FAPA: A DUAL-BRANCH FRAMEWORK FOR TIME SERIES FORECASTING WITH FREQUENCY AMPLITUDE SHAPING AND HARMONIC PERIOD ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series forecasting supports applications in energy, transportation, and meteorology. Spectral analysis reveals that time series often consist of high-amplitude periodic components and low-amplitude irregular residuals. Fitting structured periodic patterns and irregular residuals jointly with one set of shared weights is harder than modeling them separately, since the two components have very different statistical properties. Predicting the periodic component in a separate branch removes this coupling but raises two problems. First, per-window normalization, the standard remedy for distribution shift, maps windows that differ only in scale to the same input, so the amplitude of the periodic component is inherited from the input window instead of being predicted. Second, the branch has no directly available training target: re-estimating its periods on the ground truth gives periods that are unavailable at inference, while the input-window periods can fall between the frequency bins of the target window, where nearest-bin reconstruction can lose more than half of the component's energy. We propose FAPA, a plug-and-play dual-branch framework that improves existing forecasters. Its periodic branch predicts the periodic component at its original scale and refines the predicted amplitudes with a lightweight frequency Amplitude Shaping Module (AmpShape). This branch is supervised by a dedicated periodic loss whose target is built by Harmonic Period Alignment (HPA), which projects the ground truth onto sinusoids at the input-window periods and is exact for any horizon at no inference cost. A residual branch passes the residual to the backbone. Experiments on eight backbones (e.g., PatchTST, TimesNet) and five datasets show that FAPA lowers the MSE on the five-dataset average for every backbone: it falls by 10.7% overall, and by 19.6% and 18.9% on ECL and Traffic, respectively. A controlled synthetic study further shows that the gain grows monotonically with the amplitude imbalance between the periodic component and the residual.

1 INTRODUCTION

Time series forecasting supports power load scheduling, traffic management, and weather prediction (Hyndman & Athanasopoulos, 2018; Makridakis et al., 2018; Lim & Zohren, 2021; Li et al., 2018; Yu et al., 2018). Deep learning has become the prevailing approach on standard long-term forecasting benchmarks, from early RNNs and CNNs (Lai et al., 2018; Shih et al., 2019) to recent Transformer-based (Zhou et al., 2021; Nie et al., 2023; Liu et al., 2024a; Li et al., 2019; Kitaev et al., 2020) and MLP-based models (Zeng et al., 2023; Chen et al., 2023; Das et al., 2023; Yi et al., 2023). Most of these models fit every part of a series with a single set of shared weights.

In the frequency domain, the energy of the series in long-term forecasting is concentrated in a few high-amplitude periodic components, and the remaining variation forms an irregular residual of much smaller amplitude (Appendix J). A single set of weights must fit both, and their statistics differ sharply: the periodic components are regular, whereas the residual is irregular, so the fit is a compromise between the two. Because the periodic components are predictable, they can be extracted and predicted in a separate branch, which leaves the backbone only the residual. Decomposition architectures and frequency-based normalization methods already separate such components from the input (Wu et al., 2021; Zhou et al., 2022b; Ye et al., 2024); we focus on forecasting the separated

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.

periodic component accurately, which requires predicting its amplitude and supervising it with an exact target.

Neither requirement is easy to meet. First, most forecasters normalize each input window (Kim et al., 2022; Liu et al., 2023b; Fan et al., 2023), and a separate periodic branch would naturally inherit this practice. Normalization, however, removes the mean and standard deviation of each window before the model runs and restores them only after prediction, so windows that differ only in scale reach the model as the same input. The amplitude of the periodic component is thus inherited from the input window rather than predicted (Figure 1(a)), although predicting it is what periodic forecasting requires. Second, the periodic branch has no directly available training target. Its periods must be estimated on the input window, because those of the target window are unknown at inference; re-estimating them on the ground truth would therefore supervise the branch at periods it cannot know in advance. Keeping the input-window periods avoids this mismatch, but they must then be evaluated on a target window of length H, where the ratio H/p_k for a period p_k need not be an integer; in that case the periodic energy falls between frequency bins, and a nearest-bin reconstruction drifts out of phase over the horizon (Figure 1(b)).

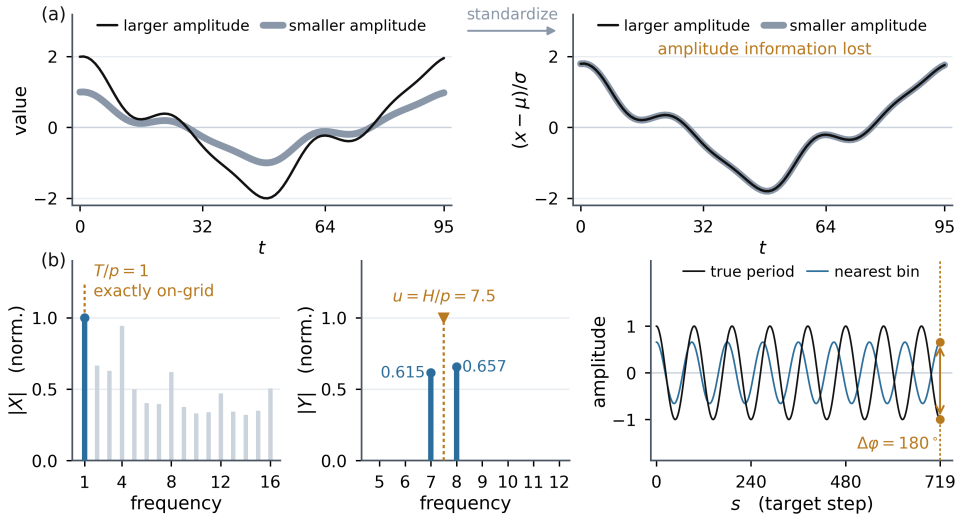

Figure 1: Two problems of a separate periodic branch. (a) Amplitude erased by normalization: two windows differing only in scale become identical after standardization, so the model inherits the amplitude from the input window instead of predicting it. (b) Off-grid periods ($T = 96$, $H = 720$): a period of 96 steps occupies exactly one bin of the input spectrum (left) but falls between two bins of the target spectrum (middle); reconstructing it from the nearest bin gives a period of 90 steps, drifting half a cycle out of phase over the target window (right).

We propose **FAPA** (**F**requency **A**mplitude shaping and harmonic **P**eriod **A**lignment), a plug-and-play dual-branch framework that attaches to existing models without changing them. For the first problem, a periodic branch forecasts the periodic component from the separated component and the raw input, without per-window normalization, so that its amplitude is predicted rather than inherited, while the residual is passed to the backbone, which keeps its own normalization. Because the periodic component is regular, a simple MLP predicts it directly; since a regression over the H steps sets the amplitude of each frequency only implicitly, a lightweight frequency **Amplitude Shaping Module (AmpShape)** then refines the predicted amplitudes in the frequency domain. For the second problem, we train the branch with a dedicated **periodic loss** and build its target by projecting the ground truth, in the time domain, onto sinusoidal bases at exactly the input-window periods $\{p_k\}$, which we call **Harmonic Period Alignment (HPA)**: the target shares its periods with the input of the branch, remains exact when H/p_k is not an integer, and is computed only during training. Our contributions are as follows:

- We analyze a periodic branch that is predicted separately from the residual and identify two problems: under per-window normalization its amplitude is inherited from the input window rather

than predicted, and its periods, which must come from the input window, need not lie on the frequency bins of the target window, so bin selection cannot always give an exact target.

- We propose **FAPA**, a plug-and-play dual-branch framework whose periodic branch forecasts the periodic component at its original, unnormalized scale and refines the predicted amplitudes with the lightweight **AmpShape** module.
- We supervise the periodic branch with a dedicated **periodic loss**: its target is built by **HPA**, i.e., by projecting the ground truth onto continuous sinusoidal bases at the input-window periods; the target uses only periods available at inference, is exact for any horizon, and coincides with bin selection whenever H/p_k is an integer.
- Experiments on five real-world datasets and eight backbones (e.g., PatchTST, TimesNet) spanning four architectural families show that FAPA lowers the MSE on the five-dataset average for every backbone, reducing the MSE by 10.7% overall, with 70.7 K additional parameters on ECL at $T = 96$ and $H = 96$, independent of the backbone. A controlled synthetic study further shows that the gain grows monotonically with the amplitude imbalance between the periodic component and the residual.

2 Related Work

2.1 Deep Learning Time Series Forecasting Models

Time series forecasting has evolved from RNN-based models (Hochreiter & Schmidhuber, 1997; Salinas et al., 2020; Lai et al., 2018; Rangapuram et al., 2018) to Transformer architectures. Following the sparse attention in Informer (Zhou et al., 2021), recent advancements focus on patch-based inputs (Nie et al., 2023) and variate-wise tokens (Liu et al., 2024a). Architectures such as Crossformer (Zhang & Yan, 2023), Pyraformer (Liu et al., 2022b), and Triformer (Cirstea et al., 2022) further optimize dependency modeling and complexity. Linear and MLP-based models (DLinear (Zeng et al., 2023), TSMixer (Chen et al., 2023), TiDE (Das et al., 2023), SparseTSF (Lin et al., 2024)) have proven effective with concise structures, and state-space models (Gu & Dao, 2024) and large pre-trained models (Liu et al., 2024b) have also been brought to forecasting.

While these backbones achieve strong benchmark results, each fits the periodic component and the residual with a single set of weights, so the two are not modeled independently. FAPA separates them into dedicated branches and can be attached to any model above, leaving the backbone to model the residual.

2.2 Time Series Decomposition Methods

Time series decomposition is a long-standing approach to forecasting. Classical models such as ARIMA (Box & Jenkins, 1970) handle trend and seasonality through differencing, and therefore operate in the time domain. Methods like STL (Cleveland et al., 1990) and RobustSTL (Wen et al., 2019) achieve trend-seasonality separation via iterative smoothing or robust trend estimation. In deep learning, architectures like Autoformer (Wu et al., 2021) and FEDformer (Zhou et al., 2022b) embed decomposition blocks directly into the network, leveraging auto-correlation or frequency-domain mechanisms to capture periodic dependencies. N-BEATS (Oreshkin et al., 2020) employs learnable basis expansions for interpretable forecasting, and N-HiTS (Challu et al., 2023) adds hierarchical interpolation with multi-rate sampling. SCINet (Liu et al., 2022a) downsamples recursively into different resolutions, and Koopa (Liu et al., 2023a) uses Koopman dynamics to disentangle time-variant and time-invariant components.

The periodic branch, however, also needs the periodic component of the ground truth on the target window, which the decomposition blocks above do not. Its periods are estimated on the input window and generally fall between the frequency bins of the target window; projecting the ground truth onto continuous sinusoidal bases at those periods represents them exactly, so the target is accurate at any horizon.

2.3 Frequency-Domain Modeling and Normalization

In the frequency domain, periodicity is directly accessible. FiLM (Zhou et al., 2022a) projects the history onto Legendre polynomials and filters it in the Fourier domain, and FreTS (Yi et al., 2023)

applies MLPs in the frequency domain. TimesNet (Wu et al., 2023) and FITS (Xu et al., 2024) further exploit multi-periodicity and frequency interpolation. A separate line of work leaves the architecture unchanged and adapts the input: RevIN (Kim et al., 2022), Non-stationary Transformers (Liu et al., 2022c), and SAN (Liu et al., 2023b) normalize each window or slice to remove distribution shift, and FAN (Ye et al., 2024) is likewise a normalization method, one that achieves normalization by decomposing out the dominant frequencies of each window and predicting them separately.

These methods standardize the input before prediction, or keep the decomposition inside the network, so the separation serves normalization rather than isolating the periodic component. FAPA separates the periodic component, consisting of only a few frequencies, into a branch that operates at the original scale, keeps the mean in the residual, projects the ground truth onto continuous sinusoidal bases at those periods, and uses the result as the target of that branch.

3 METHODOLOGY

3.1 PROBLEM DEFINITION

Given historical observations $\mathbf{X} \in \mathbb{R}^{T \times C}$ of a multivariate time series, where T is the input length and C is the number of variables, the goal of long-term forecasting is to predict the future sequence $\mathbf{Y} \in \mathbb{R}^{H \times C}$ over the next H steps, the horizon. As is standard, every channel is standardized with training-set statistics. The "original scale" below refers to this scale, without per-window normalization.

3.2 FRAMEWORK OVERVIEW

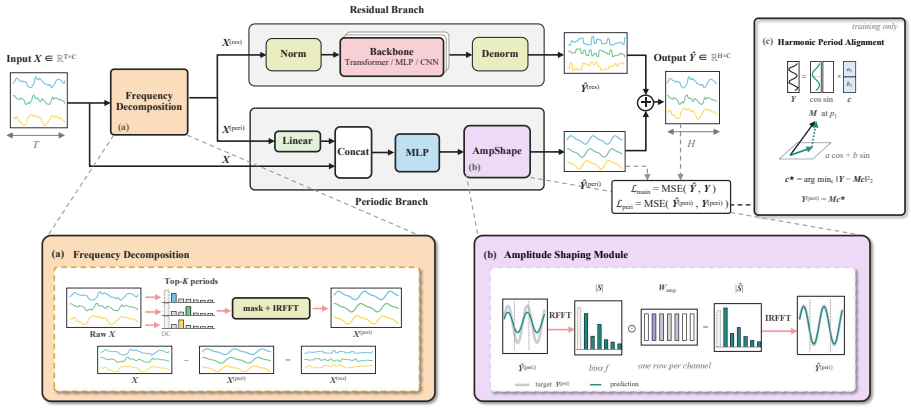

Figure 2: Framework of FAPA. (a) Frequency decomposition: the Top-K frequencies of each channel are selected and their bins kept to obtain $\mathbf{X}^{(\text{peri})}$; the zero-frequency component is excluded. (b) AmpShape rescales the amplitudes of the preliminary prediction with a learnable weight per frequency and channel. (c) Harmonic Period Alignment (training only): the ground truth is projected onto sinusoids at the input-window periods, giving the target of the periodic branch.

As shown in Figure 2, the input is first separated into a periodic component and a residual, using the Top-K periods estimated from the input spectrum. The two branches then predict the two components separately: a periodic branch predicts the periodic component with an MLP predictor, refined by AmpShape, and a residual branch passes the residual through the backbone. The final prediction is the sum of the two branch outputs, $\hat{\mathbf{Y}} = \hat{\mathbf{Y}}^{(\text{peri})} + \hat{\mathbf{Y}}^{(\text{res})}$, and the periodic branch is additionally supervised during training by the periodic loss, whose target is built from the ground truth through Harmonic Period Alignment.

3.3 FREQUENCY DECOMPOSITION

The input sequence is separated into a periodic component and a residual, $\mathbf{X} = \mathbf{X}^{(\text{peri})} + \mathbf{X}^{(\text{res})}$, where the periodic component collects the high-amplitude periods of $\mathbf{X}$ and the residual contains

everything else. To find these periods, we apply the real fast Fourier transform (RFFT) to $\mathbf{X}$ along the time dimension and select the K frequencies with the largest amplitudes (Top-K), separately for each channel and each window, where K is the number of periods. The zero-frequency (DC) component is excluded from the selection, so the mean of the series stays in the residual. Each selected frequency index i_k determines a period of $p_k = T/i_k$ steps. Appendix C compares this selection with the decomposition operators used in prior work.

Because the periods are estimated on the input window itself, T/p_k is an integer for every k, so each period occupies exactly one bin of that window. Keeping those bins and applying an inverse RFFT is therefore equivalent to projecting the window onto the sinusoidal bases at $\{p_k\}$. The residual of each channel is then orthogonal to its periodic component, so that $\|\mathbf{X}\|^2 = \|\mathbf{X}^{(\mathrm{peri})}\|^2 + \|\mathbf{X}^{(\mathrm{res})}\|^2$, and the residual keeps only the energy outside the selected bins. This agreement holds only on the input window: on the target window the same periods no longer land on bins, and the periodic component must be obtained differently (Section 3.5.1).

3.4 Dual-Branch Prediction

3.4.1 Periodic Component Branch

The periodic component branch predicts the future periodic component $\hat{\mathbf{Y}}^{(\mathrm{peri})}$ with two modules: an MLP predictor and AmpShape. The branch receives $\mathbf{X}^{(\mathrm{peri})}$ without per-window normalization, so no scale information is removed before prediction.

MLP Predictor. The separated periodic component $\mathbf{X}^{(\mathrm{peri})}$ carries the periodicity and the amplitude of the input but lacks the rest of the raw sequence, such as local trends and irregular variations. The predictor therefore processes both: the periodic component is passed through a linear layer and an activation function, and the result is concatenated with the raw input $\mathbf{X}$ before an MLP maps it to a vector of length H. The preliminary prediction is

$$\tilde{\mathbf{Y}}^{(\mathrm{peri})} = \mathrm{MLP}\left(\mathrm{Concat}\left(\psi\left(\mathrm{Linear}(\mathbf{X}^{(\mathrm{peri})})\right),\ \mathbf{X}\right)\right),$$

where $\mathrm{Linear}(\cdot)$ denotes a linear projection, ψ an activation function, $\mathrm{Concat}(\cdot)$ concatenation along the feature dimension, and $\mathrm{MLP}(\cdot)$ a multi-layer perceptron.

AmpShape. The MLP regresses the periodic component in the time domain, so the amplitude of each frequency is set only implicitly, through the H output values. As a lightweight refinement on top of the MLP prediction, we therefore add AmpShape, which adjusts these amplitudes directly. After converting the preliminary prediction to the frequency domain, we apply a learnable weight to each frequency component and each channel:

$$\hat{\mathbf{Y}}^{(\mathrm{peri})} = \mathrm{IRFFT}\left(\mathrm{RFFT}(\tilde{\mathbf{Y}}^{(\mathrm{peri})}) \odot \mathbf{W}\right), \tag{1}$$

where $\mathbf{W} = \mathbf{W}_a + \mathbf{W}_b$ is a learnable weight matrix, with $\mathbf{W}_a, \mathbf{W}_b \in \mathbb{R}^{N_f \times C}$ learnable, $N_f = \lfloor H/2 \rfloor + 1$ the number of frequency bins, C the number of channels, $\odot$ element-wise multiplication, and IRFFT the inverse of RFFT. The two matrices start at one and zero, respectively, and the DC weight is fixed at one, so the module starts from the identity and leaves the mean unchanged. The effective weight is real-valued and positive in all our runs, so only magnitudes are rescaled.

The weights are channel-independent: the amplitudes of different channels can differ substantially, so each variable learns its own correction. AmpShape introduces $O(H \times C)$ parameters, small relative to most of the backbones we test.

3.4.2 Residual Component Branch

The residual component branch processes $\mathbf{X}^{(\mathrm{res})}$ through the backbone, whose architecture is left at its default configuration. Because the input to this branch is the residual, the backbone models low-amplitude, irregular variations, which we expect to be easier to learn. Section 4.3 examines this by decomposing the validation error into the periodic subspace and its orthogonal complement.

3.5 Training Objective

The training objective combines a term on the model prediction $\hat{\mathbf{Y}}$ with a periodic loss on the periodic branch:

$$\mathcal{L} = \mathrm{MSE}(\hat{\mathbf{Y}}, \mathbf{Y}) + \lambda \mathcal{L}_{\mathrm{peri}}, \qquad \mathcal{L}_{\mathrm{peri}} = \mathrm{MSE}(\hat{\mathbf{Y}}^{(\mathrm{peri})}, \mathbf{Y}^{(\mathrm{peri})}),$$

where λ is the periodic loss weight, $\mathcal{L}_{\mathrm{peri}}$ denotes the periodic loss, and $\mathbf{Y}^{(\mathrm{peri})}$ is the target of the periodic branch, constructed in Section 3.5.1.

3.5.1 Harmonic Period Alignment

To supervise the periodic branch, we need the periodic component of the ground truth $\mathbf{Y}$ at the periods $\{p_k\}$ that define the input of the branch. Re-selecting Top-K frequencies on $\mathbf{Y}$ would not serve this purpose: those periods depend on $\mathbf{Y}$ itself and cannot be recovered from the input window, so the branch could not reproduce them at inference. We therefore keep the periods estimated on the input window and use them to extract the periodic component of $\mathbf{Y}$.

Because the periods are estimated from the input window alone, they are available at inference, where they define the input of the branch, and in training, where they define its target. Moving them to the target window changes only their bin coordinate, from i_k to $u_k = i_k H/T$, because p_k spans T/p_k cycles of the input window but H/p_k cycles of the target window.

Let $\mathbf{M} = \big[\cos(2\pi t/p_1), \sin(2\pi t/p_1), \ldots, \cos(2\pi t/p_K), \sin(2\pi t/p_K)\big]_{t=0}^{H-1} \in \mathbb{R}^{H\times 2K}$ denote the matrix whose columns are the sinusoidal bases at the periods of one channel, sampled on the target window, let $\mathbf{y} \in \mathbb{R}^H$ be that channel of $\mathbf{Y}$, and let $\mathbf{c} \in \mathbb{R}^{2K}$ denote the corresponding coefficients. The periodic component of $\mathbf{y}$ is the reconstruction from the least-squares fit

$$\mathbf{c}^\star = \arg\min_{\mathbf{c}} \|\mathbf{y} - \mathbf{M}\mathbf{c}\|_2^2, \qquad \mathbf{y}^{(\mathrm{peri})} = \mathbf{M}\mathbf{c}^\star, \tag{2}$$

where $\mathbf{c}^\star$ is the ordinary least-squares solution, so that $\mathbf{M}\mathbf{c}^\star$ is the orthogonal projection of $\mathbf{y}$ onto the span of the bases. In practice, the fit is solved with a negligible ridge term for numerical stability, to which the results are insensitive (Appendix H). Stacking $\mathbf{y}^{(\mathrm{peri})}$ over channels gives $\mathbf{Y}^{(\mathrm{peri})}$, and we refer to Eq. (2) as harmonic projection. The frequencies $1/p_k$ are not restricted to the frequency grid of the target window, so the fit is defined for any target length and recovers a sinusoid at period p_k without loss even when H/p_k is non-integer. On the target window u_k is generally fractional, so the energy is spread over the neighboring bins and a nearest-bin reconstruction picks a slightly different period that drifts out of phase over the horizon (Figure 1(b)). On the input window, by contrast, T/p_k is an integer and the fit coincides with keeping those bins (Section 3.3).

The input and the target of the periodic branch are thus described by the same periods, which is what we call Harmonic Period Alignment.

4 Experiments

We evaluate FAPA on five widely used real-world datasets: ETTm1, ETTm2, Electricity (ECL), Traffic, and Weather. Every backbone is trained with and without FAPA under an identical configuration (Appendix A.3.1), so that the two variants differ only in FAPA itself, and all results are averaged over three random seeds. FAPA itself adds two hyperparameters, the periodic loss weight λ and the number of periods K: λ is fixed to a single value per dataset, and K follows a fixed rule applied identically to every backbone. Neither is tuned per backbone or per setting (Appendix A.4). The full results for every horizon are reported in Appendix B, and the dataset descriptions, backbone configurations, and further implementation details are provided in Appendix A.

4.1 Main Results

FAPA improves every backbone on the five-dataset average, and 34 of the 40 dataset–backbone pairs. Table 1 reports the results for eight backbones from four architectural families. Unless stated otherwise, a relative change is computed between MSE values averaged over the settings being summarized; on this basis the MSE falls by 10.7% overall, while the mean of the per-setting relative changes is 7.7%. The relative effect is not driven by a single horizon. Averaged over the backbones,

the MSE falls by 19.6% on ECL and 18.9% on Traffic, against 3.5% on ETTm1 and 5.1% on Weather, while ETTm2 remains on par with the backbones (-0.1%). The standard deviation across seeds, averaged over the horizons, stays below 0.01 in every setting (Table 12), well below the gains on ECL and Traffic. Section 4.3 isolates the dependence on the amplitude ratio with synthetic data, and Section 4.4 relates the size of the gain to the spectral statistics of the input window.

Table 1: Multivariate forecasting results on five benchmark datasets with the input length fixed to $T = 96$. Every entry is the MSE of the corresponding backbone, averaged over the horizons $H \in \{96, 192, 336, 720\}$. Within each pair, **Base** is the backbone alone and **+FAPA** the same backbone with FAPA attached; both are trained by us from scratch under the protocol of Appendix A.3.1, and **bold** marks the better of the two. The last row averages each column over the five datasets. Lower is better. Results for the individual horizons and the complete mean absolute error (MAE) are reported in Appendix B.

Dataset	PatchTST		DLinear		iTransformer		TSMixer		TimesNet		FITS		SparseTSF		FreTS	
	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA
ETTm1	0.393	**0.379**	0.392	**0.387**	0.412	**0.404**	0.400	**0.399**	0.406	**0.400**	0.432	**0.392**	0.417	**0.387**	0.412	**0.403**
ETTm2	0.282	**0.278**	**0.574**	0.587	**0.290**	0.292	0.293	**0.293**	0.295	**0.292**	0.289	**0.283**	**0.288**	0.290	0.341	**0.335**
ECL	0.235	**0.199**	0.228	**0.206**	0.191	**0.186**	0.214	**0.183**	0.212	**0.178**	0.434	**0.213**	0.226	**0.191**	**0.210**	0.213
Traffic	0.581	**0.544**	0.749	**0.663**	0.473	**0.465**	0.628	**0.523**	0.633	**0.511**	1.175	**0.654**	0.730	**0.580**	0.614	**0.589**
Weather	0.252	**0.245**	**0.246**	0.250	0.263	**0.247**	0.251	**0.239**	0.260	**0.251**	0.277	**0.250**	0.275	**0.244**	0.259	**0.252**
Average	0.349	**0.329**	0.438	**0.419**	0.326	**0.319**	0.357	**0.327**	0.361	**0.326**	0.522	**0.358**	0.387	**0.338**	0.367	**0.359**

4.2 Ablation Study

To evaluate each component, we remove one component at a time on ETTm1 and ETTm2 with PatchTST as the backbone. $-$HPA keeps the periodic loss but re-selects the Top-K periods on the ground-truth window instead of reusing the input-window periods, $-$AmpShape drops the amplitude shaping module, $-\mathcal{L}_{\text{peri}}$ removes the periodic loss, $-$Concat removes the raw input from the concatenation, and $-$Unnorm normalizes the periodic branch. Normalizing the periodic branch and removing AmpShape raise the MSE in all eight settings, and removing the periodic loss hurts most at long horizons on ETTm2 (0.565 vs. 0.390 at $H = 720$). $-$HPA raises the MSE on ETTm2 at every horizon and ends up worse than the backbone alone, by 0.136 and 0.109 at $H = 96$ and 192 (0.313 and 0.351 against 0.177 and 0.242).

Table 2: Ablation study on ETTm1 and ETTm2 with PatchTST as the backbone (MSE). Full is PatchTST+FAPA as in Table 8; **bold** marks the lowest MSE in each row. Lower is better.

Dataset	H	**Full**	**$-$HPA**	**$-$AmpShape**	**$-\mathcal{L}_{\text{peri}}$**	**$-$Concat**	**$-$Unnorm**
ETTm1	96	**0.318**	0.318	0.324	0.320	0.324	0.332
	192	**0.359**	0.362	0.362	0.361	0.362	0.373
	336	**0.388**	0.400	0.393	0.392	0.391	0.401
	720	0.450	**0.448**	0.455	0.462	0.453	0.468
ETTm2	96	**0.176**	0.313	0.178	0.186	0.176	0.189
	192	**0.238**	0.351	0.241	0.272	0.240	0.253
	336	0.306	0.315	0.312	0.377	**0.300**	0.311
	720	**0.390**	0.414	0.392	0.565	0.395	0.406

Qualitative behavior. FAPA follows the amplitude of the periodic peaks, whereas the backbone, which normalizes each window, flattens them (Figure 3). This is the qualitative counterpart of the $-$Unnorm ablation in Table 2.

4.3 Results Analysis

We examine three sources of the improvement: the cost of placing a period on the discrete grid of the target window, the dependence of the gain on the amplitude ratio, and the failure mode of the coupled baseline.

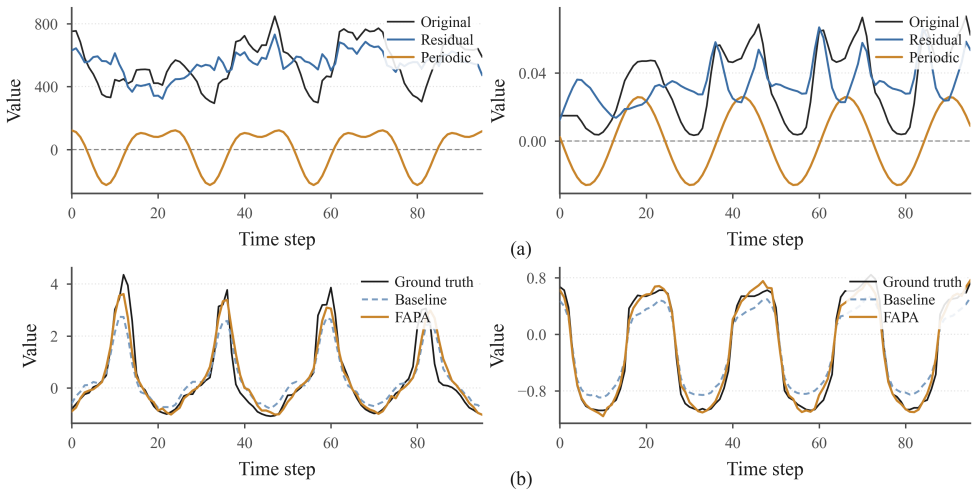

Figure 3: Decomposition and prediction at $T = 96$ on ECL and Traffic. (a) One decomposition window each from ECL (left) and Traffic (right); the original series, the periodic component, and the residual share one axis. (b) One prediction window each from Traffic (left) and ECL (right); the backbone with and without FAPA against the ground truth.

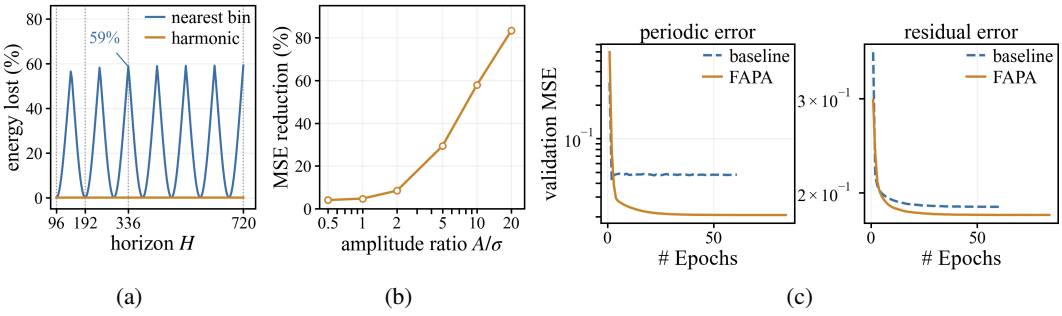

Figure 4: Sources of the improvement. (a) Energy lost by nearest-bin reconstruction and by harmonic projection when a $p = 96$ component is rebuilt on a horizon of length H. (b) Relative MSE reduction of FAPA over the backbone on synthetic series, against the amplitude ratio. (c) Validation error on ECL during training, in the periodic subspace and in its orthogonal complement.

Discretization loss on the target window. The reconstruction error on the target window is governed by the single quantity H/p. As Figure 4(a) shows, it vanishes whenever H/p is an integer, in which case the period lands on a bin. It peaks at half-integer values, where a nearest-bin reconstruction discards 59% of the component's energy, close to the asymptotic scalloping loss $1 - (2/\pi)^2$ of a half-bin offset (Harris, 1978). Harmonic projection is exact at every H/p. On the benchmarks this concerns ETTm1 and ETTm2, whose daily period of 96 steps gives $H/p = 3.5$ and 7.5 at $H = 336$ and 720; the daily period of ECL and Traffic (24 steps) divides every horizon.

Gain versus amplitude imbalance. The improvement grows monotonically with the amplitude ratio (Figure 4(b)), from 4% when the two components are comparable to 83% when the periodic component is twenty times larger, and it remains positive at every ratio. Synthetic series allow this ratio to be varied independently of every other factor: at each ratio the two variants are retrained from random initialization on a series whose periodic amplitude and noise level are fixed by construction (Appendix A.5).

Saturation of the coupled model. The coupled baseline fails mainly on the high-amplitude periodic component (Figure 4(c)). Projecting the validation error of each variant onto the periodic subspace and onto its orthogonal complement separates the two failure modes. The periodic-subspace error of the baseline stops decreasing within a few epochs and stays at more than twice that of FAPA,

whereas FAPA lowers the residual-subspace error by a much smaller margin. This is the component that instance normalization rescales and that a single set of weights must fit jointly with a residual an order of magnitude smaller.

4.4 Generalizability across Datasets and Backbones

We relate the size of the gain to simple statistics of the input window. Table 3 lists the sampling interval, the daily period in steps, the position consistency P_c of the selected frequencies, and the average change in MSE.

Spectral statistics. The position consistency is $P_c = \frac{1}{K}\sum_{j=1}^{K} q_{(j)}$, where q_f is the fraction of sample windows whose Top-K frequency set contains f and $q_{(j)}$ is the j-th largest of the q_f; it measures how consistently the same frequencies are selected across windows.

Table 3: Spectral statistics of the input window and the resulting average change in MSE over the eight backbones and $H \in \{96, 192, 336, 720\}$, where negative values indicate improvement. P_c is computed at the per-dataset K (K=2, except K=1 on Traffic).

Dataset	Sampling interval	Daily period	P_c	Avg. ΔMSE
ECL	1 h	24	0.90	-19.6%
Traffic	1 h	24	0.98	-18.9%
ETTm1	15 min	96	0.80	-3.5%
ETTm2	15 min	96	0.72	-0.1%
Weather	10 min	144	0.85	-5.1%

Differences across datasets. The two datasets with the highest P_c, Traffic (0.98) and ECL (0.90), show the largest improvements, and ETTm2, the lowest, is the only dataset on which FAPA stays on par with the backbone. ETTm1 and Weather show moderate gains (-3.5% and -5.1%). With five datasets we report these as descriptive observations rather than a fitted relation, and P_c does not measure the amplitude imbalance that governs the gain in the synthetic study of Section 4.3.

Architecture, scale, and channel structure. The gains on the five-dataset average range from 2.2% to 31.3% across the eight backbones, and what FAPA adds is the same for every architecture: 70.7 K parameters on ECL at $H = 96$, between 0.05% and 13% of the backbone for six of the eight backbones (Table 11). FAPA is effective at both extremes of this range: the two lightest backbones (FITS and SparseTSF, under 1 K parameters each) gain 22.0% on average, and TimesNet, the largest at 150 M, still gains 9.7% and improves on all five datasets. The periodic components we separate also carry more cross-channel structure than the original series, with a mean cross-channel correlation about 10% higher (Appendix I), and the channel-joint iTransformer and TSMixer still improve by 5.3% on average.

5 Conclusion

This paper proposes FAPA, a plug-and-play dual-branch framework that separates the periodic component of a series from its residual before prediction. FAPA predicts the periodic component at its own scale rather than inheriting it from the input window, refines its amplitudes in the frequency domain with AmpShape, and supervises the branch with a target obtained by projecting the ground truth onto continuous sinusoidal bases at those periods, which is exact for any horizon-to-period ratio and which we call Harmonic Period Alignment. The backbone is left to model only the residual, whose oscillations are much smaller than those of the original series. Across five benchmark datasets and eight backbones spanning four architectural families, FAPA lowers the MSE of every backbone on the five-dataset average, with reductions of 19.6% on ECL and 18.9% on Traffic and an MSE reduction of 10.7% overall, between 9% and 13% at every horizon. Controlled experiments further show that the gain grows with the amplitude imbalance between the two components, and that the failure of the coupled baseline lies in the periodic component rather than in the residual. Promising directions for future work include learnable frequency selection and automatic selection of the number of periods.

AI USE STATEMENT

Generative AI tools were used only to polish the writing of this manuscript. The authors take full responsibility for its content.

ETHICS STATEMENT

This work uses only publicly available benchmark datasets. It involves no human subjects, no personal or sensitive data, and no applications with foreseeable harmful impact.

REPRODUCIBILITY STATEMENT

All datasets used in this work are public benchmarks. Appendix A documents the datasets, the backbone configurations, the hyperparameters of FAPA, the training protocol, and the random seeds used in each run, and Appendix B reports the MSE and MAE for every horizon. The remaining appendices report the comparison against alternative decomposition operators, the efficiency measurements, the seed variance, the sensitivity analyses, and further decomposition and prediction examples (Appendices C–J). The code that reproduces our experiments is available in an anonymous repository at `https://anonymous.4open.science/r/wudisosjcb2xsju7`.

REFERENCES

George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 1970.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler-Canseco, and Artur Dubrawski. N-HiTS: Neural hierarchical interpolation for time series forecasting. In *AAAI*, 2023.

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

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In *IJCAI*, 2022.

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*, 1990.

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

Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. Dish-TS: A general paradigm for alleviating distribution shift in time series forecasting. In *AAAI*, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *COLM*, 2024.

Fredric J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2nd edition, 2018.

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

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient Transformer. In *ICLR*, 2020.

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

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting. In *NeurIPS*, 2019.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

Bryan Lim and Stefan Zohren. Time series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. SparseTSF: Modeling long-term time series forecasting with 1k parameters. In *ICML*, 2024.

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

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

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

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with Koopman predictors. In *NeurIPS*, 2023a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024a.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *ICML*, 2024b.

Zhiding Liu, Mingyue Cheng, Zhi Li, Zhenya Huang, Qi Liu, Yanhu Xie, and Enhong Chen. Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. In *NeurIPS*, 2023b.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

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

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

Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *NeurIPS*, 2018.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

Shun-Yao Shih, Fan-Keng Sun, and Hung-Yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8–9):1421–1441, 2019.

Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. RobustSTL: A robust seasonal-trend decomposition algorithm for long time series. In *AAAI*, 2019.

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

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

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with 10k parameters. In *ICLR*, 2024.

Weiwei Ye, Songgaojun Deng, Qiaosha Zou, and Ning Gui. Frequency adaptive normalization for non-stationary time series forecasting. In *NeurIPS*, 2024.

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

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

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

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

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

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

A Experimental Details

A.1 Experimental Settings

A.1.1 Model Parameters

Table 4 presents the core architectural parameters for each backbone model, including model dimension, attention heads, encoder layers, feed-forward dimension, and dropout rate. For PatchTST, we use patch length 16 and stride 8, with dropout applied to both the FC and head layers. For iTransformer, dropout is applied to the fully connected layers. TSMixer employs 8 mixing blocks with a feed-forward dimension of 256. The remaining backbones keep the configurations of their official implementations.

Table 4: Core architectural parameters of the eight backbone models (transformer-style settings). A dash means the backbone has no setting of this kind; its model-specific settings are listed in Table 5.

Model	d_{model}	n_{heads}	e_{layers}	d_{ff}	dropout
PatchTST	128	16	3	256	0.2
DLinear	–	–	–	–	–
iTransformer	512	8	2	2048	0.2
TSMixer	64	–	8	256	0.2
TimesNet	$256^{\dagger}$	–	2	$512^{\dagger}$	0.1
FITS	–	–	–	–	–
SparseTSF	–	–	–	–	–
FreTS	$128^{\ddagger}$	–	–	$256^{\ddagger}$	–

† Dataset-dependent; the value shown is that of ECL. See Table 5 for the value of each dataset. ‡ FreTS embedding and hidden size, respectively.

Table 5: Model-specific hyperparameters of the non-transformer backbones.

Model	**Model-specific setting**
DLinear	channel-independence on, kernel size 25
TimesNet	top-$k = 5$, num_kernels$= 6$; $d_{\text{model}}/d_{\text{ff}}$ are set per dataset and per horizon: ETTm1 $64/64$ ($H \in \{96, 192\}$) and $16/32$ ($H \in \{336, 720\}$); ETTm2 $32/32$ ($H \in \{96, 192, 336\}$) and $16/32$ ($H = 720$); Weather $32/32$; ECL $256/512$; Traffic $512/512$
FITS	channel-independence off; cutoff frequency selected automatically ($N = 20$ here)
SparseTSF	period length $= 4$, linear model
FreTS	channel-independence off, embed/hidden $= 128/256$

A.1.2 Training Configuration

Training follows two protocols, described in Appendix A.3.1: a shared setting for PatchTST, DLinear, iTransformer, TSMixer, and FITS, and the recipes released with their own papers for TimesNet, SparseTSF, and FreTS, which are architecturally sensitive to the schedule. The main experiments are repeated 3 times with different random seeds (2021, 2022, 2023) to assess the variability across runs.

A.1.3 Experimental Environment

All experiments are run on a server equipped with an NVIDIA A800-SXM4-80GB GPU and an Intel Xeon Platinum 8358P CPU.

A.2 Datasets

We use five publicly available benchmark datasets spanning energy, transportation, and meteorology (Table 6). ETTm1/ETTm2 record the oil temperature and load of power transformers, ECL the

electricity consumption of individual customers, Traffic the road occupancy of highway sensors in the San Francisco Bay Area, and Weather meteorological indicators.

Table 6: The five benchmark datasets.

Dataset	Channels	Sampling interval	Split
ETTm1	7	15 min	6:2:2
ETTm2	7	15 min	6:2:2
ECL	321	1 h	7:1:2
Traffic	862	1 h	7:1:2
Weather	21	10 min	7:1:2

A.3 Backbone Models

We attach FAPA to eight backbone models covering four architectural families: Transformer-based models, including PatchTST (Nie et al., 2023) and iTransformer (Liu et al., 2024a); MLP- and linear-based models, including DLinear (Zeng et al., 2023), TSMixer (Chen et al., 2023), and SparseTSF (Lin et al., 2024); the CNN-based TimesNet (Wu et al., 2023); and frequency-domain models, including FITS (Xu et al., 2024) and FreTS (Yi et al., 2023). The eight differ in whether they model channels jointly or independently, and in their parameter counts, which span more than five orders of magnitude (from 581 to 150 M).

A.3.1 Per-Backbone Training Protocols

The protocol is deliberately *not* uniform across backbones. The two groups are defined as follows, and the distinction matters when reading Table 1:

- **Shared protocol.** PatchTST, DLinear, iTransformer, TSMixer, and FITS are trained for 200 epochs with batch size 128, learning rate 10^{-4}, patience 15, and a constant-then-decaying schedule. For these five models the Base and +FAPA variants share *every* setting except FAPA itself, so the difference isolates the contribution of FAPA.
- **Official protocol.** TimesNet, SparseTSF, and FreTS follow the recipes released with their own papers, because they are architecturally sensitive to the schedule. TimesNet and FreTS use batch size 32, learning rate 10^{-4}, 10 epochs, patience 3, and a schedule that halves the learning rate after every epoch (`type1` in their code); SparseTSF uses batch size 256, learning rate 0.02, 30 epochs, and patience 5. Both variants of a given backbone always share the same recipe, so the comparison remains paired.

Table 5 collects the model-specific settings of the non-transformer backbones. TimesNet requires a separate remark: its official scripts do not set d_{model} and d_{ff} globally, but *per dataset and per horizon*, and we list the values we use in that table. Under the shared protocol, TimesNet trains less stably and its MSE increases on several settings, most notably on ETTm1 and ETTm2, so we follow its official recipe for both variants and compare them under the same schedule. The other backbones use a single configuration across datasets and horizons.

FITS is the backbone most sensitive to the input length. Its reported results are obtained by grid-searching the look-back window over $\{90, 180, 360, 720\}$ and keeping the value selected on the validation set, and its design relies on a long window to obtain a high frequency resolution (Xu et al., 2024); our protocol instead fixes $T = 96$ for every backbone, which is why its baseline errors in Table 1 are the largest of the eight.

A.4 Implementation Details

All experiments are run with PyTorch. The input length is $T = 96$ and the horizons are $H \in \{96, 192, 336, 720\}$. Each backbone keeps its own normalization strategy: PatchTST and TSMixer use RevIN (Kim et al., 2022) (affine=True); iTransformer uses its default instance normalization; TimesNet and FITS normalize each window by its mean and standard deviation; SparseTSF subtracts

the instance mean only; and DLinear and FreTS apply no normalization. The training settings and hyperparameters of the backbones are described in Appendix A.3.1.

For FAPA, the periodic loss of the training objective is weighted by λ, and the number of periods K balances decomposition granularity and model complexity: while a larger K captures more periodic components, excessive values increase the MLP burden and risk moving part of the residual into the periodic branch. We set $K = 1$ for Traffic, whose spectrum is concentrated in a single peak, and $K = 2$ for the other datasets; at the longest horizon ($H = 720$) every dataset uses $K = 1$. The rule follows from the position consistency of the input windows (Appendix G) rather than from a search over K, so it applies identically to every backbone and horizon. For λ we use a coarse grid, $\lambda \in \{0, 0.1, 0.2, 0.3, 1.0, 3.0\}$, and keep a single value per dataset for all eight backbones and all horizons. Appendices F and G analyze the effect of λ and K, respectively. Table 7 lists the FAPA-specific values used in the main experiments.

Table 7: FAPA-specific hyperparameters used in the main experiments.

Hyperparameter	**Value**
Number of periods K	1 (Traffic), 2 (others; 1 at H=720)
Periodic loss weight λ	ETTm1 (1.0), ETTm2 (3.0), ECL (0.1), Traffic (0.1), Weather (0.1)

A.5 Synthetic Data for the Analysis of Section 4.3

The synthetic experiment of Section 4.3 varies the amplitude ratio while holding every other factor fixed, so that the response can be attributed to the ratio rather than to a change of dataset. Each of the eight channels of a series is generated as

$$y_c(t) = A \sin\left(\frac{2\pi t}{p_1} + \phi_c^{(1)}\right) + 0.4A \sin\left(\frac{2\pi t}{p_2} + \phi_c^{(2)}\right) + \sigma\, r_c(t),$$

with a primary period $p_1 = 96$ steps, a secondary period $p_2 = 24$ steps, and phases $\phi_c^{(1)}, \phi_c^{(2)} \sim \mathcal{U}[0, 2\pi)$ drawn independently per channel. The irregular component r_c is a stationary AR(1) process,

$$r_c(t) = \rho\, r_c(t-1) + e_c(t), \qquad e_c(t) \sim \mathcal{N}\big(0,\ \sigma^2(1-\rho^2)\big)\,,$$

with $\rho = 0.7$, whose stationary variance is σ^2. We set $\sigma = 1$, so that A equals the amplitude ratio between the primary periodic component and the noise; because the pipeline standardizes every channel before training, this ratio is the only property of the series that reaches the model, and the sweep therefore varies it alone. Every series has 20,000 steps. The amplitude is swept over six values, $A \in \{0.5, 1, 2, 5, 10, 20\}$, and one set of phases and one noise realization are drawn once and reused at every value, so that the six settings differ only in A. At each value the backbone and the FAPA variant are retrained from random initialization under the protocol of Appendix A.3.1 (PatchTST, $T = 96$, $H = 336$), with the same architecture, optimizer, schedule, and seed. We report the relative reduction of the test MSE of FAPA over the backbone alone.

B Full Results

Table 8 reports the MSE for every horizon, and Table 9 the corresponding MAE.

C Comparison of Decomposition Operators

To test whether the gain stems from how the periodic component is defined rather than from separating the input into two components alone, we replace the separation stage of FAPA with three alternatives while leaving the rest of the framework unchanged. The alternatives are a moving-average decomposition, used by Autoformer (Wu et al., 2021) and DLinear (Zeng et al., 2023); a low-pass truncation of the spectrum, as in FITS (Xu et al., 2024); and a random selection of K frequencies, following the mode selection of FEDformer (Zhou et al., 2022b). Table 10 reports the resulting MSE.

Table 8: MSE for each horizon separately, with the same layout as Table 1. **Bold** indicates the better of the two results within each pair.

Dataset	H	PatchTST		DLinear		iTransformer		TSMixer		TimesNet		FITS		SparseTSF		FreTS	
		Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA
ETTm1	96	0.337	**0.318**	0.334	**0.327**	**0.338**	0.342	**0.323**	0.337	**0.335**	0.342	0.362	**0.329**	0.362	**0.326**	0.346	**0.335**
	192	0.373	**0.359**	0.370	**0.365**	0.383	**0.381**	**0.375**	0.376	0.389	**0.384**	0.407	**0.370**	0.396	**0.367**	0.386	**0.377**
	336	0.403	**0.388**	0.401	**0.397**	0.432	**0.413**	0.415	**0.403**	0.417	**0.412**	0.446	**0.402**	0.427	**0.397**	0.423	**0.411**
	720	0.460	**0.450**	0.463	**0.460**	0.496	**0.480**	0.486	**0.478**	0.482	**0.463**	0.513	**0.468**	0.485	**0.457**	0.492	**0.489**
ETTm2	96	0.177	**0.176**	**0.256**	0.268	**0.184**	0.188	**0.184**	0.186	**0.185**	0.191	0.186	**0.183**	0.186	**0.184**	0.217	**0.192**
	192	0.242	**0.238**	**0.431**	0.445	0.253	**0.252**	0.254	**0.249**	0.256	**0.248**	0.251	**0.243**	0.249	**0.245**	**0.265**	0.271
	336	0.307	**0.306**	**0.631**	0.641	**0.311**	0.322	**0.316**	0.318	**0.316**	0.319	0.311	**0.311**	**0.312**	0.321	0.347	**0.339**
	720	0.402	**0.390**	**0.976**	0.993	0.412	**0.405**	**0.417**	0.418	0.424	**0.409**	0.408	**0.395**	**0.407**	0.411	**0.534**	0.540
ECL	96	0.214	**0.182**	0.214	**0.187**	0.166	**0.165**	0.179	**0.160**	0.170	**0.153**	0.344	**0.194**	0.212	**0.177**	**0.192**	0.195
	192	0.216	**0.182**	0.211	**0.188**	0.174	**0.170**	0.195	**0.172**	0.188	**0.167**	0.324	**0.193**	0.203	**0.171**	**0.194**	0.197
	336	0.235	**0.198**	0.226	**0.204**	0.193	**0.187**	0.223	**0.186**	0.206	**0.178**	0.414	**0.211**	0.224	**0.192**	**0.208**	0.213
	720	0.276	**0.234**	0.262	**0.244**	0.229	**0.221**	0.258	**0.213**	0.286	**0.214**	0.654	**0.254**	0.263	**0.224**	**0.248**	0.249
Traffic	96	0.579	**0.541**	0.772	**0.678**	0.450	**0.444**	0.593	**0.499**	0.594	**0.485**	0.943	**0.650**	0.772	**0.586**	0.581	**0.570**
	192	0.564	**0.524**	0.726	**0.638**	0.457	**0.450**	0.594	**0.506**	0.622	**0.499**	0.929	**0.614**	0.701	**0.552**	0.583	**0.566**
	336	0.573	**0.535**	0.728	**0.645**	0.474	**0.466**	0.628	**0.523**	0.640	**0.511**	1.127	**0.636**	0.701	**0.570**	0.616	**0.587**
	720	0.609	**0.575**	0.770	**0.691**	0.511	**0.500**	0.699	**0.566**	0.677	**0.548**	1.702	**0.715**	0.748	**0.611**	0.677	**0.634**
Weather	96	0.174	**0.171**	**0.167**	0.171	0.181	**0.176**	**0.162**	0.164	0.176	**0.172**	0.202	**0.179**	0.200	**0.170**	0.188	**0.183**
	192	0.220	**0.214**	**0.213**	0.217	0.230	**0.217**	0.215	**0.209**	0.227	**0.222**	0.246	**0.221**	0.245	**0.213**	0.230	**0.223**
	336	0.273	**0.264**	**0.266**	0.270	0.285	**0.265**	0.276	**0.259**	0.281	**0.271**	0.296	**0.269**	0.293	**0.262**	0.276	**0.269**
	720	0.342	**0.333**	**0.339**	0.344	0.357	**0.331**	0.351	**0.324**	0.356	**0.337**	0.366	**0.332**	0.363	**0.329**	0.340	**0.334**

Table 9: MAE for each horizon separately, with the same layout as Table 8. **Bold** indicates the better of the two results within each pair.

Dataset	H	PatchTST		DLinear		iTransformer		TSMixer		TimesNet		FITS		SparseTSF		FreTS	
		Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA	Base	+FAPA
ETTm1	96	0.370	**0.360**	0.367	**0.363**	**0.372**	0.379	**0.364**	0.371	0.375	**0.375**	0.380	**0.364**	0.380	**0.368**	0.380	**0.376**
	192	0.392	**0.382**	0.385	**0.385**	**0.395**	0.397	0.393	**0.391**	0.402	**0.396**	0.405	**0.385**	0.395	**0.391**	0.401	**0.396**
	336	0.411	**0.404**	**0.406**	0.407	0.424	**0.419**	0.417	**0.412**	0.424	**0.413**	0.431	**0.406**	0.417	**0.413**	0.426	**0.421**
	720	0.443	**0.442**	**0.441**	0.443	0.462	**0.458**	**0.456**	0.457	0.459	**0.444**	0.470	**0.447**	0.448	**0.442**	**0.470**	0.471
ETTm2	96	**0.262**	0.263	**0.334**	0.341	**0.267**	0.275	**0.267**	0.273	**0.264**	0.277	0.269	**0.269**	**0.269**	0.270	0.320	**0.286**
	192	0.304	**0.303**	**0.431**	0.439	**0.312**	0.314	**0.312**	0.313	**0.308**	0.312	0.308	**0.306**	0.307	**0.307**	**0.333**	0.341
	336	**0.345**	0.353	**0.524**	0.528	**0.348**	0.364	**0.351**	0.361	**0.346**	0.363	**0.345**	0.356	**0.345**	0.364	0.393	**0.385**
	720	**0.401**	0.404	**0.658**	0.661	**0.406**	0.412	**0.407**	0.424	**0.408**	0.414	**0.399**	0.404	**0.398**	0.417	**0.503**	0.506
ECL	96	0.321	**0.273**	0.312	**0.281**	**0.257**	0.260	0.286	**0.266**	0.276	**0.258**	0.447	**0.285**	0.290	**0.268**	**0.280**	0.285
	192	0.323	**0.276**	0.313	**0.285**	**0.264**	0.267	0.300	**0.279**	0.291	**0.271**	0.427	**0.289**	0.289	**0.270**	**0.283**	0.289
	336	0.338	**0.293**	0.328	**0.302**	**0.282**	0.285	0.326	**0.293**	0.307	**0.283**	0.495	**0.307**	0.306	**0.288**	**0.298**	0.306
	720	0.367	**0.325**	0.357	**0.338**	**0.312**	0.320	0.351	**0.321**	0.362	**0.318**	0.629	**0.345**	0.338	**0.318**	**0.335**	0.336
Traffic	96	0.385	**0.350**	0.475	**0.425**	0.310	**0.308**	0.394	**0.345**	0.319	**0.309**	0.609	**0.403**	0.467	**0.375**	0.378	**0.373**
	192	0.379	**0.340**	0.454	**0.408**	0.309	**0.307**	0.402	**0.351**	0.333	**0.317**	0.603	**0.391**	0.436	**0.361**	0.374	**0.369**
	336	0.382	**0.345**	0.456	**0.412**	0.319	**0.317**	0.422	**0.362**	0.342	**0.321**	0.693	**0.406**	0.437	**0.367**	0.381	**0.378**
	720	0.397	**0.365**	0.475	**0.434**	0.343	**0.339**	0.450	**0.385**	0.362	**0.345**	0.891	**0.452**	0.460	**0.392**	0.406	**0.400**
Weather	96	**0.221**	0.226	0.241	**0.240**	**0.220**	0.232	**0.210**	0.223	**0.227**	0.231	0.243	**0.236**	0.239	**0.229**	0.241	0.241
	192	**0.260**	0.269	0.286	**0.285**	**0.261**	0.271	**0.258**	0.266	**0.266**	0.280	0.278	**0.274**	0.274	**0.268**	0.286	**0.282**
	336	**0.298**	0.308	0.328	**0.326**	**0.302**	0.310	**0.303**	0.305	**0.305**	0.318	0.313	**0.311**	0.309	**0.307**	0.321	**0.318**
	720	**0.344**	0.359	0.382	**0.379**	**0.351**	0.361	**0.351**	0.354	**0.352**	0.366	**0.359**	0.360	0.355	**0.355**	0.373	**0.366**
Average		0.347	**0.334**	0.398	**0.389**	**0.326**	0.330	0.351	**0.341**	0.336	**0.330**	0.450	**0.363**	0.358	**0.344**	0.359	0.359

Because every variant separates the input into two components, the comparison isolates the choice of separation operator from the act of separating. Top-K selection (ours) attains the lowest MSE in every column.

D EFFICIENCY

Table 11 reports the parameters added by FAPA. The extra parameters come from the MLP predictor and AmpShape. They are *backbone-independent*, because FAPA is attached to the input/output interface and is not inserted into the architecture. They depend only on the input length, the horizon, and the number of channels: at $T = 96$ and $H = 96$ they total 70.7 K, split into 39.2 K for the MLP predictor and 31.5 K for AmpShape, growing to 351.5 K at $H = 720$ since the predictor is sized by the horizon. Per window and channel, the decomposition costs $O(T \log T)$ and AmpShape

Table 10: Replacing the separation stage of FAPA by alternative decomposition operators, with PatchTST as the backbone and input length $T = 96$, at the per-dataset periodic loss weight λ of Table 7. All entries are MSE (lower is better). **Bold** marks the lowest MSE in each column.

Operator	ETTm1				ETTm2			
	H=96	H=192	H=336	H=720	H=96	H=192	H=336	H=720
Moving average	0.329	0.375	0.398	0.451	0.177	0.241	0.305	0.392
Low-pass truncation	0.338	0.374	0.406	0.485	0.188	0.274	0.350	0.532
Random-K frequency	0.318	0.363	0.416	0.471	0.183	0.244	0.307	0.396
Top-K (ours)	**0.318**	**0.359**	**0.388**	**0.450**	**0.176**	**0.238**	**0.304**	**0.389**

$O(H \log H)$, the MLP is linear in T and H, and the harmonic projection, used only in training, solves a $2K \times 2K$ system at a cost of $O(HK^2)$ with $K \leq 2$. Relative to the backbone at $H = 96$, the overhead ranges from 0.05% (TimesNet, 150 M) to 13% (PatchTST). The absolute overhead is the same 70.7 K in every row, including the two smallest backbones, FITS and SparseTSF (840 and 581 parameters), which also obtain the largest gains (Table 1).

Table 11: Parameter counts on ECL at input length $T = 96$ and $H = 96$, ordered by backbone size. The first row gives the size of FAPA itself, which is identical for every backbone; "Params" is the backbone alone and "+FAPA" the backbone with FAPA attached. FITS and SparseTSF are themselves very light backbones, under 1 K parameters each.

Backbone	**Params**	**+FAPA**
FAPA (ours)	–	70.7 K
TimesNet	150.30 M	150.38 M
iTransformer	6.40 M	6.48 M
DLinear	5.98 M	6.05 M
FreTS	3.24 M	3.31 M
TSMixer	2.09 M	2.16 M
PatchTST	0.549 M	0.620 M
FITS	0.84 K	71.5 K
SparseTSF	0.58 K	71.2 K

E Seed Variance

Table 12 reports the standard deviation of the full model across the three seeds used in the main table.

F Sensitivity to the Periodic Loss Weight

The periodic loss weight is set per dataset and held fixed across backbones and horizons. The split between the datasets that take the larger weights and those that take the smaller one appears to follow the number of periods spanned by the input window, T/p. The larger weights go to the datasets with $T/p \approx 1$, where the window covers about one full cycle: the period is then identifiable from the input and is also the principal structure of the series, so a stronger periodic loss is consistent with the larger gains. Where the window is shorter than one period ($T/p = 0.7$ on Weather) the period estimate is itself less reliable, and where it spans several ($T/p = 4$ on ECL and Traffic) the periodicity is regular enough for the backbone to model on its own; in both regimes the smaller weight is preferred. The two datasets at $T/p \approx 1$ do not take the same weight, and the difference appears to track how coherent their periodic structure is: ETTm2 has the lowest position consistency P_c of the five (Table 3), which points to a less stable period and a periodic component that is harder to learn from the data alone, so it is the dataset that takes the strongest supervision ($\lambda = 3.0$). We read this as an observation over five datasets rather than a fitted relation, and note that the ordering is confounded with other properties that co-vary across these benchmarks.

These values are a per-dataset choice rather than the per-setting optimum: one value is fixed for each dataset and held across all eight backbones and all four horizons, instead of searching separately for

Table 12: Standard deviation of the full model (+FAPA) on the test set, over the three seeds $\{2021, 2022, 2023\}$ used in the main table, on MSE and MAE, averaged over $H \in \{96, 192, 336, 720\}$.

Backbone	**ETTm1**	**ETTm2**	**ECL**	**Traffic**	**Weather**
		MSE			
PatchTST	0.0004	0.0009	0.0001	0.0007	0.0006
DLinear	0.0004	0.0018	0.0002	0.0009	0.0003
iTransformer	0.0016	0.0015	0.0002	0.0003	0.0006
TSMixer	0.0027	0.0015	0.0009	0.0012	0.0008
TimesNet	0.0040	0.0037	0.0030	0.0032	0.0034
FITS	0.0006	0.0003	0.0005	0.0019	0.0002
SparseTSF	0.0006	0.0020	0.0004	0.0097	0.0005
FreTS	0.0006	0.0036	0.0005	0.0006	0.0003
		MAE			
PatchTST	0.0005	0.0008	0.0001	0.0002	0.0007
DLinear	0.0002	0.0007	0.0002	0.0003	0.0002
iTransformer	0.0010	0.0012	0.0002	0.0001	0.0006
TSMixer	0.0014	0.0013	0.0009	0.0011	0.0008
TimesNet	0.0017	0.0029	0.0022	0.0020	0.0035
FITS	0.0004	0.0003	0.0006	0.0010	0.0003
SparseTSF	0.0008	0.0014	0.0005	0.0057	0.0009
FreTS	0.0006	0.0027	0.0006	0.0003	0.0006

each backbone and horizon. Such a search would lower the error a little further. The weight concerns how strongly the periodic branch is supervised rather than how much it improves the forecast: ETTm2 prefers the largest weight of the five. Table 13 reports the sweep with PatchTST and TSMixer as backbones. The value used is not the minimum of that table in 4 of its 10 rows, and no single per-dataset value can be optimal for every backbone: on ETTm1 the preferred λ is 1.0 for PatchTST but 0.3 for TSMixer. The dataset on which the periodic loss weight has the largest effect is ETTm2, and there the validation loss and the test loss select the same value, $\lambda = 3.0$.

Table 13: Sensitivity to the periodic loss weight λ with PatchTST and TSMixer as backbones. Each entry is the MSE averaged over the horizons run for that dataset; the values used in the main experiments are listed in Table 7.

PatchTST						
Dataset	λ=0	λ=0.1	λ=0.2	λ=0.3	λ=1.0	λ=3.0
ETTm1	0.3837	0.3812	0.3799	0.3791	0.3783	0.3806
ETTm2	0.3518	0.3325	0.3187	0.3087	0.2825	0.2770
ECL	0.1986	0.1989	0.1995	0.2001	0.2040	0.2087
Traffic	0.5516	0.5500	0.5493	0.5490	0.5502	0.5538
Weather	0.2484	0.2450	0.2432	0.2425	0.2463	0.2555
TSMixer						
Dataset	λ=0	λ=0.1	λ=0.2	λ=0.3	λ=1.0	λ=3.0
ETTm1	0.4064	0.4025	0.3994	0.3983	0.3991	0.3994
ETTm2	0.3440	0.3357	0.3298	0.3253	0.3034	0.2922
ECL	0.1852	0.1820	0.1829	0.1839	0.1884	0.1915
Traffic	0.5225	0.5210	0.5210	0.5214	0.5265	0.5337
Weather	0.2409	0.2381	0.2391	0.2400	0.2450	0.2527

G Sensitivity to the Number of Periods

The rule for K is fixed before training and uses only the input windows. Since the periodic branch shares one set of parameters across all windows, a period that is selected in some windows but not in others gives the branch a moving target; what matters is therefore whether the second period is

selected as consistently as the first, not the absolute level of P_c (Section 4.4). Admitting it changes P_c by at most 0.06 (ETTm1 $0.83 \to 0.80$, ETTm2 $0.78 \to 0.72$, ECL $0.89 \to 0.90$, Weather $0.88 \to 0.85$), so $K = 2$ is used, but it lowers P_c from 0.98 to 0.81 on Traffic, so $K = 1$ is used. ETTm2 has the lowest P_c of the five, but its two periods are selected almost equally often: its low consistency limits the size of the gain rather than the choice of K. At $H = 720$ every dataset uses $K = 1$. Over a long horizon the periodic structure of the target can drift away from the periods estimated on the input window, and each additional period adds a frequency-domain target that the branch is asked to satisfy over the whole horizon. Restricting the branch to a single period is the conservative choice: it keeps the strongest periodicity of the window as the anchor and avoids imposing corrections that a longer horizon may no longer support. Table 14 reports the MSE and MAE at $H = 96$ for each K: the error varies little across the range and the best value is $K = 1$ or $K = 2$ on every dataset, matching the rule of Table 7.

Table 14: Sensitivity to the number of periods K (MSE and MAE at $H = 96$, PatchTST). All entries are given to four decimals. **Bold** marks the best value in each column, including the Base row (PatchTST alone); ties are all bolded. Lower is better. The values used in the main experiments are listed in Table 7.

	ETTm1		ETTm2		ECL		Traffic		Weather	
	MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Base	0.3368	0.3695	0.1767	**0.2618**	0.2139	0.3208	0.5787	0.3851	0.1737	**0.2213**
FAPA K=1	0.3225	0.3626	**0.1761**	0.2632	0.1827	0.2735	**0.5413**	**0.3505**	0.1713	0.2258
FAPA K=2	**0.3182**	**0.3597**	**0.1761**	0.2626	**0.1815**	**0.2726**	0.5499	0.3556	**0.1711**	0.2247
FAPA K=3	0.3226	0.3628	0.1771	0.2636	0.1824	0.2733	0.5505	0.3568	0.1720	0.2260
FAPA K=4	0.3228	0.3602	0.1772	0.2633	0.1838	0.2741	0.5578	0.3628	0.1760	0.2307
FAPA K=5	0.3215	0.3599	0.1778	0.2641	0.1843	0.2749	0.5619	0.3668	0.1775	0.2320

Increasing K lets the branch represent more of the periodic energy of the window, which improves the fit, but it also allows periods that are less stable across windows, which degrades it; beyond the second period the second effect prevails. Since the effect of K is small, we keep the rule of Table 7 rather than tuning K per setting, and leave automatic selection of K open.

H Sensitivity to the Ridge Coefficient

The least-squares fit of Section 3.5.1 uses a ridge term λ_r for numerical stability, and is solved in double precision. Table 15 reports the effect of varying it over several orders of magnitude; the MSE is flat over the entire range.

Table 15: Sensitivity to the ridge coefficient λ_r used in the harmonic projection (PatchTST backbone, ECL, MSE averaged over $H \in \{96, 192, 336, 720\}$).

λ_r	10^{-8}	10^{-6}	10^{-4}	10^{-2}	10^{-1}
MSE	0.1989	0.1989	0.1989	0.1990	0.1991

I Channel Dependence of the Periodic Component

Figure 5 reports the Pearson correlation matrices of the original ECL series, of its periodic components, and of its residuals. It is the measurement behind the architectural discussion of Section 4.4.

J Further Decomposition and Prediction Examples

Figures 6 and 7 extend the qualitative examples of Figure 3 to a larger sample of test windows. All windows use the same selection criteria: the primary period spans between two and six full cycles of the window, it accounts for between 35% and 93% of the variance of the window, and the periodic

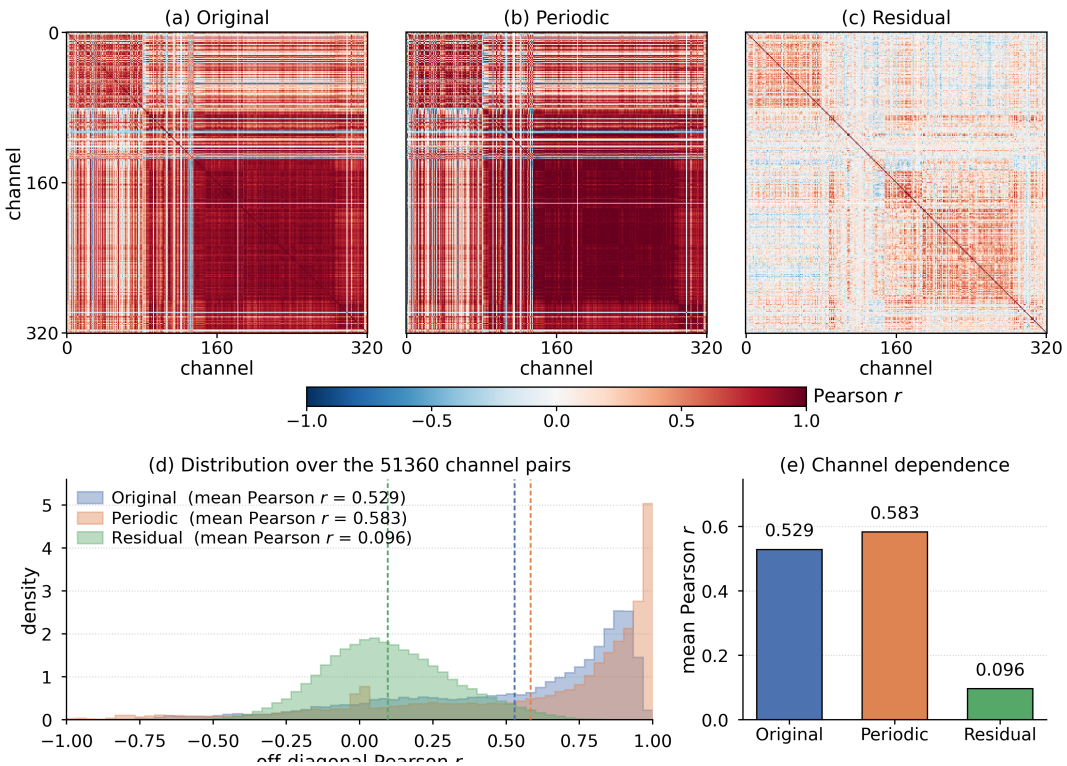

Figure 5: Channel dependence of the periodic component on ECL. (a) Pearson correlation matrices of the original series, (b) the periodic components and (c) the residuals, over the 321 channels of a test window ($T = 96$). (d) Distribution of the off-diagonal correlations, with dashed lines at the means. (e) Mean Pearson r.

component contains at least two local maxima. The decomposition panels use the same colors as Figure 3(a), and the prediction panels the same colors as Figure 3(b).

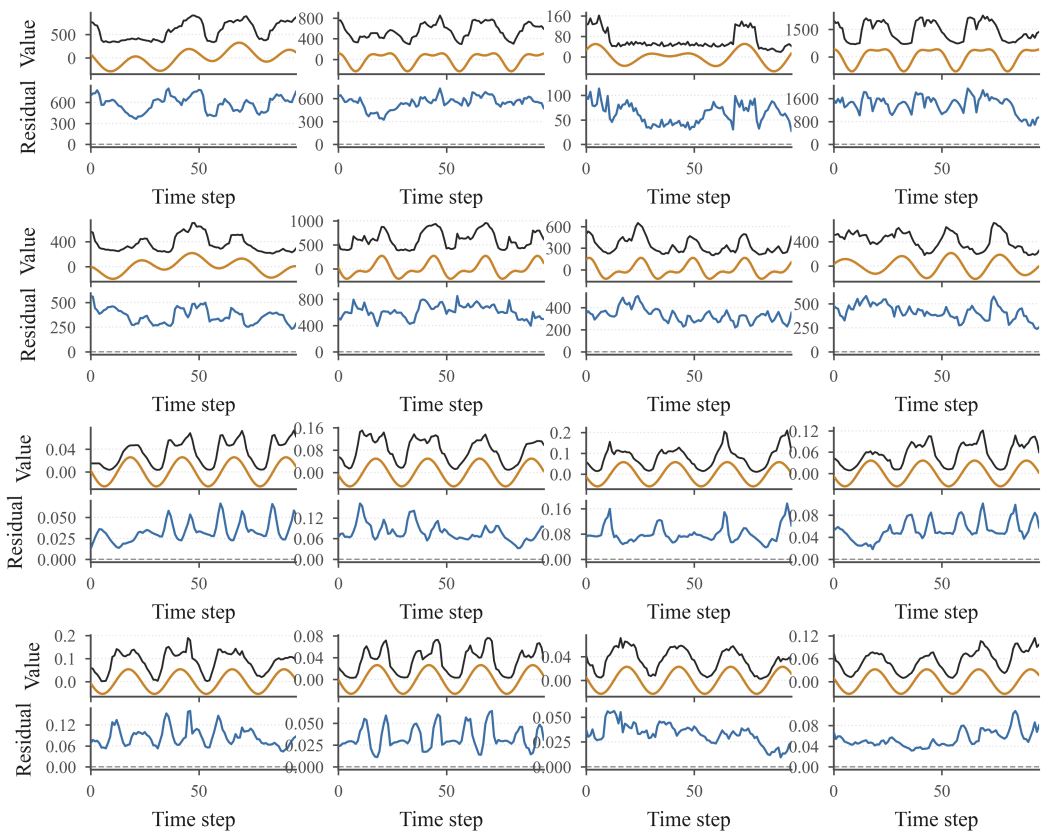

Figure 6: Decomposition of sixteen test windows at $T = 96$ (eight from ECL and eight from Traffic), one window per cell. Within each cell, the upper panel shows the original series with the periodic component, and the lower panel the residual. Blue: the residual; amber: the periodic component; black: the original series.

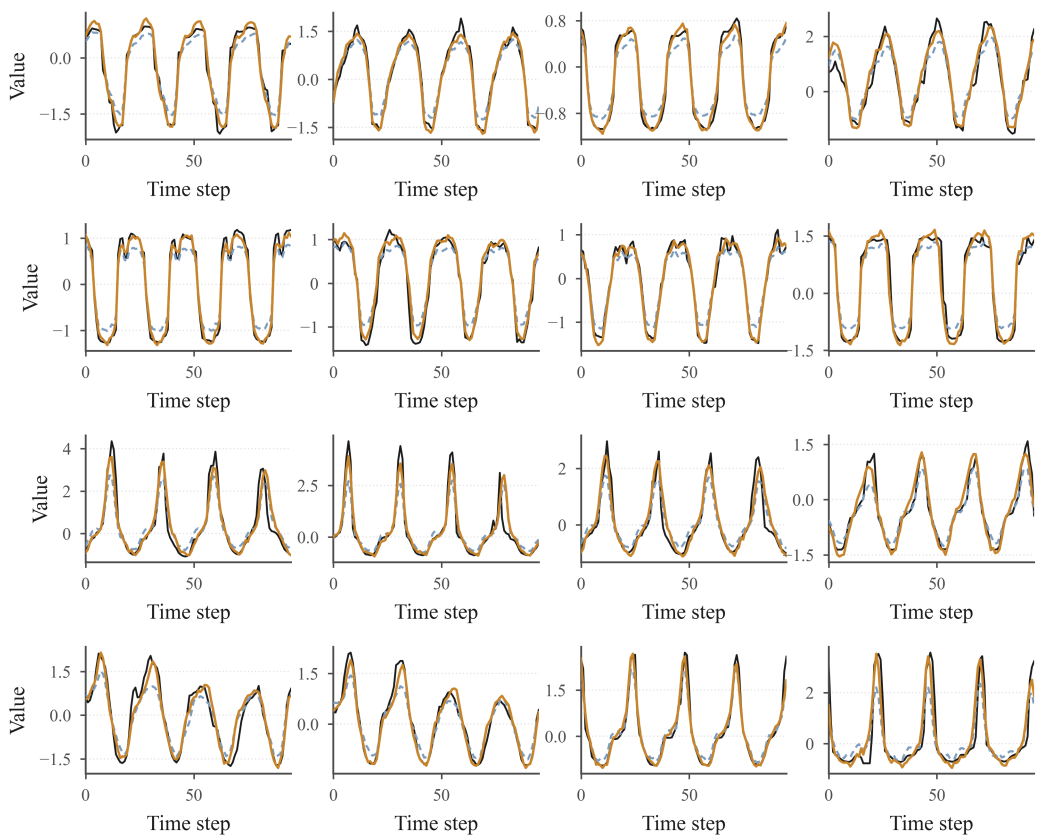

Figure 7: Prediction on sixteen test windows at $T = 96$ (eight from ECL and eight from Traffic). Blue (dashed): the backbone without FAPA; amber: the backbone with FAPA; black: the ground truth.