DCSF: Deep Convolutional Set Functions for Classification of Asynchronous Time Series

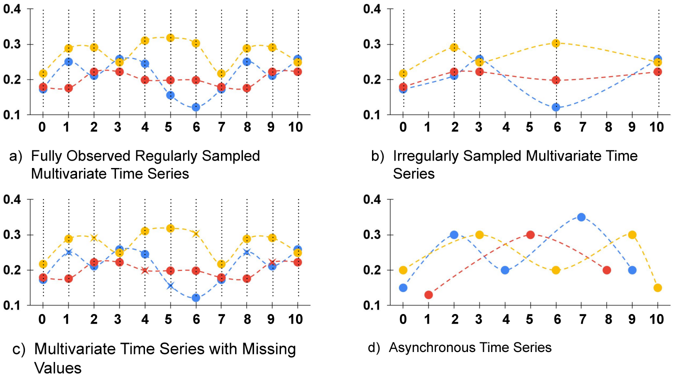
Vijaya Krishna Yalavarthi, Johannes Burchert, Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim
Germany
{yalavarthi, burchert, schmidt-thieme}@ismll.uni-hildesheim.de

***Abstract*—Asynchronous Time Series is a multivariate time series where all the channels are observed asynchronously-independently, making the time series extremely sparse when aligning them. We often observe this effect in applications with complex observation processes, such as health care, climate science, and astronomy, to name a few. Because of the asynchronous nature, they pose a significant challenge to deep learning architectures, which presume that the time series presented to them are regularly sampled, fully observed, and aligned with respect to time. This paper proposes a novel framework, that we call <u>D</u>eep <u>C</u>onvolutional <u>S</u>et <u>F</u>unctions (DCSF), which is highly scalable and memory efficient, for the asynchronous time series classification task. With the recent advancements in deep set learning architectures, we introduce a model that is invariant to the order in which time series' channels are presented to it. We explore convolutional neural networks, which are well researched for the closely related problem-classification of regularly sampled and fully observed time series, for encoding the set elements. We evaluate DCSF for AsTS classification, and online (per time point) AsTS classification. Our extensive experiments on multiple real world and synthetic datasets verify that the suggested model performs substantially better than a range of state-of-the-art models in terms of accuracy and run time. We increase the accuracy of the mini-Physionet dataset upto 2%; real datasets with synthetic setups of both AsTS, and TSMV upto 30%.**

I. Introduction

With the increase in industrialization, and new technologies, multivariate time series (MTS) datasets are becoming ubiquitous in the modern world. Traditional deep learning models for multivariate time series classification are developed for fully observed and regularly sampled multivariate time series (RMTS) where all the variables (channels) are observed simultaneously at regular frequencies (Figure 1(a)). Multivariate time series where the variables are observed simultaneously but not at regular intervals are called irregularly sampled multivariate time series (IMTS) (Figure 1(b)). Another domain of study deals with the classification of time series that are sparse, where one or more variables are not observed at a given time point.

In some time series applications, sparsity can be introduced by missing observations, called time series with missing values (TSMV) (Figure 1(c)). Missing values in time series are created, for example, due to sensor malfunctioning, power failures, and external physical interventions. However, in domains like medical applications [1], the sparsity could be created because of asynchronous observations of the sensors which we call asynchronous time series (AsTS). The sensors are observed independent of each other, making the series extremely sparse when one tries to align them at fixed time points [1]. In general, AsTS have irregularly observed channels with variable lengths, and in some physiological datasets [2], there might be unobserved channels as well.



a) Fully Observed Regularly Sampled Multivariate Time Series

b) Irregularly Sampled Multivariate Time Series

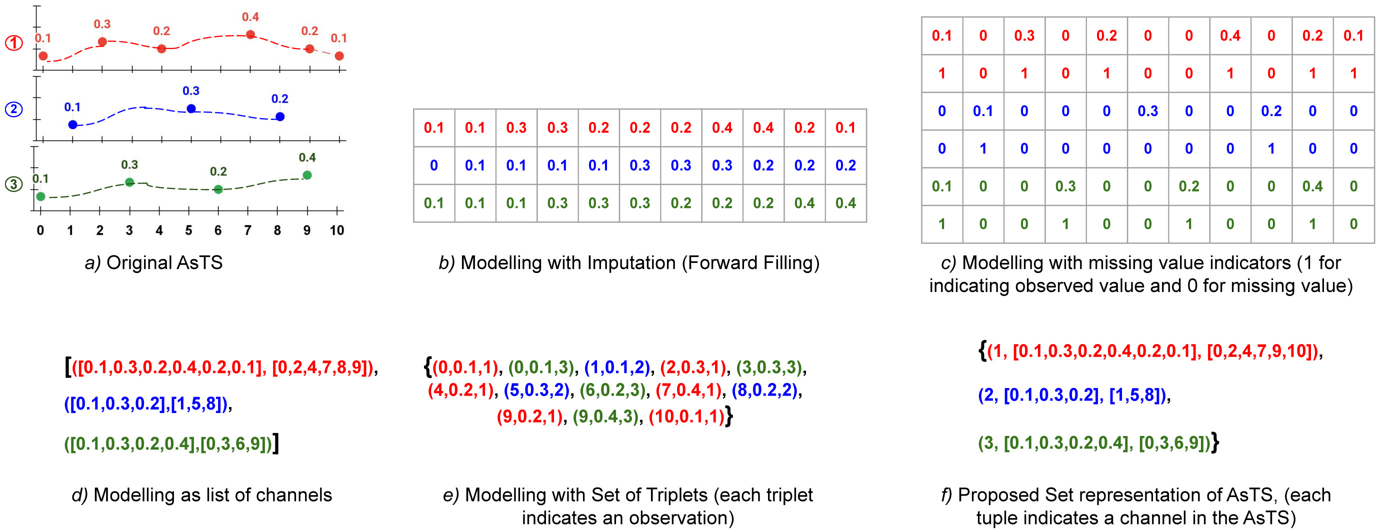
c) Multivariate Time Series with Missing Values

d) Asynchronous Time Series

Fig. 1: Demonstration of Fully observed multivariate time series (a), Irregularly Sampled Multivariate Time Series (b), Multivariate Time Series with Missing Values (c) and Asynchronous Time Series (d). observed measurements are marked with '•' and the missing values are marked with '×'.

In Figure 1, we delineate the differences among all the four categories of time series. Researchers often refer to AsTS with IMTS because the samples are observed at irregular intervals. However, irregularity can happen in Univariate Time Series as well while AsTS is specific to MTS. Because of the presence of sparsity, AsTS and TSMV are studied under the same umbrella [3]. But, the qualitative difference between them might lead to performance issues in AsTSC when one uses the models that are specific to TSMV. Though one can model AsTS as a time series with missing values, the absence of an observation in AsTS may carry its own information [4], and hence, imputation schemes used for missing value time series are not always useful. This work aims at the Classification of Asynchronous Time Series AsTSC.

Standard Time Series Classification models [5], [10], [11] assume that either the time series presented to the model is univariate time series or RMTS, and face significant challenge in learning AsTS. In order to circumvent the problem, most of the related work model AsTS into fixed dimensions, either by



a) Original AsTS

0.1	0.1	0.3	0.3	0.2	0.2	0.2	0.4	0.4	0.2	0.1
0	0.1	0.1	0.1	0.1	0.3	0.3	0.3	0.2	0.2	0.2
0.1	0.1	0.1	0.3	0.3	0.3	0.2	0.2	0.2	0.4	0.4

b) Modelling with Imputation (Forward Filling)

0.1	0	0.3	0	0.2	0	0	0.4	0	0.2	0.1
1	0	1	0	1	0	0	1	0	1	1
0	0.1	0	0	0	0.3	0	0	0.2	0	0
0	1	0	0	0	0	0	0	1	0	0
0.1	0	0	0.3	0	0	0.2	0	0	0.4	0
1	0	0	1	0	0	1	0	0	1	0

c) Modelling with missing value indicators (1 for indicating observed value and 0 for missing value)

[([0.1,0.3,0.2,0.4,0.2,0.1], [0,2,4,7,8,9]),
([0.1,0.3,0.2],[1,5,8]),
([0.1,0.3,0.2,0.4],[0,3,6,9])]

d) Modelling as list of channels

{(0,0.1,1), (0,0.1,3), (1,0.1,2), (2,0.3,1), (3,0.3,3),
(4,0.2,1), (5,0.3,2), (6,0.2,3), (7,0.4,1), (8,0.2,2),
(9,0.2,1), (9,0.4,3), (10,0.1,1)}

e) Modelling with Set of Triplets (each triplet indicates an observation)

{(1, [0.1,0.3,0.2,0.4,0.2,0.1], [0,2,4,7,9,10]),
(2, [0.1,0.3,0.2], [1,5,8]),
(3, [0.1,0.3,0.2,0.4], [0,3,6,9])}

f) Proposed Set representation of AsTS, (each tuple indicates a channel in the AsTS)

Fig. 2: Demonstration of various modelings of Asynchronous Time Series.

aligning or using non-linear functions. They apply recurrent neural networks (RNNs) for classification [3], [6]–[9], because they can accommodate series with variable lengths. RNNs are well suited for learning the dynamics of the time series for the forecasting tasks. On the other hand, deep learning models with convolutional layers are dominant in solving a closely related problem: classification of RMTS [5]. We show that by relaxing the constraint that the classification shall be performed on the synchronized channels of AsTS, using CNNs can significantly improve the accuracy of the AsTSC. In fact, in [12], Le et. al., proposed a shapelet-based model that classify RMTS by separating the channels.

Our model, Deep Convolutional Set Functions (DCSF), treats AsTS as a set of channels, and classifies it following the Deep set learning architecture [13]. For this, we represent an AsTS as a set of tuples, where each tuple indicates a channel in the AsTS. In DCSF, first, we encode variable length channels into a fixed dimensional vector in a latent representation with an encoder function built using convolutional layers. Following that, we aggregate the vector representations of all the channels and classify the aggregation using a decoder. Proposed DCSF is evaluated on Physionet2012, mini-Physionet, and MIMIC datasets for the task of AsTS classification, and used Human activity dataset for online classification of AsTS (classification per time point). Further, we demonstrate the performance of DCSF on synthetic AsTS datasets created from RMTS datasets. Additionally, because one can model AsTS and TSMV in a unified manner, we verified the performance of the proposed model for TSMV using real world RMTS datasets with a synthetic setup.

The main contributions of the proposed work are:

- We propose a novel representation of Asynchronous Time Series; set of the Asynchronous Time Series' channels.
- We show how to apply deep set functions for time series on channels rather than individual observations.
- We perform extensive experimental analysis over multiple real and synthetic datasets; compare with state-of-the-art models for the AsTSC. We improve the accuracy of the AsTSC by 2% for the mini-Physionet2012 dataset compared to state-of-the-art models with 6 times faster run time. We improve the accuracy of the real datasets with synthetic setups of both AsTS, and TSMV upto 30%. Experimental results attest to the superiority of the proposed model.

II. Preliminaries

In this section, we explain the various modeling strategies of Asynchronous Time Series (AsTS); following that, we show the proposed set representation of AsTS.

A. *Modeling AsTS*

In AsTS, each variable is observed independently, and the observation times are not synchronized, making the length of each channel different from others. Hence, it poses a significant challenge to deep learning models that operate on equal channel lengths. In order to train a deep learning model for AsTS data, we need to model all the channels in to equal length, or to a fixed dimensional vector. For this, there have been various methodologies that are studied in the literature.

Data imputation: One can consider AsTS as a TSMV by discretizing the time into non-overlapping intervals and considering the nonexistent measurement as a missing value. Following that, these missing values are imputed using various imputation techniques. In Figure 2 (a), we present an asynchronous time series where three channels are observed independently. By performing forward imputation, where the missing value is imputed with the last observed value in the channel, we achieve the fully observed time series as presented in Figure 2 (b).

Missing value indicators: Finding a suitable imputation scheme for filling the missing values in the observation space is difficult. Hence, researchers have filled the missing

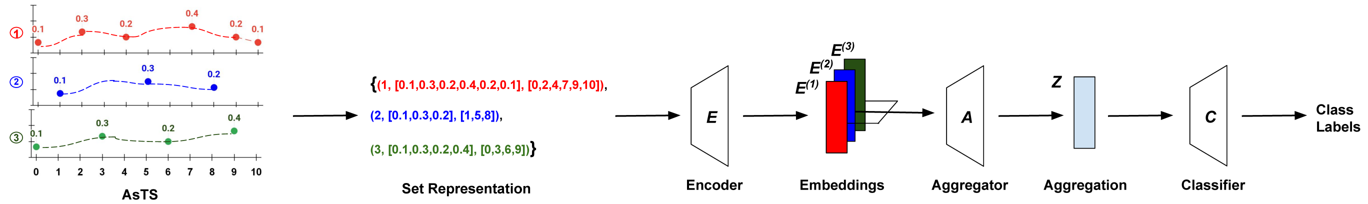


Fig. 3: Proposed overall methodology

observation with a constant value, generally zero, and provide a missing value indicator as a separate channel. It is assumed that the model understands the position of the missing observation through its indicator, and impute it in the latent domain. In Figure 2 (c), we can see the representation of AsTS with missing value indicators. Along with the missing value indicator, some models [3], [6] use time information as well for modeling AsTS.

List of channels: [8], [9] represents the AsTS as the list of channels consisting of time and observed values, as shown in Figure 2 (d). In this representation, the position of the channel is important because each channel is encoded by an independent encoder [9]. It is the closest representation to the raw data of AsTS.

Set of Triplets: [14] represented the AsTS as a set of observations where each observation is in a triplet form, which we call triplet notation. Each triplet carries the information of the time, value, and channel indicator. A differential set learning function is used for classifying the set. The triplet set representation is presented in Figure 2 (e).

Imputation methods are not desired when the missing observations carry the information of their own [4]. Also, asynchronous time series come with high sparsity (our datasets have a sparsity of around 85%), and imputing such data leads to bad predictions (we show empirically in Section IV. The main disadvantage of the triplets' set is that the time dependency in the series is not explicit, but has to be learned using the time information provided which is difficult, and fails when the sub-sequences carry more information of classification rather than individual observations. On the other hand, using separate encoders for every channel in list of channels approach [9] is computationally expensive.

Hence, we propose a representation of AsTS (trivial to extend to any MTS) which is a set of channels as shown in Figure 2(f). Unlike in [9], we share the parameters of the CNN model to encode all the channels which requires less number of parameters and share the knowledge of time dependent information among the channels. Also, because we encode entire channel rather than a single observation causal nature of the series is preserved.

B. Proposed set of channels representation for Multivariate Time Series

In contrast to the set representation in [14], where instances are represented as sets of single time slices, we represent AsTS as set of channels, each channel being an irregular one-dimensional time series. Then the classification of AsTS becomes a classification of a set of channels.

An AsTS, X, with D many channels can be represented as a set of D many elements: $X = \{s_1, ..., s_d, ..., s_D\}$. Each of its element is a tuple $s_d = (M_d, V_d, T_d)$ of i) a channel indicator $M_d \in \mathbb{N}^P$ for channel $d \in \{1, 2, ..., D\}$, where P is the dimension of the channel indicator, e.g., $P = D$ for one hot encoding, $P = \log_2 D$ for binary encoding, or $P = 1$ for nominal encoding, ii) an observation vector $V_d \in \mathbb{R}^*$ consisting of the values observed in channel d; and iii) a time vector $T_d \in \mathbb{R}^{+^*}$ containing the corresponding observation times of the values in V_d. We construct T_d in a monotonically increasing fashion to preserve the causal properties of the series. In our representation, the lengths of the vectors T_d and V_d are always equal ($|T_d| = |V_d|$). We denote the domain of these set elements by $\Omega := \mathbb{N}^P \times \mathbb{R}^* \times \mathbb{R}^{+^*}$ with $P \in \mathbb{N}$.

A time series X is considered to be fully observed if and only if $\forall d, e \in \{1, 2, ..., D\}, T_d = T_e$, meaning all the variables are observed at any given time point. Similarly, a time series X is considered to be sparse, if there exists at least two variables d and e such that $T_d \neq T_e$. Note that our representation covers both, TSMV and AsTS, although they are qualitatively different. In Figure 2 (f), we demonstrate our set representation for the AsTS of Figure 2 (a).

Problem 1 (Asynchronous Time Series Classification)**.** *Given i) a data set $\mathcal{D}^{train}$ of elements $(X, Y) \sim \rho$ sampled from an unknown distribution ρ on $\Omega^* \times \{0,1\}^L$ ($L \in \mathbb{N}$), and ii) a loss function $\mathcal{L} : \{0,1\}^L \times \mathbb{R}^L \rightarrow \mathbb{R}$, e.g., cross entropy, find a model $f : \Omega^* \rightarrow \mathbb{R}^L$ with minimal expected loss:* $\min_{(X,Y)\sim\rho} E(\mathcal{L}(Y, f(X)))$.

III. Proposed Model

Motivated by the deep set learning functions [13], we propose DCSF for the classification of AsTS. In the following, we explain the model architecture, and its building blocks.

A. Model Architecture

We define F as a function that operates on set elements, and its output is invariant to the order in which the set elements are presented. The proposed model has three different modalities: 1) Encoder (E), 2) Aggregator (A) and 3) Classifier (C). We formulate F for an AsTS instance X with D many set elements (channels) as follows:

$$F(X) = C(A(E(s_1), ..., E(s_D))), \qquad s_d \in X \qquad (1)$$

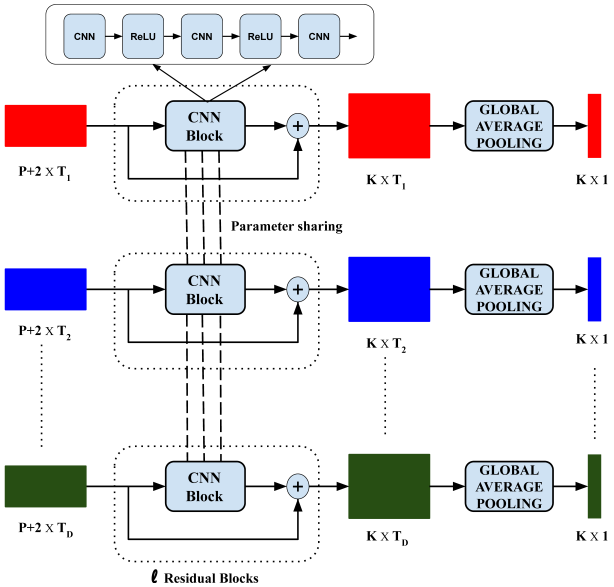
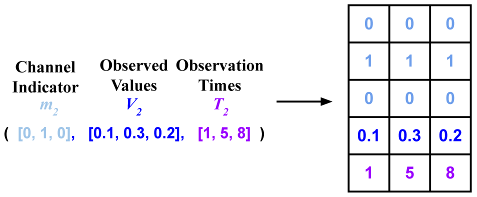


Fig. 4: Encoder (E) architecture. l residual blocks are used for extracting the latent embeddings from a set element (a channel of AsTS). Parameters are shared across all the set elements. We aggregate the variable length latent embedding into a fixed dimension using Global Average Pooling.



0	0	0
1	1	1
0	0	0
0.1	0.3	0.2
1	5	8

Fig. 5: Demonstration of conversion of set tuple into a multi-dimensional array. We show for the 2^{nd} channel in a series that contain a total of three channels ($D = 3$). One hot encoding (P=3) is used as channel encoder.

where $E : \Omega \rightarrow \mathbb{R}^K$, $A : \mathbb{R}^{K\times D} \rightarrow \mathbb{R}^K$and $C : \mathbb{R}^K \rightarrow \mathbb{R}^L$ where $K \in N^+$ is the dimensionality of the latent representation of a set element $s_d \in X$. The overall architecture of the proposed model is presented in Figure 3.

Encoder (E)

Our encoder E takes a set element as input, and provides a fixed dimensional representation $\mathbb{R}^K$. Since the length of every channel varies in AsTS, we require a model whose parameters do not depend on the length of the series presented to it. The widely used parameterized model architectures for this task are Recurrent Neural Networks (RNN), Attention Network (AtN), and Convolutional Neural Networks (CNNs). While RNN and AtN are useful for time series forecasting, state-of-the-art models for time series classification are built upon CNNs [5]. Hence, we choose a CNN based model as an encoder, and extract useful information from the channel observations. The proposed encoder architecture is presented in Figure 4. It comprises a series of l many residual blocks where each residual block consists of three convolutional layers with a ReLU activation function following each layer. In order to achieve fixed representation, one needs to aggregate the latent representation over time. For this, we employ global average pooling, which is widely used in CNNs based time series classification [5] models.

As mentioned in Section II, our set elements are tuples. Since our encoder cannot directly operate on tuples, we convert our set element s_d into a multi-dimensional array $\mathsf{s}_d \in \mathbb{R}^{(P+2)\times|T_d|}$ as shown in Figure 5. $\mathsf{s}_d^{:,t} = [M_d, V_d^t, T_d^t], t \in \{1, ..., |T_d|\}$ consists of 1) modality indicator of channel d, ($M_d \in N^P$), 2) observed value at t ($V_d^t \in \mathbb{R}$), and 3) its observation time ($T_d^t \in \mathbb{R}_+$). Our encoder E takes s_d as an input, and outputs K dimensional fixed representation of s_d. We consider K to be large enough in order to memorize the temporal locations of the signals present in s_d.

The proposed encoder share some similarity with the ResNET architecture provided in [5] in terms of using residual blocks, number of kernels, and kernel lengths in CNNs used in the residual block. However, a) we do not have a fixed number of residual blocks meaning, the number of residual blocks l is a hyper parameter, b) we do not use batch normalization after each residual block as it is hindering the performance of DCSF (refer Section IV-G).

Embeddings Aggregator (A)

Once the embeddings for each $s_d \in X$ are computed, we aggregate those embeddings $E(s_d)$ using the aggregator function A. There have been various kinds of aggregations provided in the literature such as mean aggregation [13], [15], sum aggregation [13], [15], and attention based aggregation [14]. In this work, we sum all the embeddings in order to aggregate them as shown in Equation 2 and represent the aggregated value with z. We empirically observed that the sum aggregation performs better than mean and attention based aggregation for our proposed model.

$$z = A\left(E\left(s_1\right), ..., E\left(s_D\right)\right) = \sum_{d=1}^{D} E(s_d), \quad s_d \in X \qquad (2)$$

Classifier (C)

After aggregation of the embeddings, we receive a vector representation $z \in \mathbb{R}^K$ of the multivariate time series. Then, one can use any model that is used for vector data classification. In this work, we use a fully connected artificial neural network.

As mentioned earlier, SEFT-ATTN presented in [14] also uses a deep set architecture for classifying time series. However, SEFT-ATTN uses triplet set modeling of time series, and those triplets are given as inputs to the encoder; an attention mechanism was used for aggregating the latent embeddings of the set elements. We observed that operating on individual observations has the disadvantage of losing sequence information even after providing time in the triplet. On the other hand, operating on sequences rather than observations not only preserves the causality of the sequence but also eases the learning process.

TABLE I: Statistics of the datasets used for AsTSC

Statistics	mini-Physionet	Physionet	MIMIC	Activity
Dataset Size	4000	12000	21107	6554
#Classes	2	2	2	11
Prevalence	14.6%	14.6%	13.2%	-
Sparsity	85.8%	85.7%	67.3%	75%
#Unobs. channels	10.2	10.2	2.67	0.01
Max chann. len.	170	189	246	22
Min chann. len.	1	1	1	0

B. Online classification scenario

For the online classification we need to predict the class label at every time point. Hence, we replace normal convolutions with causal convolutions, such that the current observation cannot look ahead into the future. Because we need to classify at every time point, we cannot implement global average pooling as it is implemented on entire sequence. Instead, we use causal average pooling (CAP) where at every time point we average the observations made until then, and aggregate those embeddings followed by classification using dense layer. The causal average pooling at time point τ for an array $\mathcal{E}$ is given by

$$CAP(\mathcal{E}^{\tau}) = \frac{1}{\tau}\sum_{i=1}^{\tau}\mathcal{E}^{i} \tag{3}$$

C. Supervised learning

We consider the parameters of the encoder E and the classifier C as θ and ϕ, respectively. Because we perform sum aggregation, we do not have any parameters for A. We follow [13], and train the model with all the channels present in time series. We optimize the following cost function:

$$\mathcal{J} := \mathbb{E}_{(X,Y)}\left[\mathcal{L}\left(Y, C\left(A\left(E\left(X;\theta\right)\right);\phi\right)\right)\right] \tag{4}$$

where, $\mathcal{L}$ is the loss function which is binary cross entropy for binary classification and Sigmoid cross entropy for multi-class classification tasks.

IV. Experiments

We implemented DCSF using TensorFlow and ran all the models on Nvidia GeForce RTX 3090 GPU nodes. In order to promote reproducibility, we outsource our source code in https://github.com/yalavarthivk/DCSF.

A. Asynchronous time series Datasets

Following the literature that deals with AsTSC problem, we chose three medical and one Human Activity Recognition dataset that contain asynchronous measurements. Basic statistics of the datasets are presented in Table I.

Physionet: The PhysioNet Challenge 2012 dataset [16], [17] comprises asynchronous time series data extracted from the records of patients admitted to ICU. Up to 37 variables were measured for the first 48 hours after the patient's admission, and all the time series have general descriptors like height, weight, age, and gender. The aim of this dataset is to predict whether a person dies in the hospital. The dataset comprises three different splits (a, b, and c) with 4000 observations each. Here, we combine all the splits, round the observed time points to the nearest minute, and split the 12,000 labeled observations randomly into 80% for training and 20% for testing.

mini-Physionet: Baseline models [7], [9] used only a part of the full dataset for the experiments. Hence, we would like to see how the model performs for a small part of the dataset. For this, we have taken split-c of the physionet dataset. It is also an imbalanced dataset with 4000 samples. Again, we randomly split 80% of the dataset into training, and 20% for testing.

MIMIC: The MIMIC dataset [2] also comprises records of ICU patients at Beth Israel Deaconess Medical Center. All the time series contain asynchronous measurements. Again, the task is to predict patient's mortality after the first 48 hours of ICU admission. We follow the procedures of [14], [18] for splitting the dataset that contains around 21,000 stays with 12 physiological variables being observed.

Activity: We use the human activity dataset for the online classification of Asynchronous Time Series. It contains the AsTS collected from five individuals wearing 3D sensors positioned in the belt, chest, and ankles while performing various tasks like sitting, laying, walking, etc. The dataset comprises $6,554$ time series with 12 channels, and channel consists of 50 time points. We follow the data preprocessing steps mentioned in [7], [9]. The task is to classify every time point in the series into 11 classes (activities).

B. Competing models

We use the following baseline models for the comparison.

GRU-Decay (GRUD) [3] proposed modifications to the hidden state of the GRU cell allowing a learnable decay rate to decay the past observations to the mean of the variable. **Phased-LSTM** [19] originally proposed for irregularly sampled time series, but not for unaligned measurements. We perform forward filling in order to handle the partially observed time points. **Interpolation Networks (IP-NETS)** [8] use several interpolation layers, followed by a GRU. The interpolation layers are learned along with a classifier in an end to end fashion. **LatentODE** [7] proposed model with the encoder as an ODE-RNN and decoder as a neural ODE. **Seft** [14] is a deep set model that uses triplet set modeling of time series. It extracts embeddings from all the observations, performs weighted average using an attention mechanism, and applies a fully connected network for classification. **mTAN** [9] proposed an attention-based interpolation for modeling AsTS into a fixed dimension. Later, the Variational AutoEncoder based encoder-decoder module was used for Classification and Reconstruction (auxiliary task) of AsTS. **ResNET** is a RMTS Classification model that uses CNNs presented in [5]. We fill the missing values with zeros and provide missing value indicators as additional channels in the series. **ResNET-forward** is also a ResNET model [5], we fill the unobserved value with the last observed one in the respective channel (forward filling). **DCSF** is our proposed model. In DCSF we use one hot encoding of variable as channel indicator. We normalized the observation times using min-max scaling.

TABLE II: Comparison of the results over the medical datasets namely Physionet2012, mini-Physionet2012, and Mortality (MIMIC) in terms of AUROC and run time for AsTSC task. Best results are presented in bold, and the second best are presented in italics.

Model	Physionet		mini-Physionet		MIMIC	
	AUROC	Time	AUROC	Time	AUROC	Time
GRU-D	86.3±0.3	104s	83.2±0.3	24s	*85.6±0.2*	430s
IP-NETS	86.5±0.2	14s	*84.2±0.4*	6s	84.2±0.5	297s
Phased-LSTM	79.6±0.4	70s	78.9±0.3	46s	78.9±0.4	452s
Latent-ODE	85.7±0.7*	3500*	82.9±0.4*	1200s	80.9±0.2*	4600*
SEFT-ATTN	86.0±0.4	7s	81.8±0.4	4s	85.0±0.2	29s
mTAN	*86.7±0.0*	6s	84.0±0.4	5s	83.3±0.1	55s
ResNET	84.2±0.7	8s	77.8±0.7	6s	81.5±0.6	25s
ResNET-forw.	82.6±0.9	10s	77.4±2.2	19s	82.3±1.4	51s
DCSF (Ours)	**87.1±0.2**	14s	**86.2±0.3**	1s	**85.8±0.1**	11s

C. Experimental Protocol

We randomly split off 20% of training data for validation, which is used for hyperparameter search and early stopping. For training, we use Adam optimizer with learning rate chosen from $[0.001, 0.00001]$ and the batch size from $\{32, 64, 128\}$. We also consider normalizing the time series as a hyperparameter because, for some datasets, normalization of the series before inputting to the models provides better validation results. We set the embedding length $K = 128$ as used in [5] in order to follow the universal function approximation for set functions presented in [20].

Following [14], we randomly generate 10 sets of hyperparameters for each of the competing models, and chose the setup with the best evaluation metric on the validation dataset. The models with selected hyperparameters are run independently for 5 times. For the medical datasets, we use area under receiver operating characteristic (AUROC) as the evaluation metric, because those datasets are heavily imbalanced. For the Activity dataset, we use accuracy as the evaluation metric.

D. Experiments on Asynchronous Time Series Classification

We present experimental results for classification of full AsTS using the medical datasets in Table II. The presented run time is the one that is taken by the model with best chosen hyperparameters. It could be possible that different hyperparameters may need different run times, but we use the current set up in order to be fair for all the models.

We can observe that our proposed model outperforms all the baselines in the Physionet and mini-Physionet datasets. Whereas, in the MIMIC dataset, our proposed model performs on par with the GRU-D model while outperforming all the remaining baselines. However, the run times of our proposed model are less than the GRU-D by an order of magnitude for the MIMIC dataset. Note that the second-best performing model is the Interpolation Networks, and the proposed model improves the AUROC of Interpolation Networks by around 2% in the mini-Physionet and MIMIC datasets. The Latent-ODE model could be slightly underperforming, because we run the dataset with the default hyperparameters provided in [7]. This is because of high computational complexity and run times. Especially, for the MIMIC and Physionet datasets, it took around an hour for each epoch, which is also observed in [14].

Further, we observe that it takes less time for the MIMIC dataset than for the Physionet dataset, even if the former is larger. The reason for this is: i) the total number of observations in a series (sum of lengths of all the channels in a series), which impacts the execution time, in Physionet is greater than MIMIC dataset, and ii) the choice of hyperparameters used: because we obtain different hyperparameters for different datasets through hyperparameter search, and the run time provided by each hyperparameter setup is different.

The results of the Physionet and MIMIC dataset are similar to that of the results published in [14], whereas the results on mini-Physionet dataset deviate significantly from the published results in [9]. Our results, in most cases, are better than the published ones because, unlike the procedure in [7], [9], we balance both the positive and negative class time series while sampling a batch for training in order to avoid bias towards the negative samples. Also, we could not reproduce the results published in [9] with the provided experimental setup and hyperparameters. We emailed the authors regarding the issue, and did not receive any information from them. The same issue has been raised by multiple people regarding their inability to reproduce the results for both Physionet and MIMIC datasets [1].

E. Experiments on Online Classification

The Activity dataset is an online classification dataset where one needs to provide class labels at every time point. While this is similar to segmentation in the time domain, since many researchers have explored this task, we want to see how the DCSF works in this scenario. From Table III, we observe that the DCSF outperforms all the baselines significantly, and the only model that performs close to ours is mTAN [9]. Since the source codes are not available for multiple baseline models, we did not run the experiments for the published baseline models, but took the results from [9]. Also, we could not compare the model in terms of run time because, the information is not available in the published works [9].

[1] https://github.com/reml-lab/mTAN/issues

TABLE III: Results for the Activity dataset

Model	Accuracy
GRU-D	86.2±0.5
IP-NETS	86.9±0.7
Phased-LSTM	85.5±0.5
Latent-ODE	87.0±2.8
SEFT-ATTN	81.5±0.2
mTAN	*91.0±0.2*
ResNET	88.7±0.2
ResNET-forw.	87.1±0.4
DCSF (Ours)	**91.3±0.1**

One important observation is that the Activity dataset does not heavily depend on the previous observations while predicting the class label at the current time point. CNN-based models under perform when the kernel lengths are larger than 1. Hence, we set the kernel length of CNNs to 1 for the CNN-based models, namely RESNET, RESNET-forward, and our DCSF.

F. Experiments on Regularly sampled Multivariate Time Series (RMTS) Datasets with asynchronous setup

Since the medical datasets were used by all the baseline models in their works, it is possible to have a model bias towards those datasets. Hence, we additionally use two RMTS datasets with synthetic AsTS setup for the comparison.

We consider two RMTS datasets, namely LSST and Phoneme Spectra [5], and induce artificial sparsity to them. These are among the 4 largest datasets used in [5]. The largest dataset Pen Digits, has very few time stamps (8) and channels (2) in a series. In the second largest dataset Face Detection, the accuracy of the best classifier in asynchronous setup is close to the default rate (50%) of the dataset. Hence, we choose third and fourth largest datasets.

TABLE IV: Comparison of the results over asynchronous setup on RMTS datasets: Phoneme Spectra and LSST. Sampled a single variable for every time point. Clas. full data is the classification result of the ResNET model [5] when the full (RMTS) dataset is used.

Model	Phoneme Spectra	LSST
GRU-D	08.2±0.4	41.8±0.8
IP-NETS	*11.6±0.5*	44.8±0.7
Phased-LSTM	04.3±1.2	40.7±0.6
Latent-ODE	09.3±0.7	38.5±0.5
SEFT-ATTN	10.4±0.4	*46.6±0.6*
mTAN	09.1±0.1	40.7±0.8
ResNET	10.2±1.2	**47.8±0.6**
ResNET-forward	10.1±1.0	47.3±1.2
DCSF (Ours)	**13.4±0.3**	46.1±0.5
Default rate	2.6	31.5
Class. full data	31.8	70.19

We use the following setup to generate AsTS datasets from RMTS. Because in an AsTS variables are observed independently, we assume that at every time stamp only one variable is observed. Hence, in the synthetic setup we choose one variable uniformly at random for a given time point. The total number of observations is equal to the length of the RMTS. For any two different samples, the observation times of a channel may not be consistent.

The experimental results on the datasets with synthetic asynchronous setup are presented in Table IV. Accuracy is the evaluation metric. For Phoneme Spectra, DCSF outperforms all the baseline models by a significant margin; the accuracy improvement is around 15% compared to the next best model, Interpolation Networks. For the LSST data, imputation based models, the ResNET-forward, and the RESNET perform slightly better than the DCSF. SEFT-ATTN has an accuracy gain over DCSF, but the difference is not statistically significant.

TABLE V: Comparison of the DCSF model with i) model that uses best channel for classification ii) ensemble model of all the channel classifiers. Evaluation metric is AUROC.

Dataset	DCSF	Single Channel	Ensemble
Physionet	87.1±0.2	83.0±0.1	76.4±0.0
mini-Physionet	86.2±0.3	83.2±0.2	71.0±0.2
Mimic	85.6±0.1	71.3±1.1	78.2±0.6
LSST	48.2±0.7	41.7±0.7	42.9±0.6
Phoneme Spectra	13.4±0.4	7.6±0.4	9.7±0.5

G. Ablation study

Comparison with models classifying single channel:

Since we separate the channels before inputting to the encoder, one might assume that the model is not learning from channel interactions, but just yields the accuracy of a channel that contributes maximum to the output. Hence, we perform an experiment by providing only a single channel to the model, and compared it with the proposed model. We ran the experiments with all the channels, and present the results for the best one in Table V. From the experimental results, we can conclude that the proposed model does learn from the cross channel interactions as it heavily outperforms the model that learns from only a single channel. One important observation is: for the mini-Physionet dataset, learning a single channel provides directly better results compared to many state-of-the-art models.

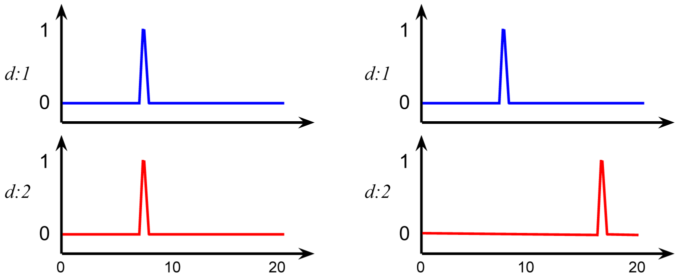
Comparison with an ensemble model:

In order to demonstrate that the aggregation after computing the latent embeddings is useful in learning the model, we compare the proposed model with an ensemble model. For this, we learn D models for D many channels present in the AsTS data. Each model takes the series from a single variable, and is trained for the classification. While testing, we take the average of encodings of the penultimate layer (while Softmax is the last layer), and compute the class label.

We present the results in Table V. We observe that the ensemble model provides better results than the model that learns with a single channel, except for the Physionet datasets. The reason for this phenomena is, the channels other than the best one do not provide useful result on their own. Hence, the linear combination used for ensemble, suppress the result of the best channel. Moreover, the proposed model outperforms the ensemble model by a significant margin, showing the advantage of aggregating the embeddings after the encoder.

Experiment to verify the importance of time information:

The purpose of having time information as a dimension in the encoder input is to capture the time dependent channel



a) Class 1

b) Class 0

Fig. 6: Toy data used for the ablation study to find the importance of time information in the encoder input. Positive class is shown in a) where peaks are observed at same time. Negative class is shown in b) where peaks are observed at different time points

TABLE VI: Comparison of the results on toy dataset for the proposed model with and without time information.

	10%	50%	90%
DCSF w\o time	91.1±0.9	76.4±0.8	59.5±0.6
DCSF (Ours)	**99.1±0.2**	**97.0±0.6**	**97.6±0.7**

interactions. Because we are encoding the channels separately, one might doubt that the proposed model is not capturing the channel interactions. In order to show that the time information provided in the encoder input serves the purpose, we performed an ablation study with a toy dataset, where the time information, or channel interactions over time is absolutely necessary for classifying it.

In the toy dataset, each time series has two channels of length T, and they are initialized with 0 for all the time steps. For positive samples, we sample a time location $t \in \{0, 1, ..., T\}$, and substitute 1 for both the channels at time point t. Where as, for negative samples, we sample two locations $t_1, t_2 \in \{0, 1, ..., T\}(t_1 \neq t_2)$ and replaced 0 with 1 at t_1 for channel 1, and t_2 for channel 2 as shown in Figure 6. For this experiment, we set $T = 20$, and induced sparsity of 10%, 50%, and 90% without altering the signals.

We conducted experiments on the proposed model with and without time information, and the results are presented in Table VI. We can observe that the proposed model performs better with time information (DCSF) compared to the one without (DCSF w\o time). It shows that the proposed model can learn the channel correlation towards the output when explicit time information is provided to it. An interesting observation is, the proposed model without time can learn from the relative position of the signal when the data is less sparse. With an increase in sparsity, the performance reduces. We can see that for 10% and 50% sparsity, the model achieves 91% and 76% accuracy, respectively.

Experiments using various time embeddings:

Here, we study multiple time embeddings in order to embed time information into the model.

Absolute time information (TE-1) Here, we provide the normalized absolute time values directly to the model, and the embeddings are learned in the latent layers, which is the proposed modeling.

TABLE VII: Ablation study with various model configurations

Configuration	mini-Physionet	MIMIC
TE-1 (DCSF)	86.88	85.60
TE-2	86.35	85.40
TE-3	82.59	82.04
TE-4	86.31	85.12
DCSF+BN	84.69	85.15
DCSF+IE	86.35	85.42

Time differences (TE-2) We consider the distance between the current observation and the previous one in time instead of absolute time information.

Sinusoidal positional embedding (TE-3) Here, we implement sinusoidal positional embedding provided in [14], which is a variant of the one presented in [21].

No Time embedding (TE-4) It is interesting to see how the models work when there is no information about time is given to it. Hence, we did not provide any explicit time information to the model.

Results on the validation dataset are provided in Table VII, shows that the absolute time values are more helpful compared to all the other embeddings. Another observation is that, model provides good accuracy even with no explicit information. This makes us think that the datasets we are using may not have significant time dependency. However, when there is a time dependency, our model excels as shown in the previous section.

Experiments with different model configurations:

Here, we study multiple model configurations and the results for the MIMIC and mini-Physionet are presented in Table VII. It can be seen that the batch normalization (DCSF+BN) is not helpful in the proposed model. We compare DCSF with a model where each channel is trained with an independent encoder (DCSF+DE). When we use D many encoders for D many channels, the channel interactions entirely depend on the aggregated embeddings. In the DCSF, because of the shared parameters, the encoder also receive the channel interactions, and with very few parameters.

H. Experiments on time series with missing values dataset

As mentioned earlier, both time series with missing values and asynchronous time series can be modelled in a unified manner. Hence, it would be interesting to see the performance of the competing models in the missing value setup. For this, we remove $p\%$ of the observations from the RMTS. As an example, if an RMTS has 5 variables observed for a length of 50, there will be 250 observations. For every time series, we randomly remove $p\%$ of those 250 observations in order to create TSMV.

We present the results on the LSST, and Phoneme Spectra with $p \in \{10, 50, 90\}$ in Table VIII. With accuracy being the evaluation metric, the best results are presented in bold and the second-best in italics. We observed that the proposed model performs better among all the competing models. The next

TABLE VIII: Comparison of the results over two RMTS datasets namely Phoneme Spectra and LSST with missing value setup. Artificially induced $p\%$ sparsity to the dataset by randomly removing $p\%$ of the observations. We set $p = 10, 50, 90$.

Model	Phoneme Spectra			LSST		
	10%	50%	90%	10%	50%	90%
GRU-D	20.8±1.2	15.8±0.6	8.6±1.7	59.4±1.0	51.2±0.5	41.1±0.8
IP-NETS	21.1±0.6	16.7±0.6	10.4±0.7	61.8±0.2	59.0±0.3	41.0±1.2
Phased-LSTM	17.1±0.6	06.2±0.3	4.5±0.4	57.8±0.6	48.8±1.1	40.3±0.7
Latent-ODE	12.3±1.3	09.2±0.8	08.2±0.4	38.3±2.0	36.6±1.6	39.3±0.7
SEFT-ATTN	10.5±0.5	10.9±0.7	11.0±0.5	60.0±2.1	47.3±1.3	**46.7**±**0.7**
mTAN	11.9±0.6	11.8±0.5	7.4±0.3	39.9±1.7	39.1±0.4	35.7±0.9
ResNET	22.5±0.2	12.7±0.6	7.1±1.4	*65.0*±*1.4*	58.7±2.2	44.1±1.7
ResNET-forw.	*29.0*±*0.7*	*19.5*±*0.7*	*11.4*±*0.2*	64.5±2.2	*59.1*±*1.3*	*44.7*±*0.4*
DCSF (Ours)	**31.5**±**1.1**	**25.8**±**0.7**	**14.4**±**0.8**	**65.8**±**1.4**	**60.2**±**0.4**	43.8±0.9

best model is ResNET-forward which is an imputation model. For Phoneme Spectra, with $p = 50\%$, DCSF provides 29% better accuracy compared to baseline models. Results show that, compared to the medical datasets, the lifts in the synthetic setup are significant, because for the medical datasets, the required information for classification is provided, but the observations are not synchronized. Whereas for the synthetic setup, we randomly remove the values, making it difficult for the models to learn. *The results indicate that the proposed DCSF can be used for both AsTS and TSMV datasets.*

Again, for the setting of 90% sparsity for LSST dataset, DCSF performs worse than ResNET, ResNET-forward and SEFT-ATTN as seen in the synthetic asynchronous setup for the LSST dataset. We observe that with extreme sparsity, in the LSST dataset, the sequence information is destroyed and applying convolutions on the observations with large time gap is not useful.

V. Literature Review

This work focus on the problem of classification for asynchronous time series (AsTS) data where the variables of the series are observed independent of each other. We briefly discuss the recent works that studied this problem.

Though one can consider the classification of both time series with missing values (TSMV), and AsTS as closely related problems, there is a qualitative difference between them. Missing values are observed due to the malfunctioning of a sensor, whereas in AsTS all the sensors work independently. The time axis can be discretized into non-overlapping intervals, and consider the intervals with no observations as missing values [22] making AsTS extremely sparse. Data imputation schemes or missing value indicators are used for the classification. For example, [23] performed semi-supervised clustering of medical data using Gaussian mixture models. Later, [24] discretized the time axis into hour-long bins, aggregated the information, and passed it through an RNN along with the missing value indicators. Chen et. al. [3], proposed various methods by combining Gated Recurrent Units (GRUs), and imputation schemes along with the one that takes observed values, missing value indicators, and the time difference between two observations. Especially in GRU-Decay, the last observed value is decayed to the mean value that is learned while training. Even though these approaches can be implemented for AsTS classification, they depend on the imputation of the time series data in input or latent domains rather than directly using the data for classification.

Rather than modeling AsTS as a TSMV, researchers developed models that can directly work on AsTS. In [25], Chen et. al., propose a Variational Auto-encoder based model that uses a neural network based decoder model combined with a latent ordinary differential equation model, for the continuous time series. Time series data is modeled using a continuous time function in the latent domain by using a neural network on its gradient field. Later, Rubanova et. al. [7], proposed a latent ODE model using an ODE-RNN model as the encoder. The encoder uses neural ordinary differential equations to model the dynamics in the hidden state, and an RNN to update it when a new observation is presented to it. De Brouer et. al. [26], proposed a continuous time version of the GRU. In [6], a neural CDE model which is a continuous analog of an RNN, while Neural ODEs are of ResNET is proposed.

Other than the ODE models, there are interpolation models where the entire series is utilized (past and future observations). In [27], [28], authors propose a multi-directional RNN for the interpolation that considers near past and near-future observations at a given time point. Sukla et. al. [8], proposed the Interpolation Network Model, where multiple semi-parametric RBF functions are used for interpolation against a set of reference time points. In [9], an time attention mechanism, where observed time points are used as keys and the reference times as queries is proposed. A bidirectional RNN followed by a Variational Autoencoder is used for the classification of the series.

In [29] the observations are represented as index-value pairs sampled from a continuous but unobserved function to address the missing values. They propose an encoder-decoder framework to learn these sequences. In [30] Wang et. al. propose a time-aware Dual-Attention and Memory-Augmented Network leveraging a vector representation of the irregularly sampled data.

In [14], Horn et. al., proposed Set Functions for Time Series where they model time series as a set of observations, and use a deep set model for the classification. The latent embeddings of each observation are computed using dense layers, and an attention-based aggregation function that has polynomial computational complexity is utilized for aggregating those embeddings. Our proposed model also uses deep set representation of time series. Instead of considering each observation as a set element, we represent each channel as a set element. We use a ResNET [31] based model for extracting the latent embedding from the given set.

VI. Conclusions

In this work, we propose a novel yet simple model, Deep Convolutional Set Function (DCSF), for the classification of asynchronous time series. Specifically, we modify the triplets' set representation of time series into channels' set representation and use a deep sets prediction model for the classification. Moreover, we apply convolutional neural networks, which are state-of-the-art models for the classification of fully observed and equally sampled time series, as the encoder to extract the embedding of the set elements. Our approach, yields the state-of-the-art results on 4 real world and 2 synthetic datasets for the tasks of both time series classification and segmentation (online classification) tasks, while providing the better run times compared to a range of the state-of-the-art models. The accuracy gains upto 2% for medical datasets, and 30% for synthetic datasets shows the superiority of the proposed model.

Acknoledgement

This work was supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany, within the framework of the IIP-Ecosphere project (project number: 01MK20006D).

References

[1] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, "Mining electronic health records (ehrs) a survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–40, 2018.

[2] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[3] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[4] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*, vol. 793. John Wiley & Sons, 2019.

[5] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021.

[6] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," *arXiv preprint arXiv:2005.08926*, 2020.

[7] Y. Rubanova, R. T. Chen, and D. Duvenaud, "Latent odes for irregularly-sampled time series," *Advances in Neural Information Processing Systems 32*, 2019.

[8] S. N. Shukla and B. M. Marlin, "Interpolation-prediction networks for irregularly sampled time series," *International Conference on Learning Representations*, 2019.

[9] S. N. Shukla and B. M. Marlin, "Multi-time attention networks for irregularly sampled time series," *International Conference on Learning Representations*, 2021.

[10] S. Jawed, J. Grabocka, and L. Schmidt-Thieme, "Self-supervised learning for semi-supervised time series classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 499–511, Springer, 2020.

[11] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[12] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K.-P. Chun, and G. L. Wong, "Shapenet: A shapelet-neural network approach for multivariate time series classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 8375–8383, 2021.

[13] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," *Advances in Neural Information Processing Systems 30*, 2017.

[14] M. Horn, M. Moor, C. Bock, B. Rieck, and K. Borgwardt, "Set functions for time series," in *International Conference on Machine Learning*, PMLR, 2020.

[15] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami, "Conditional neural processes," in *International Conference on Machine Learning*, pp. 1704–1713, PMLR, 2018.

[16] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012," in *2012 Computing in Cardiology*, pp. 245–248, IEEE, 2012.

[17] 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*, vol. 101, no. 23, pp. e215–e220, 2000.

[18] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Scientific data*, vol. 6, no. 1, pp. 1–18, 2019.

[19] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," *Conference on Neural Information Processing Systems*, 2016.

[20] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne, "On the limitations of representing functions on sets," in *International Conference on Machine Learning*, pp. 6487–6494, PMLR, 2019.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[22] M. T. Bahadori and Z. C. Lipton, "Temporal-clustering invariance in irregular healthcare time series," *arXiv preprint arXiv:1904.12206*, 2019.

[23] B. M. Marlin, D. C. Kale, R. G. Khemani, and R. C. Wetzel, "Unsupervised pattern discovery in electronic health care data using probabilistic clustering models," in *Proceedings of the 2nd ACM SIGHIT international health informatics symposium*, pp. 389–398, 2012.

[24] Z. C. Lipton, D. Kale, and R. Wetzel, "Directly modeling missing data in sequences with rnns: Improved classification of clinical time series," in *Machine learning for healthcare conference*, pp. 253–270, PMLR, 2016.

[25] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.

[26] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "Gru-ode-bayes: Continuous modeling of sporadically-observed time series," *arXiv preprint arXiv:1905.12374*, 2019.

[27] J. Yoon, W. R. Zame, and M. Van Der Schaar, "Deep sensing: Active sensing using multi-directional recurrent neural networks," in *International Conference on Learning Representations*, 2018.

[28] J. Yoon, W. R. Zame, and M. van der Schaar, "Estimating missing data in temporal data streams using multi-directional recurrent neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 5, pp. 1477–1490, 2018.

[29] S. C.-X. Li and B. Marlin, "Learning from irregularly-sampled time series: A missing data perspective," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5937–5946, PMLR, 13–18 Jul 2020.

[30] Z. Wang, Y. Zhang, A. Jiang, J. Zhang, Z. Li, J. Gao, K. Li, C. Lu, and Z. Ren, *Improving Irregularly Sampled Time Series Learning with Time-Aware Dual-Attention Memory-Augmented Networks*, p. 35233527. New York, NY, USA: Association for Computing Machinery, 2021.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.