Masking the Gaps: An Imputation-Free Approach to Time Series Modeling with Missing Data

Abhilash Neog[1], **Arka Daw**[2], **Sepideh Fatemi Khorasgani**[1], **Anuj Karpatne**[1]
[1] Virginia Tech, [2] Oak Ridge National Labs

Abstract

A significant challenge in time-series (TS) modelling is presence of missing values in real-world TS datasets. Traditional two-stage frameworks, involving imputation followed by modeling, suffer from two key drawbacks: (1) the propagation of imputation errors into subsequent TS modeling, (2) the trade-offs between imputation efficacy and imputation complexity. While one-stage approaches attempt to address these limitations, they often struggle with scalability or fully leveraging partially observed features. To this end, we propose a novel imputation-free approach for handling missing values in time series termed **Miss**ing Feature-aware **T**ime **S**eries **M**odeling (**MissTSM**) with two main innovations. *First*, we develop a novel embedding scheme that treats every combination of time-step and feature (or channel) as a distinct token. *Second*, we introduce a novel *Missing Feature-Aware Attention (MFAA) Layer* to learn latent representations at every time-step based on partially observed features. We evaluate the effectiveness of MissTSM in handling missing values over multiple benchmark datasets.

1 Introduction

Multivariate time-series (TS) modeling is important in a number of real-world applications. However, a persistent challenge is the presence of missing values on arbitrary sets of features at varying time-steps, introducing "gaps" in the data that can impair the application of State-of-the-art (SOTA) models unless specific adaptations are made. A common approach for handling missing data is to use imputation methods [1, 2, 3]. Recent deep learning (DL)-based imputation techniques [4, 5, 6] can learn complex, nonlinear temporal dynamics which are difficult for simple imputation techniques (like interpolation). However all such frameworks rely on a two-stage process, imputation of missing values, followed by feeding the imputed time-series to a TS model. This introduces two critical challenges: *first*, the propagation of imputation errors into subsequent TS modeling performance, and *second*, the inherent trade-offs between imputation efficacy and imputation complexity.

In this regard, several approaches have been proposed to model time-series with missing values, such as [7] embed time intervals between observations as additional auxiliary features to handle irregular sequences, but relies on recurrent networks which struggle with long-term dependencies. ODE-based methods [8, 9] offer a continuous-time framework for irregular sampling but are computationally demanding and difficult to scale. Recent methods, like [10] implicitly handle missing values via attention mask, or use an additional mask channel ([11]), but in a univariate scenario.

To address the above limitations, we ask the question: *"can we circumvent the need for imputation by designing a DL framework that can directly model multivariate TS with missing values?"* To answer this question, we draw inspiration from the recent success of masked modeling approaches in domains including vision [12] and language [13] where *"masked-attention"* operations embedded in Transformer blocks are effectively utilized to reconstruct data from partial observations. Based on this insight, we propose a novel **Miss**ing Feature-aware **T**ime **S**eries **M**odeling (**MissTSM**) Framework, which capitalizes on the information contained in partially observed features to perform

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

downstream TS modeling tasks without explicitly imputing the missing values. It uses two main innovations. *First*, we develop a novel embedding scheme, termed *Time-Feature Independent (TFI) Embedding*, which treats every combination of time-step and feature (or channel) as a distinct token, encoding them into a high-dimensional space. *Second*, we introduce a novel *Missing Feature-Aware Attention (MFAA) Layer* to learn latent representations at every time-step based on partially observed features. Additionally, we use the framework of Masked Auto-encoder (MAE) [12] to perform self-supervised learning of latent representations, which can be re-used for downstream tasks such as forecasting and classification. To evaluate the ability of MissTSM to model TS with missing values, we consider two synthetic masking techniques: missing completely at random (MCAR), and periodic masking, to simulate varying scenarios of missing values. We show that MissTSM achieves consistently competitive performance as SOTA models on multiple benchmark datasets without using any imputation techniques.

2 Missing Feature Time-Series Modeling (MissTSM)

2.1 Notations and Problem Formulations

Let us represent a multivariate TS as $\mathbf{X} \in \mathbb{R}^{T \times N}$, where T is the number of time-steps, and N is the dimensionality (number of variates) of the TS. We assume a subset of variates (or features) to be missing at some time-steps of $\mathbf{X}$, represented in the form of a missing-value mask $\mathcal{M} \in [0,1]^{T \times N}$, where $\mathcal{M}_{(t,d)}$ represents the value of the mask at t-th time-step and d-th dimension. Let us denote $\mathbf{X}_{(t,:)} \in \mathbb{R}^N$ as the multiple variates of the TS at a particular time-step t, and $\mathbf{X}_{(:,d)} \in \mathbb{R}^T$ as the uni-variate time-series for the variate d. In this paper, we consider two downstream tasks for TS modeling: forecasting and classification. For forecasting, the goal is to predict the future S time-steps of $\mathbf{X}$ represented as $\mathbf{Y} \in \mathbb{R}^{S \times N}$, and, for TS classification, the goal is to predict output labels $\mathbf{Y} \in \{1, 2, ..., C\}$ given $\mathbf{X}$, where C is the number of classes.

2.2 Learning Embeddings for Time-Series with Missing Features using TFI Embedding

Prior embedding techniques such as in Transformer or iTransfomer models cannot handle missing values (See Appendix A.1 for more details) directly. To address this challenge, we propose a novel *Time-Feature Independent (TFI) Embedding* scheme for TS with missing features, where the value at each combination of time-step t and variate d is considered as a single token $\mathbf{X}_{(t,d)}$, and is independently mapped to an embedding using $\texttt{TFIEmbedding} : \mathbb{R} \mapsto \mathbb{R}^D$ as: $\mathbf{h}_{(t,d)} = \texttt{TFIEmbedding}(\mathbf{X}_{(t,d)})$ In other words, the `TFIEmbedding` Layer maps $\mathbf{X} \in \mathbb{R}^{T \times N}$ into the TFI embedding $\mathbf{H}^{\text{TFI}} \in \mathbb{R}^{T \times N \times D}$ (see Figure 5(c) in the Appendix A.1). The `TFIEmbedding` is applied only on tokens $\mathbf{X}_{(t,d)}$ that are observed (for missing tokens, i.e., $\mathcal{M}_{(t,d)} = 0$, we generate a dummy embedding that gets masked out in the MFAA layer). The advantage of such an approach is that even if a particular value in the TS is missing, other observed values in TS can be embedded *"independently"* without being affected by missing values. Later, we demonstrate how our Missing Feature-Aware Attention Layer takes advantage of TFI embedding scheme to compute masked cross-attention among observed features at a time-step to account for missing features.

2.3 Missing Feature-Aware Attention (MFAA) Layer

We propose a novel *Missing Feature-Aware Attention (MFAA) Layer* (see Figure 1) to leverage the power of *"masked-attention"* for learning latent representations at every time-step using partially observed features. MFAA works by computing *attention scores* based on the partially observed features at a time-step t, which are then used to perform a weighted sum of observed features to obtain the latent representation $\mathbf{L}_t$. These latent representations are later fed into an encoder-decoder based self-supervised learning framework to reconstruct the TS.

Mathematical Formulation: To obtain *attention scores* from partially-observed features at a time-step, we apply a masked scaled-dot product operation followed by a softmax operation. We first define a learnable query vector $\mathbf{Q} \in \mathbb{R}^{1 \times D}$ which is independent of the variates and time-steps. The positionally-encoded embeddings at time-step t, $\mathbf{Z}_{(t,:)}$, are used as key and value inputs in the MFAA Layer. Specifically, the query, key, and value vectors are obtained using linear projections as, $\hat{\mathbf{Q}} = \mathbf{Q}\mathbf{W}^{\mathbf{Q}}, \quad \hat{\mathbf{K}}_t = \mathbf{Z}_{(t,:)}\mathbf{W}^{\mathbf{K}}, \quad \hat{\mathbf{V}}_t = \mathbf{Z}_{(t,:)}\mathbf{W}^{\mathbf{V}}$. Here, $\hat{\mathbf{Q}} \in \mathbb{R}^{1 \times d_k}$ and $\hat{\mathbf{K}}_t, \hat{\mathbf{V}}_t \in \mathbb{R}^{N \times d_k}$, where d_k is the dimension of the vectors after linear projection. The linear projection matrices for

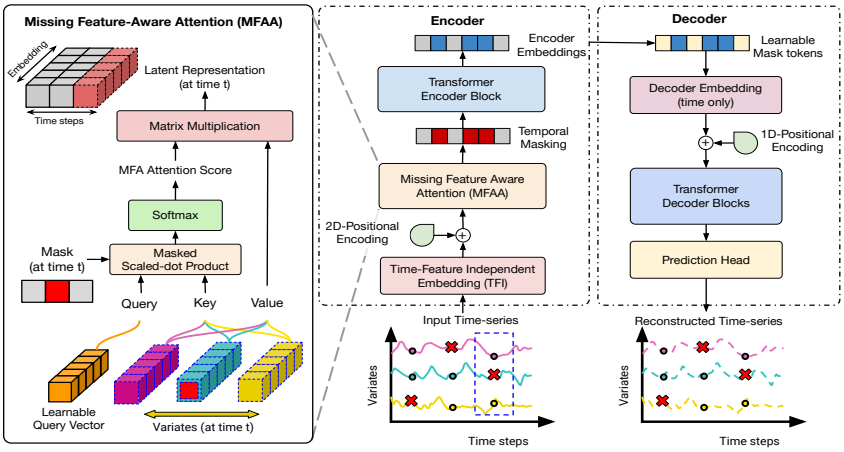


Figure 1: A schematic illustration of the overall MissTSM Framework with a zoomed-in view of the Missing Feature-Aware Attention (MFAA) Layer on the left.

the query, key, and values are defined as: $\mathbf{W^Q}, \mathbf{W^K}, \mathbf{W^V} \in \mathbb{R}^{D \times d_k}$ respectively. Note that the key $\hat{\mathbf{K}}_t$ and value $\hat{\mathbf{V}}_t$ vectors depend on the time-step t, while the query vector is learnable and doesn't change with time. This separation of roles is inspired by similar architectures in multi-modal grounding, for example, in [14], learnable object queries are kept independent of the image content that is sent as keys and values. In our setting, the learnable queries capture the interactions among variates independent of time, enabling the model to attend to the most informative aspects of observed variates at any time-step fed through keys and values. We then define the MFAA Score at a given time-step t as a masked scalar dot-product of query and key vector followed by normalization of the scores, defined as follows:

$$\mathbf{A}_t = \texttt{MFAAScore}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}_t, \mathcal{M}_{(t,:)}) = \texttt{Softmax}\Big(\frac{\hat{\mathbf{Q}}\hat{\mathbf{K}}_t^\top}{\sqrt{d_k}} + \eta\mathcal{M}_{(t,:)}\Big), \tag{1}$$

where $\mathbf{A}_t \in \mathbb{R}^N$ is the MFAA Score vector of size N corresponding to the N variates, and η is a large negative value. The negative bias term η forces the masked-elements that correspond to the missing variates in TS to have an attention score of zero. Thus, by definition, the i-th element of the MFAA Score $\mathbf{A}_{(t,i)} \neq 0 \implies \mathcal{M}_{(t,:)} \neq 0$. We then compute the latent representation $\mathbf{L}_t$ using $\mathbf{A}_t$ and the Value vector $\hat{\mathbf{V}}_t$ as, $\mathbf{L}_t = \texttt{MFAA}(\mathbf{A}_t, \hat{\mathbf{V}}_t) = \mathbf{A}_t\hat{\mathbf{V}}_t \in \mathbb{R}^{d_k}$. Similar to MHA used in transformers, we extend MFAA to multiple heads as: $\texttt{MultiHeadMFAA}(\mathbf{Q}, \mathbf{Z}_{(t,:)}, \mathcal{M}_{(t,:)}) = \text{Concat}(\mathbf{L}_t^0, \mathbf{L}_t^1, ..., \mathbf{L}_t^h)\mathbf{W^O}$, where h is number of heads, $\mathbf{W^0} \in \mathbb{R}^{hd_k \times D_o}$, $\mathbf{L}_t^i$ is latent representation obtained from i-th MHAA Layer, and D_o is output-dimension of MultiHeadMFAA Layer.

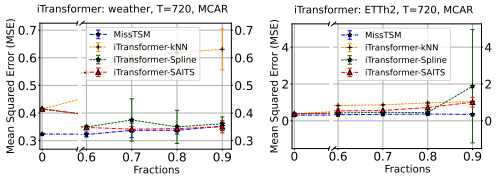
2.4 Putting Everything Together: Overall Framework of MissTSM

Figure 1 shows the MissTSM framework. We opted for a masked TS modeling approach (such as Ti-MAE [15]) due to their recent success. MissTSM has two main stages: (1) *Self-Supervised Learning Stage:* where multivariate TS (with missing values) is reconstructed using an encoder-decoder architecture, with the goal of learning meaningful representations, (2) *Fine-tuning Stage:* where latent representations learned by encoder are fed into a MLP to perform downstream tasks.

3 Experiments

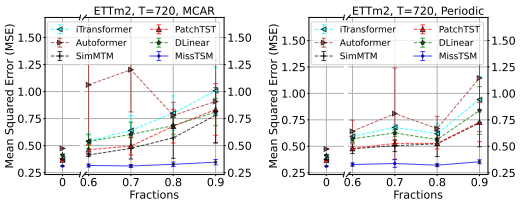
Datasets and Baselines: We consider three popular TS forecasting datasets: ETTh2 [16], ETTm2 [16] and Weather [17]. For classification, we use three real-world datasets, namely, Epilepsy, EMG, and Gesture. We follow the evaluation setup proposed in AutoFormer [18] for the forecasting datasets and evaluation setup proposed in TF-C [19] for the classification ones. . For our experiments, we consider five SOTA TS-modeling baselines, SimMTM [20], PatchTST [21], Autoformer [18], iTransformer [22] and DLinear [23]. In order to apply these methods on data with missing values,

we consider four imputation techniques—a simple 2^{nd}-order spline imputation, k-nearest neighbors (kNN) and two state-of-the-art imputation techniques, SAITS [24] and BRITS [25]. For more experimental details, refer to Appendix B.2



(a) Weather, T=720 (b) ETTh2, T=720

Figure 2: **Multiple Imputation Baselines**. Performance comparison across multiple imputation models. Imputation models considered: kNN, Spline, SAITS. TS Baselines: iTransformer



(a) ETTm2, T=720 (b) ETTm2, T=720

Figure 3: **Multiple TS Baselines.** Performance comparison with multiple TS Baselines imputed with SAITS, under MCAR and Periodic setting

Forecasting Results: For comparing the forecasting performance we consider the Mean Squared Error (MSE) metric. The performance of MissTSM is compared across an array of missing data fractions (60%, 70%, 80%, 90%), along varying forecasting horizons (T=96,192,336,720) and under different masking strategies. Figure 2 compares MissTSM with time-series baselines imputed with multiple imputation techniques. The results shown are on Weather and ETTh2 dataset and along the longest forecasting horizon. We can see that MissTSM performs better or similar in comparison to all the baselines considered. It is also interesting to note the significant drop in performance of iTransformer trained on kNN imputed data, which demonstrates the influence of imputation methods on the final performance of time-series baselines. We observe that BRITS results in significantly high MSE, and therefore not considered in result plots. We also compare MissTSM's performance against different SOTA time-series baselines (see Figure 3) imputed with SAITS, and across different masking strategies. MissTSM shows significant improvement over the baselines on ETTh2 and ETTm2, under both, MCAR and periodic masking settings (Refer to additional results in Appendix C.2). Overall, we observe that MissTSM is consistently better than the baselines for ETTh2 dataset, while it shows competitive performance on ETTm2 and Weather datasets.

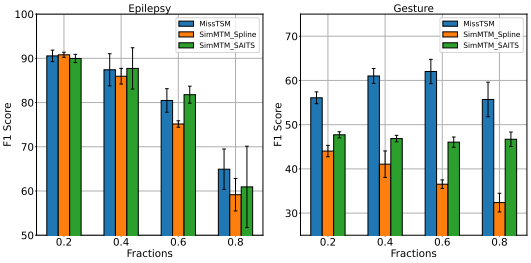


Figure 4: Classification F1 scores across varying masking fractions: $\{0.2, 0.4, 0.6, 0.8\}$.

Classification Results: For fine-tuning on the classification tasks, we add a multi-layer perceptron to the encoder as the classification layer. The same is done for the SimMTM baseline. The classification performance is reported with F1-score metric and under randomly masked (MCAR) fractions of data (20%, 40%, 60%, 80%). Figure 4 compares MissTSM with two variants of SimMTM - one, trained on Spline imputed data, and the other, trained on SAITS imputed data. As seen in the figure, MissTSM achieves roughly similar or better performance as SimMTM on EMG (performance improvement is visible over increasing masked fractions), and outperforms SimMTM on the Gesture Dataset (Refer to C.3 for results on all three datasets). These results demonstrate the effectiveness of our proposed MissTSM framework to circumvent the need for explicit imputation of missing values while achieving similar predictive performance as state-of-the-art.

4 Conclusions and Future Work

We empirically demonstrate the effectiveness of the MissTSM framework across multiple benchmark datasets and synthetic masking strategies. However, a limitation of MFAA layer is that it does not explicitly learn the non-linear temporal dynamics, and relies on subsequent transformer encoder blocks to learn the dynamics. Future work can explore modifications of MFAA layer to address this limitation.

Acknowledgments

This work was supported in part by NSF awards IIS-2239328 and DEB-2213550. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (https://www.energy.gov/doe-public-access-plan).

References

[1] Hyun Ahn, Kyunghee Sun, and Kwanghoon Pio Kim. Comparison of missing data imputation methods in time series forecasting. *Computers, Materials & Continua*, 70(1):767–779, 2022.

[2] Gustavo EAPA Batista, Maria Carolina Monard, et al. A study of k-nearest neighbour as an imputation method. *His*, 87(251-260):48, 2002.

[3] Edgar Acuna and Caroline Rodriguez. The treatment of missing values and its effect on classifier accuracy. In *Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15–18 July 2004*, pages 639–647. Springer, 2004.

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

[5] Andrea Cini, Ivan Marisca, and Cesare Alippi. Multivariate time series imputation by graph neural networks. corr abs/2108.00298 (2021). *arXiv preprint arXiv:2108.00298*, 2021.

[6] Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, and Yisong Yue. Naomi: Non-autoregressive multiresolution sequence imputation. *Advances in neural information processing systems*, 32, 2019.

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

[8] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[9] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[10] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[11] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[15] Zhe Li, Zhongwen Rao, Lujia Pan, Pengyun Wang, and Zenglin Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

[16] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[17] https://www.bgc-jena.mpg.de/wetter/.

[18] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34, 2021.

[19] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in neural information processing systems*, 35, 2022.

[20] Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[22] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[23] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

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

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

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[28] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[29] Ary L Goldberger, Luis AN 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. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[30] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[32] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pages 1651–1661. PMLR, 2020.

[33] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 computing in cardiology*, pages 245–248. IEEE, 2012.

A Additional Details: Methodology

A.1 Limitations of Existing Methods

The first step in time-series modeling using transformer-based architectures is to learn an embedding of the time-series $\mathbf{X}$, which is then fed into the transformer encoder. Traditionally, this is done using an Embedding-layer (typically implemented using a multi-layered perceptron) as $\texttt{Embedding} : \mathbb{R}^N \mapsto \mathbb{R}^D$ that maps $\mathbf{X} \in \mathbb{R}^{T \times N}$ to the embedding $\mathbf{H} \in \mathbb{R}^{T \times D}$, where D is the embedding dimension. The Embedding layer operates on every time-step independently such that the set of variates observed at time-step t, $\mathbf{X}_{(t,:)}$, is considered as a single token and mapped to the embedding vector $\mathbf{h}_t \in \mathbb{R}^D$ as $\mathbf{h}_t = \texttt{Embedding}(\mathbf{X}_{(t,:)})$ (see Figure 5(a)). An alternate embedding scheme was recently introduced in the framework of inverted Transformer [22], where the uni-variate time-series for the d-th variate, $\mathbf{X}_{(:,d)}$, is considered as a single token and mapped to the embedding vector: $\mathbf{h}_d = \texttt{Embedding}(\mathbf{X}_{(:,d)})$ (see Figure 5(b)). While both these embedding schemes have their unique advantages, they are unsuitable to handle time-series with arbitrary sets of missing values at every time-step. In particular, the input tokens to the Embedding layer of Transformer or iTransformer requires all components of $\mathbf{X}_{(t,:)}$ or $\mathbf{X}_{(:,d)}$ to be observed, respectively. If any of the components in these tokens are missing, we will not be able to compute their embeddings and thus will have to discard either the time-step or the variate, leading to loss of information.

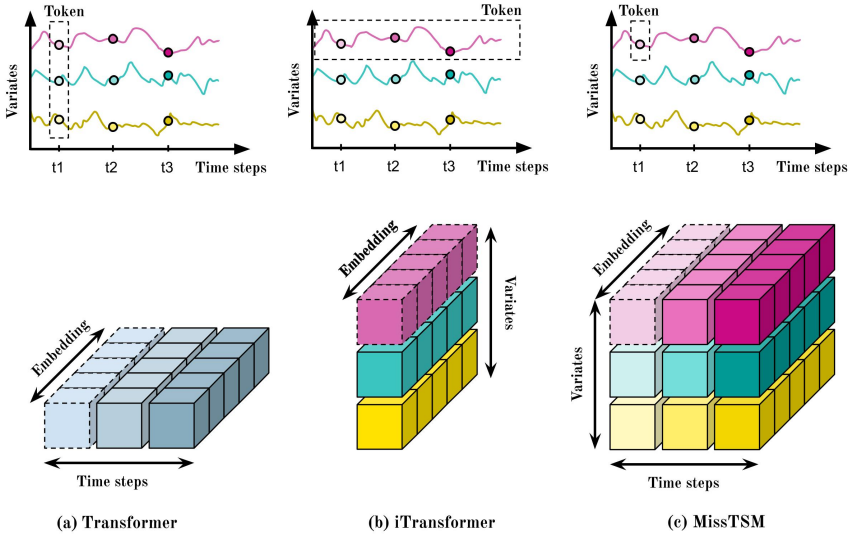


Figure 5: Schematic of the Time-Feature Independent (TFI) Embedding of MissTSM that learns a different embedding for every combination of time-step and variate, in contrast to the time-only embeddings of Transformer [26] and the variate-only embeddings of iTransformers [22].

A.2 2D Positional Encodings

We add Positional Encoding vectors $\mathbf{PE}$ to the TFI embedding $\mathbf{H}^{\text{TFI}}$ to obtain positionally-encoded embeddings, $\mathbf{Z} = \mathbf{PE} + \mathbf{H}^{\text{TFI}}$. Since TFI embeddings treat every time-feature combination as a token, we use a 2D-positional encoding scheme defined as follows:

$$\texttt{PE}(t, d, 2i) = \sin\big(\frac{t}{10000^{(4i/D)}}\big); \quad \texttt{PE}(t, d, 2i+1) = \cos\big(\frac{t}{10000^{(4i/D)}}\big), \tag{2}$$

$$\texttt{PE}(t, d, 2j + D/2) = \sin\big(\frac{d}{10000^{(4j/D)}}\big); \quad \texttt{PE}(t, d, 2j + 1 + D/2) = \cos\big(\frac{d}{10000^{(4j/D)}}\big), \tag{3}$$

where t is the time-step, d is the feature, and $i, j \in [0, D/4)$ are integers.

B Experimental Setup

B.1 Dataset Description

Forecasting Dataset Details

ETT. The ETT [16] dataset captures load and oil temperature data from electricity transformers. ETTh2 includes 17,420 hourly observations, while ETTm2 comprises 69,680 15-minute observations. Both datasets span two years and contain 7 variates each.

Weather. Weather [17] is a 10-minute frequency time-series dataset recorded throughout the year 2020 and consists of 21 meteorological indicators, like humidity, air temperature, etc.

Following previous works in this area, we use a train-validation-test split of 6:2:2 for the ETT datasets and 7:1:2 for the Weather dataset. We standardized the input features by subtracting off the mean and dividing by the standard deviation for every feature over the training set. Again, following the approach used in previous works, we compute the MSE in the normalized space of all features considering all features together.

Classification Dataset Details

Epilepsy. Epilepsy [27] contains univariate brainwaves (single-channel EEG) sampled from 500 subjects (with 11,500 samples in total), with each sample classified as having epilepsy or not (binary classification).

Gesture. Gesture [28] dataset consists of 560 samples, each having 3 variates (corresponding to the accelerometer data) and each sample corresponding to one of the 8 hand gestures (or classes)

EMG. EMG [29] dataset contains 163 EMG (Electromyography) samples corresponding to 3-classes of muscular diseases.

We make use of the following readily available data splits (train, validation, test) for each of the datasets: **Epilepsy** = 60 (30 samples per each class)/20 (10 samples per each class)/11420 (Train/Val/Test) **Gesture** = 320/20/120 (Train/Val/Test) **EMG** = 122/41/41 (Train/Val/Test)

B.2 Synthetic Masked Data Generation

Random Masking: We generated masks by randomly selecting data points across all variates and time-steps, assigning them as missing with a likelihood determined by p (masking fraction). The selected data points were then removed, effectively simulating missing values at random. For multiple runs, we created multiple such versions of the synthetic datasets and compared all baseline methods and MissTSM on the same datasets.

Periodic Masking: We use a sine curve to generate the masking periodicity with given phase and frequency values for different features. Specifically, the time-dependent periodic probability of seeing missing values is defined as $\hat{\mathtt{p}}(t) = \mathtt{p} + \alpha(1-\mathtt{p})\sin(2\pi\nu\mathtt{t} + \phi)$, where, ϕ and ν are randomly chosen across the feature space, α is a scale factor, and $\mathtt{p}$ is an offset term. We vary $\mathtt{p}$ from low to high values to get different fractions of periodic missing values in the data. To implement this masking strategy, each feature in the dataset was assigned a unique frequency, randomly selected from the range [0.2, 0.8]. This was done to reduce bias and increase randomness in periodicity across the feature space. Additionally, the phase shift was chosen randomly from the range [0, 2]. This was applied to each feature to offset the sinusoidal function over time. Like frequency, the phase value was different for different features. This generated a periodic pattern for the likelihood of missing data.

B.3 Implementation Details

The experiments have been implemented in PyTorch using NVIDIA TITAN 24 GB GPU. The baselines have been implemented following their official code and configurations. We consider Mean Squared Error (MSE) as the metric for time-series forecasting and F1-score for the classification tasks.

Forecasting experiments. MissTSM was trained with the MSE loss, using the Adam [30] optimizer with a learning rate of 1e-3 during pre-training for 50 epochs and a learning rate of 1e-4 during finetuning with an early stopping counter of 3 epochs. Batch size was set to 16. All the reported

missing data experiment results are obtained over 5 trials (5 different masked versions). During fine-tuning for different Prediction lengths (96, 192, 336, 720), we used the same pre-trained encoder and added a linear layer at the top of the encoder.

Classification experiments. MissTSM was trained using the Adam [30] optimizer, with MSE as the loss function during pre-training and Cross-Entropy loss during fine-tuning. During fine-tuning, we plugged a 64-D linear layer at the top of the pre-trained encoder. We pre-trained and fine-tuned for 100 epochs.

B.4 Hyper-parameter Details

For MissTSM, we start with the same set of hyper-parameters as reported in the SimMTM paper as initialization (see Table 1), and then search for the best learning rate in factors of 10, and encoder/decoder layers in the range [2, 4]. Note that we only perform hyper-parameter tuning on 100% data, and use the same hyper-parameters for all experiments involving the dataset, such as different missing value probabilities. Our goal is to show the generic effectiveness of our MissTSM framework even without any rigorous hyper-parameter optimization. Additionally, we would also like to note that our model sizes are relatively very small (number of parameters for ETTh2=28,080, Weather= 149,824, and ETTm2= 28,952), compared to other baselines such as SimMTM (ETTh2=4,694,186), iTransformer (ETTh2=254,944), and PatchTST (ETTh2=81,728).

Table 1: Hyperparameters for Forecasting and Classification Tasks

Task	Enc. Layers	Dec. Layers	Enc. Heads	Dec. Heads	Enc. Embed Dim	Dec. Embed Dim
Forecasting						
ETTh2	2	2	8	4	8	32
ETTm2	3	2	8	4	8	32
Weather	2	2	8	4	64	32
Classification						
All Datasets	3	2	16	16	32	32

Table 2: Hyper-parameter sensitivity of MissTSM on ETTh2 with 70% Masking Fraction, MCAR. Best results shown in bold, second best underlined. Hyper-parameter settings used in the remainder of experiments in the paper are italicized.

	Enc. Heads			Enc. Layers			Enc. Embed Dim		
	1	4	8	1	2	3	8	16	32
96	0.246	**0.245**	*0.246*	0.249	***0.243***	0.244	***0.243***	0.248	0.285
192	**0.261**	0.273	*0.266*	0.287	***0.267***	0.271	*0.267*	**0.266**	0.340
336	0.312	**0.279**	*0.310*	**0.294**	*0.392*	0.307	*0.392*	**0.316**	0.369
720	**0.326**	0.346	*0.333*	0.351	***0.323***	0.355	***0.323***	0.338	0.446

	Dec. Heads			Dec. Layers			Dec. Embed Dim		
	1	4	8	1	2	3	8	16	32
96	0.261	***0.243***	0.252	0.276	***0.242***	0.248	0.250	0.259	***0.243***
192	0.276	***0.267***	0.272	**0.266**	*0.268*	0.268	**0.257**	0.272	*0.267*
336	0.319	*0.392*	**0.301**	**0.262**	*0.352*	0.271	0.289	**0.266**	*0.392*
720	0.324	***0.323***	0.330	**0.323**	*0.364*	0.341	0.353	0.384	***0.323***

C Additional Results

C.1 Embedding of 1D data

To understand the usefulness of mapping 1D data to multi-dimensional data in TFI embedding, we present (in Table 3) an ablation comparing performances on ETTh2 with and without using high-dimensional projections in TFI Embedding under the no missing value scenario. Projecting 1D scalars independently to higher-dimensional vectors may look wasteful at the time of initialization of TFI Embedding, when the context of time and variates are not incorporated. However, it is during the cross-attention stage (using MFAA layer or later using the Transformer encoder block) that we can

leverage the high-dimensional embeddings to store richer representations bringing in the context of time and variate in which every data point resides.

From Table 3, we can see that TFI embedding with 8-dimensional vectors consistently outperform the ablation with 1D representations, empirically demonstrating the importance of high-dimensional projections in our proposed framework.

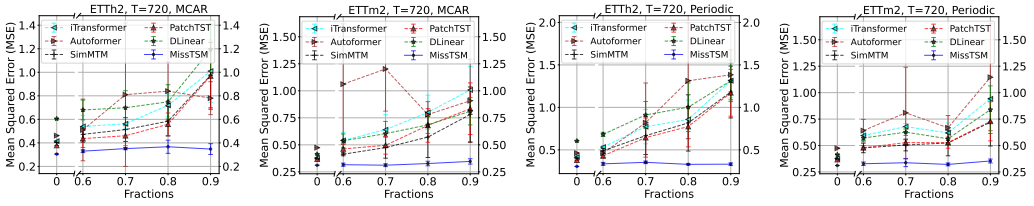
Table 3: Effect of TFI Embedding with embedding size=1 and embedding size=8 under no masking scenario. Dataset=ETTh2

Time Horizon	**TFI Embedding with embedding size = 1**	**TFI Embedding with embedding size = 8**
96	0.283 ± 0.048	**0.245 ± 0.011**
192	0.285 ± 0.078	**0.260 ± 0.023**
336	0.319 ± 0.023	**0.300 ± 0.016**
720	0.378 ± 0.022	**0.334 ± 0.032**

C.2 Forecasting

Table 4 compares the forecasting performance of MissTSM with five SOTA baseline methods in terms of the Mean Squared Error (MSE) metric on three datasets (ETTh2, ETTm2 and Weather) with varying forecasting horizons, imputation techniques (Spline and SAITS), and masking schemes. We provide the mean and standard deviations over 5 different samples of the masking schemes. We choose a missing value probability of 60% for MCAR masking and 70% for periodic masking to simulate scenarios with varying (and often extreme) amounts of missing information. We can see that in the no masking experiment, the performance of all methods (with the exception of AutoFormer) are mostly comparable to each other across all three datasets, with MissTSM and PatchTST having a slight edge on the ETTh2/ETTm2 and Weather datasets, respectively. For the MCAR masking experiments, we observe a trend across all the datasets that the MissTSM framework performs slightly better than the baselines for longer-term forecasting (such as forecasting horizon of 720), and comparable to the best-performing baselines on other forecasting horizons. For the Periodic masking experiment, we can see that MissTSM is consistently better than the baselines for ETTh2 dataset, while for the ETTm2 and Weather datasets, the forecasting performance is comparable to the other baselines. These results demonstrate the effectiveness of our proposed MissTSM framework to circumvent the need for explicit imputation of missing values while achieving comparable performance as SOTA.

By being imputation-free, MissTSM does not suffer from the propagation of imputation errors (from the imputation scheme) to forecasting errors (from the time-series models). In Appendix Figure 13, we provide empirical evidence of this error propagation, where we see a positive correlation between imputation errors and forecasting errors of baseline methods, indicating that reducing imputation errors is crucial for improving forecasting accuracy. This finding underscores the limitations of traditional two-stage approaches and suggests that using more sophisticated imputation models is necessary to achieve lower forecasting errors. We also report the computation time of SimMTM (with Spline and SAITS) and MissTSM in Appendix Table 5, where we demonstrate that MissTSM is significantly faster as it does not involve any expensive interpolations as an additional advantage.

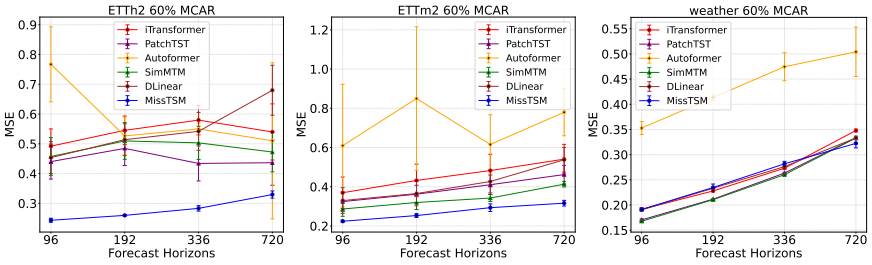


(a) ETTh2, MCAR (b) ETTm2, MCAR (c) ETTh2, Periodic (d) ETTm2, Periodic

Figure 6: **Multiple Time-series Baselines.** Performance comparison between MCAR and Periodic masking with multiple TS Baselines imputed with SAITS. TS Baselines considered: Autoformer [18], PatchTST [21], iTransformer [22], DLinear [23], SimMTM [20]

Table 4: Comparing forecasting performance of baseline methods using mean squared error (MSE) as the evaluation metric under no masking, MCAR masking, and periodic masking. For every dataset, we consider multiple forecasting horizons, $T \in \{96, 192, 336, 720\}$. Results are color-coded as Best, Second best. We report the mean and standard deviations (in brackets) across 5 random sampling of the masking schemes. Subscript $_{SP}$ refer to Spline and $_{SA}$ refer to SAITS

		ETTh2				ETTm2				Weather				Avg
		96	192	336	720	96	192	336	720	96	192	336	720	Rank
No Masking	**MissTSM**	0.255	0.234	0.316	0.305	0.183	0.209	0.261	0.311	0.164	0.210	0.254	0.324	1.9
	SimMTM	0.295	0.356	0.375	0.404	0.172	0.223	0.282	0.374	0.163	0.203	0.255	0.326	2.9
	PatchTST	0.274	0.338	0.330	0.378	0.164	0.220	0.277	0.367	0.151	0.196	0.249	0.319	1.7
	AutoFormer	0.501	0.516	0.565	0.462	0.352	0.337	0.494	0.474	0.306	0.434	0.437	0.414	5.9
	DLinear	0.288	0.383	0.447	0.605	0.168	0.224	0.299	0.414	0.175	0.219	0.265	0.323	4.1
	iTransformer	0.304	0.392	0.425	0.415	0.176	0.246	0.289	0.379	0.163	0.203	0.256	0.326	4.5
MCAR Masking	**MissTSM**	$0.243_{0.006}$	$0.259_{0.002}$	$0.283_{0.009}$	$0.329_{0.011}$	$0.224_{0.005}$	$0.253_{0.009}$	$0.293_{0.019}$	$0.316_{0.014}$	$0.191_{0.003}$	$0.234_{0.006}$	$0.281_{0.004}$	$0.322_{0.008}$	2.7
	SimMTM$_{SP}$	$0.309_{0.001}$	$0.372_{0.005}$	$0.396_{0.01}$	$0.418_{0.008}$	$0.185_{0.001}$	$0.243_{0.002}$	$0.298_{0.001}$	$0.388_{0.005}$	$0.203_{0.009}$	$0.242_{0.010}$	$0.284_{0.008}$	$0.386_{0.008}$	5.0
	SimMTM$_{SA}$	$0.457_{0.06}$	$0.510_{0.061}$	$0.503_{0.055}$	$0.472_{0.066}$	$0.287_{0.037}$	$0.320_{0.035}$	$0.342_{0.017}$	$0.413_{0.014}$	$0.187_{0.002}$	$0.240_{0.001}$	$0.280_{0.001}$	$0.385_{0.004}$	6.2
	PatchTST$_{SP}$	$0.290_{0.003}$	$0.355_{0.003}$	$0.345_{0.003}$	$0.390_{0.003}$	$0.169_{0.001}$	$0.228_{0.001}$	$0.286_{0.001}$	$0.378_{0.001}$	$0.183_{0.009}$	$0.226_{0.009}$	$0.277_{0.009}$	$0.339_{0.008}$	2.1
	PatchTST$_{SA}$	$0.440_{0.059}$	$0.484_{0.057}$	$0.434_{0.059}$	$0.436_{0.075}$	$0.324_{0.05}$	$0.362_{0.045}$	$0.410_{0.049}$	$0.462_{0.047}$	$0.175_{0.002}$	$0.211_{0.000}$	$0.264_{0.002}$	$0.335_{0.001}$	4.6
	AutoFormer$_{SP}$	$0.559_{0.05}$	$0.628_{0.101}$	$0.525_{0.037}$	$0.550_{0.143}$	$0.280_{0.006}$	$0.390_{0.158}$	$0.360_{0.018}$	$0.475_{0.033}$	$0.321_{0.008}$	$0.413_{0.013}$	$0.508_{0.036}$	$0.467_{0.032}$	8.9
	AutoFormer$_{SA}$	$0.767_{0.126}$	$0.526_{0.06}$	$0.550_{0.019}$	$0.449_{0.010}$	$0.610_{0.312}$	$0.850_{0.365}$	$0.615_{0.151}$	$1.045_{0.262}$	$0.353_{0.013}$	$0.413_{0.006}$	$0.474_{0.028}$	$0.504_{0.049}$	10.2
	DLinear$_{SP}$	$0.296_{0.003}$	$0.401_{0.018}$	$0.445_{0.006}$	$0.607_{0.013}$	$0.458_{0.169}$	$0.228_{0.001}$	$0.302_{0.000}$	$0.531_{0.144}$	$0.205_{0.007}$	$0.241_{0.007}$	$0.282_{0.008}$	$0.373_{0.009}$	6.5
	DLinear$_{SA}$	$0.454_{0.053}$	$0.514_{0.053}$	$0.542_{0.064}$	$0.680_{0.084}$	$0.330_{0.065}$	$0.365_{0.062}$	$0.427_{0.058}$	$0.538_{0.063}$	$0.190_{0.001}$	$0.233_{0.000}$	$0.276_{0.000}$	$0.333_{0.001}$	6.8
	iTransformer$_{SP}$	$0.313_{0.004}$	$0.394_{0.014}$	$0.436_{0.005}$	$0.429_{0.005}$	$0.178_{0.001}$	$0.243_{0.0004}$	$0.293_{0.001}$	$0.384_{0.008}$	$0.197_{0.006}$	$0.260_{0.007}$	$0.315_{0.008}$	$0.349_{0.006}$	4.9
	iTransformer$_{SA}$	$0.492_{0.058}$	$0.545_{0.048}$	$0.579_{0.049}$	$0.540_{0.094}$	$0.369_{0.080}$	$0.432_{0.083}$	$0.482_{0.083}$	$0.541_{0.075}$	$0.191_{0.002}$	$0.228_{0.002}$	$0.273_{0.002}$	$0.348_{0.003}$	7.7
Periodic Masking	**MissTSM**	$0.246_{0.018}$	$0.263_{0.017}$	$0.301_{0.042}$	$0.353_{0.015}$	$0.227_{0.006}$	$0.249_{0.006}$	$0.282_{0.011}$	$0.337_{0.036}$	$0.212_{0.007}$	$0.256_{0.008}$	$0.313_{0.009}$	$0.379_{0.019}$	4.1
	SimMTM$_{SP}$	$0.372_{0.122}$	$0.469_{0.198}$	$0.496_{0.198}$	$0.510_{0.200}$	$0.192_{0.010}$	$0.247_{0.009}$	$0.301_{0.008}$	$0.391_{0.008}$	$0.182_{0.004}$	$0.248_{0.003}$	$0.291_{0.009}$	$0.344_{0.005}$	4.7
	SimMTM$_{SA}$	$0.591_{0.132}$	$0.666_{0.152}$	$0.681_{0.182}$	$0.667_{0.222}$	$0.389_{0.071}$	$0.409_{0.054}$	$0.436_{0.076}$	$0.505_{0.055}$	$0.178_{0.002}$	$0.214_{0.001}$	$0.261_{0.001}$	$0.354_{0.003}$	6.0
	PatchTST$_{SP}$	$0.328_{0.047}$	$0.389_{0.040}$	$0.381_{0.050}$	$0.426_{0.058}$	$0.174_{0.004}$	$0.231_{0.003}$	$0.289_{0.004}$	$0.381_{0.004}$	$0.181_{0.004}$	$0.227_{0.005}$	$0.267_{0.005}$	$0.346_{0.003}$	2.4
	PatchTST$_{SA}$	$0.581_{0.120}$	$0.620_{0.132}$	$0.592_{0.170}$	$0.644_{0.230}$	$0.423_{0.054}$	$0.457_{0.042}$	$0.493_{0.037}$	$0.527_{0.027}$	$0.171_{0.002}$	$0.212_{0.001}$	$0.263_{0.005}$	$0.334_{0.001}$	5.2
	Autoformer$_{SP}$	$0.482_{0.041}$	$0.685_{0.165}$	$0.621_{0.166}$	$0.546_{0.035}$	$0.329_{0.109}$	$0.315_{0.010}$	$0.398_{0.090}$	$0.456_{0.021}$	$0.333_{0.0176}$	$0.387_{0.035}$	$0.406_{0.025}$	$0.453_{0.016}$	7.5
	Autoformer$_{SA}$	$1.415_{0.807}$	$0.810_{0.269}$	$1.364_{0.760}$	$0.820_{0.467}$	$1.303_{1.278}$	$0.933_{0.444}$	$1.788_{0.538}$	$0.809_{0.431}$	$0.335_{0.009}$	$0.387_{0.017}$	$0.435_{0.035}$	$0.467_{0.017}$	10.8
	DLinear$_{SP}$	$0.346_{0.069}$	$0.475_{0.108}$	$0.477_{0.044}$	$0.649_{0.068}$	$0.327_{0.188}$	$0.230_{0.002}$	$0.305_{0.003}$	$0.473_{0.038}$	$0.215_{0.018}$	$0.244_{0.013}$	$0.284_{0.008}$	$0.339_{0.007}$	5.0
	DLinear$_{SA}$	$0.605_{0.109}$	$0.674_{0.11}$	$0.728_{0.138}$	$0.911_{0.158}$	$0.447_{0.049}$	$0.475_{0.043}$	$0.523_{0.042}$	$0.626_{0.032}$	$0.190_{0.001}$	$0.233_{0.000}$	$0.276_{0.001}$	$0.333_{0.001}$	7.4
	iTransformer$_{SP}$	$0.358_{0.070}$	$0.435_{0.067}$	$0.488_{0.096}$	$0.497_{0.119}$	$0.180_{0.005}$	$0.245_{0.006}$	$0.296_{0.007}$	$0.384_{0.007}$	$0.197_{0.009}$	$0.233_{0.006}$	$0.288_{0.01}$	$0.351_{0.010}$	4.2
	iTransformer$_{SA}$	$0.691_{0.143}$	$0.715_{0.140}$	$0.763_{0.153}$	$0.773_{0.201}$	$0.512_{0.055}$	$0.578_{0.052}$	$0.662_{0.05}$	$0.680_{0.029}$	$0.194_{0.001}$	$0.229_{0.004}$	$0.274_{0.002}$	$0.350_{0.003}$	8.2



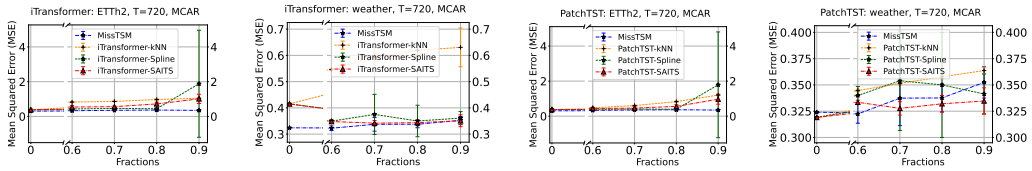
(a) ETTh2 60% MCAR (b) ETTm2 60% MCAR (c) Weather 60% MCAR

Figure 7: Forecasting performance with the horizon length $T \in 96, 192, 336, 720$ and fixed lookback length S = 336. Baseline models are imputed with SAITS

C.3 Classification

Full classification results (on all the datasets) are shown in Figure 9

Real-world results on Physio-Net: We compare the performance of the MissTSM framework with six imputation baselines— M-RNN [31], GP-VAE [32], BRITS [25], Transformer [26], and SAITS [24]—on the real-world PhysioNet classification dataset [33] that is highly sparse with 80% missing values (see Appendix for additional details), as shown in Figure 10. We follow the same evaluation setup as proposed in [24]. MissTSM achieves an F1-score of 57.84%, representing an approximately 15% improvement over SAITS, the best-performing imputation model, which scored 42.6%. This substantial performance gain on a real-world dataset with missing values highlights the advantages



(a) iTransformer, ETTh2 (b) iTransformer, Weather (c) PatchTST, ETTh2 (d) PatchTST, Weather

Figure 8: **Multiple Imputation Baselines**. Performance comparison across multiple imputation models. Imputation models considered: kNN, Spline, SAITS [24]. TS Baselines: iTransformer [22] and PatchTST [21]

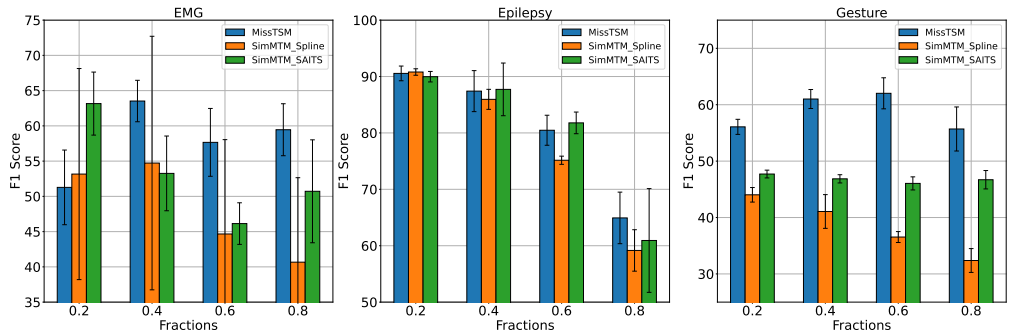


Figure 9: Classification F1 scores on three datasets - EMG, Epilepsy, Gesture. Masking fractions considered: 0.2, 0.4, 0.6, 0.8.

of MissTSM's single-stage approach compared to traditional two-stage methods, beyond synthetic masking schemes used to simulate missing values in other datasets.

C.4 Ablations on Forecasting and Classification task

In the ablation experiments, our goal is to quantify the effectiveness of the TFI-Embedding scheme and the MFAA Layer on MissTSM. To achieve this, we compare MissTSM with Ti-MAE, which can be viewed as an ablation of MissTSM without the TFI-Embedding and MFAA Layers. We refer to this ablation of MissTSM as MAE. For both the forecasting (see Fig. 11) and classification (see Fig. 12) tasks, we compare the MissTSM framework with MAE trained on spline and SAITS

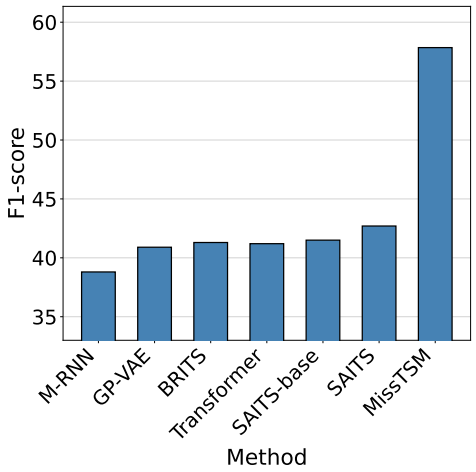


Figure 10: Classification Performance of MissTSM and other imputation baselines on PhysioNet Dataset [33].

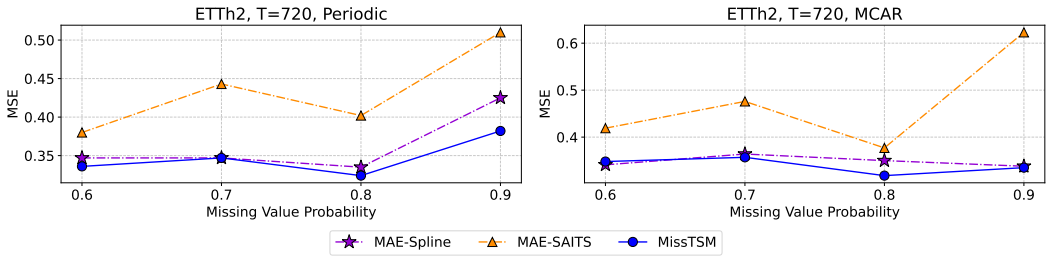


Figure 11: Ablations of MissTSM with and without MFAA layer on Forecasting datasets.

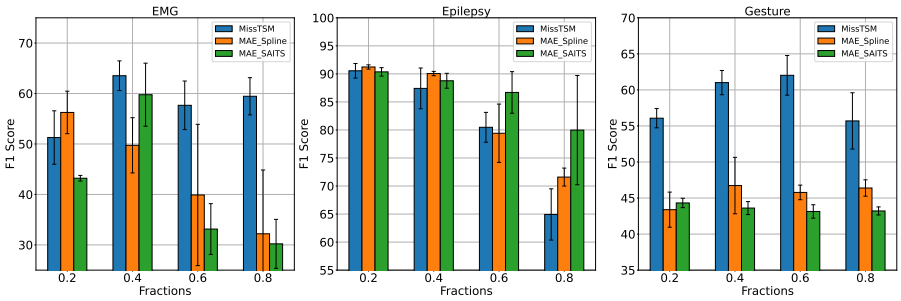


Figure 12: Ablations of MissTSM with and without the TFI+MFAA layer on the classification tasks.

imputation techniques. For forecasting on ETTh2, we observe that our proposed MissTSM framework consistently outperforms the MAE ablations without the MFAA Layer. On the other hand for the classification, we show that for all the three datasets, we are either comparable or better than the MAE ablations. This demonstrates the efficacy of the TFI-Embedding and MFAA Layer for time-series modeling with missing values.

C.5 Experiment on Computational cost comparison

We consider a case study of a classification task on the Epilepsy dataset. Dataset is 80% masked under MCAR. Spline and SAITS are the imputation techniques and SimMTM is the time-series model used. We report the total modeling time as the sum of imputation time and the time-series model training time.

In Table 5, we observe that, while SimMTM integrated with SAITS achieves the highest F1 score, the total imputation time for SAITS is significantly higher than that of Spline. This additional computational overhead substantially increases the overall modeling time. Moreover, SAITS has approximately 1.3 million trainable parameters, further increasing the overall model complexity of the time-series modeling task. This highlights the potential trade-off between imputation efficiency and complexity (by imputation complexity we are referring to both model and time complexity).

In the case of our proposed method, we do not have the extra overhead of imputation complexity. Simultaneously, MissTSM also achieves competitive performance.

Table 5: Comparison of total computational cost between MissTSM and SimMTM integrated with Spline and SAITS

Time-Series Model	Imp. Model	Imp. Time (sec)	TS Model Train Time (sec)	Total Time (sec)	F1 Score
SimMTM	SAITS	949 ± 42.9	397.59 ± 2.64	1346.59 ± 45.54	61.0 ± 9.20
	Spline	8.74 ± 0.38	397.59 ± 2.64	406.33 ± 3.02	59.16 ± 3.67
MissTSM	N/A	N/A	346.8 ± 7.32	**346.8 ± 7.32**	**64.93 ± 4.57**

C.6 Imputation error propagation

Figure 13 captures the propagation of imputation errors and forecasting errors for the weather dataset (at 720 forecasting horizon). It demonstrates that there is an overall positive correlation between the imputation error and forecasting errors, thereby demonstrating propagation of the imputation errors into the downstream time-series models.

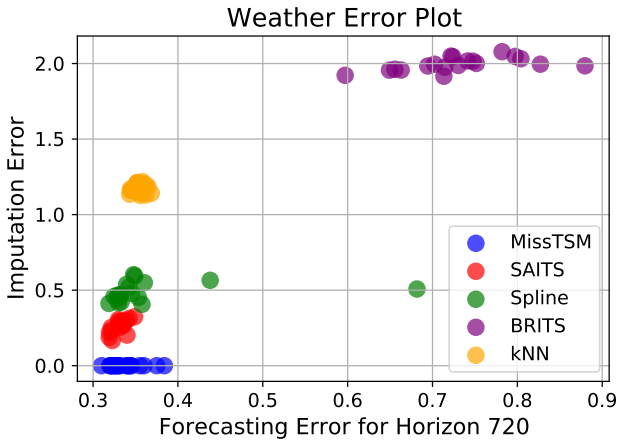


Figure 13: Imputation error vs Forecasting error across 5 trials for 4 missing fractions, 0.6, 0.7, 0.8, 0.9

C.7 Analysis of impact of frequency and phase parameters

In the following, we provide additional details regarding an ablation we conducted to understand the impact of frequency and phase parameters. Given the varying frequency and phase for each feature, we modify the intervals of both to assess their impact on the results. Dataset=ETTh2, Fraction=90%

Case 1. With the phase interval held constant, we lower the frequency range and examine two intervals: one in the high frequency region ([0.6, 0.9]) and one in the low frequency region ([0.1, 0.3]). The performance comparison between these new strategies and the original configuration is shown in Table 6.

Table 6: Effect of sampling from different frequency intervals. The best results are in bold and second-best are italicized

Time Horizon	Original Periodic Masking MSE	High Frequency MSE	Low Frequency MSE
96	**0.268 ± 0.0151**	*0.281 ± 0.028*	0.285 ± 0.023
192	**0.295 ± 0.0298**	*0.301 ± 0.037*	0.316 ± 0.049
336	0.319 ± 0.0185	*0.308 ± 0.014*	**0.307 ± 0.011**
720	0.356 ± 0.0310	**0.339 ± 0.043**	*0.351 ± 0.058*

We observe that with a reduced frequency range, for both high and low frequency intervals, the performance improves as the prediction window increases.

Case 2. Following a similar approach as Case 1, we keep the frequency interval constant and lower the range of phase values. We examine the following intervals: the positive half-cycle $[0, \pi]$ and the negative half-cycle $[\pi, 2\pi]$. Table 7 presents the results of this ablation

Table 7: Effect of sampling from different phase intervals. The best results are in bold and second-best are italicized

Time Horizon	Original Periodic Masking MSE	(+) Half Cycle MSE	(-) Half Cycle MSE
96	**0.268 ± 0.0151**	*0.287 ± 0.037*	0.293 ± 0.04
192	**0.295 ± 0.0298**	*0.309 ± 0.05*	0.313 ± 0.057
336	0.319 ± 0.0185	*0.316 ± 0.022*	**0.311 ± 0.013**
720	0.356 ± 0.0310	*0.343 ± 0.035*	**0.340 ± 0.040**

We observe a similar pattern here as well, with the performance improving as the prediction window increases when we sample from either the positive or negative cycle.

As shown in the tables above, frequency and phase values clearly impact model performance. The new strategies reduce frequency or phase-related randomness among the variates of the dataset, resulting in more consistent values. This appears to enhance the model's ability in long-term forecasting.