

TimEHR: Image-based Time Series Generation for Electronic Health Records

Hojjat Karami¹, Mary-Anne Hartley², David Atienza³ *Fellow, IEEE*, Anisoara Ionescu⁴

Abstract—Time series in Electronic Health Records (EHRs) present unique challenges for generative models, such as irregular sampling, missing values, and high dimensionality. In this paper, we propose a novel generative adversarial network (GAN) model, TimEHR, to generate time series data from EHRs. In particular, TimEHR treats time series as images by using 2D convolutional kernels and is based on two conditional GANs. The first GAN generates missingness patterns, and the second GAN generates time series values based on the missingness pattern. Experimental results on three real-world EHR datasets show that TimEHR outperforms state-of-the-art methods in terms of fidelity, utility, and privacy metrics.

Index Terms—Convolutional neural nets, Electronic health records, Generative adversarial networks, Irregularly sampled time series, Time series generation

I. INTRODUCTION

Electronic health records (EHRs) chart patients' interactions with the health system and contain critical information for improving services and supporting research. Data from these systems are routinely incorporated into machine learning and statistical models for clinical decision support on diagnostic and prognostic predictions, as well as for monitoring health and evaluating treatment response [1]. However, access to large-scale EHR datasets is challenging and governed by strict regulations on privacy and security (e.g. HIPAA and GDPR), meaning that many models are based on uncentric data with a high risk of poor generalizability [2].

Traditional approaches for anonymization can be complex and costly, often compromising the data's statistical integrity and failing to provide robust privacy guarantees [3, 4]. The use of synthetic data is thus emerging as a promising solution for optimizing the trade-off between privacy and statistical utility [5, 6].

Generative models, particularly Generative Adversarial Networks (GANs) [7], have shown great potential in producing distribution-preserving synthetic EHR data. Their effectiveness spans various data modalities, including images [8, 9], clinical

text [10, 11], discrete tabular EHRs (e.g. diagnoses and billing ICD codes) [12, 13], and time series [14, 15].

Time series data consists of sequences of observations collected over time, which can be either regularly sampled (e.g., hourly temperature measurements) or irregularly sampled, such as spontaneous clinical observations in a hospital setting. In clinical practice, missingness is often not random but rather follows structured patterns that can be informative of patient outcomes. Specifically, clinicians order tests based on prior probabilities informed by their experience and patient condition, leading to missingness that depends on unobserved factors—commonly classified as missing not at random (MNAR). This distinction is crucial because many existing models assume missing completely at random (MCAR) or missing at random (MAR), which may not reflect real-world clinical data [16, 17]. The clinical significance of MNAR lies in its impact on predictive modeling, as ignoring its structure can introduce biases. For example, critically ill patients undergo more frequent tests. A model would then interpret a high number of tests as an indicator of poor outcome irrespective of their actual value. Explicitly modeling MNAR helps create more realistic datasets, improving the generalizability of machine learning models in clinical settings.

The commonly used deep learning architectures for time series are Recurrent Neural Networks (RNNs) [18, 19], attention mechanism [20, 21], or one-dimensional Convolutional Neural Networks (1D-CNNs) [22, 23, 24] where 1D convolutional operations are applied over the time dimension to capture temporal patterns and relationships. However, image-based time series modeling based on 2D-CNNs has also been employed for time series forecasting [25, 26], classification [27, 28, 29], and anomaly detection [30]. Especially for irregularly sampled time series, ViTST aims to transform irregularly sampled time series into line graphs and employ the Vision Transformer (ViT) for the subsequent classification task [31].

We propose a novel GAN-based model, *TimEHR*, to generate synthetic EHR's time series data with irregular sampling and missing values. In particular, we treat each patient's time series as a 2-channel image (mask and values) and we use 2D-CNN architecture. TimEHR consists of two modules that are trained independently. The first module is a conditional Wasserstein GAN with gradient penalty (CWGAN-GP) that generates missing patterns (mask channel) from the noise and static data (e.g. demographics and outcome) as the conditional vector [32]. The second module is a Pix2Pix GAN [33] that generates time series values from the missing pattern and the static data as the conditional vector. Once the two modules

This work was supported by the European Union's Horizon 2020 research programme and innovation under the DIGIPREDICT project (GA 101017915).

¹Hojjat Karami is with the Signal Processing Laboratory 5 (LTS5), EPFL, Switzerland: (correspondence e-mail: hojjat.karami@epfl.ch)

²Mary-Anne Hartley is with the Laboratory for Intelligent Global Health & Humanitarian Response Technologies (LiGHT), Yale, USA.

³David Atienza is with the Embedded Systems Laboratory (ESL), EPFL, Switzerland.

⁴Anisoara Ionescu is with the Signal Processing Laboratory 5 (LTS5), EPFL, Switzerland

are trained, we can use pre-trained generators to generate time series data. The main contributions of this paper are as follows:

- TimEHR¹ can generate irregularly sampled time series with non-random missing values (MNAR). It is the first work to use image-based time series generation for EHRs.
- Our experiments on three large Electronic Health Record (EHR) datasets demonstrate that TimEHR outperforms state-of-the-art methods in terms of fidelity, utility, and privacy metrics.
- Our evaluation on simulated sinusoidal time series reveals that TimEHR is scalable to multivariate time series of up to length 128 and a number of variables up to 128, accommodating various missingness rates.

The code is available at <https://github.com/esl-epfl/TimEHR>.

II. RELATED WORKS

Regularly sampled time series. Most works on time series generation focus on regularly sampled time series with no missingness. TimeGAN [34] is one of the early works that is based on an AutoEncoder and a GAN module (AE-GAN) with RNN architectures to obtain a dense latent space within which to generate samples. DoppelGANger [35] is another RNN-based model that also handles metadata in a conditional GAN framework. Here, we do not review the details of these works as they are not inherently compatible with irregularly sampled time series.

Irregularly sampled time series. Generating irregularly sampled time series with missing values is an under-explored area. T-CGAN [22] uses a conditional GAN framework with 1D-CNN architecture to generate irregularly sampled time series values conditioned on the timestamps vector. However, the timestamp vector is randomly generated and cannot handle missing values. RTSGAN [36] is a GRU-based AE-GAN model in which an observation embedding and a novel decide-and-generate decoder are proposed to handle irregular sampling and missingness. GT-GAN [37] is also an AE-GAN method that utilizes various components, ranging from GRU-based neural ordinary/controlled differential equations to continuous time-flow processes. However, it can only generate irregularly sampled time series without missing values. EHR-Safe [14] is based on a two-stage model consisting of sequential encoder-decoder networks and GANs with stochastic normalization and explicit mask modeling ideas to improve utility and privacy. Another line of research explores the use of diffusion models for time series generation. TimeDIFF [15] uses Denoising Diffusion Probabilistic Models (DDPM) and outperforms various real-world EHR datasets in terms of data utility and privacy. TS-Diffusion [38] integrates the inhomogeneous Poisson process and observation probability into a diffusion model to overcome sample irregularity and missingness.

Among the mentioned works, RTSGAN, EHR-Safe, and TimeDIFF are the only models that can handle both irregular sampling and missingness and have been evaluated on EHR time series.

¹The code is attached to the submission

Image-based methods. The idea of converting time series data into images for generation purposes has already been explored. [39] proposed to convert single channel continuous time series to 64x64 images and use a Wasserstein GAN with gradient penalty (WGAN-GP) to generate synthetic time series. Using spectrogram images of time series data [40, 41] is another approach that works for regularly-sampled time series data. However, these methods are not designed for irregularly sampled time series with missing values.

III. TIMEHR

We begin by introducing the notation used in this paper. We then describe our proposed model in detail.

A. Problem Formulation

An EHR dataset can be denoted as $\mathcal{D} = \{(s_i, T_i)\}_{i=1}^N$ where N is the number of patients, s_i is the static data (e.g. demographics, outcomes), and T_i is the time series data such as vital signs and laboratory variables. The vector-based representation of each time series T_i is denoted by $T_i = \{(t_j, \mathbf{x}_j, \mathbf{m}_j)\}_{j=1}^L$. Here, L is the number of observations, $t_j \in \mathbb{R}_{\geq 0}$ is the time stamp, $\mathbf{x}_j \in \mathbb{R}^d$ is the vector of time series values (d variables) and $\mathbf{m}_j \in \{0, 1\}^d$ is the mask vector (1 if measured, 0 otherwise). The missing values in \mathbf{x}_j are replaced by zero. Our objective is to generate a synthetic EHR dataset \mathcal{D}_{syn} that is similar to the real EHR dataset \mathcal{D} . We simplify the problem of jointly generating static and time series data into two sub-problems: generating static data and generating time series data conditioned on the static data. This can be formulated as $P(s_i, T_i) = P(s_i)P(T_i|s_i)$. This is conceptually similar to RCGAN [42] and DoppelGANger [35] in which the authors proposed to first generate metadata and then generate time series conditioned on the generated metadata.

B. Time Series to Image

As we will treat time series data as images, we reshape each patient's time series data (T_i) into a two-channel image (I_i). For the first channel ($I_{i,value}$), we convert $\{(t_j, \mathbf{x}_j)\}_{j=1}^L$ into a matrix of shape $[d, H]$ where d is the number of variables and H is the number of time bins based on a desired time resolution (r). Here, $I_{i,value}[m, n]$ is the last available observation of m -th variable in the $((n-1)r, nr]$ interval (or zero if there is no observation). The value channel is min-max normalized to the $[-1, 1]$ range. The second channel ($I_{i,mask}$) is the corresponding mask matrix (-1 for missing values otherwise 1). It should be noted that our image is invariant to the order of variables and we can shuffle the variables to obtain a different image. Figure 1 illustrates the image-view of an example patient's time series data. This is a compact form of representing time series data that highlights the non-random missingness pattern.

C. Overall Architecture

The overall architecture of TimEHR is shown in Figure 2. The first module is a conditional Wasserstein GAN with

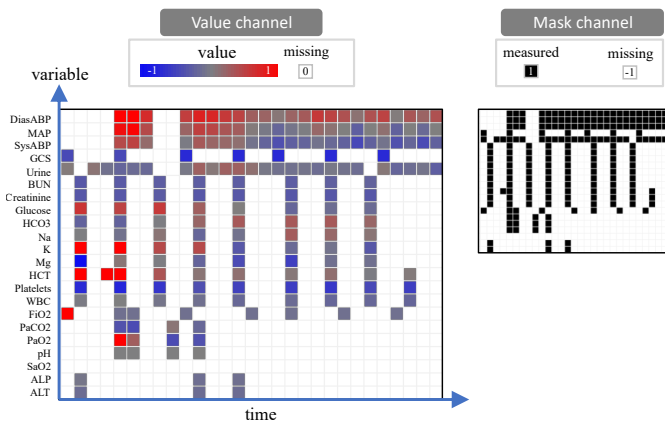


Fig. 1. An image-based representation of a patient's time series data. Colors are for visualization purposes only.

Gradient Penalty (CWGAN-GP) that generates synthetic samples (mask and values) from noise and a conditional vector (static data and label). The second module is a Pix2Pix GAN [33] that generates synthetic samples (only values) from the conditional vector and the mask channel from the training data. In the inference step (Figure 2-III), we use a Tabular GAN to generate static data, and the two trained generators to generate synthetic time series data.

Our primary innovation lies in the utilization of the reshape operation, transforming the time series into a two-channel image. Retaining the missing pattern as an additional channel facilitates the CWGAN-GP module in generating realistic missingness patterns. Despite employing 2-D convolutional kernels in our modules, the extensive layering in our CNN architecture ensures that the model's receptive field is sufficiently large to capture the temporal correlation structure of the variables.

D. Module 1: CWGAN-GP

We use a conditional Wasserstein GAN architecture with gradient penalty (CWGAN-GP) for generating images from noise and static data [32]. The generator G_1 is a convolutional neural network (CNN) that takes as input a Gaussian noise vector z ($z \sim \mathcal{N}(0, 1)$) and static data s , and outputs an image \tilde{I} . The discriminator D_1 is also a CNN that assigns scores for the generated and real images. The objective function of the CWGAN-GP is given by:

$$\begin{aligned} \min_G \max_D V(G_1, D_1) = & \mathbb{E}_{I \sim P_I} [D_1(I, s)] \\ & - \mathbb{E}_{z \sim P_z} [D_1(G_1(z, s), s)] \\ & + \lambda_{GP} \mathbb{E}_{\hat{I}} \left[(\|\nabla_{\hat{I}} D_1(\hat{I}, s)\|_2 - 1)^2 \right], \end{aligned}$$

where $\hat{I} = \epsilon I + (1 - \epsilon)G_1(z, s)$ and $\epsilon \sim U(0, 1)$. The first two terms represent the Wasserstein distance or Earth Mover's Distance (EMD) between the distribution of real and generated data. The last term is the gradient penalty that enforces the Lipschitz constraint on the discriminator. λ_{GP} is the gradient penalty coefficient.

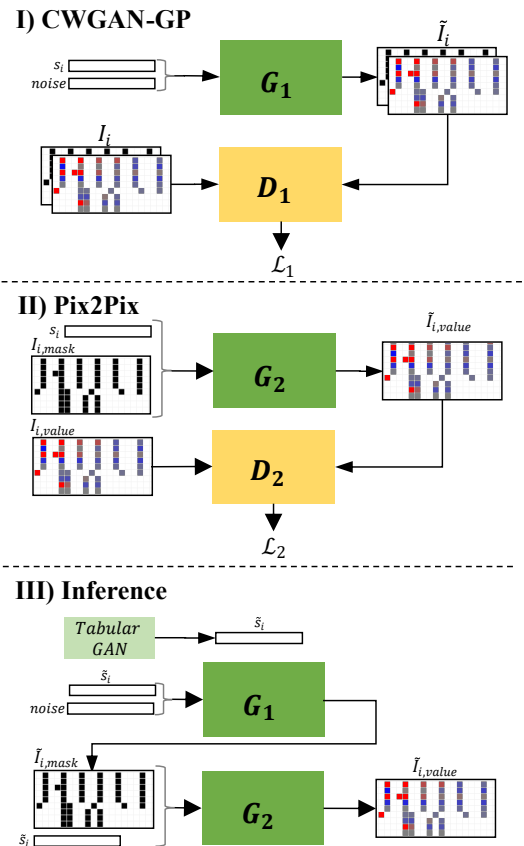


Fig. 2. Model architecture. Module 1: CWGAN-GP for generating mask, Module 2: Pix2Pix for generating values, and Inference: generating synthetic time series.

Architecture The generator G_1 consists of a series of convolutional layers with batch normalization and ReLU activation. We apply an embedding layer to the static data s and concatenate it with the noise vector z . The discriminator D_1 consists of a series of convolutional layers with batch normalization and LeakyReLU activation. We apply another embedding layer for the static data s and add it as an additional channel to the input image.

E. Module 2: Pix2Pix

We use a conditional Pix2Pix architecture to generate the value channel from the mask channel [33]. The generator G_2 takes as input a mask channel (I_{mask}) and static data s and outputs a value channel ($\tilde{I}_{value} = G_2(I_{mask}, s)$). The discriminator D_2 takes as input a value channel (I_{value} or \tilde{I}_{value}) and static data s and outputs a scalar value $D_2(I_{value}, s)$ that indicates the probability that the image is real. The objective function of the Pix2Pix is given by:

$$\begin{aligned} \mathcal{L}_2(G_2, D_2) = & \mathbb{E}_{I_{value}} [\log D_2(I_{value}, s)] \\ & + \mathbb{E}_{\tilde{I}_{value}} [\log(1 - D_2(\tilde{I}_{value}, s))]. \end{aligned}$$

The first and second terms are the log probability the discriminator correctly classifies the real data as real and the generated data as fake, respectively. Our primary experiments showed that including noise as input to the generator does

not help generate diverse value channels from a fixed mask channel which is consistent with the findings in [43, 33].

Earlier methods have discovered that it is advantageous to combine the GAN objective with a reconstruction loss such as the L2 distance [44]. While the role of the discriminator remains the same, the generator is now assigned the dual task of deceiving the discriminator and closely matching the actual output in terms of L2 similarity. We explore this idea by adding L2 loss to the objective function:

$$\mathcal{L}_{rec}(G_2) = \mathbb{E}_{I_{value}, \tilde{I}_{value}} [\|I_{value} - \tilde{I}_{value}\|_2].$$

Our final objective is:

$$\min_{G_2} \max_{D_2} V(G_1, D_1) = \mathcal{L}_2(G_2, D_2) + \lambda \mathcal{L}_{rec}(G_2).$$

Architecture The generator has a U-Net architecture with skip connections [45] with LeakyReLU activation in the down-sampling path and ReLU activation in the up-sampling path. The discriminator is a PatchGAN classifier similar to [33] that consists of several CNN layers of 2D convolutions, batch normalization and LeakyReLU activation. This discriminator attempts to classify each $P \times P$ patch in an image as real or fake, and only penalizes structure at the patch scale.

F. Inference (Time Series Generation)

Once the models in the previous steps are trained, we can use the trained generators (G_1 and G_2) to generate time series data from noise and static data. Based on conditional independence assumption in III-A, we are flexible to use any conditional tabular GAN such as CTGAN in [46] to generate static data. As the distribution of the label is important in downstream evaluation, we use the label as a conditional to generate other static variables. While the CWGAN-GP can independently generate realistic time series with missing data, we use only the mask channel as the input to the G_2 . The inference algorithm is shown in algorithm 1.

Algorithm 1: Generating Synthetic Time Series

Input: Size N , Generators G_1 and G_2
Output: synthetic dataset $\mathcal{D}_{syn} = \{(\tilde{s}_i, \tilde{T}_i)\}_{i=1}^N$
 Generate N static vectors $\{\tilde{s}_i\}_{i=1}^N$ using CTGAN
 $\mathcal{D}_{syn} = \{\}$
for $i = 1$ **to** N **do**
 generate noise z_i from $\mathcal{N}(0, 1)$
 Generate mask pattern using G_1 :
 $\tilde{I}_{i,mask} = G_1(z_i, \tilde{s}_i).mask$
 Generate values from mask pattern using G_2 :
 $\tilde{I}_{i,value} = G_2(\tilde{I}_{i,mask}, \tilde{s}_i)$
 Reshape $\tilde{I}_{i,mask}$ and $\tilde{I}_{i,value}$ into \tilde{T}_i
 $\mathcal{D}_{syn}.append((\tilde{s}_i, \tilde{T}_i))$
end for

IV. EXPERIMENTS

Datasets

We use three publicly available EHR datasets to evaluate TimEHR which enables reproducibility and fosters collaboration within the research community: (1) Medical Information Mart for Intensive Care III (MIMIC-III) [47], (2) the Physionet Challenge 2012 (P12) [48], and (3) the Physionet Challenge 2019 (P19) [49]. The overall missing rates are 80.37%, 74.58%, and 79.40% for P12, P19, and MIMIC-III, respectively and the main characteristics of the datasets are shown in Table I. We employ 5-fold cross-validation to evaluate the models. Since the number of variables and observations is less than 64, we pad time series images with zero to have an image of size $[2, 64, 64]$ (refer to VI for more details on data preprocessing).

TABLE I
DATASETS DESCRIPTION

Dataset	N	d	L_{max}	Outcome(Prevalence)
MIMIC-III	51k	48	62	Mortality(16.1%)
P12	12k	35	48	Mortality(14.2%)
P19	38k	32	64	Sepsis (7.2%)

Evaluation Metrics

We need a reasonable embedding of each time series data (T_i) to compute metrics. For each variable, we compute the common statistics (mean, standard deviation, min, max) and the missing rate, and concatenate them into a vector ($\mathbf{e}_i \in \mathbb{R}^{5d}$). We briefly describe the evaluation metrics below (more details on how these metrics are calculated are provided in VIII).

Fidelity We report precision, recall, density, and coverage (PRDC) metrics between the estimated manifold of real and generated data distributions [50]. Additionally, we define a new metric called *Temporal Correlation Difference (TCD)*, which is the $L1$ norm of the difference between the correlation matrices of the real and generated data normalized by $d(d - 1)/2$. To compute each correlation matrix, we first forward-fill time series value of each sample (T_i) and then compute the correlation matrix of the concatenated matrix of all samples along the time axis. We neglect the pairs if at least one of the variables is missing.

In addition to computing the fidelity metrics for the Train-Synthetic pair, we report the metrics for the Train-Test pair as an ideal baseline (BL). This is a valid idea, as we can consider the test data as highly authentic synthetic data. In addition, it can reveal an expected value for each metric. We use a 5-fold train-test split and report the mean and standard deviation of the metrics.

Utility Following the Train on Synthetic, Test on Real (TSTR) protocol [42], we train a LightGBM Classifier on the synthetic data and evaluate its performance on the test data. We report the area under the receiver operating characteristic curve (AUROC) as well as the area under the precision-recall

TABLE II

FIDELITY METRICS: (\uparrow) HIGHER IS BETTER, (\downarrow) LOWER IS BETTER. (BL) IS COMPUTED FOR THE TRAIN-TEST PAIR.

Dataset	Model	precision (\uparrow)	recall (\uparrow)	density (\uparrow)	coverage (\uparrow)	TCD (\downarrow)
P12	TimeGAN	0.215(0.024)	0.19(0.011)	0.178(0.024)	0.26(0.018)	0.101(0.004)
	DoppelGANger	0.268(0.036)	0.224(0.008)	0.15(0.049)	0.186(0.029)	0.116(0.009)
	RTSGAN	0.441(0.036)	0.428(0.018)	0.237(0.025)	0.381(0.025)	0.055(0.008)
	TimEHR	0.775(0.02)	0.651(0.05)	0.657(0.054)	0.782(0.024)	0.019(0.001)
	BL	0.855(0.015)	0.854(0.006)	0.94(0.051)	0.967(0.006)	0.01(0.001)
P19	TimeGAN	0.45(0.017)	0.43(0.026)	0.251(0.041)	0.392(0.014)	0.092(0.011)
	DoppelGANger	0.497(0.046)	0.481(0.014)	0.317(0.083)	0.356(0.072)	0.133(0.008)
	RTSGAN	0.633(0.048)	0.602(0.015)	0.407(0.053)	0.581(0.031)	0.031(0.019)
	TimEHR	0.791(0.026)	0.657(0.03)	0.814(0.104)	0.77(0.051)	0.021(0.002)
	BL	0.852(0.006)	0.849(0.003)	0.959(0.037)	0.964(0.005)	0.007(0.001)
MIMIC-III	TimeGAN	0.431(0.06)	0.409(0.042)	0.39(0.096)	0.326(0.036)	0.093(0.004)
	DoppelGANger	0.372(0.019)	0.31(0.044)	0.364(0.081)	0.39(0.069)	0.128(0.004)
	RTSGAN	0.687(0.041)	0.539(0.022)	0.488(0.066)	0.611(0.047)	0.052(0.007)
	TimEHR	0.762(0.027)	0.622(0.039)	0.759(0.111)	0.798(0.055)	0.035(0.001)
	BL	0.778(0.052)	0.719(0.079)	0.642(0.192)	0.839(0.074)	0.03(0.004)

curve (AUPRC) due to the high class imbalance of the datasets [51].

Privacy (1) *Membership inference attack (MIA)*: This metric measures the probability of data belonging to the training set [14]. We fit a K-Nearest Neighbors (kNN) classifier on the synthetic data and calculate the nearest distances of train and test splits to the fitted model. Any differences between the distributions of the nearest distances of train and test splits indicate that the synthetic data is not private. We report the Jensen-Shannon Divergence (JSD_{MIA}) between the two distributions, as well as the AUROC of a binary classifier that predicts whether a sample is from the training set or not ($AUROC_{MIA}$).

(2) *Nearest Neighbor Adversarial Accuracy Risk (NNAAR)*: This score measures the degree to which a generative model overfits the real training data, a factor that could raise privacy-related concerns [52]. It is the difference between two adversarial accuracies, $AA_{test}(AA(\mathcal{D}_{gen}, \mathcal{D}_{test}))$ and $AA_{train}(AA(\mathcal{D}_{gen}, \mathcal{D}_{train}))$ (see VIII).

Baselines

We use RTSGAN [36] as the main baseline because it is the only model with available code for generating irregularly sampled time series with missingness. Additionally, we adapt TimeGAN [34] and DoppelGANger [35] by treating missing pattern as additional features which increases the input dimensionality by a factor of two.

A. Overall Performance

Fidelity Table II shows the fidelity metrics for TimEHR, baseline models, as well as the test pair (BL). We observe that TimEHR outperforms baselines in terms of PRDC metrics and TCD. TimeGAN and DoppelGANger exhibit significantly poor performance which is consistent with the previous works [15, 14, 38] as they are not designed for irregularly sampled time series with missing values. Although the ideal value is typically 1 for PRDC and 0 for TCD, it is evident that this is

not the case even for the test split (BL) as authentic synthetic data. This ideal baseline can be considered as an empirical bound for the fidelity metrics in future works.

Figure 3 illustrates image-based visualization of three examples (Top) and the correlation matrix (Bottom) from real data, TimEHR, and RTSGAN for the P12 dataset. Not only can TimEHR generate more realistic time series data than RTSGAN (especially the missingness pattern), but it can better preserve the temporal correlation structure of the variables. Notably, the RTSGAN correlation matrix has many NaN values (in yellow), which implies that many variable pairs (especially those with very high missingness rates) have not been generated together. Although the input images to our model are invariant to the order of variables, our model is still capable of capturing correlations between variables that are far from each other in the input image. This is not surprising as the receptive field of the convolutional layers increases with the depth of the network, allowing the model to capture long-range dependencies [53].

Figure 4-top illustrates the t-SNE visualization of the real data, and the generated data from TimEHR and RTSGAN for P12, P19, and MIMIC-III. We can see that the generated samples by TimEHR demonstrate a notably better overlap with the real data, which is also reflected in the PRDC metrics in Table II. The histogram of the length of Stay (LOS) is also shown in Figure 4-bottom. Although TimEHR does not explicitly utilize any strategy for handling variable-length time series, it can generate realistic LOS distribution.

Utility Table III shows the utility metrics for TimEHR and baselines in the downstream task of binary classification. Here, we have also reported the "Train on Real Test on Real" (TRTR) case, where we train a classifier on the training data and evaluate it on the test data. We see that TimEHR outperforms baselines in terms of AUROC and AUPRC for all datasets. The performance gain is even higher in larger datasets (P19 and MIMIC-III), however, the AURPC is considerably lower than the TRTR case.

Privacy Table IV shows the privacy metrics for TimEHR

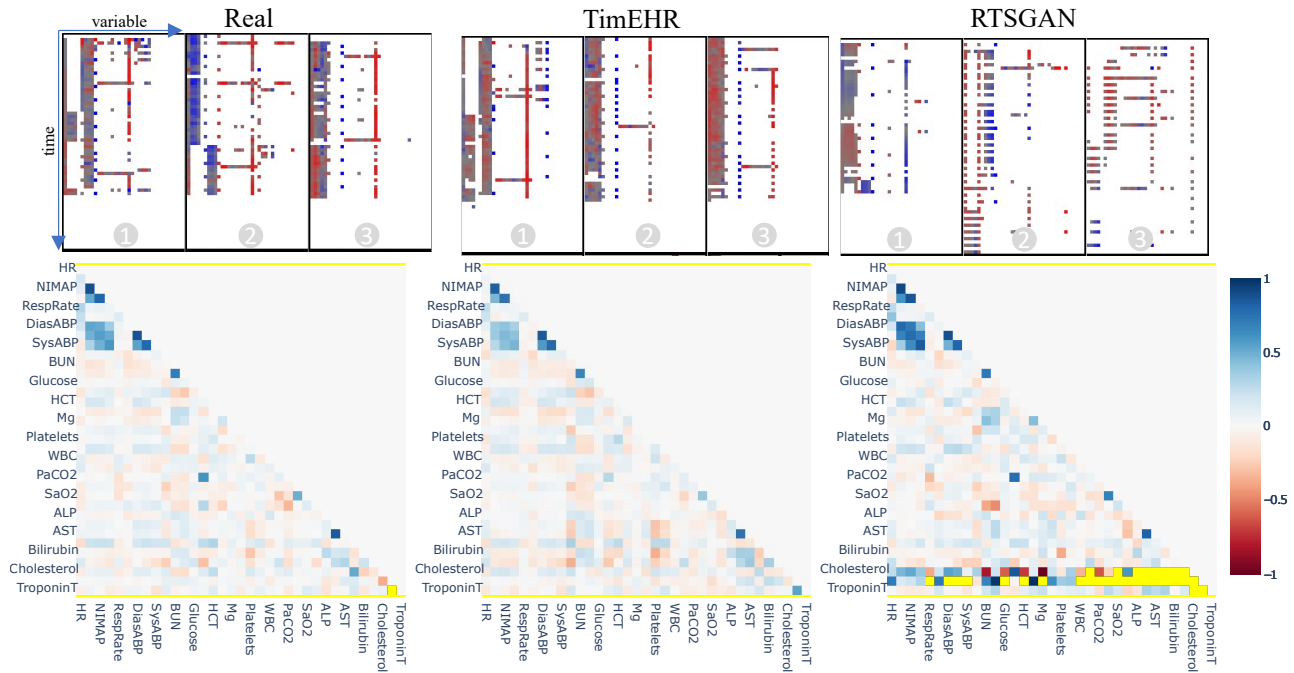


Fig. 3. Top: Image-based visualization of three examples in P12. Bottom: Temporal Correlation comparison in P12. The NaN values are shown in yellow.

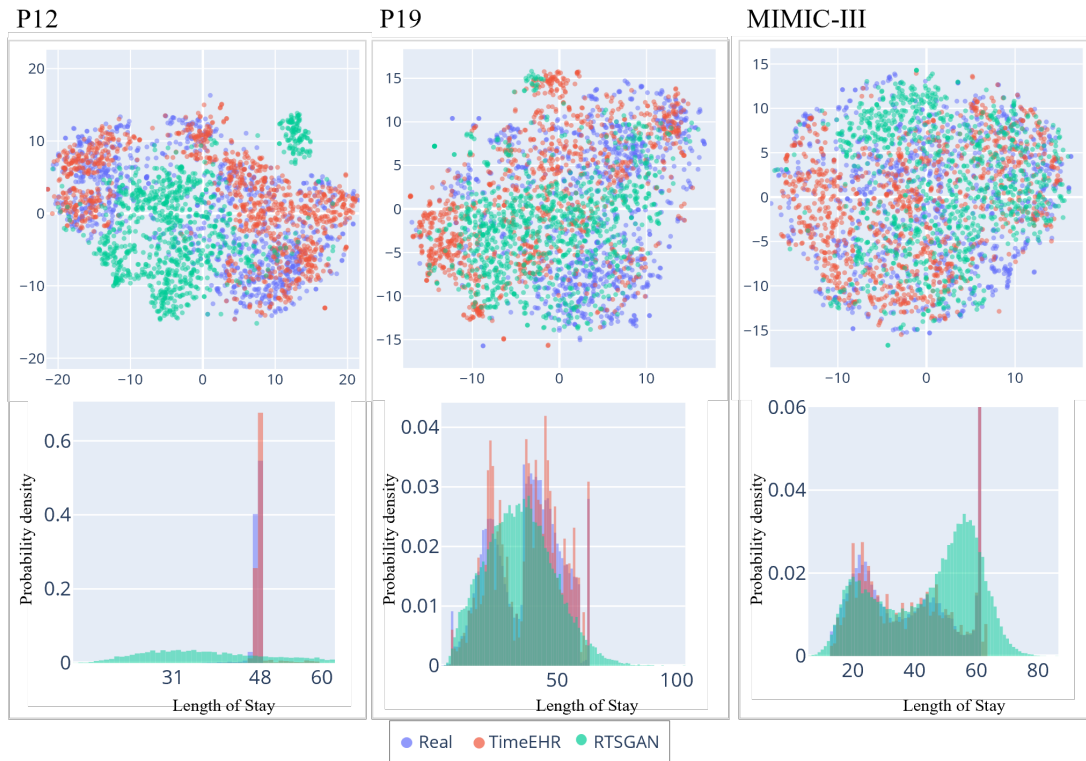


Fig. 4. t-SNE visualization (top) and length of stay (bottom) for P12, P19 and MIMIC-III datasets.

and baselines. In general, none of the metrics shows any privacy issues. The AUROC and JSD metrics for MIA are close to the baseline for all models and show that the synthetic data is private. The adversarial accuracy (AA) metrics (AA_{train} , AA_{test}) are significantly higher than the baseline

for all models. This is because these metrics are sensitive to mode collapse or mode generation, and any mismatch between the real and generated distributions can increase the adversarial accuracy (this can also be seen in PRDC metrics). Since the AA metrics do not show any privacy issues, we can conclude

TABLE III
UTILITY METRICS

Dataset	Model	AUROC	AUPRC
P12	TimeGAN	0.587(0.004)	0.204(0.033)
	DoppelGANger	0.537(0.008)	0.167(0.02)
	RTSGAN	0.745(0.015)	0.331(0.03)
	TimEHR	0.785(0.015)	0.424(0.035)
	<i>TRTR</i>	<i>0.832(0.008)</i>	<i>0.475(0.023)</i>
P19	TimeGAN	0.57(0.014)	0.094(0.028)
	DoppelGANger	0.611(0.038)	0.106(0.015)
	RTSGAN	0.677(0.037)	0.141(0.022)
	TimEHR	0.853(0.014)	0.452(0.017)
	<i>TRTR</i>	<i>0.908(0.011)</i>	<i>0.618(0.011)</i>
MIMIC-III	TimeGAN	0.637(0.016)	0.236(0.023)
	DoppelGANger	0.629(0.022)	0.227(0.031)
	RTSGAN	0.766(0.025)	0.427(0.036)
	TimEHR	0.82(0.01)	0.536(0.024)
	<i>TRTR</i>	<i>0.863(0.012)</i>	<i>0.621(0.028)</i>

that the low NNAE metrics for all models only indicate the asymmetry between AA_{train} and AA_{test} rather than the privacy risk.

B. Ablation Study

We can further investigate whether the use of each of the proposed modules improves the performance of the model. We conducted an ablation study by removing each module and comparing the results with the full model. The percent deviation of each metric from its baseline is shown in Figure 5. We observe that *TimEHR w/o Pix2Pix* (\square) and *TimEHR w/o L2* (\square) have the worst fidelity and utility scores while preserving privacy. On the other hand, *TimEHR w/o CWGAN* (\square) seems to have a perfect fidelity and utility score. However, low privacy metrics ($AUROC_{MIA}$ and AA_{train}) show that it is not a privacy-preserving model and it is memorizing the training data. This can be attributed to the deterministic nature of Pix2Pix module as it ignores the input noise. Additionally, we can argue that having fidelity scores that are better than the baseline is associated with privacy issues. Finally, *TimEHR* (\times) has the best performance in terms of fidelity and utility while preserving privacy. Based on these results, we can conclude that the CWGAN module improves the privacy of the model by generating the missingness pattern, and the Pix2Pix module with L2 reconstruction loss improves the fidelity and utility of the model by preserving the temporal correlation structure.

C. Simulated Data

We further investigate TimEHR on simulated sinusoidal time series data to understand the effect of different levels of missingness and number of variables. We generate 10,000 time series with $L = 64$ and $d \in \{16, 32, 64\}$, as well as a longer time series with $L = 128$ and $d = 128$. A 128×128 image seems to be a reasonable proxy for most of the EHR datasets as it can capture a time horizon of around 5 days (hourly data) and 128 variables. For each variable, $x_d(t) = A_d \sin(\frac{2\pi}{P_d}t + \phi_d) + \epsilon$ (full data generating process in VI). Additionally, the missing pattern follows a homogenous

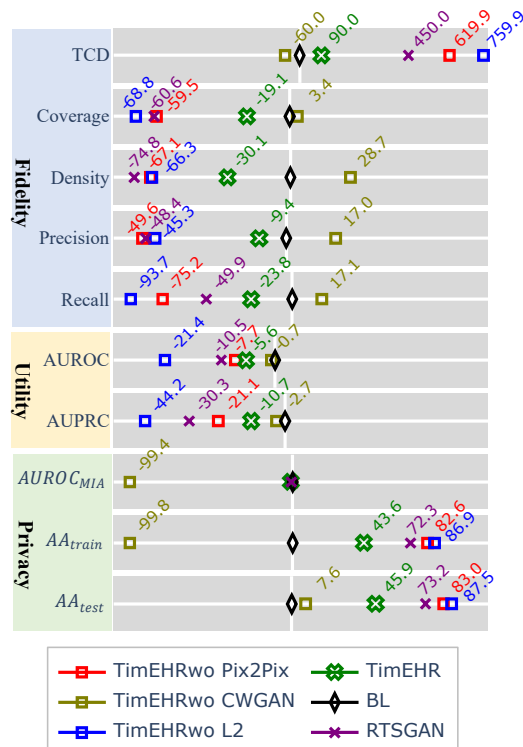


Fig. 5. Ablation study. Values are percent deviation from the baseline.

Poisson process with intensity function $\lambda^*(t) = \lambda$. We use $\lambda \in \{0.2, 0.5, 1, 2\}$ which corresponds to an overall missing rate of 81.85%, 60.66%, 36.78%, and 13.64% respectively. Figure 6 shows the TCD metric for different number of variables (d) and values of λ . The maximum TCD value is still less than 0.035, which shows that TimEHR can preserve the temporal correlation structure of the variables even for a large number of variables and a high missing rate.

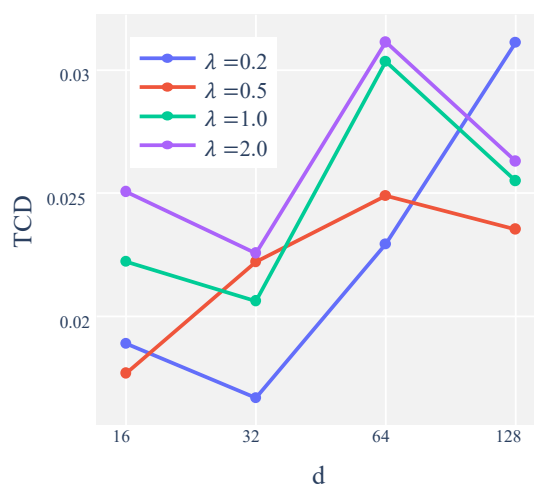


Fig. 6. TCD for simulated data.

V. CONCLUSION

We have proposed TimEHR to generate irregularly sampled time series with missingness from EHR datasets. It shows

TABLE IV

PRIVACY METRICS. NONE OF THE METRICS SHOW PRIVACY ISSUES. (BL) IS THE IDEAL VALUE FOR THE METRICS.

Dataset	Model	$AUROC_{MIA}$	JSD_{MIA}	AA_{train}	AA_{test}	$NNAA$
P12	TimeGAN	0.509(0.003)	0.001(0.001)	0.743(0.011)	0.767(0.020)	-0.024(0.014)
	DoppelGANger	0.498(0.001)	0.001(0.001)	0.604(0.010)	0.622(0.006)	-0.018(0.002)
	RTSGAN	0.496(0.008)	0.001(0.001)	0.882(0.008)	0.887(0.007)	-0.006(0.002)
	TimEHR	0.494(0.01)	0.001(0.0)	0.735(0.017)	0.747(0.015)	-0.012(0.016)
	<i>BL</i>	0.5	0	0.512(0.01)	0.512(0.01)	0
P19	TimeGAN	0.507(0.008)	0.001(0.0)	0.821(0.019)	0.835(0.007)	-0.014(0.003)
	DoppelGANger	0.491(0.004)	0.001(0.0)	0.742(0.032)	0.757(0.022)	-0.015(0.001)
	RTSGAN	0.5(0.009)	0.001(0.0)	0.767(0.01)	0.779(0.009)	-0.012(0.01)
	TimEHR	0.499(0.008)	0.001(0.0)	0.721(0.025)	0.736(0.029)	-0.015(0.012)
	<i>BL</i>	0.5	0	0.518(0.012)	0.518(0.012)	0
MIMIC-III	TimeGAN	0.488(0.027)	0.004(0.001)	0.722(0.018)	0.712(0.022)	0.010(0.002)
	DoppelGANger	0.483(0.016)	0.003(0.002)	0.76(0.032)	0.819(0.016)	-0.052(0.02)
	RTSGAN	0.491(0.025)	0.005(0.002)	0.777(0.021)	0.856(0.057)	-0.079(0.04)
	TimEHR	0.488(0.024)	0.004(0.002)	0.733(0.041)	0.837(0.061)	-0.104(0.062)
	<i>BL</i>	0.5	0	0.669(0.092)	0.669(0.092)	0

promising results in terms of fidelity, utility, and privacy metrics on three large EHR datasets.

Although our model exhibits acceptable privacy metrics, it is not a privacy-preserving model as we have not considered any privacy constraints in the training process. Thus, we leave it to future work to incorporate privacy-preserving frameworks such as differential privacy into the training process [54]. It is also interesting to explore other CNN-based architectures to see if they can improve the performance of our model in terms of model utility. In this work, we have experimented TimEHR on time series images of size 64×64 in EHR datasets and 128 in the simulated datasets. However, it would be interesting to see how the model performs on datasets with longer time series and more variables. Finally, although our PiX2Pix module relies on the missingness pattern to generate the value channel, it is still possible to generate the values using the CWGAN module. We leave it to future work to investigate the performance of our model on regularly sampled time series.

VI. APPENDIX A: DATASETS

In this work, we use three publicly available EHR datasets and a simulated dataset to evaluate TimEHR:

MIMIC-III. Medical Information Mart for Intensive Care III (MIMIC-III) [47] is a centralized database with an extensive set of electronic health record (EHR) data for individuals admitted to critical care units at Beth Israel Deaconess Medical Center from 2001 to 2012. The dataset encompasses details such as patient demographics, laboratory results, vital signs, procedures, caregiver notes, and outcomes. We used 48 time series and four static variables (age, height, weight, mortality) using the reproducibility pipeline from [55]. This led to 50,961 patients. The overall missingness rate is 79.40%. The maximum, average, and standard deviation of length of stay are 62, 45, and 17 hours, respectively.

Physionet Challenge 2012 (P12). This dataset [48] contains 35 time series and six static variables (age, gender, height, weight, ICU type, and mortality) for 12,000 ICU patients. The

overall missingness rate is 80.37%. The maximum, average, and standard deviation of length of stay are 48, 47, and 2 hours, respectively.

Physionet Challenge 2019 (P19). This dataset [49] consists of 32 time series and four static variables (age, gender, admission time, sepsis shock) for 40,333 patients. The overall missingness rate is 74.58%. The maximum, average, and standard deviation of length of stay are 64, 37, and 14 hours, respectively.

Simulated Data. The full algorithm for generating simulated data is shown in algorithm 2.

Algorithm 2: Generating Simulated Time Series

Input: Size N , number of variables d , Poisson rate λ , Time series length L , Period range $[T_{min}, T_{max}]$
 Data=np.zeros((N,L,d))
for $d = 1$ **to** D **do**
 $\phi_d = \text{uniform}(0, 2\pi)$
 $T_d = \text{rand}(T_{min}, T_{max})$
 for $i = 1$ **to** N **do**
 Generate Values:
 $t = \text{np.arange}(L)$
 $A = U(0.9, 1)$
 $b = U(-0.5, 0.5)$
 $\epsilon = \text{Normal}(0, 0.1, L)$
 $\epsilon_T = \text{rand}(1, 5)$
 $\epsilon_T = U(0, 2\pi/50)$
 $\text{Data}[i, :, d] = b + A \sin(2\pi t / (T_d + \epsilon_T) + \phi_d) + \epsilon$
 Set Missingness Pattern:
 $\text{poisson_counts} = \text{Poisson}(\lambda, L)$
 $\text{mask} = \text{np.where}(\text{poisson_counts} > 0, 1, 0)$
 $\text{Data}[i, \text{mask} == 0, d] = \text{NaN}$
 end for
end for

Data Preprocessing

We use a 5-fold train-test split. We use the first 64 (48) hours of each patient's ICU stay for P19 (P12 and MIMIC-III) and the time series are aggregated into 1-hour intervals. If we have multiple observations, we keep the last one and in the case of no observation we use NaN. Then, they are normalized to have zero mean and unit variance (NaN values are ignored), and the outlier values (more than 3 standard deviations from the mean) are clipped. Finally, we perform a min-max normalization to scale the values between -1 and 1. For static variables, we use one-hot encoding for categorical variables and standardization for continuous variables.

The next step is to reshape the time series into images. Each patient time series is a matrix of shape $[L_i, d]$ where L_i is the length of the time series and d is the number of variables. We pad this matrix with zeros to have a shape of $[64, 64]$. We form a similar mask matrix where 1 indicates a measured value and -1 indicates a missing value. We replace NaN values with 0 in the time series matrix. Finally, we concatenate the time series and mask matrices along the channel axis to form a 2-channel image of shape $[2, 64, 64]$.

VII. APPENDIX B: TRAINING DETAILS

The training algorithms for CWGAN-GP and Pix2Pix are shown in algorithm 3 and algorithm 4, respectively.

Algorithm 3: Training CWGAN-GP

Input: Training data \mathcal{D}_{train} , Batch size B
Initialize: G_1 and D_1 .
for $epoch = 1$ **to** n_{epochs} **do**
 sample minibatch of B noise samples $\{z_1, \dots, z_B\}$
 from P_z
 sample minibatch of B real data samples $\{I_1, \dots, I_B\}$
 from P_{data}
 Training D_1 :
 $\tilde{I}_i = G_1(z_i, s_i)$
 $\hat{I}_i = \epsilon I_i + (1 - \epsilon) \tilde{I}_i$
 $\mathcal{L}_{D_1} = -\frac{1}{B} \sum_{i=1}^B D_2(I_i) + \frac{1}{B} \sum_{i=1}^B D_2(\tilde{I}_i)$
 $\mathcal{L}_{D_1} = \mathcal{L}_{D_1} + \lambda_{GP} \mathbb{E}_{\hat{I} \sim P_{\hat{I}}} [(\|\nabla_{\hat{I}} D_1(\hat{I})\|_2 - 1)^2]$
 Update D_1 using $\nabla_{D_1} \mathcal{L}_{D_1}$
 Training G_1 :
 $\tilde{I}_i = G_1(z_i, s_i)$
 $\mathcal{L}_{G_1} = -\frac{1}{B} \sum_{i=1}^B D_1(\tilde{I}_i)$
 Update G_1 using $\nabla_{G_1} \mathcal{L}_{G_1}$
end for

Hyperparameters

We used Optuna library [56] for hyperparameter tuning. We explored learning rate in the range of $[1e-5, 1e-2]$ and batch size in the range of $[32, 256]$. The kernel size is 4 for all datasets. For CWGAN-GP, we used an Adam optimizer with a learning rate of $3e-4$ and a batch size of 128. We train the model for 150 epochs. We use a gradient penalty coefficient (λ_{GP}) of 10 for CWGAN-GP. For Pix2Pix, we used an Adam optimizer with a learning rate of $2e-3$ and a batch size of

Algorithm 4: Training Pix2Pix

Input: Training data \mathcal{D}_{train} , Batch size B
Initialize: G_2 and D_2 .
for $epoch = 1$ **to** n_{epochs} **do**
 sample minibatch of B real data missing pattern
 $\{I_{1,mask}, \dots, I_{B,mask}\}$ from \mathcal{D}_{train}
 Training D_2 :
 $\tilde{I}_{i,value} = G_2(I_{i,mask}, s_i)$
 $\mathcal{L}_{D_2} = -\frac{1}{B} \sum_{i=1}^B [\log D_2(I_{i,value}, s_i)] -$
 $\frac{1}{B} \sum_{i=1}^B [\log(1 - D_2(\tilde{I}_{i,value}, s_i))]$
 Update D_2 using $\nabla_{D_2} \mathcal{L}_{D_2}$
 Training G_2 :
 $\tilde{I}_{i,value} = G_2(I_{i,mask}, s_i)$
 $\mathcal{L}_{G_2} = -\frac{1}{B} \sum_{i=1}^B [\log D_2(\tilde{I}_{i,value}, s_i)]$
 $\mathcal{L}_{rec} = \frac{1}{B} \sum_{i=1}^B [\|I_{i,value} - \tilde{I}_{i,value}\|_2]$
 $\mathcal{L}_{G_2} = \mathcal{L}_{G_2} + \lambda \mathcal{L}_{rec}$
 Update G_2 using $\nabla_{G_2} \mathcal{L}_{G_2}$
end for

32. We train the model for 150 epochs. The reconstruction loss coefficient (λ_{L_2}) is 200 for P19 dataset and 100 for other datasets. We use a linear decay learning rate scheduler for the first 50 epochs. We use a gradient penalty coefficient (λ_{GP}) of 10 for CWGAN-GP.

VIII. APPENDIX C: METRICS

In this section, we describe the metrics used in the paper.

Precision, Recall, Density, Coverage

Let X and Y be two sets of samples from the real and generated data distributions. [57] first proposed to construct the *manifold* for $P(X)$ and $Q(Y)$ separately. This object is nearly identical to the probabilistic density function except that it does not sum to 1. Precision then measures the expected likelihood of fake samples against the real manifold and recall measures the expected likelihood of real samples against the fake manifold. Density improves upon the precision metric by fixing the overestimation of the manifold around real outliers. Coverage improves upon the recall metric to better quantify this by building the nearest neighbor manifolds around the real samples, instead of the fake samples, as they have fewer outliers [50].

$$\text{Precision} := \frac{1}{M} \sum_{j=1}^M \mathbb{I}_{y \in \text{manifold}(X_1, \dots, X_N)},$$

$$\text{Recall} := \frac{1}{N} \sum_{i=1}^N \mathbb{I}_{x \in \text{manifold}(Y_1, \dots, Y_M)},$$

$$\text{Density} := \frac{1}{kM} \sum_{j=1}^M \sum_{i=1}^N \mathbb{I}_{y_j \in B(X_i, NND_k(X_i))},$$

$$\text{Coverage} := \frac{1}{N} \sum_i i = 1^N \mathbb{I}_{\exists j \text{ s.t. } y_j \in B(X_i, NND_k(X_i))},$$

where N and M are the number of real and generated samples. $\mathbb{I}_{(\cdot)}$ is the indicator function, and manifolds are defined as:

$$\text{manifold}(X_1, \dots, X_N) := \bigcup_{i=1}^N B(X_i, \text{NND}(X_i)),$$

where $B(x, r)$ is the sphere in \mathbb{R}^D around x with radius r . $\text{NND}_k(X_i)$ denotes the distance from X_i to the k^{th} nearest neighbour among $\{X_j\}$ excluding itself.

Nearest Neighbor Adversarial Accuracy (NNAA)

This metric was proposed in [52] to measure the degree to which a generative model overfits the real training data. The Adversarial Accuracy (AA) between two sets of samples X and Y is defined as:

$$AA(X, Y) = \frac{1}{2} \left(\frac{1}{N} \sum_{i=1}^N \mathbb{I}\{d_{XY}(i) > d_{XX}(i)\} + \frac{1}{N} \sum_{i=1}^N \mathbb{I}\{d_{YX}(i) > d_{YY}(i)\} \right), \quad (1)$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, and the distances are defined as:

$$d_{XY}(i) = \min_j \|x_X^{(i)} - x_Y^{(j)}\|, \quad d_{YX}(i) = \min_j \|x_Y^{(i)} - x_X^{(j)}\|, \\ d_{XX}(i) = \min_{j, j \neq i} \|x_X^{(i)} - x_X^{(j)}\|, \quad d_{YY}(i) = \min_{j, j \neq i} \|x_Y^{(i)} - x_Y^{(j)}\|,$$

In principle, we expect that the nearest neighbors of a sample in either set X or Y would be from the opposite set in 50 percent of cases. Thus, the ideal value of $AA(X, Y)$ is 0.5. However, if the model overfits the training data (i.e. the model memorizes the training data), the value of $AA(X, Y)$ will be lower than 0.5 because the nearest neighbors of a sample in X or Y would be from the opposite set in most cases. Thus, we can interpret $AA(X, Y) < 0.5$ as a sign of overfitting or privacy issues. Considering three sets of samples X_{train} , X_{test} , and $X_{\text{synthetic}}$, we can define the the following metrics:

$$AA_{\text{train}} = AA(X_{\text{train}}, X_{\text{synthetic}}), \\ AA_{\text{test}} = AA(X_{\text{test}}, X_{\text{synthetic}}), \\ AA_{\text{baseline}} = AA(X_{\text{train}}, X_{\text{test}}) \\ NNAA = AA_{\text{test}} - AA_{\text{train}},$$

REFERENCES

- [1] Maryam Tayefi et al. "Challenges and Opportunities beyond Structured Data in Analysis of Electronic Health Records". In: *WIREs Computational Statistics* 13.6 (2021), e1549. ISSN: 1939-0068. DOI: 10.1002/wics.1549. (Visited on 02/02/2024).
- [2] Ismail Keshta and Ammar Odeh. "Security and Privacy of Electronic Health Records: Concerns and Challenges". In: *Egyptian Informatics Journal* 22.2 (July 2021), pp. 177–183. ISSN: 1110-8665. DOI: 10.1016/j.eij.2020.07.003. (Visited on 02/02/2024).
- [3] Clete A. Kushida et al. "Strategies for De-Identification and Anonymization of Electronic Health Record Data for Use in Multicenter Research Studies". In: *Medical care* 50.Suppl (July 2012), S82–101. ISSN: 0025-7079. DOI: 10.1097/MLR.0b013e3182585355. (Visited on 01/29/2024).
- [4] Mana Azarm-Daigle, Craig Kuziemsky, and Liam Peyton. "A Review of Cross Organizational Healthcare Data Sharing". In: *Procedia Computer Science*. The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015)/ The 5th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2015)/ Affiliated Workshops 63 (Jan. 2015), pp. 425–432. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.08.363. (Visited on 01/29/2024).
- [5] Arno Appenzeller et al. "Privacy and Utility of Private Synthetic Data for Medical Data Analyses". In: *Applied Sciences* 12.23 (Jan. 2022), p. 12320. ISSN: 2076-3417. DOI: 10.3390/app122312320. (Visited on 01/29/2024).
- [6] Mauro Giuffrè and Dennis L. Shung. "Harnessing the Power of Synthetic Data in Healthcare: Innovation, Application, and Privacy". In: *npj Digital Medicine* 6.1 (Oct. 2023), pp. 1–8. ISSN: 2398-6352. DOI: 10.1038/s41746-023-00927-3. (Visited on 01/29/2024).
- [7] Ian J. Goodfellow et al. *Generative Adversarial Networks*. June 2014. DOI: 10.48550/arXiv.1406.2661. arXiv: 1406.2661 [cs, stat]. (Visited on 01/29/2024).
- [8] Nripendra Kumar Singh and Khalid Raza. *Medical Image Generation Using Generative Adversarial Networks*. May 2020. DOI: 10.48550/arXiv.2005.10687. arXiv: 2005.10687 [cs, eess]. (Visited on 01/29/2024).
- [9] Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalonde. "GANs for Medical Image Synthesis: An Empirical Study". In: *Journal of Imaging* 9.3 (Mar. 2023), p. 69. ISSN: 2313-433X. DOI: 10.3390/jimaging9030069. (Visited on 01/29/2024).
- [10] Scott H. Lee. "Natural Language Generation for Electronic Health Records". In: *npj Digital Medicine* 1.1 (Nov. 2018), pp. 1–7. ISSN: 2398-6352. DOI: 10.1038/s41746-018-0070-0. (Visited on 01/29/2024).
- [11] Kexin Huang, Jaan Altsaar, and Rajesh Ranganath. *ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission*. Nov. 2020. DOI: 10.48550/arXiv.1904.05342. arXiv: 1904.05342 [cs]. (Visited on 01/29/2024).
- [12] Mrinal Kanti Baowaly et al. "Synthesizing Electronic Health Records Using Improved Generative Adversarial Networks". In: *Journal of the American Medical Informatics Association* 26.3 (Mar. 2019), pp. 228–241. ISSN: 1067-5027. 1527-974X. DOI: 10.1093/jamia/ocy142. (Visited on 10/19/2023).
- [13] Amirsina Torfi and Edward A. Fox. *CorGAN: Correlation-Capturing Convolutional Generative Adversarial Networks for Generating Synthetic Healthcare Records*. Mar. 2020. DOI: 10.48550/arXiv.2001.09346. arXiv: 2001.09346 [cs, stat]. (Visited on 01/29/2024).
- [14] Jinsung Yoon et al. "EHR-Safe: Generating High-Fidelity and Privacy-Preserving Synthetic Electronic Health Records". In: *npj Digital Medicine* 6.1 (Aug. 2023), pp. 1–11. ISSN: 2398-6352. DOI: 10.1038/s41746-023-00888-7. (Visited on 11/21/2023).
- [15] Muhang Tian et al. *Fast and Reliable Generation of EHR Time Series via Diffusion Models*. Oct. 2023. DOI: 10.48550/arXiv.2310.15290. arXiv: 2310.15290 [cs]. (Visited on 11/22/2023).
- [16] Amelia L. M. Tan et al. "Informative Missingness: What Can We Learn from Patterns in Missing Laboratory Data in the Electronic Health Record?" In: *Journal of Biomedical Informatics* 139 (Mar. 2023), p. 104306. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2023.104306. (Visited on 02/02/2024).

- [17] Emily Getzen et al. "Mining for Equitable Health: Assessing the Impact of Missing Data in Electronic Health Records". In: *Journal of Biomedical Informatics* 139 (Mar. 2023), p. 104269. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2022.104269. (Visited on 02/02/2024).
- [18] Edward Choi et al. "Doctor AI: Predicting Clinical Events via Recurrent Neural Networks". In: *Proceedings of the 1st Machine Learning for Healthcare Conference*. PMLR, Dec. 2016, pp. 301–318. (Visited on 01/03/2022).
- [19] Edward Choi et al. "RETAIN: An Interpretable Predictive Model for Healthcare Using Reverse Time Attention Mechanism". In: *arXiv:1608.05745 [cs]* (Feb. 2017). arXiv: 1608.05745 [cs]. (Visited on 01/03/2022).
- [20] Max Horn et al. "Set Functions for Time Series". In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 4353–4363. (Visited on 09/26/2022).
- [21] Xiang Zhang et al. *Graph-Guided Network for Irregularly Sampled Multivariate Time Series*. Mar. 2022. arXiv: 2110.05357 [cs]. (Visited on 09/26/2022).
- [22] Giorgia Ramponi et al. "T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling". In: *arXiv.org* (Nov. 2018).
- [23] Steven Cheng-Xian Li and Benjamin Marlin. "Learning from Irregularly-Sampled Time Series: A Missing Data Perspective". In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 5937–5946. (Visited on 06/18/2022).
- [24] Chrysoula Kosma, Giannis Nikolentzos, and Michalis Vazirgiannis. *Time-Parameterized Convolutional Neural Networks for Irregularly Sampled Time Series*. Aug. 2023. DOI: 10.48550/arXiv.2308.03210. arXiv: 2308.03210 [cs]. (Visited on 01/09/2024).
- [25] Srijan Sood et al. "Visual Time Series Forecasting: An Image-driven Approach". In: *Proceedings of the Second ACM International Conference on AI in Finance*. Nov. 2021, pp. 1–9. DOI: 10.1145/3490354.3494387. arXiv: 2011.09052 [cs, econ]. (Visited on 01/09/2024).
- [26] Artemios-Anargyros Semengelou, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Image-Based Time Series Forecasting: A Deep Convolutional Neural Network Approach". In: *Neural Networks* 157 (Jan. 2023), pp. 39–53. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.10.006. (Visited on 01/09/2024).
- [27] Zhiguang Wang and Tim Oates. *Imaging Time-Series to Improve Classification and Imputation*. May 2015. DOI: 10.48550/arXiv.1506.00327. arXiv: 1506.00327 [cs, stat]. (Visited on 01/03/2024).
- [28] Nima Hatami, Yann Gavet, and Johan Debayle. *Classification of Time-Series Images Using Deep Convolutional Neural Networks*. Oct. 2017. DOI: 10.48550/arXiv.1710.00886. arXiv: 1710.00886 [cs]. (Visited on 01/03/2024).
- [29] Omolbanin Yazdanbakhsh and Scott Dick. *Multivariate Time Series Classification Using Dilated Convolutional Neural Network*. May 2019. DOI: 10.48550/arXiv.1905.01697. arXiv: 1905.01697 [cs, stat]. (Visited on 01/09/2024).
- [30] Tailai Wen and Roy Keyes. *Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning*. May 2019. DOI: 10.48550/arXiv.1905.13628. arXiv: 1905.13628 [cs, stat]. (Visited on 01/09/2024).
- [31] Zekun Li, Shiyang Li, and Xifeng Yan. *Time Series as Images: Vision Transformer for Irregularly Sampled Time Series*. Oct. 2023. DOI: 10.48550/arXiv.2303.12799. arXiv: 2303.12799 [cs]. (Visited on 01/03/2024).
- [32] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. Dec. 2017. DOI: 10.48550/arXiv.1704.00028. arXiv: 1704.00028 [cs, stat]. (Visited on 01/08/2024).
- [33] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632. (Visited on 10/20/2023).
- [34] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. "Time-Series Generative Adversarial Networks". In: (Sept. 2019).
- [35] Zinan Lin et al. "Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions". In: *Proceedings of the ACM Internet Measurement Conference*. Virtual Event USA: ACM, Oct. 2020, pp. 464–483. ISBN: 978-1-4503-8138-3. DOI: 10.1145/3419394.3423643. (Visited on 01/19/2023).
- [36] Hengzhi Pei et al. "Towards Generating Real-World Time Series Data". In: (Dec. 2021). DOI: 10.1109/ICDM51629.2021.00058.
- [37] Jinsung Jeon et al. *GT-GAN: General Purpose Time Series Synthesis with Generative Adversarial Networks*. Oct. 2022. arXiv: 2210.02040 [cs]. (Visited on 11/28/2023).
- [38] Yangming Li. *TS-Diffusion: Generating Highly Complex Time Series with Diffusion Models*. Nov. 2023. DOI: 10.48550/arXiv.2311.03303. arXiv: 2311.03303 [cs]. (Visited on 12/15/2023).
- [39] Eoin Brophy, Zhengwei Wang, and Tomas E. Ward. *Quick and Easy Time Series Generation with Established Image-based GANs*. Oct. 2019. DOI: 10.48550/arXiv.1902.05624. arXiv: 1902.05624 [cs, stat]. (Visited on 08/07/2024).
- [40] Kaleb E. Smith and Anthony O. Smith. *Conditional GAN for Timeseries Generation*. June 2020. DOI: 10.48550/arXiv.2006.16477. arXiv: 2006.16477 [cs, stat]. (Visited on 08/07/2024).
- [41] Kaleb E. Smith and Anthony O. Smith. *A Spectral Enabled GAN for Time Series Data Generation*. Mar. 2021. DOI: 10.48550/arXiv.2103.01904. arXiv: 2103.01904 [cs, stat]. (Visited on 08/07/2024).
- [42] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. "Real-Valued (Medical) Time Series Generation with Recurrent Conditional GANs". In: *arXiv: Machine Learning* (June 2017).
- [43] Michael Mathieu, Camille Couprie, and Yann LeCun. *Deep Multi-Scale Video Prediction beyond Mean Square Error*. Feb. 2016. DOI: 10.48550/arXiv.1511.05440. arXiv: 1511.05440 [cs, stat]. (Visited on 02/02/2024).
- [44] Deepak Pathak et al. *Context Encoders: Feature Learning by Inpainting*. Nov. 2016. DOI: 10.48550/arXiv.1604.07379. arXiv: 1604.07379 [cs]. (Visited on 01/08/2024).
- [45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. May 2015. DOI: 10.48550/arXiv.1505.04597. arXiv: 1505.04597 [cs]. (Visited on 01/08/2024).
- [46] Lei Xu et al. *Modeling Tabular Data Using Conditional GAN*. Oct. 2019. DOI: 10.48550/arXiv.1907.00503. arXiv: 1907.00503 [cs, stat]. (Visited on 10/20/2023).
- [47] Alistair E. W. Johnson et al. "MIMIC-III, a Freely Accessible Critical Care Database". In: *Scientific Data* 3.1 (May 2016), p. 160035. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.35. (Visited on 01/05/2022).
- [48] Ikaro Silva et al. "Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012". In: ().
- [49] Matthew A. Reyna et al. "Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019". In: *Critical Care Medicine* 48.2 (Feb. 2020), pp. 210–217. ISSN: 0090-3493. DOI: 10.1097/CCM.0000000000004145. (Visited on 08/06/2021).

- [50] Muhammad Ferjad Naeem et al. *Reliable Fidelity and Diversity Metrics for Generative Models*. June 2020. DOI: 10 . 48550 / arXiv . 2002 . 09797. arXiv: 2002 . 09797 [cs, stat]. (Visited on 01/08/2024).
- [51] Matthew B. A. McDermott et al. *A Closer Look at AUROC and AUPRC under Class Imbalance*. Jan. 2024. DOI: 10 . 48550 / arXiv . 2401 . 06091. arXiv: 2401 . 06091 [cs, stat]. (Visited on 02/02/2024).
- [52] Andrew Yale et al. "Generation and Evaluation of Privacy Preserving Synthetic Health Data". In: *Neurocomputing* 416 (Nov. 2020), pp. 244–255. ISSN: 0925-2312. DOI: 10 . 1016 / j . neucom . 2019 . 12 . 136. (Visited on 01/09/2024).
- [53] Wenjie Luo et al. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. (Visited on 03/27/2024).
- [54] Liyang Xie et al. *Differentially Private Generative Adversarial Network*. Feb. 2018. DOI: 10 . 48550 / arXiv . 1802 . 06739. arXiv: 1802 . 06739 [cs, stat]. (Visited on 01/31/2024).
- [55] Alistair E. W. Johnson, Tom J. Pollard, and Roger G. Mark. "Reproducibility in Critical Care: A Mortality Prediction Case Study". In: *Proceedings of the 2nd Machine Learning for Healthcare Conference*. PMLR, Nov. 2017, pp. 361–376. (Visited on 01/28/2024).
- [56] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. July 2019. DOI: 10 . 48550 / arXiv . 1907 . 10902. arXiv: 1907 . 10902 [cs, stat]. (Visited on 01/26/2024).
- [57] Tuomas Kynkäänniemi et al. *Improved Precision and Recall Metric for Assessing Generative Models*. Oct. 2019. DOI: 10 . 48550 / arXiv . 1904 . 06991. arXiv: 1904 . 06991 [cs, stat]. (Visited on 01/25/2024).