Generating Synthetic Data Using Generative Adversarial Networks

ST449 Deep Learning and AI
21 May 2020

Contents

1 Introduction

In this project, we evaluate the use of Generative Adversarial Networks (GANs) to generate synthetic data from a database of patients' records. When generating synthetic data, one must first decide which characteristics of the real data need to be preserved in synthetic data. This is not always straightforward. Namely,

complex relationships in the data are often unknown or difficult to specify. That is the case with medical patient records which include diagnosis, treatment and treatment outcomes. It is for this reason that we decided to use GANs as they are generally effective in capturing complex data distributions in a variety of settings.

We measure the effectiveness of GANs by applying two criteria to their generated datasets:

1. Similarity of probability distributions between the generated data and real data (for example, using the Mean Square Error (MSE) of empirical probability distributions across variables in two datasets).
2. Similarity in prediction accuracy achieved on two datasets (the generated data and the real data) when pre-specified machine learning (ML) methods (e.g., Logistic Regression and Random Forest) are applied to predict values of a given variable from other variables in the data. Generative models that lead to lower difference in accuracy, e.g., lower MSE of accuracy scores, are considered more effective.

The dataset that we use to generate synthetic data consists of medical records of cancer patients. Each patient record contains categorical attributes describing patients and cancer tumours such as cancer type, stage, patient age, etc. In the generated synthetic data, we aim to preserve the following properties that we can estimate in the real data:

1. The proportions of all attribute categories; for example, the overall proportions of tumours at a given cancer stage, or the ratio of male to female patients, etc.
2. For a given cancer type, the proportions of attribute categories; for example, the proportions of lung tumours at each cancer stage, or the ratio of male to female patients who have a lung cancer, etc. These proportions can be estimated in the real data by calculating empirical distributions of the categories.
3. Unknown and complex relationships between attributes, that we do not explicitly specify.

The fact that the dataset includes only discrete data attributes poses a significant challenge in the use of GANs: a discrete output from the generator means that backpropagation from the discriminator loss to the generator parameters is not possible and, therefore, the generator parameters cannot be trained by gradient descent. In order to address this, we use a re-parameterisation method called the Gumbel-Softmax trick which is a continuous approximation of a discrete output and therefore enables backpropagation.

Our research questions relate to the construction of GANs and factors that may affect their performance:

RQ1. How effective GANs are in generating data that reflect the **general probability distribution** of variables, e.g., patient attributes.

RQ2. How effective GANs are in generating data that reflect the **conditional probability distributions,** given a specific cancer type. (Cancer type is used as an example of a specific variable, but the question applies to any other.)

For each questions, we explore the factors that influence the performance of GANs:

- Loss function. Comparison of GAN performance when using the standard loss and Wasserstein-1 loss.
- Number of hidden layers. Compare how the number of discriminator hidden layers affects performance.

Our analysis involves a detailed comparison of general and conditional GANs (conditional on the cancer type categories) based on achieved probability distribution in the generated data and prediction performance for variables using Logistic Regression and Random Forest. Our results demonstrate greater effectiveness of GAN models with Wasserstein-1 loss and gradient penalty (WGAN-GP) over standard GAN. This is confirmed both through comparison of probability distributions and the prediction accuracy using LR and RF on the two datasets (generated and real data).

In the following sections we first review the methods and provide information about the data used in the experiments. We then describe the experiment set up and experiment results. In the final section we discuss the possible extension of the project.

2 Background

2.1 Motivation for synthetic data

There are a number of reasons for generating synthetic data. First, it can increase utility of data that contains valuable information but cannot be made available publicly due to confidentiality issues. This is often the case in health care where data has patient identifiable information and, therefore, restricted access. By enabling broader use of synthetic datasets, with statistical properties similar to real data, more experts can analyse the data and contribute solutions to key problems. Second, in many situations, training effective ML models requires representative data that may not be readily available. Synthetic data can be created and used to augment training data for ML methods. Recent advances in the development of generate adversarial networks (GANs) make them a good choice for generating synthetic data.

2.1.1 Use of GANs to generate synthetic data

GANs have been primarily used for generating continuous data types. For example, deep convolutional GANs are used for generating images where each pixel in an image has a continuous value [1]. There have also been advances in using GANs for generating discrete sequential event data using recurrent neural networks with a Gumbel-Softmax activation layer [2]. MedGAN [3] is a GAN that generates patient data (using an auto-encoder) for binary and count variables in data that share the same 1 and 2 dimensional probability distributions with the real data.

Camino *et al.* [4] successfully generated multi-categorical data using a variety of GAN networks, including GAN, WGAN-GP, MedGAN and auto-encoder variants with Gumbel-Softmax activation. They trained these models on multiple

toy datasets that include categorical variables with different numbers of categories ranging from 1-10.

3 Methods

3.1 Generative Adversarial Networks (GANs)

The Generative Adversarial Network (GAN) is a generative model that learns the distribution of data p_{data}, by training two competing neural networks: a *generator* that aims to produce realistic data samples and to trick the *discriminator* and a discriminator that aims to distinguish between real data samples and the fake samples produced by the generator [5].

More specifically, the generator learns a distribution p_g through a mapping, $G(z)$ from some random noise z with a distribution p_z. It is trained to minimise the discriminator's ability to tell real output from generated, i.e., to minimise log((1-$D(G(z))$). The discriminator learns the probability $D(x)$ that a sample x came from the real data as opposed to the generated data and is trained to maximise its ability to distinguish real from generated samples log($D(x)$). The result is a minimax optimisation problem over the following objective function:

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

In this problem, when the discriminator is optimal, the optimisation of the generator is equivalent to minimizing the Jenson-Shannon Divergence between p_g, and p_{data} (see Section 3.2).

There are potential challenges when training GANs. First, there is no guarantee of convergence to the global optima through the gradient descent. Due to the opposing objectives of the generator and discriminator, it is possible that the objective reaches a "stable orbit". Second, there must be a balance between the strengths of the discriminator and generator throughout the training. If the discriminator learns to classify samples quickly, before the generator has learnt how to create realistic data, the generator is overpowered, i.e., its gradients diminish to zero and it ceases to learn. It is also possible for the generator to collapse to a setting where it generates realistic data but outputs the same data point rather than the full distribution of data. This is known as *modal collapse*.

3.2 Probability distribution distance definitions

We apply two commonly adopted distance metrics for probability distributions:

Jensen-Shannon (JS) divergence defined as:

$$\text{JS}(P_r, P_g) = KL(P_r \parallel P_m) + L{:}\,(P_g \parallel P_m) \quad (2)$$

Where P_m is the mixture (P_r+P_g)/2. JS divergence is always defined since we can set μ=P_m, and is symmetrical.

Earth-Mover (EM) distance or **Wasserstein-1**

The distance between two distributions P_r and P_g is defined as:

$$W(P_r, P_g) = \inf_{\gamma\in\prod(P_r,P_g)} E_{(x,y)\sim\gamma}[\|x - y\|] \quad (3)$$

where $\prod(P_r,P_g)$ is a set of all joint distributions $\gamma(x,y)$ with marginal P_r and P_g. Essentially, through $\gamma(x,y)$ we measure 'mass' that needs to be transported from x to y so that the distribution P_r can be transformed into the distribution P_g. In that respect, the EM distance represents a 'cost' that a transport by optimal γ will incur.

3.3 Wasserstein Generative Adversarial Networks (WGANs)

Arjovsky *et al.* [6] show that there are cases where JS divergence may not be continuous with respect to model parameters. As a consequence, it would not be possible for the generator to learn p_{data} by gradient descent. They propose an alternative loss to be minimised, the Earth-Mover (EM) distance, also referred to as Wasserstein-1 distance, as they found EM to be continuous in such cases. According to [6], Kantorovich-Rubinstein duality, EM distance is equivalent to:

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} E_{x\sim P_r}[f(x)] - E_{x\sim P_\vartheta}[f(x)] \quad (4)$$

with supremum taken over all 1-Lipschitz functions $f: X \rightarrow \mathbb{R}$. When we have a family of functions $\{f_w\}_{w\in W}$ that are all 1-Lipschitz, we could consider solving the max optimization problem:

$$\max_{w\in W} E_{x\sim P_r}[f_w(x)] - E_{x\sim p(z)}[f_w(G_\theta(z))] \quad (5)$$

The discriminator is now called a *critic*, as it is no longer classifying samples; rather it is learning the 1-Lipschitz function $f(x)$. In order to learn such a function, there is bound imposed on the critic output through clipping of its values.

3.4 WGAN with Gradient Penalty (WGAN-GP)

The weight clipping on the critic output can cause problems. For example, it may bias the generator model towards learning simpler distributions compared to the distribution of real data. Also, incorrect tuning of the clipping threshold c can lead to vanishing or exploding gradients. Thus, it is proposed to put a constraint on the gradients of the critic output in the form of a gradient penalty that also ensures the constraint [7]:

$$L = \mathop{\mathrm{E}}_{\tilde{x}\sim P_\vartheta}[D(\tilde{x})] - \mathop{\mathrm{E}}_{x\sim P_r}[D(x)] + \lambda \mathop{\mathrm{E}}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}})D(\hat{x})\|_2 - 1)^2] \quad (6)$$

where the first two terms refer to the original critic loss and the last term is the gradient penalty.

3.5 Conditional GAN

It is possible to train a GAN to learn the distribution of data conditional on a given variable y [9]. Such GANs are referred to as *conditional* GANs. Instead of learning the full distribution of the data, they learn the distribution conditional on a given input y. The resulting objective function for the conditional GAN is:

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[logD(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]\,.$$

The input variable y is concatenated with the random noise z for input into the generator and is subsequently concatenated with the output of the generator.

In our case we choose y to be cancer type since medical research often focuses on a specific cancer type. In practice this would allow users to specify the cancer type for which they would like to generate data, e.g., to specify x many records of

lung cancer patients to be created by feeding x vectors that represent lung cancer into the generator.

3.6 Gumbel-Softmax distribution

The discreteness of categorical data that is the output of the generator, poses a problem during training. To engineer an output that represents a categorical variable, in the final layer of the generator one could use a dense layer, of size of the number of categories in the variable, followed by an *argmax* transformation. This generator output could then be fed into the discriminator. However, the discreteness of the argmax transformation means that backpropagation is not possible and thus it is not possible to train the generator.

An alternative is to use a continuous approximation for the discrete layer, otherwise known as the Gumbel-Softmax trick.

The Gumbel-Max trick

The Gumbel-Max trick [8] provides an efficient way to draw samples z from a categorical distribution with class probabilities π

$$z = onehot\left(\arg\max_i [g_i + \log \pi_i]\right). \qquad (8)$$

Where $g_1,\ldots,g_k$ are independently and identically distributed samples drawn from Gumbel(0,1). The softmax function is used as a continuous and differentiable approximation of argmax.

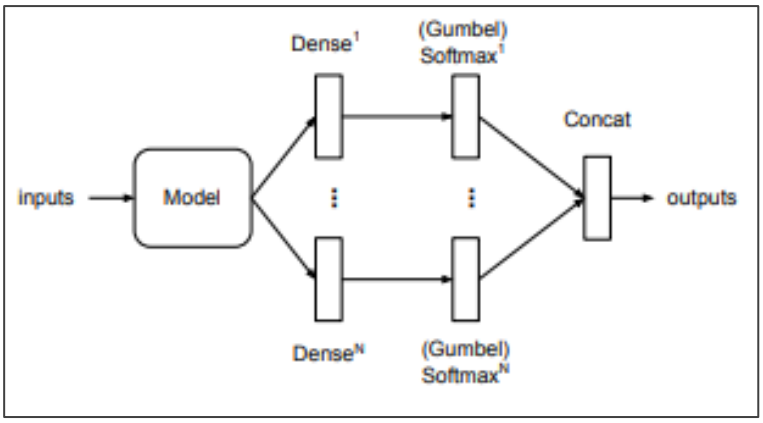

Figure 1: Final generator layer with Gumbel-Softmax activation

We generate k-dimensional sample vectors $y \in \Delta^{k-1}$ where

$$y_i = \frac{\exp(((\log \pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp(((\log \pi_j) + g_j)/\tau)}, i = 1, \ldots, k. \qquad (9)$$

The density of the Gumbel-Softmax distribution is given by

$$p_{\pi,r}(y_1, \ldots, y_k) = \Gamma(k)\ \tau^{k-1}\left(\sum_{i=1}^{k} \pi_i / y_i^{\tau}\right)^{-k} \prod_{i=1}^{k} \pi_i / \ y_i{}^{\tau+1}. \qquad (10)$$

4 Data

For the experiments we use a dataset obtained from the Simulacrum [9]. The Simulacrum is a synthetic dataset (generated by different methods from ours) that contains 1.4 million health records of patients with cancer tumours. The records have the same statistically properties and complexity as real electronic patient records held by Public Health England. There are 34 variables in total, all categorical and varying in size. The description of variables is included in Table 1. In total there are 116 cancer types.

For our data generation experiments we consider 29 out of 34 variables (Table 1), eliminating those that are unique identifiers, such as IDs, and dates, since they have a wide range of values. There is total 1,414 categories across 29 variables, ranging between 2 and 556 per variable. We apply one-hot encoding to each variable and concatenate the encoded variable into single vector for input into the discriminator.

In our experiments, we used 20% of the Simulacrum database (1,4M records) which amounts to 280,000 records. The size is restricted primarily due to the computational considerations and RAM constraints. We split the data into training and test datasets. The training set (80% of the sample) consists of 224,000 records.

5 Experiment Design and Evaluation

We compare the implement 4 GAN models: GAN, WGAN-GP, c-GAN, c-WGAN-GP, and compare their performance by evaluating their generated datasets. Each model is implemented by considering 1 and 2 discriminator hidden layers.

We take the same approach as Camino *et al.* [4] to ensure the multi-categorical output of the generator: the final layer comprises a dense layer for each categorical variable in parallel (with size equal to the number of categories of the corresponding variable) followed by a Gumbel-Softmax activation (Figure 1). The result is a one-hot encoded vector that represents a value of the categorical variable. These are concatenated together as the final output of the generator and have the same format as the one-hot encoded real data.

We evaluate the performance of each GAN by comparing the data samples generated by the generator with the real data, using the following metrics:

Average MSE (MSE_p) of empirical distributions across variables. We measure the difference in 1 dimensional probability distributions of each categorical variable. If the generator has captured the probability distribution of the data, the MSE value will be low.

Average MSE (MSE_cp) of empirical distributions given a cancer type. For within-cancer probability distributions, we measure the differences in 1 dimensional conditional probability distributions of variables. First, we split the data by cancer type, and then for each type and each variable (excluding cancer type) calculate the empirical distribution across categories for the real and generated data.

Average MSE (MSE_lr & MSE_rf) of prediction accuracy for variables. We compare the performance of Logistic Regression (LR) and Random Forest (RF) on the generated and real data for predicting each variable from all other variables

Table 1. Description of attributes considered in the data generation process. #Val refers to the number of distinct values for each variable, e.g., the number of distinct values for a category. Five variables indicated by // are not used in the experiment.

Variable	#Val	Description
'TUMOURID' //	1402817	Tumour ID
'PATIENTID' //	1,322,100	Patient ID
'DIAGNOSISDATEBEST' //	1,095	Diagnosis date
'SITE_ICD10_O2	556	Specific cancer type
'SITE_ICD10_O2_3CHAR'	116	General cancer type
'MORPH_ICD10_O2'	454	Tumour characteristics
'BEHAVIOUR_ICD10_O2'	10	
'T_BEST'	43	
'N_BEST'	22	
'M_BEST'	9	
'STAGE_BEST'	37	
'STAGE_BEST_SYSTEM'	8	
'GRADE'	7	
' AGE'	100	Patient age
'SEX'	2	Patient gender
'CREG_CODE'	8	Patient area code
'LINKNUMBER' //	1,322,100	ID link to other dataset

Variable	#Val	Description
'SCREENINGSTATUSFULL_CODE'	37	Screening status of the tumour
'ER_STATUS',	7	Breast cancer specific tumour characteristics
'ER_SCORE',	10	
'PR_STATUS',	7	
'PR_SCORE'	10	
'HER2_STATUS'	6	
'CANCERCAREPLANINTENT'	7	Cancer care plan intent
'PERFORMANCESTATUS'	11	General health of patient
'CNS'	13	Clinical nurse specialist
'ACE27'	9	Adult Comorbidity Evaluation-27 score
'GLEASON_PRIMARY'	6	Prostate cancer specific tumour characteristics
'GLEASON_SECONDARY'	6	
'GLEASON_TERTIARY'	4	
'GLEASON_COMBINED'	8	
'DATE_FIRST_SURGERY' //	1,558	Date of first surgery
'LATERALITY'	7	Tumour location (left, right etc)
'QUINTILE_2015'	5	Economic status of patient

in the data. If the generator has learnt the p_{data} well, then MSE of accuracy scores will be low.

More precisely, for each variable, we predict its values using all other variables in the data. This involves (i) splitting the dataset into training and test, (ii) training the model on the training set and (iii) evaluating the performance on the test set by computing the prediction accuracy score.

5.1 Visualisation of results

We present the comparison of probability distributions and prediction accuracies for variables in the real and generated data by showing scatterplots of corresponding statistics.

Table 2. Performance metrics for GAN and WGAN with GP generative models

	GAN				WGAN-GP			
	General		Conditional		General		Conditional	
Num of hidden layers (d)	1	2	1	2	1	2	1	2
MSE_p	0.006926	0.003852	0.008526	0.008003	0.000337	0.000364	0.000475	0.000141
MSE_cp	0.039976	0.028566	0.052108	0.04442	0.023247	0.015836	0.007957	0.004133
MSE_lr	0.051958	0.259183	0.275968	0.144916	0.034019	0.02449	0.023964	0.010692
MSE_rf	0.052775	0.251436	0.294099	0.150394	0.034909	0.025183	0.024577	0.013916

Each point on the probability scatterplot represent a single category of a variable. On the x-axis we plot the generated data probability distribution for a given category. On the y-axis is the same for the real data. Each point in the scatterplot has coordinates (P(category in gen. data), P(category in real data)). The closer the point is to the y=x line, the more similar 1 dimensional probabilities of generated and real data are.

Similarly, for each GAN output and for each ML algorithm, we present separate scatterplots of accuracy scores per variable. On x-axis we plot the prediction accuracy of a variable in the generate data. On the y axis is the same for the real data. Each point has coordinates (Acc(variable in gen. data), Acc(variable in real data)). Thus, the closer the point is to the y=x line, the more similar the results of the classifier.

We also compare empirical probabilities of individual cancer types observed in both the generated and real data. They are shown as bar plots

6 Experiment Results

All the GAN models are implemented using TensorFlow v2.0 in Google Colab. Each model was run for up to 60 episodes and stopped earlier to save time and resources only if, based on visual inspection of the log of losses over time, it was clear that the GAN had stopped training.

For each non-conditional GAN, a dataset was generated of the same size as the test data and for each conditional GAN, a data was generated of the same size and with the same proportions of cancer types as the test data. The generated datasets were then compared using the MSE evaluation metrics. The results are shown in Table 2.

6.1 Evaluation of probability distributions

6.1.1 Performance of standard GANs

We find that the standard GANs struggle to capture 1 dimensional probability distributions of individual variables. Table 2 shows that standard GANs consistently perform worse in comparison to WGAN-GP in terms of both general and conditional probability distributions, as measured by MSE_p and MSE_cp

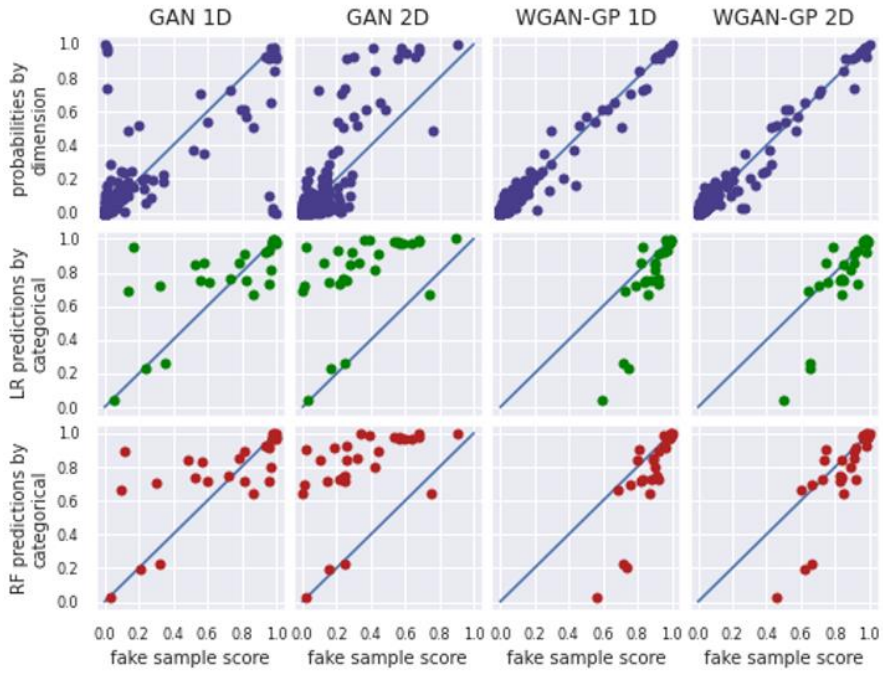

Figure 2: Scatterplots of (1) probability distributions of categories in generated and real data (top row) for 1 and 2 discriminator hidden layers for GAN and WGAN-GP and (2) accuracy scores for individual variables in generated and real data for LR (middle) and RF (bottom). The closer the points are to y=x the more similar outputs are.

across all the categories of observed variables. The difference between conditional GAN and WGAN-GP is more notable, showing that cWGAN-GP performs better, sometimes by a factor of 10 or more (cWGAN-GP with 2 layers has MSE_cp of 0.004133 compared to cGAN with 2 layers with MSE_cp of 0.04442).

This can also be seen from the scatterplots in Figure 2, top row, which shows probabilities of all the categories associated with 29 variables, both in generated and real data. Perfect alignment of points with line y=x would indicate that the empirical probability distributions of categories in both datasets are the same.

We get further insights about MSE_p and about MSE_cp per cancer type from the bar charts in Figure 4. The bar charts that GAN 1D and 2D generated data with lower frequencies of the most common cancer types than in real data. For example, GAN 1D generated very few cancer types of C50 (breast cancer), the second most common, and GAN 2D generates uniformly low numbers for 20 cancer types with highest frequency in real data.

6.1.2 Performance of conditional GANs

The conditional GAN shows lower performance in MSE_p and MSE_cp than standard GAN. This is clear from the scatterplot in Figure 3 (top row), where the points appear further away from the y=x line than in Figure 2. This is despite the perfect match of the cancer type empirical distribution which is guaranteed due to the design of the conditional GAN (shown in the bar charts in Figure 4). The low scores in MSE_cp per cancer type is shown in Figure 4.

It seems that already disadvantaged standard GAN, struggling to learn relationships between the variables from the full data distribution, is overwhelmed by adding an extra condition of cancer type. Its ability to learn decreases further.

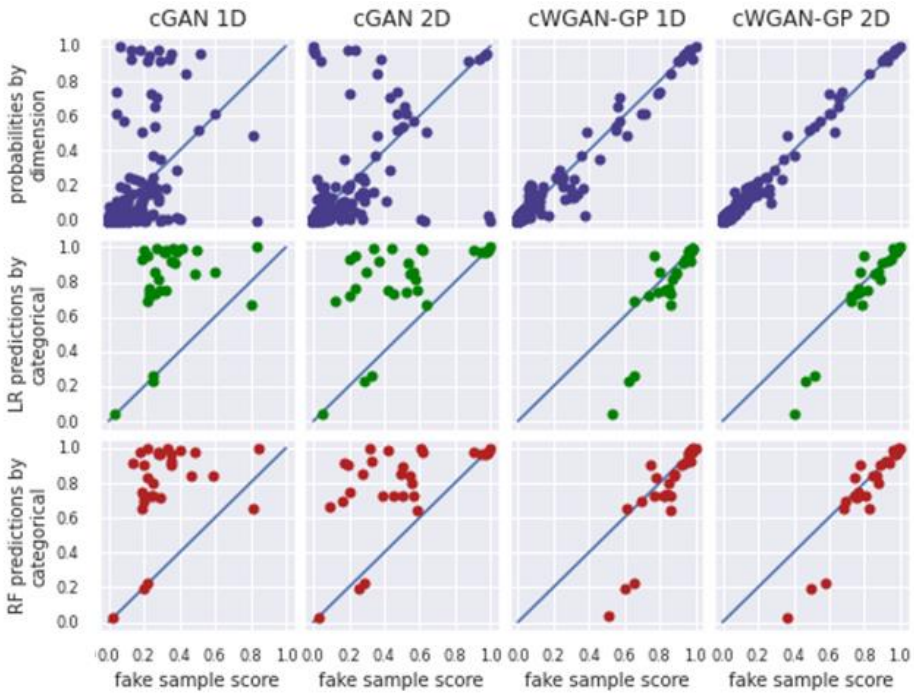

Figure 3: Scatterplots of (1) probability distributions of categories in generated and real data (top row) for 1 and 2 discriminator hidden layers for conditional GANs (cGAN) and conditional WGANs (cWGAN-GP) and (2) accuracy scores for individual variables in generated and real data for LR (middle) and RF (bottom). The closer the points are to y=x the more similar outputs are.

6.1.3 Performance of WGANs

In comparison to GANs, WGAN-GP perform better in terms of both MSE_p and MSE_cp scores. All scores improve even when the extra condition is added to the model (cancer type) and cWGAN-GPs are used. This is evident from the scatterplots in Figure 2 and 3 (top rows) showing that the category points lie very close to y=x line. The same is confirmed in Figure 5 where the frequency distributions for WGAN-GP 1D and 2D are very similar to the real data ones and the models achieve low MSE_cp metrics across cancer types.

Comparing WGAN-GP 1D and 2D, we note that 1D performs better. In case of cWGAN-GP the situation is opposite, i.e. 2D model performs better. That is not fully expected since, generally, one would presume that 2D layers would perform better because WGAN-GP relies on having a strong critic. Overall the cWGAN-GP 2D has the best MSE_p and MSE_cp scores.

6.2 Evaluation of accuracy scores

6.2.1 Prediction accuracy for GANs

We find that accuracy metrics MSE_lr and MSE_rf presented in Table 2 are poor for all GAN models. This is also evident from accuracy scatterplots in Figure 2 and 3 (middle and bottom rows). One can see that the majority of the points corresponding to the variables are located to the left of y=x line, indicating that the accuracy scores for predicted variables in the generated data are consistently lower than those in the real data. This suggests that GAN models may have failed

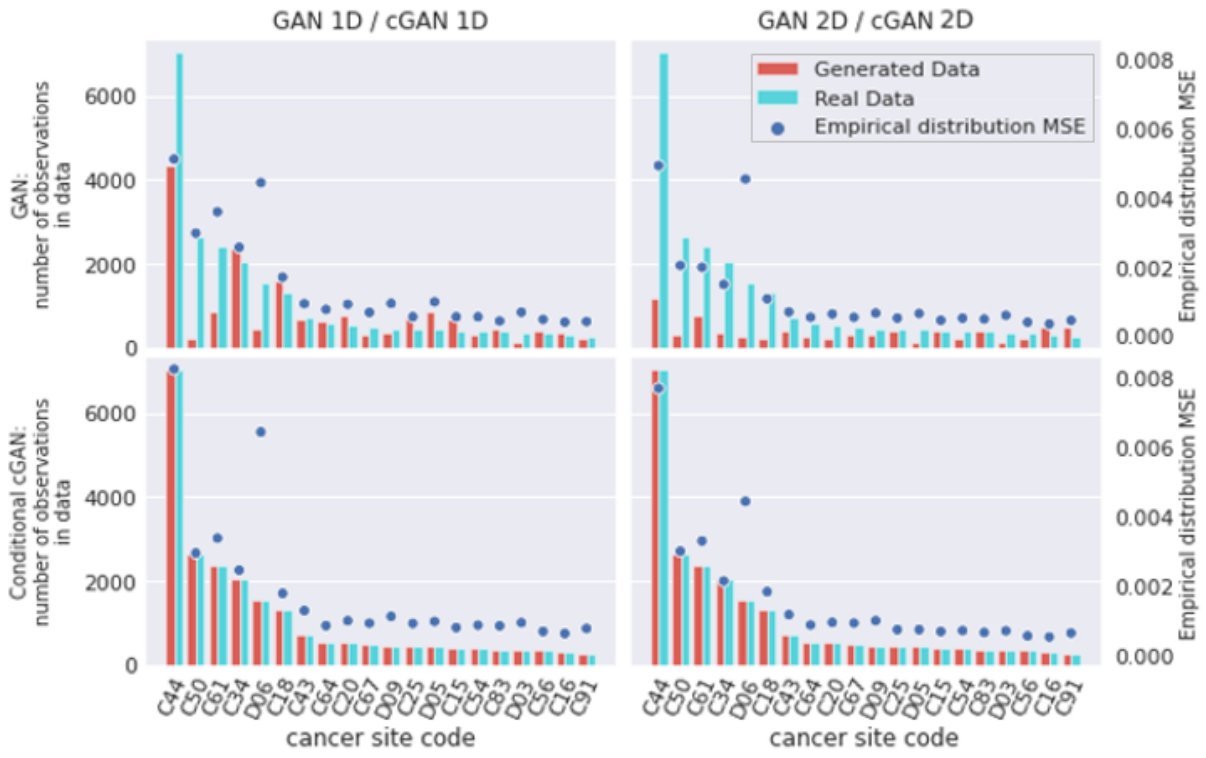

Figure 4: Probability Distribution results for GAN and cGAN for 1D and 2D. Each bar chart shows the observed frequencies of each cancer type in real (light blue bars) and generated data (red bars). For a given cancer type, it also shows the average MSE of probability distributions across all 29 variables.

to preserve some of the important relationships among variables that would potentially lead to better predictions using LR and RF.

6.2.2 Prediction accuracy for WGAN-GPs

All WGAN-GP models perform better than their corresponding GANs, with more than 10 fold improvement in MSE_lr and MSE_rf scores for most cases. This is seen in the scatterplots in Figure 2 and 3 (middle and bottom rows). The majority of points lie close to the y=x line, showing that the accuracy scores are similar for many variables.

It is interesting to observe that there are three outlier variables, located to the far right of y=x line, corresponding to much higher prediction accuracy in generated data than in real data. While this is not our objective, this may be useful in some situations, e.g., when augmenting training data to improve ML performance. Achieving such higher prediction accuracy than in real data suggests that WGAN changed the relationship of variables within data, possibly creating new ones that boost the prediction effectiveness of LR and RF.

Finally, we note that cWGAN-GP 2D model has the best MSE_lr and MSE_rf scores and, in essence, outperforms all other models across all MSE metrics.

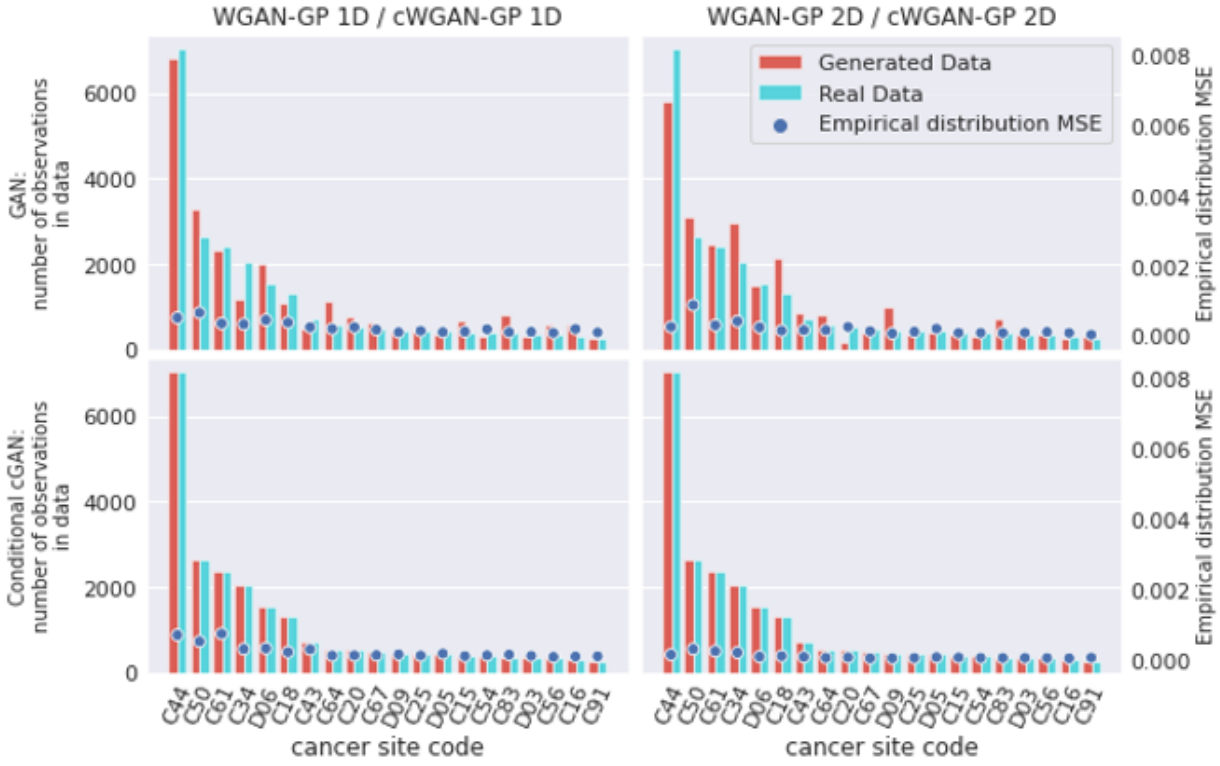

Figure 5: Probability Distribution results for WGAN and cWGAN for 1D and 2D. Each bar chart shows the observed frequencies of each cancer type in real (light blue bars) and generated data (red bars). For a given cancer type, it also shows the average MSE of probability distributions across all 29 variables.

7 Conclusion

In this project, we have demonstrated the use of general and conditional GANs for generating synthetic data from Simulacrum database that involves multi-categorical variables with varied numbers of categories, ranging from 2 to 556 per variable.

Based on comparison of empirical probability distributions in generated and real data, we conclude that standard GANs (GAN 1D, GAN 2D) do not produce data that score high on MSE metrics. In particular, they all failed to capture the 1 dimensional empirical distributions for individual variables (in the set of 29 used in the study). The same occurs when we compare empirical distribution of all variables in cancer type specific data. Furthermore, prediction of variable values using LR and RF was lower in the generated data, implying that the GANs failed to preserve relationships in the data that are important for predictions. When we considered conditional GANs (cGAN 1D, cGAN 2D) to focus on probability distributions conditioned on cancer type, the GAN's performance decreases further.

On the other hand, all WGAN-GPs performed relatively well, capturing 1 dimensional distributions and relationships among variables, as evident from improved MSE scores across the board. Overall the conditional WGAN-GPs perform the best of all the models according to all MSE scores. The best model for generating data was achieved by conditional WGAN-GP with 2D.

We expect that the performance of GAN models can be further improved through additional explorations of the hyperparameter space and alternative architectures, e.g., varying the number of hidden layers, using auto-encoders, and similar ([3],[4]). With the wide range of possible architectures, one can envision tailoring GANs for generating data with specific tasks in mind, e.g., data augmentation, or meeting specific requirements such as increased privacy guarantees.

7.1 Acknowledgement

"Data for this study/project/report used artificial data from the Simulacrum, a synthetic dataset developed by Health Data Insight CiC derived from anonymous cancer data provided by the National Cancer Registration and Analysis Service, which is part of Public Health England."

8 References

[1] Tanaka, F.H.K.D.S. and Aranha, C., 2019. Data augmentation using GANs. arXiv preprint arXiv:1904.09135.

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

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

[4] Camino, R., Hammerschmidt, C. and State, R., 2018. Generating Multi-Categorical Samples with Generative Adversarial Networks. arXiv preprint arXiv:1807.01202.

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

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

[7] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C., 2017. Improved training of wasserstein gans. In Advances in neural information processing systems (pp. 5767-5777).

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

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

[10] Simulacrum dataset: https://simulacrum.healthdatainsight.org.uk/