COMBATING MISSING VALUES IN MULTIVARIATE TIME SERIES BY LEARNING TO EMBED EACH VALUE AS A TOKEN

Chun-Kai Huang[1], **Yi-Hsien Hsieh**[1], **Shao-Hua Sun**[1,2], **Tung-Hung Su**[3,4], **Jia-Horng Kao**[3,5] **& Che Lin**[1,2,6,7,⋆]

[1] Graduate Institute of Communication Engineering, National Taiwan University (NTU)
[2] Department of Electrical Engineering, NTU
[3] Division of Gastroenterology and Hepatology, NTU Hospital
[4] Hepatitis Research Center, NTU Hospital
[5] Graduate Institute of Clinical Medicine, College of Medicine, NTU
[6] Center for Advanced Computing and Imaging in Biomedicine, NTU
[7] Smart Medicine and Health Informatics Program, NTU

ABSTRACT

Irregular and asynchronous sampled multivariate time series (MTS) data is often filled with missing values. Most existing methods embed features according to their timestamps, requiring imputing missing values. However, imputed values can drastically differ from real values, resulting in inaccurate predictions made based on imputation. To address the issue, we propose a novel concept, "each value as a token (EVAT)," treating each feature value as an independent token, which allows for bypassing imputing missing values. To realize EVAT, we propose scalable numerical embedding, which learns to embed each feature value by automatically discovering the relationship among features. We integrate the proposed embedding method with the Transformer Encoder, yielding the **S**calable n**UM**erical e**M**bedd**I**ng **T**ransformer (SUMMIT), which can produce accurate predictions given MTS with missing values. We induct experiments on three distinct electronic health record datasets with high missing rates. The experimental results verify SUMMIT's efficacy, as it attains superior performance than other models that need imputation.

1 INTRODUCTION

Multivariate time series (MTS) data constitutes a series of full observations registered at isometric timestamps, encompassing a multitude of interconnected variables. Nevertheless, most MTS data are irregular and asynchronously sampled. The irregularity causes the interval between two adjacent timestamps and their corresponding numbers to vary. Not all feature variables are observed for each timestamp, creating data with a high missing rate. Traditionally, people first impute the missing values via statistic-based (Little & Rubin, 2019) or learning-based models (Mattei & Frellsen, 2019; Du et al., 2023; Kim et al., 2023; Zhao et al., 2023) to obtain timestamp embeddings to the models (Hochreiter & Schmidhuber, 1997; Breiman, 2001; Chung et al., 2014; Chen & Guestrin, 2016; Vaswani et al., 2017). In such scenarios, imputation is inevitable. However, the rationale behind imputation is complicated and challenging to justify in some domains, especially in healthcare. Imputations that make sense to physicians may not work well for the learning model, while methods that learn well may not convince medical experts.

To combat missing values in MTS while avoiding imputation, we propose the concept of "**each value as a token** (**EVAT**)." We consider each value as an independent token, like the word tokens (Mikolov et al., 2013) in the natural language process (NLP) tasks. With this concept, we can further view the missing values as the paddings in a sentence and mask them. Moreover, the downstream

⋆Corresponding author: chelin@ntu.edu.tw

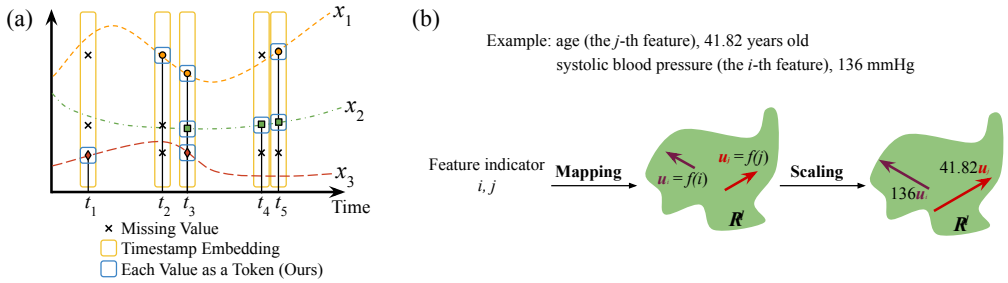

Figure 1: **(a) Embedding Multivariate Time Series Values**: The figure records MTS data with three variables and five timestamps. Colored dots are observed values. The rectangles bound by yellow lines are timestamp embedding which needs imputation to implement. Blue lines bind the proposed EVAT embeddings. Under this scenario, missing values can be neglected. **(b) Scalable Numerical Embedding**: Take "age 41.82 years old" as an example. First, the "age" feature indicator is mapped to the feature embedding $\boldsymbol{u}_j$ via a learnable function f. Second, we scale this feature embedding $\boldsymbol{u}_j$ according to its observed value.

module would have more freedom to interact with the observed variables along both the temporal and feature-wise dimensions. However, there are still several bottlenecks to embedding continuous values. The tokens with the same numerical value but different feature types should be mapped to different embedding, or it would confuse the downstream modules. Traditionally, people try to quantize the numerical values (Gao et al., 2022), but the performance is poor and highly dependent on the quantization resolution.

To implement EVAT and solve the above bottlenecks, we propose **SCAlable Numerical Embedding** (**SCANE**), a novel embedding method that learns to embed both the feature type and quantity simultaneously. We take the Transformer Encoder (Vaswani et al., 2017) with SCANE to build **Scalable nUMerical eMbeddIng Transformer** (**SUMMIT**), followed by fully connected layers to form a complete classifier. With the inherent masking mechanism in the Transformer Encoder and the help of SCANE, we can perfectly mask all missing values. Based on SCANE, SUMMIT is a truly imputation-free model. It distills the information from the observed values without interference from the missing ones, and it can freely interact with variables across temporal and feature-wise dimensions.

We evaluate SUMMIT with five other models, including the SOTA model (GRU-D). The experiments are conducted on three distinct EHR datasets, one for chronic illness prediction and the other two for acute illness prediction. Experiment results show that SUMMIT surpasses all baselines on all datasets with an average AUPRC improvement of 4.2% on these three datasets.

2 Problem Formulation

$\boldsymbol{X} = [X_{i,j}]$, is an $m \times n$ matrix, representing a time series data irregularly and asynchronously sampled and comprising n variables at m timestamps. In this matrix, missing values are denoted by Nan. The sequence of MTS data's timestamps is arranged in ascending order. We denote $X_{i,j}$ as the entry of the j-th feature at the i-th timestamp.

The goal of the EVAT problem is to map each value $X_{i,j}$, which is a scalar, to a vector in a higher dimensional vector space. The assigned vector should represent the information of the feature type and quantity simultaneously.

3 SUMMIT: Scalable Numerical Embedding Transformer

Scalable Numerical Embedding. To implement EVAT, we propose *Scalable Numerical Embedding* (SCANE). To carry the feature type and quantity information, it needs both the feature indicator and a way to represent the feature quantity. The mask $\boldsymbol{M} = [M_{i,j}]$ is the mask to indicate if the corresponding entry is missing. (0 for missing and 1 for non-missing.) SCANE first maps the feature indicator to a target vector space $\mathbf{U}$ with dimension d. We call these assigned vectors **feature embeddings**. SCANE then scales feature embeddings with each variable's observed values. The

design of scaling avoids the challenge of quantization resolution in the EVAT problem. Equation 1 illustrates how SCANE embeds a single variable into a vector.

$$\mathrm{SCANE}\left(X_{i,j},\ M_{i,j}\right) = \left(X_{i,j} \cdot M_{i,j}\right) f\left(j\right) = \left(X_{i,j} \cdot M_{i,j}\right) \boldsymbol{u}_j\ , \tag{1}$$

where $f : \mathbb{N} \to \mathbb{U}$ is realized through a single linear layer different for each feature, and $\boldsymbol{u}_j$ is feature j's feature embedding $\in \mathbb{U}$. SCANE assigns a missing value to zero vector $\mathbf{0}^d$. We do not put any restrictions on the feature embeddings. The direction and the length of the feature embeddings are updated according to the training data. It is entirely data-driven.

To generalize SCANE to its matrix form, we have:

$$\mathrm{SCANE}\left(\boldsymbol{X},\ \boldsymbol{M}\right) = \begin{bmatrix} X_{1,1}M_{1,1}\boldsymbol{u}_1 & X_{1,2}M_{1,2}\boldsymbol{u}_2 & \dots & X_{1,n}M_{1,n}\boldsymbol{u}_n \\ X_{2,1}M_{2,1}\boldsymbol{u}_1 & X_{2,2}M_{2,2}\boldsymbol{u}_2 & \dots & X_{2,n}M_{2,n}\boldsymbol{u}_n \\ \vdots & \vdots & \ddots & \vdots \\ X_{m,1}M_{m,1}\boldsymbol{u}_1 & X_{m,2}M_{m,2}\boldsymbol{u}_2 & \dots & X_{m,n}M_{m,n}\boldsymbol{u}_n \end{bmatrix}.$$

$\mathrm{SCANE}\left(\boldsymbol{X},\ \boldsymbol{M}\right)$ is an $m \times n \times d$ tensor. Every feature embedding in the $\mathrm{SCANE}\left(\boldsymbol{X},\ \boldsymbol{M}\right)$ is scaled by the corresponding observed values.

Compared to the embedding method in NLP to solve the EVAT problem, SCANE does not need to worry about the precision of the numerical value due to the "scaling" mechanism. For example, the embedding method in NLP may disassemble the embedding target "age 41.8263… years old" into "age + 4 + 1 + 8 + 2 + …". We do not know how many digits should be included to precisely embed the value. Moreover, the testing set must have some unique values. These are all unlearned values for the embedding module in the NLP flavor. SCANE avoids this situation by learning the feature embedding only. With SCANE's design, we can resolve the issues related to the EVAT problem. These are SCANE's advantages.

Transformer Encoder with Scalable Numerical Embedding. We flatten and transpose $\mathrm{SCANE}\left(\boldsymbol{X},\ \boldsymbol{M}\right)$ into a $mn \times d$ matrix. Similarly, we flatten the matrix $\boldsymbol{M}$ into a $1 \times mn$ matrix.

$$\bar{\boldsymbol{X}} = \left(\mathrm{flatten}\left(\mathrm{SCANE}\left(\boldsymbol{X}, \boldsymbol{M}\right)\right)\right)^T = \begin{bmatrix} X_{1,1}M_{1,1}\boldsymbol{u}_1 & X_{1,2}M_{1,2}\boldsymbol{u}_2 & \dots & X_{m,n}M_{m,n}\boldsymbol{u}_n \end{bmatrix}^T,$$
$$\bar{\boldsymbol{M}} = \mathrm{flatten}\left(\boldsymbol{M}\right) = \begin{bmatrix} M_{1,1} & M_{1,2} & \dots & M_{m,n} \end{bmatrix}.$$

In the Transformer Encoder's self-attention module, we take $\boldsymbol{Z} = \bar{\boldsymbol{X}} + PE$ to obtain the query $\boldsymbol{Q} = \boldsymbol{Z}\mathbf{W}_q$, the key $\boldsymbol{K} = \boldsymbol{Z}\mathbf{W}_k$, and the value $\boldsymbol{V} = \boldsymbol{Z}\mathbf{W}_v$. PE is the positional encoding. The $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$ are learnable weights. To avoid paying attention to missing values, we use the masking mechanism (Vaswani et al., 2017) to mask them in $\boldsymbol{Z}$. This can only be realized under the EVAT concept.

$$\mathrm{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \bar{\boldsymbol{M}}\right) = \mathrm{softmax}\left(\lim_{\beta \to -\infty}\left(\frac{\left(\boldsymbol{\beta}\right)^{k(n+1)\times 1}\left(\mathbf{1}^{1\times k(n+1)} - \bar{\boldsymbol{M}}\right) + \left(\boldsymbol{Q}\boldsymbol{K}^T\right)}{\sqrt{d}}\right)\right)\boldsymbol{V}, \tag{2}$$

where d is the dimension of embeddings as a suggested scaling factor (Vaswani et al., 2017) and β is a number approach negative infinity. $\mathbf{1}^{1\times k(n+1)}$ is an $1 \times k\left(n+1\right)$ matrix whose entries are all 1. The matrix $\lim_{\beta\to-\infty}\left(\boldsymbol{\beta}\right)^{k(n+1)\times 1}$ is a $k\left(n+1\right) \times 1$ and its entries all equal β. All attention weights with missing values as keys will be suppressed by the number β, which approaches negative infinity and will be zero after the $\mathrm{softmax}$. Equation 2 shows how to use the mask to avoid paying attention to missing values in the self-attention module with SCANE.

We can mask missing values independently due to EVAT. If we apply timestamp embedding instead, the missing values would be bound together with other non-missing ones. It is impossible to avoid imputation in this case. The missing value would be assigned to a zero vector in SCANE, and its contextual embedding from the second Transformer Encoder stack would be composed of other non-missing values' embeddings. So, we only mask missing values in the first layer in the entire Transformer Encoder stacks.

4 EXPERIMENTS

Datasets. For robustness, we conduct experiments on three distinct EHR datasets: the Anonymous Hospital Hepatocellular Carcinoma Dataset (private), PhysioNet2012 (public), and MIMIC-III (pub-

lic). All three datasets originate from the healthcare domain and are characterized by irregular sampling and unsynchronized measurement, thereby presenting challenges for MTS binary classification tasks. To align the number of timestamps, we follow Zheng's work to apply the summarization strategy (detailed in Appendix D) on these datasets' data.

- **Anonymous Hepatocellular Carcinoma Dataset (HCC)**: This dataset comprises records from patients over a one-year-length observation window since patients' first diagnosis record. It includes 30 numerical features and 8 categorical features. The objective is to predict if a patient will develop hepatocellular carcinoma within the ensuing five years. After the summarization with a summarization window length of 90 days, the average missing rate of all features amounts to 0.7464.
- **PhysioNet2012 (P12)**: P12 is a public dataset (Goldberger et al., 2000), encompassing 11988 intensive care unit (ICU) stays lasting at least 48 hours. The task is to predict if the patient dies during their hospital stay. The dataset consists of 40 numerical and 2 categorical features. The observation window spans 48 hours, with a summarization window length set to 2 hours. After summarization, the average missing rate of all features is 0.7377.
- **MIMIC-III (MI3)** (Harutyunyan et al., 2019): The public dataset comprises numerous ICU patients with laboratory test results, encompassing 13 numerical features and 4 categorical features. The task for this dataset entails predicting patients' survival during their hospital stay. The observation window spans 48 hours after patients' initial hospitalization, with the summarization window length set to 2 hours. After summarization, the average missing rate of all features in the summarized data is 0.4423.

More dataset details can be found in Appendix E.

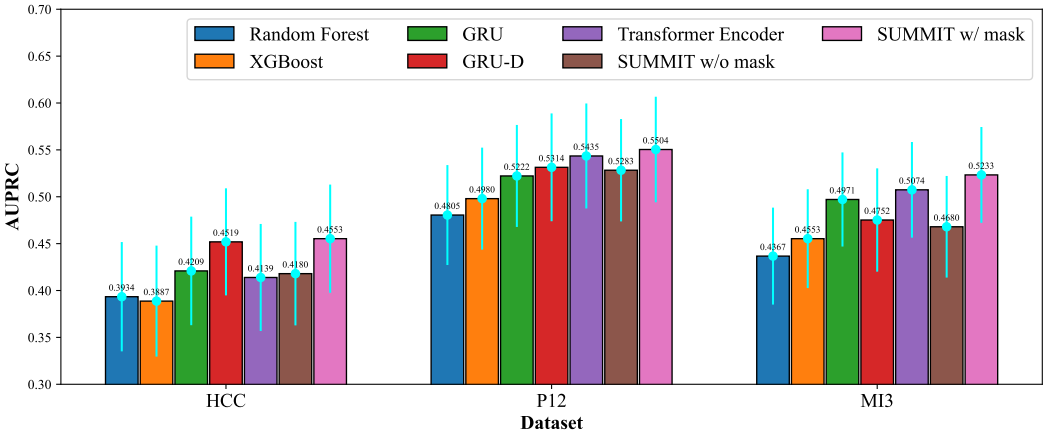

Figure 2: **Models' AUPRC Performance on Each Dataset**: Due to the imbalance issue in these datasets, we choose the area under the precision-recall curve (AUPRC) as the main metric (Saito & Rehmsmeier, 2015).

Results. Figure 2 depicts the performance of all models on the datasets. Our model, SUMMIT, outperforms other models in all datasets on AUPRC. With this result, we can confirm that it can learn well from data featuring high missing rates without the need for imputation. SUMMIT faithfully learns from what we observed and surpasses all benchmarks on the main metric, AUPRC. This supports the idea that SUMMIT is a promising choice for irregular and asynchronous MTS data. More detailed results can be found in Appendix A.

We also train and test our model on these datasets without masking missing values, dubbed SUMMIT w/o mask. The missing values here are imputed with the global mean and the global mode of the training set. Figure 2 shows that our model performs better with masking the missing on each dataset. This may imply the imputation confuses the model when training and testing, and the imputation in these three datasets may not be a good solution for missing values. Other ablation studies can be found in Appendix B. Training and testing details can be found in Appendix F.

5 CONCLUSION

We propose to combat missing values in multivariate time series by learning to embed each value as a token, which requires no imputation. We propose a novel embedding mechanism integrated with a Transformer encoder, achieving state-of-the-art results across three electronic health record datasets.

6 ACKNOWLEDGEMENT

This project is supported by the National Science and Technology Council, the Ministry of Health and Welfare, and the Ministry of Education, Taiwan. The grant numbers are MOST 110-2221-E-002-112-MY3, MOHW112-TDU-B-221-124003, and 113L900701, respectively. Shao-Hua Sun was partially supported by the Yushan Fellow Program by the Ministry of Education, Taiwan.

REFERENCES

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 2018.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

Edward Choi, Zhen Xu, Yujia Li, Michael W. Dusenberry, Gerardo Flores, Yuan Xue, and Andrew M. Dai. Learning the Graphical Structure of Electronic Health Records with Graph Convolutional Transformer, 2020. arXiv:1906.04716 [cs, stat].

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Wenjie Du, David Cote, and Yan Liu. SAITS: Self-Attention-based Imputation for Time Series. *Expert Systems with Applications*, pp. 119619, 2023. arXiv:2202.08516 [cs].

Jianliang Gao, Tengfei Lyu, Fan Xiong, Jianxin Wang, Weimao Ke, and Zhao Li. Predicting the survival of cancer patients with multimodal graph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2022.

A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 2000.

Jake Grigsby, Zhe Wang, Nam Nguyen, and Yanjun Qi. Long-Range Transformers for Dynamic Spatiotemporal Forecasting, 2023. arXiv:2109.12218 [cs, stat].

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 2019. Number: 1 Publisher: Nature Publishing Group.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

SeungHyun Kim, Hyunsu Kim, EungGu Yun, Hwangrae Lee, Jaehun Lee, and Juho Lee. Probabilistic Imputation for Time-series Classification with Missing Data, 2023. arXiv:2308.06738 [cs].

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017. arXiv:1412.6980 [cs].

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection, 2018. arXiv:1708.02002 [cs].

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

Jiawei Ma, Zheng Shou, Alireza Zareian, Hassan Mansour, Anthony Vetro, and Shih-Fu Chang. CDSA: Cross-Dimensional Self-Attention for Multivariate, Geo-tagged Time Series Imputation, 2019. arXiv:1905.09904 [cs, stat].

Pierre-Alexandre Mattei and Jes Frellsen. MIWAE: Deep Generative Modelling and Imputation of Incomplete Data, 2019. arXiv:1812.02633 [cs, stat].

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, 2013. arXiv:1301.3781 [cs].

Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, 2019.

Takaya Saito and Marc Rehmsmeier. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 2015. Publisher: Public Library of Science.

Satya Narayan Shukla and Benjamin M. Marlin. Multi-Time Attention Networks for Irregularly Sampled Time Series, 2021. arXiv:2101.10318 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2017. arXiv:1706.03762 [cs].

Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case, 2020. arXiv:2001.08317 [cs, stat].

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy, 2022. arXiv:2110.02642 [cs].

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Virtual Event Singapore, 2021.

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

Xinlu Zhang, Shiyang Li, Zhiyu Chen, Xifeng Yan, and Linda Petzold. Improving Medical Predictions by Irregular Multimodal Electronic Health Records Modeling, June 2023. arXiv:2210.12156 [cs].

He Zhao, Ke Sun, Amir Dezfouli, and Edwin Bonilla. Transformed Distribution Matching for Missing Value Imputation, 2023. arXiv:2302.10363 [cs].

Ming-Zhe Zheng. Deep sti: Deep stochastic time-series imputation on electronic medical records. (2021), 2021. Publisher: National Tsing Hua University.

Rundong Zuo, Guozhong Li, Byron Choi, Sourav S. Bhowmick, Daphne Ngar-yin Mah, and Grace L. H. Wong. SVP-T: A Shape-Level Variable-Position Transformer for Multivariate Time Series Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. Number: 9.

APPENDIX

A EXTENDED EXPERIMENTAL RESULTS

We present extended experimental results in Table 1. SUMMIT also performs well in other auxiliary metrics, getting the best and second-best values in terms of AUROC and accuracy on the P12.

Dataset	**HCC**			**P12**			**MI3**		
Metric	**AUPRC**	**AUROC**	**c-index**	**AUPRC**	**AUROC**	**accuracy**	**AUPRC**	**AUROC**	**accuracy**
Random Forest	0.3934 ±0.0583	0.8705 ±0.0232	0.8637 ±0.0227	0.4805 ±0.0533	0.8270 ±0.0228	0.8663 ±0.0146	0.4367 ±0.0517	0.8319 ±0.0209	0.8965 ±0.0105
XGBoost	0.3887 ±0.0592	0.8714 ±0.0215	0.8644 ±0.0209	0.4980 ±0.0544	0.8453 ±0.0203	0.8708 ±0.0140	0.4553 ±0.0527	0.8247 ±0.0209	<u>0.8968</u> ±0.0105
GRU	0.4209 ±0.0579	<u>0.8991</u> ±0.0156	<u>0.8915</u> ±0.0152	0.5222 ±0.0571	<u>0.8573</u> ±0.0196	0.8750 ±0.0138	0.4971 ±0.0502	<u>0.8537</u> ±0.0203	**0.9012** ±0.0107
GRU-D	<u>0.4519</u> ±0.0571	**0.9012** ±0.0171	**0.8934** ±0.0167	0.5314 ±0.0575	0.8524 ±0.0215	**0.8804** ±0.0135	0.4752 ±0.0551	0.8415 ±0.0214	0.8959 ±0.0105
Transformer Encoder	0.4139 ±0.0571	0.8964 ±0.0171	0.8888 ±0.0171	<u>0.5435</u> ±0.0560	0.8572 ±0.0200	0.8767 ±0.0131	<u>0.5074</u> ±0.0510	**0.8606** ±0.0187	0.8953 ±0.0105
SUMMIT	**0.4553** ±0.0577	0.8943 ±0.0179	0.8867 ±0.0179	**0.5504** ±0.0563	**0.8602** ±0.0197	<u>0.8783</u> ±0.0129	**0.5233** ±0.0511	0.8492 ±0.0205	0.8910 ±0.0104

Table 1: This shows the overall results of each model on the three test sets. We mark the best value in **boldface** and <u>underline</u> the second-best value for each metric. The value in parentheses is the 95% of the confidence interval of the 1000 bootstrap times in the test set.

B ABLATION STUDY

B.1 SUMMARIZATION WINDOW LENGTH

In this section, we redo the *summarization* on MI3 with the summarization window length (p) of 1 hour and 4 hours. The average missing rate in all the features is 0.5383 with the p = 1 hour, and the missing rate is 0.3225 with the p = 4 hours. We then retrain and retest all models on these MI3 datasets with different p. Again, we take AUPRC as the metric. Figure 3 depicts the models' performance in different summarization time lengths MI3. The optimal summarization window length for all models in the MI3 seems to be 2 hours. Compared to the summarization window length of 1 hour and 2 hours, the dataset will have a higher missing rate when p = 1 hour and there is more noise from imputation. As a result, models perform worse. When p changes to 4 hours, the "resolution" is lower. This may be the reason why almost all models perform worse.

According to this ablation study, we can find the trade-off between the longer p and the shorter p. When p becomes longer, the resolution and the missing rate will decrease. Despite the lower missing rate, the important events may be blurred. This may cause the model to lose the important information. On the other hand, the resolution and the missing rate increase when the p becomes shorter. Memory consumption also increases as p gets shorter.

SUMMIT and GRU-D are more stable than other benchmarks, but SUMMIT surpasses other models under different p. Unlike other benchmarks, these two models are more emphasized on both temporal and feature-wise dimensions. GRU-D takes all variables at the previous timestamp to impute the missing value, which helps it capture the feature-wise relation. SUMMIT may get help from SCANE to give the Transformer Encoder more freedom to learn the latent relation across temporal and feature-wise dimensions.

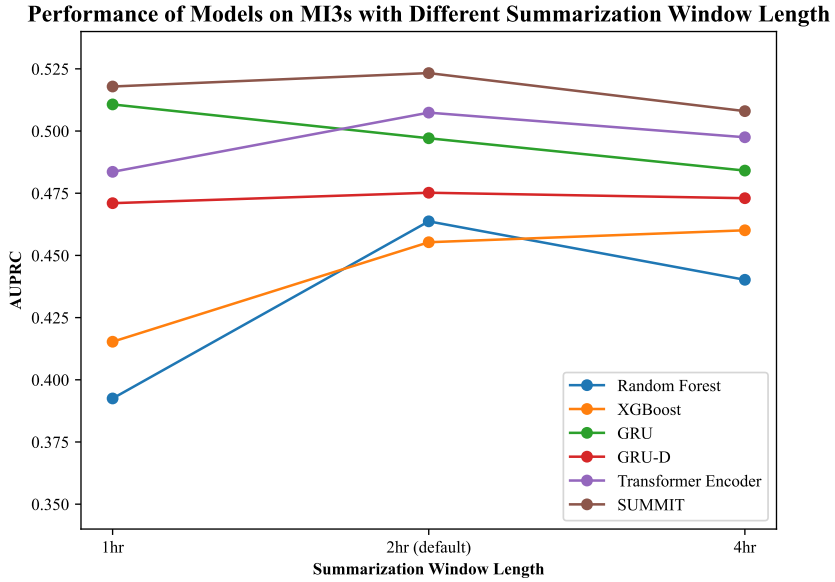

Figure 3: **Performance of All Models on MI3s with Different Summarization Window Length**: The models' setting here is based on the models trained on the default summarization window length $p = 2$ hours. The training, the validation, and the testing samples are identical in every summarization window length setting. We only adjust the p to a different scale.

C Related Work

Sequence-to-sequence models such as gated recurrent unit (GRU) (Chung et al., 2014), Long-Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), and Transformer-based models (Vaswani et al., 2017) have been widely used for MTS data (Ma et al., 2019; Xu et al., 2022; Zuo et al., 2023; Grigsby et al., 2023). Che, Purushotham, Cho, Sontag, and Liu proposed GRU-D, a GRU-based model containing an imputation module, to handle the MTS data from the healthcare domain. Sagheer and Kotb proposed a Deep LSTM architecture model (DLSTM) to forecast petroleum production. More recent works have focused on Transformer-based models (Vaswani et al., 2017). One prominent instantiation is the Time Series Transformer (TST), which proposes a Transformer-based framework for MTS representation learning (Zerveas et al., 2021). Additionally, Wu et al. have employed a Transformer encoder-decoder architecture for forecasting influenza prevalence, highlighting the superior performance of Transformer models compared to other deep learning and statistical models in forecasting tasks.

Deep learning models (Choi et al., 2020; Zhang et al., 2022; 2023) have also found their popularity in the healthcare domain dealing with EHR data. Deep STI (Zheng, 2021) proposed an RNN-VAE-based model to impute missing values in the EHR data. Deep STI can outperform traditional machine learning algorithms and statistical algorithms on the HCC prediction task with the GRU-based classifier. mTAND (Shukla & Marlin, 2021) uses an attention-mechanism-based network to learn the representation from EHR data. These studies have promoted further development of deep learning in healthcare. However, due to the inherent limitation that embeds variables according to the timestamp, most models in these works must consider imputation. This conflicts with the concern from the imputation mentioned above. Our model, SUMMIT, can perfectly relieve the issues. Because we separated each variable, there is no need for imputation. Moreover, it can entirely avoid the effect of imputation during model training.

D Summarization Strategy

To align the number of timestamps of each sample, we follow Zheng's work to apply the *summarization* strategy on the input data. Specifically, given a summarization time duration p, we obtain $k\ (= \lfloor T/p \rfloor)$ summarization intervals, where T is the observation window length. The formed summarization intervals are $[t_1,\ t_1 + p)$, $[t_1 + p,\ t_1 + 2p)$, ..., and $[t_1 + (k-1)\,p,\ t_1 + T)$. We

then assign the rows of $\boldsymbol{X}$ to the summarization intervals where their timestamps belong. Every summarization window uses the mean, the mode (the most frequently observed value), or the last observed value of the collected rows to represent the value of features in the interval. The mean is used to represent the numerical feature; the mode and the last observed value are used to represent the categorical feature. The mean and the mode are computed by dropping missing values. If there is no observation of a feature in the interval, it will assign Nan for this feature. This strategy also counts the number of rows in each summarization window and records it as an additional feature, "segment entry count," to the $\boldsymbol{X}$.

The input time series data are then summarized into a $k \times (n+1)$ matrix, $\boldsymbol{X}' = \left[x'_{i,j}\right]$. The first subscript of $x'_{i,j}$ is the index of the summarization window, and the second subscript is the feature indicator. We defined a $k \times (n+1)$ missing mask matrix $\boldsymbol{M} = [m_{i,j}]$ to indicate the entry that is *not* missing in $\boldsymbol{X}'$:

$$m_{i,j} = \begin{cases} 0\ , \text{ if } x'_{i,j} \text{ is missing.} \\ 1\ , \text{ otherwise.} \end{cases}$$

E DATASETS

Tables 2 to 4 list the full feature set of the datasets applied. In Table 2, "fatty_liver" is a categorical feature to show the fatty liver severity; "parenchymal_liver_disease" is also a categorical feature to represent the severity of cirrhosis; "hosp_days" is the number of hospitalization days; "sono" represents whether a patient has the abdominal ultrasound imaging. In Table 3, "MechVent" means whether a patient uses mechanical ventilation in the ICU.

F TRAINING SETUP

F.1 MODELS

We have selected a set of models to compare against our proposed model, SUMMIT. The non-sequential benchmarks encompass Random Forest (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016), while the deep-learning-based benchmarks include GRU (Chung et al., 2014), GRU-D (Che et al., 2018), and the original Transformer Encoder (Vaswani et al., 2017). These models have garnered widespread use in healthcare, particularly with EHR data.

For Random Forest and XGBoost, we have employed their empirically optimal default hyperparameters. The data for these two models is subjected to a *summarization* strategy, wherein the missing mask is concatenated to the original data by timestamps. We take each variable's global mean and mode from the training set to impute what is missing in the training and testing sets. Before being fed to the model, data is flattened into a one-dimensional matrix.

All the deep learning benchmarks adhere to a common architectural structure, detailed in Section **??**. These models take timestamp embedding as the models' input. They comprise a feature extractor and a classifier, consisting of a dense layer followed by a linear layer, while the feature extractors vary across these models. Specifically, GRU utilizes a GRU-based feature extractor, GRU-D employs a GRU-D-based feature extractor, and the Transformer Encoder adopts a Transformer Encoder as the feature extractor. The *summarization* strategy is also applied to these models and imputes the missing values with the same logic mentioned above. The sequence data for GRU and Transformer Encoder is concatenated with the missing mask for each timestamp.

Our proposed model, SUMMIT, adheres to the same architectural structure outlined in Section **??**. It employs a Transformer Encoder with SCANE as the feature extractor. Since we have both numerical and categorical features, we form two separate SCANE modules for each.

F.2 EXPERIMENTAL SETUP

All deep models are trained under a uniform framework and on the same platform (detailed in Appendix F.3) to ensure a fair comparison. Given the inherent class imbalance in the datasets, we adopt focal loss (Lin et al., 2018). To optimize the performance of the models, we employ a grid search strategy, with the search ranges for each model's hyperparameters provided in Appendix G. We

Feature	Feature Type
AFP (Alpha-Fetoprotein)	Numerical
ALB (Albumin)	Numerical
ALP (Alkaline Phosphatase)	Numerical
ALT (Alanine Aminotransferase)	Numerical
AST (Aspartate Aminotransferase)	Numerical
Anti-HBc (Hepatitis B Core Antibody)	Categorical
Anti-HBe (Anti-Hepatitis B e-Antigen)	Categorical
Anti-HBs (Hepatitis B Surface Antibody)	Categorical
Anti-HCV (Anti-Hepatitis C Virus Antibody)	Categorical
BUN (Blood Urea Nitrogen)	Numerical
CRE (Creatinine)	Numerical
D-BIL (Direct Bilirubin)	Numerical
GGT (gamma-Glutamyltransferase)	Numerical
Glucose AC	Numerical
HB (Hemoglobin)	Numerical
HBVDNA (Hepatitis B Virus DNA)	Numerical
HBeAg (Hepatitis B e-Antigen)	Categorical
HBsAg (Hepatitis B Surface Antigen)	Categorical
HCVRNA (Hepatitis C Virus RNA)	Numerical
HbA1c (Glycated Haemoglobin)	Numerical
Lym (Lymphocyte)	Numerical
Na (Sodium)	Numerical
PLT (Platelet)	Numerical
PT (Prothrombin Time)	Numerical
PT INR (PT International Normalized Ratio)	Numerical
Seg (Neutrophils)	Numerical
T-BIL (Total Bilirubin)	Numerical
TP (Total Protein)	Numerical
WBC (White Blood Cell)	Numerical
Height	Numerical
Weight	Numerical
fatty_liver	Categorical
paranchymal_liver_disease	Categorical
Age	Numerical
hosp_days	Numerical
Sex	Categorical
sono	Categorical

Table 2: **Feature in Anonymous Hepatocellular Carcinoma Dataset.**

select hyperparameters according to models' performance on the validation set, which constitutes 20% of the training set. Throughout the training process, we monitor the model's performance on the validation set every 5 epochs and halt the process if there is no improvement in AUPRC or the area under the receiver operating characteristic curve (AUROC) for a continuous span of 30 epochs. The batch size is fixed at 256 for all experiments. The maximal training epochs for the HCC, P12, and MI3 are set to 100, 500, and 400, respectively, ensuring adequate training for each dataset to capture underlying patterns and achieve convergence.

F.3 Platform Information

This is the information on the platform we used to conduct all the experiments in the paper:

- CPU: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
- Memory: 64GB
- GPU: RTX 3060 with 12GB VRAM

Feature	Feature Type
Weight	Numerical
ALP (Alkaline Phosphatase)	Numerical
ALT (Alanine Aminotransferase)	Numerical
AST (Aspartate Aminotransferase)	Numerical
ALB (Albumin)	Numerical
BUN (Blood Urea Nitrogen)	Numerical
Bilirubin	Numerical
Cholesterol	Numerical
Creatinine	Numerical
DiasABP (Diastolic Arterial Blood Pressure)	Numerical
FiO2 (Inspired Fraction of Oxygen)	Numerical
GCS (Glasgow Coma Scale)	Categorical
Glucose	Numerical
HCO3 (Bicarbonate)	Numerical
HCT (Hematocrit)	Numerical
HR (Heart Rate)	Numerical
K (Potassium)	Numerical
Lactate	Numerical
MAP (Mean Arterial Pressure)	Numerical
MechVent (Mechanical Ventilation)	Categorical
Mg (Magnesium)	Numerical
PaCO2 (Partial Pressure of Carbon Dioxide)	Numerical
PaO2 (Partial Pressure of Oxygen)	Numerical
PLT (Platelets)	Numerical
RespRate (Respiratory Rate)	Numerical
SaO2 (Arterial Oxygen Saturation)	Numerical
SysABP (Systolic Arterial Blood Pressure)	Numerical
Temp (Temperature)	Numerical
TroponinI	Numerical
TroponinT	Numerical
Urine	Numerical
WBC (White Blood Cell)	Numerical
pH (Body Fluid)	Numerical
Age	Numerical
Height	Numerical
Gender	Categorical
ICU Type	Categorical

Table 3: **Feature in PhysioNet2012 Dataset.**

- CUDA version: 11.4
- gcc version: 7.5.0
- pytorch version: 1.13.1
- sklearn version: 1.1.2
- xgboost version: 1.7.5

G Search Range of Hyperparameters

Table 5 shows the hyperparameter grid searching range of all deep learning models. "d_model" means the dimension of embeddings. "num_head" is the number of attention heads. "ff_dim" is the feed-forward dimension of attention module in the transformer-based models. "hidden_size" is the dimension of hidden vectors in the GRU-based models. "num_layer" is the number of unit stacks. We use Adam optimizer (Kingma & Ba, 2017) to optimize all models. After the grid searching, we will do a little perturbation on the learning rate to see if the model performs better. Restricted by the

Feature	Feature Type
Weight	Numerical
Heart Rate	Numerical
Mean Blood Pressure	Numerical
Diastolic Blood Pressure	Numerical
Systolic Blood Pressure	Numerical
Oxygen Saturation	Numerical
Respiratory Rate	Numerical
Capillary Refill Rate	Numerical
Glucose	Numerical
pH (Body Fluid)	Numerical
Temperature	Numerical
Height	Numerical
Fraction Inspired Oxygen	Numerical
Glasgow Coma Scale Eye Opening	Categorical
Glasgow Coma Scale Motor Response	Categorical
Glasgow Coma Scale Total	Categorical
Glasgow Coma Acale Verbal Response	Categorical

Table 4: **Feature in MIMIC-III Dataset.**

GPU memory, the "num_layer" of SUMMIT on the PhysioNet2012 dataset is set to 1. We also list all settings of all models in Table 6, 7, and 8.

Hyperparameter	GRU	GRU-D	Transformer Encoder	SUMMIT
d_model	X	X	112, 128, 144, 160	112, 128, 144, 160
num_head	X	X	1	1
ff_dim	X	X	64, 80, 96, ..., 240, 256	64, 80, 96, ..., 240, 256
hidden_size	64, 80, 96, ..., 240, 256	16, 18, 20, ..., 48	X	X
num_layer	1, 2, 3, ..., 15, 16	1, 2, 3, ..., 15, 16	1, 2, 4, 8, 16	1, 2, 4, 8, 16
classifier_down_factor	2	2	2	2
learning rate	3e-3, 3e-4, 3e-5	3e-3, 3e-4, 3e-5	3e-3, 3e-4, 3e-5	3e-3, 3e-4, 3e-5
Optimizer	Adam	Adam	Adam	Adam

Table 5: **Grid Searching Range of All Deep Learning Models**

Hyperparameter	GRU	GRU-D	Transformer Encoder	SUMMIT
d_model	X	X	144	144
ff_dim	X	X	144	144
hidden_size	64	38	X	X
num_layer	6	1	16	8
learning rate	3e-4	3e-4	3e-5	3.00009e-5
early stopping epoch	30	75	85	100

Table 6: **The Setting of Hyperparameter in HCC Dataset**

Hyperparameter	GRU	GRU-D	Transformer Encoder	SUMMIT
d_model	X	X	144	144
ff_dim	X	X	144	144
hidden_size	128	42	X	X
num_layer	6	1	8	1
learning rate	3e-5	3e-5	3e-5	3e-5
early stopping epoch	100	480	75	350

Table 7: **The Setting of Hyperparameter in P12 Dataset**

Hyperparameter	GRU	GRU-D	Transformer Encoder	SUMMIT
d_model	X	X	128	144
ff_dim	X	X	144	80
hidden_size	128	18	X	X
num_layer	6	1	8	1
learning rate	3e-5	3e-5	3e-5	3e-5
early stopping epoch	280	400	125	380

Table 8: **The Setting of Hyperparameter in MI3 Dataset**