

A f_{prep} FUNCTION IN GRUODE TO DEAL WITH ASYNCHRONOUS PROBLEM

There are two types of missing values in the multivariate time series. First, for one-dimensional time series, irregularly sampled observations will lead to missing values in the time dimension. Second, at a particular observed time point, values from some dimensions can be observed, but values from other dimensions cannot be observed, resulting in missing values across variables. The ODE-based model can theoretically handle the missing values from the time dimension, but it is difficult to tackle the missing values across variables.

Some common methods to deal with missing values across variables are imputation (such as impute the mean) or interpolation (such as linear or cubic interpolation), but these methods have been shown to lead to suboptimal predictions (Che et al., 2018). GRUODE (Brouwer et al., 2019) proposed to impute the missing values with the predicted means, variances, and error (see equation 3), which is handled by a preprocessing function f_{prep} . We are now going to describe the details of the f_{prep} in this section. Indeed, we outline the outputs of f_{prep} step by step.

1. Given the hidden state $\mathbf{h}(t)$, compute the parameters $\theta = f_{obs}(\mathbf{h}(t))$. Here, f_{obs} maps \mathbf{h} to the estimated parameters of the observations distribution $\mu_{\mathbf{X}}(t)$ and $\sigma_{\mathbf{X}}(t)$. Note that $\theta = [\theta^1, \dots, \theta^D]$. In the case of Gaussian, θ^d contains the means and log-variances for dimension d of $\mathbf{X}(t)$.
2. Create a vector \mathbf{q}^d that concatenates θ^d with the observed value X_t^d and the normalized error term, which for the Gaussian case is $(X_t^d - \mu^d)/\sigma^d$, where μ^d and σ^d are the mean and standard deviation derived from θ^d .
3. Multiply the vectors \mathbf{q}^d by a dimension-specific weight matrix W^d and apply a ReLU non-linear function.
4. Zero all results that did not have an observation (by multiplying them with mask m_d).

As shown above, GRUODE use prediction values (mean, variance, and error) predicted by memories from previous time point to impute missing values across variables. However, this methods can only be applied on regression/forecast tasks, and is not applicable to classification tasks since the predicted class and cross entropy loss only be computed at the last time point. Our approach gives an alternative approach to dealing with the asynchronous problem, and our models can apply to both prediction and classification tasks.

B DETAILS ABOUT HM AND HV IN THE MARGINAL BLOCK

We propose two approaches when we construct the marginal blocks, namely the HM approach and the HV approach. The meanings of the notations are different. We are going to give the interpretations in this section.

We also elucidate the meanings through an example - suppose that the number of variables (denoted as D) is set to be 3 and the size of marginal memory of each variable (denoted as s) is set to be 10, then we have $\mathbf{X}_t = [X_t^1, X_t^2, X_t^3]^T$ and $\mathbf{h}^{M,1}(t), \mathbf{h}^{M,2}(t), \mathbf{h}^{M,3}(t) \in \mathbb{R}^{10 \times 1}$. The total dimensions of memories for all the variables (denoted as H) equals $s \times D$, which is 30.

B.1 INTERPRETATIONS OF THE NOTATIONS UNDER HM APPROACH

$\mathbf{h}^M(t) = [\mathbf{h}^{M,1}(t), \mathbf{h}^{M,2}(t), \mathbf{h}^{M,3}(t)]^T$. Here, $\mathbf{r}^{M,d}(t), \mathbf{z}^{M,d}(t)$, and $\tilde{\mathbf{h}}^{M,d}(t) \in \mathbb{R}^{3 \times 10}$.

Given D matrices $\mathcal{M}_1, \dots, \mathcal{M}_D$ such that each matrix is of dimensions $m \times n$, we can concatenate them as a new tensor of dimensions $D \times m \times n$. The resulting tensor is denoted as $[\mathcal{M}_1, \dots, \mathcal{M}_D]$. As a result, given that $W_{(r,z,h)}^{M,d} \in \mathbb{R}^{10 \times 1}$ and $U_{(r,z,h)}^{M,d} \in \mathbb{R}^{10 \times 10}$, $W_{(r,z,h)}^M = [W_{(r,z,h)}^{M,1}, W_{(r,z,h)}^{M,2}, W_{(r,z,h)}^{M,3}]^T$ is a tensor of sizes $3 \times 10 \times 1$ while $U_{(r,z,h)}^M = [U_{(r,z,h)}^{M,1}, U_{(r,z,h)}^{M,2}, U_{(r,z,h)}^{M,3}]^T$ is a tensor of sizes $3 \times 10 \times 10$.

Finally, we have $W_{(r,z,h)}^M \circledast \mathbf{X}(t) = [W_{(r,z,h)}^{M,1} X^1(t), W_{(r,z,h)}^{M,2} X^2(t), W_{(r,z,h)}^{M,3} X^3(t)]^T$ and $U_{(r,z,h)}^M \circledast \mathbf{h}^M(t) = [U_{(r,z,h)}^{M,1} \mathbf{h}^{M,1}(t), U_{(r,z,h)}^{M,2} \mathbf{h}^{M,2}(t), U_{(r,z,h)}^{M,3} \mathbf{h}^{M,3}(t)]^T$. See Figure 7 for a graphical interpretation for $W_{(r,z,h)}^M \circledast \mathbf{X}(t)$ (in this figure, we assume the batch size equals 256).

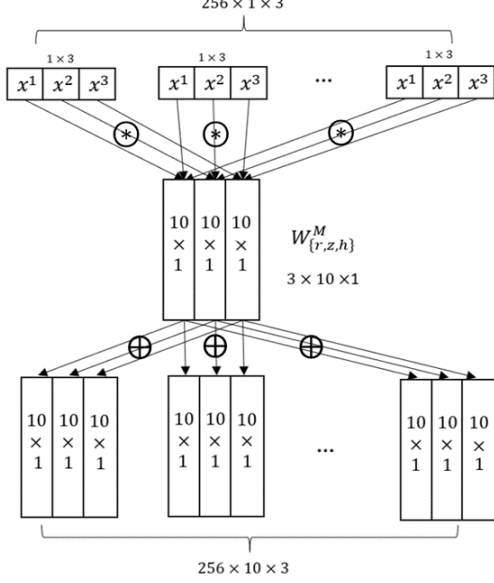


Figure 7: Visualization of HM approach

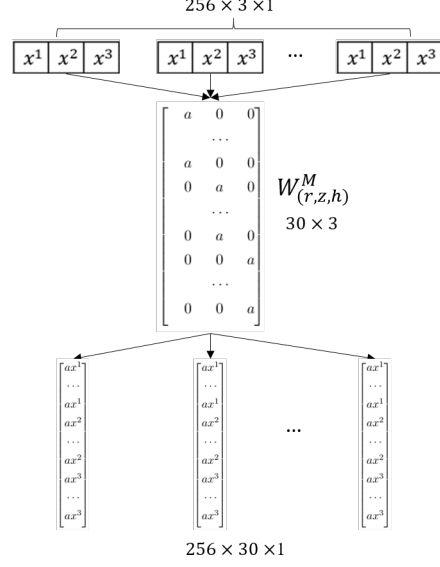


Figure 8: Visualization of HV approach

B.2 INTERPRETATIONS OF THE NOTATIONS UNDER HV APPROACH

The marginal memory $\mathbf{h}^M(t) \in \mathbb{R}^{30 \times 1}$ (30 is the product of $s = 10$ and $D = 3$). $W_{(r,z,h)}^M \in \mathbb{R}^{10 \times 3}$ and $U_{(r,z,h)}^M \in \mathbb{R}^{10 \times 10}$. We initialize $U_{(r,z,h)}^M$ as identity matrices, and $W_{(r,z,h)}^M$ as a block matrix $[A_{ij}]_{3 \times 3}^T$ such that the block A_{ij} is a matrix with all entries equal a if $i = j$ and 0 if $i \neq j$. Specifically,

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}^T = \underbrace{\begin{bmatrix} \overbrace{a \cdots a}^{10} & \overbrace{0 \cdots 0}^{10} & \overbrace{0 \cdots 0}^{10} \\ 0 \cdots 0 & \overbrace{a \cdots a}^{10} & 0 \cdots 0 \\ 0 \cdots 0 & 0 \cdots 0 & \overbrace{a \cdots a}^{10} \end{bmatrix}}_{30}^T \bigg\} 3.$$

See Figure 8 for a graphical interpretation for $W_{(r,z,h)}^M \circledast \mathbf{X}(t)$ (in this figure, we assume the batch size equals 256).

C DETAILED ARCHITECTURE OF CoGRUODE

Figure 9 shows the detailed architecture of our proposed model CoGRUODE. X^1 (red line) and X^2 (blue line) are two asynchronous time series. X^1 has observations at time points t_1 and t_2 and X^2 has observations at time points t_1 and t_3 . h^1 and h^2 are two memories that are induced by X^1 and X^2 . Each memory evolves by an ODE solver between two observed time points and updates when there is one observation for the corresponding variable. The dependence block extracts the information from the memories of each variable and forms the dependence memory.

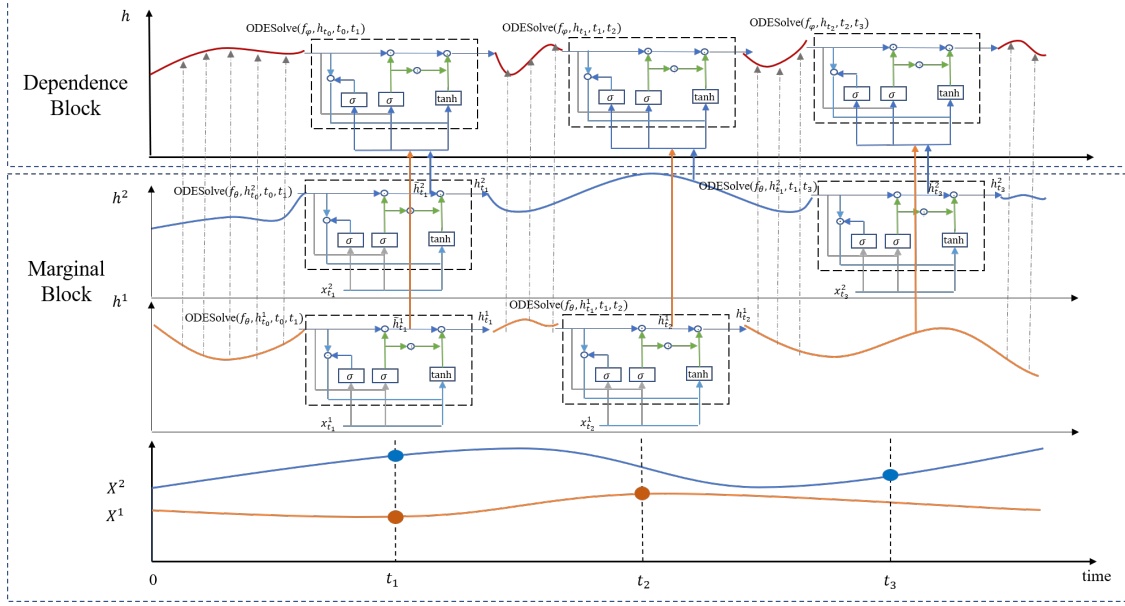


Figure 9: Detailed architecture of CoGRUODE

D ALGORITHMS OF GRUODE AND CoGRUODE

Since our model is based on the GRUODE (Brouwer et al., 2019), we display the difference of algorithm between GRUODE and our model as follows:

CoGRUODE	GRU-ODE-Bayes
Input: Observations: \mathbf{X}_t ; Observed time points: $\mathbf{t} = [t_1, \dots, t_K]$; Masks: \mathbf{m}_t ; time length: T Initialize: time = 0, $\mathbf{h}_0^M, \mathbf{h}_0^P$, and all trainable parameters Set: $\mathbf{h}^M = \mathbf{h}_0^M, \mathbf{h}^P = \mathbf{h}_0^P$ for $k = 1$ to K do $\mathbf{h}^M, \tilde{\mathbf{h}}^M = \text{GRU-ODE-M}(\mathbf{h}^M, \text{time}, t_k)$ <i>% marginal memory evolves to t_k</i> $\mathbf{h}^P = \text{GRU-ODE-P}(\mathbf{h}^P, \tilde{\mathbf{h}}^M, \text{time}, t_k)$ <i>% dependence memory evolves to t_k</i> time = t_k $\mathbf{h}^M, \tilde{\mathbf{h}}^M = \text{GRU-Bayes-M}(\mathbf{h}^M, \mathbf{X}_t, \mathbf{m}_t)$ <i>% marginal memory updates at t_k</i> $\mathbf{h}^P = \text{GRU-Bayes-P}(\mathbf{h}^P, \tilde{\mathbf{h}}^M, \text{time})$ <i>% dependence memory updates at t_k</i> end for $\mathbf{h}^M, \tilde{\mathbf{h}}^M = \text{GRU-ODE-M}(\mathbf{h}^M, t_K, T)$ <i>% marginal memory evolves to end</i> $\mathbf{h}^P = \text{GRU-ODE-P}(\mathbf{h}^P, \tilde{\mathbf{h}}^M, t_K, T)$ <i>% dependence memory evolves to end</i> $\hat{\mathbf{X}}_t = \text{MLP}(\mathbf{h}^M, \mathbf{h}^P)$ <i>% prediction/classification with marginal and dependence memory</i> return $\mathbf{h}^M, \mathbf{h}^P, \hat{\mathbf{X}}_t$	Input: Observations: \mathbf{X}_t ; Observed time points: $\mathbf{t} = [t_1, \dots, t_K]$; Masks: \mathbf{m}_t ; time length: T Initialize: time = 0, $\mathbf{h}_0^M, \mathbf{h}_0^P$ and all trainable parameters Set: $\mathbf{h} = \mathbf{h}_0$ for $k = 1$ to K do $\mathbf{h} = \text{GRU-ODE}(\mathbf{h}, \text{time}, t_k)$ <i>% memory evolves to t_k</i> time = t_k $\mathbf{h} = \text{GRU-Bayes}(\mathbf{h}, f_{prep}(\mathbf{X}_t, \mathbf{m}_t, \mathbf{h}))$ <i>% memory updates at t_k</i> end for $\mathbf{h} = \text{GRU-ODE}(\mathbf{h}, t_K, T)$ <i>% memory evolves to end</i> $\hat{\mathbf{X}}_t = \text{MLP}(\mathbf{h})$ <i>% prediction/classification with memory</i> return $\mathbf{h}, \hat{\mathbf{X}}_t$

E MGRUODE AND CO-MGRUODE

Our proposed model, CoGRUODE, extends from GRUODE, which aims to capture the marginal structure and the dependence structure separately. (Brouwer et al., 2019) also propose mGRUODE. It is a continuous-time model which is built upon the minimal GRU (a variant of GRU (Zhou et al., 2016)), which reduces one more gate based on the classic GRU. Similar to the extension of GRUODE in the main paper, we can build a new model which has two parts: the marginal part, which captures the temporal information, and the dependence part, which captures the dependence information. The constructed model is therefore called ComGRUODE. The update formulae of minimal GRU are:

$$\begin{aligned} \mathbf{z}_t &= \sigma(W_r \mathbf{X}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r), \\ \tilde{\mathbf{h}}_t &= \tanh(W_h \mathbf{X}_t + U_h (\mathbf{z}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h), \\ \mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}. \end{aligned}$$

The above formula can be rewritten as

$$\Delta \mathbf{h}_t = (1 - \mathbf{z}_t) \odot (\tilde{\mathbf{h}}_t - \mathbf{h}_{t-1}).$$

The corresponding ODE is

$$\frac{d\mathbf{h}(t)}{dt} = (1 - \mathbf{z}(t)) \odot (\tilde{\mathbf{h}}(t) - \mathbf{h}(t-1)).$$

Therefore, like CoGRUODE, we can also design ComGRUODE based on mGRUODE. The update formulae for ComGRUODE are as follows.

In the marginal block:

ODE part:

$$\begin{aligned} \mathbf{z}^M(t) &= \sigma(W_z^M \otimes \mathbf{X}(t) + U_z^M \otimes \mathbf{h}^M(t) + b_z^M), \\ \tilde{\mathbf{h}}^M(t) &= \tanh(W_h^M \otimes \mathbf{X}(t) + U_h^M \otimes (\mathbf{z}^M(t) \odot \mathbf{h}^M(t)) + b_h^M), \\ \frac{d\mathbf{h}^M(t)}{dt} &= (1 - \mathbf{z}^M(t)) \odot (\tilde{\mathbf{h}}^M(t) - \mathbf{h}^M(t)). \end{aligned}$$

Bayes part:

$$\mathbf{h}^{M,d}(t_+) = \begin{cases} \mathbf{mGRU}(\mathbf{h}^{M,d}(t_-), \mathbf{X}_t^d), & \text{if } \mathbf{m}_t^d = 1 \\ \mathbf{h}^{M,d}(t_-), & \text{if } \mathbf{m}_t^d = 0 \end{cases}.$$

In the dependence block:

ODE part:

$$\begin{aligned} \mathbf{z}^P(t) &= \sigma(W_z^P \tilde{\mathbf{h}}^M(t) + U_z^P \mathbf{h}^P(t) + b_z^P), \\ \tilde{\mathbf{h}}^P(t) &= \tanh(W_h^P \tilde{\mathbf{h}}^M(t) + U_h^P \mathbf{h}^P(t) + b_h^P), \\ \frac{d\mathbf{h}^P(t)}{dt} &= (1 - \mathbf{z}^P(t)) \odot (\tilde{\mathbf{h}}^P(t) - \mathbf{h}^P(t)). \end{aligned}$$

Bayes part:

$$\mathbf{h}^P(t_+) = \mathbf{mGRU}(\mathbf{h}^P(t_-), \tilde{\mathbf{h}}^M(t)).$$

F EXPERIMENT DETAILS

F.1 LSST DATASET

Dataset description The Photometric LSST Astronomical Time Series Classification Challenge (PLAsTiCC) is a public dataset that aims to classify simulated astronomical multivariate time-series data (Bagnall et al., 2018). The dataset is generated by measuring the photon flux using 6 different astronomical filters. Based on the 6 physical processes, this task aims to classify celestial objects into 14 astronomical classes. This dataset is a regularly sampled dataset and the time length is 36. We normalize each feature across all samples in the dataset to be in $[0, 1]$ interval.

Goal This task is a classification task and our goal is to classify samples into 14 classes based on the observations.

Loss function Let N be the number of samples. For the n -th sample, y_n is an observed one hot vector of size 14. \hat{y}_n is an estimated one hot vector of y_n . The c -th entry of y_n and \hat{y}_n are denoted as $y_{n,c}$ and $\hat{y}_{n,c}$. $p_n = [p_{n,1}, \dots, p_{n,14}]^T$ such that $p_{n,c}$ represents the predicted probability of class c . Note that

$$\hat{y}_{n,c} = \begin{cases} 1, & c = \arg \max_{1 \leq c \leq 14} p_{n,c} \\ 0, & otherwise \end{cases}.$$

CE is defined such that $\mathbf{CE}(y_n, p_n) = -\sum_{c=1}^{14} y_{n,c} \log(p_{n,c})$. Thus, the loss is

$$\text{loss} = \frac{1}{N} \sum_{n=1}^N \mathbf{CE}(y_n, p_n).$$

Training We divide the total samples (4925) into 70% for training, 15% for validation, and 15% for test. The maximum number of Epochs is 100, and the parameters for the best model during training are saved. The batch size is 128. We use Adam as the optimizer and set the initial learning rate at 0.001. The adaptive learning rate approach is used during training, and if the validation accuracy does not decrease for 5 epochs, the learning rate will decrease by half. To prevent overfitting, the dropout is used, and the dropout rate is 0.5. For the ODE-based models, the ‘‘Euler’’ method is used as the numerical solver, and the integral step is 0.02.

Model metric All experiments are conducted five times, and the mean and standard deviation of accuracy are provided. In this experiment, we use the **accuracy** as the model metric as the 14 classes are balanced. The formula of accuracy is

$$\text{Acc} = \frac{1}{N} \sum_{n=1}^N \mathbf{1}_{\{y_n = \hat{y}_n\}}.$$

F.2 HUMAN ACTIVITY DATASET

Dataset description The human Activity dataset contains multivariate time series from five individuals performing various activities. The data is made up of 3d positions of tags attached to their left ankle, right ankle, belt, and chest. At each time point, only one tag records the position data, which is 3-dimensional. In this experiment, we follow all the data preprocessing steps from Rubanova et al. (2019). For ease of understanding, we restate their preprocessing steps as follows: (1) The timestamps of the measurement are rounded to 100 ms, and such discretization does not change the overall number of points; (2) The datasets from four tags (belt, chest, right ankle, left ankle) are combined into a single time series, and the combined dataset is split into partially overlapping intervals of 50 time points (with the overlap of 25 points); (3) Sequences from all individuals are combined into a single dataset; (4) The original dataset has 11 classes, but some classes correspond to very similar activities, which are hard to be distinguished. We combine the classes within the following groups: (‘‘lying’’, ‘‘lying down’’), (‘‘sitting’’, ‘‘sitting down’’), (‘‘standing up from lying’’, ‘‘standing up from sitting’’, ‘‘standing up from sitting on the ground’’). Therefore, the final dataset

Table 5: Missing rate in Human Activity Dataset

Variables	Missing Rate
Var ₁ , Var ₂ , Var ₃	0.7359
Var ₄ , Var ₅ , Var ₆	0.7418
Var ₇ , Var ₈ , Var ₉	0.7828
Var ₁₀ , Var ₁₁ , Var ₁₂	0.7393
Total	0.7500

includes 7 classes: “walking”, “falling”, “lying”, “sitting”, “standing up”, “on all fours”, and “sitting on the ground”.

We get a dataset of 6,554 sequences such that each time series contains 12 variables and 211 time points. In addition, we normalize the values of all time series in $[0, 1]$ interval.

In Table 5, we summarize the missing rate of each variable after preprocessing steps. The missing rate of a certain variable x is defined as

$$\text{Missing Rate}(x) = 1 - \frac{\text{the number of observed time points that variable } x \text{ can be observed}}{\text{the total number of observed time points}}.$$

Goal This task is a per-time classification task and our goal is to classify the activities into 7 classes at each observed time point.

Loss function Let N be the number of samples. Let K_n be the number of observed time points for sample n . For the n -th sample at time point k_n , y_{n,k_n} is an observed one hot vector of size 7. \hat{y}_{n,k_n} is an estimated one hot vector of y_{n,k_n} . The c -th entry of y_{n,k_n} and \hat{y}_{n,k_n} are denoted as $y_{n,k_n,c}$ and $\hat{y}_{n,k_n,c}$. $p_{n,k_n} = [p_{n,k_n,1}, \dots, p_{n,k_n,7}]^T$ such that $p_{n,k_n,c}$ represents the predicted probability of class c . Note that

$$\hat{y}_{n,k_n,c} = \begin{cases} 1, & c = \arg \max_{1 \leq c \leq 7} p_{n,k_n,c} \\ 0, & \text{otherwise} \end{cases}.$$

CE is defined such that $\text{CE}(y_{n,k_n}, p_{n,k_n}) = -\sum_{c=1}^7 y_{n,k_n,c} \log(p_{n,k_n,c})$. Thus, the loss is

$$\text{loss} = \frac{1}{N} \sum_{n=1}^N \frac{1}{K_n} \sum_{k=1}^{K_n} \text{CE}(y_{n,k}, p_{n,k}).$$

Training We divide the total samples (6,554) into 70% for training, 15% for validation, and 15% for the test. The maximum number of Epochs is 50, and the parameters for the best model during training are saved. The batch size is 64. We use Adam as the optimizer and set the initial learning rate at 0.0035 and weight decay at 0.001. The adaptive learning rate approach is used for training, and if the test accuracy does not decrease for 5 epochs, the learning rate will decrease by half. To prevent overfitting, the dropout is used, and the dropout rate is 0.3. For the ODE-based models, the “Euler” method is used as the numerical solver, and the integral step is 0.01.

Model metric All experiments are conducted five times, and the mean and standard deviation of accuracy are provided. In this experiment, we use the **accuracy** as model metric as the 7 classes are balanced. The formula of accuracy is

$$\text{Acc} = \frac{1}{N} \sum_{n=1}^N \frac{1}{K_n} \sum_{k=1}^{K_n} \mathbf{1}_{\{y_{n,k}=\hat{y}_{n,k}\}}.$$

F.3 PHYSIONET DATASET

Dataset description Physionet 2012 challenge consists of records from 12,000 ICU stays. All patients are adults who are admitted for a wide variety of reasons to cardiac, medical, surgical, and

Table 6: Missing rate in Physionet Dataset

Variables	Missing Rate	Variables	Missing Rate	Variables	Missing Rate	Variables	Missing Rate
Var ₁	0.5676	Var ₁₁	0.8935	Var ₂₁	0.8980	Var ₃₁	0.9740
Var ₂	0.9919	Var ₁₂	0.7920	Var ₂₂	0.9543	Var ₃₂	0.5117
Var ₃	0.9894	Var ₁₃	0.9562	Var ₂₃	0.6693	Var ₃₃	0.7160
Var ₄	0.9891	Var ₁₄	0.9545	Var ₂₄	0.6739	Var ₃₄	0.9987
Var ₅	0.9891	Var ₁₅	0.9386	Var ₂₅	0.6689	Var ₃₅	0.9928
Var ₆	0.9890	Var ₁₆	0.2359	Var ₂₆	0.9231	Var ₃₆	0.5473
Var ₇	0.9536	Var ₁₇	0.9513	Var ₂₇	0.9233	Var ₃₇	0.9565
Var ₈	0.9989	Var ₁₈	0.9722	Var ₂₈	0.9198		
Var ₉	0.9533	Var ₁₉	0.9543	Var ₂₉	0.9523		
Var ₁₀	0.5119	Var ₂₀	0.5175	Var ₃₀	0.8131		
Total	0.8430						

trauma ICUs (Silva et al., 2012). There are 41 variables in the original dataset, and we have excluded four time-invariant features: Age, Gender, Height, and ICUType. Thus, each patients has a set of up to 37 features (‘Weight’, ‘Albumin’, ‘ALP’, ‘ALT’, ‘AST’, ‘Bilirubin’, ‘BUN’, ‘Cholesterol’, ‘Creatinine’, ‘DiasABP’, ‘FiO2’, ‘GCS’, ‘Glucose’, ‘HCO3’, ‘HCT’, ‘HR’, ‘K’, ‘Lactate’, ‘Mg’, ‘MAP’, ‘MechVent’, ‘Na’, ‘NIDiasABP’, ‘NIMAP’, ‘NISysABP’, ‘PaCO2’, ‘PaO2’, ‘pH’, ‘Platelets’, ‘RespRate’, ‘SaO2’, ‘SysABP’, ‘Temp’, ‘TroponinI’, ‘TroponinT’, ‘Urine’, ‘WBC’). Following the data preprocessing from Rubanova et al. (2019), we round up the timestamps to one minute. Therefore, each time series can contain up to 2,880 points. We normalize each feature across all patients in the dataset to be in $[0, 1]$ interval.

In Table 6, we summarize the missing rate of each variable after preprocessing steps. The missing rate of a certain variable x is defined as

$$\text{Missing Rate}(x) = 1 - \frac{\text{the number of observed time points that variable } x \text{ can be observed}}{\text{the total number of observed time points}}.$$

Goal This task is a binary classification task, and our goal is to classify each patient into 2 classes based on the measurement of 48 hours of observations.

Loss function Let N be the number of samples. For the n -th sample, y_n is an observed one hot vector of size 2. \hat{y}_n is an estimated one hot vector of y_n . The c -th entry of y_n and \hat{y}_n are denoted as $y_{n,c}$ and $\hat{y}_{n,c}$. $p_n = [p_{n,1}, p_{n,2}]^T$ such that $p_{n,c}$ represents the predicted probability of class c . Note that

$$\hat{y}_{n,c} = \begin{cases} 1, & c = \arg \max_{1 \leq c \leq 2} p_{n,c} \\ 0, & \text{otherwise} \end{cases}.$$

CE is defined such that $\text{CE}(y_n, p_n) = -\sum_{c=1}^2 y_{n,c} \log(p_{n,c})$.

Thus, the loss is

$$\text{loss} = \frac{1}{N} \sum_{n=1}^N \text{CE}(y_n, p_n).$$

Training We use data from set A and set B (8,000 in total) as the training data and validation data, and data from set C (4,000 in total) as the test data. The maximum number of Epochs is 150, and the parameters for the best model are saved during the training process. The batch size is 500. We use Adam as the optimizer and set the initial learning rate at 0.0005 and weight decay at 0.001. The adaptive learning rate approach is also used for training, and if the test AUC-ROC does not decrease for 5 epochs, the learning rate will decrease by half. For the ODE-based models, the ‘Euler’ method is used as the numerical solver, and the integral step is 0.1.

Table 7: The ratio of Frobenius norm on three dataset across five experiments

	LSST (50% missingness)			Human Activity			Physionet		
	W_r^M	W_z^M	W_h^M	W_r^M	W_z^M	W_h^M	W_r^M	W_z^M	W_h^M
exp-1	0.9969	0.9955	0.9970	0.6545	0.8567	0.9597	0.9994	0.9993	0.9902
exp-2	0.9972	0.9966	0.9973	0.7097	0.8660	0.9656	0.9995	0.9993	0.9925
exp-3	0.9956	0.9952	0.9961	0.7041	0.8762	0.9641	0.9996	0.9995	0.9942
exp-4	0.9956	0.9941	0.9959	0.7495	0.8842	0.9674	0.9992	0.9991	0.9900
exp-5	0.9966	0.9958	0.9971	0.7442	0.8794	0.9682	0.9994	0.9993	0.9938

Model metric All experiments are conducted five times, and the mean and standard deviation of AUC-ROC and AUC-PR are provided. In this experiment, we use the **AUC-ROC and AUC-PR** as the model metric as the labels in the dataset are highly imbalanced (with the positive class rate around 14%).

G FURTHER EXPERIMENTS ON MARGINAL BLOCK

In the HV method, we claim that entires in the off-diagonal blocks will move slowly from 0 during the training process. In this study, we want to further explore the size of these values. We take out $W_{(r,z,h)}^M$ at the last epoch. We extract the diagonal blocks from $W_{(r,z,h)}^M$ and form a new matrix, i.e., $W_{(r,z,h)}^{diag} = [A_{11}, A_{22}, \dots, A_{DD}]^T$ (definition of $A_{ii}, i = [1, \dots, D]$ is defined in Appendix B).

Then, we propose to use $\|W_{(r,z,h)}^{diag}\|_F / \|W_{(r,z,h)}^M\|_F$ to measure the relative ratio of the entry size in the block-diagonal to whole matrices. $\|\cdot\|_F$ is the Frobenius norm. The results in Table 7 show that in the LSST and Physionet datasets, the off-diagonal entries are extremely small compared to the entries on the main diagonal. But in the Human Activity dataset, the off-diagonal entries are relative larger in $W_{(r,z)}^M$ compared to the entries on the main diagonal. It is reasonable because data from the ankle, belt, or chest for human activity (such as walking) are highly correlated. Predicting the activity class accurately needs all data from the four sensors simultaneously.

H COMPUTING INFRASTRUCTURE

All models are run on a NVIDIA V100s GPU with 32GB RAM and AMD EPYC 7742 64-Core processor. All models are implemented in Python 3.8. The versions of main packages of our code are: Pytorch 1.8.1+cu102, torchdiffeq: 0.2.2, Sklearn: 0.23.2, Numpy: 1.19.2, Pandas: 1.1.3, Matplotlib: 3.3.2.

I CODES AND DATASETS

All experiment codes are in <https://anonymous.4open.science/r/CoGRUODE-88C4>. All the datasets are public datasets which can be downloaded from:

- **LSST:** <http://www.timeseriesclassification.com/Downloads/LSST.zip>
- **Human Activity:** <https://archive.ics.uci.edu/ml/datasets/Localization+Data+for+Person+Activity>
- **Physionet:** <https://physionet.org/files/challenge-2012/1.0.0/set-a.tar.gz>, <https://physionet.org/files/challenge-2012/1.0.0/set-b.tar.gz>, <https://physionet.org/files/challenge-2012/1.0.0/set-c.tar.gz>