PRISM: INTERPRETING TIME-SERIES FOUNDATION MODELS VIA ZERO-SHOT BASIS DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series foundation models (TSFMs) generalize across diverse domains, enabling zero-shot forecasting. However, their forecasts do not explicitly identify the contribution of trends and periodic patterns, even though interpretation is a central objective of classical time-series analysis. We introduce PRISM, which decodes a frozen TSFM's representation of the observed context into an additive forecast, fitting nothing to the target series. PRISM predicts a piecewise-linear trend, gates and weights a Fourier dictionary restricted to the periods the sampling interval and horizon admit, and adds a residual contribution, so that the forecast is a sum of components that can be inspected one at a time. On TimesFM 2.5, the components reproduce the calendar and physical structure that domain knowledge predicts in three domains, and where a cycle is present the seasonal components agree with a consensus of classical decompositions of the realized future. Where no cycle exists, as in exchange rates, the seasonal component stays near zero while the decomposers still report one. On strongly seasonal series under input noise, it revises less than the same bases fitted to the backbone's forecast. Among component-wise forecasters, Ours has the lowest error on three of five dataset groups, with dataset-averaged normalized mean absolute error (MAE/σ) 7.15% above the frozen backbone.

1 INTRODUCTION

Time series forecasting underpins a wide range of real-world decisions, from load and demand planning to traffic management and disease surveillance (Hyndman & Athanasopoulos, 2021; Petropoulos et al., 2022). Time-series foundation models (TSFMs) have changed how that prediction is obtained: pretrained on large corpora across domains and sampling rates, models such as TimesFM (Das et al., 2024), Chronos (Ansari et al., 2024), and Moirai (Woo et al., 2024) forecast unseen series zero-shot, with no fitting to the target, at accuracy competitive with supervised models. But they return a trajectory or a predictive distribution, so a user sees what is predicted but not what it is made of. That distinction is what a practitioner acts on: explicit components let a forecast be checked against domain knowledge (Taylor & Letham, 2018), and a rise attributed to trend may warrant a different response than a seasonal peak (Figure 1).

Decomposition is the established way to show what a series is made of: STL (Cleveland et al., 1990) and its successors separate trend, seasonal, and residual variation (Hyndman & Athanasopoulos, 2021). Structural time-series models (Harvey, 1990) and decomposition-based forecasters such as Prophet (Taylor & Letham, 2018) and the interpretable N-BEATS (Oreshkin et al., 2020) return a forecast that is a sum of explicit components, each of which a user can inspect. Their coefficients, however, are estimated from the target data. Prophet is refitted to every series with its periods and trend form fixed in advance; N-BEATS is retrained for every dataset and horizon over bases the user fixes beforehand. The two traditions have therefore split the two properties a practitioner wants: explicit components without generality, or generality without explicit components.

Existing bridges between the two obtain the components outside the model, and neither reads the model's own representation. Decomposing the input hands the backbone a decomposition instead of reading one from it (Cao et al., 2024). Decomposing the forecast a frozen TSFM returns, as DEFT does for candidate trajectories (Le et al., 2026), keeps zero-shot use, but derives the components from the generated output rather than reading them from the model. They are then defined relative

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

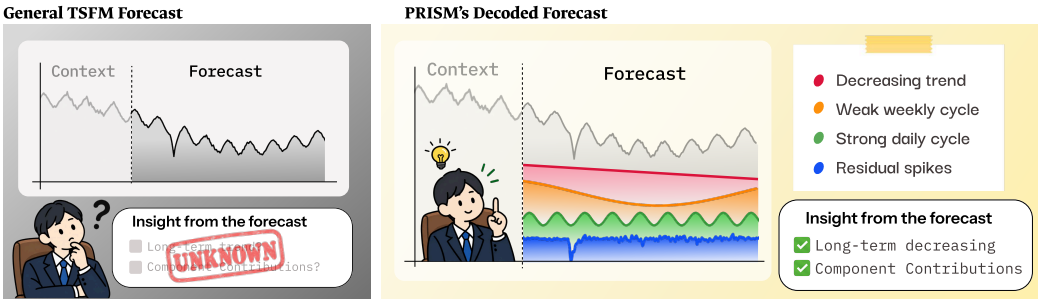

Figure 1: **Reading a forecast by its components tells a practitioner what drives it.** A TSFM alone returns only the forecast, so the reader cannot tell what moves it. PRISM returns the forecast together with its trend, weekly and daily seasonal, and residual components, read from the frozen backbone with nothing fitted to the series. Here the components show a declining trend under recurring weekly and daily cycles, with the one sharp drop assigned to the residual.

to one candidate trajectory, not read as structure the model attributes to the series. Yet a TSFM that forecasts a weekly cycle correctly should already carry that cycle in its representation of the context, where concepts such as trend and periodicity have been found (Wiliński et al., 2025). Finding a concept, however, is not the same as forecasting with it. We therefore ask: *can the components of a TSFM forecast be read from the model itself, at inference time and without fitting to the target series, and at what cost in accuracy?*

We answer it with **PRISM**, which uses **Pretrained Representations** for **Interpretable** time **Series Modeling** through *zero-shot basis decoding*. PRISM keeps the TSFM backbone frozen and replaces its forecasting head with three lightweight decoders that read the backbone's representation of the observed context. The trend decoder returns a piecewise-linear function, the seasonal decoder returns coefficients and gates over a dictionary of daily, weekly, monthly, and yearly Fourier bases, and the residual decoder returns what the two explicit bases leave out; their sum is the forecast, so every component can be inspected. The decoders are trained once per horizon, across domains, and are then applied unchanged. The same predefined trend family and Fourier dictionary are shared across datasets, without target-specific fitting. Because forecast error alone does not determine how variation splits among the components, synthetic series with known components supervise their roles alongside the forecast loss on real series, and the residual decoder is trained last.

We instantiate PRISM on TimesFM 2.5 (Google Research, 2025) and evaluate its accuracy on five dataset groups spanning electricity consumption, traffic, solar power, and transformer measurements, comparing PRISM with the frozen backbone and six component-wise forecasters, including Prophet and N-BEATS-I. We first assess the plausibility of the components using domain knowledge and classical statistical decompositions. The components reproduce the calendar and physical structure each domain is known to have. The seasonal components agree closely with a consensus of classical decompositions of the realized future, including STL, MSTL (Bandara et al., 2025), and Prophet, wherever a cycle exists.Where none exists, as on exchange rates, the decomposers still report a yearly cycle while ours stays near zero. PRISM's MAE/σ, averaged equally across the five dataset groups, is 7.15% higher than the backbone's. Decoding is stable: post-hoc fits of the same bases revise more under noise than the decoded ones, and on all four trend-dominated datasets Ours has lower median error than the backbone over the evaluation windows at $H = 336$ and 720.

In summary, our main contributions are:

- **Interpretable forecasting from a frozen TSFM.** Components that decomposition-based forecasters fit to the target are read instead from a frozen backbone's representation, zero-shot, and they match domain priors and classical decompositions, staying near zero where no cycle exists.
- **PRISM, a zero-shot basis decoder.** A gated Fourier dictionary over admissible periods, a piecewise-linear trend, and a residual decoder trained only after the other two are fixed, which gives the explicit bases the first opportunity to explain the forecast.
- **Best among component-wise forecasters, and more stable than post-hoc fits.** Lowest error on three of five dataset groups, with average error 7.15% above the backbone's; seasonal components revising less under noise than post-hoc fits, and lower median error than the backbone on all four trend-dominated datasets over the evaluation windows.

2 Related Work

Time-series foundation models. Following large-scale language-model pretraining, TSFMs such as TimesFM and Chronos forecast unseen datasets zero-shot (Das et al., 2024; Ansari et al., 2024). MOMENT serves several downstream tasks, including forecasting with a linear head over a frozen backbone (Goswami et al., 2024). Pretrained forecasters are adapted to new data through learned adapters or related series supplied in context (Benechehab et al., 2025; Faw et al., 2025). Other work calibrates predictive uncertainty, attributes forecasts to past observations and covariates, or refines forecast components with expert feedback (Achour et al., 2025; Hertel et al., 2026; Le et al., 2026). Representation analyses find concepts such as trend and periodicity inside TSFMs and steer them (Wiliński et al., 2025), but they do not return them as components of the forecast. Freezing a backbone and training a module on it is established practice; what distinguishes PRISM is that the module outputs a trend, a seasonal, and a residual contribution whose sum is the forecast.

Statistical decomposition. STL separates a series into trend, seasonal, and residual variation, and MSTL extends it to multiple seasonal periods (Cleveland et al., 1990; Bandara et al., 2025; Hyndman & Athanasopoulos, 2021). Such procedures describe an observed series rather than predict a future one. Generalized additive models carry the same structure into prediction by modeling a transform of the conditional mean as a sum of functions of the inputs (Hastie & Tibshirani, 1986). Structural time-series models express the components as latent states estimated with a Kalman filter (Harvey, 1990; Scott & Varian, 2014). Prophet fits a piecewise trend, Fourier seasonality, and holiday effects to the target series and extrapolates them (Taylor & Letham, 2018); NeuralProphet adds autoregressive and covariate modules (Triebe et al., 2021). These models estimate their components from the series being forecast; PRISM predicts them with decoders trained once across domains.

Deep decomposition and basis expansion. Autoformer, FEDformer, and DLinear build a series decomposition into the architecture (Wu et al., 2021; Zhou et al., 2022; Zeng et al., 2023). DEPTS models periodic structure explicitly, and ETSformer predicts level, growth, and seasonal components (Fan et al., 2022; Woo et al., 2022). The interpretable variant of N-BEATS (N-BEATS-I) is closest to PRISM in form: it predicts coefficients for fixed trend and seasonal bases (Oreshkin et al., 2020). In their standard supervised setting, these models require training on the target dataset. Named components also do not guarantee that each captures its intended structure, since an accurate aggregate forecast can coexist with poorly separated trend and seasonality (Woo et al., 2022). PRISM keeps its trained decoder parameters fixed across evaluation datasets. We assess its component separation through agreement with domain priors and classical decompositions.

Decomposition and bases with pretrained models. DAM trains its own backbone to output the coefficients of a continuous-time basis (Darlow et al., 2024). TEMPO and S^2IP-LLM use input decomposition when adapting pretrained language models, through additive component forecasting and component-based tokenization, respectively (Cao et al., 2024; Pan et al., 2024). PRISM instead applies structured decoders to the context representation of an existing frozen TSFM, generating interpretable forecast components without decomposing the input in advance.

3 PRISM: Zero-shot Basis Decoding for TSFMs

We consider univariate point forecasting: given an observed context $x_{1:L}$ collected at a known interval Δt (e.g., one hour), the task is to predict the next H values, $\hat{y}_{L+1:L+H}$. Following classical additive decomposition (Hyndman & Athanasopoulos, 2021), we express each prediction as $\hat{y}_{L+h} = \mu_x + \hat{T}_h + \hat{S}_h + \hat{R}_h$ for $h = 1, \ldots, H$, where μ_x is the context mean, kept separate from the predicted trend, seasonal, and residual contributions $\hat{T}_h$, $\hat{S}_h$, and $\hat{R}_h$. Section 3.1 describes simple decoders, inspired by Prophet's additive formulation (Taylor & Letham, 2018), that map a frozen TSFM's context embedding to these components for component-level interpretation. Section 3.2 explains how we train the component decoders without ground-truth decompositions of real time series.

3.1 Basis decoder architecture

Frozen TSFM embedding. We retain the backbone's context encoding and input normalization and replace its forecasting head with the component decoders (Figure 2). The component contributions and context-mean offset together form PRISM's own forecast, which is not constrained to reproduce the original head's. Our implementation uses TimesFM 2.5 (Das et al., 2024; Google Research, 2025), taking the representation of its final input patch as e.

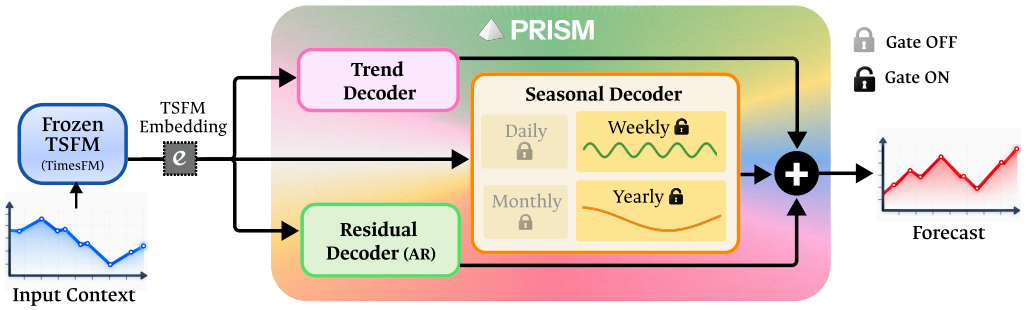

Figure 2: **PRISM decodes a frozen TSFM's embedding into additive components.** A frozen TimesFM 2.5 encoder maps the observed context (up to 512 points) to an embedding, which three decoders read. The trend decoder returns a piecewise-linear trend. The seasonal decoder weights Fourier atoms (one sine–cosine pair per admissible harmonic) within the daily, weekly, monthly, and yearly period families and gates each atom on or off individually; the diagram simplifies this to one representative atom per family, and of the four shown, daily and monthly are gated off. Conditioned on the trend and seasonal predictions, the residual decoder generates its output autoregressively. The three component contributions are rescaled and summed, then the context mean is restored as a separate offset. Each contribution can be inspected separately.

The trend and seasonal decoders generate the full horizon from this representation. The residual decoder uses their predictions together with the backbone's context state. The computation is summarized as

$$e = \mathrm{Enc}(x_{1:L}), \quad \widetilde{T} = \mathcal{B}_T(e), \quad \widetilde{S} = \mathcal{B}_S(e; \Delta t), \quad \widetilde{R} = \mathrm{AR}(x_{1:L}, \widetilde{T}, \widetilde{S}),$$
$$\hat{y} = \mathrm{Denorm}_{x_{1:L}}\left(\widetilde{T} + \widetilde{S} + \widetilde{R}\right), \tag{1}$$

where $\mathcal{B}_T$ and $\mathcal{B}_S$ denote the trend and seasonal basis decoders. AR generates the residual forecast, using autoregressive feedback when additional output patches are needed. Let μ_x and σ_x be the context mean and standard deviation used by normalization, so $\mathrm{Denorm}_{x_{1:L}}(z) = \mu_x + \sigma_x z$. The original-scale component contributions are $\hat{T}_h = \sigma_x \widetilde{T}_h$, $\hat{S}_h = \sigma_x \widetilde{S}_h$, and $\hat{R}_h = \sigma_x \widetilde{R}_h$; the context mean remains a separate offset. Thus, $\hat{y}_{L+h} = \mu_x + \hat{T}_h + \hat{S}_h + \hat{R}_h$. The backbone remains frozen throughout training and inference, and the trained decoders are applied without fitting them to each target series.

Trend decoder. We represent the trend as a continuous piecewise-linear function over normalized horizon coordinates. A small number of interior knots limits the number of slope changes, restricting the trend's flexibility to follow repeated oscillations. The decoder predicts knot locations $\kappa_k(e)$ and coefficients from the context embedding:

$$\widetilde{T}_h = \alpha_0(e) + \alpha_1(e) u_h + \sum_{k=1}^{K} \delta_k(e) \big(u_h - \kappa_k(e)\big)_+. \tag{2}$$

Here, $u_h = (h-1)/(H-1)$ is the normalized horizon coordinate, K is the number of interior knots, and $(z)_+ = \max(z, 0)$. In this piecewise-linear parameterization, α_0, α_1, and δ_k describe the initial trend level, initial slope, and changes in slope at the knots.

Seasonal decoder. We use Fourier bases to make each recurring period's contribution explicit within the seasonal forecast. Many weak periodic terms can yield similar forecasts while obscuring which recurring patterns shape them. We therefore use a broad Fourier dictionary that includes periods suggested by prior knowledge, such as daily cycles. Learned gates and a sparsity penalty encourage the forecast to rely on a few prominent periodic contributions.

For each admissible period P_j, the decoder predicts raw coefficients $(\bar{a}_j, \bar{b}_j)$ and a continuous gate $g_j \in (0, 1)$ from the context embedding e. Their product gives the effective coefficients:

$$(a_j, b_j) = g_j \, \mathbf{1}[g_j \geq \tau] \, (\bar{a}_j, \bar{b}_j). \tag{3}$$

Gates below the fixed threshold τ suppress their Fourier pairs; retained gates scale both coefficients. The seasonal forecast is

$$\widetilde{S}(t) = \sum_{j \in \mathcal{J}} \left[a_j \cos\left(\frac{2\pi t}{P_j}\right) + b_j \sin\left(\frac{2\pi t}{P_j}\right) \right]. \tag{4}$$

Here, t measures time elapsed since the first context observation; at forecast step h, $t = (L + h - 1)\Delta t$. The effective coefficients determine the amplitude and phase of each retained period. The admissible set $\mathcal{J} = \mathcal{J}(\Delta t, H)$ depends on the sampling interval and horizon. A deterministic mask excludes periods too short to be resolved at the sampling interval, retaining $P_j > 2\Delta t$ under the Nyquist criterion (Shannon, 1949). The trend knot configuration and coefficient parameterization, together with the full admissibility rules, harmonic sets, and gating details, are given in Appendix A.

Residual decoder. The residual decoder uses a free-form head to predict residual values directly over the forecast horizon. We train this head with the residual-MLP structure of TimesFM's original forecasting head (Google Research, 2025). At each step, it predicts a 128-step residual patch from the current backbone representation and the corresponding trend and seasonal forecast patches. If another output patch is needed, we feed the total forecast for the current patch into the frozen backbone to obtain the next representation. The full-horizon trend and seasonal curves remain fixed throughout the rollout. This flexibility also allows $\widetilde{R}$ to absorb variation represented by $\widetilde{T}$ and $\widetilde{S}$. The staged training in Section 3.2 is designed to limit unnecessary reliance on $\widetilde{R}$ while preserving predictive accuracy, by fitting the explicit bases first.

3.2 Sequential training procedure

Synthetic data have been used to broaden TSFM training distributions when real data offer limited coverage (Das et al., 2024; Ansari et al., 2024). We generate synthetic series from explicit bases primarily to supervise decomposition. Known trend curves supervise $\widetilde{T}$, and the generating Fourier coefficients directly supervise the effective coefficients of $\widetilde{S}$. The seasonal targets thus distinguish individual periodic contributions rather than only their summed curve. Although the synthetic distribution is limited, we repeatedly interleave these series with real data during decoder training. The synthetic targets guide the separation of trend from seasonality and the resolution of superposed periodic contributions. Forecast losses on real series penalize differences between the summed component predictions and future observations.

With the TSFM frozen, we combine prediction losses for forecast accuracy with sequential training and penalties for component separation. We first train the trend decoder and then freeze it while training the seasonal decoder to reduce the prediction error of $\widetilde{T} + \widetilde{S}$. Correlation and overlap penalties discourage redundancy between $\widetilde{T}$ and $\widetilde{S}$, while a penalty on soft gate values encourages sparsity for interpretability. Finally, with both basis decoders fixed, we train $\widetilde{R}$ to reduce the prediction error of the complete forecast on real series. The training schedule is given in Appendix A.

4 Experiments

We evaluate whether our model's zero-shot forecasts provide interpretable component structure. Because the practical value of these interpretations depends on forecasting accuracy, we also compare total-forecast performance with the original backbone and baselines that provide forecast components. The benchmarks in Sections 4.2 and 4.3 were used in neither TimesFM pretraining nor our decoder training. Throughout, *Ours* denotes our model instantiated on TimesFM.

- **Alignment with domain priors (Section 4.1).** Whether the components show the calendar and physical structure each domain is known to have.
- **Agreement with statistical decompositions (Section 4.2).** Whether seasonal components match a consensus of four statistical decompositions of the realized future.
- **Zero-shot forecasting performance (Section 4.3).** Forecasting accuracy relative to the backbone and baselines.
- **Stability of basis decoding (Section 4.4).** Component revision under input noise compared with post-hoc decompositions of the backbone forecast, and accuracy on trend-dominated series.

4.1 Alignment with domain priors

This section qualitatively evaluates whether the components PRISM predicts agree with domain priors (metadata or well-known facts) across three domains. We select data to cover human activity and biological seasonality, including daily, weekly, and yearly cycles. Dataset details and the full density analysis for influenza are given in Appendix B.

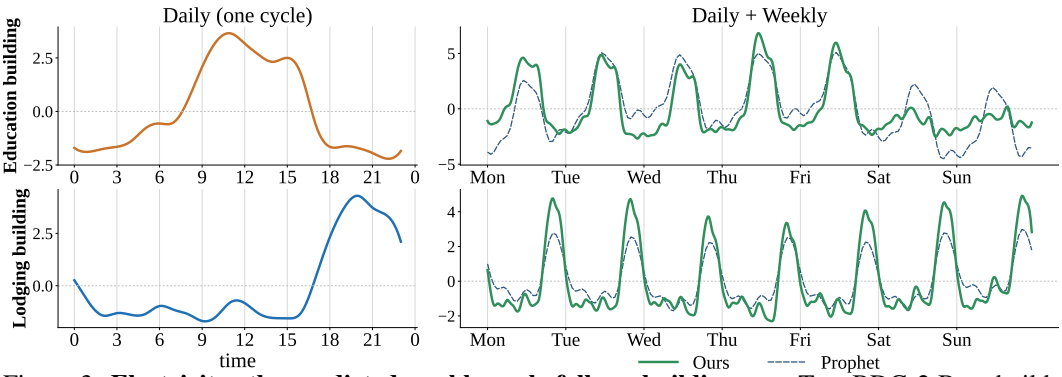

Figure 3: **Electricity: the predicted weekly cycle follows building use.** Two BDG-2 Bear buildings, one per row. Each row sets the daily component over one cycle beside the sum of the daily and weekly components over one week, where a Prophet fit to the same context is overlaid (solid: ours, dashed: Prophet). The education building peaks near 11:00 and flattens on weekends; the lodging building peaks near 20:00 every day. Prophet does not separate the two weekend patterns.

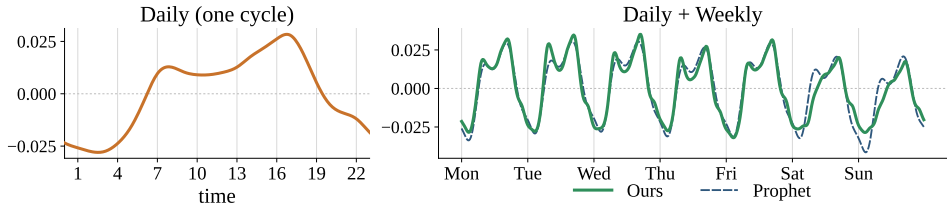

Figure 4: **Traffic: weekday rush-hour peaks flatten on both weekend days.** `traffic_hourly` sensor T837 (Monash; distinct from the `M_DENSE` Traffic row in Table 1). The daily component over one cycle sits beside the sum of the daily and weekly components over one week (Monday–Sunday), where a Prophet fit to the same context is overlaid. Ours drops 15–26% on both weekend days; Prophet's weekend and weekday amplitude ranges overlap.

Electricity. Buildings in `BDG-2 Bear` are expected to vary in electricity load by function, with education buildings showing stronger weekday daytime demand and lodging/residential buildings retaining evening demand throughout the week. As shown in Figure 3, the daily and weekly patterns identified by our model reflect these characteristics. The education building peaks around 11:00, showing strong weekday daytime demand but little weekend elevation. The lodging building peaks around 20:00, with sustained evening demand and little weekday–weekend variation. This contrast is not particular to the two buildings drawn: it holds across all 70 labelled buildings at this site.

Prophet does not clearly capture the weekend flattening; its fit instead shifts the overall pattern downward or reduces its amplitude.

Traffic. Commuting produces morning and evening peaks in weekday road occupancy, with reduced peaks on both weekend days. We predict the daily and weekly Fourier components for `traffic_hourly` sensor T837 (Figure 4), the San Francisco Bay Area road-occupancy benchmark, distinct from the `M_DENSE` Traffic row in Table 1. The predicted daily cycle shows two peaks, a smaller morning peak near 08:00 and a stronger evening peak near 17:00, consistent with commuting patterns. In the combined daily-plus-weekly component, both weekend days are 15–26% below the weekday range. Prophet's weekend amplitude range (0.053–0.062) overlaps its weekday range (0.057–0.063).

ILINet. Influenza-like illness (ILI) is expected to exhibit a yearly cycle associated with the winter flu season. Higher population density has been associated with earlier outbreak timing in county-level COVID-19 cases (Megahed et al., 2021), so we compare densely populated California with sparsely populated Wyoming.

Figure 5 shows the expected winter peak in both states, and the denser state peaks earlier (California: January) than the sparser state (Wyoming: February), consistent with the reported density–timing association; across the 51 state-level jurisdictions with a usable context and a known population figure, this density–timing correlation is weak but significant for our predicted peaks ($r = -0.289$, $p = 0.040$), where the same analysis on Prophet's peaks is not ($r = -0.191$, $p = 0.180$). Real outbreaks also vary in size and timing from year to year, so a fixed calendar cycle cannot fully

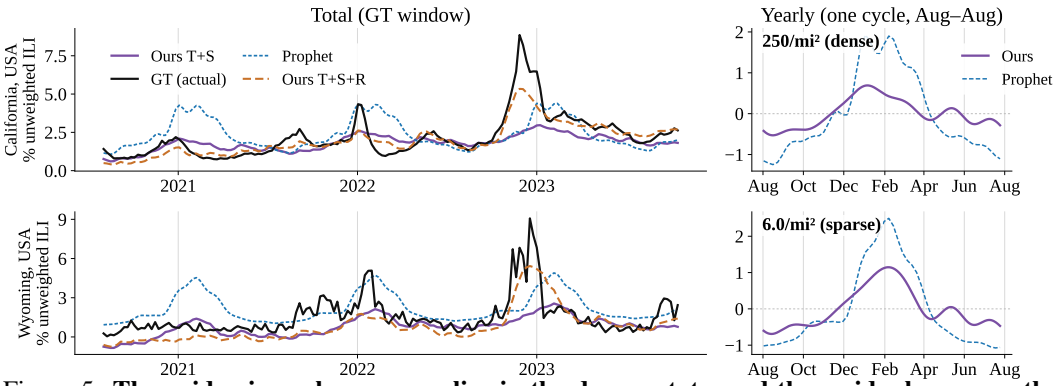

Figure 5: **The epidemic peak comes earlier in the denser state, and the residual recovers the outbreaks the trend and seasonal components smooth over.** For California (dense) and Wyoming (sparse), the total panels compare the observed ILI over 168 weeks with our T+S (trend+seasonal), our T+S+R (total prediction), and a Prophet fit, all from the same 512-week context; the yearly panels show the predicted yearly component over one cycle (solid: ours, dashed: Prophet). Adding the residual closes most of the gap between the smooth trend-plus-seasonal forecast and the sharp outbreak peaks Prophet also misses; California peaks in January, Wyoming in February.

describe them. The trend and seasonal components together (T+S) stay smooth through the sharp winter surges. Adding the residual (T+S+R) recovers these non-repeating variations and tracks the observed 168-week record more closely than Prophet's fit to the same 512-week context.

4.2 Agreement with Statistical Decompositions

Domain priors alone may be insufficient to assess the plausibility of predicted components. Classical decomposition methods offer complementary references for identifying temporal structure under different assumptions. Although these statistical reference methods can yield different components, substantial disagreement with seasonal structure on which they agree would weaken support for the plausibility of a predicted seasonal component. We therefore use their consensus as a reference.

We compare our model with N-BEATS-I (Oreshkin et al., 2020), DLinear (Zeng et al., 2023), ETSformer (Woo et al., 2022), Autoformer (Wu et al., 2021), and TEMPO (Cao et al., 2024), selected for their explicit forecast components. Like ours, these models provide trend and seasonal components at prediction time and form the total forecast by summing their component outputs. Prophet (Taylor & Letham, 2018) serves as a statistical reference here and as a forecasting baseline in Section 4.3.

For each window, we first apply STL (Cleveland et al., 1990), MSTL (Bandara et al., 2025), SSA (Golyandina et al., 2001), and Prophet (Taylor & Letham, 2018) to the 512-step context concatenated with the realized future. We retain the eligible seasonal components over the forecast horizon and take their pointwise median as the consensus.

For each forecasting model, we divide its predicted seasonal component into cycles of the longest reference period. Averaging values at the same position across these cycles gives one representative cycle. We repeat this cycle over the forecast horizon and compute its Pearson correlation with the consensus. Dataset scores are median correlations across windows, averaged over three training seeds for trained baselines. The leftmost category measures agreement among statistical reference methods by comparing each method's seasonal component, averaged by phase in the same way, with the consensus of the other methods. Reference-selection rules and implementation details are given in Appendix C.

The statistical reference methods show substantial agreement across the seven datasets, with a mean leave-one-out correlation of 0.884 (Figure 6, left). Our model's zero-shot seasonal predictions achieve the highest mean agreement among the forecasting models, with a mean correlation of 0.841 compared with 0.833 for N-BEATS-I. Our model obtains these results without adaptation on benchmarks used in neither TimesFM pretraining nor our decoder training. This agreement suggests

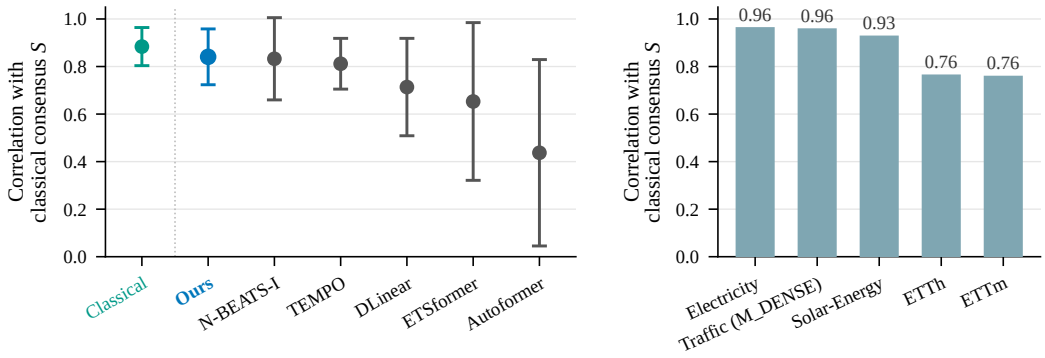

Figure 6: **Our model's seasonal components agree with statistical methods.** Left: mean ± one standard deviation across seven datasets. Classical measures agreement among statistical methods, comparing each with the consensus of the others; forecasting models use the full four-method consensus. Right: Ours by dataset, averaging ETTh1/2 and ETTm1/2.

Table 1: **PRISM inherits its backbone's generalization capability.** Values are MAE in the normalized space of each input window (lower is better). Parentheses give ΔMAE (our model minus TimesFM) in the same units. Bold and underline mark the two lowest errors, ranked before rounding. Dataset rows pool horizons by usable-window counts; Average weights the five dataset rows equally.

Dataset	Ours (Δ TimesFM)	TEMPO	Prophet	N-BEATS-I	DLinear	ETSformer	Autoformer
Electricity	**0.342** (+0.015)	0.376	0.557	<u>0.354</u>	0.354	0.560	0.510
Traffic	<u>0.308</u> (+0.021)	0.356	0.518	**0.287**	0.328	0.464	0.393
Solar-Energy	0.425 (+0.095)	0.529	0.845	**0.325**	<u>0.381</u>	0.555	0.672
ETTh	**0.813** (+0.021)	<u>0.823</u>	1.256	1.144	0.974	1.107	1.052
ETTm	**0.831** (+0.030)	<u>0.831</u>	1.339	0.927	0.888	1.178	1.310
Average	**0.544** (+0.036)	<u>0.583</u>	0.903	0.607	0.585	0.773	0.787

ETTh and ETTm each average datasets 1 and 2 equally. Traffic denotes `M_DENSE`. Trained baselines report means over seeds 2021, 2022, and 2023; other models are evaluated once. Per-dataset, horizon-level means and standard deviations are in Table 8.

that our model provides plausible seasonal components consistent with those identified by classical statistical methods.

4.3 Zero-shot forecasting performance

Table 1 reports forecasting accuracy with one decoder set (trend, seasonal, residual) per horizon applied across target series. The evaluation protocol is detailed in Appendix D. Prophet is fitted to the observed context at each evaluation window. N-BEATS-I, DLinear, ETSformer, and Autoformer are trained separately for each target dataset and horizon. TEMPO is evaluated without fine-tuning. Among these forecasting baselines, Ours has the lowest error on three of the five dataset groups, with a modest average increase of 0.036 in MAE/σ relative to TimesFM (0.544 versus 0.508; 7.1%). In an auxiliary ablation, removing the residual output raises normalized MAE by 0.044 (8.1%), suggesting that the compact trend and seasonal basis representation retains much of the model's forecasting accuracy.

4.4 Stability of basis decoding

Input noise. Post-hoc decomposition fits a model to the backbone forecast to extract its components. Such fitting may overfit noise propagated into the forecast, whereas predicting components directly from the context representation may yield a more stable decomposition. Table 2 compares these approaches under Gaussian noise injected into the observed context. We use four variance ratios, $v \in \{0.06, 0.10, 0.16, 0.25\}$, with noise standard deviation $\sqrt{v}\,\sigma$, context length 512, and horizon 336. Input normalization and context encoding are recomputed for every perturbed input, while the target future remains unchanged. OLS_H fits the decoder's trend and Fourier bases to the TimesFM forecast over the horizon alone; OLS_{L+H} fits the same bases over the context concatenated with the

Table 2: **Decoded seasonal components revise less under noise than fitted ones.** Revision of the seasonal component under injected noise, in percent of the clean-context standard deviation, macro-averaged over three hourly datasets at four variance ratios v. Ours is lowest at every v, and the gap to the post-hoc fits widens with the noise.

v	Ours	OLS_H	OLS_{L+H}	MSTL
0.06	**6.39**	6.82	7.06	8.41
0.10	**7.74**	8.20	8.93	10.34
0.16	**9.17**	9.69	11.06	12.50
0.25	**10.80**	11.34	13.54	14.92

Table 3: **Ours beats the backbone on trend-dominated datasets.** Across evaluation windows at $H \in \{336, 720\}$, relative error is the relative change in dataset-median MAE/σ (%) from TimesFM to Ours. Win is the percentage of window–horizon pairs where our model has lower error. Trend is the median fraction of input variance explained by a straight line over the same pairs.

Dataset	Rel. err. (%)	Win (%)	Trend
M4 Weekly	−8.6	68.4	0.65
Exchange	−2.0	52.0	0.48
M4 Daily	−2.8	53.8	0.53
Bitcoin	−6.8	66.6	0.38

forecast. MSTL applies its multi-seasonal-trend decomposition over the latter support using eligible calendar periods.

We measure the mean absolute revision of the seasonal component relative to its clean-input prediction, normalized by the clean-context standard deviation, on three hourly datasets with strong seasonal structure. Ours produces smaller seasonal revisions across all three datasets at each displayed noise level, with one tie at two decimals. Its macro-averaged stability advantage widens in absolute terms as v increases, particularly relative to OLS_{L+H} and MSTL. Our model's total forecast remains less accurate than TimesFM on the same perturbed inputs. The ratio of dataset-averaged MAEs decreases over the displayed noise levels ($v = 0.06$–0.25); relative to clean inputs, the error gap narrows from 8.57% to 6.31% at $v = 0.25$, a reduction of 2.26 percentage points.

Trend-dominated series. TimesFM generates forecasts autoregressively, conditioning later patches on earlier predictions. We hypothesize that this feedback can reinforce patterns introduced early in the forecast, whereas PRISM's simple trend basis captures the global trajectory with few slope changes and may thereby reduce overall MAE. PRISM's residual decoder also uses autoregressive feedback, but targets variation left unexplained by the fixed trend and seasonal forecasts, which may help limit error accumulation. Across evaluation windows at $H \in \{336, 720\}$ (Table 3), Ours reduces median MAE/σ by 2.0–8.6% across all four datasets relative to TimesFM, with window–horizon win rates of 52.0–68.4%.

5 Conclusion

We asked whether the component structure that classical time-series analysis obtains by fitting each series can be read instead out of a pretrained forecaster's representation of the observed context. PRISM answers it with three decoders over a frozen TimesFM: a piecewise-linear trend, a gated Fourier dictionary restricted to the admissible periods, and an autoregressive residual decoder trained after the basis decoders are fixed. Synthetic component supervision guides the separation of trend and seasonality, and one trained decoder set per horizon is applied across target series without parameter updates. The decoded components align with domain priors in the examined examples, and their seasonal components achieve the highest mean agreement in shape with the statistical consensus among the evaluated forecasting models. PRISM has the lowest error on three of five dataset groups among component-wise forecasters, while keeping the increase in average MAE/σ relative to the backbone to 7.15%. On three hourly datasets, its seasonal components show smaller average revisions under input noise than the evaluated context-inclusive post-hoc decompositions. PRISM also has lower median forecast error than the backbone on the selected trend-dominated datasets.

Limitations. PRISM depends on a dictionary of calendar periods and their harmonics and on a trend with few slope changes, with variation outside these bases handled by the residual. Our evaluation uses one backbone, and the separate contributions of backbone pretraining, decoder supervision, and training order remain to be isolated.

AI USE STATEMENT

In this work, we used generative AI tools to improve the readability, grammar, and consistency of the manuscript and to translate some author-written Korean text into English. We also used these tools to assist with writing code.

The research problem, core hypotheses, and conceptual design of the methodology and main experiments were developed by the authors without generative AI. All AI-assisted outputs were reviewed and verified by the authors. Reported results were checked against stored experiment outputs, checkpoint identities were verified by hashes, AI-assisted code was inspected and validated, synthetic data were checked for the intended component structure, references were verified against the original sources, and mathematical expressions were reviewed directly by the authors. The authors take full responsibility for the final manuscript, its claims, and the submitted artifacts.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our methods and experiments throughout the main paper and appendix. Sections 3.1–3.2 describe the PRISM architecture and sequential training procedure, while Appendix A provides detailed decoder configurations, basis construction, synthetic-data supervision, training schedules, and checkpoint selection. Appendices B–C document the datasets, example-selection procedures, statistical-decomposition references, window-selection criteria, and component-evaluation protocols. Appendix D specifies the forecasting datasets and splits, evaluation windows and horizons, baseline configurations and training procedures, random seeds, checkpoint selection and verification, and scoring and aggregation protocols. An anonymized implementation and the configurations used for the experiments are available at https://anonymous.4open.science/r/PRISM_review-00.

REFERENCES

Sami Achour, Yassine Bouher, Duong Nguyen, and Nicolas Chesneau. Foundation models for time series forecasting: Application in conformal prediction. *arXiv preprint arXiv:2507.08858*, 2025. doi: 10.48550/arXiv.2507.08858. URL https://arxiv.org/abs/2507.08858.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research (TMLR)*, 2024.

Kasun Bandara, Rob J Hyndman, and Christoph Bergmeir. MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *International Journal of Operational Research*, 52(1):79–98, 2025. doi: 10.1504/IJOR.2025.143957.

Abdelhakim Benechehab, Vasilii Feofanov, Giuseppe Paolo, Albert Thomas, Maurizio Filippone, and Balázs Kégl. AdaPTS: Adapting Univariate Foundation Models to Probabilistic Multivariate Time Series Forecasting. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/benechehab25a.html.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. TEMPO: Prompt-based generative pre-trained transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YH5w12OUuU.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990. URL https://www.scb.se/contentassets/ca21efb41fee47d293bbee5bf7be7fb3/stl-a-seasonal-trend-decomposition-procedure-based-on-loess.pdf.

Luke Darlow, Qiwen Deng, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Artjom Joosen, Adam Barker, and Amos Storkey. DAM: Towards a foundation model for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=4NhMhElWqP.

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

Wei Fan, Shun Zheng, Xiaohan Yi, Wei Cao, Yanjie Fu, Jiang Bian, and Tie-Yan Liu. DEPTS: Deep expansion learning for periodic time series forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AJAR-JgNw__.

Matthew Faw, Rajat Sen, Yichen Zhou, and Abhimanyu Das. In-Context Fine-Tuning for Time-Series Foundation Models. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/faw25b.html.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Nina Golyandina, Vladimir Nekrutkin, and Anatoly A Zhigljavsky. *Analysis of Time Series Structure: SSA and Related Techniques*. Chapman & Hall/CRC, 2001. doi: 10.1201/9781420035841.

Google Research. TimesFM 2.5: Official release documentation. Official TimesFM repository, release notes dated September 15, 2025, 2025. URL https://github.com/google-research/timesfm#update---sept-15-2025.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A Family of Open Time-series Foundation Models. In *International Conference on Machine Learning (ICML)*, pp. 16115–16152, 2024. URL https://arxiv.org/abs/2402.03885.

Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3): 297–310, 1986. doi: 10.1214/ss/1177013604. URL https://doi.org/10.1214/ss/1177013604.

Matthias Hertel, Alexandra Nikoltchovska, Sebastian Pütz, Benjamin Schäfer, Ralf Mikut, and Veit Hagenmeyer. Explainable load forecasting with covariate-informed time series foundation models. In *Proceedings of the 17th ACM International Conference on Future and Sustainable Energy Systems (e-Energy)*, pp. 612–626, 2026. doi: 10.1145/3744255.3811724. URL https://doi.org/10.1145/3744255.3811724.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3 edition, 2021. URL https://otexts.com/fpp3/.

Hung Le, Minh Hoang Nguyen, Manh Nguyen, Huu Hiep Nguyen, and Dai Do. Expert-Guided Forecast Editing for Time-Series Foundation Models. *arXiv preprint arXiv:2607.19659*, 2026. URL https://arxiv.org/abs/2607.19659.

Fadel M Megahed, L Allison Jones-Farmer, Longwen Zhao, and Steven E Rigdon. Modeling the differences in the time-series profiles of new COVID-19 daily confirmed cases in 3,108 contiguous U.S. counties: A retrospective analysis. *PLOS ONE*, 16(11):e0242896, 2021. doi: 10.1371/journal.pone.0242896.

Clayton Miller, Anjukan Kathirgamanathan, Bianca Picchetti, Pandarasamy Arjunan, June Young Park, Zoltan Nagy, Paul Raftery, Brodie W. Hobson, Zixiao Shi, and Forrest Meggers. buds-lab/building-data-genome-project-2: v1.0, 2020a. URL https://doi.org/10.5281/zenodo.3887306.

Clayton Miller, Anjukan Kathirgamanathan, Bianca Picchetti, Pandarasamy Arjunan, June Young Park, Zoltan Nagy, Paul Raftery, Brodie W. Hobson, Zixiao Shi, and Forrest Meggers. The building data genome project 2, energy meter data from the ASHRAE great energy predictor III competition. *Scientific Data*, 7:368, 2020b. doi: 10.1038/s41597-020-00712-x. URL https://doi.org/10.1038/s41597-020-00712-x.

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

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. S^2IP-LLM: Semantic space informed prompt learning with LLM for time series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39135–39153. PMLR, 2024. URL https://proceedings.mlr.press/v235/pan24c.html.

Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K. Barrow, Souhaib Ben Taieb, et al. Forecasting: theory and practice. *International Journal of Forecasting*, 38(3):705–871, 2022. doi: 10.1016/j.ijforecast.2021.11.001. URL https://arxiv.org/abs/2012.03854.

Steven L Scott and Hal R Varian. Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1-2):4–23, 2014.

Claude E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi: 10.1109/JRPROC.1949.232969. URL https://doi.org/10.1109/JRPROC.1949.232969.

Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. doi: 10.1080/00031305.2017.1380080.

Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir, and Ram Rajagopal. NeuralProphet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397*, 2021. URL https://arxiv.org/abs/2111.15397.

Michał Wiliński, Mononito Goswami, Willa Potosnak, Nina Żukowska, and Artur Dubrawski. Exploring representations and interventions in time series foundation models. In *International Conference on Machine Learning (ICML)*, pp. 66861–66886, 2025.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. *arXiv preprint arXiv:2202.01381*, 2022. URL https://arxiv.org/abs/2202.01381.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified Training of Universal Time Series Forecasting Transformers. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.02592.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.13008.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317.

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

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning (ICML)*, 2022. URL https://arxiv.org/abs/2201.12740.

APPENDIX

A Basis Decoder and Training Details **14**

A.1 Decoder configuration 14

A.2 Training data and objectives 15

A.3 Optimization and checkpoint selection 16

B Alignment with Domain Priors **17**

B.1 Dataset details 17

B.2 Population density and predicted peak timing 17

B.3 Building usage and predicted weekend flattening 17

C Agreement with Statistical Decompositions **20**

C.1 Statistical decomposition consensus 20

C.2 Periodic-component correspondence 21

C.3 Basis composition across windows 22

D Zero-shot Forecasting Performance **23**

D.1 Evaluation and training protocol 23

D.2 Horizon-level forecasting results 25

A Basis Decoder and Training Details

The configuration below specifies the FD-AR implementation of PRISM used in Sections 4.2 and 4.3. We train a separate decoder set for each $H \in \{96, 192, 336, 720\}$ and use a 512-step evaluation context with a frozen TimesFM 2.5 backbone.

A.1 Decoder configuration

Embedding and normalization. The final input-patch representation has 1,280 dimensions. The encoder retains TimesFM's running normalization over input patches; the mean μ_x and standard deviation σ_x after the final context patch define the decoder's output scale. Decoder targets use $(y - \mu_x)/s_x$, where $s_x = \sigma_x$ for $\sigma_x \geq 10^{-6}$ and $s_x = 1$ otherwise. We multiply each predicted component by s_x and add μ_x once to their sum. Individual components are not separately mean-centered. The initial context statistics remain the output reference throughout residual rollout, even as the backbone updates its input normalization for subsequent patches.

Trend parameterization. A two-layer MLP with width 1,280, ReLU activations, and no dropout maps the embedding to trend features. Separate linear heads predict four coefficients and two knot logits. For bounds $(l_1, r_1) = (0.20, 0.45)$ and $(l_2, r_2) = (0.55, 0.80)$, the knots are $\kappa_k = l_k + (r_k - l_k)\,\mathrm{sigmoid}(z_k)$. The knot-head weights start at zero and its biases place the initial knots at $1/3$ and $2/3$. For each context, let B_T have columns $1, u, (u - \kappa_1)_+, (u - \kappa_2)_+$ on the normalized horizon grid. We compute a reduced QR factorization $B_T = Q_T U_T$ with positive diagonal entries in U_T. The coefficient head predicts $\gamma \in \mathbb{R}^4$, and the trend is $Q_T\gamma$. The coefficients in Equation 2 are $(\alpha_0, \alpha_1, \delta_1, \delta_2)^\top = U_T^{-1}\gamma$. QR reparameterizes the trend basis internally; it does not project the trend away from the Fourier dictionary or enforce orthogonality between components.

Seasonal dictionary and gates. The dictionary contains 64 sine–cosine pairs. Their base periods and harmonic indices are listed in Table 4; the period of an atom is its family's base period divided by its harmonic index. The weekly set omits multiples of seven, which coincide with daily harmonics. For sampling interval Δt in days, an atom with period P_j is retained exactly when

$$P_j > 2\Delta t \quad \text{and} \quad \frac{H\Delta t}{P_j} \geq 2.$$

The second condition requires two cycles within the forecast horizon, with no additional context-cycle condition. The phase origin is the first context observation, so forecast time is $(L + h - 1)\Delta t$.

Table 4: **Fourier dictionary used by FD-AR.** Each retained harmonic has two coefficients and one shared gate. Base periods are fixed in days.

Family	Base period	Harmonic indices	Pairs
Daily	1	$1, \ldots, 10$	10
Weekly	7	$\{1, \ldots, 6\} \cup \{7m \pm 1, 7m \pm 2 : m = 1, \ldots, 10\}$	44
Monthly	30.4375	$1, 2$	2
Yearly	365.25	$1, \ldots, 8$	8

Each family has its own two-layer, width-1,280 ReLU MLP without dropout, followed by a linear coefficient head. A separate gate network has one 128-unit ReLU hidden layer and 64 sigmoid outputs. Its final linear layer is initialized to zero, giving initial soft gates of 0.5. The forward pass sets gates below $\tau = 0.15$ to zero and preserves the values of retained gates. During training, a straight-through derivative preserves the sigmoid gradient across this threshold. The sparsity penalty acts on the original soft gates of admissible atoms.

Residual head and rollout. The residual head concatenates the 1,280-dimensional backbone representation with the corresponding 128-step trend and seasonal patches. A $1536 \to 1280 \to 128$ MLP with a SiLU hidden activation is added to a linear $1536 \to 128$ skip path. Both output maps start at zero. The final T/S patch is zero-padded when necessary, and the output is truncated to H steps. When another patch is needed, the total T/S/R prediction is restored to the original scale and fed into the frozen backbone using its cached context state. The full-horizon T/S curves stay fixed. Feedback is detached during training, so gradients do not propagate through previous predictions or the backbone.

A.2 TRAINING DATA AND OBJECTIVES

Real data. The benchmark datasets in Sections 4.2 and 4.3 are excluded from decoder training. The admitted LOTSA collections number 50 at $H = 96, 192$ and 49 at $H = 336, 720$; `project_tycho` is omitted at the two longer horizons. Training contexts contain 512 observations for 46 collections, 256 for the weekly collections, and 64 for the monthly collection. M4 Daily and M4 Weekly are included, so the trend-dominated analysis in Section 4.4 includes datasets seen during decoder training.

Training–development separation. For collections with multiple source series, approximately 10% of series are reserved for development, keeping their windows and variates together. For a single source series, development uses a disjoint final temporal block; training windows crossing the boundary are excluded with an additional buffer. All stages use the same partition. Development evaluation uses a fixed sample of at most 128 windows per collection.

Synthetic supervision. Each horizon uses 504 synthetic groups: seven trend types, eight sampling intervals (5, 10, 15, and 30 minutes; one hour; one day; one week; and one month), and nine combinations of Fourier and wavelet content. Each group is generated with 4,096 examples and a 512-step context before validity filtering. Trend types comprise increasing, decreasing, and constant linear curves; a one-breakpoint curve with a consistent net direction; increasing and decreasing curves with a change in slope; and exponential curves with either direction and curvature. The Fourier generator has daily/weekly/monthly/yearly harmonic limits 10/4/2/8. It cycles through nonempty combinations of harmonic counts and selects harmonics within each family without replacement. Harmonic k has amplitude $\mathcal{U}(0.3, 1.7)/k$ and phase $\mathcal{U}(0, 2\pi)$. Generated periods must exceed one sampling interval and complete at least one cycle in the context; the stricter decoder mask is applied when supervising coefficients.

Fourier-plus-wavelet, Fourier-only, and wavelet-only batches are sampled with probabilities 0.6, 0.2, and 0.2. Wavelet disturbances contain detail bands at levels 1–3, with dominant family/level combinations Daubechies-1/1, Daubechies-1/3, Coiflet-1/3, and Symlet-4/2. Their context standard deviation relative to the seasonal component is drawn log-uniformly from 0.05 to 0.60, using a unit reference when the Fourier component is absent. Trend and Fourier components are individually scaled using their context statistics before combination. Gaussian noise has standard deviation $\max(0.05\ \mathrm{SD}_{\mathrm{context}}(T^*+S^*+W^*), 10^{-3})$. Wavelets and noise are nuisance content for component supervision; PRISM has no wavelet decoder. Targets then use the combined context's normalization: $T_n^* = (T^* - \mu_x)/s_x$, with Fourier coefficients divided by s_x. The decoder's additional weekly atoms receive zero synthetic coefficient targets. Coefficient supervision compares effective coefficients after masking and gating, averaging absolute error within each admissible family and then equally across families.

Objectives. All forecast and trend losses are mean absolute errors in the normalized output scale. The trend stage minimizes $\mathrm{MAE}(\widetilde{T}, y_n)$ on real batches and $\mathrm{MAE}(\widetilde{T}, T_n^*)$ on synthetic batches, where y_n is the normalized future and T_n^* is its known synthetic trend target. The seasonal stage keeps the trend decoder fixed and uses

$$\mathcal{L}_S^{\mathrm{real}} = \mathrm{MAE}(\widetilde{T} + \widetilde{S}, y_n) + 0.01\,\mathcal{P}_{\mathrm{corr}} + 0.001\,\mathcal{P}_{\mathrm{gate}} + 0.01\,\mathcal{P}_{\mathrm{overlap}}, \tag{5}$$

$$\mathcal{L}_S^{\mathrm{synthetic}} = \mathrm{MAE}(\widetilde{T}, T_n^*) + \mathcal{L}_{\mathrm{coeff}} + 0.001\,\mathcal{P}_{\mathrm{gate}} + 0.01\,\mathcal{P}_{\mathrm{overlap}}. \tag{6}$$

The synthetic trend term is constant with respect to the seasonal parameters. $\mathcal{P}_{\mathrm{corr}}$ is the batch mean squared correlation between the horizon-centered T/S curves, using 10^{-8} in the norm-product denominator. $\mathcal{P}_{\mathrm{gate}}$ is the mean soft-gate value over admissible atoms. The overlap penalty is the batch mean of

$$\frac{\|Q_T^\top \widetilde{S}\|_2^2}{H \max(H^{-1}\|y_n\|_2^2, 10^{-4})},$$

with the target energy detached. This penalizes seasonal energy in the context-specific trend subspace without removing it by projection. The coefficient-refit stage uses the same objectives with gate weight zero and updates only the four final seasonal coefficient heads; feature MLPs, gates, and the trend decoder remain fixed. Finally, the residual stage uses only real data and minimizes $\mathrm{MAE}(\widetilde{T} + \widetilde{S} + \widetilde{R}, y_n)$ with both basis decoders frozen.

A.3 Optimization and checkpoint selection

Table 5 gives the schedule for every horizon. Warm-up uses only synthetic batches. During T/S training and coefficient refitting, every fourth optimizer step uses a synthetic batch and the other steps use real data. Each step has an effective batch size of 1,024. Real collections follow a fixed sampling cycle. We use AdamW with weight decay 10^{-4} and clip the gradient norm at 1.

Table 5: **Decoder training schedule.** Each row starts a fresh optimizer. Only the named decoder or coefficient heads are updated.

Stage	Steps	Initial learning rate	Batches
Trend warm-up	125	10^{-4}	Synthetic
Trend	10,000	10^{-4}	Real and synthetic
Seasonal warm-up	125	10^{-4}	Synthetic
Seasonal	10,000	10^{-4}	Real and synthetic
Seasonal coefficient refit	2,000	2×10^{-6}	Real and synthetic
Residual	8,000	10^{-4}	Real

Learning rates follow cosine decay to zero during warm-up and residual training, and to 2×10^{-5} during the main T/S stages. The refit rate is constant. T/S stages retain their final parameters. The residual checkpoint is selected by the lowest development MAE, with equal weight per training dataset.

At residual-training step $s = 1, \ldots, 8000$, each feedback transition uses the known residual $y_n - \widetilde{T} - \widetilde{S}$ with probability $p_s = 1 - (s-1)/7999$ and the predicted residual otherwise, sampled independently per example. We add the fixed T/S patch before feedback. Development and test rollouts use predicted feedback with all decoder parameters frozen.

B ALIGNMENT WITH DOMAIN PRIORS

This section extends Section 4.1 with the identity and selection rationale of each dataset and the full population-density analysis behind the influenza states.

B.1 DATASET DETAILS

Electricity (Building, BDG-2 Bear). We use `bdg_bear`, an hourly whole-building electricity dataset containing 91 buildings in the evaluated Bear subset of Building Data Genome 2 (Miller et al., 2020b;a). BDG2's `primaryspaceusage` field follows the ENERGY STAR Portfolio Manager property-type scheme; "Education" aggregates K-12, college/university, and other instructional facilities into one label. We use these broad labels to interpret the selected examples; individual building schedules may differ. The two illustrative buildings in Figure 3 are `Bear_education_Carlo` (education building) and `Bear_lodging_Dannie` (lodging building); the education window is from 2017 and the lodging/residential window is from 2016.

Traffic (Monash). We use `traffic_hourly` from the Monash Time Series Forecasting Archive (Godahewa et al., 2021), the hourly road-occupancy-rate series (862 sensors, San Francisco Bay Area, Caltrans PEMS), distinct from the `M_DENSE` Traffic row of Table 1. Sensor T837 is used as an illustrative example; the displayed window is the sensor's first evaluated window (`window_start=0`), not a ranked selection. The Prophet comparison in Figure 4 is fit with library defaults to the same 512-hour observed context and predicted over the same 336-hour horizon; its daily-plus-weekly component is the sum of the weekly and daily terms Prophet activates here.

ILINet (state-level, unweighted ILI). We use `ilinet`, the CDC FluView ILINet weekly influenza-like illness (ILI) surveillance series. State-level `ilinet` series run only 680 weeks (from 2010-10-10), short of the 848-week context-plus-horizon window the checkpoint otherwise uses, so these examples use the 512-week context together with however much of the 336-week horizon each series actually covers (168 weeks in every case here). We rank the 53 state-level jurisdictions with a usable context by density and, within the denser and the sparser tier, take the best-predicted state by MAE divided by the context standard deviation. This gives California and Texas (250 and 117 people/mi^2) as denser examples and Kansas and Wyoming (36 and 6.0 people/mi^2) as sparser ones. Figure 5 shows the densest and the sparsest of the four, California and Wyoming, and Figure 7 highlights all four. Appendix B.2 gives the density-vs-timing analysis. These selected cases illustrate model outputs rather than estimate average performance across states.

B.2 POPULATION DENSITY AND PREDICTED PEAK TIMING

Figure 7 reports the exploratory density–timing analysis in Section 4.1: log-scaled 2020 Census population density versus the predicted yearly-component peak date across 51 ILINet state-level items with both a usable 512-week context and a known population figure. Log10 transformation reduces the skewness of population density from 6.78 to 0.01. The correlation is $r = -0.289$ ($p = 0.040$, $n = 51$). The fitted linear association accounts for 8.3% of the variation in PRISM's predicted peak dates. California, Texas, Kansas, and Wyoming are highlighted as the selected examples. This association concerns predicted seasonal peaks; it does not establish an effect of density on observed outbreak timing.

Prophet comparison. Prophet is fitted to the same 512-week context, and its peak is read from the same first 52 forecast weeks. The corresponding correlation is $r = -0.191$ ($p = 0.180$, $n = 51$, Figure 7, right). These exploratory correlations do not establish a difference between the two models.

B.3 BUILDING USAGE AND PREDICTED WEEKEND FLATTENING

Figure 3 illustrates the education/lodging contrast with one building each. This section repeats the measurement over every labelled building at that site, both to place the two drawn buildings within their groups and to test the contrast at the population level. We take all `bdg_bear` series carrying an "education" or "lodging" `primaryspaceusage` label and at least $512 + 336$ hourly samples, which is 70 buildings (66 education, 4 lodging); no building meeting those conditions is excluded. Each building contributes one window, the last $512 + 336$ samples of its series, with the

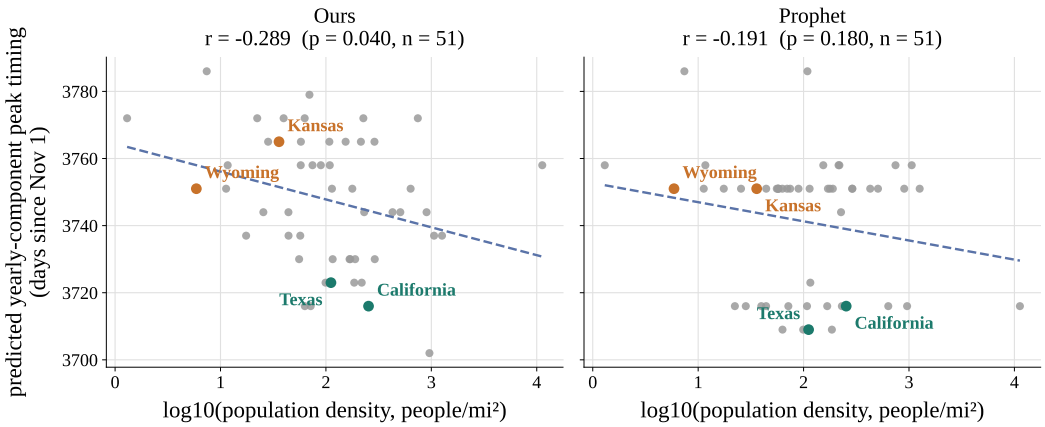

Figure 7: **Population density and predicted annual-component peak dates.** Log10 population density (2020 Census, people/mi^2) vs. predicted yearly-component peak date (days since 1 November 2010, so that 3,716 is 3 January 2021) for 51 ILINet states, read from our predicted yearly component (left) and from Prophet's fitted yearly component over the same context and forecast weeks (right). Dashed: OLS fit. California, Texas, Kansas, and Wyoming are highlighted. Left: $r = -0.289$ ($p = 0.040$). Right: $r = -0.191$ ($p = 0.180$). The reported p-values test each correlation separately.

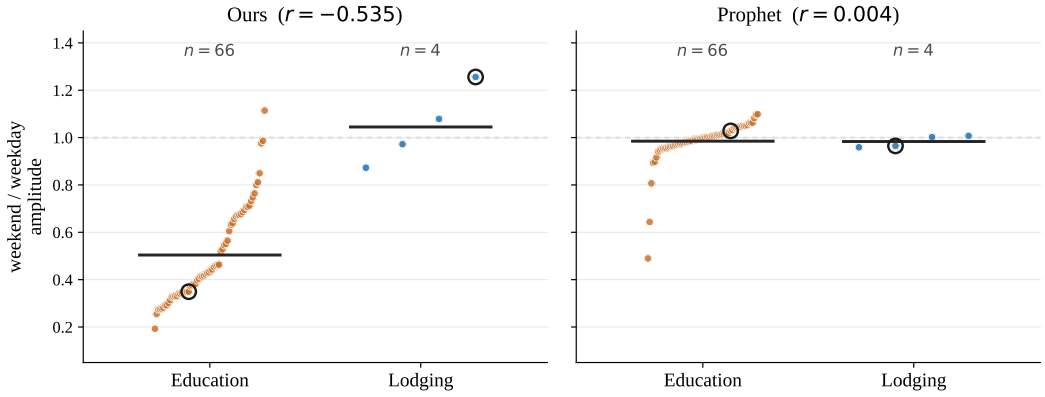

Figure 8: **Our predicted weekend amplitude separates education from lodging buildings; Prophet's does not.** Weekend within-day amplitude relative to weekday amplitude, one point per building, for all 70 labelled `bdg_bear` buildings, read from our predicted seasonal component (left) and from a Prophet fit to the same context and forecast hours (right). Points are spread horizontally by rank within their group for legibility; the horizontal bar is the group mean, and the dashed line marks equal weekend and weekday amplitude. Circled: the two buildings shown in Figure 3. Left: education 0.504 vs. lodging 1.045 ($r = -0.535$, $p < 10^{-4}$). Right: 0.985 vs. 0.983 ($r = 0.004$, $p = 0.98$). The reported p-values test each correlation separately.

first 512 as context. This collection is not part of the decoder's training pool, which contains the `fox`, `panther`, and `rat` sites but not `bear`.

Each predicted seasonal component is summarized by its *weekend amplitude ratio*: the mean weekend within-day peak-to-trough, divided by the mean weekday within-day peak-to-trough, with days assigned from each series' own calendar timestamps. A ratio near 1 corresponds to a weekend day predicted to swing as widely as a weekday; a ratio near 0.5 corresponds to half the weekday variation. Section 4.1 reports this quantity as a percentage range for Traffic.

Education buildings average 0.504 ± 0.203 and lodging buildings 1.045 ± 0.164 (point-biserial $r = -0.535$, $p < 10^{-4}$; Mann–Whitney $U = 7$, $p < 10^{-4}$), and 63 of the 66 education buildings fall below the lowest lodging building. The two buildings in Figure 3 lie inside their groups rather than at their extremes: `Bear_education_Carlo` at 0.349 is at the 30th percentile of the education group, and `Bear_lodging_Dannie` at 1.256 is the largest of the four lodging buildings. The lodging group has four members at this site, so its statistics rest on four ranks; the education group

is the well-sampled side. These ratios read the predicted component and are consistent with the usage prior; they are not a measurement of occupancy.

Prophet comparison. Prophet is fitted to each window and its seasonal component is read over the same forecast hours. Its ratio is 0.985 for education buildings and 0.983 for lodging buildings ($r = 0.004$, $p = 0.98$; Figure 8, right): its weekend forecast is no flatter than its weekday forecast for either building type, so the usage separation this subsection reports is present for our component and absent for Prophet's. Both models are measured on the same windows at one site.

Two alternative statistics. The choice of statistic determines what this measurement reports, so we state the two alternatives we examined rather than only the one that separates. Taking the weekday minus weekend *mean level* of the predicted seasonal component, instead of its amplitude, gives $r = 0.356$ ($p = 0.002$) for our component and $r = 0.403$ ($p < 0.001$) for Prophet's. The two results are consistent: a mean-level contrast is satisfied by shifting the whole weekend down while leaving its shape unchanged, which is what Prophet does here, and only the amplitude ratio distinguishes that from predicting a flatter weekend. A calendar-free statistic, the weekly family's share of the combined daily and weekly component energy, gives $r = 0.037$ ($p = 0.758$) for our component: the usage groups are separated by the weekday/weekend phase rather than by the weekly family's total magnitude.

C AGREEMENT WITH STATISTICAL DECOMPOSITIONS

This section extends Section 4.2: how the references are constructed, how the decoded periodic components correspond to them, and which bases the seasonal decoder uses across windows.

C.1 STATISTICAL DECOMPOSITION CONSENSUS

Evaluation windows. The seasonal-agreement panel uses Electricity, Traffic (`M_DENSE`), Solar-Energy, ETTh1, ETTh2, ETTm1, and ETTm2, with the test partitions in Appendix D.1. We select evenly spaced eligible origins and channels, independently of prediction or decomposition scores. Every model and training seed uses the same selection.

Reference construction. Each method is fitted to the observed context concatenated with the realized future, standardized by the context mean and population standard deviation. Only the seasonal component over the forecast interval is retained. The references are STL, MSTL, SSA, and Prophet, implemented with statsmodels 0.15.0, pyts 0.13.0, and Prophet 1.4.0.

For hourly series, MSTL, SSA, and Prophet use daily and weekly periods jointly; STL uses the weekly period. For subhourly series, all four methods use the daily period. STL uses a length-7 seasonal smoother without robust reweighting. MSTL uses seasonal smoothers of lengths 11 and 15 with two passes for two periods, or length 11 with one pass for a single period.

SSA uses half the fitted span as its embedding window. Reconstructed components are assigned to a seasonal family when their dominant nonzero frequency is within one FFT bin of a harmonic and at least 85% of their non-DC spectral energy is in the peak bin and its neighbors. The first ten harmonics are considered. Prophet uses a linear trend and explicit daily and weekly Fourier orders of 4 and 3, where applicable, with automatic seasonalities and uncertainty sampling disabled.

A reference is active if it is finite over the forecast interval and its RMS is at least 0.05 context standard deviations. A window requires at least two active references and a context standard deviation above 10^{-6}. The consensus is the pointwise median of the active references.

Model comparison and aggregation. We divide each model's predicted seasonal component by the context standard deviation and average values at matching phases of the longest reference period. This phase average is repeated over the horizon and centered before comparison with the consensus. The period is 168 steps for hourly series, 96 for 15-minute series, and 144 for 10-minute series. We phase-average the model output only; the consensus retains the time variation of the statistical references. We compute Pearson correlation over the horizon and take the median of defined correlations across windows for each dataset and seed. For trained baselines, we average these medians equally over seeds 2021, 2022, and 2023; PRISM and TEMPO each use one fixed checkpoint. Table 6 reports these dataset scores. Prophet is a consensus reference and is therefore not included as a competing model in this comparison.

Agreement among statistical methods. For Classical (LOO), we hold out each active method in turn and compare its seasonal component with the pointwise median of the remaining active references, requiring at least two remaining references. We apply the same phase averaging to the held-out component as to the forecasting models. For each method and dataset, we take the median correlation over eligible windows, then average the four method scores equally within each dataset. The plotted mean and sample standard deviation summarize these seven dataset scores. This category describes agreement with a consensus that excludes the evaluated method; forecasting models use the consensus of all active methods. It is a descriptive reference for interpreting the model scores, not a competing forecast.

Interpretation. The statistical references use the realized future, whereas the forecasting models predict from the context alone. This comparison assesses the agreement between predicted seasonal structure and post-hoc statistical structure. Correlation measures agreement in shape, not amplitude accuracy; the consensus provides a structural reference. Overlapping windows and pooled series are not independent samples, so the reported summaries are descriptive, not significance tests.

Table 6: **Seasonal agreement with four statistical references.** Model entries average window-median correlations over three training seeds for trained baselines; Ours and TEMPO each use one fixed checkpoint. Classical (LOO) averages the four methods' window-median correlations with the consensus excluding the evaluated method, using the same phase averaging. Mean and standard deviation are across seven datasets. Classical is a reference diagnostic; bold and underline rank forecasting models only.

Dataset	Classical (LOO)	Ours	N-BEATS-I	TEMPO	DLinear	ETSformer	Autoformer
Electricity	0.954	0.962	**0.971**	0.922	0.961	0.940	0.862
ETTh1	0.826	**0.836**	0.788	0.816	0.825	0.605	0.626
ETTh2	0.749	**0.690**	0.513	0.638	0.663	0.244	0.496
ETTm1	0.919	0.830	**0.879**	0.840	0.517	0.707	0.086
ETTm2	0.848	0.685	**0.723**	0.694	0.678	0.175	0.043
Solar	0.976	0.926	**0.983**	0.880	0.418	0.955	0.015
M_DENSE	0.916	0.957	**0.971**	0.892	0.933	0.945	0.933
Mean	0.884	**0.841**	0.833	0.812	0.714	0.653	0.437
Standard deviation	0.080	0.117	0.173	0.107	0.205	0.332	0.392

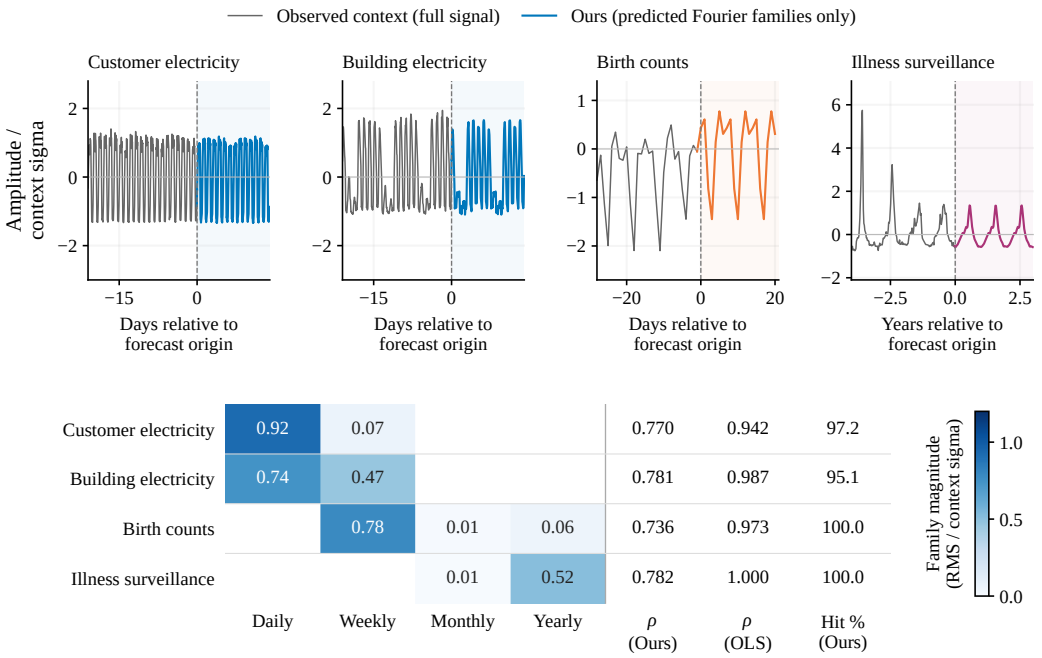

Figure 9: **Predicted periodic components and context-fitted references.** The panels show the observed context and the predicted Fourier families at $H = 336$, centered and scaled by the context standard deviation. Electricity panels combine daily and weekly components; birth counts and illness show weekly and yearly components, respectively. The grid reports family RMS, atom-energy rank correlation, and top-atom agreement. These summaries concern periodic contributions, not phase agreement.

C.2 Periodic-component correspondence

References and metrics. The family reference jointly fits an intercept, a linear trend, and the admissible Fourier dictionary to the observed context, then extends the Fourier contribution over the horizon. The atom reference fits each sine–cosine pair separately to detrended context. Component magnitude is median future-family RMS divided by context standard deviation. Spearman's ρ compares predicted atom-energy rankings with the atom reference; Hit measures agreement on the highest-energy atom, averaging over ties. Only windows where each metric is defined contribute to that metric. Blank cells indicate inactive or unavailable families.

Selection and interpretation. The summaries retain windows with finite, positive out-of-sample gain from the joint OLS reference. Gain is one minus the squared error of its detrended Fourier extrapolation divided by the squared norm of the detrended future. Waveform examples are selected from the stronger half by context-fit family RMS and the lowest quartile of forecast MAE, then chosen for similarity to the context OLS waveform. These are conditional, favorable illustrations; they do not estimate agreement over all evaluation windows.

The panels use customer electricity, an education building from BDG-2 Bear, daily birth counts, and HHS-region weighted ILINet surveillance. The latter differs from the state-level unweighted ILINet examples in Section 4.1. The displayed periodic contributions are centered for comparison; the model's output normalization is specified in Appendix A.

C.3 Basis composition across windows

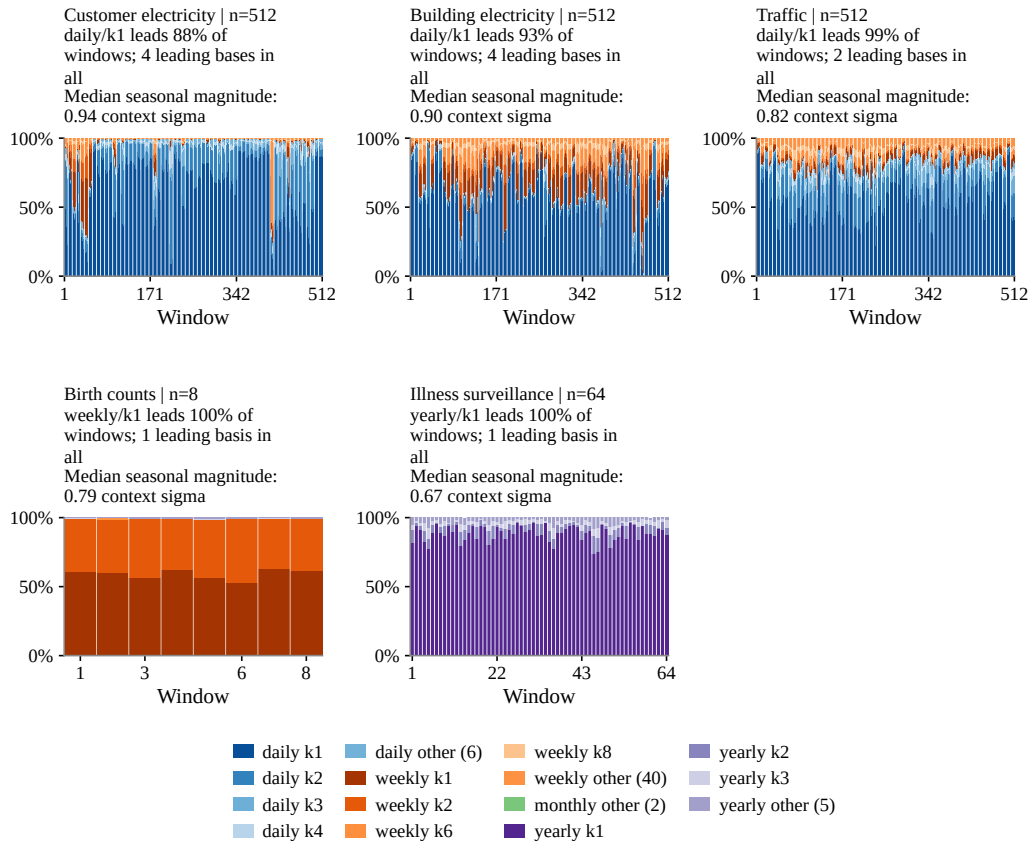

Figure 10: **Relative Fourier-atom contributions across windows.** Each bar is one window at $H = 336$, divided into shares of seasonal coefficient energy. Hue denotes the period family and shade the harmonic. Windows are ordered by series and forecast origin. Panel headers report the leading atom, its frequency of occurrence, and median seasonal RMS in context-standard-deviation units.

An atom's share is its squared effective sine and cosine coefficient energy divided by the total over admissible atoms. Shares describe relative composition; seasonal magnitude is reported separately. Windows with zero or undefined total energy are excluded from share statistics. Leader counts use unique maxima; tied windows retain their bars but do not enter the leader count. Each panel pools multiple windows, so variation across bars can reflect differences between series as well as changes within a series.

D ZERO-SHOT FORECASTING PERFORMANCE

This section extends Section 4.3 with the protocol behind Table 1 and its horizon-level results.

D.1 EVALUATION AND TRAINING PROTOCOL

Datasets. The rows are Electricity (Customer electricity, UCI, 321-channel customer-load benchmark), Traffic (the 30 forecasting targets of `M_DENSE` from LOTSA), Solar-Energy (the 137-channel Alabama solar-power benchmark), and the hourly and 15-minute ETT transformer series, ETTh and ETTm (Zhou et al., 2021). All seven channels of each ETT member are used. The dynamic covariates of `M_DENSE` are excluded for all models. The Traffic row is distinct from the `traffic_hourly` sensor example in Section 4.1. Electricity, Traffic, and ETTh are hourly; Solar-Energy is sampled every 10 minutes and ETTm every 15 minutes.

Splits and windows. The splits, windows, and horizons follow the long-horizon forecasting setup of MOMENT, which also sets the 512-step input used below. ETT uses chronological 12/4/4-month training, validation, and test partitions, with each month defined as 30 days. Observations after the twentieth month are excluded. The other datasets use chronological 70/10/20 splits. For every dataset, the prediction horizons are $H \in \{96, 192, 336, 720\}$. Each split includes every eligible origin at stride one, without padding; the entire target must lie within the split. The input may include observations from an earlier split because they are already available at the forecast origin. All models and seeds are scored on identical dataset, channel, origin, and horizon keys.

Compared models. Within a completed dataset–horizon cell, every model is evaluated on the same windows. PRISM uses the full model, with the residual decoder, and one decoder set per horizon. TimesFM 2.5 uses its pretrained checkpoint zero-shot. TEMPO uses the public `Melady/TEMPO` `TEMPO-80M_v1` checkpoint, whose declared pretraining corpus (ETTm1, ETTh1, ETTm2, electricity, traffic, weather) overlaps with our Electricity (UCI) and ETTm rows, so those are reported without fine-tuning rather than zero-shot; ETTh averages a pretraining-seen member (ETTh1) with a held-out one (ETTh2). The Traffic row uses `M_DENSE`, distinct from the PEMS traffic series in TEMPO's pretraining corpus, and Solar-Energy does not appear in it, so both remain zero-shot for TEMPO. TEMPO uses its checkpoint-native 336-step input, whereas the other models use 512 steps; the error normalization uses the same 512-step context for every model. TEMPO produces longer forecasts by feeding back successive 96-step predictions. Prophet is fitted to the observed context of each window. These fixed-checkpoint and per-window fitted comparisons are evaluated once; the three-seed repetitions below apply to the target-trained baselines, not to these columns.

Prophet configuration. The forecasting baseline uses Prophet 1.4.0 with its default linear trend, additive seasonality, and automatic daily, weekly, and yearly seasonality selection. It is refitted on each raw 512-step context, with no holidays, extra regressors, or custom seasonalities. Timestamps use a common virtual calendar beginning on 3 January 2000, advanced by the absolute source index at the dataset's sampling interval. The forecast is the model's `yhat` output. This context-only forecasting setup differs from the in-sample statistical reference in Appendix C.1.

Training and checkpoint selection. We train N-BEATS-I, DLinear, ETSformer, and Autoformer separately for each target dataset and horizon with seeds 2021, 2022, and 2023. All use 512-step inputs and minimize MSE after per-channel standardization with the population mean and standard deviation of the training split only. Each epoch traverses every eligible training timestamp window, reshuffled with the training seed and epoch; the final incomplete batch is retained. Training runs for at most 10 epochs and stops after three consecutive epochs without a strict improvement in validation MSE. Validation covers all eligible validation windows after every epoch, and the checkpoint with the lowest validation MSE is retained. The test split is not used during training or checkpoint selection, and the selected model is not refitted on training plus validation data. Its checkpoint hash is recorded at training completion and verified by the separate test-evaluation process.

Channel handling and batching. DLinear and N-BEATS-I process channels independently with shared parameters. ETSformer and Autoformer use multivariate inputs on Electricity, Traffic, and Solar-Energy, and one shared univariate model over all seven channels of each ETT dataset. Thus, the multivariate models receive cross-channel information that the channel-independent models do not. A training sample is a timestamp window carrying all retained channels; independent models fold the channel axis into the batch. Batches that exceed the forward-pass memory limit are divided

into chunks with gradients weighted by chunk size and accumulated before one optimizer step. N-BEATS-I uses $\max(1, \lfloor 1024/C \rfloor)$ timestamp windows per step for C channels; the resulting number of univariate windows is at most 1,024. Other batch sizes and nominal learning rates are listed in Table 7.

N-BEATS-I and DLinear configurations. N-BEATS-I uses trend and seasonal stacks with three blocks per stack and shared weights within each stack. Each block has four hidden layers, with widths 256 for trend and 2,048 for seasonality. The trend polynomial degree is three for Electricity and Traffic, and two for Solar-Energy and ETT. We train one model per configuration and seed, without the repeated-fit, look-back, or loss ensemble. Standardized MSE and the common epoch budget replace the original task-specific losses and fixed iteration budget. DLinear uses a 25-step moving average and shares trend and seasonal linear maps across channels (`individual=False`).

ETSformer and Autoformer configurations. Both use model width 512, eight heads, feed-forward width 2,048, and two encoder layers. ETSformer uses two decoder layers, sigmoid activation, and dropout 0.2. During training it retains the implementation's input augmentation: random scaling, shifting, and Gaussian jitter with noise scale 0.2 on standardized inputs. This augmentation is disabled during validation and test evaluation. Its Fourier selection parameter K is 3 on Electricity and Traffic, $(1, 1, 3, 3)$ on Solar-Energy, and $(2, 2, 1, 3)$ on each ETT dataset, in increasing horizon order. Autoformer uses one decoder layer, GELU activation, dropout 0.05, a 25-step moving average, and autocorrelation factor 3. Its decoder context length is 48 for Electricity, Traffic, and Solar-Energy, and 96 for ETT. Calendar features use the CSV timestamps for Electricity and ETT and the supplied start time and hourly interval for `M_DENSE`. The headerless Solar-Energy source uses a declared 10-minute grid beginning on 1 January 2006; these dates are assigned coordinates rather than timestamps recovered from the source. Dataset recipes are taken from the released implementations. Where an exact dataset recipe is unavailable, the Traffic recipe is transferred to `M_DENSE`, Weather to Solar-Energy, and the ETTm2 univariate recipe to the ETT variants requiring it. These transfers do not use test scores. The common input length, budget, channel handling, and N-BEATS-I changes above depart from the original recipes.

Optimization. N-BEATS-I, DLinear, and Autoformer use Adam. N-BEATS-I starts at 10^{-3} and halves its rate at epochs 5 and 8. DLinear and Autoformer retain their initial rate for epochs 1–2 and halve it at each subsequent epoch. ETSformer uses its released Adam implementation with separate backbone, smoothing, and damping parameter groups. At epoch 1 their rates are 10^{-30}, 100η, and 100η, respectively, where η is the nominal rate in Table 7. Following the executable schedule, all three groups share the same rate from epoch 2: approximately $\eta/4$, $\eta/2$, and $3\eta/4$ in epochs 2–4, then η at epoch 5, halved at each later epoch. N-BEATS-I and ETSformer clip the gradient norm at 1. The 10-epoch, patience-3 budget also applies to ETSformer, replacing its original 15-epoch, patience-5 setting without extending training to offset warm-up.

Table 7: **Training settings are fixed before test evaluation.** Nominal learning rates are ordered by $H = 96, 192, 336, 720$; a single value applies to all four horizons. Batches count timestamp windows except for N-BEATS-I, whose nominal batch counts univariate windows. ETT denotes all four members (ETTh1, ETTh2, ETTm1, ETTm2) unless specified.

Model	Dataset	Batch	Nominal learning rate
N-BEATS-I	All	1024	10^{-3}
DLinear	Electricity	16	10^{-3}
	Traffic	16	5×10^{-2}
	Solar-Energy	16	10^{-4}
	ETTh1, ETTh2	32	5×10^{-3}
	ETTm1	8	10^{-4}
	ETTm2	32	$(10^{-3}, 10^{-3}, 10^{-2}, 10^{-2})$
ETSformer	Electricity	32	$(3\times10^{-4}, 3\times10^{-4}, 3\times10^{-4}, 10^{-4})$
	Traffic	32	10^{-3}
	Solar-Energy	32	$(10^{-3}, 10^{-3}, 3\times10^{-4}, 3\times10^{-4})$
	ETT	32	$(10^{-5}, 10^{-3}, 10^{-3}, 3\times10^{-4})$
Autoformer	All	32	10^{-4}

Metric and aggregation. Forecasts are restored to the original data scale before scoring. For channel-window w, we compute

$$e_{w,H} = \frac{1}{H\sigma_w} \sum_{h=1}^{H} |\hat{y}_{w,h} - y_{w,h}|, \tag{7}$$

where σ_w is the population standard deviation of the same last-512-step observed context for every model. Channel-windows with $\sigma_w < 10^{-6}$ are excluded using a common mask. For each dataset, horizon, and seed, we average $e_{w,H}$ over usable channel-windows. Within each seed, the All rows pool the four horizons with usable channel-window counts as weights, rather than averaging horizon scores equally. The main table then averages ETTh1/ETTh2 equally, and likewise ETTm1/ETTm2; the appendix retains all four members separately. Finally, we average the three seed-level scores equally. The appendix also reports their sample standard deviation ($n - 1$ denominator), which measures training variability rather than uncertainty over test windows. For an aggregate row, SD is computed after that seed's horizon and dataset aggregation, not by averaging cell-level SDs. No seed is selected by test score, and all required seeds and horizons must be complete before a result is reported. Parentheses give Ours minus TimesFM 2.5 on the same evaluation support. Bold and underline mark the two lowest displayed model means before rounding, excluding TimesFM.

The Average row in Table 1 is the equally weighted mean of its five dataset rows, computed before rounding. The reported 7.15% increase over TimesFM is $100(\bar{e}_{\mathrm{PRISM}}/\bar{e}_{\mathrm{TimesFM}} - 1)$, where $\bar{e}$ denotes this dataset-averaged MAE/σ; not the mean of the five per-dataset increases.

D.2 Horizon-level forecasting results

Table 8 gives the horizon-level results summarized in Table 1, with ETTh1, ETTh2, ETTm1, and ETTm2 shown separately. All four horizons are complete for every model. The four target-trained baselines each have three complete seed evaluations per dataset and horizon; partial evaluations and partial-seed averages are excluded from every table.

Table 8: **Horizon-level forecasting accuracy for the main-text benchmark.** Values are MAE/σ (lower is better). Trained baselines report the mean over seeds 2021, 2022, and 2023, with $\pm$ sample SD below; the other models are evaluated once. Parentheses give Ours minus TimesFM 2.5. Bold and underline rank means before rounding. All is a usable-window-weighted horizon average. Traffic denotes `M_DENSE`; ETT members are reported separately.

Dataset	Horizon	Ours (Δ TimesFM)	TEMPO	Prophet	N-BEATS-I	DLinear	ETSformer	Autoformer
Electricity	96	<u>0.295</u> (+0.017)	0.316	0.402	**0.295** ±0.001	0.301 ±0.000	0.470 ±0.004	0.457 ±0.006
Electricity	192	**0.319** (+0.015)	0.345	0.459	<u>0.327</u> ±0.001	0.328 ±0.001	0.499 ±0.006	0.481 ±0.024
Electricity	336	**0.348** (+0.013)	0.382	0.557	0.366 ±0.004	<u>0.364</u> ±0.000	0.526 ±0.002	0.568 ±0.120
Electricity	720	**0.417** (+0.014)	0.471	0.844	0.439 ±0.004	<u>0.434</u> ±0.002	0.764 ±0.003	0.538 ±0.003
Electricity	All	**0.342** (+0.015)	0.376	0.557	<u>0.354</u> ±0.000	0.354 ±0.000	0.560 ±0.003	0.510 ±0.029
Traffic	96	<u>0.284</u> (+0.021)	0.322	0.459	**0.275** ±0.001	0.309 ±0.002	0.395 ±0.014	0.405 ±0.032
Traffic	192	<u>0.292</u> (+0.018)	0.337	0.480	**0.286** ±0.001	0.320 ±0.002	0.415 ±0.028	0.389 ±0.030
Traffic	336	<u>0.306</u> (+0.019)	0.356	0.520	**0.287** ±0.003	0.327 ±0.001	0.415 ±0.019	0.391 ±0.004
Traffic	720	0.360 (+0.029)	0.422	0.634	**0.301** ±0.001	<u>0.360</u> ±0.003	0.663 ±0.003	0.386 ±0.012
Traffic	All	<u>0.308</u> (+0.021)	0.356	0.518	**0.287** ±0.001	0.328 ±0.001	0.464 ±0.013	0.393 ±0.010
Solar-Energy	96	0.388 (+0.087)	0.494	0.618	**0.308** ±0.004	<u>0.362</u> ±0.001	0.498 ±0.036	0.666 ±0.041
Solar-Energy	192	0.407 (+0.083)	0.513	0.683	**0.329** ±0.003	<u>0.379</u> ±0.001	0.484 ±0.024	0.625 ±0.008
Solar-Energy	336	0.439 (+0.095)	0.542	0.844	**0.329** ±0.000	<u>0.391</u> ±0.002	0.534 ±0.010	0.601 ±0.035
Solar-Energy	720	0.468 (+0.115)	0.570	1.258	**0.333** ±0.001	<u>0.395</u> ±0.000	0.710 ±0.009	0.800 ±0.074
Solar-Energy	All	0.425 (+0.095)	0.529	0.845	**0.325** ±0.001	<u>0.381</u> ±0.001	0.555 ±0.004	0.672 ±0.024

Table 8: **Horizon-level forecasting accuracy (continued).** ETTh1, ETTh2, ETTm1, and ETTm2 are shown separately, with the same metric, statistics, and ranking as above.

Dataset	Horizon	Ours (Δ TimesFM)	TEMPO	Prophet	N-BEATS-I	DLinear	ETSformer	Autoformer
ETTh1	96	**0.616** (+0.013)	0.639	0.765	0.736 ±0.035	0.645 ±0.013	0.864 ±0.023	0.807 ±0.091
ETTh1	192	**0.666** (+0.014)	0.682	0.862	0.788 ±0.042	0.685 ±0.002	0.916 ±0.024	0.784 ±0.028
ETTh1	336	**0.690** (+0.011)	0.714	1.022	0.877 ±0.058	0.791 ±0.040	1.095 ±0.214	0.838 ±0.013
ETTh1	720	**0.730** (+0.013)	0.766	1.309	0.935 ±0.021	0.956 ±0.025	1.211 ±0.425	0.878 ±0.015
ETTh1	All	**0.672** (+0.013)	0.696	0.970	0.827 ±0.003	0.758 ±0.014	1.009 ±0.130	0.824 ±0.017
ETTh2	96	**0.779** (+0.020)	0.788	1.055	0.960 ±0.025	0.850 ±0.024	0.977 ±0.004	1.053 ±0.083
ETTh2	192	0.913 (+0.037)	**0.902**	1.303	1.219 ±0.168	1.022 ±0.029	1.204 ±0.127	1.234 ±0.131
ETTh2	336	1.005 (+0.024)	**0.997**	1.647	1.505 ±0.172	1.232 ±0.022	1.336 ±0.125	1.256 ±0.041
ETTh2	720	1.173 (+0.035)	**1.163**	2.344	2.357 ±0.434	1.785 ±0.083	1.347 ±0.016	1.661 ±0.099
ETTh2	All	0.955 (+0.029)	**0.950**	1.542	1.461 ±0.037	1.189 ±0.001	1.205 ±0.062	1.281 ±0.061
ETTm1	96	0.636 (+0.033)	0.632	0.812	0.666 ±0.008	**0.628** ±0.004	0.973 ±0.018	0.858 ±0.025
ETTm1	192	0.695 (+0.026)	0.697	0.997	0.736 ±0.017	**0.682** ±0.001	0.904 ±0.025	0.997 ±0.227
ETTm1	336	**0.744** (+0.019)	0.759	1.281	0.785 ±0.027	0.764 ±0.008	0.994 ±0.086	1.038 ±0.147
ETTm1	720	**0.826** (+0.016)	0.857	1.982	0.853 ±0.014	0.844 ±0.012	1.151 ±0.183	1.017 ±0.054
ETTm1	All	**0.724** (+0.024)	0.735	1.259	0.759 ±0.005	0.728 ±0.002	1.004 ±0.041	0.976 ±0.097
ETTm2	96	**0.720** (+0.029)	0.737	0.872	0.833 ±0.035	0.795 ±0.017	1.007 ±0.032	1.115 ±0.071
ETTm2	192	0.867 (+0.042)	**0.853**	1.114	0.975 ±0.056	0.950 ±0.017	1.211 ±0.233	1.481 ±0.303
ETTm2	336	0.991 (+0.034)	**0.974**	1.464	1.144 ±0.042	1.105 ±0.046	1.533 ±0.471	2.138 ±1.383
ETTm2	720	1.187 (+0.037)	**1.157**	2.276	1.448 ±0.073	1.358 ±0.077	1.681 ±0.202	1.857 ±0.330
ETTm2	All	0.938 (+0.035)	**0.927**	1.420	1.095 ±0.030	1.048 ±0.004	1.353 ±0.220	1.643 ±0.438