MoTM: Towards a Foundation Model for Time Series Imputation based on Continuous Modeling

Etienne Le Naour*, Tahar Nabil*, Ghislain Agoua

EDF R&D
{name.surname}@edf.fr

Abstract. Recent years have witnessed a growing interest for time series foundation models, with a strong emphasis on the forecasting task. Yet, the crucial task of out-of-domain imputation of missing values remains largely underexplored. We propose a first step to fill this gap by leveraging implicit neural representations (INRs). INRs model time series as continuous functions and naturally handle various missing data scenarios and sampling rates. While they have shown strong performance within specific distributions, they struggle under distribution shifts. To address this, we introduce *MoTM* (Mixture of Timeflow Models), a step toward a foundation model for time series imputation. Building on the idea that a new time series is a mixture of previously seen patterns, *MoTM* combines a basis of INRs, each trained independently on a distinct family of time series, with a ridge regressor that adapts to the observed context at inference. We demonstrate robust in-domain and out-of-domain generalization across diverse imputation scenarios (e.g., block and pointwise missingness, variable sampling rates), paving the way for adaptable foundation imputation models.

Keywords: Imputation · Foundation model · Implicit Neural Representations.

1 Introduction

Real-world time series from domains such as healthcare, industry and climate science are often irregularly sampled or incomplete due to sensor failures and decentralized data collection [21,6]. Reliable imputation is thus a critical first step toward downstream tasks like forecasting, classification, or anomaly detection. Yet, while recent deep learning methods have advanced imputation performance [4,9], they typically lack robustness to distribution shifts and fail to generalize to out-of-domain data.

Recently, zero-shot forecasting models have emerged in the time series community, enabling inference on unseen datasets without retraining. This shift has led to the rise of time series foundation models, offering key benefits: (i) a single,

* Equal contribution

10th Workshop on Advanced Analytics and Learning on Temporal Data (AALTD), ECML 2025

deployable model for diverse use cases, (ii) strong performance on new datasets, often exceeding supervised baselines, and (iii) emerging capabilities beyond simple memorization. While forecasting foundation models are well-studied [8,25,1], imputation-focused counterparts remain scarce. Notable attempts like NuwaTS [5] and MOMENT [12] address imputation, but overlook data heterogeneity and varying sampling rates by relying on fixed-length input segments, limiting their ability to exploit shared patterns like periodicities across datasets.

A promising direction to overcome this issue lies in time series continuous-time modeling, recently advanced through the use of implicit neural representations (INR) [26,14,15]. These models enable time series to be represented as continuous functions, making them particularly suitable for imputation tasks involving irregular sampling or unaligned timestamps. Among these, TimeFlow [14] has demonstrated competitive imputation performance, often matching or surpassing both traditional statistical methods and deep learning-based approaches. However, while TimeFlow excels within a specific data distribution, it struggles to generalize across distributions, limiting its utility in out-of-domain (OOD) settings.

To address these limitations, we introduce MoTM (Mixture of TimeFlow Models), a novel mixture-based architecture. MoTM leverages a ridge regression mechanism at inference to aggregate latent representations from multiple TimeFlow models, each trained on a distinct domain. Our contributions are as follows.

- We propose MoTM, a unified model capable of (i) handling various patterns of missing values at inference; (ii) achieving strong performance on out-of-domain datasets without retraining; (iii) effectively managing datasets sampled at different rates (e.g., 10min, 30min, 1h, 2h) by leveraging shared temporal structures across distributions. To the best of our knowledge, MoTM is the first model to meet all of these conditions (see Table 1).
- Our experiments on synthetic datasets reveal that MoTM exhibits strong zero-shot imputation capabilities that go beyond mere memorization. Notably, it generalizes effectively to time series subject to strong distribution shifts without any additional training.
- On real-world datasets, MoTM surpasses baseline models on in-domain (ID) inference and matches the performance of the strongest supervised approaches in the out-of-domain (OOD) setting.

Table 1. Comparison of imputation models on key generalization capabilities.

Method	Can Impute Various Missing Patterns	Natively Support Different Sampling Rates	Can Perform OOD Inference
BRITS [4],SAITS [9]	✓	✗	✗
NuwaTS [5],MOMENT [12]	✓	✗	✓
TimeFlow [14], ImputeINR [15]	✓	✓	✗
MoTM (Ours)	✓	✓	✓

2 Related Work

The task of imputing missing values in time series has been extensively studied, with approaches ranging from classical statistical methods to recent advances in deep learning. In this section, we review relevant lines of work, focusing on three major directions: supervised deep imputation models, continuous-time representations, and the recent emergence of time series foundation models.

Supervised imputation. Recent advances in deep learning have led to an increasing number of models for time series imputation. BRITS [4] pioneered the use of bidirectional RNNs for imputation, while subsequent methods explored alternative architectures such as GANs [17,18], VAEs [10], diffusion models [24], matrix factorization techniques [16], and Transformer-based models like SAITS [9]. Despite their success, these models assume regularly sampled data limiting their flexibility in real-world applications. Moreover, their generalization capabilities in out-of-domain (OOD) settings remain limited.

Continuous-time models. Continuous-time modeling has emerged as a promising approach to handle irregular sampling in time series. Gaussian Processes (GPs) [19] naturally represent functions over continuous domains but often struggle with scalability and kernel selection [7]. Neural Processes (NPs) [11,13] provide a more scalable alternative by parameterizing GPs through encoder-decoder architectures, yet they remain challenged by complex, high-frequency signals. More recent extensions use diffusion-based priors [2], but these approaches can be sensitive to the number of input timestamps. Other directions involve latent ODEs [3,20], or attention-based methods like mTAN [22], which model irregular time series in a continuous domain. However, these models often fall short in imputation accuracy compared to their discrete-time counterparts. Implicit neural representations (INRs) [23] have recently gained traction as a more expressive and flexible framework for continuous modeling [26,14,15], but existing models like TimeFlow [14] still exhibit limited generalization capabilities across distributions.

Foundation models. The emergence of foundation models for time series represents a shift towards models capable of zero-shot generalization across diverse datasets. In the forecasting domain, recent models such as Chronos [1], Moirai [25], and TimesFM [8] have demonstrated strong performance without fine-tuning, enabling deployment in heterogeneous environments. In contrast, foundation models for imputation remain underexplored. NuwaTS [5] and MOMENT [12] are among the few attempts in this direction, but both rely on fixed-length segments and struggle with irregular sampling or variable-resolution datasets. These limitations restrict their ability to model shared temporal structures across datasets with diverse characteristics; highlighting the need for more flexible, distribution-aware imputation models.

3 The MoTM Framework

3.1 Problem Setting

We formalize the generalizable imputation problem across heterogeneous time series datasets. Our goal is to learn a universal model capable of imputing missing values over time series that vary in sampling frequency, temporal alignment, and underlying distribution. We now describe the notations and the imputation objectives at both training and inference time.

Data notations. We consider a collection of N_{train} training datasets, denoted by $\mathcal{D}_{\text{train}} = \{\mathcal{D}_i\}_{i=1}^{N_{\text{train}}}$, where each dataset $\mathcal{D}_i$ consists of n_i time series: $\mathcal{D}_i = \{(\mathbf{x}^{(i,j)}, \mathcal{T}_{\text{obs}}^{(i,j)})\}_{j=1}^{n_i}$.

- $\mathbf{x}^{(i,j)} \in \mathbb{R}^{T_j}$ denotes the j^{th} time series in dataset i, consisting of T_j observed values.
- The temporal grid $\mathcal{T}_{\text{obs}}^{(i,j)} \subset [0,1]$ is the set of T_j observed timestamps associated with $\mathbf{x}^{(i,j)}$. To facilitate learning shared temporal patterns across datasets with varying sampling rates, we rescale all time grids to lie within the interval $[0,1]$. For example, if we consider a time period spanning four weeks, the first time point is mapped to 0 and the last to 1. This common temporal reference allows us to align heterogeneous time series, regardless of sampling frequency, missing values, or alignment issues.

A visual illustration of the notations is provided below in Fig. 1.

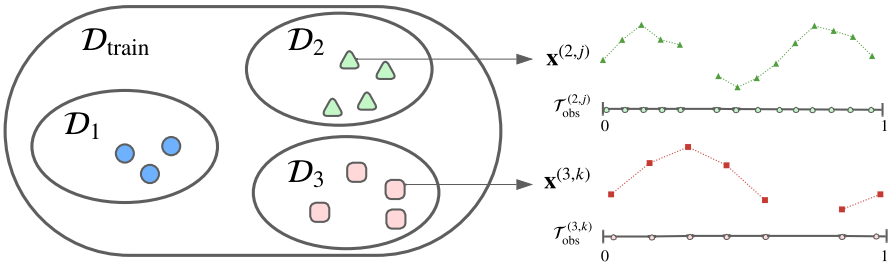


Fig. 1. Illustration of the notations used in the rest of this paper.

Imputation Task. During training, the goal is to learn a unified model f_θ capable of predicting the value x_t at any given time $t \in [0,1]$ for any time series from any training dataset in $\mathcal{D}_{\text{train}}$. *At inference time*, we aim for two generalizations:

- *In-Domain Generalization:* accurately impute values x_t for new time series from the training datasets ($\mathcal{D} \subset \mathcal{D}_{\text{train}}$).
- *Out-of-Domain (OOD) Generalization:* accurately impute values x_t for entirely new datasets not seen during training ($\mathcal{D}_{\text{new}} \not\subset \mathcal{D}_{\text{train}}$).

3.2 Key Components

Our framework is articulated around three key components:

1. **Pretraining: learn a basis of TimeFlow models on the training corpus D_{train}.** Each dataset in the training collection is used to learn a distinct TimeFlow model. These models capture per-dataset specific temporal patterns and collectively form a representative basis of diverse dynamics.
2. **At inference step 1: adapt the basis of TimeFlow models for the target time series.** For each trained model in the basis, we optimize a latent code to best fit the new target series. This results in a set of modulated Implicit Neural Representations, each proposing a reconstruction of the input series from its own perspective.
3. **At inference step 2: fit the orchestrator, here a ridge regressor, on top of the basis of TimeFlow models.** We extract hidden representations from each modulated model and combine them to form a shared feature space. A ridge regression is then trained to linearly combine these features for final imputation.

In the following sections, we will elaborate on each component of our method.

TimeFlow architecture. TimeFlow [14] is an Implicit Neural Representation (INR) model, meaning that it is a neural network capable of learning a parameterized continuous function of time $f_\theta : t \in [0,1] \mapsto f_\theta(t) \in \mathbb{R}$ that approximates a discrete time series $\mathbf{x}_t$ at any time $t \in [0,1]$.

In contrast to plain INRs, which are typically designed to represent a single function (e.g., one time series), TimeFlow is a *generalizable* INR, able to model an entire collection $(x^{(j)})_j$ of time series. This is achieved by incorporating per-sample modulations (through per-sample additive bias $\psi^{(j)}$) that condition the function f_θ on each specific instance. See Fig. 2 for a visualization. We refer to the original TimeFlow paper for a detailed description of this mechanism [14].

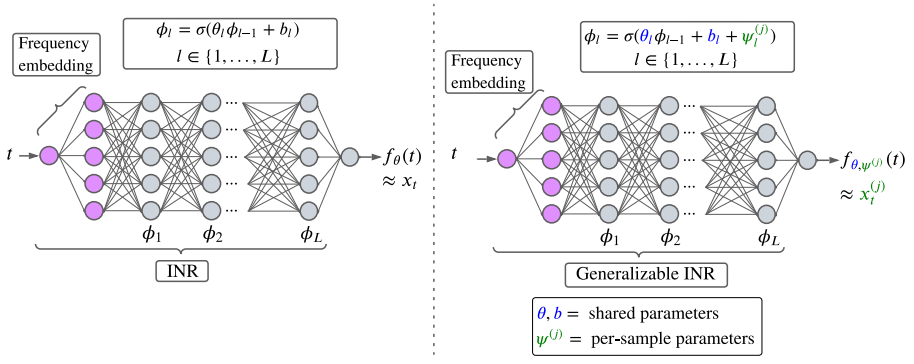


Fig. 2. Plain INR vs Generalizable INR.

Learning a basis of TimeFlow models. For each training dataset $\mathcal{D}_i \in \mathcal{D}_{\text{train}}$, we train a distinct TimeFlow model $f_{\theta^{(i)}}$. This model is an INR conditioned by modulations optimized through a latent code $z^{(i,j)}$ specific to each time series j of dataset i. Formally, the ouput of a TimeFlow model for the j-th series in dataset i is defined as:

$$\hat{x}_t^{(i,j)} = f_{\theta^{(i)},h_{w^{(i)}}}(t; z^{(i,j)}) := f_{\theta^{(i)},\psi^{(i,j)}}(t), \tag{1}$$

where $\psi^{(i,j)} = h_{w^{(i)}}(z^{(i,j)})$ is the modulation vector conditioning the INR's biases and $h_w(.)$ is a hypernetwork.

Each distinct TimeFlow instance is trained to optimize the following objective:

$$\min_{\theta^{(i)},\, w^{(i)},\, \{z^{(i,j)}\}} \sum_{j=1}^{n_i} \mathcal{L}_{\mathcal{T}_{\text{obs}}^{(i,j)}}\big(x_t^{(i,j)}, f_{\theta^{(i)},h_{w^{(i)}}}(t; z^{(i,j)})\big), \tag{2}$$

where $\mathcal{L}_{\mathcal{T}_{obs}}$ is the reconstruction loss (e.g., MSE loss function) computed over observed time points $t \in \mathcal{T}_{\text{obs}}$. The result is a **basis of N_{train} TimeFlow models** $\{f_{\theta^{(i)},h_{w^{(i)}}}\}_{i=1}^{N_{\text{train}}}$, each capturing dynamics specific to dataset i.

Adapting the basis of TimeFlow models. At inference, the basis is adapted to a new dataset $\mathcal{D}_{\text{new}} = \{(\mathbf{x}^{(j)}, \mathcal{T}_{\text{obs}}^{(j)})\}_{j=1}^{n_{\text{new}}}$ by optimizing, for each model i, a latent code $z^{(i,j)*}$ per series j, keeping the shared parameters fixed:

$$z^{(i,j)*} = \arg\min_{z} \mathcal{L}_{\mathcal{T}_{\text{obs}}^{(j)}}\big(x_t^{(j)}, f_{\theta^{(i)},h_{w^{(i)}}}(t; z)\big). \tag{3}$$

Since TimeFlow is trained using a meta-learning approach [27], adapting it to new time series involves quickly computing the corresponding latent codes $z^{(i,j)*}$ for each model i and new series j. This adaptation requires only a few optimization steps based on the observed context (see [14]). This process can be viewed as an *inner-loop* optimization, allowing the pretrained models to adjust efficiently to new samples.

This yields a family of modulated INRs $\{f_{\theta^{(i)},h_{w^{(i)}}}(t; z^{(i,j)*})\}_{i=1}^{N_{\text{train}}}$ providing different reconstructions of series j.

Fitting the orchestrator. To combine predictions from the adapted basis, we extract a latent representation $r^{(i,j)} \in \mathbb{R}^d$ from each modulated model i for each observed values of series j, typically the last hidden layer of each modulated INR of the basis:

$$r^{(i,j)}(t) := \text{hidden\_repr}\big(f_{\theta^{(i)},h_{w^{(i)}}}(t; z^{(i,j)*})\big) \quad \text{for } t \in \mathcal{T}_{obs}^{j}. \tag{4}$$

The representations obtained by each one of the N_{train} TimeFlow instances for the different observed timesteps are concatenated:

$$\mathbf{R}_{obs}^{(j)} = \begin{bmatrix} r^{(1,j)}(t_1) & r^{(2,j)}(t_1) & \dots & r^{(N_{\text{train}},j)}(t_1) & 1 \\ r^{(1,j)}(t_2) & r^{(2,j)}(t_2) & \dots & r^{(N_{\text{train}},j)}(t_2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r^{(1,j)}(T_j) & r^{(2,j)}(T_j) & \dots & r^{(N_{\text{train}},j)}(T_j) & 1 \end{bmatrix} \in \mathbb{R}^{T_j \times (N_{\text{train}} \cdot d + 1)}. \tag{5}$$

Then, for each time series j, a ridge regression model is independently fitted to predict the observed values of $\mathbf{x}^{(j)} \in \mathbb{R}^{T_j \times 1}$ as a linear combination of the representations in $\mathbf{R}_{obs}^{(j)}$:

$$\hat{\mathbf{x}}^{(j)} = \mathbf{R}_{obs}^{(j)} \mathbf{W}^{(j)}, \tag{6}$$

where $\mathbf{W}^{(j)} \in \mathbb{R}^{(N_{\text{train}} \cdot d+1) \times 1}$ contains the regression coefficients and the intercept. A visualization of this process is shown in Fig. 3.

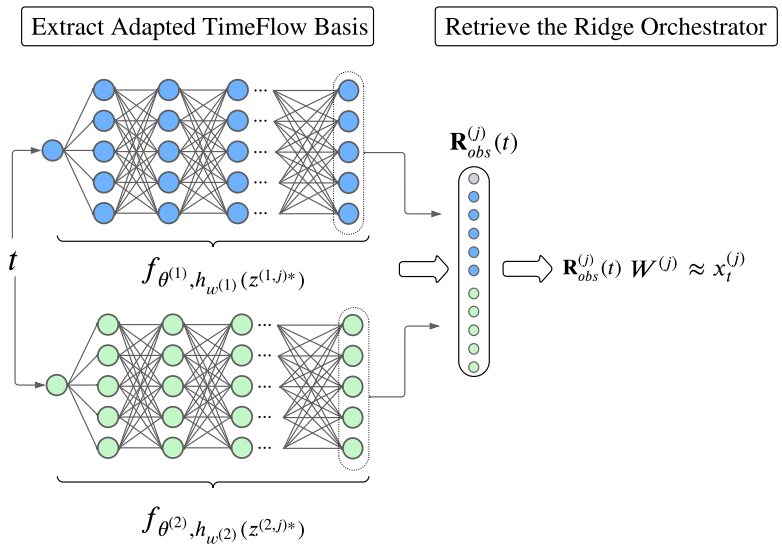


Fig. 3. Illustration of how the ridge orchestrator operates on a new time series $\mathbf{x}^{(j)}$, using a basis of two TimeFlow models. Note that the linear projection matrix $W^{(j)}$ is jointly optimized over all observed time steps of the new series $\mathbf{x}^{(j)}$.

These parameters are obtained by solving the following regularized least-squares problem:

$$\mathbf{W}^{*(j)} = \underset{\mathbf{W}^{(j)}}{\arg\min} \|\mathbf{x}^{(j)} - \mathbf{R}_{obs}^{(j)} \mathbf{W}^{(j)}\|_2^2 + \lambda \|\mathbf{W}^{(j)}\|_2^2, \tag{7}$$

with $\lambda \geq 0$. This optimization admits a closed-form solution, making the computation both efficient and scalable, even when the number of inputs is large. As such, the regression step acts as an *orchestrator*, learning how to best combine the outputs of multiple TimeFlow models to produce robust and generalizable imputations for unseen time series.

At inference: to predict for new target timestamps coordinates, we just have to build its representation $\mathbf{R}_{target}^{(j)}$ and compute $\mathbf{R}_{target}^{(j)} \mathbf{W}^{*(j)}$.

4 Experiments

We design two types of experiments to evaluate the performance of our proposed method. (i) First, we consider controlled synthetic datasets that allow us to analyze the behavior of MoTM in a well-understood setting. (ii) Second, we evaluate MoTM on real-world datasets to assess the applicability and generalization capacity of our approach in more complex and diverse scenarios.

4.1 Experiments on Synthetic Data

We begin the empirical evaluation of MoTM by designing a synthetic experiment. By controlling the seasonalities of the generated datasets, we aim to assess whether MoTM can generalize to an unseen combination of known patterns.

Data generating process. We generate three synthetic datasets summarized in Table 2, using Gaussian Processes (GP) *via* KernelSynth [1]. Each synthetic series is the sum of three components, namely: (i) a smooth trend sampled from a GP with RBF kernel; (ii) a seasonal component sampled from a GP with a periodic Exponential-Sine-Squared kernel; (iii) a residual sampled from a GP with white noise kernel. We create $N_{\texttt{train}} = 2$ datasets for pretraining, `ks1D` (resp., `ks1W`) with an hourly sampling frequency (resp., half-hourly) and daily (resp., weekly) periodicities. A chronological 0.75 - 0.25 train - test split is applied on both `ks1D` and `ks1W`. The third dataset `ks1D1W`, sampled at a 15-min rate with both a daily and a weekly periodic component, is kept for inference only.

Train protocol. Following the procedure described in Section 3, we train one TimeFlow model on `ks1D` and another on `ks1W`. Both models are trained on time grids spanning four-week periods. To condition each TimeFlow on partially observed time series, we simulate missing data during training by randomly removing a subset of timesteps from the grids $\mathcal{T}_{obs}$ at each optimization step, resulting in sparse inputs with a missingness ratio $\tau \in \{0.01, 0.2, 0.3, \ldots, 0.7\}$. At inference time, as detailed in Section 3, MoTM reuses both pretrained TimeFlow models to fit a ridge regressor on new partially observed time series.

Test protocol. The test split is divided into four-week segments. For each segment, we generate four distinct missing data scenarios, by randomly removing: either (i) 50% (*Point 1*) and (ii) 70% (*Point 2*) of the observations; or (iii) two entire days (*Block 1*) and (iv) four entire days (*Block 2*).

Table 2. Synthetic datasets generated by KernelSynth [1]. SNR: Signal-to-Noise Ratio.

Dataset	Samples	Length	Sampling Freq.	RBF Scale	Period	Average SNR (dB)
`ks1D`	100	4032	1H	1.5	1D	20.6
`ks1W`	100	5376	30min	5	1W	22.3
`ks1D1W`	100	5376	15min	1.25	1D + 1W	14.9

Implementation. TimeFlow's hyperparameters (see [14]) are chosen as follows: latent code of dimension 128, linear hypernetwork of size 256, INR with a 2×64 frequency embedding of time, five 128-dimensional hidden layers. The latent code is computed with 3 inner loop steps and a learning rate of 0.05. We train with a batch size of 64 for 5×10^3 epochs, with a 10^{-3} learning rate for the INRs and hypernetworks. After training, we set $\lambda = 2$ for MoTM in all inference settings.

Baselines. We compare MoTM against several baselines: • `TimeFlow 1D` and `TimeFlow 1W` respectively use predictions from a single TimeFlow model trained on `ks1D` and `ks1W` • `Linear` performs standard linear interpolation between observed points • `Repeat` imputes each missing value by copying the most recent available observation from the desired seasonnality. Specifically, it uses the last available value from the previous day in the case of daily seasonality, or from the previous week in the case of weekly seasonality. • `Mixture I` aggregates the predictions of both pretrained TimeFlow models using a softmax-weighted average, with weights derived from the negative reconstruction scores on the available context $\mathcal{T}_{obs}^j$ • `Mixture II` fits a ridge regressor on the observed context $\mathcal{T}_{obs}^j$, using the output predictions of both pretrained TimeFlow models as features.

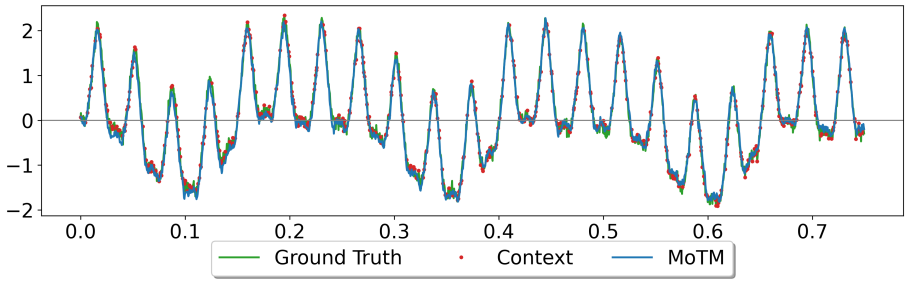
Results. Table 3 shows that MoTM achieves good performances on the in-domain datasets `ks1D` and `ks1W`, on a par with the strong supervised baselines. We also note that `Mixture II`, the ridge regression on top of the TimeFlows predictions, emerges as a simple method to match the best pretrained model at inference.

Table 3. Mean Absolute Errors (MAEs) on test series, z-normalized using the available context. Best results are shown in **bold**, and second-best are underlined. MoTM results correspond to $\lambda = 2$. MoTM improvement reports the relative improvement of MoTM over each baseline, averaged across all rows.

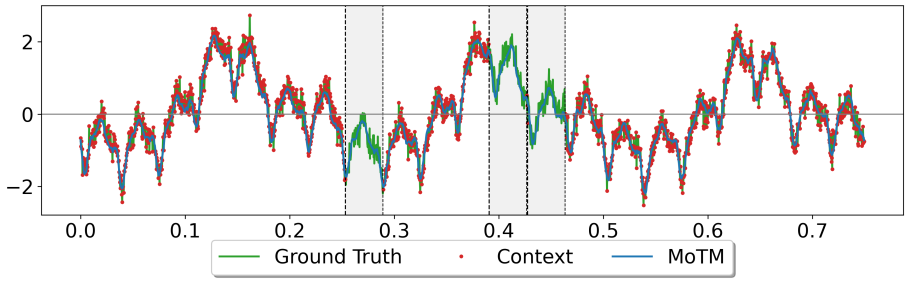
		MoTM	Mixture I	Mixture II	TimeFlow 1D	TimeFlow 1W	Linear	Repeat
MoTM In-Domain								
ks1D	*Point 1*	0.238	0.382	0.232	**0.231**	0.860	0.387	0.316
	Point 2	0.246	0.389	0.237	**0.236**	0.868	0.530	0.317
	Block 1	0.232	0.386	0.228	**0.227**	0.889	1.157	0.317
	Block 2	0.232	0.389	**0.228**	**0.228**	0.894	1.154	0.316
ks1W	*Point 1*	0.122	0.309	**0.120**	0.834	0.121	0.146	0.292
	Point 2	0.126	0.318	**0.123**	0.849	**0.123**	0.149	0.459
	Block 1	**0.122**	0.338	0.124	0.891	0.127	0.217	0.164
	Block 2	**0.123**	0.349	0.130	0.901	0.134	0.233	0.172
MoTM improvement		**0.0%**	**50.4%**	**-1.0%**	**44.4%**	**40.4%**	**48.7%**	**36.0%**
MoTM Out-Of-Domain								
ks1D1W	*Point 1*	**0.145**	0.409	0.183	0.583	0.583	0.154	0.743
	Point 2	**0.148**	0.412	0.195	0.585	0.585	0.164	0.806
	Block 1	**0.153**	0.440	0.201	0.617	0.624	0.778	0.654
	Block 2	**0.155**	0.446	0.214	0.619	0.628	0.780	0.661
MoTM improvement		**0.0%**	**64.8%**	**24.2%**	**75.0%**	**75.2%**	**48.8%**	**78.9%**

Most notably, MoTM achieves strong results on `ks1D1W` in the OOD setting, highlighting its ability to generalize beyond simple memorization. Thanks to its continuous-time formulation, MoTM naturally adapts to the new sampling rate of `ks1D1W`, while the ridge orchestrator further improves performance, reducing the MAE by approximately 75% compared to both pretrained TimeFlow variants. Indeed, the TimeFlow models trained individually on `ks1D` and `ks1W` transfer poorly to `ks1D1W`, confirming that TimeFlow can fit well within its training domain but struggles to generalize across distributions. In contrast, MoTM leverages the daily periodicity learned from `ks1D` and the weekly periodicity from `ks1W`, enabling it to handle the mixed `ks1D1W` dataset in a zero-shot manner.

Visually, Fig. 4 showcases MoTM's ability to fit the context and impute missing values in challenging scenarios.



(a) *Point 2*



(b) *Block 2*

Fig. 4. *OOD `ks1D1W` dataset.* MoTM performs imputation on (a) 70% missing timesteps and (b) three one-day missing blocks. Zoom on the first three weeks.

4.2 Experiments on Real World Data

In this section, we evaluate the performance of MoTM on various real-world datasets, covering both in-domain and out-of-domain scenarios. Our goal is to assess its ability to generalize from a small set of heterogeneous datasets to a

Table 4. Datasets used for our experiments. SNR: Signal-to-Noise Ratio.

Dataset	Samples	Total Series Length	Sampling Freq.	Average SNR (dB)	MoTM Splits	Input Length
`Electricity`	321	26 304	1H	24.6	Train, Infer.	672
`Solar`	137	52 560	10min	17.4	Train, Infer.	4 032
`SpanishW-T`	5	35 000	1H	35.3	Train, Infer.	672
`Traffic`	861	17 544	1H	18.3	Inference	672
`ETTh1`	7	17 420	1H	11.8	Inference	672
`ETTh2`	7	17 420	1H	11.6	Inference	672
`Weather`	11	35 064	1H	14.7	Inference	672
`Spanish E`	9	35 064	1H	22.0	Inference	672

new target dataset. Our experiments include comparisons with zero-shot, deep supervised, and statistical baselines under challenging imputation settings.

Datasets. For these experiments, the MoTM model is built on $N_{\text{train}} = 3$ TimeFlow models, each trained on one of the following datasets: (i) `Electricity`, (ii) `Solar`, (iii) and `SpanishW-T`. These datasets were selected to form a rich training domain, with diversity in sampling frequencies, seasonal patterns (daily and weekly for `Electricity`, daily for `Solar` and `SpanishW-T`), sample sizes (ranging from 5 to 321), and within-series variability. For out-of-domain evaluation, we use standard benchmarks: `Traffic`, `ETTh1`, `ETTh2`, `Weather`, and `Spanish Energy`. Their key characteristics are summarized in Table 4.

Protocol. All datasets are split chronologically intro train, validation and test fractions with respective ratios 0.7 - 0.1 - 0.2. The ridge regularization coefficient λ is selected from a grid search over $\{0.01, 0.1, 0.5, 1, 5, 10\}$ on the validation split of the three training datasets. The rest of the protocol for training and evaluation follows the one described for synthetic data.

Implementation. The hyperparameters for TimeFlow are the same as in Section 4.1, except for an INR hidden size of 256. We allow training for 40k epochs on `Electricity` and `Solar`, 20k epochs on `SpanishW-T`. For MoTM, the grid search yields $\lambda = 0.5$ for pointwise imputation and $\lambda = 1$ for block imputation.

Baselines. `MoTM` is compared against three groups of baselines. • We use `MOMENT` [12] as another *Zero-shot* foundation model, based on its Large version in inference mode across all datasets. • The *Supervised* baselines `TimeFlow` [14], `BRITS` [4] and `SAITS` [9] are state-of-the-art deep imputation models trained on the respective train splits of each target dataset - including those tagged as OOD for MoTM. • The *Statistical* baselines are the `Linear` and `Repeat` interpolations described in the previous section.

Results. Several observations emerge from the test results reported in Table 5. (i) MoTM consistently outperforms the zero-shot baseline MOMENT, which

Table 5. MAEs on the test fraction of the real world datasets. Best performance emphasized in **bold**, second best underlined. MoTM improvement reports the relative improvement of MoTM over each baseline, averaged across all rows.

		Zero-shot		*Supervised*			*Statistical*	
		MoTM	**MOMENT**	**TimeFlow**	**BRITS**	**SAITS**	**Linear**	**Repeat**
MoTM In-Domain								
Electricity	*Point 1*	**0.196**	0.861	0.274	0.324	0.211	0.306	0.334
	Point 2	**0.229**	0.863	0.322	0.465	0.258	0.435	0.357
	Block 1	**0.257**	0.478	0.291	0.522	0.296	1.025	0.312
	Block 2	**0.259**	0.538	0.298	0.465	0.292	1.027	0.313
Solar	*Point 1*	0.083	0.857	0.085	0.072	0.077	**0.036**	0.265
	Point 2	0.092	0.858	0.097	0.083	0.130	**0.055**	0.281
	Block 1	0.253	0.755	0.257	0.308	0.361	0.883	**0.244**
	Block 2	0.256	0.781	0.258	0.314	0.355	0.889	**0.244**
Spanish W-T	*Point 1*	0.214	0.835	0.283	0.373	0.205	**0.169**	0.520
	Point 2	**0.253**	0.838	0.309	0.473	0.295	0.277	0.585
	Block 1	0.402	0.511	**0.391**	0.685	0.444	0.889	0.484
	Block 2	0.404	0.548	**0.396**	0.689	0.451	0.898	0.470
MoTM improvement		**0.0%**	**62.8%**	**10.8%**	**30.6%**	**12.6%**	**22.5%**	**32.2%**
MoTM Out-of-Domain								
Traffic	*Point 1*	0.246	0.770	0.240	0.267	**0.201**	0.287	0.379
	Point 2	0.294	0.774	0.291	0.374	**0.241**	0.421	0.416
	Block 1	0.313	0.478	0.389	0.415	**0.227**	0.983	0.340
	Block 2	0.318	0.521	0.395	0.431	**0.231**	0.985	0.341
ETTh1	*Point 1*	0.340	0.812	0.410	0.539	0.347	**0.334**	0.594
	Point 2	**0.389**	0.814	0.482	0.633	0.421	0.426	0.635
	Block 1	**0.490**	0.633	0.553	0.723	0.535	0.845	0.558
	Block 2	**0.488**	0.664	0.557	0.730	0.536	0.834	0.559
ETTh2	*Point 1*	0.442	0.806	0.489	0.533	0.422	**0.406**	0.763
	Point 2	0.496	0.806	0.521	0.610	0.486	**0.471**	0.805
	Block 1	0.609	0.704	0.631	0.691	**0.602**	0.761	0.738
	Block 2	**0.600**	0.704	0.619	0.722	0.610	0.760	0.716
Weather	*Point 1*	0.326	0.816	0.330	0.389	0.287	**0.260**	0.739
	Point 2	0.375	0.819	0.383	0.484	0.351	**0.323**	0.803
	Block 1	**0.524**	0.640	0.627	0.689	0.584	0.621	0.677
	Block 2	**0.527**	0.669	0.633	0.691	0.582	0.620	0.682
Spanish E	*Point 1*	0.235	0.818	0.329	0.311	0.189	**0.164**	0.678
	Point 2	0.286	0.823	0.376	0.425	0.285	**0.253**	0.738
	Block 1	0.507	0.622	**0.503**	0.675	0.536	0.606	0.644
	Block 2	0.503	0.649	**0.499**	0.675	0.535	0.604	0.633
MoTM improvement		**0.0%**	**40.3%**	**10.2%**	**24.0%**	**-5.7%**	**13.2%**	**31.1%**

performs poorly across all datasets, highlighting the limitations of foundation models without adaptation. (ii) MoTM achieves substantial gains over the supervised TimeFlow model on both in-domain (ID) and out-of-domain (OOD) datasets, demonstrating the benefits of multi-source training and the effectiveness of ridge-based adaptation. (iii) Compared to other supervised deep learning methods, MoTM delivers competitive or superior performance — particularly against BRITS, reducing its error by 30.6% on ID and 24.0% on OOD datasets. In addition, MoTM slightly outperforms SAITS in the ID setting (12.6% improvement on average), while remaining competitive in the OOD setting, even surpassing SAITS on datasets like `ETTh1` or `Weather` in the *Block* scenarios. (iv) MoTM also clearly outperforms statistical baselines such as linear interpolation and value repetition, with an average relative improvement of 22.5% and 32.2% in-domain, and 13.2% and 31.1% out-of-distribution.

Overall, the results highlight the benefits of continuous modeling for time series imputation, as well as the effectiveness of MoTM's simple adaptation strategy to generalize to new domains. Qualitatively, Fig. 5 illustrates a block imputation example from the OOD dataset `SpanishE`. Visually, MoTM demonstrates strong imputation capabilities on this complex sample.

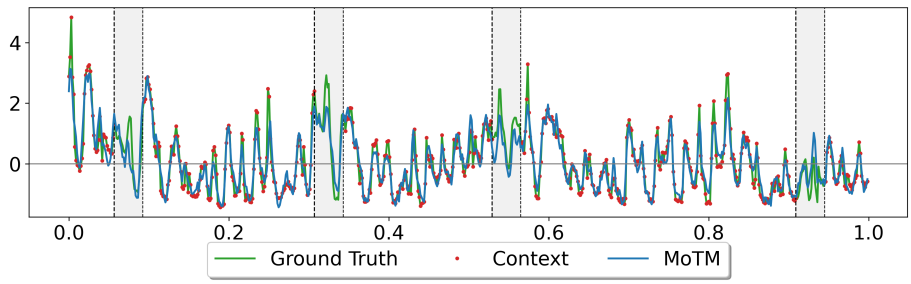
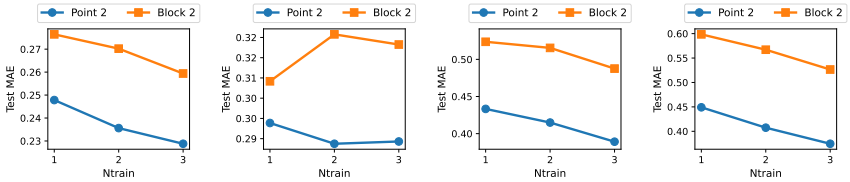


Fig. 5. *OOD `SpanishE` dataset.* MoTM imputation on four one-day missing blocks.

Ablations on the number of TimeFlow components in the MoTM basis. Can the performance of MoTM be matched by a single TimeFlow with ridge adaptation at inference? To elucidate this question, we perform an ablation on $N_{\texttt{train}}$. Fig. 6 shows the evolution of the test MAE on four datasets as the mixture grows from one component with ridge adaptation (`Electricity`) to all three components. In general, increasing $N_{\texttt{train}}$ does improve the test metrics on ID and OOD datasets, showing that MoTM leverages its multi-source pretraining. However, certain datasets and settings such as block imputation on `Traffic`, do not benefit from more base components. This calls for further improvement of the orchestration mechanism, e.g. through a better tuning of λ.

Discussion on inference time. At first glance, the inference procedure of MoTM may appear computationally expensive, due to the combination of adapting



(a) Electricity (b) Traffic (c) ETTh1 (d) Weather

Fig. 6. *Ablation: number of basis components.* Test MAE scores on one ID (`Electricity`) and three OOD (`Traffic`, `ETTh1`, `Weather`) datasets for a mixture of $N_{\text{train}} \in \{1, 2, 3\}$ TimeFlow components and ridge orchestrator. The mixture components are obtained by successively training on `Electricity`, `Solar` and `SpanishW-T`.

TimeFlow models and fitting the ridge regression. To assess its practical cost, we report in Table 6 the inference time of MoTM on the largest test dataset, `Traffic`. On a single H100 GPU, imputing 83k segments of length 672 takes approximately 61 seconds in total, corresponding to roughly 0.7 milliseconds per segment. This suggests that MoTM remains computationally efficient at inference time, even on large-scale data. For comparison, SAITS requires full retraining on this out-of-distribution dataset and takes approximately 3h16 to reach the performance reported in Table 5 on the same task.

Table 6. Computation time on the `Traffic` test dataset (NVIDIA H100 GPU). MoTM is evaluated in a zero-shot setting, while SAITS requires training.

Method	Segments	Sequence length	Compute time
MoTM (zero-shot)	83,517	672	61s
SAITS (training + inference)	83,517	672	3h16

5 Conclusion and Discussion

In this work, we introduced MoTM, a mixture-based architecture that extends the capabilities of continuous-time TimeFlow models to a zero-shot imputation setting. By aggregating specialized TimeFlow models trained on distinct distributions, MoTM effectively handles a wide range of missing data patterns and sampling rates, without the need for retraining. Our experiments confirm its strong generalization performance across both synthetic and real-world datasets, especially in out-of-distribution scenarios.

While MoTM inference is slightly slower than single-model approaches due to its mixture structure, it remains efficient relative to the substantial training time required by comparable models retrained from scratch, as shown in Table 6. Overall, the performance gains brought by MoTM are not uniform across

all ID datasets. For instance, the `Solar` dataset shows limited improvement, highlighting that the benefit of model mixing depends on the underlying data distribution. Identifying the most effective combination of models remains thus an open challenge. Moreover, the supervised baseline `SAITS` outperforms MoTM in several settings, suggesting that there is still room for improvement. Future work will explore the construction of large, unified databases enabling the training of TimeFlow models at scale for diverse distributions, and the integration of more expressive orchestration mechanisms to further enhance inference quality.

6 Acknowledgements

We would like to thank Louis Serrano for his insightful discussions about this paper. We would also like to thank Adrien Petralia and Camille Georget for their helpful feedback and proofreading assistance.

References

1. Ansari, A.F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S.S., Arango, S.P., Kapoor, S., et al.: Chronos: Learning the language of time series. Transactions on Machine Learning Research (2024)
2. Bilos, M., Rasul, K., Schneider, A., Nevmyvaka, Y., Günnemann, S.: Modeling temporal data as continuous functions with stochastic process diffusion. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) International Conference on Machine Learning, ICML. Proceedings of Machine Learning Research, vol. 202, pp. 2452–2470. PMLR (2023)
3. Brouwer, E.D., Simm, J., Arany, A., Moreau, Y.: GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In: Advances in Neural Information Processing Systems. vol. 32 (2019)
4. Cao, W., Wang, D., Li, J., Zhou, H., Li, Y., Li, L.: BRITS: Bidirectional Recurrent Imputation for Time Series. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
5. Cheng, J., Yang, C., Cai, W., Liang, Y., Wen, Q., Wu, Y.: Nuwats: a foundation model mending every incomplete time series. arXiv preprint arXiv:2405.15317 (2024)
6. Clark, J.S., Bjørnstad, O.N.: Population time series: process variability, observation errors, missing values, lags, and hidden states. Ecology **85**(11), 3140–3150 (2004)
7. Corani, G., Benavoli, A., Zaffalon, M.: Time series forecasting with gaussian processes needs priors. In: Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track. vol. 12978, pp. 103–117. Springer (2021)
8. Das, A., Kong, W., Sen, R., Zhou, Y.: A decoder-only foundation model for time-series forecasting. In: Proceedings of the 41st International Conference on Machine Learning. vol. 235, pp. 10148–10167. PMLR (2024)
9. Du, W., Côté, D., Liu, Y.: SAITS: Self-attention-based imputation for time series. Expert Systems with Applications **219**, 119619 (2023)
10. Fortuin, V., Baranchuk, D., Rätsch, G., Mandt, S.: GP-VAE: Deep Probabilistic Time Series Imputation. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. pp. 1651–1661. PMLR (2020)

11. Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D.J., Eslami, S.M.A.: Conditional neural processes. In: Proceedings of the 35th International Conference on Machine Learning, ICML. vol. 80, pp. 1690–1699. PMLR (2018)
12. Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., Dubrawski, A.: MOMENT: A family of open time-series foundation models. In: Proceedings of the 41st International Conference on Machine Learning. vol. 235, pp. 16115–16152. PMLR (2024)
13. Kim, T., Ko, W., Kim, J.: Analysis and impact evaluation of missing data imputation in day-ahead pv generation forecasting. Applied Sciences **9**(1), 204 (2019)
14. Le Naour, E., Serrano, L., Migus, L., Yin, Y., Agoua, G., Baskiotis, N., Gallinari, P., Guigue, V.: Time Series Continuous Modeling for Imputation and Forecasting with Implicit Neural Representations. Transactions on Machine Learning Research (2024)
15. Li, M., Liu, K., Guo, J., Bu, J., Wang, H., Wang, H.: Imputeinr: Time series imputation via implicit neural representations for disease diagnosis with missing data. arXiv preprint arXiv:2505.10856 (2025)
16. Liu, S., Li, X., Cong, G., Chen, Y., Jiang, Y.: Multivariate time-series imputation with disentangled temporal representations. In: The Eleventh International Conference on Learning Representations, ICLR (2023)
17. Luo, Y., Cai, X., Zhang, Y., Xu, J., Xiaojie, Y.: Multivariate time series imputation with generative adversarial networks. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
18. Luo, Y., Zhang, Y., Cai, X., Yuan, X.: E2gan: End-to-end generative adversarial network for multivariate time series imputation. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 3094–3100 (2019)
19. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press (2006)
20. Rubanova, Y., Chen, R.T.Q., Duvenaud, D.: Latent odes for irregularly-sampled time series. CoRR **abs/1907.03907** (2019)
21. Schulz, M., Stattegger, K.: Spectrum: Spectral analysis of unevenly spaced paleoclimatic time series. Computers & Geosciences **23**(9), 929–945 (1997)
22. Shukla, S.N., Marlin, B.M.: Multi-Time Attention Networks for Irregularly Sampled Time Series. In: 9th International Conference on Learning Representations, ICLR 2021 (2021)
23. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020)
24. Tashiro, Y., Song, J., Song, Y., Ermon, S.: CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In: Advances in Neural Information Processing Systems. vol. 34 (2021)
25. Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., Sahoo, D.: Unified Training of Universal Time Series Forecasting Transformers. In: Proceedings of the 41st International Conference on Machine Learning. vol. 235, pp. 53140–53164. PMLR (2024)
26. Woo, G., Liu, C., Sahoo, D., Kumar, A., Hoi, S.: Learning deep time-index models for time series forecasting. In: International Conference on Machine Learning. pp. 37217–37237. PMLR (2023)
27. Zintgraf, L., Shiarli, K., Kurin, V., Hofmann, K., Whiteson, S.: Fast context adaptation via meta-learning. In: International Conference on Machine Learning. pp. 7693–7702. PMLR (2019)