# DIM-SUM: Dynamic IMputation for Smart Utility Management

Ryan Hildebrant
University of California, Irvine
rhildebr@uci.edu

Rahul Bhope
University of California, Irvine
rbhope@uci.edu

Sharad Mehrotra
University of California, Irvine
sharad@ics.uci.edu

Christopher Tull
California Data Collaborative
chris@thecadc.org

Nalini Venkatasubramanian
University of California, Irvine
nalini@uci.edu

## ABSTRACT

Time series imputation models have traditionally been developed using complete datasets with artificial masking patterns to simulate missing values. However, in real-world infrastructure monitoring, practitioners often encounter datasets where large amounts of data are missing and follow complex, heterogeneous patterns. We introduce DIM-SUM, a preprocessing framework for training robust imputation models that bridges the gap between artificially masked training data and real missing patterns. DIM-SUM combines pattern clustering and adaptive masking strategies with theoretical learning guarantees to handle diverse missing patterns actually observed in the data. Through extensive experiments on over 2 billion readings from California water districts, electricity datasets, and benchmarks, we demonstrate that DIM-SUM outperforms traditional methods by reaching similar accuracy with lower processing time and significantly less training data. When compared against a large pre-trained model, DIM-SUM averages 2x higher accuracy with significantly less inference time.

**PVLDB Reference Format:**
Ryan Hildebrant, Rahul Bhope, Sharad Mehrotra, Christopher Tull, and Nalini Venkatasubramanian. DIM-SUM: Dynamic IMputation for Smart Utility Management. PVLDB, 18(11): 4451 - 4464, 2025.
doi:10.14778/3749646.3749705

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/Ryanhilde/dim_sum_vldb.

## 1 Introduction

This paper considers the challenge of learning imputation models for large univariate time series datasets in the wild, where the input dataset may contain a significant number of missing values. Our motivation stems from the domain of civil infrastructure monitoring (e.g. water utilities, energy metering, intelligent transportation) where time series data is collected by multiple organizations over extended periods at fixed intervals. Such datasets frequently contain substantial missing values from factors such as sensor failures, communication interruptions, system maintenance, and software issues. For example, in our analysis of water data in California, we observed a water district with 47% of the expected values missing.

Large-scale missing data issues arise across various infrastructure datasets. Electric grid monitoring systems frequently experience significant gaps in data collection, particularly during periods of high load when data is most valuable. Studies by [23, 38] found that 47.14% of individual readings can be missing, with 51.26% of readings being up to a half a day of continuous missing values. Similarly, loop sensor from the CA Dept. of Transportation's PEMS system [9] exhibits wide variations in missing data, with reporting areas experiencing between 38% to 76% missing values.

Beyond the sheer volume of missing data, infrastructure datasets often exhibit distinct missing patterns that correlate with specific operational factors. For instance, water meters from the same manufacturer may exhibit synchronized data gaps due to shared maintenance schedules or minimum required signal strengths for data transmission. In the electric domain, [23] reported that understanding the relationship between voltage fluctuations and downstream load consumption in power grids improves the characterization and imputation of missing data. Given the level of missing values and the domain knowledge required, applying plug-and-play imputation techniques across infrastructure domains is challenging.

Traditional approaches to learning imputation models begin by dividing time series data into fixed-length windows (e.g., hourly periods) to capture temporal patterns and dependencies. A window is considered complete if it contains all expected measurements within its interval [53]. For example, if sensors report readings every 15 seconds, a complete 1-minute window would contain four measurements at their expected timestamp intervals. These complete windows serve as training data, where values are artificially masked (removed), and models are trained to reconstruct these masked values. However, in the wild it can be challenging to find a sufficiently large clean dataset with complete windows for training.

A comprehensive survey of time series imputation models, including various machine learning and deep learning approaches, can be found in [18, 43]. Recent machine learning advances have demonstrated significant improvements through several architectures: SAITS [14] employs self-attention mechanisms for robust imputation; non-stationary transformers [29] are designed to handle varying temporal dependencies; MRNN [54] implements multidirectional recurrent neural networks for complex temporal relationships in streaming data with missing values; StemGNN [6] utilizes a graph neural network with a spectral GNN for capturing inter-series correlations and temporal dependencies; and CSDI [42] employs a conditional score-based diffusion model for imputing missing values by learning to reverse a noising process.

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749705

While these sophisticated architectures offer valuable insights into handling missing data, they often treat missing values as random masks or assume patterns drawn from predefined distributions, rather than as complex, domain-influenced structures. While this strategy has been effective in domains such as air quality, healthcare, and activity recognition [54], it presents challenges in large-scale infrastructure settings, where there is large datasets with significant amounts of missing values. These settings are characterized by numerous missing patterns that, despite their complexity, are recorded and reported at fixed intervals. Thus, an underlying assumption in our work is that we operate with a complete dataset. This means we presuppose a fixed data interval and an expected quantity of data points within that interval.

Several innovations have explicitly addressed how to represent large and diverse patterns of missing data to feed into an imputation model. A significant advancement is DAGAN [28], which utilizes two Generative Adversarial Networks (GANs) to learn and replicate missing patterns from real-world datasets. By training on actual production data rather than synthetic patterns, DAGAN aims to model the complexity of real-world missingness. It introduces a method of "projecting" test set values with missing patterns onto the training set to create pattern-specific masks for model training. Another strategy involves consolidating multiple datasets into a single, large pretrained model capable of processing substantial data and adapting to various missing data patterns, such as Amazon's Chronos [1]. Both of these techniques fundamentally assume that the entire dataset is known upfront. This includes both observable data and what might be termed "non-observable patterns"—essentially, the awareness that an expected value is missing, thereby reducing the anticipated amount of data. Methodologies designed to leverage real patterns of missing data inherently rely on this comprehensive, upfront knowledge of the dataset.

Hence, our goal is to explore how imputation models can be aware of real missing patterns, and "bake" them into the training process. Previous work has focused on explicitly defined patterns, single sources of missing data, or are assumed to be completely known during training time (e.g., from production datasets [28]). In contrast, infrastructure time series settings often have implicit missing patterns that can be numerous and often cluster based on characteristics such as meter type and land-use, e.g., single-family homes exhibit different usage and missing patterns than industrial factories or multi-family homes.

To address this challenge, we develop **DIM-SUM**: a generic framework for baking missing patterns into time series imputation models. DIM-SUM introduces a model-agnostic framework that enhances existing imputation methods by systematically incorporating missing patterns discovered through data analysis. Rather than developing new architectures or merging existing approaches, DIM-SUM provides a methodology that allows any imputation model to learn from and adapt to varying patterns of missing data. This enables practitioners to select models based on their specific data needs while maintaining robust performance across different missing scenarios and significantly reducing the amount of data that needs to be used for training. **Our contributions are as follows:**

- **We propose a technique to preprocess imputation models for large time series datasets using limited training data**. DIM-SUM provides a framework that can be applied with any existing imputation model (Section 3).
- **We introduce a methodology for baking data from clean and dirty sources** and projecting them into training data with minimal additional noise (Sections 4 & 5).
- **We provide probabilistic bounds for the quality of trained imputation models given diverse data sources** and unknown patterns of missing data (Section 5.3).
- **We evaluate our framework on six large-scale datasets from water, electric, and weather domains**, comparing DIM-SUM against several baselines: (i) a direct model approach using fixed missing patterns, (ii) GAN-based methods that simulate various missing patterns, and (iii) pretrained large language models (Section 6).

## 2 Related Work

The study of missing data patterns is well-established [12, 20, 26, 31], with statistics categorizing missing data as Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR) to describe the relationship between missingness and data values [27, 37]. In infrastructure time series, instances of missing values are typically known due to the data's periodic nature [37]. While traditional imputation methods often assume MCAR (where missingness is independent of data values), infrastructure settings frequently exhibit complex MNAR patterns (e.g., sensor failures due to extreme conditions, where missingness relates to unobserved values) or MAR patterns (e.g., scheduled maintenance, where missingness depends on observed variables).

Addressing these challenges, traditional database systems have adapted and scaled statistical methods for large-scale time series processing [18, 19, 21, 30, 35, 41]. Early systems integrated imputation directly into query processing, such as ImputeDB [5], while later frameworks like HoloClean [34] introduced scalable probabilistic data repair. These evolved into distributed systems like DAME [49] and EDIT [31], which implement variants of classical imputation methods while maintaining statistical guarantees [15, 25]. Furthermore, approaches like GRAIL [53] have focused on efficient time-series representation learning and processing at scale, informing scalable imputation strategies.

Deep learning has significantly advanced imputation by capturing complex temporal dependencies [52]. Recurrent Neural Networks (RNNs) like BRITS [7], GRUD [8], and MRNN [54] utilized bidirectional structures and mechanisms for variable-length or streaming data. Transformer architectures, including SAITS [14], non-stationary transformers [29], ETSformer [50], and DAMR [36], have effectively modeled long dependencies using self-attention [56]. Other innovations include ImDiffusion [10] using diffusion models, CSDI [42] employing conditional score-based diffusion, Series2Graph [4] combining imputation with graph representations, StemGNN [6] leveraging spectral graph neural networks, uncertainty-driven networks [16] aiming for efficiency, and approaches like DeepMVI [2] focusing on multidimensional time series.

DAGAN [28] made strides in incorporating realistic missing patterns but has limitations for infrastructure data. It learns mappings

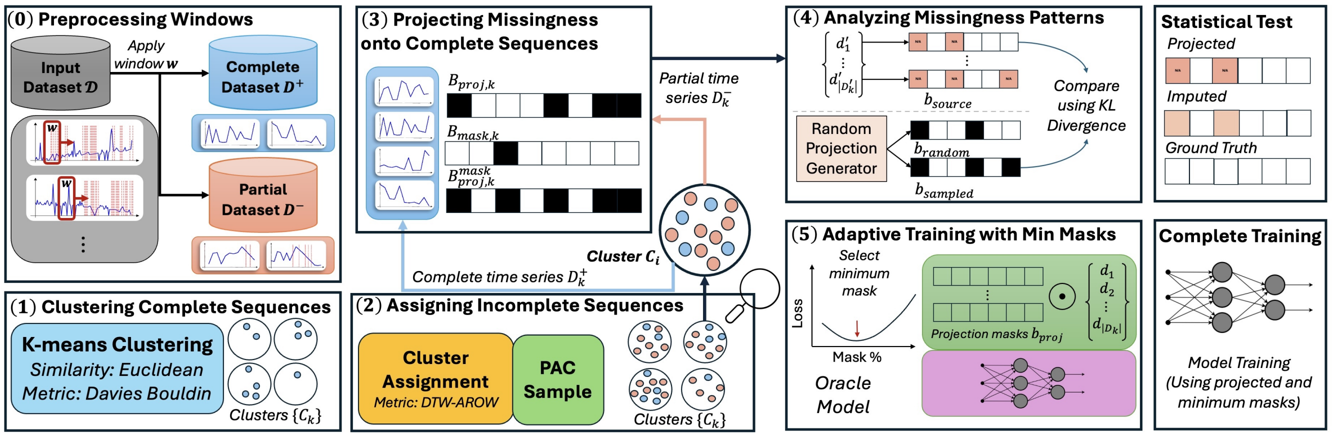

**Figure 1: Overview of the DIM-SUM Preprocessing (0, 1, 2) and Training (3, 4, 5) Framework for Time Series Imputation**

from a single clean dataset to multiple missing patterns, necessitating separate models for each source and struggling with the scale of millions or billions of readings. This approach is computationally challenging for infrastructure settings that require handling multiple source patterns mapping to multiple target patterns within a unified, efficient framework. Furthermore, DAGAN's GAN-based architecture lacks theoretical guarantees regarding the fidelity of learned missing patterns or their reproduction during training.

## 3 The DIM-SUM Approach and Architecture

Figure 1 illustrates the various steps of DIM-SUM and how the components work together to create a robust imputation framework. Algorithm 1 describes the data preparation and clustering steps (0, 1 and 2) in DIM-SUM, while Algorithm 2 specifies the model training steps (3, 4, and 5) in the DIM-SUM learning technique. A more detailed overview of 1 can be found in Section 4 and a discussion of the training process is found in 5.

**0. Windowing Function:** DIM-SUM begins by ingesting a collection of univariate time series $D = \{x_1, \ldots, x_N\}$, where each sequence $x_i \in \mathbb{R}^T$ consists of $T$ time steps that are partitioned into smaller, fixed-length windows (Algorithm 1, lines 2-5). This step standardizes sequence lengths, accommodates varying sampling rates, and ensures that imputation models operate on locally consistent patterns. Specifically, we apply a tumbling window function of size $w$, resulting in each sequence $x_i$ being divided into $\frac{T}{w}$ non-overlapping segments $x_i^{(j)}$. If a sequence contains 1,000 readings and $w = 100$, it is split into 10 partitions. Each window retains an associated mask $b_i^{(j)} \in \{0,1\}^w$, where 1 indicates an observed value and 0 denotes a missing value. After windowing, we construct two subsets: $D^+$, containing fully observed windows, and $D^-$, containing windows with missing values, where $|D^+| \ll |D^-|$.

**1. Clustering Complete Sequences.** To introduce structure in the training data, we first cluster fully observed sequences in $D^+$ (Algorithm 1, lines 6-7). Since time series data can exhibit highly variable patterns, direct imputation without accounting for inherent structure may yield poor results. We apply mini-batch $K$-means clustering to identify dominant patterns and select the number of clusters $K$ using the Davies-Bouldin index [11], which is determined by passing $D^+$ to $DBIFindOptimalK$. The clustering process aims to group similar temporal patterns, providing a structure for imputing missing values in $D^-$.

**2. Incomplete Cluster Assignment.** Once the complete sequences have been clustered, we assign the incomplete sequences in $D^-$ to the closest cluster. A naive approach would involve simple distance-based matching, but this can be unreliable when dealing with sequences that contain missing values. Instead, we employ DTW-AROW $\delta\left(x_i^{(j)}, x_p^{(q)}\right)$, a dynamic time warping (DTW)-based method that preserves temporal alignment while handling missing values effectively [39, 55]. For each incomplete sequence in $D^-$, we find the closest fit to the existing complete cluster centroids (Algorithm 1, lines 8-9). This approach ensures that incomplete sequences are mapped to clusters that best reflect their structure. To improve assignment stability, we apply a PAC-bound based sampling heuristic [46], which prevents clusters from becoming unbalanced by distributing sequences more uniformly, which we describe in detail in Section 5.2.

**3. Projecting Missing Patterns onto Complete Sequences.** To ensure that imputation models are trained in conditions that reflect real-world missing patterns, we generate realistic incomplete sequences by projecting observed missing patterns onto fully observed data. For each cluster $k$, given an incomplete sequence $x_i^{(j)} \in D_k^-$, we extract the missing patterns $b_i^{(j)} \in \{0,1\}^w$. These binary sequences captures the distribution of missing values in the time series (Algorithm 2, lines 1-2). We then apply this pattern to a randomly selected complete sequence within the same cluster $x_p^{(q)} \in D_k^+$, effectively masking values where elements of $b_i^{(j)}$ is 0. This process generates a new dataset of masked sequences $D_{p,k}$ that retain the structural properties of complete data while incorporating missing values from that cluster's distribution. Each masked sequence $b_{src} \in X_k^{proj}$ is created through the projection operation $\pi$ applied to the sequence pair (Algorithm 2, lines 3-7).

**4. Analyzing missing patterns** Once the projected datasets $X_k^{proj}$ have been constructed for each cluster $D_k$, we analyze whether the partial sequences have similarities between the centroids of

**Algorithm 1:** DIM-SUM Preprocessing & Clustering

```
Input: Dataset $D$, window size $w$, cluster range $(K_{\min}, K_{\max})$, PAC
       bound $\tau$
Output: Clusters $\{D_k\}$ with assigned sequences
1  $D^+ \leftarrow \emptyset, D^- \leftarrow \emptyset$;
   // Step 0: Window partitioning
2  for $x_i \in D$ do
3      $\{x_i^{(1)}, \ldots, x_i^{(\lfloor T/w \rfloor)}\} \leftarrow$ WindowPartition$(x_i, w)$
4      for each window $x_i^{(j)}$ do
5          if IsComplete$(x_i^{(j)})$ then $D^+ \leftarrow D^+ \cup \{x_i^{(j)}\}$ else
               $D^- \leftarrow D^- \cup \{x_i^{(j)}\}$ ;
   // Step 1: Complete sequence clustering
6  $K \leftarrow$ DBIFindOptimalK$(D^+, K_{\min}, K_{\max})$
7  $\{D_k\} \leftarrow$ ClusterSequences$(D^+, K)$
   // Step 2: Assign incomplete sequences
8  for $D_k^- \in D^-$ do
9      $D_k^- \leftarrow IncompleteClusterAssignment(D^-, C_k, \tau)$
10 return $\{D_k\}$
```

**Algorithm 2:** DIM-SUM Training

```
Input: Clusters $\{D_k\}$, assignments $\{D_k^-\}$, model $\mathcal{M}$
Output: Trained models $\{\mathcal{M}_k\}$
1  for $k \leftarrow 1$ to $K$ do
       // Step 3: Create projections
2      for $x_i^{(j)} \in D_k^-$ do
3          $b_i^{(j)} \leftarrow$ GetMask$(x_i^{(j)})$;
4          $x_p^{(q)} \leftarrow$ SampleFrom$(D_k^+)$;
5          $b_{proj} \leftarrow \pi\left(x_i^{(j)}, x_p^{(q)}\right)$;
6          $X_k^{proj} \leftarrow X_k^{proj} \cup \{b_{proj}\}$
       // Step 4: Analyze structure
7      $n_p \leftarrow 0$;
8      for $b_{src} \in X_k^{proj}$ do
9          $b_{samp} \leftarrow$ SamplePattern$(D_k^-)$;
10         $b_{rand} \leftarrow$ RandomPattern$()$;
11         if $D_{KL}(b_{src} \parallel b_{samp}) < D_{KL}(b_{src} \parallel b_{rand})$ then
               $n_p \leftarrow n_p + 1$ ;
       // Step 5: Adaptive training
12     $MinMaskSearch(m_{min}, \alpha, \omega, T)$
13 return $\{\mathcal{M}_k\}$
```

each cluster. Within a cluster, we compare each observed missing pattern $b_{src} \in X_k^{proj}$ against both a sampled pattern $b_{samp}$ from another sequence in $D_k^-$ and a randomly generated missing pattern $b_{rand}$. If the two samples from the cluster exhibit a structured relationship with the underlying behavioral pattern, imputation should account for these dependencies rather than treating missing values as independent noise (Algorithm 2, lines 9-12).

We measure the structured relationship as the similarity between observed and reference missing patterns using the KL divergence metric [22]: $D_{KL}(b_{src} \parallel b_{samp})$ and $D_{KL}(b_{src} \parallel b_{rand})$ If a majority of the missing patterns in $X_k^{proj}$ demonstrate non-random structure, the cluster exhibits similar missing data behavior. This informs the subsequent training of cluster-specific models $\mathcal{M}_k$.

**5. Adaptive Training with Minimum Masks.** Finally, DIM-SUM employs an iterative training process that determines the minimum masking necessary for robust imputation. For each cluster $k$, we begin with an initial mask size $m_0$ and progressively increase the fraction of missing values introduced during training. With our projected patterns $b_{src} \in X_k^{proj}$ already capturing real distributions, we create two additional representations: $X_k^{mask} = D_k^+ \odot b_{src}$ applying only artificial masks to complete data, and $X_k^{proj,mask} = D_k^+ \odot b_{src}$ applying artificial masks to windows of data. At each iteration, we train two models: an oracle model using only $X_k^{mask}$, and a projection model using $X_k^{proj,mask}$. If the projection model's performance remains within two standard deviations of the oracle model's performance while handling progressively larger missing regions, training continues; otherwise, the process is halted.

The details of this minimum mask selection process are detailed in Algorithm 4 in Section 5.2. Once a minimum mask is selected by sampling the training data (determined by the PAC bound, which we describe later in Section 5.3), we train the imputation model across the cluster-specific data and apply inference. We validate the performance of the model on the projected data using a statistical test to measure how well a model reconstructs missing values.

## 4 Baking Missing Patterns via Clustering

Through our empirical studies with various smart utility datasets, we have found that the large scale and intricate nature of this data frequently results in these state-of-the-art imputation models typically produce unpredictable results. This is particularly evident in the large water dataset comprising of 1.5 billion readings, where approximately 30% of values are missing. When applying the Non-stationary Transformer model [29] directly to the clean portions of this dataset and injecting 10% of MCAR missing values, we observed an MSE of 1.4876. However, when we apply DIM-SUM to same transformer-based architecture, the MSE dramatically improved to 0.7770 (47.8% error reduction).

Our performance gain comes from the recognition that utility consumption data contains repetitive behavioral patterns. Rather than treating the dataset as a monolithic entity, partitioning it into clusters of similar behavior allows us to train specialized smaller models that implicitly learn patterns of missing sequences on complete values that additionally have similarity based on what we can observe. By identifying and grouping similar temporal behaviors, these pattern-specific models better capture the distribution characteristics of each cluster, leading to more accurate imputations even with substantially fewer computational resources than using one large model on the entire dataset.

We identify distinct temporal patterns within complete windows $D^+$, where each window $x_i^{(j)}$ represents $w$ consecutive measurements. This requires balancing pattern specificity with having enough training samples per cluster. Too many clusters can create sparse sample distribution, while too few fail to capture temporal pattern diversity. We directly apply mini-batch $K$-means clustering to all windows, which reduces computational cost through batch processing while preserving clustering quality on large-scale time series data [40]. Each window is first normalized using z-score normalization to focus on pattern shapes rather than absolute values.

Normalization ensures that clustering captures similarities in the temporal dynamics (the shape of the time series) rather than being dominated by magnitude differences. This helps identify meaningful patterns across sensors or devices with different baseline readings but similar behavioral patterns, regardless of scale. Such an approach is quite effective for the cyclic nature of utility usage, i.e., single family homes may take showers in the morning, but amount of water consumed may differ. To determine the optimal number of clusters, we perform a binary search over the range $[K_{\min}, K_{\max}]$, using the Davies-Bouldin index [11] as our optimization criterion, lines 6-7 in Algorithm 1. For each candidate $k$, we compute:

$$\mathrm{DB}(k) = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i+\sigma_j}{d_{ij}}\right) \tag{1}$$

where $\sigma_i$ represents the average distance between points in cluster $i$ and their centroid, and $d_{ij}$ is the distance between the centroids of clusters $i$ and $j$. The Davies-Bouldin index measures the average similarity between each cluster and its most similar cluster, where similarity is defined as the ratio of within-cluster distances to between-cluster distances. Lower values indicate better clustering, with more compact and well-separated clusters.

We employ a binary search rather than an exhaustive search over all possible $k$ values to efficiently find the optimal number of clusters. This approach reduces the computational burden while still providing a robust estimate of the optimal $k$. Once the optimal number of clusters is determined in our sample space, we obtain the final clustering by applying ClusterSequences$(D^+, K)$. The optimal $K$ search has a time complexity of $O(\log(R_K)\cdot(tK|D^+|w + K^2w))$, for search range $R_K$, $t$ K-means iterations, $K$ clusters, $|D^+|$ complete windows, and window size $w$.

## 4.1 Assigning Incomplete Sequences

Training imputation models presents a fundamental question when working with limited complete data: *how do we balance the use of complete sequences and utilize the observable data in the incomplete sequences for training?* This challenge is amplified when the data is partitioned into clusters to capture distinct temporal patterns, as it further reduces the available training samples per model. To address this limitation while maintaining the benefits of cluster-specific models, we develop a sampling strategy that augments each cluster's training data with incomplete sequences. While we cannot directly validate imputation quality on real missing values, we can project observed missing patterns onto complete sequences where reconstruction accuracy can be measured. Consider a complete time series sequence and a projected version of that time series with missing values written over random indices:

| | |
|---|---|
| Complete: | [10, 8, 7, 5, 6, 8, 12, 15, 14, 12, 10, 9, 8, 7, 8, 10, 13, 15, 14] |
| Projection: | [10, $m$, 7, $m$, 6, 8, $m$, $m$, 14, 12, $m$, 9, 8, 7, $m$, $m$, 10, 13, $m$, 14] |

The projected sequence is fed into an model, which will output its best estimate of the original complete window. Although this is a standard practice, the projection does not account for the shared patterns of the observable values between complete and incomplete sequences. These subsequences of values also provide important information for fitting the right sequence to an existing cluster by exploiting the distributional closeness of observable values.

For assigning incomplete sequences to a cluster, we use Dynamic Time Warping with Aligned Resolutions of Warping (DTW-AROW [55]), which handles sequences with missing values while preserving temporal relationships with an extended cost function:

$$\delta(x_i, x_j) = \begin{cases} 0, & \text{if } x_i = \text{NaN or } x_j = \text{NaN} \\ (x_i - x_j)^2, & \text{otherwise} \end{cases}$$

DTW-AROW enforces synchronized advancement through missing regions and applies a correction factor, which is calculated as the ratio of observable values to all values in both sequences. This prevents artificial alignments while ensuring fair comparisons between sequences with different proportions of missing data. The computational complexity of comparing each incomplete sequence with all centroids would be $O(|D^-|\cdot|D^+|/C)$, where $C$ is the number of sequences per centroid. Given that $|D^+| \ll |D^-|$, we leverage sampling theory to reduce this to $O(n\cdot k)$, where $n$ is the number of sampled incomplete sequences and $k$ is the number of clusters.

To determine the required number of incomplete sequences, we leverage PAC learning theory [46], which is developed formally in Section 5.3. For each cluster, we compute an observable value ratio $\gamma = (1-\alpha)(1-\mathbb{M})$, where $\alpha$ is the ratio of missing values that are currently present and $\mathbb{M}$ is the masking ratio used during training, as seen in Algorithm 3 (lines 8-9, Algorithm 1).

The observable value ratio tells us how many true values the model receives during training, which we form our bound on. The goal of the cluster assignment algorithm is to sample enough missing data into each cluster, such that there is a strong trend of similar patterns to project. If there is not a strong trend, it indicates to us that the patterns are unlikely to be significant across the entire dataset. Furthermore, by sampling enough observable data to meet a PAC bound and assuming our data is independent and identically distributed (I.I.D), we can approximately ensure the expected performance achieved during training on the cluster-specific model translate to the test data that fits the properties of each cluster.

The algorithm initially calculates the PAC bound assuming $\alpha$ incomplete data. Once we sample this initial calculated amount, we examine the actual projected missing patterns, recalculate the bound, and add more samples if needed. This dynamic adjustment allows us to determine the number of samples needed to satisfy our learning guarantees more accurately.

The heuristic approach to cluster assignment first attempts to place an incomplete sequence in the closest matching cluster based on DTW-AROW distance. However, if this cluster has reached its capacity, we assign the sequence to the next closest available cluster. This ensures a more even distribution of incomplete sequences across clusters while still prioritizing similarity. The motivation for assigning to the next closest cluster is that it allows us to quickly sample and fill clusters with strong correlations to meet the bound. This helps reduce the amount of training samples we need to iterate through and fills less frequent clusters faster. In later steps, we show that this heuristic approach works well for identifying strong clusters and eliminating clusters which have no dominant pattern trends. Next, we can now prepare the data for training an

**Algorithm 3:** Incomplete Sequence Cluster Assignment

```
Input: Dataset $D^-$, clusters $\{C_k\}$, PAC threshold $\tau$,
       MaxClusterSize, $\alpha$, $\mathbb{M}$
Output: Augmented cluster assignments
for $x_i^{(j)} \in D^-$ do
    $\gamma_i \leftarrow (1-\alpha)(1-\mathbb{M})$;
    foreach $C_k$ do
        $d_k \leftarrow \delta\left(x_i^{(j)}, C_k\right)$;
    $S_c \leftarrow$ SortByDistance($d_k$);
    for $c \in S_c$ do
        if $Observable_c < \tau$ and $|C_c| < MaxClusterSize$ then
            $C_c \leftarrow C_c \cup \{x_i^{(j)}\}$;
            Update $\alpha_i$ based on in $x_i^{(j)}$;
            $\gamma_i \leftarrow (1-\alpha_i)(1-\mathbb{M})$;
            Observable$_c \leftarrow$ Observable$_c + \gamma_i$;
            Update $\tau$ if necessary;
            break;
    if $x_i^{(j)}$ not assigned then
        $c^* \leftarrow \arg\min_{c \in S_c} |C_c|$;
        $C_{c^*} \leftarrow C_{c^*} \cup \{x_i^{(j)}\}$;
return $\{C_k\}$
```

imputation model that is fitted to the distributions of that particular cluster.

## 5 Training Models with Incomplete Data

The clustering of time series based on their characteristics and missing patterns raises a practical question: *How can imputation models be effectively trained using a mix of complete sequences ($D_k^+ \in D^+$) and sequences with missing values ($D_k^- \in D^-$) within each cluster?* An effective training strategy must address two objectives:

(1) Leveraging real-world missing patterns present in $D_k^-$ is useful to represent the actual scenarios that models will encounter in production environments. These patterns contain information about how and when data becomes unavailable, but provide limited data to the model.
(2) Acquiring sufficient complete data from $D_k^+$ is required for learning the underlying temporal dynamics and distributions that govern each cluster. Complete sequences provide the ground truth necessary for understanding the range of valid behaviors and relationships within the data, but may limited in representing all distributions across the dataset.

The DIM-SUM framework addresses these challenges by combining information from both $D_k^+$ and $D_k^-$. We translate the missing patterns from $D_k^-$ into binary encodings and project them onto the complete sequences in $D_k^+$. The projection of missing patterns effectively creates regions of missing values in the training data that reflect real-world scenarios, i.e., the data is "projected" away from the complete sequences before training. This simulates an environment where the model must train on sequences with missing values, forcing it to adapt to incomplete information.

During training, the projected missing values are treated as genuinely missing, requiring the model to learn how to handle missing data in the observable regions to compute the loss. This is achieved using a masking process. However, excessively masking the remaining observable values might introduce unnecessary noise into the training process, potentially harming model performance. To address this, the framework focuses on identifying a minimal effective mask—the smallest amount of additional masking necessary to ensure the model performs effectively.

Mask size is determined by comparing against an oracle model trained only on complete sequences with the same small mask (meaning all the projected values are observable during training), representing an ideal training situation. When a model trained with both projected missing values and masked values achieves performance within a reasonable threshold of this oracle, it indicates an appropriate balance between learning from missing patterns and preserving sufficient training data. We assert this balance to be the difference in loss achieved by oracle model (which only imputes the masked values) and the projection model (imputes masked and projected values) is minimal across the various mask percentages.This process takes $O((|D^-| + \sum_k N_{\text{train},k}) \cdot w)$ time, for $|D^-|$ incomplete windows, $N_{\text{train},k}$ training sequences of length $w$ per cluster.

### 5.1 Pattern Projection and Masking

For any given cluster $D_k$, we have both a complete subset $D_k^+ \subset D^+$ containing windows of full time series and a partial subset $D_k^- \subset D^-$ containing series with missing values. Our goal is to learn from both sets while ensuring we maintain theoretical guarantees about model performance. To utilize both sequences, we use a process of projection, where the missing patterns from windows in $D_k^-$ are applied to complete windows in $D_k^+$ to create training data with a ground truth. Ultimately, we "project" these values away as input to the model for training, simulating a scenario where the model is forced to train on incomplete training data, but we are able to verify it's accuracy after training since we have the ground truth.

We define a projection $\pi$ as an operation that maps a missing patterns from a sequence in $D_k^-$ onto a sequence in $D_k^+$. This operation creates masked values in the complete sequence that mirror the missing values in the partial sequence, expressed as $\pi : D_k^- \times D_k^+ \rightarrow \{0,1\}^w \times \mathbb{R}^w$, where $w$ is the window length. For each sequence $x_i^{(j)} \in D_k^-$ and a complete sequence $x_p^{(q)} \in D_k^+$, we encode its missing patterns as a binary vector $b_{src}$ which equals 1 if a value is present and 0 if it is missing. The projection then creates a binary mask and applies it to a complete sequence: $\pi\left(x_i^{(j)}, x_p^{(q)}\right)$.

To maintain the theoretical guarantees established during cluster assignment, we ensure a one-to-one mapping between partial and complete sequences. Each partial sequence $x_i^{(j)}$ is projected onto exactly one complete sequence $x_p^{(q)}$ chosen uniformly at random, creating our projected dataset $X_k^{proj} = \left\{\pi\left(x_i^{(j)}, x_p^{(q)}\right) : x_i^{(j)} \in D_k^-, x_p^{(q)} \in D_k^+\right\}$. This bijectivity preserves the PAC-bounded properties of our original cluster assignments.

After establishing these initial projections, we analyze whether the missing patterns within each cluster exhibit meaningful structure. We maintain a count $n_{\text{pattern}}$ of windows whose missing patterns are more similar to other patterns in $D_k^-$ than to randomly generated ones ($n_p$ in Algorithm 2). For each source pattern $b_{src}$, we use Kullback-Leibler (KL) divergence [22] to compare it against both a randomly sampled pattern from the same cluster $b_{samp}$ and a synthetic pattern with matching missing rate $b_{rand}$. This comparison is formalized through the following decision criterion:

$$b_{proj} = \begin{cases} b_{samp} & \text{if } D_{KL}(b_{src}|b_{samp}) < D_{KL}(b_{src}|b_{rand}) \\ b_{rand} & \text{otherwise} \end{cases} \tag{2}$$

A cluster is considered to exhibit meaningful structure when more than two-thirds ($\omega$ in Algorithm 4) of its patterns show stronger similarity to other real patterns than to random patterns. We measure this by computing a KL divergence criterion:

$$\frac{1}{|X_k|} \sum_{b_{src} \in X_k^{proj}} \sum_{b_{samp} \sim X_k^{proj}} \mathbb{I}\left(D_{KL}(b_{src} \| b_{samp}) < D_{KL}(b_{src} \| b_{rand})\right) \tag{3}$$

where $\mathbb{I}$ is the indicator function. For each pattern that satisfies this criterion, we increment $n_{\text{pattern}}$, providing a running measure of the cluster's structural significance. If the ratio exceeds $\omega$, it suggests a dominant missing pattern within the cluster. If the ratio is below $\omega$, the missing patterns are likely rare or represented elsewhere, prompting the removal of that cluster. For clusters meeting this criterion, we proceed to create distinct training representations:

$$\begin{aligned} X_k^{proj} &= D_k^+ \odot b_{proj} \\ X_k^{mask} &= D_k^+ \odot b_{mask} \\ X_k^{proj,mask} &= D_k^+ \odot (b_{proj} \wedge b_{mask}) \end{aligned} \tag{4}$$

where $\odot$ represents element-wise masking and $\wedge$ denotes the logical AND operation. Throughout this process, we retain the ground truth values for all masked positions to enable proper evaluation.

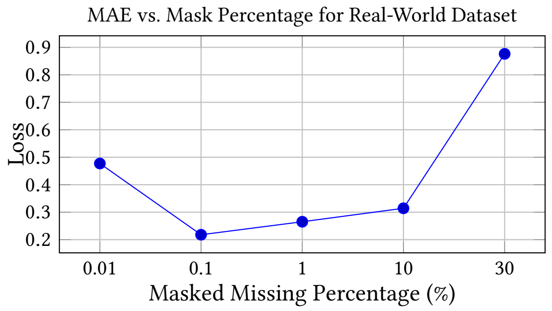

**Figure 2: An example of the U-shaped relationship between selected mask percentage and model performance with fixed level of projected missing data observed from a series of training rounds on a CA Water Dataset.**

## 5.2 Determining Minimal Effective Mask

A key aspect of our training strategy involves applying a degree of additional artificial masking after an initial projection of realistic missing patterns (derived from $D^-$) onto complete data (from $D^+$). Through our empirical findings with this two-stage approach, we observed that imputation model performance concerning validation loss often follows a U-shaped curve (as exemplified in Figure 2). Initially, moderate levels of this additional artificial masking tend to enhance model performance; we attribute this to the masking acting as a regularizer or data augmentation technique, compelling the model to learn more robust features beyond the specific projected patterns and improving generalization to diverse missingness scenarios. However, as the rate of additional artificial masking becomes excessive, performance progressively degrades.

This U-shaped performance characteristic necessitates a systematic search for a minimal effective mask ($m^*$), representing the optimal balance between induced robustness and information preservation. A key observation across the infrastructure settings we explored is that this minimum effective mask ($m^*$) is typically found at relatively small percentages; applying more masking beyond this optimal point generally increases computational costs for training without commensurate performance benefits. To determine $m^*$ for a given dataset context prepared with projected patterns, Algorithm 4 implements a logarithmic sampling strategy that concentrates evaluations in the lower range of mask percentages, where $m^*$ is often located, by starting from an initial $m_{\min} = 0.01$ and evaluating mask sizes $m_i = m_{\min}(1+r)^i$, where $r$ controls the growth rate.

For each candidate mask size $m$ two training scenarios are considered. First, an oracle model $M_{\text{oracle}}$ is trained using only $X_k^{mask}$ with a masking percentage $m_i$, representing the best achievable performance when training with complete sequences. Second, a model $M_{\text{real}}$ is trained using $X_k^{proj,mask}$, which incorporates both projected patterns and masking. Both models compute their losses $L_{\text{oracle}}(m_i)$ and $L_{\text{real}}(m_i)$ on a held-out validation set $V_k \subset D_k^+$ using the model's loss function $\ell(\cdot,\cdot)$. The absolute difference between these losses defines a gap that measures how well the model trained with projected patterns matches the performance of the oracle. To account for training variability, a convergence threshold $\omega = 2\sigma_{\text{oracle}}$ is established, where $\sigma_{\text{oracle}}$ is the standard deviation of $L_{\text{oracle}}$ in multiple training runs with random initializations. When the gap falls below $\omega$, the minimum mask size is selected.

**Algorithm 4:** Logarithmic Minimum Mask Search

```
   Input: Initial mask $m_{min} = 0.01$, growth rate $\alpha$, threshold $\omega$, int $T$
   Output: Minimum effective mask percentage $m^*$
1  gaps ← [ ]
2  for $i \leftarrow 0$ to $T-1$ do
3      $m_i \leftarrow m_{min}(1+\alpha)^i$
4      Train models on $X_{mask}k$ and $X_k^{proj,mask}$ with $m_i$
5      gap ← $|L_{\text{oracle}} - L_{\text{real}}|$
6      gaps.append(gap)
7      if $gap < \omega$ then $m^* \leftarrow m_i$ return
8  $m^* \leftarrow m_{\text{argmin(gaps)}}$
9  return $m^*$
```

## 5.3 Statistical Learning Guarantees

Section 5.2 introduced an algorithm for finding a minimal effective mask, but this empirical approach raises an important question:

*"Given that data is truly missing, how can confidence in the cluster model quality be established?"* The challenge stems from test validation: when true values are missing in production data, standard error metrics on held-out sets cannot verify model performance on the missing labels. However, by projecting patterns from $X_k^{proj}$ onto complete sequences in $D_k^+$ the model's ability to reconstruct missing values can be directly evaluated against known ground truth values. For cluster $k$, let $S_k \subseteq D_k^+$ be the set of complete sequences that have had patterns from $X_k$ projected onto them during training. For each window $w$ in these sequences, the reconstruction ability of the model can be evaluated through the test statistic:

$$T_k(w) = \begin{cases} 1, & \text{if } |\mathcal{M}_k(w) - y_i| \leq \tau \text{ for projected values } y_i \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $\mathcal{M}_k$ is the imputation model for cluster $k$, and $\tau$ is the reconstruction error tolerance. This tolerance represents the maximum acceptable difference between predicted and true values, typically set as a percentage of the data standard deviation (e.g. $\tau = 0.1\sigma$ means predictions within 10% of the standard deviation) [13]. By converting the continuous regression problem into a binary classification (success/failure) via this threshold, the test statistic enables the application of PAC learning theory [45] to bound the probability of learning an effective imputation model:

$$\begin{aligned} \alpha_k &= \frac{1}{|S_k| \cdot w} \sum_{x_i \in S_k} (1 - b_{proj}) && \text{(proportion of projected values)} \\ \beta_k &= m^* && \text{(proportion of masked values)} \\ \gamma_k &= 1 - \alpha_k - \beta_k && \text{(proportion of observable values)} \end{aligned} \tag{6}$$

Here, $\alpha_k$ measures the average proportion of projected values in the training set, $\beta_k$ represents the proportion of values masked during training using the minimal effective mask $m^*$, and $\gamma_k$ represents the remaining observable proportion. Note that $\alpha_k$ is scaled from window to window of data, while the mask $m^*$ is fixed across all windows. Let $\gamma_{\min}$ be the minimum threshold of observable values required for training. The constraint $\gamma_k \geq \gamma_{\min}$ defines a feasible region for learning [47]. The classification of whether a given configuration allows successful learning is as follows:

$$f_k(\alpha_k, \beta_k) = \begin{cases} 1 & \text{if } \mathbb{P}_{x \sim D_k^+}[T_k(x) = 1] \geq 1 - \delta \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

This creates a decision boundary in the $(\alpha_k, \beta_k)$ space that separates configurations that have reliable learning from those that do not [46]. The problem of finding this boundary can be viewed as learning a hypothesis from the class:

$$\mathcal{H}_k = \{h_\theta : h_\theta(\alpha_k, \beta_k) = \mathbb{I}(\theta_1\alpha_k + \theta_2\beta_k + \theta_3 \geq 0) \text{ for } \theta \in \mathbb{R}^3\} \tag{8}$$

where $\mathbb{I}$ is the indicator function and $\theta_3$ is a constant term. This hypothesis class is an artifact of PAC learning, corresponding to a linear classifier in the $(\alpha_k, \beta_k)$ space with a VC dimension of 3 [3, 47]. For any three points in this space, a hyperplane exists that realizes all $2^3$ possible binary classifications, i.e., if the statistical test is satisfied. However, it is impossible to shatter four points with a hyperplane in $\mathbb{R}^2$, as demonstrated by the XOR configuration [32]. Applying PAC learning theory with the VC dimension of 3 yields the following bound for the minimum number of observable sequences needed for reliable learning in cluster $k$ [3, 45]:

$$|S_k|\gamma_k \geq \frac{1}{\epsilon}\left(3 \log \frac{1}{\epsilon} + \log \frac{1}{\delta}\right) \tag{9}$$

where $\epsilon$ is the error tolerance for learning the decision boundary and $\delta$ is the failure probability. When a cluster's observable size falls below this threshold, reliable learning cannot be guaranteed with probability $1 - \delta$ [46]. These bounds establish probabilistic guarantees about the **minimum amount of observable data** needed to generalize performance of the projected values, which serve as an approximation of truly missing data.

## 6 Evaluation

This section evaluates DIM-SUM, which serves as a preprocessing framework applicable to any existing imputation model. Our evaluation specifically applies DIM-SUM to 5 distinct imputation-specific models (MRNN, SAITS, Nonstationary Transformer, StemGNN, and CSDI [6, 14, 29, 42, 54]), studying their performance across 6 datasets. These datasets span real-world and benchmark scenarios within the water, electricity, and weather infrastructure domains.

We experimentally investigate DIM-SUM's ability to enhance model accuracy by leveraging information from missing data patterns. This is examined across diverse levels and types of missingness **(Exp 1 & 2)**. For our real-world datasets (WD1, WD2, GESL-V, GESL-C, Weather) with inherent missingness, DIM-SUM's methodology involves projecting these naturally observed patterns onto complete data segments. As a benchmark, we generate and apply synthetic Missing Not at Random (MNAR) and Missing Completely at Random (MCAR) patterns for controlled experimentation. Furthermore, we compare the performance of DIM-SUM when applied to these datasets against the strategy of adapting a large pre-trained model **Exp 3**, demonstrating DIM-SUM's potential to offer competitive results with greater efficiency.

Beyond these comparisons, we evaluate DIM-SUM against another preprocessing framework, DAGAN [28], which also exploits missing data patterns prior to model training **(Exp 4)**. Lastly, we provide a comprehensive assessment of DIM-SUM's performance across all combinations of model architectures and compared techniques, focusing on critical metrics such as computational running-time and reduction in required training data **(Exp 5 & 6)**.

### 6.1 Datasets

In Table 1, we provide detailed information on the 6 datasets used in our experiments. These consist of Advanced Metering Infrastructure (AMI) data from CA cities, other infrastructure-related data, and a weather reanalysis dataset.

Table 1: Overview of Evaluation Datasets

| Infrastructure Dataset | Characteristics |
|---|---|
| **Water District 1 (WD1)**<br>Interval: 15 min<br>missing: 30% | 1.6B readings, 68,000 meters<br>Duration: 2 years<br>Feature: Flow rate |
| **Water District 2 (WD2)**<br>Interval: 15 min<br>missing: 47% | 450M readings, 18,000 meters<br>Duration: 2 years<br>Feature: Flow rate |
| **London Smart Meters (LCL)**<br>Interval: 10 min<br>missing: 10-90% | 168M readings, 360 sources<br>Duration: 1 year<br>Feature: EV charging load |
| **GESL-Current (CESL-C)**<br>Interval: 0.033s<br>missing: 10-90% | 11M readings, 13 meters<br>Duration: One month<br>Feature: Grid current |
| **GESL-Voltage (GESL-V)**<br>Interval: 0.033s<br>missing: 10-90% | 11M readings, 13 meters<br>Duration: One month<br>Feature: Grid voltage |
| **Weather**<br>Interval: 60 min<br>missing: 10-90% | 43M readings<br>Duration: One year<br>Feature: Climate |

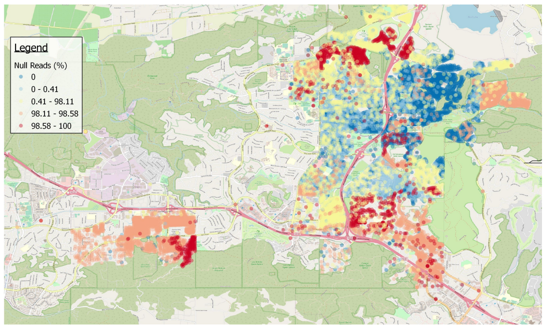

**Figure 3: A Map of WD2 Displaying the Percentage of Missing Readings from Meter Sources (Red: 98%+ missing values)**

***Water Consumption Datasets (WD1, WD2)*** We utilize data from two CA water districts collected over a two-year period. These represent real-world infrastructure data with naturally occurring missing values. WD1 contains readings from 68,000 AMI meters (1.6 billion measurements), while WD2 comprises 18,000 meters with 450 million readings. Both datasets record flow values at 15-minute intervals. We observe that 30% and 47% of values are missing in WD1 and WD2, respectively. WD1 has been anonymized and released alongside this paper in [17]. Fig. 3 illustrates the variability in the amount of missing data in various geographical regions.

***Electricity Usage Datasets (LCL, GESL-V, GESL-C)*** For electricity usage, we examine the London Smart Meters dataset from the Low Carbon London (LCL) project [44], containing 168 million readings from 360 sources at 10-minute intervals. We also use the Grid Event Signature Library (GESL) [24] with PMU measurements at 30 Hz over 3-minute intervals for current (GESL-C) and voltage magnitude (GESL-V). These datasets are typically more complete; thus, for controlled experiments, missingness is often induced.

***Weather Benchmark Dataset*** To further validate our approach on benchmarks, we utilize a weather dataset, ERA5 [33]. The dataset measures hourly estimates for a large number of atmospheric, land, and oceanic climate variables over the past 80 years. We sample 43 million points from this dataset, representing a year of data.

We process all datasets into uniform sequence lengths of 96 (corresponding to 24-hour windows at 15-minute intervals, or an 4 days for hourly intervals). For water consumption datasets with observable missing patterns, DIM-SUM directly projects these patterns onto complete data segments. For electricity and weather datasets where observations are complete, we apply MNAR and MCAR masking at specified percentages. We use a uniform 20% holdout set for evaluation for all model and dataset combinations.

## 6.2 Models and Strategies

To comprehensively evaluate DIM-SUM and understand its interaction with various imputation strategies, we first conducted a broad study across a diverse set of imputation model architectures. We examined the impact of these architectures when applied to different methodologies, including a standard direct application of each imputation model, adjusting the preprocessing to be missing pattern aware (DAGAN [28] & DIM-SUM), and utilizing a pre-trained model for inference-based imputation (Chronos [1]).

***Imputation Model Architectures Studied:*** Our initial investigation covered 8 state-of-the-art time-series imputation models, chosen to represent a range of techniques from recurrent and transformer-based to graph neural networks and diffusion models:

- **MRNN** (Multi-directional Recurrent Neural Networks for Time Series) [54]: **Recurrent architecture** that processes temporal data from multiple directions.
- **BRITS** (Bidirectional Recurrent Imputation for Time Series) [7]: Bidirectional **recurrent neural network**, with a focus on leveraging correlations in missingness.
- **SAITS** (Self-Attention-based Imputation for Time Series) [14]: A multi-staged **transformer architecture** with diagonally-masked self-attention blocks and feature reconstruction, designed specifically for imputation.
- **Nonstationary Transformer** [29]: A modified **transformer** with series stationarization and de-stationary attention mechanisms that explicitly model temporal variations and non-stationarity in data distributions.
- **Autoformer** [51]: A **transformer** model featuring an auto-correlation mechanism and series decomposition blocks for enhanced time series analysis.
- **MICN** (Multi-scale Isometric Convolutional Network) [48]: Employs multi-scale isometric **convolutions** to capture temporal patterns efficiently.
- **StemGNN** [6]: A **Graph Neural Network (GNN)** based model that captures inter-series correlations and temporal dependencies in multivariate time series through a spectral graph perspective.
- **CSDI** [42] (Conditional Score-based Diffusion Models for Imputation): Leverages score-based generative models conditioned on observed data to perform imputation through a **diffusion** process.

For the experimental evaluation, we selected five of these models that represent diverse and highly competitive architectural paradigms: MRNN, SAITS, Nonstationary Transformer, StemGNN, and CSDI. DIM-SUM's framework is model-agnostic, and its performance when integrated with these five models is compared against the other approaches detailed below.

***Strategies for Missing Pattern Integration:*** In our experiments, we evaluate performance across different overarching strategies for handling missing data for imputation:

- **Direct Model Application (Standard Approach):** This approach involves **applying each imputation model architecture directly to the dataset** (which may contain inherent or induced missing values, as specified per experiment). The model is used according to its standard implementation, relying on its own internal mechanisms or recommended setup for handling and imputing missing data, without external preprocessing frameworks. This serves as a primary point of comparison to assess the added value of preprocessing frameworks.
- **Missing Pattern Extraction Strategy (DIM-SUM):** DIM-SUM falls into this category. It first analyzes the dataset to identify and cluster data segments, then projects dominant missing data patterns onto bounded samples within each cluster for efficient and targeted model training.
- **Pre-trained Model Strategy (Chronos):** This involves leveraging large foundation models like Chronos [1], which are **pre-trained on diverse time series data from various domains** and then applied to the specific dataset. Chronos tokenizes time series and uses a language model architecture to predict missing values based on context.
- **Meta-Model Strategy (DAGAN):** This includes approaches like DAGAN [28], which use **generative models (e.g., GANs) to learn and replicate realistic missing data patterns**, or to directly impute values.

## 6.3 Experimental Results

We now present the detailed experimental results across the 6 datasets (WD1, WD2, LCL, GESL-V, GESL-C, and Weather) and evaluate how DIM-SUM compares to various model architectures and strategies for incorporating missing patterns.

*6.3.1* ***Improvement of Existing Imputation Models using DIM-SUM*** This experiment evaluates the enhancement DIM-SUM brings to the five selected imputation model architectures, focusing on imputation accuracy (MSE) at fixed levels of missingness. We explicitly compare to **direct model** strategy, where the entire dataset is used for training. Table 2 presents MSE comparisons for all five models using a fixed 50% MCAR mask and a 20% uniform holdout dataset across each of the six datasets.

As we saw previously, the loss achieved by a model can vary significantly from dataset to dataset. For example, the LCL dataset achieves near perfect loss using MRNN and SAITS, but skyrockets when the Non-stationary Transformer [29] model. Importantly, across nearly all datasets and model configurations, DIM-SUM achieve better MSE using a significantly smaller of training data, which we discuss in 6.3.4.

*6.3.2* ***DIM-SUM Performance as a Function of Missing Data*** DIM-SUM consistently delivers competitive imputation accuracy across the majority of datasets and throughout many missing pattern scenarios. This is demostrated through benchmarking direct application of models to various levels of missing data and missing patterns, as shown in Figure 5. Across each dataset, level of missing data, and type of missing data, DIM-SUM remains very competitive to direct applications, with the ability to train and produce results significantly faster, which we show in 6.3.4 and 6.3.4.

*6.3.3* ***Comparison with a Pre-trained Model*** We evaluate DIM-SUM against Chronos [1], a prominent pre-trained foundation model for time series. Figure 5 illustrates comparative MSE on the 20% holdout set for our 6 datasets. Chronos generally shows strong performance at lower missing percentages (e.g., 10-30%), where ample context is available for its predictions. However, as missingness increases, DIM-SUM tends to exhibit more robust or superior accuracy across the datasets. For example, on WD1 at 90% missing data, DIM-SUM achieves an MSE of 0.9874, whereas Chronos's MSE is visibly higher (1.6863). This is attributed to Chronos's sequential imputation, which can lead to error propagation when context is sparse. The significant computational cost of Chronos, especially its inference time, is discussed in Section 6.3.4.

*6.3.4* ***Comparison with a Meta-Model Strategy for missing pattern generation*** Next, we compare DIM-SUM with DAGAN [28], a GAN-based framework. MSE comparisons are presented in Table 3. DIM-SUM consistently demonstrates superior MSE compared to DAGAN, especially at higher missing percentages, across datasets. For instance, on WD1 with 90% missingness (Non-stationary Transformer), DIM-SUM's MSE is 1.0972, markedly better than DAGAN's 13.7267. DAGAN's tendency to overfit dominant patterns contributes to this performance gap. DIM-SUM's clustering approach, in contrast, helps preserve and learn from diverse data patterns.

***Computational Efficiency*** Computational efficiency is a significant advantage of the framework. Using DIM-SUM with the following PAC bound ($\delta = 0.1, \epsilon = 0.03$), Table 4 details the average end-to-end (E2E) computation time, directly comparing DIM-SUM with Baseline (Direct), DAGAN, and Chronos approaches at both 10% and 90% missingness levels. This allows for an assessment of how these techniques perform under varying data availability. DIM-SUM consistently demonstrates superior E2E efficiency. Note that full DAGAN preprocessing is impractical for large datasets. DIM-SUM's own complete preprocessing is far quicker (e.g., WD1: 45 seconds for DIM-SUM vs. 22 minutes for DAGAN on sampled data).

***Reduction of Training Data*** A primary benefit of DIM-SUM is its dramatic reduction in training data requirements, as detailed in Table 5. This table illustrates that DIM-SUM consistently operates with significantly fewer sequences compared to strategies like Direct Model Application or Chronos, which are trained using the entire available set of clean sequences. This efficiency also extends to comparisons with pattern-aware methods like DAGAN; even when DAGAN employs DIM-SUM's sampling, DIM-SUM's inclusion of majority cluster selection further refines data needs, resulting in smaller training sets. For instance, on the extensive

Table 2: MSE Comparison: Direct Model Application vs. DIM-SUM.

| Dataset | MRNN | | SAITS | | Nonstationary | | StemGNN | | CSDI | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Direct | DIM-SUM | Direct | DIM-SUM | Direct | DIM-SUM | Direct | DIM-SUM | Direct | DIM-SUM |
| **WD1** | 1.0887 | **0.8748** | 1.0000 | **0.9792** | 1.1491 | **1.0915** | **1.0114** | 1.2505 | 4.4240 | **3.3774** |
| **WD2** | 0.9423 | **0.8833** | **1.0027** | 1.0179 | **0.9603** | 1.0089 | 1.0798 | **1.0677** | **0.9199** | 0.9631 |
| **LCL** | 0.7190 | **0.6080** | 0.8981 | **0.6817** | 1.1910 | **1.1252** | 0.8414 | **0.8047** | 13.2458 | **6.6625** |
| **GESL-C** | 0.0079 | **0.0015** | 0.0633 | **0.0565** | 1.3024 | **1.1063** | **0.1069** | 0.1469 | 2.1340 | **0.7282** |
| **GESL-V** | 0.0007 | **0.0006** | 0.0594 | **0.0510** | 1.1122 | **1.0920** | **0.2616** | 0.9006 | 43.4276 | **26.2236** |
| **Weather** | 1.0733 | **0.8664** | 0.6229 | **0.3126** | 0.4933 | **0.3253** | 0.6135 | **0.4200** | 27.9520 | **2.6036** |

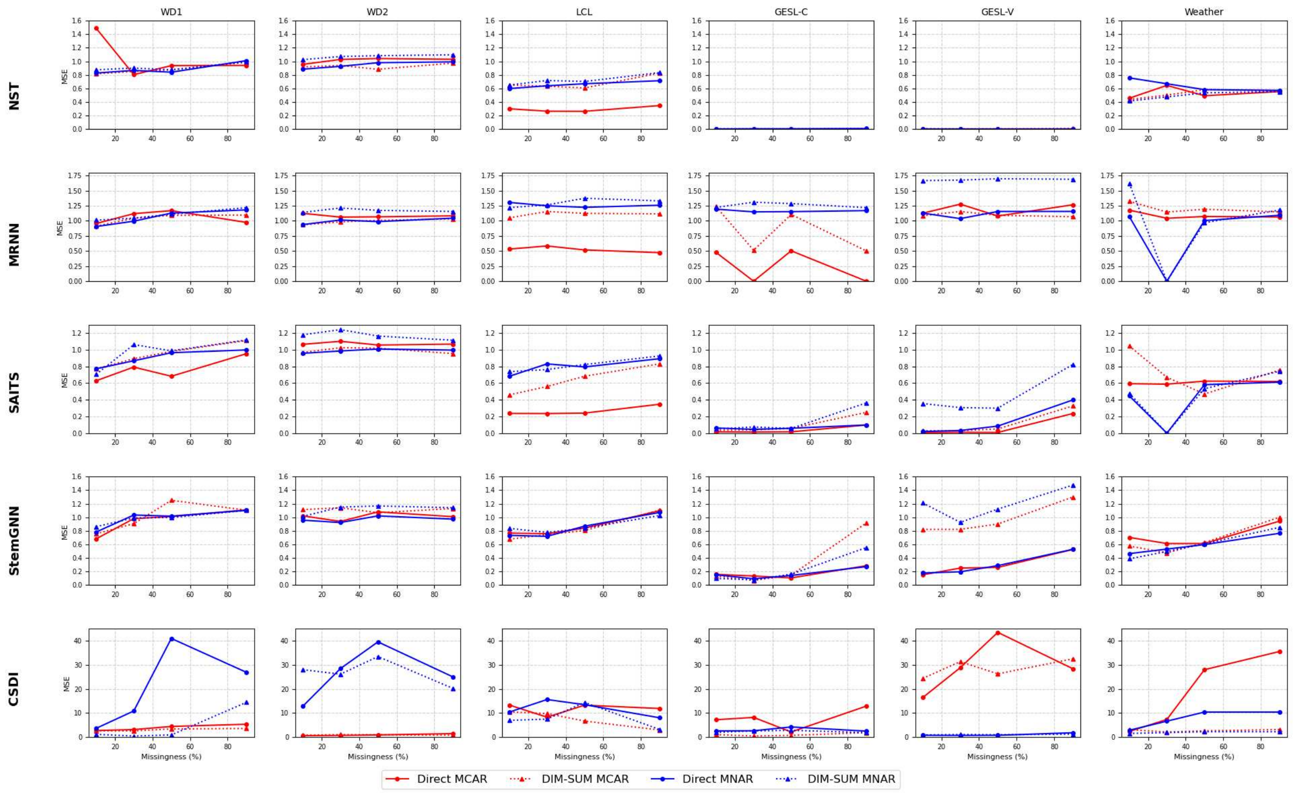

Figure 4: MSE comparison of direct models versus applying DIM-SUM across the six datasets with MCAR and MNAR

Table 3: MSE Comparison: DAGAN vs. DIM-SUM (D / DS) across Models, Datasets, and Missingness Levels (10% & 90%).

| Dataset | MRNN | | SAITS | | Nonstationary | | StemGNN | | CSDI | |
|---|---|---|---|---|---|---|---|---|---|---|
| | @10% (D / DS) | @90% (D / DS) | @10% (D / DS) | @90% (D / DS) | @10% (D / DS) | @90% (D / DS) | @10% (D / DS) | @90% (D / DS) | @10% (D / DS) | @90% (D / DS) |
| **WD1** | **0.4983** / 0.9131 | 13.7267 / **1.0972** | 6.1859 / **0.7711** | 11.1561 / **1.1104** | 4.1538 / **0.8171** | 13.3162 / **0.9874** | 6.8437 / **0.7571** | 18.9274 / **1.1046** | 24.3528 / **2.6256** | 38.4582 / **3.5908** |
| **WD2** | 1.5169 / **0.9340** | 5.7200 / **1.0330** | 1.8119 / **0.9673** | 5.8485 / **0.9536** | 1.3394 / **0.9190** | 5.7613 / **0.9754** | 1.6794 / **1.1130** | 5.8613 / **1.1297** | 5.7641 / **0.7817** | 8.2235 / **0.7962** |
| **LCL** | 1.3765 / **1.0504** | 1.3756 / **1.1158** | 0.7852 / **0.4592** | 1.0902 / **0.8304** | 0.5174 / **0.4592** | 0.9865 / **0.8304** | **0.1020** / 0.6773 | **0.5019** / 1.1064 | **0.6576** / 10.4309 | **0.9625** / 2.9519 |
| **GESL-V** | 1.1524 / **1.0862** | 2.5685 / **1.0692** | 0.0314 / **0.0293** | 3.1598 / **0.3266** | 0.0229 / **0.0010** | 0.0315 / **0.0086** | **0.0458** / 0.8221 | 3.2871 / **1.2994** | **0.0453** / 24.3325 | **0.6210** / 32.4632 |
| **GESL-C** | 1.2613 / **1.2358** | 6.4517 / **1.1706** | 0.0343 / **0.0335** | 7.9391 / **0.2492** | 0.1231 / **0.0013** | 0.3673 / **0.0022** | 0.1640 / **0.1354** | 8.0370 / **0.9166** | **0.0931** / 1.0443 | **0.5011** / 1.8508 |
| **Weather** | **1.0396** / 1.3247 | 11.5739 / **1.1486** | **0.1904** / 1.0443 | 13.3771 / **0.7553** | **0.3192** / 0.4389 | 4.0234 / **0.5671** | **0.2401** / 0.5769 | 13.4723 / **1.0015** | **0.1904** / 347.7666 | **0.5850** / 43.5254 |

WD1 dataset, DIM-SUM utilizes merely a fraction of the sequences needed by these alternative strategies, underscoring its superior data utilization efficiency and exceptional value, particularly in resource-constrained or large-scale deployment scenarios.

## 7 Conclusion

DIM-SUM presents an innovative and scalable preprocessing framework designed to enhance the training of imputation models for

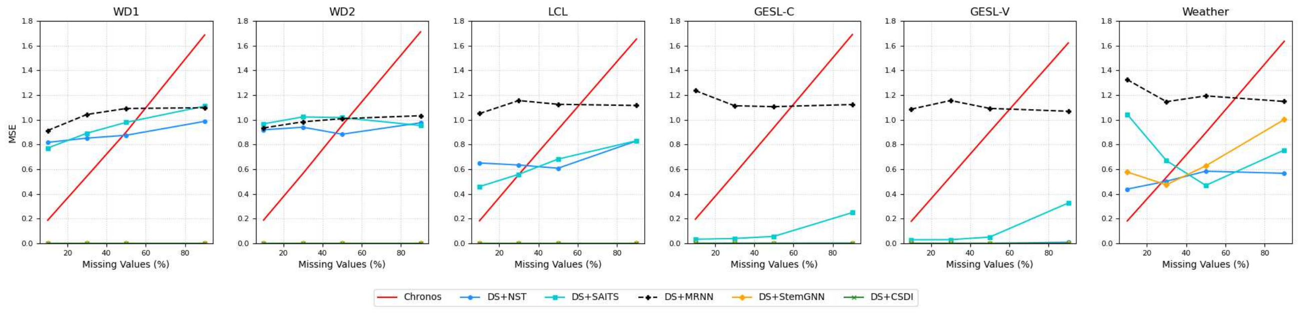

**Figure 5: MSE comparison of DIM-SUM applied to direct models and Chronos across the six datasets**

**Table 4: End-to-End (E2E) Computation Time (ms) Comparison**

| Dataset | Technique | NST | | | SAITS | | | MRNN | | | StemGNN | | | CSDI | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | @10% | @90% | Incr. | @10% | @90% | Incr. | @10% | @90% | Incr. | @10% | @90% | Incr. | @10% | @90% | Incr. |
| WD1 | DIM-SUM | **182.0** | **194.1** | +12.1 | **292.1** | **311.4** | +19.3 | **325.9** | **311.6** | -14.3 | **366.0** | **348.6** | -17.4 | **967.9** | **963.1** | -4.8 |
| | Baseline (Direct) | 1169.4 | 1269.3 | +99.9 | 1251.9 | 1332.6 | +80.7 | 1147.1 | 1233.3 | +86.2 | 381.8 | 3343.7 | +2961.9 | 993.7 | 982.7 | -11.0 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |
| WD2 | DIM-SUM | **171.1** | **173.3** | +2.2 | **263.8** | **235.6** | -28.2 | **269.7** | **276.9** | +7.2 | **373.9** | **347.1** | -26.8 | **856.0** | 869.5 | +13.5 |
| | Baseline (Direct) | 434.1 | 433.3 | -0.8 | 460.3 | 421.1 | -39.2 | 491.7 | 465.5 | -26.2 | 335.8 | 262.2 | -73.6 | 865.1 | **863.6** | -1.5 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |
| LCL | DIM-SUM | **16.2** | **14.7** | -1.5 | **199.5** | **190.6** | -8.9 | **225.4** | **227.7** | +2.3 | **29.9** | **33.0** | +3.1 | 75.2 | 72.5 | -2.7 |
| | Baseline (Direct) | 161.5 | 161.9 | +0.4 | 136.6 | 133.5 | -3.1 | 138.6 | 150.0 | +11.4 | 0.4 | 27.6 | +27.2 | **71.1** | **72.3** | +1.2 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |
| GESL-C | DIM-SUM | **8.5** | **8.4** | -0.1 | **9.7** | **14.5** | +4.8 | 11.8 | 14.1 | +2.3 | 21.0 | 21.2 | +0.2 | 44.0 | **42.2** | -1.8 |
| | Baseline (Direct) | 31.7 | 25.2 | -6.5 | 103.6 | 41.3 | -62.3 | **5.1** | **3.7** | -1.4 | **20.0** | **20.2** | +0.2 | **42.9** | 40.9 | -2.0 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |
| GESL-V | DIM-SUM | **8.9** | 8.5 | -0.4 | **15.4** | **11.4** | -4.0 | 12.3 | 14.1 | +1.8 | 20.9 | **20.4** | -0.5 | 51.4 | 50.7 | -0.7 |
| | Baseline (Direct) | 27.8 | 28.1 | +0.3 | 116.5 | 36.0 | -80.5 | **6.7** | **5.7** | -1.0 | **2.3** | 20.8 | +18.5 | **50.9** | **46.8** | -4.1 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |
| Weather | DIM-SUM | **5.1** | 6.0 | +0.9 | 10.5 | **8.5** | -2.0 | **8.0** | 9.8 | +1.8 | **12.0** | 11.3 | -0.7 | 29.4 | **28.2** | -1.2 |
| | Baseline (Direct) | 5.2 | **4.8** | -0.4 | **9.1** | **7.2** | -1.9 | 8.4 | **8.7** | +0.3 | 13.7 | **10.4** | -3.3 | **28.1** | 28.2 | +0.1 |
| | DAGAN | $5.69 \times 10^{6}$ | $6.13 \times 10^{6}$ | $+4.43 \times 10^{5}$ | $5.69 \times 10^{6}$ | $5.71 \times 10^{6}$ | $+1.93 \times 10^{4}$ | $7.47 \times 10^{6}$ | $7.41 \times 10^{6}$ | $-6.64 \times 10^{4}$ | $6.11 \times 10^{6}$ | $6.34 \times 10^{6}$ | $+2.30 \times 10^{5}$ | $6.25 \times 10^{6}$ | $6.31 \times 10^{6}$ | $+6.00 \times 10^{4}$ |
| | Chronos | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ | $1.08 \times 10^{5}$ | $8.64 \times 10^{5}$ | $+7.56 \times 10^{5}$ |

**Table 5: Reduction in training data required for DIM-SUM**

| Dataset | Clean Seq. (Baseline) | Total Seq. | DAGAN | DIM-SUM | Reduction |
|---|---|---|---|---|---|
| WD1 | 4,888,091 | 16,666,667 | 240,000 | 58,025 | 189-287x |
| WD2 | 635,274 | 4,888,091 | 100,000 | 51,241 | 12.4-95x |
| LCL | 1,746,141 | 1,746,141 | 40,000 | 7,969 | 219x |
| GESL-V | 115,622 | 115,622 | 20,000 | 2,514 | 96x |
| GESL-C | 115,622 | 115,622 | 20,000 | 2,514 | 96x |
| Weather | 427,083 | 427,083 | 40,000 | 1,564 | 273x |

time series data, particularly when faced with extensive missing values and limited complete data. By integrating pattern clustering, adaptive masking, and statistical learning guarantees, DIM-SUM effectively navigates the challenges posed by real-world missing data patterns, moving beyond traditional artificial masking techniques. This approach allows existing imputation models to learn from and adapt to the diverse and complex missing data scenarios frequently encountered in domains like smart utility management.

The framework's efficacy is demonstrated through comprehensive experiments across six diverse datasets, including water, electricity, and weather data, utilizing five different imputation model architectures. These evaluations show that DIM-SUM not only achieves comparable or improved accuracy against baseline and state-of-the-art methods but does so with significantly reduced training data and faster processing times. Furthermore, DIM-SUM provides theoretical learning guarantees, offering reliable performance even with minimal clean training data, making it a valuable tool for practical, large-scale time series applications.

## ACKNOWLEDGEMENTS

This work is supported by NSF Grants No. 2032525, 1545071, 1527536, 1952247, 2008993, 2133391, 2245372, and 2245373. Special thanks to the California Data Collaborative and California Water Districts.

## References

[1] 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. 2024. Chronos: Learning the language

of time series. *arXiv preprint arXiv:2403.07815* (2024).

[2] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. 2021. Missing Value Imputation on Multidimensional Time Series. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2533–2546. https://www.vldb.org/pvldb/vol14/p2533-bansal.pdf

[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis Dimension. *J. ACM* 36, 4 (1989), 929–965.

[4] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1185–1198. https://dl.acm.org/doi/10.14778/3407790.3407792

[5] José Cambronero, John K Feser, Michael J Smith, and Samuel Madden. 2017. Query optimization for dynamic imputation. In *Proceedings of the VLDB Endowment*, Vol. 10. 1310–1321.

[6] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xiaomo Zhu, Chao Huang, Yixiang Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. In *Advances in Neural Information Processing Systems*, Vol. 33. 17766–17778. https://proceedings.neurips.cc/paper/2020/hash/cdf6581cb7aca4b7e19ef136c6e601a5-Abstract.html

[7] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems* 31 (2018).

[8] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 6085.

[9] Chao Chen, Jaimyoung Kwon, and Pravin Varaiya. 2002. *The quality of loop data and the health of California's freeway loop detectors*. Technical Report. University of California, Berkeley. https://people.eecs.berkeley.edu/~varaiya/papers_ps.dir/QualityOfLoopData.pdf

[10] Yuhang Chen, Chaoyun Zhang, Minghua Ma, et al. 2023. ImDiffusion: Imputed Diffusion Models for Multivariate Time Series Anomaly Detection. *Proceedings of the VLDB Endowment* 17, 3 (2023), 359–372. https://www.vldb.org/pvldb/vol17/p359-zhang.pdf

[11] David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, 2 (1979), 224–227.

[12] Ton De Waal, Jeroen Pannekoek, and Sander Scholtus. 2011. *Handbook of statistical data editing and imputation*. Vol. 563. John Wiley & Sons.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019), 4171–4186.

[14] Wenjie Du, David Côté, and Yan Liu. 2023. Saits: Self-attention-based imputation for time series. *Expert Systems with Applications* 219 (2023), 119619.

[15] Wenfei Fan, Floris Geerts, and Jef Wijsen. 2014. Towards Certain Fixes with Editing Rules and Master Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 1237–1248.

[16] Zijin Feng, Miao Qiao, and Hong Cheng. 2024. Missing Data Imputation with Uncertainty-Driven Network. In *Proceedings of the 2024 ACM SIGMOD International Conference on Management of Data*. https://2024.sigmod.org/sigmod-list.html

[17] R. Hildebrant. 2025. *WD1 - AMI Dataset for Water Meters Flow Readings*. https://doi.org/10.5281/zenodo.14940605

[18] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2022. Time Series Data Management and Analytics: A Survey of Current Trends and Future Directions. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2022), 5711–5728.

[19] Mourad Khayati, Ines Arous, Zakhar Tymchenko, and Philippe Cudré-Mauroux. 2021. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams. *Proceedings of the VLDB Endowment* 14, 3 (2021), 294–306. https://www.vldb.org/pvldb/vol14/p294-khayati.pdf

[20] Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux. 2020. Mind the gap: an experimental evaluation of imputation of missing values techniques in time series. In *Proceedings of the VLDB Endowment*, Vol. 13. 768–782.

[21] Mourad Khayati, Quentin Nater, and Jacques Pasquier. 2024. ImputeVIS: An Interactive Evaluator to Benchmark Imputation Techniques for Time Series Data. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4329–4332.

[22] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[23] Sanmukh R Kuppannagari, Yao Fu, Chung Ming Chueng, and Viktor K Prasanna. 2021. Spatio-temporal missing data imputation for smart power grids. In *Proceedings of the twelfth ACM international conference on future energy systems*. 458–465.

[24] Oak Ridge National Laboratory. 2017. *Grid Echo State Lab (GESL) Dataset*. Technical Report. Oak Ridge National Laboratory. Sponsored by the U.S. Department of Energy.

[25] Yiming Lin and Sharad Mehrotra. 2023. ZIP: Lazy Imputation during Query Processing. *Proceedings of the VLDB Endowment* 17, 1 (2023), 28–40.

[26] Roderick J. A. Little and Donald B. Rubin. 2002. *Statistical Analysis with Missing Data* (2nd ed.). Wiley.

[27] Roderick J. A. Little and Donald B. Rubin. 2002. *Statistical Analysis with Missing Data*. John Wiley & Sons.

[28] Tongyu Liu, Ju Fan, Yinqing Luo, Nan Tang, Guoliang Li, and Xiaoyong Du. 2021. Adaptive data augmentation for supervised learning over missing data. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1202–1214.

[29] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems* 35 (2022), 9881–9893.

[30] Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. 2019. catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. *Data mining and knowledge discovery* 33, 6 (2019), 1821–1852.

[31] Yao Miao, Guoliang Li, and Jian Li. 2022. Efficient and Effective Data Imputation with Influence Functions. *Proceedings of the VLDB Endowment* 15, 3 (2022), 624–636. https://www.vldb.org/pvldb/vol15/p624-miao.pdf

[32] Marvin Minsky and Seymour Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA.

[33] Tung Nguyen, Jason Jewik, Hritik Bansal, Prakhar Sharma, and Aditya Grover. 2023. Climatelearn: Benchmarking machine learning for weather and climate modeling. *Advances in Neural Information Processing Systems* 36 (2023), 75009–75025.

[34] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. In *Proceedings of the VLDB Endowment*, Vol. 10. 1190–1201.

[35] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2021. Time Series Anomaly Detection: A Survey and Evaluation. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1157–1169.

[36] Xiaobin Ren, Kaiqi Zhao, Patricia J. Riddle, Katerina Taskova, Qingyi Pan, and Lianyan Li. 2023. DAMR: Dynamic Adjacency Matrix Representation Learning for Multivariate Time Series Imputation. In *Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data*. https://2023.sigmod.org/program_sigmod.shtml

[37] Donald B. Rubin. 1976. Inference and Missing Data. *Biometrika* 63, 3 (1976), 581–592.

[38] Seunghyoung Ryu, Minsoo Kim, and Hongseok Kim. 2020. Denoising Autoencoder-Based Missing Value Imputation for Smart Meters. *IEEE Access* 8 (2020), 40656–40666. https://doi.org/10.1109/ACCESS.2020.2976500

[39] Doruk Sart, Abdullah Mueen, Walid Najjar, Eamonn Keogh, and Vit Niennattrakul. 2010. Accelerating dynamic time warping subsequence search with GPUs and FPGAs. In *2010 IEEE International Conference on Data Mining*. IEEE, 1001–1006.

[40] David Sculley. 2010. Web-scale k-means clustering. (2010), 1177–1178. https://doi.org/10.1145/1772690.1772862

[41] Shaoxu Song, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2021. Missing Data Management in Database Systems: A Survey. *SIGMOD Record* 50, 2 (2021), 6–13.

[42] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems* 34 (2021), 24804–24816.

[43] Siddharth Thakur, Jaytrilok Choudhary, and Dhirendra Pratap Singh. 2021. A survey on missing values handling methods for time series data. In *Intelligent Systems: Proceedings of SCIS 2021*. Springer, 435–443.

[44] Simon Tindemans. 2023. *Low Carbon London smart meter data (refactored)*. https://doi.org/10.4121/fbbe775b-48d8-469f-a39b-b64488bfd6fd

[45] Leslie G Valiant. 1984. A Theory of the Learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.

[46] Vladimir Vapnik. 1998. *Statistical learning theory*. Wiley.

[47] Vladimir Vapnik and Alexey Chervonenkis. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications* 16, 2 (1971), 264–280.

[48] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

[49] Jingyu Wang, Jiaqi Tang, and Jiawei Han. 2020. DAME: A Distributed Adaptive Missing-value Estimation System. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2037–2050.

[50] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. 2022. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. *arXiv preprint arXiv:2202.01381* (2022).

[51] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *NeurIPS* (2021).

[52] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variations Modeling for General Time Series Analysis. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://github.com/thuml/TimesNet Published as a conference paper at ICLR 2023.

[53] Lei Yang, Zhepeng Shao, Angela Li, Saket Sathe, Jun Zhao, and Prashant Agrawal. 2021. GRAIL: Efficient Time-Series Representation Learning with Guarantees. In *Proceedings of the VLDB Endowment*, Vol. 14. 2145–2157.

[54] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. 2018. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering* 66, 5 (2018), 1477–1490.

[55] Aras Yurtman, Jonas Soenen, Wannes Meert, and Hendrik Blockeel. 2023. Estimating Dynamic Time Warping Distance Between Time Series with Missing Data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 221–237.

[56] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.