ReCTSi: Resource-efficient Correlated Time Series Imputation via Decoupled Pattern Learning and Completeness-aware Attentions

Zhichen Lai
zhla@cs.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

Dalin Zhang*
dalinz@cs.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

Huan Li*
lihuan.cs@zju.edu.cn
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou, China

Dongxiang Zhang
zhangdongxiang@zju.edu.cn
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou, China

Hua Lu
luhua@ruc.dk
Department of People and Technology, Roskilde University
Roskilde, Denmark

Christian S. Jensen
csj@cs.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

ABSTRACT

Imputation of Correlated Time Series (CTS) is essential in data preprocessing for many tasks, particularly when sensor data is often incomplete. Deep learning has enabled sophisticated models that improve CTS imputation by capturing temporal and spatial patterns. However, deep models often incur considerable consumption of computational resources and thus cannot be deployed in resource-limited settings. This paper presents ReCTSi (Resource-efficient CTS imputation), a method that adopts a new architecture for decoupled pattern learning in two phases: (1) the Persistent Pattern Extraction phase utilizes a multi-view learnable codebook mechanism to identify and archive persistent patterns common across different time series, enabling rapid pattern retrieval during inference. (2) the Transient Pattern Adaptation phase introduces completeness-aware attention modules that allocate attention to the complete and hence more reliable data segments. Extensive experimental results show that ReCTSi achieves state-of-the-art imputation accuracy while consuming much fewer computational resources than the leading existing model, consuming only 0.004% of the FLOPs for inference compared to its closest competitor. The blend of high accuracy and very low resource consumption makes ReCTSi the currently best method for resource-limited scenarios. The related code is available at https://github.com/ryanlaics/RECTSI.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Spatial-temporal systems**.

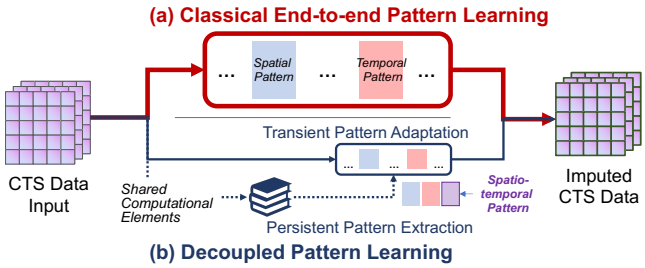

Figure 1: Two CTS imputation architectures in inference: (a) Classical end-to-end pattern learning, incorporating multiple complex spatial and temporal operators. (b) The proposed decoupled pattern learning architecture, featuring the sharing of persistent patterns and lightweight adaptation of transient patterns to improve resource-efficiency.

KEYWORDS

Correlated Time Series, Time Series Imputation, Spatio-temporal Data, Neural Network

1 INTRODUCTION

Due to the widespread and increasing digitization of processes, Correlated Time Series (CTS), derived from multiple sensors that monitor processes simultaneously are increasingly available and play a crucial role in diverse applications such as traffic management [31, 40] and predictive maintenance [9]. CTSs are often incomplete: data is missing due to sensor malfunctions [36], network problems [33], temporary obstructions [8], or other infrastructure failures. Missing data may compromise both real-time monitoring functionality and the reliability of long-term analyses [20]. As a result, **CTS (data) imputation** is an active area of research [1, 3, 11, 17, 25, 29, 30, 32, 35, 36, 38, 39].

Recent advances in CTS imputation predominantly utilize Deep Learning (DL) techniques that are known for their impressive accuracy (see Section 2) [36]. However, existing deep imputation methods have long inference latency and demand considerable

*Corresponding Authors

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671816

computational resources, including GPU/CPU and memory (see Tables 1 and 2). These shortcomings are particularly acute in settings where low-latency data processing is required, such as in emergency response systems [16], and they are further amplified in resource-limited edge settings. Thus, the deployment of existing methods is challenging in many settings.

This study aims to extend the deployability of CTS imputation by designing a deep model capable of state-of-the-art (SOTA) accuracy while consuming substantially fewer computational resources. Despite recent efforts to make DL models for various tasks more lightweight [18, 27, 41], achieving accurate CTS imputation while consuming reduced resources is non-trivial, due to the following three main limitations:

- **L1 (Rigid imputation backbone)**. Current deep imputation methods [1, 3, 11, 17, 25, 29, 30, 32, 35–39] often employ an end-to-end backbone, as depicted in Figure 1 (a). This backbone encompasses multiple temporal and spatial pattern learning modules, the aim being to create a pure deep model with standard DL computations starting from the raw input and proceeding all the way to the final result. However, such a backbone inevitably involves redundant computations related to extracting pre-existing knowledge, thus causing unnecessary computational overhead.
- **L2 (Inefficient use of periodic information in CTS)**. While integrating periodic information, such as time-of-the-day and day-of-the-week, has proven effective at improving model performance [12, 26], existing models simply treat this readily available information in the same manner as they deal with latent information that requires pattern extraction in every new input. As a result, existing models use unnecessarily high computational resources, making them less practical.
- **L3 (Inefficient iterative imputation scheme)**. Due to their success at generative tasks, diffusion models have been adopted for CTS imputation [29, 35, 37]. Imputation based on diffusion models generates missing values through an interactive process, progressing from coarse-grained to fine-grained estimation. Despite its remarkable accuracy [29, 35, 37], this form of imputation inherently consumes substantial computational resources, as each iteration involves a complete computation pass of the model. The use of this form of imputation makes it challenging to achieve high accuracy with low computational costs.

In response to these limitations, we propose ReCTSi (Resource-efficient CTS imputation via decoupled pattern learning and completeness-aware attentions). ReCTSi's primary innovation is its redesign of the conventional end-to-end imputation backbone, isolating the components that process redundant CTS knowledge (addressing limitation **L1**). Thus, ReCTSi is equipped with a decoupled pattern learning architecture, as depicted in Figure 1 (b), dividing pattern learning into two distinct phases: 1) the Persistent Pattern Extraction (PPE) phase serves as a reusable component, extracting general latent patterns; 2) the Transient Pattern Adaptation (TPA) phase then adapts the specific CTS input window, capturing transient patterns shaped by possible data dynamics, drifts, or anomalies. This decoupled design increases the adaptability of CTS pattern learning and allows for tailored strategies to optimize computations in each phase. The decoupled learning architecture is covered in Section 4.1.

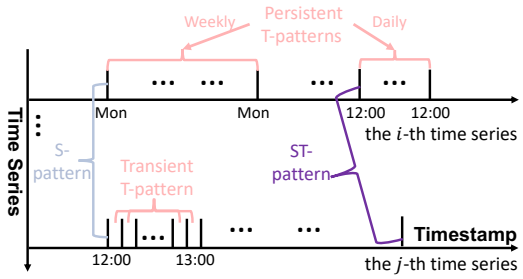

Figure 2: Multi-view patterns in CTS.

We identify three sorts of patterns in CTS: spatial, temporal, or spatio-temporal and associate each with persistent and transient components, as shown in Figure 2. Temporal patterns encompass time-related features; persistent ones recur periodically (e.g., weekly traffic flow variations), while transient ones are local and irregular (e.g., traffic spikes due to particular events). Spatial patterns concern relationships among time series; persistent spatial patterns exhibit stable and general correlations (e.g., similar traffic trends on adjacent road segments), and transient spatial patterns capture short-term dynamics (e.g., detours due to road closures). Spatio-temporal patterns combine temporal and spatial aspects with a focus on lagged spatial correlations; persistent ones are regular and stable over time and space, often reflecting a cascading effect (e.g., congestion in city centers that spreads to arterial roads), whereas transient types are characterized by varying dynamics and irregular correlations (e.g., traffic changes due to road maintenance). We propose a set of specialized techniques to achieve resource-efficient learning of both persistent and transient patterns for CTS imputation.

The PPE phase utilizes a Multi-view Learnable Codebook (MvLC) mechanism to materialize persistent patterns that can be reused among CTS input windows, providing immediate pattern access during inference. MvLC covers three kinds of persistent patterns: Persistent Temporal-pattern (PT-pattern), Persistent Spatial-pattern (PS-pattern), and Persistent Spatio-temporal-pattern (PST-pattern). Notably, PT-patterns effectively consolidate periodic information into concise, reusable representations (addressing limitation **L2**). PS-patterns identify stable correlations across different time series, while PST-patterns target correlations among neighboring timestamps across different time series. To further condense codebooks and improve resource efficiency, a Pattern Compression and Fusion (PCF) technique is designed to amalgamate the three types of patterns. The details of MvLC and PCF are provided in Section 4.2.

The TPA phase employs Completeness-aware Attention (CaA) specifically designed for the imputation task. They endeavor to recognize transient patterns, namely Transient Spatial-patterns (TS-patterns) and Transient Temporal-patterns (TT-patterns), which are key to understanding dynamic spatial and temporal correlations[1]. Self-attentions are adopted for their superior learning capability as well as tailor-ability. To be specific, completeness information (missing or not) of CTS is embedded in CaA to allocate attention

[1]Using Transient Spatio-Temporal patterns (TST-patterns) incurs quadratic space and time complexity with worse accuracy, indicating challenges in learning complex TST-patterns through a single module, excluding them from consideration in the design of CaA.

more to those complete and thus more reliable data segments. Moreover, grouped convolutions [21] are applied to the feed-forward network of CaA for further overhead refinement. When combined, these elements enable effective, fine-grained pattern learning while consuming fewer computational resources (addressing limitation **L3**). The CaA mechanism and its utility are detailed in Section 4.3.

Having covered training of ReCTSi in Section 4.4, Section 4.5 presents a detailed analysis of ReCTSi's time and space complexities, benchmarking it against the traditional end-to-end pattern learning architecture. Section 5 reports on extensive efficiency and effectiveness evaluations on five datasets spanning different domains, offering evidence that ReCTSi is capable of superior imputation accuracy while using only 0.004% of the FLOPs resource consumed by the model closest in performance.

2 RELATED WORK

Deep Models for CTS Imputation. CTS imputation has shifted from traditional methods such as MEAN, MF (Matrix Factorization), and MICE (Multivariate Imputation by Chained Equations) [38] to deep learning-based models (i.e., deep models). One school of deep models is based on the common framework used for CTS forecasting. Early instances leverage primarily temporal information, e.g., BRITS [3], a bidirectional-RNN model that incorporates information from both past and future timestamps. More recent deep models consider both spatial and temporal information [1, 10, 17, 25, 39], using graph neural networks (GNNs)[1, 17, 25] and self-attention mechanisms [10] as typical modules for spatial information extraction. Another school relies on generative schemes, such as GANs [30, 32] and VAEs [11, 36]. They also involve spatial and temporal information extraction modules, like GNNs [36] and RNNs [32]. Finally, some recent studies adopt diffusion models [29, 35, 37], which rely on an iterative scheme, to achieve performance at various generative tasks.

These deep models are capable of impressive accuracy, but at the expense of computational resource consumption and latency (see evaluations in Tables 1 and 2). In contrast, ReCTSi targets both accuracy and efficiency to extend deployability.

Lightweight Deep Models. The evolution of lightweight deep models is primarily motivated by the pressing need for edge computing applications. Two main strategies exist [5]: designing new, inherently lightweight models or compressing large, pre-existing models through techniques like knowledge distillation. Our study is aligned mostly with the former, focusing on CTS imputation, for which large pre-trained models are uncommon.

The computer vision community has been at the forefront of developing lightweight deep learning techniques such as depth-wise separable convolution [7, 14], linear transformations [13], and module scaling [34]. However, their proficiency in local feature extraction from images or videos through 2D or 3D convolutions does not extend to CTS with complex spatio-temporal characteristics. This calls for more specialized approaches. Lightweight deep learning techniques for CTS have been proposed recently but for different tasks. Lai et al. [24] target lightweight CTS forecasting that aims to predict future values, whereas CTS imputation handles missing data of both past and future. Next, David et al. [2] present a lightweight CTS classification framework using ensemble distillation, which however does not support CTS imputation. Liu et al. [28] use an adaptive embedding method to make vanilla transformer SOTA for traffic forecasting. Cheng et al. [6] propose a lightweight semi-supervised learning model for online wind turbine icing detection to improve CTS representation learning efficiency. This model, consisting of only three CNN layers with the channel calibration attention module, reduces labeling efforts and addresses the challenge of highly imbalanced data in a typical CTS application. Recently, Lai et al. [23] propose an efficient-yet-effective contrastive leaning-based method for unsupervised state detection in CTS.

Unlike existing proposals, ReCTSi is a comprehensive lightweight method tailored to CTS imputation. It is composed of novel techniques MvLC and CaA that leverage the unique characteristics of CTS and enhance performance specific to imputation, respectively.

3 PROBLEM FORMULATION

Definition 1 (Correlated Time Series, CTS). *Consider a set of* N *sensors, where each sensor generates a timestamped data sequence, constituting a time series. These series can be represented as* $\mathbf{X} \in \mathbb{R}^{\mathsf{N}\times\mathsf{T}}$, *where* $\mathbf{X}_{i,t}$ *denotes the data point from the* i*-th sensor at timestamp* t, *with* $i = 1, 2, \ldots, \mathsf{N}$ *and* $t = 1, 2, \ldots, \mathsf{T}$. *Here,* T *is the total number of timestamps. This data is referred to as* ***Correlated Time Series*** *(CTS) [40], highlighting the presence of interdependencies among data points across timestamps and devices.*

Two correlations exist in CTS: **temporal correlations**, which occur within an individual time series, and **spatial correlations**, which occur across time series. Temporal correlations thus occur in consecutive measurements in a single time series, reflecting the continuous nature of the monitored phenomena. In contrast, spatial correlations occur in concurrent measurements from different sensors, often influenced by factors like the geographical proximity among sensors. For instance, traffic flow data collected by sensors on interconnected road segments generally exhibit spatial correlations [40].

Definition 2 (CTS Imputation). *Consider raw CTS* $\mathbf{X} \in \mathbb{R}^{\mathsf{N}\times\mathsf{T}}$ *with missing values indicated by a binary mask matrix* $\mathbf{M} \in \{0, 1\}^{\mathsf{N}\times\mathsf{T}}$ *("0" denoting missing and "1", otherwise).* ***CTS imputation*** *aims to estimate the true values of all missing values in* $\mathbf{X}$, *thereby enhancing the data quality and its utility in downstream tasks. The imputation replaces only missing values, as expressed by:*

$$\hat{\mathbf{X}} = \mathbf{X} \odot \mathbf{M} + g(\mathbf{X} \mid \mathbf{M}) \odot (\mathbf{1} - \mathbf{M}), \tag{1}$$

where $\odot$ *denotes element-wise multiplication and* $g(\cdot)$ *is an imputation function that estimates missing values.*

Accurate CTS imputation hinges on the ability to extract spatial and temporal correlations from incomplete data. Deep methods, effective at capturing such correlations, are dominant despite their computational costs. As formulated next, this study aims to enable much more efficient imputation.

Problem (Resource-efficient CTS Imputation). *The aim of this study is to construct an imputation model* $\hat{g}(\cdot)$ *that can achieve accuracy comparable to those of SOTA imputation models, while*

markedly reducing the consumption of computational resources, quantified in terms of floating-point operations, FLOPs, and the amount of model parameters.

4 CONSTRUCTION OF RECTSI

We present the overall decoupled architecture of ReCTSi in Section 4.1. Then we detail the two phases, PPE and TPA, of ReCTSi for persistent pattern learning and transient pattern adaptation in Sections 4.2 and 4.3, respectively. Next, we cover the training of ReCTSi in Section 4.4. Finally, a complexity study is provided in Section 4.5.

4.1 The Decoupled Architecture of ReCTSi

In contrast to traditional deep CTS imputation models that rely heavily on an end-to-end backbone, ReCTSi employs a novel decoupled pattern learning architecture that represents a rethinking of the pattern extraction process. CTS typically encompasses two types of patterns: **persistent patterns**, capturing how data points evolve in relation to static information, such as temporal cycles and the geographic location of time series, and **transient patterns**, describing fluctuations at specific timestamps and semantic spatial correlations. Conventional approaches [1, 3, 11, 17, 25, 29, 30, 32, 35, 36, 38, 39] do not distinguish between these two types of patterns and extract both types of patterns together using an end-to-end model. However, since persistent patterns recur over time, they can be retained and reused, in contrast to transient patterns that vary over time. In line with these considerations, as illustrated in Figure 3, ReCTSi encompasses two distinct pattern learning phases: PPE and TPA.

The PPE phase focuses on representing persistent patterns that can be retained and reused. To achieve this, we design a novel Multi-view Learnable Codebook (MvLC) mechanism, detailed in Section 4.2. We follow the mechanism proposed by [22], utilizing prior knowledge to design learnable node embeddings that stores multiple persistent features. This mechanism learns and stores representations of three subtle views of persistent patterns during training, allowing for easy retrieval during imputation via periodic information updates. This not only reduces computational costs but also optimizes memory consumption.

Next, transient patterns, such as local data drifts or anomalies, also occur in CTS. To capture such patterns, ReCTSi incorporates a dedicated TPA phase, detailed in Section 4.3. This phase utilizes an innovative Completeness-aware Attention (CaA) mechanism that enables increased focus on complete and reliable data segments for both temporal and spatial transient information extraction. This mechanism enhances the method's ability to infer missing data, in contrast to a naive application of the attention mechanism that disregards the specifics of CTS imputation.

4.2 Persistent Pattern Extraction (PPE)

This phase utilizes primarily the MvLC to extract persistent patterns from three distinct views: the Persistent Temporal Pattern (PT-pattern), the Persistent Spatial Pattern (PS-pattern), and the Persistent Spatio-temporal Pattern (PST-pattern) through learnable codebooks (see Section 4.2.1). Additionally, a Pattern Compression and Fusion (PCF) module (see Section 4.2.2) is incorporated to further compress the persistent patterns and fuse them seamlessly with raw CTS, facilitating the subsequent transient pattern adaptation phase.

4.2.1 MvLC Mechanism. The **Learnable Codebook** serves as the foundation for MvLC. It functions as a dictionary that maps input information to fixed-length pattern tensors. In contrast to conventional codebooks, our learnable codebook involves random initialization of the tensors based on specific patterns, which are subsequently learned jointly with the remaining part of the model and are saved after training. During the imputation, the saved pattern tensors can be retrieved via table look-ups as an alternative to the recomputation done by many existing end-to-end models. In essence, a learnable codebook is formalized as:

$$\mathbf{CB}: \ \{k \rightarrow v \mid k \in K, v \in V\}, \tag{2}$$

where each key value k from key set K corresponds to a unique learnable tensor v from value set V mapped by the codebook $\mathbf{CB}$. By enabling the learning of pattern tensors, the same information can be encoded in different ways according to different tasks and fluctuation patterns.

PT-pattern Codebook ($\mathbf{CB}^{\text{PT}}$): Persistent Temporal Patterns (PT-patterns) represent periodic information that can employ fixed pattern representation rules for all temporal segments. For instance, information such as day-of-the-week (DoW) or time-of-the-day (ToD) is inherently periodic, and its representation should remain consistent for any given day of the week or time of day. Therefore, PT-patterns can be embedded using the following codebook:

$$\mathbf{CB}^{\text{PT}}: \{k^{\text{PT}} \rightarrow \mathbf{v}^{\text{PT}} \mid k^{\text{PT}} \in K^{\text{PT}}, \mathbf{v}^{\text{PT}} \in \mathbf{V}^{\text{PT}}\}, \tag{3}$$

where key $k^{\text{PT}} = [\text{DoW}, \text{ToD}, \ldots]$ amalgamates all periodic pieces of information, and $\mathbf{v}^{\text{PT}} \in \mathbb{R}^{\mathsf{d}_t}$ represents the corresponding PT-pattern tensor. We assign a PT-pattern tensor $\mathbf{v}_t^{\text{PT}}$ to each timestamp t and broadcast it across different time series. This is done because different timestamps possess distinct periodic information, whereas the data from different time series at the same timestamp share the same periodic information. Thus, the PT-pattern of a CTS segment is represented as follows:

$$\mathbf{E}^{\text{PT}} = [\mathbf{v}_t^{\text{PT}} \mid t = t \rightarrow t + \mathsf{T} - 1]^{\times \mathsf{N}}, \text{ where } \mathbf{E}^{\text{PT}} \in \mathbb{R}^{\mathsf{N} \times \mathsf{T} \times \mathsf{d}_t} \tag{4}$$

PS-pattern Codebook ($\mathbf{CB}^{\text{PS}}$): Similar to the PT-pattern codebook, the Persistent Spatial Pattern (PS-pattern) codebook aims to capture patterns that remain constant over time but vary spatially. This could be geographical location or identity information of time series. Therefore, value tensor $\mathbf{v}_n^{\text{PS}} \in \mathbb{R}^{\mathsf{d}_s}$ only distinguishes across distinct time series, and the key k^{PS} of the codebook $\mathbf{CB}^{\text{PS}}$ is the time series identity. These values are then broadcast across timestamps to achieve the PS-pattern:

$$\mathbf{E}^{\text{PS}} = [[\mathbf{v}_n^{\text{PS}}]^{\top} \mid n = 1 \rightarrow \mathsf{N}], \text{ where } \mathbf{E}^{\text{PS}} \in \mathbb{R}^{\mathsf{N} \times \mathsf{T} \times \mathsf{d}_s} \tag{5}$$

PST-pattern Codebook ($\mathbf{CB}^{\text{PST}}$): Finally, the MvLC also incorporates a comprehensive spatio-temporal view. Each data point has a unique spatio-temporal identity, acting as the key k^{PST}, and the corresponding value tensor $\mathbf{v}_{t,n}^{\text{PST}} \in \mathbb{R}^{\mathsf{d}_{st}}$ concerns a data point. The Persistent Spatio-Temporal (PST-pattern), $\mathbf{E}^{\text{PST}} \in \mathbb{R}^{\mathsf{N} \times \mathsf{T} \times \mathsf{d}_{st}}$, is then achieved as follows:

$$\mathbf{E}^{\text{PST}} = [[\mathbf{v}_{t,n}^{\text{PST}} \mid n = 1 \rightarrow \mathsf{N}] \mid t = t \rightarrow t + \mathsf{T} - 1] \tag{6}$$

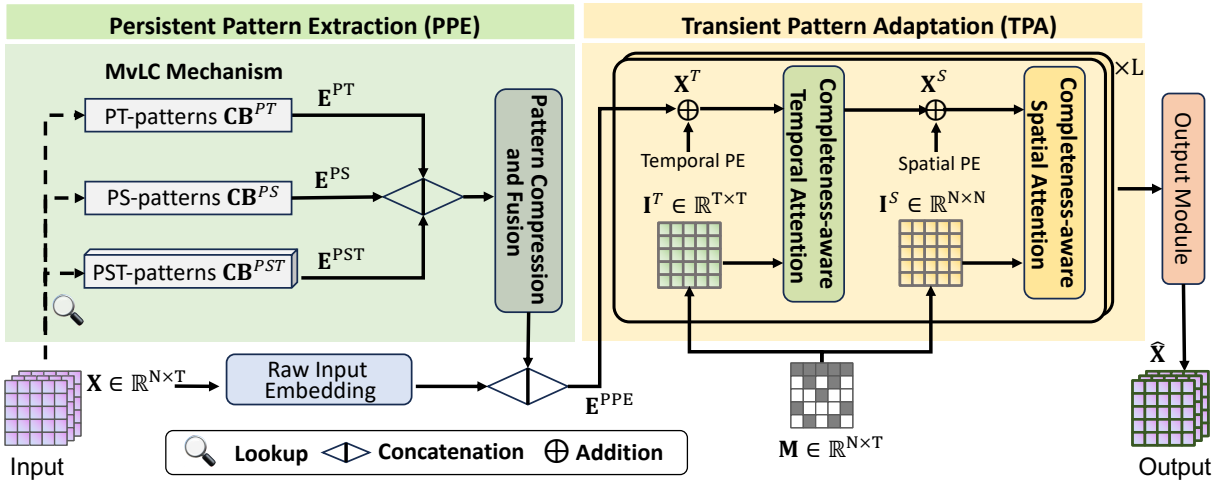

Figure 3: Overall Architecture of ReCTSi. Notably, the count of CaA modules, denoted as L, is typically set to either 1 or 2.

The codebooks $\mathbf{CB}^{\text{PS}}$ and $\mathbf{CB}^{\text{PST}}$ use identities as keys, requiring no extra input information (see Figure 3). Next, we fuse the persistent patterns from the three views to decrease redundancy and improve the efficiency of transient pattern adaptation.

4.2.2 Pattern Compression and Fusion (PCF). To emphasize more significant persistent patterns, we concatenate all three views of persistent patterns along the feature embedding dimension and then feed them into a *squeeze-and-excitation* (SE) module [15]. Specifically, a concatenated pattern is first **squeezed** with a global average pooling operation on each time series:

$$\mathbf{E}^{\circ}_{mvlc} = \text{GlobalAvgPool}(\mathbf{E}_{mvlc}), \tag{7}$$

where $\mathbf{E}_{mvlc} = [\mathbf{E}^{\text{PT}}, \mathbf{E}^{\text{PS}}, \mathbf{E}^{\text{PST}}] \in \mathbb{R}^{N\times T\times(d_t+d_s+d_{st})}$. The squeezed values are then **activated** by passing them through a non-linear transformation to capture spatial-wise dependencies:

$$S = \sigma(\mathbf{W}_{s2} \cdot \text{ReLU}(\mathbf{W}_{s1} \cdot \mathbf{E}^{\circ}_{mvlc})) \tag{8}$$

Here, $\mathbf{W}_{s1}$ and $\mathbf{W}_{s2}$ are learnable weights for dimension expansion and compression, and a sigmoid function $\sigma(\cdot)$ is applied to obtain spatial-wise scores. The computed importance scores are used to re-weight the original patterns by performing spatial-wise scaling:

$$\mathbf{E}^{\text{CF}} = S \cdot \mathbf{E}_{mvlc} \tag{9}$$

The resulting embedding $\mathbf{E}^{\text{CF}}$ is simply compressed via a linear transformation and is then merged with the linearly transformed raw CTS $\mathbf{X}$ in preparation for the transient pattern adaptation.

$$\mathbf{E}^{\text{PPE}} = [\text{Linear}(\mathbf{X}), \text{Linear}(\mathbf{E}^{\text{CF}})], \text{where } \mathbf{E}^{\text{PPE}} \in \mathbb{R}^{N\times T\times d} \tag{10}$$

4.3 Transient Pattern Adaptation (TPA)

In contrast to persistent patterns that capture the overall behavior of an CTS, transient patterns represent local data shifts deviating from routine fluctuations. The TPA phase is designed to capture such local dynamics. The conventional attention mechanisms target general purposes and are purely data-driven, learning attention scores without incorporating prior assumptions about the input. While achieving SOTA performance in purely data-driven settings, the addition of auxiliary information changes the setting and offers means of improvement. We propose a novel Completeness-aware Attention (CaA) mechanism for extracting transient spatial and temporal patterns, which customizes conventional attention to incorporate awareness of data completeness. This enhancement enables better capture of transient patterns with more focused attention on complete segments.

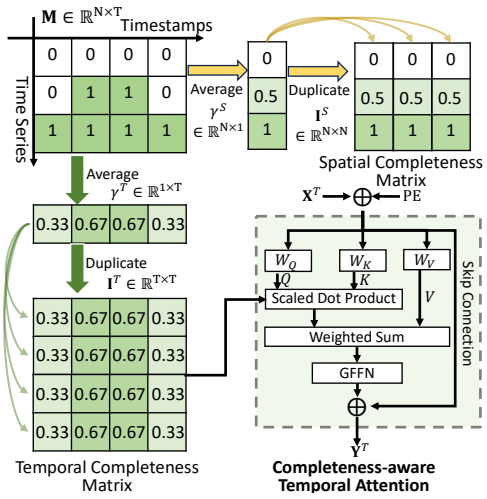

Figure 4: Completeness-aware Attention, using the temporal attention as an example. The method for Completeness-aware Spatial Attention follows a similar principle but applies self-attention across spatial data slices instead.

4.3.1 Completeness Matrices. The cornerstone of CaA is the temporal and spatial completeness matrices $\mathbf{I}^T$ and $\mathbf{I}^S$ (see Figure 3). For incomplete CTS data, data segments containing missing values contribute less to pattern learning and should be assigned less attention. As exemplified in Figure 4, the percentages of missing values in the first and fourth timestamps (columns) are 0.33, much lower than in the other two timestamps. This suggests that the features derived from these other two timestamps are less reliable

compared to those from the first two. To capture different degrees of missing data in CTS, we introduce *completeness matrices*.

Given the mask matrix $\mathbf{M}$ associated with an input CTS $\mathbf{X}$, the *temporal completeness rate* of a timestamp t, defined as its percentage of missing values, is computed as follows:

$$\gamma_t^T = \frac{1}{\mathsf{N}} \sum\nolimits_{n=1}^{\mathsf{N}} \mathbf{M}_{n,t}, \tag{11}$$

where $\mathbf{M}_{n,t}$ is the element of $\mathbf{M}$ at position (n, t). Analogously, we calculate the completeness rate for all timestamps and obtain a **temporal completeness matrix**:

$$\mathbf{I}^T = [\gamma_1^T, \gamma_2^T, \ldots, \gamma_{\mathsf{T}}^T]^{\times \mathsf{T}} \tag{12}$$

Likewise, we derive a **spatial completeness matrix** $\mathbf{I}^S$:

$$\mathbf{I}^S = [\gamma_1^S, \gamma_2^S, \ldots, \gamma_{\mathsf{N}}^S]^{\times \mathsf{N}}, \text{ where } \gamma_n^S = \frac{1}{\mathsf{T}} \sum\nolimits_{t=1}^{\mathsf{T}} \mathbf{M}_{n,t} \tag{13}$$

represents the *spatial completeness rate* for the n-th time series, as the percentage of missing values within that series.

4.3.2 Completeness-aware Attention. We integrate the completeness matrix into the conventional self-attention mechanism to construct the CaA module. The completeness matrix helps optimize attention scores, allowing the method to better leverage more complete data segments. In the following, we outline the overall CaA mechanism and highlight the distinct operations for spatial and temporal pattern extraction.

Self-attention mechanisms in the spatial and temporal attention modules have computational and memory requirements that scale quadratically with N and T, respectively. In contrast, spatio-temporal attention scale quadratically with $(\mathsf{N}\times\mathsf{T})$. Using the PeMS dataset [4] with $\mathsf{N} = 64$ as an example, spatio-temporal attention requires 4096 times more computational and memory resources than does spatial attention alone, which is prohibitive. To achieve a resource-efficient model, we therefore limit the consideration of ST-patterns to the PPE phase.

Similar to conventional self-attentions, CaA initially incorporates a positional embedding (PE) to encode permutations of the input. Given the input $\mathbf{X}^*$ ($\mathbf{X}^T$ for temporal attentions and $\mathbf{X}^S$ for spatial attentions), the positional embedding is formulated as follows:

$$\mathbf{E}^{\text{PE}} = \mathbf{X}^* + \mathbf{W}^{\text{PE}}, \tag{14}$$

where $\mathbf{W}^{\text{PE}}$ is a learnable matrix capturing position information. The resulting embedding $\mathbf{E}^{\text{PE}}$ is then used to generate the Query ($\boldsymbol{Q}$), Key ($\boldsymbol{K}$), and Value ($\boldsymbol{V}$) components of an attention module:

$$\boldsymbol{Q} = \mathbf{E}^{\text{PE}}\mathbf{W}_Q, \quad \boldsymbol{K} = \mathbf{E}^{\text{PE}}\mathbf{W}_K, \quad \boldsymbol{V} = \mathbf{E}^{\text{PE}}\mathbf{W}_V, \tag{15}$$

where $\boldsymbol{W}_* \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}/\mathsf{h}}$ are learnable weight matrices.

Unlike the conventional attention mechanism that generates attention scores solely with $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, we propose to incorporate the completeness vector $\mathbf{I}^*$ in Equations 12 and 13 to focus more on complete data segments and less on incomplete ones. Specifically, the spatial and temporal completeness-aware weighted embeddings are:

$$\mathbf{E} = \begin{cases} \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top \odot (\mathbf{I}^S)^\top}{\sqrt{\mathsf{D}/\mathsf{h}}}\right)\cdot \mathbf{V}, & \text{for spatial attentions} \\ \text{Softmax}\left(\frac{\mathbf{I}^T \odot \boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{\mathsf{D}/\mathsf{h}}}\right)\cdot \mathbf{V}, & \text{for temporal attentions} \end{cases} \tag{16}$$

where $\odot$ is element-wise multiplication, ensuring that attentions preferentially emphasize contributions from non-missing data.

A *Grouped Feedforward Network* (GFFN) with two GFFN layers [24] then takes the attentive embedding and performs lightweight non-linear activation to obtain the final output:

$$\mathbf{E}^{\text{CaA}} = \text{GFFN}(\text{GFFN}(\mathbf{E} \mid \mathsf{G}_2) \mid \mathsf{G}_1), \tag{17}$$

where G_* is the number of groups and $\text{GFFN}(\cdot \mid \mathsf{G}_*)$ is a GFFN layer, given as follows:

$$\begin{aligned} \text{GFFN}(\mathbf{E} \mid \mathsf{G}_*) &= \text{concat}(\{\text{ReLU}(\mathbf{E}_j \cdot \mathbf{W} + \mathbf{b})\}_{j=1}^{\mathsf{G}_*}), \\ \text{where} \quad \mathbf{E}_j &= \mathbf{E}\left[:, :, \frac{\mathsf{d}\times(j-1)}{\mathsf{G}_*} : \frac{\mathsf{d}\times j}{\mathsf{G}_*}\right] \end{aligned} \tag{18}$$

As shown in Figure 3, the spatial and temporal CaA modules are stacked to form a TPA block, and L TPA blocks with a skip connection for each are stacked sequentially to achieve the TPA phase (primarily, $\mathsf{L} is set to 1 or 2$). The final output embedding is denoted as $\mathbf{E}^{\text{TPA}}$.

4.4 Training of ReCTSi

ReCTSi processes the output of TPA and applies a non-linear transformation to generate the final imputation results:

$$\hat{\mathbf{X}} = \text{ReLU}(\mathbf{E}^{\text{TPA}} \cdot \mathbf{W}^o + \mathbf{b}^o), \tag{19}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{\mathsf{N}\times\mathsf{T}}$ is the imputed result and $\mathbf{W}^o$ and $\mathbf{b}^o$ are learnable weights and biases.

Despite its decoupled architecture, ReCTSi adopts an end-to-end training scheme, where PPE and TPA are trained together by the Adam optimizer. ReCTSi utilizes the Masked Mean Absolute Error (Masked MAE) [3, 10, 36] as the loss function. This function is designed specifically to optimize imputation accuracy by considering errors only in the masked (missing) regions:

$$\text{Masked MAE} = \frac{\sum_{i=1}^{\mathsf{N}}\sum_{j=1}^{\mathsf{T}}(1-\mathbf{M}_{i,j})\cdot\left|\mathbf{X}_{i,j} - \hat{\mathbf{X}}_{i,j}\right|}{\sum_{i=1}^{\mathsf{N}}\sum_{j=1}^{\mathsf{T}}(1-\mathbf{M}_{i,j})}, \tag{20}$$

where $\mathbf{X}_{i,j}$ and $\hat{\mathbf{X}}_{i,j}$ represent the ground-truth and imputed values, respectively, at the i-th time series and the j-th timestamp.

After training, the codebooks of PPE are stored and used for subsequent imputation inference via table look-ups, and TPA is used as a normal neural network to process each input CTS segment.

4.5 Complexity Analysis

Classical end-to-end models (see Figure 1 (a)) utilize layered approaches for extracting temporal and spatial features, including RNNs, CNNs, and self-attention for temporal patterns, as well as GCNs and self-attention for spatial patterns. Let L be the number of stacked modules, d the embedding size, N the number of time series, and T the window size. Then these models have time complexity $O(\mathsf{L}\cdot\mathsf{d}^2\cdot\mathsf{N}\cdot\mathsf{T})$ in the cases of RNN- and CNN-based modules, $O(\mathsf{L}\cdot\mathsf{d}\cdot\mathsf{N}\cdot\mathsf{T}\cdot(\mathsf{N}+\mathsf{d}))$ in the cases of GCN- and self-attention-based spatial modules, and $O(\mathsf{L}\cdot\mathsf{d}\cdot\mathsf{N}\cdot\mathsf{T}\cdot(\mathsf{T}+\mathsf{d}))$ in the cases of self-attention-based temporal modules, and the space complexity is $O(\mathsf{L}\cdot\mathsf{d}^2)$.

In contrast, ReCTSi introduces a novel decoupled architecture for pattern learning, shown in Figure 1 (b), employs PPE to capture persistent patterns and TPA to capture transient patterns. This approach uses MvLC codebooks to optimize persistent pattern storage

and converts persistent feature extraction during inference into an efficient $O(1)$ table look-up. Consequently, the space complexity grows by $O(\mathsf{N}\cdot\mathsf{T}\cdot\mathsf{d})$, due to the necessity to store persistent patterns in the MvLC codebook. Nevertheless, the parameter count of the codebook is relatively small (between 60 to 180KB for all datasets shown in Tables 1 and 2), which our experiments confirm.

For TPA, the space and time complexities of the CaA module align with those of traditional self-attention mechanisms. Nonetheless, due to the foundational work of PPE, TPA often needs far fewer self-attention modules, typically reducing L to 1. This reduction, along with a reduced embedding size (e.g., $\mathsf{d} = 32$ of ReCTSi vs $\mathsf{d} = 64$ of models like PoGeVon [36] on AQ36 dataset), renders the decoupled approach very efficient. Moreover, ReCTSi integrates a pattern compression and fusion module to decrease further the embedding size for the input of TPA. The computational complexity of Grouped FFNs (GFFNs) in CaA is also optimized to $\frac{(1+1/\mathsf{G})}{2}$ of the case for the standard FFN counterpart [24], enhancing the model's efficiency. The experimental results presented in Tables 1 and 2 offer evidence that ReCTSi achieves the lowest computational costs.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We use five real-world datasets that have been extensively adopted in studies [19, 36, 42] and covering domains of traffic flow, air quality, and COVID-19 infection rates. The configurations are in accordance with prior research [36]. More details can be found in Table 3.

Metrics. We evaluate the imputation **effectiveness** using standard metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Relative Error (MRE), following prior research [36]. MAE assigns equal weights to all errors, offering a general accuracy measure. MSE gives more weight to severe errors, thus emphasizing significant deviations. MRE is sensitive to errors when ground truth values are low, offering insight into the relative accuracy. Lower MAE, MSE, and MRE values indicate higher imputation accuracy.

When evaluating the imputation **efficiency**, we present standard device-agnostic metrics during *inference*: FLOPs and numbers of model parameters. These metrics align with established deep learning benchmarks [24, 34] and are calculated using the `Torchinfo` library. Additionally, we report latency and peak memory usage (Peak Mem) as runtime-specific measures. Particularly, the PeMS series datasets have identical context and statistical properties (see Table 3); therefore, ReCTSi has a uniform configuration for these PeMS datasets, and they share the same efficiency results as reported in Table 1.

5.2 Overall Comparisons

Baseline. We include ten existing methods in the evaluation. These methods were chosen to cover a broad range of approaches. Specifically, MEAN, MF, and MICE [38] exemplify classical non-DL approaches. Despite their resource efficiency, these models fall short in terms of accuracy, rendering them less desirable in many applications. Next, rGAIN [32] and PoGeVon [36] represent generative models. They leverage advanced techniques such as GANs and VAEs, respectively, to impute missing values. A different class of models, including BRITS [3], TimesNet [39], GRIN [1], NET^3 [17], and SAITS [10], integrate layered temporal and spatial modules to perform imputation.

Among these, SAITS [10] stands out for its attention mechanisms that somewhat resemble those employed by ReCTSi. The remaining models in this class employ different strategies for temporal pattern recognition, including RNNs (BRITS [3], GRIN [1], NET^3 [17], PoGeVon [36]), CNNs (TimesNet [39]), and MLPs (rGAIN [32]) models. Considering spatial pattern identification, GRIN [1] and PoGeVon [36] utilize MPNNs, whereas NET^3 [17] employs a GCN approach. Notably, we exclude diffusion-based models [29, 35, 37] as their complex iterative sampling process requires hundreds of model calls, which is far less efficient than the other models, rendering them suitable only for efficiency-insensitive scenarios.

Comparison Analysis. The comparison results are presented in Tables 1 and 2, providing evidence that ReCTSi is capable of the best performance across various datasets. Specifically, classical non-DL methods (MEAN, MF, and MICE) achieve the lowest accuracy, with the deep methods exhibiting much better accuracy. For each dataset, ReCTSi consistently achieves the lowest MAE, MSE, and MRE values, with the only exception for MAE on COVID-19. However, the slight MAE discrepancy of 0.008 between ReCTSi and the SOTA model in this dataset, comprised solely of integer values, is negligible. On PeMS-BA, PeMS-LA, and PeMS-SD datasets, ReCTSi outperforms the closest competitor in terms of accuracy, PoGeVon, by at least 11%, with evidence of exceptional computational efficiency, exhibiting much fewer FLOPs and lower latency when compared to the existing models. For instance, in comparison with PoGeVon with the closest accuracy, ReCTSi operates hundreds of times faster, with a computational cost below 1/25,000 times FLOPs. Even compared to the most efficient method, TimesNet, ReCTSi is substantially better in terms of both FLOPs and latency, while achieving superior accuracy. The few FLOPs and low latency are due to ReCTSi's novel decoupled architecture that transforms feature extraction into fast pattern look-ups.

5.3 Ablation Study

We conduct an ablation study to assess each component of ReCTSi. Initially, we evaluate each codebook individually by systematically removing them to create ReCTSi\PT, ReCTSi\PS, and ReCTSi\PST. Additionally, we scrutinize the role of the PCF module (refer to Section 4.2.2) by introducing ReCTSi\CF, a variant with PCF removed, and codebook embeddings directly concatenated. Furthermore, to explore the impact of CaA components (refer to Section 4.3), we create variants ReCTSi\CM, ReCTSi\SA, and ReCTSi\TA, removing the completeness matrices, spatial attention, and temporal attention, respectively.

The results in Table 4 show a substantial contribution of each component in ReCTSi. The variants without codebooks (ReCTSi\PT, ReCTSi\PS, and ReCTSi\PST) exhibit elevated error metrics, emphasizing the crucial role of the codebooks at capturing multi-view complex patterns. Notably, the removal of the PST-pattern codebook results in the highest errors, underscoring the crucial role of investigating cross spatio-temporal patterns. The variant without the PCF module (ReCTSi\CF) shows increased errors and resource usage, emphasizing the module's effectiveness at pattern

Table 1: Performance comparison over three traffic datasets (lower values indicate better performance for all metrics).

Model	Efficiency Metrics				PeMS-BA			PeMS-LA			PeMS-SD		
	FLOPs (M)	Params (K)	Peak Mem (MB)	Latency (ms)	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
MEAN	\	\	\	\	192.047	47504.159	0.474	216.681	62664.657	0.406	208.192	55780.002	0.529
MF	\	\	\	\	57.265	8091.407	0.103	77.339	15202.678	0.145	45.811	6044.345	0.117
MICE	\	\	\	\	50.861	6724.148	0.126	64.018	10822.355	0.120	38.978	4771.186	0.100
BRITS	358	576	**2.69**	40.65	30.274	2942.411	0.075	36.921	3681.595	0.069	21.232	1563.234	0.054
rGAIN	5880	2280	5.02	18.83	38.862	3422.914	0.096	49.611	5533.964	0.093	33.212	2341.466	0.085
SAITS	1.43	1425	7.35	45.35	46.567	5412.574	0.115	61.896	10998.854	0.116	34.117	4101.397	0.087
TimesNet	99	1399	6.24	10.69	25.859	1676.843	0.064	27.452	2058.227	0.052	21.583	1284.300	0.055
GRIN	286	195	12.62	127.81	30.057	1922.072	0.074	47.835	4561.512	0.090	41.001	3000.012	0.105
NET^3 ♣	\	\	\	130.12	35.671	2735.574	0.089	37.652	3416.784	0.071	34.111	2487.581	0.087
PoGeVon	5030	322	16.08	172.82	22.194	1248.681	0.055	23.905	1714.962	0.045	18.990	951.559	0.048
ReCTSi	**0.06**	**83**	12.54	**1.79**	**18.998**	**1050.328**	**0.046**	**21.130**	**1469.592**	**0.040**	**15.024**	**721.439**	**0.037**

♣ Due to incompatibility between NET^3 and the `Torchinfo` library, we have issues in obtaining NET^3's FLOPs, Params, and Peak Mem. Nonetheless, considering its significantly higher latency and lower accuracy metrics compared to ReCTSi, NET^3 is not competitive.

Table 2: Performance comparison over the AQ36 and COVID-19 datasets.

Model	AQ36							COVID-19						
	FLOPs (M)	Params (K)	Peak Mem (MB)	Latency (ms)	MAE	MSE	MRE	FLOPs (M)	Params (K)	Peak Mem (MB)	Latency (ms)	MAE	MSE	MRE
MEAN	\	\	\	\	62.299	6525.709	0.835	\	\	\	\	3.081	10.707	0.284
MF	\	\	\	\	39.582	4545.596	0.531	\	\	\	\	0.276	0.165	0.026
MICE	\	\	\	\	38.889	4314.435	0.521	\	\	\	\	0.077	0.013	0.007
BRITS	334	266	1.42	46.68	23.393	1276.226	0.314	186	476	**2.10**	89.36	0.386	0.293	0.036
rGAIN	250	262	**0.72**	16.35	25.032	1358.134	0.335	3080	2096	4.52	72.53	0.579	0.571	0.055
SAITS	1.38	1376	7.93	60.41	51.907	5026.475	0.685	1.41	1411	6.60	43.13	0.466	0.366	0.043
TimesNet	149	1392	6.52	10.65	40.700	3383.554	0.545	61	1397	5.97	10.32	0.028	0.002	0.003
GRIN	241	191	10.76	148.86	29.420	2050.726	0.394	146	194	6.80	76.22	0.319	0.165	0.029
NET^3	\	\	\	98.31	34.755	2473.718	0.466	\	\	\	56.76	0.547	0.682	0.051
PoGeVon	4850	323	15.42	230.10	19.581	1238.820	0.262	2930	322	9.90	98.35	**0.007**	**0.000**	**0.001**
ReCTSi	**0.06**	**76**	7.07	**1.26**	**19.483**	**1102.159**	**0.261**	**0.12**	**145**	9.81	**1.76**	0.015	**0.000**	**0.001**

Table 3: Statistics of the datasets.

Dataset	Type	Time Series	Timestamps	Interval	Missing Rate
PeMS-BA	Traffic Flow	64	25,920	5 minutes	25%
PeMS-LA	Traffic Flow	64	25,920	5 minutes	25%
PeMS-SD	Traffic Flow	64	25,920	5 minutes	25%
AQ36	Air Quality	36	8,759	1 hour	13%
COVID-19	Infection Case	50	346	1 day	25%

Table 4: Ablation study of ReCTSi on AQ36.

Model	FLOPs (M)	Params (K)	Peak Mem (MB)	Latency (ms)	MAE	MSE	MRE
ReCTSi\PT	0.06	76	6.86	1.27	21.140	1225.109	0.290
ReCTSi\PS	0.06	76	6.91	1.35	21.036	1198.529	0.282
ReCTSi\PST	0.06	63	6.91	1.21	21.847	1395.944	0.295
ReCTSi\CF	0.13	154	12.53	1.53	20.388	1173.026	0.273
ReCTSi\CM	0.06	76	7.07	1.33	20.132	1134.471	0.270
ReCTSi\SA	0.04	55	6.90	1.02	23.385	1532.564	0.313
ReCTSi\TA	0.04	55	4.33	0.94	25.414	1855.463	0.332
ReCTSi	0.06	76	7.07	1.26	**19.483**	**1102.159**	**0.261**

compression and fusion. Removing the completeness matrix degrades CaA to conventional attention, and the inferior performance of ReCTSi\CM compared to ReCTSi indicates the necessity of integrating imputation-specific information. Variants ReCTSi\SA and ReCTSi\TA demonstrate the essential nature of transient patterns for successful imputation. Overall, the superior performance of the complete ReCTSi confirms the synergistic effect of its components, highlighting their collective importance for ReCTSi's performance.

5.4 Evaluation on Varying Resource Levels

We proceed to study how ReCTSi performs under different computational resource constraints, fulfilled by varying embedding size d and window size T. The embedding size d determines the capacity of the model's feature representations for discerning nuanced patterns in the data. While a larger d can increase a model's accuracy by capturing more detailed features, it also increases the model's complexity, leading to a higher risk of overfitting and increasing the computational requirements, reflected in higher FLOPs. The window size T constrains the temporal scope for pattern learning. Increasing T allows for involving longer-term dependencies, offering more context for missing values. However, it also increases resource consumption, as a larger T necessitates processing more data points for each inference. Finding the configuration of d and T affects the balance between model expressiveness and efficiency. This balance becomes particularly crucial in resource-constrained scenarios or when deploying the method at scale.

Figure 5 shows the trade-off between ReCTSi resource consumption in terms of FLOPs and its imputation accuracy in terms of MAE on the AQ36 dataset. Increasing the **embedding size** d initially leads to a decrease in MAE, signifying an improvement in the feature representation and accuracy. This improvement plateaus and starts to decrease after reaching a certain d value, due to overfitting. This overfitting unnecessarily increases computational costs, indicating that beyond a certain point, a larger d does not yield better performance. Likewise, the **window size** T plays a pivotal role in

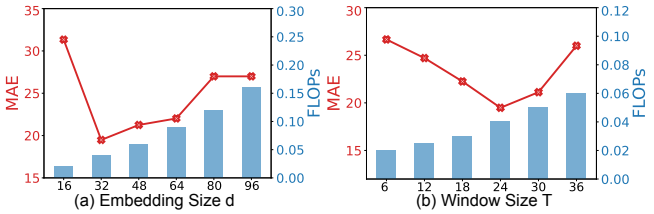

Figure 5: MAE and FLOPs when varying (a) the embedding size d and (b) the imputation window size T.

the model's ability to understand and utilize temporal patterns in the dataset. An optimal window size of $T = 24$ hours aligns with the dataset's daily periodicity, thereby optimizing the model's capabilities. Extending T beyond this period leads to the inclusion of superfluous information, which not only diminishes performance but also reduces computational efficiency.

Hence, under a specific resource constraint, higher computational overhead does not necessarily translate to better accuracy. It is imperative to select appropriate d and T values to maintain a purposeful balance between accuracy and computational resource consumption. Therefore, there is a need for a systematic mechanism to identify the optimal model configurations given a resource constraint, which we remain it as a future research direction.

6 CONCLUSION AND FUTURE WORK

We present ReCTSi, a resource-efficient model for Correlated Time Series (CTS) imputation, leveraging decoupled pattern learning and completeness-aware attention mechanisms. Inspired by rethinking existing deep models, ReCTSi adopts a decoupled architecture that distinguishes between persistent and transient patterns in CTS data. While persistent patterns can be retained and reused, transient patterns require adaptation to distinct CTS windows. To materialize and reuse persistent patterns during inference, ReCTSi introduces a Multi-view Learnable Codebook (MvLC) mechanism, enabling efficient retrieval. Additionally, novel Completeness-aware Attention (CaA) modules are proposed for transient spatial and temporal feature learning, which are tailored specifically for the imputation task. Experimental results show that ReCTSi not only achieves state-of-the-art imputation accuracy but also substantially decreases the computational resource consumption.

Promising future directions include extending ReCTSi to real-time imputation processing. Also, integrating a Neural Architecture Search framework for hyperparameter optimization under a given resource constraint would increase the usability of the method, leading to even more efficient and effective CTS imputation.

ACKNOWLEDGMENTS

This work was supported by the AAU Bridging Project (Grant no. 760848), and the Major Research Program of Zhejiang Provincial Natural Science Foundation (Grant no. LD24F020015). Huan Li and Dongxiang Zhang are supported by the Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security. Hua Lu's work was supported in part by Independent Research Fund Denmark (Grant no. 8022-00366B).

REFERENCES

[1] Cini Andrea, Marisca Ivan, Cesare Alippi, et al. 2022. Filling the g_ap_s: multivariate time series imputation by graph neural networks. In *ICLR*. 1–20.
[2] David Campos, Miao Zhang, Bin Yang, Tung Kieu, Chenjuan Guo, and Christian S. Jensen. 2023. LightTS: lightweight time series classification with adaptive ensemble distillation. In *PACMMOD*, Vol. 1. 1–27.
[3] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. BRITS: bidirectional recurrent imputation for time series. In *NeurIPS*, Vol. 31.
[4] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *TRR* 1748, 1 (2001), 96–102.
[5] Yanjiao Chen, Baolin Zheng, Zihan Zhang, Qian Wang, Chao Shen, and Qian Zhang. 2020. Deep learning on mobile and embedded devices: state-of-the-art, challenges, and future directions. *ACM Comput. Surv.* 53, 4 (2020), 1–37.
[6] Xu Cheng, Fan Shi, Xiufeng Liu, Meng Zhao, and Shengyong Chen. 2021. A novel deep class-imbalanced semisupervised model for wind turbine blade icing detection. *IEEE TNNLS* 33, 6 (2021), 2558–2570.
[7] François Chollet. 2017. Xception: deep learning with depthwise separable convolutions. In *CVPR*. 1251–1258.
[8] Francesco Concas, Julien Mineraud, Eemil Lagerspetz, Samu Varjonen, Xiaoli Liu, Kai Puolamäki, Petteri Nurmi, and Sasu Tarkoma. 2021. Low-cost outdoor air quality monitoring and sensor calibration: a survey and critical analysis. *TOSN* 17, 2 (2021), 1–44.
[9] Amin Dhaou, Antoine Bertoncello, Sébastien Gourvénec, Josselin Garnier, and Erwan Le Pennec. 2021. Causal and interpretable rules for time series analysis. In *KDD*. 2764–2772.
[10] Wenjie Du, David Côté, and Yan Liu. 2023. SAITS: self-attention-based imputation for time series. *ESWA* 219 (2023), 119619.
[11] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. 2020. Gp-vae: deep probabilistic time series imputation. In *AISTATS*. 1651–1661.
[12] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, Vol. 33. 922–929.
[13] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. 2020. Ghostnet: more features from cheap operations. In *CVPR*. 1580–1589.
[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).
[15] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *CVPR*. 7132–7141.
[16] Alejandro Jaimes and Joel Tetreault. 2021. Real-time event detection for emergency response tutorial. In *KDD*. 4042–4043.
[17] Baoyu Jing, Hanghang Tong, and Yada Zhu. 2021. Network of tensor time series. In *WWW*. 2425–2437.
[18] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: a knowledge distillation framework for recommender system. In *CIKM*. 605–614.
[19] Satya Katragadda, Ravi Teja Bhupatiraju, Vijay Raghavan, Ziad Ashkar, and Raju Gottumukkala. 2022. Examining the COVID-19 case growth rate due to visitor vs. local mobility in the United States using machine learning. *Sci. Rep.* 12, 1 (2022), 12337.
[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 *PVLDB*, Vol. 13. 768–782.
[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.
[22] Zhichen Lai, Xu Cheng, Xiufeng Liu, Lizhen Huang, and Yongping Liu. 2022. Multiscale wavelet-driven graph convolutional network for blade icing detection of wind turbines. *IEEE Sens. J.* 22, 22 (2022), 21974–21985.
[23] Zhichen Lai, Huan Li, Dalin Zhang, Yan Zhao, Weizhu Qian, and Christian S. Jensen. 2024. E2Usd: Efficient-yet-effective Unsupervised State Detection for Multivariate Time Series. In *WWW*. 3010–3021.
[24] Zhichen Lai, Dalin Zhang, Huan Li, Christian S. Jensen, Hua Lu, and Yan Zhao. 2023. LightCTS: a lightweight framework for correlated time series forecasting. In *PACMMOD*, Vol. 1. 1–26.
[25] Xiao Li, Huan Li, Hua Lu, Christian S. Jensen, Varun Pandey, and Volker Markl. 2023. Missing value imputation for multi-attribute sensor data streams via message propagation. In *PVLDB*, Vol. 17. 345–358.
[26] Youru Li, Zhenfeng Zhu, Deqiang Kong, Meixiang Xu, and Yao Zhao. 2019. Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. In *AAAI*, Vol. 33. 1004–1011.
[27] Chang Liu, Chongyang Tao, Jiazhan Feng, and Dongyan Zhao. 2022. Multi-granularity structural knowledge distillation for language model compression. In *ACL*. 1001–1011.
[28] Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. 2023. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *CIKM*. 4125–4129.

[29] Mingzhe Liu, Han Huang, Hao Feng, Leilei Sun, Bowen Du, and Yanjie Fu. 2023. PriSTI: a conditional diffusion framework for spatiotemporal imputation. In *ICDE*.
[30] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. 2019. E2GAN: end-to-end generative adversarial network for multivariate time series imputation. In *IJCAI*. 3094–3100.
[31] Zhipeng Luo, Jianqiang Huang, Ke Hu, Xue Li, and Peng Zhang. 2019. AccuAir: winning solution to air quality prediction for KDD Cup 2018. In *KDD*. 1842–1850.
[32] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. 2021. Generative semi-supervised learning for multivariate time series imputation. In *AAAI*, Vol. 35. 8983–8991.
[33] Chenmeng Qiu, Yuzhi Li, Mingyu Kang, Duxin Chen, and Wenwu Yu. 2023. CDSTTN: a data imputation method for cyber-physical systems by causal dense spatial-temporal transformer network. *IEEE J EM SEL TOP C* (2023).
[34] Mingxing Tan and Quoc Le. 2019. Efficientnet: rethinking model scaling for convolutional neural networks. In *ICML*. 6105–6114.
[35] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: conditional score-based diffusion models for probabilistic time series imputation. In *NeurIPS*, Vol. 34. 24804–24816.
[36] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *KDD*. 2256–2268.
[37] Xu Wang, Hongbo Zhang, Pengkun Wang, Yudong Zhang, Binwu Wang, Zhengyang Zhou, and Yang Wang. 2023. An observed value consistent diffusion model for imputing missing values in multivariate time series. In *KDD*. 2409–2418.
[38] Ian R White, Patrick Royston, and Angela M Wood. 2011. Multiple imputation using chained equations: issues and guidance for practice. *Stat Med* 30, 4 (2011), 377–399.
[39] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. TimesNet: temporal 2D-variation modeling for general time series analysis. In *ICLR*.
[40] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S. Jensen. 2021. AutoCTS: automated correlated time series forecasting. In *PVLDB*, Vol. 15. 971–983.
[41] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. 2021. Self-distillation: towards efficient and compact neural networks. *IEEE TPAMI* 44, 8 (2021), 4388–4403.
[42] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting fine-grained air quality based on big data. In *KDD*. 2267–2276.