GATEDLINEAR: ADAPTIVE ROUTING OF COMPLEMENTARY LINEAR BASES FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series forecasting requires models to capture diverse temporal dynamics, from smooth trend continuation to nonstationary drift and strict phase-aligned recurrence. While recent deep learning models have improved accuracy, many process these diverse patterns through a shared computational backbone with fixed architectural inductive biases. Representative mixture-of-experts models introduce multiple experts but retain homogeneous expert architectures, leaving their forecasting roles to emerge through training. To address temporal heterogeneity directly at the level of prediction mechanisms, we propose **GatedLinear**: a lightweight framework that frames forecasting as the **adaptive routing of complementary linear bases**. GatedLinear leverages **three heterogeneous forecasting experts**: a global trend-seasonal basis for smooth projection, a difference-based incremental basis for nonstationary drift, and a phase-aligned recurrence basis for explicit cyclic reuse. These experts build distinct extrapolation priors directly into their forecasting operations. To combine their predictions, we introduce a **Tri-Factorized Fusion Gate** that decomposes routing logits into channel-specific preferences, horizon-aware offsets, and phase-indexed biases derived from known future time marks. This design enables **point-wise soft routing** across variables, forecast steps, and periodic phases without stacking computationally heavy neural modules. Experiments on eight standard benchmarks show that our method achieves the lowest average MSE and MAE among the evaluated baselines, while offering interpretable routing patterns and a favorable accuracy–cost trade-off.

1 INTRODUCTION

Time series forecasting is a fundamental problem in many real-world systems, including energy management, traffic control, weather prediction, finance, and industrial monitoring. The central challenge is not merely to fit historical observations, but to extrapolate future values under multiple temporal mechanisms. A future trajectory may continue a smooth trend, evolve through short-term increments, repeat a phase-aligned periodic pattern (i.e., exact historical recurring points such as 2:00 PM daily), or combine these behaviors in different proportions across variables and forecast horizons.

Forecasting research has progressed from statistical models, including ARIMA (Box et al., 2015), vector autoregression (Toda & Phillips, 1994), and exponential smoothing (Hyndman et al., 2008), to neural models based on recurrence (Salinas et al., 2020; Lai et al., 2018; Lin et al., 2025b) and convolution (Bai et al., 2018; Liu et al., 2022). More recent approaches emphasize different temporal priors: linear and MLP-based methods exploit decomposition, multi-scale mixing, or distribution alignment (Zeng et al., 2023; Wang et al., 2024; Hu et al., 2025); Transformer-based methods model broader dependencies through attention (Wu et al., 2021; Nie et al., 2023; Liu et al., 2024; Niu et al., 2025); and frequency- or periodicity-oriented methods target recurring structure (Yi et al., 2024; Wu et al., 2023; Zhang et al., 2026; Lin et al., 2025a). These advances show that different forecasting mechanisms can be useful, depending on the temporal pattern.

Despite this progress, many advanced methods process diverse temporal patterns through a backbone with fixed architectural inductive biases. Recent mixture-of-experts (MoE) forecasters introduce multiple experts, but representative models such as Time-MoE (Shi et al., 2025) and Moirai-MoE (Liu et al., 2025) route temporal representations among structurally homogeneous feed-forward experts. The lightweight MoLE (Ni et al., 2024) similarly combines multiple instances of a linear-centric

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.

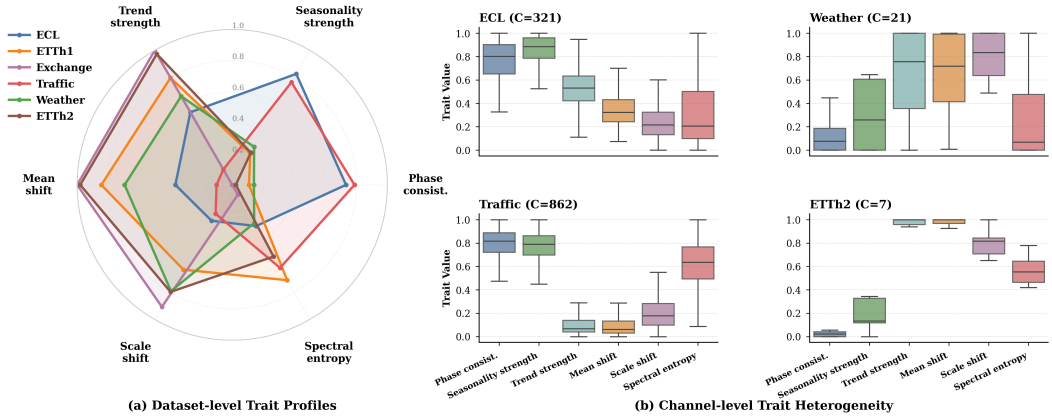


Figure 1: Trait heterogeneity in time series forecasting benchmarks. (a) Different datasets exhibit distinct profiles across temporal traits, suggesting that no single forecasting mechanism is uniformly appropriate across datasets. (b) Even within one dataset, variable channels can have broad trait distributions, indicating substantial channel-level heterogeneity. The six temporal traits are defined in the Appendix E.

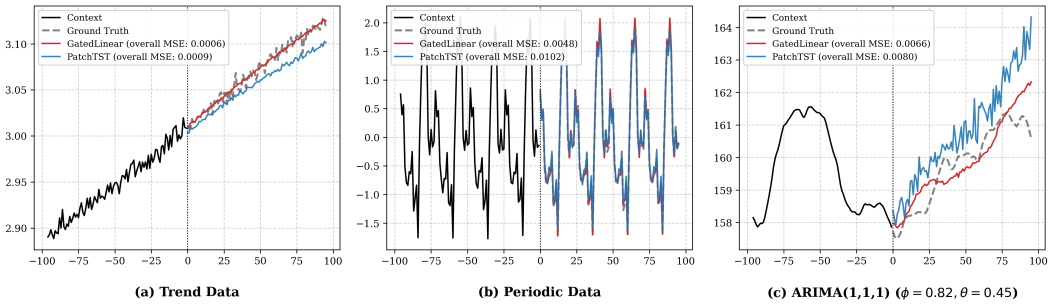


Figure 2: Performance comparison between PatchTST and GatedLinear on data with distinct temporal characteristics: (a) Trend Data with smooth continuous evolution, (b) Periodic Data with strict phase-aligned recurrence, and (c) ARIMA(1,1,1) data with nonstationary drift.

predictor. In these approaches, experts specialize through learned parameters within a shared architectural form, while distinct extrapolation mechanisms are not explicitly built into the expert pool. This suggests using experts with different forecasting mechanisms, but leaves open how such experts can be trained and combined effectively.

Real-world time series exhibit precisely this diversity of temporal dynamics. As shown in Figure 1, datasets differ substantially in trend strength, exact periodicity, and local drift, and even variables within the same dataset can follow distinct temporal regimes. SynTSBench (Tan et al., 2025) further supports this view by constructing synthetic datasets with programmable temporal patterns and showing that few deep forecasting models perform uniformly well across all pattern types. A stable daily pattern benefits from phase-aligned reuse—directly looking up the corresponding phase from historical cycles; a nonstationary drifting signal benefits from predicting increments relative to its latest state; and a smooth trajectory favors direct trend-seasonal projection. Relying on a single mechanism can lead to compromised predictions. For instance, in the examples shown in Figure 2, PatchTST underestimates the trend, attenuates recurring peaks, and overestimates the nonstationary drift, while GatedLinear more closely follows the corresponding trajectories. These observations motivate a pool of experts with complementary extrapolation priors.

Simply placing unrelated forecasting models behind a gate does not ensure that their strengths will be used. Experts with different architectures can learn at different rates and produce loss

gradients of different magnitudes. In a soft mixture, an expert assigned a small weight also receives a smaller forecast-loss gradient; an early routing preference can therefore reinforce itself, leaving other experts undertrained or effectively unused—a form of expert collapse. Our controlled comparison in Section 4.3 illustrates the practical difficulty: combining DLinear, PaiFilter, and TimeMixer under the same gate yields worse average forecasting errors than the strongest standalone expert. Architectural diversity alone is insufficient; the expert pool and the routing mechanism must be designed together.

We address this challenge with **GatedLinear**, which uses three **complementary linear forecasting bases** as heterogeneous experts. Each basis has a distinct extrapolation rule, while all retain a similarly lightweight linear form rather than mixing full models with disparate capacities. The global trend–seasonal basis directly projects decomposed components for smooth continuation. The difference-based basis predicts increments and integrates them from the last observation to track local drift. The phase-aligned basis reuses corresponding positions in recent cycles and learns a linear residual for deviations from exact recurrence. This combination gives each expert a specific forecasting role and a common level of structural simplicity.

To make these complementary predictions useful together, we introduce a **Tri-Factorized Fusion Gate**. It forms a soft mixture at each forecast point from a **channel-specific branch preference**, a **horizon-channel correction**, and a **future-phase-indexed correction** derived from known time marks. Thus, one branch need not dominate every variable, prediction distance, and periodic phase; the gate can assign different mixtures to contexts where different mechanisms are useful. Each branch predicts from the observed history, while the gate uses the forecast context to combine their outputs. The result is structured, interpretable routing among distinct mechanisms without stacking heavy forecasting networks.

Figure 3 illustrates the overall architecture of the proposed framework, including the three forecasting bases and the Tri-Factorized Fusion Gate. Our contributions are summarized as follows:

- We design three complementary linear forecasting experts for global trend–seasonal projection, difference-based incremental evolution, and phase-aligned recurrence. Their distinct extrapolation priors and comparable structural simplicity address the difficulty of combining heterogeneous full-model experts.
- We introduce a **Tri-Factorized Fusion Gate** that assigns forecast-point-level soft weights using channel, horizon, and future-phase information, enabling the experts to contribute in different temporal contexts.
- We show that this coordinated expert-pool and gate design achieves the lowest average MSE and MAE across eight evaluated benchmarks with a favorable accuracy–cost trade-off. Expert-design comparisons and ablations support the contributions of both the complementary bases and structured routing.

2 Problem formulation

Given a multivariate time series $\mathbf{X}_{t-L+1:t} \in \mathbb{R}^{L\times C}$ with L historical observations and C variables, the goal is to predict future values $\hat{\mathbf{Y}}_{t+1:t+H} \in \mathbb{R}^{H\times C}$, where H denotes the prediction length. The forecasting task can be formalized as learning a mapping

$$f : \mathbb{R}^{L\times C} \to \mathbb{R}^{H\times C}, \qquad \hat{\mathbf{Y}}_{t+1:t+H} = f(\mathbf{X}_{t-L+1:t}). \tag{1}$$

Our goal is to design a lightweight forecasting function f that can adapt its predictive mechanism across variables, forecast steps, and future temporal phases. For notational simplicity, we drop the global timestamp t in the following sections and locally index the historical window as $\mathbf{X}_{1:L}$ and the future horizon as $\hat{\mathbf{Y}}_{1:H}$.

3 Methodology

This section describes the three forecasting bases and the structured fusion gate in detail.

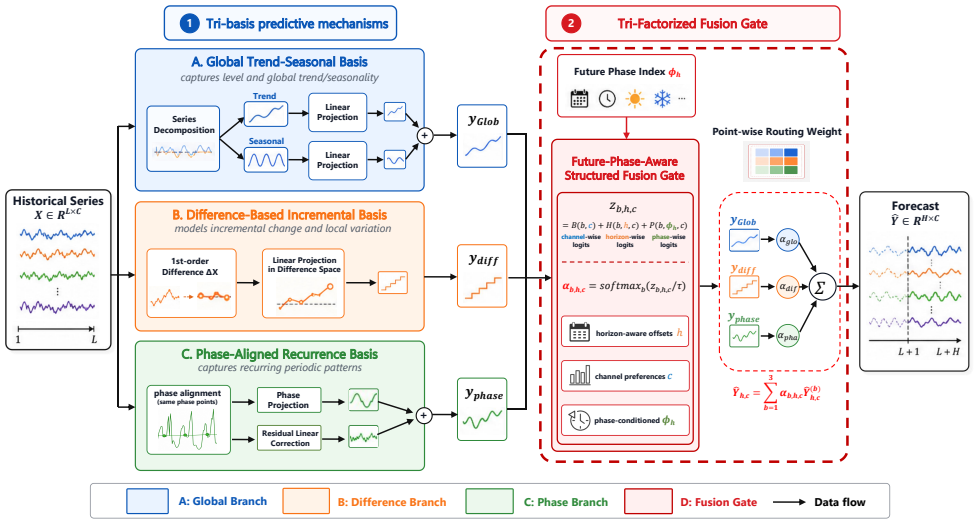


Figure 3: Overview of the proposed GatedLinear framework. The input sequence is processed by three complementary linear forecasting bases. A Tri-Factorized Fusion Gate uses horizon, channel, and future phase indices to select among the three bases for each future point.

3.1 TRI-BASIS LINEAR FORECASTING

Our model instantiates the forecasting function f using three complementary linear bases. Each basis generates a candidate future sequence $\hat{\mathbf{Y}}^{(k)} \in \mathbb{R}^{H\times C}$ for $k \in \{1, 2, 3\}$, corresponding to the Global Trend-Seasonal Basis, Difference-Based Incremental Basis, and Phase-Aligned Recurrence Basis, respectively. A fusion gate simultaneously yields mixing weights $\mathbf{W} \in \mathbb{R}^{H\times C\times 3}$, such that $\sum_{k=1}^{3} \mathbf{W}_{h,c,k} = 1$ for all future steps $h \in \{1, \ldots, H\}$ and channels $c \in \{1, \ldots, C\}$. The final sequence prediction is a convex combination of the branches:

$$\hat{\mathbf{Y}}_{h,c} = \sum_{k=1}^{3} \mathbf{W}_{h,c,k} \hat{\mathbf{Y}}^{(k)}_{h,c}. \tag{2}$$

Here, $\hat{\mathbf{Y}}_{h,c}$ denotes the forecasted value at targeted future step h for variable c, and $\mathbf{W}_{h,c,k}$ is the weight confidently assigned to basis k for that specific point.

Global Trend-Seasonal Basis ($k = 1$). The global basis is designed to learn smooth trend and seasonal structures that can be directly extrapolated over the forecast horizon. Following decomposition-based forecasting paradigms, the historical input is split into seasonal and trend components via moving-average operations:

$$\mathbf{X}^s, \mathbf{X}^t = \mathrm{Decomp}(\mathbf{X}), \tag{3}$$

where $\mathbf{X}^s, \mathbf{X}^t \in \mathbb{R}^{L\times C}$ isolate local cyclical fluctuations and global trends from the input $\mathbf{X}$. Two linear mappings then project these components from the historical length L directly to the entire future horizon H:

$$\hat{\mathbf{Y}}^{(1)}_{:,c} = \mathcal{L}_s(\mathbf{X}^s_{:,c}) + \mathcal{L}_t(\mathbf{X}^t_{:,c}), \quad \forall c \in \{1, \ldots, C\}, \tag{4}$$

where $\mathcal{L}_s, \mathcal{L}_t : \mathbb{R}^L \to \mathbb{R}^H$ are learnable temporal affine layers operating over individual channels.

Difference-Based Incremental Basis ($k = 2$). The difference basis targets nonstationary series with level shifts or persistent local drift, where predicting changes can be more stable than predicting absolute levels. We compute the first-order differences of the input series:

$$\Delta\mathbf{X}_{\tau,c} = \mathbf{X}_{\tau,c} - \mathbf{X}_{\tau-1,c}, \quad \forall \tau \in \{2, \ldots, L\}, \tag{5}$$

with $\Delta\mathbf{X}_{1,c} = 0$. Using a temporal linear projection $\mathcal{L}_d : \mathbb{R}^L \to \mathbb{R}^H$, this branch forecasts all future increments sequentially. The target future levels $\hat{\mathbf{Y}}^{(2)}$ are explicitly reconstructed via cumulative summation, integrated starting strictly from the last historically observed value $\mathbf{X}_{L,c}$:

$$\hat{\mathbf{Y}}^{(2)}_{h,c} = \mathbf{X}_{L,c} + \sum_{i=1}^{h} \left[\mathcal{L}_d(\Delta\mathbf{X}_{:,c})\right]_i , \tag{6}$$

where i indexes the predicted increments up to step h. Anchoring their cumulative sum at the last observation makes this branch equivariant to additive level shifts: a constant shift in the history produces the same shift in its forecast. Appendix C illustrates this property under a random walk with drift.

Phase-Aligned Recurrence Basis ($k = 3$). The phase-aligned basis complements the other two bases by mapping fine-grained periodic details from recent cycles to their corresponding future phases. For a given time series with characteristic base period P, assuming the historical window is sufficiently long ($L \geq K \cdot P$), we summarize the last K complete periods into a unified historical template $\mathbf{T} \in \mathbb{R}^{P\times C}$. For each recurring phase $r \in \{0, \ldots, P-1\}$:

$$\mathbf{T}_{r,c} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{X}_{L-KP+(k-1)P+r,c}. \tag{7}$$

This exact template is subsequently tiled forward across the desired prediction length H to yield the unparameterized baseline sequence $\mathbf{B}^{(p)} \in \mathbb{R}^{H\times C}$. We fine-tune this cyclical projection by supplementing an additive linearly modeled deformation residual $\mathcal{L}_p(\mathbf{X})$:

$$\hat{\mathbf{Y}}^{(3)}_{:,c} = \mathbf{B}^{(p)}_{:,c} + \mathcal{L}_p(\mathbf{X}_{:,c}). \tag{8}$$

The template provides a same-phase forecast prior, while the residual absorbs mild deviations from exact recurrence.

Notably, all three bases share the exact same structural simplicity: they are constructed solely from fundamental mathematical operations (difference, moving-average, phase alignment) combined with one-layer temporal linear mappings, entirely free of heavy neural modules.

3.2 Tri-Factorized Fusion Gate

To invoke the bases adaptively, we equip our framework with a structured gating mechanism yielding the mixing coefficients $\mathbf{W}$. We designate this as a "*Tri-Factorized Fusion Gate*" because the gate separates three lookup contexts: the variable channel, the forecast horizon, and the future phase index associated with the target timestamp. For example, under an hourly daily period, the gate can treat a noon forecast differently from a midnight forecast even when the two targets have the same horizon length. This fine-grained, localized alignment enables point-wise mechanism selection.

Accordingly, for future step $h \in \{1, \ldots, H\}$, given channel c, and candidate basis k, the unnormalized preference logit $z_{h,c,k}$ is structured into three disjoint learnable parameters:

$$z_{h,c,k} = a_{c,k} + u_{h,c,k} + v_{\phi_h,c,k}, \tag{9}$$

with respective meanings assigned as follows:

- $a_{c,k} \in \mathbb{R}$ conveys a persistent channel-specific base tendency, enabling varying dominances among different monitored variables.
- $u_{h,c,k} \in \mathbb{R}$ implements a horizon-dependent scalar offset, reflecting that smooth direct bases inherently excel farther out while increment predictions thrive primarily under tight proximities.
- $v_{\phi_h,c,k} \in \mathbb{R}$ is a phase-indexed selection bias, selected by the future phase index $\phi_h \in \{0, \ldots, P-1\}$ associated with the target timestamp. This index is obtained from known decoder-side time marks when available, or from the horizon offset modulo the period otherwise.

These structured logits are ultimately normalized into bounded fusion probabilities via a parametric softmax distribution ruled by a temperature hyperparameter τ:

$$\mathbf{W}_{h,c,k} = \frac{\exp(z_{h,c,k}/\tau)}{\sum_{j=1}^{3} \exp(z_{h,c,j}/\tau)}. \tag{10}$$

Reducing τ intrinsically prevents ambiguous blending and compels the gate to isolate the most reliable basis dominantly.

Remarkably, our constructed framework entirely circumvents weighty recursive networks or excessive cross-attention modules. It establishes a compact selection scheme governed by structured lookups over channel, horizon, and phase index.

Here, adaptivity refers to learning a dataset-specific fusion rule over complementary bases; input-conditioned routing may overreact to short-term fluctuations and was less accurate on average in our comparison (Appendix G.2).

3.3 Training objective and complexity

We train all forecasting bases and the fusion gate jointly end-to-end by minimizing the mean squared error (MSE) between fused forecasts and ground-truth values, averaged over batch samples, forecast steps, and channels.

Complexity analysis. The four channel-shared temporal affine layers use $\mathcal{O}(LH)$ parameters in total, while the gate adds $3C(1 + H + P)$ lookup parameters; the overall parameter complexity is therefore $\mathcal{O}(LH + C(H + P))$. For batch size B, the temporal projections dominate the forward cost at $\mathcal{O}(BCLH)$, with an additional $\mathcal{O}(BCH)$ for gate lookup and fusion. Appendix D provides the detailed accounting.

4 Experiments

4.1 Experimental setup

Datasets. We evaluate on standard forecasting benchmarks, including ETTm1, ETTm2, ETTh1, ETTh2, Electricity, Traffic, Weather, and Exchange (Zhou et al., 2021; Wu et al., 2021; Lai et al., 2018). Unless otherwise stated, each dataset is split into train/validation/test subsets with a ratio of 7:1:2. We use an input length of $L = 336$ and evaluate prediction lengths $H \in \{48, 96, 192, 336\}$. For datasets with clear daily or weekly periodicity, we set the period length according to the data frequency; otherwise, we use a validation-selected period.

Baselines. We compare against the same representative models reported in Table 1: DLinear (Zeng et al., 2023), TimeMixer (Wang et al., 2024), MixLinear (Ma et al., 2024), PaiFilter (Yi et al., 2024), PatchTST (Nie et al., 2023), PhaseFormer (Niu et al., 2025), CycleNet (Lin et al., 2024), TQNet (Lin et al., 2025a), TimeAlign (Hu et al., 2025), and iTransformer (Liu et al., 2024). These baselines cover decomposition-based linear models, multi-scale MLP forecasting, patch- and phase-based temporal modeling, frequency filtering, temporal-query mechanisms, distribution-aware alignment, and inverted Transformer architectures. Detailed descriptions and source repositories are provided in the appendix.

Implementation details. All experiments are implemented in PyTorch 2.6.0 based on the Time-Series-Library training pipeline and conducted on a single NVIDIA A800 GPU. We train GatedLinear with Adam using mean squared error (MSE) loss, and report mean absolute error (MAE) and MSE as evaluation metrics. Further implementation details are provided in Appendix F.4.

4.2 Main results

Table 1 shows that GatedLinear achieves the best result on 11 out of 16 dataset-level metrics across eight benchmarks, together with the lowest average MSE and MAE. Although it does not lead on every dataset, it remains competitive against recent linear, frequency-based, and Transformer-style

Table 1: Main forecasting results (averaged over prediction lengths $H \in \{48, 96, 192, 336\}$ with input length $L = 336$). The best results are shown in **red bold**, and the second-best in blue underline.

Dataset	GatedLinear (Ours)		DLinear (AAAI 2023)		TimeMixer (ICLR 2024)		MixLinear (ICLR 2026)		PaiFilter (NeurIPS 2024)		PatchTST (ICLR 2023)		PhaseFormer (ICLR 2026)		CycleNet (NeurIPS 2024)		TQNet (ICML 2025)		TimeAlign (ICLR 2026)		iTransformer (ICLR 2024)	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.138**	**0.231**	0.148	0.243	0.143	0.235	0.159	0.254	0.141	0.235	0.147	0.250	0.163	0.255	0.138	0.232	0.141	0.236	0.141	0.236	0.143	0.237
ETTh1	0.404	0.414	0.388	0.403	0.400	0.415	**0.383**	**0.399**	0.433	0.440	0.401	0.417	0.400	0.407	0.398	0.412	0.391	0.409	0.402	0.415	0.459	0.452
ETTh2	**0.300**	**0.356**	0.300	0.358	0.328	0.379	0.314	0.360	0.347	0.392	0.329	0.379	0.326	0.372	0.320	0.370	0.315	0.368	0.320	0.367	0.382	0.402
ETTm1	**0.305**	**0.344**	0.322	0.356	0.325	0.366	0.338	0.371	0.330	0.369	0.317	0.362	0.540	0.484	0.311	0.359	0.313	0.355	0.307	0.350	0.357	0.385
ETTm2	**0.196**	**0.273**	0.197	0.274	0.212	0.288	0.209	0.288	0.210	0.281	0.208	0.286	0.241	0.320	0.208	0.283	0.201	0.276	0.205	0.277	0.240	0.316
Exchange	**0.161**	0.277	0.170	**0.271**	0.184	0.287	0.168	0.281	0.196	0.294	0.172	0.281	0.188	0.298	0.184	0.286	0.193	0.294	0.171	0.278	0.211	0.312
Traffic	0.405	0.272	0.416	0.283	0.383	0.267	0.423	0.284	0.395	0.278	0.383	0.270	0.428	0.267	0.411	0.290	0.381	0.266	0.396	0.276	**0.367**	**0.262**
Weather	**0.173**	**0.218**	0.198	0.240	0.178	0.225	0.207	0.252	0.179	0.226	0.179	0.221	0.208	0.257	0.176	0.222	0.182	0.226	0.174	0.218	0.191	0.235
Average	**0.260**	**0.298**	0.267	0.303	0.269	0.308	0.275	0.311	0.279	0.314	0.267	0.308	0.312	0.333	0.268	0.307	0.265	0.304	0.264	0.302	0.294	0.325

Table 2: Forecasting expert designs on eight datasets ($L = 336$, $H = 96$). GatedDLinear uses three DLinear experts; GatedMixMoE uses DLinear, PaiFilter, and TimeMixer as three heterogeneous expert branches. Both use the same Tri-Factorized Fusion Gate as GatedLinear. Average is over all eight datasets. Lower is better; **red bold** marks the best value for each metric and dataset.

Dataset	*Expert 1: DLinear*		*Expert 2: PaiFilter*		*Expert 3: TimeMixer*		GatedMixMoE (Hetero Ensemble)		GatedDLinear (Homo Ensemble)		GatedLinear (Ours)	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	0.1411	0.2366	0.1317	0.2271	0.1353	0.2271	0.1307	0.2250	0.1300	0.2234	**0.1286**	**0.2220**
ETTh1	0.3758	0.3967	0.4343	0.4365	0.3950	0.4112	0.3882	0.4057	0.3817	0.3973	**0.3704**	**0.3914**
ETTh2	**0.2783**	**0.3390**	0.3235	0.3719	0.3114	0.3665	0.3694	0.3943	0.2935	0.3480	0.2834	0.3434
ETTm1	0.3018	0.3435	0.3103	0.3552	0.3063	0.3539	0.3193	0.3652	0.2881	0.3353	**0.2836**	**0.3301**
ETTm2	0.1640	0.2522	0.1805	0.2659	0.1792	0.2613	0.1755	0.2604	0.1650	0.2539	**0.1624**	**0.2508**
Exchange	0.0883	**0.2045**	0.0991	0.2206	0.0969	0.2226	0.0923	0.2178	0.0930	0.2103	**0.0836**	0.2054
Traffic	0.4106	0.2805	0.3883	0.2759	**0.3707**	**0.2584**	0.3764	0.2676	0.3901	0.2753	0.3967	0.2663
Weather	0.1745	0.2239	0.1511	0.2044	0.1549	0.2086	0.1512	0.2039	0.1470	0.1970	**0.1448**	**0.1956**
Average	0.2418	0.2846	0.2524	0.2947	0.2437	0.2887	0.2504	0.2925	0.2361	0.2801	**0.2317**	**0.2756**

baselines. These results support the adaptive combination of global, incremental, and phase-aligned linear bases as a lightweight approach to forecasting across heterogeneous temporal patterns.

4.3 Design of Forecasting Experts

Table 2 tests whether the benefit comes from gating alone or from the design of the expert pool. All three gated models use the same Tri-Factorized Fusion Gate. GatedDLinear repeats DLinear three times and improves on standalone DLinear on average, showing that a gate can benefit even homogeneous experts. Yet it remains behind GatedLinear, so this gain does not explain the full advantage of our design.

GatedMixMoE uses DLinear, PaiFilter, and TimeMixer as three heterogeneous expert branches, but usually fails to match the best standalone expert and sometimes underperforms all three. This pattern is consistent with routing that favors one expert or blends conflicting predictions into a compromise, rather than reliably exploiting the most suitable expert. Architectural diversity alone therefore does not guarantee an effective mixture. GatedLinear instead pairs the same gate with three purpose-designed, complementary mechanisms for global projection, incremental evolution, and phase-aligned recurrence. Its stronger overall results support this structured three-expert design. The following ablations examine the contribution of each mechanism and the gate.

4.4 Ablation Studies

We conduct five groups of ablation experiments to examine the contribution of the forecasting bases and the structured fusion gate. Table 3 reports the results on four representative datasets. The first three variants remove the global trend–seasonal branch, the difference-based incremental branch, and the phase-aligned recurrence branch, respectively, testing whether each basis provides useful complementary information. The fourth variant removes the phase-indexed logits from the fusion gate, which disables explicit future phase-aware routing. The fifth variant replaces the proposed Tri-Factorized Fusion Gate with a plain softmax gate over the three branches, whose weights are shared across horizons and variables.

Table 3: Ablation results on ECL, ETTm1, Traffic, and Weather. Numbers are averaged over prediction lengths $H \in \{48, 96, 192, 336\}$ with input length $L = 336$. Avg. Inc. reports the average relative increase (%) over all eight MSE/MAE entries compared with the full model.

Variant	ECL		ETTm1		Traffic		Weather		Avg. Inc. (%)
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
Full model	**0.1378**	**0.2311**	**0.3054**	**0.3444**	0.4046	**0.2716**	**0.1726**	**0.2176**	**0.00**
w/o Global branch	0.1432	0.2372	0.3090	0.3457	0.4116	0.2774	0.1785	0.2202	**+2.07**
w/o Difference branch	0.1389	0.2317	0.3108	0.3468	0.4144	0.2732	0.1823	0.2248	**+1.93**
w/o Phase branch	0.1433	0.2377	0.3100	0.3468	**0.4016**	0.2742	0.1771	0.2196	**+1.60**
w/o Phase-indexed logits	0.1423	0.2354	0.3112	0.3481	0.4158	0.2815	0.1752	0.2194	**+2.11**
Plain softmax gate	0.1432	0.2368	0.3122	0.3492	0.4157	0.2819	0.1765	0.2205	**+2.52**

The full model achieves the best overall performance across these ablations, indicating that the three bases and the structured gate jointly contribute to the final accuracy. Removing any single branch generally degrades performance, which suggests that global trend–seasonal projection, difference-based local evolution, and phase-aligned recurrence capture different temporal patterns and cannot be fully substituted by the remaining branches.

The gate-related ablations show a clearer performance drop, as reflected by the Avg. Inc. column. Removing the phase-indexed logits increases the averaged metric by 2.11%, indicating that explicit future phase information is important for selecting among the branches. Replacing the Tri-Factorized Fusion Gate with a plain softmax gate leads to the largest increase of 2.52%, because the same branch mixture must be shared across all horizons and variables. These results confirm that the improvement of GatedLinear comes not only from using multiple linear forecasting bases, but also from routing them with horizon-, channel-, and phase-aware information.

4.5 Interpretability Analysis

The structured gate provides a direct diagnostic view of mechanism selection. Figure 4(a) shows that different datasets learn different branch-weight preferences over the forecast horizon, indicating that the model does not rely on a fixed mixture of forecasting mechanisms. Instead, it adjusts the relative importance of the global, difference, and phase-aligned branches according to dataset-specific temporal properties. A consistent pattern is that the difference branch tends to receive larger weights at shorter horizons and gradually decreases as the horizon extends, while the phase-aligned branch often becomes more prominent at longer horizons. This behavior is intuitive: difference-based incremental prediction is useful for capturing near-term local evolution, but its errors may accumulate over longer forecasts; in contrast, the phase-aligned branch can provide a more stable reference by reusing recurring periodic structure. The gate therefore dynamically shifts the mixture from short-term incremental evolution toward more reliable long-horizon periodic cues when appropriate.

Figure 4(b) further shows that even within the same dataset, different variables can have distinct branch preferences. In each branch-specific rectangular heatmap, each row corresponds to one variable channel and each column corresponds to a forecast horizon step. The heterogeneous row patterns suggest that GatedLinear learns variable-specific routing behavior, allowing each variable to adaptively select the branch mixture that matches its own temporal characteristics. Additional cross-dataset channel-wise preferences are shown in Appendix E.1.

To further quantify this alignment, Figure 5 aggregates all variable channels from the evaluated datasets and reports the Spearman correlations between channel-level temporal traits and learned branch preferences. The difference branch is closely associated with shift-related traits such as Mean Shift and Scale Shift, indicating that it is emphasized for variables that require incremental evolution. In contrast, the phase branch is positively correlated with Seasonality Strength and Phase Consistency, matching its intended role of exploiting stable recurring phases. These correlations show that the gate learns semantically meaningful branch preferences rather than arbitrary weights, further supporting the interpretability and effectiveness of the adaptive routing mechanism.

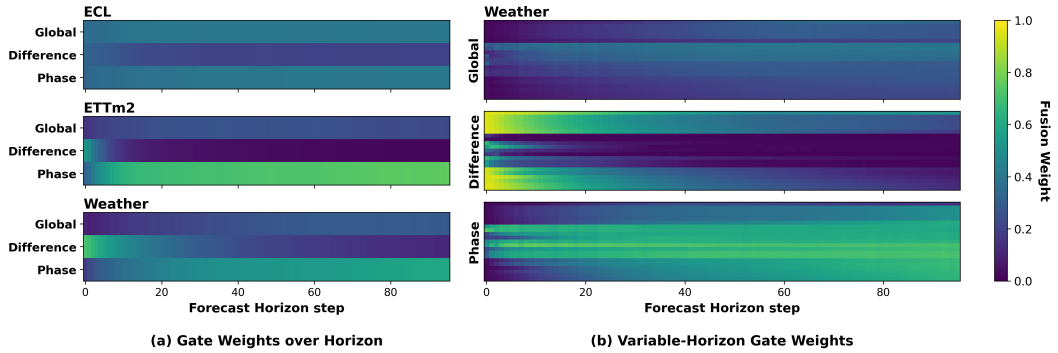


Figure 4: Learned fusion gate weights across horizons and variables. (a) Different datasets allocate branch weights over the forecast horizon. (b) Variable- and horizon-specific gate weights on the Weather dataset.

4.6 Efficiency Analysis

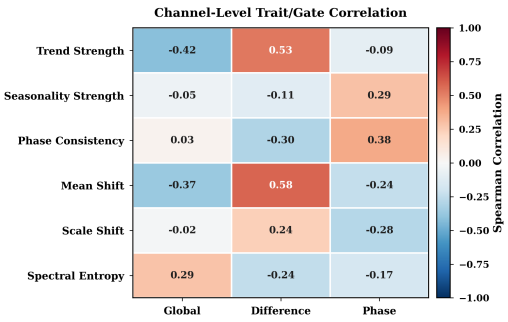


Figure 5: Spearman correlation heatmap between channel-level temporal traits and learned branch preferences. The computation of Spearman correlations is provided in Appendix E.

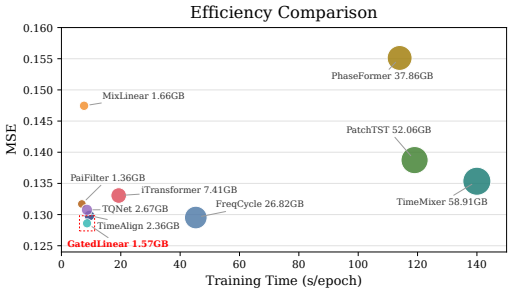


Figure 6: Efficiency comparison on the Electricity dataset using input length $L = 336$, prediction horizon $H = 96$, and batch size 128. Bubble size indicates peak GPU memory.

Figure 6 compares forecasting accuracy, training time, and peak GPU memory on Electricity. GatedLinear lies in the favorable lower-left region, combining low forecasting error with short training time and small memory consumption. These results demonstrate a favorable accuracy–cost trade-off among the evaluated baselines.

5 Conclusion

Time series forecasting requires different extrapolation mechanisms across datasets, variables, and future points, yet simply gating unrelated full-model experts can degrade performance. GatedLinear addresses this challenge by jointly designing complementary forecasting bases with comparable lightweight structure and a Tri-Factorized Fusion Gate that combines them at each forecast point using channel, horizon, and future-phase information. This pairing makes mechanism diversity useful without the performance degradation observed in a naive heterogeneous mixture, while retaining an efficient predictor. Across eight benchmarks, GatedLinear achieves the lowest average MSE and MAE among the evaluated methods with a favorable accuracy–cost trade-off. Expert-design comparisons, ablations, and gate analyses support the value of both the purpose-designed bases and fine-grained routing. Together, these results show that coordinating distinct predictive mechanisms and their selection is an effective way to handle temporal heterogeneity.

AI USE STATEMENT

We used generative AI tools to assist with implementing and revising code for the proposed method and experiments. We also used these tools to help draft and edit portions of the manuscript, improve clarity and language, and organize and summarize relevant literature. The authors reviewed and revised the AI-assisted material before including it in this work. We take full responsibility for the final code, references, analyses, claims, and manuscript content.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5 edition, 2015.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 6989–6997, 2023.

Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *ICLR 2024: The Twelfth International Conference on Learning Representations.*, 2024.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023.

Yifan Hu, Jie Yang, Tian Zhou, Peiyuan Liu, Yujin Tang, Rong Jin, and Liang Sun. Bridging past and future: Distribution-aware alignment for time series forecasting. *arXiv preprint arXiv:2509.14181*, 2025.

Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024.

Shengsheng Lin, Haojun Chen, Haijie Wu, Chunyun Qiu, and Weiwei Lin. Temporal query network for efficient multivariate time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025a.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *IEEE Internet of Things Journal*, 2025b.

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. *Advances in neural information processing systems*, 35:5816–5828, 2022.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering time series foundation models with sparse mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 38940–38962. PMLR, 2025. URL https://proceedings.mlr.press/v267/liu25an.html.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Aitian Ma, Dongsheng Luo, and Mo Sha. Mixlinear: Extreme low resource multivariate time series forecasting with 0.1 k parameters. *arXiv preprint arXiv:2410.02081*, 2024.

Ronghao Ni, Zinan Lin, Shuaiqi Wang, and Giulia Fanti. Mixture-of-Linear-Experts for long-term time series forecasting. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4672–4680. PMLR, 2024. URL https://proceedings.mlr.press/v238/ni24a.html.

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

Yiming Niu, Jinliang Deng, and Yongxin Tong. Phaseformer: From patches to phases for efficient and effective time series forecasting. *arXiv preprint arXiv:2510.04134*, 2025.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150*, 2024.

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.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/558d48c1f08675daa636e09bfe94a89e-Abstract-Conference.html.

Qitai Tan, Yiyun Chen, Mo Li, Ruiwen Gu, Yilin Su, and Xiao-Ping Zhang. Syntsbench: Rethinking temporal pattern learning in deep learning models for time series. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Hiro Y Toda and Peter CB Phillips. Vector autoregression and causality: a theoretical overview and simulation study. *Econometric reviews*, 13(2):259–285, 1994.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 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 *The Eleventh International Conference on Learning Representations*, 2023.

Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. Filternet: Harnessing frequency filters for time series forecasting. *Advances in Neural Information Processing Systems*, 37:55115–55140, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Boya Zhang, Shuaijie Yin, Huiwen Zhu, and Xing He. Freqcycle: A multi-scale time-frequency analysis method for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 28159–28167, 2026.

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*, pp. 27268–27286. PMLR, 2022.

A Additional Related Work

Efficient linear and MLP-based forecasting. Long-sequence forecasting has recently seen a clear shift from increasingly complex attention backbones toward Transformer-free architectures that rely on simple temporal mappings, fully connected blocks, and MLP-style mixing. N-BEATS (Oreshkin et al., 2019) introduced a deep fully connected architecture based on residual stacks and neural basis expansion, while N-HiTS (Challu et al., 2023) improved this basis-style view through hierarchical interpolation and multi-rate sampling for long-horizon forecasting. DLinear (Zeng et al., 2023) demonstrates that decomposing a series into trend and seasonal components followed by one-layer linear mappings can already form a highly competitive baseline. TSMixer (Chen et al., 2023) uses all-MLP time- and feature-mixing blocks to capture temporal and cross-variate information, TimeMixer (Wang et al., 2024) extends this efficiency-oriented view with decomposable multi-scale MLP mixing, and MixLinear (Ma et al., 2024) further targets low-resource multivariate forecasting through segment-based temporal modeling and adaptive low-rank spectral filtering. These methods show that lightweight predictors can challenge heavier Transformer-style models, but they usually instantiate one dominant forecasting mechanism for all variables and forecast positions. GatedLinear follows this efficient forecasting lineage, yet treats lightweight linear operations as complementary bases rather than as a single fixed backbone, allowing smooth projection, incremental evolution, and periodic recurrence to be selected adaptively.

Periodicity and phase-aware modeling. Another line of work explicitly exploits frequency structure, cyclic recurrence, or phase alignment in time series. FEDformer (Zhou et al., 2022) introduces frequency-enhanced decomposition inside a Transformer architecture, TimesNet (Wu et al., 2023) discovers dominant periods and converts one-dimensional sequences into two-dimensional period-aligned representations, and FilterNet (Yi et al., 2024) learns frequency-domain filters for shaping informative spectral components. CycleNet (Lin et al., 2024) models periodic patterns with learnable recurrent cycles and predicts residual components, while FreqCycle (Zhang et al., 2026) combines shared-cycle learning in the time domain with segmented frequency modeling for residual components. PhaseFormer (Niu et al., 2025) replaces patch tokens with phase-wise modeling and lightweight cross-phase routing for periodic forecasting. These approaches encode periodicity through spectral transforms, learned filters, learnable cycles, period-to-2D transformations, or phase-aware tokenization. In contrast, GatedLinear uses a direct time-domain recurrence basis that reuses historical values at matched future phases, and its phase-indexed gate logits decide when such exact phase reuse should dominate over the other bases.

Dataset heterogeneity and adaptive routing. Comprehensive benchmark studies suggest that forecasting data are highly heterogeneous across domains, variables, and temporal regimes. TFB (Qiu et al., 2024) broadens evaluation coverage across diverse real-world domains and characterizes datasets to support fairer comparisons among statistical, machine-learning, and deep-learning forecasters. SynTSBench (Tan et al., 2025) further constructs synthetic datasets with programmable temporal patterns and shows that existing deep forecasting models rarely perform uniformly well across all pattern types. These observations support the view that trend strength, periodicity, stationarity, and local drift vary substantially across datasets and even across variables within the same dataset. Adaptive routing is therefore a natural response: mixture-of-experts models (Jacobs et al., 1991; Shazeer et al., 2017) use gates to assign inputs to specialized predictors, and recent time-series architectures such as Pathformer (Chen et al., 2024) and PhaseFormer (Niu et al., 2025) introduce adaptive pathways or routers for scale and phase interactions. GatedLinear shares this motivation but routes among interpretable linear bases instead of deep experts, with routing logits factorized by channel, horizon, and future phase so that each forecast point can select the mechanism most compatible with its temporal traits.

B Limitations of Our Work

Channel-independent design. GatedLinear follows the common channel-independent forecasting setting: the temporal linear branches are shared across variables and each channel is mainly forecast from its own historical trajectory. Although the fusion gate learns channel-specific routing preferences, the current design does not explicitly model cross-variable dependencies. This choice keeps the model lightweight and interpretable, but it may be less effective when multivariate interactions are central to

the forecasting problem. A representative case is Traffic, which contains 862 sensor variables; on this dataset, methods that directly model variate-level relations, such as iTransformer and TQNet, obtain stronger results.

Dependence on meaningful periods and time marks. The phase-aligned recurrence basis and the phase-indexed gate assume that the data contain a usable base period and that future phase indices are meaningful. When future time marks are available, these indices can be obtained from the decoder-side timestamps; otherwise, the model falls back to horizon-based phase indices. This assumption may be weakened for weakly periodic series, drifting periods, holiday or event effects, abrupt disruptions, and irregularly sampled observations. In such cases, reusing historical values from the same phase may become less reliable.

Limited expressiveness of the linear basis pool. Our framework intentionally restricts its forecasting mechanisms to three simple linear bases: smooth trend–seasonal projection, difference-based local evolution, and phase-aligned recurrence with a linear residual. This restriction is important for the model's efficiency and interpretability, but it also bounds the expressive power of the predictor. Highly nonlinear dynamics, abrupt regime shifts, or trajectories driven by external events may not be fully captured by these three branches alone.

C Theoretical Motivation of the Difference-Based Basis

We use a simple random walk with drift to illustrate how predicting increments and reconstructing from the last observation affect forecasting. Let

$$x_t = x_{t-1} + c + \epsilon_t, \qquad \epsilon_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2), \tag{11}$$

where c is a constant drift. At forecast origin T, the future value is

$$x_{T+h} = x_T + hc + \sum_{i=1}^{h} \epsilon_{T+i}. \tag{12}$$

If the drift c is known, the conditional-mean predictor is $x_T + hc$, with mean squared error $h\sigma^2$. The difference branch predicts increments $\hat{d}_i$ from the observed history and reconstructs

$$\hat{x}^{(\text{diff})}_{T+h} = x_T + \sum_{i=1}^{h} \hat{d}_i. \tag{13}$$

Because future innovations are independent of the history, its conditional mean squared error is

$$\mathbb{E}\Big[(x_{T+h} - \hat{x}^{(\text{diff})}_{T+h})^2 \mid x_{\le T}, c\Big] = h\sigma^2 + \left(hc - \sum_{i=1}^{h} \hat{d}_i\right)^2. \tag{14}$$

Thus, if the increment layer outputs the true drift $\hat{d}_i = c$, the branch represents the conditional-mean predictor. For learned increments, the second term records the effect of estimation error.

The construction also provides additive shift equivariance. If every historical value is shifted by a constant δ, its first differences and predicted increments remain unchanged, while the reconstruction anchor becomes $x_T + \delta$. Hence, for any learned increment weights,

$$\hat{x}^{(\text{diff})}_{T+h}(x_{\le T} + \delta) = \hat{x}^{(\text{diff})}_{T+h}(x_{\le T}) + \delta. \tag{15}$$

An unnormalized direct linear mapping has the same property when its history weights sum to one; the difference construction builds it into the parameterization. Our implementation also uses RevIN, which handles additive shifts for the other branches. The analysis therefore motivates incremental reconstruction as a useful forecasting prior for drifting series, while the empirical results assess its contribution in the full model.

D Complexity Analysis

The four channel-shared affine mappings $\mathcal{L}_s, \mathcal{L}_t, \mathcal{L}_d, \mathcal{L}_p : \mathbb{R}^L \to \mathbb{R}^H$ each contain LH weights and H biases. They therefore contribute $4H(L+1)$ trainable parameters. The three gate terms in Eq. equation 9 contain $3C$, $3HC$, and $3PC$ parameters, respectively, for a total of $3C(1+H+P)$. Thus, the complete parameter count for these components is $4H(L+1) + 3C(1+H+P) = \mathcal{O}(LH + C(H+P))$. RevIN is used without affine parameters and does not change this count.

For a batch of B windows, each temporal affine projection costs $\mathcal{O}(BCLH)$ operations. Moving-average decomposition, differencing, and non-affine normalization each take $\mathcal{O}(BCL)$ operations with the fixed-size pooling implementation; constructing the phase template from K periods takes $\mathcal{O}(BCKP)$, where $KP \le L$. Gate lookup, softmax, and three-branch fusion take $\mathcal{O}(BCH)$. Hence, the temporal projections dominate the forward complexity at $\mathcal{O}(BCLH)$, while the gate adds only linear work in the number of forecast points. These asymptotic costs describe the model operations; measured training time and peak GPU memory are reported separately in Figure 6.

E Temporal Trait and Gate-Correlation Analysis

This section provides the definitions used to construct the temporal-trait visualization in Figure 1 and the channel-level gate-correlation heatmap in Figure 5. For a single variable channel, let the training sequence be $x_{1:T} = (x_1, \dots, x_T)$, where T is the training length. Let P denote the dataset-specific base period, e.g., $P = 24$ for hourly ETT data and $P = 168$ for weekly hourly datasets such as Electricity and Traffic. For multivariate datasets, all traits are computed independently for each channel.

Trend and seasonality strength. We first estimate a low-frequency trend component using a centered moving average with window size P,

$$\tau_t = \mathrm{MA}_P(x)_t. \tag{16}$$

The detrended sequence is $d_t = x_t - \tau_t$. We then estimate the seasonal component by averaging detrended values that share the same phase,

$$s_t = \frac{1}{|\mathcal{I}_{\phi(t)}|} \sum_{i \in \mathcal{I}_{\phi(t)}} d_i, \qquad \phi(t) = t \bmod P, \tag{17}$$

where $\mathcal{I}_\phi = \{i : i \bmod P = \phi\}$. The residual is $r_t = x_t - \tau_t - s_t$. Trend strength and seasonality strength are then computed as

$$\mathrm{TrendStrength} = 1 - \frac{\mathrm{Var}(r)}{\max(\mathrm{Var}(x-s), \epsilon)}, \qquad \mathrm{SeasonalityStrength} = 1 - \frac{\mathrm{Var}(r)}{\max(\mathrm{Var}(x-\tau), \epsilon)}, \tag{18}$$

and clipped to $[0, 1]$. Here ϵ is a small constant for numerical stability.

Phase consistency. Phase consistency measures whether values at the same phase of the base period are reproducible across cycles. Let $K = \lfloor T/P \rfloor$ be the number of complete cycles and fold the sequence into $X_{k,p}$ for $k = 1, \dots, K$ and $p = 0, \dots, P-1$. Denote

$$\bar{X}_p = \frac{1}{K} \sum_{k=1}^{K} X_{k,p}, \qquad \bar{X} = \frac{1}{KP} \sum_{k=1}^{K} \sum_{p=0}^{P-1} X_{k,p}. \tag{19}$$

The phase-consistency score is the fraction of total variance explained by phase-specific means:

$$\mathrm{PhaseConsistency} = \frac{K \sum_{p=0}^{P-1} (\bar{X}_p - \bar{X})^2}{\sum_{k=1}^{K} \sum_{p=0}^{P-1} (X_{k,p} - \bar{X})^2 + \epsilon}. \tag{20}$$

A larger value indicates more stable phase-aligned recurrence and therefore stronger support for the phase-aligned branch.

Mean and scale shift. Mean shift captures low-frequency level nonstationarity. For a rolling window of length P, define

$$\mu_t^{(P)} = \frac{1}{P} \sum_{i=t}^{t+P-1} x_i. \tag{21}$$

The mean-shift score is

$$\text{MeanShift} = \text{Std}\left(\{\mu_t^{(P)}\}_{t=1}^{T-P+1}\right). \tag{22}$$

Scale shift captures time-varying volatility or amplitude. With rolling standard deviation

$$\sigma_t^{(P)} = \sqrt{\frac{1}{P} \sum_{i=t}^{t+P-1} \left(x_i - \mu_t^{(P)}\right)^2}, \tag{23}$$

we compute

$$\text{ScaleShift} = \frac{\text{Std}\left(\{\sigma_t^{(P)}\}_{t=1}^{T-P+1}\right)}{\text{Mean}\left(\{\sigma_t^{(P)}\}_{t=1}^{T-P+1}\right) + \epsilon}. \tag{24}$$

Larger values indicate stronger level drift or scale nonstationarity.

Spectral entropy. Spectral entropy measures how concentrated the frequency-domain energy is. Let $S(f_j)$ be the power spectral density estimated by Welch's method and normalize it as

$$p_j = \frac{S(f_j)}{\sum_m S(f_m)}. \tag{25}$$

The normalized spectral entropy is

$$\text{SpectralEntropy} = -\frac{1}{\log_2 M} \sum_{j=1}^{M} p_j \log_2(p_j + \epsilon), \tag{26}$$

where M is the number of frequency bins. A larger value indicates more dispersed spectral energy and more complex frequency composition.

Robust normalization for visualization. For the radar and boxplot visualizations, each trait is robust-normalized across all selected channel samples. Given channel-level trait values $\{z_i\}_{i=1}^N$, let q_2 and q_{98} be the 2nd and 98th percentiles. The normalized score is

$$\tilde{z}_i = \text{clip}\left(\frac{z_i - q_2}{q_{98} - q_2 + \epsilon}, 0, 1\right). \tag{27}$$

Dataset-level radar values are obtained by averaging these robust-normalized channel-level scores.

Gate weight aggregation and correlation. The fusion gate produces branch weights $w_{h,c,b}$ for forecast horizon h, channel c, and branch $b \in \{\text{Global}, \text{Difference}, \text{Phase}\}$. For channel-level correlation analysis, we aggregate each channel's branch weight over the prediction horizon:

$$\bar{w}_{c,b} = \frac{1}{H} \sum_{h=1}^{H} w_{h,c,b}. \tag{28}$$

The heatmap in Figure 5 reports Spearman rank correlation between channel-level temporal traits and aggregated branch preferences over all selected channel samples. We use Spearman correlation because it captures monotonic relationships and is less sensitive to outliers or nonlinear scaling than Pearson correlation.

For completeness, Pearson correlation between trait values a_i and branch weights b_i is

$$\rho_{\text{Pearson}}(a, b) = \frac{\sum_{i=1}^N (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^N (a_i - \bar{a})^2} \sqrt{\sum_{i=1}^N (b_i - \bar{b})^2}}. \tag{29}$$

Spearman correlation first converts the two variables into ranks and then computes Pearson correlation on the ranked values:

$$\rho_{\text{Spearman}}(a, b) = \rho_{\text{Pearson}}\left(R(a), R(b)\right). \tag{30}$$

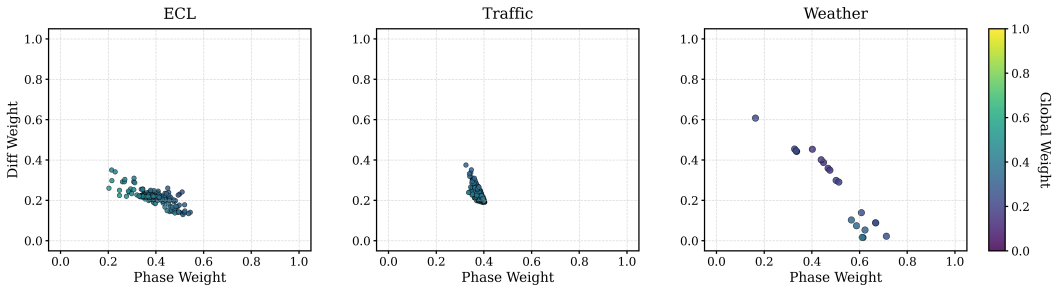


Figure 7: Channel-wise branch-weight preferences across different datasets. Each point denotes a variable channel.

E.1 Channel-wise Gate Preferences

Figure 7 visualizes channel-wise branch preferences for ECL, Traffic, and Weather. Each point denotes one variable, and its position reflects the relative reliance on the phase-aligned and difference-based branches. The Traffic variables form a compact cluster, while the Weather variables are more dispersed. This pattern is consistent with the channel-level trait distributions in Figure 1(b): Weather exhibits stronger variable-level heterogeneity, whereas Traffic shows less variation among variables. Together with Figure 4, this visualization provides an additional view of how learned gate preferences vary across datasets and channels.

F Experimental Details

F.1 Datasets

We provide additional details for the forecasting benchmarks used in our experiments.

- The ETT benchmark contains two sub-datasets: ETTh, collected at 1-hour intervals, and ETTm, collected at 15-minute intervals. Both were recorded from electricity transformers between July 2016 and July 2018 (Zhou et al., 2021).
- Electricity records the hourly electricity consumption of 321 clients from 2012 to 2014 (Wu et al., 2021).
- Traffic provides hourly measurements from 862 sensors on San Francisco Bay Area freeways from January 2015 to December 2016 (Wu et al., 2021).
- Weather includes 21 meteorological indicators recorded every 10 minutes during 2020 at the Weather Station of the Max Planck Biogeochemistry Institute (Wu et al., 2021).
- Exchange contains daily exchange rates for eight countries, including Australia, Britain, Canada, Switzerland, China, Japan, New Zealand, and Singapore, from 1990 to 2016 (Lai et al., 2018).

Unless otherwise stated, each dataset is split into train/validation/test subsets with a ratio of 7:1:2. The detailed dataset statistics are summarized in Table 4.

The base period P is set according to the dataset frequency: $P = 24$ for ETTh1/ETTh2, $P = 96$ for ETTm1/ETTm2, $P = 168$ for Electricity and Traffic, $P = 144$ for Weather, and $P = 24$ for Exchange.

F.2 Evaluation Protocol and Metrics

All real-dataset experiments use input length $L = 336$, prediction lengths $H \in \{48, 96, 192, 336\}$, and label length 0. We report mean squared error (MSE) and mean absolute error (MAE), following standard practice in time series forecasting. Lower values indicate better performance. GatedLinear results are averaged over three random seeds, and the validation set is used for early stopping and hyperparameter selection.

Table 4: Dataset statistics.

Dataset	Frequency	Duration	Features	Samples
ETTh	1 hour	2016–2018	7	17,420
ETTm	15 min	2016–2018	7	69,680
Electricity	1 hour	2012–2014	321	26,304
Traffic	1 hour	2015–2016	862	17,544
Weather	10 min	2020	21	52,696
Exchange	1 day	1990–2016	8	7,588

F.3 BASELINES

We summarize the representative forecasting baselines and their source repositories below.

DLinear. DLinear is a simple linear forecasting model that decomposes each input sequence into trend and seasonal components and applies one-layer linear mappings to both parts. Despite its simplicity, it provides a strong and efficient baseline for long-term time series forecasting. The official implementation is available at `https://github.com/cure-lab/LTSF-Linear`.

TimeMixer. TimeMixer is an MLP-based forecasting model that exploits decomposable multi-scale patterns. It mixes seasonal and trend information across multiple temporal resolutions and aggregates multi-scale predictors for future forecasting. The official implementation is available at `https://github.com/kwuking/TimeMixer`.

MixLinear. MixLinear is an ultra-lightweight multivariate forecasting model designed for resource-constrained settings. It combines segment-based temporal modeling with adaptive low-rank spectral filtering, reducing the parameter scale while preserving both temporal and frequency-domain information. The official implementation is available at `https://github.com/aitianma/MixLinear`.

PaiFilter. PaiFilter is a plain shaping filter introduced in FilterNet. It learns a universal frequency-domain filter to selectively shape time series signals and is recommended by FilterNet for datasets with a relatively small number of variables. The official implementation is available at `https://github.com/aikunyi/FilterNet`.

PatchTST. PatchTST adapts Transformers to long-term time series forecasting by segmenting each series into patch tokens and using channel-independent processing. This design enables the model to capture long-range temporal dependencies with improved efficiency. The official implementation is available at `https://github.com/yuqinie98/PatchTST`.

PhaseFormer. PhaseFormer replaces patch-based tokenization with phase-wise modeling for periodic time series. It uses compact phase embeddings and a lightweight cross-phase routing mechanism to capture periodic dynamics with very low parameter cost. The official implementation is available at `https://github.com/neumyor/PhaseFormer_TSL`.

CycleNet. CycleNet uses Residual Cycle Forecasting to model periodic patterns with learnable recurrent cycles. It removes the modeled cycle from the input, forecasts the residual component with a linear layer or shallow MLP, and adds the future cycle back to form the prediction. The official implementation is available at `https://github.com/ACAT-SCUT/CycleNet`.

TQNet. TQNet introduces temporal query vectors to model multivariate correlations efficiently. It uses periodically shifted learnable vectors as queries in an attention mechanism, combining global temporal priors with sample-specific key-value representations. The official implementation is available at `https://github.com/ACAT-SCUT/TQNet`.

TimeAlign. TimeAlign is a lightweight plug-and-play alignment framework for time series forecasting. It aligns past and future representations through a reconstruction branch and a distribution-aware alignment module, helping reduce the mismatch between historical inputs and future targets. The official implementation is available at `https://github.com/TROUBADOUR000/TimeAlign`.

iTransformer. iTransformer inverts the conventional Transformer design by treating variates as tokens and modeling multivariate correlations through attention. This architecture improves the effectiveness of Transformer modules on multivariate time series forecasting tasks. The official implementation is available at `https://github.com/thuml/iTransformer`.

F.4 Implementation Details

The code implementation uses four linear branches: seasonal, trend, difference, and phase residual. The seasonal and trend branches jointly form the decomposed direct basis. The difference branch predicts increments and reconstructs future levels with cumulative summation. The phase branch averages the most recent complete cycles into a phase template and adds a learned residual. Fusion logits are stored as a base table, a horizon table, and a phase table. When available, future time marks select phase logits; otherwise, horizon modulo period is used.

Each model is trained for at most 30 epochs using Adam with mean squared error loss, an initial learning rate of 0.005, batch size 128, and early stopping patience 20 based on validation loss. The moving-average kernel for trend–seasonal decomposition is set to 25. For the Tri-Factorized Fusion Gate, we use fusion temperature $\tau = 0.8$. RevIN is enabled with $\epsilon = 10^{-5}$ and without affine parameters.

The phase branch uses the most recent available complete cycles, with a target of 3 cycles for ETTh1 and 2 cycles for the other datasets, clipped by the number of complete cycles contained in the input window. Future time marks are used to select phase-indexed gate logits when the phase can be decoded; otherwise, the model falls back to horizon-modulo-period indexing.

G Additional Empirical Results

G.1 Detailed Forecasting Results

Table 5 reports the detailed MSE and MAE results for input length $L = 336$ and prediction lengths $H \in \{48, 96, 192, 336\}$. The horizon-wise breakdown further clarifies the averaged results in Table 1. GatedLinear is consistently strong on ETTm1, ETTm2, ETTh2, Exchange, and Weather, where it achieves the best or second-best results for most horizons. On ECL, CycleNet is close to GatedLinear, which has the lower four-horizon MSE and MAE. GatedLinear also shows particularly strong results on Weather as the prediction horizon becomes longer. On ETTh1 and Traffic, specialized baselines remain stronger for some horizons, indicating that a fixed lightweight channel-independent design is not uniformly optimal for every dataset. Nevertheless, GatedLinear obtains the best overall average performance, supporting the claim that adaptive mechanism selection provides a robust accuracy profile across heterogeneous benchmarks.

Table 5: Detailed forecasting results with input length $L = 336$. The best results are shown in **red bold**, and the second-best in blue underline. Lower MSE/MAE is better.

Dataset	H	GatedLinear		DLinear		TimeMixer		MixLinear		PaiFilter		PatchTST		PhaseFormer		CycleNet		TQNet		TimeAlign		iTransformer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	48	0.112	0.207	0.125	0.223	0.113	0.207	0.135	0.231	0.114	0.211	0.122	0.228	0.146	0.241	**0.108**	**0.204**	0.110	0.206	0.115	0.211	0.110	0.205
	96	**0.129**	**0.222**	0.141	0.237	0.135	0.227	0.147	0.242	0.132	0.227	0.139	0.242	0.155	0.248	0.131	0.224	0.131	0.226	0.130	0.224	0.133	0.227
	192	**0.146**	**0.239**	0.155	0.249	0.151	0.244	0.174	0.270	0.150	0.244	0.156	0.257	0.167	0.258	0.147	0.241	0.154	0.248	0.152	0.245	0.153	0.250
	336	**0.164**	**0.257**	0.172	0.265	0.172	0.262	0.178	0.271	0.167	0.261	0.173	0.274	0.183	0.273	0.166	0.260	0.171	0.265	0.169	0.263	0.174	0.267
	AVG	**0.138**	**0.231**	0.148	0.243	0.143	0.235	0.159	0.254	0.141	0.235	0.147	0.250	0.163	0.255	0.138	0.232	0.141	0.236	0.141	0.236	0.143	0.237
ETTh1	48	0.339	**0.370**	0.343	0.376	0.356	0.387	0.343	0.374	0.382	0.412	0.346	0.386	0.363	0.387	0.354	0.385	**0.337**	0.377	0.343	0.378	0.378	0.408
	96	0.370	0.391	0.376	0.397	0.395	0.411	**0.366**	**0.388**	0.434	0.437	0.379	0.405	0.383	0.395	0.379	0.401	0.387	0.407	0.384	0.405	0.438	0.438
	192	0.432	0.435	**0.405**	**0.412**	0.409	0.419	0.407	0.413	0.444	0.446	0.432	0.433	0.419	0.419	0.420	0.425	0.410	0.418	0.429	0.433	0.509	0.475
	336	0.473	0.461	0.429	0.428	0.438	0.442	**0.418**	**0.420**	0.471	0.464	0.445	0.443	0.435	0.428	0.438	0.436	0.432	0.433	0.450	0.446	0.509	0.486
	AVG	0.404	0.414	0.388	0.403	0.400	0.415	**0.383**	**0.399**	0.433	0.440	0.401	0.417	0.400	0.407	0.398	0.412	0.391	0.409	0.402	0.415	0.459	0.452
ETTh2	48	0.224	0.300	**0.217**	**0.298**	0.241	0.323	0.242	0.311	0.254	0.325	0.242	0.316	0.252	0.321	0.237	0.314	0.233	0.313	0.230	0.305	0.262	0.333
	96	0.283	0.343	**0.278**	**0.339**	0.311	0.366	0.293	0.344	0.324	0.372	0.310	0.364	0.301	0.355	0.299	0.354	0.296	0.352	0.293	0.349	0.369	0.393
	192	**0.333**	**0.379**	0.337	0.384	0.375	0.402	0.347	0.381	0.408	0.432	0.365	0.410	0.351	0.388	0.360	0.395	0.357	0.394	0.369	0.394	0.455	0.435
	336	**0.359**	0.403	0.370	0.411	0.387	0.424	0.374	**0.403**	0.402	0.438	0.398	0.428	0.398	0.423	0.383	0.415	0.375	0.413	0.388	0.418	0.442	0.448
	AVG	**0.300**	**0.356**	0.300	0.358	0.328	0.379	0.314	0.360	0.347	0.392	0.329	0.379	0.326	0.372	0.320	0.370	0.315	0.368	0.320	0.367	0.382	0.402
ETTm1	48	**0.251**	**0.309**	0.275	0.328	0.277	0.335	0.285	0.337	0.285	0.340	0.265	0.325	0.579	0.492	0.257	0.323	0.254	0.312	0.252	0.313	0.309	0.346
	96	**0.284**	**0.330**	0.302	0.344	0.306	0.354	0.324	0.365	0.310	0.355	0.304	0.350	0.499	0.463	0.292	0.347	0.293	0.344	0.291	0.340	0.340	0.376
	192	0.327	**0.359**	0.337	0.365	0.344	0.379	0.350	0.377	0.348	0.381	0.333	0.376	0.542	0.490	0.330	0.371	0.335	0.372	**0.324**	0.363	0.369	0.396
	336	0.360	**0.380**	0.374	0.386	0.372	0.397	0.391	0.404	0.377	0.399	0.367	0.396	0.541	0.492	0.366	0.393	0.371	0.391	**0.359**	0.385	0.410	0.423
	AVG	**0.305**	**0.344**	0.322	0.356	0.325	0.366	0.338	0.371	0.330	0.369	0.317	0.362	0.540	0.484	0.311	0.359	0.313	0.355	0.307	0.350	0.357	0.385
ETTm2	48	**0.124**	**0.221**	0.128	0.226	0.136	0.234	0.142	0.243	0.134	0.230	0.130	0.229	0.181	0.289	0.134	0.230	0.127	0.223	0.130	0.224	0.144	0.247
	96	**0.162**	**0.251**	0.164	0.252	0.179	0.261	0.178	0.268	0.180	0.266	0.178	0.267	0.209	0.299	0.174	0.261	0.166	0.254	0.168	0.253	0.212	0.300
	192	**0.221**	0.292	0.221	**0.291**	0.237	0.306	0.232	0.304	0.243	0.301	0.233	0.304	0.266	0.331	0.232	0.301	0.226	0.296	0.237	0.298	0.278	0.343
	336	0.276	0.329	**0.275**	**0.325**	0.296	0.351	0.286	0.338	0.281	0.330	0.292	0.343	0.308	0.361	0.291	0.339	0.285	0.332	0.285	0.333	0.326	0.373
	AVG	**0.196**	**0.273**	0.197	0.274	0.212	0.288	0.209	0.288	0.210	0.281	0.208	0.286	0.241	0.320	0.208	0.283	0.201	0.276	0.205	0.277	0.240	0.316
Exchange	48	0.048	0.151	**0.044**	**0.143**	0.047	0.153	0.058	0.173	0.052	0.159	0.047	0.154	0.058	0.172	0.049	0.156	0.053	0.161	0.046	0.150	0.053	0.166
	96	**0.084**	0.205	0.088	**0.205**	0.097	0.223	0.102	0.227	0.099	0.221	0.095	0.219	0.103	0.232	0.093	0.214	0.100	0.223	0.103	0.228	0.110	0.239
	192	0.200	0.328	0.192	0.307	0.197	0.317	0.187	0.311	0.238	0.346	0.188	0.315	0.214	0.335	0.192	0.314	0.196	0.317	**0.180**	**0.302**	0.236	0.354
	336	**0.313**	0.424	0.356	0.428	0.395	0.456	0.325	**0.415**	0.393	0.451	0.356	0.435	0.378	0.452	0.404	0.461	0.425	0.477	0.355	0.432	0.444	0.488
	AVG	**0.161**	0.277	0.170	**0.271**	0.184	0.287	0.168	0.281	0.196	0.294	0.172	0.281	0.188	0.298	0.184	0.286	0.193	0.294	0.171	0.278	0.211	0.312
Traffic	48	0.383	0.260	0.394	0.275	0.347	0.252	0.402	0.276	0.363	0.263	0.352	0.256	0.414	0.264	0.382	0.281	0.342	**0.237**	0.367	0.262	**0.329**	0.243
	96	0.397	0.266	0.411	0.280	0.371	0.258	0.416	0.279	0.388	0.276	0.376	0.268	0.425	0.266	0.405	0.286	0.364	**0.252**	0.376	0.261	**0.363**	0.260
	192	0.410	0.275	0.424	0.285	0.393	**0.264**	0.430	0.287	0.407	0.284	0.395	0.276	0.432	0.268	0.420	0.296	0.399	0.278	0.413	0.288	**0.385**	0.270
	336	0.428	0.285	0.436	0.291	0.422	0.293	0.443	0.295	0.421	0.291	0.409	0.282	0.442	**0.272**	0.435	0.296	0.420	0.297	0.428	0.295	**0.392**	0.274
	AVG	0.405	0.272	0.416	0.283	0.383	0.267	0.423	0.284	0.395	0.278	0.383	0.270	0.428	0.267	0.411	0.290	0.381	0.266	0.396	0.276	**0.367**	**0.262**
Weather	48	0.116	0.160	0.136	0.184	0.120	0.169	0.146	0.200	0.118	0.163	**0.114**	**0.155**	0.153	0.207	0.120	0.164	0.120	0.163	0.116	0.157	0.131	0.176
	96	**0.145**	**0.196**	0.174	0.224	0.155	0.209	0.183	0.236	0.151	0.204	0.150	0.198	0.184	0.244	0.148	0.199	0.156	0.209	0.149	0.199	0.167	0.221
	192	**0.188**	**0.237**	0.217	0.259	0.194	0.242	0.225	0.269	0.197	0.250	0.197	0.244	0.225	0.273	0.191	0.241	0.202	0.250	0.189	0.237	0.207	0.254
	336	**0.242**	**0.278**	0.264	0.293	0.242	0.279	0.273	0.304	0.249	0.288	0.254	0.287	0.272	0.307	0.243	0.282	0.249	0.284	0.242	0.279	0.257	0.291
	AVG	**0.173**	**0.218**	0.198	0.240	0.178	0.225	0.207	0.252	0.179	0.226	0.179	0.221	0.208	0.257	0.176	0.222	0.182	0.226	0.174	0.218	0.191	0.235

G.2 Comparison with Input-Conditioned Gates

We compare the proposed Tri-Factorized Gate with four variants that condition fusion weights on each input window. The three forecasting bases are retained while the gate is varied:

- **FeatureMLP** generates routing logits from the observed history using an MLP.
- **Tri-Factor + FeatureMLP** adds the MLP-generated logits to the structured logits of the proposed gate.
- **FeaturePatchAttn** uses patch-based attention over the observed history to generate routing logits.
- **Tri-Factor + FeaturePatchAttn** adds these attention-generated logits to the structured logits.

The two combined variants preserve the structured gate and add corrections based on the current window, directly testing whether input conditioning improves its learned preferences.

Table 6: MSE comparison of the proposed gate and input-conditioned alternatives on eight datasets. TF denotes the Tri-Factorized Gate, MLP denotes FeatureMLP, and Attn denotes FeaturePatchAttn. Average is the arithmetic mean across datasets. Lower is better; **red bold** marks the best result in each row.

Dataset	TF (Ours)	MLP	TF + MLP	Attn	TF + Attn
ECL	0.1286	0.1339	0.1300	0.1400	**0.1277**
ETTh1	**0.3704**	0.3917	0.3952	0.3734	0.3889
ETTh2	**0.2834**	0.3515	0.3450	0.2889	0.2947
ETTm1	0.2836	0.2992	0.3029	0.3013	**0.2830**
ETTm2	**0.1624**	0.1825	0.2093	0.1653	0.1740
Exchange	**0.0836**	0.1109	0.1233	0.0979	0.0973
Traffic	0.3967	**0.3826**	0.3862	0.4108	0.4023
Weather	**0.1448**	0.1554	0.1571	0.1525	0.1462
Average	**0.2317**	0.2510	0.2561	0.2413	0.2393

Table 6 shows that the proposed gate achieves the lowest average MSE, 0.2317, while the four input-conditioned alternatives obtain 0.2393–0.2561. Adding history-dependent corrections to the structured logits does not improve the average result in either combined variant. One possible explanation is that window-conditioned corrections respond to short-term fluctuations even when dataset-level branch preferences are relatively stable.

FeatureMLP improves on the proposed gate on Traffic, and Tri-Factor + FeaturePatchAttn achieves lower MSE on ECL and ETTm1. The proposed gate nevertheless has the lowest average error across the eight evaluated datasets.

G.3 Synthetic-Regime Comparison with MoE Forecasters

Inspired by the programmable temporal patterns in SynTSBench (Tan et al., 2025), we construct synthetic series with different forecasting characteristics and compare GatedLinear with representative mixture-of-experts forecasters. Table 7 presents four illustrative regimes selected after examining a twelve-regime sweep. The selected regimes cover stable phase-aligned repetition, nonstationary autoregressive dynamics, a smooth trend combined with periodicity, and a random walk. All methods are evaluated on the same 128 test origins with input length $L = 336$ and prediction length $H = 96$. The trained models use three-seed means, whereas the two foundation models are evaluated zero-shot with public weights.

GatedLinear has the lowest MSE and MAE in each of these four selected regimes. The results are consistent with the benefit of combining complementary linear forecasting bases across distinct temporal patterns. Because this is a selected subset, the table supports this narrower observation rather than a claim of dominance across all twelve regimes; comparisons with the zero-shot models also reflect a different evaluation protocol from the trained-from-scratch models.

Table 7: Selected synthetic-regime comparison ($L = 336$, $H = 96$; 128 shared test origins). Entries are normalized MSE and MAE; lower is better. Trained models report three-seed means, while Time-MoE and Moirai-MoE use public weights zero-shot. This illustrative subset was selected after examining twelve regimes.

Regime	Trained from scratch						Zero-shot public weights			
	GatedLinear		MoLE-DLinear		PathFormer		Time-MoE		Moirai-MoE	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Stable periodic pulse	**0.002183**	**0.03718**	0.002359	0.03869	0.003596	0.04594	0.02929	0.1396	0.1947	0.2877
ARIMA(1,1,1)	**0.2524**	**0.3433**	0.5715	0.4948	0.2736	0.3739	0.3607	0.4215	0.3412	0.3869
Trend + period	**0.0005654**	**0.0187**	0.0008295	0.0227	0.00058	0.01897	0.04254	0.1629	0.0535	0.1843
Random walk	**0.3225**	**0.3905**	0.4304	0.4382	0.3528	0.4151	0.6995	0.535	0.6285	0.4543

G.4 Forecasting Visualizations

Figure 8 provides qualitative examples for the same forecasting setting. The vertical dashed line separates the observed input window from the prediction region. Across strongly periodic examples such as ECL and Traffic, GatedLinear preserves the phase and amplitude of repeated peaks more closely than the shown baselines, which is consistent with the role of the phase-aligned branch. On ETTm2 and ETTh2, the predictions also follow the dominant future shape while avoiding large phase shifts. For more irregular or nonstationary series such as ETTh1, Exchange, and Weather, all methods have difficulty matching abrupt local changes exactly; GatedLinear tends to produce smoother and more conservative trajectories, tracking the main level and trend while not overreacting to short-lived fluctuations. These visualizations complement the quantitative results by showing that the proposed gate can select useful forecasting mechanisms under diverse temporal patterns, while the remaining errors often occur around sudden regime changes or high-frequency variations.

Figure 9 further shows forecasting results with the three branch contributions on ECL and Traffic. The visualizations make the fused forecast and the global, difference-based, and phase-aligned components explicit, illustrating how the final prediction is formed by combining complementary branches over the forecast horizon.

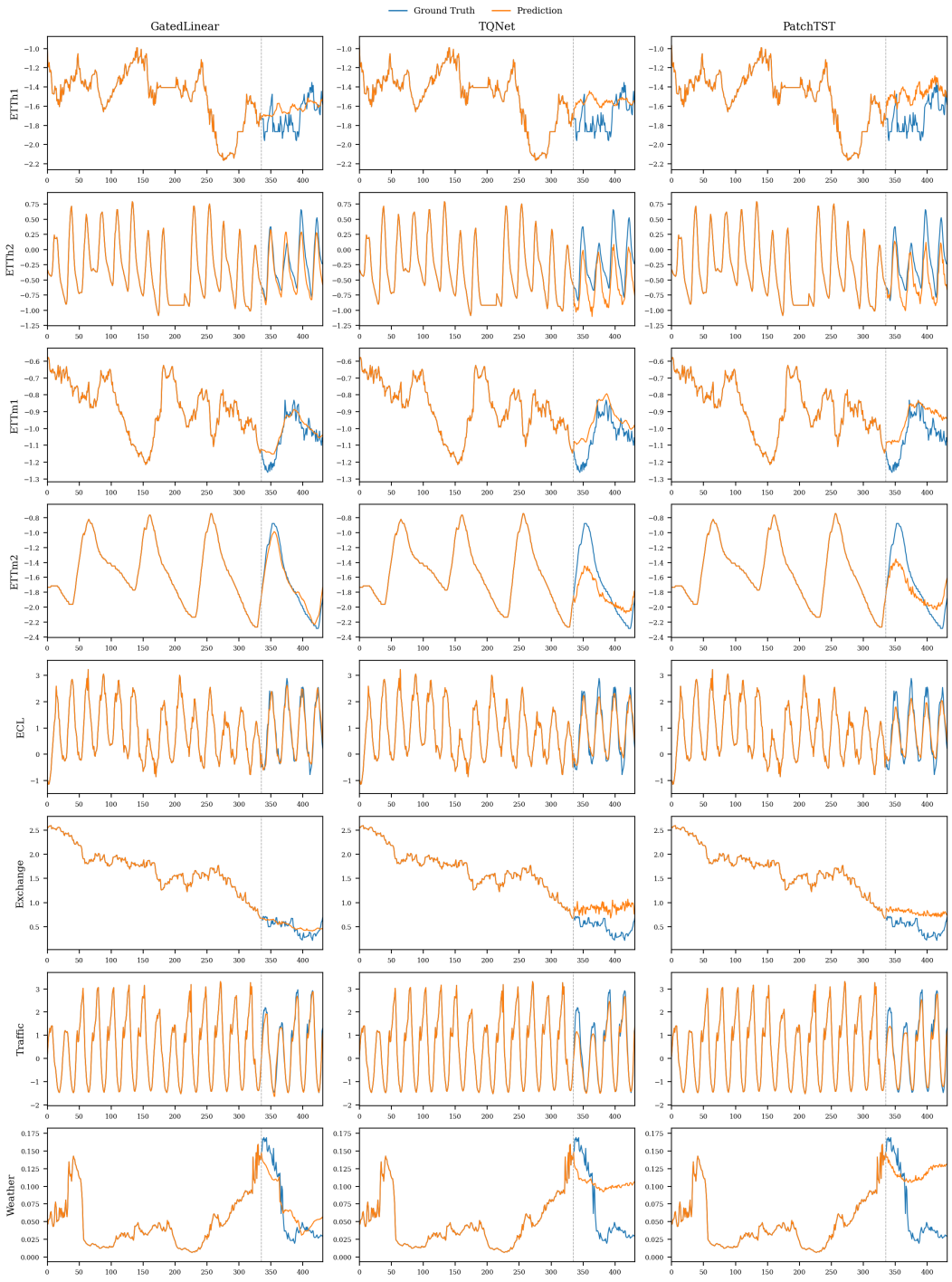


Figure 8: Forecasting visualization examples on eight benchmark datasets with input length $L = 336$ and prediction length $H = 96$. Each panel compares the predicted trajectory of our model with representative baselines against the ground truth, providing a qualitative view of how GatedLinear tracks diverse temporal patterns across datasets.

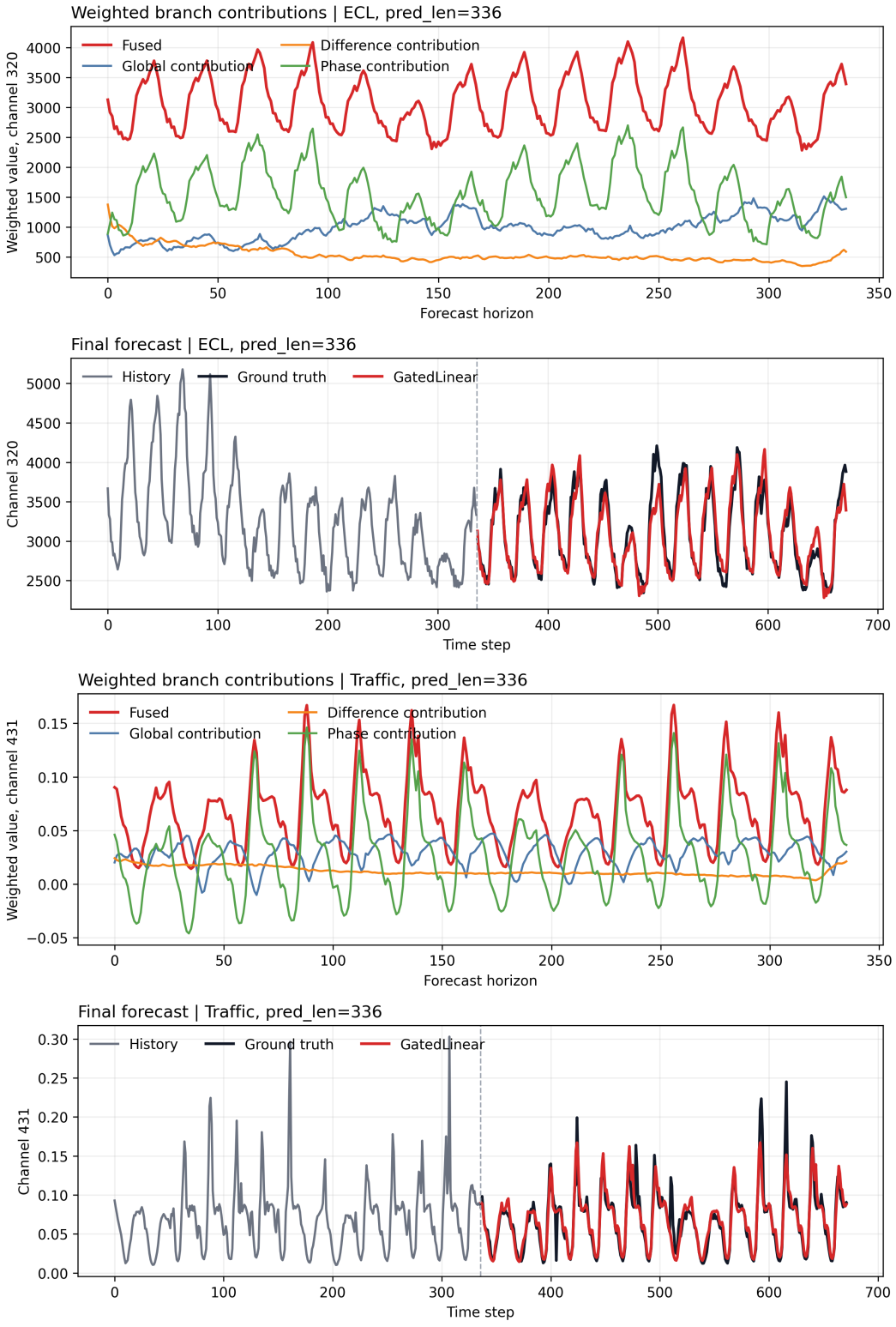


Figure 9: Forecasting visualizations with weighted tri-branch contributions on ECL and Traffic, using input length $L = 336$ and prediction length $H = 336$. For each dataset, the top panel shows the fused prediction together with the weighted contributions from the global, difference-based, and phase-aligned branches, and the bottom panel compares the final forecast with the ground truth.

G.5 Effect of Different Input Windows

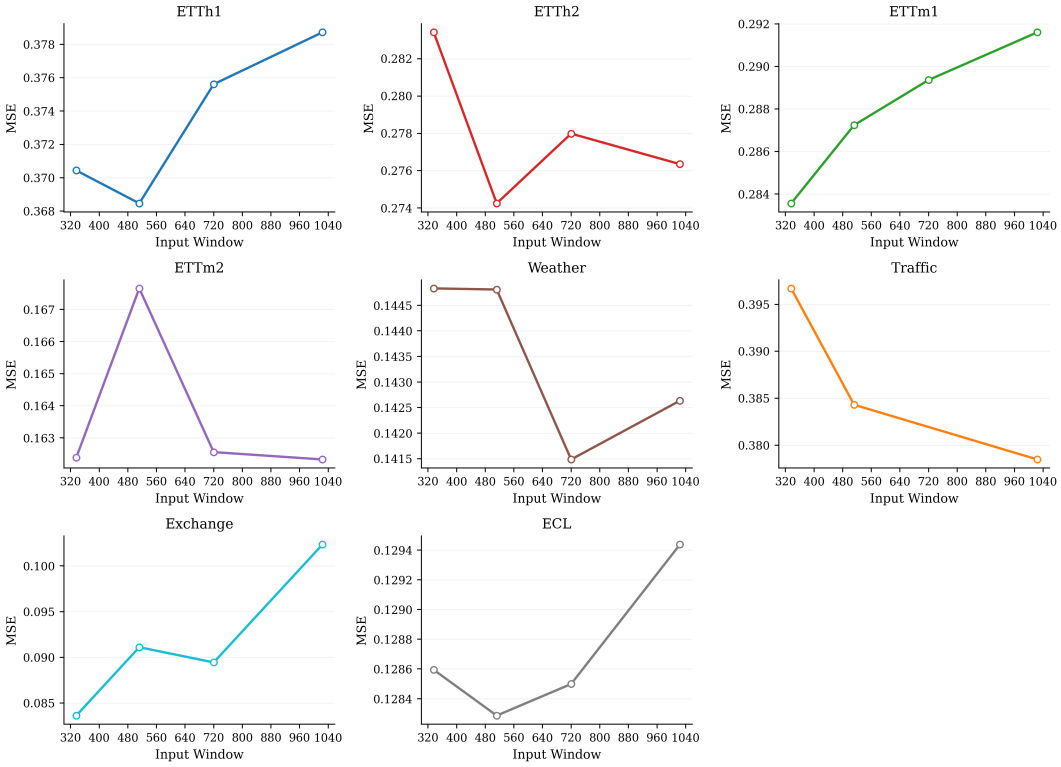


Figure 10: Performance comparison of GatedLinear under different input lengths across benchmark datasets, with prediction length fixed to $H = 96$. The results show the effect of using different historical context windows for a fixed forecasting horizon.

Figure 10 compares GatedLinear under different input-window lengths and examines whether longer input histories always improve forecasting accuracy. The effect is dataset-dependent. Traffic benefits from a longer window, and Weather also improves when more historical context is provided, which is plausible because both datasets contain stable daily or weekly patterns that can be reinforced by additional cycles. In contrast, ETTm1 and Exchange become worse when the window is excessively long, suggesting that older observations may introduce stale or nonstationary information. ETTh1, ETTh2, ETTm2, and ECL show their best or near-best results at moderate window lengths, rather than at the maximum window. Overall, the comparison indicates that the useful context size depends on the balance between informative periodic evidence and outdated historical information.

G.6 Parameter Sensitivity

Figure 11 analyzes two hyperparameters of the proposed gate and phase branch. For the fusion temperature τ, very sharp routing gives higher error, while moderate temperatures lead to lower MSE and MAE. The performance remains close across $\tau \in [0.5, 1.0]$, and the default value $\tau = 0.8$ lies in this stable low-error region, indicating that the model does not rely on a finely tuned temperature. For the number of phase cycles K, using a small number of recent cycles is beneficial, with the best region around $K = 3$ to $K = 5$ in this study. Increasing K to 10 degrades both metrics, likely because older cycles dilute the current phase template when the series undergoes level or amplitude changes. These results support the design choice of using compact phase evidence and a softly adaptive fusion gate.

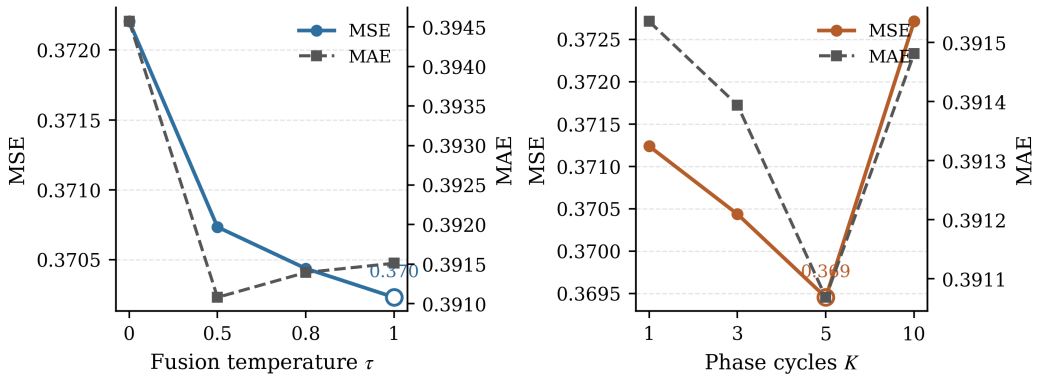


Figure 11: Parameter sensitivity study on the ETTh1 dataset. The left panel varies the fusion temperature τ in the fusion gate, and the right panel varies the number of phase cycles K. MSE and MAE are reported to show how these two hyperparameters affect forecasting performance.

G.7 Period Robustness of the Phase Branch

To assess when phase-aligned recurrence helps and how it behaves under a misspecified period, we vary the assumed period P on Exchange and ECL, which represent weakly and strongly periodic data, respectively. Table 8 reports the MSE for each setting, including a model without the phase branch. Both relative-change columns use the corresponding baseline MSE as the denominator; positive values indicate higher error. The default periods are $P = 24$ for Exchange and $P = 168$ for ECL.

Table 8: Sensitivity to the assumed period P on Exchange and ECL. Lower MSE is better; positive relative changes indicate worse performance than the indicated baseline.

Dataset	Period P	MSE	Δ vs. no phase	Δ vs. default
Exchange	No phase	0.084355	+0.00%	+0.86%
	1	0.084163	−0.23%	+0.63%
	7	0.084036	−0.38%	+0.47%
	12	0.083988	−0.43%	+0.42%
	24 (default)	0.083639	−0.85%	+0.00%
	48	0.083232	−1.33%	−0.49%
ECL	No phase	0.134469	+0.00%	+4.57%
	1	0.134706	+0.18%	+4.75%
	24	0.132646	−1.36%	+3.15%
	84	0.134705	+0.18%	+4.75%
	144	0.134364	−0.08%	+4.49%
	160	0.134462	−0.00%	+4.56%
	168 (default)	0.128594	−4.37%	+0.00%

The correctly specified weekly period on ECL reduces MSE by 4.37% relative to removing the phase branch. With other tested periods, this gain largely disappears: $P = 1$ and $P = 84$ are 0.18% worse than the no-phase setting, while $P = 160$ is nearly identical to it. Thus, a poor period choice can erase the phase branch's benefit on strongly periodic data, although the degradation relative to the no-phase model is small in this experiment. Exchange is less sensitive across the tested periods: every phase-enabled setting improves on the no-phase model by 0.23–1.33%, and $P = 48$ slightly outperforms the default $P = 24$. These results suggest that the phase branch is most useful when a stable recurrence can be aligned with an appropriate period; on weakly periodic data, its contribution is smaller and the tested period choices have limited impact.