Synthetic Data Generation Using Bayesian Neural Networks

Abhishek Jha, Preetom Saha Arko, Yuan Zhou

Abstract

With the increasing demand for diverse datasets in machine learning applications, the generation of synthetic data has emerged as a popular research area. Bayesian Neural Networks introduce a probabilistic framework, enabling modeling of uncertainty and providing a principled mechanism for generating diverse samples. This study explores three different applications of Bayesian Neural Networks (BNN) for generating synthetic data - synthetic data generation using Genetic Algorithm, synthesis of Electronic Health Records using Bayesian GAN, and deep generative second-order ODEs with Bayesian neural networks.

1 Introduction

Traditional neural networks use point estimates for their weights, which may not capture uncertainties in the data. In contrast, Bayesian neural networks (BNN) treat the weights as probability distributions, allowing them to model uncertainty and generate synthetic data that reflects this uncertainty. In this study, we will explore three different approaches to synthetic data generation using Bayesian neural networks.

The increasing digitization of sensitive information has led to heightened concerns regarding privacy and the security of individual-level data. Particularly, where datasets contain private details. The risk of identity and attribute disclosure poses a significant challenge. Addressing these concerns necessitates innovative approaches that balance the preservation of privacy with using data for analytical purposes. Here, the exploration of synthetic data emerges as a promising strategy. Statistical properties of an original dataset are employed to generate artificial samples.

This study explores the utilization of a genetic algorithm for customizing Bayesian networks to improve the efficiency and privacy-preserving capabilities of synthetic data. By replacing the conventional greedy algorithm, the genetic algorithm offers a new perspective on mitigating disclosure risks while maintaining the utility of the data for machine learning tasks. Through empirical experiments on a set of three datasets, the study aims to assess the algorithm's efficacy and highlight its potential as a privacy-preserving option in section 2.

Generating Electronic Health Records (EHR) is a good application of synthetic data generation. EHRs are scarce and have a lot of privacy and confidentiality concerns. That's why there is a pressing need to generate good diverse EHRs from the already available ones, while ensuring that privacy and confidentiality do not get compromised if the synthetic data gets leaked. In section 3, we will discuss on synthesis of electronic health records using Bayesian Generative Adversarial Networks.

In section 4, we will discuss how to utilize BNN to variational Autoencoder (VAE). The VAE model is a generative model that encodes input data into a compact representation and decodes it back. However, VAEs assume independently distributed latent representations, limiting their applicability to sequential data. Addressing this limitation, the Ordinary Differential Equation Variational Auto-Encoder (ODE2VAE) model is proposed for high-dimensional sequential data. It introduces a latent second-order Ordinary Differential Equation (ODE) to capture position and momentum, accounting for acceleration in the latent dynamic state. To handle uncertainty and overfitting, the model employs

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

a Bayesian neural network for second-order differential equation solutions and utilizes variational inference for optimization.

2 Synthetic Data Generation using Genetic Algorithm

2.1 Motivation

The concept of using synthetic data as a privacy-preserving measure in micro-data is explored in the context of mitigating identity and attribute disclosure risks. The approach involves learning the statistical properties of an original dataset, storing this information in a model, and generating artificial samples using a Bayesian network. The focus during this discussion is on fully synthetic data, where all columns of the original dataset are synthesized. The efficiency is improved with the replacement of the greedy algorithm by a genetic algorithm. In addressing disclosure risks, particularly in micro-data, the distinction between identity and attribute disclosure is highlighted. While fully synthetic data eliminates one-to-one links between real individuals and synthetic samples, the risk of attribute disclosure remains. The idea of customizing network structures for specific machine learning tasks is introduced to reduce this risk, enhancing the overall applicability of synthetic data for privacy preservation. The approach holds promise for decreased disclosure risks and increased privacy. In the original implementation of PrivBayes and later in DataSynthesizer, a greedy algorithm is utilized to learn a Bayesian network from the input dataset [23]. However, as the maximum number of parents for each node increases or when dealing with large dataset dimensions, the computational cost escalates significantly . To address this challenge and enhance the efficiency of the data description process, an alternative to the greedy approach is explored, namely a genetic algorithm [12].

2.2 Implementation

The genetic algorithm aims to maximize the total sum of pairwise mutual information scores when assessing the quality of each individual. This scoring mechanism evaluates the strength and relevance of relationships between attributes within the represented Bayesian Network structure. Individuals with higher total mutual information scores are considered more effective at capturing meaningful dependencies among attributes.

$$MI(A,B) = \sum_{a \in A} \sum_{b \in B} P(a \cap b) \log \left(\frac{P(a \cap b)}{P(a)P(b)} \right)$$

The initialization process creates a population of individuals, where each individual represents a potential Bayesian Network structure. The ordering chromosome determines the order in which attributes appear in the network, and the connectivity chromosome represents the parents of each attribute based on the connectivity constraints. The process ensures that each individual is initialized with a valid set of connectivity relationships [12].

Algorithm 1: Initialization

Data: Parameters P, d, and k
for $i = 1$ *to* P **do**
  Generate a random permutation O of $f_1, \ldots, f_d$;
  Let O be the ordering chromosome of individual i;
  for $j = 1$ *to* d **do**
    Let F be the set of attributes that appear prior to f_j in O;
    if $|F| \geq k$ **then**
      Let C_j be a set of k distinct random elements of F;
    else
      Let C_j contain all elements of F and $k - |F|$ distinct random attributes not in F;
  Let $C = [C_1, \ldots, C_d]$ be the connectivity chromosome of individual i;
  Save the individual $i = (O, C)$;

The crossover operation combines information from two parent individuals by dividing the connectivity chromosome at a random point j and taking the first part from one parent and the second part from

the other parent. This operation introduces diversity into the population and helps explore different combinations [12].

Algorithm 2: Crossover(i_1, i_2)

Data: Individuals $i_1 = (O_1, C_1)$ and $i_2 = (O_2, C_2)$
Result: New individual $i = (O, C)$
// Generate a random integer j in $[0, d]$
$j \leftarrow$ random integer in $[0, d]$;
// Create the new individual
for $k = 1$ *to* j **do**
  Let C_k be the k-th set of C_1;
for $k = j + 1$ *to* d **do**
  Let C_k be the $k - j$-th set of C_2;
// Construct the new individual
Let O be the ordering chromosome of i_1;
Let $C = [C_1, \ldots, C_d]$ be the connectivity chromosome of the new individual i;
return $i = (O, C)$;

The order flip operation randomly selects an attribute in the ordering chromosome and swaps its position with another randomly selected attribute. The probability of performing the flip is controlled by the random number x and the threshold r. This operation introduces small variations in the ordering of attributes, contributing to the genetic diversity [12]. The swap operation

Algorithm 3: Order Flip(i)

Data: Individual $i = (O, C)$
Result: Individual i with updated ordering chromosome O
for f *in* O **do**
  // Generate a random number x in $[0, 1]$
  $x \leftarrow$ random number in $[0, 1]$;
  // Check if the flip should be performed
  if $x < r$ **then**
    // Generate a random integer j in $[1, d]$
    $j \leftarrow$ random integer in $[1, d]$;
    // Flip f with f_j in O
    Flip f with f_j in O;
return i with updated ordering chromosome O;

modifies the connectivity chromosome by swapping attributes within the sets G and Cj. The probability of performing these swaps is controlled by random numbers and the specified mutation

rate. This operation introduces variations in the connectivity relationships between attributes [12].

Algorithm 4: Swap(i)

Data: Individual $i = (O, C)$
Result: Individual i with updated connectivity chromosome C

for $j = 1$ *to* d **do**
  // Let G be the set of attributes in C_j that appear later than f_j in O
  Let G be the set of attributes in C_j that appear later than f_j in O;
  for g *in* G **do**
    // Let F be the set of attributes that appear prior to f_j in O and are not already in C_j
    Let F be the set of attributes that appear prior to f_j in O and are not already in C_j;
    if $F \neq \emptyset$ **then**
      // Swap g with a random element in F
      Swap g with a random element in F;
  for c *in* C_j **do**
    // Generate a random number x in $[0, 1]$
    $x \leftarrow$ random number in $[0, 1]$;
    if $x < r$ **then**
      // Let F be the set of attributes that appear prior to f_j in O and are not already in C_j
      Let F be the set of attributes that appear prior to f_j in O and are not already in C_j;
      if $F \neq \emptyset$ **then**
        // Swap c with a random element in F
        Swap c with a random element in F;
return i with updated connectivity chromosome C;

The next generation is generated by starting with the fittest individuals from the previous generation. It then iteratively creates new individuals through genetic operations, specifically applying crossover, order flip, and swap with a certain probability. The resulting individuals are added to the new generation until the desired population size is achieved. This process helps explore and exploit potential solutions to the optimization problem [12].

Algorithm 5: Next Generation

Data: Set of individuals E from the previous generation, population size P, mutation rate r
Result: Set of individuals G for the next generation

Let $G = E$;
while $|G| < P$ **do**
  // Choose a random individual i in E
  $i \leftarrow$ randomly select an individual from E;
  // Generate a random number x in $[0, 1]$
  $x \leftarrow$ random number in $[0, 1]$;
  if $x < r$ **then**
    // Choose a random individual i' in E
    $i' \leftarrow$ randomly select another individual from E;
    // Perform Crossover, Order Flip, and Swap operations
    $i \leftarrow$ Crossover(i, i');
    $i \leftarrow$ Order Flip(i);
    $i \leftarrow$ Swap(i);
  // Add i to G
  Add i to G;
return G;

2.3 Discussion and Results

The algorithm's performance was assessed in capturing complex relationships within three datasets– namely contraception, seeds, and lung cancer, to compare its outcomes.

The first one was contraception dataset, where even for small degrees of Bayesian network(n=2,3,4), we observe a rapid increase in computational time. Despite this computational demand, the genetic algorithm demonstrates its efficacy in capturing intricate data patterns, as evidenced by the generated heat maps. These visual representations provide insight into the relationships between variables and showcase the algorithm's ability to discern meaningful structures within the contraception dataset. For the contraception dataset, the relationship between a wife's education and contraception use is

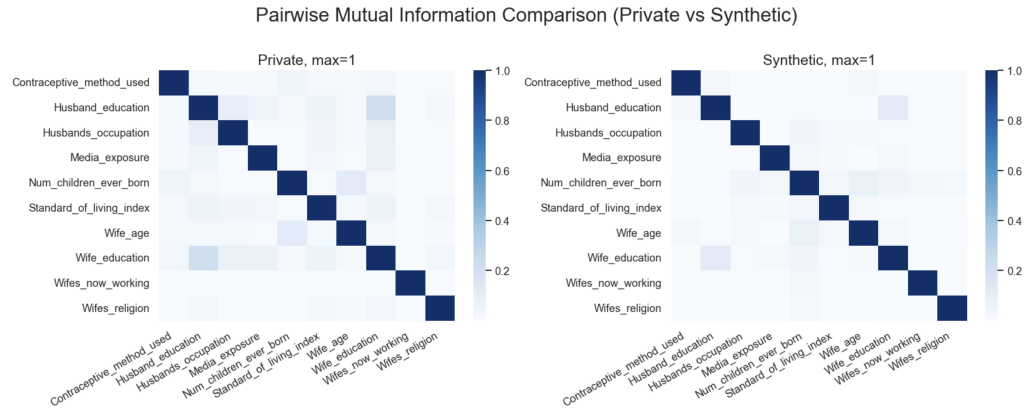

Figure 1: Heatmap for contracpetion dataset, n=2

observed. This assess the genetic algorithm's ability to capture relationships between these variables. Notably, even for a n=2, the initial results appear promising. Similar trends are observed in the seeds

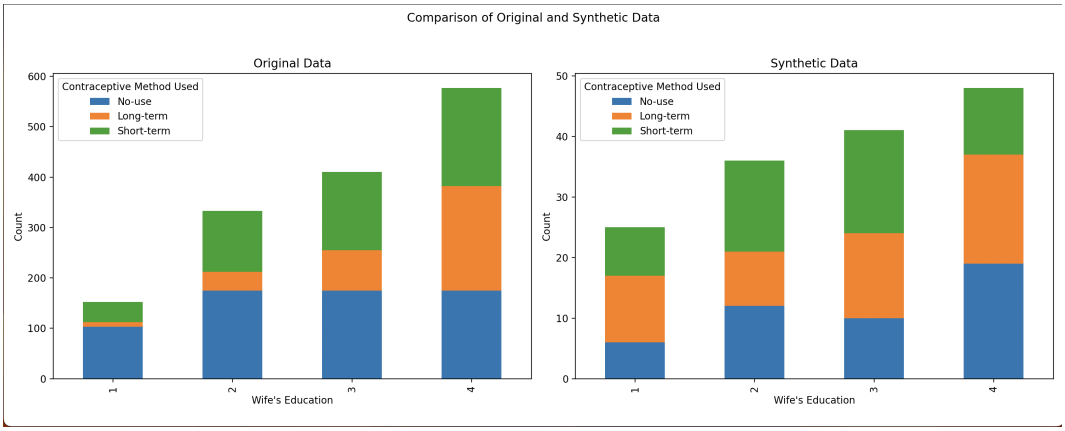

Figure 2: Observing the relationship between education and contraception use in original and synthetic dataset

dataset, where the genetic algorithm proves adept at handling creating the synthetic data using low computation power while producing insightful heat maps. The algorithm's capability to capture nuanced patterns becomes evident, emphasizing its potential utility.

In the case of the lung cancer dataset, where the number of features is significantly greater, the genetic algorithm remains effective in capturing the essence of the original data relationships. The generated heat maps showcase this as evidenced by the heatmap. However, it's crucial to note that due to the increased complexity with a larger number of features, nuances in the relationships demand more computational power. Despite these challenges, the algorithm showcases its capability to unveil meaningful patterns in the lung cancer data.

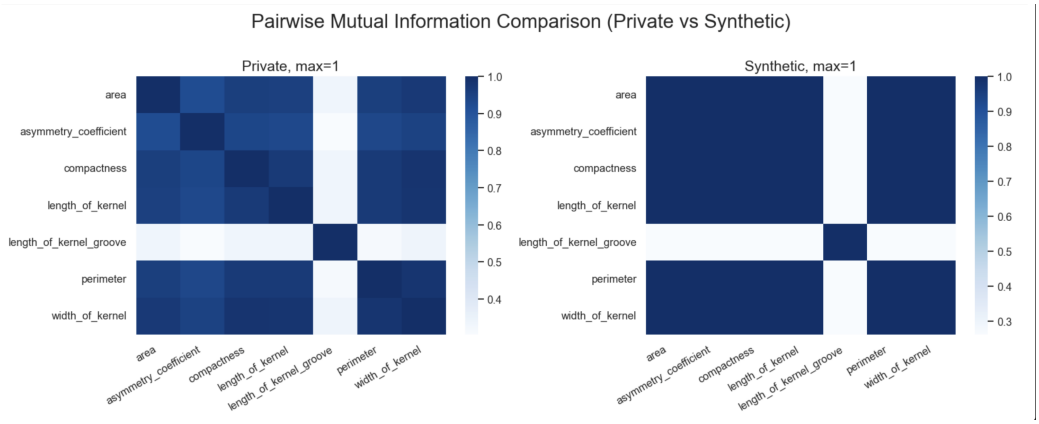

Figure 3: Heatmap for seeds dataset, n=2

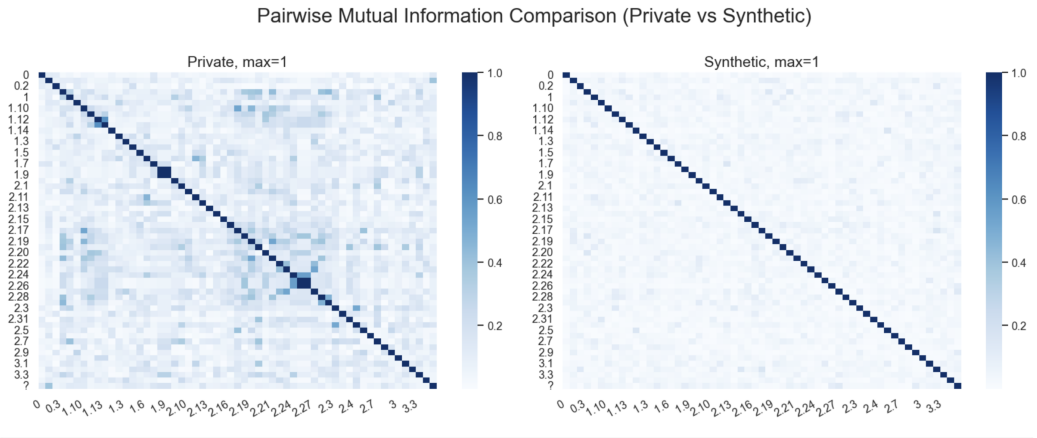

Figure 4: Heatmap for lung cancer dataset, n=3

3 Synthesis of Electronic Health Records Using Bayesian GAN

3.1 Motivation and Related Works

Generative adversarial networks (GAN) [9] are deep neural network architectures consisting of two components: a generator and a discriminator. Given some unlabeled input data, the generator G tries to generate truly new samples from the input data distribution, therefore implicitly trying to approximate the true data distribution as best as possible. The discriminator D tries to determine as best as possible if any input was part of the real input data or was generated by the generator G. By providing feedback on each other's outputs, the generator and discriminator incrementally improve on their respective tasks.

Due to privacy concerns and patient confidentiality regulations, research on Electronic Health Records (EHR) typically requires synthetic data, and the generator of GAN seems a promising tool for that. For this, we need a good GAN model that can faithfully capture the distribution of actual health data and later its generator can generate synthetic data from which no confidential information can be inferred. However, a GAN exhibits remarkable performance in generating real-valued continuous data, but it has limitations in generating discrete data [18]. The main reason is that a GAN fails to learn the distribution of discrete data in its original form during the gradient update process in training. To overcome this limitation, MEDGAN [7] incorporated an autoencoder with the original GAN to learn the distribution of discrete data and then generated synthetic discrete EHR data. The output of the generator is passed through the decoder of the autoencoder before going to the discriminator. SynthEHR [4] proposed two improved design concepts of the original GAN, namely, Wasserstein GAN with gradient penalty (WGAN-GP) [10] and boundary-seeking GAN (BGAN) [13]

as alternatives to the GAN in the MEDGAN framework. The proposed models named MEDBGAN and MEDWGAN both outperformed MEDGAN.
GANs are notoriously hard to train for various reasons. One such reason is that if the generator G finds some convenient way to fool the discriminator D, it can overly exploit this and overfit to a specific part of the distribution. We will then have a generator that does not correctly approximate the distribution of the data, but has instead "collapsed" to one of its modes. The resulting generated data will exhibit very poor diversity. This phenomenon is known as **mode collapse** and this is a very common problem of GANs. To solve this issue, AdaGAN [21], MGAN [14], MADGAN [8] used multiple generators instead of a single one with the intuition that multiple generators can better model multi-modal distributions since each generator only needs to capture a subset of the modes. To entail model aggregation, probabilistic modeling is a natural and principled framework to articulate the aggregation process.

Recently some probabilistic approaches have been tried to address the issue of mode collapse. For example, Bayesian GAN [19] aims to overcome mode collapse by simulating the discriminator and generator from their respective conditional posterior distributions; however, the two conditional posterior distributions are incompatible and can lead to unpredictable behavior [1], especially for minimax-style GAN objective. The authors of ProbGAN [11] showed that Bayesian GAN suffers from bad convergence due to its fixed and weakly informative prior. To solve this issue, ProbGAN imposes an adaptive prior on the generator and updates the prior by successively multiplying the likelihood function at each iteration; consequently, the generator converges to a fixed point instead of a distribution. The most recent approach, Empirical Bayesian GAN (EBGAN) [16] improves over both Bayesian GAN and ProbGAN to solve mode collapse. In this approach, the discriminator converges to a fixed point while the generator converges to a distribution at the Nash equilibrium. Multiple generators are simulated from the posterior distribution conditioned on the discriminator using momentum stochastic gradient Langevin dynamics (MSGLD) algorithm, and the discriminator is updated using stochastic gradient descent along with simulations of the generators. EBGAN has a strong theoretical guarantee to solve the issue of mode collapse.

MEDGAN previously incorporated the minibatch averaging method into their adversarial framework to prevent the problem of mode collapse, but it has no strong theoretical guarantee. As SynthEHR architecture is based on MEDGAN, it also employs the same method. In our study, we will try to find out whether EBGAN (current state-of-the-art method for addressing mode collapse) can outperform MEDGAN or SynthEHR.

3.2 Dataset

We used Medical Information Mart for Intensive Care (MIMIC-III) database [15], a freely available public database comprising de-identified electronic health records associated with approximately 60K patient admissions to the critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. *MIMIC-III* contains various types of health-related data, of which we used patients' diagnoses data (DIAGNOSES_ICD) and procedures (PROCEDURES_ICD) data, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system. Following the previous approaches [7] [4], we reduced the ICD codes to 3-digit codes. In the *MIMIC-III* dataset, each row corresponds to a patient's admission record of diagnoses and procedures data, represented by ICD codes. A patient likely visits a hospital more than once, so s/he may have multiple records in the dataset. We aggregated each patient's longitudinal record into a single fixed-sized vector of ICD codes. Thus, we represented the dataset as a multidimensional matrix, in which a row corresponds to a patient's record and a column to a specific ICD code (eg, diagnoses code or procedure code). Since ICD codes are aggregated by the patients, they are all count variables. The count variables indicate the number of times a patient was associated with a specific ICD code. In the end, our matrix has 46517 patients and 1358 ICD codes.

3.3 Experimental Setup

Interpreting synthetic data: The raw generated data values from the generator were any continuous nonnegative numbers. We rounded the continuous values of the synthetic data to the nearest integer values.

Source code: We obtained the Pytorch implementation of Bayesian GAN [2], Tensorflow implementation of MEDGAN, MEDBGAN and MEDWGAN [3] and Pytorch implementation of EBGAN [20] from Github. For Bayesian GAN, We had to change some in-place operations to make it work. We had to downgrade our Tensorflow version to 1.14 to train MEDGAN, MEDBGAN and MEDWGAN.

Method for evaluating synthetic EHRs: We performed dimension-wise Kolmogorov–Smirnov test (K–S) test on 2 data samples (synthetic data and real data) to examine whether the 2 data samples originate from the same distribution. In the K–S test, the statistic is calculated by finding the maximum absolute value of the differences between 2 samples' cumulative distribution functions. The null hypothesis is that both samples originate from a population with the same distribution. In our experiment, we rejected the null hypothesis when P-value ≤ 0.05.
We used 80% of our dataset for training and the rest 20% for testing. Specifically, during testing, we generated the same number of samples as the number of samples in the test set using the generator and then performed dimension-wise K-S test between the generated samples and the test samples. We report the average value along all dimensions.

System Information: We used an AMD Ryzen 9 7940HS laptop with NVIDIA RTX 4060 GPU to train the models.

3.4 Experiments and Results

As our study is based on Bayesian Neural Networks and Bayesian GAN is the first Bayesian approach to address mode collapse, we trained Bayesian GAN in its default configuration on CIFAR-10 dataset. Surprisingly, Bayesian GAN performed worse in semi-supervised learning than conventional GAN (Figure 5).

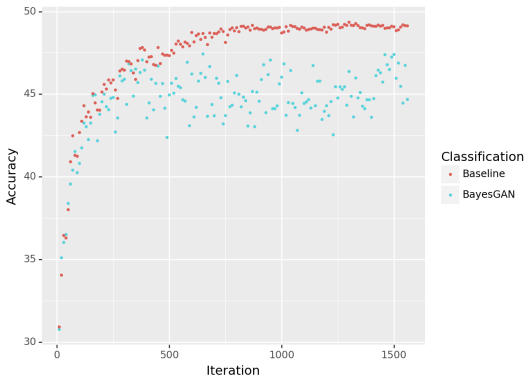

Figure 5: Baeysian GAN vs Conventional GAN in semi-supervised setting on CIFAR-10 dataset

To maintain the good aspects of both MEDGAN and EBGAN, we tried to follow the model architecture of MEDGAN while designing the generator and discriminator in EBGAN. Following the approach of MEDGAN, we added an autoencoder to EBGAN, trained it on our training data and then trained the generator and discriminator of EBGAN. As mentioned before, the output of the generator goes through the decoder before going to the discriminator in this approach. During training, generator loss initially decreased upto 15 epochs, then increased continuously, leading to unstable training. Changing the learning rate had almost no effect on this trend. Probably the autoencoder has somewhat memorized the data and it directly contradicts our approach to find a distribution for the generator. The highest K-S test similarity we got in this setting is only 52.45%.

Then we removed the autoencoder from EBGAN. We had to remove some skip connections in the generator to keep the dimensions suitable for matrix operations in the neural network. Following MEDGAN, we trained EBGAN in this setting for 1000 epochs with a batch size of 1000. We fine-tuned the learning rates of generator and discriminator and found that generator learning rate of $0.2 - 0.3$ and discriminator learning rate of $0.01 - 0.05$ works well for our dataset. The GAN typically converges in 200 epochs and no significant change can be observed afterwards. We got the same K-S test similarity for 200 epoch training and 1000 epoch training.

Model	K-S test similarity
MEDGAN	87.63%
MEDWGAN	90.06%
MEDBGAN	90.79%
EBGAN	86.97%

Table 1: K-S test similarity shown by various models on our dataset

We have also trained MEDGAN, MEDBGAN and MEDWGAN models on our dataset. MEDWGAN was found to perform the best among these three models on the dataset we used [4]. A comparison of the K-S test similarity between different models on our dataset can be seen in Table 1.

From Table 1, we can see that the data generated by MEDBGAN model resembles the actual data the most. This means the probabilistic approach of EBGAN cannot outperform the minibatch averaging method and boundary-seeking GAN objective of MEDBGAN.

EBGAN also proposes Kullback-Leibler divergence-based prior which enhances the similarity between synthetic data and real data at the density level. But implementing it resulted in CUDA Runtime error.

Our implementation of EBGAN and Bayesian GAN can be found here: `https://github.com/preetom-saha-arko/EHR_Synthesis`

4 Deep Generative Second-Order ODEs with Bayesian Neural Networks

The Variational Autoencoder (VAE)[17] is a type of generative model in machine learning. As a specialized autoencoder, it learns to encode input data into a compact representation and decode it back. Notably, VAEs introduce a probabilistic element, enabling diverse data generation through sampling in a lower-dimensional space. They are commonly used for tasks like image generation and representation learning. However, because the VAE model has a prior assumption that latent representations of samples are independently distributed (iid), it can not deal with sequential samples like medical images or autonomous driving. Because sequential samples should have similar latent representations and they should be correlated.

The Ordinary Differential Equation Variational Auto-Encoder (ODE2VAE) model is a latent second-order ODE model for high-dimensional sequential data[22], decomposing the latent dynamic ODE state into position and momentum capturing the acceleration and modeling the latent space for the sequential data. Additionally, to deal with the uncertainty in dynamics and overfitting, the author used a Bayesian neural network to get the answer for the second-order differential equation and used variational inference to optimize the model.

4.1 Ordinary Differential Equations

Consider a residual model like ResNet, the transformation for the hidden layer is shown in Eq. 1, where h_t represents the input of the t^{th} layer, θ_t is the parameters in the t^{th} layer and f is the output of each layer.

$$h_{t+1} = h_t + f(h_t, \theta_t) \tag{1}$$

Based on this, what if we have a model with infinite layers and each layer only adds a small step to the output? The equation can change to the Eq. 2, which means the output of the T^{th} layer of this model is the solution to this ordinary differential equation at time T.

$$\frac{dh(t)}{dt} = f(h(t), t, \theta) \tag{2}$$

Because the residual model is discrete, we can create a model that solves the ODE continually, which means an ODE network defines a vector field, which continuously transforms the state. To train the ODE model, treat it as a black box, and compute gradients using the adjoint sensitivity method to solve the equation[6].

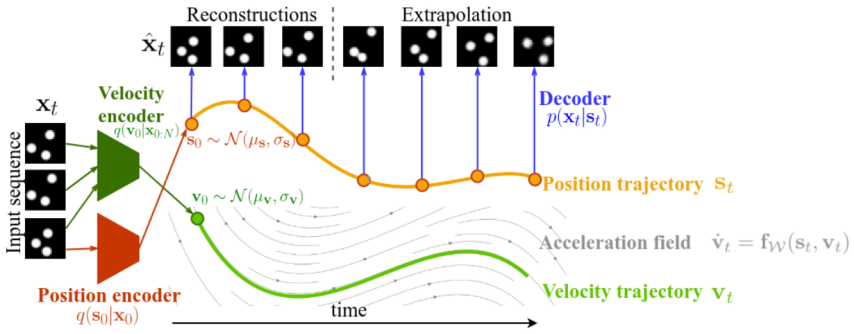

Figure 6: The illustration of ODE2VAE model

4.2 Bayesian Second-order ODEs

However, the first-order ODE model encounters limitations in addressing systems with high-order dynamics, such as acceleration or the motion of a pendulum. Furthermore, deterministic ODE models face challenges in handling uncertainty. In light of these considerations, the Bayesian second-order ODE model emerges as a viable solution to rectify these issues. The equation for second-order ODE is shown in Eq. 3,

$$\ddot{z}_t \coloneqq \frac{d^2\mathbf{z}_t}{d^2t} = \mathrm{f}_\mathcal{W}\left(\mathbf{z}_t, \dot{\mathbf{z}}_t\right) \tag{3}$$

which can be reduced by a couple of first-order ODE models shown in Eq. 4. The velocity of change of the sequential samples is governed by a neural network $\mathrm{f}_\mathcal{W}$, where $\mathcal{W}$ is a set of parameters $\{W_l\}$ of the layers in the neural network and the distribution of $\mathcal{W}$ is resulting from a BNN and its prior.

$$\begin{cases} \dot{s}_t = v_t \\ \dot{v}_t = \mathrm{f}_\mathcal{W}\left(s_t, v_t\right) \end{cases}, \begin{bmatrix} s_T \\ v_T \end{bmatrix} = \begin{bmatrix} s_0 \\ v_0 \end{bmatrix} + \int_0^T \begin{bmatrix} v_t \\ \mathrm{f}_\mathcal{W}\left(s_t, v_t\right) \end{bmatrix} dt \tag{4}$$

4.3 ODE2VAE Model

The OED2VAE model[22] is based on the Bayesian second-order ODE model, enabling the derivation of latent representations for sequential samples and the generation of samples across the temporal domain.

First, characterizing the position and velocity s_0 and v_0 of the input sequence, respectively. Subsequently, a second-order ODE model is utilized to compute the current velocity representation, v_t. The dynamics of the system are further modeled through integration, yielding the current position representation, s_t. Finally, the obtained latent representation is passed through a decoder to generate the corresponding output figure. The structure of the model is shown in Fig. 6.

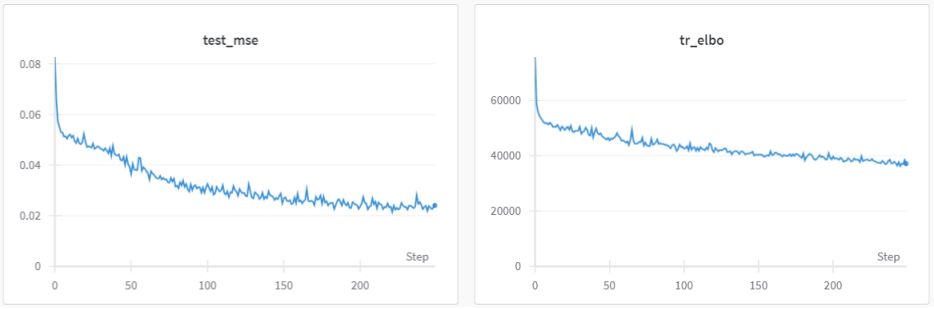

Figure 7: The loss and test accuracy of ODE2VAE

Because using the VAE model to generate new samples, we use variational inference to optimize the whole model including the BNN. And the Evidence Lower Bound can become as follows:

$$\log p(X) \geq \underbrace{-\mathrm{KL}\left[q(\mathcal{W}, Z|X)\,||p(\mathcal{W}, Z)\right] + \mathbb{E}_{q(W,Z|x)}\left[\log p(X|\mathcal{W}, Z)\right]}_{\text{ELBO}} \tag{5}$$

$$= -\mathbb{E}_{q(\mathcal{W},Z|X)}\left[\log \frac{q(\mathcal{W})\,q(Z|\mathcal{W}, X)}{p(\mathcal{W})\,p(Z)}\right] + \mathbb{E}_{q(\mathcal{W},Z|X)}\left[\log p(X|\mathcal{W}, Z)\right] \tag{6}$$

$$= -KL\left[q(\mathcal{W})\,||p(\mathcal{W})\right] + \mathbb{E}_{q(\mathcal{W},Z|X)}\left[-\log \frac{q(Z|\mathcal{W}, X)}{p(Z)} + \log p(X|\mathcal{W}, Z)\right] \tag{7}$$

$$= \underbrace{-KL\left[q(\mathcal{W})\,||p(\mathcal{W})\right]}_{\text{ODEregularization}} + \underbrace{\mathbb{E}_{q_{enc}(\mathbf{z}_0|X)}\left[-\log \frac{q_{enc}(\mathbf{z}_0|X)}{p(\mathbf{z}_0)} + \log p(\mathbf{x}_0|\mathbf{z}_0)\right]}_{\text{VAEloss}} \tag{8}$$

$$+ \underbrace{\sum_{i=1}^{N} \mathbb{E}_{q_{ode}(\mathcal{W},\mathbf{z}_i|X,\mathbf{z}_0)}\left[-\log \frac{q_{ode}(\mathbf{z}_i|\mathcal{W}, X)}{p(\mathbf{z}_i)} + \log p(\mathbf{x}_i|\mathbf{z}_i)\right]}_{\text{dynamicloss}} \tag{9}$$

The first term functions as the BNN weight penalty, strategically introduced to forestall the potential for overfitting. Following this, the second term is the bound for the VAE model, effectively reverting to the classical VAE formulation for sequences of length 1. The pivotal disparity between the second and third terms resides in the methodology employed for expectation computation; notably, the third term calculates expectations contingent upon the variational distribution induced by the second-order ODE.

4.4 Experiments and Results

We test the performance of the model on a rotating MNIST dataset[5], which is rotating the images of handwritten "3" digit in 16 angles from 0 to 360. We left the same angle for testing. During training, we randomly removed four angles of the training dataset. During the testing phase, the ODE2VAE model exclusively acquires the latent representation at the initial time point and subsequently performs decoding at a specified time instance. This design imparts the capability for the model to generate images characterized by arbitrary rotation angles. For test accuracy, we calculate the MSE between the prediction and the label.

For the hyperparameters in the experiment, we set the $batch\_size = 64$, $epoch = 250$, $learningrate = 10^{-3}$, using the Adam optimizer and using the NVIDIA GeForce GTX 1650 GPU. The loss and test accuracy for each epoch is shown in Fig. 7. The code of the experiments can be found at `https://github.com/cagatayyildiz/ODE2VAE`. The generated samples are shown in Fig. 8, in each alternate row, labels are presented, while the subsequent row displays samples generated by the model. Notably, the results exhibit similarity to the original labels, even in instances where the handwritten style changes.

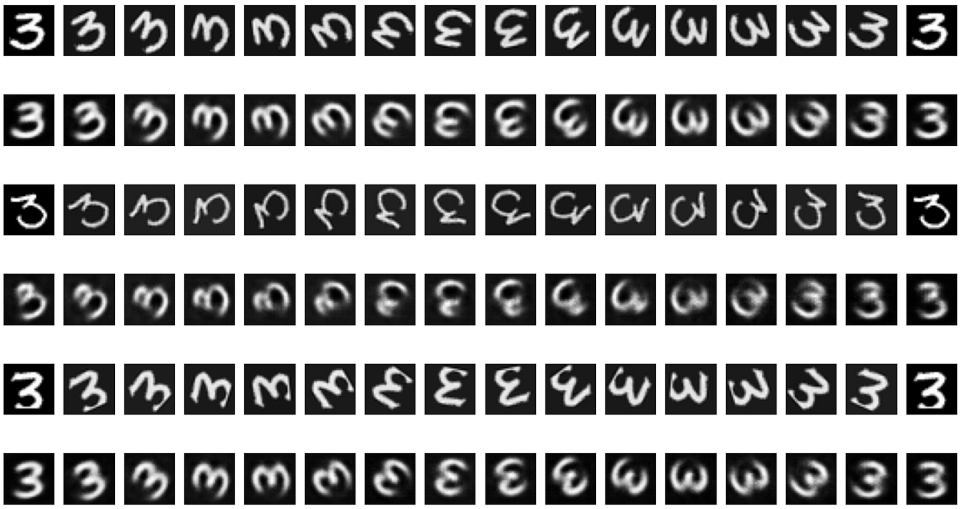

Figure 8: Samples generated by ODE2VAE

5 Conclusion and Future Work

We have discussed three different approaches for synthetic data generation using Bayesian Neural Networks. The experiments conducted in this study reveal that achieving the goal of preserving privacy does not necessitate a significant loss in the utility of the data for machine learning purposes. Consequently, the customization of Bayesian networks proves to be a viable addition to the toolkit of measures for statistical disclosure control. Exploring parallelization techniques and using advanced hardware accelerators could accelerate the genetic algorithm's rate of convergence.

Though our approach to generate synthetic EHRs could not outperform the current state-of-the-art in terms of K-S test similarity, the disclosure risk of our model may be lower. We plan to investigate it in the future. Adopting KL prior may boost the faithfulness of data generation.

The ODE2VAE model represents an extension of Variational Autoencoders (VAEs) for continuous-time dynamic modeling. This approach involves decomposing the latent space into position and velocity components and introducing a robust neural second-order differential equation system. In future enhancements, a potential avenue for improving the model lies in refining the ODE flow to better capture time variance in the samples. Furthermore, optimizing the model's performance can be explored through the refinement of the optimization method.

One way to improve the performance of our approaches involves using more computational resources. The noticeable computational demands, particularly in datasets with higher dimensions and with probabilistic neural networks, suggest that increasing computing power would noticeably enhance the performance of the models.

References

[1] Barry C Arnold and S James Press. Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156, 1989.

[2] Ben Athiwaratkun. pytorch-bayesgan. `https://github.com/vasiloglou/mltrain-nips-2017/tree/master/ben_athiwaratkun/pytorch-bayesgan`, 2017. [Online; accessed 6 December 2023].

[3] Mrinal Kanti Baowaly. SynthEHR. `https://github.com/baowaly/SynthEHR`, 2019. [Online; accessed 6 December 2023].

[4] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2019.

[5] Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[6] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[7] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[8] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8513–8521, 2018.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[11] Hao He, Hao Wang, Guang-He Lee, and Yonglong Tian. Probgan: Towards probabilistic gan with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

[12] Markus Hittmeir, Rudolf Mayer, and Andreas Ekelhart. Efficient bayesian network construction for increased privacy on synthetic data. *2022 IEEE International Conference on Big Data (Big Data)*, 2022.

[13] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

[14] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations 2018*. OpenReview, 2018.

[15] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[16] Sehwan Kim, Qifan Song, and Faming Liang. A new paradigm for generative adversarial networks based on randomized decision rules. *arXiv preprint arXiv:2306.13641*, 2023.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[19] Yunus Saatci and Andrew G Wilson. Bayesian gan. *Advances in neural information processing systems*, 30, 2017.

[20] sehwankimstat. EBGAN. `https://github.com/sehwankimstat/EBGAN/tree/main`, 2023. [Online; accessed 6 December 2023].

[21] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *Advances in neural information processing systems*, 30, 2017.

[22] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems*, 42(4), 2017.

Appendix

Study on synthetic data generation using genetic algorithm was performed by Abhishek Jha 2. Preetom Saha Arko studied synthesis of Electronic Health Records using Bayesian GAN 3. Investigation on deep generative second order ODEs with Bayesian neural networks was done by Yuan Zhou 4.