Revisiting the Generic Transformer: Deconstructing a Strong Baseline for Time Series Foundation Models

Yunshi Wen*
Rensselaer Polytechnic Institute

Wesley M. Gifford
IBM Research
wmgifford@us.ibm.com

Chandra Reddy
IBM Research
creddy@us.ibm.com

Lam M. Nguyen
IBM Research
lamnguyen.mltd@ibm.com

Jayant Kalagnanam
IBM Research
jayant@us.ibm.com

Anak Agung Julius
Rensselaer Polytechnic Institute
agung@ecse.rpi.edu

Abstract

The recent surge of interest in Time Series Foundation Models has rapidly advanced the field. Yet, heterogeneous training setups across studies make it difficult to determine whether performance improvements came from architectural innovations or effective data engineering. In this work, we investigate whether a standard patch Transformer architecture, paired with a straightforward training protocol, can match the zero-shot forecasting performance of state-of-the-art TSFMs. We conduct a comprehensive ablation study covering model scaling, data composition, and training techniques to examine the ingredients behind this performance. Our findings show that pretraining data composition, contiguous patch masking, and context length have a substantial effect on forecasting accuracy, and that the generic architecture itself can deliver high performance with good scalability. By strictly controlling these experimental variables, we provide comprehensive empirical results on model scaling across multiple dimensions. Furthermore, we establish that the effectiveness of this training protocol generalizes robustly across benchmarks and alternative architectures. We release our pretrained models and document the detailed training protocol and findings to establish a transparent, reproducible baseline for future research.

1 Introduction

The recent shift from domain-specific models to cross-domain Time Series Foundation Models (TSFMs) capable of zero-shot generalization marks a significant advancement in the field. These models have demonstrated remarkable success across a diverse range of domains and downstream tasks (Godahewa et al., 2021; Jiang et al., 2023; Emami et al., 2023; Liu et al., 2023; Nguyen et al., 2023), achieving performance that matches or exceeds state-of-the-art (SOTA) supervised methods (Goswami et al., 2024; Wen et al., 2024; Shi et al., 2025). Given the ubiquity of time series data and the proven feasibility of this approach, TSFMs have garnered substantial attention in the recent literature.

The performance of a foundation model is fundamentally the joint product of its backbone architecture and its training protocol. However, as the field of TSFMs rapidly expands, a concerning trend has emerged: many recent studies claim advantage in architecture while evaluating models trained using

*Work done while the author was a Ph.D. student at Rensselaer Polytechnic Institute.
Correspondence to: yunshi-wen@outlook.com.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

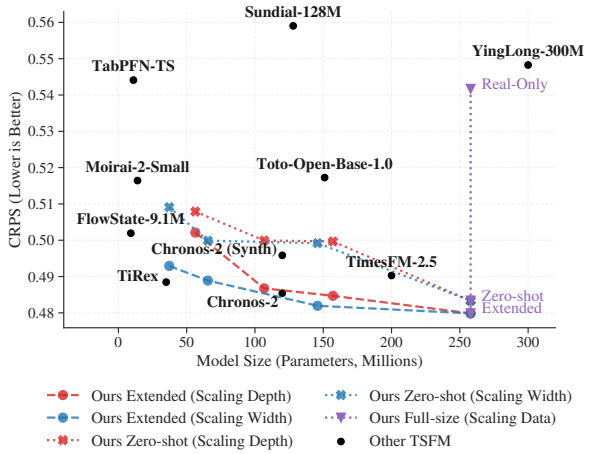

Figure 1: **The Landscape of Time Series Foundation Models on Probabilistic Forecasting**. We compare our generic Transformer with SOTA TSFMs on the GIFT-Eval benchmark. The results establish a strong reference for model scaling in TSFMs and show that, under our protocol, enriching the pretraining data (purple line) yields larger gains than scaling the model size. The Extended model is trained on a corpus that overlaps 23 GIFT-Eval test cases (Section 3.1).

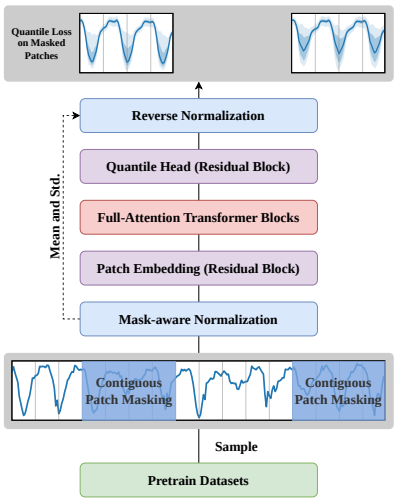

Figure 2: **An overview of the generic Transformer architecture**, an encoder-only Transformer with patched inputs and quantile outputs.

highly heterogeneous datasets and processing techniques. This lack of standardization introduces three critical challenges for the community. First, it creates a confounding factor, making it difficult to attribute performance improvements to actual architectural innovations rather than superior data engineering. Second, although many models are open-sourced, their training protocols are often proprietary or under-documented, severely limiting reproducibility. Finally, for researchers developing new models, the absence of a proven, transparent training recipe makes it nearly impossible to determine whether a novel architectural idea is inherently flawed or simply trained incorrectly. Consequently, the true impact of recent architectural advancements remains obscured.

In this work, we demonstrate that a generic patch Transformer architecture, when coupled with a straightforward training recipe, achieves zero-shot performance on par with the leading TSFMs on the GIFT-Eval probabilistic forecasting benchmark (Aksu et al., 2025) and the TIME benchmark (Qiao et al., 2026). Following this finding, we investigate the fundamental question: *What are the essential ingredients of this effectiveness?* To answer this, we conduct a comprehensive empirical study to examine the contribution of each ingredient required for a successful TSFM.

As illustrated in Figure 1, we compare the scaling behavior of our generic patch Transformer against recent SOTA TSFMs. We observe that this standard architecture delivers performance comparable to, and sometimes exceeding, deliberately designed TSFMs across various model sizes. Notably, within this controlled setting, enriching the pretraining data composition yields larger gains than scaling the model size, highlighting the central role of data in TSFM performance.

Our contributions are summarized as follows:

- **Competitive Performance with a Generic Architecture**: We demonstrate that a standard Transformer architecture, utilizing patched inputs and quantile outputs, achieves zero-shot performance on the GIFT-Eval and TIME benchmarks that is statistically indistinguishable from that of the leading TSFMs (Section 3.2, Appendix E). This result serves as strong evidence for the effectiveness of generic, scalable backbones.
- **Comprehensive Component Analysis**: We conduct a systematic ablation study to examine the impact of critical training components, including model scaling laws, pretraining data composition, and training techniques. With comprehensive empirical experiments on each aspect, we unveil the essential ingredients for the success of TSFM.

- **Key Insights and Analysis**: Our analysis identifies pretraining data composition as an essential factor where both real-world and synthetic datasets contribute to a successful TSFM. We also provide extensive empirical evidence detailing how other key aspects affect the zero-shot performance of TSFM, including model scaling, contiguous patch masking, context length.
- **Open Research**: We release our pretrained model checkpoints and model implementation[1]. We further document the complete training protocol, including pretraining data composition and sampling, masking strategy, optimization settings, and compute (Appendix C.6), together with detailed empirical results and statistical analyses. By providing a transparent and reproducible baseline built from publicly available data sources, we aim to facilitate fair comparisons and accelerate future research in TSFMs.

Scope of this Work: To ensure clarity and rigor, we define the scope of our study as follows:

- **Univariate Modeling and Zero-shot Forecasting:** Our primary evaluation metric is zero-shot performance on univariate time series forecasting tasks. Specifically, our study focuses on the recipe of training a high-quality base foundation model. Therefore, no adaptation approaches, such as finetuning or agentic methods, are compared.
- **Ablation of Intrinsic Factors:** We focus exclusively on analyzing the scaling behaviors and component contributions within our proposed generic architecture. Rather than attempting to reproduce and retrain alternative architectures, which introduces significant variance due to differing implementations and resource requirements, we prioritize a controlled, in-depth analysis of the standard Transformer to establish a robust baseline for the field.

2 Architecture

In this section, we present the formulation of our Time Series Foundation Model (TSFM) architecture, as illustrated in Figure 2. Our design leverages a standard Transformer backbone augmented with specific data processing and training techniques designed to enhance zero-shot generalization.

2.1 Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^T$ denote a univariate input time series of length T. To perform probabilistic predictions, our architecture takes $\mathbf{x}$ as input and predicts a set of quantiles $\hat{\mathbf{q}} \in \mathbb{R}^{T \times K}$ representing the distribution of the target values. This enables the model to capture the inherent uncertainty in time series data via a deterministic mapping.

To form semantically-meaningful tokens from real-value inputs, the time series $\mathbf{x}$ is segmented into non-overlapping patches $\mathbf{p} \in \mathbb{R}^{N \times L}$, where L is the patch size and $N = \frac{T}{L}$ denotes the number of patches (Nie et al., 2023). Note that we pad or truncate the raw inputs to make a constant length T that is always divisible by L.

2.2 Masking for Long-Horizon Prediction Training

When modeling real-valued continuous systems with neural networks, conventional training paradigms—such as single, step next-token prediction or scattered masked autoencoding, often fail to capture long-horizon dynamics. To mitigate this, various disciplines have adopted strategies that explicitly optimize for multi-step predictions over extended horizons (Hafner et al., 2019; Gloeckle et al., 2024). Within the context of TSFMs, an instantiation of this principle is termed Contiguous Patch Masking (CPM) (Auer et al., 2025).

We define a binary mask $\mathbf{m} \in \{0, 1\}^T$, where $m_t = 1$ indicates a masked observation and $m_t = 0$ indicates a visible observation. Unlike the conventional approaches that uniformly sample scattered masks (He et al., 2022; Woo et al., 2024), CPM selects random starting positions (multiples of L) and masks blocks of $N_{\text{cpm}} \cdot L$ consecutive observations, as illustrated by the blue shaded regions in Figure 2. This creates substantial "information gaps" between the visible context and the targets, compelling the model to learn the underlying dynamics of the time series to bridge these gaps.

[1] Model checkpoints: `https://huggingface.co/ibm-research/patchtst-fm-r1`; model code: `https://github.com/ibm-granite/granite-tsfm/tree/patchtst-fm`.

2.3 Patch Embedding and Transformer Backbone

We first apply a mask-aware variant of Reversible Instance Normalization (Kim et al., 2022) that computes statistics only over visible time points to prevent leakage from masked targets (see Appendix C.1 for details). The normalized patches $\bar{\mathbf{p}}$ and the masking $\mathbf{m}$ are projected into latent embeddings $\mathbf{h} \in \mathbb{R}^{N \times d}$ using a residual block (Das et al., 2024; Auer et al., 2025; Ansari et al., 2025), defined as

$$\mathbf{h}_{\text{out}} = f_{\text{res}}(\mathbf{h}_{\text{in}}) + f_{\text{out}}\big(\texttt{sigmoid}(f_{\text{in}}(\mathbf{h}_{\text{in}}))\big), \tag{1}$$

where f_{res}, f_{in}, f_{out} are linear layers. Learnable positional embeddings are then added to these patch embeddings. The resulting sequence of embeddings is processed by a stack of full-attention Transformer blocks (i.e., an encoder-only Transformer) with depth N_{layer} and embedding size d. We adhere to the standard Transformer formulation with pre-normalization.

2.4 Probabilistic Prediction

To generate probabilistic predictions, we employ a Quantile Head (Auer et al., 2025; Ansari et al., 2025), implemented as a residual block, which maps the latent representations to the quantile distribution of the patches. Each patch embedding $\mathbf{h}_i \in \mathbb{R}^d$ is decoded into the normalized quantile values of the corresponding patch $\hat{\mathbf{q}}_i \in \mathbb{R}^{L \times K}$. The predictions for all patches are concatenated into $\hat{\mathbf{q}} \in \mathbb{R}^{T \times K}$. For each $q_t \in \mathbb{R}^K$, the model predicts the values for a set of quantile levels $Q = \{\tau_1, \tau_2, \ldots, \tau_K\}$.

During inference, the predicted quantiles are scaled back using the stored statistics with $\mathbf{y} = \texttt{sinh}(\hat{\mathbf{q}}) \cdot \sigma + \mu$.

2.5 Training Objective

The model is trained to minimize the Pinball Loss (Quantile Loss), computed exclusively on the masked real values. Formally, the objective function is defined as:

$$\begin{aligned} &\mathcal{L}(\hat{\mathbf{q}}, \mathbf{x}) = \frac{1}{\sum_{t=1}^{T} \mathbb{1}(m_t = 1)} \sum_{t=1}^{T} \mathbb{1}(m_t = 1) \mathcal{L}_q(\hat{\mathbf{q}}_t, x_t), \\ &\text{where} \quad \mathcal{L}_q(\hat{\mathbf{q}}_t, x_t) = \sum_{\tau \in Q} (x_t - \hat{q}_t^{(\tau)}) \cdot (\tau - \mathbb{1}(x_t \leq \hat{q}_t^{(\tau)})). \end{aligned} \tag{2}$$

Here, $\hat{q}_t^{(\tau)}$ denotes the predicted value for quantile level τ at timestamp t.

2.6 Design Analysis

Our architectural design is guided by the following core principles:

- **Simplicity**: We deliberately adhere to the generic Transformer formulation, demonstrate its effectiveness for TSFMs, and study the scaling laws in Section 4.
- **Flexibility**: Our use of CPM with randomized positions and high masking ratios prevents the model from overfitting to a specific prediction horizon, which results in flexible forecast lengths (see Figure 7 and Appendix F.3).
- **Single-step Inference**: The encoder-only architecture enables parallel decoding of all target patches simultaneously. Unlike autoregressive models that generate predictions sequentially, our approach produces the entire forecast horizon in a single forward pass. This minimizes the propagation of prediction error or model's inherent bias, and thus maximizes the role of real context from data. (See Appendix F.5.)

3 Empirical Evaluations

In this section, we rigorously evaluate the performance of our proposed architecture. We begin by detailing the experimental setup, including model hyperparameters and the composition of our pretraining corpus. Then, we demonstrate the performance of our models on probabilistic forecasting on a public benchmark.

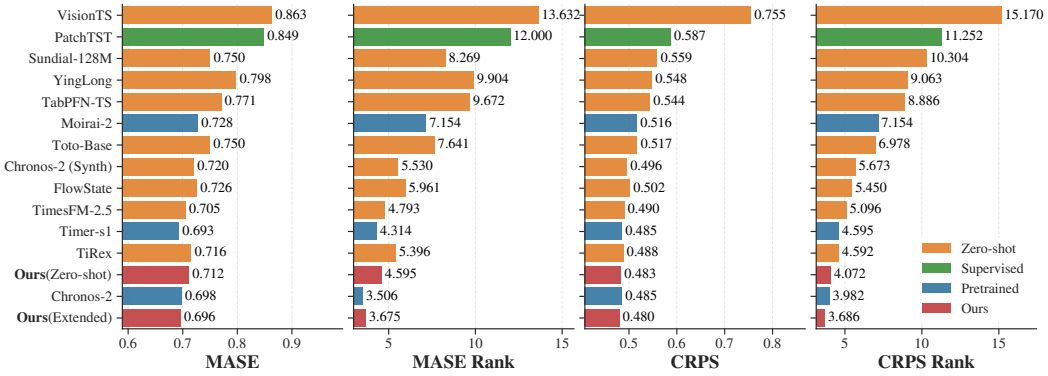

Figure 3: Forecasting performance on the GIFT-Eval benchmark. The four metrics (MASE, MASE Rank, CRPS, CRPS Rank) are aggregated over the 97 test cases. Lower values indicate better performance. The methods are sorted based on CRPS Rank. Ours (Extended) is trained on a corpus that overlaps 23 of the 97 test cases and is therefore not a zero-shot result (Section 3.1).

3.1 Experiment Setups

Model Configuration: The default hyperparameters, presented in Table 6, are selected heuristically based on established best practices in the Transformer literature. The detailed implementations are summarized in Appendix C, and Appendix F.1.6 summarizes the empirical evidence behind each choice.

Pretrain Data: A critical component of our study is the curation of a diverse and high-volume pretraining corpus. Our dataset comprises three sources:

- **Real Data**: We utilize the GIFT-Eval-Pretrain collection, consisting of 132 datasets from various domains totaling approximately 4.5 million time series and 230 billion observations.
- **KernelSynth** (Synthetic): To support long-context training ($T = 8192$), we generate 10 million univariate time series using the KernelSynth procedure (Ansari et al., 2024). As suggested by Auer et al. (2025), we employ a diverse set of periodic kernels and a small volume of augmentations.
- **TSMixup** (Augmented): To enhance temporal diversity, we incorporate TSMixup data (Ansari et al., 2024), generated by randomly mixing real-world time series. We evaluate two variants to study the impact of data leakage: (1) a "Leaky" set of 10 million series that includes series drawn from datasets represented in 23 of the 97 GIFT-Eval test cases, and (2) a "Clean" set of 4 million series with all test-set overlaps removed. This comparison allows us to quantify the true zero-shot capability of the model versus performance gains from domain exposure.

Evaluated Models: Based on these data collections, we present results for two distinct model variants:

- **Zero-shot Model**: Trained on Real Data + 10M KernelSynth + 4M Clean TSMixup. This model has no exposure to any test distribution and is used for all zero-shot comparisons on GIFT-Eval.
- **Extended Model**: Trained on Real Data + 10M KernelSynth + 10M Leaky TSMixup. Because its corpus overlaps 23 GIFT-Eval test cases, its GIFT-Eval results are not zero-shot; we report them as a diagnostic of the effect of broader data coverage and domain exposure, mark them with † in the tables, and note the overlap in the figure captions. On the TIME benchmark, none of the 98 cases is seen by either variant, so both models are evaluated zero-shot there.

3.2 Benchmark Results

To assess zero-shot probabilistic forecasting performance, we utilize the GIFT-Eval benchmark (Aksu et al., 2025), a widely adopted standard for TSFMs. The benchmark encompasses 55 diverse real-world datasets spanning 97 distinct test cases, covering multiple domains of time series data and various forecast lengths.

Compared Methods: We compare our approach against the current SOTA TSFMs on the GIFT-Eval leaderboard, selecting the most recent high-performing TSFM variants, including TabPFN-TS (Hoo

et al., 2024), VisionTS (Chen et al., 2025), Sundial (Liu et al., 2025), Moirai-2 (Woo et al., 2024), Toto (Cohen et al., 2025), YingLong (Wang et al., 2025), FlowState (Graf et al., 2026), TimesFM-2.5 (Das et al., 2024), TiRex (Auer et al., 2025), Timer-S1 (Liu et al., 2026), and Chronos-2 (Ansari et al., 2025). To provide a comprehensive context, we also include two statistical baselines (Naive and Seasonal Naive) and a supervised deep learning baseline PatchTST (Nie et al., 2023). The baseline results are obtained from the official GIFT-Eval Leaderboard. Detailed information about the baselines is summarized in Appendix D.2.

Note on Comparability: It is important to emphasize that the baseline TSFMs utilize heterogeneous training setups, varying significantly in pretraining data scale, composition, model configurations, and optimization recipes. Consequently, the benchmarking results below reflect the aggregate performance of the final model artifacts (the combination of architecture, data, and training) rather than the isolated efficacy of the architectures themselves. Our goal is not to claim architectural superiority, but to demonstrate that a generic architecture is sufficient to match the leading TSFMs when coupled with a robust training recipe.

Evaluation Metrics: Following the standard GIFT-Eval protocol, we report three key metrics:

- **Mean Absolute Scaled Error (MASE)**: Evaluates point forecasting accuracy by scaling the Mean Absolute Error of the predicted median (i.e., the 50th quantile $\hat{\mathbf{q}}^{(0.5)}$) against the error of a naive forecast.
- **Continuous Ranked Probability Score (CRPS)**: Assesses probabilistic forecasting quality by measuring the difference between the predicted cumulative distribution function and the ground truth observation. It effectively generalizes MAE to the probabilistic setting.
- **Rank**: Measures the relative standing of each method among all the compared methods above. Rankings are computed per-dataset based on MASE and CRPS and averaged over all cases. The models are sorted by CRPS Rank as our primary focus on probabilistic forecasting.

Then, on each test dataset and case, the average MASE and CRPS over all test samples are normalized w.r.t. the performance of Seasonal Naive. The metrics over the 97 cases are aggregated with geometric mean for MASE and CRPS, and arithmetic mean for Rank, as an overall evaluation of the methods on the GIFT-Eval benchmark (refer to Appendix D.1 and Aksu et al. (2025) for more details).

Results Analysis: Figure 3 summarizes the comparative performance on the GIFT-Eval benchmark. Our Zero-shot model ranks among the top TSFMs on all metrics. A paired bootstrap analysis over test cases (Appendix E) shows that its aggregate MASE and CRPS are statistically indistinguishable from those of the strongest baselines, Chronos-2, TimesFM-2.5, and Timer-S1: all 95% confidence intervals of the score ratios include 1. The Extended model obtains the best MASE and CRPS point estimates; since its corpus overlaps 23 test cases, we interpret this result as a diagnostic of domain exposure rather than as a zero-shot result (Section 4.1). These findings provide strong empirical evidence that a generic Transformer architecture, when paired with a robust training recipe, is sufficient to match the leading TSFMs on time-series forecasting. Furthermore, our results demonstrate that the effective utilization of existing open-source data can yield models that match the recent TSFMs which rely on proprietary synthetic generation pipelines.

4 Systematic Ablation Study

Recognizing that the generic Transformer architecture can match the leading TSFMs on the GIFT-Eval benchmark, we now turn to the critical question: *What are the essential ingredients for this success?* As discussed in the previous sections, the heterogeneous, sometimes even opaque, training recipes of existing TSFMs often obscure whether improvements stem from architectural innovations or simply from better data and training protocols.

In this section, we conduct a large-scale ablation study to systematically break down these factors. We vary one component at a time and analyze its contribution. By strictly controlling variables within our standardized training pipeline, we aim to provide a transparent, empirical "recipe" for training effective TSFMs.

Unless stated otherwise, the ablations in this section start from the Extended configuration and are evaluated on all 97 GIFT-Eval cases; they compare training choices within a single recipe and are not zero-shot results (Section 3.1). Throughout this section, the Rank metric is the CRPS Rank computed strictly within the context of each specific ablation experiment. For instance, when comparing a set

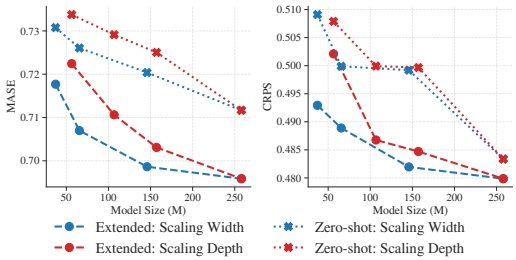

Figure 4: **Impact of model scaling on forecasting performance**: We compare increasing model capacity via embedding dimension (width) versus layer count (depth) for both the Extended and Zero-shot corpora (settings in Appendix F.1.5).

Table 1: Ablation study on **context length**. Extending the context yields significant improvements across all metrics. Models pretrained with longer context lengths provide more robust and general usage across diverse forecast lengths.

Ctx. Len.	MASE ↓	CRPS ↓	Rank ↓
2048	0.721	0.500	4.592
4096	0.717	0.497	4.501
8192	**0.696**	**0.480**	**3.381**

of ablation variants, their ranks are derived by comparing them against each other and the fixed set of baseline methods from Section 3.2. Consequently, the reported Rank values are relative measures intended for comparison within a specific table or figure, and should not be compared directly across different ablation studies.

4.1 Pretrain Data Composition

Table 12 and the purple line in Figure 1 investigate the impact of training data diversity on downstream performance. All variants share the same training steps, context length, and effective batch size, i.e., the same nominal token budget. We conclude the following:

1. **Insufficiency of Single Source**: Training on KernelSynth alone yields the poorest results, indicating that while synthetic periodic kernels capture basic seasonality, they fail to model complex real-world noise and non-stationary dynamics. Using only real data provides a stronger baseline, yet performance remains suboptimal.
2. **The "Synergy" Effect**: The Zero-shot configuration, combining real and synthetic data, delivers a substantial performance boost. This suggests a complementary relationship: synthetic data acts as a robust regularizer, filling the gaps between diverse real-world distributions, while the real data provides patterns that cannot be easily synthesized.
3. **Benefits of Domain Exposure**: The Extended model, which includes the broader leaky TSMixup set, achieves the best point estimates on GIFT-Eval. While the architecture generalizes well to unseen distributions (the Zero-shot variant), exposing the model to the target domain's patterns, even via mixed augmentations, provides an additional performance edge.

Quantifying the Impact of Leakage: The performance gap between our variants raises a critical question: *does the leaked data merely allow the model to memorize specific test cases, or does it contribute to better feature learning?* To investigate this, we excluded all 23 leaked test cases from the benchmark and compared the models exclusively on the remaining 74 unseen cases. As illustrated in Figure 5, both of our variants remain among the top models on this non-overlapping subset. Notably, the gap between the Zero-shot and Extended variants narrows, and even reverses slightly on the overall CRPS. To determine whether the increased data diversity still yields genuine zero-shot improvements, we evaluated both models on the TIME benchmark (Qiao et al., 2026), where no test case is seen by either model. As detailed in Table 4, the Extended model outperforms the Zero-shot model on both metrics, and the 95% intervals of the Extended/Zero-shot score ratios exclude 1 (Appendix E). Together, these results support our core finding: pretraining data composition has a substantial effect on benchmark performance, and expanding the diversity of the pretraining corpus improves generalization to unseen datasets, although these comparisons vary volume, composition, and domain exposure jointly and do not isolate any single factor.

4.2 Transformer Backbone Dimensions

We investigate the impact of model capacity on forecasting accuracy by independently scaling the Transformer backbone along two axes: (1) keeping $N_{\text{layer}} = 20$ and scaling the embedding

dimension $d \in \{384, 512, 768, 1024\}$, and (2) keeping $d = 1024$ and scaling the model depth $N_{\text{layer}} \in \{4, 8, 12, 20\}$. Figure 4 illustrates the scaling behavior across both MASE and CRPS metrics. We observe two key trends:

1. **Predictable Monotonic Scaling**: Performance improves consistently as parameter count increases, in line with the scaling behavior observed in other domains. This also suggests that the current architecture has not yet saturated and could benefit from further scaling.
2. **Depth versus Width**: At similar parameter counts, deep-narrow models consistently outperform shallow-wide models. This indicates that univariate TSFM benefits more from the hierarchical reasoning capabilities of a deep Transformer stack than from the increased feature dimension. Consequently, for parameter-efficient designs, prioritizing network depth yields better performance returns than expanding embedding size.

These sweeps characterize the accuracy–size trade-off within the Transformer backbone (see Appendix F.1.5 for the exact settings) rather than arguing for larger models per se, since larger models also incur higher training and inference cost.

4.3 Context Length

We evaluate the importance of historical information by varying the input context length $T \in \{2048, 4096, 8192\}$. Table 1 demonstrates a clear benefit of scaling the lookback window on the general forecasting performance. However, we observe that these gains are most pronounced in time series that exhibit long-term seasonality (i.e., where the seasonal period is large relative to the sampling rate). In such cases, shorter lookback windows often lead to ambiguity between periodic patterns and trends, leading to suboptimal predictions. (See Figure 12 for examples)

4.4 Pretrain Masking Strategy

Table 2: Impact of **Contiguous Patch Masking (CPM) block size**. The results show a significant performance improvement from CPM over scattered masking (i.e., $N_{\text{CPM}} = 1$).

Num. CPM	MASE ↓	CRPS ↓	Rank ↓
1	0.750	0.514	6.675
4	0.704	0.482	3.790
8	**0.696**	**0.480**	**3.566**

Table 3: Impact of **Masking Ratio**. The architecture remains relatively robust to this hyperparameter.

Mask Ratio	MASE ↓	CRPS ↓	Rank ↓
20%	0.698	0.484	**3.900**
40%	**0.696**	**0.480**	3.972
60%	0.702	0.484	4.420
80%	0.703	0.482	4.678

We analyze the design of the training objective by ablating the masking strategy, which serves as the primary mechanism for masked autoencoding, self-supervised learning. Based on results in Table 2 and Table 3, we conclude that:

- **Necessity of Training for Long-term Predictions**: Table 2 investigates the impact of the CPM block length $N_{\text{CPM}} \in \{1, 4, 8\}$. We observe that standard random patch masking (i.e., $N_{\text{CPM}} = 1$) yields poor performance. In contrast, introducing CPM significantly improves the performance by creating large "information gaps" and forcing the model to learn the underlying system dynamics for long-term predictions. Additionally, using larger N_{CPM} yields a performance edge by further pushing the model towards long-horizon understanding. (See Figure 13 for examples.)
- **Robustness to Mask Ratio**: Table 3 explores the optimal density of the masking signal. We find that the architecture is generally robust to the masking ratio, with no catastrophic failure observed even at high ratios (i.e., 80%). A ratio of 40% emerges as a "sweet spot", achieving slightly better MASE and CRPS.

4.5 Generalization to Other Benchmarks and Architectures

We demonstrate that our results are not limited to the GIFT-Eval benchmark. We evaluate our generic Transformer on the TIME benchmark, which includes 50 fresh datasets for zero-shot evaluation. As shown in Table 4, our models remain comparable with the leading TSFMs (bootstrap intervals in

Table 4: **Generalization to novel benchmarks.** Forecasting performance of our generic Transformer on the TIME benchmark, compared against SOTA TSFMs. Metrics are aggregated across 50 distinct datasets and 98 test cases.

	MASE ↓		CRPS ↓	
Model	**Score**	**Rank**	**Score**	**Rank**
Chronos-2	**0.660**	3.929	**0.549**	**3.796**
Ours (Extended)	**0.660**	**3.776**	**0.549**	4.071
Ours (Zero-shot)	0.669	5.224	0.556	4.724
TimesFM-2.5	0.667	5.112	0.559	5.510
Timer-S1	0.674	5.612	0.565	5.694
TiRex	0.681	6.286	0.565	5.827
Toto	0.696	8.408	0.576	7.643
Moirai2	0.701	8.735	0.580	8.204

Table 5: **Generalization across architectures.** Forecasting performance on the GIFT-Eval benchmark comparing top-tier TSFMs against four generic architectures trained with our training protocol (shaded rows).

		MASE ↓		CRPS ↓	
Model	**Size**	**Value**	**Rank**	**Value**	**Rank**
Transformer	33M	0.731	7.766	0.509	7.368
Toto	151M	0.750	8.181	0.517	7.544
Moirai2	14M	0.728	7.576	0.516	7.906
MLP-Mixer	30M	0.745	9.259	0.518	8.718
Bi-Mamba2	37M	0.755	9.517	0.518	8.994
TabPFN-TS	11M	0.771	10.856	0.544	9.911
YingLong	300M	0.798	11.113	0.548	10.144
Sundial	128M	0.750	9.103	0.559	11.711
CNN-ResNet	30M	0.791	12.447	0.555	12.297

Appendix E). This indicates that our training recipe yields robust models that transfer well to unseen domains and datasets.

Finally, we test if the training protocol generalizes to other backbone architectures. With the zero-shot protocol, we train three additional generic models that represent different types of basic inductive biases: MLP-Mixer (Ekambaram et al., 2023), bidirectional Mamba2 (Bi-Mamba2) (Dao and Gu, 2024), and a CNN-ResNet. All four backbones share the data recipe, training steps, context length, and batch size (i.e., the same nominal token budget) and are roughly matched in parameter count but not in FLOPs; each is a single run (Appendix F.1.5). Table 5 details their performance on the GIFT-Eval benchmark. Notably, when coupled with our training protocol, even the very basic architectures could match or surpass deliberately designed TSFMs. This supports our core claim: a rigorous training protocol can matter as much as architectural innovation for a foundation model's empirical success, and performance differences between models trained under different data and protocols cannot be attributed to architecture alone.

5 Discussion and Conclusion

Figure 1 visualizes the broader landscape of the capability of existing TSFMs on probabilistic forecasting, synthesizing our benchmarking results in Section 3.2 with the scaling dynamics observed in our ablations in Section 4. By comparing the performance trajectories of our generic Patch Transformer architecture against the current SOTA TSFMs, we conclude two insights regarding the current progress and direction of the field.

5.1 Architectural Progress

The scaling curves of our generic Patch Transformer show performance comparable to that of the recent SOTA TSFMs with similar sizes, matching other deliberately designed architectures. This suggests that for large-scale time series modeling and pretraining, the "vanilla" Transformer architecture remains a competitive choice by demonstrating high forecasting performance and scalability. Alternatively, we also observe distinct advantages of State-Space-Models (TiRex and FlowState) which demonstrate better parameter efficiency, achieving competitive results with more compact models. Notably, FlowState achieves strong performance with *only real data from GIFT-Eval-Pretrain*, suggesting that timestamp-level modeling via SSMs may offer a valuable direction for future research. However, the scalability of SSMs has not yet been demonstrated.

5.2 The Role of Data and Training Protocol

In Figure 1, the vertical trajectory (purple line) shows the substantial impact of data composition on the quality and performance of TSFMs, which is further supported by Section 4.1 and Figure 5. Furthermore, our ablations in Section 4 identify that specific training techniques, especially CPM and extended context lengths, are essential mechanisms for learning dynamics of time series and performing generalized predictions. Additionally, as presented in Section 4.5, we demonstrate that

other generic, standard architectures could also achieve comparable performance as the deliberately designed TSFMs when coupled with our training protocol. These findings suggest that the performance leaps observed in recent studies may stem, at least in part, from advancements in training methodology rather than from innovation in the core architecture.

5.3 Recommendations for Future Research

Based on our findings, we make following recommendations for the community of TSFM research:

- **Benchmark Artifacts and Architectures Separately**: We should distinguish between releasing high-utility checkpoints (an engineering goal) and discovering new architectures (a scientific goal). Releasing models trained on specific data is valuable, but it often obscures the true source of performance gains. To measure scientific progress accurately, we advocate evaluating architectures independently of their pretraining data. Future research should explicitly state whether contributions stem from architectural innovation or data curation.
- **Standardize the Pretraining Corpus**: To enable fair comparisons, the community needs a unified, publicly available pretraining corpus, analogous to ImageNet. Currently, heterogeneous pretraining datasets make cross-paper architectural comparisons virtually impossible. With our proven protocol as a proposal and potential starting point, we advocate for a standardized framework where models are trained on a fixed, shared dataset to isolate architectural innovations from data curation techniques.
- **Bridging Research and Application with High-Quality Benchmark Datasets**: Standardized benchmarks must not compromise model quality. Our results show that data diversity and broad domain coverage have a substantial effect on performance. Therefore, a unified pretraining corpus must be large and diverse enough to train models near "production-grade" levels, rather than relying on limited "toy" datasets. A high-quality standard corpus ensures that architectural improvements observed in research will reliably translate to real-world applications.

5.4 Conclusion

In this work, we revisited the generic Patch Transformer to evaluate its effectiveness as a Time Series Foundation Model. We demonstrated that this generic architecture, when supported by an effective training recipe, achieves performance statistically indistinguishable from that of the leading TSFMs on the GIFT-Eval and TIME forecasting benchmarks. Through extensive ablation studies, we examined the factors contributing to this success, finding that pretraining data composition, contiguous patch masking, and context length have the largest effect within our protocol.

These findings imply that the rapid leaderboard advancements observed in recent TSFMs may be partially confounded by non-uniform training protocols, which obscure the actual progress in architectural innovations. By establishing this robust baseline, we highlight the central role of data and training techniques in current benchmarks. We hope this work encourages the community to adopt more rigorous benchmarking protocols, redirecting focus toward architectural innovations that demonstrate clear value beyond data engineering.

Acknowledgments and Disclosure of Funding

This work was supported by IBM through the IBM-Rensselaer Future of Computing Research Collaboration (FCRC). The authors acknowledge the National Artificial Intelligence Research Resource (NAIRR) Pilot and Mass Open Cloud for providing computational resources that contributed to this work.

References

Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., Xiong, C., and Sahoo, D. (2025). GIFT-eval: A benchmark for general time series forecasting model evaluation.

Ansari, A. F., Shchur, O., Küken, J., Auer, A., Han, B., Mercado, P., Rangapuram, S. S., Shen, H., Stella, L., Zhang, X., et al. (2025). Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*.

Ansari, A. F., Stella, L., Turkmen, A. C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, B. (2024). Chronos: Learning the language of time series. *Transactions on Machine Learning Research*.

Auer, A., Podest, P., Klotz, D., Böck, S., Klambauer, G., and Hochreiter, S. (2025). 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*.

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M. K., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2024). xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chen, M., Shen, L., Li, Z., Wang, X. J., Sun, J., and Liu, C. (2025). VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In *Forty-second International Conference on Machine Learning*.

Cohen, B., Khwaja, E., Doubli, Y., Lemaachi, S., Lettieri, C., Masson, C., Miccinilli, H., Ramé, E., Ren, Q., Rostamizadeh, A., du Terrail, J. O., Toon, A.-M., Wang, K., Xie, S., Xu, Z., Zhukova, V., Asker, D., Talwalkar, A., and Abou-Amal, O. (2025). This time is different: An observability perspective on time series foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Dao, T. and Gu, A. (2024). Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Das, A., Kong, W., Sen, R., and Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Ekambaram, V., Jati, A., Nguyen, N., Sinthong, P., and Kalagnanam, J. (2023). Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pages 459–469.

Emami, P., Sahu, A., and Graf, P. (2023). Buildingsbench: A large-scale dataset of 900k buildings and benchmark for short-term load forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Gloeckle, F., Idrissi, B. Y., Rozière, B., Lopez-Paz, D., and Synnaeve, G. (2024). Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.

Godahewa, R. W., Bergmeir, C., Webb, G. I., Hyndman, R., and Montero-Manso, P. (2021). Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., and Dubrawski, A. (2024). Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*.

Graf, L., Ortner, T., Woźniak, S., and Pantazi, A. (2026). Flowstate: Sampling-rate-equivariant time-series forecasting. In *Forty-third International Conference on Machine Learning*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009.

Hoo, S. B., Müller, S., Salinas, D., and Hutter, F. (2024). The tabular foundation model tabPFN outperforms specialized time series forecasting models based on simple features. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

Howard, A., Yui, H., McDonald, M., and Cukierski, W. (2017). Recruit restaurant visitor forecasting. `https://kaggle.com/competitions/recruit-restaurant-visitor-forecasting`. Kaggle.

Jiang, J., Han, C., Jiang, W., Zhao, W. X., and Wang, J. (2023). Libcity: A unified library towards efficient and comprehensive urban spatial-temporal prediction. *arXiv preprint arXiv:2304.14343*.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. (2022). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104.

Liu, X., Xia, Y., Liang, Y., Hu, J., Wang, Y., Bai, L., Huang, C., Liu, Z., Hooi, B., and Zimmermann, R. (2023). LargeST: A benchmark dataset for large-scale traffic forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Liu, Y., Qin, G., Shi, Z., Chen, Z., Yang, C., Huang, X., Wang, J., and Long, M. (2025). Sundial: A family of highly capable time series foundation models. In *Forty-second International Conference on Machine Learning*.

Liu, Y., Su, X., Wang, S., Zhang, H., Liu, H., Wang, Y., Ye, Z., Xiang, Y., Wang, J., and Long, M. (2026). Timer-s1: A billion-scale time series foundation model with serial scaling. *arXiv preprint arXiv:2603.04791*.

Nguyen, T., Jewik, J. K., Bansal, H., Sharma, P., and Grover, A. (2023). Climatelearn: Benchmarking machine learning for weather and climate modeling. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*.

Palaskar, S., Ekambaram, V., Jati, A., Gantayat, N., Saha, A., Nagar, S., Nguyen, N. H., Dayama, P., Sindhgatta, R., Mohapatra, P., et al. (2024). Automixer for improved multivariate time-series forecasting on business and it observability data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 22962–22968.

Qiao, Z., Pan, S., Wang, A., Zhukova, V., Liu, Y., Jiang, X., Wen, Q., Long, M., Jin, M., and Liu, C. (2026). It's time: Towards the next generation of time series forecasting benchmarks. *arXiv preprint arXiv:2602.12147*.

Shen, S., Van Beek, V., and Iosup, A. (2015). Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing*, pages 465–474. IEEE.

Shi, X., Wang, S., Nie, Y., Li, D., Ye, Z., Wen, Q., and Jin, M. (2025). Time-moe: Billion-scale time series foundation models with mixture of experts. In *The Thirteenth International Conference on Learning Representations*.

Smith, J. T., Warrington, A., and Linderman, S. (2023). Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*.

Trindade, A. (2015). ElectricityLoadDiagrams20112014. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58C86.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Wang, X., Zhou, T., Gao, J., Ding, B., and Zhou, J. (2025). Output scaling: Yinglong-delayed chain of thought in a large pretrained time series forecasting model. *arXiv preprint arXiv:2506.11029*.

Wen, Y., Ma, T., Weng, T.-W., Nguyen, L. M., and Julius, A. A. (2024). Abstracted shapes as tokens - a generalizable and interpretable model for time-series classification. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. (2024). Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

A Related Work

To consume large-scale pretraining data, TSFMs adopt scalable architectures paired with explicit input processing or "tokenization" strategies. Following its success in computer vision and natural language processing, the Transformer architecture (Vaswani et al., 2017) also serves as the dominant backbone choice for TSFMs. For time series data, tokenization approaches can vary between timestamp-level and patch-level granularities.

For instance, Chronos (Ansari et al., 2024) discretizes values at the timestamp level to construct a "language of time series," enabling forecasting via a decoder-only autoregressive Transformer. TimesFM (Das et al., 2024) and Moirai (Woo et al., 2024) adopt patch-level tokenization to capture local semantics and improve computational efficiency, utilizing decoder-only and encoder-only (i.e., full-attention Transformer with masked inputs) architectures, respectively. Very recently, Chronos-2 (Ansari et al., 2025) extends the field from univariate TSFM to generic multivariate modeling and forecasting, while heavily relying on synthetic data. Beyond deterministic mapping, Sundial (Liu et al., 2025) integrates patch-level modeling with flow-matching decoding to provide more "natural" probabilistic forecasts from point predictions.

Recent works also explore state-space models as alternatives to the Transformer backbone, achieving high performance with relatively compact models. For example, TiRex (Auer et al., 2025) combines patch-level processing with the xLSTM (Beck et al., 2024) architecture. FlowState (Graf et al., 2026) formulates timestamp-level inputs as a discrete ODE using an S5 backbone (Smith et al., 2023), utilizing a specialized scaling factor to handle various sampling rates.

However, while these models continue to rapidly advance the SOTA results, their heterogeneous training protocols obscure the attribution of these gains. In particular, the increasing reliance on proprietary synthetic data generation pipelines, where the exact synthetic corpus is often not publicly released, introduces a significant uncontrolled variable. Consequently, it remains unclear whether recent improvements stem from the core architectural innovations or from superior data engineering (see Appendix D.2 for details).

B Broader Impacts

This work aims to support the development of TSFMs by establishing a transparent and reproducible baseline. We discuss the broader impacts of our contributions along three dimensions.

Establishing Transparent Baselines. Many existing TSFM studies emphasize final model performance while remaining opaque about their training protocols. Our results show that the protocol can be as essential as the architecture itself in determining downstream performance. By releasing our model checkpoints, documenting our training protocol in full, and reporting detailed empirical findings, we provide a strong and reproducible baseline that future work can build upon. We hope this encourages a research culture in which architectural contributions are reported alongside the training conditions under which they were obtained, so that genuine progress in architecture can be more reliably identified.

Advancing a Unified Pretraining Corpus. Fair benchmarking of TSFM architectures requires a high-quality and standardized pretraining corpus. Our pretraining data composition is openly documented and can be reproduced from public sources, which we view as an initial step toward such a standard. We hope this effort motivates the community to develop even larger and more diverse corpora in the future. Standardized evaluation benchmarks alone are not sufficient. A shared pretraining corpus is equally important for isolating the contribution of a new architecture from the contribution of its training data, analogous to the role that standardized benchmarks have played in accelerating progress in adjacent fields such as computer vision (Deng et al., 2009).

Lowering the Barrier to Entry. Developing a competitive TSFM currently requires substantial resources and considerable trial and error to identify an effective training recipe. This creates a high barrier for entry-level researchers and small research groups, and it makes it difficult to determine whether a new architecture is genuinely ineffective or simply trained under a suboptimal protocol. By providing a protocol that produces a generic Transformer competitive with models from leading

industrial groups, we hope to lower this barrier and enable a broader range of researchers to contribute to the field.

C Implementation Details

C.1 Mask-Aware Normalization

Heterogeneous value distribution and non-stationarity are major challenges in time-series modeling, where Reversible Instance Normalization (Kim et al., 2022) is proposed to mitigate such issues. In our masked autoencoding formulation, computing statistics including the masked (target) values would introduce information leakage about the target distribution.

To address this, we utilize Mask-Aware Normalization, which computes the mean μ and standard deviation σ using only the visible time points:

$$\mu = \frac{\sum_{t=1}^{T} \mathbb{1}(m_t = 0)\, x_t}{\sum_{t=1}^{T} \mathbb{1}(m_t = 0)}, \qquad \sigma = \sqrt{\frac{\sum_{t=1}^{T} \mathbb{1}(m_t = 0)\, (x_t - \mu)^2}{\sum_{t=1}^{T} \mathbb{1}(m_t = 0)}}. \tag{3}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. These statistics are stored and applied later for Reverse Normalization to restore the predicted quantiles into the original scale of the input data. Furthermore, we follow Ansari et al. (2025) to normalize the input time series with $\bar{\mathbf{x}} = \texttt{sinh}^{-1}\left(\frac{\mathbf{x}-\mu}{\sigma}\right)$, which suppresses the influence of extreme outliers.

C.2 Masking Strategy

We provide supplementary details regarding the input processing described in Section 2. The design of the input mask is critical for ensuring model flexibility and generalized usage.

Mask Processing In our implementation, the unified input mask $\mathbf{m}$ is a composite of three distinct components:

- Prediction mask $\mathbf{m}^{\text{pred}}$: Designates the observed values in the sequence that are dropped to serve as prediction targets. To optimize the model for forecasting tasks, with probability 0.5 we additionally mask a contiguous block of up to $4 \cdot N_{\text{CPM}}$ patches at the end of each training sample.
- Missing mask $\mathbf{m}^{\text{miss}}$: Identifies inherent missing values within the raw data. These values are retained in the attention mechanism (to preserve temporal position) but are marked as missing in $\mathbf{m}$ and strictly excluded from the loss calculation.
- Padding mask $\mathbf{m}^{\text{pad}}$: Identifies padding tokens used to fill the context window when the input sequence length is shorter than the context window T. We apply left-padding to align the sequence. Patches consisting entirely of padding values are masked out of the attention mechanism, where we empirically find this approach could improve performance by saving attention.

The model processes the logical union of these masks, defined as $\mathbf{m} = \mathbf{m}^{\text{pred}} \vee \mathbf{m}^{\text{miss}} \vee \mathbf{m}^{\text{pad}}$. Note that in Section 2 and Equation 2, we simplified the notation to assume $\mathbf{m} \approx \mathbf{m}^{\text{pred}}$ for clarity of presentation.

Adaptive Contiguous Patch Masking CPM masks blocks of $N_{\text{CPM}} \cdot L$ consecutive observations. However, applying a fixed block size to short input sequences can inadvertently mask the entire valid history. Therefore, we implement an Adaptive CPM strategy. Formally for a time series $\mathbf{x} \in \mathbb{R}^{T^{(x)}}$ with length $T^{(x)} \ll T$, we dynamically adjust the block size $N_{\text{CPM}}^{(x)}$ as follows:

$$N_{\text{CPM}}^{(x)} = \min\left(\left\lceil \frac{\lceil T^{(x)}/L \rceil}{4} \right\rceil, N_{\text{CPM}}\right).$$

This preserves sufficient context for the model to learn meaningful dependencies even in short samples.

C.3 Quantile Head and Inference Pipeline

Pipeline Given a raw series, we compute the normalization statistics μ and σ from the visible context only (Appendix C.1), apply the $\sinh^{-1}$ transform, segment the series into non-overlapping patches of length L, and mark the forecast positions as masked. The encoder processes the visible and masked patches jointly. The quantile head, implemented as a residual block, maps the embedding of each masked patch to $K = 99$ quantile values at levels $\{0.01, 0.02, \ldots, 0.99\}$ for every timestamp in the patch. During training, the pinball loss in Equation 2 is computed only on masked positions that correspond to real observations; missing and padded positions are excluded. At inference, all target patches are decoded in a single non-autoregressive forward pass, and the predicted quantiles are mapped back to the original scale with $\mathbf{y} = \texttt{sinh}(\hat{\mathbf{q}}) \cdot \sigma + \mu$.

Monotone Quantiles The head does not predict the K quantile values independently. For each timestamp t, it predicts an anchor value b_t and K raw increments $\delta_{t,1}, \ldots, \delta_{t,K}$, and outputs

$$\hat{q}_t^{(\tau_k)} = b_t + \sum_{j=1}^{k} \frac{\mathrm{softplus}(\delta_{t,j})}{K}, \qquad k = 1, \ldots, K, \tag{4}$$

so that $\hat{q}_t^{(\tau_k)} - \hat{q}_t^{(\tau_{k-1})} = \mathrm{softplus}(\delta_{t,k})/K > 0$ for $k = 2, \ldots, K$. The predicted quantiles are therefore ordered by construction, and no post-hoc sorting is required to prevent quantile crossing.

Non-overlapping Patches We use non-overlapping patches (stride equal to L) throughout. This keeps the token count at T/L and avoids redundant computation over overlapping windows, at the possible cost of fine-scale information at patch boundaries. Patch overlap was not ablated in this study; a patch-size ablation is reported in Appendix F.1.3.

C.4 Hyperparameters

Table 6: Detailed Hyperparameter Configuration. The values are selected based on standard practices in TSFMs and Transformers.

Hyperparameter	Value	Rationale / Source
Architecture		
Context Length (T)	8192	A large context window for general usage. (Ansari et al., 2025)
Patch Size (L)	16	Non-overlapping (stride $=$ L), fixed across datasets and frequencies, following PatchTST (Nie et al., 2023); see Appendix F.1.3.
Number of Layers (N_{layer})	20	Modified from ViT-Large.
Embedding Dimension (d)	1024	From ViT-Large.
Attention Head Dimension	64	Standard choice for Transformers.
Number of Attention Heads	$d/64$	Standard choice for Transformers.
Number of Quantiles (K)	99	
Quantile Levels Q	$\{0.01, 0.02, \ldots, 0.99\}$	
Training Configurations		
Masking Ratio	40%	An intermediate value based on BERT (Devlin et al., 2019) and MAE (He et al., 2022).
CPM Block Size (N_{CPM})	8	
Optimization		
Total Training Steps	100k	
Effective Batch Size	2048	
Optimizer	AdamW	Standard choice for foundation models.
Learning Rate Schedule	Cosine with Linear Warmup	Standard choice for foundation models.
Peak Learning Rate	$3 \cdot 10^{-4}$	Standard choice, conservative for training stability.
Minimum Learning Rate	$1 \cdot 10^{-5}$	
Weight Decay	0.1	Empirical choice to ensure training stability.
(β_1, β_2) in AdamW	$(0.9, .95)$	Empirical choice to ensure training stability.
Gradient Norm Clipping	1	Standard choice, conservative for training stability.
Precision	FP16 Mixed	Accelerate training, supported by V100 GPUs.

C.5 Alternate Generic Architectures

In this section, we detail the formulations of the three additional generic architectures discussed in Section 4.5 and Table 5. All three share the same normalization and patching pipeline as the Transformer backbone, differing only in how the resulting patch embeddings are processed. We deliberately instantiate each backbone in its most canonical form so that it cleanly represents a single type of architectural inductive bias, enabling a controlled comparison of their relative effectiveness. In practice, these primitives are often combined to achieve broader and more robust utility (e.g., SSM+MLP, CNN+MLP hybrids).

MLP-Mixer backbone. The backbone consists of L identical Mixer blocks operating on a univariate token sequence $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{model}}}$, where N is the number of patches and each token is the embedding of one patch. Each block alternates two MLPs that act along orthogonal axes of the token matrix: an *inter-patch* MLP applied across the patch dimension and an *intra-patch* MLP applied across the embedding dimension. Both sub-layers follow a pre-norm residual formulation:

$$\mathbf{U}_\ell = \mathbf{X}_\ell + \left[f_{\text{inter}}\big(\text{LN}(\mathbf{X}_\ell)^\top\big) \right]^\top, \tag{5}$$

$$\mathbf{X}_{\ell+1} = \mathbf{U}_\ell + f_{\text{intra}}(\text{LN}(\mathbf{U}_\ell)), \tag{6}$$

where f_{inter} and f_{intra} are two-layer MLPs with GELU activation and dropout. Transposition before and after f_{inter} exposes the patch axis as the input dimension of the MLP, allowing it to learn position-dependent mixing weights over the entire sequence in a single operation.

Bidirectional Mamba2 backbone. The backbone is a stack of L bidirectional Mamba-2 blocks operating on a token sequence $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{model}}}$. Each block applies two independent Mamba-2 selective state-space modules (Dao and Gu, 2024) to opposite traversal orders of the same layer-normalized input and combines their outputs additively within a single residual connection:

$$\mathbf{X}_{\ell+1} = \mathbf{X}_\ell + \text{SSM}_{\rightarrow}(\text{LN}(\mathbf{X}_\ell)) + \mathcal{R}(\text{SSM}_{\leftarrow}(\mathcal{R}(\text{LN}(\mathbf{X}_\ell)))), \tag{7}$$

where $\mathcal{R}$ denotes reversal along the patch axis and $\text{SSM}_{\rightarrow}, \text{SSM}_{\leftarrow}$ are two separately parameterized Mamba-2 modules. Summing rather than concatenating the two directional outputs preserves the channel dimension and avoids doubling the parameter count of downstream projections.

This bidirectional construction is motivated by the non-causal nature of the main modeling setup with the encoder-only Transformer: at inference time the entire context window is observed, so restricting the backbone to left-to-right state propagation would discard information that a unidirectional SSM cannot recover.

CNN-ResNet backbone. The backbone is a stack of L residual convolutional blocks operating on a token sequence $\mathbf{X} \in \mathbb{R}^{d_{\text{model}} \times N}$, where N is the number of patches. Each block applies two 1D convolutions with "same" (non-causal) padding, separated by a GELU activation and dropout, followed by a residual connection and a LayerNorm along the channel dimension:

$$\mathbf{X}_{\ell+1} = \text{LN}\Big(\mathbf{X}_\ell + \text{Conv}^{(2)}_{k_\ell}\Big(\sigma\Big(\text{Conv}^{(1)}_{k_\ell}(\mathbf{X}_\ell)\Big)\Big)\Big), \tag{8}$$

where σ denotes GELU and k_ℓ is the kernel size of the ℓ-th block. All convolutions preserve sequence length, so the temporal resolution is unchanged throughout the network.

To capture dependencies at multiple temporal scales, we adopt a coarse-to-fine kernel schedule that monotonically decreases with depth, with the default configuration $\{31, 21, 15, 11, 7, 3\}$ for $L = 6$. Early layers employ large kernels to aggregate broad context and expose low-frequency structure, while later layers use progressively smaller kernels to refine local, high-frequency patterns.

C.6 Experiment Framework

Our models, training pipeline, and experiments are implemented using PyTorch and the Hugging Face ecosystem. To ensure broad compatibility with open-source frameworks, we leverage standard libraries including the `PretrainedModel` class, `Trainer`, `Accelerate`, and `Datasets`. Unless noted below, pretraining was conducted on NVIDIA V100 GPUs using FP16 mixed-precision training to optimize computational efficiency.

Compute All pretraining runs reported in this paper, including every ablation variant, use the same training configuration (Table 6). With the exception noted below, they were run on 96 NVIDIA V100-32G GPUs for approximately 13 hours per run, i.e., 1,248 allocated V100 GPU-hours. The 96-GPU allocation was chosen for its host memory, which keeps the full corpus in memory, rather than for model parallelism. For reference, we reproduced the full-scale run on 8 NVIDIA H100-80G GPUs in about 18 hours (144 H100 GPU-hours), and in about 6.5 hours (52 H100 GPU-hours) with FlashAttention and by skipping computation on padding tokens; these H100 runs serve only as timing references, and none of the reported results uses their checkpoints. Owing to a change in the compute available to us, the patch-size ablation in Appendix F.1.3 was run later on 32 NVIDIA A100-80G GPUs with the same training configuration.

Data Sampling The pretraining corpus is organized as a collection of datasets: every dataset–frequency pair of GIFT-Eval-Pretrain, where each variate of a multivariate series is treated as a separate univariate series, together with the KernelSynth and TSMixup collections. Training samples are generated on the fly by the data loader in four steps, summarized in Algorithm 1:

- **Series selection.** Datasets are sampled with equal probability, except that small datasets with fewer than 100 series are down-weighted by a factor of 100; within a dataset, series are drawn uniformly. Each series is standardized by its own mean and standard deviation.
- **Frequency augmentation.** For real series with a known sampling frequency, the original resolution is kept with probability at least $1/2$; otherwise the series is mean-pooled to one of up to three coarser calendar frequencies (e.g., hourly to daily), which widens the range of seasonal periods seen during pretraining.
- **Window cropping.** A window is cut at a random position. With probability 0.2 its length is the full series length (capped at $T = 8192$); otherwise it is drawn uniformly between $\ell_{\min} = 64$ and the capped length.
- **Masking and padding.** Missing values are recorded in the missing mask, the adaptive CPM mask of Appendix C.2 is generated at the patch level, and the window and its masks are left-padded to length T.

Algorithm 1 Generation of one pretraining sample

Require: Corpus $\mathcal{D}$ with dataset weights $\{w_d\}$; context length T; patch size L; masking ratio ρ; CPM block size N_{CPM}; minimum window length $\ell_{\min}$

Sample a dataset $d \propto w_d$ and a series $\mathbf{x} \in d$ uniformly; standardize $\mathbf{x}$ ▷ series selection

if the sampling frequency of d is known **then**
    Sample a pooling factor $s \in \{1, 1, 1, s_1, s_2, s_3\}$ and set $\mathbf{x} \leftarrow \mathrm{MeanPool}(\mathbf{x}, s)$ ▷ frequency augmentation
end if

With probability 0.2 set $\ell \leftarrow \min(|\mathbf{x}|, T)$; otherwise sample $\ell \sim \mathcal{U}\{\ell_{\min}, \ldots, \min(|\mathbf{x}|, T)\}$

Sample a start $t_0 \sim \mathcal{U}\{0, \ldots, |\mathbf{x}| - \ell\}$ and crop $\mathbf{x} \leftarrow \mathbf{x}_{t_0:t_0+\ell}$ ▷ window cropping

$\mathbf{m}^{\text{miss}} \leftarrow \mathrm{isnan}(\mathbf{x})$; replace missing values by 0

$N \leftarrow \lceil \ell / L \rceil$; $N'_{\text{CPM}} \leftarrow \min(\lceil N/4 \rceil, N_{\text{CPM}})$ ▷ adaptive CPM

Sample $\lfloor \rho N / N'_{\text{CPM}} \rfloor$ block starts without replacement and mask N'_{CPM} consecutive patches from each

With probability 0.5, additionally mask up to $4N'_{\text{CPM}}$ patches at the end of the window ▷ forecast-style tail

Expand the patch-level mask to $\mathbf{m}^{\text{pred}}$; left-pad $\mathbf{x}$, $\mathbf{m}^{\text{pred}}$, $\mathbf{m}^{\text{miss}}$ to length T and record $\mathbf{m}^{\text{pad}}$

return $(\mathbf{x}, \mathbf{m}^{\text{pred}}, \mathbf{m}^{\text{miss}}, \mathbf{m}^{\text{pad}})$

D Benchmark and Baselines

D.1 GIFT-Eval

Evaluation Metrics Here, we provide the detailed formulations of the GIFT-Eval metrics.

MASE is a metric for point forecasting. It compares the prediction's absolute error against the mean absolute error of a Naive forecasting baseline. Formally, given the target $\mathbf{x} \in \mathbb{R}^T$ and predicted median $\hat{\mathbf{q}}^{(0.5)} \in \mathbb{R}^T$, the MASE is calculated as follows:

$$\texttt{MASE}(\hat{\mathbf{q}}^{(0.5)}, \mathbf{x}) = \frac{\frac{1}{T}\sum_{t=1}^{T} \left|x_t - \hat{q}^{(0.5)}\right|}{\frac{1}{T-1}\sum_{t=2}^{T} |x_t - x_{t-1}|}. \tag{9}$$

CRPS evaluates the accuracy of probabilistic forecasts. CRPS measures the difference between the predicted cumulative distribution functions (CDFs) and the target values. In practice, computing the CDFs via integration is computationally expensive. The GIFT-Eval benchmarking protocol approximates the CRPS using a discrete sum over a finite set of quantile levels, often referred to as the mean weighted quantile loss.

Let $\hat{\mathbf{q}} \in \mathbb{R}^{T \times K}$ denote the predicted values for the quantile levels $Q = \{\tau_1, \ldots, \tau_K\}$. The approximated CRPS is defined as follows:

$$\texttt{CRPS}(\hat{\mathbf{q}}, \mathbf{x}) \approx \frac{1}{K}\sum_{k=1}^{K} \texttt{wQL}[\tau_k], \text{ where } \texttt{wQL}[\tau_k] = 2\frac{\sum_{t=1}^{T}(\tau_k - \mathbb{1}(x_t < \hat{q}_t^{(\tau_k)}))(x_t - \hat{q}_t^{(\tau_k)})}{\sum_{t=1}^{T}|x_t|}. \tag{10}$$

Dataset Information The GIFT-Eval benchmark contains 55 datasets from 7 macro domains, including Nature, Web and Cloud Applications, Sales, Energy, Transportation, Economy and Finance, and Healthcare. The datasets are obtained from popular time series forecasting benchmarks, including Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), AutoMixer (Palaskar et al., 2024), LibCity (Jiang et al., 2023), LSTNet (Lai et al., 2018), Monash Archive (Godahewa et al., 2021), Grid Workloads Archive (Shen et al., 2015), and Restaurant Competition (Howard et al., 2017), and UCI ML Archive (Trindade, 2015). Please refer to Aksu et al. (2025) for the detailed information about the GIFT-Eval and GIFT-Eval-Pretrain datasets.

D.2 Baselines

Table 7: Comparison of recent foundation models. Pretraining corpora sources are summarized for brevity. Publicly-available data are highlighted with blue. Corpus overlap with the GIFT-Eval test cases are highlighted with red.

Model	Size	Arch.	Context Len.	Pretrain Corpus
TabPFN-TS (Hoo et al., 2024)	11M	Encoder	4096[(1)]	Synthetic (Gaussian Process, Structured Causal Models).
Sundial (Liu et al., 2025)	128M	Decoder	2880	Subset of Chronos, LOTSA, Synthetic, Finance, IoT.
YingLong (Wang et al., 2025)	300M	Encoder	8192	Subset of Chronos, KernelSynth.
Moirai-2 (Woo et al., 2024)	14M	Encoder	512	GIFT-Eval-Pretrain, Training split of GIFT-Eval, TSMixup, KernelSynth, Internal Operations Data. Curated to remove the "low-quality" samples.
Toto (Cohen et al., 2025)	151M	Decoder	4096	Observability Metrics, GIFT-Eval-Pretrain, Subset of Chronos, Synthetic.
FlowState (Graf et al., 2026)	9.1M	S5	2048	GIFT-Eval-Pretrain.
TimesFM-2.5 (Das et al., 2024)	200M	Decoder	16384	GIFT-Eval-Pretrain, Wiki Pageviews, Google Trends, Synthetic.
TiRex-1.1 (Auer et al., 2025)	35M	xLSTM	2048[(2)]	Subset of GIFT-Eval-Pretrain, Subset of Chronos, Synthetic.
Chronos-2 (Ansari et al., 2025)	120M	Encoder	8192	Chronos, High-volume Synthetic.

Table 7 summarizes the key specifications of the baseline Time Series Foundation Models (TSFMs) evaluated in this study, including model size, backbone architecture, context length, and pretraining corpus composition. This information was collated from the original publications, official GitHub repositories, and Hugging Face model cards. We note the following specific configurations:

1. **TabPFN:** While the model supports variable input lengths, we report a context length of 4096, consistent with the default setting in the official evaluation code.
2. **TiRex:** We report the context length of 2048 based on the initial version described in Auer et al. (2025). Although TiRex-1.1 introduces updates to address data leakage in the original release and some improvements, the context length is not explicitly redefined in the documentation; we therefore assume it remains unchanged.
3. **Synthetic Data**: The inclusion of synthetic data has become a standard practice for pretraining TSFMs. Notably, many recent works generate their own synthetic corpora. While the generation methodologies are often briefly described in technical reports or publications, the exact synthetic datasets are rarely released.

E Statistical Analysis of Benchmark Results

The GIFT-Eval and TIME protocols report aggregate scores over 97 and 98 test cases, respectively, where a case is a dataset $\times$ frequency $\times$ horizon combination. Since several of the gaps in Figure 3 and Table 4 are small, we quantify the uncertainty of these aggregate comparisons across benchmark cases. All numbers below are computed from the case-level results of each model.

E.1 Protocol

Normalization and Aggregation For case i, metric $m \in \{\text{MASE}, \text{CRPS}\}$, and model M, let $s_{M,i}^{(m)}$ denote the case-level score. Following the official protocols, each score is normalized by the same-case Seasonal-Naive score,

$$x_{M,i}^{(m)} = \frac{s_{M,i}^{(m)}}{s_{\text{SN},i}^{(m)}}, \tag{11}$$

and aggregated over a set of cases I with the geometric mean, $A_M^{(m)}(I) = \exp\left(\frac{1}{|I|}\sum_{i\in I} \log x_{M,i}^{(m)}\right)$.

Paired Stratified Cluster Bootstrap Cases from the same dataset–frequency pair share data and are not independent. We therefore treat each dataset–frequency pair as a cluster and keep all of its horizons together when resampling. Clusters with one horizon and clusters with three horizons are resampled separately (with replacement), so that each bootstrap replicate preserves the composition of the benchmark. GIFT-Eval has 34 one-horizon and 21 three-horizon clusters; the 74-case non-overlapping subset has 26 and 16; TIME has 26 and 24. For each replicate b with resampled multiset of cases I_b, the same I_b is used for both models of a comparison, and we compute the ratio of aggregate scores

$$R_{A/B}^{(m,b)} = \frac{A_A^{(m)}(I_b)}{A_B^{(m)}(I_b)}. \tag{12}$$

We draw 10,000 replicates (seed 20260728) and report the ratio on the full benchmark together with its 2.5th and 97.5th bootstrap percentiles as a 95% confidence interval. A ratio below 1 favors the numerator model, and an interval that contains 1 indicates that the two models are not statistically distinguishable at this level.

Scope The intervals are pointwise and are not adjusted for multiple comparisons. They capture variation across benchmark cases, not variation due to initialization, data ordering, or optimization: each full-scale variant corresponds to a single pretraining run, and we do not repeat pretraining across seeds.

E.2 GIFT-Eval

Table 8 compares the Zero-shot model with the three strongest baselines on the 97 GIFT-Eval cases. All six intervals include 1; the same analysis on the 74 non-overlapping cases leads to the same conclusion. The Zero-shot model is therefore competitive with, but not distinguishable from, these baselines. Case-level outputs of the Extended model on GIFT-Eval are not available, so it is not included here.

Table 8: Paired cluster-bootstrap 95% confidence intervals for the ratio of aggregate scores on **GIFT-Eval** (97 cases). A ratio below 1 favors the numerator model.

Comparison	MASE ratio [95% CI]	CRPS ratio [95% CI]
Zero-shot / Chronos-2	1.013 [0.987, 1.041]	0.995 [0.965, 1.022]
Zero-shot / TimesFM-2.5	1.003 [0.977, 1.032]	0.985 [0.958, 1.012]
Zero-shot / Timer-S1	1.020 [0.982, 1.062]	0.995 [0.963, 1.031]

E.3 TIME

Table 9 reports the same analysis on the 98 TIME cases, where no case is seen by either of our models. The Extended model improves over the Zero-shot model on both metrics with intervals that exclude 1, and likewise over Timer-S1; it is on par with Chronos-2 on both metrics and with TimesFM-2.5 on MASE, while its CRPS interval against TimesFM-2.5 lies below 1. The Zero-shot model is on par with Chronos-2 and TimesFM-2.5 and improves over Timer-S1.

Table 9: Paired cluster-bootstrap 95% confidence intervals for the ratio of aggregate scores on **TIME** (98 cases). A ratio below 1 favors the numerator model.

Comparison	**MASE ratio [95% CI]**	**CRPS ratio [95% CI]**
Extended / Zero-shot	0.986 [0.975, 0.995]	0.988 [0.977, 0.997]
Extended / Chronos-2	1.000 [0.986, 1.016]	1.001 [0.986, 1.017]
Extended / TimesFM-2.5	0.990 [0.976, 1.005]	0.981 [0.965, 0.998]
Extended / Timer-S1	0.943 [0.905, 0.975]	0.946 [0.911, 0.975]
Zero-shot / Chronos-2	1.014 [0.997, 1.032]	1.013 [0.997, 1.030]
Zero-shot / TimesFM-2.5	1.004 [0.987, 1.021]	0.994 [0.976, 1.012]
Zero-shot / Timer-S1	0.956 [0.916, 0.990]	0.958 [0.923, 0.987]

E.4 Dispersion and Ranks across Cases

Tables 10 and 11 summarize how the normalized scores and the within-case ranks vary across cases. For each model, we report the sample standard deviation of $x_{M,i}^{(m)}$ over cases and the minimum, first quartile, median, third quartile, and maximum of its within-case rank. Ranks on GIFT-Eval are computed among the 14 models of Figure 3 with complete matched outputs, and ranks on TIME among the six models of Table 4 for which case-level results are available; ties receive average ranks. On both benchmarks, every model ranks first on some cases and near the bottom on others, which reflects the heterogeneity of the benchmarks rather than a consistent ordering of the models.

Table 10: Case-level dispersion on **GIFT-Eval** (97 cases). SD is the sample standard deviation of the Seasonal-Naive-normalized score across cases; ranks are min/Q25/median/Q75/max among 14 models.

Model	**MASE SD**	**MASE ranks**	**CRPS SD**	**CRPS ranks**
Zero-shot	0.202	1/2/4/7/14	0.198	1/2/5/7/14
Chronos-2	0.179	1/2/3/6/14	0.192	1/2/4/6/14
TimesFM-2.5	0.181	1/3/5/7/11	0.204	1/3/5/8/13
Timer-S1	0.187	1/2/5/7/12	0.217	1/2/5/8/13

Table 11: Case-level dispersion on **TIME** (98 cases). SD is the sample standard deviation of the Seasonal-Naive-normalized score across cases; ranks are min/Q25/median/Q75/max among six models.

Model	**MASE SD**	**MASE ranks**	**CRPS SD**	**CRPS ranks**
Zero-shot	0.175	1/2/3/5/6	0.176	1/2/3/5/6
Extended	0.159	1/1.25/3/4/6	0.170	1/2/3/4/6
Chronos-2	0.150	1/1/2/4/6	0.163	1/1/3/4/6
TimesFM-2.5	0.152	1/2/4/5/6	0.163	1/3/4/5.75/6
Timer-S1	0.205	1/3/5/6/6	0.190	1/3/4/5/6

F Additional Experimental Results and Visualizations

F.1 Additional Benchmarking Results

F.1.1 Pretraining Data Composition

Table 12: Ablation analysis of **pretraining data composition**. We compare models trained on individual data sources and composite datasets, as introduced in Section 3.1. †The Extended corpus overlaps 23 GIFT-Eval test cases (Section 3.1).

Pretrain Data	MASE ↓	CRPS ↓	Rank ↓
KernelSynth	0.832	0.584	12.220
Real Only	0.790	0.542	7.008
Zero-shot	0.712	0.483	4.267
Extended†	**0.696**	**0.480**	**3.383**

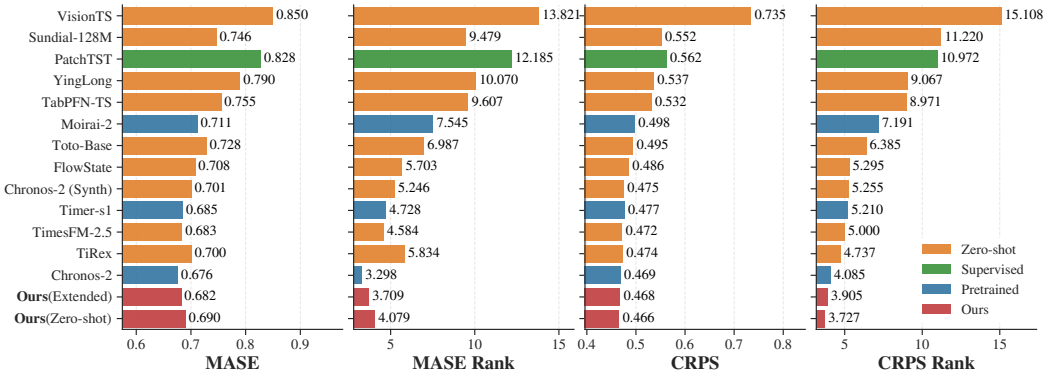

Figure 5: **Zero-shot** forecasting performance on the GIFT-Eval benchmark. The three metrics (MASE, CRPS, Rank) are aggregated with geometric mean over the **74 non-overlapping** test cases, none of which is seen by either of our models. Lower values indicate better performance. The methods are sorted based on Aggregated Rank.

F.1.2 Output Granularity

Table 13: Ablation study on **output probability granularity**. We compare the performance of models trained to predict 9, 21, and 99 quantiles.

Num. Quantiles	MASE ↓	CRPS ↓	Rank ↓
9	0.699	0.482	3.758
21	0.700	0.481	4.087
99	**0.696**	**0.480**	**3.598**

We investigate the impact of the output distribution's resolution by varying the number of predicted quantiles $K = \{9, 21, 99\}$ with the corresponding quantile levels:

- $K = 9$: $Q = \{0.1, 0.2, \ldots, 0.9\}$,
- $K = 21$: $Q = \{0.01, 0.05, 0.1, \ldots, 0.9, 0.95, 0.99\}$,
- $K = 99$: $Q = \{0.01, 0.02, \ldots, 0.99\}$.

As shown in Table 13, the three variants result in very similar performance, while $K = 99$ has slight advantage over the coarser predictions. This indicates that while a coarse set of quantiles is sufficient to capture the general distributional, higher granularity is more general and precise for fine-grained predictions. Note that, for fair comparison, the CRPS score of the three variants are computed only on the quantile levels $\{0.1, 0.2, \ldots, 0.9\}$.

F.1.3 Patch Size

Table 14: Ablation study on **patch size** L on GIFT-Eval. Tokens is the number of patches at the default context length $T = 8192$. The three models form a separate set of pretraining runs; see the text for the training setup.

Patch Size	Tokens	MASE ↓	CRPS ↓	Rank ↓
8	1024	**0.714**	**0.487**	5.495
16	512	0.719	**0.487**	**5.340**
32	256	0.724	0.497	6.722

Table 15: Patch-size ablation broken down by sampling frequency and forecast term. Each entry is the Seasonal-Naive-normalized geometric mean over the cases in the group. Minute-level groups the 5-, 10-, and 15-minute datasets; Monthly+ groups the monthly, quarterly, and yearly datasets.

		MASE ↓			CRPS ↓		
Group	**Cases**	$L = 8$	$L = 16$	$L = 32$	$L = 8$	$L = 16$	$L = 32$
10-second	6	**0.676**	0.705	0.715	**0.478**	0.546	0.514
Minute-level	30	0.734	**0.723**	0.737	0.500	**0.489**	0.511
Hourly	31	**0.693**	0.696	0.697	0.434	**0.432**	0.439
Daily	15	0.661	**0.660**	0.671	0.416	**0.411**	0.418
Weekly	8	0.779	**0.773**	0.798	0.624	**0.621**	0.631
Monthly+	7	**0.823**	0.906	0.843	**0.780**	0.809	0.811
Short	55	**0.687**	0.699	0.701	**0.508**	0.509	0.516
Medium	21	0.736	**0.735**	0.741	**0.458**	0.467	0.475
Long	21	0.770	**0.756**	0.770	0.464	**0.455**	0.469

We vary the patch size $L \in \{8, 16, 32\}$ while keeping the context length, data recipe, and all other hyperparameters fixed, so that the number of tokens per sample changes from 1024 to 256. This ablation was conducted after the main experiments on a different cluster (32 NVIDIA A100-80G GPUs instead of 96 V100-32G GPUs; see Appendix C.6) under the Zero-shot data recipe, so the three runs are compared with each other rather than with the ablations above; the $L = 16$ run therefore differs slightly from the Zero-shot model in Figure 3. Table 14 reports the GIFT-Eval aggregates. $L = 8$ and $L = 16$ obtain the same CRPS and differ by 0.005 in MASE; across the 97 cases, each is better than the other on roughly half of them (49 vs. 48 cases on CRPS), and 74 cases differ by less than 5%. $L = 32$ is consistently worse: its CRPS is 2% higher, and it loses to $L = 16$ on 65 of the 97 cases.

Table 15 breaks the comparison down by sampling frequency and forecast term. On the 84 cases sampled at minute to weekly resolution, $L = 8$ and $L = 16$ differ by only 1–2%, $L = 32$ is within 2% of $L = 16$ except on minute-level data (4.5% in CRPS), and $L = 16$ is the best or on par with the best. The advantage of $L = 8$ is concentrated in the six 10-second cases, where it improves CRPS over $L = 16$ by 24% at the medium and 10% at the long horizon, and in short-horizon forecasts, where it is slightly better; at long horizons on minute-level data it is about 6% worse than $L = 16$. The degradation of $L = 32$ is uniform across terms (about 1.4% in median CRPS) and most visible on minute-level data. The Monthly+ group contains only seven cases and is dominated by two of them (`car_parts` and `us_births`), so we do not draw conclusions from it. Overall, since $L = 8$ doubles the token count without an aggregate gain, the choice $L = 16$ is still a reasonable default.

F.1.4 Scaling Training Budget

We investigate the relationship between training duration and downstream performance by evaluating checkpoints throughout the pretraining process (see Figure 6). The results show a steady improvement in both MASE and CRPS. The trajectories suggest that the models have not yet reached their asymptotic limit, implying that further performance gains can be achieved simply by extending the training budget.

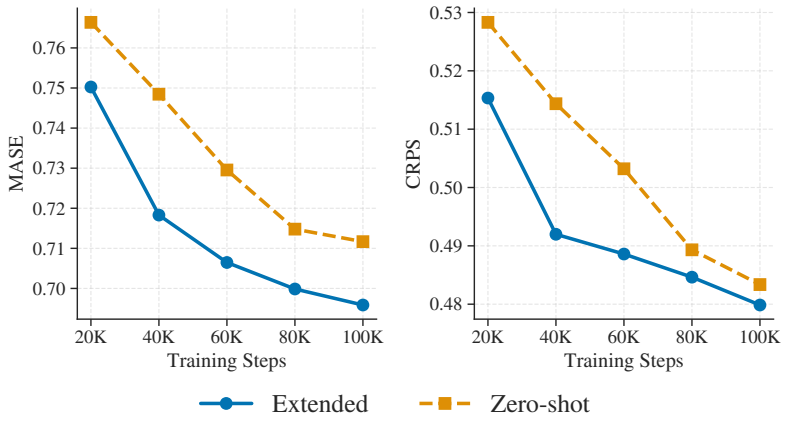

Figure 6: **Impact of training scaling on forecasting performance**: We evaluate the intermediate model checkpoints during training. MASE and CRPS for both the Extended and Zero-shot variants exhibit a consistent monotonic improvement.

F.1.5 Details of the Scaling Study

Figure 4 reports two one-factor sweeps for each pretraining corpus (Zero-shot and Extended). The width sweep fixes $N_{\text{layer}} = 20$ and varies $d \in \{384, 512, 768, 1024\}$; the depth sweep fixes $d = 1024$ and varies $N_{\text{layer}} \in \{4, 8, 12, 20\}$. All runs use the same data recipe, 100k training steps, context length 8192, and effective batch size; they are therefore matched in training steps and nominal token budget but not in FLOPs, and each point is a single pretraining run. The sweeps characterize the accuracy–size trade-off within the Transformer backbone and should not be read as a universal scaling law.

F.1.6 Summary of Design Choices

We summarize the empirical evidence behind the default configuration in Table 6.

- **Model size** (Figure 4): at similar parameter counts, deeper and narrower variants perform better, so parameter allocation matters beyond the total size.
- **Context length** (Table 1): increasing T from 2048 to 4096 to 8192 lowers CRPS from 0.500 to 0.497 to 0.480.
- **CPM block size** (Table 2): $N_{\text{CPM}} = 1/4/8$ gives CRPS 0.514/0.482/0.480; most of the gain is obtained by $N_{\text{CPM}} = 4$.
- **Masking ratio** (Table 3): 40% gives the best MASE and CRPS, while 20% gives the best rank; the preferred ratio depends on the metric.
- **Number of quantiles** (Table 13): 9/21/99 quantiles give CRPS 0.482/0.481/0.480; the gain from denser quantiles is small.
- **Training steps** (Figure 6): both metrics improve monotonically from 20k to 100k steps within a single run.
- **Patch size** (Table 14, separate runs): $L = 8$ and $L = 16$ give the same CRPS (0.487) and $L = 32$ is worse (0.497); $L = 16$ is a reasonable default choice since $L = 8$ doubles the token count without an aggregate gain.

F.2 When will it fail? Explore the maximum supported forecast length

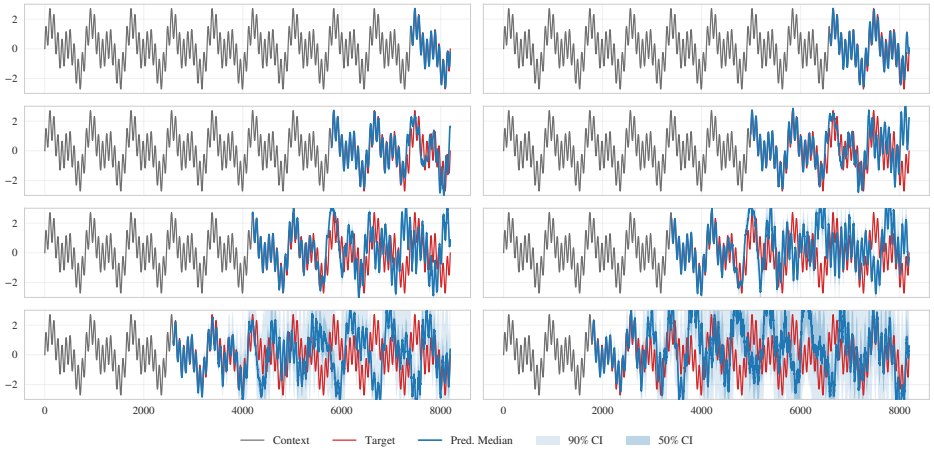

Figure 7: Examples of predictions on a synthetic signal with different predictions lengths.

Our generalized pretraining masking strategy offers the distinct advantage of flexible forecast lengths. In Figure 7, we investigate the effective maximum forecasting horizon supported by the model. We evaluate performance on a synthetic multi-frequency sine wave across forecast lengths $\{800, 1600, \ldots, 6400\}$. We observe that while predictions remain accurate for shorter horizons $\{800, 1600\}$, performance gradually degrades as the forecast length increases. These findings imply the following:

- For optimal performance, the maximum reliable forecast length is approximately 2000 timestamps. Extending beyond this limit can be achieved with autoregressive decoding.
- Neural models may struggle to extrapolate "simple" periodic cases over long horizons, whereas such patterns are often trivially predicted by classical statistical models.

F.3 Horizon Dependence on Real Data

To complement the synthetic stress test above, Table 16 reports absolute case-level MASE and CRPS on three TIME datasets at their short, medium, and long horizons, for our two models and the three strongest baselines. The three datasets were chosen to span different sampling frequencies (hourly and 20-minute), horizon lengths (9 to 336 steps), and domains. The effect of the horizon depends on both the dataset and the metric: CRPS decreases with the horizon on `Australia_Solar/H` for all five models, whereas both errors increase with the horizon on `Coastal_T_S/20T` and `CPHL/H`. The ordering of the models also changes across datasets and horizons. We therefore do not draw general conclusions about horizon dependence from these three examples.

Table 16: Case-level MASE and CRPS on three TIME datasets at short (S), medium (M), and long (L) horizons. Values are absolute (not normalized) case-level scores; lower is better.

Dataset (S/M/L)	**Model**	**MASE-S**	**MASE-M**	**MASE-L**	**CRPS-S**	**CRPS-M**	**CRPS-L**
Australia_Solar/H (24/72/168)	Extended	0.5746	0.6031	0.5789	0.2320	0.2102	0.1972
	Zero-shot	0.6278	0.6745	0.6832	0.2322	0.2215	0.2193
	Chronos-2	0.5689	0.6116	0.6182	0.2583	0.2250	0.2103
	TimesFM-2.5	0.6053	0.6114	0.6012	0.2657	0.2237	0.2067
	Timer-S1	0.5861	0.5986	0.5776	0.2519	0.2215	0.2038
Coastal_T_S/20T (9/72/216)	Extended	0.4701	0.9593	1.5430	0.0018	0.0040	0.0062
	Zero-shot	0.4421	0.9737	1.5763	0.0017	0.0042	0.0067
	Chronos-2	0.4837	0.9894	1.4011	0.0019	0.0041	0.0055
	TimesFM-2.5	0.4038	0.9137	1.3247	0.0015	0.0038	0.0055
	Timer-S1	0.5812	1.0121	1.5084	0.0022	0.0041	0.0060
CPHL/H (24/168/336)	Extended	1.2066	1.7346	2.0068	0.2201	0.3007	0.3402
	Zero-shot	1.2206	1.7413	1.9741	0.2214	0.2977	0.3302
	Chronos-2	1.1943	1.7754	2.0681	0.2199	0.3050	0.3557
	TimesFM-2.5	1.2108	1.7389	1.9857	0.2182	0.2931	0.3347
	Timer-S1	1.2209	1.7995	2.0225	0.2226	0.3141	0.3645

F.4 Visualizations of Predictions

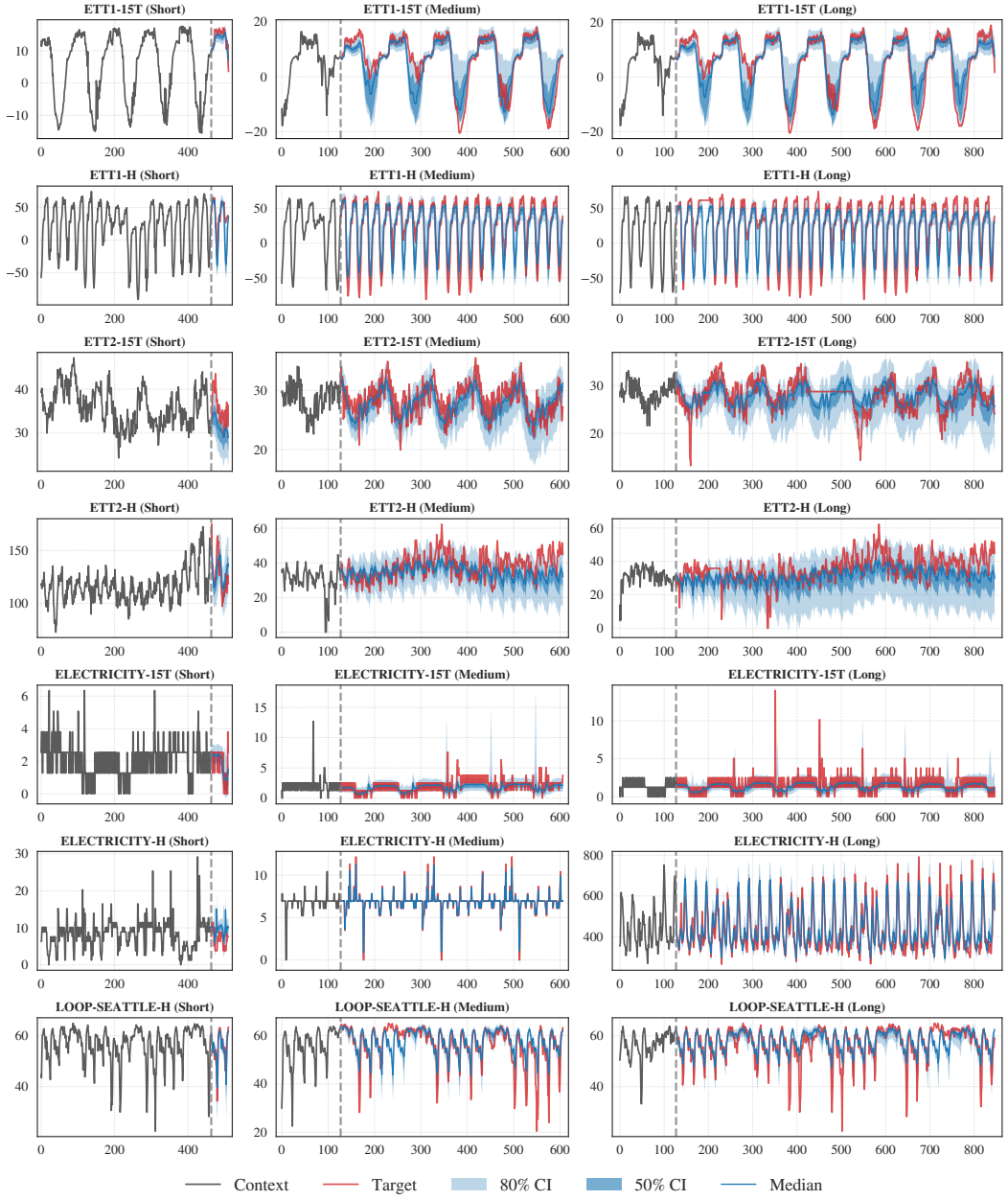

Figure 8: Examples of predictions from our **Extended** model on datasets with significant seasonal patterns.

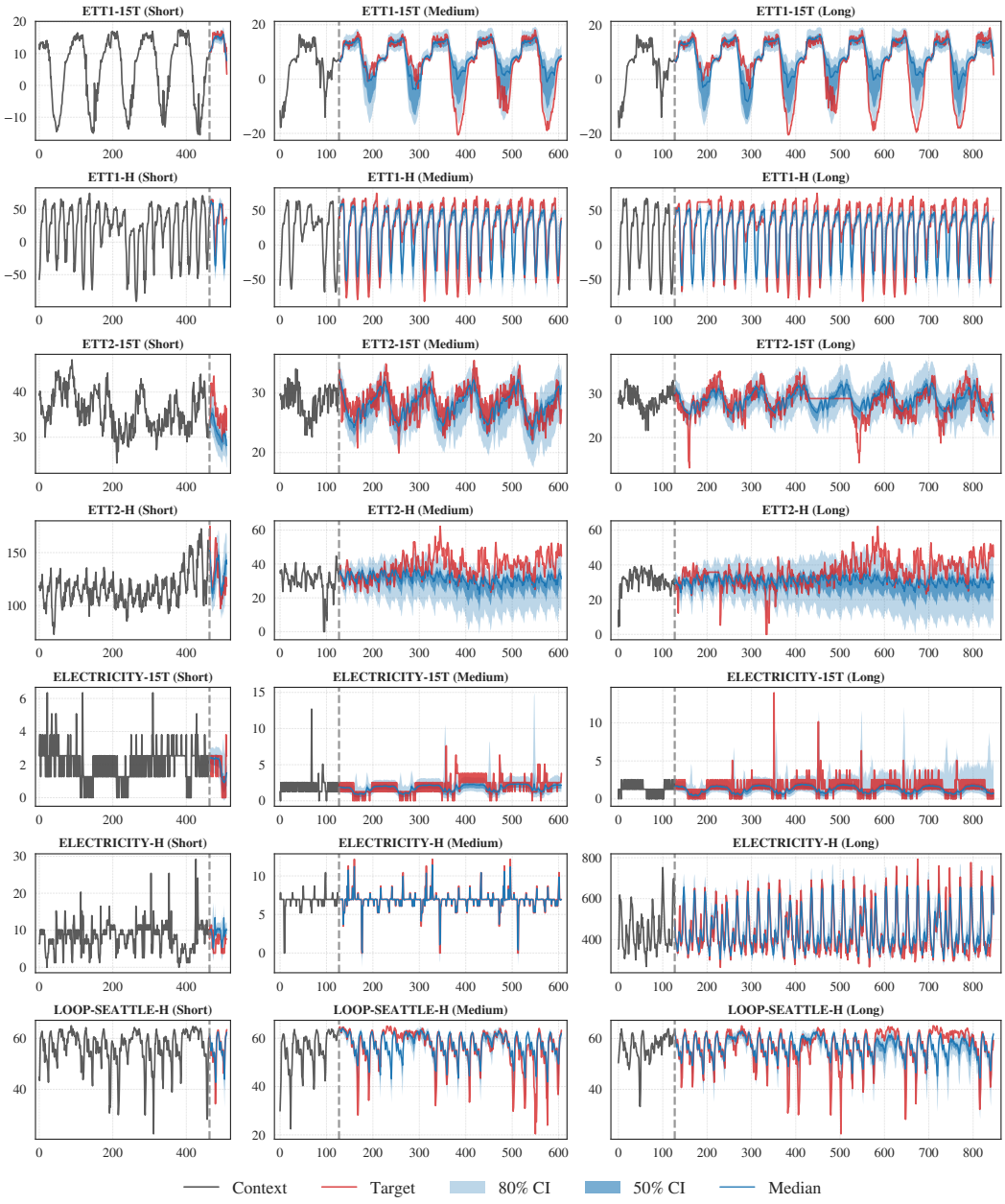

Figure 9: Examples of predictions from our **Zero-shot** model on datasets with significant seasonal patterns.

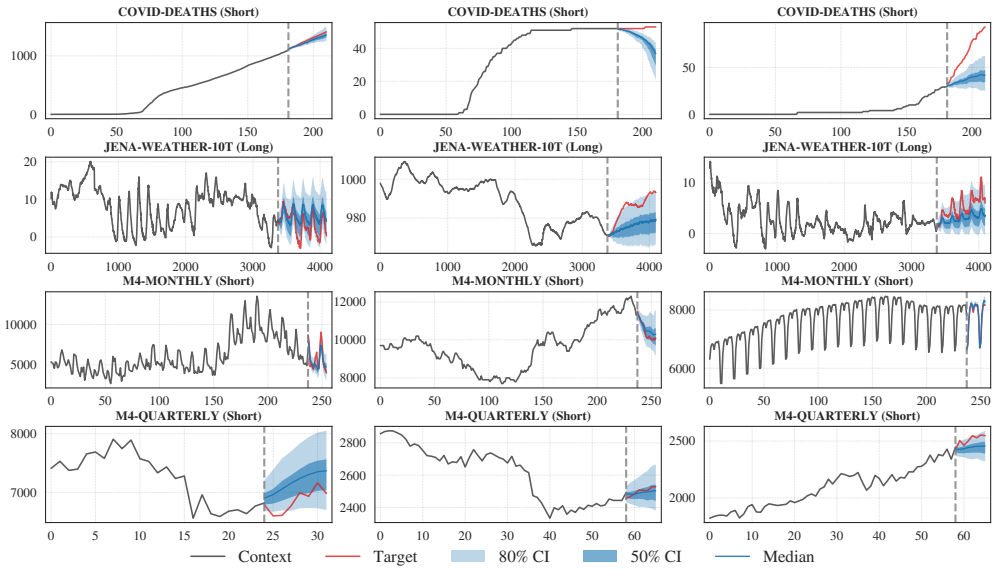

Figure 10: Examples of predictions from our **Extended** model on datasets with significant trend patterns.

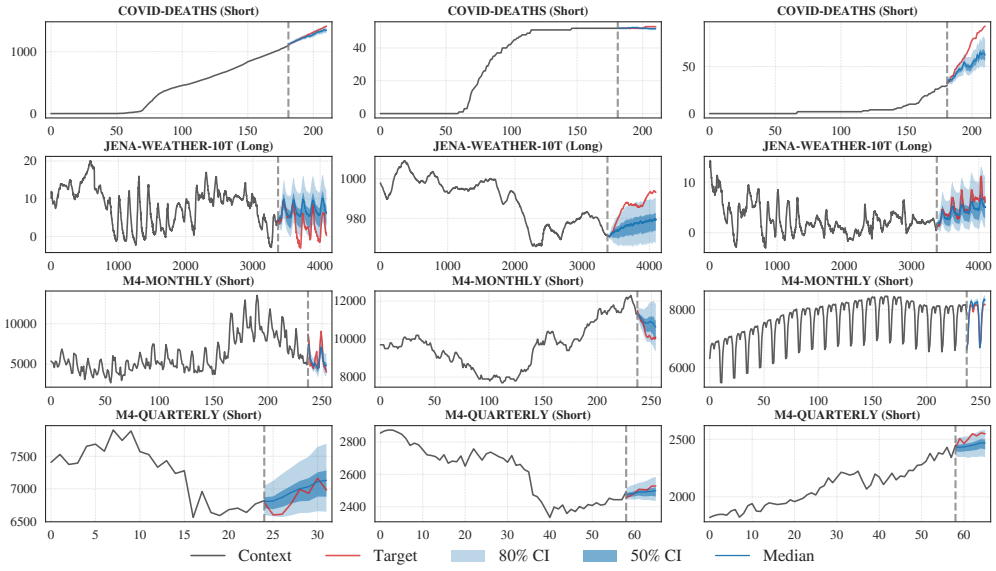

Figure 11: Examples of predictions from our **Zero-shot** model on datasets with significant trend patterns.

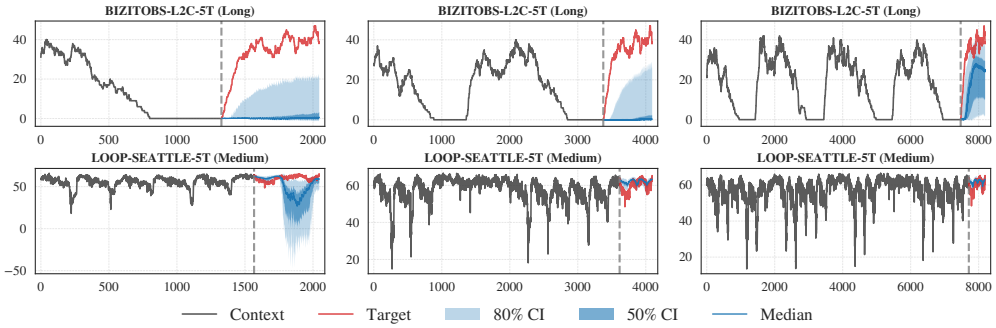

Figure 12: Example of the predictions from models with context length (left) 2048, (middle) 4096, and (right) 8192. Shorter context lengths may fail to capture periodicity and produce poor forecasts, while longer context lengths are more robust for general usage.

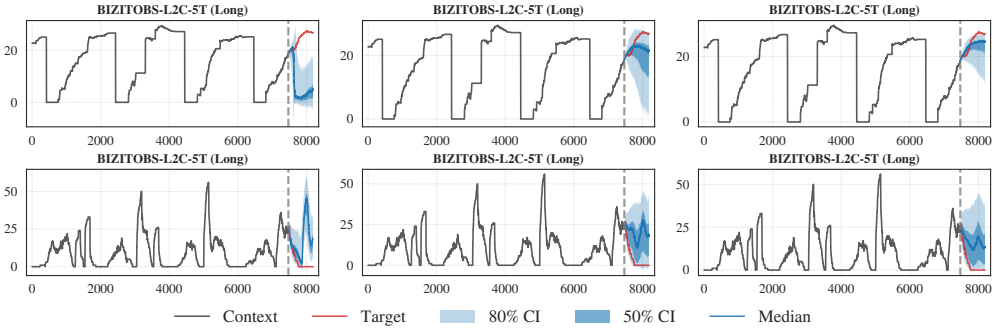

Figure 13: Examples of the predictions from models pretrained with CPM Block Size (left) $N_{\text{CPM}} = 1$, (middle) $N_{\text{CPM}} = 4$, and (right) $N_{\text{CPM}} = 8$. Larger N_{CPM} encourages the model to infer the underlying time series dynamics, providing better forecasts for general usage.

F.5 Inference Time

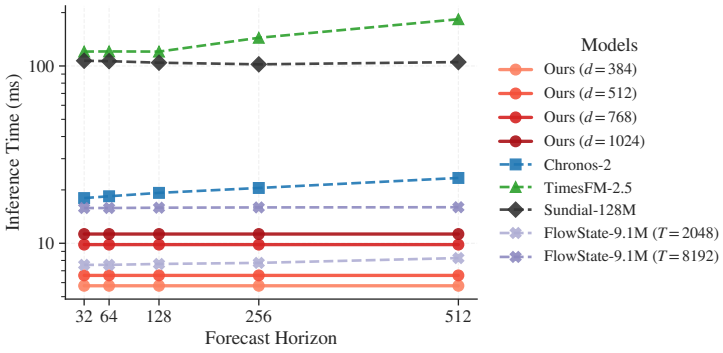

Figure 14: Inference time of our models and other TSFMs. Benefiting from high mask ratio, single-step inference, and architecture simplicity, our models also excel in inference time which is invariant to the forecast horizon.

Figure 14 benchmarks the inference latency of our proposed architecture against state-of-the-art TSFMs. All evaluations were conducted on an NVIDIA RTX 6000 Blackwell GPU using a batch size of 1. To ensure a rigorous comparison, we set the input context length to $T = 8192$ for all compatible models, while adjusting for architectural constraints in baselines such as Sundial ($T = 2880$). While FlowState is pretrained with $T = 2048$, it also supports longer context length as a state-space model. Therefore, we include the evaluation for both cases and compare their inference speed. Results for comparison models were derived using their official implementations for inference. We note that TiRex is excluded from this specific comparison due to hardware-specific compilation incompatibilities with its custom state-space model kernels.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state four major claims. All of them are supported and justified by the main body, including the statistical analysis in Appendix E that backs the claim of performance on par with the leading TSFMs. They reflect the paper's contribution discussed in the introduction and conclusion within the stated scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Scope of this Work paragraph in Section 1 explicitly bounds the study to univariate zero-shot forecasting and to ablations of the proposed architecture rather than retraining alternative TSFMs. Section 4.1 notes that the data-composition comparisons vary volume, composition, and domain exposure jointly; Section 4.5 and Appendix F.1.5 note that the scaling and backbone comparisons are matched in training steps rather than FLOPs and are single runs; Appendix E states that the reported confidence intervals capture variation across benchmark cases rather than training randomness; Appendix C.3 notes that the patch stride was fixed and not ablated, and Appendix F.1.3 notes that the patch-size ablation is a separate set of single pretraining runs.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical and contains no formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2 and Appendix C fully specify the architecture, normalization, masking, quantile head, and inference pipeline; Section 3.1 and Appendix C.6 describe the pretraining corpora, sampling procedure, and compute; Table 6 lists all hyperparameters with rationales. Pretrained checkpoints and the model implementation are publicly released (Section 1, Open Research contribution), and all data sources are public or generated with published procedures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the pretrained model checkpoints (`https://huggingface.co/ibm-research/patchtst-fm-r1`) and the model implementation (`https://github.com/ibm-granite/granite-tsfm/tree/patchtst-fm`), which reproduce all reported benchmark results through the official GIFT-Eval and TIME evaluation pipelines. The pretraining corpus is built from publicly available sources (GIFT-Eval-Pretrain) and from the published KernelSynth and TSMixup procedures, documented in Section 3.1; the pretraining pipeline is documented in Appendix C.6, while the training code and the generated synthetic series are not released.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.1 specifies model variants, pretraining data composition, and evaluation setup. Table 6 in Appendix C reports all training and architectural hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Following the standard GIFT-Eval and TIME benchmarking protocols, we report aggregate metrics (geometric mean MASE/CRPS, arithmetic mean Rank) over 97 and 98 test cases respectively. Appendix E additionally reports paired, stratified cluster-bootstrap 95% confidence intervals (10,000 replicates) for the ratios of aggregate MASE and CRPS between our models and the strongest baselines on both benchmarks, together with case-level standard deviations and rank quantiles. These intervals capture variation across benchmark cases; each full-scale variant is a single pretraining run, and this is stated explicitly.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.6 reports the GPU type and count, wall-clock time, and GPU-hours of the pretraining runs on V100 and H100 GPUs, and states that all ablation variants use the same training configuration, with the patch-size ablation run on 16 A100 GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms with the NeurIPS Code of Ethics. The work uses publicly available time-series datasets and contains no human-subjects data.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work studies foundation models for general time-series forecasting, a foundational research direction with no specific deployment context. The contribution is a transparent baseline and analysis methodology that promotes reproducibility, which we view as a positive influence on the field, with no direct path to negative societal applications beyond those generic to forecasting models.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released models are general-purpose univariate time-series forecasters trained on public datasets and pose no identified high risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, baselines, and frameworks used are cited to their original publications in Section 3.2 and Appendix D.2. Pretraining data sources (GIFT-Eval-Pretrain, TSMixup) and baseline models are publicly released by their original authors under their respective licenses, and the implementation builds on the open-source PyTorch and Hugging Face ecosystems.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the pretrained model checkpoints (`https://huggingface.co/ibm-research/patchtst-fm-r1`) and the model implementation (`https://github.com/ibm-granite/granite-tsfm/tree/patchtst-fm`), accompanied by usage instructions and license information (Section 1, Open Research contribution).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as a component of the research methodology. Any use was limited to incidental writing assistance and does not affect the core methods, scientific rigor, or originality of the work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.