TimeOperator: A Function-to-Function Approach to Time Series Modeling

Sheo Yon Jhin*
KAIST
Daejeon, South Korea
sheoyon.jhin@kaist.ac.kr

B. Aditya Prakash
Georgia Institute of Technology
Atlanta, GA,USA
badityap@cc.gatech.edu

Noseong Park
KAIST
Daejeon, South Korea
noseong@kaist.ac.kr

Abstract

Most time-series foundation models operate on fixed temporal grids, representing signals as vectors, tokens, or patches and learning maps between discretized observations. This grid-to-grid view is convenient for standard forecasting, but it is not the natural object for reusable time-series modeling, where temporal resolution, forecast horizon, and query locations may change across tasks and deployments. We argue that time-series foundation models should instead learn sample-conditioned function-to-function operators. Given finite observations of an underlying signal, such an operator maps the observed context to an output function that can be evaluated at user-specified times. We propose **TimeOperator**, a compact branch-trunk backbone that realizes this view with a shared context encoder, spectrally informed query features, context-adaptive trunk parameters, and multi-basis temporal responses. Forecasting and imputation are handled by evaluating the same operator on different query sets, while classification reuses the shared context representation. Across deterministic and probabilistic forecasting, imputation, and classification benchmarks, TimeOperator is competitive with existing time-series foundation models. Further analyses show that cross-frequency and cross-horizon prediction can be handled by changing only the query set, rather than changing the decoder or relying on post-hoc resampling.

1 Introduction

Time series are a fundamental modality across science and industry, including weather, healthcare, energy systems, finance, and industrial monitoring. Modern applications often require more than a single task: the same data may be used to forecast future trajectories, impute missing observations, or classify temporal patterns. This has motivated time-series foundation models and general-purpose backbones such as MOMENT (Goswami et al., 2024), UniTS (Gao et al., 2024), and FeDaL (Chen et al., 2025), which learn shared temporal representations or unified task specifications across downstream tasks.

Despite their architectural differences, many widely used time-series models share a discretized formulation. They represent signals as vectors, tokens, or patches on a chosen temporal grid, and learn maps between grid-indexed representations. This formulation can forecast well on the grid for which it is trained, but the output remains tied to a fixed decoder shape. When the output frequency changes, the horizon is extended, or values are requested at arbitrary time coordinates, the model typically requires resampling, interpolation, autoregressive rollout, token repetition, or task-specific decoding. We argue that this creates an object-level mismatch. Scaling data or architecture can improve accuracy on a chosen grid, but it does not by itself turn a grid-indexed vector map into a function-to-function operator that can be evaluated at user-specified physical times.

*Part of the work was done as a visiting researcher at Georgia Tech

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

Formally, for value-prediction tasks such as forecasting and imputation, the natural target is an operator over the underlying signal,

$$G^\star : C([0, t_{obs}]) \to C([0, T]), \tag{1}$$

which maps an observed continuous-time context to an output function that can be evaluated at arbitrary times. In practice, the input function is observed only through finite samples. TimeOperator therefore learns a queryable output function conditioned on those samples. In contrast, a discrete-grid model learns a finite-dimensional map such as $f_\theta : \mathbb{R}^L \to \mathbb{R}^H$ for a chosen input length L and prediction length H. Thus, the mismatch lies in the learned object: a grid-indexed vector map is not an intrinsic operator over continuous-time functions.

This motivates an operator-learning formulation for multi-task time-series modeling. We treat a time series as partial observations of an underlying continuous-time function u, and learn a conditional operator G_θ that maps the observed context to a queryable output function. The output is evaluated at a user-chosen set of query times $\mathcal{T}_q \subset [0, T]$, rather than decoded on a fixed training grid. Under this formulation, cross-frequency prediction corresponds to changing the spacing of $\mathcal{T}_q$, horizon extension corresponds to extending $\mathcal{T}_q$ beyond t_{obs}, and forecasting and imputation differ only in whether the queried times lie outside or within the observed context. Classification reuses the same context representation with a lightweight classification head.

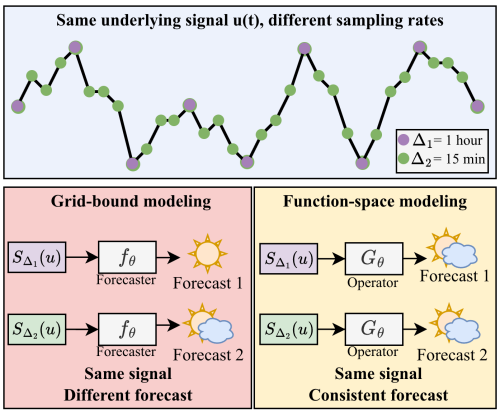

Figure 1: **Grid-bound prediction vs. function-space modeling.** The same signal $u(t)$ can be sampled at different rates, yielding different grid-indexed vectors. Conventional forecasters learn maps between sampled vectors, whereas **TimeOperator** treats them as observations of an underlying function and evaluates a conditional operator at user-specified query times.

We instantiate this idea as **TimeOperator**, a compact branch–trunk neural operator for time series inspired by DeepONet (Lu et al., 2021). A shared branch encoder summarizes the observed context, while a continuous trunk evaluates context-conditioned temporal basis responses at each query time. To support diverse temporal dynamics, TimeOperator uses context-adaptive trunk parameters, spectrally informed query features, and multiple basis families for periodic, trend-like, localized, and high-frequency components. This yields a queryable operator for value prediction and a shared representation for classification.

Our contributions are summarized as follows:

1. **Operator formulation.** We reformulate forecasting and imputation as query-time evaluation of a conditional operator over continuous-time signals.
2. **TimeOperator architecture.** We propose a compact branch–trunk neural operator with a shared context encoder, context-adaptive trunk parameters, multi-basis temporal functions, and spectrally informed query features.
3. **Evaluation.** We evaluate TimeOperator on forecasting, imputation, and classification, with a focus on cross-frequency and cross-horizon prediction by modifying only the query set $\mathcal{T}_q$.

2 Related Work

2.1 Time Series Foundation Models

Time Series Foundation Models (TSFMs) aim to replace dataset-specific models with reusable pretrained backbones that transfer across time-series datasets and tasks. Early forecasting-oriented models such as Chronos (Ansari et al., 2024), LPTM (Kamarthi & Prakash, 2024), Lag-Llama (Rasul et al., 2023), and TimesFM (Das et al., 2024) adopt large-scale sequence modeling interfaces for zero-shot forecasting, using tokenization, lagged inputs, or patch-based decoding. Moirai (Woo

et al., 2024) extends this direction toward universal forecasting by addressing frequency, variate, and distributional heterogeneity, while TiRex (Auer et al., 2025) revisits backbone design with xLSTM-based state tracking for short- and long-horizon prediction.

A related line broadens TSFMs beyond forecasting by learning shared representations or unified task specifications. MOMENT (Goswami et al., 2024) pretrains a patched Transformer encoder with masked time-series reconstruction and evaluates it across forecasting, imputation, classification, and anomaly detection. UniTS (Gao et al., 2024) represents time-series samples, prompts, and task specifications as tokens, using generative and classification tokens for different downstream tasks. FeDaL (Chen et al., 2025) studies general TSFM pretraining in federated environments, addressing dataset-level heterogeneity across forecasting, imputation, anomaly detection, and classification. Together, these works demonstrate the effectiveness of shared time-series backbones for transfer across datasets, domains, and task specifications.

2.2 Continuous-Time Time Series Modeling

Continuous-time modeling has long been studied as a way to handle irregular observations, non-uniform sampling, and latent dynamical structure in time series. Neural ODEs (Chen et al., 2018) parameterize the evolution of a hidden state through a differential equation, allowing the state to be evaluated continuously in time. Neural CDEs (Kidger et al., 2020) extend this idea by treating the observed sequence as a continuous control path that drives the hidden dynamics. EXIT (Jhin et al., 2022) extends neural CDEs through learned latent paths that support interpolation and extrapolation, while LEAP (Jhin et al., 2023) learns task-adaptive control paths instead of relying solely on a fixed interpolation scheme. In parallel, implicit neural representations, such as sinusoidal representation networks (Sitzmann et al., 2020), represent signals as coordinate-based functions that can be queried at arbitrary points.

Together, these works show that temporal data can be modeled beyond fixed observation grids, especially when observations are irregular, sparse, or naturally viewed as samples from an evolving continuous process.

2.3 Neural Operator Learning

Neural operator learning studies maps between functions, rather than maps between fixed-dimensional vectors. Given function spaces $\mathcal{U}$ and $\mathcal{V}$, the goal is to learn

$$G : \mathcal{U} \to \mathcal{V}, \tag{2}$$

so that an input function $u \in \mathcal{U}$ is mapped to an output function $G(u) \in \mathcal{V}$. This view separates the function being modeled from the particular grid on which it is observed or evaluated.

DeepONet (Lu et al., 2021) is a representative neural operator based on a branch–trunk factorization. The branch network receives samples of the input function and produces coefficients, while the trunk network receives a query coordinate and produces basis values at that coordinate. The operator output is written as

$$G(u)(y) = \sum_{k=1}^{p} b_k(u)\, \phi_k(y), \tag{3}$$

where $b_k(u)$ are branch coefficients and $\phi_k(y)$ are trunk basis values at query location y. This factorization makes the sample-to-query structure explicit. The input samples are encoded once, and the resulting operator can be evaluated at different output coordinates.

Fourier Neural Operators (Li et al., 2020) provide another influential formulation by learning global transformations over functions in the spectral domain. FNOs are especially effective when functions are represented on regular meshes, and they have been widely used for learning solution operators of partial differential equations. Together, DeepONet and FNO have made neural operators a standard framework for learning function-to-function maps in scientific machine learning.

For time series, the operator view is useful because the observation grid and the prediction grid need not coincide. It gives a natural formulation for models that condition on sampled temporal contexts and evaluate predictions at requested time coordinates.

3 Proposed Method

3.1 Problem Setup and Operator Form

We view a time series as finite observations of an underlying temporal signal. For clarity, we present the univariate case, where $u : \Omega \to \mathbb{R}$ is defined on $\Omega = [0, T]$. Given an observed context and a set of query times,

$$\mathcal{C} = \{(t_i, x_i)\}_{i=1}^{n_{\text{ctx}}}, \qquad x_i = u(t_i), \qquad t_i \in \mathcal{T}_{\text{ctx}} \subset [0, t_{obs}], \qquad \mathcal{T}_q = \{\tau_j\}_{j=1}^{n_q} \subset \Omega, \tag{4}$$

the goal is to predict the values $\{u(\tau_j)\}_{j=1}^{n_q}$ from $\mathcal{C}$. The signal u defines the function-level target, but the model is conditioned only on the sampled context $\mathcal{C}$. The query set $\mathcal{T}_q$ specifies where the output function is evaluated.

Different choices of $\mathcal{T}_q$ define different value-prediction tasks. Forecasting queries future times $\tau_j > t_{obs}$, imputation queries masked in-context times, and cross-frequency prediction queries a grid whose spacing differs from that of $\mathcal{T}_{\text{ctx}}$. Thus, output locations are specified by the query set rather than by a fixed decoder shape.

TimeOperator learns a sample-conditioned operator that maps the observed context to a queryable output function. Let $z = B_\theta(\mathcal{C})$ be the branch representation of the context. At a query time τ, the trunk produces K_{tr} vector-valued temporal responses $r_k(\tau, z) \in \mathbb{R}^{d_b}$. The final prediction is obtained by a context-dependent branch-trunk readout,

$$\hat{u}_\theta(\tau) = G_\theta(\mathcal{C})(\tau) = \sum_{k=1}^{K_{\text{tr}}} \beta_k(z)^\top r_k(\tau, z), \tag{5}$$

where $\beta_k(z) \in \mathbb{R}^{d_b}$ is the readout vector for the k-th trunk response. Each inner product $\beta_k(z)^\top r_k(\tau, z)$ is a scalar contribution to the prediction. The responses $r_k(\tau, z)$ are defined by the multi-basis hyper-trunk in Section 3.3.3. This operator form is used for forecasting and imputation. For classification, the trunk is bypassed and the same context representation z is passed to a lightweight classification head, as described in Section 3.4.

Proposition 1 (Approximation of sample-conditioned time-series operators)**.** *Let $s \geq 1$ be an integer denoting the smoothness order of the signals. Let $\mathcal{U} \subset C^s([0, t_{obs}])$ be a compact class of signals, and let $G^\star : \mathcal{U} \to C(\Omega_q)$ be a continuous target operator on a compact query domain $\Omega_q \subset [0, T]$. For each $u \in \mathcal{U}$, suppose the sampled context*

$$\mathcal{C}_n(u) = \{(t_i, u(t_i) + \eta_i)\}_{i=1}^n \tag{6}$$

has fill distance h_n over $[0, t_{obs}]$, noise level $\|\eta\|_\infty \leq \delta$, and admits a reconstruction $R_n\mathcal{C}_n(u)$ such that $G^\star(R_n\mathcal{C}_n(u))$ is well-defined and

$$\sup_{u \in \mathcal{U}} \|u - R_n\mathcal{C}_n(u)\|_\infty \leq \rho_n, \qquad \rho_n = ch_n^s + \delta. \tag{7}$$

Then, for any $\epsilon > 0$, there exists a TimeOperator G_θ with sufficiently expressive branch, trunk, and readout modules such that

$$\sup_{u \in \mathcal{U}} \|G^\star(u) - G_\theta(\mathcal{C}_n(u))\|_{L^\infty(\Omega_q)} \leq \omega_{G^\star}(\rho_n) + \epsilon, \tag{8}$$

where $\omega_{G^\star}$ is the modulus of continuity of $G^\star$. In the noiseless dense-context limit, $h_n \to 0$ and $\delta \to 0$, TimeOperator can uniformly approximate $G^\star$ on $\mathcal{U}$.

Proposition 1 formalizes the role of sampled contexts in the operator view. The term ρ_n captures the information loss from observing a continuous signal through a finite context, while ϵ captures the approximation error of the branch-trunk operator. The result is an expressivity statement, and the proof follows by applying DeepONet-style operator approximation to the reconstructed context. Details are given in Appendix A.

3.2 Overall Workflow

Figure 2 summarizes the TimeOperator workflow. The observed context is encoded once into a branch representation z, while FFT descriptors are combined with each query time $t \in \mathcal{T}_q$ to form an informed query. A hyper head conditions the trunk parameters and readout weights on z, and the multi-basis trunk evaluates query-time responses used by the operator readout.

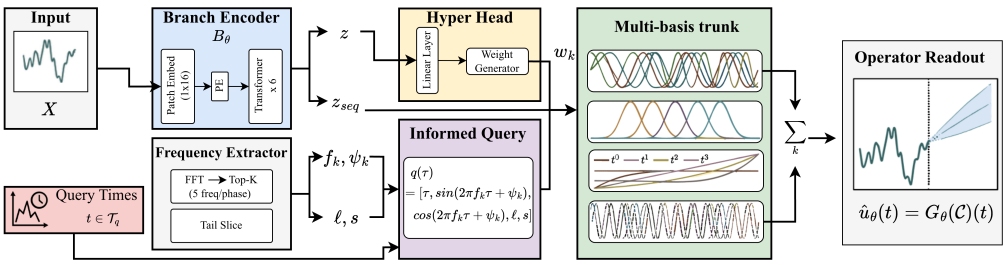

Figure 2: Overview of TimeOperator. The context X is encoded into z, while FFT descriptors and query times form informed queries. A hyper head generates context-dependent projection weights and readout vectors. The multi-basis trunk evaluates vector-valued responses $r_k(t, z)$, and the operator readout predicts $\hat{u}_\theta(t)$ using $\sum_k \beta_k(z)^\top r_k(t, z)$.

3.3 Time Operator

3.3.1 Context encoder and Hyper-network

Patch tokenization. The context $X = (x_1, \ldots, x_L) \in \mathbb{R}^L$ is divided into non-overlapping patches of length P, giving $n_p = L/P$ patches. Each patch is linearly embedded into a d-dimensional token and augmented with a sinusoidal positional encoding,

$$h_i^{(0)} \;=\; W_{\text{patch}}\, x_{(i-1)P+1\,:\,iP} \;+\; \text{PE}(i), \qquad i = 1, \ldots, n_p, \tag{9}$$

yielding a token sequence $H^{(0)} \in \mathbb{R}^{n_p \times d}$.

Transformer encoder. The patch tokens are processed by an N-layer Transformer encoder (N=6),

$$z_{\text{seq}} \;=\; \text{Transformer}(H^{(0)}) \;\in\; \mathbb{R}^{n_p \times d}, \qquad z \;=\; \frac{1}{n_p} \sum_{i=1}^{n_p} (z_{\text{seq}})_i \;\in\; \mathbb{R}^d. \tag{10}$$

The patch-level sequence z_{seq} retains positional information used later by per-query cross-attention, while the pooled summary z serves as a global descriptor of the context.

Hyper-network. Different time series can require different temporal bases even when they share the same trunk architecture. A fixed trunk basis would make all contexts use the same basis functions and adapt only through the final coefficients. We instead use a lightweight hyper head to generate context-dependent projection parameters and readout vectors from the pooled summary z,

$$\{W_k(z), \beta_k(z)\}_{k=1}^{K_{\text{tr}}} = H_\theta(z), \tag{11}$$

where $W_k(z)$ shapes the k-th basis response and $\beta_k(z)$ controls its contribution to the operator readout. A learned scale α_k stabilizes the amplitude of each branch. This makes the operator context-adaptive at inference time while keeping all model parameters shared across series.

3.3.2 Spectral lifting of query times

A trunk that receives only the raw query coordinate τ must learn temporal alignment from the context through downstream layers. We instead lift each query time using frequency and phase descriptors extracted from the observed context. This gives the trunk oscillatory coordinates aligned with the current series, while keeping the output query coordinate continuous. All query times are represented in normalized coordinates. For a physical query time t and a context ending at t_0 with time scale s_{time}, we use $\tau = 1 + (t - t_0)/s_{\text{time}}$, so that the context endpoint corresponds to $\tau = 1$. FFT frequencies are measured in cycles per normalized time unit, and phases are defined with respect to the same coordinate origin. We omit the dependence of q on the fixed context descriptors for notational simplicity.

Spectral and tail descriptors. We extract two complementary summaries from the observed context. The first is the set of top-K Fourier components, selected by magnitude and represented by their normalized frequencies and phases,

$$\{(f_k, \psi_k)\}_{k=1}^{K} = \mathrm{TopK}_{\mathrm{mag}}(\mathrm{FFT}(X)), \qquad K = 5, \tag{12}$$

where f_k is measured in cycles per normalized time unit and ψ_k is the corresponding phase. The second is a pair of tail statistics,

$$\ell = x_L, \qquad s = x_L - x_{L-1}. \tag{13}$$

The spectral descriptors summarize global periodic structure, while the tail statistics encode the local level and slope near the end of the observed context.

Informed query. For every query time $\tau \in \mathcal{T}_q$, these descriptors are combined into a single query feature,

$$q(\tau) = \big[\tau,\, \{\sin(2\pi f_k \tau + \psi_k),\, \cos(2\pi f_k \tau + \psi_k)\}_{k=1}^{K},\, \ell,\, s\big]. \tag{14}$$

Given a fixed context, $q(\tau)$ is defined for any normalized query coordinate. Changing the output rate, horizon, or imputation locations therefore changes only the queried set $\mathcal{T}_q$, not the decoder shape or the trunk architecture.

3.3.3 Multi-basis Trunk and Operator Readout

The trunk combines the context path and the query path. From the context path, it receives the global representation z, the patch sequence z_{seq}, and the hyper-head outputs $\{W_k(z), \beta_k(z)\}_{k=1}^{K_{\mathrm{tr}}}$. From the query path, it receives the informed query feature $q(\tau)$. At each query time, the trunk builds several temporal responses and combines them into one scalar prediction. We use four basis families to cover periodic, local, trend-like, and high-frequency behavior.

Basis branches. We use $K_{\mathrm{tr}} = 4$ branches. Each branch starts from a deterministic lift of the normalized query coordinate τ:

$$\begin{aligned}
\phi^{F}_{\mathrm{base}}(\tau) &= [\,\tau, \sin(2\pi f \tau), \cos(2\pi f \tau)\,]_{f=1}^{32}, \\
\phi^{R}_{\mathrm{base}}(\tau) &= [\,\tau, \exp(-20(\tau - c_n)^2)\,]_{n=1}^{20}, \\
\phi^{P}_{\mathrm{base}}(\tau) &= [\,1, \tau, \tau^2, \ldots, \tau^6\,], \\
\phi^{H}_{\mathrm{base}}(\tau) &= [\,\tau, \sin(2\pi f \tau), \cos(2\pi f \tau)\,]_{f=1}^{256}.
\end{aligned} \tag{15}$$

The four branches correspond to Fourier, RBF, polynomial, and high-Fourier bases. The high-Fourier branch provides a fixed high-frequency coordinate channel for rapidly varying components.

Per-query context attention. For branch k and query time τ, we first combine the basis lift and the informed query,

$$e_k(\tau) = [\,\phi^{k}_{\mathrm{base}}(\tau), q(\tau)\,]. \tag{16}$$

This query feature attends to the patch sequence to retrieve a location-specific context summary,

$$a_k(\tau) = \mathrm{CrossAttn}_k\big(P_k e_k(\tau), z_{\mathrm{seq}}, z_{\mathrm{seq}}\big), \qquad u_k(\tau) = [\,e_k(\tau), a_k(\tau)\,], \tag{17}$$

where P_k forms the attention query and z_{seq} provides keys and values. Thus, different query times can use different parts of the observed context while sharing the same global summary z.

Hyper-trunk evaluation. The hyper-head projection $W_k(z)$ maps the query feature $u_k(\tau)$ to a vector-valued temporal response,

$$r_k(\tau, z) = \mathrm{GELU}\big(\alpha_k W_k(z) u_k(\tau)\big) \in \mathbb{R}^{d_b}, \tag{18}$$

where α_k is a learned scalar that stabilizes the branch amplitude. Through $W_k(z)$, the same trunk architecture produces context-dependent responses for different input series.

Operator readout. The final prediction is the sum of scalar contributions from all branches:

$$\hat{u}_\theta(\tau) = G_\theta(\mathcal{C})(\tau) = \sum_{k=1}^{K_{\mathrm{tr}}} \beta_k(z)^\top r_k(\tau, z), \qquad \beta_k(z) \in \mathbb{R}^{d_b}. \tag{19}$$

Each term $\beta_k(z)^\top r_k(\tau, z)$ is a scalar, so the readout dimension is explicit. Changing the task changes only the queried coordinates $\mathcal{T}_q$, while the same trunk and readout are reused for forecasting, imputation, and cross-frequency prediction.

3.4 Training Objectives

TimeOperator is pretrained for value prediction using forecasting and imputation query sets. Both tasks use the same operator loss and differ only in the choice of $\mathcal{T}_q$. Classification is evaluated separately by applying a lightweight linear head to the branch representation.

Operator loss. For value-prediction tasks, we sample a context $\mathcal{C}$ and a query set $\mathcal{T}_q$. Forecasting places query times after the context endpoint, while imputation places them at masked observed positions. The operator is trained with a point-wise reconstruction loss,

$$\mathcal{L}_{\mathrm{op}} = \frac{1}{|\mathcal{T}_q|} \sum_{\tau \in \mathcal{T}_q} \left\| G_\theta(\mathcal{C})(\tau) - u(\tau) \right\|_2^2 . \tag{20}$$

For multivariate series, the same loss is applied channel-wise and averaged.

Classification head. For classification evaluation, the trunk is bypassed. The pooled branch representation $z = B_\theta(X)$ is fed to a lightweight head h_ψ and optimized with cross-entropy,

$$\mathcal{L}_{\mathrm{cls}} = \mathrm{CE}\big(h_\psi(z), y\big). \tag{21}$$

Only the classification head is trained for this evaluation, while the operator backbone remains fixed.

4 Experiments

We evaluate TimeOperator on forecasting, imputation, and classification. The first two tasks are performed by querying the learned operator at target times, while classification reuses the shared branch representation.

4.1 Forecasting

4.1.1 Standard forecasting benchmarks

We evaluate standard forecasting benchmarks to assess the forecasting accuracy of TimeOperator. On zero-shot ETT and Weather forecasting, TimeOperator obtains best or second-best MSE on all five datasets over horizons $H \in \{96, 192, 336, 720\}$ and observation lengths $\{512, 1024, 2048, 3072\}$. Full deterministic results and baseline configurations are provided in Appendix C.1.

Table 1 reports zero-shot normalized CRPS on 15 held-out GIFT-Eval configurations spanning transport, nature, energy, web/cloud, and sales domains. The evaluation uses the frozen TimeOperator backbone with the lightweight quantile head described in Appendix B.4. With a 40M-parameter backbone and a reported 1B data scale, TimeOperator achieves the lowest mean normalized CRPS and the most first-place results, winning on five configurations and ranking second on average. These results demonstrate strong probabilistic forecasting performance across diverse domains.

4.1.2 Cross-frequency forecasting

A reusable time-series foundation model should not be tied to a single temporal grid. We evaluate zero-shot cross-frequency forecasting, where the context is observed at the dataset's native resolution

Table 1: Zero-shot normalized CRPS on 15 held-out GIFT-Eval configurations. Scores are normalized by the seasonal naive baseline. Lower is better. Best and second-best results are bold and underlined, respectively.

Configuration	Domain	Ours	TempoPFN	TiRex	Moirai-L	Sundial	YingLong	FlowState r1.1	Chronos-B
sz_taxi/15T/long	Transport	0.473	0.485	**0.463**	0.498	0.517	0.477	<u>0.469</u>	0.581
kdd_cup_2018/D/short	Nature	0.554	0.566	0.557	0.565	0.587	**0.551**	0.567	<u>0.552</u>
ett2/H/medium	Energy/ETT	<u>0.552</u>	0.586	0.575	0.634	0.610	**0.548**	0.569	0.616
bizitobs_l2c/H/long	Web/Cloud	0.306	0.312	<u>0.285</u>	0.638	0.345	0.401	**0.2634**	0.295
loop_seattle/5T/short	Transport	0.675	0.688	<u>0.593</u>	**0.512**	0.614	0.710	0.598	0.675
temperature_rain/D/short	Nature	**0.918**	1.328	1.281	<u>1.122</u>	1.452	1.333	1.298	1.429
saugeen/W/short	Nature	0.737	0.723	0.664	0.816	0.769	**0.642**	<u>0.656</u>	0.898
electricity/D/short	Energy	**0.638**	0.865	<u>0.766</u>	0.979	0.908	0.780	0.783	0.865
hierarchical_sales/W/short	Sales	0.672	<u>0.666</u>	0.672	0.695	0.755	0.712	**0.650**	0.710
car_parts/M/short	Sales	**0.857**	0.985	0.952	1.129	1.137	1.111	<u>0.897</u>	1.024
m_dense/H/medium	Transport	0.515	0.673	<u>0.482</u>	**0.465**	0.532	0.669	0.484	0.565
bitbrains_fast_storage/5T/long	Web/Cloud	0.899	**0.859**	<u>0.864</u>	0.921	1.043	0.925	0.953	0.914
solar/10T/medium	Energy	1.532	1.841	<u>1.518</u>	3.396	1.696	2.078	**1.508**	3.119
ett1/D/short	Energy	**0.707**	<u>0.731</u>	0.734	0.757	0.988	0.739	0.734	1.025
restaurant/D/short	Sales	**0.609**	0.631	<u>0.624</u>	0.660	0.700	0.660	0.637	0.682
Mean CRPS ↓	–	**0.710**	0.796	<u>0.735</u>	0.919	0.844	0.822	0.738	0.930
Average rank ↓	–	<u>2.73</u>	4.50	**2.60**	5.50	6.67	4.90	3.17	5.93
Win count ↑	–	**5**	1	1	2	0	<u>3</u>	<u>3</u>	0
Data scale	–	1B	∼400B	231B	231B	∼1T	∼78B	231B	84B
Parameters	–	40M	40M	35M	91M	128M	50M	18.5M	200M

Table 2: Cross-frequency forecasting MSE (z-score-normalized, median across test windows). Native denotes the dataset's original sampling rate; coarse denotes one coarser output rate. Lower is better.

	Native	Coarse	Ours		Chronos		Moirai		Fourier		DeepONet	
Dataset	rate	rate	native	coarse	native	coarse	native	coarse	native	coarse	native	coarse
Weather	10min	hourly	**0.334**	**0.318**	0.342	0.333	0.957	0.889	1.048	1.047	0.799	0.797
Sz_taxi	15min	hourly	**0.863**	1.407	0.881	1.413	1.276	1.466	1.034	1.423	0.902	**1.320**
Loop_Seattle	5min	hourly	**0.275**	**0.120**	0.288	0.123	0.304	0.143	0.782	0.823	0.299	0.138
BizITObs-L2C	5min	hourly	4.771	**4.754**	**4.265**	4.912	6.322	6.192	5.824	5.482	5.214	4.982
ETTh1	1H	daily	**0.453**	0.313	0.629	0.505	0.842	0.556	2.510	**0.243**	1.465	0.330
ETTh2	1H	daily	**0.439**	**0.201**	0.478	0.219	0.565	0.377	0.651	0.237	0.713	0.234

and the output is requested either on the native grid or on one coarser grid. For TimeOperator, changing the output resolution only changes the query set $\mathcal{T}_q$. Chronos and Moirai produce native-rate forecasts that are aggregated to the coarser grid, while Fourier and DeepONet are queryable baselines evaluated directly at the requested times. A controlled comparison with a fixed-grid variant further supports the benefit of queryable outputs on coarse grids (Section 5.3).

Table 2 shows that TimeOperator obtains the lowest error in 9 out of 12 native/coarse settings and 4 out of 6 coarse-rate settings. Against fixed-grid TSFMs, TimeOperator is strongest in every coarse-rate setting. The Fourier and DeepONet comparisons further show that the gains are not only from accepting query coordinates, since TimeOperator remains strongest across most datasets. Figure 3 shows the same behavior on bursty service traffic. TimeOperator follows the coarse-grid trajectory directly, while Chronos and Moirai smooth or distort spikes after aggregating native-rate forecasts.

We further evaluate TimeOperator against TiRex and FlowState-r1.1 across five output sampling intervals. TimeOperator achieves the lowest MSE in 20 of 25 settings, including 18 of 20 non-native settings. These results support its cross-frequency forecasting capability against strong recent baselines. Full results are provided in Appendix C.3.

4.2 Imputation

We evaluate zero-shot random-mask imputation by treating masked positions as query times. Given a binary mask m_i, the observed context and query set are

$$\mathcal{C} = \{(t_i, x_i) : m_i = 0\}, \qquad \mathcal{T}_q = \{t_i : m_i = 1\}. \tag{22}$$

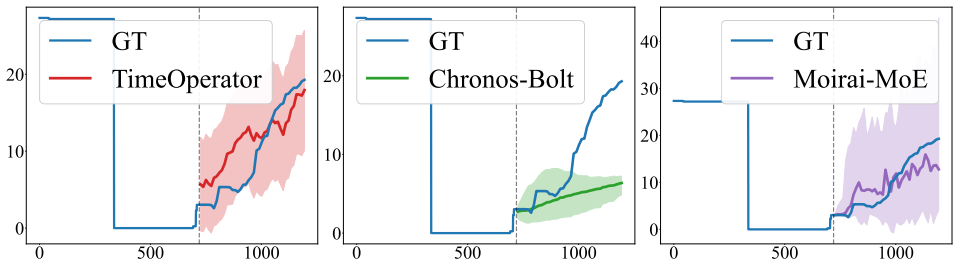

Figure 3: Cross-frequency forecasting on `bizitobs_l2c`. Panels show 5min→1-hour forecasts from TimeOperator, Chronos-Bolt, and Moirai-MoE. TimeOperator queries the coarse grid directly, while Chronos-Bolt and Moirai-MoE aggregate native-rate forecasts.

The same pretrained operator used for forecasting then evaluates $G_\theta(\mathcal{C})(\tau)$ at the masked locations, without an imputation-specific decoder, prompt, or fine-tuning.

Table 3: Zero-shot random-mask imputation averaged over five datasets and four mask ratios.

Model	Avg. MSE	Avg. MAE	1st Count
TimeOperator	**0.0297**	0.0923	**5**
FeDaL	0.0298	**0.0909**	**5**
FFTS	0.0334	0.1042	0
GPT4TS	0.0393	0.1121	0
DLinear	0.0414	0.1170	0

Table 3 shows that TimeOperator achieves the lowest average MSE among the compared models and ties with FeDaL for the highest first-place count across the ten dataset–metric pairs after averaging over mask ratios. FeDaL achieves the lowest average MAE. TimeOperator ranks first or second in all ten pairs. Full results are provided in Table 13. These results demonstrate effective imputation through query-time evaluation of the pretrained operator.

4.3 Classification

We evaluate classification as an auxiliary test of representation transfer using the frozen TimeOperator branch encoder. A linear classifier is trained on the concatenated channel-wise pooled features, with the trunk bypassed. Full UEA results and a comparison of branch-only and branch–trunk representations are provided in Appendix C.5.

5 Analysis and Ablations

We analyze how TimeOperator extends the forecast horizon through query times, how its architectural components affect accuracy, and how its queryable decoder compares with a fixed-grid prediction head.

5.1 Cross-Horizon Inference

The forecast horizon is specified by the extent of the query set $\mathcal{T}_q$. We evaluate ETTh1 with a fixed 512-step context and normalize latency and MSE by each model's $H{=}96$ value.

Table 4: Cross-horizon ratios on ETTh1.

Quantity	Ratio
TimeOperator latency	$\approx 1.0\times$
Chronos latency	$6.0\times$
TimeOperator MSE	$1.8\times$
Chronos/Moirai MSE	$2.3\times$

Table 4 shows that TimeOperator extends the horizon with much smaller latency growth than fixed-grid decoding. Its error also grows more mildly at long horizons. This supports the view that horizon extension is handled by adding query times to $\mathcal{T}_q$, rather than by changing the prediction head or rolling out an autoregressive decoder.

Additional profiling with a fixed 720-step context shows that increasing the query count from 96 to 1440 raises inference latency by only 1.3% in the measured setting. Context encoding and hypernetwork outputs are reused across queries, while query-dependent computation scales linearly with the query count. Detailed query-scaling and hypernetwork cost measurements are provided in Appendix C.6.

Table 5: Zero-shot MSE averaged over 5 datasets.

Variant	H=96	H=192	H=336	H=720	Avg.
Full TimeOperator	**0.293**	**0.343**	**0.380**	**0.434**	**0.362**
w/o hyper-trunk H_θ	0.323	0.372	0.405	0.456	0.389
w/o spectral query $q(\tau)$	0.339	0.393	0.425	0.479	0.409
Fourier-only trunk	0.311	0.360	0.393	0.443	0.377
+ wavelet basis branch	0.301	0.348	0.382	0.439	0.367

Table 6: Forecasting MSE of TimeOperator-Grid and TimeOperator on native and $6\times$ coarser output grids. Lower is better. Best results for each grid are bold.

	Native grid		$6\times$ coarser grid	
Configuration	Grid	Ours	Grid	Ours
ETT2/D/long	0.092	**0.085**	0.134	**0.071**
ETT2/H/medium	**1.522**	1.569	1.837	**1.381**
LoopSeattle/5T/short	**0.264**	0.275	0.288	**0.086**
BizITObs-L2C/H	0.302	**0.283**	0.348	**0.268**

5.2 Architectural Ablations

We ablate the main components of TimeOperator while keeping the branch encoder, training data, and optimizer fixed. Table 5 reports zero-shot forecasting MSE averaged over five ETT/Weather datasets.

The largest degradation comes from removing the spectrally informed query, showing that FFT-derived frequency and phase features are central to query-time prediction. Removing the hyper-trunk also hurts performance, indicating that context-adaptive basis parameters are useful beyond fixed basis functions. The Fourier-only trunk remains competitive but is consistently worse than the full multi-basis design, while adding a wavelet branch does not improve performance.

5.3 Comparison with a Fixed-Grid Decoder

To isolate the contribution of the queryable output structure, we compare TimeOperator with TimeOperator-Grid, which uses the same branch encoder, pretraining data, training objective, and optimization settings but replaces the queryable trunk with a fixed-grid prediction head. For the $6\times$ coarser output grid, TimeOperator-Grid aggregates its native-grid predictions, while TimeOperator directly evaluates the requested coordinates. As shown in Table 6, TimeOperator maintains competitive native-grid accuracy and achieves lower MSE in all four coarse-grid settings, with reductions ranging from 23% to 70%.

6 Conclusion

We presented TimeOperator, an operator-style backbone that maps sampled time-series contexts to a queryable output function. By mapping an observed context to a queryable output function, TimeOperator replaces fixed-grid decoding with evaluation at user-specified query times. This unifies forecasting and imputation through the choice of $\mathcal{T}_q$, while classification reuses the shared branch representation. Across forecasting, imputation, and classification benchmarks, TimeOperator remains competitive with existing time-series foundation models. Its main advantage appears in operator-specific settings: changing output frequency or forecast horizon is handled by changing only $\mathcal{T}_q$, rather than by interpolation, averaging, or autoregressive rollout. These results support the view that learning function-to-function operators is a promising direction for time-series foundation models beyond fixed temporal grids.

Acknowledgments

This work was supported by the following grants funded by the Ministry of Science and ICT (MSIT), Republic of Korea: an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (No. 2022-0-00857, Development of Financial and Economic Digital Twin Platform based on AI and Data, 33%), an Electronics and Telecommunications Research Institute (ETRI) grant (No. 2026-26ZB1200, Development of AI and Big Data-Based Policy Intelligence Technology, 33%), an IITP grant (No. RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain, 1%), and an IITP grant (No. RS-2026-25526850, 33%). This work was also partly supported by the U.S. NSF (Expeditions CCF-1918770, CAREER IIS-2028586, Medium IIS-1955883, Medium IIS-2403240, Medium IIS-2106961), NIH (1R01HL184139), Meta, and Dolby faculty gifts.

References

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation, 2024. URL https://arxiv.org/abs/2410.10393.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. *arXiv preprint arXiv:2505.23719*, 2025.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018, 2018. URL https://arxiv.org/abs/1811.00075.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Shengchao Chen, Guodong Long, and Jing Jiang. Fedal: Federated dataset learning for time series foundation models. *arXiv preprint arXiv:2508.04045*, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first international conference on machine learning*, 2024.

Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: A unified multi-task time series model. *Advances in Neural Information Processing Systems*, 37:140589–140631, 2024.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Sheo Yon Jhin, Jaehoon Lee, Minju Jo, Seungji Kook, Jinsung Jeon, Jihyeon Hyeong, Jayoung Kim, and Noseong Park. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 3102–3112, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512030. URL https://doi.org/10.1145/3485447.3512030.

Sheo Yon Jhin, Minju Jo, Seungji Kook, and Noseong Park. Learnable path in neural controlled differential equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7): 8014–8022, Jun. 2023. doi: 10.1609/aaai.v37i7.25969. URL https://ojs.aaai.org/index.php/AAAI/article/view/25969.

Harshavardhan Kamarthi and B Aditya Prakash. Large pre-trained time series models for cross-domain time series analysis tasks. *NeurIPS*, 2024.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

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

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 the operator formulation, architecture, and evaluation scope, and the claims are supported by Sections 4–5.

   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 limitations paragraph discusses finite sampled inputs, FFT descriptors computed from discrete observations, and deployment risks under distribution shift or noisy contexts.

   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: [Yes]

Justification: Proposition 1 states the assumptions, and Appendix A provides the full proof and a band-limited Fourier special case.

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: Sections 3–4 and Appendix B–C describe the architecture, objectives, data filtering, leakage control, training settings, and evaluation protocols.

   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: An anonymized supplementary release provides code, configuration files, and data preparation instructions for reproducing the reported experiments.

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: Appendix B reports the pretraining corpus, filtering rules, model and optimizer hyperparameters, quantile-head training, imputation protocol, and classification linear-probe protocol.

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: The cross-frequency evaluation is computed over 20 test windows; Table 2 reports median MSE, and the supplementary material reports variability over the same windows.

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 B reports the GPU type, memory, batch size, precision, training time, and compute used for backbone pretraining and task-specific heads.

   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 work uses public time-series datasets and standard benchmarks, preserves anonymous submission formatting, and does not involve human-subject experiments or restricted 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: [Yes]

Justification: The broader impact paragraph discusses potential uses in energy, weather, healthcare sensors, and industrial telemetry, as well as risks under distribution shift, sparse context, and noisy measurements.

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 work does not release high-risk assets such as personal data, scraped image data, or generative media models.

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: Appendix B.1 credits existing datasets and pretrained baselines, and reports the corresponding source and license or terms when available.

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: [N/A]

    Justification: The paper does not introduce a new dataset or benchmark asset; released code and configurations are for reproducing the proposed method.

    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: The paper does not involve crowdsourcing or human-subject experiments.

    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: The paper does not involve human-subject 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: No LLM is used as a core methodological component, model, baseline, data generator, or evaluation tool.

    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.

A Theoretical Justification

A.1 Proof of Proposition 1

We prove the proposition by separating the discretization error from the operator approximation error. Let

$$\rho_n = ch_n^s + \delta. \tag{23}$$

By the reconstruction assumption, for every $u \in \mathcal{U}$,

$$\|u - R_n\mathcal{C}_n(u)\|_\infty \le \rho_n. \tag{24}$$

Since $G^\star$ is continuous on the compact signal class, it has a modulus of continuity $\omega_{G^\star}$. Therefore,

$$\|G^\star(u) - G^\star(R_n\mathcal{C}_n(u))\|_{L^\infty(\Omega_q)} \le \omega_{G^\star}(\rho_n). \tag{25}$$

This term measures the information loss caused by observing the signal through a finite context.

It remains to approximate the reconstructed-context operator. Define

$$F_n(\mathcal{C}, \tau) = G^\star(R_n\mathcal{C})(\tau), \qquad \tau \in \Omega_q. \tag{26}$$

The admissible context set

$$\mathcal{K}_n = \{\mathcal{C}_n(u) : u \in \mathcal{U}\} \tag{27}$$

is compact, and F_n is continuous on $\mathcal{K}_n \times \Omega_q$. By the universal approximation theorem for continuous operators, equivalently the DeepONet branch–trunk approximation theorem, for any $\epsilon > 0$ there exist continuous coefficient functions $a_k(\mathcal{C})$ and trunk functions $\varphi_k(\tau)$ such that

$$\sup_{\mathcal{C}\in\mathcal{K}_n,\, \tau\in\Omega_q} \left| F_n(\mathcal{C}, \tau) - \sum_{k=1}^{K} a_k(\mathcal{C})\varphi_k(\tau) \right| < \epsilon. \tag{28}$$

TimeOperator can realize this DeepONet form as a special case. The branch encoder and readout can approximate the coefficient functions $a_k(\mathcal{C})$. The hyper-network can output context-independent trunk parameters, and the weights connected to the spectral query features can be set to zero. Thus the trunk can approximate fixed functions $\varphi_k(\tau)$, yielding

$$G_\theta(\mathcal{C})(\tau) \approx \sum_{k=1}^{K} a_k(\mathcal{C})\varphi_k(\tau). \tag{29}$$

Combining the reconstruction error and the operator approximation error gives

$$\sup_{u\in\mathcal{U}} \|G^\star(u) - G_\theta(\mathcal{C}_n(u))\|_{L^\infty(\Omega_q)} \le \omega_{G^\star}(ch_n^s + \delta) + \epsilon. \tag{30}$$

In the noiseless dense-context limit, $h_n \to 0$ and $\delta \to 0$, so the discretization term vanishes. This proves the proposition. □

Proposition 1 is an expressivity result. It does not imply finite-sample generalization or optimization guarantees. The reconstruction term is necessary because finite samples cannot identify arbitrary infinite-dimensional signals without regularity or injectivity assumptions.

A.2 Band-limited Fourier Operators as a Special Case

We next give a concrete special case that connects the operator approximation view to the spectrally informed query used by TimeOperator.

Consider the periodic domain $[0, 2\pi]$ with normalized Fourier basis

$$\psi_k(t) = \frac{1}{\sqrt{2\pi}} e^{ikt}. \tag{31}$$

Assume the input lies in an n-dimensional band-limited Fourier subspace

$$V_n = \operatorname{span}\{\psi_{\kappa_1}, \ldots, \psi_{\kappa_n}\}, \qquad u(t) = \sum_{i=1}^{n} a_i \psi_{\kappa_i}(t), \tag{32}$$

and the output lies in an m-dimensional Fourier subspace

$$W_m = \mathrm{span}\{\psi_{\lambda_1}, \ldots, \psi_{\lambda_m}\}. \tag{33}$$

Let the target operator be a truncated Fourier operator

$$F_m u(y) = \sum_{j=1}^{m} d_j \psi_{\lambda_j}(y), \qquad d = Ma, \tag{34}$$

for some matrix $M \in \mathbb{C}^{m\times n}$.

Let the input be observed at sensor locations $\{x_\ell\}_{\ell=1}^{q}$, and define the sample vector

$$S(u) = \big(u(x_1), \ldots, u(x_q)\big)^\top. \tag{35}$$

The samples satisfy

$$S(u) = Ea, \qquad E_{\ell i} = \psi_{\kappa_i}(x_\ell). \tag{36}$$

Assume that E has full column rank. This is the no-aliasing condition for the chosen sensors and frequencies. Then the Fourier coefficients are recovered by

$$a = E^\dagger S(u). \tag{37}$$

Thus

$$d = ME^\dagger S(u). \tag{38}$$

A DeepONet with $p = m$ can represent F_m exactly by choosing the branch and trunk as

$$B(S(u)) = ME^\dagger S(u), \qquad T_j(y) = \psi_{\lambda_j}(y). \tag{39}$$

Then

$$N(u)(y) = \sum_{j=1}^{m} B_j(S(u))T_j(y) = \sum_{j=1}^{m} (Ma)_j \psi_{\lambda_j}(y) = F_m u(y). \tag{40}$$

Therefore,

$$\sup_{u\in V_n} \|F_m u - N(u)\|_{L^2} = 0. \tag{41}$$

Since TimeOperator contains the DeepONet branch–trunk form as a special case, TimeOperator can also exactly represent this finite band-limited Fourier operator under the same no-aliasing condition.

Approximate branch and trunk. If the branch coefficients and trunk basis are approximated rather than represented exactly, define

$$\delta_B = \sup_{\|u\|_{L^2}\le R} \|\widehat{d}(u) - Ma\|_2, \qquad \delta_T = \left(\sum_{j=1}^{m} \|\widehat{T}_j - \psi_{\lambda_j}\|_{L^2}^2\right)^{1/2}. \tag{42}$$

Then

$$\sup_{\|u\|_{L^2}\le R} \|F_m u - N(u)\|_{L^2} \le \delta_B + (\|M\|_2 R + \delta_B)\,\delta_T. \tag{43}$$

In particular, when the branch implements the linear coefficient map exactly and the trunk uses the exact Fourier basis, $\delta_B = \delta_T = 0$ and the approximation error is zero.

Role of the spectral query. This special case explains why spectrally informed queries are useful for oscillatory time series. When dominant frequency and phase information can be estimated from the context, the trunk does not need to infer oscillatory coordinates from raw time alone. The prediction problem is reduced to estimating coefficients over appropriate temporal bases. The Fourier special case is not a general guarantee for arbitrary time series. It assumes a finite band-limited input class and an injective sensor map. If the sensor matrix E is not full column rank, two different inputs can produce the same samples, and no branch–trunk model that only observes those samples can distinguish them. This is why Proposition 1 explicitly separates context reconstruction error from operator approximation error.

B Implementation and Training Details

Reproducibility statement. We provide architecture details, training objectives, hyperparameters, dataset filtering procedures, and evaluation protocols in Appendix B and Appendix C. Full per-dataset results are reported in the appendix, and code/configuration files are provided in the supplementary material for reproducibility. `https://github.com/sheoyon-jhin/NeurIPS2026-TimeOperator`

B.1 Existing Assets and Licenses

Table 7 summarizes the external datasets, pretrained models, and code assets used in this work. We use all assets for research purposes only, credit the original creators, and follow the license or terms stated by the corresponding source. For assets whose dataset license is not explicitly stated on the source page, we cite the original archive and do not redistribute modified copies.

Table 7: Existing assets used in this work.

Asset	Use	Source	License / Terms	Citation
LOTSA	Pretraining source	Hugging Face Hub: `Salesforce/lotsa_data`	Apache-2.0; research-use release notice on the dataset card	(Woo et al., 2024)
GIFT-Eval	Probabilistic forecasting evaluation	Hugging Face Hub: `Salesforce/GiftEval`; official GIFT-Eval repository	Apache-2.0; research-use release notice on the dataset card	(Aksu et al., 2024)
ETT	Deterministic forecasting evaluation	Official ETDataset repository	CC BY-ND 4.0, as stated in the ETDataset license file	(Zhou et al., 2021)
Weather / Jena Climate	Deterministic forecasting evaluation	Jena Climate data recorded at the Max Planck Institute for Biogeochemistry; commonly distributed through LTSF benchmark copies	Dataset license not explicitly specified on the benchmark source page; original data source is credited	(Wu et al., 2021)
UEA MTSC Archive	Classification evaluation	Time Series Classification website / UEA multivariate archive	Dataset license not explicitly specified on the archive page; archive citation and source terms are followed	(Bagnall et al., 2018)
Chronos-Bolt	Fixed-grid forecasting baseline	Hugging Face Hub: `amazon/chronos-bolt-base`; Chronos GitHub repository	Apache-2.0	(Ansari et al., 2024)
Moirai / Moirai-MoE	Fixed-grid forecasting baseline	Hugging Face Hub: `Salesforce/moirai-*`; Uni2TS repository	Model weights: CC BY-NC 4.0; Uni2TS code: Apache-2.0	(Woo et al., 2024)

The generator is regime-targeted rather than dataset-targeted. Each synthetic sample is drawn by first selecting a temporal-regime template and then sampling its parameters from broad priors. Table 8 summarizes the regime axes used in the synthetic component. The axes cover trend, periodicity, basis-specific morphology, and nonstationary or event-driven behavior. No template is fitted to any held-out evaluation dataset. Concrete frequencies, amplitudes, phases, noise variances, segment lengths, and trend slopes are independently randomized per sample. The synthetic generator contributes 1M trajectories per training run, each of length up to 720 steps.

B.2 Pretraining Data and Leakage Control

TimeOperator is pretrained on a mixture of (i) a curated subset of the public LOTSA corpus and (ii) a synthetic time-series generator that covers a broad range of temporal regimes. Both sources are combined at each optimization step to balance the realism of measured data with the diversity and coverage of programmatically generated trajectories. To support a clean zero-shot evaluation, the LOTSA subset is filtered before any window is sampled for training.

B.2.1 Real-world subset of LOTSA

Real-world series are drawn from LOTSA, a publicly released collection of time series spanning energy, transport, finance, climate, healthcare, retail, and industrial monitoring (231B observations in total). We explicitly remove from the LOTSA corpus **any dataset that overlaps with the GIFT-Eval test split or with the standard ETT/Weather benchmarks**, before any window is sampled for training. Concretely, the excluded set is:

- **GIFT-Eval overlap (12 datasets):** `LOOP_SEATTLE`, `M_DENSE`, `SZ_TAXI`, `car_parts_with_missing`, `covid_deaths`, `hierarchical_sales`,

Table 8: Synthetic pretraining regimes with randomized parameters. Generators are not fitted to held-out datasets.

Axis	Approx. weight	Regime classes
Trend	0.12	Linear, piecewise-linear, random walk, exponential drift
Periodicity	0.39	Sub-hourly, diurnal, weekly, monthly, annual, business-cycle
Basis morphology	0.08	Fourier, polynomial, RBF bumps, mixed bases
Nonstationarity and events	0.40	Regime shifts, bursts, multiscale signals, heteroskedastic noise, chirps, pulses

`hospital`, `kdd_cup_2018_with_missing`, `restaurant`, `saugeenday`, `temperature_rain_with_missing`, `us_births`;

- **ETT and weather overlap:** all variants of the ETT and Weather benchmarks, together with `oikolab_weather`;
- **M-competition overlap:** all M1, M3, and M4 datasets in their LOTSA-mirrored form (6 datasets total).

After this filtering step, every dataset used to train TimeOperator is disjoint from the datasets used in our zero-shot evaluations. The exclusion is enforced at the corpus level: an assertion at dataset-loading time refuses to start training if any excluded dataset remains in the cache, so future re-runs cannot accidentally reintroduce overlapping series.

After exclusion, the LOTSA subset that we use yields approximately **1B observations** across 16M sliding windows of length 720 (in the Data Scale unit reported by the FeDaL convention; see Tab. 10). This is roughly 0.4% of full LOTSA (231B observations), reflecting both the conservative exclusion list and a windows-per-series cap that we use to keep training cost bounded.

B.2.2 Synthetic temporal regimes

Removing the GIFT-Eval/ETT/Weather/M-competition datasets thins out several temporal regimes that are otherwise abundant in LOTSA – notably daily and sub-hourly cycles, short series with strong seasonality, regime-switching dynamics, and heavy-tailed event series. To restore broad coverage without re-introducing any held-out dataset, we additionally sample from a synthetic generator that produces univariate trajectories with controllable temporal structure.

The generator is regime-targeted rather than dataset-targeted: each synthetic sample is drawn by first selecting a temporal-regime template and then sampling its parameters from broad priors. The templates are organized along four orthogonal axes:

- **Axis 1 – Trend.** Linear, piecewise-linear, random walk, and exponential drift.
- **Axis 2 – Periodicity.** Fundamental periods spanning sub-hourly to annual, with one or several harmonics, optional amplitude modulation, and additive seasonal residuals.
- **Axis 3 – Basis-targeted morphology.** Realizations produced from a single trunk family (Fourier, polynomial, or RBF bumps) so that each branch of the multi-basis trunk receives positive supervision on inputs that cleanly excite it.
- **Axis 4 – Non-stationary level shifts.** Non-stationary level shifts, time-varying variance (heteroskedastic), regime-switching, and multi-scale superpositions.

Domains within each axis are sampled with fixed mixture weights (detailed in Tab. 8). Importantly, no template is constrained to reproduce any specific dataset. Concrete frequencies, amplitudes, phases, noise variances, segment lengths, and trend slopes are independently randomized per sample, so two draws from the same template differ in all these dimensions. The synthetic generator contributes 1 M trajectories per training run, each of length up to 720 steps.

B.2.3 Mixing strategy

The two sources address complementary needs.

- **Real-world LOTSA series** provide the measurement characteristics that synthetic generators cannot easily reproduce: irregular noise structure, exogenous shocks, slowly-varying biases, and the long-tailed distribution of anomalies that occurs in deployed sensors and economic indicators.
- **Synthetic series** provide coverage: they fill temporal regimes that are sparse in the filtered LOTSA pool (notably daily-to-weekly cycles, short series with strong seasonality, and basis-pure morphologies that supervise specific trunk branches), and they let us match the volume of training data needed for a foundation-scale backbone without re-introducing any held-out dataset.

At each optimization step we sample mini-batches from the union of LOTSA windows and synthetic trajectories with batch composition set so that the two sources contribute roughly equal gradient mass over an epoch. This mixed-data regime preserves the realism of foundation-scale pretraining while keeping the test datasets strictly unseen.

B.2.4 Training corpus statistics

After applying the exclusion list, the real-world subset contains approximately **1B observations** across 1,000+ datasets, spanning sub-second to monthly sampling frequencies. The synthetic generator contributes a configurable additional 1 M trajectories per training run, each of length up to 720 steps. We use a sliding window of 192–720 steps with random forecast horizons; full hyper-parameters are listed in Appendix B.

B.3 Spectral Query Normalization

The trunk receives a query feature $q(\tau)$ that combines deterministic basis lifts with an informed feature extracted from the observed context by FFT. This appendix specifies the normalization used for query coordinates and spectral descriptors, so that the same trunk can be evaluated across heterogeneous context lengths and sampling rates.

Query coordinate normalization. Let L be the effective context length after rounding down to the nearest multiple of the patch size $P = 16$. For a raw query time t, let t_0 be the last observed context time and let s_{time} be the duration of the observed context. We first form the context-relative coordinate

$$\bar{\tau} = \frac{t - t_0}{s_{\text{time}}}. \tag{44}$$

In implementation, we use the shifted coordinate $\tau = 1 + \bar{\tau}$, so that the last context point corresponds to $\tau = 1$. For a future query k native steps after the context endpoint, this gives

$$\tau = 1 + \frac{k}{L}, \qquad k = 1, \ldots, H. \tag{45}$$

Thus, horizons are represented as fractions of the context length. Changing the output rate or horizon changes only the queried coordinates $\mathcal{T}_q$.

Informed feature z_{iq}. For each normalized query coordinate τ, the informed query feature is

$$z_{\text{iq}} = \big[\tau,\, \sin(2\pi f_1 \tau + \psi_1),\, \cos(2\pi f_1 \tau + \psi_1),\, \ldots,\, \sin(2\pi f_K \tau + \psi_K),\, \cos(2\pi f_K \tau + \psi_K),\, \ell,\, s\big], \tag{46}$$

where $\{(f_k, \psi_k)\}_{k=1}^{K}$ are the top-K Fourier components selected by magnitude from the context FFT. We use $K = 5$ by default. The scalar features ℓ and s are the last context value and local slope, matching the main text.

Frequency and magnitude normalization. The FFT frequencies are converted to cycles per normalized context unit. A component with period p native steps is therefore represented as $f = L/p$. For example, the same physical periodicity receives the same normalized frequency when the context covers the same physical duration, even if the sampling interval changes. This prevents the same harmonic pattern from appearing different solely because the series is sampled every 5 minutes rather than every hour.

The tail features ℓ and s are computed after the same per-window z-score normalization used by the encoder. This keeps the informed query bounded across datasets with different amplitudes and units.

This construction supports the cross-frequency evaluation in Table 2. Since both τ and z_{iq} are defined in a length-normalized coordinate system, evaluating the trunk at $\tau = 1.5$ means querying half a context length beyond the observed window, independent of whether one native step corresponds to 5 minutes or 1 hour.

B.4 Probabilistic Forecasting Head and CRPS

For probabilistic forecasting evaluations (Sec. 4.1.1, Tab. 1), we attach a lightweight quantile head on top of the (frozen) backbone. This appendix specifies its architecture, training objective, and the exact CRPS formula used in our reported numbers.

Architecture. The head is a per-trunk two-layer MLP followed by a sorting step:

$$\texttt{Linear}(d_{\text{model}} \to 128) \;\to\; \texttt{GELU} \;\to\; \texttt{Linear}(128 \to N_q) \;\to\; \texttt{cumsum-softplus},$$

where N_q=9 is the number of quantile levels. The cumulative-softplus post-processing enforces monotonic, non-crossing quantiles. Total trainable parameters: 347,146 (the backbone with ∼40M parameters is frozen). The supported quantile levels are $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, which match the GIFT-Eval probabilistic protocol.

Training objective. The head is trained with the standard pinball loss

$$\mathcal{L}_{\text{pinball}}(y, \hat{q}_\tau) \;=\; \max\big(\, \tau\,(y - \hat{q}_\tau),\; (\tau - 1)\,(y - \hat{q}_\tau)\,\big), \tag{47}$$

averaged over the nine quantile levels and over forecast horizons. We run 8 epochs of head-only fine-tuning on the same synthetic+LOTSA mixture used for backbone pretraining (Sec. B.2); detailed hyper-parameters are listed in Appendix B.6.

Point forecasts from quantile output. For deterministic-forecasting evaluations (Tab. 10, Tab. 2), we use the **median quantile** (τ=0.5, the fifth output of the head) as the point prediction. This avoids any model-specific calibration step.

CRPS computation. Following the GIFT-Eval reporting convention, we compute the (Gneiting–Raftery) continuous ranked probability score from the nine quantile estimates:

$$\mathrm{CRPS}(y, \hat{q}) \;=\; \frac{2}{N_q} \sum_{i=1}^{N_q} \mathcal{L}_{\text{pinball}}\big(y, \hat{q}_{\tau_i}\big). \tag{48}$$

We report the dataset-level *normalized* CRPS, defined as the ratio of the model's CRPS to that of a seasonal-naive baseline computed on the same context window with period equal to `get_seasonality(freq)`. By this construction the seasonal-naive baseline scores 1.000 on every dataset, and TimeOperator together with all baseline foundation models in Tab. 1 are directly comparable on the same scale.

B.5 Multivariate Handling and Classification Head

The TimeOperator backbone is intrinsically univariate: each forward pass ingests a single channel of length L and produces a continuous functional forecast. This appendix describes how we handle multivariate forecasting benchmarks (ETT, Weather, ECL, etc.) and how we build a classification head on top of the same backbone for the UEA benchmarks.

Channel-independent forecasting. For multivariate forecasting evaluations, we apply the channel-independent protocol popularized by PatchTST: each channel is forecast independently using the same backbone, with per-channel z-score normalization computed from the corresponding channel's context window. Concretely, given an input of shape (B, L, C) and a forecast horizon H,

1. For each $b \in [B]$ and channel $c \in [C]$, compute $\mu_{b,c}, \sigma_{b,c}$ from $x_{b,:,c}$ and form the normalized context $\tilde{x}_{b,:,c} = (x_{b,:,c} - \mu_{b,c})/\sigma_{b,c}$.

2. Run the (univariate) backbone on each $\tilde{x}_{b,:,c}$ to obtain a H-step forecast $\tilde{y}_{b,:,c}$ on the normalized scale.
3. De-normalize: $\hat{y}_{b,:,c} = \tilde{y}_{b,:,c} \cdot \sigma_{b,c} + \mu_{b,c}$.

This is the same protocol we use to evaluate baselines (Chronos, Moirai) on multivariate datasets, ensuring a like-for-like comparison.

Classification head. On the UEA multivariate classification benchmark, we follow the standard linear-probe protocol used by FeDaL/TimesBERT. The pretrained backbone is kept frozen and a single linear classifier is trained on top of a pooled context representation:

1. Each channel of the input is processed independently by the backbone encoder, producing a $(n_{\text{patches}}, d_{\text{model}})$ patch-level representation per channel.
2. Mean-pool across patches to obtain a d_{model}=512 feature per channel; concatenate channel features into a single $C \cdot d_{\text{model}}$-dimensional embedding per series.
3. A single linear layer $\texttt{Linear}(C \cdot d_{\text{model}} \to n_{\text{classes}})$ is trained per dataset with cross-entropy loss for 100 epochs; all backbone parameters are frozen throughout.

This is the same recipe used by FeDaL. It tests whether the operator-pretrained representation is linearly separable for class identity, without giving the backbone any opportunity to specialize to any individual UEA dataset. Full hyper-parameters are listed in Appendix B.8.

This appendix details the training protocol for each task. All experiments share the same pretrained backbone unless noted otherwise; task-specific heads are trained on top of the frozen (or lightly fine-tuned) backbone. Pretraining excludes 12 GIFT-Eval test datasets and 6 M-competition datasets from the LOTSA corpus to ensure a strict zero-shot evaluation protocol.

B.6 Forecasting

Backbone pretraining. We pretrain TimeOperator from scratch on a mixture of LOTSA and synthetic time series. Context lengths are sampled from $L \in \{96, 192, 384, 512, 720\}$, with forecast horizons sampled uniformly from $[\max(4, L/16), 2L]$. We use patch size $P = 16$ and augment window sampling rates by factors in $\{0.5, 1.0, 2.0\}$. Synthetic data follow Table 8, with independently randomized generator parameters and no fitting to held-out datasets.

The 40M-parameter backbone is trained for 15 epochs using AdamW with learning rate 1.2×10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 10^{-2}, batch size 1024, and bfloat16 precision. Each epoch includes approximately 17M sequences from 16M LOTSA windows, 1M synthetic trajectories, and 460k short-series donor windows. Training takes approximately 8 hours on a single A100 (80GB).

Quantile head fine-tuning. For probabilistic forecasting, we train a two-layer MLP head per trunk output with hidden dimension 128 and approximately 347k total trainable parameters, keeping the backbone frozen. The head predicts nine quantiles $\{0.1, 0.2, \ldots, 0.9\}$ and is trained for 8 epochs on the same synthetic+LOTSA mixture using pinball loss, learning rate 3×10^{-4}, batch size 512, and bfloat16 precision. Training takes approximately 30 minutes on a single A100. Deterministic ETT/Weather evaluations use the median quantile as the point prediction.

B.7 Imputation

Setup. Imputation is performed in a fully zero-shot manner using the same pretrained backbone and the same operator readout as for forecasting; no imputation-specific decoder, prompt, fine-tuning, or per-dataset hyperparameter adjustment is applied. Following Section 3, imputation differs from forecasting only in the choice of query set $\mathcal{T}_q$: given a binary mask m_i, the observed context is $\mathcal{C} = \{(t_i, x_i) : m_i = 0\}$ and the query set is $\mathcal{T}_q = \{t_i : m_i = 1\}$. The model evaluates $G_\theta(\mathcal{C})(t)$ at each masked timestamp using the same trunk and operator readout used for forecasting.

Protocol. We follow the masking ratio protocol of TimesBERT/FeDaL: $r \in \{12.5\%, 25\%, 37.5\%, 50\%\}$ of randomly chosen time points are masked per series. Imputation is evaluated on ETTh1, ETTh2, ETTm1, ETTm2, and Weather; for each (dataset, r) pair we

report MSE and MAE averaged over the test split. Sequence length is 720 with batch size 16 at inference. The same backbone weights are used across all five benchmarks and all four mask ratios, with no per-(dataset, r) re-tuning.

B.8 Classification

Linear probe protocol. We adopt the standard linear-probe protocol on the 10 UEA multivariate classification datasets used by FeDaL/TimesBERT (`EthanolConcentration`, `FaceDetection`, `Handwriting`, `Heartbeat`, `JapaneseVowels`, `PEMS-SF`, `SelfRegulationSCP1`, `SelfRegulationSCP2`, `SpokenArabicDigits`, `UWaveGestureLibrary`). The same pre-trained backbone used for forecasting and imputation is kept frozen; we only train a single linear classifier on top of the pooled context representation per dataset. The trunk is not used for classification (Section 3); we directly read the pooled branch representation. Each channel of the input is processed independently by the backbone encoder; the per-channel patch features are mean-pooled to a 512-dimensional vector and the channel features are concatenated into a single $C \cdot 512$-dimensional embedding per series.

Training. For each UEA dataset, we train the linear head with AdamW (lr $= 10^{-3}$, weight decay 10^{-4}) for 100 epochs with cross-entropy loss, batch size 32, sequence length 720. We use the standard train/test split provided by the UEA archive. No data augmentation, no fine-tuning of the backbone, and no per-dataset hyperparameter tuning is performed. We report classification accuracy on the test set.

C Additional Experimental Results

C.1 Forecasting

C.1.1 Zero-shot forecasting

Table 9: Zero-shot forecasting MSE and MAE. Avg. is the arithmetic mean over four horizons. Lower is better. Bold and underlined values indicate the best and second-best distinct scores. First-place counts exclude Avg. rows and include ties. TimesFM is omitted on Weather due to pretraining overlap.

Dataset	H	**Operator Learning**		**Federated Learning Methods**						**Pretrained Time Series Foundation Models**													
		TimeOperator		FeDaL		FFTS		FedAvg		$\text{Moirai}_{\text{small}}$		$\text{Moirai}_{\text{base}}$		$\text{Moirai}_{\text{large}}$		TimesFM		MOMENT		$\text{Chronos}_{\text{small}}$		$\text{Chronos}_{\text{base}}$		$\text{Chronos}_{\text{large}}$	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.158**	0.212	0.159	0.212	0.172	0.218	0.201	**0.207**	0.198	0.222	0.220	0.217	0.199	0.211	–	–	0.243	0.255	0.211	0.243	0.203	0.238	0.194	0.235
	192	0.218	0.264	**0.217**	0.264	0.235	0.278	0.250	0.278	0.247	0.265	0.271	0.259	0.246	**0.251**	–	–	0.278	0.329	0.263	0.294	0.256	0.290	0.249	0.285
	336	0.284	0.302	0.285	0.312	0.290	0.321	0.304	0.340	0.283	0.303	0.286	0.297	**0.274**	**0.291**	–	–	0.306	0.346	0.321	0.339	0.314	0.336	0.302	0.327
	720	0.343	**0.331**	0.359	0.348	0.351	0.383	0.353	0.395	0.373	0.354	0.373	0.354	**0.337**	0.340	–	–	0.350	0.374	0.404	0.397	0.397	0.396	0.372	0.378
	Avg.	**0.2508**	0.2773	0.2550	0.2840	0.2620	0.3000	0.2770	0.3050	0.2753	0.2860	0.2875	0.2818	0.2640	**0.2733**	–	–	0.2943	0.3260	0.2998	0.3183	0.2925	0.3150	0.2793	0.3063
ETTh1	96	0.351	0.380	0.347	0.381	**0.344**	0.382	0.412	0.409	0.401	0.402	0.376	0.392	0.349	**0.379**	0.414	0.404	0.688	0.557	0.466	0.409	0.440	0.393	0.441	0.390
	192	0.395	**0.407**	0.398	0.410	0.395	0.410	0.420	0.421	**0.388**	0.412	0.412	0.413	0.434	0.415	0.465	0.434	0.688	0.560	0.530	0.450	0.492	0.426	0.502	0.424
	336	**0.413**	0.441	0.425	0.452	0.438	0.445	0.440	0.444	0.433	**0.428**	0.433	**0.428**	0.495	0.445	0.503	0.456	0.675	0.563	0.570	0.486	0.550	0.462	0.576	0.467
	720	0.445	0.458	0.457	0.469	0.445	0.457	0.480	0.522	**0.439**	0.454	0.447	**0.444**	0.611	0.510	0.511	0.481	0.683	0.585	0.615	0.543	0.882	0.591	0.835	0.583
	Avg.	**0.4010**	0.4215	0.4068	0.4280	0.4055	0.4235	0.4380	0.4490	0.4153	0.4240	0.4170	**0.4193**	0.4723	0.4373	0.4733	0.4438	0.6835	0.5663	0.5453	0.4720	0.5910	0.4680	0.5885	0.4660
ETTh2	96	**0.293**	0.352	0.307	0.355	0.325	0.332	0.340	0.350	0.297	0.336	0.294	**0.330**	0.296	**0.330**	0.315	0.349	0.342	0.396	0.307	0.356	0.308	0.343	0.320	0.345
	192	**0.345**	0.379	0.349	0.372	0.355	**0.359**	0.378	0.388	0.368	0.381	0.365	0.375	0.361	0.371	0.388	0.395	0.354	0.402	0.376	0.401	0.384	0.392	0.406	0.399
	336	0.382	0.397	0.387	0.395	0.391	0.393	0.415	0.421	0.370	0.393	0.376	**0.390**	0.390	**0.390**	0.422	0.427	**0.356**	0.407	0.408	0.431	0.429	0.430	0.492	0.453
	720	**0.392**	0.416	0.401	**0.406**	0.409	0.412	0.427	0.441	0.411	0.426	0.416	0.433	0.423	0.418	0.443	0.454	0.395	0.434	0.604	0.533	0.501	0.477	0.603	0.511
	Avg.	**0.3530**	0.3860	0.3610	0.3820	0.3700	**0.3740**	0.3900	0.4000	0.3615	0.3840	0.3628	0.3820	0.3675	0.3773	0.3920	0.4063	0.3618	0.4098	0.4238	0.4303	0.4055	0.4105	0.4553	0.4270
ETTm1	96	**0.287**	0.347	0.289	**0.346**	0.307	0.352	0.303	0.371	0.418	0.392	0.363	0.356	0.380	0.361	0.361	0.370	0.654	0.527	0.511	0.423	0.454	0.408	0.457	0.403
	192	0.318	**0.367**	**0.317**	0.369	0.324	0.392	0.344	0.384	0.431	0.405	0.388	0.375	0.412	0.383	0.414	0.405	0.662	0.532	0.618	0.485	0.567	0.477	0.530	0.450
	336	**0.367**	0.418	0.370	0.420	0.399	0.438	0.426	0.432	0.433	0.412	0.416	**0.392**	0.436	0.400	0.445	0.429	0.672	0.537	0.683	0.524	0.662	0.525	0.577	0.481
	720	0.461	0.422	0.464	0.426	**0.426**	0.498	0.438	0.453	0.462	0.432	0.460	**0.418**	0.462	0.420	0.512	0.471	0.692	0.551	0.748	0.566	0.900	0.591	0.660	0.526
	Avg.	**0.3583**	0.3885	0.3600	0.3903	0.3640	0.4200	0.3778	0.4100	0.4360	0.4103	0.4068	**0.3853**	0.4225	0.3910	0.4330	0.4188	0.6700	0.5368	0.6400	0.4995	0.6458	0.5003	0.5560	0.4650
ETTm2	96	0.206	0.284	0.207	0.283	0.222	0.278	0.219	0.289	0.214	0.288	0.205	0.273	0.211	0.274	0.202	**0.270**	0.260	0.335	0.209	0.291	0.199	0.274	**0.197**	0.271
	192	**0.239**	0.331	0.248	0.333	0.273	0.320	0.270	0.333	0.284	0.332	0.275	0.316	0.281	0.318	0.289	0.321	0.289	0.350	0.280	0.341	0.261	0.322	0.254	**0.314**
	336	0.315	0.338	0.316	0.340	0.320	**0.327**	0.321	0.340	0.331	0.362	0.329	0.350	0.341	0.355	0.360	0.366	0.324	0.369	0.354	0.390	0.326	0.366	**0.313**	0.353
	720	**0.390**	**0.401**	0.397	0.408	0.453	0.523	0.478	0.498	0.402	0.408	0.437	0.411	0.485	0.428	0.462	0.430	0.394	0.409	0.553	0.499	0.455	0.439	0.416	0.415
	Avg.	**0.2875**	0.3385	0.2920	0.3410	0.3170	0.3620	0.3220	0.3650	0.3078	0.3475	0.3115	**0.3375**	0.3295	0.3438	0.3283	0.3468	0.3168	0.3658	0.3490	0.3803	0.3103	0.3503	0.2950	0.3383
1st Count		**13**		4		4		1		3		6		7		1		1		0		0		3	

C.1.2 Full-shot forecasting

C.2 Comparison with FlowState Checkpoints

We compare TimeOperator with two official FlowState checkpoints, containing 9.1M and 18.5M parameters, on the same 15 GIFT-Eval configurations. As shown in Table 11, TimeOperator achieves lower normalized CRPS on 10 configurations against FlowState-9.1M and eight against FlowState-r1.1, with the lowest mean score across the evaluated configurations. We use FlowState-r1.1 as the primary comparator in Table 1.

Table 10: Full long-term forecasting results comparing our proposed TimeOperator with advanced deep time series models. **Bold**: the best, Underline: the second best.

Dataset	Method	**TimeOperator**		FeDaL		TimeMixer		Time-LLM		GPT4TS		DLinear		PatchTST		TimesNet	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.131**	**0.126**	0.139	0.199	0.162	0.209	0.147	0.201	0.162	0.212	0.176	0.237	0.149	0.198	0.172	0.220
	192	**0.168**	**0.225**	0.174	0.230	0.208	0.250	0.189	0.234	0.204	0.248	0.220	0.282	0.194	0.241	0.219	0.261
	336	**0.234**	**0.279**	0.240	**0.279**	0.252	0.287	0.262	**0.279**	0.254	0.286	0.265	0.319	0.245	0.282	0.280	0.306
	720	**0.284**	**0.302**	**0.300**	**0.310**	0.340	0.343	0.304	0.316	0.326	0.337	0.333	0.362	0.314	0.334	0.365	0.359
	Avg	**0.204**	**0.249**	**0.213**	**0.255**	0.241	0.272	0.225	0.257	0.237	0.270	0.248	0.300	0.225	0.264	0.259	0.287
ILI	24	1.310	0.726	**1.265**	**0.714**	1.979	0.860	1.285	0.727	2.063	0.881	2.215	1.081	1.319	0.754	2.317	0.934
	36	1.331	0.804	**1.329**	**0.800**	1.893	0.862	1.404	0.814	1.868	0.892	1.963	0.963	1.430	0.834	1.972	0.920
	48	1.410	0.771	**1.409**	**0.768**	2.129	0.936	1.523	0.807	1.790	0.884	2.130	1.024	1.553	0.815	2.238	0.940
	60	1.442	0.830	**1.418**	0.810	2.155	0.938	1.531	0.854	1.979	0.957	2.368	1.096	1.470	**0.788**	2.027	0.928
	Avg	1.373	0.783	**1.355**	**0.773**	2.039	0.899	1.435	0.801	1.925	0.903	2.169	1.041	1.443	0.797	2.139	0.931
ETTh1	96	**0.337**	**0.368**	0.340	0.371	0.373	0.401	0.362	0.392	0.376	0.397	0.375	0.399	0.370	0.399	0.384	0.402
	192	0.373	0.403	**0.371**	**0.400**	0.436	0.430	0.398	0.418	0.416	0.418	0.405	0.416	0.413	0.421	0.436	0.429
	336	**0.390**	**0.418**	0.392	0.420	0.484	0.458	0.430	0.427	0.442	0.433	0.439	0.443	0.422	0.436	0.491	0.469
	720	**0.402**	0.447	0.417	**0.445**	0.497	0.482	0.442	0.457	0.477	0.456	0.472	0.490	0.447	0.466	0.521	0.500
	Avg.	**0.375**	**0.409**	0.380	**0.409**	0.448	0.443	0.408	0.423	0.465	0.455	0.422	0.437	0.413	0.430	0.458	0.450
ETTh2	96	**0.261**	0.328	0.267	**0.325**	0.289	0.340	0.268	0.328	0.285	0.342	0.289	0.353	0.274	0.336	0.340	0.374
	192	**0.327**	**0.361**	0.333	0.369	0.370	0.389	0.329	0.375	0.354	0.389	0.383	0.418	0.339	0.379	0.402	0.414
	336	0.339	0.389	0.359	0.392	0.386	0.413	0.368	0.409	0.373	0.407	0.448	0.465	**0.329**	**0.380**	0.452	0.452
	720	**0.362**	**0.410**	0.377	0.420	0.412	0.432	0.372	0.420	0.406	0.441	0.605	0.551	0.379	0.422	0.462	0.468
	Avg	**0.322**	**0.372**	0.334	0.377	0.364	0.394	0.334	0.383	0.381	0.412	0.431	0.446	0.330	0.379	0.414	0.427
ETTm1	96	**0.256**	0.324	0.258	**0.321**	0.320	0.357	0.272	0.334	0.292	0.346	0.299	0.343	0.290	0.342	0.338	0.375
	192	0.299	0.347	**0.297**	**0.343**	0.361	0.380	0.310	0.358	0.332	0.372	0.335	0.365	0.332	0.369	0.374	0.387
	336	0.330	0.384	**0.327**	**0.381**	0.392	0.404	0.352	0.384	0.366	0.394	0.369	0.386	0.366	0.392	0.410	0.411
	720	0.394	0.413	0.392	0.415	0.452	0.440	**0.383**	**0.411**	0.417	0.421	0.425	0.421	0.416	0.420	0.478	0.450
	Avg	**0.319**	0.367	**0.319**	**0.365**	0.381	0.395	0.329	0.372	0.388	0.403	0.357	0.378	0.351	0.380	0.400	0.406
ETTm2	96	**0.156**	**0.238**	0.157	0.240	0.175	0.258	0.161	0.253	0.173	0.262	0.167	0.269	0.165	0.255	0.187	0.267
	192	**0.217**	**0.289**	0.220	0.292	0.235	0.298	0.219	0.293	0.229	0.301	0.224	0.303	0.220	0.292	0.249	0.309
	336	**0.262**	0.340	0.267	0.342	0.298	0.341	0.271	**0.329**	0.286	0.341	0.281	0.342	0.274	**0.329**	0.321	0.351
	720	0.378	0.397	0.399	0.400	0.391	0.395	**0.352**	**0.379**	0.378	0.401	0.397	0.421	0.362	0.385	0.408	0.403
	Avg	0.253	0.316	0.261	0.319	0.275	0.323	**0.251**	**0.313**	0.284	0.339	0.267	0.333	0.255	0.315	0.291	0.333

C.3 Extended Cross-Frequency Evaluation

We evaluate TimeOperator, TiRex, and FlowState-r1.1 under a zero-shot protocol across five forecasting settings. The output sampling interval is varied over $\{1, 3, 6, 12, 14\}$ times the native interval, with larger multiples corresponding to coarser output grids. For each setting, all models use the same observation windows, physical forecast spans, target timestamps, and normalization. TimeOperator changes only the query set to evaluate each output grid.

Table 12 shows that TimeOperator achieves the lowest MSE in 20 of 25 settings and outperforms FlowState-r1.1 in 21 settings. On non-native grids, TimeOperator ranks first in 18 of 20 settings, supporting its ability to adapt the output resolution through query-time evaluation.

C.4 Imputation

C.5 Classification

We evaluate representation transfer on 10 UEA datasets using a frozen TimeOperator backbone and a linear classifier. For each channel, patch-level branch features are mean-pooled and concatenated across channels before classification. Only the classifier is trained. Full training details are provided in Appendix B.8.

Table 14 reports all per-dataset results. TimeOperator achieves an average accuracy of 76.4%, compared with 76.1% for FeDaL. Performance varies across datasets, and the aggregate advantage is influenced by the strong result on EthanolConcentration.

Table 11: Zero-shot normalized CRPS comparison with two FlowState checkpoints on 15 GIFT-Eval configurations. Lower is better. Best results are bold.

Configuration	TimeOperator (40M)	FlowState (9.1M)	FlowState-r1.1 (18.5M)
sz_taxi/15T/long	0.473	0.484	**0.469**
kdd_cup_2018/D/short	**0.554**	0.565	0.567
ett2/H/medium	**0.552**	0.560	0.569
bizitobs_l2c/H/long	0.306	**0.2628**	0.2634
loop_seattle/5T/short	0.675	0.618	**0.598**
temperature_rain/D/short	**0.918**	1.287	1.298
saugeen/W/short	0.737	0.676	**0.656**
electricity/D/short	**0.638**	0.775	0.783
hierarchical_sales/W/short	0.672	0.657	**0.650**
car_parts/M/short	**0.857**	0.964	0.897
m_dense/H/medium	0.515	**0.476**	0.484
bitbrains_fast_storage/5T/long	**0.899**	1.062	0.953
solar/10T/medium	1.532	1.619	**1.508**
ett1/D/short	**0.707**	0.742	0.734
restaurant/D/short	**0.609**	0.627	0.637
Mean CRPS ↓	**0.7096**	0.7583	0.7378
TimeOperator wins vs. checkpoint	–	10/15	8/15

Table 12: Zero-shot cross-frequency forecasting MSE at five output sampling intervals relative to the native interval. Lower is better. Best results are bold.

Dataset and setting	Model	1×	3×	6×	12×	14×
ETT2/D, long	TimeOperator	**0.085**	**0.080**	**0.071**	**0.047**	**0.033**
	TiRex	0.147	0.158	0.148	0.112	0.082
	FlowState-r1.1	0.157	0.152	0.137	0.094	0.067
ETT2/H, medium ($H = 192$)	TimeOperator	**1.569**	**1.481**	**1.381**	**1.232**	**1.182**
	TiRex	1.927	1.904	1.871	1.840	1.944
	FlowState-r1.1	2.079	2.022	1.931	1.810	1.641
Loop Seattle/5T, short	TimeOperator	0.275	**0.128**	**0.086**	**0.050**	**0.042**
	TiRex	**0.264**	0.172	0.163	0.149	0.108
	FlowState-r1.1	0.277	0.164	0.125	0.104	0.043
Loop Seattle/5T, medium ($H = 480$)	TimeOperator	0.863	**0.772**	**0.719**	**0.672**	0.622
	TiRex	**0.841**	0.832	0.817	0.804	0.729
	FlowState-r1.1	0.849	0.806	0.760	0.681	**0.592**
BizITObs-L2C/H	TimeOperator	0.283	0.279	**0.268**	**0.247**	**0.153**
	TiRex	0.281	0.278	0.282	0.262	0.193
	FlowState-r1.1	**0.276**	**0.274**	0.271	0.258	0.182

Table 15 compares branch-only and branch–trunk representations under frozen-backbone linear probing. The branch-only representation achieves higher average accuracy, supporting its use for classification. We therefore interpret classification as auxiliary representation transfer, while forecasting and imputation directly evaluate the queryable operator.

C.6 Computational Cost

Scaling with query count. We profile the full model on an NVIDIA A6000 using FP32, batch size one, and a fixed context length of 720. Table 16 reports inference cost as the number of query coordinates Q increases from 96 to 1440. This $15\times$ increase raises median latency by 1.3% and peak memory by 5.6%. The hypernetwork runs once per context, with its generated parameters reused across all query coordinates. For a fixed context length and model architecture, query-dependent

Table 13: Imputation MSE and MAE at four mask ratios. Avg denotes their arithmetic mean. Lower is better. Bold and underlined values indicate the best and second-best distinct scores.

Dataset	Mask ratio	**TimeOperator**		FeDaL.		FFTS		FedAvg		GPT4TS		DLinear		PatchTST		TimesNet		FED.		Auto.		Stationary		ETS.		LightTS		In.		Re.	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	12.5%	**0.013**	**0.031**	**0.013**	0.036	0.020	0.053	0.020	0.053	0.026	0.049	0.025	0.045	0.029	0.049	0.057	0.141	0.047	0.101	0.039	0.084	0.041	0.107	0.027	0.051	0.026	0.047	0.037	0.093	0.031	0.076
	25%	0.021	**0.024**	**0.018**	0.025	0.024	0.057	0.024	0.057	0.028	0.052	0.029	0.052	0.031	0.053	0.065	0.155	0.052	0.111	0.048	0.103	0.064	0.163	0.029	0.056	0.030	0.054	0.042	0.100	0.035	0.082
	37.5%	**0.022**	0.064	0.025	0.060	0.034	0.063	0.034	0.063	0.033	0.060	0.031	**0.057**	0.035	0.058	0.081	0.180	0.058	0.121	0.057	0.117	0.107	0.229	0.033	0.062	0.032	0.060	0.049	0.111	0.040	0.091
	50%	0.029	**0.050**	**0.027**	**0.050**	0.038	0.061	0.038	0.061	0.037	0.065	0.034	0.062	0.038	0.063	0.102	0.207	0.065	0.133	0.066	0.134	0.183	0.312	0.037	0.068	0.037	0.067	0.053	0.114	0.046	0.099
	Avg	0.0213	**0.0423**	**0.0208**	0.0428	0.0290	0.0585	0.0290	0.0585	0.0310	0.0565	0.0298	0.0540	0.0333	0.0558	0.0763	0.1708	0.0555	0.1165	0.0525	0.1095	0.0988	0.2028	0.0315	0.0593	0.0313	0.0570	0.0453	0.1045	0.0380	0.0870
ETTh1	12.5%	**0.012**	0.074	0.013	**0.073**	0.015	0.075	0.020	0.105	0.017	0.085	0.023	0.101	0.041	0.130	0.096	0.229	0.093	0.206	0.080	0.193	0.052	0.166	0.032	0.119	0.046	0.144	0.063	0.180	0.042	0.146
	25%	**0.013**	**0.070**	**0.013**	0.071	0.015	0.081	0.022	0.109	0.022	0.096	0.023	0.101	0.044	0.135	0.096	0.229	0.093	0.206	0.080	0.193	0.052	0.166	0.032	0.119	0.046	0.144	0.063	0.180	0.042	0.146
	37.5%	0.033	**0.089**	0.033	0.091	0.036	0.094	0.046	0.131	**0.029**	0.111	**0.029**	0.111	0.049	0.143	0.133	0.271	0.113	0.231	0.103	0.219	0.069	0.191	0.039	0.131	0.057	0.161	0.079	0.200	0.063	0.182
	50%	**0.030**	0.115	**0.030**	**0.110**	**0.030**	0.111	0.048	0.135	0.040	0.128	0.036	0.124	0.055	0.151	0.186	0.323	0.134	0.255	0.132	0.248	0.089	0.218	0.047	0.145	0.067	0.174	0.093	0.218	0.082	0.208
	Avg	**0.0220**	0.0870	0.0223	**0.0863**	0.0240	0.0903	0.0340	0.1200	0.0270	0.1050	0.0278	0.1093	0.0473	0.1398	0.1278	0.2630	0.1083	0.2245	0.0988	0.2133	0.0655	0.1853	0.0375	0.1285	0.0540	0.1558	0.0745	0.1945	0.0573	0.1705
ETTh2	12.5%	**0.011**	0.066	0.014	**0.062**	0.014	0.070	0.020	0.085	0.017	0.076	0.018	0.080	0.026	0.094	0.108	0.239	0.034	0.127	0.062	0.166	0.056	0.159	0.021	0.088	0.023	0.092	0.133	0.270	0.108	0.228
	25%	**0.015**	**0.070**	0.017	**0.070**	0.017	0.074	0.024	0.090	0.020	0.080	0.020	0.085	0.028	0.099	0.164	0.294	0.042	0.143	0.085	0.196	0.080	0.195	0.024	0.096	0.026	0.101	0.135	0.272	0.136	0.262
	37.5%	**0.016**	0.074	0.018	**0.070**	0.017	0.078	0.028	0.106	0.022	0.087	0.023	0.091	0.030	0.104	0.237	0.356	0.051	0.159	0.106	0.222	0.110	0.231	0.027	0.103	0.030	0.108	0.155	0.293	0.175	0.300
	50%	0.021	0.082	0.022	**0.079**	**0.019**	0.086	0.032	0.111	0.025	0.095	0.026	0.098	0.034	0.110	0.323	0.421	0.059	0.174	0.131	0.247	0.156	0.276	0.030	0.108	0.035	0.119	0.200	0.333	0.211	0.329
	Avg	**0.0158**	0.0730	0.0178	**0.0703**	0.0168	0.0770	0.0260	0.0980	0.0210	0.0845	0.0218	0.0885	0.0295	0.1018	0.2080	0.3275	0.0465	0.1508	0.0960	0.2078	0.1005	0.2153	0.0255	0.0988	0.0285	0.1050	0.1558	0.2920	0.1575	0.2798
ETTm1	12.5%	**0.028**	**0.106**	0.030	0.116	0.034	0.132	0.038	0.132	0.043	0.140	0.057	0.159	0.093	0.201	0.126	0.263	0.240	0.345	0.151	0.267	0.070	0.190	0.060	0.165	0.074	0.182	0.114	0.234	0.074	0.194
	25%	**0.034**	**0.121**	0.038	0.130	0.041	0.147	0.046	0.142	0.054	0.156	0.069	0.178	0.107	0.217	0.169	0.304	0.265	0.364	0.180	0.292	0.106	0.236	0.080	0.189	0.090	0.203	0.140	0.262	0.102	0.227
	37.5%	0.062	0.151	0.061	**0.150**	0.065	0.160	**0.058**	0.162	0.072	0.180	0.084	0.196	0.120	0.230	0.220	0.347	0.296	0.382	0.215	0.318	0.124	0.258	0.102	0.212	0.109	0.222	0.174	0.293	0.135	0.261
	50%	0.084	0.189	0.086	0.192	0.091	0.202	**0.074**	**0.180**	0.107	0.216	0.102	0.215	0.141	0.248	0.293	0.402	0.334	0.404	0.257	0.347	0.165	0.299	0.133	0.240	0.137	0.248	0.215	0.325	0.179	0.298
	Avg	**0.0520**	**0.1418**	0.0538	0.1470	0.0578	0.1603	0.0540	0.1540	0.0690	0.1730	0.0780	0.1870	0.1153	0.2240	0.2020	0.3290	0.2838	0.3738	0.2008	0.3060	0.1163	0.2458	0.0938	0.2015	0.1025	0.2138	0.1608	0.2785	0.1225	0.2450
ETTm2	12.5%	0.024	0.106	**0.020**	**0.100**	0.028	0.102	0.045	0.128	0.039	0.125	0.040	0.130	0.057	0.152	0.187	0.319	0.101	0.231	0.100	0.216	0.095	0.212	0.042	0.133	0.044	0.138	0.305	0.431	0.163	0.289
	25%	0.025	0.124	**0.021**	**0.104**	0.034	0.133	0.052	0.153	0.044	0.135	0.046	0.141	0.061	0.158	0.279	0.390	0.115	0.246	0.127	0.247	0.137	0.258	0.049	0.147	0.050	0.149	0.322	0.444	0.206	0.331
	37.5%	**0.048**	0.118	**0.048**	**0.110**	0.063	0.153	0.077	0.175	0.051	0.147	0.052	0.151	0.067	0.166	0.400	0.465	0.126	0.257	0.158	0.276	0.187	0.304	0.056	0.158	0.060	0.163	0.353	0.462	0.252	0.370
	50%	0.052	0.121	0.048	**0.110**	**0.033**	0.151	0.075	0.160	0.059	0.158	0.060	0.162	0.073	0.174	0.602	0.572	0.136	0.268	0.183	0.299	0.232	0.341	0.065	0.170	0.068	0.173	0.369	0.472	0.316	0.419
	Avg	0.0373	0.1173	**0.0343**	**0.1080**	0.0395	0.1348	0.0623	0.1540	0.0483	0.1413	0.0495	0.1460	0.0645	0.1625	0.3670	0.4365	0.1195	0.2505	0.1420	0.2595	0.1628	0.2788	0.0530	0.1520	0.0555	0.1558	0.3373	0.4523	0.2343	0.3523

Table 14: Full results for the classification task on UEA Archive in the Transformers indicates the name of $*$former. We report the classification accuracy (%) as the result. The standard deviation is within 0.1%. **Bold**: the best, Underline: the second best.

Datasets / Models	Transformers									MLP		CNN		Pre-trained FMs				
	Trans.	Re.	In.	Pyra.	Auto.	Station.	FED.	ETS.	Flow.	DLinear	LightTS.	TimesNet	MTCN	MOMENT	FedAvg	FFTS	FeDaL	**TimeOperator**
EthanolConcentration	32.7	31.9	31.6	30.8	31.6	32.7	31.2	28.1	33.8	32.6	29.7	35.7	36.3	30.0	27.7	32.1	35.4	89.7
FaceDetection	67.3	68.6	67.0	65.7	68.4	68.0	66.0	66.3	67.6	68.0	67.5	68.6	70.8	68.9	65.2	69.0	70.0	67.2
Handwriting	32.0	27.4	32.8	29.4	36.7	31.6	28.0	32.5	33.8	27.0	26.1	32.1	30.6	35.1	35.0	37.2	37.3	26.6
Heartbeat	76.1	77.1	80.5	75.6	74.6	73.7	73.7	71.2	77.6	75.1	75.1	78.0	77.2	73.7	74.0	75.2	80.2	72.7
JapaneseVowels	98.7	97.8	98.9	98.4	96.2	99.2	98.4	95.9	98.9	96.2	96.2	98.4	98.8	95.7	94.2	94.7	99.6	99.8
PEMS-SF	82.1	82.7	81.5	83.2	82.7	87.3	80.9	86.0	83.8	75.1	88.4	89.6	89.1	85.5	86.0	84.5	85.6	98.8
SelfRegulationSCP1	92.2	90.4	90.1	88.1	84.0	89.4	88.7	89.6	92.5	87.3	89.8	91.8	93.4	88.7	89.1	90.5	95.1	76.5
SelfRegulationSCP2	53.9	56.7	53.3	53.3	50.6	57.2	54.4	55.0	56.1	50.5	51.1	57.2	60.3	55.0	56.2	55.7	59.5	61.7
SpokenArabicDigits	98.4	97.0	100.0	99.6	100.0	100.0	100.0	100.0	98.8	81.4	100.0	99.0	98.7	99.4	98.0	99.5	100.0	99.0
UWaveGestureLibrary	85.6	85.6	85.6	83.4	85.9	87.5	85.3	85.0	86.6	82.1	80.3	85.3	86.7	90.0	87.6	80.6	98.5	72.19
Average Accuracy	71.9	71.5	72.1	70.8	71.1	72.7	70.7	71.0	73.0	67.5	70.4	73.6	74.2	72.2	71.5	71.7	76.1	**76.4**

computation scales linearly with Q. The small measured latency increase reflects parallel query evaluation rather than constant computational complexity.

Hypernetwork overhead. Table 17 compares the full model with the variant without the context-adaptive hypernetwork. The hypernetwork adds 11.0M parameters and increases training-step time by 3.6% and inference latency by 2.7%. Training memory increases from 6.86 to 7.05 GiB, while peak inference memory increases from 120.4 to 154.5 MiB. These measurements show that context adaptation primarily adds parameter and memory costs, with a smaller effect on measured runtime.

C.7 Nonstationarity stress test

We compare the full multi-basis trunk with a Fourier-only trunk under level shifts, trend reversals, variance shifts, and dominant-frequency shifts using existing checkpoints without retraining or fine-tuning. Each transition occurs within the observed context, and MSE is measured over the subsequent forecast region. Table 18 shows that the full trunk improves performance under all four changes, reducing average MSE by 25.3%.

C.8 Boundary stability

We evaluate stability near the context–forecast boundary on ETTh1, averaging over horizons $H \in \{96, 192, 336, 720\}$. All models use the same test windows, context lengths, normalization, and forecast horizons. We report full-horizon MSE and two metrics over the first 10% of the forecast horizon: prediction MSE and curvature error, defined as the MSE between predicted and ground-truth

Table 15: Average linear-probe accuracy on 10 UEA datasets. Higher is better.

Representation	Accuracy (%)
Branch only	**76.40**
Branch + trunk	60.08

Table 16: Inference cost across query counts with a fixed 720-step context on an NVIDIA A6000 using FP32 and batch size one.

Queries Q	Median latency (ms)	Peak memory (MiB)	Hypernetwork time (ms)
96	1.780	137.9	0.110
336	1.777	138.9	0.111
720	1.781	141.5	0.111
1440	1.803	145.6	0.111

second differences. TimeOperator achieves the lowest full-horizon MSE and curvature error, and the second-lowest boundary MSE (Table 19). These results provide empirical evidence of boundary stability under the evaluated settings.

D Limitations.

TimeOperator does not remove discretization from the input side. The model is conditioned on finite sampled observations, and both the branch encoder and FFT-based spectral descriptors are computed from those samples. Our claim is about the output interface: once the context is encoded, output frequency, forecast horizon, and imputation locations are controlled through the query set $\mathcal{T}_q$ rather than by a fixed decoder shape.

E Broader impact.

A queryable time-series operator can be useful in domains where users need predictions at different temporal resolutions, such as energy systems, weather monitoring, healthcare sensors, and industrial telemetry. Potential risks arise if forecasts or imputations are used without domain validation in high-stakes settings, especially under distribution shift, sparse context, or noisy measurements. We recommend reporting uncertainty when available, checking for dataset leakage, and validating the model under deployment-specific sampling patterns before use.

Table 17: Resource costs of TimeOperator with and without the context-adaptive hypernetwork.

Metric	w/o hyper-trunk	Full TimeOperator
Parameters (M)	29.5	40.5
Training-step time (ms)	151.9	157.3
Training memory (GiB)	6.86	7.05
Inference latency (ms)	1.734	1.781
Peak inference memory (MiB)	120.4	154.5

Table 18: Zero-shot forecasting MSE under four types of nonstationarity. Lower is better.

Change	Fourier-only	Full multi-basis
Level shift	1.2326	**1.0186**
Trend reversal	1.2012	**0.7885**
Variance shift	1.4439	**0.8742**
Frequency shift	1.7270	**1.5057**
Average	1.4012	**1.0468**

Table 19: Boundary stability on ETTh1 averaged over four forecast horizons. Boundary metrics cover the first 10% of the forecast horizon. Lower is better.

Model	Full-horizon MSE	Boundary MSE	Curvature error
TimeOperator	**0.467**	<u>0.411</u>	**0.416**
Fourier-only	<u>0.485</u>	0.451	0.466
TiRex	0.489	**0.402**	0.436
FlowState-r1.1	0.512	0.449	0.431
Chronos-Bolt	0.556	0.438	<u>0.430</u>