LATENTTIME: EFFICIENT LONG-CONTEXT TIME SERIES FOUNDATION MODELS VIA LATENT FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) have emerged as a promising approach to general-purpose forecasting. As public pre-training corpora expand, recent TSFMs increasingly combine larger model capacities with longer input contexts. However, fine-grained patching and padding of variable-length sequences introduce substantial computational overhead. To solve this problem, we propose LatentTime, an efficient long-context forecasting framework that combines continuous latent forecasting with a Length-Aware Bucket strategy. Its lightweight TS-AE encoder compresses long input contexts by $2\times$, enabling forecasting in the resulting latent space. The decoder then performs $2\times$ temporal super-resolution on the latent predictions to restore the original resolution. Together, these operations approximately halve input tokens and double the forecast span per prediction block. The Length-Aware Bucket strategy further reduces padding overhead and controls training exposure across context lengths. Experiments on the GIFT-Eval benchmark demonstrates that LatentTime achieves forecasting accuracy comparable to that of raw-space PatchTST with lower forward FLOPs, while TS-AE maintains high reconstruction fidelity for long contexts across diverse time series.

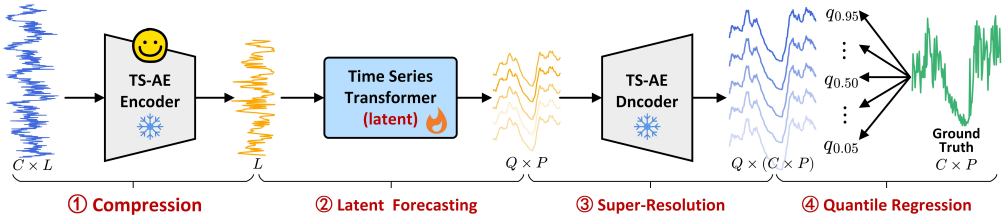


Figure 1: Overview of LatentTime: input context compression for latent forecasting and super-resolution of latent predictions. $C = 2$ is the compression factor, and Q is the number of quantiles.

1 INTRODUCTION

Time series analysis (TSA) is widely used in weather forecasting, healthcare, macroeconomics, and industrial monitoring, where accurate forecasting supports decision-making and risk management (Yi et al., 2025; Chang et al., 2026). Recent research in TSA has increasingly focused on task-specific models to time series foundation models (TSFMs) pre-trained on large-scale time series corpora (Woo et al., 2024; Auer et al., 2025; Fu et al., 2026). These models can generalize to unseen datasets and perform zero-shot time series forecasting (TSF).

As public time series pre-training corpora grow in scale and diversity (Shi et al., 2025), TSFMs are being scaled up to pursue better forecasting performance. Alongside this growth in model capacity, TSFMs adopt longer input contexts to make fuller use of historical information (Yao et al., 2025). For example, Chronos-2 and Timer-S1 support post-training context lengths of approximately 8.2K and 11.5K, respectively (Ansari et al., 2025; Liu et al., 2026c). Many of these models adopt Transformer architectures and tokenize raw time series through patching (Wen et al., 2026), typically using small patches, such as $P = 16$, to capture local temporal patterns. An input of length L therefore produces $n = \lceil L/P \rceil$ tokens. Since self-attention has quadratic complexity in n, processing

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

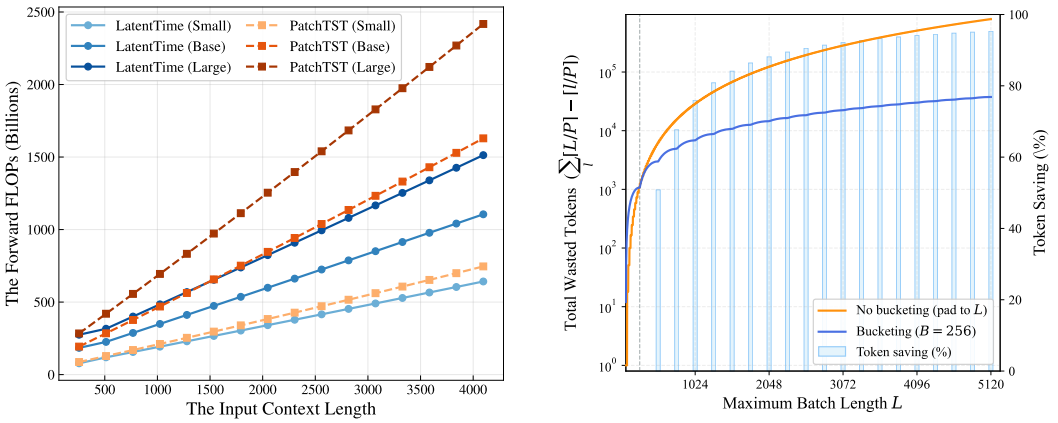


Figure 2: Computational motivation of LatentTime. **(Left)** Forward FLOPs of **PatchTST** and **LatentTime** with Small (21M), Base (47M), and Large (68M) with 64 batch size in Table 3. The parameter counts refer to the PatchTST, while LatentTime additionally includes a 1.8M TS-AE. We count all the operations. The results show that computational savings increase with predictor size and context length. **(Right)** Wasted tokens under global padding and length-aware bucket strategy with bucket width $B = 256$. We present the quantitative analysis for bucketing in Appendix A.

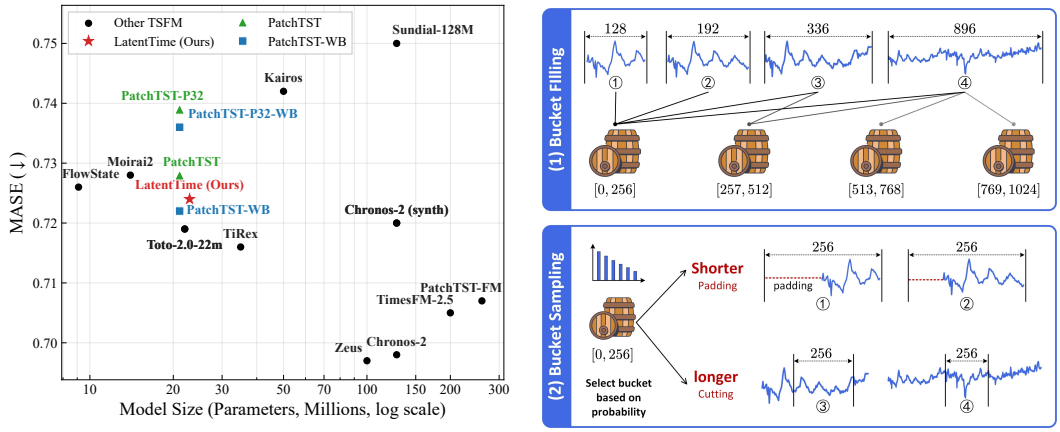


Figure 3: Performance and bucketing strategy of LatentTime. **(Left)** GIFT-Eval (Aksu et al., 2024) benchmark results measured by MASE. **(Right)** Length-Aware Bucket strategy. Sequences are first assigned to buckets by length, and then padded or cut as needed based on their length.

long contexts incurs substantial computational overhead. Figure 2 (left) illustrates the increasing forward FLOPs of channel-independent PatchTST (Nie et al., 2023) as context length grows.

Furthermore, most TSFMs left-pad each sequence to the maximum length within a batch to support variable-length inputs and parallel computation (Graf et al., 2026; Liu et al., 2026a). When sequence lengths differ substantially, this introduces considerable computation on padding, as illustrated in Figure 2 (right). Beyond reducing this overhead, variable-length training should provide adequate exposure to both short and long contexts. This motivates a batching strategy that reduces padding while enabling explicit control over training exposure across different context-length ranges.

To address the computational costs of long contexts and variable-length batching, we propose **LatentTime**, a framework for efficient long-context TSF. Inspired by latent modeling in audio, image, and video generation (Fu et al., 2025; Rombach et al., 2022; Zheng et al., 2024; Sun et al., 2024), LatentTime performs forecasting in a temporally compressed latent space to reduce sequence-processing costs. It further designs a **length-aware bucketing strategy** to reduce padding overhead and control the frequency with which different context-length ranges are sampled during training.

As illustrated in Figure 1, LatentTime employs TS-AE, a lightweight, channel-independent convolutional autoencoder (AE) with 1.8M parameters and linear computational complexity Chen et al. (2025a). Its encoder compresses long contexts by $2\times$, approximately halving the input token count at a fixed patch size. Its decoder performs $2\times$ temporal super-resolution, upsampling predicted latent sequences into forecasts at the original temporal resolution. Consequently, each latent prediction block covers twice as many original time steps, reducing the autoregressive iterations required for long-horizon forecasting. As shown in Figure 3 (left), our experimental results on the Gift-Eval benchmark show that LatentTime via latent forecasting still maintains good performance.

The length-aware bucket strategy in Figure 3 (right) groups sequences into predefined length ranges and pads them only to their corresponding bucket boundaries, substantially reducing redundant tokens. During training, bucket sampling probabilities explicitly control the frequency with which different context-length ranges are encountered, allowing training exposure to short and long contexts to be adjusted as needed. During inference, we process non-empty buckets sequentially. When sufficient computational resources are available, different buckets can be processed in parallel.

Our main contributions are as follows:

- We propose **LatentTime**, which uses a lightweight 1.8M-parameter TS-AE for $2\times$ context compression and $2\times$ temporal super-resolution of latent forecasts, reducing input tokens and extending the forecast span per forward pass.
- We introduce the **Length-Aware Bucket strategy** to reduce padding overhead and explicitly control training exposure across context-length ranges through bucket sampling. This strategy also better complements TS-AE in handling variable-length time series.
- Experiments on **GIFT-Eval** benchmark demonstrate the forecasting accuracy comparable to raw-space PatchTST with lower computational cost. Further analyses evaluate representation fidelity of TS-AE and the effects of bucket sampling and training parallelism.

2 Related Work

Large-scale time series foundation model. Analogous to large language models trained on massive text corpora, we refer to large-scale models pre-trained on vast time series datasets for zero-shot forecasting as TSFMs (Ansari et al., 2024; Wang et al., 2025a; Liu et al., 2026b;c). Early work from Salesforce introduced LOTSA, the largest time series pre-training dataset, and used it to train the first TSFM Moirai for universal forecasting (Woo et al., 2024). Recently, larger pre-training datasets and a growing number of TSFMs have emerged, including Sundial based on flow matching (Liu et al., 2025), TiRex with xLSTM architecture (Auer et al., 2025; Podest et al., 2026), Chronos-2 based on autoregressive Transformer (Ansari et al., 2025), PatchTST-FM (Wen et al., 2026) and Toto 2.0 (Khwaja et al., 2026b) based on contiguous patch masking, as well as multi-agent frameworks such as TSorchestra (Cao et al., 2025) and TimeRouter (Ning et al., 2026). However, while these works increasingly rely on scaling pre-training data, model size, or context length for better forecasting performance, they have not adequately addressed the high computational cost of excessively long contexts and the substantial token waste caused by variable-length padding.

Latent representations for time series forecasting. Latent diffusion models reduce the cost of image generation by operating on compressed AE representations (Rombach et al., 2022). Related approaches have also been explored in time series forecasting Tao et al. (2026). TLAE learns low-dimensional multivariate representations and temporal evolution for probabilistic forecasting (Nguyen & Quanz, 2021), while LDT performs conditional diffusion in a compressed latent space (Feng et al., 2024). LatentTSF maps observations at each time step into a higher-dimensional latent to encourage temporal coherence (Yang et al., 2026), and Re-Diffusion investigates diffusion-based latent residual modeling (Zhang et al., 2026a). Another line of work uses discrete representations: TOTEM employs a channel-independent VQ-VAE to compress temporal contexts for downstream forecasting (Talukder et al., 2024), while UniTok-FM performs autoregressive prediction using a causal tokenizer with finite scalar quantization (Zhang et al., 2026b). Unlike diffusion-based latent models and discrete tokenizers, LatentTime uses deterministic continuous representations to shorten the temporal sequence processed by the forecasting backbone. Its lightweight deterministic TS-AE provides $2\times$ context compression and learned $2\times$ temporal upsampling of latent forecasts, complemented by the Length-Aware Bucket strategy for efficient variable-length training.

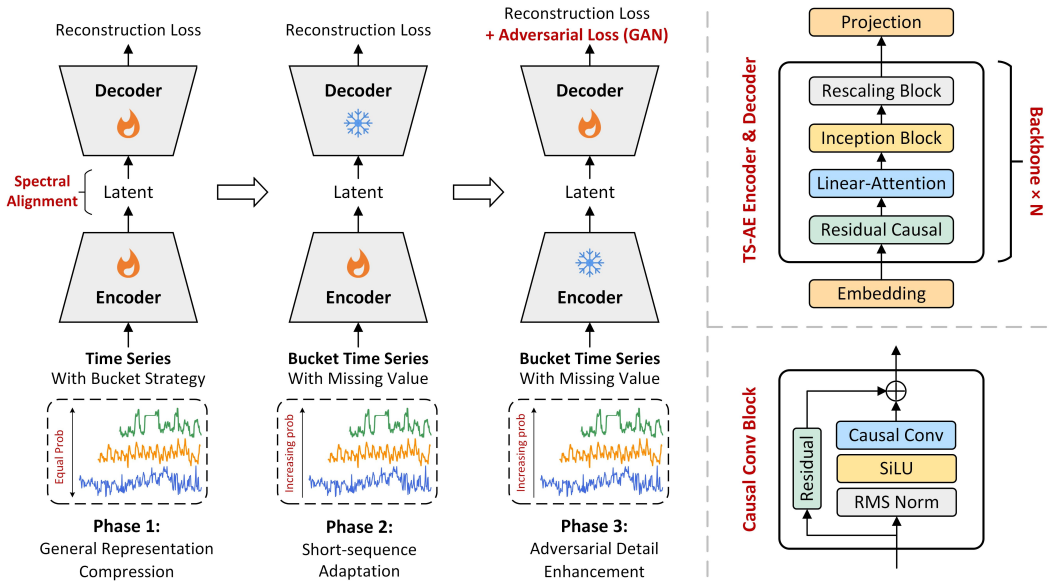


Figure 4: **(Left)** Three-stage pre-training of TS-AE. **(Right)** The encoder–decoder backbone.

3 METHODS

3.1 A LIGHTWEIGHT $\mathcal{O}(L)$ AUTOENCODER WITH LENGTH-AWARE BUCKET STRATEGY

Architecture. We develop TS-AE, a lightweight deterministic autoencoder with 1.8M parameters to provide continuous compressed representations. TS-AE processes each channel independently and takes normalized values, temporal encodings, and observation masks as input (Ansari et al., 2025). As shown in Figure 4, its symmetric encoder–decoder architecture combines: (1) causal convolutions for local temporal dependencies; (2) linear attention for global context (Cai et al., 2023); (3) Inception modules for multi-scale features (Ismail Fawaz et al., 2020); and (4) rescaling blocks for $2\times$ compression and upsampling. With a fixed architecture, both encoding and decoding have $\mathcal{O}(L)$ complexity in sequence length L. We provide more details of TS-AE in Appendix B.1.

Bucket construction and sampling. As shown in Figure 3 (right), inspired by bucketed training in OpenSora (Zheng et al., 2024), we group variable-length training windows into predefined length buckets. Each process or GPU independently samples a bucket and constructs a local batch padded to its upper boundary, while long records can supply cropped windows to multiple eligible buckets. This reduces padding and unnecessary convolutional computation, which cannot be eliminated by masking the reconstruction loss alone. Moreover, bucketing also controls exposure to different context lengths: we sample buckets uniformly in Stage 1 and favor shorter buckets in Stages 2 and 3 to support short-context adaptation during the pre-training of TS-AE in Figure 4. The sampling probabilities are defined in Appendix A.3. During inference, each sequence is assigned to the smallest bucket covering its context length, and non-empty buckets are processed separately. For TS-AE pre-training, we use buckets $[64, 256]$, $[257, 512]$, ..., and $[6401, 6656]$, with upper boundaries increasing in 256-step increments up to the maximum training context length of 6656.

3.2 THREE-STAGE PRE-TRAINING OF TS-AE

To promote both latent representation fidelity and accurate reconstruction, we adopt three-stage pre-training, as illustrated in Figure 4 (left). First, we jointly optimize the encoder and decoder with uniform bucket sampling. Second, we freeze the decoder and adapt the encoder through increased sampling of shorter buckets and masked reconstruction. Finally, we freeze the encoder and refine the decoder using an additional adversarial objective from a compact InceptionTime discriminator (Ismail Fawaz et al., 2020), retaining the emphasis on short sequences and masking augmentation. Originally missing observations are excluded during data preparation; random point masks and con-

tiguous patch masks are introduced artificially in the latter two stages to improve robustness to missing inputs. We provide more details for the three-stage pre-training in Appendix B.2.

Reconstruction losses. Let $\mathbf{z}_i = E_\theta(\tilde{\mathbf{x}}_i)$ and $\hat{\mathbf{x}}_i = D_\phi(\mathbf{z}_i)$ denote the latent representation and reconstruction of a possibly masked input $\tilde{\mathbf{x}}_i$, with clean target $\mathbf{x}_i$. For a batch of N windows, the target mask $\mathbf{m}_i$ excludes padding while retaining artificially masked observations, and $n_i = \sum_t m_{i,t}$. We use masked MSE and the empirical Wasserstein-1 distance (WD) (Chen et al., 2017):

$$\mathcal{L}_{\mathrm{MSE}} = \frac{\sum_{i=1}^{N}\sum_t m_{i,t}(x_{i,t}-\hat{x}_{i,t})^2}{\sum_{i=1}^{N} n_i}, \quad \mathcal{L}_{\mathrm{WD}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}\left|x_{i,(j)}-\hat{x}_{i,(j)}\right|. \tag{1}$$

Parenthesized indices denote independently sorted valid values. MSE preserves pointwise temporal alignment, while WD aligns empirical value distributions through sorting. Their combination defines $\mathcal{L}_{\mathrm{rec}} = \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{WD}}\mathcal{L}_{\mathrm{WD}}$.

Latent spectral alignment. To encourage preservation of periodic structure, we align the seasonal power spectra of the latent representation and the clean input downsampled to the same temporal resolution. Let A_2 denote $2\times$ average pooling and $\mathcal{S}$ extract the seasonal component from a trend–seasonality decomposition. Define $\mathbf{p}_i^x = \mathrm{PSD}(\mathcal{S}(A_2(\mathbf{x}_i)))$ and $\mathbf{p}_i^z = \mathrm{PSD}(\mathcal{S}(\mathbf{z}_i))$. Decomposition and spectral estimation are performed on aligned non-padding regions using identical settings (O'Shea, 2000). Let $\mathcal{I}$ denote the eligible windows with sufficient valid length and non-negligible reference spectral energy, as specified in Appendix B.3. The alignment loss is

$$\mathcal{L}_{\mathrm{spec}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\left[1 - \frac{\langle \mathbf{p}_i^x, \mathbf{p}_i^z\rangle}{\max\left(\|\mathbf{p}_i^x\|_2\|\mathbf{p}_i^z\|_2, \epsilon\right)}\right], \tag{2}$$

where $\epsilon > 0$ ensures numerical stability. We set $\mathcal{L}_{\mathrm{spec}} = 0$ when $\mathcal{I}$ is empty.

Adversarial and overall objectives. In Stage 3, to optimize the details of decoder reconstruction, the decoder further minimizes the non-saturating adversarial loss: $\mathcal{L}_{\mathrm{adv}} = -\mathbb{E}_i\left[\log D_\psi(\hat{\mathbf{x}}_i; \mathbf{m}_i)\right]$, where D_ψ is the discriminator (Esser et al., 2021). The overall TS-AE objective at stage s is

$$\mathcal{L}^{(s)} = \begin{cases} \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{spec}}\mathcal{L}_{\mathrm{spec}}, & s \in \{1,2\}, \\ \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{adv}}\mathcal{L}_{\mathrm{adv}}, & s = 3. \end{cases} \tag{3}$$

We set $\lambda_{\mathrm{WD}} = 0.25$ and $\lambda_{\mathrm{adv}} = 0.075$, while $\lambda_{\mathrm{spec}} = 0.005$ controls latent spectral regularization. The spectral term is omitted in Stage 3 because the encoder is frozen. The discriminator is trained using binary cross-entropy with detached reconstructions; its complete objective is provided in Appendix B.4.

3.3 Probabilistic Forecasting in Latent Space

Predictor pre-training. After pre-training TS-AE, we train a channel-independent PatchTST (Nie et al., 2023) based on a standard Transformer encoder with rotary position embeddings (Su et al., 2024) in the original time domain. Probabilistic forecasts are learned using the quantile regression

$$\mathcal{L}_{\mathrm{quantile}} = \frac{1}{HQ}\sum_{t=1}^{H}\sum_{q\in\mathcal{Q}}\Big[q(y_t-\hat{y}_{t,q})_+ + (1-q)(\hat{y}_{t,q}-y_t)_+\Big], \tag{4}$$

where H is the forecast horizon and $\hat{y}_{t,q}$ is the quantile-q prediction. We use $\mathcal{Q}$=$\{0.01$,0.05, $0.1, 0.2, \ldots, 0.9, 0.95, 0.99\}$, with the median providing the point forecast. Following Chronos-2 (Ansari et al., 2025), we adopt short-context pre-training followed by long-context post-training, both using the Length-Aware Bucket strategy. Pre-training uses buckets $[16, 32]$, $[33, 64]$, $[65, 128]$, and $[129, 256]$, followed by buckets of width 256 up to length 1280. Post-training extends the bucket range to 4096 with the same 256-step spacing. With a patch size of 16, the predictor outputs up to 16 future patches per forward pass, corresponding to a maximum forecast length of 256 time steps.

Adaptation to latent forecasting. As illustrated in Figure 1, we initialize LatentTime with the pre-trained PatchTST weights and place the predictor between the frozen TS-AE encoder and decoder. Then, we freeze TS-AE and post-train the predictor using Equation equation 4, evaluated on decoded forecasts in the original time domain. This adapts the predictor to latent representations while

enabling both $2\times$ context compression and $2\times$ temporal upsampling of its latent forecasts. Quantile decoding and autoregressive forecasting details are provided in Appendix B.5.

Context-dependent forecasting. We activate latent forecasting only when every sequence processed together contains at least 512 valid context steps, excluding padding. Otherwise, the predictor operates directly in the original time domain. This avoids the lower reconstruction fidelity of short contexts, where compression also offers smaller computational benefits. After training, LatentTime performs zero-shot forecasting without fine-tuning on the target datasets. To distinguish the two modes, we insert a learnable token between the context tokens and prediction tokens: `[RAW]` for original-space forecasting and `[LAT]` for latent-space forecasting.

4 EXPERIMENTS

To evaluate the effectiveness of latent forecasting, the compression of TS-AE and the length-aware bucket strategy, we conduct experiments to address the following 5 research questions:

- **RQ1**: How do $2\times$ context compression with TS-AE and the Length-Aware Bucket strategy affect zero-shot generalization in time series forecasting?
- **RQ2**: Can TS-AE maintain reconstruction fidelity across domains and context lengths, and how do its pre-training stages and architectural choices affect reconstruction, robustness to missing inputs, and downstream forecasting?
- **RQ3**: How do TS-AE's latent representations differ from the original time series, and do they preserve temporal structure and predictive information more effectively than $2\times$ average pooling?
- **RQ4**: Does channel-independent compression with TS-AE preserve the inter-channel correlation structure of multivariate time series?
- **RQ5**: How do training parallelism and bucket sampling probabilities affect TSFM optimization and forecasting performance under the Length-Aware Bucket strategy?

The Pre-training of TS-AE and LatentTime. We pre-train TS-AE exclusively on the leakage-free GIFT-Eval pre-training corpus (Aksu et al., 2024) and synthetic time series generated using KernelSynth (Ansari et al., 2024), ForecastPFN (Dooley et al., 2023), and S2Generator (Wang et al., 2025a). For all prediction models, including the PatchTST variant and LatentTime, we added an additional GIFT-Eval training set to the original TS-AE pre-training dataset, with its validation set used for evaluation and model selection. The forecasting backbone is a channel-independent PatchTST with six Transformer encoder layers and approximately 21M parameters. Training configurations and datasets are detailed in Appendices C.1 and C.2, respectively.

4.1 **(RQ1)** FORECASTING PERFORMANCE OF LATENTTIME ON GIFT-EVAL

Setup. We assess the effects of latent compression and the length-aware bucket strategy on the GIFT-Eval test splits using MASE and CRPS defined in Appendix C.4. The baseline TSFMs are introduced in Appendix C.3. Then, we compare LatentTime with PatchTST and PatchTST-WB using patch size 16, and their patch-size-32 counterparts, PatchTST-P32 and PatchTST-P32-WB. WB denotes bucketed training. For contexts processed by TS-AE, the P32 variants provide approximately the same input token count through larger patches, allowing comparison with learned compression.

Results. Figure 5 shows that bucketing improves both metrics at either patch size. LatentTime achieves a MASE of 0.724 and CRPS of 0.488, outperforming standard PatchTST while remaining close to PatchTST-WB, with relative error increases of 0.28% and 0.62%. Compared with PatchTST-P32-WB, LatentTime reduces MASE by 1.63% and CRPS by 2.01%. These results support latent forecasting as an effective way to reduce long-context input tokens while largely retaining raw-space forecasting accuracy. We visualized the forecasting performance of LatentTime in Appendix E.3.

4.2 **(RQ2)** RECONSTRUCTION FIDELITY AND ABLATION STUDIES OF TS-AE

Setup. We evaluate TS-AE on GIFT-Eval using sliding windows with 50% overlap at lengths of 128, 256, 512, 1024, 2048, 3072, 4096, 6656, 7168, and 8192, aggregating results by context length

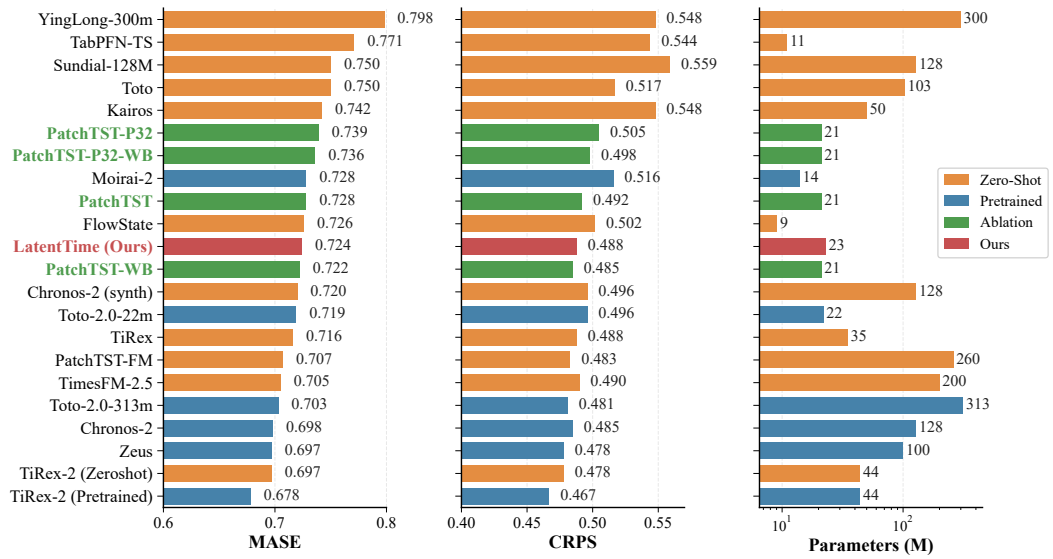


Figure 5: The forecasting results of Gift-Eval benchmark with full results shown in Appendix D.1.

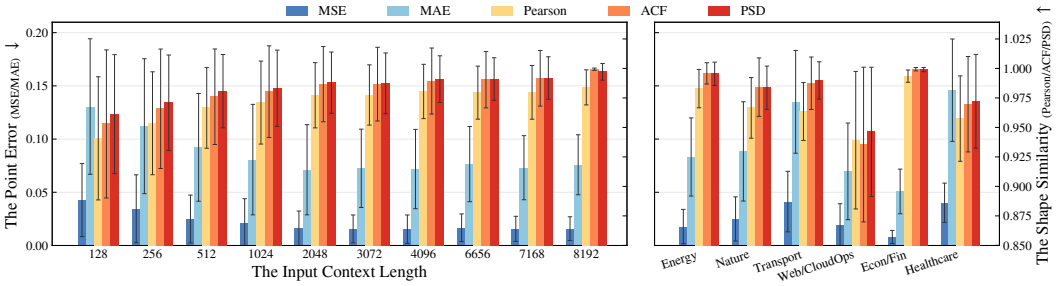


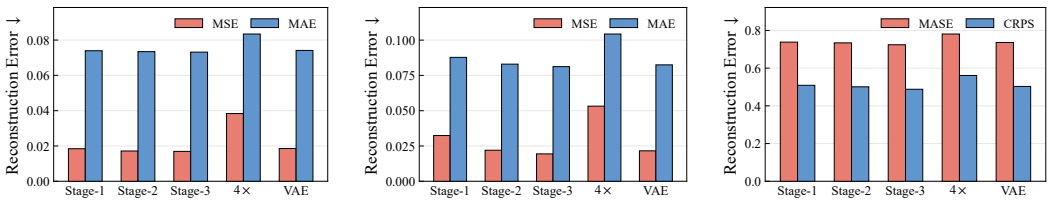
Figure 6: TS-AE reconstruction fidelity across context lengths (left) and domains (right) on GIFT-Eval. Error bars indicate standard deviations across multiple sub-datasets or context length.

and domain. The last two lengths exceed the maximum training context of 6656. MSE and MAE measure pointwise errors, while Pearson, ACF, and PSD similarities assess structural fidelity; definitions are provided in Appendix C.5. We further compare checkpoints from the three pre-training stages, a $4\times$ compression variant, and a VAE variant with KL regularization weighted by 10^{-6}. Reconstruction ablations use length-2048 contexts under complete inputs and inputs with 10% artificially masked observations, retaining clean targets. We also evaluate LatentTime forecasting with each autoencoder using MASE and CRPS.

Results. Figure 6 shows high reconstruction fidelity for long contexts, including evaluated lengths beyond the training range. Fidelity varies across domains and decreases for short contexts, motivating the bypass in LatentTime. Figure 7 shows progressive improvements in reconstruction, robustness to missing inputs, and downstream forecasting across pre-training stages. The final TS-AE outperforms both the higher-compression and VAE variants, supporting the three-stage training procedure and the $2\times$ deterministic design. We present the full results for reconstruction at the length of 2048 in Appendix D.2. Then, we explain why we used AE instead of VAE for context compression in Appendix C.6. The visualization of TS-AE reconstruction performance is in Appendix E.1.

4.3 **(RQ3)** Temporal Structure and Predictive Utility of Latent Space

Setup. Since LatentTime performs forecasting in the TS-AE latent space, we examine how its representations differ from original contexts and their $2\times$ average-pooled counterparts in representation structure and predictive utility. We first visualize length-512 windows and their corresponding



(a) Reconstruction (b) 10% Missing Inputs (c) Downstream Forecasting

Figure 7: Ablations of TS-AE pre-training stages and architecture. Reconstruction experiments use contexts of length 2048. The $4\times$ variant increases temporal compression, while VAE introduces KL regularization with weight 10^{-6}. Lower values indicate better performance for all reported metrics.

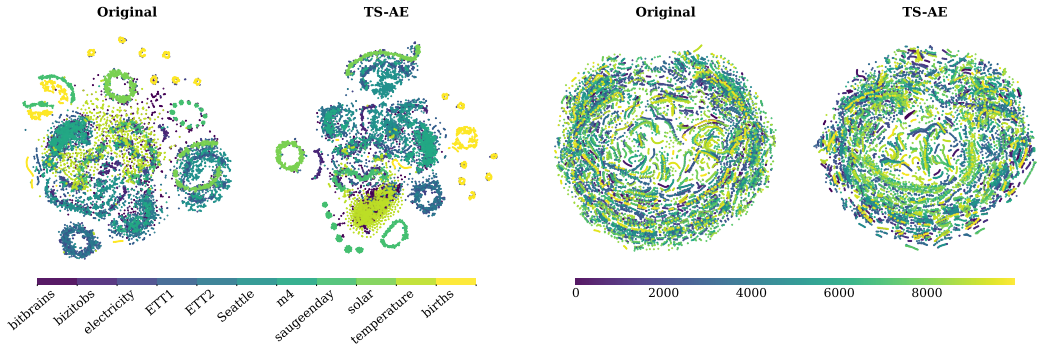


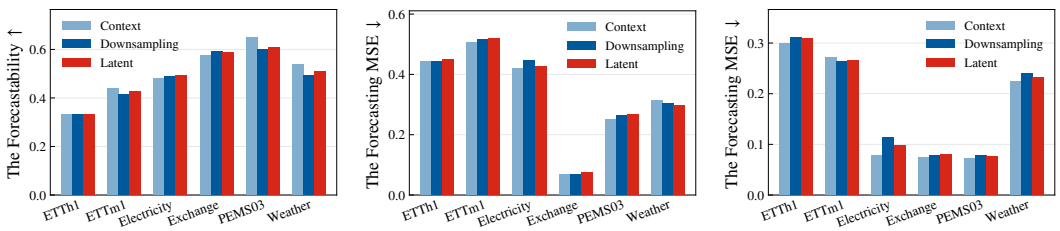
Figure 8: **(Left)** TSNE embeddings in different subsets of the Gift-Eval benchmark and **(Right)** in different timestamps of the Electricity dataset's OT channel with context length of 512 per samples.

latents using t-SNE from two perspectives: dataset-level organization across selected GIFT-Eval datasets and temporal continuity across windows sampled at different timestamps within an Electricity channel. Then, we quantitatively assess predictive utility on six multivariate datasets using spectral forecastability, defined as $\Omega = 1 - H$ with normalized spectral entropy H, and downstream forecasting with ARIMA and frozen Chronos-2-Synth (Ansari et al., 2025). All forecasts are evaluated against the same future targets at the original temporal resolution using MSE and MAE. Spectral estimation and the detailed evaluation protocol are provided in Appendices C.7 and C.8.

Results. Figure 8 shows broadly similar dataset-level organization and locally continuous timestamp-ordered trajectories in the original and latent spaces, providing qualitative evidence of structural preservation. The spectral evaluation in Figure 9(a) further shows higher latent forecastability than average pooling in 19 of the 24 dataset–length settings, with Exchange being a consistent exception. Forecasting results in Figure 9(b–c) reveal that the benefits depend on the predictor. With Chronos-2-Synth, latent forecasting yields lower MSE than average pooling across all evaluated context–horizon combinations on PEMS03, Electricity, and Weather. Improvements with ARIMA are less consistent, while forecasting from uncompressed contexts remains more accurate in several settings. These findings support the predictive utility of the compressed representations, with advantages over average pooling varying across datasets and forecasting models.

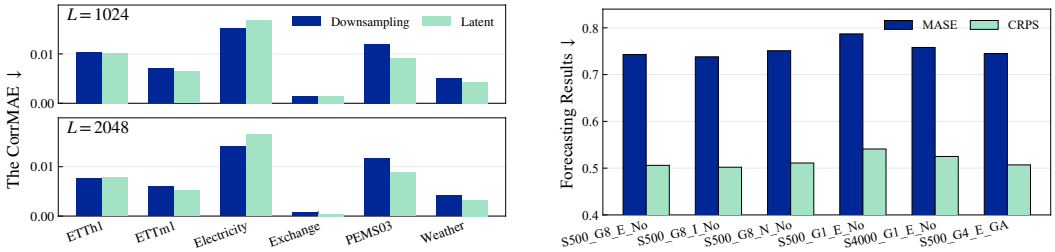
4.4 **(RQ4)** Preservation of Inter-Channel Correlations

Setup. Multivariate forecasting is an important direction for TSFMs. To accommodate variable channel counts with its fixed-channel convolutional backbone, TS-AE encodes each channel independently. We examine whether this compression preserves inter-channel correlations at context lengths of 1024 and 2048. We compare TS-AE latents and $2\times$ average-pooling against the original contexts using CorrMAE, which measures the mean absolute change in pairwise Pearson correlations. Lower values indicate better preservation; calculation details are provided in Appendix C.9.



(a) Forecastability (b) ARIMA (c) Chronos-2-Synth

Figure 9: Comparison of original contexts, $2\times$ average-pooled sequences, and TS-AE latent representations. From left to right: (a) spectral forecastability $\Omega = 1 - H$ (↑); (b) forecasting MSE with ARIMA (↓); (c) forecasting MSE with frozen Chronos-2-Synth (↓).



(a) Inter-channel correlation preservation (b) Training configuration comparison

Figure 10: (a) Inter-channel CorrMAE of average-pooled and TS-AE representations at context lengths of 1024 and 2048. (b) Forecasting MASE and CRPS after PatchTST short-context pre-training under six configurations, defined in Appendix C.10.

Results. Figure 10a shows low latent CorrMAE across all six datasets at both context lengths, generally comparable to average pooling. These results indicate that TS-AE largely preserves inter-channel linear correlation structure despite compressing each channel independently.

4.5 **(RQ5)** Training Parallelism and Bucket Sampling

Setup. We investigate GPU parallelism, gradient accumulation, and bucket sampling during PatchTST's short-context pre-training, before long-context post-training, with the training configuration in Appendix 4. Each bucketed microbatch contains contexts from one length range, making the composition of each optimizer update dependent on these settings. Using `S500_G8_E_No` as the reference, we compare six configurations and evaluate forecasting performance on GIFT-Eval. Configuration definitions and results analysis are provided in Appendix C.10.

Results. Figure 10b shows that `S500_G8_I_No` achieves the lowest errors, and both bucketed eight-GPU settings outperform `S500_G8_N_No`. The single-GPU configuration `S500_G1_E_No` performs worse; extending training to `S4000_G1_E_No` narrows but does not close the gap. Meanwhile, `S500_G4_E_GA` closely matches `S500_G8_E_No`, indicating that gradient accumulation can maintain comparable forecasting accuracy with fewer GPUs.

5 Conclusion

We presented LatentTime, a framework for efficient long-context time series forecasting using continuous latent representations. Its lightweight TS-AE combines $2\times$ context compression with $2\times$ temporal upsampling of latent forecasts, while the Length-Aware Bucket strategy reduces padding and controls exposure to different context lengths. Experiments on Gift-Eval benchmark demonstrate the accuracy comparable to raw-space PatchTST with lower forward FLOPs, while further analyses support latent representation fidelity and predictive utility. Short-context reconstruction remains a limitation, motivating our context-dependent forecasting strategy.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and proofreading, which is a task with recommended disclosure. We have not used generative AI tools for experimental design, experimental analysis, or any other task with required disclosure, and the remaining required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies efficient time series forecasting using existing public datasets and synthetic time series. It does not involve recruiting participants or collecting new personal data. Dataset sources and the training and evaluation protocol are described in Appendix C.2. Forecasting performance varies across domains and context lengths, and benchmark results alone do not establish suitability for safety-critical applications. Deployment in healthcare, finance, or industrial monitoring requires application-specific validation and appropriate human oversight.

REPRODUCIBILITY STATEMENT

The model architecture, training objectives, and forecasting procedure are described in Section 3. Additional TS-AE architecture details and model training configurations are provided in Appendices B.1 and C.1, respectively. Dataset sources, preprocessing, and data partitions are documented in Appendix C.2. Forecasting and reconstruction metrics are defined in Appendices C.4 and C.5. Detailed protocols for evaluating latent predictive utility, inter-channel correlations, and bucket training configurations are provided in Appendices C.8, C.9, and C.10.

REFERENCES

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024. URL https://openreview.net/forum?id=Z2cMOOANFX.

Robert Lund and. Time series analysis and its applications: With r examples. *Journal of the American Statistical Association*, 102(479):1079–1079, 2007. doi: 10.1198/jasa.2007.s209. URL https://doi.org/10.1198/jasa.2007.s209.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR. Expert Certification.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=v7UqniC9pF.

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17256–17267, 2023. doi: 10.1109/ICCV51070.2023.01587.

Defu Cao, Michael Gee, Jinbo Liu, Hengxuan Wang, Wei Yang, Rui Wang, and Yan Liu. Conversational time series foundation models: Towards explainable and effective forecasting. *arXiv preprint arXiv:2512.16022*, 2025.

Ching Chang, Yidan Shi, Defu Cao, Wei Yang, Jeehyun Hwang, Haixin Wang, Jiacheng Pang, Wei Wang, Yan Liu, Wen-Chih Peng, and Tien-Fu Chen. A survey of reasoning and agentic systems in time series with large language models. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=l3QW42g6u3. Survey Certification.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=wH8XXUOUZU.

Yongxin Chen, Tryphon T. Georgiou, Lipeng Ning, and Allen Tannenbaum. Matricial wasserstein-1 distance. *IEEE Control Systems Letters*, 1(1):14–19, 2017. doi: 10.1109/LCSYS.2017.2699319.

Youqiang Chen, Cheng Zhang, Ridong Zhang, and Furong Gao. Deep network model fusion of wide kernel feature learning for industrial process modeling and fault diagnosis. *Process Safety and Environmental Protection*, 194:1283–1302, 2025b. ISSN 0957-5820. doi: https://doi.org/10.1016/j.psep.2024.12.066. URL https://www.sciencedirect.com/science/article/pii/S0957582024016379.

Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability, 2024. URL https://arxiv.org/abs/2407.07874.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 2403–2426. Curran Associates, Inc., 2023. doi: 10.52202/075280-0112. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0731f0e65559059eb9cd9d6f44ce2dd8-Paper-Conference.pdf.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12873–12883, June 2021.

Kun Feng, Shaocheng Lan, Yuchen Fang, Wenchao He, Sihan Lu, Shuqi Gu, Lintao Ma, Xingyu Lu, and Kan Ren. Kairos: Toward adaptive and parameter-efficient time series foundation models, 2026. URL https://arxiv.org/abs/2509.25826.

Shibo Feng, Chunyan Miao, Zhong Zhang, and Peilin Zhao. Latent diffusion transformer for probabilistic time series forecasting. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i11.29085. URL https://doi.org/10.1609/aaai.v38i11.29085.

Yisong Fu, Zezhi Shao, Chengqing Yu, Yujie Li, Yongjun Xu, Xueqi Cheng, and Fei Wang. Zeus: Towards tuning-free foundation model for time series analysis. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Dtue4WLYSk.

Youquan Fu, Ruiyang Si, Hongfa Wang, Dongzhan Zhou, Jiacheng Sun, Ping Luo, Di Hu, Hongyuan Zhang, and Xuelong Li. Object-avedit: An object-level audio-visual editing model, 2025. URL https://arxiv.org/abs/2510.00050.

Georg Goerg. Forecastable component analysis. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 64–72, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/goerg13.html.

Lars Graf, Thomas Ortner, Stanisław Woźniak, and Angeliki Pantazi. Flowstate: Sampling-rate-equivariant time-series forecasting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9yNGBueKgZ.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending tabpfn-v2 to time series forecasting, 2026. URL https://arxiv.org/abs/2501.02945.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026a. URL https://arxiv.org/abs/2605.20119.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026b. URL https://arxiv.org/abs/2605.20119.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting, 2026a. URL https://arxiv.org/abs/2511.11698.

Yiding Liu, Yifan Hu, Hongjie Xia, Peiyuan Liu, Hongzhou Chen, Xilin Dai, Zewei Dong, and Jiang-Ming Yang. Falcon-x: A time series foundation model for heterogeneous multivariate modeling, 2026b. URL https://arxiv.org/abs/2605.27286.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LO7ciRpjI5.

Yong Liu, Xingjian Su, Shiyu Wang, Haoran Zhang, Haixuan Liu, Yuxuan Wang, Zhou Ye, Yang Xiang, Jianmin Wang, and Mingsheng Long. Timer-s1: A billion-scale time series foundation model with serial scaling, 2026c. URL https://arxiv.org/abs/2603.04791.

Nam Nguyen and Brian Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9117–9125, 2021. doi: 10.1609/aaai.v35i10.17101. URL https://doi.org/10.1609/aaai.v35i10.17101.

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

Kanghui Ning, Yushan Jiang, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. Timerouter: Efficient and adaptive routing of time-series foundation models, 2026. URL https://arxiv.org/abs/2606.11625.

P. O'Shea. The use of sliding spectral windows for parameter estimation in power system disturbance monitoring. *IEEE Transactions on Power Systems*, 15(4):1261–1267, 2000. doi: 10.1109/59.898099.

Patrick Podest, Marco Pichler, Elias Bürger, Levente Zólyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex-2: Generalizing tirex to multivariate data and streaming, 2026. URL https://arxiv.org/abs/2607.01204.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e1wDDFmlVu.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, Jiahao Bu, Zhongzhi Chen, Xuemeng Huang, Fengzong Lian, Saiyong Yang, Jianfeng Yan, Yuyuan Zeng, Xiaoqin Ren, Chao Yu, Lulu Wu, Yue Mao, Jun Xia, Tao Yang, Suncong Zheng, Kan Wu, Dian Jiao, Jinbao Xue, Xipeng Zhang, Decheng Wu, Kai Liu, Dengpeng Wu, Guanghui Xu, Shaohua Chen, Shuang Chen, Xiao Feng, Yigeng Hong, Junqiang Zheng, Chengcheng Xu, Zongwei Li, Xiong Kuang, Jianglu Hu, Yiqi Chen, Yuchi Deng, Guiyang Li, Ao Liu, Chenchen Zhang, Shihui Hu, Zilong Zhao, Zifan Wu, Yao Ding, Weichao Wang, Han Liu, Roberts Wang, Hao Fei, Peijie Yu, Ze Zhao, Xun Cao, Hai Wang, Fusheng Xiang, Mengyuan Huang, Zhiyuan Xiong, Bin Hu, Xuebin Hou, Lei Jiang, Jianqiang Ma, Jiajia Wu, Yaping Deng, Yi Shen, Qian Wang, Weijie Liu, Jie Liu, Meng Chen, Liang Dong, Weiwen Jia, Hu Chen, Feifei Liu, Rui Yuan, Huilin Xu, Zhenxiang Yan, Tengfei Cao, Zhichao Hu, Xinhua Feng, Dong Du, Tinghao Yu, Yangyu Tao, Feng Zhang, Jianchen Zhu, Chengzhong Xu, Xirui Li, Chong Zha, Wen Ouyang, Yinben Xia, Xiang Li, Zekun He, Rongpeng Chen, Jiawei Song, Ruibin Chen, Fan Jiang, Chongqing Zhao, Bo Wang, Hao Gong, Rong Gan, Winston Hu, Zhanhui Kang, Yong Yang, Yuhong Liu, Di Wang, and Jie Jiang. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent, 2024. URL https://arxiv.org/abs/2411.02265.

Sabera J Talukder, Yisong Yue, and Georgia Gkioxari. TOTEM: Tokenized time series embeddings for general time series analysis. In *ICLR 2024 Workshop on Learning from Time Series For Health*, 2024. URL https://openreview.net/forum?id=jzIdR2TXlK.

Xiaoyu Tao, Shilong Zhang, Mingyue Cheng, Daoyu Wang, Tingyue Pan, Bokai Pan, Changqing Zhang, and Shijin Wang. From values to tokens: An llm-driven framework for context-aware time series forecasting via symbolic discretization, 2026. URL https://arxiv.org/abs/2508.09191.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Wenxuan Wang, Kai Wu, Yujian Betterest Li, Dan Wang, and Xiaoyu Zhang. Synthetic series-symbol data generation for time series foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=xB1ZNgq0Xp.

Xue Wang, Tian Zhou, Jinyang Gao, Bolin Ding, and Jingren Zhou. Output scaling: Yinglong-delayed chain of thought in a large pretrained time series forecasting model, 2025b. URL https://arxiv.org/abs/2506.11029.

Yunshi Wen, Wesley M. Gifford, Chandra Reddy, Lam M. Nguyen, Jayant Kalagnanam, and Anak Agung Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models, 2026. URL https://arxiv.org/abs/2602.06909.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Yd8eHMY1wz.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S. Yu. From observations to states: Latent time series forecasting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=s49fw3BVU0.

Qingren Yao, Chao-Han Huck Yang, Renhe Jiang, Yuxuan Liang, Ming Jin, and Shirui Pan. Towards neural scaling laws for time series foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uCqxDfLYrB.

Kun Yi, Qi Zhang, Wei Fan, Longbing Cao, Shoujin Wang, Hui He, Guodong Long, Liang Hu, Qingsong Wen, and Hui Xiong. A survey on deep learning based time series analysis with frequency transformation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6206–6215, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3736571. URL https://doi.org/10.1145/3711896.3736571.

Boning Zhang, Haishuai Wang, Zehong Hu, Jiajun Wang, Hongyi Zhang, and Jia Jia. Re-diffusion: Modeling latent residuals with diffusion for time-series forecasting. In *Proceedings of the ACM Web Conference 2026*, WWW '26, pp. 7390–7400, New York, NY, USA, 2026a. Association for Computing Machinery. ISBN 9798400723070. doi: 10.1145/3774904.3792539. URL https://doi.org/10.1145/3774904.3792539.

Yunhao Zhang, Ruiying Qi, Jiale Zheng, Jianfeng Zhang, Lujia Pan, and Junchi Yan. Time series as language: A universal tokenizer for general-purpose time series foundation models, 2026b. URL https://arxiv.org/abs/2606.09861.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. URL https://arxiv.org/abs/2412.20404.

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

A Analysis of the Bucketing Strategy

This section explains the motivation for the Length-Aware Bucket strategy and quantifies padding overhead with and without bucketing. We also specify the bucket sampling probabilities used to control training exposure across context lengths.

A.1 Main Motivation for Bucketing

Increasing Input Context Length in Time Series Foundation Models. Pre-training large-scale time series foundation models on massive datasets has emerged as a prominent trend. As model scales grow, input context lengths expand accordingly—for instance, Timer-S1 supports contexts up to 11.5K. However, to preserve local features, these models typically employ short patch lengths (e.g., P=16) for token embedding. Long contexts consequently generate an enormous number of tokens, and since Transformer complexity scales as $\mathcal{O}(n^2)$ for n tokens, computational costs become prohibitively high.

Token Waste from Variable-Length Training Strategies. Time series foundation models are commonly trained with variable-length inputs to accommodate arbitrary sequence lengths. The standard practice is left-side zero-padding all sequences in a batch to match the longest sequence. For example, training on lengths $\{256, 512, 1024\}$ requires padding 256- and 512-length sequences with 768 and 512 zeros, respectively. Such large padding disparities introduce substantial computational waste from meaningless tokens.

Patch-Based Token Construction. Tokens are formed by splitting raw sequences into fixed-length patches, where each patch corresponds to one token. With P=16, a batch containing sequences of lengths 128, 256, and 1024 is padded to 1024, yielding 64 tokens per sequence. However, the 128-length sequence only needs 8 valid tokens yet computes all 64, wasting 56 padding tokens; the 256-length sequence similarly wastes 48 tokens. These invalid tokens consume both compute and memory during training and inference.

Proposed Bucketing Algorithm. To mitigate token waste from zero-padding, we propose a bucketing algorithm. Sequences are grouped by length intervals, and each group is padded and processed independently within its bucket. This prevents short sequences from being padded to the global maximum length. The symbols we use and their meanings are shown in Table 1.

Table 1: Definitions and default values of hyperparameters

Symbol	Meaning	Value	Symbol	Meaning	Value
P	Patch length	16	B	Width for bucketing	256
l	Length of sequence	-	L	Max sequence length	-
$l_{\min}$	Min context length	64	N	Batch size	$L - l_{\min} + 1$

A.2 Quantitative Analysis of Bucketing

In this section, we present the specific calculation method and theoretical analysis for Figure 2 (right). We define the function mapping sequence length x to the number of patch tokens as:

$$T(x) = \left\lceil \frac{x}{P} \right\rceil. \tag{5}$$

After padding a time series of length l to target length L_{pad} and patching, the total token count is $T(L_{\text{pad}})$, of which $T(l)$ are valid tokens. The number of invalid tokens for a single sequence is thus $T(L_{\text{pad}}) - T(l)$.

To enable reproducible quantitative comparison of token waste between padding strategies, we construct the following deterministic batch generation rule. Given a maximum batch length L, we build a batch containing one sequence for every integer length in the interval $[l_{\min}, L]$:

$$\mathcal{L} = \{l_{\min},\ l_{\min}+1,\ \ldots,\ L\}, \qquad N = L - l_{\min} + 1. \tag{6}$$

This setting fully covers a uniform length distribution over $[l_{\min}, L]$. For the non-bucketing strategy, it also constitutes the worst-case (upper-bound) token waste: all sequences are padded to L, and the invalid token contribution $T(L) - T(l)$ increases monotonically as l decreases. Including all shorter sequences in the batch maximizes total padding waste. Under random uniform sampling with smaller batch sizes, expected waste falls below this upper bound.

Token Waste under Non-Bucketing Strategy. Without bucketing, all sequences are left-zero-padded to the maximum batch length L. The wasted tokens for a sequence of length l are:

$$w_{\text{no}}(l;\, L) = T(L) - T(l) = \left\lceil \frac{L}{P} \right\rceil - \left\lceil \frac{l}{P} \right\rceil . \tag{7}$$

Summing over the entire batch:

$$W_{\text{no}}(L) = \sum_{l=l_{\min}}^{L} w_{\text{no}}(l;\, L) = N \cdot T(L) - \sum_{l=l_{\min}}^{L} T(l). \tag{8}$$

For example, with $P = 16$, $L = 1024$, and $l = 128$: total tokens after padding are $T(L) = \lceil 1024/16 \rceil = 64$, of which $T(l) = \lceil 128/16 \rceil = 8$ are valid, yielding $64 - 8 = 56$ invalid tokens.

Token Waste under Bucketing Strategy. Sequences are grouped by length with bucket width B. A sequence of length l falls into bucket $k = \lceil l/B \rceil$, with padding target equal to the bucket upper bound:

$$C(l) = B \cdot \left\lceil \frac{l}{B} \right\rceil . \tag{9}$$

For $B = 256$: $l = 128 \mapsto 256$; $l = 300 \mapsto 512$; $l = 1024 \mapsto 1024$. Each bucket forms batches and computes independently—the padding target of a sequence depends on neither other batch elements nor the global maximum L. After bucketing, wasted tokens for length l are:

$$w_{\text{bkt}}(l) = T\big(C(l)\big) - T(l) = \left\lceil \frac{C(l)}{P} \right\rceil - \left\lceil \frac{l}{P} \right\rceil . \tag{10}$$

Summing over all time series in $[l_{\min}, L]$:

$$W_{\text{bkt}}(L) = \sum_{l=l_{\min}}^{L} w_{\text{bkt}}(l) = \sum_{l=l_{\min}}^{L} \left[T\big(C(l)\big) - T(l) \right]. \tag{11}$$

Note that L here only defines the summation range (which lengths appear in the batch) and is not the padding target for individual time series.

Saving Ratio and Strategy Comparison. We define the token saving ratio of bucketing relative to non-bucketing as:

$$\text{Saving}(L) = \left(1 - \frac{W_{\text{bkt}}(L)}{W_{\text{no}}(L)} \right) \times 100\%. \tag{12}$$

where a positive value indicates bucketing reduces invalid tokens; a negative value indicates no benefit under the given setting. In Figure 2, we plot the token waste curves for the non-bucketing in Equation equation 8 and bucketing in Equation equation 11 as a function of maximum input context length L. The saving ratio from Equation equation 12 is additionally shown as bar charts. Results clearly demonstrate that token savings reach approximately 80% when the maximum sequence length exceeds 1024.

The sole limitation is reduced parallel throughput under single-threaded execution, as buckets must be processed sequentially. This can be addressed via multi-threaded inference: when the number of threads matches the number of buckets, the bucketing strategy imposes no parallel speed overhead. In practical use, to accommodate predictions with shorter contexts, we additionally set finer-grained buckets such as [0, 16], [17, 32], [33, 64], [65, 128], and [129, 256]. We then began bucketing at intervals of 256.

A.3 Bucket Sampling Probabilities

Let K denote the number of length buckets, ordered by their upper boundaries $C_1 < \cdots < C_K$. At pre-training stage $s \in \{1, 2, 3\}$, the probability of sampling bucket k is

$$\pi_k^{(s)} = \frac{\exp\left[-\alpha_s(k-1)\right]}{\sum_{j=1}^{K} \exp\left[-\alpha_s(j-1)\right]}, \qquad k = 1, \ldots, K, \tag{13}$$

where α_s controls the sampling bias toward shorter buckets. Stage 1 uses $\alpha_1 = 0$, giving uniform probabilities $\pi_k^{(1)} = 1/K$. Stages 2 and 3 use $\alpha_2, \alpha_3 > 0$, respectively, so that probabilities decrease exponentially with bucket index. Larger coefficients place greater emphasis on shorter contexts. At each training step, every GPU independently draws a bucket from the stage-specific distribution and constructs its local batch from that bucket's sample pool.

B Model Architecture Analysis

This section details the TS-AE architecture and its three-stage pre-training strategy, including latent spectral alignment and adversarial reconstruction. We then describe predictor adaptation to the latent space and the decoding of quantile-indexed predictions into forecasts at the original temporal resolution.

B.1 Detailed Architecture of TS-AE

TS-AE is a deterministic autoencoder with 1.8M parameters. It processes each input channel independently using shared weights and provides $2\times$ temporal compression and corresponding reconstruction.

Input preparation. Each univariate context is left-padded to its assigned bucket boundary and aligned to a multiple of the compression factor. Let L denote the resulting length. The encoder input comprises three channels: a temporal index $\mathbf{j}$, normalized values $\mathbf{u}$, and an observation mask $\mathbf{m}$ Ansari et al. (2025); Podest et al. (2026):

$$\begin{aligned} \mathbf{u} &= \mathrm{Fill}_0\left(\mathcal{N}(\mathbf{x}; \mu, s)\right), \\ j_t &= \frac{t - L - 1}{\kappa}, \qquad t = 1, \ldots, L, \\ \mathbf{U} &= \mathrm{Stack}(\mathbf{j}, \mathbf{u}, \mathbf{m}) \in \mathbb{R}^{3 \times L}. \end{aligned} \tag{14}$$

Here, $\mathcal{N}$ denotes reversible instance normalization (Kim et al., 2022), and μ and s are its stored location and scale statistics. Missing and padded entries are represented as NaN during normalization and replaced with zero afterward. The mask distinguishes these entries from observed values. The fixed scale κ controls the temporal index magnitude; indexing backward from the context endpoint provides consistent temporal coordinates across input lengths.

For missing-value augmentation, normalization statistics are computed before applying point or patch masking. Artificially masked values are then replaced with zero, and the encoder-visible mask is updated. The clean normalized values remain reconstruction targets.

Convolutional embedding. The encoder embeds $\mathbf{U}$ through parallel wide and causal convolutional branches. Each branch applies convolution, RMSNorm, SiLU, and dropout. Their outputs are concatenated along the feature dimension and fused by a kernel-size-3 convolution:

$$\mathbf{H}^{(0)} = \mathrm{Conv}_3\left(\mathrm{Concat}_{\mathrm{feat}}\left[f_{\mathrm{wide}}(\mathbf{U}), f_{\mathrm{causal}}(\mathbf{U})\right]\right). \tag{15}$$

The wide branch captures broad temporal patterns and provides neighboring context that can reduce sensitivity to isolated noise or missing observations (Chen et al., 2025b). The causal branch extracts local, history-dependent features and fine temporal details. The fusion convolution preserves temporal length; compression is performed by subsequent rescaling blocks. The decoder instead embeds its three-channel latent input using RMSNorm, SiLU, and a causal projection convolution.

Backbone components. Each backbone stage combines residual causal convolution, linear attention, and an Inception module, followed by optional temporal rescaling.

(1) Residual causal convolution. Two causal convolutions, preceded by RMSNorm and SiLU, refine local temporal dependencies. Dropout is applied before the second convolution. A residual connection uses either an identity mapping or a kernel-size-1 projection to match feature dimensions.

(2) Linear temporal attention. After RMSNorm, a kernel-size-1 convolution produces queries, keys, and values, which are divided into multiple heads. For one head, let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d_h}$, where T is the current sequence length and d_h is the head dimension. We apply a non-negative feature map:

$$\widetilde{\mathbf{Q}} = \mathrm{ReLU}(\mathbf{Q}), \qquad \widetilde{\mathbf{K}} = \mathrm{ReLU}(\mathbf{K}). \tag{16}$$

Attention is evaluated through associative multiplication:

$$\begin{aligned} \mathbf{M} &= \widetilde{\mathbf{K}}^\top \mathbf{V}, \qquad \mathbf{s} = \widetilde{\mathbf{K}}^\top \mathbf{1}_T, \\ \mathbf{O}_{t,:} &= \frac{\widetilde{\mathbf{Q}}_{t,:}\mathbf{M}}{\max\left(\widetilde{\mathbf{Q}}_{t,:}\mathbf{s}, \epsilon_{\mathrm{attn}}\right)}, \qquad \epsilon_{\mathrm{attn}} = 10^{-6}. \end{aligned} \tag{17}$$

Head outputs are concatenated, projected with a kernel-size-1 convolution, and added to the residual input. This aggregates information over the entire temporal context without constructing a $T \times T$ attention matrix. Its attention computation costs $\mathcal{O}(T n_h d_h^2)$ for n_h heads and is linear in T for fixed feature dimensions.

Long temporal reductions can overflow in FP16, as observed for inputs exceeding approximately 9K time steps in our experiments. We therefore disable autocast locally and compute $\mathbf{M}$, $\mathbf{s}$, the denominator, and the normalized readout in FP32. The result is cast back to the mixed-precision dtype before the output projection. This stabilization is applied at every attention block, irrespective of input length.

(3) Multi-scale Inception convolution. Following RMSNorm and SiLU, parallel convolutions with different kernel sizes extract temporal patterns at multiple scales (Ismail Fawaz et al., 2020). Their outputs are concatenated and combined with a residual connection, using a kernel-size-1 shortcut when feature dimensions change.

(4) Temporal rescaling and output projection. Encoder downsampling uses a stride-2 convolution followed by SiLU Wan et al. (2025), RMSNorm, and a causal refinement convolution. Decoder upsampling uses nearest-neighbor interpolation followed by convolution, SiLU, RMSNorm, and causal refinement. The configuration used in this work provides an overall compression factor of two. Both backbones end with RMSNorm, SiLU, and a causal convolution that projects hidden features to a single output channel. With fixed depth, feature dimensions, and kernel sizes, encoding and decoding both scale as $\mathcal{O}(L)$.

Masking variable-length latent sequences. Let $\mathbf{z}^{\mathrm{raw}} \in \mathbb{R}^{L/2}$ denote the encoder output before masking, and let $\mathbf{m}$ be the encoder-visible input mask. Each latent position is considered valid if at least one of its two aligned input positions is observed:

$$\begin{aligned} m_j^z &= \mathbf{1}\left[m_{2j-1} + m_{2j} > 0\right], \\ z_j &= m_j^z z_j^{\mathrm{raw}}, \qquad j = 1, \ldots, L/2. \end{aligned} \tag{18}$$

Thus, entirely padded or missing groups are explicitly zeroed, while partially observed groups remain valid. During augmented training, this mask follows the observations available to the encoder, rather than the reconstruction-target mask.

The decoder receives the masked latent values, their validity mask, and a temporal index constructed at the latent resolution:

$$\mathbf{U}^z = \mathrm{Stack}(\mathbf{j}^z, \mathbf{z}, \mathbf{m}^z) \in \mathbb{R}^{3 \times L/2}, \qquad j_j^z = \frac{j - L/2 - 1}{\kappa}. \tag{19}$$

The decoder reconstructs the sequence at the original temporal resolution. When outputs in the original measurement scale are required, the stored normalization statistics are used to invert $\mathcal{N}$.

B.2 THREE-STAGE PRE-TRAINING STRATEGY

We adopt three-stage pre-training to progressively develop general compression and reconstruction capabilities, improve adaptation to short and incomplete contexts, and refine reconstruction details (Chen et al., 2025a). All stages use the Length-Aware Bucket strategy, with stage-specific sampling probabilities and optimization targets, as illustrated in Figure 4 (left).

- **Stage 1: General Representation Compression.** We jointly train the encoder and decoder while sampling all length buckets with equal probability. This exposes TS-AE to different context lengths and establishes its basic compression and reconstruction capabilities. The encoder learns continuous compressed representations, while the decoder learns to recover the original temporal resolution from these representations.
- **Stage 2: Short-Sequence Adaptation.** After Stage 1, reconstruction is generally more accurate for long sequences than for short ones. We therefore freeze the decoder and optimize the encoder while increasing the sampling probabilities of shorter buckets. Keeping the decoder fixed encourages the encoder to improve short-context representations while maintaining compatibility with the learned decoding function. We additionally introduce masked reconstruction to improve robustness to incomplete inputs. Originally missing observations are excluded during data preparation; random point masks and contiguous patch masks are then applied artificially. The encoder receives the corrupted inputs, while the corresponding clean observations remain valid reconstruction targets. This stage jointly addresses short-context compression and missing-input robustness.
- **Stage 3: Adversarial Detail Enhancement.** We freeze the adapted encoder and further train the decoder, retaining short-bucket oversampling and the masking augmentation used in Stage 2. A compact InceptionTime discriminator (Ismail Fawaz et al., 2020) provides an adversarial objective in addition to the reconstruction objective. The discriminator learns to distinguish original from reconstructed sequences, while the decoder learns to produce reconstructions that better preserve temporal details. This stage improves reconstruction fidelity from the established compressed representations while keeping the encoder's latent mapping fixed. Together, the three stages balance general compression, adaptation to challenging inputs, and detailed reconstruction.

B.3 ROLE OF LATENT SPECTRAL ALIGNMENT

Motivation. When training the 1.8M-parameter TS-AE, we observed that accurate reconstruction did not necessarily imply faithful latent trajectories. Although the encoder captured broad trends, its outputs could appear sign-reversed relative to the input and preserve fine-scale oscillations less accurately. A jointly trained decoder can compensate for transformations of the latent representation, allowing good reconstruction despite these differences. We therefore introduce latent spectral alignment to encourage preservation of periodic structure directly in the compressed representation. Figure 11 visualizes the impact of using and not using latent spectral alignment on the fidelity of the model's latent space, comparing it with a $2\times$ average downsampling. While our TS-AE still achieves high accuracy even without this loss, adding it clearly demonstrates significantly improved fidelity in the latent space, noticeably enhancing its detail.

Downsampling and seasonal decomposition. For each normalized target window $\mathbf{x}_i$, we construct a reference sequence using non-overlapping average pooling:

$$a_{i,j} = \frac{x_{i,2j} + x_{i,2j+1}}{2}, \qquad b_{i,j} = m_{i,2j} m_{i,2j+1}, \tag{20}$$

where $\mathbf{m}_i$ is the target validity mask and $\mathbf{b}_i$ identifies pooling positions containing no padding. The reference $\mathbf{a}_i$ and the latent representation $\mathbf{z}_i = E_\theta(\tilde{\mathbf{x}}_i)$ are aligned at the same temporal resolution. We retain their corresponding valid contiguous regions before decomposition and spectral estimation. Artificially masked inputs remain valid reconstruction and spectral targets.

For either sequence $\mathbf{v} \in \{\mathbf{a}_i, \mathbf{z}_i\}$, we estimate its trend using a centered moving average with a window size of 29, measured at the compressed temporal resolution:

$$\mathcal{T}_{29}(\mathbf{v})_t = \frac{1}{29} \sum_{r=-14}^{14} v_{t+r}^{\text{ext}}, \qquad \mathcal{S}(\mathbf{v})_t = v_t - \mathcal{T}_{29}(\mathbf{v})_t, \tag{21}$$

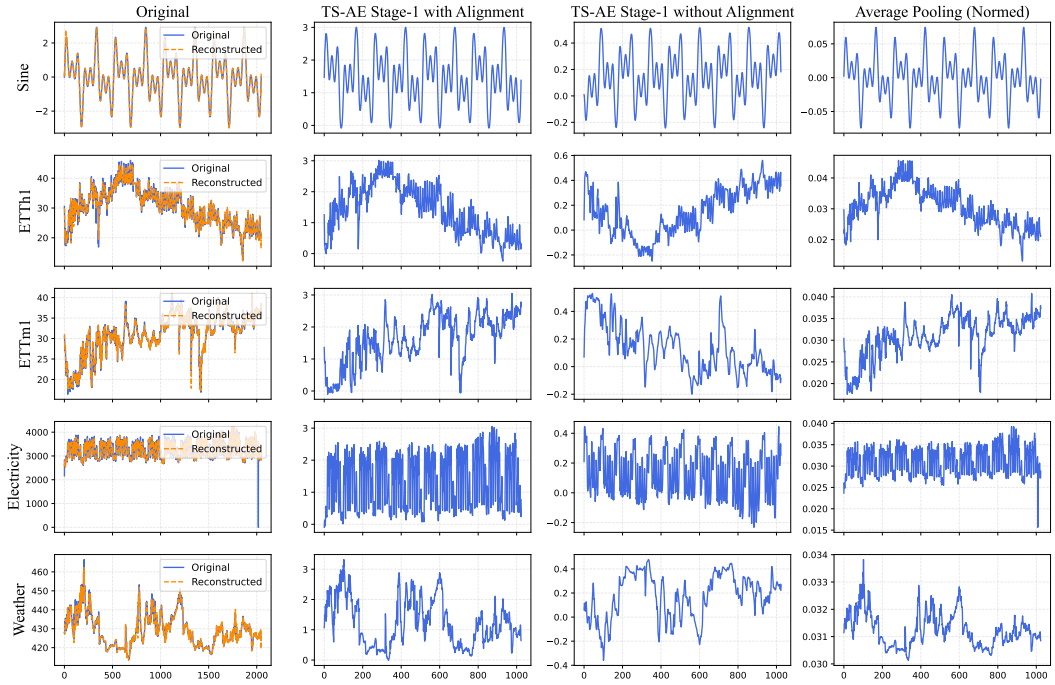


Figure 11: Using simple Sine signals and several datasets such as ETTh1, ETTm1, Electricity, and Weather as examples, we demonstrate the reconstruction performance of TS-AE in one-stage pre-training and the impact of using latent spectral alignment loss on the latent space.

where $\mathbf{v}^{\text{ext}}$ uses endpoint replication to preserve sequence length. The residual $\mathcal{S}(\mathbf{v})$ serves as the seasonal component for spectral alignment. No explicit matching loss is imposed on the trend.

Windowed power spectrum estimation. We estimate the seasonal PSD using non-parametric windowed power spectrum estimation (WPSE) (O'Shea, 2000), implemented as a windowed periodogram. For a valid seasonal sequence $\mathbf{s} \in \mathbb{R}^M$, we subtract its mean and apply a symmetric Hann window:

$$\begin{aligned} w_t &= \frac{1}{2}\left[1 - \cos\left(\frac{2\pi t}{M-1}\right)\right], \\ S_{\mathbf{s}}(\omega_k) &= \frac{\left|\mathcal{F}_M\left[(\mathbf{s} - \bar{s}) \odot \mathbf{w}\right]_k\right|^2}{\sum_{t=0}^{M-1} w_t^2}, \end{aligned} \tag{22}$$

where $\bar{s}$ is the sample mean, $\odot$ denotes elementwise multiplication, and $\mathcal{F}_M$ is the M-point discrete Fourier transform computed using an FFT. The sampling rate is normalized to one in compressed-time coordinates.

For real-valued inputs, we retain the non-negative frequencies $\omega_k = 2\pi k/M$, with $k = 0, \ldots, \lfloor M/2 \rfloor$. This yields $\lfloor M/2 \rfloor + 1$ coefficients, including DC and, for even M, the Nyquist frequency. These coefficients are used directly without doubling the interior frequency bins. Both branches use identical spectral estimation settings.

Spectral alignment and numerical handling. Let $\mathbf{p}_i^a$ and $\mathbf{p}_i^z$ denote the spectral vectors obtained from $\mathcal{S}(\mathbf{a}_i)$ and $\mathcal{S}(\mathbf{z}_i)$, respectively. We exclude windows that are too short for the estimator or whose reference seasonal spectrum has negligible energy. Specifically,

$$\mathcal{I} = \left\{ i : M_i \geq 3,\ \sum_k p_{i,k}^a > \eta \right\}, \tag{23}$$

where M_i is the common valid length and η is a numerical energy threshold. The loss is averaged over these eligible windows:

$$\mathcal{L}_{\text{spec}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left[1 - \frac{\langle \mathbf{p}_i^a, \mathbf{p}_i^z \rangle}{\max\left(\|\mathbf{p}_i^a\|_2 \|\mathbf{p}_i^z\|_2, \epsilon \right)} \right], \tag{24}$$

where $\epsilon > 0$ prevents division by zero. If $\mathcal{I}$ is empty, the spectral loss is set to zero. Eligibility depends on the reference spectrum rather than the latent spectrum, so a degenerate latent prediction is not excluded when its reference has nonzero spectral energy.

Optimization and gradient propagation. We use $\lambda_{\text{spec}} = 0.005$ and optimize

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{rec}} + 0.005\, \mathcal{L}_{\text{spec}} \tag{25}$$

during the first two pre-training stages. The reference branch is treated as fixed, while moving-average decomposition, FFT, power computation, and cosine similarity retain gradients through the latent branch to the encoder. The spectral term is omitted in Stage 3 because the encoder is frozen. FFT-based spectral estimation introduces $\mathcal{O}(M \log M)$ computation per window during pre-training and is not required for forecasting inference.

B.4 Adversarial Reconstruction Objective

In the third pre-training stage, we freeze the TS-AE encoder and refine the decoder using reconstruction and adversarial losses. A lightweight InceptionTime discriminator (Ismail Fawaz et al., 2020) distinguishes real sequences from reconstructions through convolutional branches with different kernel sizes. The decoder acts as the generator, while the discriminator provides an additional learning signal for improving reconstruction details.

Inputs and reconstruction. For a clean target window $\mathbf{x}_i$, we construct an augmented input $\tilde{\mathbf{x}}_i$ using point and contiguous patch masking. Its latent representation and reconstruction are

$$\mathbf{z}_i = \text{sg}\left(E_\theta(\tilde{\mathbf{x}}_i) \right), \qquad \hat{\mathbf{x}}_i = D_\phi(\mathbf{z}_i), \tag{26}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation. The encoder parameters θ remain fixed throughout this stage.

Let $\mathbf{m}_i$ denote the target validity mask. Padding is excluded, whereas artificially masked observations remain valid targets. We write $D_\psi(\mathbf{v}; \mathbf{m}_i) \in (0, 1)$ for the discriminator's estimated probability that $\mathbf{v}$ is real, with the validity mask used in its evaluation. Real and reconstructed inputs receive the same padding-mask treatment.

Discriminator update. The discriminator is trained to assign high probabilities to clean targets and low probabilities to reconstructions. For a batch of N windows, its binary cross-entropy objective is

$$\mathcal{L}_D = -\frac{1}{N} \sum_{i=1}^{N} \Big[\log D_\psi(\mathbf{x}_i; \mathbf{m}_i) + \log\left(1 - D_\psi\left(\text{sg}(\hat{\mathbf{x}}_i); \mathbf{m}_i \right) \right) \Big]. \tag{27}$$

The clean targets have binary label 1, and the reconstructions have label 0. Reconstructions are detached during this update so that gradients modify only the discriminator parameters ψ. The encoder and decoder remain unchanged.

Decoder update. We next hold the discriminator parameters fixed and optimize the decoder. Its non-saturating adversarial loss encourages reconstructed sequences to receive high real-sequence probabilities:

$$\mathcal{L}_{\text{adv}} = -\frac{1}{N} \sum_{i=1}^{N} \log D_\psi\left(D_\phi(\mathbf{z}_i); \mathbf{m}_i \right). \tag{28}$$

During this update, reconstructions are not detached. Although the discriminator parameters are fixed, gradients propagate through its operations to the decoder:

$$\nabla_\phi \mathcal{L}_{\text{adv}} = \sum_{i=1}^{N} \left(\frac{\partial \hat{\mathbf{x}}_i}{\partial \phi} \right)^\top \nabla_{\hat{\mathbf{x}}_i} \mathcal{L}_{\text{adv}}. \tag{29}$$

Thus, freezing the discriminator parameters does not disable differentiation with respect to its inputs.

The decoder combines this adversarial signal with the reconstruction objective:

$$\begin{aligned} \mathcal{L}_{\text{dec}} &= \mathcal{L}_{\text{rec}} + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}, \\ \mathcal{L}_{\text{rec}} &= \mathcal{L}_{\text{MSE}} + \lambda_{\text{WD}} \mathcal{L}_{\text{WD}}, \end{aligned} \tag{30}$$

where $\lambda_{\text{WD}} = 0.25$ and $\lambda_{\text{adv}} = 0.075$. The reconstruction terms preserve pointwise accuracy and the empirical value distribution, while the adversarial term encourages realistic temporal details. Latent spectral alignment is omitted because the frozen encoder produces representations that do not depend on the decoder parameters.

Alternating optimization. Training alternates between discriminator and decoder updates. A discriminator update clears its gradients, evaluates $\mathcal{L}_D$ using detached reconstructions, and updates ψ. A decoder update clears the decoder gradients, evaluates $\mathcal{L}_{\text{dec}}$ with the discriminator parameters fixed, and updates ϕ. The encoder remains frozen in both updates. Separate optimizer states are maintained for the discriminator and decoder.

B.5 Latent Forecasting Adaptation and Quantile Decoding

Supervision in the original time domain. During latent adaptation, we freeze the TS-AE encoder and decoder and optimize only the pre-trained PatchTST predictor (Rombach et al., 2022). Supervision is applied to decoded forecasts at the original temporal resolution, without an additional forecasting loss in latent space (Chen et al., 2025a).

This choice has two motivations. First, each forward pass predicts between 1 and 16 latent patches of length 16, corresponding to 16–256 latent positions and 32–512 original time steps after $2\times$ upsampling. Constructing latent targets by separately encoding these relatively short future windows can provide less reliable supervision, given TS-AE's lower compression fidelity on short sequences. Second, latent supervision requires careful alignment of normalization scales between encoded future targets and context-conditioned predictions. Applying the loss after decoding avoids introducing separate latent forecasting targets and their associated normalization choices.

Let P denote the forecast length in the original time domain, $Q = |\mathcal{Q}|$, and $\hat{y}_{t,q}$ denote the decoded prediction for quantile q. We use the same quantile regression objective as in raw-space pre-training:

$$\mathcal{L}_{\text{forecast}} = \frac{1}{PQ} \sum_{t=1}^{P} \sum_{q \in \mathcal{Q}} \left[q(y_t - \hat{y}_{t,q})_+ + (1-q)(\hat{y}_{t,q} - y_t)_+ \right], \tag{31}$$

where $(u)_+ = \max(u, 0)$. Decoded predictions and targets use a consistent normalization scale. Although the decoder parameters are frozen, gradients propagate through the decoder to update the predictor.

Joint decoding of quantile-indexed predictions. For each input series, the predictor produces $\widehat{\mathbf{Z}} \in \mathbb{R}^{Q \times P/2}$, with one latent trajectory for each target quantile. We concatenate these trajectories in a fixed quantile order, preserving temporal order within each trajectory:

$$\mathbf{v} = \text{Concat}_{q \in \mathcal{Q}} \left(\widehat{\mathbf{Z}}_{q,:} \right) \in \mathbb{R}^{QP/2}. \tag{32}$$

The decoder upsamples this sequence by a factor of two, and its output is reshaped into the final quantile forecasts:

$$\widehat{\mathbf{Y}} = \text{Reshape}_{Q \times P} \left(D_\phi(\mathbf{v}) \right) \in \mathbb{R}^{Q \times P}. \tag{33}$$

This operation is performed separately for each original input series, without concatenating different series.

For a batch of N series, this arrangement keeps the decoder's effective batch size at N, compared with NQ when treating each quantile trajectory as an independent sample. It also presents the decoder with longer sequences, motivated by TS-AE's stronger reconstruction performance on longer inputs. The total number of latent positions remains unchanged.

Because concatenation occurs along the temporal dimension, decoder operations can couple different quantile trajectories across their boundaries. The reshaped outputs are therefore jointly trained

toward their assigned quantile levels through Equation equation 31; the intermediate latent trajectories are not independently supervised as latent quantiles.

C Implementation Details

This section documents model hyperparameters, datasets, baselines, and evaluation metrics. It also explains the choice of a deterministic autoencoder and provides protocols for assessing spectral forecastability, downstream predictive utility, and inter-channel correlation preservation. Finally, we present and analyze training configurations that examine bucket sampling, GPU parallelism, and gradient accumulation.

C.1 Model Hyperparameters

We use TS-AE-Small, a channel-independent deterministic autoencoder with approximately 1.8M parameters and $2\times$ temporal compression. Its encoder and decoder each contain four backbone stages combining causal convolutions, linear attention with four heads, and multi-scale Inception modules. TS-AE is trained with a maximum context length of 6656; its detailed architecture and input-processing settings are provided in Table 2. For forecasting, we use the Small PatchTST configuration with 21.08M parameters, comprising six Transformer encoder layers, a hidden dimension of 512, a feed-forward dimension of 2048, and eight attention heads. Only this Small predictor is trained for forecasting experiments. The Base and Large configurations in Table 3 are used solely for the computational complexity analysis in Figure 2 (left).

Table 4 summarizes the training settings. All models are trained on eight GPUs using AdamW, FP16 mixed precision, and linear learning-rate decay with 2048 warm-up steps.

C.2 Pre-training and Evaluation Datasets

GIFT-Eval pre-training corpus. We pre-train TS-AE, PatchTST, and LatentTime using the public GIFT-Eval pre-training corpus (Aksu et al., 2024). The released corpus contains 88 datasets, including 71 univariate and 17 multivariate datasets, with approximately 4.5 million time series and 230 billion observations across 13 sampling frequencies. Curated from LOTSA, this corpus excludes the GIFT-Eval evaluation datasets to support leakage-free pre-training. We maintain this separation throughout training. Dataset statistics grouped by domain are provided in Tables 5–13. These statistics describe the released source corpus; training examples are subsequently constructed by sampling variable-length windows.

Synthetic data and augmentation. To increase training diversity, we supplement the real-world corpus with synthetic time series generated using KernelSynth (Ansari et al., 2024), the synthetic data generator introduced in ForecastPFN (Dooley et al., 2023), and S2Generator (Wang et al., 2025a). KernelSynth samples from Gaussian processes with randomly composed covariance kernels, while ForecastPFN combines global trends, multi-scale seasonality, and noise. S2Generator generates response sequences by applying sampled symbolic expressions to excitation time series. We additionally apply TSMixup (Ansari et al., 2024), which constructs random convex combinations of scaled training windows to combine patterns from different series. Synthetic generation and augmentation are performed without using GIFT-Eval evaluation records.

GIFT-Eval evaluation datasets. We evaluate the forecasting performance on GIFT-Eval benchmark (Aksu et al., 2024), which comprises 23 dataset collections containing approximately 144,000 time series and 177 million observations. The benchmark covers 10 sampling frequencies and seven domains: Nature, Web/CloudOps, Energy, Transport, Sales, Econ/Fin, and Healthcare. Dataset, frequency, and forecast-horizon combinations define 97 evaluation tasks covering short-, medium-, and long-term forecasting, with prediction lengths ranging from 6 to 900 time steps. We follow the official rolling-window evaluation protocol and apply the pre-trained models without fine-tuning on the evaluation datasets. Detailed series counts, average sequence lengths, numbers of variables, prediction lengths, and rolling-window counts are reported in Tables 14–20. When pre-training TS-AE, we did not include any data from Gift-Eval. Our model was trained entirely on the leak-free pre-training dataset and synthetic dataset. However, when training all prediction models, including

Table 2: Hyperparameters of TS-AE-Small with $2\times$ temporal compression. Feature widths list the embedding dimension followed by the outputs of the four backbone stages.

Hyperparameter	Value
Architecture	
Model preset	`ts-ae-small-f2-wb`
Total parameters	Approximately 1.8M
Backbone stages per encoder/decoder	4
Base hidden dimension	32
Channel multipliers	$(2, 4, 4, 4)$
Encoder feature widths	$(32, 64, 128, 128, 128)$
Decoder feature widths	$(128, 128, 128, 64, 32)$
Latent channels per input series	1
Temporal compression factor	$2\times$
Encoder downsampling location	After Stage 3
Decoder upsampling location	After Stage 2
Wide convolution kernel size	17
Residual convolution kernel size	3
Causal convolution kernel size	3
Inception kernel sizes	$(3, 5, 9)$
Attention mechanism	ReLU-based linear attention
Attention heads	4
Backbone normalization	RMSNorm
Backbone activation	SiLU
Dropout rate	0.0
Convolutional bias	Enabled
Input Processing and Bucketing	
Input feature channels	Time index, value, mask
Instance normalization	Enabled
Arcsinh transformation	Enabled
Maximum training context length	6656
Temporal encoding scale κ	6656
Length-Aware Bucket strategy	Enabled
Default bucket upper boundaries	$256, 512, \ldots, 6656$
Number of default buckets	26
Default Missing-Value Augmentation	
`batch_missing_rate`	0.05
`sequence_missing_rate`	0.30
Point-mask sampling probability	0.70
Patch-mask sampling probability	0.30
`missing_patch_size`	48
`min_patch_size`	8
`max_num_patches`	4

all PatchTST and LatentTime datasets, we strictly followed the Gift-Eval guidelines, dividing the dataset into training, validation, and test sets. We added an additional training set portion to the TS-AE training data and evaluated the model's predictive performance on its test dataset. Using the validation set defined in Gift-Eval as the evaluation benchmark, we will select the best-performing or five second-best checkpoints on it for testing, and take the best result among them.

C.3 Baseline Models

We compare LatentTime with the following forecasting models, including different model sizes and pre-training variants.

Table 3: Detailed parameter settings of PatchTST predictors in LatentTime.

Hyperparameter	small	base	large
Hidden Size (d_{model})	512	768	768
Feed Foreward Dimension (d_{ff})	2048	3072	3072
Attention Head Dimension (d_{kv})	64	64	64
Number of Attention Heads	8	12	12
Number of Layers	6	6	9
Total Parameters	21.08 M	46.96 M	68.21 M

- **YingLong-300m** (Wang et al., 2025b) is the 300M-parameter variant of YingLong, an encoder-only Transformer with bidirectional attention. It learns forecasting through masked token recovery and jointly models future outputs, with a multi-input ensemble used to reduce prediction variance.
- **TabPFN-TS** (Hoo et al., 2026; Grinsztajn et al., 2026) reformulates forecasting as tabular regression using calendar, automatically extracted seasonal, and time-index features. The pre-trained TabPFN-v2 model conditions on historical feature–target pairs to produce point and probabilistic forecasts without additional time-series-specific pre-training.
- **Toto** (Cohen et al., 2024) is a decoder-only Transformer developed for observability time series. It combines temporal and variate attention to model multivariate dependencies, and uses a Student-t mixture output head for probabilistic next-patch prediction.
- **Sundial-128M** (Liu et al., 2025) is a 128M-parameter generative forecasting Transformer operating on continuous-valued patches. Its flow-matching-based TimeFlow objective learns conditional next-patch distributions, enabling probabilistic forecasts through sampled future trajectories.
- **Kairos** (Feng et al., 2026) is an adaptive Transformer-based forecasting model. Its dynamic patching tokenizer and mixture-of-size encoder adjust tokenization granularity to local information density, while dynamic rotary position embeddings incorporate instance-level spectral characteristics.
- **Moirai-2** (Woo et al., 2024) adopts a decoder-only Transformer architecture for universal forecasting. Compared with the original Moirai, it uses quantile regression instead of a parametric distribution objective and predicts multiple future tokens per forward pass.
- **FlowState** (Graf et al., 2026) combines a state-space encoder with a functional basis decoder. This architecture represents forecasts in continuous time, supporting changes in sampling resolution and forecast horizon without retraining.
- **Chronos-2 (synth)** (Ansari et al., 2025) retains the Chronos-2 architecture but is pre-trained exclusively on synthetic univariate and multivariate time series. This variant supports forecasting without exposure to real-world datasets during pre-training.
- **Toto-2.0-22m** (Khwaja et al., 2026a) is the 22M-parameter variant of Toto 2.0. It retains a decoder-only backbone with temporal and variate attention, while introducing contiguous patch masking and a quantile output head for predicting multiple future patches in a single forward pass.
- **TiRex** (Auer et al., 2025) is a recurrent forecasting foundation model based on xLSTM. Its pre-training combines contiguous patch masking and time series augmentation to strengthen in-context learning and support probabilistic forecasting across short and long horizons.
- **PatchTST-FM** (Wen et al., 2026) is a large pre-trained forecasting model built on a PatchTST-style Transformer encoder. It combines residual input and output projections with a quantile prediction head, learning forecasting and missing-value reconstruction through contiguous patch and forecast-horizon masking.
- **TimesFM-2.5** (Das et al., 2024) is a 200M-parameter model in the TimesFM family of decoder-only forecasting Transformers. It supports contexts of up to 16K time steps and provides probabilistic forecasts through an optional continuous quantile prediction head.

Table 4: Detailed training hyperparameters of TS-AE, PatchTST, and LatentTime. Global batch sizes account for GPU parallelism and gradient accumulation. K denotes one thousand training steps.

Hyperparameter	Value	Stage / Scope
Shared Training Settings		
Optimizer	AdamW	All stages
Learning-rate schedule	Linear decay warm-up	All stages
Warm-up steps	2048	All stages
Minimum learning rate†	5×10^{-6}	TS-AE and predictor
Weight decay	0.1	All stages
AdamW (β_1, β_2)	$(0.9, 0.95)$	All stages
Gradient norm clipping	1.0	All stages
Training precision	FP16 mixed precision	All stages
Number of GPUs	8	All stages
Gradient accumulation steps	1	All stages
TS-AE		
Peak learning rate	5×10^{-5}	
Batch size per GPU	64	
Global batch size	512	Stage 1
Training steps	10K	
Bucket sampling coefficient α	0 (uniform)	
Peak learning rate	2.5×10^{-5}	
Batch size per GPU	64	
Global batch size	512	Stage 2
Training steps	10K	
Bucket sampling coefficient α	0.05	
Peak learning rate	1×10^{-5}	
Batch size per GPU	128	
Global batch size	1024	
Training steps	5K	Stage 3
Bucket sampling coefficient α	0.05	
Discriminator	InceptionTime	
Discriminator learning rate	1×10^{-6}	
PatchTST		
Peak learning rate	5×10^{-4}	
Batch size per GPU	512	
Global batch size	4096	Short-context pre-training
Training steps	500K	
Bucket sampling coefficient α	0.05	
Peak learning rate	1×10^{-4}	
Batch size per GPU	256	
Global batch size	2048	Long-context post-training
Training steps	300K	
Bucket sampling coefficient α	0.05	
LatentTime		
Peak learning rate	5×10^{-5}	
Batch size per GPU	256	
Global batch size	2048	Latent forecasting adaptation
Training steps	500K	
Bucket sampling coefficient α	0.05	

† The minimum learning rate applies to TS-AE and the forecasting predictor, excluding the discriminator.

- **Toto-2.0-313m** (Khwaja et al., 2026a) is the 313M-parameter variant of Toto 2.0. It follows the same temporal–variate attention architecture, contiguous patch masking strategy, and quantile forecasting formulation as the 22M variant, with greater model capacity.

Table 5: Pretraining datasets of Gift-Eval in the Energy domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
BDG-2 Panther	H	105	1	0	919,800
BDG-2 Fox	H	135	1	0	2,324,568
BDG-2 Rat	H	280	1	0	4,728,288
BDG-2 Bear	H	91	1	0	1,482,312
Low Carbon London	H	713	1	0	9,543,348
SMART	H	5	1	0	95,709
IDEAL	H	219	1	0	1,265,672
Sceaux	H	1	1	0	34,223
Borealis	H	15	1	0	83,269
Buildings900K	H	1,792,328	1	0	15,702,590,000
Covid19 Energy	H	1	1	6	31,912
GEF12	H	20	1	1	788,280
GEF14	H	1	1	1	17,520
GEF17	H	8	1	1	140,352
PDB	H	1	1	1	17,520
Spanish	H	1	1	1	35,064
BDG-2 Hog	H	24	1	5	421,056
BDG-2 Bull	H	41	1	3	719,304
BDG-2 Cockatoo	H	1	1	5	17,544
ELF	H	1	1	0	21,792
London Smart Meters	30T	5,520	1	0	166,238,880
Wind Farms	T	337	1	0	172,165,370
Wind Power	4S	1	1	0	7,397,147
Solar Power	4S	1	1	0	7,397,222
Elecdemand	30T	1	1	0	17,520
Australian Electricity Demand	30T	5	1	0	1,153,584
KDD Cup 2022	10T	134	1	9	4,727,519
Residential Load Power	T	271	3	0	145,994,559
Residential PV Power	T	233	3	0	125,338,950

Table 6: Pretraining datasets of Gift-Eval in the Climate domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
CMIP6	6H	1,351,680	53	0	1,973,453,000
ERA5	H	245,760	45	0	2,146,959,000
Subseasonal	D	862	4	0	14,097,148
Subseasonal Precipitation	D	862	1	0	9,760,426
Oikolab Weather	H	8	1	0	800,456
Weather	D	3,010	1	0	42,941,700

- **Chronos-2** (Ansari et al., 2025) is an encoder-only forecasting Transformer pre-trained on real and synthetic time series. It combines temporal attention with group attention to support univariate, multivariate, and covariate-informed forecasting, and produces probabilistic predictions using a quantile output head.
- **Zeus** (Fu et al., 2026) is a multi-task time series foundation model pre-trained on real and synthetic data. It supports point and probabilistic forecasting, imputation, anomaly detection, and classification without task-specific fine-tuning.
- **TiRex-2 (Zeroshot)** (Podest et al., 2026) extends the recurrent TiRex framework to multivariate forecasting with past and future-known covariates. We distinguish its GIFT-Eval zero-shot checkpoint, whose pre-training corpus excludes overlap with both GIFT-Eval and GIFT-Eval-Pretrain.

Table 7: Pretraining datasets of Gift-Eval in the CloudOps domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
Azure VM Traces 2017	5T	159,472	1	2	885,522,908
Borg Cluster Data 2011	5T	143,386	2	5	537,552,854
Alibaba Cluster Trace 2018	5T	58,409	2	6	95,192,530

Table 8: Pretraining datasets of Gift-Eval in the Transport domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
Taxi	30T	67,984	1	0	54,999,060
Uber TLC Daily	D	262	1	0	47,087
Uber TLC Hourly	H	262	1	0	1,129,444
LargeST	5T	42,333	1	0	4,452,510,528
PEMS03	5T	358	1	0	9,382,464
PEMS04	5T	307	3	0	5,216,544
PEMS07	5T	883	1	0	24,921,792
PEMS08	5T	170	3	0	3,035,520
PEMS Bay	5T	325	1	0	16,937,700
Los-Loop	5T	207	1	0	7,094,304
Beijing Subway	30T	276	2	11	248,400
SHMetro	15T	288	2	0	1,934,208
HZMetro	15T	80	2	0	146,000
Q-Traffic	15T	45,148	1	0	264,386,688
Covid Mobility	D	362	1	0	148,602
Traffic Weekly	W	862	1	0	82,752
Traffic Hourly	H	862	1	0	14,978,112
Rideshare	H	2,304	1	0	859,392
Vehicle Trips	D	329	1	0	32,512
Pedestrian Counts	H	66	1	0	3,130,762

- **TiRex-2 (Pretrained)** (Podest et al., 2026) uses the same forecasting architecture as the zero-shot variant but includes GIFT-Eval-Pretrain in its training corpus. Its pre-training data still exclude overlap with the GIFT-Eval evaluation datasets.

C.4 Forecasting Metrics

We evaluate point forecasts using Mean Absolute Scaled Error (MASE) and probabilistic forecasts using Continuous Ranked Probability Score (CRPS), following the GIFT-Eval evaluation protocol (Aksu et al., 2024). Both metrics are evaluated in the original observation space, with lower values indicating better performance.

For a target series, let T denote the forecast origin, H the forecasting horizon, and s the seasonal period specified by the benchmark. We denote the observed value by y_t, the point forecast by $\hat{y}_t$, and the predicted quantile at level q by $\hat{y}_{t,q}$. The median forecast is used for point evaluation.

Mean Absolute Scaled Error (MASE). MASE measures absolute forecasting error relative to the historical error of a seasonal naive predictor:

$$\mathrm{MASE} = \frac{\frac{1}{H}\sum_{t=T+1}^{T+H} |y_t - \hat{y}_t|}{\frac{1}{T-s}\sum_{t=s+1}^{T} |y_t - y_{t-s}|}. \tag{34}$$

Table 9: Pretraining datasets of Gift-Eval in the Web domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
Wiki-Rolling	D	47,675	1	0	40,619,100
Kaggle Web Traffic Weekly	W	145,063	1	0	16,537,182
Extended Web Traffic	D	145,063	1	0	370,926,091

Table 10: Pretraining datasets of Gift-Eval in the Sales domain.

Dataset	Frequency	# Time Series	# Targets	# Covariates	# Obs.
M5	D	30,490	1	0	58,327,370
Favorita Sales	D	111,840	1	0	139,179,538
Favorita Transactions	D	54	1	0	84,408

The denominator is computed from the observed history and provides a series-specific scale, enabling comparisons across time series with different magnitudes.

Continuous Ranked Probability Score (CRPS). CRPS evaluates the predictive distribution by comparing its cumulative distribution function with the realized observation. Let $G_t(u)$ denote the predictive cumulative distribution function at time t. The horizon-averaged CRPS is

$$\mathrm{CRPS} = \frac{1}{H} \sum_{t=T+1}^{T+H} \int_{-\infty}^{\infty} \left[G_t(u) - \mathbf{1}\{y_t \le u\} \right]^2 \mathrm{d}u, \tag{35}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function.

For quantile forecasts, GIFT-Eval uses mean weighted quantile loss as a scale-normalized approximation to CRPS. The evaluation quantiles are $\mathcal{Q}_{\mathrm{eval}} = \{0.1, 0.2, \ldots, 0.9\}$. We define the pinball loss for a residual e as

$$\rho_q(e) = e\,(q - \mathbf{1}\{e < 0\})\,. \tag{36}$$

For a single forecast window, the benchmark score is

$$\mathrm{CRPS}_{\mathrm{GIFT}} = \frac{1}{|\mathcal{Q}_{\mathrm{eval}}|} \sum_{q \in \mathcal{Q}_{\mathrm{eval}}} \frac{2 \sum_{t=T+1}^{T+H} \rho_q\left(y_t - \hat{y}_{t,q}\right)}{\sum_{t=T+1}^{T+H} |y_t|}. \tag{37}$$

Normalization by the absolute target values removes the dependence on the overall magnitude of the series. We refer to this benchmark score as CRPS in the reported results.

For task-level evaluation, the numerator and denominator of the weighted quantile loss are accumulated over all valid forecast targets across series and evaluation windows. Missing-value handling and aggregation across benchmark configurations follow the official GIFT-Eval implementation.

C.5 Reconstruction Metrics

We evaluate TS-AE using five complementary metrics that assess pointwise accuracy and structural fidelity. Each metric is computed separately for each channel-independent window and then averaged over the evaluated windows.

Let $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^L$ denote an instance-normalized input window and its reconstruction in the same normalized space, respectively. We disable inverse normalization during evaluation (`denormalize=False`), so reconstruction errors are measured in normalized units. For MSE, MAE, and Pearson correlation, we retain only paired finite observations:

$$\mathcal{V} = \{t \in \{1, \ldots, L\} \mid x_t \text{ and } \hat{x}_t \text{ are finite}\}\,. \tag{38}$$

For ACF and PSD similarity, non-finite entries are replaced with zeros while preserving the window length.

Table 11: Pretraining datasets of Gift-Eval in the Economics and Finance domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
M1 Yearly	Y	106	1	0	3,136
M1 Quarterly	Q	198	1	0	9,854
M1 Monthly	M	617	1	0	44,892
M3 Yearly	Y	645	1	0	18,319
M3 Quarterly	Q	756	1	0	37,004
M3 Monthly	M	1,428	1	0	141,858
M3 Other	Q	174	1	0	11,933
NN5 Daily	D	111	1	0	81,585
NN5 Weekly	W	111	1	0	11,655
Tourism Yearly	Y	419	1	0	11,198
Tourism Quarterly	Q	427	1	0	39,128
Tourism Monthly	M	366	1	0	100,496
CIF 2016	M	72	1	0	6,334
FRED MD	M	107	1	0	76,612
Bitcoin	D	18	1	0	74,824
GoDaddy	M	3,135	2	0	128,535

Table 12: Pretraining datasets of Gift-Eval in the Nature domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
Sunspot	D	1	1	0	73,894
China Air Quality	H	437	6	0	5,739,234
Beijing Air Quality	H	12	11	0	420,768

Mean Squared Error (MSE). MSE measures the average squared deviation between the input and reconstruction:

$$\mathrm{MSE}(\hat{\mathbf{x}}, \mathbf{x}) = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} (\hat{x}_t - x_t)^2. \tag{39}$$

Mean Absolute Error (MAE). MAE measures the average absolute deviation over the same valid observations:

$$\mathrm{MAE}(\hat{\mathbf{x}}, \mathbf{x}) = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} |\hat{x}_t - x_t|. \tag{40}$$

MAE is less sensitive to large individual errors than MSE.

Pearson Correlation. Pearson correlation measures linear shape agreement between the input and reconstruction. Let $\bar{x}$ and $\bar{\hat{x}}$ denote their respective means over $\mathcal{V}$, and define the centered vectors

$$\mathbf{x}_c = (x_t - \bar{x})_{t \in \mathcal{V}}, \qquad \hat{\mathbf{x}}_c = (\hat{x}_t - \bar{\hat{x}})_{t \in \mathcal{V}}. \tag{41}$$

The numerically stabilized coefficient is

$$r(\hat{\mathbf{x}}, \mathbf{x}) = \frac{\langle \hat{\mathbf{x}}_c, \mathbf{x}_c \rangle}{\max\left(\|\hat{\mathbf{x}}_c\|_2 \|\mathbf{x}_c\|_2, \epsilon\right)}, \qquad \epsilon = 10^{-8}. \tag{42}$$

The denominator is lower-bounded to avoid division by zero for constant sequences. The coefficient lies in $[-1, 1]$, with values closer to one indicating stronger positive linear agreement.

Autocorrelation Function Similarity (ACF-Sim). ACF-Sim evaluates the preservation of temporal dependence over a fixed range of lags. Let $\mathbf{u}$ denote either window after replacing non-finite

Table 13: Pretraining datasets of Gift-Eval in the Healthcare domain.

Dataset	Frequency	Time Series	Targets	Covariates	# Obs.
CDC Fluview ILINet	W	75	5	0	63,903
CDC Fluview WHO NREVSS	W	74	4	0	41,760
Project Tycho	W	1,258	1	0	1,377,707

Table 14: GIFT-Eval datasets in the Nature domain. Forecast settings are reported as Pred/Win, denoting the prediction length and number of rolling windows, respectively.

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
Jena Weather	10T	1	52,704	21	48 / 20	480 / 11	720 / 8
	H	1	8,784	21	48 / 19	480 / 2	720 / 2
	D	1	366	21	30 / 2	–	–
Saugeen	D	1	23,741	1	30 / 20	–	–
	W-THU	1	3,391	1	8 / 20	–	–
	M	1	780	1	12 / 7	–	–
Temperature Rain	D	32,072	725	1	30 / 3	–	–
KDD Cup 2018	H	270	10,898	1	48 / 20	480 / 2	720 / 2
	D	270	455	1	30 / 2	–	–

Table 15: GIFT-Eval datasets in the Web/CloudOps domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
BizITObs - Application	10S	1	8,834	2	60 / 15	600 / 2	900 / 1
BizITObs - Service	10S	21	8,835	2	60 / 15	600 / 2	900 / 1
BizITObs - L2C	5T	1	31,968	7	48 / 20	480 / 7	720 / 5
	H	1	2,664	7	48 / 6	480 / 1	720 / 1
Bitbrains - Fast Storage	5T	1,250	8,640	2	48 / 18	480 / 2	720 / 2
	H	1,250	721	2	48 / 2	–	–
Bitbrains - rnd	5T	500	8,640	2	48 / 18	480 / 2	720 / 2
	H	500	720	2	48 / 2	–	–

Table 16: GIFT-Eval datasets in the Energy domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
ETT1	15T	1	69,680	7	48 / 20	480 / 15	720 / 10
	H	1	17,420	7	48 / 20	480 / 4	720 / 3
	D	1	725	7	30 / 3	–	–
	W-THU	1	103	7	8 / 2	–	–
ETT2	15T	1	69,680	7	48 / 20	480 / 15	720 / 10
	H	1	17,420	7	48 / 20	480 / 4	720 / 3
	D	1	725	7	30 / 3	–	–
	W-THU	1	103	7	8 / 2	–	–
Solar	10T	137	52,560	1	48 / 20	480 / 11	720 / 8
	H	137	8,760	1	48 / 19	480 / 2	720 / 2
	D	137	365	1	30 / 2	–	–
	W-FRI	137	52	1	8 / 1	–	–
Electricity	15T	370	140,256	1	48 / 20	480 / 20	720 / 20
	H	370	35,064	1	48 / 20	480 / 8	720 / 5
	D	370	1,461	1	30 / 5	–	–
	W-FRI	370	208	1	8 / 3	–	–

entries with zeros, and let $\bar{u} = L^{-1} \sum_{t=1}^{L} u_t$. We compute the sample autocorrelation using nor-

Table 17: GIFT-Eval datasets in the Transport domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
Loop Seattle	5T	323	105,120	1	48 / 20	480 / 20	720 / 15
	H	323	8,760	1	48 / 19	480 / 2	720 / 2
	D	323	365	1	30 / 2	–	–
SZ-Taxi	15T	156	2,976	1	48 / 7	480 / 1	720 / 1
	H	156	744	1	48 / 2	–	–
M_DENSE	H	30	17,520	1	48 / 20	480 / 4	720 / 3
	D	30	730	1	30 / 3	–	–

Table 18: GIFT-Eval datasets in the Sales domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
Restaurant	D	807	358	1	30 / 1	–	–
Hierarchical Sales	D	118	1,825	1	30 / 7	–	–
	W-WED	118	260	1	8 / 4	–	–
Car Parts	M	2,674	51	1	12 / 1	–	–

Table 19: GIFT-Eval datasets in the Econ/Fin domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
M4 Yearly	A	22,974	37	1	6 / 1	–	–
M4 Quarterly	Q	24,000	100	1	8 / 1	–	–
M4 Monthly	M	48,000	234	1	18 / 1	–	–
M4 Weekly	W	359	1,035	1	13 / 1	–	–
M4 Daily	D	4,227	2,371	1	14 / 1	–	–
M4 Hourly	H	414	902	1	48 / 2	–	–

Table 20: GIFT-Eval datasets in the Healthcare domain. Forecast settings are reported as prediction length/number of rolling windows (Pred/Win).

Dataset Name	Freq.	#Series	Avg. length	#Vars	Short-term	Med-term	Long-term
Hospital	M	767	84	1	12 / 1	–	–
COVID Deaths	D	266	212	1	30 / 1	–	–
US Births	D	1	7,305	1	30 / 20	–	–
	W-TUE	1	1,043	1	8 / 14	–	–
	M	1	240	1	12 / 2	–	–

malized biased autocovariances:

$$\rho_k(\mathbf{u}) = \frac{\sum_{t=1}^{L-k}(u_t - \bar{u})(u_{t+k} - \bar{u})}{\sum_{t=1}^{L}(u_t - \bar{u})^2}, \quad k = 0, \ldots, K, \tag{43}$$

where $K = \min(32, L-1)$. This follows the convention of `tsa.stattools.acf` in `statsmodels` with `adjusted=False` and `fft=False`. We exclude lag zero and collect the remaining coefficients into vectors $\mathbf{r}_x$ and $\mathbf{r}_{\hat{x}}$ for the input and reconstruction, respectively. Their cosine similarity is

$$\text{ACF-Sim}(\hat{\mathbf{x}}, \mathbf{x}) = \frac{\langle \mathbf{r}_{\hat{x}}, \mathbf{r}_x \rangle}{\|\mathbf{r}_{\hat{x}}\|_2 \|\mathbf{r}_x\|_2}. \tag{44}$$

Values closer to one indicate better preservation of the shape of the autocorrelation profile over the evaluated lags.

Power Spectral Density Similarity (PSD-Sim). PSD-Sim measures agreement in spectral shape. For each zero-filled window $\mathbf{u}$, we subtract its mean and apply a periodic Hann window:

$$w_t = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi(t-1)}{L}\right)\right], \qquad t = 1, \ldots, L. \tag{45}$$

Let U_j denote the j-th coefficient of the L-point real-input FFT of $w_t(u_t - \bar{u})$. The one-sided PSD is

$$S_j = \frac{c_j |U_j|^2}{f_s \sum_{t=1}^{L} w_t^2}, \qquad j = 0, \ldots, \lfloor L/2 \rfloor, \tag{46}$$

where the normalized sampling frequency is $f_s = 1$. The correction factor c_j equals one for the DC bin and, when L is even, the Nyquist bin; it equals two for all other bins. The Hann window reduces spectral leakage.

We normalize each PSD vector across frequency bins:

$$\tilde{\mathbf{S}} = \frac{\mathbf{S}}{\|\mathbf{S}\|_1}. \tag{47}$$

The similarity between the normalized input and reconstruction spectra is

$$\text{PSD-Sim}(\hat{\mathbf{x}}, \mathbf{x}) = \frac{\left\langle \tilde{\mathbf{S}}_{\hat{x}}, \tilde{\mathbf{S}}_x \right\rangle}{\|\tilde{\mathbf{S}}_{\hat{x}}\|_2 \|\tilde{\mathbf{S}}_x\|_2}. \tag{48}$$

Values closer to one indicate better agreement in the distribution of spectral energy across frequencies. This metric assesses spectral shape independently of overall power.

Aggregation and Interpretation. For each dataset and context length, the dataset-level score is the arithmetic mean of the corresponding window-level metric:

$$\overline{m} = \frac{1}{N} \sum_{i=1}^{N} m(\hat{\mathbf{x}}_i, \mathbf{x}_i), \tag{49}$$

where N is the number of evaluated windows and m denotes any of the five metrics. Each window receives equal weight. Lower MSE and MAE indicate better pointwise accuracy, while higher Pearson correlation, ACF-Sim, and PSD-Sim indicate greater structural fidelity. In the main text and figures, ACF and PSD refer to their respective similarity scores. Error bars are grouped from subsets of datasets of varying lengths or from multiple domains.

C.6 Why a Deterministic Autoencoder Rather Than a VAE?

Inspired by DC-AE (Chen et al., 2025a), which demonstrates the potential of high-fidelity compression for efficient latent modeling, we adopt a deterministic autoencoder for TS-AE. Our objective is to preserve temporal information useful for forecasting while reducing sequence length.

Deterministic representations. For a fixed preprocessed input, TS-AE produces a reproducible latent sequence $\mathbf{z} = E_\theta(\mathbf{x})$. A Gaussian VAE instead parameterizes an approximate posterior $q_\theta(\mathbf{z} \mid \mathbf{x})$ and typically uses reparameterized samples during training (Kingma & Welling, 2022):

$$\mathbf{z} = \boldsymbol{\mu}_\theta(\mathbf{x}) + \boldsymbol{\sigma}_\theta(\mathbf{x}) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{50}$$

Deterministic encoding avoids this sampling variability in the context supplied to the forecaster. A VAE can also use its posterior mean at inference, but its representations are still learned under a variational training objective.

Representation fidelity and prior regularization. The VAE objective includes a KL term that encourages the approximate posterior to match a prescribed prior, commonly $\mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\mathcal{L}_{\mathrm{KL}} = \mathbb{E}_{\mathbf{x}} \left[D_{\mathrm{KL}} \left(q_\theta(\mathbf{z} \mid \mathbf{x}) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I}) \right) \right]. \tag{51}$$

This regularization introduces a trade-off between reconstruction and prior alignment. LatentTime conditions predictions on observed histories and does not require sampling context representations from a predefined prior. We therefore prioritize reconstruction fidelity and temporal structure through reconstruction losses and latent spectral alignment. Predictive uncertainty is learned through quantile regression on decoded forecasts, so probabilistic forecasting does not require a stochastic context encoder.

Empirical evidence. The RQ2 ablation in Figure 7 supports this choice. With KL weight 10^{-6}, the evaluated VAE variant yields reconstruction MSE of 0.018556, compared with 0.016924 for the final TS-AE at context length 2048. Under 10% missing inputs, the corresponding errors are 0.02158 and 0.01940. Downstream forecasting also favors TS-AE, with MASE/CRPS of 0.724/0.488 versus 0.736/0.503 for the VAE variant. These results support the deterministic design for our forecasting pipeline under the evaluated configuration.

C.7 Normalized Spectral Entropy and Forecastability

Following the entropy-based notion of forecastability introduced by ForeCA (Goerg, 2013), we use normalized spectral entropy to characterize the spectral regularity of the input context, its average-pooled counterpart, and the TS-AE latent representation. The following specifies the estimator used in our experiments. The full results of the forecastability in ETTh1, ETTh2, PEMS03, Electricity, Exchange, Weather datasets (Zhou et al., 2021) are shown in Figure 9.

Compared Representations. For each univariate window $\mathbf{x} \in \mathbb{R}^L$, we evaluate the context itself, its non-overlapping average-pooled sequence $\mathbf{x}^{\downarrow} \in \mathbb{R}^{L/c}$, and the corresponding deterministic TS-AE representation $\mathbf{z} \in \mathbb{R}^{L/c}$. Average pooling is defined as

$$x_j^{\downarrow} = \frac{1}{c} \sum_{r=1}^{c} x_{c(j-1)+r}, \qquad j = 1, \ldots, L/c. \tag{52}$$

We use $c = 2$, require L to be divisible by c, and evaluate windows with $L \geq 4c$. Latent outputs are truncated to L/c positions.

The context and pooled sequence are evaluated in the same space. When inverse normalization is disabled, $\mathbf{x}$ denotes the encoder input after preprocessing, including instance normalization and the optional arcsinh transformation; otherwise, it is expressed in the original observation space. Latent representations are never inverse-normalized. No additional standardization is applied when computing spectral entropy, and non-finite entries are replaced with zero before spectral estimation.

Spectral Estimation and Normalization. For each sequence $\mathbf{y} \in \mathbb{R}^T$, we estimate its one-sided Welch power spectrum using a sampling rate of 1, a segment length of $n = \min(256, T)$, and an overlap of $\lfloor n/2 \rfloor$. Each segment is demeaned and multiplied by a periodic Hann window. Segment periodograms are averaged using the `scaling="spectrum"` convention.

After removing the zero-frequency component, we retain the nonnegative powers $\mathbf{s} = (s_1, \ldots, s_m)$ at positive frequencies, where $m = \lfloor n/2 \rfloor$. Frequencies are measured in cycles per sample and lie in $(0, 1/2]$. We mirror these powers to construct a symmetric discrete spectral mass vector:

$$\widetilde{\mathbf{s}} = (s_m, \ldots, s_1, s_1, \ldots, s_m). \tag{53}$$

The initial probability masses are

$$p_i^{(0)} = \frac{\widetilde{s}_i}{\sum_{j=1}^{2m} \widetilde{s}_j}, \qquad i = 1, \ldots, 2m. \tag{54}$$

If the total spectral power is zero, we instead use the uniform distribution $p_i^{(0)} = 1/(2m)$. For numerical stability, we mix these probabilities with a uniform prior:

$$p_i = (1 - \lambda) p_i^{(0)} + \frac{\lambda}{2m}, \qquad \lambda = 10^{-3}. \tag{55}$$

Table 21: Complete spectral forecastability results for original contexts, $2\times$ average-pooled sequences, and TS-AE latent representations across datasets and context lengths. Scores are defined as $\Omega = 1 - H$, where H denotes normalized spectral entropy. Higher scores indicate greater spectral regularity Goerg (2013).

Dataset	Context Length	Context	Downsampling	Latent
ETTh1	512	0.3413	0.3544	0.3550
	1024	0.3335	0.3258	0.3260
	1536	0.3317	0.3264	0.3315
	2048	0.3276	0.3180	0.3203
ETTm1	512	0.4403	0.4270	0.4391
	1024	0.4384	0.4095	0.4167
	1536	0.4403	0.4085	0.4307
	2048	0.4392	0.4079	0.4201
PEMS03	512	0.6528	0.6127	0.6100
	1024	0.6474	0.5991	0.6060
	1536	0.6501	0.6002	0.6073
	2048	0.6415	0.5885	0.6148
Electricity	512	0.4794	0.4941	0.5025
	1024	0.4690	0.4760	0.4852
	1536	0.4856	0.4926	0.4990
	2048	0.4797	0.4857	0.4912
Exchange	512	0.5675	0.5719	0.5689
	1024	0.5753	0.5878	0.5853
	1536	0.5643	0.5905	0.5856
	2048	0.5880	0.6109	0.6088
Weather	512	0.5449	0.5107	0.5199
	1024	0.5242	0.4828	0.4892
	1536	0.5348	0.4833	0.5088
	2048	0.5421	0.4916	0.5167

Probabilities no greater than 10^{-9} are subsequently set to zero, and the remaining masses are renormalized.

Entropy and Forecastability. Normalized spectral entropy is computed as

$$H(\mathbf{y}) = -\frac{\sum_{i=1}^{2m} p_i \ln p_i}{\ln(2m)}, \tag{56}$$

with $0 \ln 0 = 0$. This dimensionless quantity lies in $[0, 1]$: a uniform spectral distribution has entropy 1, while a more concentrated distribution has lower entropy. Forecastability is defined as its complement:

$$\Omega(\mathbf{y}) = 1 - H(\mathbf{y}) \in [0, 1]. \tag{57}$$

We report Ω without converting it to a percentage. For zero-power inputs, $H = 1$ and $\Omega = 0$ follow from the uniform fallback and represent a numerical convention.

Dataset-level scores are arithmetic means over evaluated windows. Because sequence length affects Welch averaging and, for short sequences, frequency resolution, we emphasize comparisons between latent and average-pooled sequences with identical compressed lengths. The uncompressed context provides a descriptive reference. Lower spectral entropy indicates greater spectral concentration, but does not by itself establish preservation of predictive information; this is assessed separately through forecasting experiments.

C.8 Evaluation of Temporal Structure and Predictive Utility

Qualitative visualization. We extract length-512 windows and obtain their corresponding length-256 latent representations using the frozen TS-AE encoder. The dataset-level visualization compares windows from selected GIFT-Eval datasets, while the timestamp-level visualization compares windows sampled at different temporal positions within a selected Electricity channel. Dataset identities and temporal order are used to interpret the resulting t-SNE embeddings. We examine whether dataset-level grouping and local temporal continuity remain visible after compression. These visualizations provide qualitative evidence of representation structure; they do not establish equality between the original and latent distributions.

Spectral forecastability. We evaluate ETTh1, ETTm1, PEMS03, Electricity, Exchange, and Weather. Multivariate records are processed channel independently, using windows of lengths $L \in \{512, 1024, 1536, 2048\}$ with 50% overlap. For every window, we compute normalized spectral entropy and forecastability for the uncompressed context, its non-overlapping $2\times$ average-pooled sequence, and its TS-AE latent representation, following Appendix C.7. Scores are averaged over windows separately for each dataset and context length. The latent and pooled sequences have identical lengths, making them the primary comparison; the uncompressed context serves as a descriptive reference. Higher forecastability indicates greater spectral concentration and is interpreted together with the downstream forecasting results.

Forecasting windows and sampling. The forecasting evaluation uses the same six datasets, with context lengths $L \in \{1024, 2048\}$ and prediction lengths $H \in \{32, 48, 64\}$. We extract channel-independent history–future pairs with stride H, producing non-overlapping future segments. For each dataset and context–horizon configuration, we evaluate up to 32 windows with ARIMA (and, 2007) and up to 256 windows with Chronos-2-Synth (Ansari et al., 2025). The smaller ARIMA sample budget reflects the cost of model-order selection. Within each predictor, all three representations are evaluated on the same selected windows and future targets.

Forecasting from different representations. Let $\mathbf{x} \in \mathbb{R}^L$ denote the history and $\mathbf{y} \in \mathbb{R}^H$ the corresponding future. Let $\mathcal{F}_h$ denote a forecasting procedure producing h future values, A_2 denote non-overlapping average pooling with kernel size and stride two, and U_2 denote upsampling by repeating each predicted value twice. The three forecasting routes are

$$\begin{aligned} \widehat{\mathbf{y}}^{\mathrm{ctx}} &= \mathcal{F}_H(\mathbf{x}), \\ \widehat{\mathbf{y}}^{\mathrm{pool}} &= U_2\big(\mathcal{F}_{H/2}(A_2(\mathbf{x}))\big), \\ \mathbf{z} &= E_\theta(\mathbf{x}), \\ \widehat{\mathbf{y}}^{\mathrm{lat}} &= \mathrm{Tail}_H\left[D_\phi\left([\mathbf{z}; \mathcal{F}_{H/2}(\mathbf{z})]\right)\right], \end{aligned} \tag{58}$$

where $[\cdot;\cdot]$ denotes temporal concatenation and Tail_H selects the final H decoded positions. Concatenating the historical latent sequence with its predicted continuation provides the decoder with temporal context when restoring the forecast resolution. Only historical observations and predicted latent values enter this decoding step. Both the TS-AE encoder and decoder remain frozen throughout the evaluation. The full results are shown in Table 22 and Table 23.

Forecasting probes. ARIMA provides a classical statistical forecasting probe. For each history window, we select its order (p, d, q) by AIC using the uncompressed context, then reuse that order for the pooled and latent representations. Model coefficients are fitted separately to each representation, and failed fits fall back to naive forecasting. This protocol controls model order across representations, although it does not independently optimize the order for each compressed sequence.

Chronos-2-Synth provides a pre-trained neural forecasting probe. We use the `autogluon/chronos-2-synth` checkpoint, which was pre-trained exclusively on synthetic univariate and multivariate data (Ansari et al., 2025). The same checkpoint is applied to all three representations without fine-tuning, using its median output for point forecasting. Thus, this experiment evaluates compatibility with an independent pre-trained predictor, without adapting that predictor to the TS-AE latent space.

Table 22: Complete forecasting results of ARIMA (and, 2007) using original contexts, 2× average-pooled sequences, and TS-AE latent representations across datasets and context–horizon configurations. All forecasts are evaluated at the original temporal resolution.

Dataset	Methods	Metircs	Context		Downsampling		**Latent**	
	Context	Prediction	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	1024	32	0.3797	0.4813	0.3631	0.4745	0.3765	0.4879
		48	0.4178	0.5192	0.4139	0.5189	0.3990	0.5094
		64	0.4608	0.5363	0.4744	0.5491	0.4726	0.5394
	2048	32	0.4339	0.5241	0.3864	0.4966	0.4139	0.5093
		48	0.5751	0.5621	0.6139	0.5918	0.5816	0.5717
		64	0.3824	0.4765	0.4043	0.5060	0.4529	0.5282
ETTm1	1024	32	0.3868	0.4760	0.4365	0.5026	0.4069	0.4980
		48	0.5506	0.5542	0.5178	0.5520	0.5077	0.5457
		64	0.6608	0.6590	0.6701	0.6623	0.6679	0.6636
	2048	32	0.3953	0.4652	0.3952	0.4668	0.3994	0.4707
		48	0.5313	0.5627	0.5488	0.5645	0.5493	0.5703
		64	0.5193	0.5597	0.5303	0.5635	0.5748	0.5901
PEMS03	1024	32	0.1870	0.3164	0.1675	0.3059	0.2547	0.3379
		48	0.2592	0.3791	0.2757	0.4000	0.2575	0.4002
		64	0.2890	0.4050	0.3078	0.4149	0.2722	0.4061
	2048	32	0.2833	0.3792	0.2962	0.3826	0.3010	0.4018
		48	0.2323	0.3619	0.2657	0.3837	0.2412	0.3698
		64	0.2566	0.3623	0.2678	0.3759	0.2845	0.3902
Electricity	1024	32	0.3372	0.4755	0.3689	0.4975	0.3376	0.4690
		48	0.4387	0.5378	0.4687	0.5744	0.4854	0.5834
		64	0.5517	0.5966	0.5288	0.5950	0.4916	0.5742
	2048	32	0.3616	0.4898	0.3957	0.5018	0.3736	0.4836
		48	0.3781	0.4874	0.4166	0.5188	0.3727	0.4844
		64	0.4541	0.5484	0.4955	0.5798	0.5032	0.5761
Exchange	1024	32	0.0782	0.1614	0.0781	0.1573	0.0798	0.1639
		48	0.1182	0.2025	0.1187	0.2016	0.1261	0.2087
		64	0.1013	0.2262	0.0966	0.2180	0.1142	0.2636
	2048	32	0.0352	0.1200	0.0327	0.1145	0.0349	0.1134
		48	0.0313	0.1188	0.0317	0.1187	0.0369	0.1314
		64	0.0487	0.1606	0.0478	0.1598	0.0483	0.1616
Weather	1024	32	0.3155	0.3514	0.2656	0.3097	0.2531	0.3069
		48	0.2003	0.2904	0.2079	0.3109	0.2050	0.3011
		64	0.5591	0.4955	0.5574	0.4935	0.5579	0.4939
	2048	32	0.1524	0.2595	0.1565	0.2582	0.1514	0.2585
		48	0.1731	0.2793	0.2057	0.3025	0.1866	0.2903
		64	0.4793	0.4847	0.4306	0.4522	0.4257	0.4505

Evaluation and interpretation. All forecasts are compared with the same future observations at the original temporal resolution. MSE and MAE are computed in the instance-normalized space, consistently with the reconstruction evaluation. The uncompressed-context route serves as a reference baseline, rather than a guaranteed upper bound on forecasting performance.

The latent route evaluates the combined effects of compression, predictor compatibility, and learned decoding. Lower error than the pooling route demonstrates an advantage of this complete forecasting pipeline, but does not isolate the contribution of the encoder from that of the decoder. Conversely, higher error can reflect information loss, predictor mismatch, or decoding error. The different outcomes obtained with ARIMA and Chronos-2-Synth therefore indicate that predictive utility depends on both the representation and the forecasting procedure.

Table 23: Complete forecasting results of Chronos-2-Synth (Ansari et al., 2025) using original contexts, 2× average-pooled sequences, and TS-AE latent representations across datasets and context–horizon configurations. The predictor is applied without fine-tuning, and all forecasts are evaluated at the original temporal resolution.

Dataset	Methods	Metircs	Context		Downsampling		**Latent**	
	Context	Prediction	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	1024	32	0.2540	0.3511	0.2692	0.3669	0.2695	0.3674
		48	0.3330	0.3991	0.3526	0.4171	0.3529	0.4202
		64	0.3129	0.3928	0.3265	0.4099	0.3223	0.4096
	2048	32	0.2380	0.3258	0.2502	0.3441	0.2497	0.3478
		48	0.3241	0.3798	0.3329	0.3912	0.3292	0.3936
		64	0.3304	0.3934	0.3309	0.3989	0.3303	0.3998
ETTm1	1024	32	0.2149	0.3189	0.2129	0.3185	0.2159	0.3278
		48	0.3018	0.3732	0.3147	0.3787	0.3113	0.3841
		64	0.3223	0.3839	0.3192	0.3883	0.3166	0.3924
	2048	32	0.2241	0.3090	0.2069	0.3027	0.2116	0.3134
		48	0.3002	0.3615	0.2885	0.3599	0.2908	0.3653
		64	0.2602	0.3486	0.2449	0.3451	0.2487	0.3524
PEMS03	1024	32	0.0630	0.1629	0.0692	0.1716	0.0653	0.1646
		48	0.0765	0.1765	0.0837	0.1829	0.0807	0.1806
		64	0.1328	0.2251	0.1354	0.2287	0.1319	0.2220
	2048	32	0.0343	0.1283	0.0347	0.1303	0.0346	0.1311
		48	0.0720	0.1618	0.0779	0.1667	0.0770	0.1664
		64	0.0595	0.1607	0.0647	0.1690	0.0617	0.1636
Electricity	1024	32	0.0761	0.1624	0.1137	0.2187	0.0985	0.1998
		48	0.0703	0.1666	0.1140	0.2252	0.0968	0.2033
		64	0.0799	0.1761	0.1305	0.2368	0.1158	0.2161
	2048	32	0.0703	0.1584	0.0981	0.2075	0.0828	0.1889
		48	0.0644	0.1567	0.0973	0.2089	0.0811	0.1879
		64	0.1028	0.1802	0.1231	0.2239	0.1100	0.2047
Exchange	1024	32	0.0761	0.1624	0.1137	0.2187	0.0985	0.1998
		48	0.0703	0.1666	0.1140	0.2252	0.0968	0.2033
		64	0.0799	0.1761	0.1305	0.2368	0.1158	0.2161
	2048	32	0.0703	0.1584	0.0981	0.2075	0.0828	0.1889
		48	0.0644	0.1567	0.0973	0.2089	0.0811	0.1879
		64	0.1028	0.1802	0.1231	0.2239	0.1100	0.2047
Weather	1024	32	0.1582	0.2335	0.1816	0.2535	0.1670	0.2478
		48	0.2137	0.2641	0.2228	0.2799	0.2184	0.2697
		64	0.4560	0.3403	0.4917	0.3632	0.4765	0.3554
	2048	32	0.1471	0.2030	0.1518	0.2117	0.1493	0.2048
		48	0.1747	0.2348	0.1874	0.2460	0.1766	0.2355
		64	0.1897	0.2505	0.2005	0.2637	0.2000	0.2636

C.9 Inter-Channel Correlation Preservation

We evaluate whether channel-independent compression with TS-AE preserves the inter-channel Pearson correlation structure of multivariate time series.

Compared Representations. We extract multivariate windows $\mathbf{X} \in \mathbb{R}^{C \times L}$, where $C \geq 2$, using a default stride of L. All channels remain temporally aligned within each window. We compare the input context $\mathbf{X}$, its average-pooled representation $\mathbf{A} \in \mathbb{R}^{C \times L/c}$, and the TS-AE latent representation

$\mathbf{Z} \in \mathbb{R}^{C \times L/c}$. Average pooling is applied independently to each channel:

$$A_{i,j} = \frac{1}{c} \sum_{r=1}^{c} X_{i,c(j-1)+r}, \qquad j = 1, \ldots, L/c. \tag{59}$$

We use $c = 2$ and require L to be divisible by c. TS-AE encodes each channel independently, and the resulting latent sequences retain their original channel ordering.

When inverse normalization is disabled, $\mathbf{X}$ denotes the instance-normalized context; otherwise, it is expressed in the original observation space. Latent representations are evaluated without inverse normalization. Pearson correlation is invariant to channel-wise shifts and positive rescaling. If nonlinear preprocessing is applied, correlation preservation is assessed relative to the processed context.

Inter-Channel Correlation Matrices. For a representation $\mathbf{Y} \in \mathbb{R}^{C \times T'}$, let $\mathcal{V}_i$ denote the finite-valued time indices of channel i, and let $\mathcal{V}_{ij} = \mathcal{V}_i \cap \mathcal{V}_j$. The channel mean μ_i is computed over $\mathcal{V}_i$. Following the finite-value handling used in our implementation, we calculate

$$R_{ij}(\mathbf{Y}) = \frac{\sum_{t \in \mathcal{V}_{ij}} (Y_{i,t} - \mu_i)(Y_{j,t} - \mu_j)}{\sqrt{\sum_{t \in \mathcal{V}_i} (Y_{i,t} - \mu_i)^2} \sqrt{\sum_{t \in \mathcal{V}_j} (Y_{j,t} - \mu_j)^2}}. \tag{60}$$

For missing observations, this formulation is equivalent to channel-wise mean imputation followed by Pearson correlation. Off-diagonal entries involving channels with fewer than two finite observations or negligible variance are marked as undefined. Diagonal entries are set to one and excluded from the evaluation.

Each window therefore yields three correlation matrices:

$$\mathbf{R}^X = R(\mathbf{X}), \qquad \mathbf{R}^A = R(\mathbf{A}), \qquad \mathbf{R}^Z = R(\mathbf{Z}). \tag{61}$$

Correlation Mean Absolute Error. We measure the change in inter-channel correlation using Correlation Mean Absolute Error (CorrMAE). For $V \in \{A, Z\}$, let $\mathcal{P}_V$ contain the channel pairs (i, j) with $i < j$ for which both R^V_{ij} and R^X_{ij} are finite. CorrMAE is defined as

$$\mathrm{CorrMAE}(V; X) = \frac{1}{|\mathcal{P}_V|} \sum_{(i,j) \in \mathcal{P}_V} \left| R^V_{ij} - R^X_{ij} \right|. \tag{62}$$

When all channel pairs are valid, $|\mathcal{P}_V| = C(C-1)/2$. Lower values indicate better preservation of inter-channel linear correlation. CorrMAE measures absolute deviations in correlation coefficients, rather than a percentage of correlation information lost.

For reference, we also report the mean absolute correlation strength of the input:

$$\overline{|R^X|} = \frac{1}{|\mathcal{P}_X|} \sum_{(i,j) \in \mathcal{P}_X} |R^X_{ij}|, \tag{63}$$

where $\mathcal{P}_X$ contains the valid upper-triangular entries of $\mathbf{R}^X$.

Aggregation and Interpretation. Dataset-level scores are arithmetic means of the valid window-level scores. Windows containing no valid channel pairs are excluded. Correlation heatmaps show element-wise means across windows, omitting undefined entries. Error heatmaps average the absolute differences $|\mathbf{R}^A - \mathbf{R}^X|$ and $|\mathbf{R}^Z - \mathbf{R}^X|$ computed within each window.

Comparing latent representations with average pooling assesses correlation preservation under the same temporal compression ratio. This metric specifically evaluates zero-lag linear relationships; downstream multivariate forecasting performance requires separate evaluation.

We present the inter-channel correlation matrices of datasets such as ETTh1 and ETTm1 in matrix form in Figures 12–17.

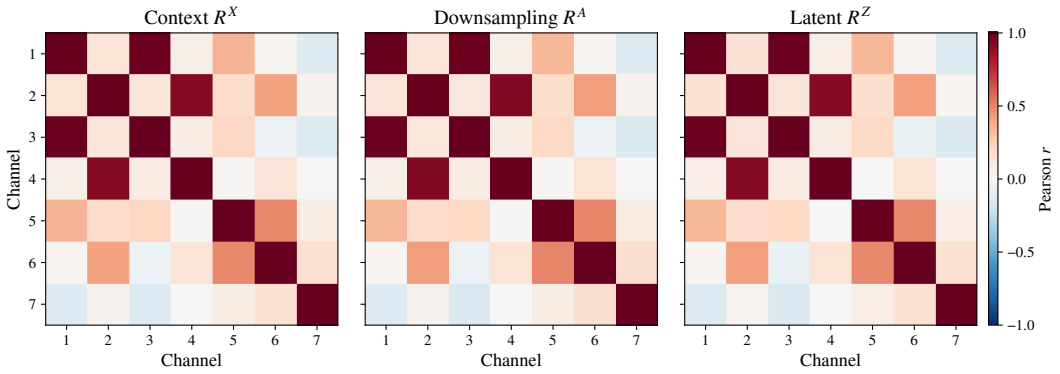


Figure 12: The inter-channel correlation matrices of ETTh1 dataset.

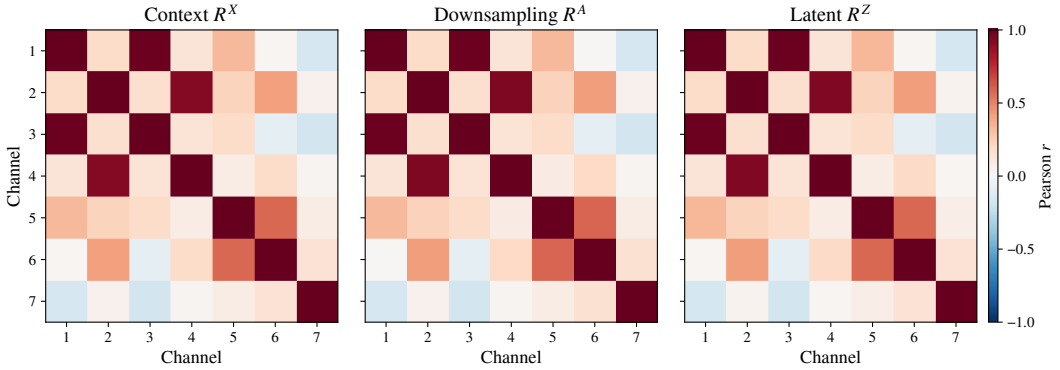


Figure 13: The inter-channel correlation matrices of ETTm1 dataset.

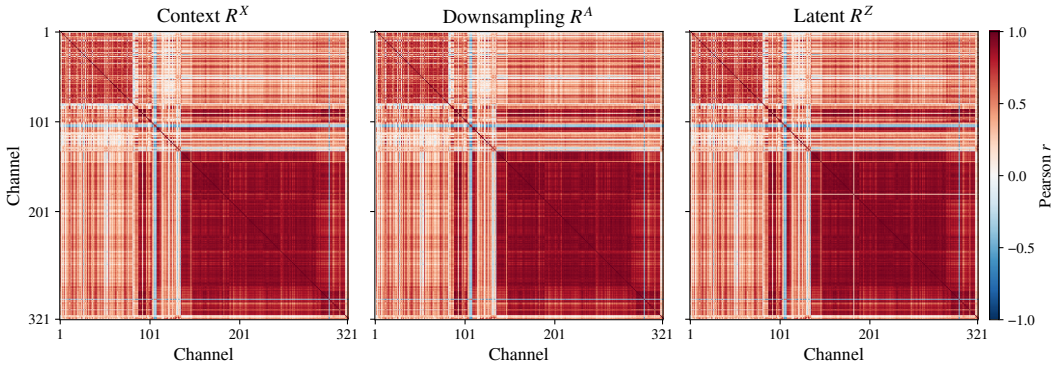


Figure 14: The inter-channel correlation matrices of Electricity dataset.

C.10 Training Configurations for the Length-Aware Bucket Strategy

Experimental scope. We study the short-context pre-training stage of PatchTST, whose standard training budget is 500K optimizer updates on eight GPUs. The eight buckets are $[16, 32]$, $[33, 64]$, $[65, 128]$, $[129, 256]$, $[257, 512]$, $[513, 768]$, $[769, 1024]$, and $[1025, 1280]$. Models are evaluated on the GIFT-Eval test splits after this stage, without subsequent long-context post-training or latent forecasting adaptation.

Configuration notation. Each identifier follows `S{S}_G{G}_{B}_{GA}`, where S denotes optimizer updates in thousands and G is the number of GPUs. The batching and sampling scheme B is `E` for uniform bucket sampling, `I` for sampling biased toward shorter buckets, and `N` for train-

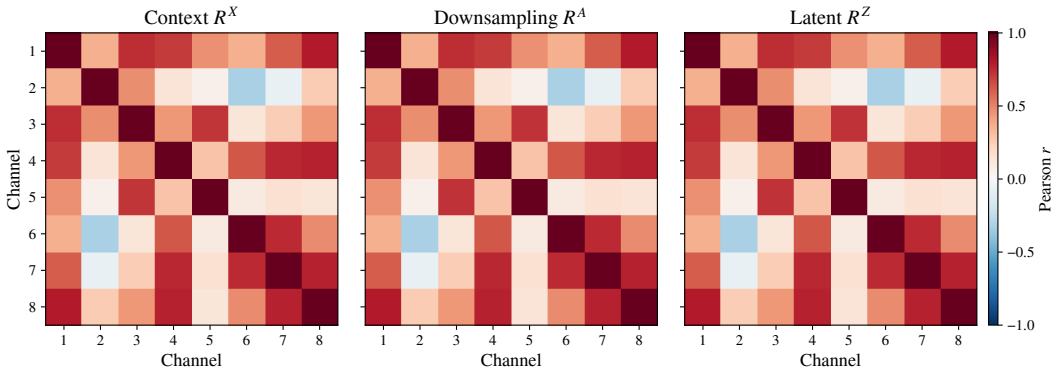


Figure 15: The inter-channel correlation matrices of Exchange dataset.

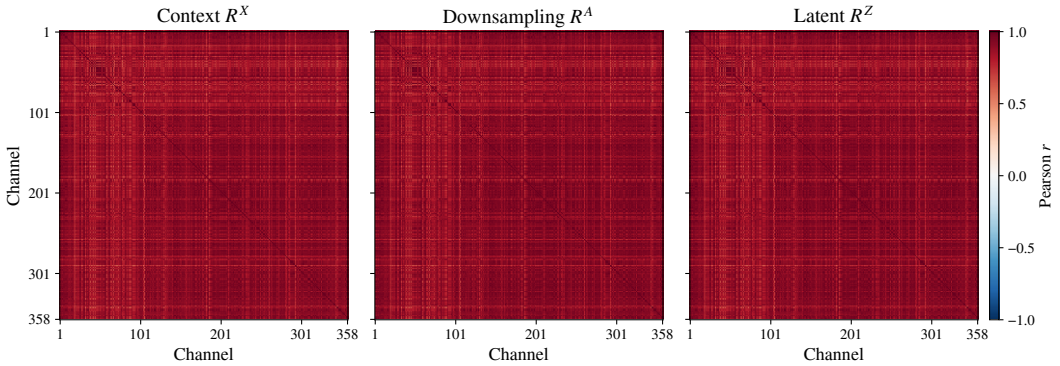


Figure 16: The inter-channel correlation matrices of PEMS03 dataset.

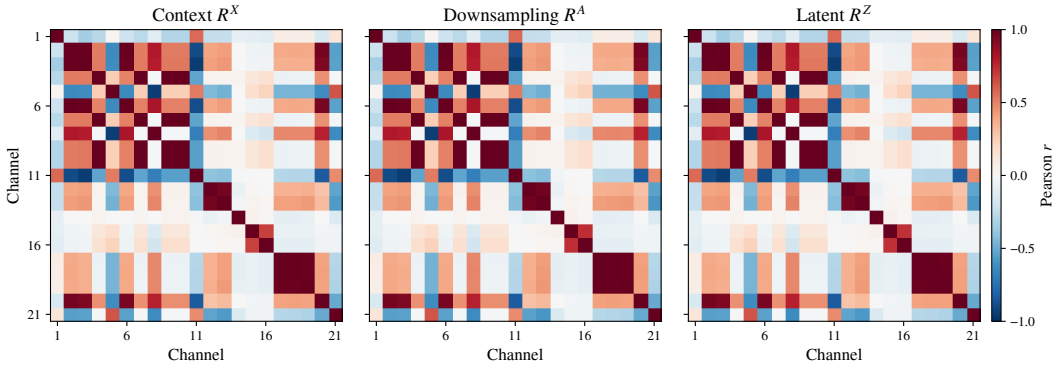


Figure 17: The inter-channel correlation matrices of Weather dataset.

ing without bucketing. `GA` denotes two microbatches accumulated per GPU before each optimizer update, whereas `No` denotes one microbatch per GPU per update. Thus, the accumulation factor is $A = 2$ for `GA` and $A = 1$ otherwise.

With a local microbatch size of $b = 512$, the effective batch size and cumulative number of sampled windows are

$$b_{\text{eff}} = bGA, \qquad N_{\text{draw}} = 10^3 SbGA. \tag{64}$$

Here, N_{draw} counts sampled windows, including repeated draws, rather than distinct training examples.

Comparison groups. Table 24 supports three comparisons:

Table 24: PatchTST short-context pre-training configurations and forecasting results for RQ5. S is measured in thousands of optimizer updates, A is the gradient accumulation factor, and b_{eff} is the effective global batch size. Lower MASE and CRPS indicate better performance.

Configuration	S	G	B	A	b_{eff}	MASE	CRPS
S500_G8_E_No	500	8	E	1	4096	0.743	0.506
S500_G8_I_No	500	8	I	1	4096	**0.738**	**0.502**
S500_G8_N_No	500	8	N	1	4096	0.751	0.511
S500_G1_E_No	500	1	E	1	512	0.787	0.541
S4000_G1_E_No	4000	1	E	1	512	0.758	0.525
S500_G4_E_GA	500	4	E	2	4096	0.745	0.507

- **Bucket sampling.** The three S500_G8 configurations compare uniform sampling, short-bucket-biased sampling, and unbucketed training at the same update count and effective batch size.
- **Parallelism and training exposure.** S500_G1_E_No uses one-eighth of the reference effective batch size and cumulative sampled-window count. Extending it to S4000_G1_E_No matches the reference cumulative count while retaining the smaller batch size and using eight times as many optimizer updates.
- **Gradient accumulation.** S500_G4_E_GA matches the effective batch size, update count, and cumulative sampled-window count of S500_G8_E_No using four GPUs. Matching the number of independent bucket draws per update additionally requires sampling a bucket independently for each accumulated microbatch.

C.11 Analysis of Training Configurations

Table 24 and Figure 10b report forecasting results after PatchTST's short-context pre-training, before long-context post-training or latent adaptation. Each GPU processes 512 windows per microbatch. We use S500_G8_E_No as the reference configuration and examine the following comparisons.

Effect of length-aware bucketing. With eight GPUs and 500K optimizer updates, uniform bucket sampling (S500_G8_E_No) achieves lower MASE and CRPS than training without bucketing (S500_G8_N_No), reducing the two metrics from 0.751 to 0.743 and from 0.511 to 0.506, respectively. Thus, although each bucketed microbatch contains a restricted range of context lengths, this organization does not impair aggregate forecasting accuracy in the evaluated setting. Together with the reduction in padding, these results support length-aware bucketing for variable-length pre-training.

Effect of bucket sampling probabilities. Increasing the sampling frequency of shorter-context buckets (S500_G8_I_No) further improves MASE to 0.738 and CRPS to 0.502, producing the lowest errors among the six configurations. Relative to uniform sampling, these correspond to reductions of approximately 0.67% and 0.79%. This modest improvement suggests that explicitly adjusting exposure across context-length ranges can benefit forecasting. However, the aggregate results do not identify which context lengths benefit most or establish an optimal sampling bias.

Reducing GPU parallelism at a fixed update count. Using one GPU for 500K updates (S500_G1_E_No) increases MASE to 0.787 and CRPS to 0.541. With the same microbatch size per GPU, this configuration reduces the effective batch size from 4096 to 512 and processes one-eighth as many sampled windows. Moreover, each update receives samples from only one bucket, whereas eight-GPU training can combine gradients from several independently sampled buckets. Consequently, this comparison changes effective batch size, cumulative sample exposure, and potential length diversity per update simultaneously; their individual contributions cannot be isolated.

Extending single-GPU training. Increasing single-GPU training to 4000K updates (S4000_G1_E_No) improves MASE to 0.758 and CRPS to 0.525. This configuration matches the

cumulative number of sampled windows in the eight-GPU reference, including repeated samples, while retaining the smaller effective batch size. The remaining errors are approximately 2.02% and 3.75% higher than the reference, respectively. Therefore, additional updates recover part of the performance gap, but matching cumulative sample exposure alone does not recover the reference accuracy under the evaluated schedule. The different batch sizes and optimization trajectories remain possible contributors to this gap.

Gradient accumulation with fewer GPUs. Accumulating two microbatches on each of four GPUs (`S500_G4_E_GA`) restores the effective batch size to 4096. At 500K optimizer updates, it also matches the cumulative sample exposure of the eight-GPU reference. Its MASE of 0.745 and CRPS of 0.507 are only approximately 0.27% and 0.20% higher, respectively. These results indicate that gradient accumulation can maintain comparable forecasting accuracy with fewer GPUs. If buckets are independently resampled for each accumulated microbatch, this configuration also combines eight bucket draws per update, providing the same number of opportunities to mix context-length ranges as eight-GPU training. The accuracy comparison does not establish equivalent training throughput, since accumulation requires additional sequential microbatch processing.

D Full Results of Forecasting and Reconstruction

D.1 Full Results of Time Series Forecasting Benchmark Gift-Eval

The full results of Gift-Eval time series forecasting benchmark are shown in Table 25.

Table 25: Detailed forecasting results on GIFT-Eval. PatchTST-WB denotes PatchTST trained with the Length-Aware Bucket strategy. Lower MASE and CRPS indicate better performance.

Dataset	PatchTST		PatchTST-WB		LatentTime	
	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓
bitbrains_fast_storage/5T/long	0.8453	0.5095	0.8102	0.3958	0.8155	0.4057
bitbrains_fast_storage/5T/medium	0.9509	0.5041	0.9292	0.4107	0.9364	0.4136
bitbrains_fast_storage/5T/short	0.6605	0.3608	0.6589	0.3392	0.6711	0.3456
bitbrains_fast_storage/H/short	1.0744	0.5555	1.1383	0.5044	1.1881	0.5081
bitbrains_rnd/5T/long	3.2752	0.5640	3.2265	0.3770	3.2374	0.3932
bitbrains_rnd/5T/medium	4.3557	0.5304	4.3401	0.4324	4.3494	0.4640
bitbrains_rnd/5T/short	1.6289	0.3599	1.6236	0.3086	1.6229	0.3098
bitbrains_rnd/H/short	5.7718	0.6705	5.7787	0.5887	5.7789	0.6152
bizitobs_application/10S/long	3.1076	0.0512	3.1963	0.0544	3.1674	0.0550
bizitobs_application/10S/medium	2.1199	0.0318	2.5090	0.0425	2.4951	0.0429
bizitobs_application/10S/short	1.2085	0.0117	1.3468	0.0130	1.3359	0.0134
bizitobs_l2c/5T/long	0.7553	0.3377	0.7854	0.3555	0.8669	0.4047
bizitobs_l2c/5T/medium	0.6258	0.2601	0.5991	0.2540	0.6410	0.2744
bizitobs_l2c/5T/short	0.2919	0.0783	0.3104	0.0832	0.3174	0.0852
bizitobs_l2c/H/long	0.5584	0.2567	0.5639	0.2611	0.5681	0.2623
bizitobs_l2c/H/medium	0.5064	0.2400	0.5174	0.2423	0.5243	0.2438
bizitobs_l2c/H/short	0.4733	0.2003	0.5128	0.2161	0.5180	0.2185
bizitobs_service/10S/long	1.3327	0.0523	1.3088	0.0536	1.3165	0.0539
bizitobs_service/10S/medium	1.0212	0.0225	1.0848	0.0270	1.1110	0.0282
bizitobs_service/10S/short	0.7689	0.0117	0.7772	0.0132	0.7861	0.0131
car_parts/M/short	0.8420	0.9710	0.8524	0.9904	0.8567	0.9904
covid_deaths/D/short	32.2119	0.0370	31.7297	0.0317	33.4614	0.0324
electricity/15T/long	0.8748	0.0749	0.8634	0.0729	0.8643	0.0726
electricity/15T/medium	0.8229	0.0738	0.8147	0.0729	0.8159	0.0728
electricity/15T/short	0.9419	0.0804	0.9168	0.0788	0.9159	0.0788

Continued on the next page.

Table 25: Detailed forecasting results on GIFT-Eval (continued).

Dataset	PatchTST		PatchTST-WB		LatentTime	
	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓
electricity/D/short	1.3821	0.0543	1.3708	0.0531	1.3706	0.0529
electricity/H/long	1.2214	0.0915	1.1857	0.0845	1.1882	0.0851
electricity/H/medium	1.0927	0.0789	1.0694	0.0740	1.0708	0.0744
electricity/H/short	1.0013	0.0731	0.9457	0.0674	0.9425	0.0676
electricity/W/short	1.5171	0.0529	1.5996	0.0496	1.5807	0.0491
ett1/15T/long	1.0542	0.2554	1.0442	0.2476	1.0496	0.2486
ett1/15T/medium	1.0344	0.2503	1.0422	0.2524	1.0381	0.2514
ett1/15T/short	0.7024	0.1659	0.7071	0.1629	0.7124	0.1648
ett1/D/short	1.7401	0.2981	1.8186	0.3183	1.8095	0.3184
ett1/H/long	1.4122	0.2958	1.4122	0.2793	1.3986	0.2785
ett1/H/medium	1.3466	0.2757	1.2924	0.2590	1.3025	0.2603
ett1/H/short	0.8279	0.1804	0.8365	0.1813	0.8348	0.1810
ett1/W/short	1.5806	0.2657	1.5688	0.2617	1.5940	0.2649
ett2/15T/long	0.9058	0.0938	0.9180	0.0936	0.9153	0.0934
ett2/15T/medium	0.8729	0.0897	0.8777	0.0901	0.8784	0.0902
ett2/15T/short	0.7538	0.0653	0.7473	0.0648	0.7503	0.0650
ett2/D/short	1.7145	0.1065	1.4743	0.0988	1.5102	0.0995
ett2/H/long	1.2156	0.1127	1.0374	0.1008	1.0338	0.1003
ett2/H/medium	1.1677	0.1130	1.0444	0.1033	1.0415	0.1033
ett2/H/short	0.7510	0.0653	0.7416	0.0649	0.7460	0.0653
ett2/W/short	0.8647	0.0887	1.0398	0.0943	1.0691	0.0955
hierarchical_sales/D/short	0.7490	0.5805	0.7457	0.5729	0.7450	0.5729
hierarchical_sales/W/short	0.7200	0.3529	0.7302	0.3660	0.7291	0.3636
hospital/M/short	0.7573	0.0503	0.7611	0.0500	0.7641	0.0504
jena_weather/10T/long	0.6278	0.0511	0.6172	0.0486	0.6149	0.0484
jena_weather/10T/medium	0.5966	0.0508	0.6024	0.0488	0.6028	0.0485
jena_weather/10T/short	0.2862	0.0306	0.2916	0.0288	0.2920	0.0288
jena_weather/D/short	1.0816	0.0505	1.0232	0.0516	1.0290	0.0517
jena_weather/H/long	1.1101	0.0617	0.9590	0.0552	0.9635	0.0556
jena_weather/H/medium	0.7823	0.0528	0.7465	0.0499	0.7457	0.0503
jena_weather/H/short	0.5330	0.0424	0.5261	0.0416	0.5268	0.0417
kdd_cup_2018/D/short	1.2121	0.3796	1.2037	0.3792	1.2013	0.3790
kdd_cup_2018/H/long	0.9769	0.4336	0.9340	0.4111	0.9366	0.4119
kdd_cup_2018/H/medium	1.0010	0.4167	0.9543	0.3962	0.9618	0.3975
kdd_cup_2018/H/short	0.9162	0.3760	0.8652	0.3529	0.8676	0.3537
loop_seattle/5T/long	0.8796	0.0778	0.8011	0.0708	0.8039	0.0711
loop_seattle/5T/medium	0.8182	0.0725	0.7576	0.0670	0.7586	0.0671
loop_seattle/5T/short	0.5751	0.0488	0.5563	0.0472	0.5650	0.0479
loop_seattle/D/short	0.8851	0.0431	0.8897	0.0433	0.8895	0.0434
loop_seattle/H/long	0.8904	0.0619	0.8865	0.0621	0.8897	0.0625
loop_seattle/H/medium	0.9299	0.0652	0.9297	0.0657	0.9342	0.0662
loop_seattle/H/short	0.8442	0.0596	0.8383	0.0593	0.8418	0.0597
m4_daily/D/short	7.9845	0.0595	8.7924	0.0668	6.9591	0.0520
m4_hourly/H/short	0.8987	0.0257	0.9382	0.0261	0.9370	0.0264
m4_monthly/M/short	0.9242	0.0916	0.9325	0.0918	0.9308	0.0916
m4_quarterly/Q/short	1.1601	0.0739	1.1660	0.0741	1.1581	0.0741
m4_weekly/W/short	2.2482	0.0411	2.2437	0.0432	2.3409	0.0443
m4_yearly/A/short	3.4245	0.1183	3.3249	0.1152	3.2777	0.1139
m_dense/D/short	0.6884	0.0666	0.6920	0.0665	0.6929	0.0664

Continued on the next page.

Table 25: Detailed forecasting results on GIFT-Eval (continued).

Dataset	PatchTST		PatchTST-WB		LatentTime	
	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓	MASE ↓	CRPS ↓
m_dense/H/long	0.7242	0.1180	0.7268	0.1191	0.7333	0.1201
m_dense/H/medium	0.7024	0.1154	0.7039	0.1160	0.7092	0.1170
m_dense/H/short	0.8228	0.1375	0.8165	0.1370	0.8216	0.1379
restaurant/D/short	0.7036	0.2651	0.7009	0.2640	0.7014	0.2641
saugeen/D/short	3.1148	0.3831	2.9810	0.3591	3.0014	0.3633
saugeen/M/short	0.7472	0.2994	0.7529	0.3044	0.7553	0.3055
saugeen/W/short	1.1468	0.3444	1.1621	0.3517	1.1606	0.3510
solar/10T/long	0.7666	0.2833	0.7836	0.2855	0.7629	0.2808
solar/10T/medium	0.8058	0.3005	0.8005	0.2967	0.7828	0.2925
solar/10T/short	0.9274	0.4426	0.7850	0.3822	0.7868	0.3834
solar/D/short	0.9729	0.2800	0.9786	0.2816	0.9821	0.2821
solar/H/long	0.8989	0.3062	0.8396	0.2885	0.8506	0.2927
solar/H/medium	0.8627	0.3054	0.7805	0.2805	0.7900	0.2847
solar/H/short	0.8302	0.2977	0.7699	0.2779	0.7775	0.2813
solar/W/short	1.1413	0.1538	1.1898	0.1588	1.1511	0.1548
sz_taxi/15T/long	0.5035	0.1949	0.5012	0.1937	0.5010	0.1937
sz_taxi/15T/medium	0.5352	0.2018	0.5329	0.2007	0.5326	0.2005
sz_taxi/15T/short	0.5471	0.2017	0.5459	0.2012	0.5457	0.2011
sz_taxi/H/short	0.5640	0.1365	0.5635	0.1365	0.5631	0.1364
temperature_rain/D/short	1.3677	0.5669	1.3648	0.5637	1.3654	0.5646
us_births/D/short	0.4232	0.0218	0.4091	0.0211	0.4095	0.0211
us_births/M/short	0.8735	0.0173	0.7324	0.0146	0.7153	0.0144
us_births/W/short	1.0439	0.0121	0.9711	0.0118	0.9833	0.0119

D.2 Full Results of TS-AE reconstruction

The full results of reconstruction of TS-AE on Gift-Eval datasets with 2048 context length are shown in Table 26.

Table 26: Detailed reconstruction results of TS-AE on GIFT-Eval. Lower MSE and MAE indicate better pointwise accuracy, while higher Pearson correlation, ACF similarity, and PSD similarity indicate better structural preservation. Dashes indicate unavailable results.

Dataset	Domain	MSE ↓	MAE ↓	Pearson ↑	ACF ↑	PSD ↑
bitbrains_fast_storage/5T	Web/CloudOps	0.0325	0.1046	0.8966	0.8887	0.906
bitbrains_fast_storage/H	Web/CloudOps	-	-	-	-	-
bitbrains_rnd/5T	Web/CloudOps	0.0349	0.1112	0.9051	0.8812	0.9098
bitbrains_rnd/H	Web/CloudOps	-	-	-	-	-
bizitobs_application/10S	Web/CloudOps	0.0015	0.0232	0.9988	1	1
bizitobs_l2c/5T	Web/CloudOps	0.0011	0.0174	0.9993	1	1
bizitobs_l2c/H	Web/CloudOps	0.0026	0.0315	0.9981	1	1
bizitobs_service/10S	Web/CloudOps	0.0027	0.0308	0.9343	0.9365	0.9365
car_parts/M	Sales	-	-	-	-	-
covid_deaths/D	Healthcare	-	-	-	-	-
electricity/15T	Energy	0.0122	0.0726	0.9864	0.9953	0.9952
electricity/D	Energy	-	-	-	-	-
electricity/H	Energy	0.0131	0.0808	0.9884	0.9982	0.9981
electricity/W	Energy	-	-	-	-	-

Continued on next page

Table 26: Detailed reconstruction results of TS-AE on GIFT-Eval (continued).

Dataset	Domain	MSE ↓	MAE ↓	Pearson ↑	ACF ↑	PSD ↑
ett1/15T	Energy	0.0138	0.0764	0.9889	1	0.9999
ett1/D	Energy	-	-	-	-	-
ett1/H	Energy	0.0113	0.0742	0.9909	0.9999	0.9996
ett1/W	Energy	-	-	-	-	-
ett2/15T	Energy	0.0207	0.089	0.9767	0.9908	0.9955
ett2/D	Energy	-	-	-	-	-
ett2/H	Energy	0.0159	0.0826	0.9865	0.9999	0.9997
ett2/W	Energy	-	-	-	-	-
hierarchical_sales/D	Sales	-	-	-	-	-
hierarchical_sales/W	Sales	-	-	-	-	-
hospital/M	Healthcare	-	-	-	-	-
jena_weather/10T	Nature	0.008	0.041	0.9904	0.9958	0.9972
jena_weather/D	Nature					
jena_weather/H	Nature	0.0082	0.0481	0.9868	0.9955	0.9923
kdd_cup_2018/D	Nature	-	-	-	-	-
kdd_cup_2018/H	Nature	0.0482	0.1259	0.9444	0.9984	0.9777
loop_seattle/5T	Transport	0.028	0.1147	0.972	0.9996	0.9993
loop_seattle/D	Transport	-	-	-	-	-
loop_seattle/H	Transport	0.0209	0.0989	0.9793	0.9981	0.9982
m4_daily/D	Transport	-	-	-	-	-
m4_hourly/H	Transport	0.0136	0.0818	0.9897	0.9998	0.9998
m4_monthly/M	Transport	0.0136	0.0818	0.9897	0.9998	0.9998
m4_quarterly/Q	Transport	0.014	0.0827	0.9893	0.9998	0.9998
m4_weekly/W	Econ/Fin	0.0021	0.0281	0.9984	1	1
m4_yearly/A	Econ/Fin	-	-	-	-	-
m_dense/D	Econ/Fin	0.0023	0.0253	0.998	1	1
m_dense/H	Econ/Fin	0.0027	0.0299	0.9977	1	0.9999
restaurant/D	Sales	-	-	-	-	-
saugeen/D	Nature	0.0048	0.041	0.9941	1	0.9998
saugeen/M	Nature	-	-	-	-	-
saugeen/W	Nature	0.0197	0.0905	0.9772	0.9995	0.9997
solar/10T	Energy	0.0021	0.0234	0.9984	1	1
solar/D	Energy	-	-	-	-	-
solar/H	Energy	0.0051	0.0473	0.9962	0.9999	0.9999
solar/W	Energy	-	-	-	-	-
sz_taxi/15T	Transport	0.0631	0.1666	0.9449	0.9897	0.9927
sz_taxi/H	Transport	-	-	-	-	-
temperature_rain/D	Nature	-	-	-	-	-
us_births/D	Healthcare	0.0449	0.1652	0.9661	0.9736	0.9745
us_births/M	Healthcare	-	-	-	-	-
us_births/W	Healthcare	-	-	-	-	-

E VISUALIZATION

This section presents visualizations of TS-AE compression and reconstruction, its pre-trained latent representations, and LatentTime forecasts. Examples from real time series and synthetic signals complement the quantitative evaluations.

E.1 The Compression and Reconstruction of TS-AE

To illustrate TS-AE's reconstruction behavior under $2\times$ temporal compression, we visualize real-world time series from ETTh1, ETTm1, Electricity, and Weather, synthetic series generated by KernelSynth (Ansari et al., 2024) and ForecastPFN (Dooley et al., 2023), and representative signals with diverse temporal patterns. Figures 18–33 cover input lengths from 1024 to 6656, complementing the quantitative evaluation with examples of oscillations, changing frequencies, abrupt transitions, and electrocardiogram waveforms.

E.2 The Latent Space of Pre-trained TS-AE

Figure 34 and Figure 35 present the TSNE embeddings for several subsets of the Gift-Eval benchmark, with each time series sample having lengths of 256 and 512, respectively. This approach primarily demonstrates the differences between the latent space of TS-AE and the original time series in terms of overall representation. Subsequently, using the multivariate time series dataset ETTh1 as an example, we sampled each of the seven channels and embedded data using different channels as units. The results for sample lengths of 256 and 512 are shown in Figure 36 and Figure 37. Finally, to simply illustrate the continuity of the latent space of TS-AE, we used the OT channel of the Electricity dataset as an example, dividing the samples according to timestamps. The results for sample lengths of 256 and 512 are shown in Figure 38 and Figure 39.

E.3 The Forecasting of LatentTime

To qualitatively assess LatentTime's forecasting performance, we visualize predictions on synthetic sinusoidal signals and real-world datasets from GIFT-Eval (Aksu et al., 2024). Synthetic examples include single-frequency signals with $\omega = 2 \cdot 2\pi$ (Figure 40) and $\omega = 4 \cdot 2\pi$ (Figure 41), and a superposition of three sinusoidal components (Figure 42). Real-world examples include ETTh1 (Figure 43), ETTm1 (Figure 44), Electricity (Figure 45), Weather (Figure 46), BizITObs Application (Figure 47), Loop Seattle (Figure 48), daily M_DENSE (Figure 49), hourly M_DENSE (Figure 50), SZ-Taxi (Figure 51), Hierarchical Sales (Figure 52), Saugeen (Figure 53), and Solar (Figure 54).

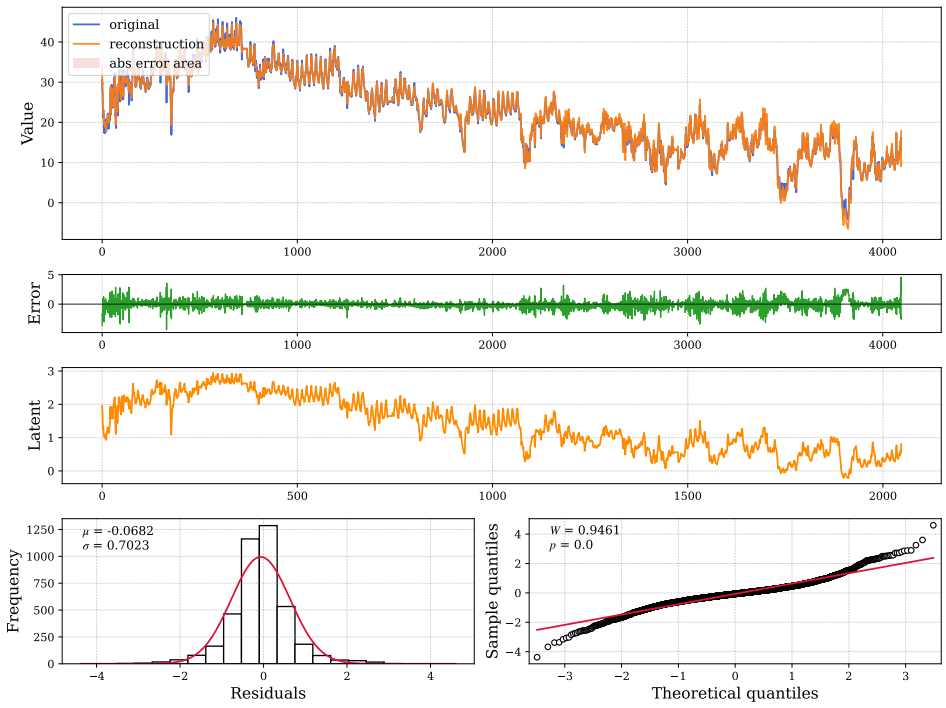


Figure 18: TS-AE reconstruction on the ETTh1 dataset with input length $L = 4096$.

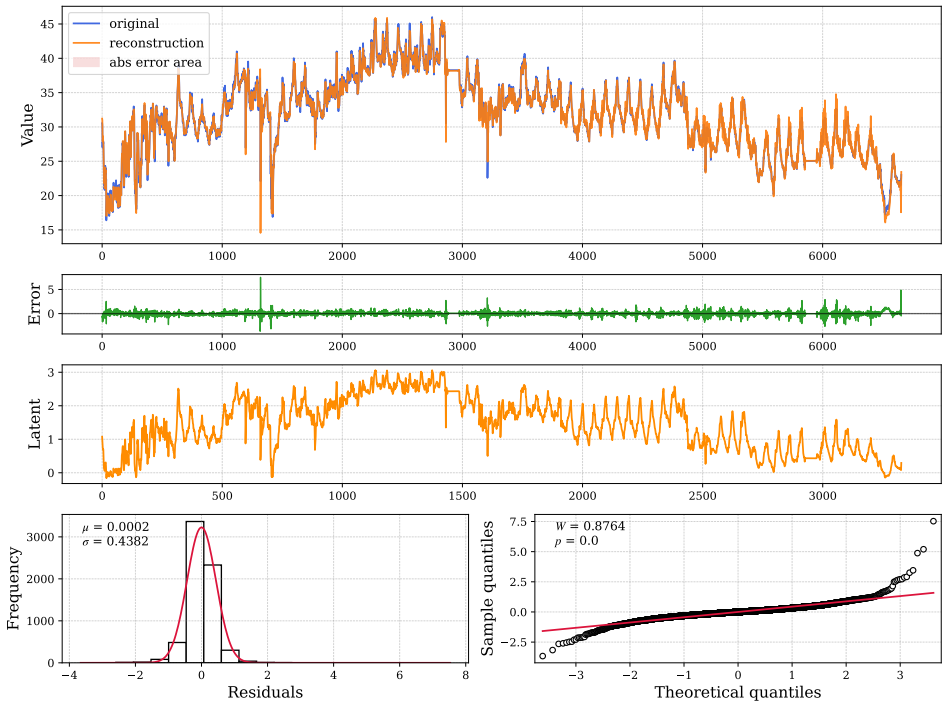


Figure 19: TS-AE reconstruction on the ETTm1 dataset with input length $L = 6656$.

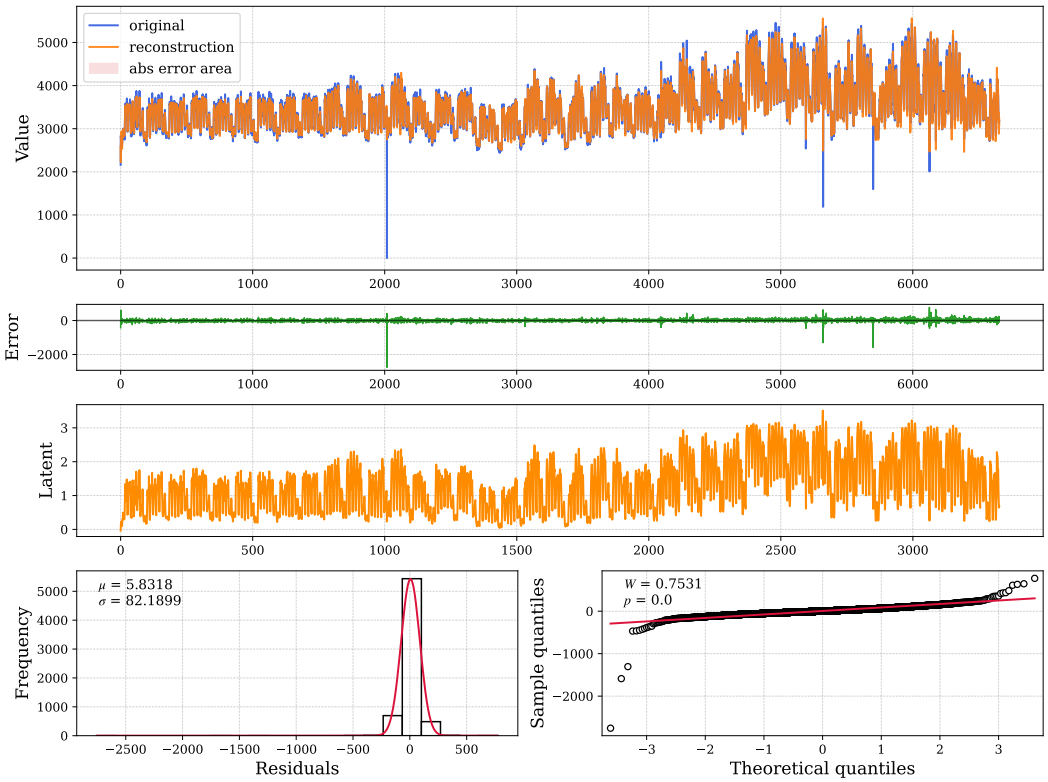


Figure 20: TS-AE reconstruction on the Electricity dataset with input length $L = 6656$.

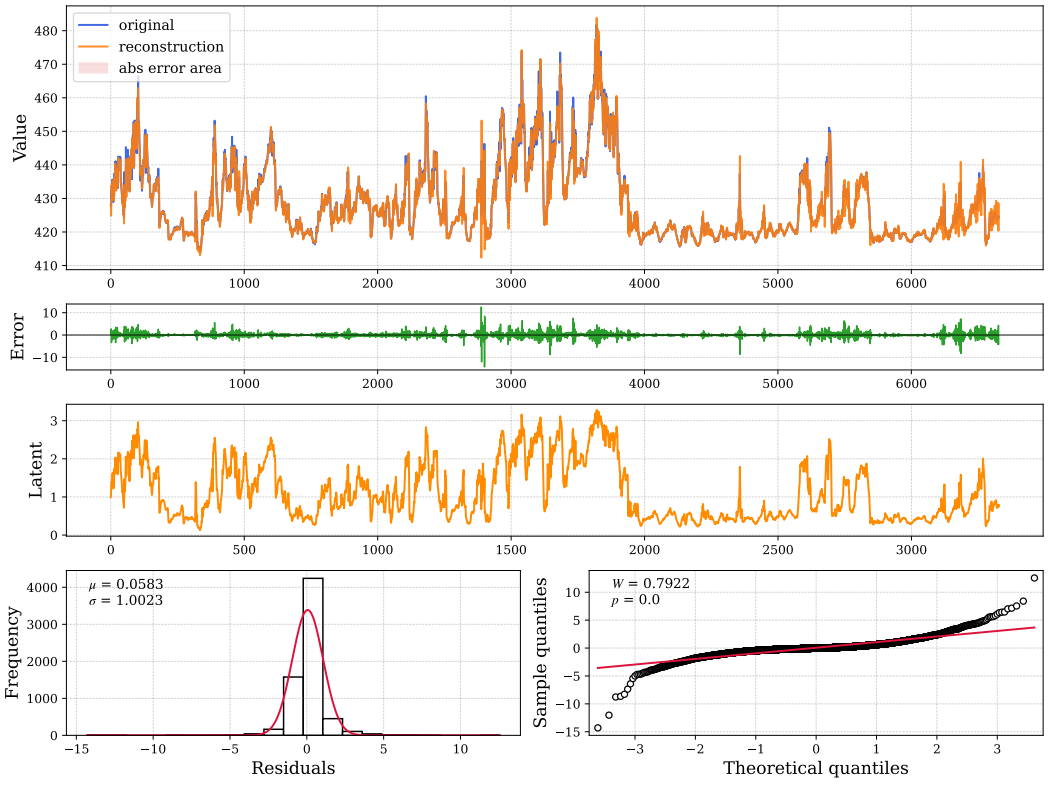


Figure 21: TS-AE reconstruction on the Weather dataset with input length $L = 6656$.

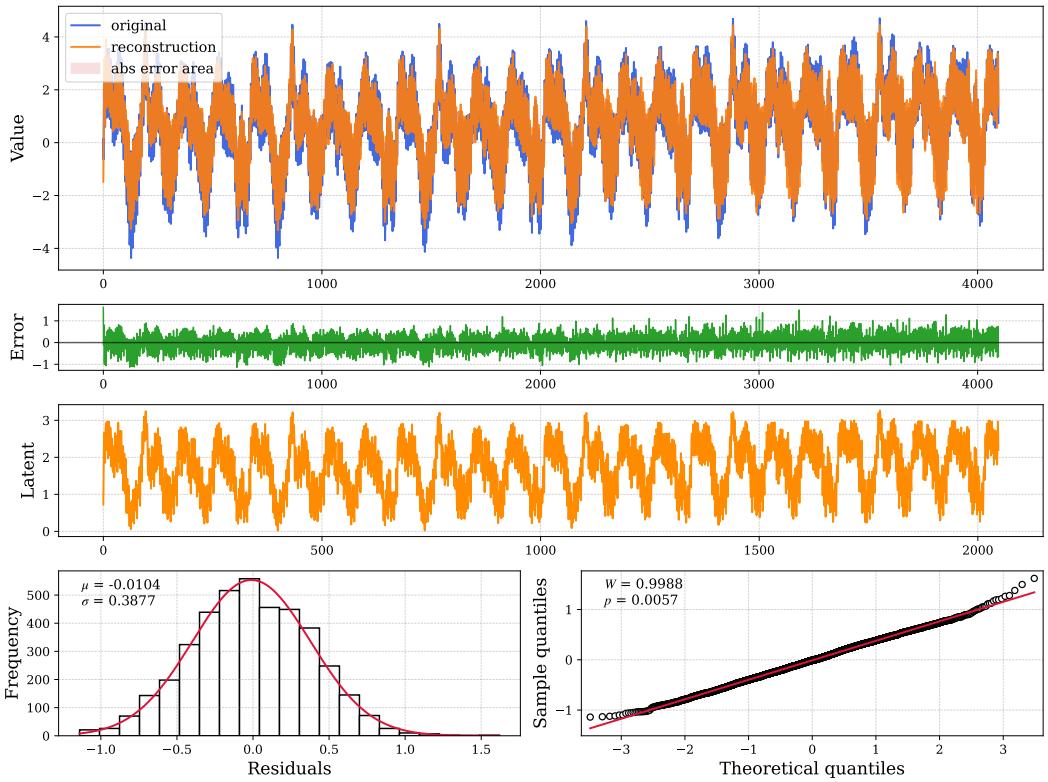


Figure 22: Synthetic time series generated by KernelSynth (Ansari et al., 2024).

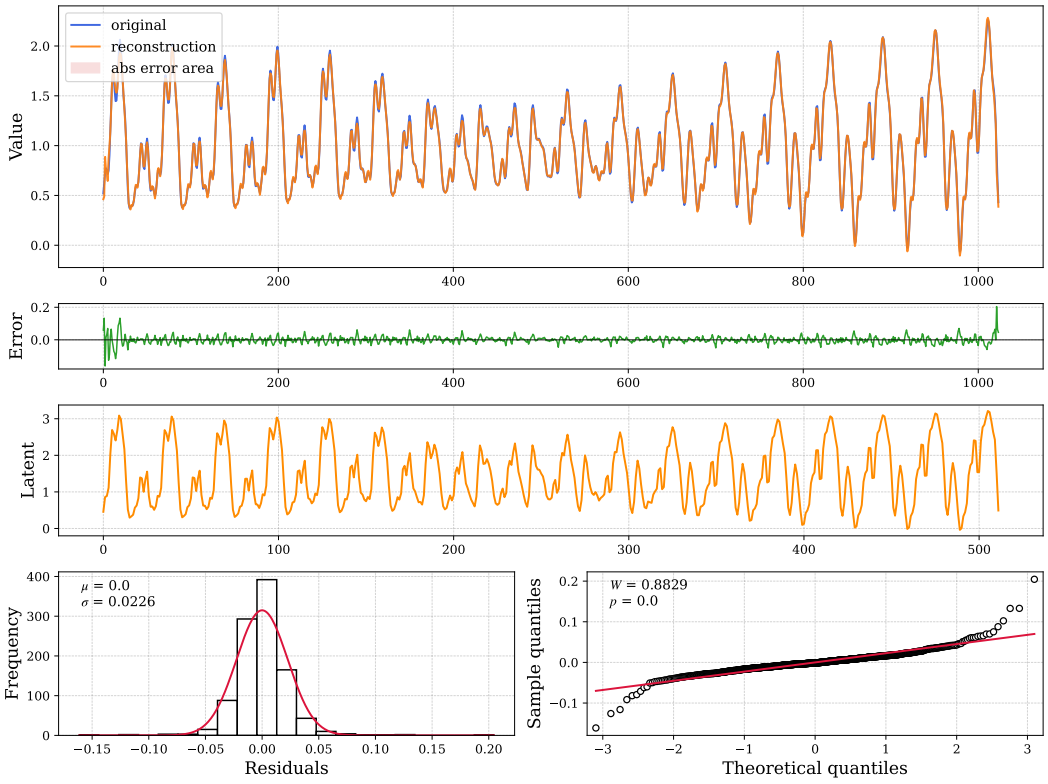


Figure 23: Synthetic time series generated by ForecastPFN (Dooley et al., 2023).

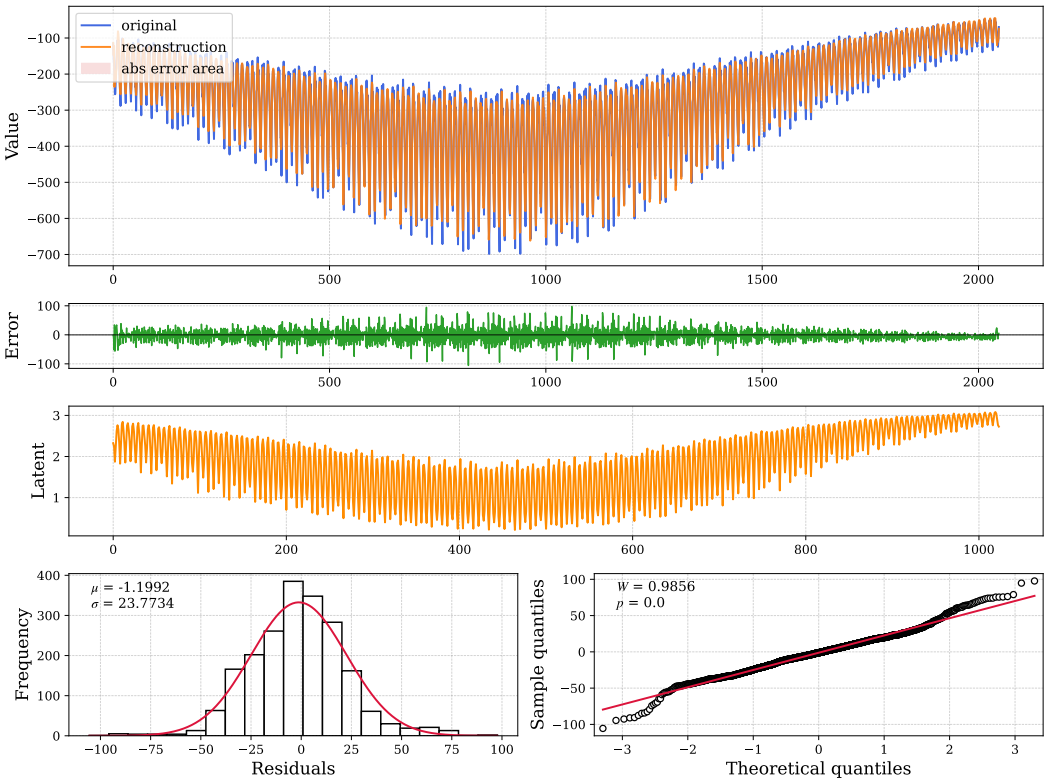


Figure 24: Synthetic time series generated by ForecastPFN (Dooley et al., 2023).

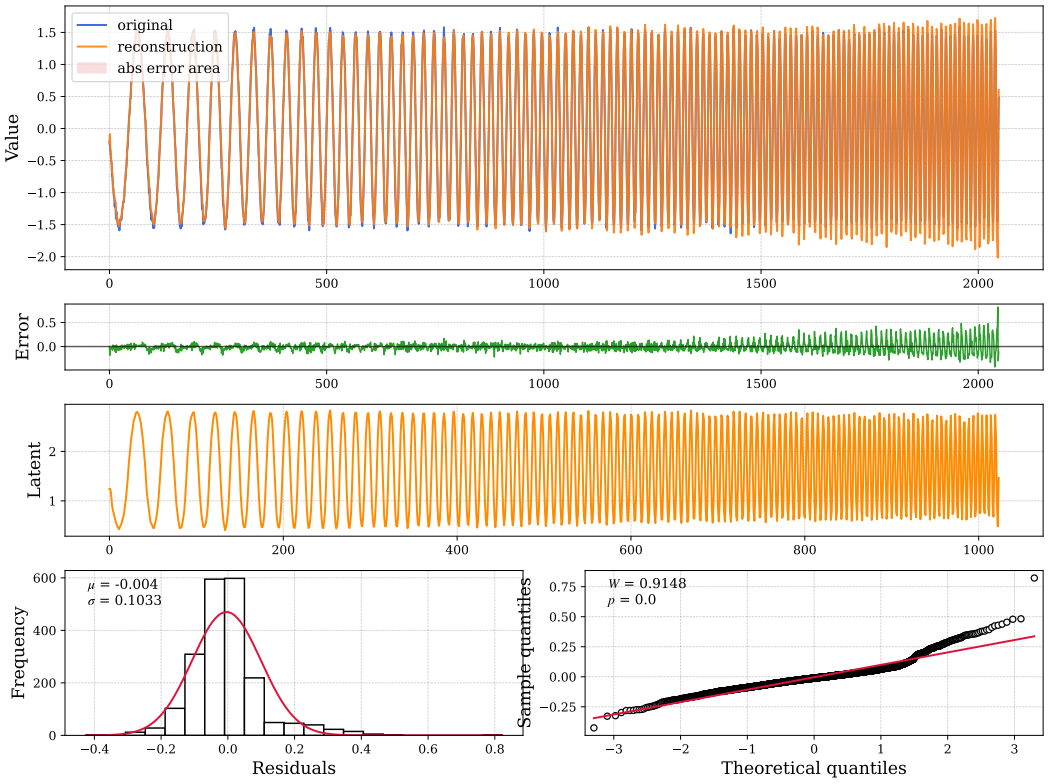


Figure 25: TS-AE reconstruction of a chirp signal with input length $L = 2048$.

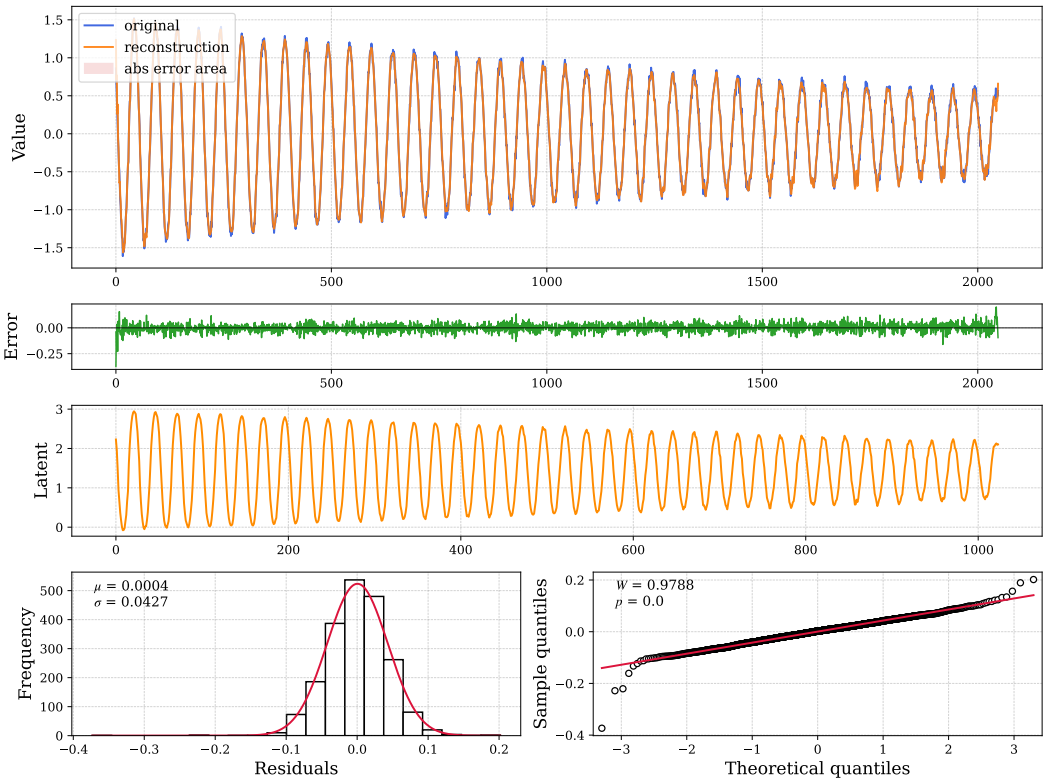


Figure 26: TS-AE reconstruction of a damped oscillation signal with input length $L = 2048$.

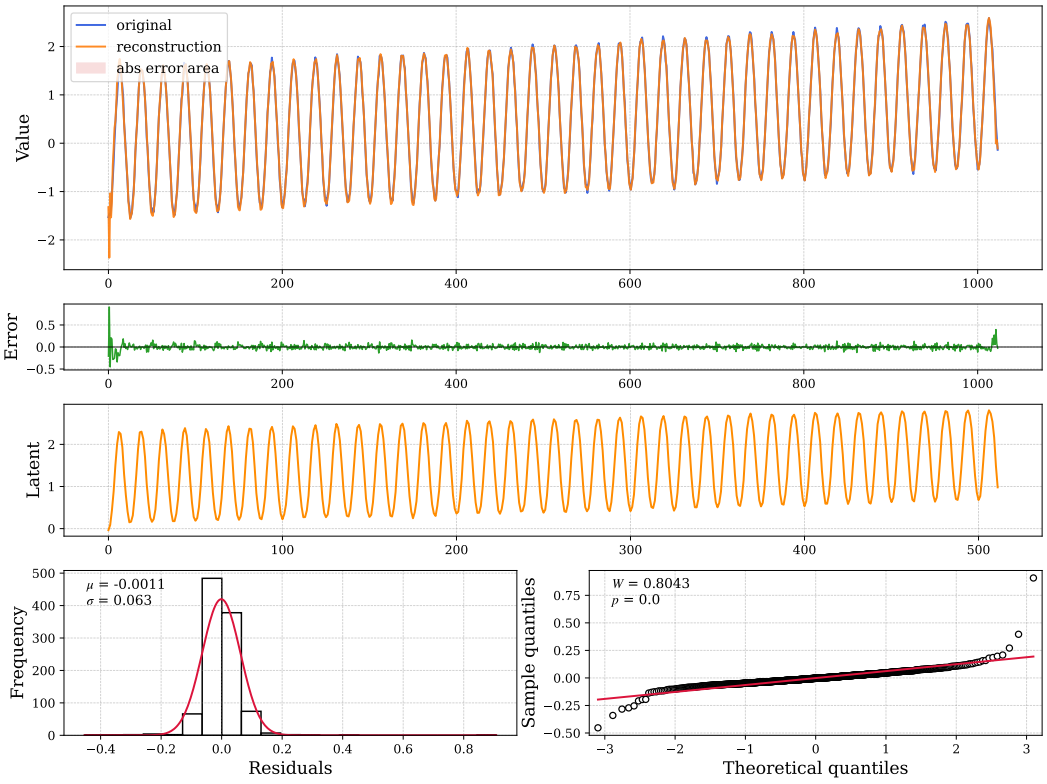


Figure 27: TS-AE reconstruction of a nonstationary sinusoidal signal with input length $L = 1024$.

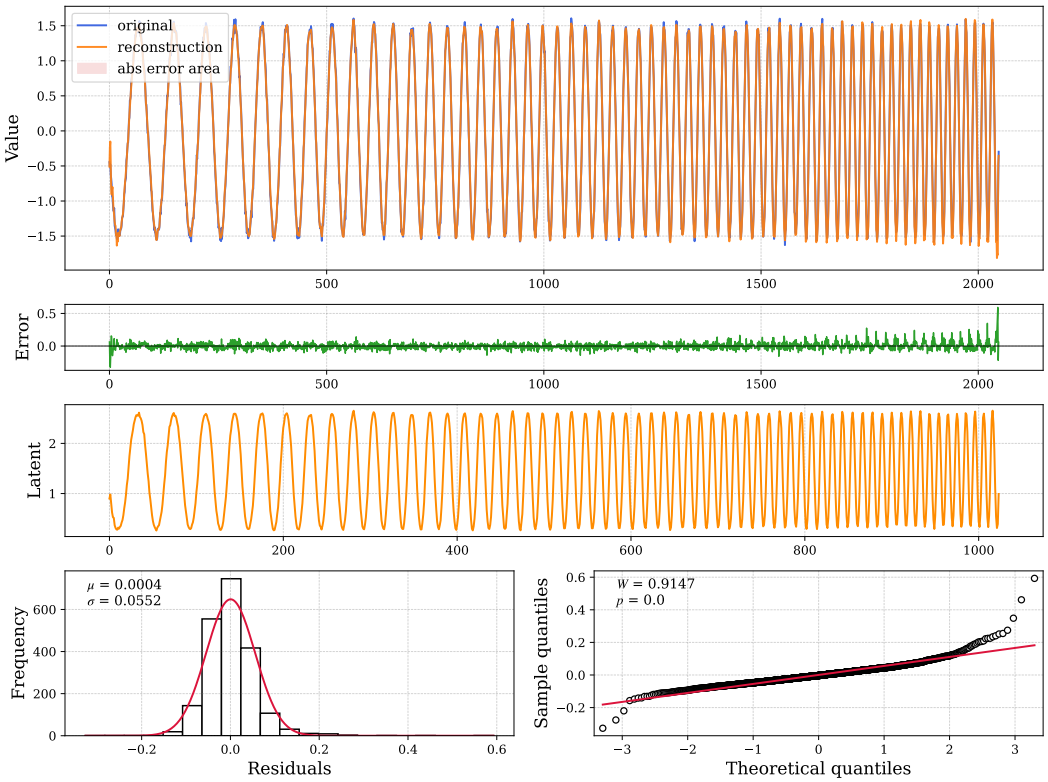


Figure 28: TS-AE reconstruction of a sinusoidal signal.

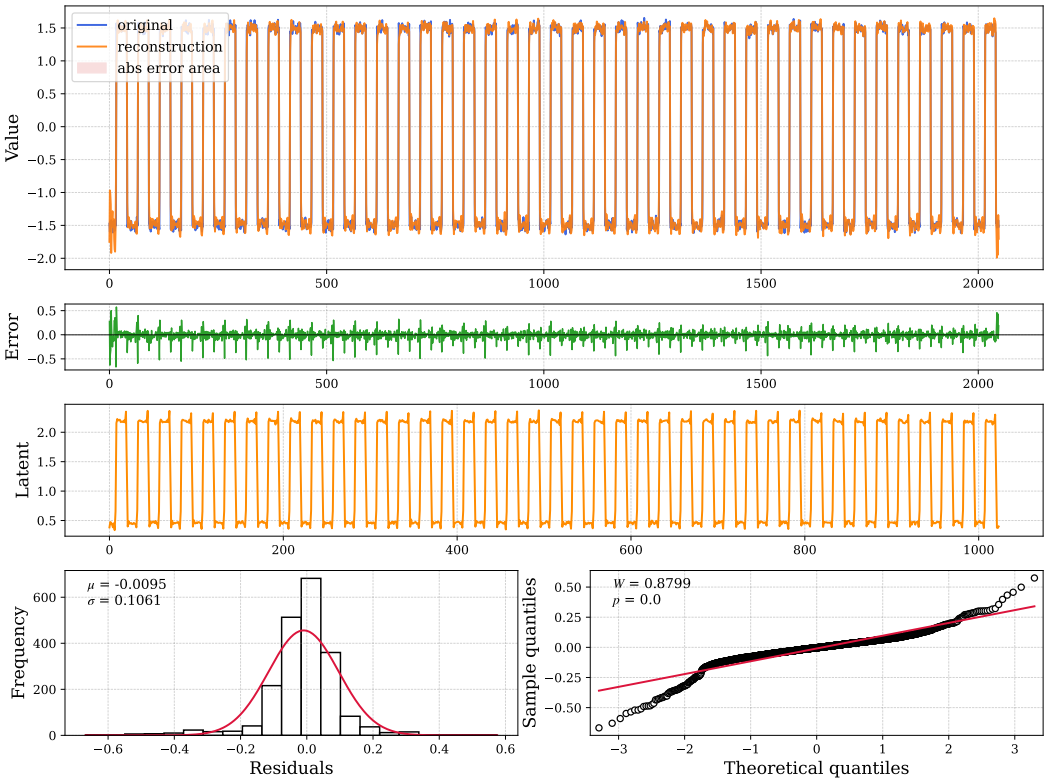


Figure 29: TS-AE reconstruction of a square wave with input length $L = 2048$.

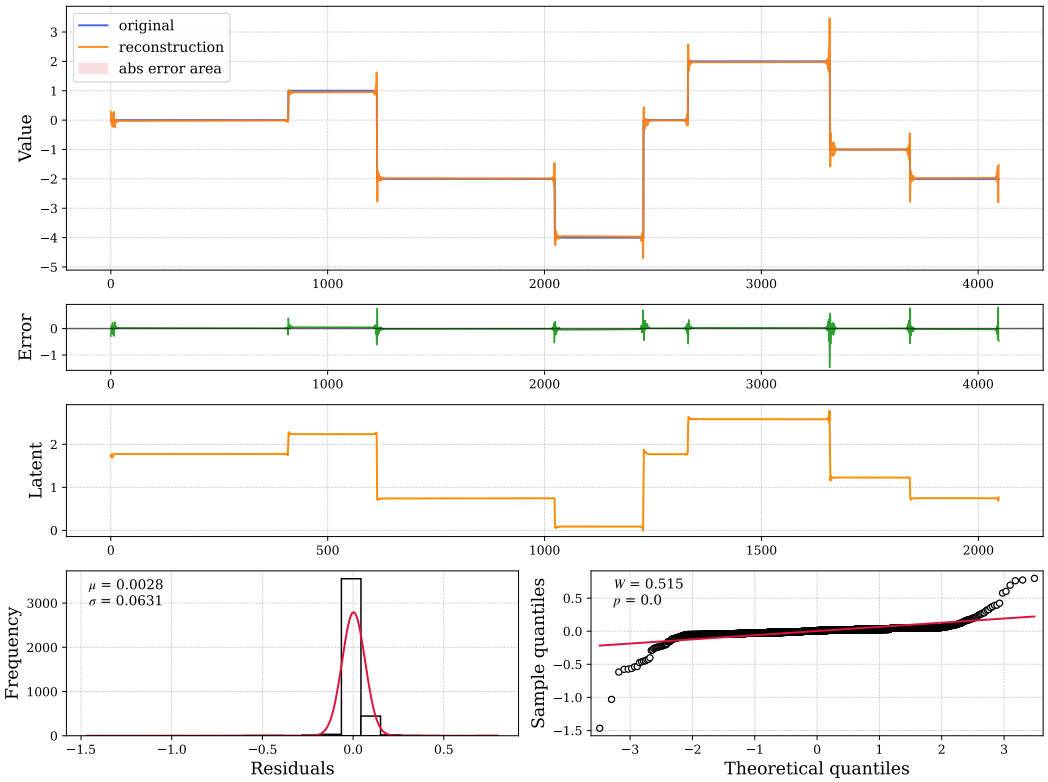


Figure 30: TS-AE reconstruction of a step signal.

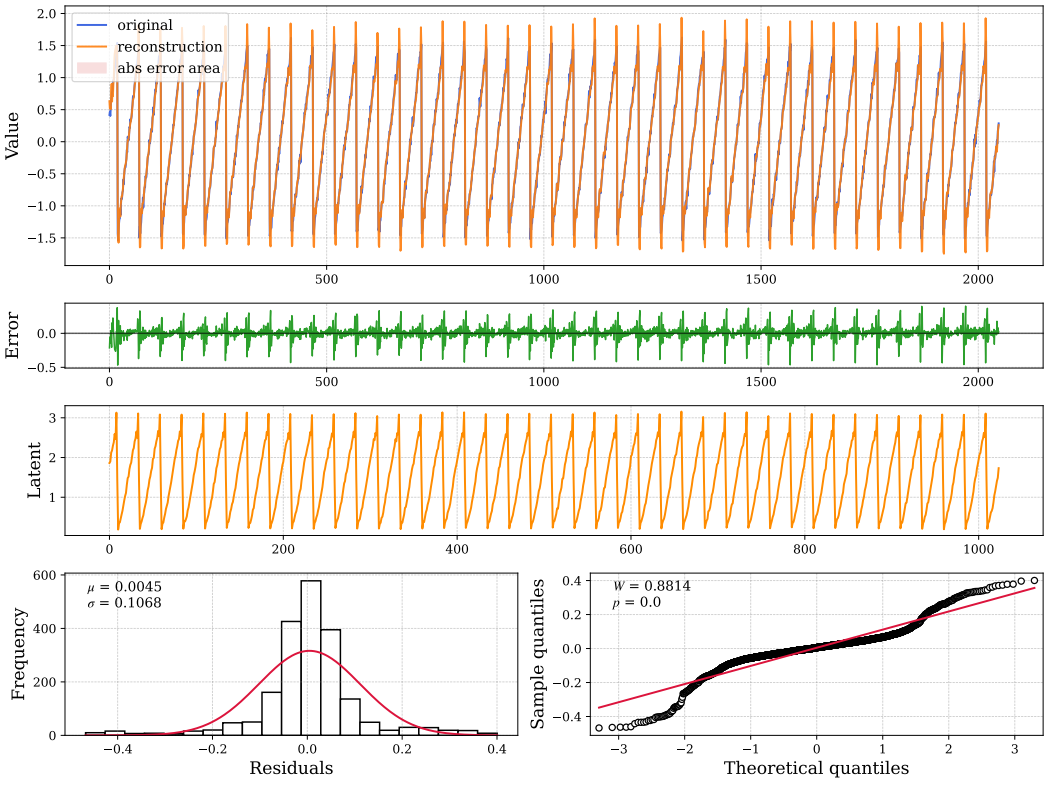


Figure 31: TS-AE reconstruction of a sawtooth wave with input length $L = 2048$.

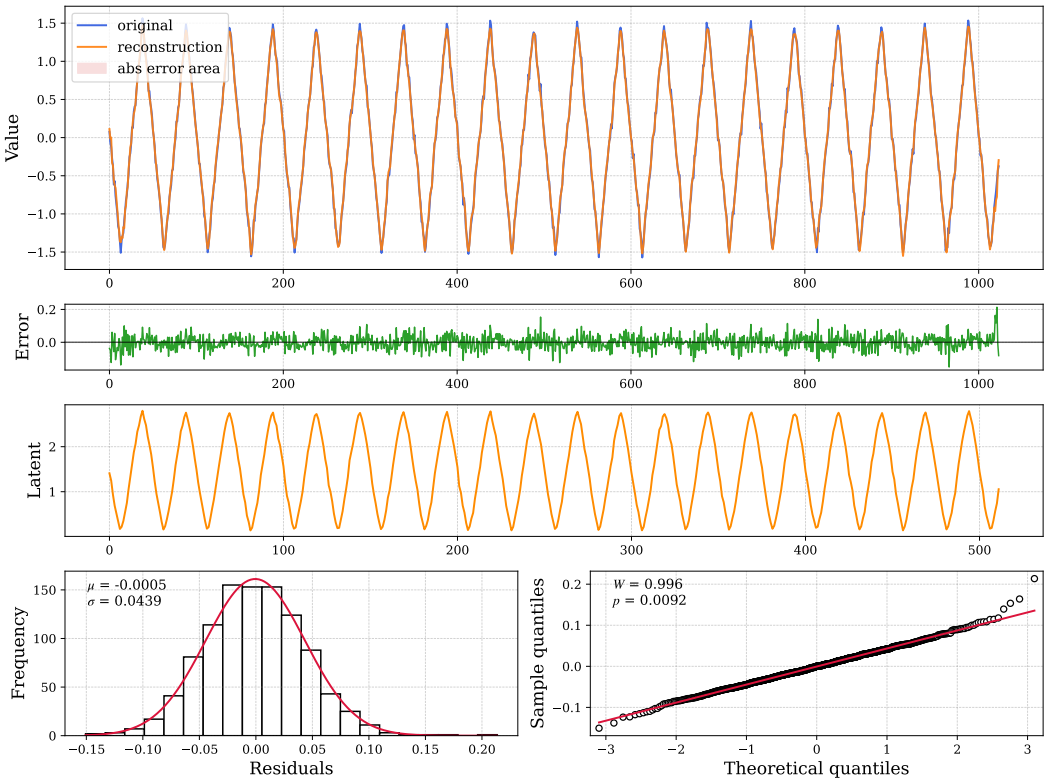


Figure 32: TS-AE reconstruction of a triangle wave with input length $L = 1024$.

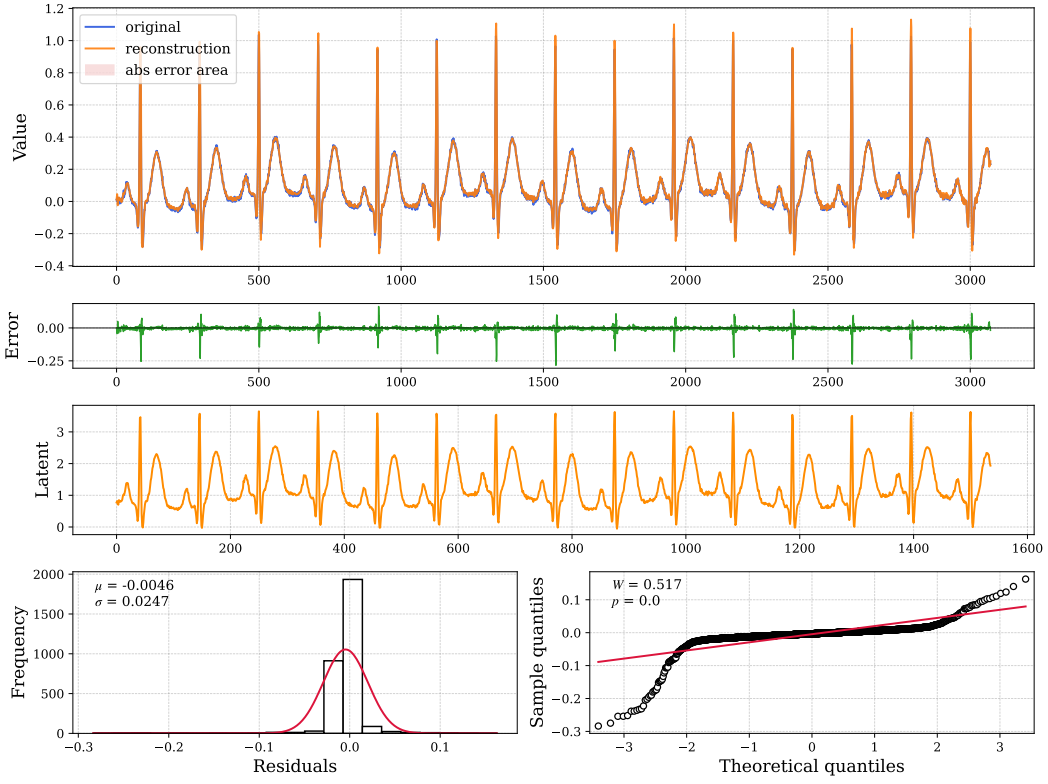


Figure 33: TS-AE reconstruction of an electrocardiogram (ECG) signal with input length $L = 3072$.

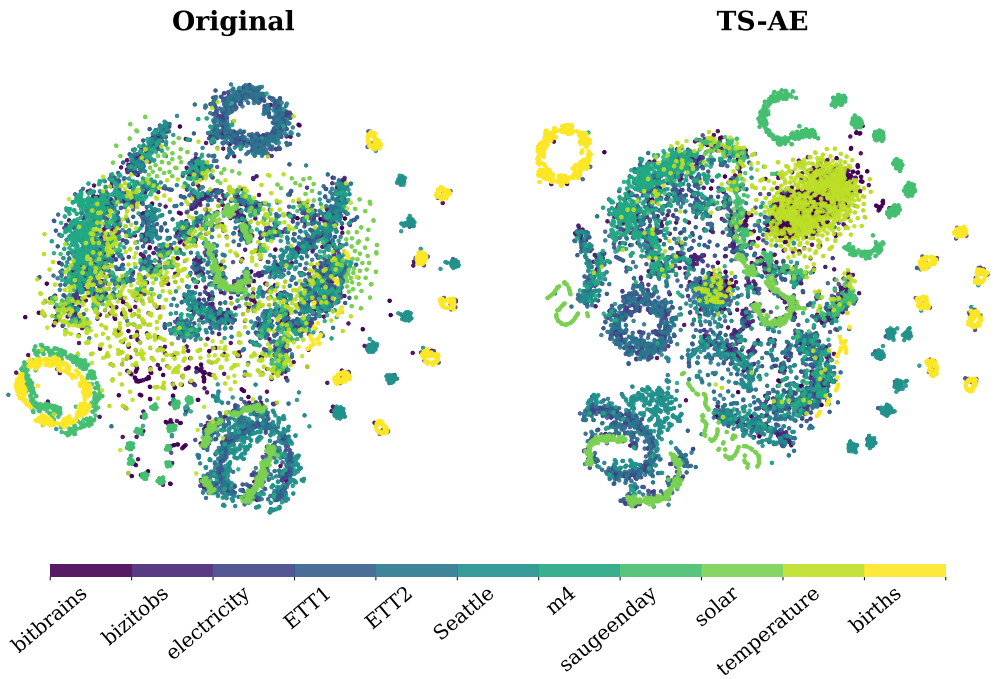


Figure 34: The t-SNE visualization on different datasets in Gift-Eval with context length of 256.

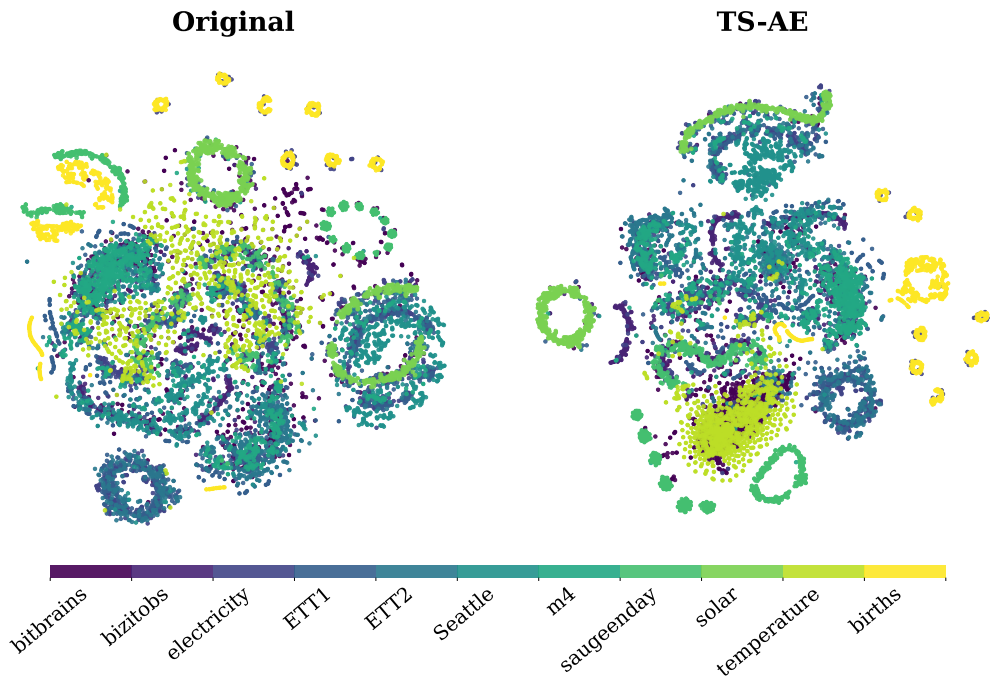


Figure 35: The t-SNE visualization on different datasets in Gift-Eval with context length of 512.

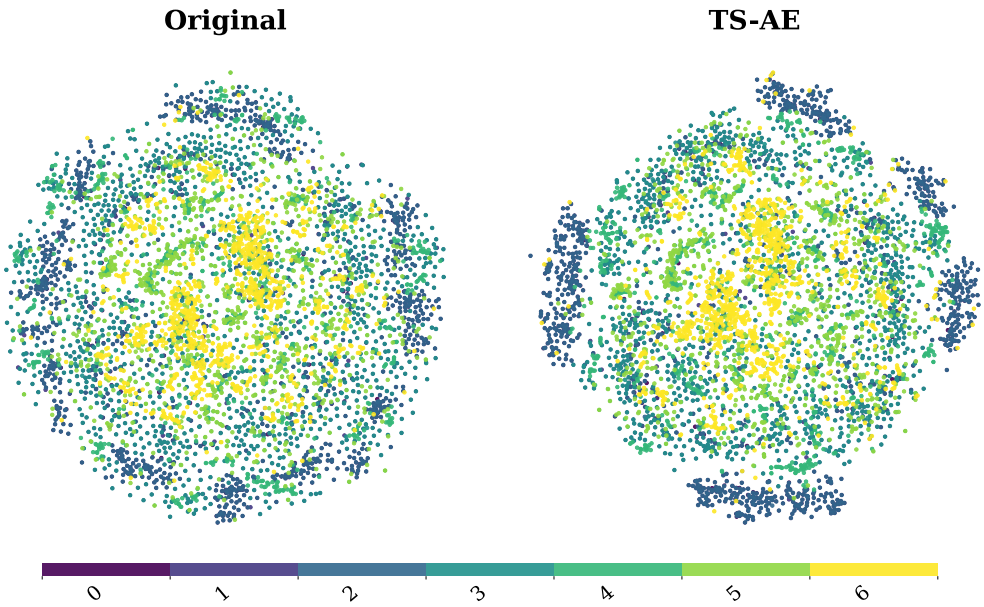


Figure 36: The t-SNE visualization of different channels on ETTh1-15T dataset with context length of 256.

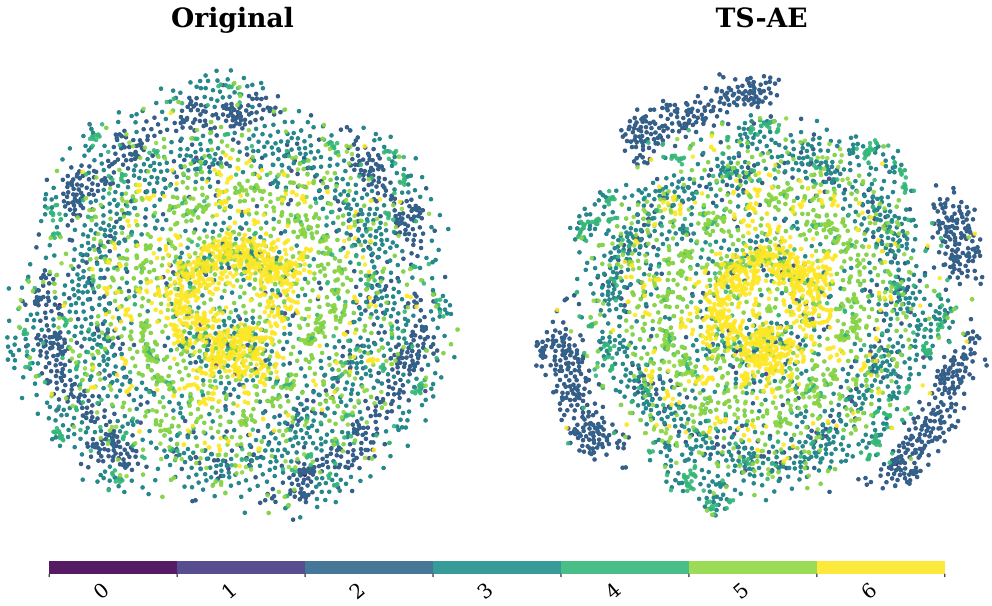


Figure 37: The t-SNE visualization of different channels on ETTh1-15T dataset with context length of 512.

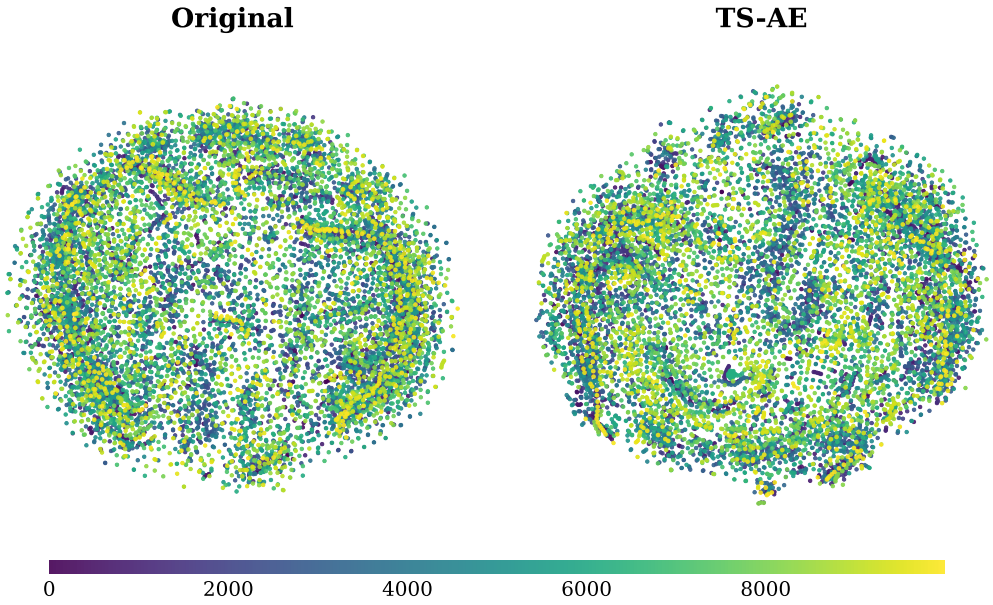


Figure 38: The t-SNE visualization of 256-length time series at different timestamps within the same channel in Electricity dataset.

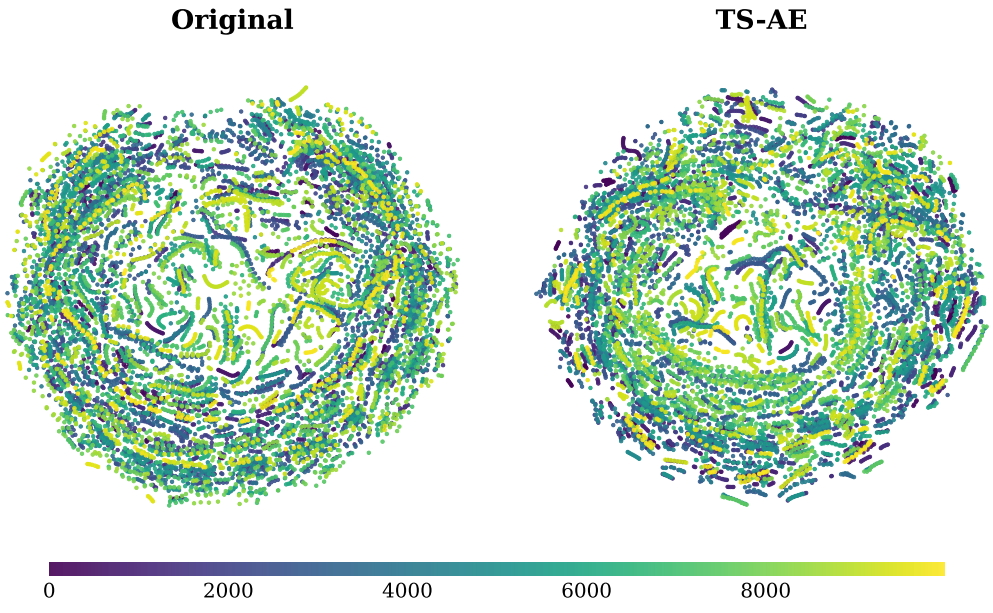


Figure 39: The t-SNE visualization of 512-length time series at different timestamps within the same channel in Electricity dataset.

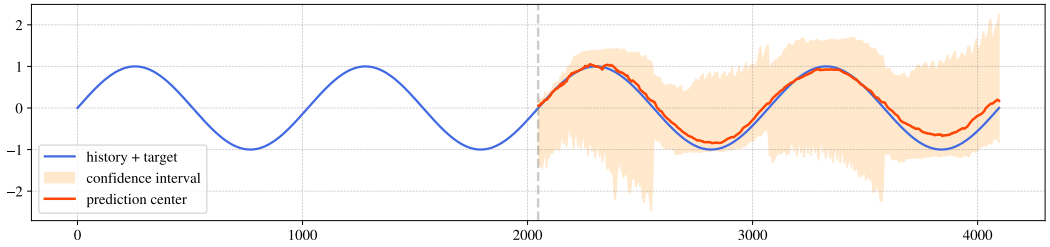


Figure 40: LatentTime forecasts on a synthetic sinusoidal signal with angular frequency $\omega = 2 \cdot 2\pi$.

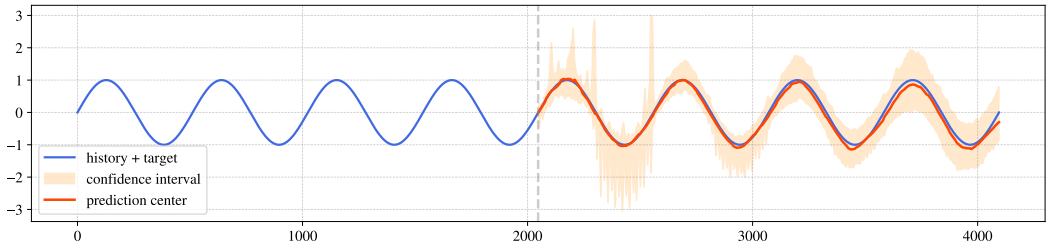


Figure 41: LatentTime forecasts on a synthetic sinusoidal signal with angular frequency $\omega = 4 \cdot 2\pi$.

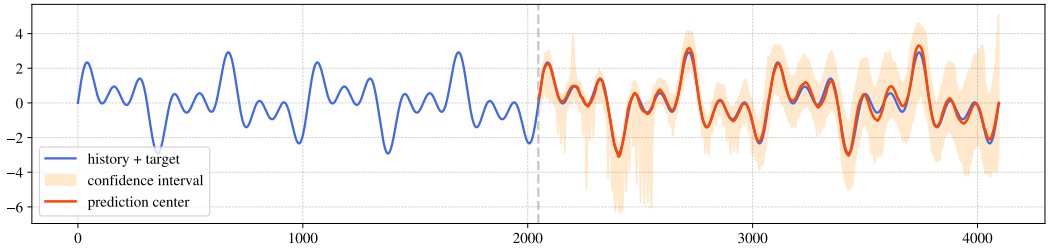


Figure 42: LatentTime forecasts on a synthetic signal formed by superimposing sinusoidal components with angular frequencies $4 \cdot 2\pi$, $10 \cdot 2\pi$, and $16 \cdot 2\pi$.

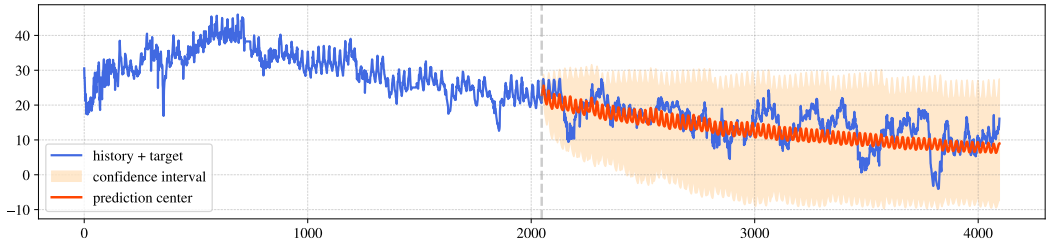


Figure 43: LatentTime forecasts on the ETTh1 dataset with hourly sampling.

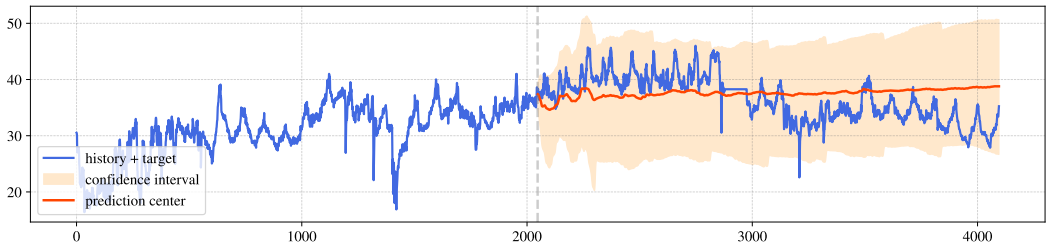


Figure 44: LatentTime forecasts on the ETTm1 dataset with 15-minute sampling.

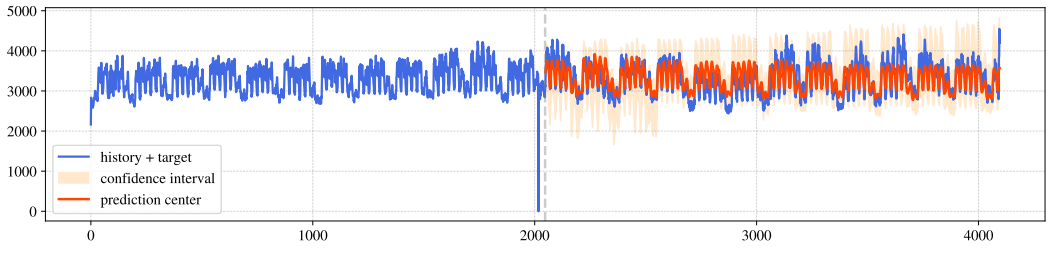


Figure 45: LatentTime forecasts on the Electricity dataset.

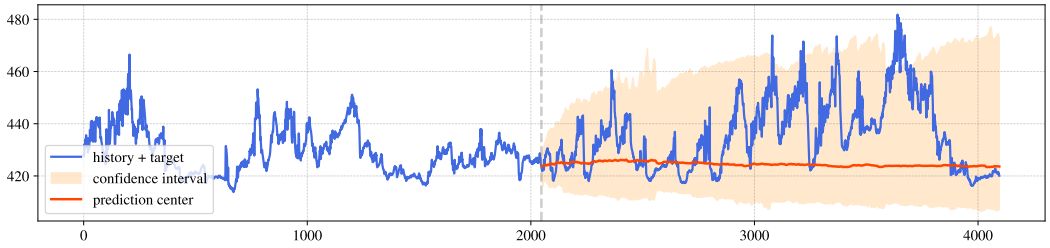


Figure 46: LatentTime forecasts on the Weather dataset.

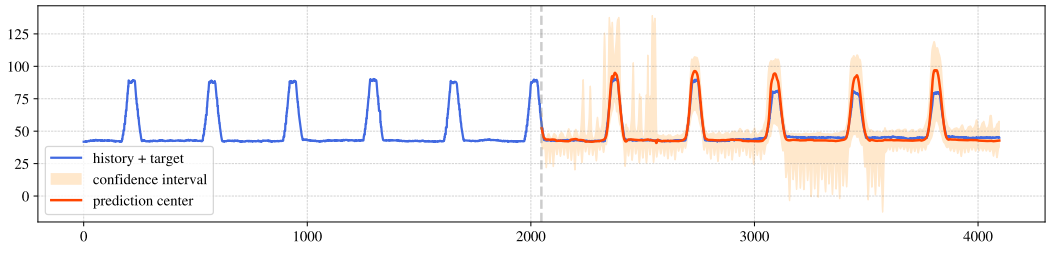


Figure 47: LatentTime forecasts on the BizITObs Application dataset.

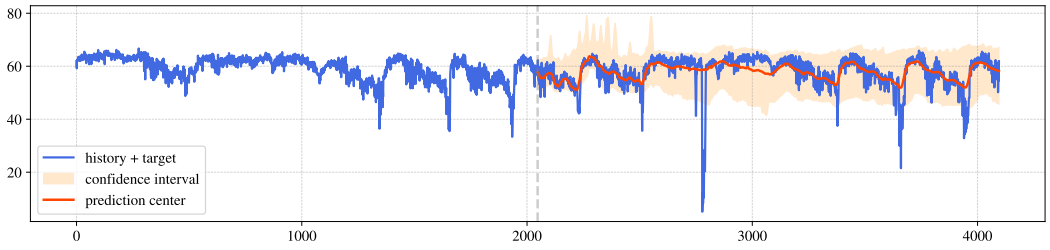


Figure 48: LatentTime forecasts on the Loop Seattle dataset with 5-minute sampling.

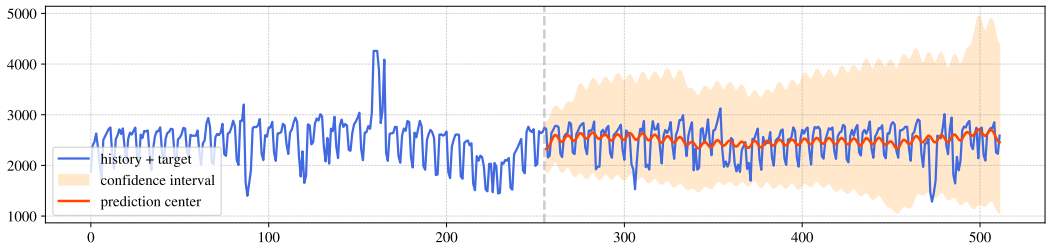


Figure 49: LatentTime forecasts on the M_DENSE dataset with daily sampling.

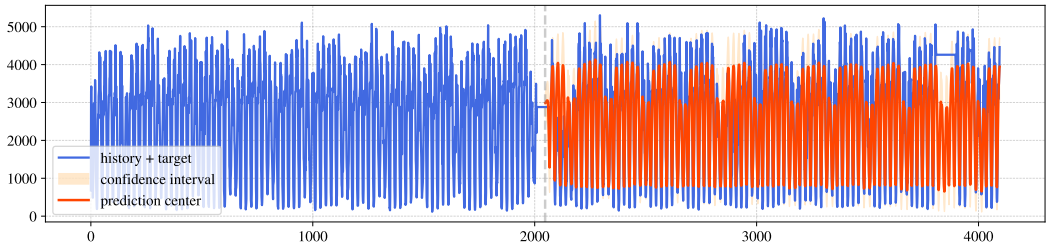


Figure 50: LatentTime forecasts on the M_DENSE dataset with hourly sampling.

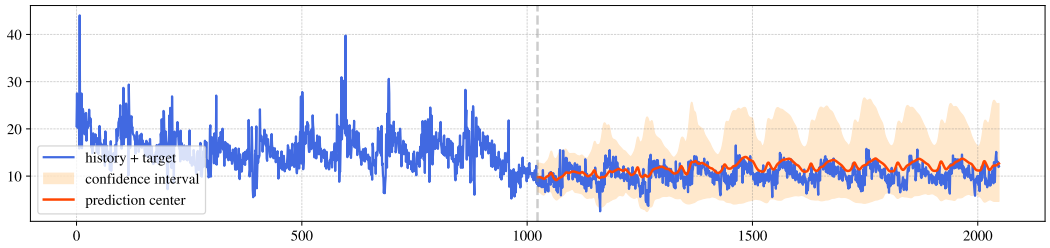


Figure 51: LatentTime forecasts on the SZ-Taxi dataset with 15-minute sampling.

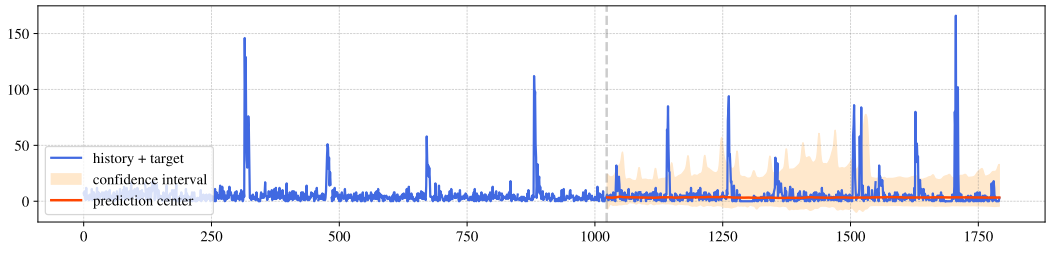


Figure 52: LatentTime forecasts on the Hierarchical Sales dataset with daily sampling.

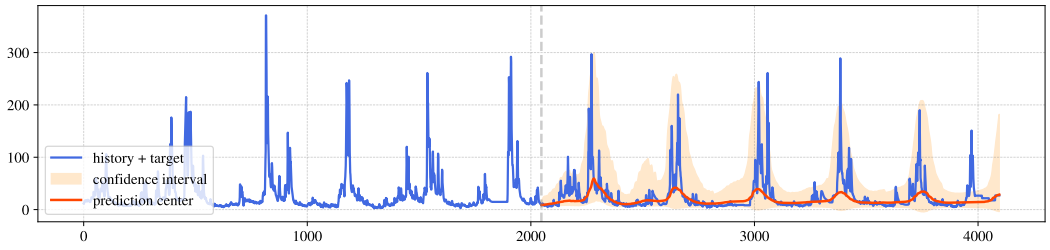


Figure 53: LatentTime forecasts on the Saugeen dataset with daily sampling.

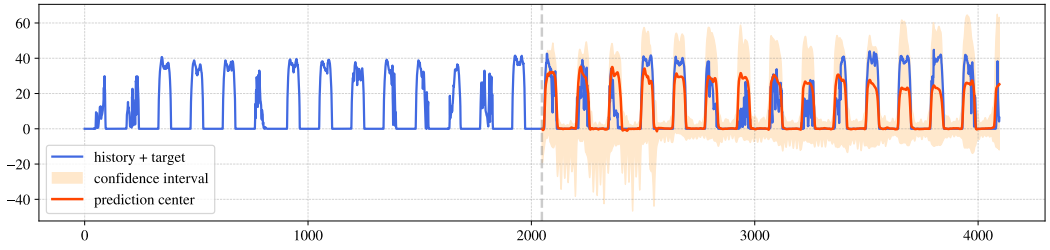


Figure 54: LatentTime forecasts on the Solar dataset with 10-minute sampling.