Generating Multi-label Discrete Patient Records using Generative Adversarial Networks

Edward Choi[1] MP2893@GATECH.EDU
Siddharth Biswal[1] SBISWAL7@GATECH.EDU
Bradley Malin[2] BRADLEY.MALIN@VANDERBILT.EDU
Jon Duke[1] JON.DUKE@GATECH.EDU
Walter F. Stewart[3] STEWARWF@SUTTERHEALTH.ORG
Jimeng Sun[1] JSUN@CC.GATECH.EDU

[1] GEORGIA INSTITUTE OF TECHNOLOGY [2] VANDERBILT UNIVERSITY [3] SUTTER HEALTH

Abstract

Access to electronic health record (EHR) data has motivated computational advances in medical research. However, various concerns, particularly over privacy, can limit access to and collaborative use of EHR data. Sharing synthetic EHR data could mitigate risk.

In this paper, we propose a new approach, medical Generative Adversarial Network (`medGAN`), to generate realistic synthetic patient records. Based on input real patient records, `medGAN` can generate high-dimensional discrete variables (e.g., binary and count features) via a combination of an autoencoder and generative adversarial networks. We also propose minibatch averaging to efficiently avoid mode collapse, and increase the learning efficiency with batch normalization and shortcut connections. To demonstrate feasibility, we showed that `medGAN` generates synthetic patient records that achieve comparable performance to real data on many experiments including distribution statistics, predictive modeling tasks and a medical expert review. We also empirically observe a limited privacy risk in both identity and attribute disclosure using `medGAN`.

1. Introduction

The adoption of electronic health records (EHR) by healthcare organizations (HCOs), along with the large quantity and quality of data now generated, has led to an explosion in *computational health*. However, the wide adoption of EHR systems does not automatically lead to easy access to EHR data for researchers. One reasons behind limited access stems from the fact that EHR data are composed of personal identifiers, which in combination with potentially sensitive medical information, induces privacy concerns. As a result, access to such data for secondary purposes (e.g., research) is regulated, as well as controlled by the HCOs groups that are at risk if data are misused or breached. The review process by legal departments and institutional review boards can take months, with no guarantee of access (Hodge Jr et al., 1999). This process limits timely opportunities to use data and may slow advances in biomedical knowledge and patient care (Gostin et al., 2009).

HCOs often aim to mitigate privacy risks through the practice of de-identification (for Civil Rights, 2013), typically through the perturbation of potentially identifiable attributes (e.g., dates of birth) via generalization, suppression or randomization. (El Emam et al., 2015) However, this approach is not impregnable to attacks, such as linkage via residual information to re-identify the individuals to whom the data corresponds (El Emam et al., 2011b). An alternative approach to de-identification is to generate synthetic data (McLachlan et al., 2016; Buczak et al., 2010; Lombardo and Moniz, 2008). However, realizing this goal in practice has been challenging because the resulting synthetic data are often not sufficiently realistic for machine learning tasks. Since many machine learning models for EHR data use aggregated discrete features derived from longitudinal EHRs, we concentrate our effort on generating such aggregated data in this study. Although it is ultimately

desirable to generate longitudinal event sequences, in this work we focus on generating high-dimensional discrete variables, which is an important and challenging problem on its own.

Generative adversarial networks (GANs) have recently been shown achieve impressive performance in generating high-quality synthetic images (Goodfellow et al., 2014; Radford et al., 2015; Goodfellow, 2016). To understand how, it should first be recognized that a GAN consists of two components: a *generator* that attempts to generate realistic, but fake, data and a *discriminator* that aims to distinguish between the generated fake data and the real data. By playing an adversarial game against each other, the generator can learn the distribution of the real samples - provided that both the generator and the discriminator are sufficiently expressive. Empirically, a GAN outperforms other popular generative models such as variational autoencoders (VAE) (Kingma and Welling, 2013) and PixelRNN/PixelCNN (van den Oord et al., 2016b,a) on the quality of data (i.e., fake compared to real), in this case images, and on processing speed (Goodfellow, 2016). However, GANs have not been used for learning the distribution of discrete variables.

To address this limitation, we introduce `medGAN`, a neural network model that generates high-dimensional, multi-label discrete variables that represent the events in EHRs (e.g., diagnosis of a certain disease or treatment of a certain medication). Using EHR source data, `medGAN` is designed to learn the distribution of discrete features, such as diagnosis or medication codes via a combination of an autoencoder and the adversarial framework. In this setting, the autoencoder assists the original GAN to learn the distribution of multi-label discrete variables. The specific contributions of this work are as follows:

- We define an efficient algorithm to generate high-dimensional multi-label discrete samples by combining an autoencoder with GAN, which we call `medGAN`. This algorithm is notable in that it handles both binary and count variables.
- We propose a simple, yet effective, method called *minibatch averaging* to cope with the situation where GAN overfits to a few training samples (i.e., mode collapse), which outperforms previous methods such as *minibatch discrimination*.
- We demonstrate a close-to-real data performance of `medGAN` using real EHR datasets on a set of diverse tasks, which include reporting distribution statistics, classification performance and medical expert review.
- We empirically show that `medGAN` leads to acceptable privacy risks in both presence disclosure (i.e., discovery that a patient's record contributed to the GAN) and attribute disclosure (i.e., discovery of a patient's sensitive medical data).

2. Related work

In this section, we begin with a discussion of existing methods for generating synthetic EHR data. This is followed by a review recent advances in generative adversarial networks (GANs). Finally, we summarize specific investigations into generating discrete variables using GANs.

Synthetic Data Generation for Health Data: De-identification of EHR data is currently the most generally accepted technical method for protecting patient privacy when sharing EHR data for research in practice (Johnson et al., 2016). However, de-identification does not guarantee that a system is devoid of risk. In certain circumstances, re-identification of patients can be accomplished through residual distinguishable patterns in various features (e.g., demographics (Sweeney, 1997; El Emam et al., 2011a), diagnoses (Loukides et al., 2010), lab tests (Atreya et al., 2013), visits across healthcare providers (Malin and Sweeney, 2004), and genomic variants (Erlich and Narayanan, 2014)) To mitigate re-identification vulnerabilities, researchers in the statistical disclosure control community have investigated how to generate synthetic datasets. Yet, historically, these approaches have been limited to summary statistics for only several variables at a time (e.g., (Dreschsler, 2011; Reiter, 2002). For instance McLachlan et al.(2016) used clinical practice guidelines and health incidence statistics with a state transition machine to generate synthetic patient datasets.

There is some, but limited, work on synthetic data generation in the healthcare domain and, the majority that has, tend to be disease specific. For example, Buczak et al. (2010) generated EHRs to explore questions related to the outbreak of specific illnesses, where care patterns in the source EHRs were applied to generate synthetic datasets. Many of these methods often rely heavily upon domain-specific knowledge along with actual data to generate synthetic EHRs (Lombardo and Moniz, 2008). More recently, and most related to our

work, a privacy-preserving patient data generator was proposed based on a perturbed Gibbs sampler (Park et al., 2013). Still, this approach can only handle binary variables and its utility was assessed with only a small, low-dimensional dataset. By contrast, our proposed `medGAN` directly captures general EHR data without focusing on a specific disease, which makes it suitable for a greater diversity of applications.

GAN and its Applications: Attempts to advance GANs (Goodfellow et al., 2014) include, but are not limited to, using convolutional neural networks to improve image processing capacity (Radford et al., 2015), extending GAN to a conditional architecture for higher quality image generation (Mirza and Osindero, 2014; Denton et al., 2015; Odena et al., 2016), and text-to-image generation (Reed et al., 2016). We, in particular, pay attention to the recent studies that attempted to handle discrete variables using GANs.

One way to generate discrete variables with GAN is to invoke reinforcement learning. SeqGAN (Yu et al., 2016) trains GAN with REINFORCE (Williams, 1992) and Monte-Carlo search to generate word sequences. Although REINFORCE enables an unbiased estimation of the gradients of the model via sampling, the estimates come with a high variance. Moreover, SeqGAN focuses on sampling one word (*i.e.* one-hot) at each timestep, whereas our goal is to generate multi-label binary/count variables. Alternatively, one could use specialized distributions, such as the Gumbel-softmax (Jang et al., 2016; Kusner and Hernández-Lobato, 2016), a concrete distribution (Maddison et al., 2016) or a soft-argmax function (Zhang et al., 2016) to approximate the gradient of the model from discrete samples. However, since these approaches focus on the softmax distribution, they cannot be directly invoked for multi-label discrete variables, especially in the count variable case. Yet another way to handle discrete variables is to generate distributed representations, then decode them into discrete outputs. For example, Glover (2016) generated document embeddings with a GAN, but did not attempt to simulate actual documents.

To handle high-dimensional multi-label discrete variables, `medGAN` generates the distributed representations of patient records with a GAN. It then decodes them to simulated patient records with an autoencoder.

3. Method

This section begins with a formalization of the structure of EHR data and the corresponding mathematical notation we adopt in this work, This is followed by a detailed description of the `medGAN` algorithm.

3.1 Description of EHR Data and Notations

We assume there are $|\mathcal{C}|$ discrete variables (*e.g.*, diagnosis, medication or procedure codes) in the EHR data that can be expressed as a fixed-size vector $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$, where the value of the i^{th} dimension indicates the number of occurrences (*i.e.*, counts) of the i-th variable in the patient record. In addition to the count variables, a visit can also be represented as a binary vector $\mathbf{x} \in \{0,1\}^{|\mathcal{C}|}$, where the i^{th} dimension indicates the absence or occurrence of the i^{th} variable in the patient record. It should be noted that we can also represent demographic information, such as age and gender, as count and binary variables, respectively.

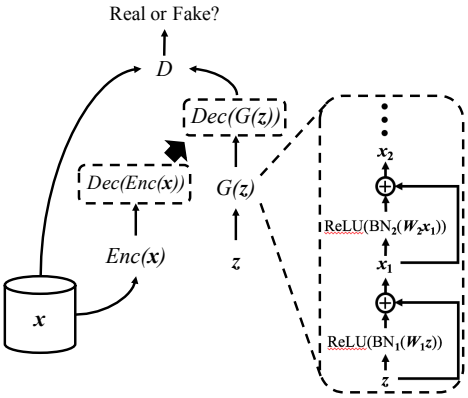
Learning the distribution of count variables is generally more difficult than learning the distribution of binary variables. This is because the model needs to learn more than simple co-occurrence relations between the various dimensions. Moreover, in EHR data, certain clinical concepts tend to occur much more frequently (e.g., essential hypertension) than others. This is problematic because it can skew a distribution around different dimensions.

3.2 Preliminary: Generative Adversarial Network

In a GAN, the generator $G(\mathbf{z};\theta_g)$ accepts a random prior $\mathbf{z} \in \mathbb{R}^r$ and generates synthetic samples $G(\mathbf{z}) \in \mathbb{R}^d$, while the discriminator $D(\mathbf{x};\theta_d)$ determines whether a given sample is real or fake. The optimal discriminator D^* would perfectly distinguish real samples from fake samples. The optimal generator G^* would generate fake samples that are indistinguishable from the real samples so that D is forced to make random guesses. Formally, D and G play the following minimax game with the value function $V(G,D)$:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

Figure 1: Architecture of `medGAN`: The discrete $\mathbf{x}$ comes from the source EHR data, $\mathbf{z}$ is the random prior for the generator G; G is a feedforward network with shortcut connections (right-hand side figure); An autoencoder (i.e, the encoder Enc and decoder Dec) is learned from $\mathbf{x}$; The same decoder Dec is used after the generator G to construct the discrete output. The discriminator D tries to differentiate real input $\mathbf{x}$ and discrete synthetic output $Dec(G(\mathbf{z}))$.



where p_{data} is the distribution of the real samples and $p_{\mathbf{z}}$ is the distribution of the random prior, for which $\mathcal{N}(0,1)$ is generally used. Both G and D iterate in optimizing the respective parameters θ_g and θ_d as follows,

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_i) + \log(1 - D(G(\mathbf{z}_i)))$$

$$\theta_g \leftarrow \theta_g - \alpha \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}_i)))$$

where m is the size of the minibatch and α the step size. In practice, however, G can be trained to maximize $\log(D(G(\mathbf{z}))$ instead of minimizing $\log(1 - D(G(\mathbf{z}))$ to provide stronger gradients in the early stage of the training (Goodfellow et al., 2014) as follows,

$$\theta_g \leftarrow \theta_g + \alpha \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log D(G(\mathbf{z}_i)) \tag{1}$$

Henceforth, we use Eq.(1) as it showed significantly more stable performance in our investigation. We also assume throughout the paper that both D and G are implemented with feedforward neural networks.

3.3 `medGAN`

Since the generator G is trained by the error signal from the discriminator D via backpropagation, the original GAN can only learn to approximate discrete patient records $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$ with continuous values. We alleviate this limitation by leveraging the autoencoder. Autoencoders are trained to project given samples to a lower dimensional space, then project them back to the original space. Such a mechanism leads the autoencoder to learn salient features of the samples and has been successfully used in certain applications, such as image processing (Goodfellow et al., 2016; Vincent et al., 2008).

In this work, We apply the autoencoder to learn the salient features of discrete variables that can be applied to decode the continuous output of G. This allows the gradient flow from D to the decoder Dec to enable the end-to-end fine-tuning. As depicted by Figure 1, an autoencoder consists of an encoder $Enc(\mathbf{x}; \theta_{enc})$ that compresses the input $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$ to $Enc(\mathbf{x}) \in \mathbb{R}^h$, and a decoder $Dec(Enc(\mathbf{x}); \theta_{dec})$ that decompresses $Enc(\mathbf{x})$ to $Dec(Enc(\mathbf{x}))$ as the reconstruction of the original input $\mathbf{x}$. The objective of the autoencoder is to minimize the reconstruction error:

$$\frac{1}{m} \sum_{i=0}^{m} ||\mathbf{x}_i - \mathbf{x}'_i||_2^2 \tag{2}$$

$$\frac{1}{m} \sum_{i=0}^{m} \mathbf{x}_i \log \mathbf{x}'_i + (1 - \mathbf{x}_i) \log(1 - \mathbf{x}'_i) \tag{3}$$

$$\text{where } \mathbf{x}'_i = Dec(Enc(\mathbf{x}_i))$$

where m is the size of the mini-batch. We use the mean squared loss (Eq.(2)) for count variables and cross entropy loss (Eq.(3)) for binary variables. For count variables, we use rectified linear units (ReLU) as the

activation function in both Enc and Dec. For binary variables, we use tanh activation for Enc and the sigmoid activation for Dec.[1]

With the pre-trained autoencoder, we can allow GAN to generate distributed representation of patient records (i.e., the output of the encoder Enc), rather than generating patient records directly. Then the pre-trained decoder Dec can pick up the right signals from $G(\mathbf{z})$ to convert it to the patient record $Dec(G(\mathbf{z}))$. The discriminator D is trained to determine whether the given input is a synthetic sample $Dec(G(\mathbf{z}))$ or a real sample $\mathbf{x}$. The architecture of the proposed model `medGAN` is depicted in Figure 1. `medGAN` is trained in a similar fashion as the original GAN as follows,

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_i) + \log(1 - D(\mathbf{x}_{\mathbf{z}_i}))$$

$$\theta_{g,dec} \leftarrow \theta_{g,dec} + \alpha \nabla_{\theta_{g,dec}} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_{\mathbf{z}_i})$$

$$\text{where } \mathbf{x}_{\mathbf{z}_i} = Dec(G(\mathbf{z}_i))$$

It should be note that we can round the values of $Dec(G(\mathbf{z}))$ to their nearest integers to ensure that the discriminator D is trained on discrete values instead of continuous values. We experimented both with and without rounding and empirically found that training D in the latter scenario led to better predictive performance in section 4.2. Therefore, we assume, for the remainder of this paper, that D is trained without explicit rounding.

We fine-tune the pre-trained parameters of the decoder θ_{dec} while optimizing for G. Therefore, the generator G can be viewed as a neural network with an extra hidden layer pre-trained to map continuous samples to discrete samples. We used ReLU for all of G's activation functions, except for the output layer, where we used the tanh function[2]. For D, we used ReLU for all activation functions except for the output layer, where we used the sigmoid function for binary classification.

3.4 Minibatch Averaging

Since the objective of the generator G is to produce samples that can fool the discriminator D, G could learn to map different random priors $\mathbf{z}$ to the same synthetic output, rather than producing diverse synthetic outputs. This problem is denoted as *mode collapse*, which arises most likely due to the GAN's optimization strategy often solving the max-min problem instead of the min-max problem (Goodfellow, 2016). Some methods have been proposed to cope with mode collapse (e.g., minibatch discrimination and unrolled GANs), but they require *ad hoc* fine-tuning of the hyperparameters and scalability is often neglected (Salimans et al., 2016; Metz et al., 2016).

By contrast, `medGAN` offers a simple and efficient method to cope with mode collapse when generating discrete outputs. Our method, *minibatch averaging*, is motivated by the philosophy behind minibatch discrimination. It allows the discriminator D to view the minibatch of real samples $\mathbf{x}_1, \mathbf{x}_2, \ldots$ and the minibatch of the fake samples $G(\mathbf{z}_1), G(\mathbf{z}_2), \ldots$, respectively, while classifying a real sample and a fake sample. Given a sample to discriminate, minibatch discrimination calculates the distance between the given sample and every sample in the minibatch in the latent space. Minibatch averaging, by contrast, provides the average of the minibatch samples to D, modifying the objective as follows:

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_i, \bar{\mathbf{x}}) + \log(1 - D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}}))$$

$$\theta_{g,dec} \leftarrow \theta_{g,dec} + \alpha \nabla_{\theta_{g,dec}} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}})$$

$$\text{where } \bar{\mathbf{x}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i, \quad \mathbf{x}_{\mathbf{z}_i} = Dec(G(\mathbf{z}_i)), \quad \bar{\mathbf{x}}_{\mathbf{z}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_{\mathbf{z}_i}$$

where m denotes the size of the minibatch. Specifically, the average of the minibatch is concatenated on the sample and provided to the discriminator D.

1. We considered a denoising autoencoder (dAE) (Vincent et al., 2008) as well, but there was no discernible improvement in performance.
2. We also applied tanh activation for the encoder Enc for consistency.

Binary variables: When processing binary variables $\mathbf{x} \in \{0,1\}^{|\mathcal{C}|}$, the average of minibatch samples $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}_{\mathbf{z}}$ are equivalent to the maximum likelihood estimate of the Bernoulli success probability $\hat{p}_k$ of each dimension k. This information makes it easier for D to ascertain whether a given sample is real or fake, if $\hat{p}_k$'s of fake samples are considerably different from those of real samples. This is especially likely when mode collapse occurs because the $\hat{p}_k$'s for most dimensions of the fake samples become dichotomized (either 0 or 1), whereas the $\hat{p}_k$'s of real samples generally take on a value between 0 and 1. Therefore, if G wants to fool D, it will have to generate more diverse examples within the minibatch $Dec(G(\mathbf{z}_1, \mathbf{z}_2, \ldots))$.
Count variables: Count variables are a more accurate description of clinical events. They can indicate the number of times a certain diagnosis was made or a certain medication was prescribed over multiple hospital visits. For count variables $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$, the average of minibatch samples $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}_{\mathbf{z}}$ can be viewed as the estimate of the binomial distribution mean $n\widehat{p_k}$ of each dimension k, where n is the number of hospital visits. Hence minibatch averaging for the count variables also provides helpful statistics to the discriminator D, guiding the generator G to generate more diverse and realistic samples. As our experiments show, minibatch averaging works surprisingly well and does not require additional parameters like minibatch discrimination. As a consequence, it has minimal impact to the training time. It is further worth mentioning that, for both binary and count variables, a minibatch that is larger than usual is recommended to properly capture the statistics of the real data. We use 1,000 records for a minibatch in this investigation.

3.5 Enhanced Generator Training

Similar to image processing GANs, we observed that balancing the power of D and G in the multi-label discrete variable setting was quite challenging (Goodfellow, 2016). Empirically, we observed that training `medGAN` with minibatch averaging demonstrated D consistently overpowering G after several iterations. While G still managed to learn under such situation, the performance seemed suboptimal, and updating θ_g and θ_{dec} more often than θ_d in each iteration only degraded performance. Considering the importance of an optimal D (Goodfellow, 2016), we chose not to limit the discriminative power of D, but rather improve the learning efficiency of G by applying batch normalization (Ioffe and Szegedy, 2015) and shortcut connection (He et al., 2016). G's k^{th} layer is now formulated as follows:

$$\mathbf{x}_k = \text{ReLU}(\text{BN}_k(\mathbf{W}_k\mathbf{x}_{k-1})) + \mathbf{x}_{k-1}$$

where ReLU is the rectified linear unit, BN_k is the batch normalization at the k-th layer, $\mathbf{W}_k$ is the weight matrix of the k-th layer, and $\mathbf{x}_{k-1}$ is the input from the previous layer. The right-hand side of Figure 1 depicts the first two layers of G. Note that we do not incorporate the bias variable into each layer because batch normalization negates the necessity of the bias term. Additionally, batch normalization and shortcut connections could be applied to the discriminator D, but the experiments showed that D was consistently overpowering G without such techniques, and we empirically found that a simple feedforward network was sufficient for D. We describe the overall optimization algorithm in the Appendix A.

3.6 Privacy Consideration

When EHRs are de-identfied via methods such generalization or randomization, there often remains a 1-to-1 mapping to the underlying records from where they were derived. However, in our case, the mapping between the generated data from `medGAN` and the training data of specific patients is not explicit. Intuitively, this seems to imply that the privacy of the patients can be better preserved with `medGAN`; however, it also begs the question of how to evaluate the privacy in the system. We perform a formal assessment of `medGAN`'s privacy risks based on two definitions of privacy.
Presence disclosure occurs when an attacker can determine that `medGAN` was trained with a dataset including the record from patient x. (Nergiz and Clifton, 2010) Presence disclosure for `medGAN` happens when a powerful attacker, one who already possesses the complete records of a set of patients P, can determine whether anyone from P are in the training set by observing the generated patient records. More recently, for machine learned models, this has been referred to as an *membership inference attack* (Shokri et al., 2017). the knowledge gained by the attacker may be limited, if the dataset is well balanced in its clinical concepts.
Attribute disclosure occurs when attackers can derive additional attributes such as diagnoses and medications about patient x based on a subset of attributes they already know about x. (Matwin et al., 2015) We believe

Table 1: Basic statistics of datasets A, B and C

Dataset	(A) Sutter PAMF	(B) MIMIC-III	(C) Sutter Heart Failure
# of patients	258,559	46,520	30,738
# of unique codes	615	1071	569
Avg. # of codes per patient	38.37	11.27	53.02
Max # of codes for a patient	198	90	871
Min # of codes for a patient	1	1	2

that attribute disclosure for medGAN could be a more prominent issue because the attacker only needs to know a subset of attributes of a patient. Moreover, the goal of the attacker is to gain knowledge of the unknown attributes by observing similar patients generated by medGAN.

Considering the difficulty of deriving analytic proof of privacy for GANs and simulated data, we report the empirical analysis of both risks to understand the extent to which privacy can be achieved, as commonly practiced in the statistical disclosure control community. (Domingo-Ferrer and Torra, 2003)

4. Experiments

We evaluated medGAN with three distinct EHR datasets. First, we describe the datasets and baseline models. Next, we report the quantitative evaluation results using both binary and count variables. We then perform a qualitative analysis through medical expert review. Finally, we address the privacy aspect of medGAN. The source code of medGAN is publicly available at https://github.com/mp2893/medgan.

4.1 Experimental Setup

Source data: The datasets in this study were from A) Sutter Palo Alto Medical Foundation (PAMF), which consists of 10-years of longitudinal medical records of 258K patients, B) the MIMIC-III dataset (Johnson et al., 2016; Goldberger et al., 2000), which is a publicly available dataset consisting of the medical records of 46K intensive care unit (ICU) patients over 11 years old and C) a heart failure study dataset from Sutter, which consists of 18-months observation period of 30K patients. From dataset A and C, we extracted diagnoses, medications and procedure codes, which were then respectively grouped by Clinical Classifications Software (CCS) for ICD-9[3], Generic Product Identifier Drug Group[4] and for CPT[5]. From dataset B, we extracted ICD9 codes only and grouped them by generalizing up to their first 3 digits. Finally, we aggregate a patient's longitudinal record into a single fixed-size vector $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$, where $|\mathcal{C}|$ equals 615, 1071 and 569 for dataset A, B and C respectively. Note that datasets A and B are binarized for experiments regarding binary variables while dataset C is used for experiments regarding count variables. A summary of the datasets are in Table 1.

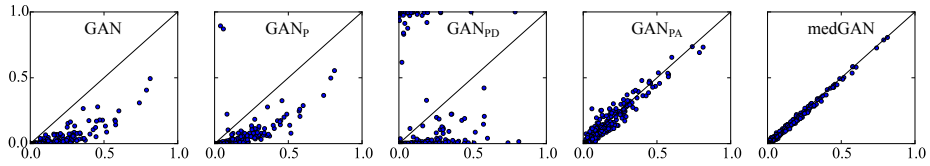
Models for comparison: To assess the effectiveness of our methods, we tested multiple versions of medGAN:

- **GAN:** We use the same architecture as medGAN with the standard training strategy, but do not pre-train the autoencoder.
- **GAN$_P$:** We pre-train the autoencoder (in addition to the GAN).
- **GAN$_{PD}$:** We pre-train the autoencoder and use minibatch discrimination (Salimans et al., 2016).
- **GAN$_{PA}$:** We pre-train the autoencoder and use minibatch averaging.
- **medGAN:** We pre-train the autoencoder and use minibatch averaging. We also use batch normalization and a shortcut connection for the generator G.

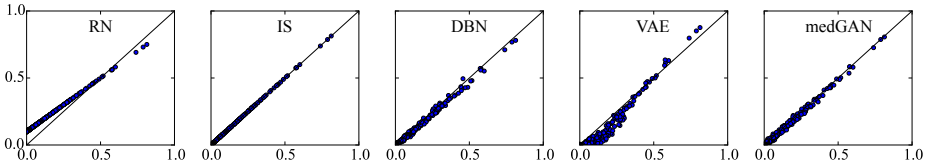
We also compare the performance of medGAN with several popular generative methods as below.

- **Random Noise (RN):** Given a real patient record $\mathbf{x}$, we invert the binary value of each code (i.e., dimension) with probability 0.1. This is not strictly a generative method, but rather it is a simple implementation of a privacy protection method based on randomization.
- **Independent Sampling (IS):** For the binary variable case, we calculate the Bernoulli success probability of each code in the real dataset, based on which we sample binary values to generate the synthetic dataset. For the count variable case, we use the kernel density estimator (KDE) for each code then sample from that distribution.

3. https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
4. http://www.wolterskluwercdi.com/drug-data/medi-span-electronic-drug-file/
5. https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ ccssvcproc.jsp



(a) Dimension-wise probability performance of various versions of `medGAN`.



(b) Dimension-wise probability performance of baseline models and `medGAN`.

Figure 2: Scatterplots of dimension-wise probability results. Each dot represents one of 615 codes. The x-axis represents the Bernoulli success probability for the real dataset A, and y-axis the probability for the synthetic counterpart generated by each model. The diagonal line indicates the ideal performance where the real and synthetic data show identical quality.

- **Stacked RBM (DBM):** We train a stacked Restricted Boltzmann Machines (Hinton and Salakhutdinov, 2006), then, using Gibbs sampling, we can generate synthetic binary samples. There are studies that extend RBMs beyond binary variables (Hinton and Salakhutdinov, 2009; Gehler et al., 2006; Tran et al., 2011). In this work, however, as our goal is to study `medGAN`'s performance in various aspects, we use the original RBM only.
- **Variational Autoencoder (VAE):** We train a variational autoencoder (Kingma and Welling, 2013) where the encoder and the decoder are constructed with feed-forward neural networks.

Implementation details: We implemented `medGAN` with TensorFlow 0.12 (Team, 2015). For training models, we used Adam (Kingma and Ba, 2014) with the learning rate set to 0.001, and a mini-batch of 1,000 patients on a machine equipped with Intel Xeon E5-2630, 256GB RAM, four Nvidia Pascal Titan X's and CUDA 8.0. The hyperparameter details are provided in Appendix B.

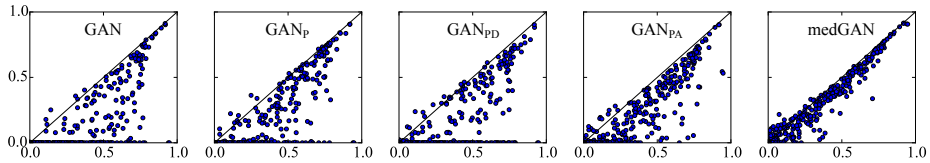
4.2 Quantitative Evaluation for Binary Variables

We evaluate the model performance for binary variables in this section, and provide the evaluation results of count variables in Appendix D. For all evaluations, we divide the dataset into a training set $R \in \{0,1\}^{N\times|\mathcal{C}|}$ and a test set $T \in \{0,1\}^{n\times|\mathcal{C}|}$ by 4:1 ratio. We use R to train the models, then generate synthetic samples $S \in \{0,1\}^{N\times|\mathcal{C}|}$ that are assessed in various tasks. For `medGAN` and VAE, we round the values of the generated dataset to the nearest integer values.

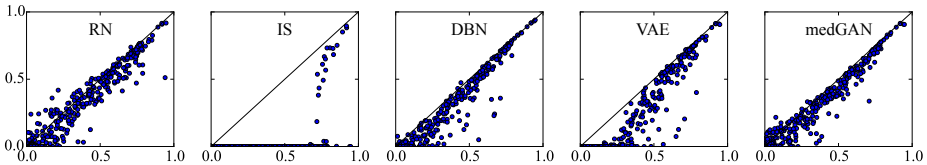
- **Dimension-wise probability:** This is a basic sanity check to confirm the model has learned each dimension's distribution correctly. We use the training set R to train the models, then generate the same number of synthetic samples S. Using R and S, we compare the Bernoulli success probability p_k of each dimension k.

- **Dimension-wise prediction:** This task indirectly measures how well the model captures the inter-dimensional relationships of the real samples. After training the models with R to generate S, we choose one dimension k to be the label $\mathbf{y}_{R_k} \in \{0,1\}^N$ and $\mathbf{y}_{S_k} \in \{0,1\}^N$. The remaining $R_{\setminus k} \in \{0,1\}^{N\times|\mathcal{C}|-1}$ and $S_{\setminus k} \in \{0,1\}^{N\times|\mathcal{C}|-1}$ are used as features to train two logistic regression classifiers LR_{R_k} and LR_{S_k} to predict $\mathbf{y}_{R_k}$ and $\mathbf{y}_{S_k}$, respectively. Then, we use the model LR_{R_k} and LR_{S_k} to predict label $\mathbf{y}_{T_k} \in \{0,1\}^n$ of the test set T. We can assume that the closer the performance of LR_{S_k} to that of LR_{R_k}, the better the quality of the synthetic dataset S. We use F1-score to measure the prediction performance, with the threshold set to 0.5.

To mitigate the repetition of results, we present our evaluation of dataset A in this section and direct the reader to Appendix C for the results from dataset B.



(a) Dimension-wise prediction performance of various versions of medGAN.



(b) Dimension-wise prediction performance of baseline models and medGAN.

Figure 3: Scatterplots of dimension-wise prediction results. Each dot represents one of 615 codes. The x-axis represents the F1-score of the logistic regression classifier trained on the real dataset A. The y-axis represents the F1-score of the classifier trained on the synthetic counterpart generated by each model. The diagonal line indicates the ideal performance where the real and synthetic data show identical quality.

4.2.1 Dimensions-wise probability

There are several notable findings that are worth highlighting. The dimension-wise probability performance increased as we used more advanced versions of medGAN, where the full medGAN shows the best performance as depicted by figure 2a. Note that minibatch averaging significantly increases the performance. Since minibatch averaging provides Bernoulli success probability information of real data to the model during training, it is natural that the generator learns to output synthetic data that follow a similar distribution. Minibatch discrimination does not seem to improve the results. This is most likely due to the discrete nature of the datasets. Improving the learning efficiency of the generator G with batch normalization and shortcut connection clearly helped improve the results.

Figure 2b compares the dimension-wise probability performance of baseline models with medGAN. Independent sampling (IS) naturally shows great performance as expected. DBM, given its stochastic binary nature, shows comparable performance as medGAN. VAE, although slightly inferior to DBM and medGAN, seems to capture the dimension-wise distribution relatively well, showing specific weakness at processing codes with low probability. Overall, we can see that medGAN clearly captures the independent distribution of each code.

4.2.2 Dimensions-wise prediction

Figure 3a shows the dimension-wise prediction performance of various versions of medGAN. The full medGAN again shows the best performance as it did in the dimension-wise probability task. Although the advanced versions of medGAN do not seem to dramatically increase the performance as they did for the previous task, this is due to the complex nature of inter-dimensional relationship compared to the independent dimension-wise probability. Figure 3b shows the dimension-wise prediction performance of baseline models compared to medGAN. As expected, IS is incapable of capturing the inter-dimensional relationship, given its naive sampling method. VAE shows similar behavior as it did in the previous task, showing weakness at predicting codes with low occurrence probability. Again, DBM shows comparable, if not slightly better performance to medGAN, which seems to come from its binary nature.

4.3 Qualitative Evaluation for Count Variables

We conducted a qualitative evaluation of medGAN with the help from a medical doctor. A discussion with the doctor taught us that count data are easier to assess its *realistic-ness* than binary data. Therefore we use dataset C to train medGAN and generate synthetic count samples. In this experiment, we randomly pick 50 records from real data and 50 records from synthetic data, randomly shuffle the order, present them to a medical doctor (specialized in internal medicine) who is asked to score how realistic each record is using scale 1 to 10 (10 being most realistic). Here the human doctor is served as the role of discriminator to provide the quality assessment of the synthetic data generated by medGAN.

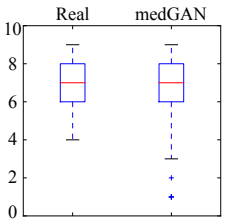


Figure 4: Boxplot of the impression scores from a medical expert.

The results of this assessment is shown in Figure 4. The findings suggest that medGAN's synthetic data are generally indistinguishable to a human doctor except for several outliers. In those cases, the fake records identified by the doctor either lacked appropriate medication codes, or had both male-related codes (*e.g.* prostate cancer) and female-related codes (*e.g.* menopausal disorders) in the same record. The former issue also existed in some of the real records due to missing data, but the latter issue demonstrates a current limitation in medGAN which could potentially be alleviated by domain specific heuristics. In addition to medGAN's impressive performance in statistical aspects, this medical review lends credibility to the qualitative aspect of medGAN.

4.4 Privacy Risk Evaluation

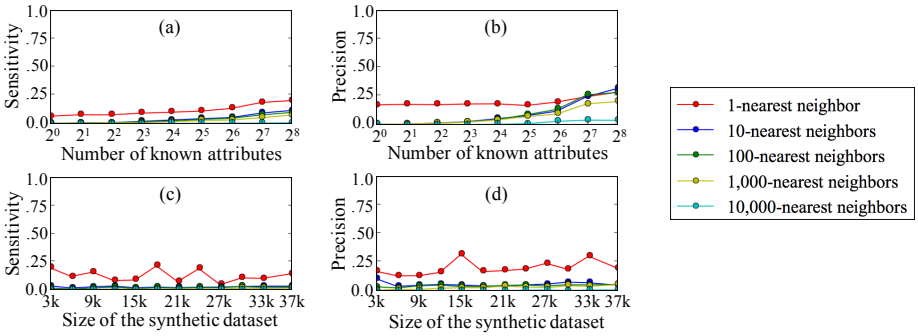


Figure 5: **a,b:** Sensitivity and precision when varying the number of known attributes. The total number of attributes (*i.e.* codes) of dataset B is 1,071. **c,d:** Sensitivity and precision when varying the size of the synthetic dataset. The maximum size of the synthetic dataset $S \in \{0,1\}^{N\times|\mathcal{C}|}$ is matched to the size of the training set $R \in \{0,1\}^{N\times|\mathcal{C}|}$.

We evaluate both presence and attribute disclosure using dataset B with binary variables. Due to the space constraint, we present the results of the attribute disclosure in the main paper and leave out the results of presence disclosure in Appendix F.

Experiment setup: We randomly sample 1% of the training set R as the compromised records, which is approximately 370 records. For each record r, we randomly choose s attributes as those which are known to the attacker. Next, the attacker performs k-nearest neighbor classifications to estimate the values of unknown attributes based on the synthetic records. More specifically, based on the known attributes, k-nearest neighbors in the synthetic dataset S are retrieved for each compromised record. Then, $|\mathcal{C}| - s$ unknown attributes are estimated based on the majority vote of the k nearest neighbors. Finally, for each unknown attribute, we calculate classification metrics in the form of precision and sensitivity. We repeat this process for all records of the 1% samples and obtain the mean precision and mean sensitivity. We vary the number of known attributes s and the number of neighbors k to study the attribute disclosure risk of medGAN. Note that the s attributes are randomly sampled across patients, so the attacker may know different s attributes for different patients.

Impact of attacker's knowledge: Figures 5a and 5b depict the sensitivity (i.e., recall) and the precision of the attribute disclosure test when varying the number of attributes known to the attacker. In this case, $x\%$ sensitivity means the attacker, using the known attributes of the compromised record and the synthetic data, can correctly estimate $x\%$ of the positive unknown attributes (i.e., attribute values are 1). Likewise, $x\%$ precision means the positive unknown attributes estimated by the attacker are on average $x\%$ accurate. Both figures show that an attacker who knows approximately 1% of the target patient's attributes (8 to 16 attributes) will estimate the target's unknown attributes with less than 10% sensitivity and 20% precision.

Impact of synthetic data size: Next, we fixed the number of known attributes to 16 and varied the number of records in the synthetic dataset S. Figures 5c and 5d show that the size of the synthetic dataset has little

influence on the effectiveness of the attack. In general, 1 nearest neighbor seems to be the most effective attack, although the sensitivity is still below 25% at best.

Overall, our privacy experiments indicate that medGAN does not simply remember the training samples and reproduce them. Rather, medGAN generates diverse synthetic samples that reveal little information to potential attackers unless they already possess significant amount of knowledge about the target patient.

5. Conclusion

In this work, we proposed medGAN, which uses generative adversarial framework to learn the distribution of real-world multi-label discrete electronic health records (EHR). Through rigorous evaluation using real datasets, medGAN showed impressive results for both binary variables and count variables. Considering the difficult accessibility of EHRs, we expect medGAN to make a contribution for healthcare research. We also provided empirical evaluation of privacy, which demonstrates very limited risks of medGAN in attribute disclosure. For future directions, we plan to explore the sequential version of medGAN, and also try to include other modalities such as lab measures, patient demographics, and free-text medical notes.

Acknowledgments

This work was supported by the National Science Foundation, award IIS-#1418511 and CCF-#1533768, Children's Healthcare of Atlanta, Google Faculty Award, UCB and Samsung Scholarship. Dr. Malin was supported by the National Science Foundation, award IIS-#1418504.

References

R. V. Atreya, J. C. Smith, A. B. McCoy, B. Malin, and R. A. Miller. Reducing patient re-identification risk for laboratory results within research datasets. *Journal of the American Medical Informatics Association*, 20(1): 95–101, 2013.

Anna Buczak, Steven Babin, and Linda Moniz. Data-driven approach for creating synthetic electronic medical records. *BMC Medical Informatics and Decision Making*, 10(1):59, 2010.

Emily Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pages 1486–1494, 2015.

Josep Domingo-Ferrer and Vicenç Torra. Disclosure risk assessment in statistical microdata protection via advanced record linkage. *Statistics and Computing*, 13(4):343–354, 2003. ISSN 1573-1375. doi: 10.1023/A:1025666923033. URL `http://dx.doi.org/10.1023/A:1025666923033`.

J Dreschsler. *Synthetic datasets for statistical disclosure control*. Springer Press, 2011.

K. El Emam, D. Buckeridge, R. Tamblyn, A. Neisa, E. Jonker, and A. Verma. The re-identification risk of canadians from longitudinal demographics. *BMC Medical Informatics and Decision Making*, 11:46, 2011a.

K. El Emam, E. Jonker, L. Arbuckle, and B. Malin. A systematic review of re-identification attacks on health data. *PLoS ONE*, 6(12):e28071, 2011b.

K. El Emam, S. Rodgers, and B. Malin. Anonymising and sharing individual patient data. *British Medical Journal*, 350:h1139, 2015.

Y. Erlich and A. Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014.

Office for Civil Rights. *Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule*. U.S. Department of Health and Human Services, 2013.

Peter V Gehler, Alex D Holub, and Max Welling. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of the 23rd international conference on Machine learning*, pages 337–344. ACM, 2006.

John Glover. Modeling documents with generative adversarial networks. *arXiv:1612.09122*, 2016.

Ary Goldberger et al. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 2000.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Lawrence Gostin, Laura Levit, Sharyl Nass, et al. *Beyond the HIPAA Privacy Rule: Enhancing Privacy, Improving Health Through Research*. National Academies Press, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614, 2009.

James Hodge Jr, Lawrence O Gostin, and Peter Jacobson. Legal issues concerning electronic health information: privacy, quality, and liability. *Jama*, 282(15):1466–1471, 1999.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv:1611.01144*, 2016.

Alistair Johnson et al. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Diederik Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv:1611.04051*, 2016.

Joseph S Lombardo and Linda J Moniz. Ta method for generation and distribution. *Johns Hopkins APL Technical Digest*, 27(4):356, 2008.

G. Loukides, J. C. Denny, and B. Malin. The disclosure of diagnosis codes can breach research participants' privacy. *J Am Med Inform Assoc*, 17(3):322–327, 2010.

Chris Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*, 2016.

B. Malin and L. Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):179–192, 2004.

Stan Matwin, Jordi Nin, Morvarid Sehatkar, and Tomasz Szapiro. *A Review of Attribute Disclosure Control*, pages 41–61. Springer International Publishing, Cham, 2015. ISBN 978-3-319-09885-2. doi: 10.1007/978-3-319-09885-2_4. URL http://dx.doi.org/10.1007/978-3-319-09885-2_4.

Scott McLachlan, Kudakwashe Dube, and Thomas Gallagher. Using the caremap with health incidents statistics for generating the realistic synthetic electronic healthcare record. In *Healthcare Informatics (ICHI), 2016 IEEE International Conference on*, pages 439–448. IEEE, 2016.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv:1611.02163*, 2016.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

Mehmet Nergiz and Chris Clifton. δ-presence with complete world knowledge. *IEEE Transactions on Knowledge Engineering*, 22:868–883, 2010.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv:1610.09585*, 2016.

Yubin Park, Joydeep Ghosh, and Mallikarjun Shankar. Perturbed gibbs samplers for generating large-scale privacy-safe synthetic health data. In *Healthcare Informatics (ICHI), 2013 IEEE International Conference on*, pages 493–498. IEEE, 2013.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

J. Reiter. Satisfying disclosure restrictions with synthetic datasets. *Journal of Official Statistics*, 18(4):531–543, 2002.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pages 2226–2234, 2016.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Security & Privacy Conference*, page in press, 2017.

L. Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine, and Ethics*, 25(2-3):98–110, 1997.

TensorFlow Team. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Truyen Tran, Dinh Phung, and Svetha Venkatesh. Mixed-variate restricted boltzmann machines. In *Asian Conference on Machine Learning*, pages 213–229, 2011.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NIPS*, pages 4790–4798, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016b.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv:1609.05473*, 2016.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. *NIPS Workshop on Adversarial Training*, 2016.

Algorithm 1 `medGAN` Optimization

$\theta_d, \theta_g, \theta_{enc}, \theta_{dec} \leftarrow$ Initialize with random values.
repeat // Pre-train the autoencoder
  Randomly sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ from $\mathbf{X}$
  Update $\theta_{enc}, \theta_{dec}$ by minimizing Eq.(2) (or Eq.(3))
until convergence or fixed iterations
repeat
  for k steps **do** // Update the discriminator.
    Randomly sample $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m$ from $p_{\mathbf{z}}$
    Randomly sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ from $\mathbf{X}$
    $\mathbf{x}_{\mathbf{z}_i} \leftarrow Dec(G(\mathbf{z}_i))$
    $\bar{\mathbf{x}}_{\mathbf{z}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_{\mathbf{z}_i}$
    $\bar{\mathbf{x}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$
    Ascend θ_d by the gradient:
      $\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m} \log D(\mathbf{x}_i, \bar{\mathbf{x}}) + \log(1 - D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}}))$
  end for
  // Update the generator and the decoder.
  Randomly sample $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m$ from $p_{\mathbf{z}}$
  $\mathbf{x}_{\mathbf{z}_i} \leftarrow Dec(G(\mathbf{z}_i))$
  $\bar{\mathbf{x}}_{\mathbf{z}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_{\mathbf{z}_i}$
  Ascend θ_g, θ_{dec} by the gradient:
    $\nabla_{\theta_{g,dec}} \frac{1}{m}\sum_{i=1}^{m} \log D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}})$
until convergence or fixed iterations

Appendices

Appendix A. `medGAN` training algorithm

Algorithm 1 describes the overall optimization process of `medGAN`. Note that θ_d is updated k times per iteration, while θ_g and θ_{dec} are updated once per iteration to ensure optimality of D. However, typically, a larger k has not shown a clear improvement (Goodfellow, 2016). And we set $k = 2$ in our experiments.

Appendix B. Hyperparameter details

We describe the architecture and the hyper-parameter values used for each model. We tested all models by varying the number of hidden layers (while matching the number of parameters used for generating synthetic data), the size of the minibatch, the learning rate, the number of training epochs, and we report the best performing configuration for each model.

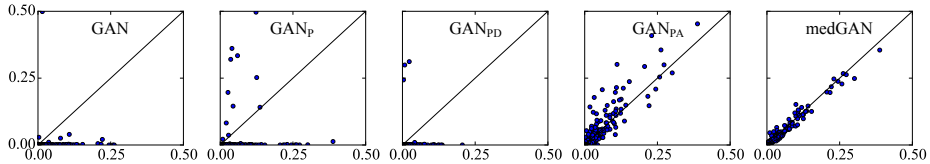
- **`medGAN`**: Both the encoder Enc and the decoder Dec are single layer feedforward networks, where the original input $\mathbf{x}$ is compressed to a 128 dimensional vector. The generator G is implemented as a feedforward network with two hidden layers, each having 128 dimensions. For the batch normalization in the generator G, we use both the scale parameter γ and the shift parameter β, and set the moving average decay to 0.99. The discriminator D is also a feedforward network with two hidden layers where the first layer has 256 dimensions and the second layer has 128 dimensions. `medGAN` is trained for 1,000 epochs with the minibatch of 1,000 records.

- **DBM**: In order to match the number of parameters used for data generation in `medGAN` (G + Dec), we used four layers of Restricted Boltzmann Machines where the first layer is the input layer. All hidden layers used 128 dimensions. We performed layer-wise greedy persistent contrastive divergence (20-step

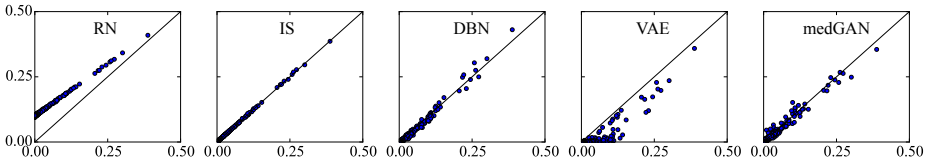
Gibbs sampling) to train DBM. We used 0.01 for learning rate and 100 samples per minibatch. All layers were separately trained for 100 epochs. Synthetic samples were generated by performing Gibbs sampling at the two two layers then propagating the values down to the input layer. We ran Gibbs sampling for 1000 iterations per sample. Using three stacks showed small performance degradation.

- **VAE**: In order to match the number of parameters used for data generation in medGAN ($G + Dec$), both the encoder and the decoder were implemented with feedforward networks, each having 3 hidden layers. The encoder accepts the input $\mathbf{x}$ and compresses it to a 128 dimensional vector and the decoder reconstructs it to the original dimension space. VAE was trained with Adam for 1,000 iterations with the minibatch of 1,000 records. Using two hidden layers for the encoder and the decoder showed similar performance.

Appendix C. Quantitative evaluation results for binary dataset B

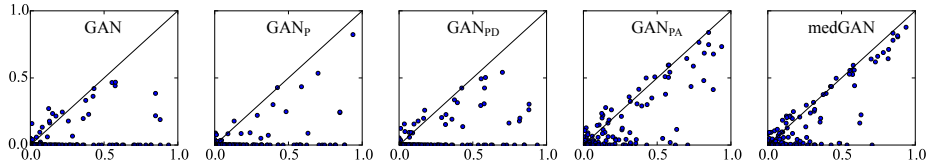


(a) Dimension-wise probability performance of various versions of medGAN.

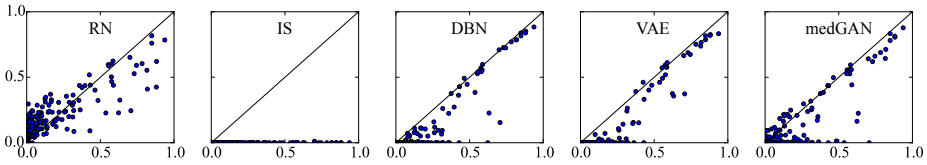


(b) Dimension-wise probability performance of baseline models and medGAN.

Figure 6: Scatterplots of dimension-wise probability results. Each dot represents one of 1,071 codes. The x-axis represents the Bernoulli success probability for the real dataset B, and y-axis the probability for the synthetic counterpart generated by each model. The diagonal line indicates the ideal performance where the real and synthetic data show identical quality.



(a) Dimension-wise prediction performance of various versions of medGAN.



(b) Dimension-wise prediction performance of baseline models and medGAN.

Figure 7: Scatterplots of dimension-wise prediction results. Each dot represents one of 1,071 codes. The x-axis represents the F1-score of the logistic regression classifier trained on the real dataset B. The y-axis represents the F1-score of the classifier trained on the synthetic counterpart generated by each model. The diagonal line indicates the ideal performance where the real and synthetic data show identical quality.

C.1 Dimension-wise probability

Figure 6a shows the consistent superiority of the full version of medGAN compared other versions. The effect of minibatch averaging is even more dramatic for dataset B. Figure 6b shows that VAE has some difficulty capturing the dimension-wise distribution of dataset B. Again, DBM shows comparable performance to medGAN, slightly outperforming medGAN for low-probability codes, but slightly underperforming for high-probability codes. Overall, dimension-wise probability performance is somewhat weaker for dataset B than for dataset A, most likely due to smaller data volume and sparser code distribution.

C.2 Dimension-wise prediction

Figure 7a shows the dimension-wise predictive performance for different versions of medGAN where the full version outperforms others. Figure 7b shows similar pattern as Figure 3b. Independent sampling completely fails to make any meaningful prediction. VAE demonstrates weakness at predicting low-probability codes. DBM seems to slightly outperform medGAN, especially for highly predictable codes. Again, due to the nature of the dataset, all models show weaker predictive performance for dataset B than they did for dataset A.

Appendix D. Quantitative results for count variables

In order to evaluate for count variables, we use dataset C, consisting of 30,738 patients whose records were taken for exactly 18 months. The same subset was used to perform qualitative evaluation in section 4.3. The details of constructing dataset C for heart failure studies are described in Appendix E. Note that each patient's number of hospital visits within the 18 months period can vary, which is a perfect test case for count variables. Again, we aggregate the dataset into a fixed-size vector and divide it into the training set $R \in \mathbb{Z}_+^{N \times |\mathcal{C}|}$ and the test set $T \in \mathbb{Z}_+^{n \times |\mathcal{C}|}$ in 4:1 ratio. Since we have confirmed the superior performance of full medGAN compared to other versions of GANs in binary variables evaluation, we focus on the comparison with baseline models in this section. Note that, to generate count variables, we replaced all activation functions in both VAE and medGAN (except the discriminator's output) to ReLU. We also use kernel density estimator with Gaussian kernel (bandwidth=0.75) to perform the independent sampling (IS) baseline. We no longer test random noise (RN) method in this section as it is difficult to determine how much noise should be injected to count variables to keep them sufficiently realistic but different enough from the training set.

For count variables, we conduct similar quantitative evaluations as binary variables with slight modifications. We first calculate dimension-wise average count instead of dimension-wise probability. For dimension-wise prediction, we use the binary labels $\mathbf{y}_{R_k} \in \{0,1\}^N$ and $\mathbf{y}_{S_k} \in \{0,1\}^N$ as before, but we train the logistic regression classifier with count samples $R_{\setminus k} \in \mathbb{Z}_+^{N \times |\mathcal{C}|-1}$ and $S_{\setminus k} \in \mathbb{Z}_+^{N \times |\mathcal{C}|-1}$. The classifiers use count features as oppose to binary features while the evaluation metric is still F1-score.

D.0.1 Dimensions-wise average count

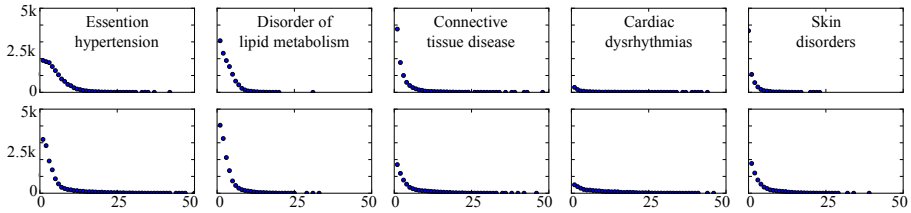


Figure 8: Histogram of counts of five most frequent codes from dataset C. The top row was plotted using the training dataset, the bottom row using medGAN's synthetic dataset.

Figure 9 shows the performance of baseline models and medGAN. The discontinuous behavior of VAE is due to its extremely low-variance synthetic samples. We found that, on average, VAE's synthetic samples had

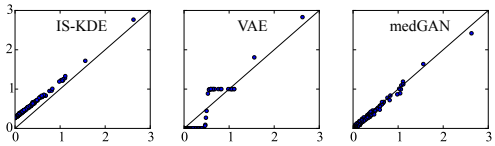


Figure 9: Scatterplot of dimension-wise average count of the training dataset (x-axis) versus the synthetic counterpart (y-axis).

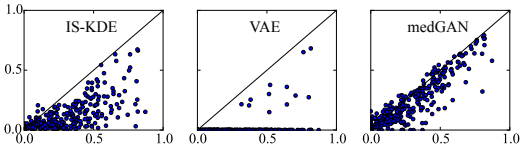


Figure 10: Scatterplot of dimension-wise prediction F1-score of logistic regression trained on the training dataset (x-axis) versus the classifier trained on the synthetic counterpart (y-axis).

nine orders of magnitude smaller standard deviation than medGAN's synthetic samples. medGAN, on the other hand, shows good performance with just a simple substitution of the activation functions.

Figure 8 shows the count histograms of five most frequent codes from the count dataset, where the top row was plotted with the training dataset and the bottom row with medGAN's synthetic dataset. We can see that medGAN's synthetic counterpart has very similar distribution as the real data. This tells us that medGAN is not just trying to match the average count of codes (*i.e.* binomial distribution mean), but learns the actual distribution of the data.

D.0.2 Dimensions-wise prediction

Figure 10 shows the performance of baseline models and medGAN. We can clearly see that medGAN shows superior performance. The experiments on count variables is especially interesting, as medGAN seems to make a smooth transition from binary variables to count variables, with just a replacement of the activation function. We also speculate that the medGAN's dimension-wise prediction performance will increase with more training data, as the count dataset used in this section consists of only 30,738 samples.

Appendix E. Dataset construction for heart failure studies

Case patients were 40 to 85 years of age at the time of HF diagnosis. HF diagnosis (HFDx) is defined as: 1) Qualifying ICD-9 codes for HF appeared in the encounter records or medication orders. Qualifying ICD-9 codes are displayed in Table 2. 2) a minimum of three clinical encounters with qualifying ICD-9 codes had to occur within 12 months of each other, where the date of diagnosis was assigned to the earliest of the three dates. If the time span between the first and second appearances of the HF diagnostic code was greater than 12 months, the date of the second encounter was used as the first qualifying encounter. The date at which HF diagnosis was given to the case is denoted as HFDx. Up to ten eligible controls (in terms of sex, age, location) were selected for each case, yielding an overall ratio of 9 controls per case. Each control was also assigned an index date, which is the HFDx of the matched case. Controls are selected such that they did not meet the operational criteria for HF diagnosis prior to the HFDx plus 182 days of their corresponding case. Control subjects were required to have their first office encounter within one year of the matching HF case patients first office visit, and have at least one office encounter 30 days before or any time after the cases HF diagnosis date to ensure similar duration of observations among cases and controls.

Appendix F. Presence disclosure

We performed a series of experiments to assess the extent to which medGAN leaks the presence of a patient. To do so, we randomly sample r patient records from each of the training set $R \in \{0,1\}^{N \times |C|}$ and the test set

Table 2: Qualifying ICD-9 codes for heart failure

ICD-9 Code	Description
398.91	Rheumatic heart failure (congestive)
402.01	Malignant hypertensive heart disease with heart failure
402.11	Benign hypertensive heart disease with heart failure
402.91	Unspecified hypertensive heart disease with heart failure
404.01	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.03	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage V or end stage renal disease
404.11	Hypertensive heart and chronic kidney disease, benign, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.13	Hypertensive heart and chronic kidney disease, benign, with heart failure and chronic kidney disease stage V or end stage renal disease
404.91	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.93	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and chronic kidney disease stage V or end stage renal disease
428.0	Congestive heart failure, unspecified
428.1	Left heart failure
428.20	Systolic heart failure, unspecified
428.21	Acute systolic heart failure
428.22	Chronic systolic heart failure
428.23	Acute on chronic systolic heart failure
428.30	Diastolic heart failure, unspecified
428.31	Acute diastolic heart failure
428.32	Chronic diastolic heart failure
428.33	Acute on chronic diastolic heart failure
428.40	Combined systolic and diastolic heart failure, unspecified
428.41	Acute combined systolic and diastolic heart failure
428.42	Chronic combined systolic and diastolic heart failure
428.43	Acute on chronic combined systolic and diastolic heart failure
428.9	Heart failure, unspecified

$T \in \{0,1\}^{n \times |\mathcal{C}|}$. We assume the attacker has complete knowledge on those $2r$ records. Then for each record, we calculate its hamming distance to each sample from the synthetic dataset $S \in \{0,1\}^{N \times |\mathcal{C}|}$. If there is at least one synthetic sample within a certain distance, we treat that as its claimed match. Now, since we sample from both R and T, the match could be a true positive (i.e., attacker correctly claims their targeted record is in the GAN training set), false positive (i.e., attacker incorrectly claims their targeted record is in the GAN training set), true negative (i.e., attacker correctly claims their targeted record is not in the GAN training set), or false negative (i.e., attacker incorrectly claims their targeted record is not in the GAN training set).

We varied the number of patients r and the hamming distance threshold and calculated the sensitivity and precision.

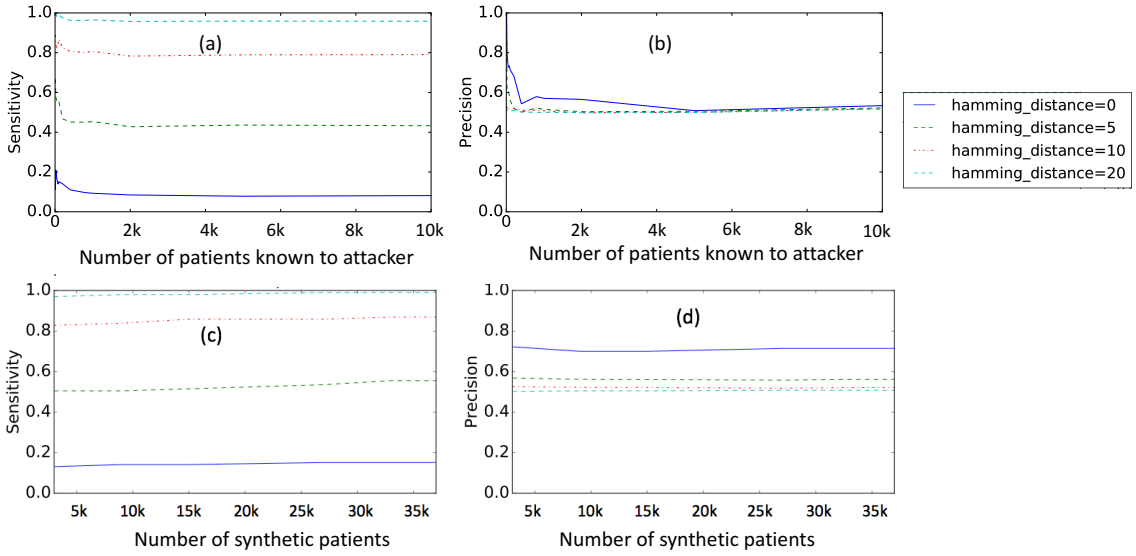


Figure 11: **a,b:** Sensitivity and precision while varying the number of patients known to the attacker. **c,d:** Sensitivity and precision while varying the number of synthetic patients.

Impact of attacker's knowledge: Figures 11a and 11b depict the sensitivity (*i.e.* recall) and the precision of the presence disclosure test when varying the number of real patient the attacker knows. In this setting, $x\%$ sensitivity means the attacker has successfully discovered that $x\%$ of the records that he/she already knows were used to train `medGAN`. Similarly, $x\%$ precision means, when an attacker claims that a certain number of patients were used for training `medGAN`, only $x\%$ of them were actually used. Figure 11a shows that with low threshold of hamming distance (e.g. hamming distance of 0) attacker can only discover 10% percent of the known patients to attacker were used to train `medGAN`. Figure 11b shows that, the precision is mostly 50% except when the number of known patients are small. This indicates that the attacker's knowledge is basically useless for presence disclosure attack unless the attacker is focusing on a small number of patients (less than a hundred), in which case the precision is approximately 80%.

We conducted an additional experiment to evaluate the impact of the size of the synthetic data on presence disclosure risk. In this experiment, we fix the number of known real patients to 100 and varied the number of records in the synthetic dataset S. Figures 11c and 11d show that the size of the generated synthetic dataset has almost no impact on presence disclosure.