Categorical EHR Imputation with Generative Adversarial Nets

1st Yinchong Yang
Siemens AG
Munich, Germany
yinchong.yang@siemens.com

2nd Zhiliang Wu
Siemens AG
Ludwig-Maximilians University of Munich
Munich, Germany
zhiliang.wu@siemens.com

2nd Volker Tresp
Siemens AG
Ludwig-Maximilians University of Munich
Munich, Germany
volker.tresp@siemens.com

3rd Peter A. Fasching
University Clinics Erlangen
Department of Gynecology and Obstetrics
Erlangen Germany
peter.fasching@uk-erlangen.de

***Abstract*—Electronic Health Records often suffer from missing data, which poses a major problem in clinical practice and clinical studies. A novel approach for dealing with missing data are Generative Adversarial Nets (GANs), which have been generating huge research interest in image generation and transformation. Recently, researchers have attempted to apply GANs to missing data generation and imputation for EHR data: a major challenge here is the categorical nature of the data. State-of-the-art solutions to the GAN-based generation of categorical data involve either reinforcement learning, or learning a bidirectional mapping between the categorical and the a real latent feature space, so that the GANs only need to generate real-valued features. However, these methods are designed to generate complete feature vectors instead of imputing only the subsets of missing features. In this paper we propose a simple and yet effective approach that is based on previous work on GANs for data imputation. We first motivate our solution by discussing the reason why adversarial training often fails in case of categorical features. Then we derive a novel way to re-code the categorical features to stabilize the adversarial training. Based on experiments on two real-world EHR data with multiple settings, we show that our imputation approach largely improves the prediction accuracy, compared to more traditional data imputation approaches.**

***Index Terms*—Data Imputation, Multiple Imputation, Generative Adversarial Nets**

I. Introduction

The increasing importance of data quality in healthcare: Electronic Health Records (EHR) present a rich data source and are, e.g., used for intra- and inter-departmental information exchange, for documentation purposes, and, most recently, as the basis for many analytic studies. Typically, data involving critical clinical decision paths are of good quality, but less critical data are often incomplete, e.g., due the huge workload of clinical personnel; this poses a significant problem for the secondary use of EHR data. In particular the value of a clinical study greatly depends on data completeness and correctness. Although the prime solution would be to enhance the EHR quality by improving the EHR system design and the data collection process, the missing data problem is not likely to completely disappear. [1] provides an overview on missing data approaches in statistics and [2] presents solutions to the neural network setting. When using nonlinear models, data imputation is often used [3], [4], which is also the approach pursued in this paper. Data imputation is often based on parametric or nonparametric probability density estimation. In this paper we investigate a recently developed GAN architecture. It imputes data without calculating a probability density first, and might become an important method of choice in the future.

Multiple instead of single imputation: More specifically, we discuss a novel realization of the well-known multiple imputation approach [1], [5], [6]. By embedding certain randomness into the imputation method and performing imputation multiple times, one can achieve more flexibility and reliability than with single imputation. This allows for —in contrast to an averaged point estimate of each missing value— estimating the statistical reliability of the imputation methods [7]. Multiple Imputation by Chained Equations (MICE) [8] fits one regression model for each feature that contains missing values, conditioned on all complete features. This method can model the dependency between features but the number of necessary regression models increases quadratically with the number of features. One could also simply assume a multivariate Gaussian distribution for the missing features and draw multiple random samples as imputation. The covariance matrix represents the dependency between features but the Gaussian distribution cannot handle categorical features, which are often present in EHR data. In this paper we investigate a novel approach that takes into account the categorical nature of the features while modeling inter-feature dependency in an efficient way.

GAN as multiple imputation: In recent years, a new class of neural networks, called Generative Adversarial Nets (GANs), have been developed and have generated huge interest in the research community. The original paper [9] proposes to train a network that can learn the underlying distribution of the data, allowing for generating unlimited amount of

data instances. When applied to images, the generated images often appear quite real. Since their initial introduction, a large variety of exciting improvements and modifications of the GAN framework have been proposed to solve different and yet related tasks, such as generating labeled data [10], image translation [11], deriving super-resolution [12] and image augmentation [13]. [14] proposes a new variant, the Generative Adversarial Imputation Nets (GAINs), to perform data imputation and shows promising results on multiple benchmark datasets. This method presents in fact a novel realization of the multiple imputation concept, and it is also related to the MICE algorithm. Instead of trying all possible orders to build the regression chain, it exploits the expressiveness of deep neural networks to model all features with missing values simultaneously. However, this approach cannot immediately be applied to EHR data, as we discuss now.

Challenge in EHR data for GAN: Most of the GAN models have been designed for image data, where the features, i.e., the pixel values, are real numbers. This enables error back-propagation within the GAN framework. In EHR data, however, a large proportion of patient features are categorical. In order to generate categorical data, even when binary coded, requires operations that are not differentiable, meaning that standard adversarial training is not possible. Due to the same reason, GANs have not seen many successful applications in NLP data [15]. A few approaches to generate discrete data with GANs have recently been proposed; most promising are approaches involving reinforcement learning [16], more specifically policy gradients [17]. Another proposed solution is to learn a mapping function from the discrete space of words to a latent real space as well as a reverse mapping [18]. [19] applies this idea to handle categorical features in EHR data and develops auto-encoders to function as the mapping. In such cases, the GAN model only needs to generate real valued vectors that represent the originally discrete data instances, allowing for the gradient propagation from discriminator and generator. In our related works section, we will review this approach in more detail. It is to note that the mapping functions between discrete and real spaces serve as pre- and post-processing steps, and are crucial for the quality of the translation between the discrete and real spaces. Such mapping functions are often trained in an Auto-encoder fashion and thus rely on the completeness of the input features. In a data imputation setting, however, the input features are by definition incomplete and the learned mappings must learn to map incomplete data. That is to say, this proposed approach is only applicable to generating complete feature vector instead of subsets of features.

Our contributions: The Generative Adversarial Imputation Nets framework [14] has been proposed to apply adversarial training to impute missing data of real values. In this work, we adjust this framework so that it can also perform data imputation for categorical features. We hypothesize that the reason that adversarial training often fails with softmax activation in the generator is that, while the true data features contain exclusively 0s and 1s, the softmax function can only produce a probability value between 0 and 1. On one hand, within a couple of epochs, the discriminator with sufficient expressiveness can learn to discriminate the generated values from real data exploiting this fact. On the other hand, it typically takes more epochs of training before the generator can produce real values close to 0 and 1. This phenomenon, i.e., that the discriminator always makes correct decisions and the generator always receives negative feedback from the very beginning, results in the divergence of the adversarial training [20]. In other words, the generator fails to learn anything useful to improve itself.

One of our major contributions is to propose a small but very effective modification to the data processing step. We perform a fuzzy binary coding of categorical features, i.e., we encode the binary values using real numbers between 0 and 1, while retaining the category information. In this way we guarantee that from the very beginning of the adversarial training, values produced by the generator already resemble the real values in their domain. To this end, the discriminator can not "cheat" and exploit the simple fact that real data are all binary while generated data are all real. The discriminator can only focus on the true and informative characteristics of the real and generated data, such as the dependency between features. Thus, the generator can receive more useful gradient updates from the discriminator, which improves the data generation process.

The rest of this paper is organized as follows. In section II we provide an overview of related works in three research fields of GANs: generation of categorical data, application GANs for EHR data and for data imputation. After a brief introduction to the GAN framework in section III, we elaborate the methods we propose in detail, including the fuzzy binary coding and the GAN for categorical data generation in section IV. In section V we present our experimental results on two EHR datasets and show that imputation based on the GAN framework with fuzzy binary coding can be quite effective in dealing with missing categorical data in EHRs.

II. Related Works

GANs that generate categorical features: There are currently three different approaches for generating categorical features with GANs. The first approach modifies the output activation function in the generator so that the gradient can flow from the discriminator to the generator while the latter generates pseudo-discrete features. Examples are so-called Gumbel Softmax [21], [22] or a soft argmax function [23]. The second approach modifies the training objectives. [16], [17] apply REINFORCE [24] algorithm for adversarial training. The third approach, including [18], [19], learns a mapping from the raw discrete feature space to a latent real space, as well as the reverse mapping. These mapping functions are, e.g., realized as an auto-encoder. With the first mapping one transforms all training data that are originally categorical into real representations. Then, the GANs framework only has to operate in this real space, learning to generate real feature vectors. As a post processing step, the generated vectors are

transformed back into the discrete space using the second mapping function.

GANs in EHR data analysis: GANs have already found various interesting applications in healthcare. [19] and [25] aim at generating pseudo-synthetic EHR data for the purpose of de-identification. The former focuses on the challenge of generating categorical features by applying an auto-encoder that can map between the discrete feature space and a real latent space. It is pointed out that applying differentiable Gumbel softmax or soft argmax functions does not completely solve the categorical problem, because patient features could be multinomial (i.e., multiclass) as well as multiple Bernoulli distributed (i.e., multi-label). The latter paper develops GANs that are based on Recurrent Neural Networks (RNN) to generate high dimensional time series EHR data.

Missing Data Imputation using Generative Adversarial Nets: [14] adjusted the GAN framework for the specific task of data imputation. It can be interpreted as a special case of conditional GAN, in the sense that both discriminator and generator take as input a mask vector indicating the missingness of feature values. It is shown that this novel training framework can efficiently impute real-valued features, especially in case where the missing rate is relatively high. Our method is largely inspired by this work, but we focus on the specific techniques to perform adversarial imputation of categorical features.

III. Preliminary: The Generative Adversarial Nets Framework

In its simplest case, a GAN framework [9] consists of two neural networks. The first one is often referred to as the *generator*, which consumes as input some random seeds $\boldsymbol{r}$ and generate data instances $\boldsymbol{g}$ that are supposed to resemble real data $\boldsymbol{x}$. The generator can be seen as a function of

$$\boldsymbol{g} = G(\boldsymbol{r}|\Theta_G). \tag{1}$$

Each generated sample is provided to the second neural network, the *discriminator*, i.e., $D(\boldsymbol{g}|\Theta_D)$. The discriminator also consumes as input the real data samples as $D(\boldsymbol{x}|\Theta_D)$. The training of the generator and the discriminator is adversarial, in that, while the discriminator is trained to correctly classify a sample to be either real or generated, the generator learns to fool the discriminator so that it classify generated samples to be real. More specifically, in term of the log-loss function $\mathcal{H}(a,b) = b \cdot \log(a) + (1-b) \cdot \log(1-a)$, we can write the discriminator loss and the generator loss as

$$\begin{aligned} loss_D = &- \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_{\text{real}}} \mathcal{H}(D(\boldsymbol{x}|\Theta_D), 1) \\ &- \mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}} \mathcal{H}(D(\boldsymbol{g}|\Theta_D), 0) \\ = &- \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_{\text{real}}} \log(D(\boldsymbol{x}|\Theta_D)) \\ &- \mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}} \log(1 - D(\boldsymbol{g}|\Theta_D)) \\ loss_G = &- \mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}} \mathcal{H}(D(\boldsymbol{g}|\Theta_D), 1) \\ = &- \mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}} \log(D(\boldsymbol{g}|\Theta_D)) \end{aligned} \tag{2}$$

respectively. With sufficient training, D will not be able to differentiate between real and generated samples by assigning neutral values to both cases. G will learn to map random seeds from an arbitrary distribution $\mathcal{P}_{\text{seed}}$ to the underlying distribution of the real data $\mathcal{P}_{\text{real}}$. Denoted as $\hat{\mathcal{P}}_{\text{real}}$, this estimate of the real data distribution allows for unlimited sampling.

IV. Method

In this section we give a detailed introduction to our method, which consists of two major components: the fuzzy binary coding and a modification of the Generative Adversarial Imputation Nets [14] for categorical feature generation.

A. Fuzzy binary coding

It is important to distinguish between *multinomial* and *multi-Bernoulli* distributed categorical features. In the former case, the random variable is realized by taking only one single category, i.e., the categories are mutually exclusive. For instance, the estrogen-receptor status of a patient could only be either positive, negative or unknown. In machine learning, especially if such features appear as targets, they are often referred to as *multiclass* features and modelled with the softmax function. In the *multi-bernoulli* case, a categorical feature can realize more than one categories, such as the location of metastasis, which could be multiple organs at the same time, or multiple (serious) adverse events (AE/SAE) could be triggered by certain treatment. For such a feature with non-mutual exclusive categories, one often use the term *multilabel*. For a concise terminology, we adopt the convention from machine learning and refer to these two cases as multiclass and multilabel features for the rest of the paper.

Assume that we observe p categorical features on one data instance and the j-th feature is a multiclass one, denoted as

$$\xi_j \in \Omega_j \text{ where } |\Omega_j| = q_j \tag{3}$$

As the first step, we perform regular binary coding $\xi_j \rightarrow \boldsymbol{z}_j \in \{0,1\}^{q_j}$. We use the term *inactive category* to refer to a category that is represented by 0; and an *active category* is represented by a 1. It is easy to see that the sum of all elements in $\boldsymbol{z}_j$ is strictly 1 if ξ_j is of multiclass, and could be $\mathbb{N}_0$ if ξ_j is a multilabel feature. These binary codings are also known as one-hot and multi-hot encodings, respectively.

In the second step, we transform the binary coded variable $\boldsymbol{z}_j$ in its fuzzy representation.

a) Multiclass case: We propose a transformation denoted as $f(\cdot)$ of $\boldsymbol{z}_j$ as:

$$\boldsymbol{x}_j(k) = f(\boldsymbol{z}_j(k)) = \begin{cases} \mathcal{U}[0, \frac{1}{q_j}) & \forall k : \boldsymbol{z}_j(k) = 0, \\ 1 - \sum_k \boldsymbol{x}_j(k) & \text{for } k : \boldsymbol{z}_j(k) = 1. \end{cases} \tag{4}$$

Please note that we use $\boldsymbol{x}_j(k)$ to denote the k-th element in the vector $\boldsymbol{x}_j$, in order to avoid double subscripts; $\boldsymbol{\mathcal{U}}[a,b)$ denotes a continuous uniform distribution in the interval of $[a,b)$. Assuming any active category to be k^*, then each of the $q_j - 1$ inactive categories is represented by a fraction $\boldsymbol{x}_j(k)$ which is uniformly sampled from $[0, \frac{1}{q_j})$. With this smoothing,

we can retain exactly the same information encoded in $\boldsymbol{z}_j$. It is easy to see that,

$$1 - \sum_{\forall k \neq k^*} \boldsymbol{x}_j(k) > \frac{1}{q_j}. \tag{5}$$

In other words, the left side of the inequation (5), which represents the active category k^*, is guaranteed to be larger than any fraction encoding an inactive category. Operations such as max, min, $argmax$ and $argmin$ applied on the fuzzy $\boldsymbol{x}_j$ are always able to decode the same information in $\boldsymbol{z}_j$.

b) Multilabel case: Since the categories are no more mutual exclusive, we can derive a fuzzy binary coding by simply taking 0.5 instead of $\frac{1}{q_j}$ as the upper bound of uniform sampling:

$$\boldsymbol{x}_j(k) = f(\boldsymbol{z}_j(k)) = \begin{cases} \mathcal{U}[0, 0.5) \text{ for } \boldsymbol{z}_j(k) = 0, \\ \mathcal{U}[0.5, 1] \text{ for } \boldsymbol{z}_j(k) = 1. \end{cases} \tag{6}$$

It is also guaranteed that the category information in $\boldsymbol{z}_j$ remains intact, since we can always recover $\boldsymbol{z}_j$ applying $I(\boldsymbol{x}_j \geq 0.5)$, where $I(\cdot)$ denotes the indicator function.

Transforming the binary codes into fuzzy binary codes prevents the discriminator from exploiting the fact that the generated values are all fractions and the real values only contain 0's and 1's. This fuzzy binary coding, especially the samplings in Eq. (4) and (6), can be performed only once as pre-processing step, or alternatively, prior to each training epoch. In our experiments, we implement the first variant.

In Fig. 3 we provide some empirical results based on our experiments, demonstrating that without the fuzzy binary coding trick, the adversarial training tends to diverge, i.e., the discriminator keeps improving itself by exploiting the obvious difference between the generated and real data. The generator, therefore, receives no gradients from the discriminator for improvement.

Applying the fuzzy encoding, the discriminator can be forced to focus on discovering the true difference between the real and generated data in term of their distributions and dependencies instead of their different domains. These discoveries in turn shall encourage the generator to approximate the real data distribution.

Lastly, we concatenate the feature vectors of all categorical features as

$$\bar{\boldsymbol{x}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_p] = [\boldsymbol{x}_j]_{j=1}^{p} \in [0, 1]^{\sum_{j=1}^{p} q_j}, \tag{7}$$

which form the inputs to the generative adversarial imputation network. Note here that we do not use another subscript denoting the data instance in $\boldsymbol{x}$, and simply assume that they are all i.i.d. samples.

B. Categorical Generative Adversarial Imputation Nets (Categorical GAINs)

a) Data notation: In order to represent the missingness of data, [14] introduced a binary mask vector $\boldsymbol{m}$ indicating which features are missing in a data instance represented by a real vector $\boldsymbol{\xi}$. Here $\boldsymbol{m}$ and $\boldsymbol{\xi}$ have exactly the same size and each element $\boldsymbol{m}(k)$ is 1 if $\boldsymbol{\xi}(k)$ is not missing, and 0 otherwise.

In case of categorical features, however, we introduce two masking mechanisms. Firstly, we use $\boldsymbol{\mu}$ to denote the missingness of $\boldsymbol{\xi}$, i.e.,

$$\mu_j = \begin{cases} 0 & \text{if } \xi_j \text{ is missing,} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

Once the features are binary and fuzzy coded, we construct another mask vector as:

$$\boldsymbol{m}_j = \begin{cases} \mathbf{0} & \text{if } \xi_j \text{ is missing,} \\ \mathbf{1} & \text{otherwise} \end{cases} \in [0, 1]^{q_j}, \tag{9}$$

where we denote a *vector* of 0's and 1's using $\mathbf{0}$ and $\mathbf{1}$, respectively. It can be interpreted as simply repeating μ_j for q_j times for the j-th feature. The rationale for these two kinds of masking is that the discriminator's prediction is in fact equivalent to the missingness of the data. For real-valued features discussed in [14], one could simply reuse the masking vector as the target of the discriminator. But for categorical features that are coded as binary or fuzzy binary, doing so would imply making a prediction for each single *category* instead of each *feature*. In the following introduction to the generator and discriminator, we shall give a more detailed explanation.

Analogously to the construction of $\bar{\boldsymbol{x}}$, we have the concatenation of $\boldsymbol{m}_j$'s:

$$\bar{\boldsymbol{m}} = [\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_p] = [\boldsymbol{m}_j]_{j=1}^{p} \in [0, 1]^{\sum_{j=1}^{p} q_j} \tag{10}$$

b) The generator: The generator takes as input i) the fuzzy binary coded feature vector $\bar{\boldsymbol{x}}$ that is expected to contain missing values, ii) the equally sized mask vector $\bar{\boldsymbol{m}}$ and iii) a random vector $\bar{\boldsymbol{r}} = [\boldsymbol{r}_j]_{j=1}^{p}$ functioning as seeds. The generator produces as output a single vector denoted $\bar{\boldsymbol{g}}$ that is supposed to contain imputed missing values in $\bar{\boldsymbol{x}}$:

$$\bar{\boldsymbol{g}} = G(\bar{\boldsymbol{x}}, \bar{\boldsymbol{m}}, \bar{\boldsymbol{r}}). \tag{11}$$

Specifically in our implementation, we build as generator a neural network with 3 hidden layers:

$$\boldsymbol{h}_1^G = \text{relu}(\boldsymbol{W}_1^G \cdot [\bar{\boldsymbol{x}} + (1 - \bar{\boldsymbol{m}}) \circ \bar{\boldsymbol{r}},\ \bar{\boldsymbol{m}}] + \boldsymbol{b}_1^G) \tag{12}$$

$$\boldsymbol{h}_2^G = \text{relu}(\boldsymbol{W}_2^G \cdot \boldsymbol{h}_1^G + \boldsymbol{b}_2^G) \tag{13}$$

$$\boldsymbol{h}_3^G = \text{relu}(\boldsymbol{W}_3^G \cdot \boldsymbol{h}_2^G + \boldsymbol{b}_3^G) \tag{14}$$

$$\boldsymbol{g}_j = \sigma(\boldsymbol{W}_o^G(j) \cdot \boldsymbol{h}_3^G + \boldsymbol{b}_o^G(j))\ \forall j \in [1, p] \tag{15}$$

$$\bar{\boldsymbol{g}} = \bar{\boldsymbol{m}} \circ \bar{\boldsymbol{x}} + (1 - \bar{\boldsymbol{m}}) \circ [\boldsymbol{g}_j]_{j=1}^{p} \tag{16}$$

As proposed by [14], the operation carried out in Eq. (12) first fills the missing values in $\bar{\boldsymbol{x}}$ with random seeds $\bar{\boldsymbol{r}}$, before feeding it to the neural network. The hidden layers $\boldsymbol{h}_1^G, \boldsymbol{h}_2^G, \boldsymbol{h}_3^G$ extract hierarchically the global context information from the input. In the last layer, we define for each categorical feature j a specific classification model. Depending on the distribution assumption of the feature, the activation function can be either sigmoid or softmax, both of which are denoted using σ for the sake of simplicity. In Eq. (16), the outputs from all p

activation functions are concatenated as $[\boldsymbol{g}_j]_{j=1}^p$. And if a specific feature is in fact not missing, the generated values are replaced by the real values. Similar to the Multiple Imputation by Chained Equations [5], this generator in fact attempts to approximate the real distribution $\boldsymbol{\pi}_{j^*}$ of each missing variable $\boldsymbol{X}_{j^*}$ conditioned on all other observed features $\boldsymbol{x}_j$, i.e.,

$$\hat{\boldsymbol{\pi}}_{j^*} = \boldsymbol{g}_{j^*} = \mathbb{P}(\boldsymbol{m}_{j^*} = \mathbf{1} \mid \{\boldsymbol{X}_j = \boldsymbol{x}_j\}_{\forall j:\mu_j=1}) \tag{17}$$

This architecture is illustrated in Fig. 1 with only two categorical features as examples.

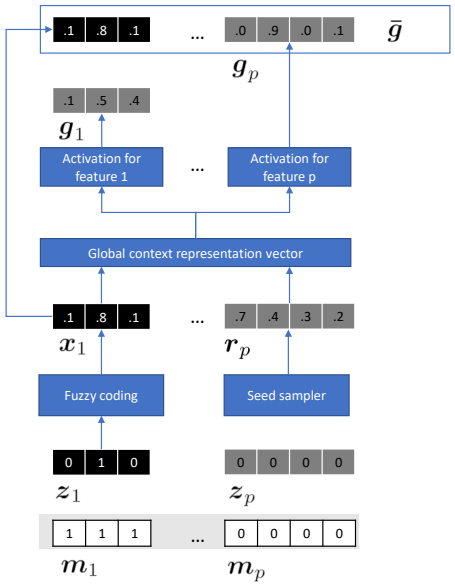

Fig. 1. A detailed illustration of the generator in *categorical GAIN* architecture. As an example, we visualize two categorical features that are binary coded as $\boldsymbol{z}_1$ and $\boldsymbol{z}_p$. The first is labeled as observed while the second as missing in $\boldsymbol{m}_1$ and $\boldsymbol{m}_p$. Therefore, the observed binary input of $\boldsymbol{z}_1 = [0, 1, 0]$ is transformed into a fuzzy representation of $\boldsymbol{x}_1 = [0.1, 0.8, 0.1]$. And the seed sampler fills these positions with random values in $\boldsymbol{r}_p$. On the output side, the generated values for the first feature are replaced by the original, fuzzy codes, since the true values are observed, i.e., $\boldsymbol{g}_1 := \boldsymbol{x}_1$. Only the generated values for the other feature $\boldsymbol{g}_p$ are exposed to the discriminator. The concatenation of the overall generator output is denoted as $\bar{\boldsymbol{g}}$.

c) The discriminator: Like the generator, the discriminator also consumes two input vectors. The first input is the concatenated output of the generator $\bar{\boldsymbol{g}}$. The second input is a hint vector as proposed by [14], which can be interpreted as a *masked mask vector*: For each data instance, one randomly samples a predefined portion of features and sets the corresponding entries in the mask vector $\bar{\boldsymbol{m}}$ to be 0.5. On the output side of the discriminator, we have again an concatenated vector $\hat{\boldsymbol{\mu}} = [\hat{\mu}]_{j=1}^p$. Each $\hat{\mu}_j$ is a point estimate of μ_j as defined in Eq. (8), indicating whether the j-th feature in the input, denoted as $\boldsymbol{g}_j$ is generated or real,

$$\hat{\boldsymbol{\mu}}_{j^*} = \mathbb{P}(\boldsymbol{g}_{j^*}\text{is real} \mid \{\boldsymbol{g}_j\}_{\forall j:j\neq j^*}). \tag{18}$$

Generally, we can describe the discriminator as

$$\hat{\boldsymbol{\mu}} = D(\bar{\boldsymbol{g}}, \bar{\boldsymbol{h}}). \tag{19}$$

Specifically for our experiments, we have a neural network with two hidden layers:

$$\boldsymbol{h}_1^D = \text{relu}(\boldsymbol{W}_1^D \cdot [\bar{\boldsymbol{g}}, \bar{\boldsymbol{h}}] + \boldsymbol{b}_1^G) \tag{20}$$

$$\boldsymbol{h}_2^D = \text{relu}(\boldsymbol{W}_2^D \cdot \boldsymbol{h}_1^D + \boldsymbol{b}_2^D) \tag{21}$$

$$\hat{\mu}_j = \sigma(\boldsymbol{w}_o^D(j)^T \cdot \boldsymbol{h}_3^G + b_o^D(j)) \ \forall j \in [1, p] \tag{22}$$

In parallel to the architecture of the generator, the first two hidden layers represent the global context information, while the last layer contains p logistic regression models. Each of them attempts to predict whether the j-th feature in the input $\boldsymbol{g}_j$ is generated. In the original GANs, each input vector to the discriminator is typically either generated or real. In GAIN, however, one input vector to the discriminator may contain generated and real data simultaneously, and the discriminator performs multiple predictions correspondingly.

In the original setting in [14], where the features are of real values, the training target of the discriminator is in fact identical with the mask vector. In case of categorical features, however, one should not directly utilize the mask vector $\bar{\boldsymbol{m}}$ as training target. Because in order to mask a (fuzzy) binary coded vector $\boldsymbol{x}_j$ completely, we have to define a same sized vector $\boldsymbol{m}_j$. Training a discriminator that attempts to recover every element in $\boldsymbol{m}_j$ is in fact a prediction for each *category* instead of *feature*. To this end, we propose to train the discriminator so that each $\hat{\mu}_j$ would approximate μ_j as in Eq. (8) for all real data. The generator, on the other hand, should make the discriminator assign a $\hat{\mu}_j$ that is close to $1 - \mu_j$ to all generated values.

The hint mechanism is also a crucial component in training the discriminator. Once a subset of entries in the mask vector is set to a neutral value of 0.5, the discriminator is enforced to predict whether the corresponding values in $\bar{\boldsymbol{g}}$ are real or generated. Such prediction is supposed to rely on other entries in $\bar{\boldsymbol{g}}$ that are provided to discriminator. The proportion of features that are neutralized in the hint vector therefore controls the amount of information from which the discriminator is supposed to learn the decision. In order to see that one could consider two extreme cases: With the proportion close to 1, the discriminator would attempt to perform prediction for a large amount of features in $\bar{\boldsymbol{g}}$, based on very few features that are denoted as either real or generated. This could be a challenging task for the discriminator and, more importantly, the discriminator may not learn to build the prediction based on the dependency between features. With a proportion that is close to 0, the hint vector becomes almost identical to the mask vector. In the original setting in [14], where features are of real values and the mask vector is in fact the prediction target of the discriminator, having two almost identical vectors as input and output of a neural network would cause the discriminator to simply learn an identity function, not being able to tell the difference between real and generated data. This is slightly less of a problem in case of categorical features, because as stated above, our mask vector as input to the discriminator and training target are not exactly identical, although they contain the same information on the missingness of the data.

To this end, for experiments, we include the hint mechanism and use a relatively small proportion of 0.1. This reveals 90% of available information of the data missingness to the discriminator, which is encouraged to build its prediction based on the dependency among features.

The hint vector also has to be adjusted for categorical features. Similar to the mask vectors, we define for each categorical feature j a hint vector $\boldsymbol{h}_j$ that consists of exclusively either 0 or 1, and denote the concatenation of all hint vectors as $\bar{\boldsymbol{h}} = [\boldsymbol{h}_j]_{j=1}^{p}$. The proposed approach in [14] would imply masking the missingness information for each *category* instead of each *feature*. To this end, we propose to first sample a subset out of the p features, and set the entire corresponding hint vectors to be 0.5, i.e. $\boldsymbol{h}_j = \mathbf{0.5}$, as can be seen in the illustration in Fig. 2

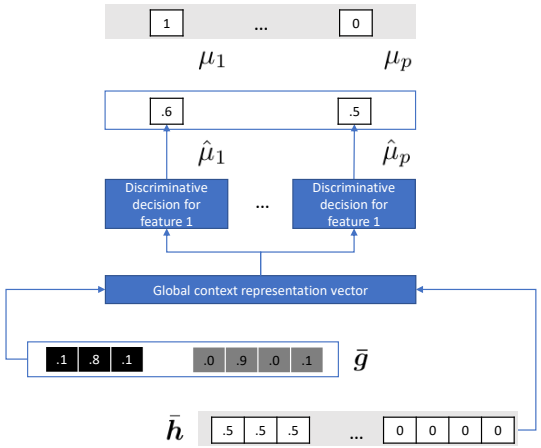

Fig. 2. A detailed illustration of the discriminator in categorical GAIN architecture. The input to the discriminator is the concatenated output $\bar{\boldsymbol{g}}$ from the generator, as well as the hint vector $\bar{\boldsymbol{h}}$. The output of the discriminator here consists of 2 scalars of $\hat{\mu}_1$ and $\hat{\mu}_p$. They are trained against the scalars μ_1 and μ_p, encoding the missingness of both features respectively.

d) Loss functions: Similar to the original GANs framework as in Eq. (2), GAIN also contains two adversarial loss functions $loss_D$ and $loss_G$:

$$loss_D = -\Sigma_j \left(\mu_j \cdot \log(\hat{\mu}_j) + (1-\mu_j) \cdot \log(1-\hat{\mu}_j)\right) \quad (23)$$

$$loss_G = -\Sigma_j (1-\mu_j) \cdot \log(\hat{\mu}_j) \quad (24)$$

The discriminator adjusts itself to make correct classification by minimizing the $loss_D$ in Eq. (23). This objective forces the discriminator to produce large $\hat{\mu}_j$ if $\mu_j = 1$, indicating that $\boldsymbol{x}_j$ is real. The generator learns to fool the discriminator by minimizing $loss_G$ in Eq. (24). This loss is adversarial to the second additive term in the discriminator loss. The generator encourages the discriminator to assign large probability $\hat{\mu}_j$ to features where $\mu_j = 0$, implying that the generated data should be classified as real.

As defined in Eq. (16), once a feature j is observed instead of missing, whatever is generated by the generator gets replaced by the actually observed values. The weight parameters responsible for these features will not get gradient signals for this specific training sample. Therefore, [14] proposes a new loss function that measures the similarity between generated and the observed feature values. In case of real-valued features this loss could be realized as mean-squared error. In our case, we apply the log-loss to measure the distance between probabilities and binary codes:

$$loss_{sim} = \Sigma_j \boldsymbol{m}_j^T (-\boldsymbol{x}_j) \log(\boldsymbol{g}_j). \quad (25)$$

This loss mechanism implies that, in case a feature is observed, the generator should learn to reproduce it based on all other observed features; and in case a feature is missing, the adversarial training forces the generator to produce values that the discriminator would believe to be real.

In comparison to the original GAIN architecture in [14], there are three adjustments that we propose for categorical features. First, the output activation function in the generator: in order to take into account the discrete distribution of the data features, we apply softmax or sigmoid activation functions instead of linear activation. Second, the target variable of the discriminator: In case of real valued features, the discriminator only needs to predict the mask vector $\bar{\boldsymbol{m}}$ which has the same shape as the feature vector $\bar{\boldsymbol{x}}$. This is because each element in the mask vector can represent the missingness of the corresponding feature. However, in order to encode the missingness of a feature containing multiple categories $\boldsymbol{x}_j$, it is unnecessary for the discriminator to recover the corresponding mask vector $\boldsymbol{m}_j$, since all values in this vector are either all 0's or all 1's. Instead, it is much more efficient to train the discriminator to predict the scalar μ_j. Thirdly, due to the same reason, the hint mechanism also has to be defined on the level of feature instead of categories. In other words, for a feature j we initialize a vector $\boldsymbol{h}_j$ from $\boldsymbol{m}_j$, and set all elements to be 0.5 if necessary.

V. Experiments

In this section, we provide experiments conducted on two datasets. The first dataset is publicly available and a well known benchmark for breast cancer classification based on categorical features. The second dataset is provided by the PRAEGNANT study [26], a Germany-wide clinical study for breast cancer research.

Please recall that we perform fuzzy binary coding of the categorical features and our generator produces values that range between 0 and 1. We recover the binary codes applying $I(\boldsymbol{x}_j \geq 0.5)$ for multilabel and $I(\boldsymbol{x}_j = \max(\boldsymbol{x}_j))$ for multi-class features as a post processing step. Because, as discussed in subsection IV-A, the encoded categorical information is always retained after the fuzzy binary coding and can be recovered completely.

A. Experiments on a public dataset

The breast cancer dataset is available on UCI data repository [27]. It contains 9 multiclass features (Tab. I) observed on 286 patient cases. The prediction target is to differentiate between 201 recurrence and 85 no-recurrence cases of the cancer.

Feature	#Categories
age	6
menopause	3
tumor-size	11
inv-nodes	7
node-caps	2
deg-malig	3
breast	2
breast-quad	5
irradiat	2

TABLE I
PATIENT FEATURES FROM THE UCI BREAST CANCER DATASET

We perform 5-fold cross-validation on the complete dataset, applying logistic regressions with ridge regularization (Tab. II) and report the prediction accuracy and AUROC scores. As sanity check we also provide these scores produced by random and most popular predictions, the latter of which constantly produces the frequency of the label class in the training set.

Methods	Accuracy	AUROC
Random prediction	0.516 ± 0.051	0.484 ± 0.053
Most popular prediction	0.707 ± 0.051	0.500 ± 0
Prediction on complete data	0.737 ± 0.056	0.721 ± 0.051

TABLE II
SANITY CHECKS FOR THE PREDICTION TASK ON THE UCI BREAST CANCER DATASET

For each cross-validation split, we randomly mask 10%, 20%, 30% 40% and 50% of the features. We then apply different imputation approaches to recover the masked values. Note that the imputation model is only trained on the training set, and applied on both training and test sets, in order to simulate a realistic setting. The predictive model is then trained on *imputed* training set and validated on the *imputed* test set.

As the first baseline method we implement a low-rank reconstruction model using SVD. Assuming $\boldsymbol{X}_{tr}$ and $\boldsymbol{X}_{te}$ as training and test sets containing missing values, we compose the former as $\boldsymbol{X}_{tr} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, and impute the training and test sets as $\tilde{\boldsymbol{X}}_{tr} = \boldsymbol{U}_r\boldsymbol{D}_r\boldsymbol{V}_r^T$ and $\tilde{\boldsymbol{X}}_{te} = \boldsymbol{X}_{te}(\boldsymbol{D}_r\boldsymbol{V}_r^T)^\dagger(\boldsymbol{D}_r\boldsymbol{V}_r^T)$, respectively. Here we denote the the low rank representation of $\boldsymbol{U}, \boldsymbol{D}$ and $\boldsymbol{V}^T$ using $\boldsymbol{U}_r, \boldsymbol{D}_r$ and $\boldsymbol{V}_r^T$ with a specific rank r. Please note that we do not perform any *argmax* to the reconstructed values.

The same ranks also apply to the second baseline model, which is an auto-encoder with non-linear $\tanh$ activation for the hidden layer. We summarize all prediction performances in term of accuracy and AUROC scores in Tab. III as average and standard deviation of the 5 cross-validation splits. For both baseline models we conduct experiments using 4 different ranks, i.e., the size of the hidden layer in AE, of 4, 8, 16 and 32, and report the best results. For the categorical GAIN model we perform 100-fold multiple imputation.

In term of accuracy, SVD reconstruction turns out to be more effective for this dataset, achieving the best accuracy in 4 out of 5 settings of masking proportions. In term of AUROC, categorical GAIN achieves 4 out of 5 cases. It is therefore interesting to note that for this dataset, the SVD decomposition

	Methods	Accuracy	AUROC
10%	No imputation	0.718 ± 0.067	0.66 ± 0.11
	Avg imputation	0.744 ± 0.05	0.639 ± 0.088
	SVD reconstruction	**0.776** ± 0.062	0.689 ± 0.114
	Auto-encoder	0.751 ± 0.047	0.652 ± 0.089
	Categorical GAIN	0.739 ± 0.066	**0.697** ± 0.098
20%	No imputation	0.711 ± 0.039	0.634 ± 0.07
	Avg imputation	0.707 ± 0.036	0.671 ± 0.065
	SVD reconstruction	**0.747** ± 0.046	0.664 ± 0.082
	Auto-encoder	0.729 ± 0.051	0.636 ± 0.038
	Categorical GAIN	0.71 ± 0.046	**0.697** ± 0.087
30%	No imputation	0.726 ± 0.031	0.644 ± 0.086
	Avg imputation	0.729 ± 0.04	0.665 ± 0.071
	SVD reconstruction	0.726 ± 0.053	0.689 ± 0.083
	Auto-encoder	0.726 ± 0.038	0.641 ± 0.053
	Categorical GAIN	**0.737** ± 0.032	**0.704** ± 0.042
40%	No imputation	0.678 ± 0.052	0.686 ± 0.058
	Avg imputation	0.708 ± 0.027	0.54 ± 0.066
	SVD reconstruction	**0.751** ± 0.026	**0.709** ± 0.059
	Auto-encoder	0.737 ± 0.033	0.638 ± 0.054
	Categorical GAIN	0.7 ± 0.017	0.686 ± 0.051
50%	No imputation	0.701 ± 0.044	0.607 ± 0.091
	Avg imputation	0.704 ± 0.044	0.632 ± 0.057
	SVD reconstruction	**0.747** ± 0.029	0.665 ± 0.063
	Auto-encoder	0.74 ± 0.034	0.635 ± 0.041
	Categorical GAIN	0.713 ± 0.025	**0.72** ± 0.044

TABLE III
PREDICTION PERFORMANCES ON IMPUTED UCI BREAST CANCER DATASET USING DIFFERENT APPROACHES

does not take into account the the fact that the feature values are in fact binary And yet the SVD reconstruction achieves comparable performances as categorical GAIN. This relatively simple technique, as well as many approaches that it has inspired, are widely applied in recommender systems and knowledge graph, where the most essential task is the completion of matrices and tensors. Therefore, it could very well present a simple and effective solution for data imputation as well. However, one should also note that the label distribution in this dataset is relatively unbalanced (201:85). Consequently, the most popular prediction as in Tab. II can already reach 70% accuracy. And even with complete data the prediction model cannot improve beyond 73.7%. The AUROC, in contrast, seems to be a more informative and convincing measurement, because the prediction on complete data achieves 72% while the most popular prediction 50%. Therefore, for this dataset, ROC seems to be a more reliable means to measure the prediction quality.

B. Experiments on the PRAEGNANT dataset

1) Cohort and Features: For our experiment, we extract EHR data on 1234 patients with metastatic breast cancer who have met the first line of treatment from the PRAEGNANT study network [26]. We build our predictive models based on features that are clinically relevant, as well as those that are based on an earlier study [28] aiming at automatically inferring the feature relevance in EHR data. The features included are listed in Tab. IV. We have 10 multiclass features and 9 multi-label featues, both of which are fuzzy-binary coded. The one

numeric feature is normalized between 0 and 1. Features such as current metastasis, metastasis estrogen receptor, metastasis progesterone receptor, AE/SAE and ECOG life status were originally temporal features. We aggregate and normalize these w.r.t the time dimension as in [29]. For these patients it is especially important for the physicians to decide, whether they should receive antihormone therapy or chemo therapy. The recorded clinical decision serve as ground truth, i.e. the target of our prediction. 750 of the 1234 patients have received antihormone, and the rest chemo therapy.

Multiclass features	#Categories
Staging at breast	15
Staging at axilla	8
Ever received antihormone therapy	8
Ever received chemo therapy	8
Metastasis by diagnostics	5
Tumor estrogen receptor status	4
Tumor progesterone receptor	4
Immunohistochemistry for HER2	6
Tumor grading	5
KI67	3
Multilabel features	**#Categories**
Staging of metastasis	10
Location of earlier metastasis	14
Current metastasis	4
Metastasis estrogen receptor	3
Metastasis progesterone receptor	3
HER2 IHC	5
Metastasis grading	4
AE/SAE	20
ECOG life status	4
Numerical features	**#Dimension**
Age	1

TABLE IV
PATIENT FEATURES FROM THE PRAEGNANT STUDY.

Here we apply almost exactly the same experimental setting as with the public dataset, except that, considering the feature space of higher dimension, we train the SVD and auto-encoder imputation models with an additional rank of 64.

Methods	Accuracy	AUROC
Random prediction	0.516 ± 0.029	0.526 ± 0.041
Most popular prediction	0.607 ± 0.046	0.500 ± 0
Prediction on complete data	0.710 ± 0.029	0.774 ± 0.039

TABLE V
SANITY CHECKS FOR THE PREDICTION TASK ON THE PRAEGNANT DATASET

2) Experimental Results: In Tab. V we could see there is a large improvement from most popular prediction to the prediction on complete data in term of both accuracy and AUROC.

In Tab. VI we could see that, the advantage of categorical GAIN only becomes visible as the masking proportion increases. For smaller proportion like 10% and 20%, simpler methods such as average imputation and SVD shows superior performances. With a proportion larger than 30%, categorical GAIN outperforms all other methods and the improvement grows with masking proportion. In term of AUROC, for instance, categorical GAIN can always achieve a score above

	Methods	Accuracy	AUROC
10%	No imputation	0.674 ± 0.017	0.718 ± 0.016
	Avg imputation	0.689 ± 0.008	$\mathbf{0.727} \pm 0.024$
	SVD reconstruction	$\mathbf{0.7} \pm 0.015$	0.645 ± 0.011
	Auto-encoder	0.609 ± 0.023	0.506 ± 0.022
	Categorical GAIN	0.645 ± 0.012	0.725 ± 0.024
20%	No imputation	0.669 ± 0.014	0.69 ± 0.011
	Avg imputation	$\mathbf{0.684} \pm 0.015$	0.707 ± 0.014
	SVD reconstruction	0.663 ± 0.025	0.621 ± 0.032
	Auto-encoder	0.609 ± 0.021	0.496 ± 0.03
	Categorical GAIN	0.649 ± 0.016	$\mathbf{0.716} \pm 0.022$
30%	No imputation	0.645 ± 0.03	0.696 ± 0.018
	Avg imputation	0.658 ± 0.039	0.695 ± 0.021
	SVD reconstruction	0.662 ± 0.04	0.599 ± 0.018
	Auto-encoder	0.609 ± 0.043	0.528 ± 0.017
	Categorical GAIN	$\mathbf{0.665} \pm 0.018$	$\mathbf{0.723} \pm 0.01$
40%	No imputation	0.652 ± 0.008	0.663 ± 0.009
	Avg imputation	0.643 ± 0.012	0.66 ± 0.014
	SVD reconstruction	0.658 ± 0.01	0.6 ± 0.017
	Auto-encoder	0.608 ± 0.017	0.494 ± 0.034
	Categorical GAIN	$\mathbf{0.666} \pm 0.017$	$\mathbf{0.711} \pm 0.015$
50%	No imputation	0.635 ± 0.027	0.646 ± 0.029
	Avg imputation	0.649 ± 0.041	0.643 ± 0.038
	SVD reconstruction	0.644 ± 0.015	0.566 ± 0.018
	Auto-encoder	0.608 ± 0.013	0.509 ± 0.038
	Categorical GAIN	$\mathbf{0.654} \pm 0.05$	$\mathbf{0.705} \pm 0.029$

TABLE VI
PREDICTION PERFORMANCES ON IMPUTED PRAEGNANT DATASET USING DIFFERENT APPROACHES

70%, while the other performance of other methods drop much faster as the proportion of missing data increases. This agrees with findings in [14], that it is especially advantageous to apply GAIN to impute data in case of a relatively higher missing rate.

One might also hypothesize that the GAIN framework, consisting of relatively complex neural networks, profit from increasing number of training samples. For a smaller dataset such as the public breast cancer dataset, it seems more reasonable to first experiment with simpler methods such as SVD reconstruction. The GAIN approach, on the other hand, turns out to be more appropriate in case of large number of training samples and more complex feature dependencies.

We also present in Fig. 3 the development of the losses of discriminator (top) and generator (bottom), trained on binary (left) and fuzzy coded (right) features. In case of plain binary coded features, it is clear that the adversarial training fails since the generator loss increases, while the discriminator loss decreases constantly. This implies that the discriminator can always tell the real data from generated ones. Consequently, the generator cannot improve itself by learning to generate important characteristics in the feature distribution. When we apply the fuzzy binary coding, in contrast, the generator can improve itself by lowering its loss, i.e., it gets harder and harder for the discriminator to make the decision. In addition, as expected, varying proportion of missing data (masking) has impact on the adversarial training losses. With larger proportion of missing data, the imputation task becomes more

challenging and both discriminator loss and generator loss are expected to increase with larger proportion. This verifies empirically our hypothesis, that, if one applies softmax as the final activation in the generator to generate categorical data, the adversarial training fails as the discriminator can learn to exploit the huge difference in the generated and real data. This typically results in divergence of the adversarial training. By re-coding the binary features in a fuzzy way while retaining the information, we enforce the real data to resemble what softmax would produce. Thus we can make both discriminator and generator converge in training.

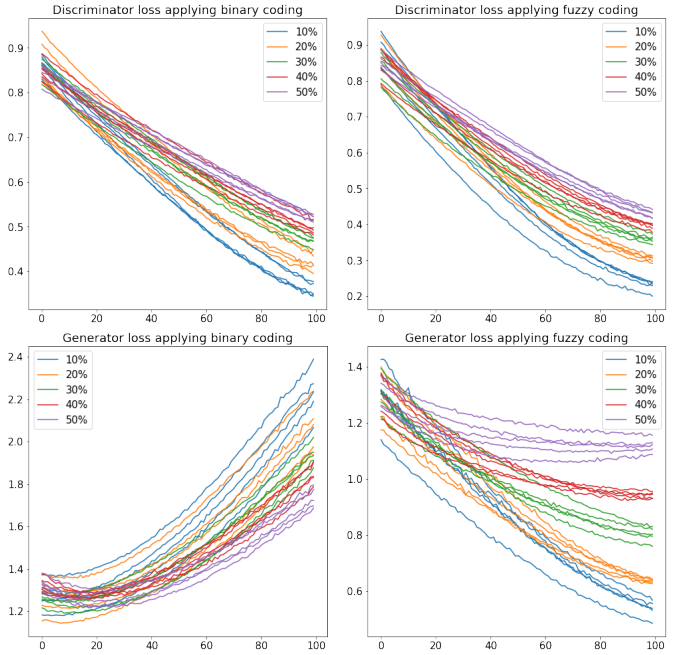

Fig. 3. Losses in adversarial training on the PREAGNANT dataset. X-axis: training epochs; Y-axis: adversarial loss. Above: Discriminator losses with binary coding (left) and fuzzy binary coding (right). Bottom: Generator losses with binary coding (left) and fuzzy binary coding (right).

VI. Summary

In this paper, we have proposed a Categorical Generative Adversarial Nets (*Categorical GAIN*) for EHR data imputation, based on a framework that is originally designed for real values. First, we have hypothesized that applying softmax functions as output activation in the generator directly often results in the discriminator exploiting the obvious difference between generated and real values. And the adversarial training typically ends up in divergence. We have proposed to perform fuzzy coding of the binary values so that they resemble generated values while retaining the encoded information. Secondly, we have performed multiple modifications in the architectures of both generator and discriminator, in order to handle the fuzzy binary coded features.

We have compared our methods with a variety of benchmark methods on two EHR datasets. We have simulated different proportions of missing data by masking out known values and then attempting to perform prediction tasks based on imputed data. We could show that the more complex method of generative adversarial nets turned out to be advantageous in case of relatively higher missing rate and larger training data set, while the simpler methods such as SVD reconstruction and average imputation are more reliable to impute smaller proportion of missing data.

Acknowledgment

The authors acknowledge support by the German Federal Ministry for Education and Research (BMBF), funding project "MLWin" (grant 01IS18050).

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

References

[1] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2014, vol. 333.

[2] V. Tresp, R. Neuneier, and S. Ahmad, "Efficient methods for dealing with missing data in supervised learning," in *Advances in neural information processing systems*, 1995, pp. 689–696.

[3] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan, "Strategies for handling missing data in electronic health record derived data," *eGEMs*, vol. 1, no. 3, 2013.

[4] E. M. Mirkes, T. J. Coats, J. Levesley, and A. N. Gorban, "Handling missing data in large healthcare dataset: A case study of unknown trauma outcomes," *Computers in biology and medicine*, vol. 75, pp. 203–216, 2016.

[5] E. A. Stuart, M. Azur, C. Frangakis, and P. Leaf, "Multiple imputation with large data sets: a case study of the children's mental health initiative," *American journal of epidemiology*, vol. 169, no. 9, pp. 1133–1139, 2009.

[6] D. B. Rubin, *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, 2004, vol. 81.

[7] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[8] T. E. Raghunathan, J. M. Lepkowski, J. Van Hoewyk, and P. Solenberger, "A multivariate technique for multiply imputing missing values using a sequence of regression models," *Survey methodology*, vol. 27, no. 1, pp. 85–96, 2001.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[10] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[12] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network." in *CVPR*, vol. 2, no. 3, 2017, p. 4.

[13] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training." in *CVPR*, vol. 2, no. 4, 2017, p. 5.

[14] J. Yoon, J. Jordon, and M. van der Schaar, "Gain: Missing data imputation using generative adversarial nets," *arXiv preprint arXiv:1806.02920*, 2018.

[15] S. Rajeswar, S. Subramanian, F. Dutil, C. Pal, and A. Courville, "Adversarial generation of natural language," *arXiv preprint arXiv:1705.10929*, 2017.

[16] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[17] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[18] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," *arXiv preprint arXiv:1706.03850*, 2017.

[19] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," *arXiv preprint arXiv:1703.06490*, 2017.

[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[21] M. J. Kusner and J. M. Hernández-Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," *arXiv preprint arXiv:1611.04051*, 2016.

[22] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[23] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, vol. 21, 2016.

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[25] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[26] P. Fasching, S. Brucker, T. Fehm, F. Overkamp, W. Janni, M. Wallwiener, P. Hadji, E. Belleville, L. Häberle, F.-A. Taran *et al.*, "Biomarkers in patients with metastatic breast cancer and the praegnant study network," *Geburtshilfe und Frauenheilkunde*, vol. 75, no. 01, pp. 41–50, 2015.

[27] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[28] Y. Yang, V. Tresp, M. Wunderle, and P. A. Fasching, "Explaining therapy predictions with layer-wise relevance propagation in neural networks," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2018, pp. 152–162.

[29] C. Esteban, D. Schmidt, D. Krompaß, and V. Tresp, "Predicting sequences of clinical events by using a personalized temporal latent embedding model," in *Healthcare Informatics (ICHI), 2015 International Conference on*. IEEE, 2015, pp. 130–139.