Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks

Edward Choi[1], Siddharth Biswal[1], Bradley Malin[2], Jon Duke[1], Walter F. Stewart[3], Jimeng Sun[1]
[1]Georgia Institute of Technology [2]Vanderbilt University [3]Sutter Health
{mp2893,sbiswal7}@gatech.edu, bradley.malin@vanderbilt.edu,
jon.duke@gatech.edu, stewarwf@sutterhealth.org, jsun@cc.gatech.edu

Abstract

Access to electronic health records (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 EHRs. Based on an input EHR dataset, `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 EHR datasets that achieve comparable performance to real data on many experiments including distribution statistics, predictive modeling tasks and medical expert review.

1 Introduction

The adoption of electronic healthcare records (EHR) by healthcare providers, and 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 of the reasons for limited access is because EHR data are composed of personally identifiable information, which in combination with potentially sensitive medical data, induces privacy concerns. As a result, access to such data for secondary purposes (e.g., research) is highly regulated, as well as controlled by provider 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).

Healthcare institutions often aim to mitigate privacy risks through the practice of de-identification (for Civil Rights, 2013). Typically, de-identification is accomplished 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 to privacy protection is not impregnable to re-identification attack (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 approach in practice has been challenging because the resulting synthetic data are often not sufficiently realistic for machine learning tasks. Since many machine learning models on EHR data are using aggregated discrete features derived from longitudinal EHR records, we concentrate our effort on generating such aggregated data in this work. 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 demonstrated impressive performance in generating high-quality synthetic images (Goodfellow et al., 2014; Radford et al., 2015; Goodfellow, 2016). A GAN consists of two components: a *generator* that attempts to generate realistic, but fake, images and a *discriminator* that aims to distinguish between the generated fake images and the real images. 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., 2016a,b) on the quality of images (i.e., fake compared to real) and on processing speed (Goodfellow, 2016). However, a GAN cannot learn the distribution of discrete variables in its original form.

On that premise, we propose `medGAN`, a neural network model that generates high-dimensional discrete variables representing events documented in patients' EHRs (e.g., diagnosis of a certain disease or treatment of a certain medication). Using an 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 is applied to overcome the original GAN's inability to generate discrete samples. The specific contributions of this work are as follows:

- We propose `medGAN`, an efficient algorithm to generate high-dimensional multi-hot discrete samples by combining an autoencoder with GAN. In particular, `medGAN` can handle both binary and count variables.
- `medGAN` translates input EHR data to a program that can generate arbitrarily large volume of high-quality, high-dimensional synthetic patient data.
- We propose a simple, yet effective, method called *minibatch averaging* to cope with the situation where GAN learns to generate samples of low diversity, the "mode collapse" problem, which outperforms previous methods such as *minibatch discrimination.*
- We demonstrate the close-to-real-data performance of `medGAN` using real EHR datasets on diverse tasks including distribution statistics, classification performance and medical expert review.

2 Related work

We first discuss existing methods to generate synthetic EHR data, followed by recent advances in generative adversarial networks (GAN) and specific works on generating discrete variables using GAN.

Synthetic Data Generation for Health Data: De-identification of EHR data is the prominent technical method for protecting patient privacy when sharing EHR data for research (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 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 EHR to explore questions related to the outbreak of specific illnesses, where care patterns in the source EHR 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 only on a small, low-dimensional dataset. By contrast, our proposed `medGAN` directly captures general EHR data without focusing on specific diseases, which makes it suitable for diverse applications.

GAN and its Applications: Attempts to advance GAN (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;

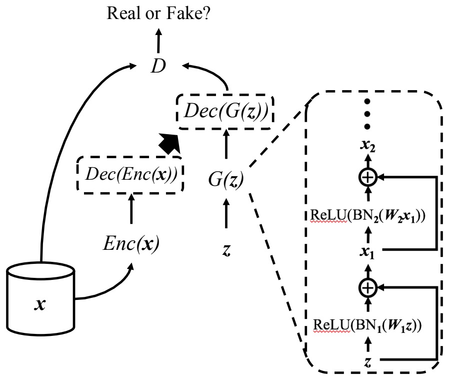


Figure 1: Architecture of medGAN: The discrete $\mathbf{x}$ comes from the source EHR data, $\mathbf{z}$ is the random prior for the genrator G; G is a feedforward network with shortcut connections (righthand side figure); An autoencoder (i.e, the encoder Enc and decoder Dec) is learned from $\mathbf{x}$; 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 $Dec(G(\mathbf{z}))$.

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 GAN.

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 unbiased estimation of 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. 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 GAN, but did not attempt to generate actual documents.

To handle high-dimensional multi-label discrete variables, we propose medGAN to efficiently generate discrete samples by generating the distributed representations of patient records with GAN, then decoding them into actual discrete records with an autoencoder.

3 Method

We first describe the structure of EHR data and the related mathematical notations, followed by details on medGAN.

3.1 Description of EHR Data and Notations

We assume there are $|\mathcal{C}|$ number of 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 record 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. Note 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 codes tend to occur much more frequently (*e.g.*, essential hypertension) than others. This is problematic because it can skew a distribution among different dimensions.

3.2 Preliminary: Generative Adversarial Network

In a GAN, the generator $G(\mathbf{z};\theta_g)$, accepts the 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 $V(G,D)$:

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

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,

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

In the remainder of this paper we use Eq.(1) as it showed significantly more stable performance. 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 cannot directly learn the distribution of discrete patient records $\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}|}$. We overcome 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, like image processing (Goodfellow et al., 2016; Vincent et al., 2008). We apply the autoencoder to learn the salient features of discrete variables that can be used 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 $Dec(Enc(\mathbf{x});\theta_{dec})$ 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:

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

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

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 autoencoder, we can allow GAN to generate distributed representation of patient records (i.e., the output of the encoder *Enc*), rather than generating discrete records directly. As the generator G and the encoder *Enc* both generate similar continuous values, the decoder *Dec* can pick up the right signals to convert synthetic continuous samples $G(\mathbf{z}) \in \mathbb{R}^h$ to discrete samples $Dec(G(\mathbf{z})) \in \mathbb{Z}_+^{|\mathcal{C}|}$. 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,

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

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

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

Note that 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 such as minibatch discrimination and unrolled GANs, but they require some fine-tuning of the hyperparameters, or scalability has not been addressed (Salimans et al., 2016; Metz et al., 2016).

`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; allow 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 sample and all samples 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:

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

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

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

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

[1] We considered the denoising autoencoder (dAE) (Vincent et al., 2008) but there was no visible performance improvement.

[2] Note that we also used 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 if 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 $Dec(G(\mathbf{z}_1, \mathbf{z}_2, \ldots))$. For EHR data generation, the binary variables can be applied to present whether diseases or treatments are present or not.
Count variables: Count variables are a more accurate representation of clinical events they can indicate the number of times a diagnosis was made or a certain medication was prescribed over multiple hospital visits. In this case of 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. Minibatch averaging works surprisingly well and does not require additional parameters as minibatch discrimination, therefore has minimal impact to the training time. It is worth mentioning that, for both binary and count variables, a bigger minibatch than usual is recommended to properly capture the statistics of the real data. We use 1,000 records for a minibatch in this work.

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:

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

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 righthand side of Figure 1 depicts the first two layers of G. Note that we do not incporate 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.

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.

4 Experiments

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

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 $\theta_d$ 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 $\theta_g, \theta_{dec}$ 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
```

Table 1: Basic statistics of Sutter PAMF and MIMIC-III

Dataset	**Sutter PAMF**	**MIMIC-III**
# of patients	258,559	46,520
# of unique codes	615	1071
Avg. # of codes per patient	38.37	11.27
Max # of codes for a patient	198	90
Min # of codes for a patient	1	1

4.1 Experimental Setup

Source data: The datasets in this study were from: 1) the Sutter Palo Alto Medical Foundation (PAMF), which consists of 10-years of longitudinal medical records of 258K patients with age 50 to 90 and 2) 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. From the PAMF dataset, we extracted diagnoses, medications and and procedure codes, which were then respectively grouped by Clinical Classifications Software for ICD-9[3], Generic Product Identifier Drug Group[4] and Clinical Classifications Software for CPT[5] into a total of 615 codes. From the MIMIC-III, we extracted diagnosis codes and grouped them by generalizing up to their 3-digit ICD9 codes, yielding a total of 1071 codes. A summary of the datasets are in Table 1. 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 and 1071 for PAMF and MIMIC-III, respectively.

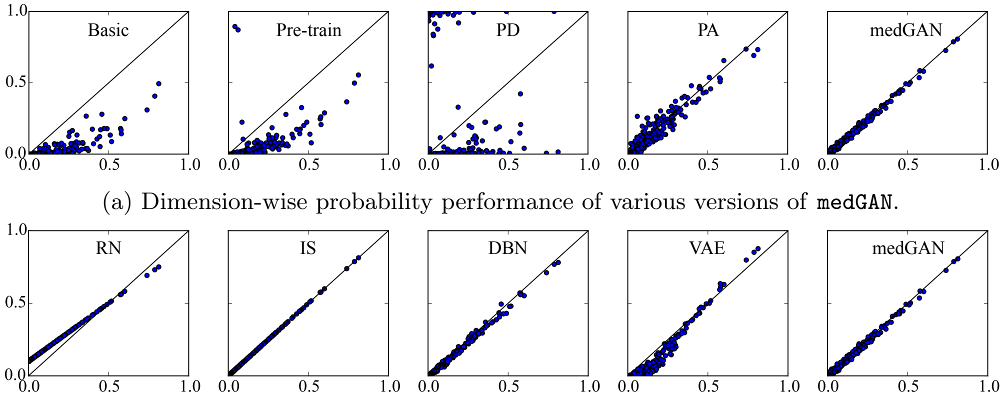
Models for comparison: To assess the effectiveness of our methods, we tested multiple versions of `medGAN`:

- **Basic:** We use the same architecture as `medGAN` with the standard training strategy, but do not pre-train the autoencoder.

[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 PAMF data, 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.

- **Pre-train:** We pre-train autoencoder in addition to GAN.
- **PD:** We pre-train the autoencoder and use minibatch discrimination (Salimans et al., 2016).
- **PA:** We pre-train the autoencoder and use minibatch averaging.
- Full `medGAN`**:** We pre-train the autoencoder and use minibatch averaging. We also use batch normalization and shortcut connection for the generator G.

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

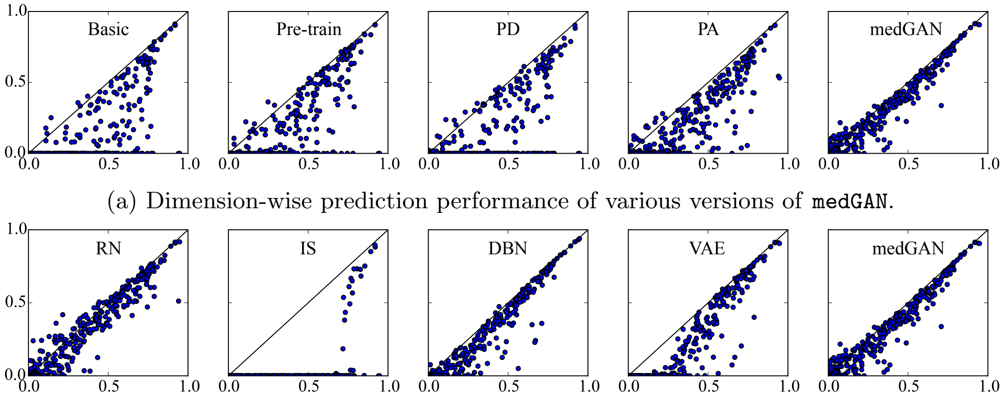
- **Random Noise (RN):** Given a real patient record $\mathbf{x}$, we invert the binary value of each code (*i.e.*, dimension) with probability 0.1. This is not strictly a generative method but a simple implementation of a privacy protection method based on randomization.
- **Independent Sampling (IS):** 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.
- **Stacked RBM (DBN):** We train a stacked Restricted Boltzmann Machines (Hinton and Salakhutdinov, 2006) and generate synthetic samples using Gibbs sampling. This was used as a baseline for binary variables only, as it is a binary network.
- **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 a mini-batch of 100 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 .1.

4.2 Quantitative Evaluation for Binary Variables

We evaluate the model performance for binary variables first, then proceed to the more challenging count variables. 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 $S \in \{0,1\}^{N \times |\mathcal{C}|}$ to use it for various tasks. For `medGAN` and VAE, we round the continuous values to the nearest integer values.

- **Dimension-wise probability:** 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



(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 Sutter PAMF. 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.

of synthetic samples S. Using the R and S, we compare the Bernoulli success probability p_k of each dimension k.

- **Dimension-wise prediction:** A task to indirectly measure how well the model captures the inter-dimensional relationship 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 $R_{\backslash k} \in \{0,1\}^{N \times |\mathcal{C}|-1}$ and $S_{\backslash k} \in \{0,1\}^{N \times |\mathcal{C}|-1}$ are used as features to train two logistic regression classifiers LR_{R_k} and LR_{S_k} to predict respectively $\mathbf{y}_{R_k}$ and $\mathbf{y}_{S_k}$. 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 relieve the reader of repetitive experiment results, we present evaluation results using Sutter PAMF in this section and provide the results from MIMIC-III in Appendix .3.

4.2.1 Dimensions-wise probability

The dimension-wise probability performance increased as we used more advanced version 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 generate synthetic data that follow the similar distribution. Minibatch discrimination does not seem to improve the results, 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 improving the results on both datasets.

Figure 2b compares the dimension-wise probability performance of baseline models with `medGAN`. Independent sampling (IS) naturally shows great performance as expected. DBN, given its stochastic binary nature, shows comparable performance as `medGAN`. VAE, although slightly inferior to DBN 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` can clearly capture 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, DBN shows comparable, if not slightly better performance to medGAN, which seems to come from its binary nature.

4.3 Quantitative Evaluation for Count Variables

In order to evaluate for count variables, we use Sutter heart failure (HF) dataset, which is a subset of Sutter PAMF, consisting of 30,738 patients whose records were taken for exactly 18 months. Note that each patient's total hospital visits within the 18 months period can vary, which is a perfect test case for count variables. Detailed process of constructing Sutter HF dataset is provided in Appendix .2. Again, we aggregate the dataset into a fixed-size vector and divide it into $R \in \mathbb{Z}_+^{N\times|\mathcal{C}|}$ and $T \in \mathbb{Z}_+^{n\times|\mathcal{C}|}$ in 4:1 ratio, Since we have confirmed the superior performance of full medGAN for binary variables, 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.

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.

4.3.1 Dimensions-wise average count

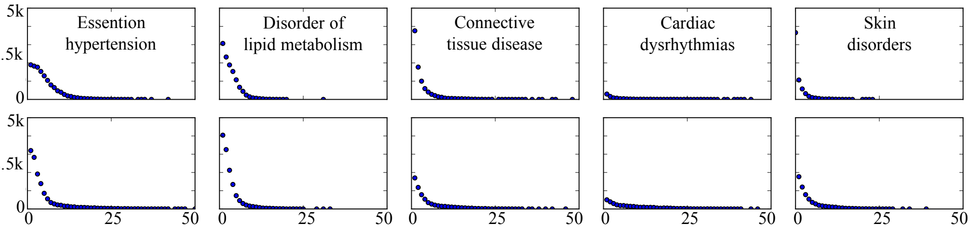


Figure 4: Histogram of counts of five most frequent codes from the Sutter heart failure dataset. The top row was plotted using the real data, the bottom row using medGAN's synthetic data.

Figure 5 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 nine orders of magnitude smaller standard deviation than medGAN's synthetic samples. medGAN, on the other hand, shows good performance with the simple substitution of activation functions.

Figure 4 shows the count histograms of five most frequent codes from Sutter HF dataset, where the top row was plotted with the real data and the bottom row with medGAN's synthetic data. We can see that medGAN's synthetic counterpart has very similar distribution as the real data. This tells us that medGAN is

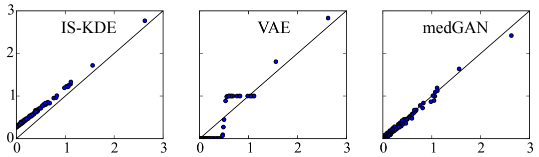


Figure 5: Scatterplot of dimension-wise average count of real Sutter heart failure dataset (x-axis) versus the synthetic counterpart (y-axis).

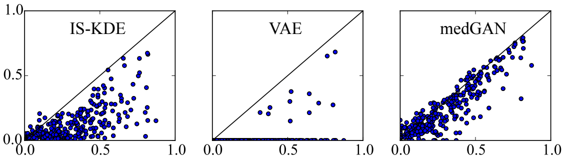


Figure 6: Scatterplot of dimension-wise prediction F1-score of logistic regression trained on real Sutter heart failure dataset (x-axis) versus the classifier trained on the synthetic counterpart (y-axis).

not just trying to match the average count of codes (*i.e.* binomial distribution mean), but learns the actual distribution of the data.

4.3.2 Dimensions-wise prediction

Figure 6 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 Sutter HF dataset consists of only 30K samples.

4.4 Qualitative Evaluation

To conduct a qualitative evaluation of medGAN, we use Sutter HF dataset to train medGAN and generate synthetic samples with count variables. We then randomly pick 50 records from real data and 50 records from synthetic data, 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.

The results of this assessment is shown in Figure 7. 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.

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 two datasets, medGAN showed impressive results for both binary variables and count variables. Considering the

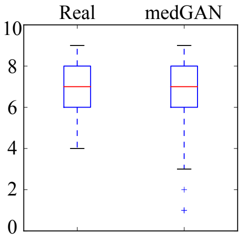


Figure 7: Boxplot of the impression scores from a medical expert. The synthetic data from medGAN provides comparable impression scores in terms of realisticness level.

difficulty accessibility of EHRs, we expect medGAN to make a contribution for healthcare research. As we have shown the versatility of GAN framework, we also expect this will encourage other researchers to apply GAN framework on various datasets and tasks. 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.

References

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

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

Emily Denton, Soumith Chintala, Rob Fergus, and others. 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *NIPS*. 1486–1494.

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

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

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

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

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

Office for Civil Rights. 2013. *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.

John Glover. 2016. Modeling documents with Generative Adversarial Networks. *arXiv:1612.09122* (2016).

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

Ian Goodfellow. 2016. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160* (2016).

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

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

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

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

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

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

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144* (2016).

Alistair Johnson and others. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3 (2016).

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

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

Matt J Kusner and José Miguel Hernández-Lobato. 2016. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv:1611.04051* (2016).

Joseph S Lombardo and Linda J Moniz. 2008. TA Method for Generation and Distribution. *Johns Hopkins APL Technical Digest* 27, 4 (2008), 356.

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

Chris Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv:1611.00712* (2016).

B. Malin and L. Sweeney. 2004. 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 (2004), 179–192.

Scott McLachlan, Kudakwashe Dube, and Thomas Gallagher. 2016. Using the CareMap with Health Incidents Statistics for Generating the Realistic Synthetic Electronic Healthcare Record. In *Healthcare Informatics (ICHI), 2016 IEEE International Conference on.* IEEE, 439–448.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled Generative Adversarial Networks. *arXiv:1611.02163* (2016).

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

Augustus Odena, Christopher Olah, and Jonathon Shlens. 2016. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv:1610.09585* (2016).

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

Alec Radford, Luke Metz, and Soumith Chintala. 2015. 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. 2016. Generative adversarial text to image synthesis. In *ICML*.

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

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

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

TensorFlow Team. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). `http://tensorflow.org/` Software available from tensorflow.org.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, and others. 2016b. Conditional image generation with pixelcnn decoders. In *NIPS*. 4790–4798.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016a. Pixel Recurrent Neural Networks. *arXiv:1601.06759* (2016).

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

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

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv:1609.05473* (2016).

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating Text via Adversarial Training. *NIPS Workshop on Adversarial Training* (2016).

.1 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.
- **DBN**: 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 DBN. 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.

.2 Heart failure cohort construction

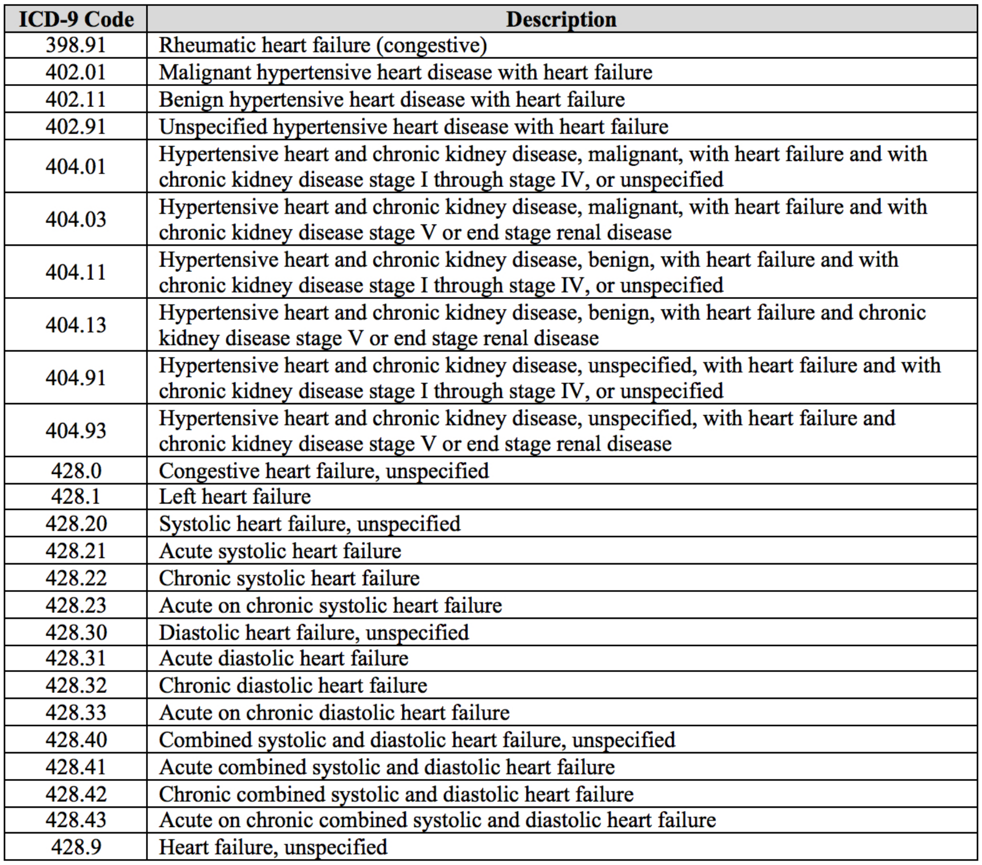
Case patients were 40 to 85 years of age at the time of HF diagnosis. HF diagnosis (HFDx) is defined as: 1) Qualifying ICD-9 codes for HF appeared in the encounter records or medication orders. Qualifying ICD-9 codes are displayed in Table 2. 2) a minimum of three clinical encounters with qualifying 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.

.3 Quantitative Evaluation Results for MIMIC-III

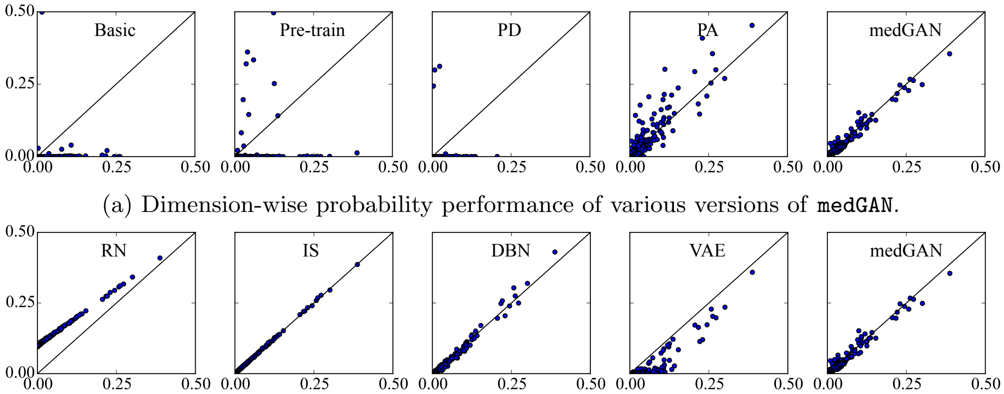
.3.1 Dimension-wise probability

Figure 8a shows the consistent superiority of the full version of medGAN compared other versions. The effect of minibatch averaging is even more dramatic for MIMIC-III. Figure 8b shows that VAE has some difficulty capturing the dimension-wise distribution of MIMIC-III. Again, DBN shows comparable performance to

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



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

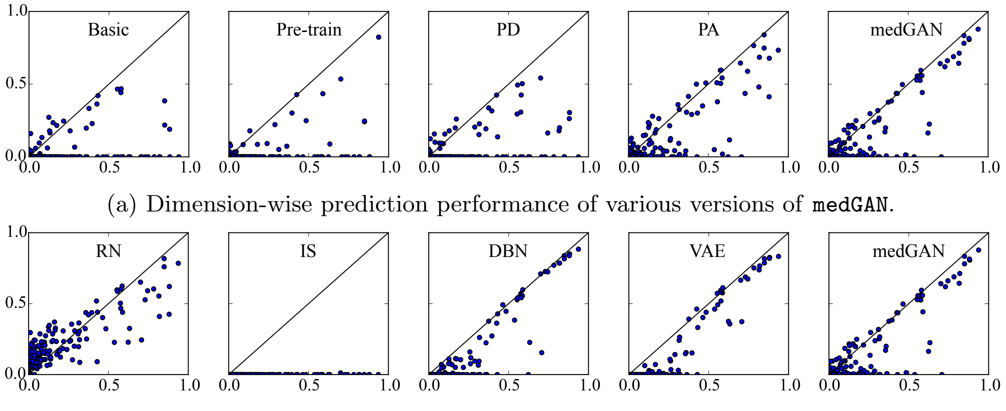
(b) Dimension-wise probability performance of baseline models and `medGAN`.

Figure 8: 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 MIMIC-III, 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.

`medGAN`, slightly outperforming `medGAN` for low-probability codes, but slightly underperforming for high-probability codes. Overall, dimension-wise probability performance is somewhat weaker for MIMIC-III than for Sutter PAMF, most likely due to smaller data volume and sparser code distribution.

.3.2 Dimension-wise prediction

Figure 9a shows the dimension-wise predictive performance for different versions of `medGAN` where the full version outperforms others. Figure 9b shows similar pattern as Figure 3b. Independent sampling completely fails to make any meaningful prediction. VAE demonstrates weakness at predicting low-probability codes. DBN 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 MIMIC-III than they did for Sutter PAMF.



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

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

Figure 9: 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 MIMIC-III. 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.