BRIDGING HORIZON-SPECIFIC REPRESENTATIONS FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing time series forecasting methods typically train separate models for each forecasting horizon, as different horizons demand distinct representations. However, such a horizon-specific method limits its flexibility in real-world scenarios with diverse forecasting demands. In this paper, we provide a theoretical analysis, which reveals that optimal representation can vary with the horizon: short-term forecasting relies on local dynamic patterns, while long-term forecasting depends more on global trends. We thereby propose HorizonBridge, a novel framework that leverages a representation translator to bridge horizon-specific representations for cross-horizon forecasting. Specifically, HorizonBridge shares a base representation encoder across different horizons to preserve common structure. Then, we devise a horizon-conditioned representation translator to generate adaptive residual adjustments for the base representation, thereby adapting it to each target horizon. To further improve cross-horizon representation compatibility, we incorporate an adaptive patch embedding mechanism that partitions time series according to local information density and leverages temporal periodicity to construct cyclical embeddings. Extensive experiments show that HorizonBridge achieves competitive forecasting performance across multiple benchmark datasets, while offering improved efficiency and flexibility. The source code and models will be released.

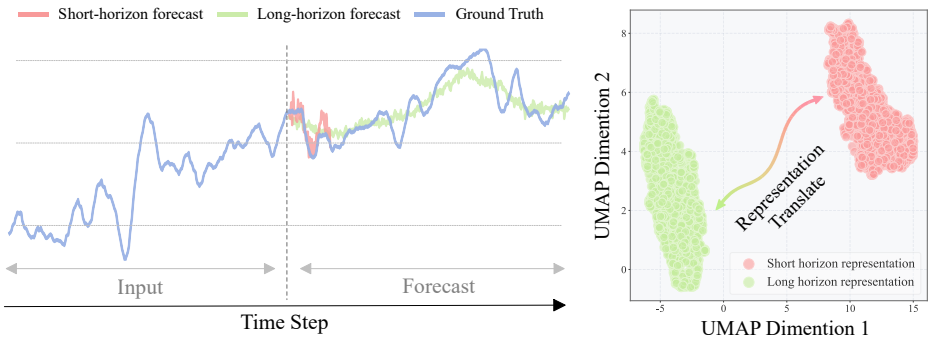
1 INTRODUCTION

Time series forecasting is widely used in a variety of real-world applications, including electricity dispatch (Zaballa et al., 2026), traffic coordination (Shao et al., 2026), and financial investment (Zhang et al., 2025a). In recent years, the rapid development of deep learning has significantly advanced time series forecasting. Many efforts have designed sophisticated architectures and modeling mechanisms, including RNN-based (Zhang et al., 2025b), LSTM-based (Kraus et al., 2025), GNN-based (Yin et al., 2024), and Transformer-based forecasting frameworks. These frameworks have continuously enhanced the representational capacity for historical series, leading to improved forecasting performance.

However, diverse forecasting requirements in real-world applications introduce new challenges. For example, in power load forecasting, models are expected to provide both short-term forecasts (*e.g.*, the next 1–24 hours) for real-time dispatch and medium- to long-term forecasts (*e.g.*, 7 days or longer) for planning and scheduling. Existing advanced time series forecasting models (Liu et al., 2024; Niu et al., 2026) still follow separate training and deployment for different forecasting horizons, *i.e.*, once trained for a specific horizon, such models often do not generalize well to another. We find that the fundamental difficulty of cross-horizon forecasting lies in the learned representations. Given that forecasting tasks at different horizons emphasize different aspects of temporal dynamics, the representations learned for one horizon may mismatch another. This makes cross-horizon forecasting inherently challenging without retraining the model.

Although pre-trained time series foundation models and some flexible frameworks have alleviated this issue to some extent, they still suffer from notable limitations. Autoregressive models (Liu et al., 2025; Das et al., 2024; Shi et al., 2025) can naturally support forecasting over varying horizons, yet they are prone to error accumulation in long-horizon prediction. Some studies adopt the masked

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.



(a) Short- vs. Long Horizon Forecast (b) Representation Space Visualization

Figure 1: Illustration of horizon-dependent forecasting patterns and representation discrepancy. (a) Short- and long-horizon forecasts exhibit different temporal patterns, where short horizons preserve local fluctuations while long horizons emphasize global trends. (b) UMAP visualization (McInnes et al., 2018) reveals distribution differences between representations of different horizons, motivating our horizon-conditioned representation translation.

modeling paradigm from language modeling (Devlin et al., 2019) to enable flexible forecasting through masked tokens (Woo et al., 2024; Zhang et al., 2024).

However, in long-term forecasting, introducing numerous mask tokens may not only reduce encoding efficiency, but also dilute effective context, thereby leading to degraded predictive performance. Another practical solution is to share a pre-trained time series encoder by freezing its backbone and training only a forecasting head for the target horizon (Goswami et al., 2024). Nevertheless, this strategy only adjusts how the learned representation is read out for forecasting, while the representation mismatch across horizons remains unresolved. Overall, these methods fail to simultaneously achieve strong cross-horizon adaptability, high forecasting accuracy, and good computational efficiency.

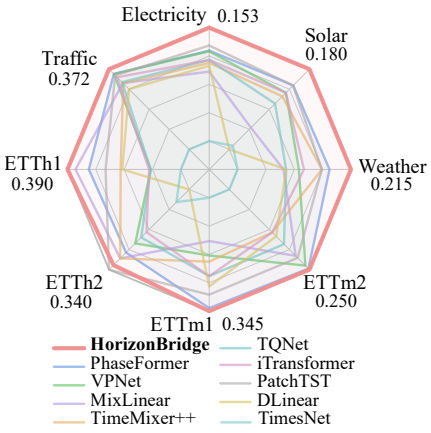


Figure 2: Performance comparison (average MSE) of HorizonBridge and state-of-the-art models on eight public datasets.

In this paper, we aim to optimize representations to realize cross-horizon forecasting. Our theoretical analysis reveals that the optimal representation can vary with the forecasting horizon, as temporal components at different time scales contribute unequally to the prediction target. As shown in Fig. 1(a), short-term forecasting tends to rely more on local dynamics, whereas long-term forecasting depends more on global trends. Nevertheless, these temporal patterns are not entirely isolated, as they share a common underlying dynamic structure. Therefore, directly sharing a fixed encoder across different horizons can lead to representation mismatch, while retraining independent models for each horizon is computationally expensive and lacks flexibility. Moreover, most existing Transformer-based forecasting models adopt a fixed patching strategy to tokenize time series, ignoring the non-uniform distribution of information density across temporal patches. This may further exacerbate the representation mismatch across different horizons. As a result, a question is raised: *without retraining the representation backbone, can we "translate" horizon-specific representations for cross-horizon forecasting by alleviating the representation mismatch?*

We answer this question with HorizonBridge, a novel framework for cross-horizon time series forecasting. Specifically, we first train a base representation encoder on a source horizon and then freeze it. Based on this frozen encoder, we introduce a horizon-conditioned representation translator. The translator takes the context representation and the target forecasting horizon as inputs, and learns dynamic transformations to generate residual adjustments. This design adapts the shared representation to other forecasting horizons, thereby eliminating the need to retrain the backbone network.

In addition, to further improve cross-horizon representation compatibility, we design an adaptive patching mechanism. It formulates patching as a Minimum Description Length (MDL)-based optimization problem (Grünwald, 2007), which can be solved offline via dynamic programming without additional training overhead. The resulting dynamic patches are then embedded cyclically by leveraging temporal periodicity to accommodate different patch lengths.

Our contributions are summarized as follows:

- We propose HorizonBridge, a novel framework of "base encoder + translator" that realizes time series forecasting across different horizons without retraining the representation backbone.
- We design a horizon-conditioned representation translator that generates adaptive residual adjustments for the shared base representation, bridging the gap between horizon-specific representations.
- We devise an adaptive patch embedding mechanism consisting of MDL-based patching and periodic embedding to further improve cross-horizon representation compatibility.
- Extensive experiments on multiple public benchmark datasets show that HorizonBridge achieves competitive forecasting performance, while exhibiting strong adaptability and flexibility across different horizon settings.

2 Related Work

2.1 Time Series Embedding

Time series embedding is a fundamental component of Transformer-based forecasting models. Early methods commonly adopt point-wise embedding, which maps the observation at each time step into a token (Wu et al., 2021; Zhou et al., 2021). However, this strategy not only increases computational costs in long-term forecasting, but also mixes heterogeneous variables within the same token, potentially causing semantic entanglement. PatchTST (Nie et al., 2023) introduces the patching strategy from Vision Transformers (Dosovitskiy et al., 2021) into time series forecasting, preserving local temporal semantics while reducing token numbers and attention costs. iTransformer (Liu et al., 2024) proposes inverted embedding, which represents the complete sequence of each variable as a token. PhaseFormer (Niu et al., 2026) replaces patch tokens with phase tokens to model cross-period dynamics more efficiently. In recent years, patch embedding has become a mainstream design in Transformer-based forecasting models. Despite its simplicity and effectiveness, conventional fixed-interval patching processes highly fluctuating and smooth regions with the same encoding granularity, making it difficult to adapt to non-uniform information density of real-world time series. To address this issue, TimeSqueeze (Ankireddy et al., 2025) encodes the sequence with a state space model and extracts boundary embeddings as patch representations, which may discard fine-grained variations within patches; TimeMosaic (Ding et al., 2026) determines the encoding granularity through a selection network, but it relies on predefined regions and still applies a uniform granularity within each region. In contrast, our periodic embedding method exploits periodic structures to uniformly encode patches of different lengths, thereby enabling more flexible modeling of heterogeneous temporal patterns and improving cross-horizon representation compatibility.

2.2 Conditional Representation Learning

Conditional representation learning aims to adjust intermediate representations or model parameters according to auxiliary conditions, such as text descriptions, class labels, or time steps. Common implementations include conditional feature modulation mechanisms, such as AdaIN (Huang & Belongie, 2017) and FiLM (Perez et al., 2018), as well as conditional parameter generation mechanisms, such as hypernetworks (Ha et al., 2017). These methods have been widely adopted in various domains. For example, in computer vision and generative modeling, diffusion models commonly condition denoising networks on the diffusion timestep (Ho et al., 2020). In natural language processing, prompt tuning methods introduce trainable task-specific prompts into frozen pre-trained language models, enabling task adaptation without updating the original model parameters (Liu et al., 2022). In time series forecasting, HN-MVTS (Savchenko & Kachan, 2026) conditions on variables to generate channel-specific prediction-head weights. APT (Li et al., 2026) conditions on

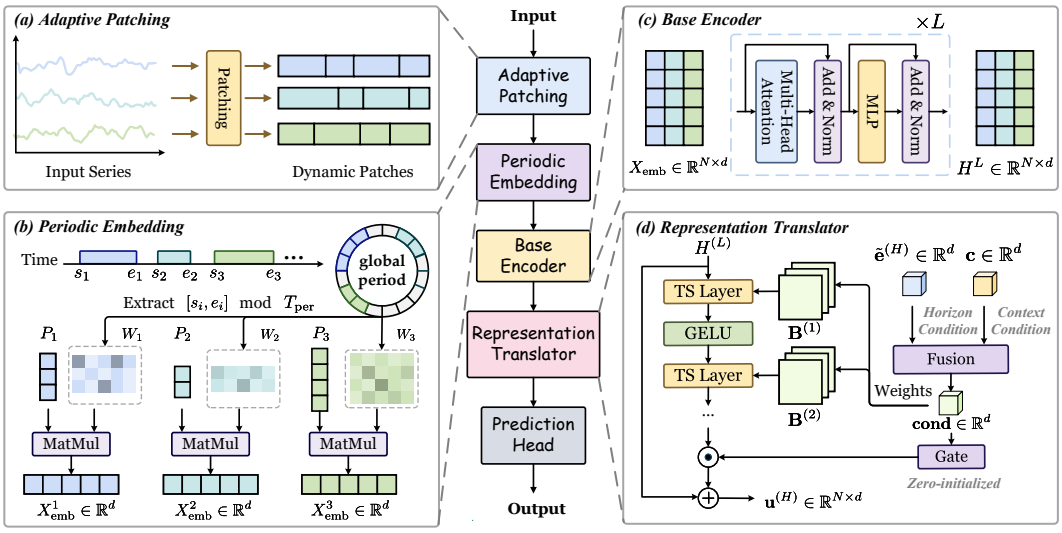


Figure 3: Overall architecture of HorizonBridge. (a) **Adaptive Patching** partitions the series into dynamic-length patches; (b) **Periodic Embedding** extracts phase-aligned components from the periodic matrix and embeds each patch as a token; (c) Base Encoder adopts a Transformer architecture to learn base representations; (d) **Representation Translator** generates dynamic residual adjustments, bridging the gap between horizon-specific representations.

timestamps to generate dynamic affine parameters for representation modulation, thereby improving robustness under distribution shift. However, the forecasting horizon is usually still treated as output configuration rather than as an explicit condition for representation. In this paper, we take the forecasting horizon and the context representation as conditions, and design a representation translator that transforms the base representation to adapt to the target horizon. As a result, our HorizonBridge achieves cross-horizon forecasting without retraining the backbone network.

3 Theoretical Motivation

To explain the representation mismatch across different forecasting horizons, we consider a linear dynamical system:

$$s_{t+1} = As_t + \epsilon_t, \qquad x_t = c^\top s_t + \eta_t, \tag{1}$$

where $s_t \in \mathbb{R}^M$ denotes the latent state, $A \in \mathbb{R}^{M\times M}$ is the system dynamics matrix, and $c \in \mathbb{R}^M$ is the observation vector. Given historical observations $\mathcal{X} = \{x_1, x_2, \ldots, x_T\}$, we denote the posterior mean of the latent state by $\mu_T = \mathbb{E}[s_T \mid \mathcal{X}]$. Under the MSE risk, the Bayes optimal H-step prediction can be written as

$$m_H(X) = \mathbb{E}[Y^{(H)} \mid \mathcal{X}] = G_H \mu_T, \tag{2}$$

where $G_H = \begin{bmatrix} c^\top A & c^\top A^2 & \cdots & c^\top A^H \end{bmatrix}^\top \in \mathbb{R}^{H\times M}$. Accordingly, the prediction target can be decomposed as $Y^{(H)} = G_H \mu_T + \xi^{(H)}$, where $\xi^{(H)}$ denotes the unpredictable residual and satisfies $\mathbb{E}[\xi^{(H)} \mid \mathcal{X}] = 0$. This form shows that the forecasting horizon H not only determines the output length, but also determines which aspects of the latent state contribute to future prediction.

Real-world time series are often driven by latent high-dimensional system dynamics, while the encoder of a deep model compresses historical information into a lower-dimensional latent representation. To describe this process, we regard the representation learned by the encoder as a rank-D linear compression of the posterior mean of the latent state: $z = R\mu_T \in \mathbb{R}^D$, where $R \in \mathbb{R}^{D\times M}$ with $\mathrm{rank}(R) = D$ and $D < M$. The model prediction can then be written as $\hat{Y}^{(H)} = WR\mu_T$, where the projection $W \in \mathbb{R}^{H\times D}$. To simplify the subsequent analysis, we consider minimizing the MSE loss:

$$\mathcal{L}_H(R, W) = \mathbb{E}\left[\left\| G_H \mu_T + \xi^{(H)} - WR\mu_T \right\|_2^2\right]. \tag{3}$$

For horizon H, let (R_H^*, W_H^*) denote an optimal solution to the forecasting objective over all rank-D representation matrices R. The optimized loss with respect to R is then given by:

$$\mathcal{L}_H^*(R) = \min_W \mathcal{L}_H(R, W). \tag{4}$$

We define the corresponding optimal representation subspace as:

$$S_H^* := \mathrm{Row}\left(R_H^* \Sigma^{1/2}\right), \quad \Sigma = \mathbb{E}\left[\mu_T \mu_T^\top\right] \succ 0. \tag{5}$$

This subspace is determined by:

$$K_H = \Sigma^{1/2} G_H^\top G_H \Sigma^{1/2}. \tag{6}$$

Specifically, S_H^* is spanned by the eigenvectors associated with the D largest eigenvalues of K_H, with the detailed proof shown in ***appendix D***. This leads to the following proposition.

Proposition [Representation Mismatch]. For two forecasting horizons $H_1 \neq H_2$, suppose that $S_{H_1}^* \neq S_{H_2}^*$, and K_{H_2} has a spectral gap, i.e., $\lambda_D^{(H_2)} > \lambda_{D+1}^{(H_2)}$. Then reusing a representation optimized for H_1 on horizon H_2 incurs excess risk:

$$\mathcal{L}_{H_2}^*(R_{H_1}^*) - \min_R \mathcal{L}_{H_2}^*(R) \geq \frac{1}{2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right) \left\|\Pi_{H_1}^* - \Pi_{H_2}^*\right\|_F^2 \tag{7}$$

where Π_H^* denotes the orthogonal projection onto S_H^*.

This proposition shows that cross-horizon forecasting involves a representation-level mismatch, rather than merely a change in the output dimension. When the optimal subspaces for two horizons differ, a representation optimized for one horizon cannot fully capture the predictive directions required by the other, leading to unavoidable excess risk. Directly reusing a fixed representation across horizons is therefore generally suboptimal, motivating us to design the representation translator for more flexible and efficient cross-horizon forecasting.

4 METHODOLOGY

Problem Formulation. For simplicity, channel dimension is omitted in the following formulation, since our core designs are channel-independent. Given a historical time series $X = [x_1, \ldots, x_T] \in \mathbb{R}^T$, the goal is to predict the future sequence $Y^{(H)} = [x_{T+1}, \ldots, x_{T+H}] \in \mathbb{R}^H$, where H denotes the target forecasting horizon. Unlike conventional methods that train separate models for fixed forecasting horizons, we aim to use a unified model to handle different target horizons. Specifically, a shared encoder $f_\theta(\cdot)$ first learns a base representation from the input sequence, which is then adapted to the target horizon by a representation translator:

$$\mathbf{u}^{(H)} = \mathrm{Translator}\left(f_\theta(X), H\right). \tag{8}$$

The resulting horizon-adapted representation $\mathbf{u}^{(H)}$ is then mapped to the corresponding forecast $\hat{Y}^{(H)} \in \mathbb{R}^H$.

4.1 ADAPTIVE PATCH EMBEDDING

MDL-based Patching. Existing patch-based time series forecasting models typically adopt fixed-length patches. While simple and effective, this imposes a uniform temporal granularity across the series, which is unsuitable for the non-uniform local structures often present in real-world data. To address this, we formulate patching as a Minimum Description Length (MDL) (Grünwald, 2007) optimization problem and evaluate each candidate patch by its description length under a local Gaussian model. By minimizing the total description length (TDL), the model avoids grouping heterogeneous intervals into the same patch, making each patch easier to represent as a token in the subsequent embedding.

Formally, given a time series X of length T, we partition it into K consecutive patches $\mathcal{P} = \{P_i\}_{i=1}^K$. The TDL of a patch P_k is approximated as:

$$\mathrm{TDL}(P_k) = \frac{1}{2}|P_k| \log\left(2\pi e \cdot \max(\hat{\sigma}_{P_k}^2, \epsilon)\right) + \log|P_k|, \tag{9}$$

where $\hat{\sigma}^2_{P_k}$ denotes the empirical variance within patch P_k, and ϵ is a small variance floor used to avoid degenerate description lengths for zero-variance patches. The optimal partition is then obtained by minimizing the sum of TDL of all patches:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} \sum_{k=1}^{K} \mathrm{TDL}(P_k), \quad \text{s.t. } |P_k| \geq L_{\min} \tag{10}$$

The optimization is solved using dynamic programming, with details provided in ***Appendix B***.

Periodic Embedding. Since MDL-based patching produces patches of dynamic lengths, a conventional linear projection with a fixed input dimension cannot be directly applied. We therefore propose a phase-aligned periodic embedding, which assigns embedding weights according to the phase of each time point within a global period rather than its local position within the patch. Specifically, each time index t is mapped to its phase by $\phi(t) = t \bmod T_{\mathrm{per}}$, where T_{per} denotes a predefined period, such as 24 for hourly data. The embedding of patch P_k is computed as

$$X^k_{\mathrm{emb}} = \frac{1}{\sqrt{|P_k|}} \sum_{t=s_k}^{e_k} W_{\mathrm{per}}[:, \phi(t)] x_t, \tag{11}$$

where $[s_k, e_k]$ denotes the interval of P_k, and $W_{\mathrm{per}} \in \mathbb{R}^{d \times T_{\mathrm{per}}}$ is a learnable embedding matrix. The normalization factor $1/\sqrt{|P_k|}$ alleviates scale variations caused by different patch lengths. By sharing embedding weights at the same phase positions, this design maps variable-length patches into fixed-dimensional tokens while preserving consistent periodic semantics.

4.2 Horizon-conditioned Representation Translator

Following the standard paradigm of patch-based Transformer forecasters (Nie et al., 2023; Zhang & Yan, 2023), we employ L encoder-only Transformer layers as the base encoder to process the patch tokens:

$$\mathbf{H}^l = \mathrm{EncoderLayer}^l(\mathbf{H}^{l-1}), \quad l = 1, \ldots, L, \tag{12}$$

where $\mathbf{H}^l \in \mathbb{R}^{N \times d}$ denotes the representation produced by the l-th encoder layer. The final encoder output $\mathbf{H}^L$ captures shared temporal dependencies from the historical sequence.

As discussed earlier, different forecasting horizons generally require different representation directions. We therefore introduce a horizon-conditioned representation translator to adapt $\mathbf{H}^L$ to the target horizon. Since the required adjustment may also vary with the current input sequence, the translator is conditioned on both the target horizon and the input context, as further discussed in ***Appendix D***. Specifically, we first construct a condition vector $\mathbf{cond} \in \mathbb{R}^d$ by fusing the horizon embedding with the context pooled from $\mathbf{H}^L$:

$$\tilde{\mathbf{e}}_H = \mathrm{Linear}(\mathbf{e}_H), \quad \mathbf{c} = \mathrm{Linear}\left(\mathrm{Pool}(\mathbf{H}^L)\right), \tag{13}$$

$$\mathbf{cond} = \mathrm{Linear}\left(\mathrm{GELU}\left(\tilde{\mathbf{e}}_H \oplus \mathbf{c}\right)\right), \tag{14}$$

where $\mathbf{e}_H \in \mathbb{R}^d$ denotes the horizon embedding, $\oplus$ denotes the concatenation operation. Since the representation adjustments required by different horizons share common structures, directly generating a full transformation matrix for each condition would introduce unnecessary overhead. Inspired by hypernetworks (Ha et al., 2017), we model each dynamic transformation as a mixture of K learnable basis matrices. For the r-th translator layer, the mixing logits are generated by:

$$\boldsymbol{\gamma}^{(r)} = \mathrm{Linear}^{(r)}_{\gamma}\left(\mathbf{cond}\right), \tag{15}$$

and the corresponding dynamic weight is given by

$$W^{(r)}_{\mathrm{ts}} = \frac{1}{\sum_{k=1}^{K} e^{\gamma^{(r)}_k}} \sum_{k=1}^{K} e^{\gamma^{(r)}_k} \mathbf{B}^{(r)}_k. \tag{16}$$

Here, $\{\mathbf{B}^{(r)}_k\}_{k=1}^{K}$ denotes the learnable basis matrices of the r-th translator layer. The resulting dynamic weights W_{ts} parameterize the translator network $\mathrm{TSNet}(\cdot)$ to produce a residual adjustment for $\mathbf{H}^L$:

$$\mathbf{u}_H = \mathbf{H}^L + \phi_{\mathrm{gate}}(\mathbf{cond}) \odot \mathrm{TSNet}(\mathbf{H}^L), \tag{17}$$

where $\phi_{\mathrm{gate}}(\cdot)$ is a zero-initialized gate projection, which makes the translator approximate an identity mapping at the start of training, thereby stabilizing optimization and avoiding premature disruption of the base representation.

Shared Prefix Forecasting Head. After representation translation, $\mathbf{u}^{(H)}$ has already incorporated horizon-specific structural information. Therefore, instead of assigning independent prediction heads to different horizons, we adopt a shared prefix forecasting head to support cross-horizon forecasting flexibly:

$$\hat{Y}^{(H)} = \mathrm{PrefixProj}_H\left(\mathrm{Flatten}_{[N,d]}(\mathbf{u}^{(H)})\right). \tag{18}$$

During training, HorizonBridge samples multiple target horizons for the same input window and jointly optimizes the shared head with supervision from different horizons. This encourages a horizon-agnostic readout structure, which prevents the prediction head from overfitting to horizon-specific discrepancies.

5 Experiments

5.1 Experimental Setup

Table 1: Comparison of average multi-horizon forecasting performance with unified look-back window $T = 720$. The best and second-best results are highlighted in **red** and underlined. Full results are reported in ***Appendix F***.

Models	**HorizonBridge (Ours)**		PhaseFormer (2026)		VPNet (2026)		MixLinear (2026)		TimeMixer++ (2025)		TQNet (2025)		iTransformer (2024)		PatchTST (2023)		DLinear (2023)		TimesNet (2023)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.345**	0.376	0.346	**0.374**	0.372	0.383	0.379	0.400	0.369	0.378	0.362	0.388	0.362	0.391	0.353	0.382	0.357	0.379	0.400	0.406
ETTm2	**0.250**	**0.305**	0.251	0.314	0.252	0.306	0.257	0.319	0.269	0.320	0.263	0.323	0.269	0.329	0.256	0.317	0.267	0.331	0.291	0.333
ETTh1	**0.390**	0.417	0.403	**0.415**	0.440	0.442	0.395	0.424	0.422	0.432	0.440	0.452	0.439	0.448	0.413	0.434	0.423	0.437	0.458	0.450
ETTh2	0.340	**0.379**	0.347	0.389	0.359	0.391	0.339	0.383	0.339	0.380	0.367	0.406	0.374	0.406	**0.324**	0.381	0.431	0.447	0.414	0.427
Weather	**0.215**	**0.246**	0.223	0.261	0.235	0.259	0.241	0.281	0.226	0.262	0.240	0.279	0.233	0.271	0.226	0.264	0.240	0.300	0.259	0.287
Electricity	**0.153**	**0.246**	0.161	0.250	0.161	0.255	0.168	0.261	0.165	0.253	0.164	0.261	0.164	0.261	0.159	0.253	0.166	0.264	0.192	0.295
Traffic	**0.372**	**0.245**	0.387	0.250	0.387	0.265	0.415	0.282	0.416	0.264	0.412	0.289	0.397	0.282	0.391	0.264	0.434	0.295	0.620	0.336
Solar	**0.180**	0.214	0.194	0.240	0.200	**0.213**	0.229	0.263	0.203	0.258	0.209	0.264	0.200	0.260	0.194	0.245	0.247	0.309	0.244	0.334

Datasets. We evaluate HorizonBridge on widely used long-term forecasting benchmarks (Zhou et al., 2021; Wu et al., 2021; Liu et al., 2024), including ETT, Weather, Electricity, Traffic, and Solar. Their diverse temporal dynamics and periodic patterns enable a comprehensive evaluation across different forecasting horizons.

Baselines. We compare HorizonBridge with recent state-of-the-art forecasting models including: (1) Transformer-based models: PhaseFormer (Niu et al., 2026), TimeMixer++ (Wang et al., 2025), TQNet (Lin et al., 2025), iTransformer (Liu et al., 2024), and PatchTST (Nie et al., 2023); (2) Linear models: MixLinear (Ma et al., 2026) and DLinear (Zeng et al., 2023); (3) CNN-based models: VPNet (Liu et al., 2026) and TimesNet (Wu et al., 2023).

Implementation Details. All experiments are conducted on two NVIDIA RTX PRO 6000 96GB GPUs using the PyTorch framework (Paszke et al., 2019) and the Adam optimizer (Kingma & Ba, 2015). Following prior works (Niu et al., 2026), we set the look-back window to 720 and evaluate cross-horizon forecasting with target horizons $H \in \{96, 192, 336, 720\}$. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are adopted as evaluation metrics. For all baselines, each forecasting horizon is trained independently. HorizonBridge adopts a two-stage training strategy. In the first stage, the embedding and base encoder are trained with $H = 96$. These components are then frozen, and the representation translator and shared prediction head are jointly optimized for other horizons. The detailed training procedure and implementation settings are provided in ***Appendix C***.

5.2 Main Results

Table 1 reports the average cross-horizon forecasting results of HorizonBridge against representative baselines. HorizonBridge delivers consistently competitive performance across diverse datasets,

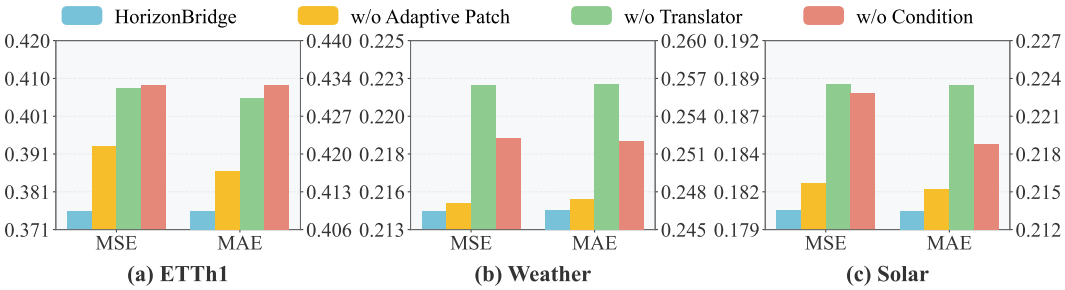


Figure 4: Framework components ablation of HorizonBridge on ETTh1, Weather, and Solar datasets. The left and right y-axes represent the MSE and MAE values, respectively.

Table 2: Ablation study on multi-horizon forecasting strategies, where the reported results are averaged over four forecasting horizons $H \in \{96, 192, 336, 720\}$.

Dataset	HorizonBridge		Prefix		AR	
Metric	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.390**	**0.417**	0.413	0.437	0.429	0.433
ETTm1	**0.345**	0.376	0.352	0.382	0.352	**0.373**
Solar	**0.180**	**0.214**	0.183	0.222	0.195	0.219
Weather	**0.215**	**0.246**	0.221	0.256	0.256	0.271

demonstrating robust cross-horizon forecasting capability. Notably, HorizonBridge reuses the base encoder trained on the $H = 96$ task for all target horizons, while other models are trained independently for each horizon. HorizonBridge achieves strong forecasting performance and outperforms the recent advanced forecaster VPNet by reducing MSE and MAE by 6.27% and 3.23% on average, respectively. These results demonstrate that horizon-conditioned representation translation can effectively mitigate representation mismatch across horizons and enable flexible adaptation to diverse forecasting requirements.

5.3 Ablation Studies

Framework Components. We conduct framework component ablations to examine the contribution of each design in HorizonBridge with three variants. **w/o Translator:** removes the horizon-conditioned translator and performs cross-horizon forecasting by training only linear heads; **w/o Adaptive Patch:** replaces adaptive patch embedding with a conventional fixed-length patch embedding; **w/o Condition:** removes the horizon and context conditioning signals, so that the translator acts as a standard MLP. As shown in Fig. 4, removing the representation translator causes a substantial performance degradation across all datasets. This indicates that directly reusing the shared base representations is insufficient for different forecasting horizons, highlighting the importance of representation translation. The w/o condition variant is slightly less degraded, as the remaining MLP provides limited transformation, but it lacks horizon-aware adaptation. Replacing adaptive patch embedding also degrades performance, suggesting that fixed patches cannot fully capture the non-uniform information density in time series.

Multi-horizon Training. We compare HorizonBridge with two multi-horizon forecasting strategies. **AR:** autoregressive forecasting following the next-patch-token prediction paradigm (Liu et al., 2025); **Prefix:** training on the longest horizon and obtaining shorter predictions by extracting the corresponding prefix. As shown in Table 2, HorizonBridge achieves the best average performance across different forecasting horizons. This improvement demonstrates the advantage of explicitly modeling horizon-dependent representation adaptation. Detailed analysis across horizons and full results are provided in ***Appendix E***.

Period Length Sensitivity. We further study the sensitivity of the period length T_{per} in the periodic embedding. As shown in Fig. 5, HorizonBridge achieves better performance when the selected

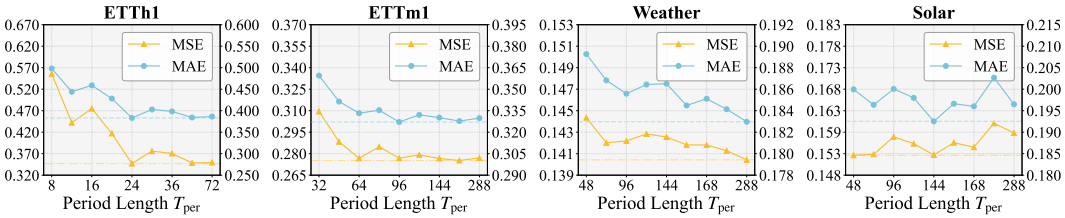


Figure 5: Sensitivity analysis of the period size T_{per} on ETTh1, ETTm1, Weather and Solar datasets. The left and right y-axes represent the MSE and MAE values, respectively.

Table 3: Plug-and-play analysis of the representation translator with PatchTST, Crossformer, and TimeXer for cross-horizon forecasting. Params and Time denote the total trainable parameters and accumulated per-iteration training time required to support all forecasting horizons, respectively. Full results are reported in ***Appendix F***.

Models		PatchTST				Crossformer				TimeXer			
Metric		MSE	MAE	Params	Time	MSE	MAE	Params	Time	MSE	MAE	Params	Time
Solar	Original	0.193	0.256	17.1M	976.7ms	0.210	0.259	16.2M	529.6ms	0.187	0.263	19.9M	262.4ms
	+Translator	0.196	0.261	10.0M	374.5ms	0.191	0.249	4.5M	112.1ms	0.188	0.260	5.6M	135.9ms
	Δ *IMP*	−1.52%	−1.72%	**41.7%**	**61.7%**	**8.90%**	**3.92%**	**72.2%**	**78.8%**	−0.54%	**1.10%**	**71.7%**	**48.2%**
Weather	Original	0.221	0.263	17.1M	200.8ms	0.239	0.302	13.6M	223.4ms	0.233	0.273	14.1M	105.5ms
	+Translator	0.219	0.260	10.1M	81.6ms	0.216	0.279	4.1M	60.2ms	0.227	0.268	5.7M	43.1ms
	Δ *IMP*	**1.18%**	**0.97%**	**41.1%**	**59.4%**	**9.38%**	**7.76%**	**69.9%**	**73.1%**	**2.30%**	**1.82%**	**59.7%**	**59.2%**
Traffic	Original	0.415	0.292	17.1M	828.2ms	0.564	0.311	13.5M	412.1ms	0.400	0.279	61.2M	960.0ms
	+Translator	0.401	0.278	10.1M	284.6ms	0.550	0.310	3.1M	99.0ms	0.397	0.276	19.0M	350.6ms
	Δ *IMP*	**3.40%**	**4.80%**	**41.1%**	**65.6%**	**2.48%**	**0.23%**	**76.9%**	**76.0%**	**0.81%**	**1.16%**	**69.0%**	**63.5%**

period length aligns with the temporal periodicity of the data. Specifically, ETTh1, ETTm1, and Solar achieve their best performance when $T_{\text{per}} = 24$, 96, and 144, respectively, corresponding to daily cycles under their sampling frequencies. Weather achieves the best result at $T_{\text{per}} = 288$, corresponding to a two-day cycle. In contrast, mismatched period lengths lead to performance degradation.

5.4 Plug-and-Play Analysis

To verify the plug-and-play property of the proposed representation translator, we integrate it into three representative patch-based backbones: PatchTST (Nie et al., 2023), Crossformer (Zhang & Yan, 2023), and TimeXer (Wang et al., 2024). To ensure a fair evaluation, the look-back window is uniformly set to 720. As shown in Table 3, the variants with the translator (+Translator) achieve comparable or slightly better forecasting accuracy while substantially improving computational efficiency. Quantitatively, representation translation reduces the average MSE and MAE by 2.58% and 1.95%, respectively, while decreasing parameters and cumulative training time by 60.35% and 58.92%. These results indicate that training separate models for each horizon often introduces substantial computational redundancy, which can be effectively alleviated via the "base encoder + translator" design.

6 Conclusion

In this paper, we study the problem of cross-horizon adaptation in time series forecasting. We argue that different forecasting horizons often require different representations, and directly reusing a fixed representation can lead to cross-horizon representation mismatch. To address this problem, we propose HorizonBridge, which employs a horizon-conditioned translator to adapt shared base representations to target forecasting horizons. We also devise an adaptive patch embedding mechanism to better capture non-uniform information density and improve cross-horizon representation compatibility. Extensive experiments show that HorizonBridge achieves competitive performance across multiple forecasting benchmarks and effectively adapts to different target horizons without retraining the backbone network. Horizon-conditioned representation adaptation provides a promising direction toward more flexible and efficient forecasting models.

AI USE STATEMENT

We used generative AI solely to polish the language of the manuscript. The research ideas, methodology, figures, and analysis were produced by the authors. AI-assisted revisions were limited to wording and expression, and the authors checked all changes to preserve the intended technical meaning.

REFERENCES

Sravan Kumar Ankireddy, Nikita Seleznev, Nam H. Nguyen, Yulun Wu, Senthil Kumar, Furong Huang, and C. Bayan Bruss. TimeSqueeze: Dynamic patching for efficient time series forecasting. In *NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models*, 2025. URL https://openreview.net/forum?id=eBrZLICc6S.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Kuiye Ding, Fanda Fan, Chunyi Hou, Zheya Wang, Lei Wang, Zhengxin Yang, and Jianfeng Zhan. TimeMosaic: Temporal heterogeneity guided time series forecasting via adaptive granularity patch and segment-wise decoding. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 20790–20798. AAAI Press, 2026. doi: 10.1609/AAAI.V40I25.39218. URL https://doi.org/10.1609/aaai.v40i25.39218.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the 9th International Conference on Learning Representations*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Ky Fan. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences*, 35(11):652–655, 1949.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=FVvf69a5rx.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Ao Hu, Liangjian Wen, Jiang Duan, Yong Dai, Dongkai Wang, Jun Wang, Yan He, Yukun Zhang, Ruoxi Jiang, and Zenglin Xu. PMDformer: Patch-mean decoupling transformer for long-term forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rfJ41gK9Ct.

Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 1510–1519. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.167. URL https://doi.org/10.1109/ICCV.2017.167.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *Proceedings of the 3rd International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980.

Maurice Kraus, Felix Divo, Devendra Singh Dhami, and Kristian Kersting. xLSTM-mixer: Multivariate time series forecasting by mixing via scalar memories. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=JlVn0XRpy0.

Yujie Li, Zezhi Shao, Chengqing Yu, Yisong Fu, Tao Sun, Yongjun Xu, and Fei Wang. APT: affine prototype-timestamp for time series forecasting under distribution shift. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 15180–15188. AAAI Press, 2026. doi: 10.1609/AAAI.V40I18.38542. URL https://doi.org/10.1609/aaai.v40i18.38542.

Shengsheng Lin, Haojun Chen, Haijie Wu, Chunyun Qiu, and Weiwei Lin. Temporal query network for efficient multivariate time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, 2025. URL https://proceedings.mlr.press/v267/lin25e.html.

Kun Liu, Renjun Jia, Ruifeng Yang, Xirui Zeng, Yuqi Liang, and Cen Chen. Are global dependencies necessary? scalable time series forecasting via local cross-variate modeling. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CNVL194fO5.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

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 *Proceedings of the 12th International Conference on Learning Representations*. OpenReview.net, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *Proceedings of the 13th International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=KMCJXjlDDr.

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

Aitian Ma, Dongsheng Luo, and Mo Sha. MixLinear: Extreme low-resource multivariate time series forecasting with 0.1K parameters. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=QUj0KuCumD.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the 11th International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Yiming Niu, Jinliang Deng, and Yongxin Tong. PhaseFormer: From patches to phases for efficient and effective time series forecasting. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Lk9SqMQzhX.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, volume 32, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 3942–3951. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11671. URL https://doi.org/10.1609/aaai.v32i1.11671.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024. doi: 10.14778/3665844.3665863. URL https://www.vldb.org/pvldb/vol17/p2363-hu.pdf.

Andrey V. Savchenko and Oleg Kachan. HN-MVTS: hypernetwork-based multivariate time series forecasting. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 25200–25208. AAAI Press, 2026. doi: 10.1609/AAAI.V40I30.39711. URL https://doi.org/10.1609/aaai.v40i30.39711.

Minlan Shao, Zijian Zhang, Yili Wang, Yiwei Dai, Xu Shen, and Xin Wang. HyperD: Hybrid periodicity decoupling framework for traffic forecasting. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 15689–15697. AAAI Press, 2026. doi: 10.1609/AAAI.V40I18.38599. URL https://doi.org/10.1609/aaai.v40i18.38599.

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 *Proceedings of the 13th International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e1wDDFmlVu.

Shiyu Wang, Jiawei Li, Xiaoming Shi, Zhou Ye, Baichuan Mo, Wenze Lin, Shengtong Ju, Zhixuan Chu, and Ming Jin. TimeMixer++: A general time series pattern machine for universal predictive analysis. In *Proceedings of the 13th International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=1CLzLXSFNn.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. TimeXer: Empowering transformers for time series forecasting with exogenous variables. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 469–498. Curran Associates, Inc., 2024. doi: 10.52202/079017-0015. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0113ef4642264adc2e6924a3cbbdf532-Paper-Conference.pdf.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, pp. 22419–22430, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Nan Yin, Li Shen, Huan Xiong, Bin Gu, Chong Chen, Xian-Sheng Hua, Siwei Liu, and Xiao Luo. Messages are never propagated alone: Collaborative hypergraph neural network for time-series forecasting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2333–2347, 2024. doi: 10.1109/TPAMI.2023.3331389.

Guoqi Yu, Jing Zou, Xiaowei Hu, Angelica I Aviles-Rivero, Jing Qin, and Shujun Wang. Revitalizing multivariate time series forecasting: Learnable decomposition with inter-series dependencies and intra-series variations modeling. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=87CYNyCGOo.

Onintze Zaballa, Verónica Álvarez, and Santiago Mazuelas. Adaptive multi-task learning for probabilistic load forecasting. *IEEE Transactions on Power Systems*, 41(3):1992–2004, 2026. doi: 10.1109/TPWRS.2025.3648811.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128. AAAI Press, 2023. doi: 10.1609/AAAI.V37I9.26317. URL https://doi.org/10.1609/aaai.v37i9.26317.

Jiawen Zhang, Shun Zheng, Xumeng Wen, Xiaofang Zhou, Jiang Bian, and Jia Li. ElasTST: Towards Robust Varied-Horizon Forecasting with Elastic Time-Series Transformer. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119174–119197. Curran Associates, Inc., 2024. doi: 10.52202/079017-3786. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d7aa002885ccbe68cf6880da583761b2-Paper-Conference.pdf.

Yang Zhang, Wenbo Yang, Jun Wang, Qiang Ma, and Jie Xiong. CAMEF: causal-augmented multi-modality event-driven financial forecasting by integrating time series patterns and salient macroeconomic announcements. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy (eds.), *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pp. 3867–3878. ACM, 2025a. doi: 10.1145/3711896.3736872. URL https://doi.org/10.1145/3711896.3736872.

Yitian Zhang, Liheng Ma, Antonios Valkanas, Boris N. Oreshkin, and Mark Coates. SKOLR: Structured koopman operator linear RNN for time-series forecasting. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75734–75756. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/zhang25be.html.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

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

A DATASETS

We conduct experiments on widely used multivariate time series forecasting benchmarks covering different real-world scenarios, including energy, traffic, weather, and solar power systems. These benchmarks cover diverse temporal patterns and data scales, providing a broad evaluation setting for assessing HorizonBridge under different forecasting scenarios. Following the standard evaluation protocol in previous forecasting works (Liu et al., 2024; Nie et al., 2023), we split each dataset in chronological order into training, validation, and test sets. Specifically, the split ratio is set to 6:2:2 for the ETT datasets and 7:1:2 for the remaining datasets. Detailed dataset statistics are summarized in Table 4.

- **ETT**[1]**:** The ETT dataset contains transformer oil temperature and load-related measurements collected from electricity transformers over two years. It includes four subsets: ETTh1 and ETTh2 are recorded hourly, while ETTm1 and ETTm2 are sampled every 15 minutes.
- **Weather**[2]**:** The Weather dataset consists of 21 meteorological variables, such as temperature, humidity, and pressure, collected at 10-minute intervals in Germany during 2020 (Wu et al., 2021).
- **Electricity**[3]**:** The Electricity dataset records the hourly electricity consumption of 321 clients in Portugal from 2012 to 2014.
- **Traffic**[4]**:** The Traffic dataset contains hourly road occupancy measurements collected by 862 sensors on highways in the San Francisco Bay Area. This dataset exhibits strong spatial and temporal variability due to complex traffic dynamics.
- **Solar**[5]**:** The Solar dataset contains solar power generation records from 137 photovoltaic plants, sampled every 10 minutes in 2007 by the U.S. National Renewable Energy Laboratory.

Table 4: Summary statistics of all evaluation datasets. "Dataset size" indicates the number of timesteps in the training, validation, and testing splits.

Dataset	Variates	Dataset size	Frequency
ETTh1	7	(8,545, 2,881, 2,881)	1 hour
ETTh2	7	(8,545, 2,881, 2,881)	1 hour
ETTm1	7	(34,465, 11,521, 11,521)	15 min
ETTm2	7	(34,465, 11,521, 11,521)	15 min
Weather	21	(36,792, 5,271, 10,540)	10 min
Traffic	862	(12,185, 1,757, 3,509)	1 hour
Solar	137	(36,792, 5,271, 10,540)	10 min
Electricity	321	(18,317, 2,633, 5,261)	1 hour

B DETAILS OF ADAPTIVE PATCH PARTITIONING

MDL Cost. We adopt the minimum description length (MDL) principle as the criterion for adaptive patch partitioning. Under this criterion, the sequence is partitioned into local patches that can be compactly described by simple statistical models. For locally stationary regions with consistent patterns, a relatively long patch can provide an effective representation. In contrast, for regions containing abrupt changes, trend shifts, or strong local fluctuations, new boundaries are needed to

[1] https://github.com/zhouhaoyi/ETDataset

[2] https://www.bgc-jena.mpg.de/wetter/

[3] https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

[4] https://pems.dot.ca.gov/

[5] https://www.nrel.gov/grid/solar-power-data.html

reduce the residual description cost. Given data D and a description model $\mathcal{M}$, the total description length (TDL) can be formulated as

$$\mathrm{TDL} = \mathcal{L}(\mathcal{M}) + \mathcal{L}(D|\mathcal{M}), \tag{19}$$

where $\mathcal{L}(\mathcal{M})$ denotes the coding length required to describe the model itself, and $\mathcal{L}(D|\mathcal{M})$ denotes the coding length required to describe the data residuals given the model. For a candidate patch P, we describe it using a local Gaussian model:

$$\mathcal{M}_P : x_t \sim \mathcal{N}(\mu_P, \sigma_P^2), \quad t \in P, \tag{20}$$

where μ_P and σ_P^2 denote the local mean and variance, respectively. These two parameters are estimated by the sample mean and variance:

$$\hat{\mu}_P = \frac{1}{|P|}\sum_{t\in P} x_t, \quad \hat{\sigma}_P^2 = \frac{1}{|P|}\sum_{t\in P}(x_t - \hat{\mu}_P)^2. \tag{21}$$

The likelihood of patch P can then be written as

$$p(P|\hat{\mu}_P, \hat{\sigma}_P^2) = \prod_{t\in P} \frac{1}{\sqrt{2\pi\hat{\sigma}_P^2}} \exp\left(-\frac{(x_t - \hat{\mu}_P)^2}{2\hat{\sigma}_P^2}\right). \tag{22}$$

According to Shannon's source coding theorem, the optimal coding length can be given by the negative log-likelihood. The data description length of a patch is therefore:

$$\begin{aligned}\mathcal{L}(x_P|\mathcal{M}_P) &= -\log p(P|\hat{\mu}_P, \hat{\sigma}_P^2) \\ &= \frac{|P|}{2}\log(2\pi\hat{\sigma}_P^2) + \frac{1}{2\hat{\sigma}_P^2}\sum_{t\in P}(x_t - \hat{\mu}_P)^2 \\ &= \frac{|P|}{2}\log\left(2\pi e\hat{\sigma}_P^2\right).\end{aligned} \tag{23}$$

For the model description length, we use the commonly adopted asymptotic parameter coding cost in MDL (Grünwald, 2007):

$$\mathcal{L}(\mathcal{M}_P) \approx \frac{k}{2}\log|P|, \tag{24}$$

where k denotes the number of model parameters. Since the local Gaussian model contains two parameters, i.e., mean and variance, we have $k = 2$. Therefore, the total description length of patch P can be formulated as

$$\begin{aligned}\mathrm{Cost}_{\mathrm{MDL}}(P) &\approx \frac{|P|}{2}\log\left(2\pi e\hat{\sigma}_P^2\right) + \frac{k}{2}\log|P| \\ &= \frac{|P|}{2}\log\left(2\pi e\hat{\sigma}_P^2\right) + \log|P|.\end{aligned} \tag{25}$$

Dynamic Programming Algorithm. Based on the MDL cost of each candidate patch, the optimal patch partition can be solved by dynamic programming. To reduce the search overhead, we introduce a search unit size q, and only search patch boundaries at positions separated by q. For an input sequence of length T, we divide it into T/q coarse units. Accordingly, the candidate boundary set is reduced to $\{0, q, 2q, \ldots, T\}$. In this case, the time complexity of the dynamic programming algorithm is

$$\mathcal{O}\left(N\left(\frac{T}{q}\right)^2\right). \tag{26}$$

Compared with directly searching over the original time steps, which leads to $\mathcal{O}(NT^2)$ complexity, introducing the search unit size q significantly reduces the boundary search cost. A smaller q provides finer-grained boundary choices, while a larger q further reduces the computational overhead. Algorithm 1 summarizes the overall procedure.

Algorithm 1: MDL-based Adaptive Patch Partitioning

Input: Time series $x_{1:T}$, number of patches N, unit size q
Output: Partition boundaries $\mathcal{B} = \{b_k\}_{k=0}^{N}$

Set $U = T/q$ and search candidate boundaries over coarse units $\{0, 1, \ldots, U\}$.
Initialize $\mathrm{DP}(0,0) = 0$, and $\mathrm{DP}(0,j) = \infty$ for $j > 0$.
for $m = 1$ **to** N **do**
  for $j = m$ **to** U **do**
    $\mathrm{DP}(m,j) = \infty$
    for $i = m-1$ **to** $j-1$ **do**
      Compute the MDL cost $\mathrm{DL}(i,j)$ of patch $(i,j]$.
      $c = \mathrm{DP}(m-1,i) + \mathrm{DL}(i,j)$
      if $c < \mathrm{DP}(m,j)$ **then**
        $\mathrm{DP}(m,j) = c$
        $\mathrm{Prev}(m,j) = i$
      end if
    end for
  end for
end for
Backtrack from (N,U) using Prev to obtain unit-level boundaries $\{\beta_k\}_{k=0}^{N}$.
Map them back to time indices: $b_k = q\beta_k$, $k = 0, 1, \ldots, N$.
return $\mathcal{B} = \{b_k\}_{k=0}^{N}$.

C Implementation Details

C.1 Architecture Details

Reversible Instance Normalization. Real-world time series may exhibit distribution shifts across different periods, which can affect the generalization ability of forecasting models. To address this issue, HorizonBridge adopts reversible instance normalization. Specifically, each input instance is standardized before being fed into the model, and the prediction is transformed back to the original scale at the output side:

$$\begin{aligned} X_t &= \frac{X_t - \mu}{\sqrt{\sigma^2 + \epsilon}}, \\ \bar{Y}_t &= \bar{Y}_t \cdot \sqrt{\sigma^2 + \epsilon} + \mu. \end{aligned} \tag{27}$$

Here, μ and σ^2 denote the mean and variance of the input sequence, respectively, and ϵ is a small constant for numerical stability.

Base Encoder. Transformer architectures have become a widely used backbone in time series forecasting due to their strong capability for long-sequence modeling (Niu et al., 2026; Hu et al., 2026; Nie et al., 2023). Following these works, HorizonBridge adopts a standard patch-based Transformer architecture that performs attention over both the temporal and variable dimensions. Specifically, for the temporal attention layer, we apply the attention mechanism within the sequence corresponding to each variable to capture temporal dependencies:

$$\begin{aligned} \hat{\mathbf{H}}_{c,:}^{(l)} &= \mathrm{LN}\left(\mathbf{H}_{c,:}^{(l-1)} + \mathrm{Attn}(\mathbf{H}_{c,:}^{(l-1)}, \mathbf{H}_{c,:}^{(l-1)}, \mathbf{H}_{c,:}^{(l-1)})\right), \\ \mathbf{H}_{c,:}^{(l)} &= \mathrm{LN}\left(\hat{\mathbf{H}}_{c,:}^{(l)} + \mathrm{MLP}(\hat{\mathbf{H}}_{c,:}^{(l)})\right). \end{aligned} \tag{28}$$

Here, $\mathbf{H}_{c,:} \in \mathbb{R}^{N \times d}$ denotes the patch token sequence corresponding to variable c, $\mathrm{Attn}(\cdot)$ denotes the self-attention operation, and $\mathrm{LN}(\cdot)$ denotes layer normalization. For the variable attention layer, we model the relationships among all variables within each patch position:

$$\begin{aligned} \hat{\mathbf{H}}_{:,n}^{(l)} &= \mathrm{LN}\left(\mathbf{H}_{:,n}^{(l-1)} + \mathrm{Attn}(\mathbf{H}_{:,n}^{(l-1)}, \mathbf{H}_{:,n}^{(l-1)}, \mathbf{H}_{:,n}^{(l-1)})\right), \\ \mathbf{H}_{:,n}^{(l)} &= \mathrm{LN}\left(\hat{\mathbf{H}}_{:,n}^{(l)} + \mathrm{MLP}(\hat{\mathbf{H}}_{:,n}^{(l)})\right). \end{aligned} \tag{29}$$

Here, $\mathbf{H}_{:,n} \in \mathbb{R}^{C\times d}$ denotes the tokens of all variables corresponding to patch index n. By alternately performing attention modeling along the temporal and variable dimensions, the base encoder can capture both temporal dependencies and cross-variate dependencies. The final output $\mathbf{H}^{(L)}$ serves as the shared base representation and is fed into the subsequent representation translator.

Horizon Embedding. To provide the representation translator with forecasting horizon information, HorizonBridge encodes each forecasting horizon H into a horizon embedding $\mathbf{e}_H \in \mathbb{R}^d$. In implementation, we support two encoding forms: learnable horizon embedding and sinusoidal horizon embedding. For the learnable form, we assign an independent trainable vector to each horizon: $\mathbf{e}_H = \mathrm{Embed}(H)$, where $\mathrm{Embed}(\cdot)$ denotes a learnable embedding table. For the sinusoidal form, we adopt a fixed encoding scheme similar to the positional encoding used in Transformer. Given a target horizon H, we construct the horizon embedding using sinusoidal functions with different frequencies:

$$\mathbf{e}_H = [\cos(H\omega_1), \sin(H\omega_1), \ldots, \cos(H\omega_{d/2}), \sin(H\omega_{d/2})], \tag{30}$$

where the frequencies ω_i follow an exponential decay:

$$\omega_i = \exp\left(-\frac{i-1}{\lfloor d/2 \rfloor - 1}\log P\right) \in \left[\frac{1}{P},\ 1\right]. \tag{31}$$

Here, $i = 1, \ldots, \lfloor d/2 \rfloor$, and P controls the variation range of the frequencies. Compared with learnable embeddings, sinusoidal embeddings introduce no additional trainable parameters and can encode arbitrary horizons without additional training.

C.2 Training and Testing.

Experimental Setup. All experiments are implemented with PyTorch 2.5.1 (Paszke et al., 2019) and conducted on two NVIDIA RTX PRO 6000 96GB GPUs. The computing environment runs Ubuntu 22.04.5 LTS with Python 3.10.20 and CUDA 12.6, equipped with AMD EPYC 9654 96-Core Processors. We optimize the model using the Adam optimizer (Kingma & Ba, 2015), with a maximum of 50 training epochs, and an early stopping strategy with a patience of 5. Following the fair evaluation protocol proposed by TFB (Qiu et al., 2024), we disable the `drop_last` option during testing to ensure that all test samples are included. We investigate different model configurations by adjusting several key hyperparameters. The numbers of layers in the base encoder and representation translator are selected from $\{1, 2, 3, 4, 5\}$ and $\{1, 2, 3\}$, respectively. The hidden dimension d and number of basis matrices are chosen from $\{64, 128, 256, 512\}$ and $\{3, 5, 8, 10\}$. The learning rate is searched from 10^{-4} to 0.05. The period length T_{per} is determined according to the dominant temporal periodicity of each dataset.

Metric. We evaluate forecasting performance using Mean Absolute Error (MAE) and Mean Squared Error (MSE) on test sets, which measure the average absolute deviation and squared deviation between the predictions and ground truth, respectively:

$$\mathrm{MAE} = \frac{1}{|Y|}\sum_{i=1}^{|Y|}\left|Y_i - \hat{Y}_i\right|, \mathrm{MSE} = \frac{1}{|Y|}\sum_{i=1}^{|Y|}\left(Y_i - \hat{Y}_i\right)^2. \tag{32}$$

Training Strategy. For all baselines, results are obtained by independently training a model for each forecasting horizon. In contrast, HorizonBridge adopts a two-stage training strategy. In the first stage, the embedding module and base encoder are trained with the base horizon $H = 96$ to learn shared historical representations. These components are then frozen, while the representation translator and shared prediction head are jointly optimized to adapt to other target horizons. During the second stage, the training and validation sets are constructed based on the longest horizon $H = 720$. Shorter-horizon targets are obtained by truncating the corresponding prefixes of these sequences. This strategy ensures consistent sample sizes across different horizons and prevents the model from biasing toward any specific horizon. During testing, we follow the horizon-specific test splits adopted by previous works to ensure a fair comparison with baseline methods.

Loss Function. For each forecasting horizon H, we adopt a hybrid MAE loss in both the temporal and frequency domains to jointly optimize forecasting accuracy and spectral consistency:

$$\begin{aligned}\mathcal{L}_{\text{time}}^{(H)} &= \frac{1}{H}\sum_{i=1}^{H}\left|Y_i - \hat{Y}_i\right|,\\ \mathcal{L}_{\text{freq}}^{(H)} &= \frac{1}{F}\sum_{k=0}^{F-1}\left|\text{FFT}\left(\hat{Y}^{(H)}\right)_k - \text{FFT}\left(Y^{(H)}\right)_k\right|,\\ \mathcal{L}^{(H)} &= (1-\lambda)\mathcal{L}_{\text{time}}^{(H)} + \lambda\mathcal{L}_{\text{freq}}^{(H)},\end{aligned} \tag{33}$$

where FFT denotes the Fast Fourier Transform, $F = \lfloor H/2 \rfloor + 1$ is the number of frequency components, and $\lambda \in [0, 1]$ balances the contributions of temporal and frequency-domain losses.

For joint training over multiple horizons $\mathcal{H} = \{H_1, H_2, \ldots, H_M\}$, the overall objective is defined as:

$$\mathcal{L}_{\text{joint}} = \frac{1}{|\mathcal{H}|}\sum_{H\in\mathcal{H}}\mathcal{L}^{(H)}. \tag{34}$$

D Theoretical Analysis

This section provides a theoretical analysis of the representation mismatch problem in cross-horizon time-series forecasting, and explains why a horizon-conditioned representation translator is needed. We consider a linear dynamical system:

$$s_{t+1} = As_t + \epsilon_t,\ x_t = c^\top s_t + \eta_t, \tag{35}$$

where $s_t \in \mathbb{R}^M$ is the latent state, $A \in \mathbb{R}^{M\times M}$ is the system dynamics matrix, $c \in \mathbb{R}^M$ is the observation vector, and ϵ_t and η_t are independent zero-mean Gaussian noises. Given an input sequence $\mathcal{X}_T = x_{1:T} \in \mathbb{R}^T$, let

$$\mu_T := \mathbb{E}[s_T \mid \mathcal{X}_T] \in \mathbb{R}^M \tag{36}$$

be the posterior mean of the latent state given $\mathcal{X}_T$. Under the MSE loss, the Bayes-optimal H-step forecast is the conditional expectation:

$$m_H(\mathcal{X}_T) = \mathbb{E}\left[Y^{(H)} \mid \mathcal{X}_T\right] = G_H\mu_T, \tag{37}$$

where

$$G_H = \begin{bmatrix}c^\top A & c^\top A^2 & \cdots & c^\top A^H\end{bmatrix}^\top \in \mathbb{R}^{H\times M} \tag{38}$$

collects the effects of the system dynamics and the readout over the next H steps. It characterizes which directions in the current latent state μ_T influence the outputs at future steps $1, \ldots, H$. The true future sequence can further be written as

$$Y^{(H)} = G_H\mu_T + \xi^{(H)}, \tag{39}$$

where $\xi^{(H)}$ denotes the unpredictable residual. By construction, $\mathbb{E}\left[\xi^{(H)} \mid \mathcal{X}_T\right] = 0$, $\mathbb{E}[\xi^{(H)}\mu_T^\top] = 0$.

Real-world time series are often driven by latent high-dimensional system dynamics, while the encoder of a deep model compresses historical information into a lower-dimensional latent representation. To describe this process, we regard the representation learned by the encoder as a linear compression of the posterior mean of the latent state:$z = R\mu_T \in \mathbb{R}^D$, where $R \in \mathbb{R}^{D\times M}$, $D < M$. The model prediction can then be written as $\hat{Y}^{(H)} = WR\mu_T$. To simplify the subsequent analysis, we consider minimizing the MSE loss:

$$\begin{aligned}\mathcal{L}_H(R, W) &= \mathbb{E}\left[\left\|G_H\mu_T + \xi^{(H)} - WR\mu_T\right\|_2^2\right]\\ &= \mathbb{E}\left[\left\|\left(G_H - WR\right)\mu_T + \xi^{(H)}\right\|_2^2\right]\\ &= \mathbb{E}\left[\|(G_H - WR)\mu_T\|_2^2\right] + \mathbb{E}\left[\left\|\xi^{(H)}\right\|_2^2\right]\\ &\quad + 2\mathbb{E}\left[\left((G_H - WR)\mu_T\right)^\top \xi^{(H)}\right].\end{aligned} \tag{40}$$

For the last term in (40), since $\xi^{(H)}$ is orthogonal to μ_T, we have

$$\mathbb{E}\left[((G_H - WR)\mu_T)^\top \xi^{(H)}\right] = \mathbb{E}\left[\left(\xi^{(H)}\right)^\top (G_H - WR)\mu_T\right] = 0. \tag{41}$$

Define $\Sigma := \mathbb{E}[\mu_T \mu_T^\top]$ and assume $\Sigma \succ 0$. Extracting the part of the risk that depends on (R, W) gives

$$\begin{aligned}\tilde{\mathcal{L}}_H(R, W) &:= \mathbb{E}\|(G_H - WR)\mu_T\|_2^2 \\ &= \mathbb{E}\left[\mu_T^\top (G_H - WR)^\top (G_H - WR)\mu_T\right] \\ &= \mathrm{tr}\left((G_H - WR)^\top (G_H - WR)\mathbb{E}[\mu_T \mu_T^\top]\right) \\ &= \mathrm{tr}\left((G_H - WR)\Sigma(G_H - WR)^\top\right). \end{aligned} \tag{42}$$

For a fixed representation matrix R, the optimal prediction head can be obtained by minimizing $\tilde{\mathcal{L}}_H(R, W)$ with respect to W. Expanding the loss gives

$$\tilde{\mathcal{L}}_H(R, W) = \mathrm{tr}(G_H \Sigma G_H^\top) - 2\mathrm{tr}(WR\Sigma G_H^\top) + \mathrm{tr}(WR\Sigma R^\top W^\top). \tag{43}$$

Taking the derivative with respect to W and setting it to zero yields

$$\frac{\partial \tilde{\mathcal{L}}_H}{\partial W} = -2G_H \Sigma R^\top + 2W(R\Sigma R^\top) = 0. \tag{44}$$

Therefore, when $R\Sigma R^\top$ is invertible, the optimal prediction head is given by

$$W_H^*(R) = G_H \Sigma R^\top (R\Sigma R^\top)^{-1}. \tag{45}$$

Proposition 1 [Optimal Representation Subspace]. Let $R \in \mathbb{R}^{D \times M}$ be full-rank, and define

$$K_H := \Sigma^{1/2} G_H^\top G_H \Sigma^{1/2} \in \mathbb{R}^{M \times M}. \tag{46}$$

Let its eigendecomposition be written as

$$K_H u_i^{(H)} = \lambda_i^{(H)} u_i^{(H)},\ \lambda_1^{(H)} \geq \cdots \geq \lambda_M^{(H)}. \tag{47}$$

Among all representation matrices R with rank D, the minimum value of the risk is

$$\min_{\mathrm{rank}(R)=D} \tilde{\mathcal{L}}_H(R, W_H^*) = \|F_H\|_F^2 - \sum_{i=1}^{D} \lambda_i^{(H)}, \tag{48}$$

where $F_H := G_H \Sigma^{1/2}$. This minimum is attained when the row space of $C_R = R\Sigma^{1/2}$ equals the subspace spanned by the top D eigenvectors of K_H, i.e.,

$$\mathrm{Row}(C_R) = \mathrm{span}\{u_1^{(H)}, \ldots, u_D^{(H)}\}. \tag{49}$$

We refer to this subspace as the optimal representation subspace for horizon H.

Proof. Substituting the optimal head into $\tilde{\mathcal{L}}_H(R, W)$ gives

$$\tilde{\mathcal{L}}_H^*(R) = \mathrm{tr}(G_H \Sigma G_H^\top) - \mathrm{tr}\left(G_H \Sigma R^\top (R\Sigma R^\top)^{-1} R\Sigma G_H^\top\right). \tag{50}$$

To reveal its structure, define

$$F_H = G_H \Sigma^{1/2},\ C_R = R\Sigma^{1/2}. \tag{51}$$

Then

$$R\Sigma R^\top = C_R C_R^\top,\ G_H \Sigma R^\top = F_H C_R^\top. \tag{52}$$

Therefore, the second term in (50) can be written as

$$\mathrm{tr}\left(F_H C_R^\top (C_R C_R^\top)^{-1} C_R F_H^\top\right). \tag{53}$$

Define

$$\Pi_R = C_R^\top (C_R C_R^\top)^{-1} C_R. \tag{54}$$

Since R is full-rank, C_R has full row rank, and thus Π_R is the orthogonal projection matrix onto the row space of C_R. Hence $\Pi_R^\top = \Pi_R$ and $\Pi_R^2 = \Pi_R$. We obtain

$$\begin{aligned}\tilde{\mathcal{L}}_H^*(R) &= \mathrm{tr}(F_H F_H^\top) - \mathrm{tr}(F_H \Pi_R F_H^\top) \\ &= \|F_H\|_F^2 - \|F_H \Pi_R\|_F^2. \end{aligned} \tag{55}$$

Since $\|F_H\|_F^2$ is independent of R, minimizing $\tilde{\mathcal{L}}_H^*(R)$ is equivalent to maximizing $\|F_H \Pi_R\|_F^2$. Moreover,

$$\begin{aligned}\|F_H \Pi_R\|_F^2 &= \mathrm{tr}(F_H \Pi_R \Pi_R F_H^\top) \\ &= \mathrm{tr}(F_H \Pi_R F_H^\top) \\ &= \mathrm{tr}(\Pi_R K_H), \end{aligned} \tag{56}$$

where $K_H = F_H^\top F_H$. The problem is therefore reduced to

$$\max_{\Pi_R} \mathrm{tr}(\Pi_R K_H), \tag{57}$$

where Π_R is a rank-D orthogonal projection matrix, and Π_R is equivalent to the row space of C_R. By the Ky Fan maximum principle (Fan, 1949), the maximum value is the sum of the top D eigenvalues of K_H, namely $\sum_{i=1}^D \lambda_i^{(H)}$. Substituting this value into (55) gives

$$\tilde{\mathcal{L}}_H^*(R) = \|F_H\|_F^2 - \sum_{i=1}^D \lambda_i^{(H)}. \tag{58}$$

The maximum in (57) is attained under the condition in (49). This completes the proof.

Proposition 2 [Representation Mismatch]. Let

$$S_H^* = \mathrm{span}\{u_1^{(H)}, \ldots, u_D^{(H)}\} \tag{59}$$

be the subspace spanned by the eigenvectors corresponding to the top D eigenvalues of K_H. For two prediction horizons $H_1 \neq H_2$, suppose that $S_{H_1}^* \neq S_{H_2}^*$ and that K_{H_2} has a spectral gap between its D-th and $(D+1)$-th eigenvalues, i.e., $\lambda_D^{(H_2)} > \lambda_{D+1}^{(H_2)}$. Then reusing the representation that is optimal for H_1 on horizon H_2 incurs a strictly positive excess risk, with the following lower bound:

$$\begin{aligned}\tilde{\mathcal{L}}_{H_2}^*(R_{H_1}^*) - \min_R \tilde{\mathcal{L}}_{H_2}^*(R) &\geq \frac{1}{2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right) \left\|\Pi_{H_1}^* - \Pi_{H_2}^*\right\|_F^2 \\ &\geq \frac{4D}{\pi^2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right) \bar{\theta}^2, \end{aligned} \tag{60}$$

where Π_H^* is the orthogonal projection onto S_H^*, and $\bar{\theta}$ is the RMS principal angle between $S_{H_1}^*$ and $S_{H_2}^*$.

Proof. Consider the eigendecomposition of K_{H_2}:

$$K_{H_2} = U \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} U^\top, \tag{61}$$

where $\Lambda_1 = \mathrm{diag}(\lambda_1, \ldots, \lambda_D)$ and $\Lambda_2 = \mathrm{diag}(\lambda_{D+1}, \ldots, \lambda_M)$. For any rank-$D$ representation subspace $\mathrm{Row}(C_R)$, denote its orthogonal projection matrix by Π. In the eigenbasis of K_{H_2}, write

$$\bar{\Pi} = U^\top \Pi U = \begin{bmatrix} \Pi_{11} & \Pi_{12} \\ \Pi_{21} & \Pi_{22} \end{bmatrix}. \tag{62}$$

Here $U = [U_1, U_2]$, where $U_1 \in \mathbb{R}^{M \times D}$ corresponds to the top D eigenvectors and $U_2 \in \mathbb{R}^{M \times (M-D)}$ corresponds to the remaining eigenvectors. Moreover, $\Pi_{11} = U_1^\top \Pi U_1$, $\Pi_{22} = U_2^\top \Pi U_2$. Since Π is a rank-D orthogonal projection matrix, we have $\mathrm{tr}(\Pi_{11}) + \mathrm{tr}(\Pi_{22}) = D$, and therefore

$$\mathrm{tr}(I_D - \Pi_{11}) = D - \mathrm{tr}(\Pi_{11}) = \mathrm{tr}(\Pi_{22}). \tag{63}$$

For any rank-D representation subspace, Proposition 1 implies that its excess risk over the optimal subspace $S_{H_2}^*$ is

$$\Delta\mathcal{L} = \mathrm{tr}(\Pi_{H_2}^* K_{H_2}) - \mathrm{tr}(\Pi K_{H_2}). \tag{64}$$

By Proposition 1,

$$\mathrm{tr}(\Pi^*_{H_2} K_{H_2}) = \mathrm{tr}(\Lambda_1), \tag{65}$$

and

$$\begin{aligned}\mathrm{tr}(\Pi K_{H_2}) &= \mathrm{tr}(\Pi U \Lambda U^\top) = \mathrm{tr}(\bar{\Pi}\Lambda) \\ &= \mathrm{tr}(\Pi_{11}\Lambda_1) + \mathrm{tr}(\Pi_{22}\Lambda_2).\end{aligned} \tag{66}$$

Therefore,

$$\begin{aligned}\Delta\mathcal{L} &= \mathrm{tr}(\Pi^*_{H_2} K_{H_2}) - \mathrm{tr}(\Pi K_{H_2}) \\ &= \mathrm{tr}(\Lambda_1) - \mathrm{tr}(\Pi_{11}\Lambda_1) - \mathrm{tr}(\Pi_{22}\Lambda_2) \\ &= \mathrm{tr}((I_D - \Pi_{11})\Lambda_1) - \mathrm{tr}(\Pi_{22}\Lambda_2).\end{aligned} \tag{67}$$

Since $\Lambda_1 \succeq \lambda_D I_D$ and $\Lambda_2 \preceq \lambda_{D+1} I_{M-D}$, we obtain

$$\begin{aligned}\Delta\mathcal{L} &\geq \lambda_D \mathrm{tr}(I_D - \Pi_{11}) - \lambda_{D+1}\mathrm{tr}(\Pi_{22}) \\ &= (\lambda_D - \lambda_{D+1})\mathrm{tr}(\Pi_{22}).\end{aligned} \tag{68}$$

On the other hand, two rank-D orthogonal projection matrices satisfy

$$\begin{aligned}\|\Pi - \Pi^*_{H_2}\|_F^2 &= \mathrm{tr}\left((\Pi - \Pi^*_{H_2})^2\right) \\ &= \mathrm{tr}(\Pi^2) + \mathrm{tr}((\Pi^*_{H_2})^2) - 2\mathrm{tr}(\Pi\Pi^*_{H_2}) \\ &= 2D - 2\mathrm{tr}(\Pi\Pi^*_{H_2}).\end{aligned} \tag{69}$$

Moreover,

$$\begin{aligned}\mathrm{tr}(\Pi\Pi^*_{H_2}) &= \mathrm{tr}\left(U^\top \Pi U \begin{bmatrix} I_D & 0 \\ 0 & 0 \end{bmatrix}\right) \\ &= \mathrm{tr}\left(\begin{bmatrix} \Pi_{11} & \Pi_{12} \\ \Pi_{21} & \Pi_{22} \end{bmatrix} \begin{bmatrix} I_D & 0 \\ 0 & 0 \end{bmatrix}\right) \\ &= \mathrm{tr}(\Pi_{11}).\end{aligned} \tag{70}$$

Thus,

$$\|\Pi - \Pi^*_{H_2}\|_F^2 = 2\left(D - \mathrm{tr}(\Pi_{11})\right) = 2\mathrm{tr}(\Pi_{22}), \tag{71}$$

which gives

$$\mathrm{tr}(\Pi_{22}) = \frac{1}{2}\|\Pi - \Pi^*_{H_2}\|_F^2. \tag{72}$$

Substituting (72) into (68) yields

$$\Delta\mathcal{L} \geq \frac{1}{2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right)\left\|\Pi - \Pi^*_{H_2}\right\|_F^2. \tag{73}$$

Taking $\Pi = \Pi^*_{H_1}$ gives

$$\Delta\mathcal{L} \geq \frac{1}{2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right)\left\|\Pi^*_{H_1} - \Pi^*_{H_2}\right\|_F^2. \tag{74}$$

Let U_{H_1} and U_{H_2} be orthonormal basis matrices of $S^*_{H_1}$ and $S^*_{H_2}$, respectively. Then $\Pi^*_H = U_H U_H^\top$. The principal angles $\theta_1, \ldots, \theta_D \in [0, \pi/2]$ between the two subspaces are defined by

$$\cos\theta_i = \sigma_i(U_{H_1}^\top U_{H_2}),\ i = 1, \ldots, D, \tag{75}$$

where $\sigma_i(\cdot)$ denotes the i-th singular value. We have

$$\|\Pi^*_{H_1} - \Pi^*_{H_2}\|_F^2 = 2\sum_{i=1}^{D}\sin^2\theta_i \geq \frac{8D}{\pi^2}\bar{\theta}^2, \tag{76}$$

where $\bar{\theta} = \sqrt{\frac{1}{D}\sum_{i=1}^{D}\theta_i^2}$ is the RMS principal angle, which reflects the average deviation between the two subspaces. Substituting this bound into (74) gives

$$\Delta\mathcal{L} \geq \frac{4D}{\pi^2}\left(\lambda_D^{(H_2)} - \lambda_{D+1}^{(H_2)}\right)\bar{\theta}^2. \tag{77}$$

This completes the proof.

To clarify the genericity of Proposition 2, the following lemma shows that, when the predictive rank increases within the representation capacity, the longer-horizon optima occupy only a measure-zero subset of the shorter-horizon optimum manifold.

Lemma 1. Let $H_1 < H_2$, $V_i = \mathrm{range}(K_{H_i})$, and $r_i = \dim V_i$. Let $\mathrm{Gr}(D, M)$ denote the Grassmann manifold of all D-dimensional subspaces of $\mathbb{R}^M$, and define

$$\mathcal{O}_i = \operatorname*{argmin}_{S \in \mathrm{Gr}(D,M)} \mathrm{tr}((I - P_S)K_{H_i}),$$

where P_S is the orthogonal projector onto S. If $r_1 < r_2$, $r_2 \le D$, and $D < M$, then $\mathcal{O}_2 \subseteq \mathcal{O}_1$ and

$$\dim \mathcal{O}_i = (D - r_i)(M - D).$$

Consequently,

$$\mathrm{codim}_{\mathcal{O}_1} \mathcal{O}_2 = (r_2 - r_1)(M - D) > 0,$$

so $\mathcal{O}_2$ has zero volume measure within $\mathcal{O}_1$. Therefore, although common optimal representations exist, they form a measure-zero subset of the shorter-horizon optimum manifold.

Proof. Since $K_H = \sum_{h=1}^{H} v_h v_h^\top$, we have $V_1 \subseteq V_2$. For $r_i \le D$, a D-dimensional subspace S is optimal for horizon H_i if and only if $V_i \subseteq S$. Hence

$$\mathcal{O}_i \cong \mathrm{Gr}(D - r_i, M - r_i),$$

and therefore

$$\dim \mathcal{O}_i = (D - r_i)\big((M - r_i) - (D - r_i)\big) = (D - r_i)(M - D).$$

Since $r_1 < r_2$, we have $V_1 \subsetneq V_2$, and thus $\mathcal{O}_2 \subseteq \mathcal{O}_1$. Its codimension in $\mathcal{O}_1$ is

$$(r_2 - r_1)(M - D) > 0.$$

A positive-codimension embedded submanifold has zero volume measure in its ambient manifold. Hence $\mathcal{O}_2$ has measure zero within $\mathcal{O}_1$. This completes the proof.

Effect of the Horizon on K_H. Expanding K_H gives

$$K_H = \Sigma^{1/2} \left[\sum_{h=1}^{H} (A^h)^\top c c^\top A^h \right] \Sigma^{1/2}. \tag{78}$$

As H increases, additional propagation terms $(A^h)^\top c c^\top A^h$ are accumulated in K_H, so the spectral structure of K_H may also change. To further illustrate this effect, suppose that A is diagonalizable:

$$A = V \Lambda V^{-1},\ \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_M). \tag{79}$$

Let $a := V^\top c \in \mathbb{R}^M$. Then

$$K_H = \Sigma^{1/2} V^{-T} M_H V^{-1} \Sigma^{1/2},$$
$$M_H := \sum_{h=1}^{H} \Lambda^h a a^\top \Lambda^h \in \mathbb{R}^{M \times M}. \tag{80}$$

The (i, j)-th entry of M_H is

$$[M_H]_{ij} = a_i a_j \sum_{h=1}^{H} (\lambda_i \lambda_j)^h. \tag{81}$$

This indicates that each entry of M_H is controlled by a horizon-dependent cumulative exponential factor $\sum_{h=1}^{H} (\lambda_i \lambda_j)^h$. Therefore, different dynamical modes are weighted and accumulated differently as the prediction horizon changes. When the modes have different growth or decay rates, these factors evolve at different rates with H, thereby changing the relative contributions of different modal directions in K_H. Such changes may further affect the spectral structure of K_H, and consequently shift the optimal representation subspace.

This phenomenon is consistent with the evolution patterns of time series. Real-world time series are usually composed of multiple dynamical modes, such as short-term fluctuations, periodic components, and trend components. These modes have different temporal scales and decay effects. Short-term forecasting is more easily affected by local fluctuations or high-frequency components. As the prediction horizon increases, slowly decaying or persistently growing long-term modes may

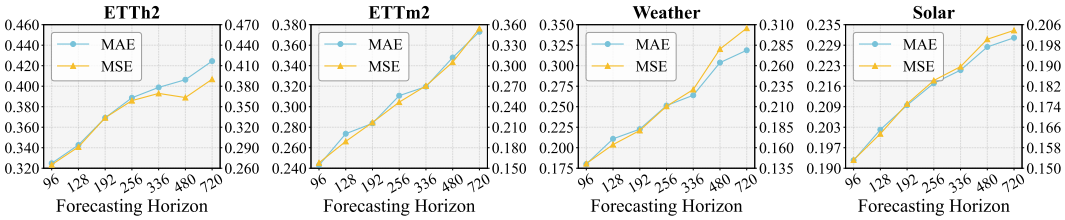


Figure 6: Generalization performance on unseen forecasting horizons for ETTh2, ETTm2, Weather and Solar datasets. The left and right y-axes represent the MSE and MAE values, respectively.

account for a larger proportion, while the influence of rapidly decaying short-term components is gradually diluted.

The above analysis is based on a time-invariant linear dynamical system. For more general non-stationary time series, the underlying dynamical structure may vary over time. In this case, the system dynamics matrix can be written as A_t, and the corresponding K_H becomes

$$\Sigma^{1/2}\left[\sum_{h=1}^{H}\left(\prod_{r=0}^{h-1}A_{T+h-1-r}\right)^{\top}cc^{\top}\left(\prod_{r=0}^{h-1}A_{T+h-1-r}\right)\right]\Sigma^{1/2}. \tag{82}$$

Therefore, the optimal predictive representation depends on both the prediction horizon and the local temporal context. This provides a theoretical motivation for introducing the horizon-conditioned representation translator in HorizonBridge.

E Additional Experiments

Ablation Study on Multi-horizon Training. We compare HorizonBridge with two commonly used strategies for multi-horizon forecasting: autoregressive forecasting and longest-horizon prefix extraction. For the autoregressive strategy, we follow the next-patch-token prediction paradigm adopted by recent time-series foundation models (Liu et al., 2025; Das et al., 2024). For the prefix extraction strategy, the model is trained to forecast the longest horizon ($H = 720$), and shorter-horizon results are obtained by extracting the corresponding prefixes from the output sequence. As shown in Table 5, the autoregressive strategy achieves relatively competitive performance on short horizons but suffers from apparent degradation on longer horizons, which may be attributed to the accumulation of prediction errors during recursive generation. The prefix strategy performs better on longer horizons but becomes less effective when the target horizon is shortened, as representations optimized for long-term forecasting may not match the requirements of short-term forecasting tasks. In contrast, HorizonBridge generally achieves better performance across different horizons, demonstrating the effectiveness of the strategy of jointly optimizing multiple horizons.

Generalization on Unseen Horizons. To further evaluate the generalization capability of HorizonBridge to unseen horizons during training, we conduct experiments on unseen forecasting horizons. Specifically, we train HorizonBridge on the standard horizons $\{96, 192, 336, 720\}$ and evaluate it on additional horizons $\{128, 256, 480\}$ without further training. The representation translator directly generates horizon-adapted representations based on the horizon condition. As shown in Fig. 6, MSE and MAE exhibit a smooth increasing trend as the forecasting horizon becomes longer, which is consistent with the increasing difficulty of long-term forecasting. Although some unseen horizons exhibit slight upward bends in the error plots, the overall trajectories remain smooth and follow the expected trend as the forecasting horizon increases. These results indicate that the representation translator can provide effective adaptation to unseen forecasting lengths.

Efficiency Analysis. We evaluate the computational efficiency of HorizonBridge by comparing peak GPU memory consumption, training time, and forecasting performance (MSE) with the following forecasters: VPNet (Liu et al., 2026), TQNet (Lin et al., 2025), Leddam (Yu et al., 2024), ModernTCN (Luo & Wang, 2024), iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023), TimesNet (Wu et al., 2023) and DLinear (Zeng et al., 2023). For a fair comparison, we adopt

Table 5: Ablation study on multi-horizon forecasting strategies. Prefix denotes training with the longest horizon and obtaining shorter predictions by extracting the corresponding prefixes. AR denotes the autoregressive forecasting strategy. The best results are highlighted in **red**.

Variant		HorizonBridge		Prefix		AR	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.347**	**0.385**	0.385	0.415	0.365	0.395
	192	**0.383**	**0.408**	0.402	0.427	0.409	0.422
	336	**0.403**	**0.423**	0.413	0.436	0.443	0.442
	720	**0.428**	**0.453**	0.451	0.471	0.497	0.471
ETTm1	96	**0.277**	**0.330**	0.303	0.354	0.295	0.334
	192	**0.333**	0.366	0.333	0.371	0.334	**0.358**
	336	**0.360**	**0.382**	0.361	0.388	0.362	0.384
	720	**0.410**	0.424	0.412	**0.415**	0.419	0.416
Solar	96	**0.153**	0.193	0.162	0.209	0.159	**0.190**
	192	**0.175**	**0.210**	0.177	0.219	0.183	0.211
	336	**0.190**	**0.221**	0.191	0.226	0.203	0.225
	720	0.204	**0.231**	**0.200**	0.234	0.236	0.250
Weather	96	**0.141**	**0.180**	0.145	0.192	0.152	0.197
	192	**0.181**	**0.223**	0.188	0.232	0.207	0.244
	336	**0.231**	**0.264**	0.239	0.273	0.278	0.290
	720	**0.306**	**0.319**	0.313	0.327	0.388	0.354

the official implementations of all baselines and follow their recommended experimental settings. All models are evaluated under a unified look-back window of 720 and a batch size of 64. For HorizonBridge, the two training stages are separately reported as two points in the figure, corresponding to the base encoder pre-training stage and the representation translator optimization stage. As shown in Fig. 7, HorizonBridge achieves efficiency only inferior to the linear model DLinear. However, DLinear consists of only two linear layers and lacks the capacity to capture complex temporal dependencies. TQNet and iTransformer also exhibit favorable efficiency, benefiting from their variable-level tokenization strategies that significantly reduce the number of tokens. Nevertheless, such coarse-grained representations may limit their ability to model non-uniform temporal patterns, resulting in inferior forecasting performance compared with HorizonBridge, which leverages adaptive patch embedding. These results demonstrate that HorizonBridge achieves a favorable trade-off between forecasting accuracy and computational efficiency.

F Full Results

Full Results of Cross-horizon Forecasting. Table 6 reports the complete cross-horizon forecasting results across all datasets and forecasting horizons, with the look-back window fixed to 720. Notably, HorizonBridge only trains a single base encoder on the base horizon and reuses it across all target horizons, whereas baseline models independently train separate models for each horizon. Despite this more challenging setting, HorizonBridge delivers consistently competitive performance across diverse datasets, achieving the best results in 53 out of 64 evaluation settings. These results highlight the effectiveness of horizon-conditioned representation translation in enabling flexible adaptation to diverse forecasting requirements.

We conduct two-sided paired Wilcoxon signed-rank tests over all dataset–horizon settings to evaluate the statistical significance of the performance improvements. As shown in Table 7, HorizonBridge significantly outperforms both PhaseFormer and VPNet in terms of MSE and MAE ($p < 0.001$), demonstrating the robustness of its gains.

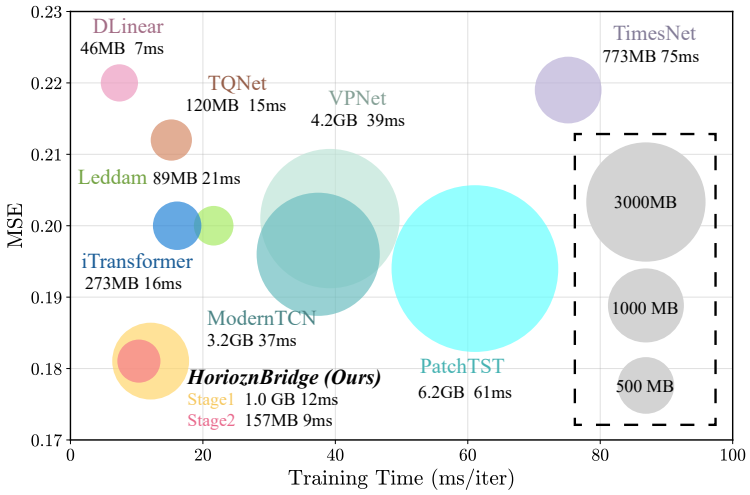


Figure 7: Efficiency analysis on the Weather dataset with $H = 192$. The x-axis, y-axis, and bubble size represent training time, forecasting error (MSE), and peak memory usage, respectively. Larger bubbles indicate higher GPU memory consumption.

Full Results of Plug-and-Play Analysis. Table 8 and Table 9 report the complete performance and efficiency results of the plug-and-play analysis under the cross-horizon forecasting setting. For each backbone, the original model requires independent training for different forecasting horizons, while the variants equipped with the representation translator (+Translator) only require training the base encoder once and subsequently optimizing the translator to adapt to different horizons. These results demonstrate the plug-and-play capability of the proposed method.

Table 6: Full results of cross-horizon forecasting with horizon $H \in \{96, 192, 336, 720\}$ for HorizonBridge and all baselines. Count indicates the number of cases where the model achieves the best performance under the corresponding metric. The results of HorizonBridge are averaged over three runs with different random seeds. The best and second-best results are highlighted in **red** and underlined, respectively.

Models		**HorizonBridge (Ours)**		PhaseFormer (2026)		VPNet (2026)		MixLinear (2026)		TimeMixer++ (2025)		TQNet (2025)		iTransformer (2024)		PatchTST (2023)		DLinear (2023)		TimesNet (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.277**	**0.330**	0.293	0.344	0.292	0.337	0.333	0.375	0.310	0.334	0.296	0.349	0.300	0.353	0.293	0.343	0.299	0.343	0.338	0.375
	192	0.333	0.366	**0.323**	**0.361**	0.345	0.371	0.353	0.382	0.348	0.362	0.337	0.374	0.345	0.382	0.333	0.370	0.335	0.365	0.371	0.387
	336	0.360	0.382	**0.358**	0.381	0.404	0.400	0.391	0.407	0.376	0.391	0.369	0.393	0.374	0.398	0.369	**0.369**	0.369	0.386	0.410	0.411
	720	**0.410**	0.424	0.412	**0.410**	0.447	0.424	0.439	0.435	0.440	0.423	0.447	0.434	0.429	0.430	0.416	0.420	0.425	0.421	0.478	0.450
	Avg	**0.345**	0.376	0.346	**0.374**	0.372	0.383	0.379	0.400	0.369	0.378	0.362	0.388	0.362	0.391	0.353	0.382	0.357	0.379	0.400	0.406
ETTm2	96	**0.158**	**0.243**	0.163	0.256	0.162	0.245	0.170	0.261	0.170	0.245	0.169	0.257	0.175	0.266	0.166	0.256	0.167	0.260	0.187	0.267
	192	**0.217**	**0.284**	0.219	0.293	0.219	0.285	0.223	0.298	0.229	0.291	0.231	0.299	0.242	0.312	0.223	0.296	0.224	0.303	0.249	0.309
	336	0.270	0.320	**0.269**	0.326	0.270	**0.319**	0.275	0.332	0.303	0.343	0.282	0.337	0.282	0.340	0.274	0.329	0.281	0.342	0.321	0.351
	720	0.354	**0.373**	**0.351**	0.379	0.357	0.376	0.360	0.384	0.373	0.399	0.371	0.398	0.378	0.398	0.362	0.385	0.397	0.421	0.497	0.403
	Avg	**0.250**	**0.305**	0.251	0.314	0.252	0.306	0.257	0.319	0.269	0.320	0.263	0.323	0.269	0.329	0.256	0.317	0.267	0.331	0.291	0.333
ETTh1	96	**0.347**	0.385	0.359	**0.382**	0.381	0.402	0.351	0.383	0.376	0.403	0.379	0.404	0.386	0.405	0.370	0.400	0.375	0.399	0.384	0.402
	192	**0.383**	0.408	0.397	**0.404**	0.418	0.427	0.395	0.423	0.416	0.441	0.429	0.441	0.424	0.440	0.413	0.429	0.405	0.416	0.557	0.436
	336	**0.403**	**0.423**	0.425	0.424	0.439	0.438	0.411	0.434	0.430	0.434	0.454	0.455	0.449	0.460	0.422	0.440	0.439	0.443	0.491	0.469
	720	0.428	0.453	0.431	**0.450**	0.520	0.500	**0.423**	0.456	0.467	0.451	0.499	0.506	0.495	0.487	0.447	0.468	0.472	0.490	0.521	0.500
	Avg	**0.390**	0.417	0.403	**0.415**	0.440	0.442	0.395	0.424	0.422	0.432	0.440	0.452	0.439	0.448	0.413	0.434	0.423	0.437	0.458	0.450
ETTh2	96	**0.265**	**0.325**	0.275	0.338	0.275	0.335	0.283	0.340	0.276	0.328	0.288	0.354	0.297	0.348	0.274	0.337	0.289	0.353	0.340	0.374
	192	0.334	**0.369**	0.341	0.376	0.351	0.382	0.337	0.376	0.342	0.379	0.377	0.403	0.371	0.403	**0.314**	0.382	0.383	0.418	0.402	0.414
	336	0.369	0.399	0.369	0.405	0.387	0.409	0.356	0.395	0.346	0.398	0.377	0.415	0.404	0.428	**0.329**	**0.384**	0.448	0.465	0.452	0.452
	720	0.390	0.424	0.402	0.436	0.420	0.440	0.380	0.423	0.392	**0.415**	0.424	0.452	0.424	0.444	**0.379**	0.422	0.605	0.551	0.462	0.468
	Avg	0.340	**0.379**	0.347	0.389	0.359	0.391	0.339	0.383	0.339	0.380	0.367	0.406	0.374	0.406	**0.324**	0.381	0.431	0.447	0.414	0.427
Weather	96	**0.141**	**0.180**	0.148	0.195	0.144	0.181	0.171	0.226	0.155	0.205	0.160	0.213	0.159	0.208	0.149	0.198	0.152	0.237	0.172	0.220
	192	**0.181**	**0.223**	0.193	0.237	0.201	0.237	0.212	0.260	0.201	0.245	0.212	0.261	0.200	0.248	0.194	0.241	0.220	0.282	0.219	0.261
	336	**0.231**	**0.264**	0.242	0.278	0.249	0.276	0.257	0.294	0.237	0.265	0.260	0.299	0.253	0.289	0.245	0.282	0.265	0.319	0.280	0.306
	720	**0.306**	**0.319**	0.309	0.332	0.347	0.343	0.324	0.344	0.312	0.334	0.328	0.343	0.321	0.338	0.314	0.334	0.323	0.362	0.365	0.359
	Avg	**0.215**	**0.246**	0.223	0.261	0.235	0.259	0.241	0.281	0.226	0.262	0.240	0.279	0.233	0.271	0.226	0.264	0.240	0.300	0.259	0.287
Electricity	96	**0.125**	**0.221**	0.129	0.221	0.132	0.225	0.139	0.233	0.135	0.222	0.143	0.244	0.138	0.237	0.129	0.222	0.140	0.237	0.168	0.272
	192	**0.143**	**0.234**	0.148	0.238	0.154	0.248	0.154	0.248	0.147	0.235	0.151	0.247	0.157	0.256	0.147	0.240	0.152	0.249	0.184	0.289
	336	**0.157**	0.251	0.165	0.257	0.169	0.264	0.171	0.265	0.164	**0.245**	0.166	0.261	0.167	0.264	0.163	0.259	0.169	0.267	0.198	0.300
	720	**0.186**	**0.280**	0.201	0.285	0.189	0.284	0.209	0.297	0.212	0.310	0.194	0.291	0.194	0.286	0.197	0.290	0.203	0.301	0.220	0.320
	Avg	**0.153**	**0.246**	0.161	0.250	0.161	0.255	0.168	0.261	0.165	0.253	0.164	0.261	0.164	0.261	0.159	0.253	0.166	0.264	0.192	0.295
Traffic	96	**0.336**	**0.235**	0.361	0.238	0.361	0.247	0.390	0.268	0.392	0.253	0.398	0.297	0.363	0.265	0.360	0.249	0.410	0.282	0.593	0.321
	192	**0.356**	**0.234**	0.373	0.243	0.373	0.263	0.404	0.276	0.402	0.258	0.397	0.277	0.385	0.273	0.379	0.256	0.423	0.287	0.617	0.336
	336	**0.380**	**0.246**	0.385	0.248	0.385	0.264	0.416	0.282	0.428	0.263	0.403	0.279	0.396	0.277	0.392	0.264	0.436	0.296	0.629	0.336
	720	**0.416**	**0.265**	0.428	0.270	0.428	0.287	0.452	0.301	0.441	0.282	0.448	0.304	0.445	0.312	0.432	0.286	0.466	0.315	0.640	0.350
	Avg	**0.372**	**0.245**	0.387	0.250	0.387	0.265	0.415	0.282	0.416	0.264	0.412	0.289	0.397	0.282	0.391	0.264	0.434	0.295	0.620	0.336
Solar	96	**0.153**	**0.193**	0.177	0.229	0.186	0.199	0.211	0.256	0.171	0.231	0.181	0.242	0.188	0.242	0.178	0.229	0.216	0.287	0.219	0.314
	192	0.175	0.210	0.194	0.237	**0.174**	**0.205**	0.227	0.261	0.218	0.263	0.203	0.261	0.193	0.258	0.189	0.246	0.244	0.305	0.231	0.322
	336	**0.190**	0.221	0.202	0.245	0.213	**0.221**	0.241	0.270	0.212	0.269	0.219	0.272	0.195	0.259	0.198	0.249	0.263	0.319	0.246	0.337
	720	0.204	0.231	**0.203**	0.249	0.226	**0.225**	0.240	0.265	0.212	0.270	0.231	0.281	0.223	0.281	0.209	0.256	0.264	0.324	0.280	0.363
	Avg	**0.180**	0.214	0.194	0.240	0.200	**0.213**	0.229	0.263	0.203	0.258	0.209	0.264	0.200	0.260	0.194	0.245	0.247	0.309	0.244	0.334
1st Count		29	24	5	7	1	5	1	0	0	2	0	0	0	0	4	2	0	0	0	0

Table 7: Statistical significance test between HorizonBridge and baseline models using the two-sided paired Wilcoxon signed-rank test.

Comparison	Metric	Wilcoxon Statistic	p-value	Significant
HorizonBridge vs PhaseFormer	MSE	38.0	3.1×10^{-6}	Yes
HorizonBridge vs VPNet	MSE	3.0	2.3×10^{-9}	Yes
HorizonBridge vs PhaseFormer	MAE	62.0	5.4×10^{-5}	Yes
HorizonBridge vs VPNet	MAE	29.0	8.1×10^{-7}	Yes

Table 8: Full forecasting performance for the plug-and-play experiments.

Models		PatchTST				Crossformer				TimeXer			
		Original		+Translator		Original		+Translator		Original		+Translator	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Solar	96	0.170	0.232	0.170	0.232	0.161	0.220	0.161	0.220	0.167	0.246	0.167	0.246
	192	0.190	0.253	0.193	0.263	0.203	0.244	0.188	0.244	0.183	0.257	0.183	0.254
	336	0.203	0.272	0.204	0.270	0.199	0.237	0.200	0.251	0.200	0.276	0.196	0.264
	720	0.208	0.268	0.215	0.277	0.276	0.334	0.215	0.280	0.199	0.272	0.207	0.276
	Avg.	0.193	0.256	0.196	0.261	0.210	0.259	0.191	0.249	0.187	0.263	0.188	0.260
Weather	96	0.146	0.197	0.146	0.197	0.147	0.215	0.147	0.215	0.150	0.203	0.150	0.203
	192	0.190	0.240	0.189	0.239	0.214	0.288	0.189	0.259	0.208	0.257	0.194	0.246
	336	0.244	0.286	0.237	0.278	0.261	0.326	0.235	0.297	0.251	0.293	0.245	0.286
	720	0.305	0.328	0.302	0.326	0.332	0.379	0.295	0.343	0.322	0.341	0.321	0.339
	Avg.	0.221	0.263	0.219	0.260	0.239	0.302	0.216	0.279	0.233	0.273	0.227	0.268
Traffic	96	0.391	0.286	0.391	0.286	0.512	0.283	0.512	0.283	0.366	0.257	0.366	0.257
	192	0.390	0.263	0.384	0.265	0.541	0.316	0.542	0.311	0.388	0.276	0.384	0.270
	336	0.421	0.303	0.397	0.272	0.553	0.305	0.555	0.316	0.402	0.280	0.399	0.277
	720	0.458	0.318	0.432	0.290	0.648	0.341	0.588	0.332	0.445	0.303	0.437	0.298
	Avg.	0.415	0.292	0.401	0.278	0.564	0.311	0.550	0.310	0.400	0.279	0.397	0.276

Table 9: Full efficiency results for the plug-and-play experiments. Params denotes trainable parameters in millions (M) and Time denotes training time per iteration in milliseconds (ms).

Models		PatchTST				Crossformer				TimeXer			
		Original		+Translator		Original		+Translator		Original		+Translator	
Metric		Params	Time	Params	Time	Params	Time	Params	Time	Params	Time	Params	Time
Solar	96	1.5M	239.2ms	1.5M	239.2ms	3.6M	93.1ms	3.6M	93.1ms	2.1M	62.3ms	2.1M	62.3ms
	192	2.6M	241.2ms			3.8M	148.3ms			3.3M	63.1ms		
	336	4.3M	244.5ms	8.5M	135.3ms	4.1M	136.4ms	0.9M	19.0ms	5.0M	65.6ms	3.5M	73.6ms
	720	8.7M	251.8ms			4.8M	151.8ms			9.5M	71.4ms		
	Sum	17.1M	976.7ms	10.0M	374.5ms	16.2M	529.6ms	4.5M	112.1ms	19.9M	262.4ms	5.6M	135.9ms
Weather	96	1.5M	50.8ms	1.5M	50.8ms	3.1M	44.7ms	3.1M	44.7ms	2.1M	22.0ms	2.1M	22.0ms
	192	2.6M	52.3ms			3.2M	46.1ms			2.4M	28.7ms		
	336	4.3M	48.0ms	8.6M	30.8ms	3.4M	56.1ms	1.0M	15.5ms	4.9M	25.5ms	3.6M	21.1ms
	720	8.7M	49.6ms			3.9M	76.5ms			4.6M	29.3ms		
	Sum	17.1M	200.8ms	10.1M	81.6ms	13.6M	223.4ms	4.1M	60.2ms	14.1M	105.5ms	5.7M	43.1ms
Traffic	96	1.5M	178.2ms	1.5M	178.2ms	2.7M	72.7ms	2.7M	72.7ms	11.0M	219.5ms	11.0M	219.5ms
	192	2.6M	194.4ms			3.0M	93.3ms			13.2M	211.2ms		
	336	4.3M	226.9ms	8.6M	106.4ms	3.4M	115.9ms	0.4M	26.3ms	14.0M	256.4ms	8.0M	131.0ms
	720	8.7M	228.6ms			4.5M	130.3ms			23.0M	272.8ms		
	Sum	17.1M	828.2ms	10.1M	284.6ms	13.5M	412.1ms	3.1M	99.0ms	61.2M	960.0ms	19.0M	350.6ms