MTM: A Multi-Scale Token Mixing Transformer for Irregular Multivariate Time Series Classification

Shuhan Zhong
The Hong Kong University of Science and Technology
Hong Kong SAR, China
szhongaj@cse.ust.hk

Weipeng Zhuo*
Beijing Normal-Hong Kong Baptist University
Zhuhai, China
weipengzhuo@uic.edu.cn

Sizhe Song
The Hong Kong University of Science and Technology
Hong Kong SAR, China
ssongad@cse.ust.hk

Guanyao Li
Guangzhou Urban Planning and Design Survey Research Institute
Guangzhou, China
gliaw@connect.ust.hk

Zhongyi Yu
Beijing Normal-Hong Kong Baptist University
Zhuhai, China
zhongyiyu@uic.edu.cn

S.-H. Gary Chan
The Hong Kong University of Science and Technology
Hong Kong SAR, China
gchan@cse.ust.hk

Abstract

Irregular multivariate time series (IMTS) is characterized by the lack of synchronized observations across its different channels. In this paper, we point out that this channel-wise asynchrony can lead to poor channel-wise modeling of existing deep learning methods. To overcome this limitation, we propose MTM, a **m**ulti-scale **t**oken **m**ixing transformer for the classification of IMTS. We find that the channel-wise asynchrony can be alleviated by down-sampling the time series to coarser timescales, and propose to incorporate a masked concat pooling in MTM that gradually down-samples IMTS to enhance the channel-wise attention modules. Meanwhile, we propose a novel channel-wise token mixing mechanism which proactively chooses important tokens from one channel and mixes them with other channels, to further boost the channel-wise learning of our model. Through extensive experiments on real-world datasets and comparison with state-of-the-art methods, we demonstrate that MTM consistently achieves the best performance on all the benchmarks, with improvements of up to 3.8% in AUPRC for classification.

CCS Concepts

• **Computing methodologies → Neural networks**; • **Mathematics of computing → Time series analysis**.

Keywords

Irregular Multivariate Time Series, Channel-wise Asynchrony, Token Mixing, Multi-Scale, Transformer

*Corresponding Author

ACM Reference Format:
Shuhan Zhong, Weipeng Zhuo, Sizhe Song, Guanyao Li, Zhongyi Yu, and S.-H. Gary Chan. 2025. MTM: A Multi-Scale Token Mixing Transformer for Irregular Multivariate Time Series Classification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737058

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15600519.

1 Introduction

Irregular multivariate time series (IMTS) is a common data modality in various science domains and real-world applications [10, 29, 30, 33]. An IMTS is a sequence of *irregularly sampled* observations indexed in time order, where observations of different variates form the "channel" dimension. The irregularity of sampling is characterized by the uneven time interval between consecutive observations, as well as partially observed channels. Specifically, we find that the irregularity of IMTS can lead to a serious lack of synchronized observations across its different channels, as shown in Figure 1(a). Such channel-wise asynchrony presents significant challenges for deep learning methods to model channel-wise correlation of IMTS data. Early efforts on IMTS analysis mainly focus on improving the temporal modeling of IMTS data by considering the uneven time interval with specially designed deep learning modules. However, most of them have not fully considered the channel-wise asynchrony [5, 36, 37]. They either bypass channel-wise asynchrony by simply processing different channels independently, or impute missing channels to obtain a fixed-size vector per timepoint as model input, which leads to poor channel-wise modeling of the data.

In light of this, some methods propose Transformer models that tokenize each single observation in an IMTS and rely on attention modules along the temporal and channel dimensions to model their correlations [38, 42, 44]. Nevertheless, we argue that the channel-wise attention in these methods is also *ineffective* for channel-wise modeling, as it relies on the *synchronized* observations. Due to the channel-wise asynchrony, such channel-wise attention is always

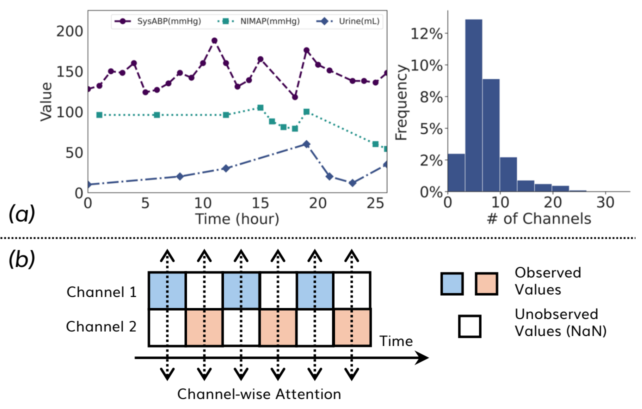

Figure 1: (a) Left: An example of irregular multivariate time series. Right: The distribution of number of channels observed per timepoint in the P12 dataset. (b) A failure example of channel-wise attention in existing Transformers. Channels 1 and 2 have no synchronized observation, and will be processed as independent even though they are potentially correlated.

computed over a small part of channels. Moreover, if some channels do not have any synchronized observations, which is a common case in real-world IMTS data, no channel-wise attention will be computed between these channels (Figure 1(b)). These channels are thus processed independently despite that the potential correlations between them may help the downstream task. As presented in Table 1, we find almost no performance drop when substituting the channel-wise attention module with simple multi-layer perceptrons (MLPs) in such Transformers.

In this work, we overcome the limitation on channel-wise modeling of existing methods in two aspects, i.e., to improve the effectiveness of the channel-wise attention with multi-scale down-sampling, and to propose a new token mixing mechanism to further enhance the channel-wise modeling. First, we find that the lack of synchronized observations can be gradually alleviated when we down-sample the IMTS to coarser timescales. However, the limited existing works on this are either uni-scale patching [45], or cannot guarantee to mitigate the channel-wise asynchrony [44]. On the other hand, different from the general down-sampling on regular dense data, the irregularity and sparsity of IMTS makes it more critical to retain information during the down-sampling process. The effectiveness of general down-sampling methods remains *unknown* on IMTS. In this work, we conduct a comprehensive comparison of potential down-sampling methods, and find that a simple concatenation of max and average temporal pooling, with the missing values masked, is the most effective IMTS down-sampler for our purpose.

Furthermore, although down-sampling can help cross-channel feature learning in coarser timescales, it sacrifices the temporal resolution of the data and still suffers in finer timescales. To solve this problem, we propose a novel token mixing mechanism to further enhance the channel-wise feature learning without down-sampling. The intuition behind is to mix tokens across channels *even if they*

Table 1: The channel-wise attention module in Transformer is ineffective for IMTS classification.

Transformer	P12 AUROC	P19 AUROC	PAM F1-Score
w/ Channel Attention	86.1 ±1.6	87.3 ±3.2	95.8 ±1.2
w/o Channel Attention	85.8 ±1.6	87.6 ±2.7	96.1 ±0.7

are not synchronized, which is achieved by proactively choosing important tokens from one channel, and mixing them with unsynchronized tokens in other channels. Specifically, based on the attention scores from a set of specially designed `[CLS]` tokens, we find out the importance of tokens within each channel. We then choose the pivotal tokens to fill the missing channels of the same timepoint. Then, we mix the information of each channel's original tokens with the pivotal tokens from other channels with another self-attention step. After the mixing operation, we reset the missing channels to keep the original sampling patterns *unchanged*. Through this token mixing mechanism, the pivotal features from one channel can be effectively mixed with other channels in both coarse and fine timescales, such that channel-wise features can be better captured.

To this end, we propose MTM, a **m**ulti-scale **t**oken **m**ixing transformer for IMTS classification that effectively integrates the masked concat pooling and the token mixing mechanism. We carry out extensive experiments on 3 real-world datasets with various experiment settings, and compare MTM with state-of-the-art methods. The results show that MTM consistently achieves the best performance on all the benchmarks, with significant improvements of up to 3.8% in classification AUPRC.

To summarize, we make the following contributions:

- *A thorough investigation* into the channel-wise asynchrony in IMTS and how it impacts channel-wise modeling of existing deep learning models for IMTS.
- *A masked concat pooling method* that down-samples IMTS to mitigate its channel-wise asynchrony and improves the effectiveness of channel-wise attention, which is designed based on a comprehensive comparison of different down-sampling methods on IMTS.
- *A channel-wise token mixing mechanism* to further enhance the channel-wise feature learning of IMTS in all timescales.
- *A multi-scale token mixing Transformer model MTM* that integrates the pooling and token mixing modules for the classification of IMTS, with extensive experiments to validate its effectiveness.

The remainder of this paper is organized as follows: We first review related works in Section 2, and formally define the problem we study in Section 3. Afterwards, we elaborate on the design of MTM and its modules in Section 4. We discuss the experimental results in Section 5 and conclude in Section 6.

2 Related Works

In this section, we review related works below from two aspects. First, we summarize and compare prior studies on IMTS thoroughly in Section 2.1. Second, although token down-sampling has not been applied for the analysis of IMTS before, we also list relevant works analyzing regular data in the literature, and discuss the down-sampling for IMTS in 2.2.

2.1 Modeling Irregular Multivariate Time Series

IMTS is prevalent in various real-world domains such as healthcare [10], climate science [33], traffic [39], biology [29], and astronomy [30]. Despite the recent success of deep learning in the analytics of regular MTS [20, 23, 24, 43, 48], how to tackle the irregularities in IMTS with deep learning still remains a challenging problem [35]. Earliest approaches [11, 14, 16, 18, 34, 40, 41, 46] basically rely on data imputation to preprocess the time series to be regular and leverage general time series models. Nevertheless, this will inevitably introduce bias, artifacts, or cause information loss to the dataset, thus impacting the analysis of the data [35, 47].

To address this problem, efforts have been made to directly model IMTS without imputation. Recent works try to adapt deep sequence models to deal with the temporal irregularity of IMTS, such as introducing the time interval information into the state transition of Recurrent Neural Networks [4, 19, 37], learning neural ordinary differential equations along the temporal dimension [2, 5–7, 28, 31, 32], and encoding the irregular time for attention mechanism [15, 36]. However, most of these methods still cannot handle channel-wise asynchrony. In light of this, some Transformer-based methods [38, 42, 44] propose to tokenize each single observation in an IMTS by encoding the corresponding time and channel information together with the observed value, and use attention modules to capture their pair-wise correlations. Since a global attention over all tokens is prohibitive considering the quadratic complexity of self-attention, these models generally compute the attention along the temporal and channel dimensions separately. However, the lack of synchronized observations in IMTS limits the ability of channel-wise attention to learn channel-wise correlations, which leads to poor channel-wise modeling of these methods.

There are only few discussion on the channel-wise modeling problem of IMTS. t-PatchGNN [45] proposes a patching approach to transform the IMTS into uniform and aligned temporal patches to avoid the irregularity. Despite it mitigates the asynchrony to some extent, the patching is applied only once at the beginning of their model, where multi-scale down-sampling is needed for better channel-wise modeling. Warpformer [44] is the first multi-scale model for IMTS, with a learnable irregular warping approach to project IMTS into multiple timescales. However, this approach does not always help align different channels. In some cases, it even exacerbates the asynchrony by projecting originally aligned observations apart. ViTST [12] converts IMTS into images and utilizes Vision Transformer with multi-scale modeling abilities for classification. Nonetheless, the conversion is complicated and time-consuming. Moreover, the computation overhead is much larger than other methods. In summary, all these methods are designed to improve the effectiveness of channel-wise learning on coarser timescales. Besides, they all rely on imputed data to some extent, which hampers their modeling abilities.

In contrast, our MTM addresses the channel-wise asynchrony in two aspects, i.e., improving the existing channel-wise learning module by gradually down-sampling the data to coarser timescales, and introducing a new channel-wise feature learning mechanism to enhance channel-wise modeling in all timescales. As shall be demonstrated in Section 5, our proposed MTM shows significantly better performance than existing methods, without the need of any data imputation.

2.2 Token Down-Sampling in Transformers

Token down-sampling can be broadly categorized into the structured down-sampling and the unstructured down-sampling based on how they partition and reduce the target dimension. Structured down-sampling methods, such as max pooling and average pooling, partition the dimension into fixed-sized patches, with reduction operation performed within each patch. On the contrary, unstructured down-sampling methods offer a more flexible approach by adapting the partition [9]. Recent methods [3, 13, 17, 25] generally rely on a scoring function, such as the attention scores in the Transformer, to partition the dimension into varying sized patches, which allows the model to balance its focus on regions of interest and potentially leading to improved performance.

Since these methods are mostly studied for dense and regular data modalities such as regular time series and images, they focus more on extracting essential features from a redundancy of data and reducing computational cost. However, the irregularity and sparsity of IMTS make it more critical to retain information during the down-sampling process compared with regular data. In addition, as channel-wise asynchrony has not been attended before, the effectiveness of general down-sampling methods to mitigate the channel-wise asynchrony remains unknown on IMTS. In this work, we identify the value of down-sampling in tackling the channel-wise asynchrony, and conduct a comprehensive comparison of potential down-sampling methods to find that a simple concatenation of max and average temporal pooling, with the missing values masked, is the most effective IMTS down-sampler for our purpose.

3 Problem Definition

An IMTS of T timepoints and C channels is a matrix of chronologically ordered observations $\boldsymbol{X} = [x_{i,j}] \in (\mathbb{R} \cup \{\mathtt{NaN}\})^{T \times C}$ together with its observation time $\boldsymbol{T} = [t_i]$, where $x_{i,j}$ is observation of channel j at time t_i for $i \in [1, T]$ and $j \in [1, C]$. Specifically, the *irregularity* of IMTS refers to that $t_{i+1} - t_i$ may not be constant, and each observation may contain missing channels which makes $x_{i,j} = \mathtt{NaN}$. In this paper, we consider the series-level classification of IMTS that predicts one categorical label for each time series. Given a dataset $\mathcal{D}$ containing sample pairs $(\boldsymbol{X}, \boldsymbol{T}, y)$, where $\boldsymbol{X}$ and $\boldsymbol{T}$ are the input IMTS and its observation time, $y \in \{0, \ldots, M-1\}$ is the categorical label of M classes, the classification problem is to obtain an optimal function $\mathcal{F}(\cdot)$ that maps the input $(\boldsymbol{X}, \boldsymbol{T})$ to its corresponding label as $\hat{y} = \mathcal{F}(\boldsymbol{X}, \boldsymbol{T})$ such that the difference between $\hat{y}$ and the ground truth y is minimized.

4 MTM

In this section, we first overview the architecture and workflow of our proposed MTM in Section 4.1, and then elaborate on the designs of MTM from Sections 4.2 to 4.4.

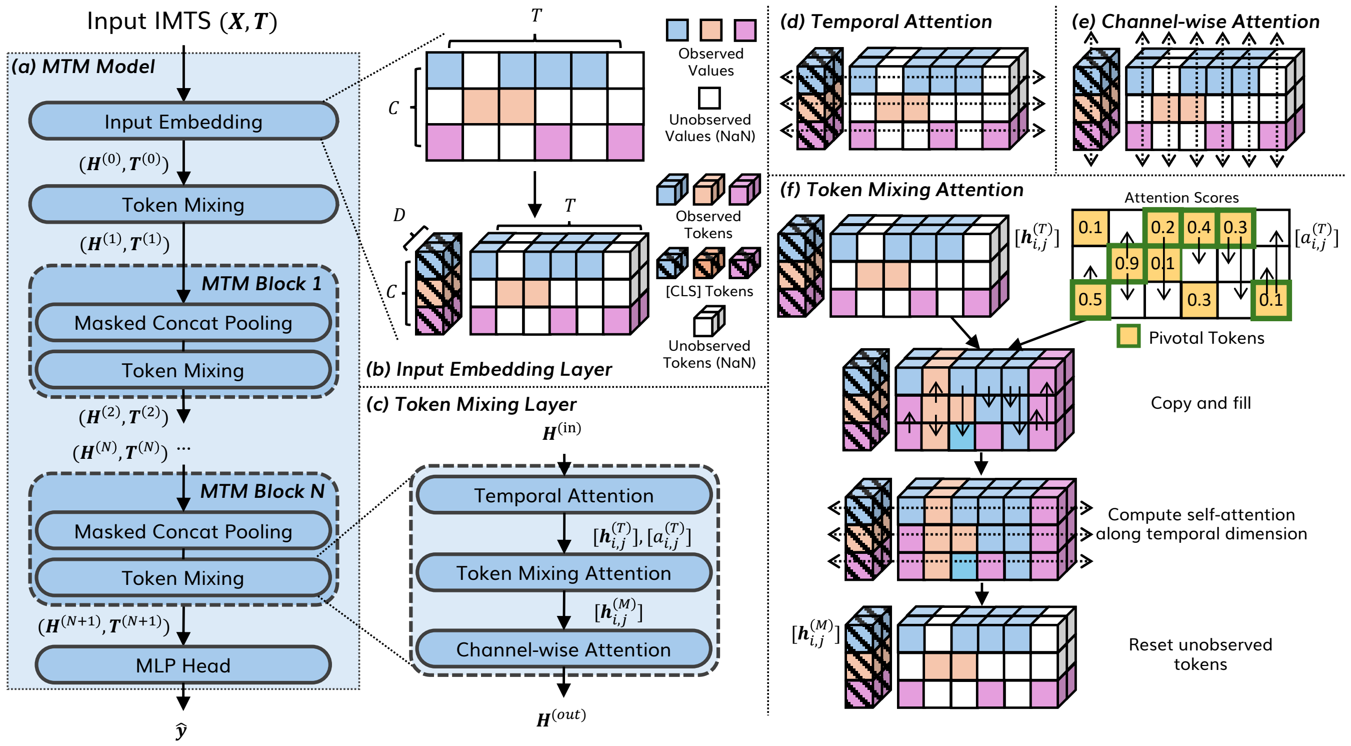

Figure 2: Overview of MTM.

4.1 MTM Overview

Figure 2(a) shows the overall architecture of MTM. MTM comprises an Input Embedding layer, a Token Mixing layer, and a stack of N MTM Blocks. Each MTM Block is further composed of a Masked Concat Pooling layer and a Token Mixing layer. In MTM, the input IMTS $\boldsymbol{X}$ is first embedded by the Input Embedding layer to obtain the high dimensional token embeddings of the observations. Meanwhile, in the Input Embedding layer, we also introduce a set of special [CLS] tokens attached together with the observation tokens. These [CLS] tokens will be used in our token mixing mechanism in the Token Mixing layers, and also be used to predict the final class label at the end. We denote the initial token embeddings as $\boldsymbol{H}^{(0)}$. After the Input Embedding layer, $\boldsymbol{H}^{(0)}$ is first fed into a Token Mixing layer with the output denoted as $\boldsymbol{H}^{(1)}$, followed by a stack of N MTM Blocks for feature extraction. Denote $\boldsymbol{T}^{(1)} = \boldsymbol{T}$, then the n-th MTM Block of MTM takes the token embeddings and observation time $(\boldsymbol{H}^{(n)}, \boldsymbol{T}^{(n)})$ as input, down-samples the data and mixes the feature to get $(\boldsymbol{H}^{(n+1)}, \boldsymbol{T}^{(n+1)})$. The output of the last MTM Block in MTM is $(\boldsymbol{H}^{(N+1)}, \boldsymbol{T}^{(N+1)})$. We extract the embeddings of the [CLS] tokens from $\boldsymbol{H}^{(N+1)}$ and compute the max pooling of these tokens, followed by an MLP layer to get the class label prediction $\hat{y}$.

4.2 Input Embedding

MTM treats each single observation in an IMTS as a token, and employs the attention mechanism to extract features from the tokens for classification. In the Input Embedding layer of MTM shown as Figure 2(b), we map each observation in the input IMTS $\boldsymbol{X}$ to a high dimensional token embedding $Z = [z_{i,j}] \in (\mathbb{R} \cup \{\text{NaN}\})^{T \times C \times D}$ by considering the observed value, observation time, and channel of each observation, with

$$z_{i,j} = x_{i,j} \cdot \boldsymbol{e}_j + \text{PE}(t_i), (i \in [1, T], j \in [1, C], x_{i,j} \neq \text{NaN}), \quad (1)$$

where $\boldsymbol{e}_j \in \mathbb{R}^D$ is the trainable embedding vector for the j-th channel, D is the size of the embedding dimension, and $\text{PE}(\cdot)$ is the sinusoidal positional encoding function of D dimensions. To facilitate the computation in the Token Mixing layers, we introduce a tailor-made [CLS] token for each of the C channels. The embedding of the [CLS] tokens is randomly initialized as D dimensions and trainable. We denote the [CLS] tokens as $\boldsymbol{S} = [\boldsymbol{s}_j] \in \mathbb{R}^{C \times D}$. We concatenate the [CLS] tokens together with the observation tokens to obtain

$$\boldsymbol{H}^{(0)} = [S||Z] = [\boldsymbol{h}_{i,j}^{(0)}], (i \in [0, T], j \in [1, C]), \quad (2)$$

where $\boldsymbol{h}_{0,j}^{(0)}$ denotes the [CLS] token for channel j.

4.3 Masked Concat Pooling Layer

To mitigate the asynchrony between channels of IMTS, in MTM we propose to down-sample the data along the temporal dimension by merging temporally adjacent observations, such that tokens can be better aligned along the channel dimension for the attention mechanism to effectively learn the channel-wise features. In MTM, we employ the Masked Concat Pooling layers to perform the down-sampling as shown in Figure 3. Each Masked Concat Pooling layer is associated with a down-sampling rate R. In each layer we first partition the original time span into non-overlapping patches of

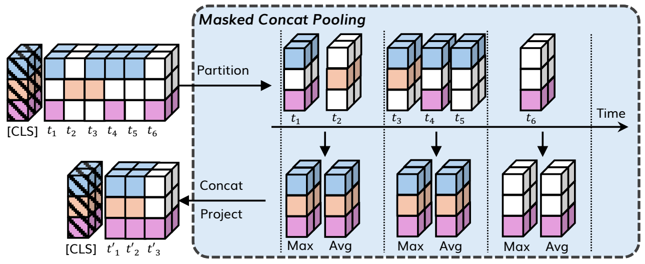

Figure 3: An example of IMTS down-sampling by masked concat pooling.

size R, then compute the max and average of token embeddings within each patch. Timepoints with no observations are masked out and will not be computed. We then concatenate the max and average token embeddings of each patch, and use a trainable linear projection to map the embeddings of size $2D$ back to D. The projected embeddings for the corresponding timepoints are used in the down-sampled timescale. Last, we update the observation time of the tokens accordingly to match the down-sampled timescale, and fed the obtained token embeddings and their observation time to next layers for further processing.

4.4 Token Mixing Layer

The channel-wise asynchrony of IMTS makes the channel-wise attention in existing Transformer models less effective, especially without down-sampling. To this end, we propose Token Mixing layer in MTM by introducing a novel token mixing mechanism that share important information of one channel across other channels, as it is more likely to be important for other channels.

As is shown in Figure 2(c), a Token Mixing layer is composed of a Temporal Attention module, a Token Mixing Attention module, and a Channel-wise Attention module. Here we omit the layer and block index in the notations for simplicity, and denote the input token embeddings as $\boldsymbol{H}^{(in)} = [\boldsymbol{h}_{i,j}^{(in)}], (i \in [0,T], j \in [1,C])$, where $[\boldsymbol{h}_{0,:}^{(in)}]$ are embeddings of the [CLS] tokens, and $[\boldsymbol{h}_{1:,:}^{(in)}]$ are embeddings of the observation tokens. The Temporal Attention module takes $\boldsymbol{H}^{(in)}$ as input, and compute the temporal attention by treating tokens from each channel as a sequence (Figure 2(d)). Taking the j-th channel as an example, we compute the pairwise attention scores $[a_{i,k}^{(T)}]$ of sequence $[\boldsymbol{h}_{0,j}^{(in)}, \dots, \boldsymbol{h}_{T,j}^{(in)}]$, and update the token embeddings as

$$a_{i,k}^{(T)} = \frac{(\boldsymbol{W}_Q^{(T)} \boldsymbol{h}_{i,j}^{(in)})(\boldsymbol{W}_K^{(T)} \boldsymbol{h}_{k,j}^{(in)})^T}{\sqrt{D}}, \quad \boldsymbol{h}_{i,j}^{(T)} = \sum_k \text{Softmax}(a_{i,k}^{(T)}) \cdot (\boldsymbol{W}_V^{(T)} \boldsymbol{h}_{k,j}^{(in)}), \tag{3}$$

where $\boldsymbol{W}_Q^{(T)}$, $\boldsymbol{W}_K^{(T)}$, and $\boldsymbol{W}_V^{(T)}$ are learnable parameters defined on $\mathbb{R}^{D \times D}$.

According to Equation (3), in this step, each [CLS] token embedding will be updated as a linear combination of all tokens from the same channel, with the combination weights being the attention scores in $[a_{i,k}^{(T)}]$. Considering that the embeddings of [CLS] tokens will be used to predict the class label in the end, the model will learn to aggregate important information into the [CLS] tokens, which naturally makes the attention scores of the [CLS] tokens a good indicator of the importance of observation tokens [22]. Therefore, we propose to use the attention score of the [CLS] tokens with respect to the observation tokens to identify the *pivotal tokens*. We compare such attention scores for all channels within each timepoint, and define the token with the largest attention score as the pivotal token for that timepoint.

The pivotal tokens have been well mixed within their original channels by the Temporal Attention module. However, the lack of synchronized observations along the channel dimension makes it hard for them to be mixed with tokens in other channels. Hence, we propose to copy the pivotal tokens to fill the missing tokens in other channels of the same timepoint, such that the pivotal tokens can be mixed to other channels *even without synchronized tokens* with another temporal attention (Figure 2(f)). We conduct the copy-and-fill operation on the token embeddings $[\boldsymbol{h}_{i,j}^{(T)}]$, and denote the output as $[\boldsymbol{r}_{i,j}]$. Taking the j-th channel as an example, we compute their pairwise attention scores as

$$a_{i,k}^{(M)} = \frac{(\boldsymbol{W}_Q^{(M)} \boldsymbol{r}_{i,j})(\boldsymbol{W}_K^{(M)} \boldsymbol{r}_{k,j})^T}{\sqrt{D}}, \tag{4}$$

where $\boldsymbol{W}_Q^{(M)}$, $\boldsymbol{W}_K^{(M)}$, and $\boldsymbol{W}_V^{(M)}$ are learnable parameters defined on $\mathbb{R}^{D \times D}$. To avoid the original information of less observed channels being overwhelmed by the pivotal tokens, we down-weight the pivotal tokens' attention score by the length of the IMTS as

$$\beta_{i,j} = \begin{cases} a_{i,j}^{(M)}/T, & \boldsymbol{h}_{i,j}^{(T)} \text{ is NaN} \\ a_{i,j}^{(M)}, & \text{otherwise} \end{cases}, \tag{5}$$

and update the token embeddings as

$$\boldsymbol{r}'_{i,j} = \sum_k \text{Softmax}(\beta_{i,k}) \cdot (\boldsymbol{W}_V^{(M)} \boldsymbol{r}_{k,j}). \tag{6}$$

After this process, we reset the missing tokens in $[\boldsymbol{r}'_{i,j}]$ as NaN to keep the original sampling pattern *unchanged*, and denote the output as $[\boldsymbol{h}_{i,j}^{(M)}]$. We refer to this process as the Token Mixing Attention.

The Channel-wise Attention module takes $[\boldsymbol{h}_{i,j}^{(M)}]$ as input, and compute the channel wise attention by treating tokens from each timepoint as a sequence. Taking the i-th timepoint as an example, we compute the pairwise attention scores of $[\boldsymbol{h}_{i,1}^{(M)}, \dots, \boldsymbol{h}_{i,C}^{(M)}]$, and update the token embeddings as

$$a_{j,k}^{(C)} = \frac{(\boldsymbol{W}_Q^{(C)} \boldsymbol{h}_{i,j}^{(M)})(\boldsymbol{W}_K^{(C)} \boldsymbol{h}_{i,k}^{(M)})^T}{\sqrt{D}}, \quad \boldsymbol{h}_{i,j}^{(out)} = \sum_k \text{Softmax}(a_{k,j}^{(C)}) \cdot (\boldsymbol{W}_V^{(C)} \boldsymbol{h}_{i,k}^{(M)}), \tag{7}$$

where $\boldsymbol{W}_Q^{(C)}$, $\boldsymbol{W}_K^{(C)}$, and $\boldsymbol{W}_V^{(C)}$ are learnable parameters defined on $\mathbb{R}^{D \times D}$. We denote the final output as $\boldsymbol{H}^{(out)} = [\boldsymbol{h}_{i,j}^{(out)}]$.

Table 2: Statistics of benchmark datasets.

Dataset	#Samples	#Channels	#Classes (positive%)	Sparsity
P12	11988	36	2 (14.2%)	88%
P19	38303	34	2 (4.19%)	95%
PAM	5333	17	8 (N/A)	60%

Note that for both Equations (3) and (7), the unobserved tokens (with embedding as NaNs) are masked out and do not participate in the computation.

5 Illustrative Experimental Results

In order to demonstrate the modeling ability of MTM, we conduct extensive experiments on three well-adopted benchmark datasets for classification of IMTS. In this section, we first describe the experiment settings and implementation details of models in Section 5.1. We then compare the performance of MTM with state-of-the-art methods in Section 5.2. We further study the modeling abilities of MTM for data with different degrees of channel-wise asynchrony in Section 5.3, followed by an ablation study on our proposed key modules in Section 5.4. To gain more insight into the design of MTM, we compare performance of MTM with different potential down-sampling methods in Section 5.5, and analyze key hyperparameters of MTM in 5.6. Lastly, we empirically study the efficiency of MTM by comparing the number of model parameters and running speed with other baselines in Section 5.7.

5.1 Experiment Settings

In this work, we experiment on three widely used real-world healthcare and human activity IMTS datasets. Among them, the P12 [8] and P19 [27] datasets contain electronic health records as IMTS, collected from patients in their hospital stays. The objectives of analyzing these two datasets are to predict in-hospital mortality and the occurrence of sepsis, respectively. The PAM dataset [26] contains IMTS that measure daily living activities of the subjects with motion sensors. The objective is to predict from each IMTS a categorical label that indicates one out of eight activities of the subject.

The P12 and P19 datasets are highly biased regarding the target labels. Therefore, we compute the area under the receiver operating characteristic curve (AUROC) and the area under the precision-recall curve (AUPRC) of the prediction result for evaluation. For the PAM dataset, we compare the accuracy, precision, recall, and F1-score of the prediction result for evaluation. We follow previous studies [12, 47] to preprocess the data, and split each dataset into 5 subsets. Each subset is further split by 80%, 10%, and 10% for training, validation, and testing, respectively. We apply the same fixed split provided by [47] for MTM and all baselines. For each dataset, we experiment with the models on the 5 subsets, and report the mean and standard deviation of their test results. Information of the processed datasets is summarized in Table 2.

We mainly compare our proposed MTM with seven state-of-the-art methods, including Raindrop [47], ContiFormer [6], Coformer [38], Warpformer [44], and ViTST [12], which are the best-performing methods for IMTS classification reported in recent studies. We also compare with GraFITi [42] and t-PatchGNN [45], which are competitive methods for IMTS forecasting, by adapting them for classification tasks.

In the experiments, we implement MTM and all the baseline methods with PyTorch [21]. For MTM we perform grid search to obtain the best combination of hyperparameters, including the hidden dimension, number of layers, dropout rate, learning rate, and the down-sampling rate. For the baseline methods, we follow the implementation provided by the original papers. All experiments are done on an experiment platform of an NVIDIA GeForce RTX 4090 with 24 GB GPU memory.

5.2 Comparison With State-Of-The-Art

The full results of the experiment are shown in Table 3. In this table, we also show results of older baselines reported in past papers [12, 47]. MTM achieves the best performance on all evaluation metrics over the three datasets. Among the baselines, GraFITi is based on a bipartite graph representation of IMTS, which is the only method among the baselines that does not rely on any imputed values. Therefore, it has a more satisfactory performance than other baselines even though it is originally designed for forecasting. However, GraFITi does not consider the channel-wise asynchrony, thus still suffering from poor channel-wise modeling. Meanwhile, despite that Warpformer considers multi-scale modeling of IMTS, its warping-based down-sampling approach does not always mitigate the channel asynchrony, thus cannot improve its channel-wise modeling ability, which leads to a suboptimal performance. t-PatchGNN avoid the irregularity of IMTS by transforming the data into uniform and aligned temporal patches. Despite its patching approach mitigates the asynchrony to some extent, the patching is applied only once at the beginning of their model, which is not enough to deal with different degrees of asynchrony. Its suboptimal performance also indicates that multi-scale down-sampling is needed for better channel-wise modeling. Another strong baseline is ViTST, which converts IMTS data into line-graph images, and utilizes ViTs to classify these images. Despite its simplicity in directly applying the existing ViTs, we notice that its conversion from IMTS data to images can be very complicated and time-consuming. Any changes to the raw IMTS data or the image generation settings may require a re-generation of all the images, leading to its inflexibility in applications. Its performance also relies on its backbone ViT which is pretrained on extra large image datasets. It may not perform very well if we train it from scratch like other methods in our experiments.

In comparison, MTM addresses the poor channel-wise modeling problem of existing methods with a carefully-designed down-sampling module to mitigate the asynchrony, and a token mixing mechanism to enhance channel-wise feature extraction, which gives MTM better channel-wise modeling ability, and makes MTM significantly more powerful than other baselines. Meanwhile, MTM directly works on the IMTS data and does not rely on any imputed

Table 3: Classification results. The best results are in bold and the second bests are underlined.

Methods	P12		P19		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1-Score
GRU-D (2018)	81.9 ±2.1	46.1 ±4.7	83.9 ±1.7	46.9 ±2.1	83.3 ±1.6	84.6 ±1.2	85.2 ±1.6	84.8 ±1.2
SeFT (2020)	73.9 ±2.5	31.1 ±4.1	81.2 ±2.3	41.9 ±3.1	67.1 ±2.2	70.0 ±2.4	68.2 ±1.5	68.5 ±1.8
mTAND (2020)	84.2 ±0.8	48.2 ±3.4	84.4 ±1.3	50.6 ±2.0	74.6 ±4.3	74.3 ±4.0	79.5 ±2.8	76.8 ±3.4
IP-Net (2018)	82.6 ±1.4	47.6 ±3.1	84.6 ±1.3	38.1 ±3.7	74.3 ±3.8	75.6 ±2.1	77.9 ±2.2	76.6 ±2.8
DGM2-O (2021)	84.4 ±1.6	47.3 ±3.6	86.7 ±3.4	44.7 ±11.7	82.4 ±2.3	85.2 ±1.2	83.9 ±2.3	84.3 ±1.8
MTGNN (2020)	74.4 ±6.7	35.5 ±6.0	81.9 ±6.2	39.9 ±8.9	83.4 ±1.9	85.2 ±1.7	86.1 ±1.9	85.9 ±2.4
Raindrop (2022)	82.8 ±1.7	44.0 ±3.0	87.0 ±2.3	51.8 ±5.5	88.5 ±1.5	89.9 ±1.5	89.9 ±0.6	89.8 ±1.0
ContiFormer (2023)	82.1 ±2.2	44.8 ±3.5	84.4 ±2.1	50.4 ±4.3	85.2 ±2.8	86.8 ±2.7	86.7 ±2.6	86.3 ±2.7
Coformer (2023)	85.8 ±1.9	52.4 ±4.3	89.2 ±1.8	57.3 ±3.3	91.2 ±0.6	92.4 ±0.7	93.7 ±0.7	92.8 ±0.5
Warpformer (2023)	86.5 ±1.2	54.3 ±2.7	88.7 ±2.0	52.4 ±4.5	94.2 ±2.3	95.1 ±2.2	94.7 ±2.3	94.9 ±2.2
ViTST (2023)	85.1 ±0.8	51.1 ±4.1	89.2 ±2.0	53.1 ±3.4	95.8 ±1.3	96.2 ±1.3	96.1 ±1.1	96.5 ±1.2
t-PatchGNN (2024)	84.5 ±0.9	50.8 ±2.6	87.0 ±1.4	51.5 ±5.2	93.9 ±1.2	94.9 ±0.9	94.8 ±1.3	94.8 ±1.2
GraFITi (2024)	86.6 ±1.1	54.8 ±3.1	89.1 ±2.6	56.6 ±6.3	96.0 ±0.8	96.3 ±0.6	96.0 ±0.8	96.1 ±0.7
MTM (ours)	**88.0 ±1.0**	**58.6 ±4.1**	**90.3 ±2.0**	**58.3 ±5.3**	**97.5 ±0.2**	**97.8 ±0.3**	**97.5 ±0.4**	**97.6 ±0.2**

Table 4: Statistical significance test results.

Dataset	P12		P19		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Acc.	Prec.	Rec.	F1
P-value	0.008	0.007	0.023	0.026	0.003	0.000	0.007	0.000

values, which makes the data processing simple, flexible, and not subjective to extra artifacts.

We conduct statistical significance tests to further justify the improvement of MTM over the baselines. On the first subset of each each dataset, we run MTM and the best baseline for the dataset 5 times with different random seeds, and apply independent samples t-test to examine the significance of the improvement. Table 4 shows the p-values from the tests, which represent the statistical probabilities if there were no difference between the performances of MTM and the baselines. All the p-values in Table 4 are less than 0.05. which indicates that the performance gain to be highly significant.

5.3 Experiments with Varying Sparsity

To further study the modeling abilities of MTM for data with different degrees of channel-wise asynchrony, we randomly mask out a subset of channels in the PAM dataset and compare the performance of different methods. This setting emulates real-world situations where some sensors fail or become unreachable during the data collection period. The PAM dataset is of 60% sparsity. We experiment with 10%-50% data masked out, corresponding to 64%-80% sparsity, which emulates moderate to severe channel-wise asynchrony for our purpose. The masking is applied to the training, validation, and testing sets. We report the average F1-score over the 5 subsets as the result for each method.

As shown in Figures 4, MTM leads the board with the highest F1-scores when 10% to 50% of the channels are masked out. With 50% of the channels masked, MTM can still accomplish the classification task well with an F1-score up to 91%. This highlights the excellent capability of MTM to model IMTS data with different degrees of channel-wise asynchrony. On the other hand, we observe that the performance of Raindrop and ContiFormer flops faster than other methods when the masking ratio increases. This is because they both heavily rely on data imputation to handle the channel-wise irregularity of IMTS. Specifically, ContiFormer requires to impute missing channels in each observation to get fixed-sized vectors as its input. Raindrop learns to generate the representation of the missing channels from the observed channels. Their performance is thus more severely affected when the sparsity of the data increases.

5.4 Ablation Study

We strongly believe that the good performance of MTM is rooted in our proposed multi-scale down-sampling and token mixing mechanisms. To validate the efficacy of these novel designs, we implement variants of MTM by removing the Masked Concat Pooling layers, the use of `[CLS]` tokens, and the Token Mixing Attention modules, respectively. We carry out the experiments for these MTM variants, and report the results in Table 5. All the three variants show inferior performance to the full MTM model in the classification tasks. We note that by using the token mixing mechanism in MTM, the AUROC increases up to 2% on the P19 dataset. This improvement is much larger than that of the other two datasets, and also larger than the improvement brought by the down-sampling module. This is because the P19 dataset is the most sparse dataset among the three datasets. The severe lack of synchronized observations even makes the down-sampling module less helpful. On the other hand, the down-sampling modules works better than token-mixing on the PAM dataset, which is because PAM is the least sparse dataset. By comparing the variants with and without the `[CLS]` tokens, we discover that the introduction of the `[CLS]` tokens alone can also bring some performance gain to MTM even without the full token mixing mechanism. These results provide strong evidence

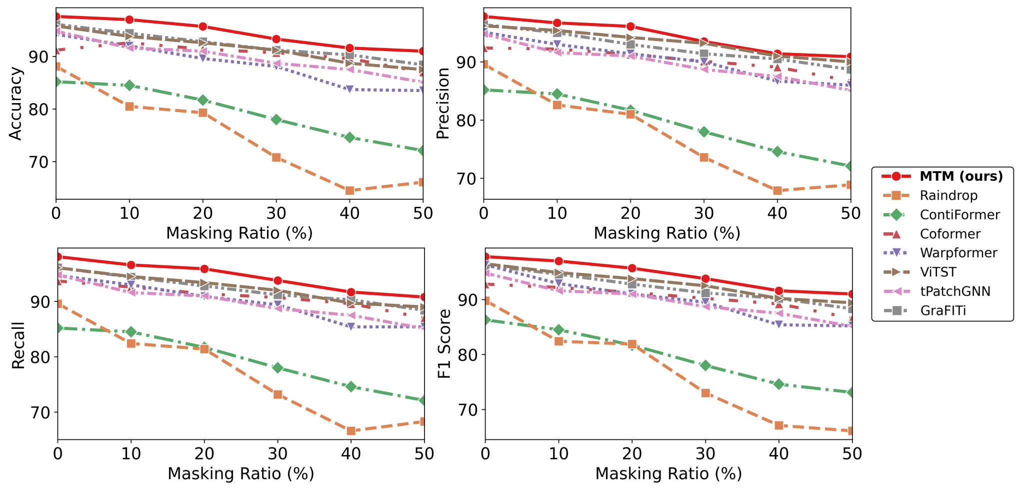

Figure 4: Performance with varying masking ratio.

Table 5: Ablation study results.

MTM Variants			P12		P19		PAM			
Pooling	[CLS]	Mixing	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1-Score
N	N	N	86.1 ±1.6	52.6 ±6.6	87.3 ±3.2	51.1 ±5.5	95.6 ±1.3	95.9 ±1.3	95.6 ±1.1	95.8 ±1.2
Y	N	N	86.9 ±1.3	56.1 ±3.7	87.5 ±2.9	52.0 ±5.6	96.9 ±0.6	97.2 ±0.3	97.2 ±0.5	96.9 ±0.4
Y	Y	N	87.1 ±1.0	57.7 ±4.1	88.4 ±2.5	55.3 ±3.7	96.2 ±0.9	96.4 ±0.9	96.2 ±0.9	96.3 ±0.9
N	Y	Y	87.2 ±1.2	56.9 ±2.9	89.3 ±2.6	56.8 ±4.8	96.1 ±0.6	96.7 ±0.6	96.1 ±0.6	96.4 ±0.6
Y	Y	Y	88.0 ±1.0	58.6 ±4.1	90.3 ±2.0	58.3 ±5.3	97.5 ±0.2	97.8 ±0.3	97.5 ±0.4	97.6 ±0.2

to demonstrate the effectiveness of our proposed masked concat pooling and token mixing mechanism.

5.5 Comparing Down-Sampling Methods

As there is a lack of related study on the down-sampling of IMTS, we implement four representative down-sampling methods on MTM. The three structured methods include max pooling, average pooling, and a concatenation of max and average pooling. The one unstructured method base on the attention score from its previous layer to select the top scored timepoints, and merge the observations to its nearest selected timepoints. We experiment these methods on our datasets, and show the results in Table 6.

From the results we observe that the masked concat pooling, which is the method we use in our MTM, achieves the best performance among its structured and unstructured counterparts. We think the reason behind is that the concat pooling can maximally retain information from the data. In contrast to the regular dense data where down-sampling is applied for extracting essential information from a redundancy and achieving computational efficiency, IMTS is already very sparse, and we employ down-sampling to mitigate the channel-wise asynchrony, which makes the retaining of information more important than reducing them. This is why the concat pooling is consistently better on all the datasets despite max and average pooling are more frequently adopted for image and regular time series. On the other hand, we observe that the attention-based unstructured pooling method has inferior performance comparing the structured methods, despite its success in computer vision researches. Besides the ability of retaining information when reducing the dimension, we think another reason is that its irregular partitioning of the time dimension adds up to the difficulties for the model to learn the temporal correlations in the down-sampled timescale. We think this is also the reason why the unstructured warping-based multi-scale module in the Warpformer baseline does not actually outperform the structured pooling method we propose in MTM.

5.6 Hyperparameter Study

To gain deeper insights into our proposed MTM with different setups, we compare the performance of MTM with different down-sampling rate and number of layers on the P12 and P19 datasets. The P12 and P19 datasets are of severe channel-wise asynchrony, which is good for us to understand the sensitivity of the related hyperparameters. The results are shown in Figure 5. When we fix the number of Token Mixing layers in MTM, and experiment with different down-sampling rates, we find that the performance is suboptimal when we set the down-sampling rate too large or too small for both datasets. The best performing values are 3 for the P12 dataset and 2 for the P19 dataset. When we fix the down-sampling rates to each dataset's optimal, and experiment with varying number of layers, we find that, with increasing number of layers, the

Table 6: Performance of MTM with different down-sampling methods.

Pooling Method	P12 AUROC	P12 AUPRC	P19 AUROC	P19 AUPRC	PAM Accuracy	PAM Precision	PAM Recall	PAM F1-Score
Max Pooling	87.5 ±1.2	57.3 ±3.8	89.8 ±2.4	56.8 ±3.3	97.0 ±0.4	97.4 ±0.5	97.0 ±0.4	97.2 ±0.4
Avg Pooling	87.4 ±1.7	57.2 ±5.2	89.5 ±2.2	55.9 ±3.7	96.5 ±0.6	96.8 ±0.8	96.5 ±0.6	96.6 ±0.7
Attention Scoring	87.2 ±1.3	56.2 ±3.5	86.5 ±3.2	49.9 ±5.4	96.4 ±0.9	96.8 ±0.6	96.4 ±0.9	96.6 ±0.7
Concat(max, avg)	**88.0** ±1.0	**58.6** ±4.1	**90.3** ±2.0	**58.3** ±5.3	**97.5** ±0.2	**97.8** ±0.3	**97.5** ±0.4	**97.6** ±0.2

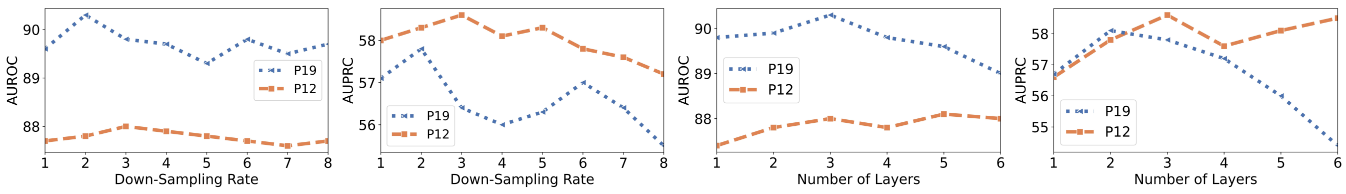

Figure 5: Performance with different hyperparameter setups.

performance of MTM stabilizes around the optimal value, while it flops on the P19 datasets. These findings shed light on the importance of adapting these hyperparameters to the specific properties of the datasets.

5.7 Model Efficiency

Our proposed MTM mainly comprises attention operations. It does not require extra computation-intensive operations, and operations in our proposed masked concat pooling and token mixing mechanism are mostly parallelizable to enjoy accelerated computation on modern GPUs. Moreover, our multi-scale down-sampling approach by masked concat pooling allows MTM to compute the attention over less tokens and further reduces the computational overhead. These characteristics enable MTM to remain efficient while being very powerful. In this section, we empirically study the computation efficiency of our proposed MTM by comparing the number of model parameters and training time consumption of different models on the PAM dataset. From the results shown in Figure 6, we can see that MTM achieves the best performance while also maintaining a small model size and short training time. Specifically, MTM runs much faster than the other two methods Warpformer and Coformer with similar Transformer-based designs. On the other hand, despite achieving a good classification performance, we can see that ViTST is significantly larger in model size and slower in training time than most methods. We also note that the Contiformer consumes significantly longer training time than all other methods because of its neural ODE-based designs. Therefore, we are not showing it in Figure 6.

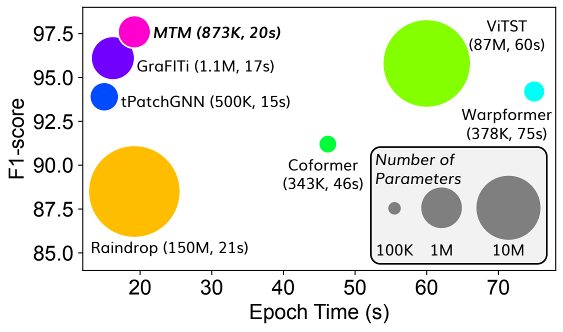

Figure 6: Model efficiency comparison.

6 Conclusion

In this paper, we focus on the classification of IMTS, especially regarding the channel-wise feature extraction. We first point out the ineffectiveness of the channel-wise attention in existing Transformers to be caused by the severe channel-wise asynchrony of IMTS data, and propose to overcome this limitation in two aspects, i.e., improving the effectiveness of existing channel-wise learning module with multi-scale modeling, and introducing a new channel-wise learning mechanism. To this end, we propose MTM, a multi-scale token mixing transformer. We first propose a masked concat pooling in MTM to gradually down-sample IMTS. Moreover, we propose a novel channel-wise token mixing mechanism which learns to proactively choose important tokens from one channel and mix them with other channels. We experiment on three real-world datasets from the healthcare and human action recognition area, which demonstrate the applicability, feasibility, and SOTA performance of our proposed MTM in real-world IMTS classification scenarios, with improvements of up to 3.8% in AUPRC. Meanwhile, we also realize that the key designs of MTM are currently limited to the classification task. An extension of this work to forecasting and other generative tasks of IMTS can be meaningful research directions for future efforts.

Acknowledgments

This work was supported, in part, by Research Grants Council Collaborative Research Fund under grant number C1045-23G, and the Guangdong Provincial / Zhuhai Key Laboratory of IRADS (2022B1212010006).

References

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

[2] Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. 2021. Neural Flows: Efficient Alternative to Neural ODEs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 21325–21337.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461* (2022).

[4] 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.

[5] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.

[6] Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. 2023. ContiFormer: Continuous-Time Transformer for Irregular Time Series Modeling. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., New Orleans, USA, 47143–47175.

[7] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. 2019. GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[8] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation* 101, 23 (2000), e215–e220. doi:10.1161/01.CIR.101.23.e215

[9] Joakim Bruslund Haurum, Sergio Escalera, Graham W Taylor, and Thomas B Moeslund. 2023. Which tokens to use? investigating token reduction in vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 773–783.

[10] Peter B Jensen, Lars J Jensen, and Søren Brunak. 2012. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics* 13, 6 (2012), 395–405.

[11] Steven Cheng-Xian Li and Benjamin Marlin. 2020. Learning from Irregularly-Sampled Time Series: A Missing Data Perspective. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 5937–5946.

[12] Zekun Li, Shiyang Li, and Xifeng Yan. 2023. Time Series as Images: Vision Transformer for Irregularly Sampled Time Series. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 49187–49204. https://proceedings.neurips.cc/paper_files/paper/2023/file/9a17c1eb808cf012065e9db47b7ca80d-Paper-Conference.pdf

[13] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations. arXiv:2202.07800 [cs.CV]

[14] Zachary C Lipton, David Kale, and Randall Wetzel. 2016. Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series. In *Proceedings of the 1st Machine Learning for Healthcare Conference (Proceedings of Machine Learning Research, Vol. 56)*. PMLR, Northeastern University, Boston, MA, USA, 253–270.

[15] Junyu Luo, Muchao Ye, Cao Xiao, and Fenglong Ma. 2020. HiTANet: Hierarchical Time-Aware Attention Networks for Risk Prediction on Electronic Health Records. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 647–656. doi:10.1145/3394486.3403107

[16] Liantao Ma, Junyi Gao, Yasha Wang, Chaohe Zhang, Jiangtao Wang, Wenjie Ruan, Wen Tang, Xin Gao, and Xinyu Ma. 2020. AdaCare: Explainable Clinical Health Status Representation Learning via Scale-Adaptive Feature Extraction and Recalibration. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 825–832. doi:10.1609/aaai.v34i01.5427

[17] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. 2023. Token pooling in vision transformers for image classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 12–21.

[18] Benjamin M. Marlin, David C. Kale, Robinder G. Khemani, and Randall C. Wetzel. 2012. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium* (Miami, Florida, USA) *(IHI '12)*. Association for Computing Machinery, New York, NY, USA, 389–398. doi:10.1145/2110363.2110408

[19] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates, Inc., Barcelona,Spain.

[20] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. arXiv:2211.14730 [cs.LG] https://arxiv.org/abs/2211.14730

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[22] Yao Qiang, Deng Pan, Chengyin Li, Xin Li, Rhongho Jang, and Dongxiao Zhu. 2022. AttCAT: Explaining Transformers via Attentive Class Activation Tokens. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 5052–5064.

[23] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. In *Proc. VLDB Endow.* 2363–2377.

[24] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. 2025. DUET: Dual Clustering Enhanced Multivariate Time Series Forecasting. arXiv:2412.10859 [cs.LG] https://arxiv.org/abs/2412.10859

[25] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems* 34 (2021), 13937–13949.

[26] Attila Reiss and Didier Stricker. 2012. Introducing a New Benchmarked Dataset for Activity Monitoring. In *2012 16th International Symposium on Wearable Computers*. 108–109. doi:10.1109/ISWC.2012.13

[27] Matthew A Reyna, Christopher S Josef, Russell Jeter, Supreeth P Shashikumar, M Brandon Westover, Shamim Nemati, Gari D Clifford, and Ashish Sharma. 2020. Early prediction of sepsis from clinical data: the PhysioNet/Computing in Cardiology Challenge 2019. *Critical care medicine* 48, 2 (2020), 210–217.

[28] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., Vancouver, Canada.

[29] T Ruf. 1999. The Lomb-Scargle periodogram in biological rhythm research: analysis of incomplete and unequally spaced time-series. *Biological Rhythm Research* 30, 2 (1999), 178–201.

[30] Jeffrey D Scargle. 1982. Studies in astronomical time series analysis. II-Statistical aspects of spectral analysis of unevenly spaced data. *Astrophysical Journal, Part 1.* 263 (1982), 835–853.

[31] Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. 2022. Modeling Irregular Time Series with Continuous Recurrent Units. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 19388–19405.

[32] Randolf Scholz, Stefan Born, Nghia Duong-Trung, Mariano Nicolas Cruz-Bournazou, and Lars Schmidt-Thieme. 2023. Latent Linear ODEs with Neural Kalman Filtering for Irregular Time Series Forecasting.

[33] Michael Schulz and Karl Stattegger. 1997. SPECTRUM: Spectral analysis of unevenly spaced paleoclimatic time series. *Computers & Geosciences* 23, 9 (1997), 929–945.

[34] Siyuan Shan, Yang Li, and Junier B. Oliva. 2023. NRTSI: Non-Recurrent Time Series Imputation. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10095054

[35] Satya Narayan Shukla and Benjamin M. Marlin. 2021. A Survey on Principles, Models and Methods for Learning from Irregularly Sampled Time Series. arXiv:2012.00168 [cs.LG]

[36] Sindhu Tipirneni and Chandan K. Reddy. 2022. Self-Supervised Transformer for Sparse and Irregularly Sampled Multivariate Clinical Time-Series. *ACM Trans. Knowl. Discov. Data* 16, 6, Article 105 (jul 2022), 17 pages. doi:10.1145/3516367

[37] Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, and Wendell Ela. 2021. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing* 441 (2021), 161–178.

[38] Yuxi Wei, Juntong Peng, Tong He, Chenxin Xu, Jian Zhang, Shirui Pan, and Siheng Chen. 2023. Compatible Transformer for Irregularly Sampled Multivariate Time Series. In *2023 IEEE International Conference on Data Mining (ICDM)*. 1409–1414. doi:10.1109/ICDM58522.2023.00183

[39] Ronghui Xu, Hanyin Cheng, Chenjuan Guo, Hongfan Gao, Jilin Hu, Sean Bin Yang, and Bin Yang. 2025. MM-Path: Multi-modal, Multi-granularity Path Representation Learning – Extended Version. arXiv:2411.18428 [cs.LG] https://arxiv.org/abs/2411.18428

[40] Yanbo Xu, Siddharth Biswal, Shriprasad R. Deshpande, Kevin O. Maher, and Jimeng Sun. 2018. RAIM: Recurrent Attentive and Intensive Model of Multimodal Patient Monitoring Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2565–2573.

doi:10.1145/3219819.3220051

[41] Zhen Xu, David R. So, and Andrew M. Dai. 2021. MUFASA: Multimodal Fusion Architecture Search for Electronic Health Records. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 12 (May 2021), 10532–10540. doi:10.1609/aaai.v35i12.17260

[42] Vijaya Krishna Yalavarthi, Kiran Madhusudhanan, Randolf Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Jawed, Stefan Born, and Lars Schmidt-Thieme. 2024. GraFITi: Graphs for Forecasting Irregularly Sampled Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 15 (Mar. 2024), 16255–16263. doi:10.1609/aaai.v38i15.29560

[43] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Are Transformers Effective for Time Series Forecasting? arXiv:2205.13504 [cs.AI] https://arxiv.org/abs/2205.13504

[44] Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. 2023. Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 3273–3285. doi:10.1145/3580305.3599543

[45] Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. 2024. Irregular Multivariate Time Series Forecasting: A Transformable Patching Graph Neural Networks Approach. In *Forty-first International Conference on Machine Learning*.

[46] Xianli Zhang, Buyue Qian, Yang Li, Yang Liu, Xi Chen, Chong Guan, and Chen Li. 2021. *Learning Robust Patient Representations from Multi-modal Electronic Health Records: A Supervised Deep Learning Approach.* 585–593. doi:10.1137/1.9781611976700.66

[47] Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Graph-Guided Network for Irregularly Sampled Multivariate Time Series. arXiv:2110.05357 [cs.LG]

[48] Shuhan Zhong, Sizhe Song, Weipeng Zhuo, Guanyao Li, Yang Liu, and S-H Gary Chan. 2024. A Multi-Scale Decomposition MLP-Mixer for Time Series Analysis. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1723–1736.

A More Results on Applicability

Our proposed MTM is designed for general IMTS data. It does not rely on any domain-specific properties, which gives it great applicability to different domains. Section 5.2 compares MTM with baseline methods on datasets from the healthcare and human action recognition domains. To further validate the applicability of MTM, we experiment on the SpokenArabicDigits (SAD) dataset [1] from the speech recognition domain. SAD contains 8800 regular multivariate time series of speech signals collected from native Arabic speakers' pronunciations of 10 Arabic digits. Each time series consists of 13 channels representing different feature types. The objective of this dataset is to predict the corresponding digits from the time series data. Since the data is regular, we randomly mask 80% of the observations in each time series to generate highly sparse IMTS. We compare the classification accuracy of MTM with the most competitive baselines from Table 3. As the results shown in Table 7, our MTM also achieves the best performance in comparison with these competitive baselines.

Table 7: Classification accuracy on SpokenArabicDigits.

Method	Acccuracy (%)
Coformer	93.7
Warpformer	89.7
ViTST	94.0
t-PatchGNN	91.8
GraFITi	93.6
MTM (ours)	**94.5**

B More Results on Extremely Sparse IMTS

We further extend our experiments with IMTS of varying sparsity in Section 5.3 to extreme conditions. Table 8 shows performance of MTM and the most competitive baselines on the PAM dataset with 60%-90% data masked out, which corresponds to overall spasity of 84%-96%. By comparison, we observe that the advantage of MTM gradually expands with the increase of data sparsity, which demonstrates the effectiveness of our proposed method in addressing the channel-wise asynchrony problem.

C Visualization

To better understand the behavior of our proposed method, we study two examples of the attention scores used for choosing the pivotal tokens in our proposed token mixing mechanism. The examples are taken from the P12 dataset. For each example we visualize the attention scores $[a_{i,j}^{(T)}]$ from the first and second Token Mixing layers, where the first Token Mixing layer works on the original timescale, and the second layer works on a coarser timescale down-sampled by 4. From the attention maps in Figure 7 we can see that the data at the original timescale severely suffers from channel-wise asynchrony, and a down-sampling process effectively mitigates the problem. On the other hand, tokens with the highest attention scores are chosen as the pivotal tokens and mixed with other channels in the Token Mixing Attention that follows, which further enhances MTM's channel-wise modeling ability to deal with the channel-wise asynchrony of IMTS.

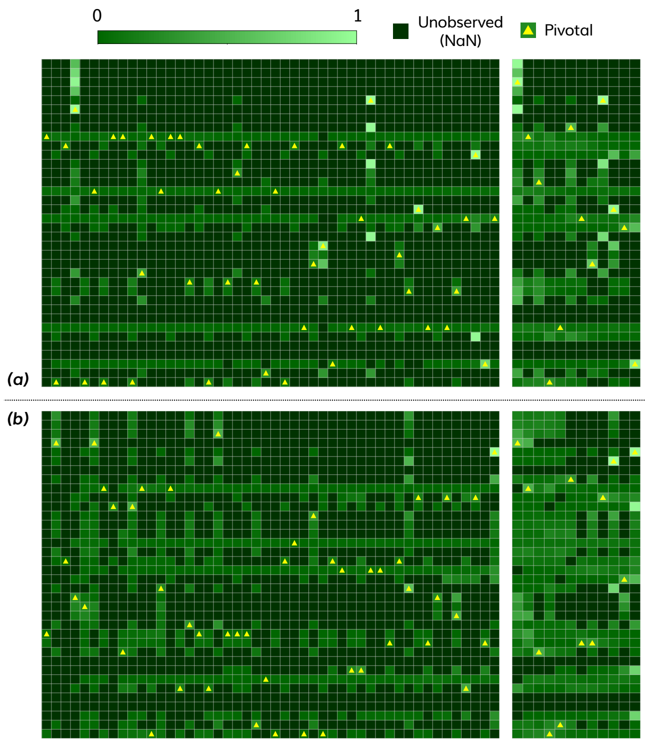

Figure 7: Attention score visualizations.

Table 8: Classification performance on PAM with extremely high masking ratios.

Methods	Accuracy (%)				Precision(%)				Recall(%)				F1-Score(%)			
	60%	70%	80%	90%	60%	70%	80%	90%	60%	70%	80%	90%	60%	70%	80%	90%
Coformer (2023)	85.7	80.5	69.7	60.2	84.7	78.6	69.7	60.2	85.7	80.5	68.5	60.2	85.0	79.2	68.7	59.5
Warpformer (2023)	79.4	73.9	63.5	59.1	82.5	77.2	68.3	62.4	81.5	76.3	64.0	59.6	81.4	76.3	64.9	59.0
ViTST (2023)	86.8	80.5	75.6	69.6	88.7	84.1	78.4	72.7	88.6	82.6	77.7	**70.9**	88.6	83.1	77.8	70.9
tPatchGNN (2024)	81.1	75.5	68.1	59.8	84.7	79.9	72.3	63.0	83.5	77.1	70.2	60.5	83.9	78.1	70.7	61.2
Grafiti (2024)	85.8	80.5	71.0	63.2	85.9	81.0	73.1	64.5	85.8	80.5	71.1	63.2	85.6	80.5	71.4	63.3
MTM (ours)	**88.7**	**85.2**	**77.8**	**70.5**	**88.8**	**85.0**	**79.4**	**72.9**	**88.7**	**85.2**	**77.8**	70.5	**88.7**	**85.0**	**78.2**	**71.3**