.4	$+52.1 \pm 0.6$	$+4.1 \pm 0.4$	$+37.7 \pm 0.6$	-2.9 ± 0.8	$+39.2 \pm 0.8$
ETTh1	-0.1 ± 0.4	$+2.9 \pm 1.6$	-0.2 ± 0.3	$+3.1 \pm 1.9$	$+0.3 \pm 0.4$	$+5.4 \pm 1.9$
ETTh2	-2.6 ± 1.1	$+11.1 \pm 3.3$	$+0.9 \pm 1.1$	$+14.3 \pm 1.3$	-0.1 ± 0.6	$+7.8 \pm 1.9$
Weather	$+17.6 \pm 0.4$	$+24.0 \pm 1.8$	$+19.4 \pm 0.6$	$+14.8 \pm 1.5$	$+13.2 \pm 1.0$	$+25.8 \pm 2.1$
Electricity	$+12.5 \pm 0.4$	$+30.5 \pm 0.5$	$+11.7 \pm 0.5$	$+26.4 \pm 1.2$	$+9.8 \pm 0.5$	$+14.3 \pm 0.6$
Exchange	$+0.2 \pm 0.4$	$+12.0 \pm 2.3$	$+0.7 \pm 0.5$	$+6.8 \pm 1.5$	-0.5 ± 0.8	$+19.6 \pm 3.5$
Traffic	$+11.5 \pm 0.3$	$+26.5 \pm 0.6$	$+10.9 \pm 0.5$	$+20.0 \pm 0.6$	$+6.2 \pm 0.4$	$+15.8 \pm 0.6$

Table 11: PatchTST full layer probing results on ETTm1 (9-run mean ± std). All 15 diagnostic tasks plus the random-label control. On the constructed tasks the Layer-1 FFN gives the largest rise for volatility and spectral_entropy, and attention for diff_energy; after Layer 1, no sub-layer changes their accuracy by more than 1.9 pp. Table 18 gives, for all 8 datasets, the FFN deltas summed over the three layers.

Task	embed	1a_attn	1b_ffn	2a_attn	2b_ffn	3a_attn	3b_ffn
Innate and degenerate (classification)							
trend_direction	99.6±0.0	99.6±0.0	99.4±0.1	99.2±0.2	98.9±0.2	98.6±0.1	98.5±0.1
dominant_period	96.2±0.1	96.2±0.1	96.9±0.2	96.8±0.1	96.9±0.1	96.8±0.1	96.8±0.1
Constructed (classification)							
volatility$_{q4}$	38.9±0.4	49.2±2.1	83.7±1.0	83.4±1.4	83.8±0.8	83.2±1.2	82.5±0.9
diff_energy$_{q4}$	42.1±0.4	55.6±1.5	66.2±1.1	68.1±0.6	68.2±0.8	68.3±0.7	67.7±1.0
spectral_entropy$_{q4}$	30.2±1.0	47.0±0.6	80.2±0.7	79.5±1.0	80.4±0.7	79.7±0.7	78.8±0.9
Boundary tasks and local_global_trend (classification)							
slope_magnitude$_{q4}$	33.4±0.0	40.2±0.8	80.5±1.8	77.6±1.9	79.1±1.9	76.2±2.3	76.5±1.6
change_point	70.0±0.2	74.5±1.1	97.2±0.5	96.8±0.5	97.2±0.5	96.7±0.5	96.8±0.3
local_global_trend	60.1±0.2	59.9±0.3	60.6±0.6	59.8±0.6	60.2±0.9	58.8±0.7	59.1±0.4
Inter-token (classification)							
adjacent_similarity	54.5±0.2	72.0±0.6	87.1±0.2	86.7±0.2	86.9±0.2	86.7±0.5	86.7±0.4
relative_magnitude	99.5±0.1	99.8±0.0	99.8±0.0	99.8±0.0	99.7±0.0	99.7±0.0	99.6±0.0
position_quarter	100.0±0.0	99.9±0.0	95.5±1.2	88.3±1.9	82.4±2.3	74.3±1.7	69.5±1.6
trend_consistency	61.5±0.1	65.5±0.5	72.5±0.5	73.1±0.3	73.2±0.3	72.9±0.5	73.0±0.4
Regression (R^2)							
reg_variance	0.009±0.000	0.253±0.041	0.961±0.005	0.949±0.009	0.951±0.005	0.941±0.009	0.944±0.006
reg_slope	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	0.999±0.000	0.998±0.000	0.998±0.000
reg_next_mean	0.923±0.001	0.933±0.001	0.933±0.001	0.934±0.001	0.934±0.001	0.935±0.001	0.935±0.001
Control							
random_label	25.3±0.2	25.3±0.1	25.3±0.1	25.3±0.1	25.3±0.1	25.3±0.1	25.2±0.0

B.3 PatchTST Regression Results

Table 12 gives reg_variance and reg_slope on all 8 datasets.

Table 12: PatchTST reg_variance R^2 from embed to 1b_ffn (9-run mean ± std); the last column is reg_slope R^2 at the embedding (9-run mean). At 1b_ffn, R^2 stays low on ETTh1, ETTh2, and Exchange (0.175, 0.310, and 0.251) and is at least 0.890 on the other 5 datasets.

Dataset	embed	1b_ffn	1b_ffn − embed	reg_slope (embed)
ETTm1	0.009±0.000	0.961±0.005	+0.952	1.000
ETTm2	0.006±0.003	0.890±0.013	+0.884	1.000
ETTh1	0.043±0.002	0.175±0.072	+0.132	0.996
ETTh2	0.002±0.001	0.310±0.067	+0.308	0.993
Weather	0.023±0.002	0.923±0.005	+0.900	1.000
Electricity	0.152±0.003	0.932±0.003	+0.780	1.000
Exchange	−0.015±0.010	0.251±0.069	+0.265	0.983
Traffic	0.496±0.003	0.943±0.002	+0.446	1.000

B.4 iTransformer Full Results

Labels are computed on each variate's series before the model's per-variate normalization, because after it every series has nearly the same variance and volatility would be nearly constant. A token, a linear map of the normalized series, therefore lacks the series' mean and standard deviation, on which the constructed-task labels depend. Volatility is the variance itself, diff_energy scales with it, and spectral_entropy, whose spectrum includes the zero-frequency bin, changes with the ratio of the mean to the standard deviation; trend_direction is unaffected, since the normalization does not change its sign. Part of a gain on these tasks can therefore come from other tokens (other variates and the appended time-feature tokens), which only attention reads. Table 13 gives the per-task deltas.

Table 13: iTransformer per-task Layer-1 delta (9-run mean ± std). Avg: mean of the three tasks; Mean: mean of the 8 datasets.

Dataset	Vol Attn Δ	Vol FFN Δ	DE Attn Δ	DE FFN Δ	SE Attn Δ	SE FFN Δ	Avg Attn Δ	Avg FFN Δ
ETTm1	+15.5±1.5	+5.0±2.1	+13.9±0.8	+3.8±0.9	+10.5±1.0	+6.6±1.9	+13.3±1.1	+5.1±1.6
ETTm2	+5.7±0.7	+1.9±0.6	+8.8±2.6	+4.2±0.9	+14.1±2.9	+1.5±1.7	+9.5±2.1	+2.5±1.0
ETTh1	+24.7±1.5	+7.5±1.5	+23.8±0.9	+8.8±1.4	+14.6±1.0	+9.3±0.9	+21.0±1.1	+8.5±1.3
ETTh2	+10.4±1.4	+7.1±1.1	+6.2±0.8	+3.4±0.9	+11.9±1.1	+2.4±1.1	+9.5±1.1	+4.3±1.0
Weather	+12.0±0.7	+13.3±0.7	+15.4±1.7	+12.7±0.8	+14.4±0.4	+10.6±0.4	+13.9±0.9	+12.2±0.7
Electricity	+13.5±0.4	+2.6±0.4	+15.6±0.6	+4.0±0.5	+21.9±0.5	+2.9±0.4	+17.0±0.5	+3.2±0.4
Exchange	+2.6±1.2	+4.8±1.4	+3.5±1.2	+2.4±1.2	+4.8±1.1	+2.8±1.1	+3.6±1.2	+3.3±1.2
Traffic	+9.2±0.4	+2.9±0.5	+19.1±0.6	+5.3±0.5	+39.4±0.6	+6.5±0.9	+22.6±0.5	+4.9±0.6
Mean	+11.7±1.0	+5.6±1.1	+13.3±1.1	+5.6±0.9	+16.5±1.1	+5.3±1.0	+13.8±1.1	+5.5±1.0

B.5 CATS Full Results

Table 14: CATS per-task results (9-run mean ± std). Absolute accuracy of the cross-attention output (1a_cross) and FFN delta from 1a_cross to 1b_ffn. Query-side labels are those of the future patch each query forecasts.

Dataset	Vol 1a_cross	Vol FFN Δ	DE 1a_cross	DE FFN Δ	SE 1a_cross	SE FFN Δ
ETTm1	41.8±1.0	+2.2±1.1	48.5±0.5	+1.2±0.7	41.3±0.3	+1.4±0.4
ETTm2	39.3±0.3	+1.2±0.5	46.7±0.3	+1.5±0.7	38.7±0.1	+1.5±0.5
ETTh1	36.5±0.6	+2.5±0.7	46.5±0.4	+2.2±0.5	43.7±0.3	+0.2±0.9
ETTh2	37.0±0.4	+1.9±0.6	38.5±0.7	+8.2±1.1	37.1±0.4	+3.5±1.0
Weather	48.1±0.3	+3.5±0.4	56.9±0.2	+3.8±0.3	45.7±0.1	+3.5±0.2
Electricity	38.2±0.2	+1.4±0.2	51.6±0.2	+1.0±0.3	53.4±0.4	+1.8±0.4
Exchange	38.3±0.6	+2.2±0.7	41.5±1.2	+3.6±1.3	39.6±0.8	+1.6±0.9
Traffic	54.2±0.1	+3.9±0.2	52.1±0.1	+3.7±0.2	45.5±0.1	+3.2±0.1

CATS's learnable queries start without per-instance input information and hold no input points, so their labels are computed on the future patch each query forecasts, normalized with the input window's mean and standard deviation as in the model; the accuracies thus also reflect how predictable these properties are from the input. The absolute accuracy at 1a_cross (cross-attention output) does not isolate cross-attention: the learned queries already encode the position and, without query independence (Appendix A), the variate. The FFN adds $+0.2$ to $+8.2$ pp afterward (Table 14).

B.6 PatchTST-NoAttn Full Results

Table 15: Per-dataset Layer-1 constructed-task deltas without attention (9-run mean ± std). FFN change and total change are differences from PatchTST, where the total is Attn Δ + FFN Δ (NoAttn's Attn Δ is in Table 16). The 8-dataset means are in Table 3.

	PatchTST		NoAttn, ablated			NoAttn, from scratch		
Dataset	Attn Δ	FFN Δ	FFN Δ	FFN change	Total change	FFN Δ	FFN change	Total change
ETTm1	+13.5±1.6	+26.1±1.7	+38.8±1.2	+12.7	−0.8	+44.9±2.4	+18.8	+5.2
ETTm2	+0.2±0.5	+43.0±0.7	+44.2±0.5	+1.2	+0.7	+43.9±0.6	+0.9	+0.3
ETTh1	+0.0±0.4	+3.8±1.8	+4.1±1.9	+0.3	+0.3	+4.1±1.7	+0.3	+0.3
ETTh2	−0.6±0.9	+11.1±2.2	+12.1±2.8	+1.0	+1.4	+9.8±3.9	−1.3	−0.8
Weather	+16.7±0.7	+21.5±1.8	+36.1±2.5	+14.6	−2.2	+40.1±0.9	+18.6	+1.7
Electricity	+11.3±0.5	+23.7±0.8	+38.1±0.5	+14.3	+3.0	+40.6±0.4	+16.9	+5.7
Exchange	+0.1±0.6	+12.8±2.4	+13.9±3.0	+1.1	+0.9	+13.8±4.3	+1.0	+0.8
Traffic	+9.5±0.4	+20.8±0.6	+32.3±0.4	+11.5	+2.1	+33.0±0.5	+12.2	+3.3

Table 16: PatchTST-NoAttn per-task Layer-1 delta (9-run mean ± std), ablated (top) and trained from scratch (bottom). "Attn Δ" is near zero because the attention output is zero.

Dataset	Vol Attn Δ	Vol FFN Δ	DE Attn Δ	DE FFN Δ	SE Attn Δ	SE FFN Δ
Ablated						
ETTm1	+0.1±0.6	+46.2±1.0	−0.6±0.7	+21.0±0.9	+0.6±1.8	+49.1±1.7
ETTm2	+0.0±0.4	+52.8±0.6	−0.4±0.2	+42.6±0.4	−0.4±0.4	+37.3±0.5
ETTh1	+0.0±0.3	+2.9±1.2	+0.0±0.3	+3.5±1.9	+0.0±0.3	+5.8±2.6
ETTh2	−0.2±1.2	+10.6±4.2	−0.2±1.0	+16.4±1.7	−0.1±0.6	+9.2±2.5
Weather	+0.0±0.3	+41.3±2.7	+0.2±0.6	+31.7±1.9	−0.5±0.7	+35.4±2.9
Electricity	+0.0±0.2	+49.6±0.5	+0.1±0.4	+42.0±0.7	+0.0±0.1	+22.6±0.2
Exchange	+0.1±0.4	+12.8±2.6	+0.0±0.4	+8.2±2.0	−0.5±0.6	+20.8±4.2
Traffic	+0.3±0.1	+40.2±0.3	+0.0±0.3	+33.6±0.5	+0.0±0.3	+23.0±0.4
From scratch						
ETTm1	+0.2±0.6	+50.8±1.5	−1.0±0.9	+31.5±4.8	+0.6±0.8	+52.4±1.1
ETTm2	−0.1±0.2	+52.7±0.7	−0.4±0.3	+41.7±0.4	−0.7±0.7	+37.3±0.8
ETTh1	+0.0±0.3	+3.3±1.0	+0.1±0.3	+3.8±1.9	+0.0±0.3	+5.1±2.2
ETTh2	−0.2±1.8	+7.1±5.1	+0.0±1.3	+14.3±3.4	−0.1±1.2	+8.0±3.4
Weather	+0.0±0.3	+45.1±0.9	+0.5±0.6	+36.5±0.7	−1.1±0.7	+38.8±1.1
Electricity	+0.2±0.3	+50.2±0.4	+0.1±0.3	+47.5±0.5	+0.3±0.3	+24.1±0.3
Exchange	+0.1±0.4	+13.6±3.9	+0.1±0.3	+8.7±2.8	−0.4±0.6	+19.2±6.2
Traffic	+0.9±0.6	+40.9±0.4	+0.3±0.4	+34.0±0.5	+0.6±0.4	+24.0±0.6

B.7 PatchTST (P=1) Full Results

Table 17: PatchTST (P=1) with the mean adapter, per-task Layer-1 delta (9-run mean ± std). Avg: mean of the three tasks; Mean: mean of the 8 datasets. The FFN delta is small on every dataset except ETTh2, while the attention delta (mean +10.9 pp) is positive for every task on every dataset but Exchange.

Dataset	Vol Attn Δ	Vol FFN Δ	DE Attn Δ	DE FFN Δ	SE Attn Δ	SE FFN Δ	Avg Attn Δ	Avg FFN Δ
ETTm1	+15.0±1.7	+1.4±3.4	+16.5±2.5	+2.5±3.6	+4.1±2.4	+2.6±3.1	+11.9±2.2	+2.2±3.4
ETTm2	+14.0±0.7	+2.0±0.9	+14.8±1.1	+4.2±1.7	+15.7±1.0	+2.0±1.3	+14.8±1.0	+2.7±1.3
ETTh1	+18.2±3.3	−3.4±3.5	+14.3±2.3	−3.4±2.4	+3.5±1.9	+4.1±2.1	+12.0±2.5	−0.9±2.7
ETTh2	+3.4±1.0	+10.0±1.8	+5.7±1.3	+7.8±1.9	+12.0±4.7	+5.2±5.7	+7.0±2.3	+7.7±3.1
Weather	+16.3±0.8	+0.5±1.0	+15.8±1.3	+2.7±1.3	+14.3±0.6	−0.1±0.5	+15.5±0.9	+1.0±0.9
Electricity	+19.8±1.3	+5.7±1.9	+10.7±1.2	+3.7±1.8	+8.0±0.7	+0.9±1.3	+12.8±1.1	+3.4±1.7
Exchange	−0.3±0.5	+0.5±0.6	+0.2±0.4	+0.1±0.5	+1.6±1.3	−0.2±1.6	+0.5±0.8	+0.1±0.9
Traffic	+19.1±0.6	+1.9±0.2	+14.6±0.6	+1.1±0.6	+4.4±0.8	+2.2±0.7	+12.7±0.7	+1.7±0.5
Mean	+13.2±1.2	+2.3±1.7	+11.6±1.3	+2.3±1.7	+8.0±1.7	+2.1±2.0	+10.9±1.4	+2.3±1.8

The constructed-task averages over the 8 datasets are $\Delta_{\text{attn}} = +10.9$ and $\Delta_{\text{FFN}} = +2.3$. ETTh2 is the exception: its FFN delta (+7.7) exceeds its attention delta (+7.0), although it stays below its P=16 value (+11.1, Table 2); on the other 7 datasets the FFN delta is at most +3.4. On Exchange both components stay small (+0.5 and +0.1).

Table 18: Constructed-task deltas summed over the three encoder layers for PatchTST at P=16 and P=1 (deltas computed from the rounded 9-run means at each probing point). L1: Layer 1; Σ: sum over the three layers; "Max deeper FFN" is the larger of the Layer-2 and Layer-3 FFN deltas.

	P=16		P=1			
Dataset	L1 FFN Δ	Σ FFN Δ	Σ Attn Δ	L1 FFN Δ	Σ FFN Δ	Max deeper FFN
ETTm1	+26.1	+25.8	+17.0	+2.2	+1.4	−0.2
ETTm2	+43.0	+43.8	+21.9	+2.7	+2.2	−0.2
ETTh1	+3.8	+5.2	+17.5	−0.9	−3.0	−0.4
ETTh2	+11.1	+12.4	+14.9	+7.7	+7.9	+0.2
Weather	+21.5	+22.8	+25.6	+1.0	−0.9	−0.7
Electricity	+23.7	+23.6	+14.7	+3.4	+2.9	+1.3
Exchange	+12.8	+15.1	+1.0	+0.1	−0.1	0.0
Traffic	+20.8	+19.5	+13.9	+1.7	−0.6	+0.3
8-dataset mean	**+20.4**	**+21.0**	**+15.8**	**+2.3**	**+1.2**	

Read-out. The mean adapter averages the representations of 16 consecutive tokens, which could hide what the FFN writes into individual tokens. With a concatenation adapter that keeps all 16 vectors (a $16d_{\text{model}}$-dimensional probe input), the FFN delta on the 4 sweep datasets is $+0.8$ (ETTh1), $+3.4$ (ETTm1), $+3.1$ (Weather), and $+3.4$ (Electricity), and attention still leads on all 4 ($+10.9$ to $+17.5$). With concatenation, the embedding already exposes every linear functional of the 16 points, as at $P{=}16$, so attention's lead is not an artifact of what mean pooling discards.

B.8 Control Results

Table 19: Random-label control on PatchTST (top; 9-run mean $\pm$ std per cell) and on each model of Figure 3 and Table 3 (bottom). Each row of the bottom block gives the mean $\pm$ std and the range of the model's 9-run cell means over the 8 datasets and its probing points (Layer 1 only for NoAttn from scratch). Labels have 4 classes drawn independently for each token (chance 25.0%); this control marks the chance floor used in Appendix F.

Dataset	embed	1a_attn	1b_ffn	2a_attn	2b_ffn	3a_attn	3b_ffn
ETTm1	$25.3{\pm}0.2$	$25.3{\pm}0.1$	$25.3{\pm}0.1$	$25.3{\pm}0.1$	$25.3{\pm}0.1$	$25.3{\pm}0.1$	$25.2{\pm}0.0$
ETTm2	$25.4{\pm}0.1$	$25.3{\pm}0.1$	$25.4{\pm}0.2$	$25.6{\pm}0.2$	$25.5{\pm}0.2$	$25.4{\pm}0.2$	$25.4{\pm}0.2$
ETTh1	$25.4{\pm}0.2$	$25.4{\pm}0.2$	$25.4{\pm}0.1$	$25.4{\pm}0.2$	$25.4{\pm}0.1$	$25.4{\pm}0.2$	$25.5{\pm}0.1$
ETTh2	$25.2{\pm}0.2$	$25.1{\pm}0.2$	$25.2{\pm}0.2$	$25.2{\pm}0.2$	$25.2{\pm}0.2$	$25.2{\pm}0.2$	$25.3{\pm}0.3$
Weather	$25.3{\pm}0.1$	$25.3{\pm}0.2$	$25.3{\pm}0.1$	$25.3{\pm}0.2$	$25.3{\pm}0.1$	$25.4{\pm}0.1$	$25.3{\pm}0.1$
Electricity	$25.3{\pm}0.1$	$25.3{\pm}0.2$	$25.4{\pm}0.1$	$25.3{\pm}0.2$	$25.3{\pm}0.2$	$25.3{\pm}0.2$	$25.3{\pm}0.2$
Exchange	$25.2{\pm}0.1$	$25.3{\pm}0.1$	$25.2{\pm}0.2$	$25.2{\pm}0.2$	$25.2{\pm}0.1$	$25.2{\pm}0.2$	$25.3{\pm}0.1$
Traffic	$25.3{\pm}0.2$	$25.4{\pm}0.3$	$25.3{\pm}0.2$	$25.3{\pm}0.2$	$25.3{\pm}0.3$	$25.2{\pm}0.3$	$25.2{\pm}0.3$

Model	Mean $\pm$ std	Range
PatchTST	$25.3{\pm}0.1$	25.1 to 25.6
NoAttn, ablated	$25.3{\pm}0.1$	25.2 to 25.5
NoAttn, from scratch	$25.3{\pm}0.1$	25.2 to 25.5
PatchTST ($P{=}1$)	$25.4{\pm}0.1$	25.3 to 25.5
iTransformer	$25.6{\pm}0.3$	25.2 to 26.4
CATS	$25.3{\pm}0.2$	25.1 to 26.0

B.9 Informer Multi-Dataset Probing

We probe Informer on 4 datasets (ETTm1, ETTh1, Weather, Electricity; Table 20). Informer's tokens are not single scalars: each encoder token is a three-step circular convolution over all V variates plus time-feature and positional embeddings. The model also differs from PatchTST in its ProbSparse attention, the absence of RevIN, seq_len $= 96$, and a single architecture setting for all datasets; a nonlinear decoder, rather than a linear head, reads its encoder output. The single-scalar argument of §5.4 therefore does not apply, and we report this as a confounded auxiliary result.

Table 20: Informer Layer-1 deltas (constructed task average, 9-run mean, mean-adapter probe) across 4 datasets. Both components are flat, but Informer tokens are not single scalars, so this is a confounded auxiliary result (see text).

Dataset	Attn Δ	FFN Δ
ETTm1	$+1.62$	$+2.66$
ETTh1	$+1.44$	$+1.75$
Weather	$+0.37$	$+1.50$
Electricity	-0.04	-0.02
Mean	$\mathbf{+0.85}$	$\mathbf{+1.47}$

Probe-design confound. Because an Informer token holds all variates, the mean adapter pools 16 consecutive token representations (stride 8) into one vector, replicates it V times ($V = 7/7/21/321$ for ETTm1/ETTh1/Weather/Electricity), and pairs each copy with a label from one variate's own 16-step patch. Each probe input thus appears V times with labels that need not agree, which caps the attainable probing accuracy at every layer and can compress the deltas, more so as V grows. An earlier run probed per-time-step representations against variate-averaged labels on the 16-step window ending at each step and gave FFN deltas of $+5.33/+5.09/+5.97/-0.94$ (ETTm1/ETTh1/Weather/Electricity).

Because that run differed not only in the labels but also in the probe unit and (except on ETTh1) the number of test windows, the gap to the FFN deltas above cannot be attributed to label design alone. We therefore read the flat deltas as an open empirical observation, with the model differences above, Informer's training settings, and this probe design as candidate causes.

B.10 Inter-Token Task Results

Table 21 reports Layer-1 probing deltas for the four inter-token tasks (adjacent_similarity, relative_magnitude, position_quarter, trend_consistency) across the 4 models analyzed here (PatchTST, iTransformer, CATS, and PatchTST-NoAttn). These tasks' labels depend on a token's position or on other tokens rather than on its own points alone.

Table 21: Inter-token task Layer-1 deltas across architectures (9-run means averaged over the 8 datasets; PatchTST-NoAttn is the ablated model). For CATS, the Attn column gives the accuracy after cross-attention (abs), as in Table 14 (see note below).

Model	Task	Attn Δ	FFN Δ
PatchTST	adjacent_similarity	+5.7	+12.5
	relative_magnitude	+0.0	−0.3
	position_quarter	−0.9	−3.4
	trend_consistency	+0.7	+2.9
iTransformer	adjacent_similarity	+2.2	+1.2
	relative_magnitude	+5.0	+3.5
	position_quarter	+11.3	+8.1
	trend_consistency	+1.5	+1.2
CATS[†]	adjacent_similarity	76.2 (abs)	+1.4
	relative_magnitude	63.4 (abs)	+2.2
	position_quarter	98.5 (abs)	+0.9
	trend_consistency	90.0 (abs)	+0.2
PatchTST-NoAttn	adjacent_similarity	+0.1	+18.5
	relative_magnitude	+0.0	−0.3
	position_quarter	−0.0	−2.5
	trend_consistency	+0.0	+3.5

[†] Attn: absolute accuracy (abs) at `1a_cross`, not a delta: the embedding lies on the key-value side and `1a_cross` on the query side, whose labels are those of the forecast patches. FFN Δ: from `1a_cross` to `1b_ffn`, within the query side.

These results reveal architecture-dependent patterns on the inter-token tasks that are qualitatively distinct from the intra-token feature construction patterns analyzed in the main text. PatchTST's FFN strongly raises the linear decodability of inter-patch similarity (+12.5 pp on adjacent_similarity), and PatchTST-NoAttn's FFN raises it even more (+18.5 pp). NoAttn has no token mixing, so this gain arises within each token: with stride 8, a 16-point patch already shares 8 time points with the next. iTransformer's attention across variates contributes more than its FFN to position and relative-magnitude information (+11.3 pp on position_quarter, +5.0 pp on relative_magnitude). For variate tokens, the tasks are defined across variates: adjacent_similarity is whether a variate's series correlates positively with the next variate's (the previous variate's for the last variate), relative_magnitude whether its mean exceeds the mean over variates, position_quarter which quarter of the variate order it falls in, and trend_consistency whether the sign of its end-minus-start change agrees with the majority of variates. Because relative_magnitude is computed before the normalization, it depends on the variates' means, which no token holds. A mechanistic interpretation of these patterns, particularly of the role of attention heads in inter-token routing versus the FFN's encoding of token-level features, requires dedicated analytical machinery (e.g., per-head attention attribution) and is left for future work.

B.11 Additional Probing Targets

Table 22 reports Layer-1 deltas for the three targets added in §3.2. *Seasonal concentration* is the share of nonzero-frequency power in the largest spectral bin, $\max_k p_k / \sum_k p_k$, where p_k is bin k

of $|\text{rfft}(\tilde{x})|^2$ and k runs over the nonzero frequencies. The *multi-scale ratio* is $\text{var}(c)/\text{var}(\tilde{x} - c')$, where c averages adjacent pairs of the patch and c' repeats each average twice. Both are 4-class quartile tasks, with embedding probing accuracies of 29.5 to 52.0% and 29.2 to 41.7%. *Long-range consistency* is binary: whether the sign of the slope within token i equals the sign of the slope through the means of tokens $i-8, \ldots, i-1$ (tokens with $i < 8$ are excluded). Unlike the class boundaries of Appendix A, the quartile thresholds of the first two targets are computed on the probe's training part only. The FFN delta exceeds the attention delta on 7, 7, and 6 of the 8 datasets, and on 5 of the 8 for each target it also reaches at least 3 pp. The exceptions are Weather on the first two targets and, on long-range consistency, Traffic (attention ahead) and ETTh2 (a tie at zero); ETTh1 and Exchange are also weak on the first two targets, and ETTh1 is near flat on long-range consistency. Weather is attention-leaning on both intra-token targets, consistent with its having the weakest FFN/attention ratio in Table 2 (1.3×).

Table 22: Layer-1 deltas for the additional probing targets on PatchTST (9-run mean ± std). The control is a binary random label on the long-range subset.

	Seasonal conc.		Multi-scale ratio		Long-range cons.		Control	
Dataset	Attn Δ	FFN Δ	Attn Δ	FFN Δ	Attn Δ	FFN Δ	Attn Δ	FFN Δ
ETTm1	+3.5±0.3	+19.3±0.5	+4.8±0.9	+15.3±0.9	+2.1±0.2	+7.6±0.4	+0.0±0.3	+0.0±0.3
ETTm2	+10.2±0.6	+13.3±0.8	+14.5±0.4	+15.7±1.8	+3.0±0.3	+8.7±0.4	−0.2±0.3	+0.1±0.3
ETTh1	−0.2±0.4	+1.3±1.2	+1.3±0.5	+2.9±1.4	+0.0±0.4	+0.3±0.6	−0.1±0.1	+0.0±0.1
ETTh2	−0.6±1.1	+5.9±1.2	−0.4±0.8	+4.3±1.2	+0.0±0.4	+0.0±0.4	+0.0±0.3	+0.0±0.3
Weather	+10.0±0.8	+7.4±1.1	+12.5±0.7	+8.1±1.1	+5.8±0.8	+16.4±0.8	+0.0±0.2	+0.0±0.3
Electricity	+10.7±0.4	+22.6±0.7	+12.0±0.4	+15.2±1.3	+1.3±0.6	+3.9±0.7	+0.0±0.2	+0.0±0.2
Exchange	+0.4±0.5	+1.0±1.1	+0.0±0.3	+2.7±0.6	−0.5±0.5	+8.6±1.9	+0.1±0.1	+0.0±0.1
Traffic	+4.6±0.4	+25.1±0.8	+6.2±0.3	+11.8±0.4	+5.1±0.4	+0.6±0.4	+0.0±0.2	+0.1±0.1
8-dataset mean	**+4.8**	**+12.0**	**+6.4**	**+9.5**	**+2.1**	**+5.8**	**+0.0**	**+0.0**

B.12 Patch-Length Sweep

Table 23: PatchTST Layer-1 constructed-task deltas by patch length (9-run mean ± std). P=1 and P=16 (stride 8) are the main checkpoints; $P \in \{2, 4, 8, 32\}$ (stride $P/2$) and the non-overlapping P=16 (stride 16) are separately trained PatchTST checkpoints (§4). For $P \geq 2$ the tasks are computed on each token's own P points, so rows probe statistics of different lengths (at P=2, volatility and diff_energy are both quartiles of $|x_2 - x_1|$ and share their labels); at P=1 the tasks describe 16-point segments read through the mean adapter. We report deltas rather than FFN/attention ratios because Attn Δ crosses zero on ETTh1; on ETTm1 the ratio is 0.18× at P=1 and 1.6 to 3.0× for $P \geq 2$, on Electricity 0.27× and 2.0 to 5.1×, and on Weather 0.07× and 0.9 to 1.9×.

	ETTh1		ETTm1		Weather		Electricity	
P (stride)	Attn Δ	FFN Δ	Attn Δ	FFN Δ	Attn Δ	FFN Δ	Attn Δ	FFN Δ
1 (1)	+12.0±2.5	−0.9±2.7	+11.9±2.2	+2.2±3.4	+15.5±0.9	+1.0±0.9	+12.8±1.1	+3.4±1.7
2 (1)	−2.1±0.8	+23.3±3.3	+10.4±0.7	+31.4±4.2	+13.9±2.4	+13.0±3.2	+5.9±4.7	+30.1±4.8
4 (2)	+0.8±0.8	+20.3±2.9	+14.2±1.6	+31.5±4.3	+14.5±1.2	+17.2±1.6	+7.4±0.8	+30.7±2.9
8 (4)	+1.0±0.7	+16.2±2.6	+16.2±1.5	+27.3±3.1	+18.5±1.0	+16.1±1.5	+8.4±0.6	+35.7±1.1
16 (8)	+0.0±0.4	+3.8±1.8	+13.5±1.6	+26.1±1.7	+16.7±0.7	+21.5±1.8	+11.3±0.5	+23.7±0.8
32 (16)	−0.1±0.6	+0.5±1.2	+13.0±0.9	+21.5±1.1	+13.5±1.0	+26.1±1.3	+10.1±0.6	+20.5±1.0
16 (16)	−0.2±0.8	+4.0±1.4	+14.3±0.8	+26.8±0.9	+16.1±1.0	+18.6±3.6	+10.8±0.5	+25.5±0.8

Whenever a token holds the points a task depends on ($P \geq 2$), the FFN delta is large on ETTm1, Weather, and Electricity, and on ETTh1 for $P \leq 8$. It exceeds the attention delta in 18 of the 20 pairs of dataset and patch length; on Weather, attention is slightly ahead at P=2 and ahead at P=8. The non-overlapping P=16 row differs from the overlapping one by less than 3 pp.

B.13 Prediction Length Verification

We check whether the FFN dominance pattern persists across forecasting horizons by retraining PatchTST with pred_len $\in \{192, 336, 720\}$ on 4 datasets (ETTm1, ETTh1, Weather, Electricity) and applying our probing protocol.

Table 24: Cross-horizon check of FFN dominance: constructed-task deltas for PatchTST at 4 forecasting horizons (9-run mean). FFN $\Delta \geq$ Attn Δ holds at every horizon. Values for pred_len=96 are from Table 2.

Dataset	pred_len	Attn Δ	FFN Δ	Ratio
ETTm1	96	$+13.5$	$+26.1$	1.9×
ETTm1	192	$+15.7$	$+24.5$	1.6×
ETTm1	336	$+13.9$	$+25.6$	1.8×
ETTm1	720	$+13.3$	$+26.9$	2.0×
ETTh1	96	$+0.0$	$+3.8$	—
ETTh1	192	-0.1	$+4.1$	—
ETTh1	336	$+0.0$	$+4.0$	—
ETTh1	720	$+0.0$	$+4.3$	—
Weather	96	$+16.7$	$+21.5$	1.3×
Weather	192	$+16.6$	$+20.2$	1.2×
Weather	336	$+16.6$	$+18.7$	1.1×
Weather	720	$+15.4$	$+15.8$	1.0×
Electricity	96	$+11.3$	$+23.7$	2.1×
Electricity	192	$+10.0$	$+23.0$	2.3×
Electricity	336	$+11.3$	$+25.1$	2.2×
Electricity	720	$+10.7$	$+25.2$	2.4×

FFN $\Delta \geq$ Attn Δ holds in all 16 pairs of dataset and horizon (Table 24). On ETTm1 the FFN/Attn ratio ranges from 1.6× to 2.0× with no systematic trend; on Electricity it stays at 2.1 to 2.4×; on Weather it falls from 1.3× to 1.0× as the horizon grows, and at pred_len=720 the two deltas nearly tie ($+15.8$ against $+15.4$). ETTh1 shows minimal Attn Δ at all horizons. The FFN's leading role in construction under patch tokens thus extends to longer horizons on these 4 datasets.

C Data Quality Verification

C.1 Quartile Boundary Verification

Table 25: Volatility quartile boundaries Q1 to Q3: quartiles of the variances of the 210,000 RevIN-normalized 16-point patches in each dataset's PatchTST probe sample (Appendix A), that is, the class boundaries of the volatility labels. zero%: share of these patches with variance below 10^{-6}. Status: collapsed if two adjacent boundaries coincide, and normal otherwise.

Dataset	Q1	Q2	Q3	zero%	Status
ETTm1	0.064	0.198	0.399	0.3%	✓normal
ETTm2	0.092	0.348	0.668	12.9%	✓normal
ETTh1	0.327	0.646	1.041	1.3%	✓normal
ETTh2	0.244	0.582	0.951	11.4%	✓normal
Weather	0.003	0.020	0.109	14.7%	✓normal
Electricity	0.483	0.810	1.099	0.4%	✓normal
Exchange	0.023	0.054	0.121	0.0%	✓normal
Traffic	0.348	0.726	1.192	0.0%	✓normal
Solar	0.000	0.000	0.137	55.4%	× collapsed

In Solar (Table 25), Q1 and Q2 are both 0 because 55.4% of patches (nighttime) have variance ≈ 0. Quartile classification then effectively reduces to a 3-class problem, and high accuracy is achievable simply by detecting whether the variance is zero. The other 8 datasets in the table have well-separated quartile boundaries, making 4-class classification meaningful.

C.2 Channel Correlation Verification

Table 26: Inter-channel Pearson correlation in the same probe samples, computed within each input window between the concatenated patches of two variates and averaged over the sample's first 20 windows (fewer for Electricity and Traffic) and over all pairs of variates. >0.5 ratio: share of pairs whose window-averaged correlation exceeds 0.5. Status describes the mean correlation: low below 0.3, moderate from 0.3 to 0.5, high from 0.5 to 0.8, and redundant above 0.8.

Dataset	#Variables	Mean correlation	>0.5 ratio	Status
ETTm1	7	0.062	9.5%	✓low
ETTm2	7	0.241	19.0%	✓low
ETTh1	7	0.113	14.3%	✓low
ETTh2	7	0.276	28.6%	✓low
Weather	21	0.033	22.4%	✓low
Electricity	321	0.491	55.6%	△ moderate
Exchange	8	0.335	35.7%	△ moderate
Traffic	862	0.593	69.5%	△ high
Solar	137	0.855	100.0%	× redundant

Solar's 137 plants all lie in one state (Alabama; Lai et al., 2018), yielding a mean correlation of 0.855 (Table 26). Under channel-independent treatment, Solar thus amounts to nearly the same time series repeated 137 times. Electricity, Exchange, and Traffic also show channel correlation but keep well-separated quartile boundaries, so their probing results remain interpretable.

C.3 Solar-Energy Exclusion Rationale

Solar-Energy exhibits both issues (quartile collapse and channel redundancy) and is therefore excluded from the main analysis. Table 27 compares the main 8-dataset configuration with a 9-dataset configuration that adds Solar.

Table 27: Effect of adding Solar to the main 8 datasets: Layer-1 deltas and, for CATS cross-attention, absolute accuracy (constructed task average, 9-run mean).

Metric	8 datasets (main)	8 datasets + Solar	Diff
PatchTST Attn Δ	$+6.4$	$+6.5$	$+0.1$
PatchTST FFN Δ	$+20.4$	$+18.6$	-1.8
iTransformer Attn Δ	$+13.8$	$+14.2$	$+0.4$
iTransformer FFN Δ	$+5.5$	$+5.7$	$+0.2$
CATS cross-attn accuracy	44.2	47.4	$+3.2$
CATS FFN Δ	$+2.5$	$+2.6$	$+0.1$

Adding Solar introduces only minor numerical shifts on most metrics (≤ 0.4 pp), with the largest changes on PatchTST FFN Δ (-1.8 pp) and CATS cross-attention ($+3.2$ pp). The architecture-level dominance pattern is preserved (PatchTST: FFN-dominant, iTransformer: attention-dominant, CATS: cross-attention-dominant), so excluding Solar does not change it.

Solar full probing results. The quartile collapse is directly visible at the embedding layer (Table 28): volatility (86.1%), diff_energy (83.7%), and spectral_entropy (80.4%) are already high on Solar, against 27.7 to 54.9% on the 8 main datasets (§5.1; e.g., ETTm1 volatility 38.9%, Table 11). With only ~ 12 pp gained by `1b_ffn` (vs. ~ 40 pp on ETTm1), these tasks no longer test construction on Solar: their labels largely separate nighttime patches of near-zero variance from the rest (Appendix C.1), so they are decodable at the embedding for a reason specific to the data, not because a linear read-out of the patch determines them (§3.2).

Table 28: PatchTST full layer probing results on Solar (9-run mean ± std). The constructed-task group (volatility, diff_energy, spectral_entropy) starts above 80% at the embedding layer due to Solar's quartile collapse, leaving little room for FFN-driven gains and undermining the constructed/innate dichotomy used in our main analysis. This is the data quality issue that motivates Solar's exclusion (Appendix C.1).

Task	embed	1a_attn	1b_ffn	2a_attn	2b_ffn	3a_attn	3b_ffn
Innate and degenerate (classification)							
trend_direction	99.4±0.0	99.4±0.0	99.2±0.0	98.9±0.1	98.7±0.0	98.3±0.1	97.9±0.2
dominant_period	99.4±0.0	99.4±0.0	99.6±0.0	99.6±0.0	99.6±0.0	99.5±0.0	99.5±0.0
Constructed (classification)							
volatility$_{q4}$	86.1±0.1	92.8±0.5	97.0±0.2	96.9±0.2	96.6±0.1	96.1±0.2	95.6±0.3
diff_energy$_{q4}$	83.7±0.1	89.9±0.4	95.5±0.4	95.4±0.2	95.2±0.3	94.7±0.1	94.2±0.1
spectral_entropy$_{q4}$	80.4±0.3	90.3±0.7	93.1±0.7	94.9±0.5	94.8±0.5	95.1±0.4	94.2±0.4
Boundary tasks and local_global_trend (classification)							
slope_magnitude$_{q4}$	74.5±0.3	82.4±1.0	86.5±0.3	86.9±0.4	86.7±0.4	87.0±0.2	85.7±0.3
change_point	88.4±0.1	93.6±0.6	97.1±0.3	96.4±0.3	96.3±0.4	95.6±0.2	95.3±0.2
local_global_trend	99.4±0.0	99.4±0.0	99.2±0.0	98.9±0.1	98.7±0.0	98.3±0.1	97.9±0.2
Inter-token (classification)							
adjacent_similarity	93.6±0.1	95.2±0.4	95.4±0.2	95.5±0.1	95.2±0.2	94.8±0.4	94.2±0.2
relative_magnitude	100.0±0.0	99.9±0.0	99.9±0.0	99.8±0.0	99.7±0.1	99.5±0.1	99.4±0.0
position_quarter	100.0±0.0	100.0±0.0	99.7±0.0	98.8±0.1	97.5±0.3	93.9±0.6	90.9±0.9
trend_consistency	81.3±0.1	87.5±0.2	87.9±0.3	87.4±0.4	87.0±0.3	85.3±0.2	84.2±0.6
Regression (R^2)							
reg_variance	0.402±0.002	0.823±0.016	0.956±0.003	0.950±0.006	0.949±0.004	0.934±0.006	0.933±0.008
reg_slope	1.000±0.000	1.000±0.000	0.999±0.000	0.997±0.000	0.995±0.001	0.989±0.001	0.984±0.002
reg_next_mean	0.976±0.000	0.979±0.000	0.979±0.000	0.978±0.000	0.978±0.001	0.975±0.001	0.975±0.001
Control							
random_label	25.3±0.2	25.3±0.2	25.3±0.1	25.4±0.2	25.5±0.2	25.4±0.3	25.4±0.2

D ModernTCN Component Breakdown

We also probe ModernTCN [Luo and Wang, 2024], a CNN-based time series model with structure Patch Embed → [DWConv + ConvFFN1 + ConvFFN2] $\times L$, where DWConv performs linear inter-patch mixing while the two ConvFFNs perform nonlinear processing. We evaluate on ETTm1 using ModernTCN's official 3-layer configuration ($L = 3$); a 1-layer configuration cannot, by construction, show cancellation across layers.

ModernTCN's block decomposes into three components per layer: DWConv (linear inter-patch mixing), ConvFFN1 (cross-feature mixing, nonlinear), and ConvFFN2 (cross-variable mixing, nonlinear). Table 29 reports per-component cumulative deltas across all three layers on ETTm1.

Table 29: ModernTCN per-component cumulative deltas across all three layers. ConvFFN1 contributes most of the nonlinear feature construction; ConvFFN2 contributes additionally; DWConv shows sign reversal across layers (positive at Layer 1, negative at Layers 2 and 3) with net cancellation when summed.

Task	Total DWConv Δ	Total ConvFFN1 Δ	Total ConvFFN2 Δ	Total ConvFFN Δ
volatility	−3.1	+11.7	+4.1	+15.8
diff_energy	+2.2	+23.9	+2.3	+26.2
spectral_entropy	−0.1	+15.8	+5.9	+21.7
Constructed avg	**−0.3**	**+17.1**	**+4.1**	**+21.2**

Per-layer DWConv contributions are +13.8, −7.9, −6.2 (Layers 1, 2, and 3, respectively), illustrating the sign reversal and net cancellation: DWConv's gain at Layer 1 is offset by its negative contributions at Layers 2 and 3. ConvFFN1 contributions are stable across layers (+6.5, +5.5, +5.1), and ConvFFN2 adds a smaller but consistent increment (+1.0, +1.9, +1.2). The linear DWConv leads in the first block, which is where the main analysis compares components, but the ConvFFNs lead from the second block on and carry most of the construction summed over the three blocks, as the FFN does in PatchTST (§6).

E Probe Convergence and Expressiveness

E.1 Convergence

Table 30: Probing accuracy by gradient steps on ETTm1 (top) and the FFN delta minus the attention delta on three more datasets (bottom; positive: the FFN leads), for PatchTST with the three model seeds of the main analysis (9-run means). The probes were retrained for this table, and their random draws differ from those behind Table 11, so the ETTm1 30k column differs from it by up to 0.6 pp at embed and 0.1 pp after attention and the FFN. The component that leads on each task is the same at every step count on ETTm1, and from 5k on elsewhere.

Layer	Task	1k	5k	10k	20k	30k	50k	80k
embed	volatility	38.6	38.7	38.7	38.7	38.7	38.7	38.7
1a_attn	volatility	43.9	46.8	47.9	48.7	49.2	49.7	50.2
1b_ffn	volatility	70.2	79.7	81.6	83.0	83.8	84.6	85.4
embed	diff_energy	41.4	41.5	41.5	41.5	41.5	41.5	41.5
1a_attn	diff_energy	49.6	52.8	54.1	55.1	55.7	56.5	57.1
1b_ffn	diff_energy	54.6	61.2	63.5	65.3	66.2	67.2	68.0
embed	spectral_entropy	28.4	29.8	30.0	30.2	30.2	30.3	30.4
1a_attn	spectral_entropy	41.5	44.4	45.5	46.5	47.1	47.8	48.4
1b_ffn	spectral_entropy	68.2	75.9	77.9	79.4	80.1	80.9	81.5
FFN Δ minus Attn Δ (pp)								
ETTh1	volatility	$+0.8$	$+2.3$	$+2.8$	$+3.0$	$+3.0$	$+3.0$	$+3.0$
ETTh1	diff_energy	$+2.0$	$+3.0$	$+3.1$	$+3.1$	$+3.1$	$+3.1$	$+3.1$
ETTh1	spectral_entropy	$+1.2$	$+4.3$	$+5.2$	$+5.3$	$+5.2$	$+5.1$	$+5.0$
Weather	volatility	-1.4	$+2.8$	$+4.3$	$+5.7$	$+6.3$	$+6.9$	$+7.4$
Weather	diff_energy	-5.8	-5.3	-5.0	-4.9	-4.8	-4.4	-4.7
Weather	spectral_entropy	$+2.7$	$+6.8$	$+9.4$	$+11.2$	$+11.9$	$+12.7$	$+13.6$
Electricity	volatility	$+17.7$	$+18.7$	$+18.7$	$+18.2$	$+17.9$	$+17.6$	$+17.4$
Electricity	diff_energy	$+13.0$	$+14.8$	$+15.2$	$+15.1$	$+15.0$	$+14.9$	$+14.7$
Electricity	spectral_entropy	$+4.5$	$+5.0$	$+4.8$	$+4.7$	$+4.5$	$+4.3$	$+4.2$

On ETTm1, embed accuracy changes by at most 0.6 pp after 5k steps (Table 30, top). FFN-layer accuracy increases modestly even at 80k steps ($+1.6$, $+1.8$, and $+1.4$ pp on volatility, diff_energy, and spectral_entropy from 30k to 80k), which we attribute to the standard logistic-regression dynamic of weight norms increasing under cross-entropy loss without weight decay. The component that leads on each task is the same at every step count (the FFN on volatility and spectral_entropy, attention on diff_energy). From 30k to 80k, the FFN/Attn Δ ratio falls by only 5 to 7% (volatility: $3.3\times$ vs. $3.1\times$; diff_energy: $0.74\times$ vs. $0.70\times$; spectral_entropy: $2.0\times$ vs. $1.8\times$), so on ETTm1 the conclusion on component dominance does not depend on the number of training steps. On ETTh1, Weather, and Electricity, the leading component of each task is also the same at every step count from 5k on, and the FFN-layer accuracy, not shown in Table 30, rises by at most 3.0 pp from 30k to 80k. We adopt 30k steps as a balance between near-saturation and computational cost.

E.2 Linear Probe vs. MLP Probe

Because a linear probe can miss nonlinearly encoded information, we compare it with a 2-layer MLP probe (hidden width 128, ReLU) on identical representations across 4 datasets (ETTm1, ETTh1, Weather, Electricity) and four tasks (three constructed: volatility, diff_energy, spectral_entropy; one innate: trend_direction). The MLP probe is matched to the linear probes: it is trained and scored as in §3.4, with the same split for each probe seed, Adam at the same learning rate, the same batch size and number of steps, and the best test accuracy over evaluations every 500 steps. Each cell of Table 31 is a 9-run mean.

Table 31: Linear vs. MLP probe comparison on PatchTST across 4 datasets and four tasks (9-run mean). MLP probes saturate at the embed layer for constructed tasks; linear probes gain at the FFN instead. Linear probes measure what is linearly decodable, the format a linear head reads, while MLP probes measure information present in some form.

		ETTm1		ETTh1		Weather		Electricity	
Task	Probe	embed	FFN Δ	embed	FFN Δ	embed	FFN Δ	embed	FFN Δ
volatility	Linear	38.9	+34.5	37.6	+2.9	33.6	+24.0	41.6	+30.5
	MLP	**99.0**	−0.2	**95.5**	−0.4	**99.1**	−0.3	**97.5**	+0.2
diff_energy	Linear	42.1	+10.6	38.7	+3.1	33.8	+14.8	38.4	+26.4
	MLP	**98.6**	−0.2	**92.9**	−1.4	**97.5**	−0.8	**96.4**	+0.5
spectral_entropy	Linear	30.2	+33.2	35.8	+5.4	36.2	+25.8	41.6	+14.3
	MLP	**98.5**	−0.3	**91.0**	−1.4	**98.7**	−0.4	**96.0**	−0.1
trend_direction	Linear	99.6	−0.2	97.9	−1.4	98.5	−0.3	99.6	−0.4
	MLP	100.0	0.0	99.6	−0.1	99.9	−0.1	99.9	−0.1

Observation. On constructed tasks (volatility, diff_energy, spectral_entropy), MLP probes saturate at the embed layer across all 4 datasets, achieving 91 to 99% with negligible change through attn/FFN ($|\Delta| \leq 1.4$ pp). Linear probes, in contrast, start at about 30 to 42% (chance 25%) and gain $+2.9$ to $+34.5$ pp at the FFN, least on ETTh1. On the innate task (trend_direction), both probes saturate near ceiling at the embed layer.

Interpretation. PatchTST's patch embedding W_P is a full-rank linear projection from the patch dimension (16) to d_{model} (128, or 16 on ETTh1). MLP probes can effectively invert this projection to recover raw patches and reimplement task formulas (variance, diff energy, spectral entropy) directly. Hence, MLP probing accuracy reflects where information exists in some form, not where it is linearly decodable.

Linear probes are appropriate. PatchTST's forecasting head, like those of its variants and ModernTCN, is the linear projection $\text{Linear}(d_{\text{model}} \cdot N_{\text{patches}}, \text{pred_len})$. iTransformer and CATS map each token linearly to its part of the forecast (a variate's horizon, a query's future patch); only Informer reads its encoder output with a nonlinear decoder (Appendix B.9). Linear decodability is thus the format these heads read, and it is what our linear probes measure. The linear-probe deltas at the FFN show that the FFN re-encodes raw patch information into a linearly decodable form, the property relevant to our central question.

F Selectivity Analysis

We compute the selectivity of Hewitt and Liang [2019] on the same probing data used in §5, with a random-label control in place of their word-type control tasks, which give each word type a fixed random label. For each (dataset, layer, task) triple we report

$$\text{selectivity}(d, \ell, t) \;=\; \text{acc}(d, \ell, t) \;-\; \text{acc}_{\text{ctrl}}(d, \ell),$$

where acc_{ctrl} is the accuracy of the same probe trained on a 4-class *random-label* control task. All values are 9-run means and aggregated across the 8 datasets used in the main analysis. The control task stays near chance at every probing point of the 4 models analyzed here (PatchTST, PatchTST-NoAttn, iTransformer, and CATS; Table 19). Its labels are drawn independently for each token and carry nothing a probe could transfer to held-out tokens, so the control stays at chance by construction and marks the chance floor at each probing point. Unlike word-type control tasks, it does not measure probe capacity, which we fix instead by restricting the read-out to the forecasting heads' linear family (a matched MLP probe saturates at the embedding; Appendix E.2). Throughout this section, we write Δ_a for the delta from the embedding to the output of the first sub-layer (attention or cross-attention) and Δ_b for the delta from there to the FFN output.

PatchTST Layer-1 selectivity. The three constructed tasks gain on average $+26.70$ pp through Layer 1 (paired t-test pooled over the 24 dataset-task pairs, $p < 10^{-6}$, Cohen's $d_z = +1.67$, very large), with the FFN sub-layer contributing $+20.33$ pp ($d_z = +1.59$) versus attention's $+6.37$ pp ($d_z = +0.89$). The three tasks of a dataset are probed on the same forecasters, so these pairs are not independent; a test on the 8 dataset means gives $p < 10^{-3}$. Through Layer 1, the innate trend_direction

changes by -1.2 pp, the degenerate dominant_period by $+1.1$ pp, and local_global_trend (outside the classification) by $+0.6$ pp. In contrast, the boundary tasks slope_magnitude$_{q4}$ and change_point gain $+28.5$ and $+16.4$ pp, mainly at the FFN, as §3.2 notes for ETTm1.

Table 32: PatchTST Layer-1 selectivity (8-dataset mean; 9-run probing). Columns: selectivity at embedding, post-attention, and post-FFN; total Layer-1 delta; Cohen's d_z and one-sided p (for an increase) of a paired t-test on $\mathrm{embed} \to \mathrm{1b_ffn}$ over the 9-run means of the 8 datasets (task rows) or of the 24 dataset-task pairs (Constructed avg row). Constructed and boundary tasks show very large selectivity gains, mainly at the FFN, while the other tasks change by at most 1.2 pp.

Task	embed	1a_attn	1b_ffn	Total Δ	Cohen's d_z	p
Constructed						
volatility$_{q4}$	$+13.1$	$+19.2$	$+43.4$	$+30.3$	$+1.58$	0.0014
diff_energy$_{q4}$	$+12.4$	$+20.0$	$+36.7$	$+24.3$	$+1.69$	0.0010
spectral_entropy$_{q4}$	$+11.2$	$+16.6$	$+36.7$	$+25.5$	$+1.65$	0.0011
Constructed avg	$\mathbf{+12.2}$	$\mathbf{+18.6}$	$\mathbf{+38.9}$	$\mathbf{+26.7}$	$\mathbf{+1.67}$	$\mathbf{< 10^{-6}}$
Innate and degenerate						
trend_direction	$+72.8$	$+72.3$	$+71.6$	-1.2	-0.80	0.97
dominant_period	$+68.3$	$+68.4$	$+69.4$	$+1.1$	$+0.91$	0.019
Boundary tasks and local_global_trend						
slope_magnitude$_{q4}$	$+10.3$	$+16.9$	$+38.9$	$+28.5$	$+1.55$	0.0016
change_point	$+46.1$	$+50.0$	$+62.5$	$+16.4$	$+1.62$	0.0013
local_global_trend	$+36.3$	$+37.0$	$+36.9$	$+0.6$	$+0.98$	0.014

Selectivity tracks raw accuracy. For the 8-dataset means of all 4 models, the maximum gap between the selectivity-based delta and the raw-accuracy-based delta was 0.05 pp before rounding (up to 0.63 pp on a single dataset). For PatchTST constructed tasks, $\mathrm{embed} \to \mathrm{1b_ffn}$ selectivity increased by $+26.70$ pp while raw accuracy increased by $+26.71$ pp. This near-equality follows directly from the flatness of $\mathrm{acc}_{\mathrm{ctrl}}$ across layers (25.30 to 25.33 for PatchTST), which the random-label design guarantees.

Cross-architecture validation. The architecture-specific dominance pattern reported in Figure 3 (FFN-dominant in PatchTST, attention-dominant in iTransformer, cross-attention-dominant in CATS) is reproduced under the selectivity metric (Table 33; for CATS, by the query-side trajectory of Table 34).

Table 33: Cross-architecture Layer-1 selectivity vs. raw accuracy (constructed task average; 8-dataset mean). The dominant sub-layer (**bold** in each row but CATS) and its sign are reproduced under selectivity. For PatchTST-NoAttn, Δ_a is near zero because the attention sub-layer is ablated.

Architecture	First sub-layer	Δ_a (sel)	Δ_b (sel)	Δ_a (raw)	Δ_b (raw)
PatchTST	attn	$+6.4$	$\mathbf{+20.3}$	$+6.4$	$+20.4$
PatchTST-NoAttn	attn (ablated)	-0.1	$\mathbf{+27.5}$	-0.1	$+27.4$
iTransformer	attn	$\mathbf{+13.8}$	$+5.5$	$+13.8$	$+5.5$
CATS†	cross	$-9.9^{\dagger}$	$+2.5$	-9.9	$+2.5$

† **CATS side-switch caveat.** CATS `embed` probes the KV-side (key and value) input embedding, while `1a_cross` and subsequent layers probe the Q-side post-cross-attention representation (a learnable query attending to KV). Their direct subtraction therefore reflects a side switch rather than a layer transition, and the labels also change from the input patches to the future patches the queries forecast (the KV-side labels are computed on the input padded with zeros, whereas the model pads it by repeating its last value). The negative Δ_a is thus not information loss. Table 34 shows the within-Q trajectory: the first cross-attention output reaches $+18.85$ pp of constructed-task selectivity, measured against chance rather than the query before cross-attention, while the subsequent five sub-layers add only $+4.03$ pp cumulatively.

Table 34: CATS within-Q selectivity trajectory (constructed task average, 8-dataset mean). Most of the Q-side selectivity is already present at the first cross-attention output.

Q-side layer	Selectivity	Sub-layer Δ
1a_cross	$+18.85$	— (start; cross-attention output)
1b_ffn	$+21.39$	$+2.54$
2a_cross	$+21.97$	$+0.58$
2b_ffn	$+22.65$	$+0.68$
3a_cross	$+22.69$	$+0.04$
3b_ffn	$+22.88$	$+0.19$
Total post-1a_cross gain		$+4.03$ *over 5 sub-layers*

The boundary tasks. The FFN gives most of the Layer-1 gain of both boundary tasks in Table 32: $+22.0$ of $+28.5$ pp for slope_magnitude$_{q4}$ and $+12.5$ of $+16.4$ pp for change_point. A linear read-out cannot realize an absolute value, but their labels only take the absolute value of a linear quantity that the embedding already exposes (the slope, or the difference between the two half means), so §3.2 reports them as boundary tasks rather than main tasks.

G Activation Patching

To complement the representational analysis (§5.2) on the forecasting side, we apply activation patching [Meng et al., 2022] to PatchTST on all 8 datasets, and additionally to PatchTST at $P{=}1$ and to iTransformer (Appendices G.4 and G.5). Restoring a clean component output into a corrupted run is a denoising test: it checks whether that output suffices to restore the forecast [Heimersheim and Nanda, 2024], not whether the component is necessary.

G.1 Methodology

Procedure. Following Meng et al. [2022], for each input we run two forward passes: a *clean* run on the original input, and a *corrupt* run where the input is replaced by a corrupted version. We then run a third *patched* pass: the model receives the corrupt input, but we replace the output of a chosen component (e.g., the Layer-1 FFN) with its activation from the clean run. The patching effect is

$$\text{effect}(c) = \text{MSE}_{\text{corrupt}} - \text{MSE}_{\text{patched}(c)},$$

where c is the patched component. A positive effect means that injecting the clean component output into the corrupt run *recovers* forecasting performance, that is, the clean output of c suffices to restore part of the forecast. Because the FFN's input is the normalized sum of the residual stream and the attention output, a restored FFN output also carries the part of clean attention that passes through the FFN.

Corruption protocol. Per-sample non-overlapping chunk shuffle: the input time axis is divided into 21 chunks of size 16, and each sample receives an independent permutation of its chunks. This preserves the per-sample marginal distribution exactly while disrupting temporal structure.

Hook position. Hooks are placed at `dropout_attn` and `dropout_ffn` (pre-residual component outputs), not at post-residual norm layers. With post-residual hooks, L_i_attn and L_i_ffn collapse to identical effects in eval mode (once the post-attention stream is restored, the deterministic FFN and normalization reproduce the clean stream downstream); pre-residual hooks correctly separate the two components.

Why focus on Layer 1. Two reasons: (i) probing places most construction at Layer 1 (Table 18), so Layer 1 is where a forecasting-side check of attention's contribution is most direct; (ii) deeper-layer patching is subject to *cross-context incompatibility*: the clean output enters a corrupt residual stream that has had more layers to diverge from the clean one, so the patched activation can be off-distribution downstream. Zhang and Nanda [2024] raise this concern for any intervention that introduces out-of-distribution inputs.

Experimental scale. In every patching experiment, each dataset contributes its first 100 test windows (consecutive, one step apart), and the same windows with the same corruption are used for each of the 3 model seeds, giving 300 paired comparisons per (dataset, component) cell. The main analysis thus has 8 datasets $\times$ 2 Layer-1 components = 16 cells, or 4,800 patched runs, and an additional 9,600 runs cover Layers 2 and 3 (Appendix G.3). Because each window enters a cell three times, the paired t-tests average each window's FFN-minus-attention difference over the 3 seeds and test the 100 seed-averaged windows; their p-values still treat these overlapping windows as independent. Cohen's d_z is computed over the 300 pairs.

G.2 Layer-1 Results

Table 35: Layer-1 activation patching effects on PatchTST (MSE recovered; mean over 300 paired samples per cell, 3 seeds $\times$ 100). Damage: MSE of the corrupt run minus that of the clean run, averaged over the same 300 samples. Bold marks the larger component within each dataset. Cohen's d_z is the effect size of the FFN-minus-attention difference over the 300 pairs; the paired t-test on the 100 seed-averaged windows gives $p < 10^{-16}$ on every dataset.

Dataset	Damage	L1_attn	L1_ffn	Δ (FFN $-$ attn)	Cohen's d_z
ETTm1	1.129	$+0.229$	$\mathbf{+0.473}$	$\mathbf{+0.243}$	$\mathbf{+2.10}$
ETTm2	0.100	$+0.022$	$\mathbf{+0.042}$	$\mathbf{+0.021}$	$\mathbf{+1.64}$
ETTh1	0.637	$+0.007$	$\mathbf{+0.202}$	$\mathbf{+0.194}$	$\mathbf{+3.09}$
ETTh2	0.089	$+0.000$	$\mathbf{+0.038}$	$\mathbf{+0.037}$	$\mathbf{+1.07}$
Weather	0.112	$\mathbf{+0.036}$	$+0.030$	-0.006	-0.85
Electricity	0.691	$+0.058$	$\mathbf{+0.085}$	$\mathbf{+0.027}$	$\mathbf{+1.35}$
Exchange	0.183	$+0.005$	$\mathbf{+0.075}$	$\mathbf{+0.069}$	$\mathbf{+1.08}$
Traffic	1.071	$+0.108$	$\mathbf{+0.262}$	$\mathbf{+0.153}$	$\mathbf{+3.30}$

FFN restoration exceeds attention restoration on 7 of 8 datasets. On every dataset except Weather, restoring the Layer-1 FFN recovers more forecasting performance than restoring Layer-1 attention ($d_z = 1.07$ to 3.30). Because a restored FFN output is computed from the clean attention output as well (see Procedure), the instrument favors FFN restoration, which carries little signal by itself; only attention's recovery is informative. On ETTh1, for example, attention recovers $+0.007$ against the FFN's $+0.202$.

Agreement with probing on whether attention contributes. Restoring attention recovers at most 0.022 on the 4 datasets where probing puts Attn Δ near zero (ETTh1, ETTh2, ETTm2, Exchange; Table 2) and 0.036 to 0.229 on the other 4. The agreement is binary (near zero or not) and holds only in absolute MSE: relative to the damage (Table 35), attention recovers at most 3% on ETTh1, ETTh2, and Exchange but 22% on ETTm2, about as much as on ETTm1 (20%) and more than on Electricity (8%) or Traffic (10%), so the recovered shares do not follow the ordering of the probing deltas.

Weather. On Weather, attention restoration is slightly but significantly larger at Layer 1 (0.036 vs. 0.030, $d_z = -0.85$), and the FFN leads at Layers 2 and 3 (Layer 3: 0.053 vs. 0.044). Weather also has the weakest FFN/attention probing ratio and favors attention on two additional probing targets (§5.2, Appendix B.11), so it recurs as an exception across instruments; we do not attribute this to a specific data property.

G.3 Multi-Layer Results

The Layer-1 conclusions largely extend to Layer 2 (Table 36); Layer 3 introduces cross-context incompatibility. On Traffic, attention restores more than the FFN at Layer 3 ($+0.167$ vs. $+0.035$).

Table 36: Per-layer patching effects on PatchTST across all encoder layers (MSE recovered; mean across 300 paired comparisons). Bold marks the per-layer dominant component (largest positive effect).

Component	ETTm1	ETTm2	ETTh1	ETTh2	Weather	Electricity	Exchange	Traffic
L1_attn	$+0.229$	$+0.022$	$+0.007$	$+0.000$	$\mathbf{+0.036}$	$+0.058$	$+0.005$	$+0.108$
L1_ffn	$\mathbf{+0.473}$	$\mathbf{+0.042}$	$\mathbf{+0.202}$	$\mathbf{+0.038}$	$+0.030$	$\mathbf{+0.085}$	$\mathbf{+0.075}$	$\mathbf{+0.262}$
L2_attn	$+0.177$	$+0.028$	$+0.019$	$+0.002$	$+0.039$	$\mathbf{+0.040}$	$+0.016$	$+0.122$
L2_ffn	$\mathbf{+0.409}$	$\mathbf{+0.037}$	$\mathbf{+0.197}$	$\mathbf{+0.041}$	$\mathbf{+0.040}$	$+0.039$	$\mathbf{+0.079}$	$\mathbf{+0.180}$
L3_attn	$+0.187$	$+0.029$	$+0.035$	$+0.006$	$+0.044$	$\mathbf{+0.049}$	$+0.036$	$\mathbf{+0.167}$
L3_ffn	$\mathbf{+0.356}$	$\mathbf{+0.037}$	$\mathbf{+0.147}$	$\mathbf{+0.036}$	$\mathbf{+0.053}$	-0.020	$\mathbf{+0.075}$	$+0.035$

Cross-context incompatibility in deeper layers. At Layer 3, a patched clean output enters a corrupt residual stream that has already passed through two layers, so it can lower forecasting performance instead of restoring it, as on Electricity's $L3$_ffn (-0.020). The main claims therefore rest on Layer 1, where the clean and corrupt streams have diverged least.

G.4 PatchTST at P=1

We repeat Layer-1 patching on PatchTST at P=1 for ETTh1 and ETTm1 with the same corruption. Two sanity checks (hooks off, and clean activations patched into the clean run) give zero difference, and the same code reproduces the P=16 ETTh1 values of Table 35. Restoration stays FFN-dominant at P=1 (Table 37), although probing puts construction in attention there (§5.4). Attention's restoration rises relative to the FFN's from P=16 to P=1. This instrument favors the FFN by construction and cannot confirm or refute the shift seen in probing.

Table 37: Layer-1 activation patching at P=16 and P=1 (MSE recovered; mean over 300 paired samples, 3 seeds $\times$ 100). d_z is the paired effect size of FFN minus attention at P=1 ($p < 10^{-45}$ on both datasets for the 100 seed-averaged windows).

	P=16		P=1		
Dataset	L1_attn	L1_ffn	L1_attn	L1_ffn	d_z
ETTh1	$+0.007$	$+0.202$	$+0.065$	$+0.269$	$+2.68$
ETTm1	$+0.229$	$+0.473$	$+0.267$	$+0.443$	$+2.08$

G.5 iTransformer

We also patch Layer 1 of iTransformer under two corruptions: the temporal chunk shuffle (6 chunks of 16 steps over the 96-step input) and a cross-variate corruption that circularly shifts each variate by an independent random offset of 8 to 88 steps, which destroys alignment across variates while keeping each variate's values. Both sanity checks give zero difference. Restoration is FFN-dominant for every dataset and corruption tested (Table 38), although probing puts construction in attention for iTransformer (§5.5). In iTransformer's post-norm layer, too, the clean FFN output is computed from the clean attention output, so patching favors the FFN by construction. The circular shift also misaligns each variate with its own forecast target. We therefore attribute this result to the instrument and use component knockout for the forecasting-side comparison across architectures (Appendix H).

Table 38: iTransformer Layer-1 activation patching (MSE recovered; mean over 300 paired samples, 3 seeds $\times$ 100). d_z is the paired effect size of FFN minus attention ($p < 10^{-18}$ in every row for the 100 seed-averaged windows).

Corruption	Dataset	L1_attn	L1_ffn	d_z
Temporal	ETTh1	$+0.073$	$+0.408$	$+1.74$
Temporal	Electricity	$+0.045$	$+0.133$	$+0.77$
Cross-variate	ETTh1	$+0.039$	$+0.401$	$+2.77$
Cross-variate	Electricity	$+0.011$	$+0.090$	$+2.97$
Cross-variate	Traffic	$+0.011$	$+0.096$	$+1.29$

G.6 Limitations of the Patching Analysis

In PatchTST at Layer 1, patching agrees with probing on whether attention contributes (8 datasets); the larger FFN restoration on 7 of 8 datasets reflects the instrument's bias toward the FFN and does not by itself locate construction. Activation patching gives component-level, forecasting-side evidence; finer mechanistic validation would require path patching [Wang et al., 2023] across components, head-level decomposition, and interventions designed for each architecture.

H Forecasting Without a Component

We evaluate forecasting MSE on all 8 datasets, comparing the trained PatchTST checkpoint with two models without attention: the same checkpoint with its attention output set to zero (ablated) and PatchTST-NoAttn trained from scratch (Table 39). Ablation raises MSE on all 8 datasets, with a mean increase of $+15.7\%$ over the 24 runs and $+14.4\%$ on the 8-dataset mean MSE; the increase is near zero on ETTh2 ($+0.3\%$, not positive for every seed) and small on ETTh1 ($+1.6\%$). The from-scratch model's MSE is 0.4 to 5.4% higher than PatchTST's on 7 datasets and 2.6% lower on Exchange.

Table 39: Forecasting MSE without attention across 8 datasets (3 seeds per dataset; mean $\pm$ sample std). Ablated: the trained checkpoint with the attention output set to zero; its change is computed per seed before averaging. From scratch: PatchTST-NoAttn trained with the attention output fixed to zero; its change is computed on the 3-seed means because its runs are not paired with PatchTST seeds.

Dataset	PatchTST	Ablated	Change (%)	From scratch	Change (%)
ETTm1	$0.289_{\pm 0.002}$	$0.340_{\pm 0.007}$	$+17.6_{\pm 1.7}$	$0.295_{\pm 0.001}$	$+2.0$
ETTm2	$0.164_{\pm 0.001}$	$0.183_{\pm 0.001}$	$+11.4_{\pm 0.8}$	$0.165_{\pm 0.001}$	$+0.6$
ETTh1	$0.375_{\pm 0.001}$	$0.381_{\pm 0.001}$	$+1.6_{\pm 0.2}$	$0.377_{\pm 0.001}$	$+0.4$
ETTh2	$0.275_{\pm 0.000}$	$0.276_{\pm 0.001}$	$+0.3_{\pm 0.5}$	$0.276_{\pm 0.000}$	$+0.5$
Weather	$0.151_{\pm 0.000}$	$0.204_{\pm 0.002}$	$+34.8_{\pm 1.4}$	$0.157_{\pm 0.000}$	$+4.3$
Electricity	$0.130_{\pm 0.001}$	$0.160_{\pm 0.004}$	$+22.5_{\pm 4.0}$	$0.135_{\pm 0.000}$	$+3.1$
Exchange	$0.092_{\pm 0.001}$	$0.103_{\pm 0.005}$	$+11.4_{\pm 4.6}$	$0.090_{\pm 0.000}$	-2.6
Traffic	$0.367_{\pm 0.000}$	$0.463_{\pm 0.002}$	$+26.3_{\pm 0.5}$	$0.386_{\pm 0.000}$	$+5.4$
Mean (24 runs)			$\mathbf{+15.7}$		
8-dataset mean MSE	0.2305	0.2636	$+14.4$	0.2352	$+2.0$

Without attention. The 4 datasets with the largest FFN delta change (ETTm1, Weather, Electricity, Traffic; Table 15) also show the largest ablation costs ($+17.6$ to $+34.8\%$), but ablation costs $+11.4\%$ on ETTm2 and Exchange with almost no change in probing, so the cost does not depend only on how much construction moves from attention to the FFN. Training from scratch removes most of it: the from-scratch model recovers 75 to 95% of the ablation damage on 6 datasets and beats the ablated model on every dataset except ETTh2. On the 7 datasets shared with RAM [Guo et al., 2024], the from-scratch model's MSE at horizon 96 is within 0.003 of RAM's attention-free PatchTST (mean of per-dataset changes $+2.3\%$, against RAM's $+2.2\%$). The setups are not identical. RAM deletes the attention layer but keeps the feed-forward, residual, and normalization layers, and it averages five runs; its text gives a lookback of 96 but labels the model PatchTST/42, the 336-step setting of Nie et al. [2023]. We fix the attention output to zero, use a lookback of 336, and average three seeds.

Knockout. For each frozen checkpoint, we zero the output of every attention sub-layer or of every FFN sub-layer, keep the residual connections and normalization, and measure the relative increase in test MSE over the unmodified model at pred_len $= 96$. A no-op hook reproduces the baseline MSE exactly in every run; Appendix A compares the baselines with the published MSEs. For PatchTST, the attention knockout is the ablated NoAttn model of §5.3, and its mean costs match Table 39. Table 40 reports the costs.

Table 40: Component knockout: MSE increase (%) when the attention or FFN output is zeroed in all layers of a frozen model (mean ± sample std over 3 seeds). For CATS, "Attn" is cross-attention. Ratio: mean attention cost over the 8 datasets divided by the mean FFN cost.

	PatchTST		iTransformer		CATS	
Dataset	Attn	FFN	Attn	FFN	Attn	FFN
ETTm1	+17.6±1.7	+53.1±8.5	+6.4±1.5	+11.2±2.2	+122.6±4.2	+8.3±0.5
ETTm2	+11.4±0.8	+22.8±2.0	+7.5±1.2	+1.8±0.3	+29.1±1.2	+1.0±0.5
ETTh1	+1.6±0.2	+19.5±2.0	+4.9±1.1	+8.2±0.7	+85.2±0.9	+2.6±0.3
ETTh2	+0.3±0.5	+12.4±0.2	+1.6±0.5	+3.2±0.1	+19.5±0.2	+5.2±0.6
Weather	+34.8±1.4	+36.9±1.7	+28.8±3.0	+13.6±0.6	+31.4±0.5	+7.9±1.6
Electricity	+22.5±4.0	+10.2±2.2	+38.0±0.6	+30.3±1.0	+471.9±0.4	+9.5±0.5
Exchange	+11.4±4.6	+106.1±5.6	−1.6±1.0	+3.6±1.7	+68.5±1.9	+1.4±0.3
Traffic	+26.3±0.5	+20.2±1.0	+62.3±3.0	+40.0±0.4	+234.7±0.1	+12.6±0.4
8-dataset mean	**+15.7**	**+35.1**	**+18.5**	**+14.0**	**+132.8**	**+6.1**
Ratio (attn/FFN)	0.45		1.32		21.9	
FFN cost > attn cost	6 of 8		4 of 8		0 of 8	

Ordering. The FFN cost orders the architectures PatchTST > iTransformer > CATS and the ratio orders them PatchTST < iTransformer < CATS, the same order as probing; both orderings hold in all 8 leave-one-dataset-out subsets (ratios 0.37 to 0.65, 1.19 to 1.40, and 15.2 to 25.4). CATS's cross-attention costs are large by construction, since cross-attention is its only path from the input. Per dataset, PatchTST relies more on attention for Electricity and Traffic, and iTransformer relies more on the FFN for ETTm1, ETTh1, ETTh2, and Exchange.

Single-scalar tokens. In PatchTST at P=1, the FFN cost exceeds the attention cost on every dataset tested, as at P=16 (Table 41), although probing puts construction in attention at P=1 (§5.4). Knockout thus measures what the frozen model relies on, not which component makes the probed properties decodable.

Table 41: PatchTST component knockout at P=1 and P=16 (MSE increase %, mean ± sample std over 3 seeds; the P=16 row repeats Table 40). Weather at P=1 comes from a separate run with another test batch size, which changes how many windows at the end of the test split are kept; on ETTh1 and ETTm1, that run also gives +5.1/+39.8 and +28.2/+48.3 (Attn/FFN), close to the P=1 row.

	ETTh1		ETTm1		Weather		Exchange	
P	Attn	FFN	Attn	FFN	Attn	FFN	Attn	FFN
1	+5.8±1.0	+41.4±0.8	+28.4±8.2	+48.4±9.1	+5.6±8.4	+34.5±4.8	+19.2±11.2	+65.8±1.9
16	+1.6±0.2	+19.5±2.0	+17.6±1.7	+53.1±8.5	+34.8±1.4	+36.9±1.7	+11.4±4.6	+106.1±5.6

Scope. Knockout shows what a frozen model relies on; a retrained model can compensate (§5.3). Per-dataset costs vary widely, so the comparison across architectures rests not on single cells but on the ordering of the 8-dataset means, which holds when any one dataset is left out.

I Feature-Level Erasure

We ask whether the forecaster uses an individual probed property by erasing it from the Layer-1 output (`1b_ffn`) of frozen PatchTST (P=16) and measuring the forecast cost. For each constructed

task, iterative nullspace projection (INLP) [Ravfogel et al., 2020] repeatedly trains a 4-class linear probe on the quartile labels and projects the span of the probe's mean-centered weight rows (at most 3 directions) out of the representation, until a probe retrained on the projected representation reaches at most 30% (chance 25%) or 40 iterations have run. The projection is applied to every token before Layer 2, and the cost is the increase in test MSE over the unmodified model on 200 test windows; the probes are fit on 200 other windows, whose own quartiles set the class boundaries. As a control, the same procedure runs on randomly permuted labels and stops at the same number of removed dimensions, so that, unlike the random-direction control of amnesic probing [Elazar et al., 2021], both removals come from the same probe mechanism. The two removals still take different shares of the representation energy (the summed squared norm of the token representations), so we compare the MSE increase per unit of removed energy.

On ETTm1, Weather, and Electricity ($d_{\text{model}} = 128$; Table 42), erasure removes 33 to 94% of the dimensions; it stays incomplete in the three cells with runs at the iteration cap (ETTm1 volatility and diff_energy, Weather spectral_entropy), where the probe after erasure still reaches 34 to 37%. In these 9 cells, the energy-normalized cost exceeds the control in 6 and falls below it in 3 (Weather diff_energy, Electricity volatility and diff_energy), and the difference exceeds one standard deviation over model seeds only for ETTm1 diff_energy (above the control) and Electricity volatility (below). With $d_{\text{model}} = 16$, each INLP step removes up to 3 of the 16 dimensions, so the ETTh1 rows are coarse and shown for direction only. The test is therefore inconclusive about whether the forecaster uses an individual property: the properties may be spread over many directions, or INLP may remove more than the property itself: LEACE erases the linear information about a k-class label with an edit of rank at most $k-1$ [Belrose et al., 2023], far fewer dimensions than INLP removes here. Minimal-rank erasure is left for future work.

Table 42: Feature-level erasure in frozen PatchTST (P=16) at the Layer-1 output `1b_ffn` (mean over 3 model seeds; $\pm$: sample std of the cost difference). Removed: share of dimensions removed by INLP. Probe: in-sample accuracy (chance 25%) of a 4-class linear probe on the 200 fitting windows, trained for 1,500 steps before and 800 steps after erasure; not comparable to the main-text probing accuracies. Energy: share of representation energy removed by erasure and by the random-label control. Cost difference: MSE increase divided by the removed energy share, erasure minus control.

Dataset (d_{model})	Task	Removed (%)	Probe (%)	Energy (%)	Cost difference
ETTm1 (128)	volatility	94	80.6 → 35.9	52.8 / 32.0	$+0.05_{\pm 0.07}$
ETTm1 (128)	diff_energy	80	62.7 → 36.5	54.8 / 16.4	$+0.24_{\pm 0.16}$
ETTm1 (128)	spectral_entropy	33	76.9 → 30.3	2.2 / 0.6	$+0.22_{\pm 0.26}$
Weather (128)	volatility	66	68.3 → 31.3	12.4 / 3.1	$+0.04_{\pm 0.09}$
Weather (128)	diff_energy	55	62.7 → 31.7	7.9 / 2.2	$-0.11_{\pm 0.17}$
Weather (128)	spectral_entropy	93	66.4 → 33.9	75.6 / 41.4	$+0.06_{\pm 0.09}$
Electricity (128)	volatility	80	82.8 → 29.8	34.3 / 67.9	$-0.15_{\pm 0.01}$
Electricity (128)	diff_energy	84	74.5 → 28.7	44.4 / 72.8	$-0.03_{\pm 0.04}$
Electricity (128)	spectral_entropy	84	64.7 → 29.5	45.6 / 72.5	$+0.02_{\pm 0.08}$
ETTh1 (16)	volatility	88	38.2 → 25.1	67.1 / 72.6	$-0.05_{\pm 0.06}$
ETTh1 (16)	diff_energy	88	43.6 → 29.6	70.8 / 67.4	$-0.04_{\pm 0.08}$
ETTh1 (16)	spectral_entropy	63	40.0 → 29.7	25.9 / 22.7	$+0.14_{\pm 0.03}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and §1 state our three contributions (temporal probing framework, role allocation by tokenization and architecture, and a representational reading of prior results), which are supported in §5, with the reading of prior results in §6. Forecasting-side interventions (activation patching, training without attention, and component knockout) are stated as establishing component-level reliance and sufficiency in the standard PatchTST, not feature-level causal construction; across architectures, knockout is stated only to match the ordering seen in probing, and whether individual probed features are used is stated as open (§8).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: §8 states (i) the reach of the forecasting-side evidence: component-level reliance and sufficiency in PatchTST at P=16, no verdict on the shift seen in probing at P=1, only the ordering across architectures, and an inconclusive feature-level erasure test (Appendix I); (ii) the scope of the allocation: what varies, what does not, which regimes lie outside, and that lookback, depth, and the form of the mixer were not varied; (iii) the limits of the task suite: it is hand-picked to make construction visible, and the main conclusions rest on three second-order statistics of a token's segment (16 points in PatchTST); (iv) the conditions under which the framework applies, with pretrained foundation models left to future work; and (v) the absence of a formal theory: the phenomenon and its boundary conditions are established experimentally, not proven.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper presents an empirical study without formal theorems or proofs. The structural arguments for patch tokens (§5.2) and single-scalar tokens (§5.4) are informal explanations grounded in the architecture, not theoretical results (§8).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The forecasters are trained with each model's official code and scripts (Informer with the implementation in the PatchTST code base); Appendix A gives their architecture-level and per-dataset architectural settings (Tables 5 to 7 and the per-model paragraphs) and how probe samples are drawn, labeled, and split. §3.4 gives the probe training protocol (optimizer, learning rate, batch size, and step count), Appendices G to I the forecasting-side interventions, and §4 the 8 standard public benchmarks with standard splits. The probing results of §5 are 9-run means (3 model $\times$ 3 probe seeds), and the forecasting-side interventions use 3 model seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and the per-run result files are released at `https://github.com/mingyu-11/tokenization-component-roles`, with instructions for training the models, running the probes, and recomputing the tables. All datasets used (ETT, Weather, Electricity, Exchange, Traffic, Solar) are publicly available standard time series forecasting benchmarks, which we use following the PatchTST protocol, and are not redistributed. The paper also gives the probe architecture and training protocol (§3.4), the per-dataset architectural hyperparameters (Appendix A, Tables 5 to 7), the corruption procedure for activation patching (Appendix G), and the seeds (3 model $\times$ 3 probe seeds for the probing results of §5; 3 model seeds for the forecasting-side interventions).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: §4 specifies models, datasets, and ablations. §3.4 reports the probe training protocol (Adam, lr=10^{-3}, batch=32,768, 30k steps, 80/20 split; evaluated every 500 steps, with the best test accuracy over the entire training trajectory reported), and Appendix A describes how probe samples are drawn, labeled, and split. The forecasters are trained with each model's official code and scripts (Informer with the implementation in the PatchTST code base); Appendix A gives their architecture-level and per-dataset architectural settings and the settings used for datasets the scripts do not cover.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 2 and the per-dataset appendix probing tables behind Table 3 and Figure 3 report 9-run means over 3 model seeds $\times$ 3 probe seeds (the probe seed sets the split, the initialization, and the batch draws; Appendix A), with $\pm$ built from 9-run population standard deviations (for deltas, the standard deviations of the two probing points combined as in Table 2; see Appendix B), and the forecasting tables of Appendices H and I report means $\pm$ sample standard deviations over 3 seeds. Statistical tests include paired t-tests with Cohen's d_z for the Layer-1 selectivity gain (§5.1 and Appendix F: $d_z = +1.67$, one-sided $p < 10^{-6}$, pooled over 24 dataset-task pairs) and for activation patching (Appendix G: d_z from -0.85 to $+3.30$ over 300 window-seed pairs and $p < 10^{-16}$ on the 100 seed-averaged windows per dataset, which the test treats as independent although they overlap).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Experiments ran on NVIDIA A100 (40GB), A40 (48GB), and RTX A6000 (48GB) GPUs (Appendix A). Training a forecaster takes from minutes on ETTh1 to over 100 GPU-hours per seed for PatchTST at $P{=}1$ on Traffic, a probing run takes minutes, and activation patching takes under an hour per dataset. All reported results took over 700 GPU-hours, and the full project used more.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: This work uses only publicly available time series forecasting benchmarks and does not involve human subjects, personal data, or sensitive applications. All authors have reviewed the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This paper presents foundational research on the internal workings of time series transformers. It introduces no new datasets, deploys no models, and develops no applications, and we see no direct path from it to negative societal applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out

that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release any models, datasets, or other high-risk artifacts. It analyzes forecasters that we train with the public code of PatchTST (which also provides our Informer implementation), iTransformer, CATS, and ModernTCN on standard forecasting benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Code: PatchTST [Nie et al., 2023] (Apache-2.0), whose code base also provides the Informer [Zhou et al., 2021] implementation we train, and iTransformer [Liu et al., 2024], CATS [Kim et al., 2024], and ModernTCN [Luo and Wang, 2024] (MIT); we train every forecaster ourselves and use no released weights. Data (§4): ETT [Zhou et al., 2021] (CC BY-ND 4.0); Weather [Wu et al., 2021], recorded by the Max Planck Institute for Biogeochemistry, Jena (CC BY 4.0); and, as prepared by Lai et al. [2018], Electricity from the UCI ElectricityLoadDiagrams20112014 data (CC BY 4.0), Traffic from Caltrans PeMS (public domain under its conditions of use), Exchange (no license stated), and Solar-Energy from NREL's Solar Power Data for Integration Studies, for which, as NREL's data terms ask, we credit the U.S. Department of Energy (DOE)/NREL/ALLIANCE. No dataset is redistributed, and the released code keeps each upstream license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code and per-run result files are documented in the repository README, and the repository states the license of our code (MIT) and those of the included upstream code. No new datasets or pre-trained models are released. The repository is `https://github.com/mingyu-11/tokenization-component-roles`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research; no IRB review applies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not a component of the core methodology. The probing framework, ablation experiments, and activation patching analyses use standard linear/MLP probes and intervention techniques without LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.