VAEs for Synthetic Data Generation

Dominic Danks

1 Executive Summary

This report summarises the work carried out as part of the NHSX PhD internship project entitled "Synthetic Data Exploration: Variational Autoencoders for Healthcare Data".

The primary purpose of the project was to investigate the suitability of Variational Autoencoders (VAEs) as synthetic health data generators within an NHS context. Experiments suggest that the VAE is a viable candidate for synthetic data generation, with performance at least on par with that of currently utilised Generative Adversarial Network-based approaches. The project also investigated the notion of differential privacy with the VAE as a case study, confirming that increasingly stringent differential privacy guarantees lead to a reduction in synthetic data quality.

Other important contributions of the project include: i) an accompanying open-source code repository containing code to train a VAE with differential privacy and benchmark synthetic data generation methods within the Synthetic Data Vault (SDV) software framework; ii) a discussion within this report of the frameworks and implementation considerations encountered during the project; iii) the inclusion within this report of explicit, otherwise undocumented details of SDV required to understand its outputs.

2 Introduction

Analysis of real-world healthcare datasets can provide significant insights into diseases and treatments which can in turn be used to improve patient outcomes. However, the sensitive nature of personal medical data has historically complicated analysis pathways, with often very few users able to access, and hence analyse, the data. With the size and granularity of medical datasets continually increasing, it is becoming increasingly important to devise solutions which allow for the analysis of these datasets in a privacy-preserving way.

One approach to this problem is synthetic data generation, which takes a private dataset as input and provides a new, synthetic dataset as output. This synthetic dataset should retain the important relationships and insights present in the original dataset, but should not contain information which could be used to reidentify the original dataset's subjects.

The most common form of synthetic data generation techniques, and those which are most generally applicable, are based around attempting to estimate the underlying distribution from which observations are drawn and then sampling from that distribution at generation time. The simplest example of such a method (referred to later in this work as "Independent") would be to assume independence across variables, model continuous variables via Gaussian distributions and categorical variables via categorical distributions and estimate the relevant sufficient statistics empirically. A synthetic dataset could then be generated by sampling independently from these learnt distributions. However, this approach does not capture dependencies between variables and does not consider privacy. The challenge when performing synthetic data generation in the healthcare setting is to generate data which accurately matches the distribution of each of the original variables, but whilst also capturing the dependencies between variables and introducing privacy.

Most modern work aimed at performing this task makes use of Generative Adversarial Networks (GANs, Goodfellow et al. (2014)). These are a form of generative model which are well-known for their ability to generate convincing synthetic images (see e.g. Brock et al. (2019)), but are also known for being "black-box" in their operation and unstable during training (Arjovsky and Bottou, 2017; Arjovsky et al., 2017).

Another form of generative model, which performs many of the same tasks as a GAN, is a Variational Autoencoder (VAE, Kingma and Welling (2014)). Variants of VAEs have demonstrated similar performance to GANs on image generation (see e.g. Vahdat and Kautz (2021); Child (2021)) but offer the ability to map observations onto a learnt latent space and allow for explicit specification of a likelihood model. This makes VAEs arguably more interpretable than GANs and allows for a greater level of user-definition within the generative model, which can be useful in healthcare settings where prior knowledge may be present.

Although VAEs are a well-regarded generative model, little work has been done to evaluate their suitability as a synthetic data generation tool. The ONS Data Science Campus released preliminary work including a VAE-based model, but in a limited capacity and without privacy considerations (see their online page). Wan et al. (2017) and Salim (2018) also consider VAE data generation, but without extensive data similarity metrics and privacy considerations.

This project investigates the potential suitability of VAEs as a synthetic data generation tool in the context of the NHS. To effectively address this direction, it explicitly focuses on 4 key aspects: quality, privacy, ease of use, and interpretability. Briefly, quality refers to how faithfully the synthetic data represents the real data, i.e. to what extent patterns present in the original dataset exist within the synthetic data. Privacy considers the extent to which the synthetic data retains sensitive aspects of the original dataset. Ease of use considers how readily a typical NHS user could progress from introduction to the approach and software to generating synthetic data for their use case. Finally, interpretability refers to how easy it is to understand why the model's results are as they are and to edit model design accordingly.

Section 3 details work which has been carried out in the area of synthetic data generation and is relevant to this project's scope and NHS context. Section 4 provides the necessary background information to understand the VAE approach implemented. Section 5 provides a detailed narrative of the implementation process, including decisions to use certain frameworks, potential alternatives and notes to be aware of when working with the codebase of this project and related frameworks. Section 6 details various experiments to evaluate the VAE-based method alongside other alternatives. Section 7 discusses the suitability of the VAE-based approach in light of the implementation aspects and observed results, as well as discussing natural continuation directions.

3 Related Work

This section presents work relevant to the project's investigation. It particularly focuses on methods which are compared against the VAE-based approach in the experiments of section (Section 6).

3.1 Copula-Based Generation

A d-dimensional copula is a joint cumulative distribution function over the d-dimensional hypercube $[0, 1]^d$ with uniform marginals. The utility of a copula is derived from the idea that a multivariate distribution can be decomposed into its marginal distributions and its correlation structure, the latter of which is described by a copula. This idea is formalised by Sklar's theorem, which states that a multivariate CDF, F, can be written in terms of its marginal CDFs $\{F_i\}$ and a copula C as

$$F(x_1, x_2, \ldots, x_d) = C(F_1(x_1), F_2(x_2), \ldots, F_d(x_d)).$$

This in turn implies that an $\mathbf{x}$ sample can be generated by first sampling $\mathbf{c}$ from the copula distribution and then setting $x_i = F_i^{-1}(c_i)$.

In the context of synthetic data generation, one can specify a probability distribution in terms of a set of marginals and a copula, both with learnable parameters. These parameters can then be learnt via e.g. maximum likelihood estimation with respect to the real data, thus providing a distribution from which samples are drawn to obtain the synthetic dataset.

An especially common and useful form of copula is the Gaussian copula, which defines its CDF in terms of Gaussian-based functions (see e.g. Martin Haugh's Introduction to Copulas for details) and is easily sampled from. A Gaussian copula model is one of the models evaluated in Section 6.

3.2 CTGAN

Conditional Tabular Generative Adversarial Network (CTGAN) is a method proposed in Xu et al. (2019) to generate synthetic tabular data. At its heart is a GAN, the generator of which can be used to generate synthetic data after training on real data. CTGAN is the prevailing choice of GAN model within the Synthetic Data Vault (SDV) framework and is evaluated in Section 6.

3.3 CopulaGAN

CopulaGAN is an adaptation of CTGAN which adds an additional transform of the input data before passing it to the GAN model, in particular transforming it to a distribution close to a standard normal. This is done by passing each variable through an estimate of its marginal CDF and then applying the inverse CDF of a standard normal. This can be seen as a more refined version of the typical standardisation and normalisation process within which one subtracts the mean and divides by the standard deviation. The effect of this step on generator performance can be seen by inspecting CTGAN vs CopulaGAN performance in the experiments section (Section 6).

3.4 ONS & NHS Digital

In 2019, the ONS Data Science Campus released preliminary work on synthetic data generation in an online page. That project shared similar motivations to this project but only lightly considered VAEs, did not consider categorical variables and did not include privacy as a consideration. More recently, in 2021, a team from the University of Warwick and NHS Digital published a more detailed piece of work (Arvanitis et al., 2021), however this also focused on GANs and did not explicitly consider privacy.

3.5 PATE

The two prevailing ways to introduce differential privacy into a synthetic data generation model during training are: i) DP-SGD (utilised in this work and discussed in Section 4) and ii) Private Aggregation of Teacher Ensembles (PATE) (Papernot et al., 2017). PATE involves the training of multiple "teacher" models, each trained on a separate partition of the dataset. A "student" model is then trained using noised outputs of the teacher models. This student model is the final output of the training procedure and has a calculable differential privacy budget. The PATE procedure was applied to a GAN by Yoon et al. (2019b) of the van der Schaar group. The group has also published work which introduces privacy to a GAN via an identifiability loss during training (Yoon et al., 2020) and also work considering time-series generation using GANs (Yoon et al., 2019a).

The benefit of DP-SGD over PATE is its conceptual and implementational simplicity. DP-SGD can be readily included in a PyTorch model via the Opacus framework (see Section 5.3). In contrast, PATE requires fundamental changes to the training procedure, with data partitioning and multiple parallel models required. We therefore prioritise DP-SGD in this work. A valid future direction could be to investigate the application of PATE to VAE training, comparing and contrasting it to the DP-SGD approach we use.

4 Background

This section presents in more detail the fundamental concepts relevant to our privacy-aware VAE-based approach.

4.1 Variational Autoencoders

At the heart of the Variational Autoencoder (VAE) framework is the idea that the variation within observed (potentially high-dimensional) data $\mathbf{x}$ can typically be captured by a low-dimensional latent variable $\mathbf{z}$ and an appropriate mapping between the latent space, in which $\mathbf{z}$ exists, and the data space, in which $\mathbf{x}$ exists. More formally, the VAE posits the generative model

$$\begin{aligned} \mathbf{z}_i &\sim p(\mathbf{z}_i), \ i = 1, \dots, n \\ \mathbf{x}_i|\mathbf{z}_i &\sim p_{\tilde{\theta}}(\mathbf{x}_i|\mathbf{z}_i), \end{aligned} \tag{1}$$

where $\mathbf{z}_i \in \mathbb{R}^p$ is the latent variable corresponding to the i-th data point $\mathbf{x}_i \in \mathbb{R}^d$ in the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1,\dots,N}$.

It is typical to place a standard normal prior on $\mathbf{z}$, i.e. $\mathbf{z} \sim \mathcal{N}(0, I)$. Meanwhile, the mapping from $\mathbf{z}$ space to $\mathbf{x}$ space is provided by the conditional likelihood $p_{\tilde{\theta}}(\mathbf{x}|\mathbf{z})$. A neural network (NN) referred to as the

decoder is used to compute this quantity. Typically the NN decoder, denoted f_θ, is used to parameterise the mean such that $\mathbb{E}[\mathbf{x}|\mathbf{z}] = f_\theta(\mathbf{z})$. Any additional parameters in the conditional likelihood (e.g. variance for a Gaussian) are treated as constant across samples and learnt. For example, in the case of a Gaussian conditional likelihood, $p_{\tilde{\theta}}(x_j|\mathbf{z}) = \mathcal{N}\left(x_j|f_\theta^{(j)}(\mathbf{z}), \sigma_j^2\right)$, with $\{\sigma_j^2\}$ learnable, j denoting output dimension, θ representing neural network parameters and $\tilde{\theta} = \left(\theta, \{\sigma_j^2\}\right)$.

Given data, the learning problem is to infer the values of parameters $\tilde{\theta} = \left(\theta, \{\sigma_j^2\}\right)$ most consistent with the observed data. The most immediate way to attempt this is maximum likelihood inference, that is to attempt to maximise the overall log-likelihood marginalised over z, i.e.

$$\log p_{\tilde{\theta}}(\mathbf{x}) = \log \int p_{\tilde{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z} \,. \tag{2}$$

This is generally intractable and cannot be optimised directly. To make progress, a distribution $q(\mathbf{z})$ is introduced via

$$\log p_{\tilde{\theta}}(\mathbf{x}) = \log \int \frac{p_{\tilde{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) \, \mathrm{d}\mathbf{z} \,.$$

Applying Jensen's inequality to the above yields

$$\log p_{\tilde{\theta}}(\mathbf{x}) \geq \int q(\mathbf{z}) \log \frac{p_{\tilde{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q(\mathbf{z})} \, \mathrm{d}\mathbf{z} \,.$$

The right hand side of this equation is referred to as the Evidence Lower Bound, or ELBO, and can be rewritten in a number of ways. The two most useful ways when considering VAEs are

$$\begin{aligned} \mathrm{ELBO}(\tilde{\theta}, q) &= \mathop{\mathbb{E}}_{\mathbf{z}\sim q(\mathbf{z})} \left[\log p_{\tilde{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left[q(\mathbf{z}) \| p(\mathbf{z})\right] && (3) \\ &= \log p_{\tilde{\theta}}(\mathbf{x}) - \mathrm{KL}\left[q(\mathbf{z}) \| p_{\tilde{\theta}}(\mathbf{z}|\mathbf{x})\right] . && (4) \end{aligned}$$

The second of these, i.e. (4), implies that the ELBO is tight to the log-likelihood if and only if $q(\mathbf{z}) = p_{\tilde{\theta}}(\mathbf{z}|\mathbf{x})$, i.e. if $q(\mathbf{z})$ matches the true posterior. Therefore, one could imagine optimising $\mathrm{ELBO}(\tilde{\theta}, q)$ by alternating two steps: i) fix $\tilde{\theta}$, compute the true posterior and set $q(\mathbf{z})$ to it; ii) maximise (3) w.r.t. $\tilde{\theta}$ with $q(\mathbf{z})$ fixed. This algorithm is exactly Expectation Maximisation (Dempster et al., 1977), and is applicable in simple cases (e.g. a linear decoder, which is equivalent to Factor Analysis), however in general it is not possible to compute the posterior, i.e. to perform step i).

One way to proceed, and the approach taken in the VAE framework, is to accept that the true posterior cannot be computed and instead model $q(\mathbf{z})$ as a member of a family of distributions (e.g. Gaussian) with parameters (e.g. mean and variance) given by a mapping parameterised by ϕ, so that $q(\mathbf{z})$ becomes $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. This is referred to as *amortised variational inference*, and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is referred to as the *variational posterior*. Specifically, in the VAE framework, a NN mapping, referred to as the *encoder*, is used to map from an observation $\mathbf{x}$ to parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}^2$ of the variational posterior $q_{\boldsymbol{\phi}}(z_j|x) = \mathcal{N}(z_j; \mu_j, \sigma_j^2)$ (this is for a Gaussian variational posterior, but other distribution families can be utilised in a similar way). By parameterising $q(\mathbf{z})$ in this way, the ELBO becomes a function of just decoder parameters $\tilde{\theta}$ and encoder parameters ϕ, such that the overall VAE training objective is to minimise

$$\begin{aligned} -\mathrm{ELBO}(\tilde{\theta}, \phi; \mathcal{X}) = &- \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} \left[\log p_{\tilde{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\right] \\ &+ \sum_{i=1}^{N} \mathrm{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) \| p(\mathbf{z}_i)\right] , \end{aligned} \tag{5}$$

which can be performed using standard gradient descent-based methods, e.g. Adam (Kingma and Ba, 2017) or RMSProp. In the experiments of this report, Adam is used. Figure 1 provides a pictorial representation of the VAE. A reconstruction of a data point $\mathbf{x}$ can be obtained by following the diagram from far left to far right. Specifically, $\mathbf{x}$ is fed to the encoder, which outputs the parameters of the variational posterior distribution. This posterior is then sampled from to obtain the associated $\mathbf{z}$ value. $\mathbf{z}$ is then passed through

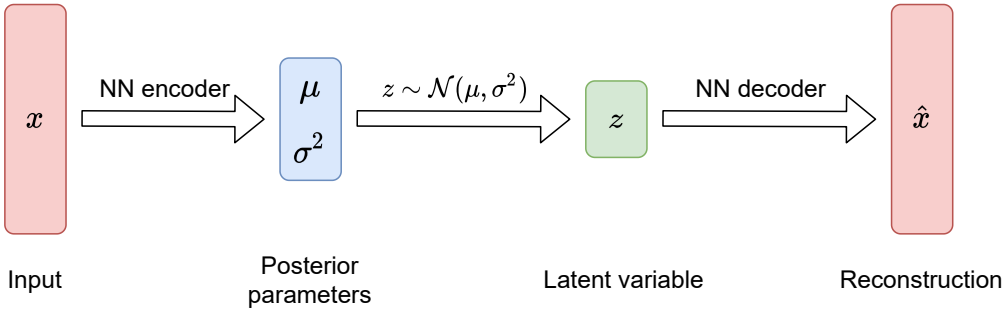

Figure 1: Schematic representation of a Variational Autoencoder.

the decoder to obtain the sufficient statistics of $p(x|z)$ which can then be sampled from to obtain $\hat{\mathbf{x}}$. While useful to understand a model's behaviour, this process is not what is used to generate synthetic data with a VAE. Instead, synthetic data is generated by simply sampling according to the generative model (1) with the trained decoder. This is the same as the process above, but replacing the encoder output with a sample from the prior. This explanation highlights the encoder's primary purpose as a training tool. Once training is complete, unless detailed internal model behaviour is to be investigated, the encoder can be discarded and synthetic data still generated.

4.2 Differential Privacy

An immediate challenge when attempting to include privacy within machine learning models is to precisely define and evaluate the privacy of an algorithm. Differential Privacy (DP) addresses this challenge by providing a formal privacy metric using the intuition that if the model preserves privacy, then the removal of any single individual should not significantly affect the output of the model.

More formally, a randomised algorithm $\mathcal{M}$ is defined to be (ϵ, δ) differentially private if

$$P(\mathcal{M}(\mathcal{D}) \in \mathcal{S}) \leq \mathrm{e}^{\epsilon} P(\mathcal{M}(\mathcal{D}') \in \mathcal{S}) + \delta,$$

where $\mathcal{D}$ is the dataset $\mathcal{D}'$ with any single entry removed and $\mathcal{S}$ is any subset of the image of $\mathcal{M}$, with P being the probability over the randomised aspect of $\mathcal{M}$. Note that if $\mathcal{M}$ is $(0, 0)$ differentially private, then the model is unaffected by the presence of an additional datapoint and is hence completely private. However, such a model would be unlikely to be useful. In general, a tradeoff must be made between complete privacy and optimal performance. Differential privacy (if it can be computed for the given procedure) allows the amount of privacy sacrificed during training to be quantified.

4.3 DP-SGD

Gradient descent (GD) is an iterative optimisation procedure which, at each iteration, performs a parameter update of the form

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta}),$$

where $\boldsymbol{\nabla}$ denotes the gradient with respect to $\boldsymbol{\theta}$, α is the step size (also referred to as the learning rate) and $\mathcal{L}$ is the loss function.

For convex optimisation objectives, it can be shown that (for sufficiently small α) gradient descent will converge to the global minimum, whilst in general it will tend to find local minima. Variants of gradient descent optimisation (e.g. Adam (Kingma and Ba, 2017), RMSProp) are currently the standard choice for optimisation of NN-based models (we use Adam).

Standard GD methods do not consider privacy, they focus only on decreasing the model's loss. DP-SGD attempts to retain the basic intuition and functionality of standard GD whilst introducing and quantifying privacy. It does this by editing the updates involved in GD in two ways: 1) it clips the gradients corresponding to any particular example (intuitively this can be thought of as preventing the model overfitting to outliers, hence sacrificing their privacy) and 2) it adds noise to the gradients. Its formal algorithmic definition is given in in Algorithm 1 in Figure 2.

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Figure 2: The DP-SGD algorithm. Credit: Abadi et al. (2016).

If Algorithm 1 is followed, then the resulting model is (ϵ, δ) differentially private with an (ϵ, δ) which can be calculated via a "moments accountant", which is included within an Opacus `PrivacyEngine` object. Any model which is trained via gradient descent can therefore be made differentially private by editing the training process in this way via use of Opacus (see Section 5.3).

5 Implementation

5.1 PyTorch vs. Alternatives

Typical approaches for training the models of interest in this project (e.g. VAEs, GANs) require the use of automatic differentiation and gradient-based optimisation. Various machine learning libraries offer this functionality in numerous languages, and the project could in principle have been carried out using any of these (e.g. TensorFlow in C++, TensorFlow in Python, Flux in Julia). PyTorch was chosen due to i) its ease of use, ii) its didactic code structure and iii) the large number of PyTorch-aware open-source libraries relevant to this project (in particular the Synthetic Data Vault discussed below).

5.2 Synthetic Data Vault (SDV)

Synthetic Data Vault (SDV) is an open-source project which provides synthetic data generation and evaluation functionality. In particular, it allows for numerous modern methods (e.g. Copulas, CTGAN) to be trained on a particular dataset and their performance compared, using a variety of metrics, with little code overhead. To our knowledge, it is the most advanced and thorough synthetic data framework in Python. Furthermore it is built using PyTorch, leading to particular compatibility with our VAE-based approach. We use various models and metrics from SDV in Section 6.

5.3 Opacus

Our VAE-based approach requires the use of DP-SGD and associated privacy budget calculations which are not implemented within standard PyTorch. Opacus provides this functionality in a natural way. In particular, a `PrivacyEngine` is attached to the optimiser used in a standard PyTorch model, after which the model can be trained as with standard PyTorch. This `PrivacyEngine` covertly i) edits the gradients used to update the parameters by clipping and noising as in DP-SGD and ii) computes the privacy budget associated with the learning process, providing an (ϵ, δ) DP value for the trained model.

5.4 Points of Note

It should be noted that SDV is still in the early stages of development, leading to a lack of documentation on some functions and the need for additional user-side code to complete the desired functionality. In addition, Opacus is largely tailored to and demonstrated on standard supervised learning neural network models, with some additions needed to enable private training of a VAE-based approach. This section contains some of the steps required to perform the experiments detailed below and aspects which should be kept in mind when working with our codebase and Opacus and SDV in general.

5.4.1 Opacus Layer Support

At the time of writing, Opacus only supports PyTorch `module`s with either no learnable parameters, parameters whose values are frozen (i.e. with `require_grad=False`) or made up of elements which are contained within the `opacus.SUPPORTED_LAYERS` dictionary. Entries include most standard layers such as `torch.nn.Linear` and `torch.nn.Conv2d`. However, recall from Section 4.1 that there are 3 sets of parameters for the VAE: the decoder parameters $\boldsymbol{\theta}$, the encoder parameters $\boldsymbol{\phi}$ and the data space noise parameters $\{\sigma_j^2\}$. The former two can be immediately captured within the Opacus framework as all network parameters are within supported layers. However, the $\{\sigma_j^2\}$ parameters are most naturally included using a basic `torch.nn.Parameter()` specification, which is not supported by Opacus. To circumvent this problem we use the idea that if the weights of a linear layer are fixed to zero, then $\mathrm{W}\mathbf{x} + \mathbf{b} = \mathbf{b}$. Specifically, we define $\{\log \sigma_j\}$ as the output of a linear layer with $\mathrm{W} = \mathbf{0}$ and allow the bias $\mathbf{b}$ to play the role of what would usually be specified via `torch.nn.Parameter()`. This edit allows the model to be readily compatible with Opacus while retaining the $\{\sigma_j^2\}$ parameters which are integral to the VAE model.

5.4.2 Opacus Per-Example Gradient Calculation

DP-SGD requires the computation of the contribution of each datapoint to the overall gradient of the loss with respect to learnable parameters ($\mathbf{g}_t(x_i)$ within Algorithm 1 of Figure 2). This is not a natural operation in PyTorch as most models require the gradient only once it has been aggregated over a batch of observations. As a result, Opacus relies on a relatively specialised PyTorch feature known as a *hook*. Running our VAE-based approach in Opacus in its standard form results in a `Warning` that the use of PyTorch's `register_backward_hook` does not appropriately capture all of the gradients required in our model and that PyTorch's `register_full_backward_hook` should be used instead. We follow this advice and therefore use a local version of Opacus with this edit in our codebase.

6 Experiments

6.1 Metrics

SDV provides a variety of metrics which allow quantification of synthetic data quality and privacy. More precisely, SDV provides three types of metrics, namely *detection* metrics, *distribution* metrics, and *privacy* metrics, detailed in turn below.

6.1.1 Detection Metrics

Detection metrics aim to communicate how easily synthetic data points can be distinguished from real data points. SDV attempts to quantify this by training either a logistic regression model (denoted as Logistic in tables) or support vector classifier (denoted as SVC in tables) to perform this task and reports an Area Under ROC Curve (AUC)-based metric with a value in $[0, 1]$. This metric takes value 1 for AUC ≤ 0.5 and $2(1 - \text{AUC})$ otherwise. If it is straightforward to distinguish the two data types, the AUC will be close to 1, resulting in an SDV metric output of close to 0. If the trained model is unable to distinguish the two types of data, the AUC will be close to 0.5, making the SDV metric output close to 1. Therefore, when evaluating synthetic data generators on detection, we desire generators with metric values closer to 1 rather than those with metric values closer to 0.

6.1.2 Distribution Metrics

Distribution metrics aim to quantify how similar the empirical distributions of the real and synthetic data are.

The GMLL (Gaussian Mixture Log-Likelihood) metric reports the log-likelihood of the synthetic data under a Gaussian Mixture Model fitted to the real data. The idea here is that if the real and synthetic distributions are similar, the likelihood of obtaining the synthetic data from the learnt model will be high, and vice versa. Hence, a higher performing synthetic data generator should exhibit a higher GMLL metric than a less well-performing generator.

The CS (Chi-Squared) metric reports the p-value (hence in $[0, 1]$) associated with performing a Chi-Squared test on the categorical columns of the synthetic data under the null hypothesis that that the synthetic data is drawn from the same distribution as the real data. As the p-value communicates the probability of obtaining data "at least as different" to the real data as the synthetic data, a large value is what is desired. This essentially means that there is insufficient evidence to suggest that the null hypothesis that the two distributions are the same should be rejected.

Similar to the CS metric, the KS (Kolmogorov-Smirnov) metric reports the p-value (hence in $[0, 1]$) associated with performing a Kolmogorov-Smirnov test on the continuous columns of the synthetic data under the null hypothesis that that the synthetic data is drawn from the same distribution as the real data. By the same arguments as for CS, a larger value communicates greater distributional similarity.

Cont KL (Continuous Kullback-Leibler) reports

$$\frac{1}{1 + \mathrm{KL}},$$

where KL is the Kullback-Leibler divergence between the synthetic and real data distributions over the continuous variables. This metric can therefore take values between zero and one with values becoming more desirable as they approach one. Disc KL (Discrete Kullback-Leibler) computes the corresponding quantity for the categorical variables.

6.1.3 Privacy Metrics

SDV's privacy metrics aim to empirically estimate the extent to which the privacy of sensitive variables within the original dataset is retained in the synthetic dataset. More specifically, they quantify how accurately the value of a sensitive variable within the real data can be predicted by a model trained on the synthetic data. For example, if the `sdv.metrics.tabular.NumericalMLP` method is chosen, a Multilayer Perceptron (i.e. standard feedforward neural network) is trained using the synthetic data to predict the numerical sensitive attribute (e.g. age) from a number of "key" attributes (e.g. medical biomarkers). Its ability to compute the sensitive attribute given the key attributes is then tested on the real data and a score computed. In the case of categorical sensitive variables, this is the probability of an incorrect prediction. In the case of continuous sensitive variables, it is the L_2 norm between CDF(real) and CDF(predicted), where CDF is the cumulative distribution function of the sensitive variable fitted to the real data. Therefore, the privacy metrics also lie in $[0, 1]$, with a larger value communicating greater privacy.

6.2 SUPPORT

SUPPORT (Knaus et al., 1995) is a dataset derived from a prognostic study of seriously ill patients containing contains 8873 entries with 14 covariates (age, sex, race, number of comorbidities, presence of diabetes, presence of dementia, presence of cancer, mean arterial blood pressure, heart rate, respiration rate, temperature, white blood cell count, serum sodium, and serum creatinine) and associated survival times (or censorship times). The dataset therefore contains a desirable mix of continuous and categorical variables. Furthermore, the dataset is widely used as a benchmark dataset in the context of survival modelling. For example, Katzman et al. (2018) use this dataset in their analysis and freely release the preprocessed version within the PyCox framework, which we make use of.

6.3 Results

6.3.1 Models

We evaluate the performance of the VAE approach alongside 5 alternative methods currently available, namely Gaussian Copula, CTGAN, CopulaGAN, TVAE and Independent. Gaussian Copula, CTGAN and CopulaGAN are as described in Section 3. TVAE refers to SDV's own implementation of a VAE following the description in Xu et al. (2019). Independent refers to our baseline, namely a model which assumes independence across variables, models continuous variables via Gaussian distributions and categorical variables via categorical distributions, and estimates the relevant sufficient statistics empirically. Evaluating this set of models provides context to the performance of the VAE with respect to both basic (e.g. Independent) and complex (e.g. CTGAN) approaches.

6.3.2 Quality and Privacy

The first question to address via experiments is how well a standard VAE performs in a quality and privacy sense as a synthetic data generator relative to the alternative methods. The second question to address is how the incorporation of differential privacy affects synthetic data quality and privacy as measured by other privacy metrics. This section addresses each of these questions in turn.

The first question can be addressed by attempting synthetic data generation from the SUPPORT dataset using each method and comparing overall performance using the metrics defined previously. Table 1 shows the detection metrics (see Section 6.1.1) for each of the methods considered. The first point to note is that in each case the AUC-based metric value associated with the SVC model is smaller than that associated with the Logistic model, i.e. the SVC is more successful at distinguishing between real and synthetic data. This is to be expected due to the greater expressivity of support vector classification relative to Logistic Regression and is reassuring to observe. It should also be noted that the SVC metric is therefore of the greatest importance as adversarial attacks will typically use highly expressive models (such as SVC), not logistic regression. The table therefore suggests that the VAE produces data which is the most difficult to distinguish from real observations, with the two GAN-based approaches achieving similar results. Predictably, the independent baseline performed poorly when exposed to the powerful SVC classifier, however performed surprisingly well with respect to the Logistic classifier, stressing the importance of evaluating detection using a suitably powerful model. A final Table 1 result to highlight is the relatively poor performance of TVAE relative to VAE. The source of this is not entirely clear, hence we only utilise SDV's TVAE implementation when generating its metric entries in Tables 1–3. For other VAE experimentation and evaluation we use our own implementation, represented as VAE in Tables 1–3.

Table 2 shows the distribution-based metrics (see Section 6.1.2) obtained by each of the generators. The first point of note with respect to this table is that each method actually performs relatively similarly according to these metrics. As will be seen in visualisations later in this section, this is not the case when evaluating the synthetic data qualitatively. For example, an inspection of this table would suggest that Independent is particularly desirable, however, we know by definition that it is unable to capture correlations.

Finally, Table 3 shows the privacy-based metrics (see Section 6.1.3). We use age as the sensitive attribute and all other variables as key attributes. Here we see that the VAE and independent models typically provide the greatest privacy according to these metrics. The large privacy metric values associated with the Independent method is perhaps unsurprising as there is no correlation structure within the data, reducing the amount of information able to be drawn upon to regress from the key to the sensitive features. The large values associated with the VAE are encouraging, as these suggest that the VAE, a model capable of capturing correlations, also demonstrates competitive privacy metrics.

Tables 1–3 provide quantitative metrics on the various aspects of similarity, however it is also of interest to present the synthetic vs real data comparison visually. A good choice for such a visual comparison is the Pearson correlation matrix plotted as a heatmap. In particular, it is of interest to visualise i) the real data's correlations, ii) the synthetic data's correlations and iii) the difference between them. This is shown for a cross section of the models in Figure 3, with i), ii) and iii) corresponding to the left, middle and right columns respectively. Also shown (in each of the right column plot titles) are two additional metrics for each model, namely the "Gower Distance" and "Correlation RMSE". The Gower Distance (Gower, 1971) quantifies the dissimilarity between two datasets, with a larger value communicating greater dissimilarity. Meanwhile,

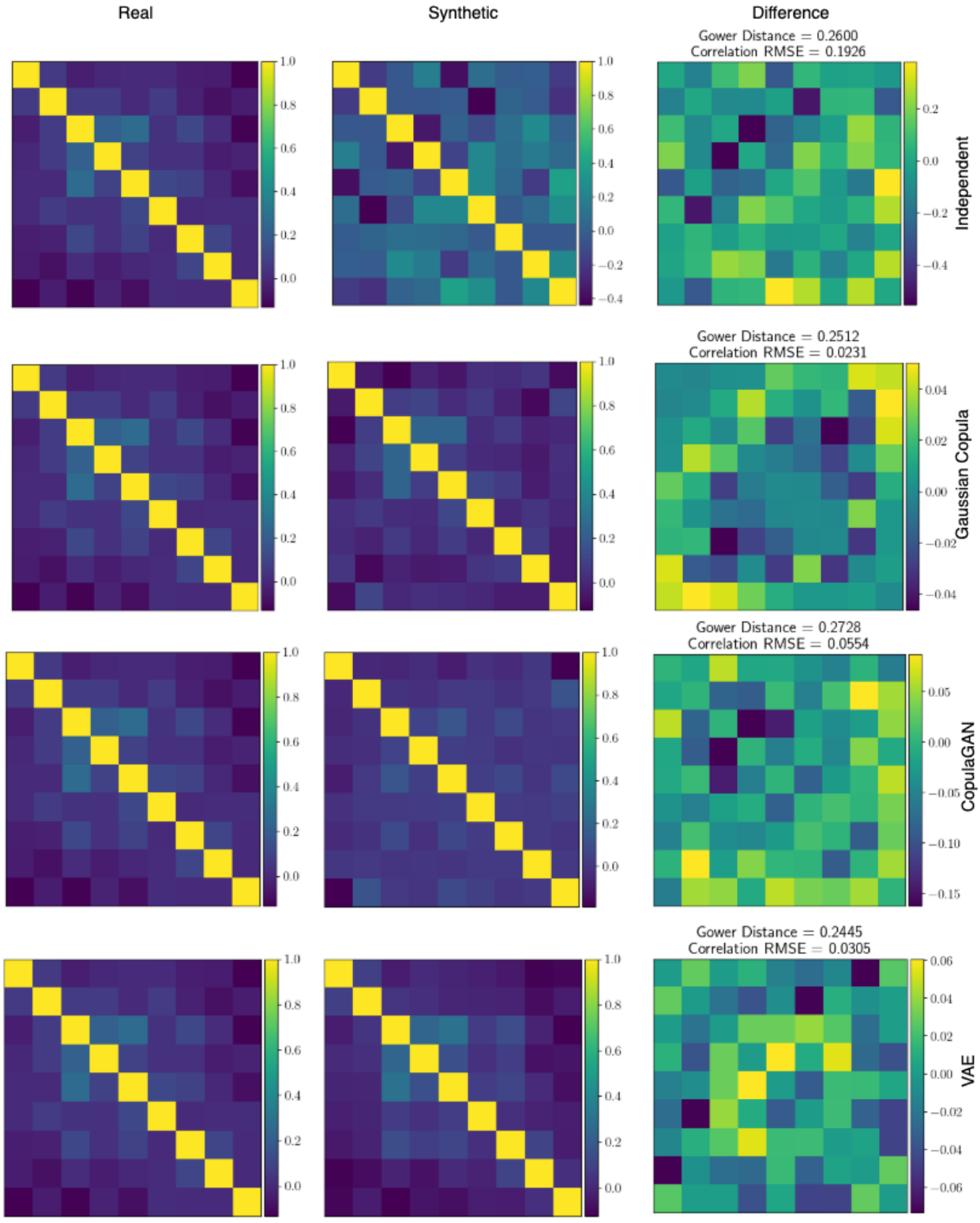

Figure 3: **Real vs Synthetic Data Correlation Structures.** A visual comparison of within-dataset correlations for real data and synthetic data generated from various models.

Correlation RMSE is the root mean square of the difference between the two correlation matrices, hence a larger value communicates greater disagreement between the real and synthetic data correlations.

Inspection of Figure 3 reveals that, as expected, Independent has the greatest Correlation RMSE, whilst Gaussian Copula has the smallest, with VAE demonstrating similar performance. Similarly, VAE exhibits the smallest Gower Distance, with Gaussian Copula performing comparably. Visually, it is also clear that whilst the three non-independent methods capture the important central correlation structure, greater similarity is exhibited by the Gaussian Copula and VAE models.

We now turn to the second key question laid out at the beginning of the section, namely how the introduction of differential privacy changes the behaviour of the VAE-based model. Of particular interest is how the metrics and visualisations evaluated in the previous question's analysis change as a function of differential privacy budget. We follow the convention of similar work (e.g. Abadi et al. (2016)) and fix δ (in this case to 10^{-3}) while allowing ϵ to vary. Tables 4–6 show the effect of varying ϵ on the key evaluation metrics. The first thing to note is that Table 4 behaves much as expected in that the SVC metric tends to increase with increasing ϵ up to a maximum at $\epsilon = \infty$, i.e. using standard gradient descent rather than DP-SGD. Table 5's distribution metrics largely follows this pattern also, i.e. they exhibit less desirable metric values at low ϵ, however with exceptions, e.g. a strong performance by the model with $\epsilon \approx 5$ in this case. Of particular interest is the effect of changing ϵ on the value of the privacy metrics. One would be expect that an ϵ decrease would lead to an improvement, and hence increase, in the privacy metric values. Table 6 does support this idea to a degree, for example at $\epsilon = \infty$ the privacy metrics are generally at their lowest. However it also exhibits unexpected behaviour, in particular that the metrics associated with $\epsilon \approx 1$ and $\epsilon \approx 2$ (the lowest ϵs considered) are not the largest, even though they generally exhibit the worst performance in Tables 4 & 5. This result could arise due to genuinely low adversarial robustness at low ϵ, however this is inconsistent with the motivation of DP, and a more likely alternative is that the SDV privacy metrics may not always quantify privacy appropriately.

Further insight is again provided by visualising the correlations within the real and synthetic data and comparing Gower Distance and Correlation RMSE values (see Figure 4). From this figure it can be seen that as ϵ is decreased from 10 to 1 the Correlation RMSE is largely unchanged but the synthetic correlations take on a more "washed out" appearance, with a larger number of entries exhibiting a small erroneous value. Meanwhile, the Gower Distance increases monotonically with each ϵ decrease.

6.3.3 Ease of Use and Interpretability

In the context of data generation within the NHS, it is also important to consider i) how readily the VAE-based tool could be utilised by users with varied familiarity with coding and ii) how interpretable and/or understandable the model and its behaviours are. This section considers these aspects in turn.

Given that any synthetic data tool adopted by the NHS would be of interest to users with a wide variety of programming familiarity, a conscious effort was made to run the aforementioned experiments in their most natural way, as would be done by a fresh user. In particular, where possible SDV algorithms were run with default settings, and the same simple feedforward network was used within the VAE implementation at all times. Consequently, the results presented in this report are representative of what would be obtained by the "typical" user, rather than one with unlimited time and coding experience. Indeed, in its current form, a user with a basic knowledge of Python and PyTorch could generate synthetic data of their own using any of the methods provided here with relative ease (generally by editing only a few lines of code, the location of which is highlighted within the relevant files). We recognise however that this still restricts access to those confident enough to use GitHub and run Python scripts. As synthetic data tools such as this move closer to the wider NHS community it will be important to consider in greater detail the default user interface and whether a GUI-based approach, potentially with AutoML aspects for additional methodological assistance, is a more viable option for the typical user than code-based APIs.

Advanced users may like to understand model performance and behaviour beyond basic metrics and visualisations as well as edit a model's default setup. Our VAE approach is particularly appealing is this respect. Firstly, it is easily customisable by an experienced Python user with regard to aspects such as network architecture, optimizer choice and generative model design. Whilst it is in principle possible to do the same with the alternative methods offered via the SDV framework, this requires editing the source code of the framework, adding an additional layer of complexity. In addition, note that compared to a GAN the

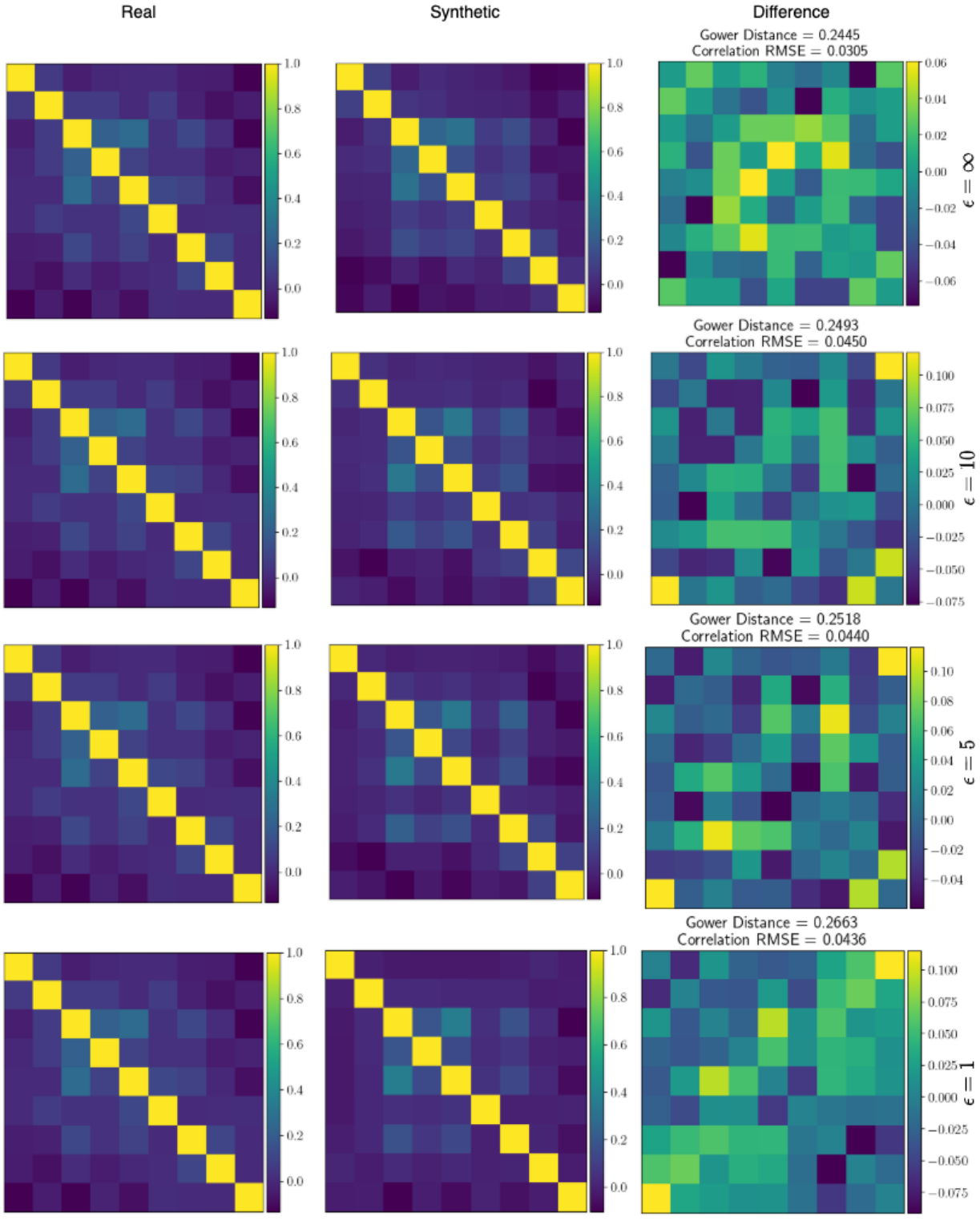

Figure 4: **Real vs Synthetic Data Correlation Structures.** A visual comparison of within-dataset correlations for real data and VAE-generated synthetic data with varied differential privacy budgets.

Table 1: **SUPPORT: Detection metrics.** Quantification of the difficulty of distinguishing synthetic and real data entries. Larger values mean harder to distinguish. Metrics are given as mean±std over 10 randomly seeded runs.

Method	Logistic	SVC
Gaussian Copula	0.4457 ± 0.0049	0.2687 ± 0.0043
CTGAN	0.5507 ± 0.0885	0.3464 ± 0.0466
CopulaGAN	0.5432 ± 0.0887	0.3474 ± 0.0436
TVAE	0.3023 ± 0.0450	0.0798 ± 0.0125
VAE	0.9282 ± 0.0280	$\mathbf{0.3555 \pm 0.0368}$
Independent	$\mathbf{0.9992 \pm 0.0012}$	0.0484 ± 0.0081

Table 2: **SUPPORT: Distribution metrics.** Quantification of the distributional similarity of synthetic and real datasets. Larger values mean greater similarity. Metrics are given as mean ± std over 10 randomly seeded runs.

Method	GMLL	CS	KS	Cont KL	Disc KL
Gaussian Copula	-34.36 ± 0.03	0.9407 ± 0.0016	0.9207 ± 0.0016	0.8987 ± 0.0015	0.7657 ± 0.0011
CTGAN	-34.18 ± 0.89	0.8996 ± 0.0235	0.8571 ± 0.0226	0.8864 ± 0.0358	0.9250 ± 0.0135
CopulaGAN	-33.78 ± 0.41	0.9005 ± 0.0222	0.8627 ± 0.0242	$\mathbf{0.8998 \pm 0.0170}$	0.9231 ± 0.0219
TVAE	$\mathbf{-30.91 \pm 0.19}$	0.8915 ± 0.0152	0.8314 ± 0.0169	0.8254 ± 0.0241	0.8408 ± 0.0236
VAE	-36.42 ± 0.09	0.9883 ± 0.005	0.8820 ± 0.0041	0.8693 ± 0.0059	$\mathbf{0.9817 \pm 0.0011}$
Independent	-33.94 ± 0.17	$\mathbf{0.9962 \pm 0.0017}$	$\mathbf{0.9592 \pm 0.0021}$	0.8444 ± 0.0053	0.9809 ± 0.0009

VAE has two additional aspects which can aid model understanding, namely i) an encoder which is able to associate each data point with a low-dimensional embedding and ii) an explicit generative likelihood model. The former can be used to understand latent factors within the data, whilst the latter can be used to enforce or encourage behaviour within the output distribution. In contrast, a GAN's discriminator does not provide i) as its only purpose is to discriminate between genuine and generated data. Moreover, a GAN's generator does not provide ii) as it does not have an associated likelihood due to it only ever having to be sampled from, not evaluated. We do not explicitly visualise the embeddings or significantly edit the generative model in this report, however this would be valuable future work.

7 Discussion

Throughout the report it has been clear that the VAE is a viable synthetic health data generator. Its likelihood-based nature and incorporation of a deep neural network allows for interpretable modelling of varied and complex data distributions, and its performance in Tables 1– 3 and Figure 3 compares favourably with alternative methods. Additionally it can be easily adapted to accommodate (ϵ, δ) differential privacy as demonstrated in the accompanying code repository. However, whilst differential privacy can be introduced easily, it was seen that restrictive privacy budgets led to reduced synthetic dataset quality. Furthermore, it was seen that the measure of privacy provided by DP did not always correlate directly with the observed level of privacy according to SDV's privacy metrics. This observation motivates further work. One natural follow-on question is whether the performance decrease associated with lower ϵs is worthwhile, and by extension whether DP is necessary at all or whether the noise inherent to a variational encoder in the $\mathbf{z}$ and $\mathbf{x}$ sampling steps is enough to disguise the real data. To answer this satisfactorily would require the development and

Table 3: **SUPPORT: Privacy metrics.** Quantification of the privacy of the synthetic datasets produced by different methods. Larger values mean greater privacy. Metrics are given as mean $\pm$ std over 10 randomly seeded runs.

Method	**LR**	**MLP**	**SVR**
Gaussian Copula	0.0736 ± 0.0245	0.0845 ± 0.0303	0.0749 ± 0.0248
CTGAN	0.0803 ± 0.0276	0.0911 ± 0.0324	0.0841 ± 0.0291
CopulaGAN	0.0836 ± 0.0288	0.0877 ± 0.0307	0.0869 ± 0.0299
TVAE	0.0841 ± 0.0291	0.0917 ± 0.0320	0.0880 ± 0.0298
VAE	0.0828 ± 0.0013	$\mathbf{0.1078 \pm 0.0430}$	$\mathbf{0.0987 \pm 0.0373}$
Independent	$\mathbf{0.0882 \pm 0.0351}$	0.1077 ± 0.0430	$\mathbf{0.0987 \pm 0.0372}$

Table 4: **SUPPORT: Detection metrics with a differentially private VAE.** Quantification of the effect of privacy budget on detection metrics for the VAE. Larger values mean harder to distinguish.

ϵ	**Logistic**	**SVC**
0.9994	0.4957	0.1453
2.0179	0.7567	0.1875
5.1455	0.6780	0.2561
9.9502	0.7092	0.2511
21.3812	0.7749	0.2742
∞	**0.9184**	**0.4018**

Table 5: **SUPPORT: Distribution metrics with a differentially private VAE.** Quantification of the effect of privacy budget on distribution metrics for the VAE. Larger values mean greater similarity.

ϵ	**GMLL**	**CS**	**KS**	**Cont KL**	**Disc KL**
0.9994	-36.12	0.7845	0.8888	0.8416	0.7344
2.0179	-36.14	0.8184	0.8914	0.8422	0.8139
5.1455	$\mathbf{-35.17}$	0.8333	**0.8979**	0.8556	0.8252
9.9502	-35.21	0.8349	0.8942	0.8534	0.8333
21.3812	-35.42	0.8598	0.8933	0.8568	0.8703
∞	-36.39	**0.9851**	0.8909	**0.8726**	**0.9806**

application of a range of more advanced adversarial attacks and metrics than those offered by SDV and the standard VAE to resist these effectively.

Another natural immediate continuation of the work presented here is to attempt to incorporate (ϵ, δ) DP into models other than the VAE and compare their behaviour as a function of ϵ with the aim of elucidating the extent to which each model is able to resist quality loss as ϵ is lowered. This would be done by accessing the internal PyTorch `optimizer`s associated with the SDV `fit` methods and attaching Opacus `PrivacyEngine`s. A linked question which could be investigated is whether the use of PATE, rather than DP-SGD, allows for greater performance retention as the privacy budget is tightened. It could also be beneficial to more explicitly demonstrate the interpretable aspects of a VAE, namely latent factor identification and generative model

Table 6: **SUPPORT: Privacy metrics with a differentially private VAE.** Quantification of the effect of privacy budget on privacy metrics for the VAE. Larger values mean greater privacy.

ϵ	**LR**	**MLP**	**SVR**
0.9994	0.0826	0.0858	0.0817
2.0179	0.0810	0.0825	0.0817
5.1455	**0.0863**	0.0877	**0.0825**
9.9502	0.0853	0.0869	0.0812
21.3812	0.0829	**0.1018**	0.0805
∞	0.0801	0.0826	0.0804

specification, whilst comparing to the GAN equivalent, that is probing GAN behaviour via the generator's decision boundary and discriminator's generative samples. Finally, it would also be informative to apply all methods to additional datasets with a variety of data types, data heterogeneity and missingness rates, with MIMIC-III an appropriate first choice.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA. Association for Computing Machinery.

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan.

Theodoros N. Arvanitis, Sean White, Stuart Harrison, Rupert Chaplin, and George Despotou. 2021. A method for machine learning generation of realistic synthetic datasets for validating healthcare applications. *medRxiv*.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large scale gan training for high fidelity natural image synthesis.

Rewon Child. 2021. Very deep vaes generalize autoregressive models and can outperform them on images.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

J. C. Gower. 1971. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871.

Jared L. Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. 2018. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes.

William A. Knaus, Frank E. Harrell, Joanne Lynn, Lee Goldman, Russell S. Phillips, Alfred F. Connors, Neal V. Dawson, William J. Fulkerson, Robert M. Califf, Norman Desbiens, Peter Layde, Robert K. Oye, Paul E. Bellamy, Rosemarie B. Hakim, and Douglas P. Wagner. 1995. The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122(3):191–203.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data.

Ally Salim. 2018. Synthetic patient generation: A deep learning approach using variational autoencoders.

Arash Vahdat and Jan Kautz. 2021. Nvae: A deep hierarchical variational autoencoder.

Zhiqiang Wan, Yazhou Zhang, and Haibo He. 2017. Variational autoencoder based synthetic data generation for imbalanced learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. *CoRR*, abs/1907.00503.

Jinsung Yoon, Lydia N. Drumright, and Mihaela van der Schaar. 2020. Anonymization through data synthesis using generative adversarial networks (ads-gan). *IEEE Journal of Biomedical and Health Informatics*, 24(8):2378–2388.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019a. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2019b. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.