Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Econometrics and Management Science

An Evaluation Framework For Synthetic Medical Data

Supervisor:

O'Neill, E.P.

Author:

Jim Achterberg

481797

Second Assessor:

Gruber, K.

December 19, 2022

Abstract

This thesis presents a framework for evaluating synthetic medical data, which typically includes a combination of static and temporal variables of mixed data types. To assess the similarity of synthetic medical data to real data, we extend the tSNE algorithm to visualize sequences with mixed data types and use dynamic time warping and Gower distance to measure the distances between points. Additionally, we propose a new two-sample goodness-of-fit test for medical data that uses classification-based testing. We also introduce an interpretable metric for evaluating the privacy risk of synthetic data through attribute inference attacks. To demonstrate these evaluation methods, we apply them to synthetic medical data samples generated from two real-world datasets using a generative adversarial network, and a novel application of a probabilistic autoregressive network.

Contents

1 Introduction

In the past decade, there has been an increase in healthcare data breaches in the United States, resulting in the exposure of over 300 million healthcare records ("Healthcare Data Breach Statistics" 2021). These breaches can lead to the public release of sensitive medical information of individuals, prompting the search for new methods to preserve privacy in healthcare data. Traditional methods such as anonymizing and perturbing identifiable or sensitive attributes often still leave medical records at risk of re-identification (Henriksen-Bulmer and Jeary, 2016). One method that has gained popularity in recent years is the use of synthetic data, which is generated by machine learning models trained on real data, but is not traceable back to real individuals. This allows for the protection of sensitive information while still providing insight into realistic patient data.

The focus of this thesis is on the proper evaluation of synthetic medical data. When generating synthetic data to replace real data, three components should be considered: fidelity (similarity to real data), utility (performance in data-driven tasks compared to real data), and privacy risk (risk of violating the privacy of real individuals when distributing synthetic data).

Medical data often consists of both static and temporal variables, and can be both categorical and continuous in nature. For example, a patient in a clinical drug trial may have measurements taken on drug response and health factors over several weeks, as well as static attributes such as sex and ethnicity. In this thesis, we refer to this combination of static and temporal variables of mixed data types as medical data.

There is a lack of existing literature on the evaluation of synthetic medical data, either due to the use of simpler data or the use of evaluation methods that are not suitable for this type of data. This thesis aims to address this gap by presenting new methodologies for evaluating synthetic medical data. These include a new method for visualizing high-dimensional medical data in two dimensions, a new two-sample Goodness-Of-Fit (GOF) test called the Medical Goodness-Of-Fit (MGOF) test, and a new interpretable method for evaluating privacy risk when disclosing synthetic medical data.

We present these new methodologies in an evaluation framework and apply it on generated synthetic samples of two real-world medical datasets, generated using two easy-to-use deep learning software libraries.[1][2] This framework allows researchers in health institutions to easily generate synthetic samples of medical data and evaluate them for fidelity, utility, and privacy risk. The

[1]Library available at:`https://synthetics.docs.gretel.ai/en/stable/models/timeseries_dgan.html`

[2]Library available at:`https://sdv.dev/SDV/user_guides/timeseries/par.html`

models contained in these libraries consist of a Generative Adversarial Network (GAN), and a Probabilistic Autoregressive Network (PAN). This is a novel application of a PAN, since it has not yet been used to generate synthetic medical data.

Synthetic data is considered to have good fidelity if its distribution is similar to that of real data. To assess this, we use various methods including descriptive statistics, two-dimensional visualization of the empirical distribution, and GOF testing. We provide both static and temporal descriptive statistics to account for the mixed static and temporal nature of the data. To visualize the data in two dimensions, we use t-distributed Stochastic Neighbor Embedding (tSNE) with a modified distance metric suitable for mixed categorical and continuous sequences. Additionally, following Lee et al. (2020), Yoon, Jarrett, et al. (2019) and Desai et al. (2021), we apply a GOF metric for synthetic data using a classification model trained to distinguish synthetic from real samples. If the classification accuracy is close to 0.5, it indicates that the synthetic and real distributions are similar. With MGOF, we perform a novel two-sample GOF test for mixed static and temporal data of mixed types using this classification model.

To evaluate the utility of synthetic data, we assess its performance in data-driven tasks and compare it to real data. Following Yoon, Jarrett, et al. (2019) and Desai et al. (2021), we consider next-step forecasting tasks using the Train Synthetic Test Real (TSTR) and Train Real Test Real (TRTR) approaches. Here, we train a one-step ahead forecasting model on either synthetic or real data, and compare predictive performance on a real test set. If the predictive performance is similar for both approaches, it suggests that synthetic data can be used in place of real data without compromising performance while preserving privacy.

Finally, we investigate privacy risk of disclosing synthetic data through an Attribute Inference Attack (AIA). In this attack, we train a predictive model on synthetic data and attempt to infer sensitive attributes in a real test set. Since synthetic data is ideally similar to real data, such an attack can potentially be accurate, thereby successfully inferring sensitive information and harming privacy. Until now, AIAs were only applied in a real data setting where some individuals were willing to share sensitive data. We are now extending the use of AIAs to the synthetic data setting.

We generate and evaluate synthetic samples of two real-world medical datasets: one consisting of patient data on neuromuscular monitoring during surgeries, and the other consisting of data on a randomized control drug trial for cervical dystonia. Both datasets consist of both static and temporal variables, which are of both categorical and continuous types.

Our evaluation framework demonstrated good fidelity and utility in the generated synthetic samples of the cervical dystonia dataset. However, there was a reasonable risk of inferring the sensitive attribute of age. In contrast, the synthetic samples of the neuromuscular monitoring dataset displayed poor fidelity and utility, as well as a low risk of sensitive attribute inference.

The remainder of this thesis presents major findings from the literature in Section 2, followed by a description of the real data used to generate synthetic data in Section 3. Section 4 provides a detailed description of the methods used to generate and evaluate the synthetic medical data, and the corresponding results are presented in Section 5. Finally, the main conclusions of this study are presented in Section 6, with a discussion on possible improvements and further research in Section 7.

2 Related work

2.1 *Synthetic Data Generation*

Hernandez et al. (2022) provide a systematic literature review on research in synthetic tabular medical data, containing the most popular methods for synthetic data generation. The review distinguishes three approaches: classical approaches, deep learning models, and other approaches. Firstly, classical approaches relate to baseline methods such as anonymization or to probabilistic models which capture correlation structures in the original data, from which synthetic data is sampled. Also, it can relate to simple supervised machine learning approaches such as linear regression and logistic regression. Secondly, deep learning models relate to approaches using neural networks. The review distinguishes three types of models, namely Generative Adversarial Networks (GANs), auto-encoders and ensembles. Lastly, other approaches mainly consist of rule-based methods. These methods are based on predefining a set of rules about the data, based on knowledge of professionals. Then, based on some statistics and predefined rules, synthetic data can be sampled. From this review, it is clear that deep learning approaches are the most popular and successful in literature. This is the approach we use in this thesis.

We use two deep learning based software libraries to generate synthetic samples. Firstly, we use a GAN with DoppelGANger, from Lin et al. (2019). Secondly, we use a Probabilistic Autoregressive Network (PAN) with CPAR, from K. Zhang et al. (2022). First, we provide an introduction to GANs and PANs.

2.1.1 Probabilistic Autoregressive Networks

Autoregressive neural networks have a similar structure to regular neural networks. They consist of an input layer, one or more hidden layers, and an output layer. The crucial difference is that autoregressive neural networks pass p-lagged input values from the input layer to the first hidden layer. This allows for modelling temporal dependencies between current and previous inputs.

Whereas standard autoregressive networks model the next value in a sequence, PANs model the distribution parameters of the next value in a sequence. For continuous variables this could be the mean and variance of any continuous distribution (like Gaussian), for discrete values the stopping parameter and probability of success of the Negative binomial distribution, and so on.

To generate synthetic data, we can predict next-step distribution parameters using lagged inputs and randomly sample from a parameterized distribution. With a suitable sequence initialization, this method can produce entire synthetic sequences.

It should be noted that 'probabilistic neural networks' are used as an umbrella term in the literature. Generally, it is used to denote a neural network which approximates the (posterior) distribution of the target value. This way, uncertainty about the predictions is incorporated in the model. However, the way this is achieved can be considerably different. More sophisticated probabilistic networks, like variational autoencoders (Kingma and Welling, 2013), use variational inference to estimate the posterior distribution. In Specht (1990), probabilistic neural networks relate to neural network classifiers where the activation function is replaced by a density function to incorporate uncertainty. However in this thesis, we use the definition from K. Zhang et al. (2022): a neural network which makes parametric assumptions on the uncertainty of the target value - it follows a Gaussian or Negative binomial distribution - and estimates distribution parameters.

2.1.2 Generative Adversarial Networks

In short, a GAN (Goodfellow et al., 2014) can be described in a game-theoretic way as a game between two players: a generator and a discriminator. Here, the generator consists of a neural network with the aim to generate realistic synthetic data from random input. Crucially, the generating network only learns from interaction with the discriminator and does not obtain information directly through real data. The discriminator receives both real data and synthetic data produced by the generator. Here, the discriminator consists of a neural network with the aim to tell apart real data and synthetic data produced by the generator. The generator is successful if it adequately learns the distribution of the real data via interaction with the discriminator. In this case, the discriminator accurately classifies 50 percent of provided samples, corresponding to randomly guessing if provided data is real or synthetic. Figure 1 provides a schematic overview of a GAN.

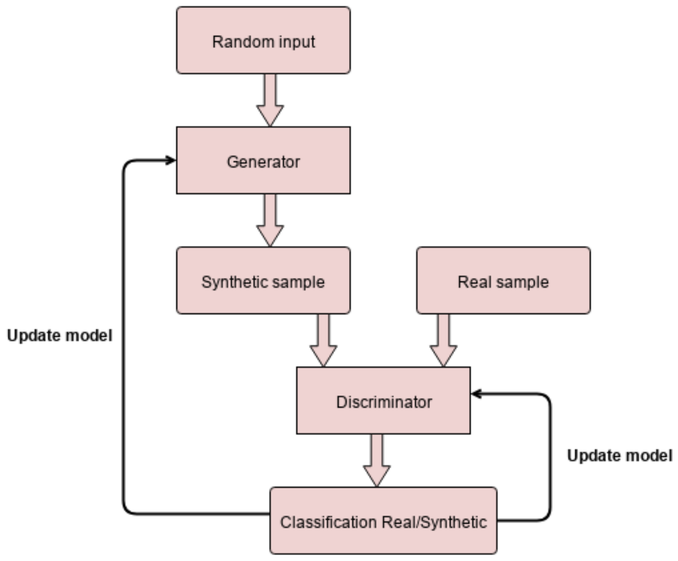


Figure 1: Schematic Overview of a Generative Adversarial Network

2.1.3 Synthetic Data Generation in Healthcare

GANs are already used extensively for generating synthetic healthcare data, for example tabular patient data. Choi et al. (2017), Baowaly et al. (2019), Torfi and Fox (2020) and Torfi, Fox, and Reddy (2022) generate synthetic discrete tabular patient data using a GAN combined with an auto-encoder. This way, the auto-encoder can decode continuous output of the GAN to a discrete space if the original features are discrete. Another popular model architecture is the conditional GAN. Here, the model conditions on class labels in order to learn separate distributions for, for example, continuous and discrete features. Yoon, Drumright, et al. (2020) and Rashidian et al. (2020) use this architecture to generate synthetic mixed-type tabular patient data.

However, a large part of patient data consists not only of static, but also of time-varying data. To capture temporal dynamics, a suitable model is needed. Esteban et al. (2017) generate synthetic continuous sequential patient data using a conditional GAN, capturing temporal dynamics by using

Recurrent Neural Networks (RNNs) for the generator and discriminator. Yoon, Jarrett, et al. (2019) generate synthetic mixed-type sequential patient data by training a GAN from an embedded space and adding stepwise supervised loss. This way, they reduce the dimensionality of the learning space and capture stepwise distributions of sequences. To capture temporal dynamics, the embedding function, generator and discriminator again employ RNNs. Lee et al. (2020) use a similar approach, by training a GAN from an embedded space and using RNNs in the embedding function, generator and discriminator. Crucially, they add a second GAN to adversarially train the embedding network to produce better samples. This way, the embedding network not only minimizes reconstruction error from the embedding, but is also guided to include realistic features.

2.1.4 Synthetic Data Generation in this Thesis

Crucially, this thesis requires joint generation of temporal sequences and static attributes of mixed data types. Firstly, Lin et al. (2019) design DoppelGANger: a GAN using RNNs in the generator to capture temporal dynamics, conditioning on static attributes. To generate these static attributes, the model contains a separate generator and discriminator. Pei et al. (2021) use DoppelGANger to generate synthetic medical data. Secondly, K. Zhang et al. (2022) design CPAR: a PAN, conditioning on static attributes. Static attributes are generated using Gaussian copulas. To our knowledge, no research yet exists on generating synthetic medical data using CPAR.

This thesis uses DoppelGANger and CPAR to generate synthetic samples. Firstly, this is because these methods are able to generate the correct type of data: both static and temporal variables, of categorical and continuous type. Secondly, these models are contained in easy and free to use software libraries, so researchers can quickly and easily create synthetic samples to test the evaluation framework provided in this research. Thirdly, these models are faster to train than many alternatives for these types of data. For example when implementing the models from Yoon, Jarrett, et al. (2019) and Lee et al. (2020), jointly training an embedding network and GAN turned out to be slow in practice. Lastly, the model architectures of DoppelGANger and CPAR have some desirable properties for generating a wide variety of synthetic data. We discuss this in Section 4.

2.2 *Evaluation*

2.2.1 Fidelity

To evaluate synthetic data fidelity to real data, investigating descriptive statistics like range, mean and standard deviation can be considered a first check on whether synthetic samples are viable. For example Rashidian et al. (2020) and Yang et al. (2019) investigate these properties, when generating synthetic healthcare records. Since this thesis generates synthetic temporal data, we also plot the mean of time-varying variables over time as a first check on whether stepwise distributions are captured.

The next indicator whether a synthetic sample is viable, is visualization of the synthetic versus real empirical distribution. Since the data is multidimensional, this requires some dimensionality reduction technique. Yoon, Jarrett, et al. (2019) and Desai et al. (2021) consider tSNE plotting of synthetic multivariate temporal data. This method, introduced in Van der Maaten and Hinton (2008), is a dimensionality reduction technique which excels at preserving both local and global structure, making it a good choice for visualizing high dimensional data. The method relies on converting data into a matrix of pairwise similarities. Then, samples are plotted in two dimensions, where distances between points reflect their similarity. This implies that standard tSNE cannot take three dimensional sequence data as input, since pairwise similarities will not result in a single matrix. To overcome this, Yoon, Jarrett, et al. (2019) and Desai et al. (2021) simply flatten the temporal dimension. However, this means differences in temporal dynamics across samples are not taken into account in the visualization. For this, sequences should be visualized as a whole. Furthermore, they do not consider mixed data types.

An algorithm for measuring similarity between entire sequences, is Dynamic Time Warping (DTW). DTW allows for elastic transformation of sequences across time, to detect similar shapes with possibly different phases (Senin, 2008). The algorithm calculates distances between sequences after aligning them to their similar shapes. Figure 2 shows the idea of sequence alignment.

In standard DTW, distances between sequences are calculated using Euclidean norm. However, this is not suitable for mixed data types, as the Euclidean norm between categorical vectors is ill-defined. A distance measure suited for mixed data types, is Gower distance (Gower, 1971). This is a weighted measure of categorical and continuous distance measures, between the individual variables of respective data types. Escudero et al. (2022) provide a software package for calculating sequence

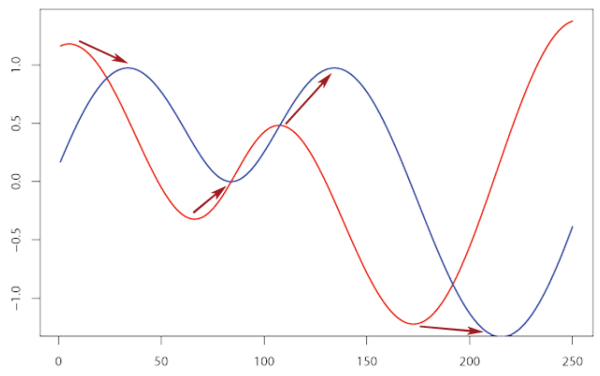


Figure 2: Alignment through Dynamic Time Warping. Arrows show desirable points of alignment (Senin, 2008).

similarity through DTW with Gower distance as distance metric.

Wong and F. L. Chung (2019) use DTW to calculate similarities in tSNE visualization of continuous sequences. However, no literature yet exists on using DTW with Gower distance in tSNE visualization for mixed-type sequences. This thesis provides a novel visualization method for medical data, by using DTW with Gower distance in tSNE visualization.

To further investigate fidelity of synthetic to real data, Lee et al. (2020), Yoon, Jarrett, et al. (2019) and Desai et al. (2021) train a classification model to discriminate between real and synthetic samples. If the model can classify samples accurately, this indicates poor fidelity: synthetic and real samples can easily be distinguished. If the distributions of synthetic and real data are equal, the best theoretical accuracy of the model is 0.5, corresponding to randomly guessing whether a sample is synthetic or real. However, there is no clear cutoff accuracy value to signify when synthetic and real distributions are equal.

To make a conclusion on whether two distributions are equal, one must perform a proper statistical test: a two-sample GOF test. This tests the null hypothesis of equal distribution between two samples. In the synthetic data literature, only univariate GOF testing per feature is considered (Baowaly et al., 2019, Yoon, Drumright, et al., 2020, Dash et al., 2020). However, this ignores the multivariate structure of the data, and thereby the fact that we try to generate indistinguishable samples rather than features. So, to further evaluate synthetic data fidelity, we wish to perform a multivariate two-sample GOF test.

There are extensions of standard nonparametric univariate two-sample GOF tests, to the multivariate setting. Biswas and Ghosh (2014) provide a test for multivariate tabular data, where the test statistic is based on the difference in inter-point differences between two samples. Li and Y. Zhang (2020) provide another test for multivariate tabular data, based on a projective ensemble approach. This approach relies on the idea that if two samples are equally distributed, projections of these samples are as well. Lastly, Friedman (2004) also provides a test for multivariate tabular data, based on testing binary classification scores of two samples. This test relies on the idea that if two samples come from the same distribution, they are scored in an equal manner by a (machine learning) classifier.

The notion of Friedman (2004) can be used to construct a GOF test for medical data. Crucially, this requires a suitable classifier, which can score samples of mixed-type temporal sequences and static attributes. Intuitively, this is the same model required for the discriminative scoring evaluation, earlier in this Section. RNNs are flexible models, suited for this task. Since this is a novel test, we provide a Monte Carlo style simulation to verify the size and power of this test in Section 4.5.3.

2.2.2 Utility

To evaluate utility of synthetic data, we investigate performance in data-driven tasks compared to real data. For example, Yoon, Jarrett, et al. (2019) and Desai et al. (2021) consider forecasting of temporal data. They predict next-step temporal vectors using RNNs, trained on synthetic data and tested on real data (TSTR approach). When the predictive model is accurate in predicting temporal vectors using the TSTR approach, this shows good fidelity of synthetic data. Since then, synthetic data has accurately captured stepwise distributions. Also, this shows usefulness in practice: synthetic data can be used to train a predictive model, so no exposure is necessary before making predictions using real data. This reduces privacy risk. We compare forecasts made using the TSTR approach, with forecasts made with the TRTR approach. If forecast accuracy between these approaches is equal, synthetic data can be used to replace real data in predictive modelling without losing accuracy.

2.2.3 Privacy risk

Lastly, we evaluate privacy risk introduced by distributing synthetic data. In this research, we define privacy risk as the risk of sensitive attributes from real samples becoming known to some third party. The goal of synthetic data is to replace real data, so real sensitive attributes can never become public due to negligence by an institution or a cyber-attack by some malicious party. However, since synthetic data tries to resemble real data, making synthetic data available may still give opportunity to malicious parties to find out the identities or identifiable attributes of real individuals.

N. Park et al. (2018), Norgaard et al. (2018), Rashidian et al. (2020) and Yale et al. (2019) consider some similarity measure between individual synthetic and real samples, as privacy risk evaluation. Here, if a synthetic sample is within some specified range of similarity to a real sample, they conclude that privacy is not well preserved. Since when similarity between a synthetic and real sample is high, sensitive attributes of real individuals are practically being disclosed. However, we argue that this evaluation is not thorough enough. Even when no single synthetic sample is too close to a real sample, malicious parties can potentially infer sensitive attributes through predictive modelling.

Membership Inference Attacks (MIAs) are another popular evaluation method for privacy risk. Hayes et al. (2019) presents MIAs, as determining whether a given set of individuals was used in training a model. Z. Zhang et al. (2022) and Yale et al. (2019) use MIAs as a way to evaluate privacy preservation capabilities of synthetic medical data. If MIAs can successfully determine which individuals were used to train a model to generate synthetic data, privacy is not well preserved.

MIAs rely on a sample containing full information being available to an attacker. However, a more immediate threat to privacy is a set of non-sensitive variables being used to infer sensitive attributes. Since, non-sensitive information on individuals is more likely to be available to the public. An attack where non-sensitive variables are used to infer sensitive attributes, is an AIA. In AIAs, some malicious party trains a predictive model on a sample containing non-sensitive variables, and sensitive targets. Then, it predicts sensitive attributes in another sample. Finally, privacy risk can be inferred from the accuracy of these predictions: high accuracy means high privacy risk, since sensitive attributes can accurately be inferred.

AIAs have been used in a variety of domains. For example, Gong and B. Liu (2018) evaluate privacy risk in social network data using AIAs, and Wu et al. (2020) in health data. In these datasets, only part of the individuals is willing to disclose sensitive information. This gives reason for using AIAs to measure the risk of sensitive attribute inference, by training a model on the individuals willing to disclose sensitive information and predicting for those who are not. However, little research has been done on whether synthetic data can be used in AIAs. If AIAs using synthetic data are accurate in inferring sensitive information, this can be a reason not to make synthetic data available.

In this thesis, we perform an AIA by training a predictive model using temporal explanatory variables, and sensitive static target variables. To measure privacy risk of synthetic data, we employ the TSTR approach. However, when accuracy of an AIA is low, this can be due to either: low predictability of sensitive attributes in real data, or poor preservation of correlation between static attributes and temporal sequences in synthetic data. In the latter case, there is a clear tradeoff between fidelity and privacy risk. Low accuracy in an AIA due to poor preservation of correlation, signifies poor fidelity. To investigate this, we compare the TSTR approach with the TRTR approach. If accuracy of an AIA is high with TRTR and low with TSTR, this indicates poor fidelity. If accuracy is low in both, this indicates poor predictability of the sensitive attribute, and low privacy risk. If accuracy is high in both, this indicates significant privacy risk.

3 Data

As mentioned in Section 1, this research considers a combination of static attributes and temporal sequences of mixed data types. In general, sequences and attributes can be both categorical and continuous. However, in the datasets used in this thesis, sequences are all continuous and static attributes are both continuous and categorical.

3.1 *Neuromuscular Monitoring*

We use an open-source dataset on neuromuscular monitoring, collected during a clinical trial at the University Hospital of Brussels (Verdonck et al., 2021).[3] Data consists of static patient attributes, and measurements on muscular activity of patients under general anesthesia during the course of an operation. We will later abbreviate this dataset as NM.

The dataset contains $N = 136$ patients, with a total of 21891 intra-operative observations. There is large variability in the amount of observations per patient. However, the methods used in this research can only handle fixed-length sequences. Therefore, we drop all samples of patients with less than $T = 50$ observations and cutoff all other sequences at $T = 50$, resulting in $N = 128$

[3]available at: `https://www.kaggle.com/datasets/michalverdonck/neuromuscular-monitoring-data`

patients. This seems like quite a small sample size for training neural networks. We discuss this issue in Section 4.8, after presenting the methods used in this thesis. Furthermore, we drop all engineered features and only consider real measurements. The engineered features contain no new information, as they can easily be constructed from the other variables. Furthermore, the dataset contains 8 numerical time-varying features, and 2 numerical and 2 categorical static attributes.

Table 1: Descriptive Statistics of Neuromuscular Monitoring Dataset

	Temporal	Type	Min	Max	Mean	Median	St. Deviation
Age	No	Num	12.000	95.000	54.055	56.000	20.635
BMI	No	Num	16.110	47.860	26.575	26.175	5.780
Sex	No	Cat	0	1	0.594	1	0.491
typeStudy	No	Cat	0	1	0.367	0	0.482
ExpSevo	Yes	Num	0.000	6.800	1.144	1.400	0.876
InspSevo	Yes	Num	0.000	8.000	1.426	1.760	1.091
TOF	Yes	Num	0.000	109.800	18.763	0.000	32.948
Count	Yes	Num	0.000	4.000	2.081	1.000	1.456
Esmeron	Yes	Num	0.000	80.000	0.832	0.000	5.740
Temp	Yes	Num	34.000	38.300	36.293	36.300	0.468
T1	Yes	Num	0.000	119.500	23.423	22.700	22.809
Bridion	Yes	Num	0.000	160.000	0.072	0.000	2.947

This table displays descriptive statistics of the Neuromuscular Monitoring dataset. Numerical variables are abbreviated to Num, and categorical variables to Cat.

The NM dataset contains eponymous sensitive static attributes on age, Body Mass Index (BMI) and sex (male is 0, female is 1), and an attribute on the type of study (0 is retrospective, 1 is prospective). Furthermore, it contains time-varying features on concentration of narcotic gas in inhales and exhales (**ExpSevo** and **InspSevo**), muscle twitch responses (**TOF**, **Count**, and **T1**), administered muscle relaxer and reversal agent of muscle relaxer (**Esmeron** and **Bridion** respectively), and patient temperature (**Temp**). Here, numerical variables **Age, Count, Esmeron**, and **Bridion** are discrete, the rest are continuous.

Note that only the CPAR model handles continuous and discrete numerical variables differently. For all other methods, all numerical variables are assumed to be continuous. For this reason, when this thesis speaks on mixed data types, it mostly uses the terms continuous and categorical.

Looking at the range and median of **Esmeron** and **Bridion**, these variables are likely highly sparse. Indeed **Esmeron** contains around 97% zeros, and **Bridion** over 99%.

3.2 *Cervical Dystonia*

We also use an open-source dataset on a drug (botulinum toxin type B) for cervical dystonia, collected from a randomized controlled trial at nine U.S. sites (Davis, 2002).[4] Data consists of static attributes, and measurements on a rating scale for severity of cervical dystonia during the trial. We will later abbreviate this dataset as CD.

The dataset contains $N = 109$ patients, with a total of 632 intra-trial observations. There is small variability in the amount of observations. We drop samples of patients with less than $T = 5$ observations, and cutoff all other sequences at $T = 5$, resulting in $N = 105$ patients. The sequences in this dataset are considerably shorter than in the NM dataset. This will likely make it easier for the data generating models to capture temporal dependencies. Furthermore, the dataset contains 1 numerical time-varying feature, and 1 numerical and 2 categorical static attributes.

Table 2: Descriptive Statistics of Cervical Dystonia Dataset

	Temporal	Type	Min	Max	Mean	Median	St. Deviation
Age	No	Num	26.000	83.000	55.581	56.000	12.147
Treat	No	Cat	1	3	1.990	2	0.811
Sex	No	Cat	0	1	0.371	0	0.483
TWSTRS	Yes	Num	6.000	71.000	40.731	42.000	12.633

This table displays information and descriptive statistics of the Cervical Dystonia dataset. Numerical variables are abbreviated to Num, and categorical variables to Cat.

The CD dataset contains eponymous sensitive static attributes on age and sex (female is 0, male is 1), and an attribute on the treatment type (**Treat**: placebo and two different drug dosages). Lastly, it contains a single time-varying feature on a score measuring severity, pain, and disability of cervical dystonia (**TWSTRS**). Numerical variables **Age** and **TWSTRS** are discrete.

The previous notes on sparsity in variables should be regarded when investigating descriptive statistics of synthetic data. We discuss data preprocessing in Section 4, since this may differ in data

[4]available at: `https://hbiostat.org/data/repo/cdystonia.html`

generating models and models used in the evaluation framework.

4 Methodology

This Section describes the methods for generating synthetic data, as well as the evaluation methodologies proposed in this research. First we describe methods used for generating synthetic data, then we describe methods for evaluation.

4.1 *Generative Adversarial Networks*

Section 2.1.2 illustrates that GANs should be seen in a game-theoretic way, as a game between a generator and a discriminator. Goodfellow et al. (2014) first introduces GANs, and describes the standard form in the following way.

The generator aims to generate synthetic data following the distribution of data $\mathbf{x}$, from noise $\mathbf{z}$ with prior distribution $p_{\mathbf{z}}(\mathbf{z})$. Now, $G(\mathbf{z}, \theta_g)$ is some differentiable function mapping $\mathbf{z}$ to the data space, where θ_g are the function parameters. Here, $G(\cdot)$ represents some neural network. Here, the choice for $G(\cdot)$ is crucial. For example, in standard GANs it can be a Multi-Layer Perceptron (MLP), in GANs for synthesizing images it can be a convolutional neural network, in GANs for time series it can be an RNN. The choice in network architecture depends on the task at hand.

The discriminator aims to assign the correct label to training data and data generated by $G(\cdot)$. Now, $D(\mathbf{x}, \theta_d)$ is some differentiable function that determines whether $\mathbf{x}$ is from training data, or from the generator distribution p_g. Again, $D(\cdot)$ represents some neural network, where the network architecture depends on the task at hand. Training network $D(\cdot)$, maximizes the probability of assigning correct labels to synthetic and real samples. Simultaneously, training network $G(\cdot)$ minimizes $\log(1 - D(G(\mathbf{z})))$. This results in a two-player minimax game, with objective function:

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

Section 2 shows GANs have been largely successful in generating synthetic medical data. However, training of standard GANs can be notoriously unstable, resulting in issues like vanishing gradients and mode collapse (Creswell et al., 2018).

As the discriminator becomes accurate at classifying samples, the generator's output is consistently rejected, resulting in the gradient of the loss function with respect to the generator's parameters becoming zero (vanishing gradient). This way, the generator is unable to learn to produce better samples.

In mode collapse, the generator has learned to output a small variety of highly plausible samples which the discriminator cannot accurately classify. Although fooling the discriminator is the goal of the generator, generating only a small variety of samples is undesirable in synthetic data. A realistic synthetic dataset covers the entire range of the distribution of the training data.

Multiple solutions have been provided in the literature to mitigate vanishing gradients and mode collapse in GANs. Among the most popular are altering the loss function to a Wasserstein loss function (Arjovsky et al., 2017), and 'unrolling' the alternating training process (Metz et al., 2016). The Wasserstein loss function is more likely to be differentiable than the standard GAN loss function, thus mitigating the vanishing gradient problem. Furthermore, this causes the discriminator to be able to be trained more optimally without worrying about vanishing gradients. Then, the discriminator can learn to negate samples with low variety (mode collapse). We provide a more in-depth explanation of the Wasserstein loss function in Section 4.2.4. 'Unrolling' the training process entails generating multiple samples from the generator, and back-propagating over all samples using only a single discriminator. The discriminator only back-propagates over the first sample. This way, the generator is able to learn more quickly relative to standard GANs, and it becomes less likely that all generated samples are rejected by the discriminator. Hereby it mitigates the vanishing gradient problem.

4.2 *DoppelGANger*

This research generates synthetic samples using DoppelGANger proposed in Lin et al. (2019). DoppelGANger makes changes to the standard GAN architecture to capture temporal dependencies, correlations between static attributes and time-varying features, model mixed-type variables and mitigate training instability issues in GANs.

4.2.1 Capturing Static-Temporal Correlations

Lin et al. (2019) find that standard GANs can not adequately capture correlations between time-varying features, and static attributes. To alleviate this issue, DoppelGANger uses a conditional GAN architecture. Here, generation of static attributes is decoupled from time-varying features by separate generators (attribute generator and feature generator). Then, the feature generator can condition on generated static attributes. Lin et al. (2019) shows that by decoupling feature and attribute generation in this way, correlations between static attributes and time-varying features are captured better in synthetic data.

4.2.2 Capturing Temporal Dependencies

As feature generator, DoppelGANger uses an RNN to capture temporal dependencies. Specifically, it uses Long Short-Term Memory (LSTM) networks. These are a type of RNN which contain a memory cell, to store information on past network outputs. The memory cell contains three gates: input gate, output gate and forget gate. The output gate modulates the amount of exposure to the memory (past instance of the model), the forget gate modulates the extent to which the existing memory is forgotten, and the input gate modulates how much new content (due to new data) is added to the memory (J. Chung et al., 2014). This memory cell allows for capturing longer term temporal dependencies than standard RNNs, in which past network outputs are taken as input but not stored as memory. However, this memory cell considerably increases the amount of model parameters, influencing computational costs.

DoppelGANger uses a Wasserstein loss function to improve training stability. This influences the choice for the discriminator network. At the time of writing, leading software packages do not include tools to calculate second order derivatives of this loss function, which is necessary for loss optimization in RNNs. For this reason, the discriminator network is an MLP.

DoppelGANger uses an MLP for the attribute generator and discriminator network. No RNN is necessary here, since attributes are static.

Lastly, to further capture temporal dependencies and output more realistic samples, DoppelGANger outputs batched instead of single timesteps. The recurrent network in the feature generator can output sample $X^i_j, \ldots, X^i_{j+S}$ instead of only X^i_j. Here, S is a tunable parameter, where total sample length must be divisible by S. When outputting batched steps, the MLP discriminator can negate samples which poorly capture temporal dependencies. This is not possible when outputting single steps, since MLPs do not take past model output as input.

4.2.3 Modelling Mixed-Type Variables

To model mixed-type variables, DoppelGANger uses appropriate activation functions in the output layer of the neural networks. Here, the activation depends on the data type we wish to generate:

- **Continuous variables**: sigmoid activation. This function outputs a continuous value in the $(0, 1)$ range, so is appropriate after $[0, 1]$ scaling of the data. Another choice is tanh activation, with $[-1, 1]$ scaling.
- **Categorical variables**: softmax activation. With K classes in a categorical variable, this function normalizes input vectors to a probability distribution of K probabilities, corresponding to the confidence that a sample belongs to each class. Based on highest class probabilities, this can then be converted to a categorical output vector.

4.2.4 Improving Training Stability

To improve training stability and mitigate mode collapse, DoppelGANger implements the Wasserstein loss function as proposed in Arjovsky et al. (2017). There, it is shown that in standard GANs the objective function is equivalent to minimizing the Jensen-Shannon divergence (JS):

$$JS(p_{\mathbf{x}}, p_g) = KL(p_{\mathbf{x}}|\frac{p_{\mathbf{x}} + p_g}{2}) + KL(p_g|\frac{p_{\mathbf{x}} + p_g}{2}) \tag{2}$$

$$KL(p|m) = \int_{-\infty}^{\infty} p(x) \log(\frac{p(x)}{m(x)})dx \tag{3}$$

Arjovsky et al. (2017) shows that when minimizing JS, the resulting loss function is not always continuous or differentiable. When this happens, it is not possible to train the network optimally, since we can not improve on the loss. Also, as the discriminator is trained using JS, the gradient goes to zero resulting in the vanishing gradient problem. With a zero gradient, weights in the neural network can not be updated properly.

By implementing the Wasserstein loss function, the training objective is equivalent to minimizing the Wasserstein distance:

$$W(p_{\mathbf{x}}, p_g) = \inf_{\gamma \in \Pi(p_{\mathbf{x}}, p_g)} \mathbb{E}_{(\mathbb{x},\mathbb{y}) \sim \gamma}[||x - y||] \tag{4}$$

Here, $\Pi(p_{\mathbf{x}}, p_g)$ is the set of all joint distributions $\gamma(x, y)$ with marginals respectively $p_{\mathbf{x}}$ and p_g.

Arjovsky et al. (2017) shows that the resulting loss function is continuous and differentiable everywhere (under some assumptions), alleviating the vanishing gradient problem. Furthermore, it allows the discriminator to be trained more optimally while still giving useful gradient updates to the generator. This way, the discriminator can learn to negate samples with low variety and thus mitigate mode collapse.

Furthermore, Lin et al. (2019) finds that when the variance of time-varying features across samples is very large, standard GANs exhibit mode collapse. DoppelGANger alleviates this cause of mode collapse by implementing auto-normalization. For this, it normalizes features prior to training. Next, it adds two static attributes to the i-th sample: $(max\{f^i\} \pm min\{f^i\})$, where f corresponds to the time-varying features. These two attributes can scale back features in the generated sample to a realistic range. However, in our datasets auto-normalization led to the range of synthetic variables becoming more unrealistic compared to real data. This is mostly due to strictly positive variables incidentally being scaled to negative values. For this reason, we did not use auto-normalization.

Now, Figure 3 shows the complete architecture of DoppelGANger.

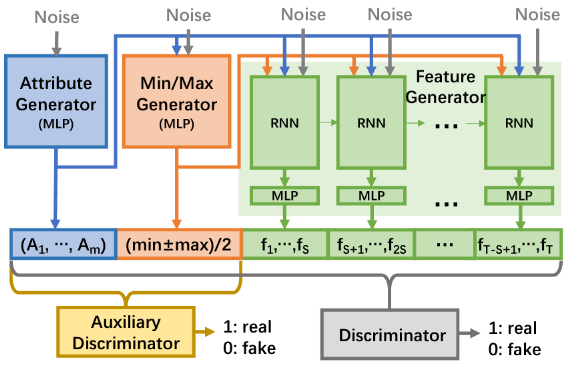


Figure 3: DoppelGANger architecture (Lin et al., 2019)

4.2.5 Preprocessing and Hyperparameters

Data preprocessing in DoppelGANger consists of scaling continuous variables to $[0,1]$, and one-hot encoding categorical variables. Since the goal of this research is not to create the most realistic synthetic samples, but to improve upon evaluation metrics, we fix hyperparameters and train DoppelGANger until convergence. This can take up to 200,000 batches (Lin et al., 2019). We use single hidden layers with 100 nodes in all networks inside DoppelGANger. This simple architecture can limit the capability to learn (nonlinear) dependencies in the data. However, when trying more layers and more nodes, generated synthetic data did not become significantly more realistic judging by the metrics provided in this thesis. So for the two datasets we use, this simple architecture seems to suffice. However, when the goal is to generate the best possible synthetic samples, one should perform proper hyperparameter optimization. Learning rates of all networks are fixed at 0.001, and we fix the batch size at 30. We train for the amount of epochs it takes per dataset to train approximately 200,000 batches. This ranges between 40,000 and 70,000 epochs. Lastly, we set the size of batched timestep output to 5 (see Section 4.2.2).

4.3 *Probabilistic Autoregressive Networks*

Section 2.1.1 illustrates that autoregressive neural networks are similar to regular neural networks, only they pass lagged inputs to the hidden layers. So in the case of a single hidden layer with k neurons, the formulation of the network is (Panja et al., 2022):

$$f(x^*) = \alpha_0 + \sum_{j=1}^{k} \alpha_j \phi(\beta_j + \theta_j x^*) \tag{5}$$

where x^* is p-lagged input, α_0, α_j, β_j are connecting weights, θ_j is a weight vector of dimension p and ϕ some specified (non-linear) activation function.

Since PANs model distributional parameters of next sequence values instead of the value itself, this requires a custom training objective. This objective has three components: it minimizes loss over all sequences S^i, each timestep S_t^i, and all distribution parameters $\pi_{t,j}^i$ (K. Zhang et al., 2022):

$$\mathcal{L} = \sum_i \sum_t \sum_{j=0}^{k-1} \mathcal{L}(S_t^{(i)}, \pi_{t,j}^{(i)}) \tag{6}$$

Here, $\mathcal{L}(\cdot)$ is some loss function suitable for the type of data.

4.4 *CPAR*

Next to DoppelGANger, this research generates synthetic samples using CPAR proposed in K. Zhang et al. (2022). CPAR consists of a PAN tailored to capture temporal dependencies, correlations between static attributes and time-varying features and model mixed-type variables.

4.4.1 Capturing Static-Temporal Correlations

As in DoppelGANger, CPAR uses a conditional architecture to better capture correlations between time-varying features and static attributes. First, CPAR models static attributes using Gaussian copulas. Then, it models time-varying features conditioned on static attributes using a conditional PAN.

Copulas model multivariate distributions of variables with uniform marginal distributions. To model static attributes, the Gaussian copula applies a probability integral transformation to each component: $(U_1, U_2, \ldots, U_d) = (\phi(X_1), \phi(X_2), \ldots, \phi(X_d))$, so $(U_1, U_2, \ldots, U_d)$ has uniform marginals. Here, ϕ denotes the univariate Gaussian distribution. Probability integral transformation states for continuously distributed random variable X with CDF F_X, that the random variable $Y = F_X(X)$ is uniformly distributed. Now for given (d, d) correlation matrix $\mathbf{\Sigma} = \begin{bmatrix} 1 & \rho_{ij} \\ \rho_{ij} & 1 \end{bmatrix}$, where ρ_{ij} denotes correlation between columns $i \neq j \in \{1, \ldots, d\}$, the copula $C_{\mathbf{\Sigma}}$ of $X_1, X_2, \ldots, X_d$ is defined as:

$$C_{\mathbf{\Sigma}}(u_1, u_2, \ldots, u_d) = \mathbf{\Phi}_{\mu,\mathbf{\Sigma}}(\phi^{-1}(u_1), \phi^{-1}(u_2), \ldots, \phi^{-1}(u_d)) \tag{7}$$

where $\mathbf{\Phi}_{\mu,\mathbf{\Sigma}}$ denotes the multivariate Gaussian distribution with zero mean vector μ and correlation matrix $\mathbf{\Sigma}$. Now to generate static attributes, we can simply sample $(Z_1, Z_2, \ldots, Z_d)$ from the multivariate Gaussian and apply probability integral transformation $(U_1, U_2, \ldots, U_d) = (\phi(Z_1), \phi(Z_2), \ldots, \phi(Z_d))$.

Then, CPAR generates time-varying features conditioned on static attributes. Here, a PAN network takes lagged values of the input sequence and static attributes as input, and predicts next-step distribution parameters. Then, CPAR draws next-step features from the parameterized distribution. CPAR can draw a next-step sample S times, and choose the sample which maximizes the likelihood. Here, S is a tunable hyperparameter.

4.4.2 Capturing Temporal Dependencies

To model temporal dependencies in the lagged inputs, CPAR uses an RNN. Here, hidden layers consist of a dense layer followed by a recurrent layer. Specifically, it uses Gated Recurrent Units (GRUs) in the recurrent layer. Unlike LSTM nodes, GRUs do not have a memory cell. However, they do possess a similar gated structure to modulate information flow: an update gate to modulate the amount of new data to take into account, and a reset gate to modulate the amount of information from the previous state to disregard (J. Chung et al., 2014). Since there is no memory cell, GRUs are suited for modelling shorter term temporal dependencies than LSTMs, but have less computational overhead. Figure 4 shows a schematic overview of CPAR.

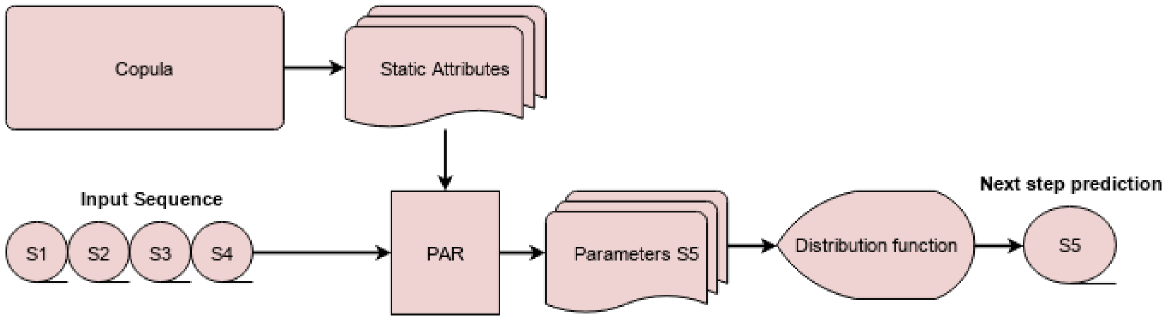


Figure 4: Schematic overview CPAR

4.4.3 Modelling Mixed-Type Variables

Equation (6) shows the custom loss function for PANs. To model mixed-type variables, CPAR specifies separate loss functions $\mathcal{L}(\cdot)$ for different data types, and uses appropriate activation functions in the neural network:

- **Continuous variables**: $\mathcal{L}(x;\mu,\sigma^2) = -(\log(f_{\mu,\sigma^2}(x)))$, where $f_{\mu,\sigma^2}(\cdot)$ denotes a Gaussian distribution with mean μ and variance σ^2. Parameter μ uses linear activation since it can take any positive or negative value, and σ^2 uses softplus activation to scale output to a positive range, since variance is strictly positive.
- **Discrete variables**: $\mathcal{L}(x;r,p) = -(\log(f_{r,p}(x)))$, where $f_{r,p}(x)$ denotes a Negative binomial distribution with number of successes r, and probability of success p. Parameter r uses softplus activation, since the number of successes is strictly positive. Parameter p uses sigmoid activation, since it represents a probability between 0 and 1.
- **Categorical variables**: $\mathcal{L}(x;\pi_0,\pi_1,\ldots,\pi_{K-1}) = -\sum_{j=0}^{K-1} x_j \log(\pi_j)$, where each $j \in \{0,\ldots,K-1\}$ represents a category with corresponding probability π_j. Parameters $\pi_0,\pi_1,\ldots,\pi_{K-1}$ use softmax activation, since they represent class probabilities (also see Section 4.2.5).

4.4.4 Training Stability Compared to DoppelGANger

CPAR is not prone to the same type of training stability issues as DoppelGANger, like mode collapse. Since we randomly sample data from a distribution, we are not likely to output many samples close together. Random sampling ensures a certain level of uncertainty, and thus variation in the generated samples.

However, CPAR does rely on some extra assumptions compared to DoppelGANger. Most importantly, that the likelihood of occurrence of a sequence value can be characterized by a Gaussian (continuous) or Negative binomial (discrete) distribution. Of course, other continuous and discrete distributions exist to sample from, and may be more suited.

4.4.5 Preprocessing and Hyperparameters

CPAR standardizes continuous variables, scales discrete variables to $[0, 1]$ and one-hot encodes categorical variables. As for DoppelGANger, we fix the hyperparameters of CPAR. We use the standard network architecture of the software package: two hidden layers (dense followed by recurrent) with 32 nodes each. The learning rate for the network is fixed at 0.001. CPAR learns from individual sequences at a time, so the batch size is fixed at 1. We train the model for approximately the same amount of time as DoppelGANger, which amounts to a range between 400 and 1000 epochs, depending on the dataset. We set S, the amount of samples to draw per timestep to maximize likelihood, at 10.

This seems like a small amount of epochs to train in comparison to DoppelGANger. However, first note that the amount of distinct training samples in CPAR is different from DoppelGANger. Since we separately model each timestep in every sequence (see Equation 6), the amount of distinct training samples is $N \cdot T$, whereas in DoppelGANger this is N. Furthermore, since batch size is fixed at 1 in CPAR and 30 in DoppelGANger, CPAR trains on 30 times more batches per epoch. This gives some intuition behind why a low amount of epochs in CPAR has similar training time to a high amount of epochs in DoppelGANger. CPAR runs through more distinct training samples per epoch, and through a higher number of batches.

4.5 *Evaluation: Fidelity*

The first step in evaluating synthetic data, is evaluating resemblance to real data. We investigate descriptive statistics, low dimensional visualizations of high dimensional data and GOF metrics for comparing the distribution of synthetic versus real data.

4.5.1 Descriptive Statistics

As a first sanity check, we compare descriptive statistics from Section 3 with those of synthetic samples. The distribution of synthetic samples should show similar range, location, and standard deviation. Secondly, since we consider temporal features, stepwise statistics should also be similar. We plot the mean of each time-varying feature over time. If the first check on descriptive statistics already shows poor resemblance, researchers should further tune their methods of data generation to produce better samples.

4.5.2 Visualization

Visual evaluation of synthetic samples requires some method to reduce the data to two dimensions. This research uses standard tSNE, and an extension to tSNE for multivariate mixed-type sequence data. Van der Maaten and Hinton (2008) introduce tSNE for visualizing high dimensional data in two dimensions. This method is derived from Stochastic Neighbor Embedding (SNE), with an altered cost function which is easier to optimize and solves the 'crowding problem'. We discuss this later in this Section.

Stochastic Neighbor Embedding

SNE computes similarities between two datapoints x_i and x_j, as conditional probabilities that these points are 'neighbors' to each other. Here, neighbors are picked in proportion to their probability density under a Gaussian distribution centered around x_i (Hinton and Roweis, 2002). The conditional probabilities are defined as:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \tag{8}$$

where σ_i^2 is the variance of the Gaussian around datapoint x_i. Note that here we compute similarities using Euclidean norm. However, other ways to calculate similarities exist, as we will show later in this Section.

Now, we define datapoint y_i as the low dimensional representation of high dimensional datapoint x_i. For y_i we can similarly compute conditional probabilities $q_{j|i}$, where we set variance of the Gaussian σ_i^2 to $\frac{1}{\sqrt{2}}$. Notice that then, the variance term drops out.

To correctly visualize the high dimensional data, $q_{j|i}$ should model the conditional probabilities to minimize the difference with $p_{j|i}$. It does so by using a gradient descent method on the sum of Kullback-Leibler divergences (also see Equation (3)) between $q_{j|i}$ and $p_{j|i}$. A drawback of this cost function is the asymmetry: large cost for low $q_{j|i}$ to model high $p_{j|i}$, and small cost for high $q_{j|i}$ to model low $p_{j|i}$. So, close real datapoints will likely not be represented far away, but distant real datapoints might be represented close together. This means local structure is preserved well, but global structure might not be. Also, the gradient descent on the sum of Kullback-Leibler divergences may be hard to optimize. Since it is a non-convex problem, results are influenced by initialization and hyperparameter values.

Now it is still required to select σ_i^2 in Equation (8). This is done implicitly via the user-specified perplexity parameter. This can be interpreted as a smooth measure for the effective number of neighbors for each datapoint:

$$Perp(P_i) = 2^{H(P_i)} \tag{9}$$

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{10}$$

SNE performs a binary search to find σ_i^2. It computes $H(P_i)$ by calculating all $p_{j|i}$ for given σ_i^2, to then compute $Perp(P_i)$. For each point i, SNE selects σ_i^2 which induces perplexity closest to the user-specified value. Intuitively, higher perplexity means more effective neighbors, which requires higher conditional probabilities of distant points, which requires higher variance σ_i^2.

Lastly, SNE exhibits the 'crowding problem'. When conditional probabilities of distant points are too low to accommodate the fixed perplexity measure, many distant points are pulled close together, thus poorly visualizing the data structure. Remember that low conditional probabilities are caused by datapoints being in the tail of a Gaussian distribution, instantiated over another datapoint. To increase conditional probabilities, the algorithm can either pull points closer together (crowding problem) or increase the probabilities in some other way (tSNE).

t-Distributed Stochastic Neighbor Embedding

To alleviate the crowding problem, and the asymmetry and optimization issues of the SNE cost function, tSNE presents two solutions: a symmetric version of the SNE cost function, and a Student-t distribution instead of Gaussian to compute similarities in the low dimensional representation.

To obtain a symmetric version of the cost function, tSNE minimizes a single Kullback-Leibler divergence between the joint distributions of the high and low dimensional space, instead of the sum of divergences of conditional probabilities. Here, pairwise similarities between a point i and j are defined as in Equation (8). Then, it sets joint probabilities $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2n}$, so datapoints x_i with low $p_{j|i}$ will have a more significant contribution to the cost function. This alleviates the previously described asymmetry issue, causing poor preservation of global structure. Furthermore, the gradient of this symmetric cost function is faster to compute.

To alleviate the crowding problem, tSNE uses a Student-t distribution instead of a Gaussian to convert distances between points into probabilities in the low dimensional space. Since tail densities in the Student-t are higher than in the Gaussian, distances of distant points are converted into higher conditional probabilities than in standard SNE. This way, there is less need for the SNE algorithm to pull distant points close together, thus alleviating the crowding problem.

Lastly, standard tSNE is not suitable for the three dimensional input of sequence data. Similarities between points result in a three dimensional array, which can not be represented in two dimensions. A simple alternative is to flatten the temporal dimension. However, visualizations of similarities per timestep are difficult to interpret and likely not suitable for the task at hand. This shows how close each point in a sample is to a point in another sample, while we wish to visualize how close one sample is to another. Also, this disregards differences in temporal dynamics across samples.

Multivariate Sequence t-Distributed Stochastic Neighbor Embedding

To visualize entire samples instead of single timesteps using tSNE, we calculate similarities per sequence using DTW. With DTW, we align sequences to account for differences in phase. Phase differences in sequences can cause large distances between samples, which are otherwise very similar. First, DTW calculates distances between samples of sequences $X = x_1, x_2, \dots, x_n$ and $Y = y_1, y_2, \dots, y_m$:

$$d(x_i, y_j) = \|x_i - y_j\| \tag{11}$$

where each element (i, j) corresponds to the alignment between point x_i and y_j. Then, DTW defines a warping path $W = w_1, w_2, \dots, w_K$, $\max(m, n) \leq K < m + n - 1$ which maps X to Y (Keogh and Pazzani, 2001). Here, w_k contains the set of warping elements $(i, j)_k$. The warping path is subject to three main constraints, to ensure proper alignment. Firstly, the warping path starts and ends at the start and end of sequences x_i and y_j, for proper boundaries. Secondly, steps in the warping path are restricted to adjacent cells for continuity. Lastly, the points in the warping path are monotonically spaced in time.

Now to find the warping path, DTW finds the warping path which minimizes distances between points after alignment. Figure 5 shows an example warping path. Here, the heatmap displays the distances between points, in two sequences of 250 points. Clearly, the optimal warping path is found by aligning points to have minimal distance, subject to the earlier mentioned constraints. The optimal warping path can be found through dynamic programming.

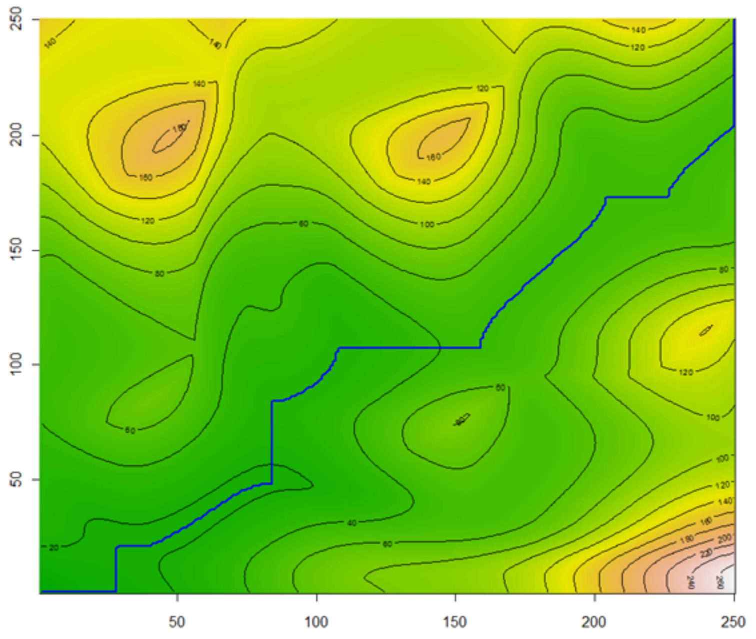


Figure 5: Example Warping Path (Senin, 2008)

In multivariate DTW, we can find an independent warping path for each variable (independent DTW), or a single warping path for all variables (dependent DTW). While independent DTW may improve performance, this greatly increases computational costs. For this reason, we use dependent DTW.

Since medical data consists of both continuous and categorical data, we use Gower distance instead of Euclidean to compute $d(x_i, y_j)$ in Equation (11). Gower distance between x_i and y_j equals the average distance over all variables p, where individual distances differ per data type:

$$D_{i,j} = \frac{\sum_{p=1}^{P} s_{i,j,p}}{P} \tag{12}$$

$$\tag{13}$$

where for categorical variables p, $s_{i,j,p} = 1$ if $x_{i,p} = y_{j,p}$, and 0 otherwise. For numerical variables

p, $s_{i,j,p} = 1 - \dfrac{|x_{i,p} - y_{j,p}|}{\max(x_p, y_p) - \min(x_p, y_p)}$ (Gower, 1971).

Now, we can compute the conditional probabilities for tSNE in Equation (8), by replacing the Euclidean distances with the distances computed through DTW with Gower distance. Then, we can proceed with tSNE to visualize the sequences.

Preprocessing and Hyperparameters

In tSNE with DTW and Gower distance, preprocessing only requires standardizing continuous variables. In standard tSNE we also one-hot encode categorical variables.

Perplexity is one of the most influential hyperparameters in tSNE visualization. Van der Maaten and Hinton (2008) recommend a value between 5 and 50. Following Wattenberg et al. (2016), we try multiple plots within this range and choose the plot which gives the clearest visualization of both global and local structure.

4.5.3 Goodness-Of-Fit

As a next evaluation of synthetic data fidelity, we train a classification model to discriminate between real and synthetic samples. If classification accuracy is high, this indicates poor fidelity. Furthermore, we provide a novel two-sample GOF test for medical data using classification based GOF testing.

Classifying Synthetic Samples

The goal of a (binary) classification model, is to predict the label of sample i given the measurements:

$$F(\mathbf{X}) = y \tag{14}$$

where $y \in \{0, 1\}$, $\mathbf{X}$ a sample of measurements and $F(\cdot)$ some classifier. For $y \in \{0, 1\}$, $F(\mathbf{X})$ outputs a continuous score $s \in (0, 1)$ which reflects confidence that $y = 1$. This score is used to make predictions using a decision rule (Friedman, 2004):

$$\hat{y}(x) = \begin{cases} 1 \text{ if } F(\mathbf{X}) > t \\ 0 \text{ otherwise} \end{cases} \tag{15}$$

where t is some chosen threshold value, usually 0.5.

To predict scores s, the classifier learns a transformation function from multivariate data to univariate confidence levels that a sample belongs to a label, which minimizes loss. If $F_{syn} = F_{real}$, the model is unable to learn an accurate transformation function, since there are no differences in the labelled data to exploit. In this case, the highest theoretical accuracy the model can achieve is 0.5, corresponding to randomly guessing labels. This gives rise to a first possible metric for evaluating GOF of synthetic samples: the closer accuracy of label predictions is to 0.5, the closer F_{syn} is to F_{real}.

Now, we can specify an appropriate binary classifier and make label predictions. Since medical data consists of a mix of temporal sequences and static attributes, we require a flexible classifier which can take both as input. RNNs are a suitable choice, since we can specify separate input and hidden layers for time-varying and static variables. The separate hidden layers share a single fully-connected output node with sigmoid activation, to output scores $s \in (0, 1)$. The hidden layers for sequences consist of LSTM nodes, whereas the hidden layers for static variables consist of regular fully-connected nodes.

However, it is still not obvious how close the prediction accuracy must be to 0.5, to conclude $F_{syn} = F_{real}$. For this, we require an explicit two-sample GOF test.

Two-Sample Testing

As mentioned above, if $F_{syn} = F_{real}$, the classifier is unable to learn an accurate transformation function since there are no differences in the labelled data to exploit. Furthermore, the classifier will similarly transform samples from the identical distributions into label confidence scores s. This means we can test $F_{syn} = F_{real}$ through testing $p(s_{syn}) = p(s_{real})$. This is the general idea of classification based GOF testing (Friedman, 2004).

Existing classification based multivariate GOF testing has mostly been performed for tabular data (Friedman, 2004, Kim et al., 2021 and Hediger et al., 2022). We use RNNs as classifiers, to construct a two-sample GOF test for a mix of sequences and static attributes of mixed data types, called MGOF. This is a novel statistical test. The general testing procedure is as follows:

Let $(\mathbf{X^s_{1,t}}, \mathbf{X^a_1}), \ldots, (\mathbf{X^s_{N,t}}, \mathbf{X^a_N})$ be a sample containing multivariate sequences $\mathbf{X^s_{i,t}}$ at timestep t, and static attributes $\mathbf{X^a_i}$. Now take $\mathbf{X} = (\mathbf{X^s_{1,t}}, \mathbf{X^a_1}), \ldots, (\mathbf{X^s_{N,t}}, \mathbf{X^a_N})$ and $\mathbf{Y} = (\mathbf{Y^s_{1,t}}, \mathbf{Y^a_1}), \ldots, (\mathbf{Y^s_{M,t}}, \mathbf{Y^a_M})$ from respective distributions F_X and F_Y. We test $H_0 : F_X = F_Y$ against $H_1 : F_X \neq F_Y$. To do this, assign opposite binary labels to the samples in $\mathbf{X}$ and $\mathbf{Y}$. Next, pool the samples from $\mathbf{X}$ and

$\mathbf{Y}$ and create independent stratified labelled train and test sets. Then, train an appropriate (RNN) classifier on the training set and make score predictions s on the test set. Here, scores s reflect the confidence that a sample belongs to label 1. Now, we split scores s into s_0 and s_1, corresponding to the correct label of the test sample. Finally, we apply a univariate two-sample GOF test on s_0 and s_1, and set the test statistic and p-value of MGOF to the outcome of this test. According to Friedman (2004), this is appropriate as long as independent datasets are used for training the classifier and predicting scores s.

In the last step, we can apply any two-sided univariate two sample GOF test. Nonparametric tests are favorable, since the distribution of discriminative scores s can be expected to be nonstandard. A good choice for a nonparametric univariate GOF test, is the Kolmogorov-Smirnoff test. Here, for two samples from population CDFs F and G with empirical CDFs $F_m(x)$ and $G_n(x)$, null hypothesis is $H_0 : F = G$ with test statistic $D = \sup_x |F_m(x) - G_n(x)|$. So, the test statistic equals the maximum possible difference between the empirical CDFs, see Figure 6.

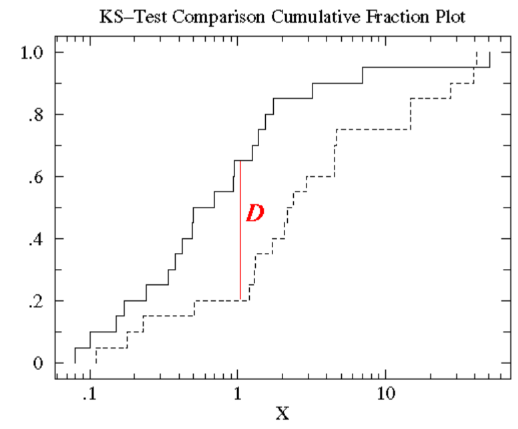


Figure 6: Kolmogorov-Smirnov test statistic (Kirkman, 1996)

The null hypothesis is rejected when $D > D_\alpha$ for some chosen level α. Generally, we can find the critical value through the asymptotic Kolmogorov distribution: $F_\infty(x) = 1 - 2e^{-2x^2}$ (Knuth, 1981). Including a sample size correction term, the critical value can now be found through the inverse of the asymptotic distribution at a specified level: $D_\alpha = \sqrt{-\ln(\alpha/2)\frac{1+m/n}{2m}}$.

To verify the size and power of this GOF test for medical data, we perform a Monte Carlo style simulation at the end of this Section.

Finally, note that the classification score measure and MGOF should give similar results. When

MGOF does not reject H_0 of equal distribution, we can expect classification accuracy to be around 0.5. The utility of MGOF comes in when we are not sure whether classification accuracy is close enough to 0.5, to conclude that two distributions are equal.

Hyperparameters

Since we use the same classification model for the score metric and MGOF, we apply the same hyperparameters. We apply a 60% train-test split on the data, and find optimal hyperparameters through stratified 3-fold cross-validation on a validation set within the training set. Here, we tune hyperparameters for highest accuracy in binary label prediction. We tune the amount of nodes in the recurrent layer between 10, 50 and 100, the batch size between 16, 32 and 64, and the amount of epochs between 50, 100 and 500. We set the learning rate at 0.001, and use the Adam optimizer for performing gradient descent at each iteration. Lastly, we add 10% dropout between the recurrent layer and the output node to regularize the network. Since we predict binary labels, we use the binary cross-entropy loss function, and sigmoid activation in the output layer.

Verifying Size and Power

We verify the size and power of MGOF, by investigating rejection percentages when performing the test on samples from known Data Generating Processes (DGPs). The size and power of a test equal the probability of false and correct rejection of the null hypothesis, respectively. The size of a test should be below the set confidence level.

To verify the size and power, we use different specifications of VAR Gaussian models. The general form of a p-lagged VAR process is:

$$\mathbf{Y_t} = \sum_{j=1}^{p} \mathbf{Y_{t-j}} \mathbf{A_j} + \epsilon_\mathbf{t} \tag{16}$$

where $\mathbf{Y_t}$ is an (N,P) matrix of N samples with P features at time t. $\mathbf{A}$ is a positive semi-definite (P,P) matrix of coefficients between features, with eigenvalues below 1, as to ensure stationarity. This way, sequences do not explode. Note that the definition of $\mathbf{A}$ differs from most standard VAR specifications: we simulate correlation between features within sequences, instead of correlation between sequences. This type of correlation is more likely to be present in real-world medical data, considered in this research. Furthermore, $\epsilon_\mathbf{t}$ is an (N,P) matrix where $\epsilon_\mathbf{t} \sim \mathbf{N}(\mu, \mathbf{\Sigma})$. As per definition of VAR processes, μ is a vector of zeros and $\mathbf{\Sigma}$ is a positive semi-definite matrix. Also,

$\epsilon_\mathbf{t}$ is uncorrelated across time. This last assumption may not be realistic for medical data, as there are likely to be some latent variables correlated across time, relating to overall health (Blackwood, 1988 and Rabe-Hesketh and Skrondal, 2008).

Since this research also considers static attributes and categorical data types, we randomly draw discrete values and add these as static categorical attributes to the VAR data. We draw binary values, with 0.5 probability to draw 0 and 1.

For simplicity, we simulate VAR data with a single lag. We generate $N = 150$ samples of length $T = 25$, containing 4 time-varying continuous features and 2 static categorical attributes. This is similar to the size of the two real-world datasets used in this thesis.

We consider three alternative distributions, by changing three parameters in the VAR Gaussian process:

- Different coefficient matrix $\mathbf{A}$, causing difference in temporal dependency
- Different error covariance matrix $\mathbf{\Sigma}$, causing difference in noise levels
- Different probabilities of drawing static categorical features, causing difference in static attribute distribution.

These differences are all in comparison to a baseline VAR Gaussian process. We report the rejection percentage at a 5% confidence level, when performing 10000 iterations of MGOF versus the baseline. Results can be found in Table 3.

With similar DGP, 2% of the tests are rejected, showing false rejections are indeed below the set level. For all three alternative distributions, 100% of the tests are rejected, showing good power of the test. The test has power against changes in temporal dependencies and noise levels in temporal variables, and class probabilities in categorical variables.

Table 3: Results Monte Carlo Simulation for Verifying Size and Power of MGOF

Difference in VAR Process	Rejection Percentage
No	2%
Coefficient Matrix	100%
Error Covariance Matrix	100%
Categorical probabilities	100%

However, there may be differences in multivariate distribution for which MGOF does not have power. Since the test depends on an RNN classifier, the test only has power for properties which can be detected by this model. In general, this is always an issue in classification based GOF testing. Neural network classifiers are flexible models which can capture linear and nonlinear dependencies, and thus provide power against many alternative distributions. However, this is still dependent on the specific network architecture used. For instance, RNNs can not capture both correlation in time and between features (spatiotemporal correlation). When we aspire to also test for differences in spatiotemporal correlation, we can alter the network structure by implementing a combination of convolutional and recurrent layers (L. Zhang et al., 2017). However, the methods for synthetic data generation in this research also do not implement a network architecture to capture this, so we do not investigate this property any further.

Marginal and Conditional Two-Sample Testing

DoppelGANger and CPAR both separately model the marginal distribution of static attributes and the distribution of time-varying features conditioned on static attributes, to model the joint multivariate distribution:

$$F(\mathbf{X_t^s}, \mathbf{X^a}) = F(\mathbf{X^a})F(\mathbf{X_t^s}|\mathbf{X^a}) \tag{17}$$

where $\mathbf{X^a}$ are static attributes and $\mathbf{X_t^s}$ are time-varying features at timestep t.

With MGOF, we test the null hypothesis of equal joint multivariate distribution $F(\mathbf{X_t^s}, \mathbf{X^a})$. To further investigate the outcome of this test, we also test the null hypotheses of equal marginal and conditional distributions $F(\mathbf{X^a})$ and $F(\mathbf{X_t^s}|\mathbf{X^a})$. This way, in case MGOF rejects, we can investigate whether this is due to poor capturing of the marginal or conditional distribution by DoppelGANger and CPAR. For this, we use the same classification based test setup as MGOF, with an altered network architecture. Here, we only need to alter the hidden layer.

To test the null hypothesis $F_{syn}(\mathbf{X^a}) = F_{real}(\mathbf{X^a})$, we use a single dense hidden layer. A dense layer suffices, since attributes are static. To test the null hypothesis $F_{syn}(\mathbf{X_t^s}|\mathbf{X^a}) = F_{real}(\mathbf{X_t^s}|\mathbf{X^a})$, we use a single conditional recurrent hidden layer from the implementation of Remy (2020). Here, the recurrent layer conditions on static attributes at the first timestep.

4.6 *Evaluation: Utility*

For synthetic data to be useful to researchers, it should perform similarly to real data in data-driven tasks. In sequential data a realistic scenario is researchers wanting to predict next-step data, for example to get early notice of a patient's condition worsening. Also, this is another measure for synthetic data fidelity. Good performance in predicting next-step data shows synthetic data has adequately captured conditional distributions over time.

We evaluate performance of the TSTR versus TRTR approach in next-step forecasting of temporal features conditioned on static attributes. Here, we train predictive model $F(\cdot)$ to model $F(\mathbf{X^s_T}|\mathbf{X^s_{1,\ldots,T-1}}, \mathbf{X^a})$, where $\mathbf{X^s_t}$ is timestep t in a sample of multivariate sequences, and $\mathbf{X^a}$ is a set of static attributes.

To forecast time-varying features conditioned on static attributes, we use conditional RNNs. We use a single conditional recurrent hidden layer from Remy (2020), and compare the accuracy of TSTR forecasts with TRTR forecasts. If these are equally accurate, it is justified to replace real data with synthetic data in forecasting. Since in this case, privacy risk is potentially reduced while performance in data-driven tasks is not harmed. To test equality of forecast accuracy between two sets of forecasts, we perform the Diebold-Mariano (DM) test with adjustment for small samples by Harvey et al. (1997), for each target variable. In the DM test, the null hypothesis is of equal forecast accuracy between two sets of forecasts $\hat{y}_1$ and $\hat{y}_2$ for target y. Specifically, $H_0 : DM \sim N(0, 1)$, where for variable p:

$$DM_p = \frac{\bar{d_p}}{\sqrt{(\gamma_{0,p} + 2\sum\limits_{k=1}^{h-1}\gamma_{k,p})/n}} \tag{18}$$

$$\gamma_{k,p} = \frac{1}{n}\sum_{i=k+1}^{n}(d_{i,p} - \bar{d_p})(d_{i-k,p} - \bar{d_p}) \tag{19}$$

$$d_{i,p} = e^2_{1,i,p} - e^2_{2,i,p} \tag{20}$$

$$e_{j,i,p} = y_{i,p} - \hat{y}_{j,i,p} \tag{21}$$

where h is the forecast horizon (1 in this research), and n is the amount of forecasts. Harvey et al. (1997) find that DM in Equation (18) rejects too often in small samples, and propose H_0 : $\sqrt{\frac{n+1-2h+h(h-1)/n}{n}}DM \sim T(n-1)$. This is the test statistic we use.

Since forecasts are multivariate, we also apply a multivariate generalization of the DM test from

Ziel and Weron (2018). Here, we find the loss differential from Equation (20) as:

$$d_i = \|\mathbf{e^2_{1,i}}\|_\mathbf{1} - \|\mathbf{e^2_{2,i}}\|_\mathbf{1} \tag{22}$$

A vital assumption of the DM test, is stationarity of the loss differential. We test if this assumption holds, by performing the Augmented Dickey-Fuller (ADF) test on the loss differential for each DM test. The ADF test tests the null hypothesis of a unit root, namely $H_0 : \gamma = 1$ in:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p-1} \delta_i \Delta y_{t-i} + \epsilon_t \tag{23}$$

Here, the test statistic is $DF = \dfrac{\hat{\gamma} - 1}{\hat{\sigma}_{\hat{\gamma}}}$, where under H_0, DF follows a nonstandard distribution. Critical values have been derived in literature (Chang and J. Y. Park, 2002).

Hyperparameters

As for the sequence classifier, we apply a 60% train-test split and find optimal hyperparameters through 3-fold cross validation on a validation set. We tune hyperparameters for minimal Mean Squared Forecast Error (MSFE). This is suitable, since datasets in this research only contain continuous time-varying features. Again, we tune the amount of nodes in the hidden layer, batch size and amount of epochs, for the same possible values as for MGOF. Other parameters like learning rate, optimizer and dropout are also equal.

4.7 *Evaluation: Privacy Risk*

Synthetic data introduces privacy risk, if a predictive model trained on synthetic data can accurately infer real unknown sensitive attributes. Here, we consider sensitive attributes to be *static*. Thus, we again adopt the TSTR approach, to evaluate privacy risk.

We train predictive model $F(\cdot)$ on synthetic data to model $F(\mathbf{A_s}|\mathbf{X_1}, \ldots, \mathbf{X_T})$, where $\mathbf{A_s}$ a set of static sensitive attributes, and $\mathbf{X}_t$ a sample of multivariate sequences at timestep t. Then, we test how accurately sensitive attributes can be predicted in real samples. Since the input data for the predictive model only consists of time-varying features, we employ an RNN with a single hidden layer as predictive model.

To measure the amount of privacy risk induced by making synthetic data available, we evaluate the attribute inference accuracy. High attribute inference accuracy means high risk of sensitive attributes being inferred, by using synthetic data. For categorical attributes we use the simple accuracy metric, for continuous attributes we use Mean Absolute Percentage Error (MAPE). These metrics provide an interpretable score on how well sensitive attributes can be inferred using synthetic data. Whether accuracy and MAPE of attribute inference denotes high privacy risk, should be assessed by policy makers per inferred variable. In part, this depends on the sensitivity of the particular variable, since this might vary. An example of such an assessment, can be found in Section 5.3.

Low accuracy of an AIA can be due to either poor predictability of sensitive attributes, or poor fidelity of synthetic data. When time-varying features have poor predictive power over sensitive attributes in real data, this will be reflected in synthetic data. Then, an AIA trained on synthetic data will have low accuracy. In this case, both the TRTR and TSTR approach will show low accuracy in an AIA. On the other hand, when time-varying features have high predictive power over sensitive attributes, but this is poorly reflected in synthetic data, only the TSTR approach will show low accuracy in an AIA. In this case, data fidelity is poor, since $F(\mathbf{X}^{\mathbf{s}}_{\mathbf{t}}|\mathbf{X}^{\mathbf{a}})$ is poorly captured in the generative model. To investigate whether low accuracy in an AIA is due to poor predictability of sensitive attributes or poor fidelity of synthetic data, we compare the outcomes of a TRTR and TSTR approach.

Hyperparameters

Again, we apply a 60% train-test split and find optimal hyperparameters through 3-fold cross validation on a validation set. We tune nodes in the hidden layer, batch size, and epochs, for the same possible values as MGOF and the forecasting model. Other parameters like learning rate, optimizer and dropout are also equal. Now, since sensitive attributes are a mix of continuous and binary variables (for example age and sex), we apply separate output layers after the hidden layer. The continuous output layer uses linear activation and mean squared error loss, and the categorical output layer uses sigmoid activation and binary cross-entropy loss. We tune the network for a joint low score in mean squared error for continuous variables, and high score in accuracy for binary variables.

4.8 *Note on Sample Sizes*

As mentioned in Section 3, sample sizes in this thesis are relatively small. This can be an issue since both synthetic data generation and evaluation methods rely on machine learning models, which often require large datasets to perform well. The tendency for machine learning models trained on small datasets to quickly overfit to the training samples, can result in poor generalization performance. In synthetic data generation, generalization performance can be seen as the ability to produce a wide variety of unique samples following the distribution of real samples.

Firstly in DoppelGANger, GANs learn not only from input data, but also from generated fake samples. This way, the discriminator (and consequently the generator) learn from more distinct samples than just the amount of input samples. Possibly, this can cause GANs to perform well even in small samples. For example Y. Liu et al. (2019) achieve good results, when training a GAN on only 60 real samples. However, Karras et al. (2020) point out that GANs can quickly overfit to the training samples in small datasets, causing issues like mode collapse and training instability. In synthetic data generation, this can be seen as poor generalization performance.

As mentioned in Section 4.4.5, CPAR takes more distinct training samples than DoppelGANger: $N \cdot T$ instead of N. Because of this, sample sizes are much larger than for DoppelGANger, although one can argue that the CD dataset still has few samples. Furthermore, CPAR is not likely to exhibit generalization issues like mode collapse due to random sampling of synthetic data. With reasonable sampling sizes, it is expected that random samples cover the entire range of the sampling distribution. How close the parameterized sampling distribution follows the true data distribution, depends on the validity of the sampling distribution and how well the model is trained.

Small datasets can also be an issue in the evaluation metrics. GOF metrics for fidelity, forecasting accuracy for utility, and attribute inference for privacy evaluation all rely on neural networks which are prone to overfitting in small datasets. We include small hidden layers and few training epochs in the hyperparameter grid, and introduce dropout between the hidden layer and output layer, to mitigate this issue.

5 Results

Now, we apply the evaluation framework on generated synthetic data of two real-world datasets. We investigate fidelity, utility and privacy risk of synthetic samples.

5.1 *Fidelity*

To evaluate fidelity, we investigate descriptive statistics, low dimensional visualization, and classification based GOF metrics.

Table 4 and Table 5 show descriptive statistics of synthetic versus real variables. In the NM dataset, samples from CPAR and DoppelGANger generally show similar range, location and standard deviation as real data. Samples from DoppelGANger seem to deviate more from real data than samples from CPAR, especially the range. Only for **Bridion**, synthetic samples show noteworthy differences from real data in descriptive statistics. Here, samples from DoppelGANger and CPAR are biased in opposite directions. DoppelGANger greatly underestimates and CPAR greatly overestimates standard deviation relative to the mean. DoppelGANger suffers from mode collapse, and only outputs values around zero. The generator is able to fool the discriminator in this way, since **Bridion** is highly sparse. In CPAR, sparsity causes poor estimation of the parameters of the Negative binomial distribution. Here, a small difference in estimated nonzero values (2% in synthetic versus 1% in real data) causes a large difference in standard deviation. Sparsity causes poor estimation, since the network receives too little information to correctly model the distribution of the variable.

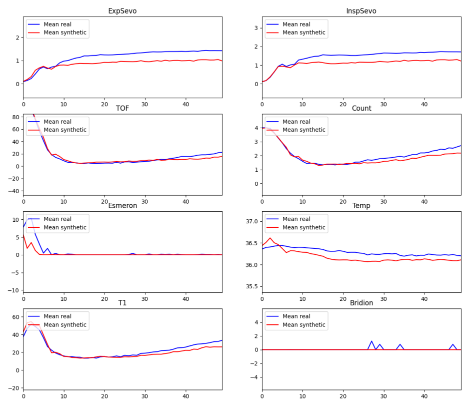
In the CD dataset, samples from both DoppelGANger and CPAR show similar range, location and standard deviation as the real data.

Table 4: Descriptive Statistics of Synthetic and Real Cervical Dystonia Data

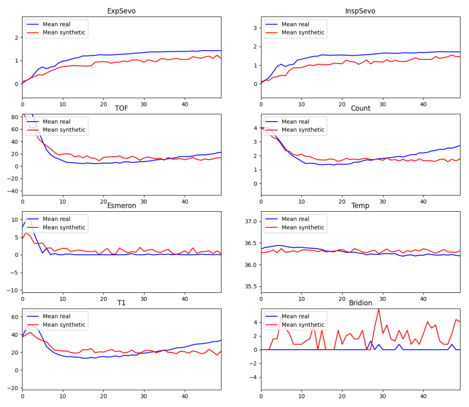
Variable	Dataset	Temporal	Type	Min	Max	Mean	Median	St. Deviation
Age	Real	No	Num	26.000	83.000	55.581	56.000	12.147
	DGAN			29.199	80.142	58.600	60.307	11.306
	CPAR			26.000	79.000	55.152	54.000	12.078
Treat	Real	No	Cat	1	3	1.990	2	0.811
	DGAN			1	3	1.762	2	0.763
	CPAR			1	3	2.048	2	0.832
Sex	Real	No	Cat	0	1	0.371	0	0.483
	DGAN			0	1	0.333	0	0.471
	CPAR			0	1	0.390	0	0.488
TWSTRS	Real	Yes	Num	6.000	71.000	40.731	42.000	12.633
	DGAN			7.630	69.087	41.264	42.159	11.486
	CPAR			6.000	65.000	40.703	43.000	11.345

Table 5: Descriptive Statistics of Synthetic and Real Neuromuscular Monitoring Data

Variable	Dataset	Temporal	Type	Min	Max	Mean	Median	St. Deviation
Age	Real	No	Num	12.000	95.000	54.055	56.000	20.635
	DGAN			12.225	94.989	45.991	40.138	25.109
	CPAR			12.000	85.000	53.484	56.000	20.560
BMI	Real	No	Num	16.110	47.860	26.575	26.175	5.780
	DGAN			19.126	45.487	25.701	24.973	4.858
	CPAR			16.110	47.860	25.817	24.530	5.829
Sex	Real	No	Cat	0	1	0.594	1	0.491
	DGAN			0	1	0.609	1	0.488
	CPAR			0	1	0.531	1	0.499
typeStudy	Real	No	Cat	0	1	0.367	0	0.482
	DGAN			0	1	0.430	0	0.495
	CPAR			0	1	0.320	0	0.467
ExpSevo	Real	Yes	Num	0.000	6.800	1.144	1.400	0.876
	DGAN			0.008	5.247	0.867	0.606	0.668
	CPAR			0.000	6.800	0.856	0.840	0.891
InspSevo	Real	Yes	Num	0.000	8.000	1.426	1.760	1.091
	DGAN			0.009	6.2910	1.085	0.634	0.930
	CPAR			0.000	8.000	1.052	1.000	1.104
TOF	Real	Yes	Num	0.000	109.800	18.763	0.000	32.948
	DGAN			0.000	109.800	18.513	3.011	32.063
	CPAR			0.000	107.600	20.177	0.000	36.076
Count	Real	Yes	Num	0.000	4.000	2.081	1.000	1.456
	DGAN			0.005	4.000	1.934	1.564	1.267
	CPAR			0.000	4.000	1.969	1.000	1.411
Esmeron	Real	Yes	Num	0.000	80.000	0.832	0.000	5.740
	DGAN			0.000	76.899	0.254	0.000	3.118
	CPAR			0.000	80.000	1.427	0.000	7.629
Temp	Real	Yes	Num	34.000	38.300	36.293	36.300	0.468
	DGAN			35.312	37.465	36.181	36.199	0.337
	CPAR			34.000	38.300	36.310	36.300	0.384
T1	Real	Yes	Num	0.000	119.500	23.423	22.700	22.809
	DGAN			1.470	112.207	22.013	12.315	20.654
	CPAR			0.000	119.500	23.054	25.900	24.785
Bridion	Real	Yes	Num	0.000	160.000	0.072	0.000	2.947
	DGAN			0.000	0.571	0.000	0.000	0.008
	CPAR			0.000	160.000	1.878	0.000	14.741

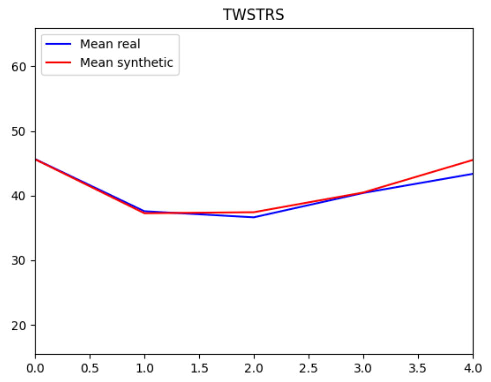


(a) DGAN

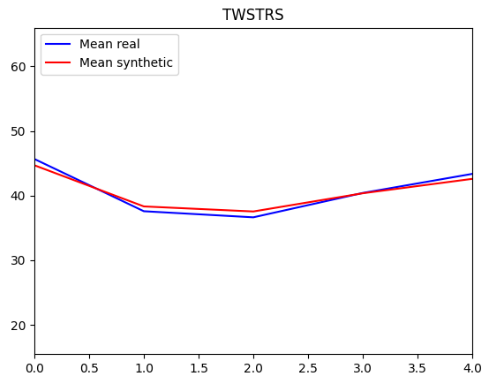


(b) CPAR

Figure 7: Mean Over Time, Neuromuscular Monitoring Data



(a) DGAN

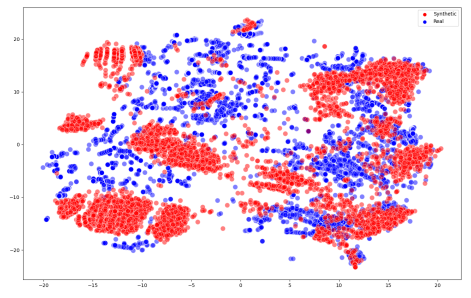


(b) CPAR

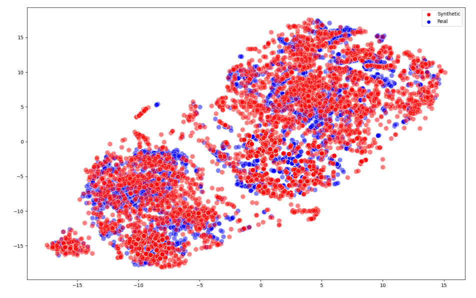
Figure 8: Mean Over Time, Cervical Dystonia Data

Figure 7 and Figure 8 show the mean over time of each variable, of the NM and CD dataset respectively. In the NM dataset, DoppelGANger seems to accurately capture the structure of the mean over time for all variables except **Bridion**. CPAR is not able to capture the increase in mean in the second half of sequences in **TOF**, **Count** and **T1**. Possibly this is due to the use of GRU instead of LSTM units in the recurrent layer, which can model shorter term dependencies. For **Bridion**, the mean over time shows the same issues as before. DoppelGANger exhibits mode collapse and only outputs values around zero, whereas CPAR overestimates the amount of nonzero values.

In the CD dataset, both DoppelGANger and CPAR capture the structure of the mean over time, which first decreases and then increases.

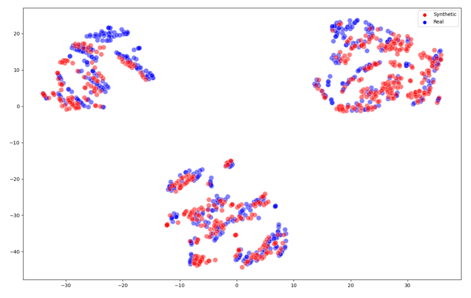


(a) DGAN

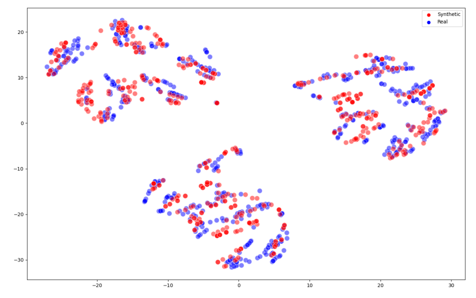


(b) CPAR

Figure 9: tSNE (standard) Neuromuscular Monitoring Data



(a) DGAN

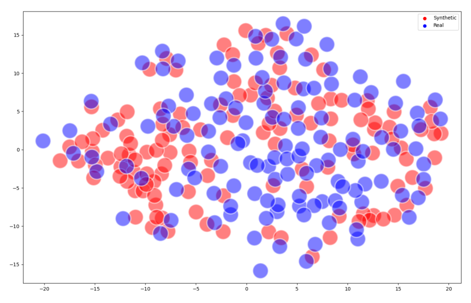


(b) CPAR

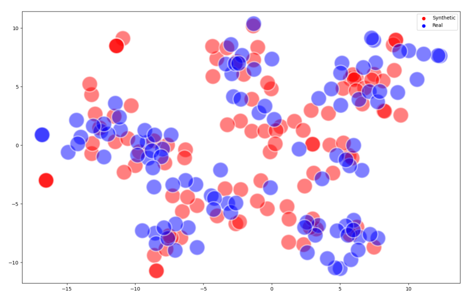
Figure 10: tSNE (standard) Cervical Dystonia Data

Figure 9 and Figure 10 show tSNE plots of the NM and CD dataset respectively, where the temporal dimension is flattened. In the NM dataset, datapoints from CPAR show good preservation of global structure. Local structure is more difficult to distinguish. Points from DoppelGANger show quite poor preservation of the data structure, mostly due to points being less dispersed. This could be due to less variability in some variables in the generated samples, which is a known issue in GANs (mode collapse).

In the CD dataset, points from both DoppelGANger and CPAR show good preservation of global structure. Again, points from DoppelGANger are somewhat less dispersed than the real data, especially looking at the top left and top right clusters.

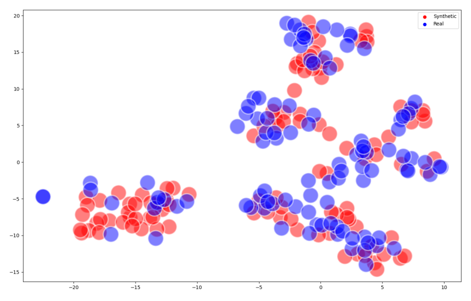


(a) DGAN

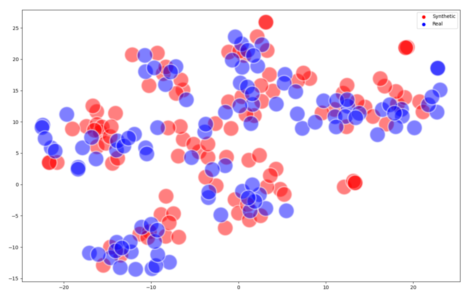


(b) CPAR

Figure 11: Multivariate Sequence tSNE Neuromuscular Monitoring data



(a) DGAN



(b) CPAR

Figure 12: Multivariate Sequence tSNE Cervical Dystonia data

Figure 11 and Figure 12 show tSNE plots where we visualize entire sequences instead of single timesteps, by computing distances through DTW with Gower distance. In the NM dataset, samples from DoppelGANer again show some mode collapse. Many synthetic samples are close together in the left side of the plot, whereas samples from CPAR are again more dispersed. In general, global structure is captured well for both samples from DoppelGANger and CPAR.

In the CD dataset, we can see from Figure 12 that samples from DoppelGANger again exhibit some mode collapse. It outputs a large amount of samples capturing the leftmost cluster, and samples are generally closer together than real samples. This is more easily visible here, than in the standard tSNE plot. Furthermore, some samples from CPAR seem to be quite distant from real samples. This was not easily visible in standard tSNE plots. Besides this, global and local structure seem to be preserved well in generated samples from both models.

Table 6: Classification Based GOF Testing

Dataset	Model	Distribution	Accuracy	p-value
NM	DGAN	$F(\mathbf{X^s_t}, \mathbf{X^a})$	0.932	0.000
		$F(\mathbf{X^s_t} \vert \mathbf{X^a})$	0.961	0.000
		$F(\mathbf{X^a})$	0.515	0.684
	CPAR	$F(\mathbf{X^s_t}, \mathbf{X^a})$	0.971	0.000
		$F(\mathbf{X^s_t} \vert \mathbf{X^a})$	0.951	0.000
		$F(\mathbf{X^a})$	0.505	0.684
CD	DGAN	$F(\mathbf{X^s_t}, \mathbf{X^a})$	0.524	0.610
		$F(\mathbf{X^s_t} \vert \mathbf{X^a})$	0.583	0.436
		$F(X^a)$	0.524	0.186
	CPAR	$F(\mathbf{X^s_t}, \mathbf{X^a})$	0.524	0.292
		$F(\mathbf{X^s_t} \vert \mathbf{X^a})$	0.583	0.610
		$F(\mathbf{X^a})$	0.464	0.791

Table 6 shows results from predictions of a classification model trained to discriminate synthetic from real samples. Here, accuracy close to 0.5 indicates good fidelity. The last column shows p-values from the MGOF test for the joint multivariate distribution $F(\mathbf{X^s_t}, \mathbf{X^a})$ of time-varying features $\mathbf{X^s_t}$ and static attributes $\mathbf{X^a}$. Here, we set the level of the test at 5%. Since DoppelGANger and CPAR separately model $F(\mathbf{X^s_t}|\mathbf{X^a})$ and $F(\mathbf{X^a})$ to model the joint distribution, we also include p-values for testing these multivariate distributions using classification based GOF testing.

For samples from both DoppelGANger and CPAR, the classification model achieves accuracy well above 0.5 for the joint distribution in NM data. As expected this corresponds to p-values below 0.05, rejecting the null hypothesis of equal joint distribution. Interestingly, for static attributes generated by both DoppelGANger and CPAR, a classification model is not able to distinguish synthetic from real samples. Accuracy is close to 0.5 and p-value above 0.05, thereby not rejecting the null hypothesis of equal marginal distribution of static attributes. The differences in joint distribution between synthetic and real data, thus seem to stem from the generation of the time-varying features. In general for the NM dataset, we can conclude that both CPAR and DoppelGANger have not accurately captured the joint multivariate distribution in generated synthetic samples.

In the CD dataset, accuracy of the classification model for samples from both DoppelGANger and CPAR for the joint, conditional and marginal distribution are all close to 0.5. Furthermore, p-values of classification based GOF testing are all above 0.05, thus not rejecting the null hypothesis of equal distribution. For the CD dataset, we can conclude that both CPAR and DoppelGANger have accurately captured the joint multivariate distribution in generated synthetic samples.

5.2 *Utility*

Next, we evaluate utility of synthetic samples in data-driven tasks, namely forecasting. Table 7 shows results of next-step forecasting performance of temporal variables from the NM and CD datasets. The Table shows MSFE and p-values of the DM test for equal forecast accuracy, and p-values of the ADF test for testing the stationarity assumption in the DM test. We set the level for both tests at 5%. Note that the alternative hypothesis in the DM test is of unequal forecast accuracy, so the test is two-sided. The ADF test is one-sided. In the NM dataset, we accept the null hypothesis of equal forecasting accuracy between the TSTR and TRTR approach for samples from DoppelGANger for variables **TOF**, **Count**, **Esmeron**, **Temp**, and **Bridion**. However for **Bridion**, we also accept the null hypothesis of a unit root in the loss differential of the DM test, so the result is invalid. For samples from CPAR, we reject the null hypothesis of equal forecasting accuracy for all variables. Lastly, for the multivariate DM test, we reject the null hypothesis of equal forecasting accuracy for samples from DoppelGANger and CPAR.

Therefore in the NM dataset, neither when using synthetic samples from DoppelGANger nor CPAR, we can forecast as accurately as when using real data. However, since when using samples from DoppelGANger some univariate DM tests do not reject, we can conclude that DoppelGANger captures stepwise distributions of some variables better than CPAR. This is also consistent with conclusions from Figure 7.

In the CD dataset, we accept the null hypothesis of equal forecasting accuracy when using samples from DoppelGANger, and reject when using samples from CPAR. Again, DoppelGANger captures stepwise distributions better than CPAR. Since the CD dataset contains very short se-

Table 7: Forecast performance TSTR versus TRTR approach

Variable	Dataset	MSFE	DM	ADF
ExpSevo	Real	0.051		
	DGAN	0.204	0.001	0.000
	CPAR	2.053	0.000	0.000
InspSevo	Real	0.044		
	DGAN	0.187	0.000	0.000
	CPAR	2.205	0.000	0.000
TOF	Real	0.192		
	DGAN	0.318	0.061	0.000
	CPAR	1.401	0.000	0.011
Count	Real	0.265		
	DGAN	0.472	0.103	0.000
	CPAR	1.556	0.000	0.000
Esmeron	Real	0.013		
	DGAN	0.014	0.672	0.000
	CPAR	0.127	0.009	0.000
Temp	Real	0.250		
	DGAN	0.583	0.038	0.000
	CPAR	2.417	0.000	0.000
T1	Real	0.147		
	DGAN	0.511	0.000	0.000
	CPAR	1.651	0.000	0.000
Bridion	Real	0.029		
	DGAN	0.018	0.249	0.414
	CPAR	2.725	0.0315	0.000
Multivariate	Real	0.124		
	DGAN	0.289	0.000	0.000
	CPAR	1.7675	0.000	0.000
TWSTRS	Real	0.213		
	DGAN	0.506	0.183	0.000
	CPAR	1.397	0.000	0.000

quences, this is likely not due to the difference in type of recurrent nodes (LSTM versus GRU). Other possible explanations are a higher number of recurrent nodes in DoppelGANger (100 versus 32), or batched output samples in DoppelGANger (see Section 4.2.2).

5.3 *Privacy Risk*

Finally, we evaluate privacy risk introduced by distributing synthetic data. Here, we evaluate whether synthetic data can be used to infer sensitive attributes, by training a predictive model on synthetic data and inferring sensitive attributes in real samples.

Table 8: Predictive accuracy of Attribute Inference Attack.

Metric	Variable	Model	Dataset	
Accuracy	**Sex**		NM	CD
		Real	0.692	0.690
		DGAN	0.558	0.500
		CPAR	0.558	0.595
MAPE	**Age**		NM	CD
		Real	0.392	0.155
		DGAN	0.695	0.193
		CPAR	0.337	0.17
	BMI		NM	
		Real	0.181	
		DGAN	0.210	
		CPAR	0.178	

Table 8 shows results on the accuracy of the AIA for both datasets. Both datasets contain sensitive attributes **Sex** and **Age**, and the NM dataset also contains **BMI**.

We compare AIA accuracy of the TSTR approach with the TRTR approach. This way, we can investigate whether low AIA accuracy is due to poor predictability of the sensitive attributes by time-varying features, or due to loss of attribute-feature correlation (and thus fidelity) in synthetic data.

For **Sex** in both datasets, the TRTR approach achieves considerably higher accuracy than the TSTR approach. AIA accuracy is low for this variable, due to loss in attribute-feature correlation in synthetic data. Since inference accuracy is close to 50%, corresponding to random guessing, policy makers will likely not find synthetic samples a risk to privacy.

For **Age** in the NM dataset, AIA accuracy is low in both the TSTR and TRTR approach. Here, AIA accuracy is low due to poor predictability of the attribute by time-varying features. MAPE is over 30%, so with a mean Age of 54 years, this amounts to an average inference error of over 15 years. Policy makers will likely not find synthetic samples a risk to privacy. In the CD dataset, AIA accuracy is much higher: in synthetic samples, MAPE is between 17 and 19%, corresponding to an average inference error of 9 to 10 years. Policy makers might find this a risk to privacy.

For **BMI** in the NM dataset, MAPE in syntethic samples is between 18 and 21%, corresponding to an average inference error between 4 and 6 points. Since classification cutoffs for health status through the BMI index are typically every 5 points (20, 25 and 30) (Freedman and Sherry, 2009), 4 to 6 points can be seen as a large average error. So, researchers will likely not find synthetic samples a threat for inference of BMI. Here, since AIA accuracy is close for the TSTR and TRTR approach, low accuracy is due to poor predictability of the attribute.

6 Conclusion

In conclusion, this thesis proposes an evaluation framework for synthetic medical data, which is defined as a combination of static and temporal variables of mixed data types. To assess synthetic medical data, the framework introduces new methodologies for visualization, GOF testing, and privacy risk evaluation.

To evaluate the fidelity of synthetic medical data to real data, we use an adapted version of the tSNE algorithm. This adaptation involves using DTW with Gower distance as a distance metric within tSNE, enabling the visualization of entire sequences and the computation of distances between mixed data types. Additionally, to assess the similarity of the distributions of synthetic and real medical data, we propose a novel two-sample GOF test with MGOF. This test involves training a classifier to distinguish synthetic from real samples, and applying a univariate GOF test on the classification scores. If classification scores of synthetic and real samples are not distributed equally, we conclude that the synthetic and real distribution are not equal. We demonstrate that the test has false rejection rates below the set level, and power against changes in temporal dependencies, noise levels, and class probabilities of categorical data, through a Monte Carlo simulation.

Next, to evaluate utility of synthetic data in data-driven tasks, we evaluate forecasting performance of a model trained on synthetic data on a real test set. Following Yoon, Jarrett, et al. (2019) and Desai et al. (2021), we compare these forecasts with forecasts made with a model trained on real data. If forecast accuracy between these two approaches is equal, we conclude that synthetic data can replace real data in data-driven tasks.

Lastly, to evaluate privacy risk associated with distributing synthetic data, we perform AIAs on synthetic samples. These AIAs involve training a predictive model on synthetic data to predict sensitive static attributes, and then using this model to make predictions on a real test set.

We apply the proposed evaluation framework on synthetic medical data generated using a GAN and a PAN. Synthetic medical data is generated from two real-world medical datasets: one related to neuromuscular monitoring during surgeries, and another to a randomized controlled drug trial for cervical dystonia. The results show good fidelity and utility for synthetic cervical dystonia data, but also risk of inference of the sensitive attribute of age. For synthetic neuromuscular monitoring data, the framework indicates poor fidelity and utility, but also low privacy risk. In both datasets we find evidence that the GAN exhibits a certain level of mode collapse, resulting in the generation of many highly similar samples. In contrast, the PAN does not display such issues.

7 Discussion

As discussed in Section 4.8, this thesis uses relatively small datasets for machine learning methods. This is due to difficult access to suitable open-source medical data. This can cause poor results, as we see for the neuromuscular monitoring dataset. On the other hand, this clearly shows the great potential for synthetic data in the medical domain. Since there is generally poor access to medical data, synthetic data can be used instead.

Furthermore, we use very simple network architectures in the methods for synthetic data generation. More hidden layers and nodes may improve the quality of synthetic samples. Moreover, the GAN exhibits varying degrees of mode collapse, which can potentially be mitigated through better hyperparameter tuning and including methods like ‘unrolled’ training (Metz et al., 2016). However, the goal of this thesis is not to generate the most realistic samples. Rather, it is to present novel evaluation methodologies.

Also, more research can be done into classification based GOF testing for synthetic medical data. By implementing different network architectures, the test may have power against more alternatives. For example by implementing a combination of recurrent and convolutional layers, the test can have power against differences in spatiotemporal correlation (L. Zhang et al., 2017).

Lastly, further research should be done by implementing the novel methods from this thesis on larger medical datasets. This can further prove the usefulness of the presented methods in evaluating fidelity, utility, and privacy risk of large synthetic medical datasets.

References

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). “Wasserstein generative adversarial networks”. In: *International conference on machine learning*. PMLR, pp. 214–223.

Baowaly, Mrinal Kanti et al. (2019). “Synthesizing electronic health records using improved generative adversarial networks”. In: *Journal of the American Medical Informatics Association* 26.3, pp. 228–241.

Biswas, Munmun and Anil K Ghosh (2014). “A nonparametric two-sample test applicable to high dimensional data”. In: *Journal of Multivariate Analysis* 123, pp. 160–171.

Blackwood, Larry (1988). “Latent variable models for the analysis of medical data with repeated measures of binary variables”. In: *Statistics in Medicine* 7.9, pp. 975–981.

Chang, Yoosoon and Joon Y Park (2002). “On the asymptotics of ADF tests for unit roots”. In: *Econometric Reviews* 21.4, pp. 431–447.

Choi, Edward et al. (2017). “Generating multi-label discrete patient records using generative adversarial networks”. In: *Machine learning for healthcare conference*. PMLR, pp. 286–305.

Chung, Junyoung et al. (2014). “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555*.

Creswell, Antonia et al. (2018). “Generative adversarial networks: An overview”. In: *IEEE signal processing magazine* 35.1, pp. 53–65.

Dash, Saloni et al. (2020). “Medical time-series data generation using generative adversarial networks”. In: *International Conference on Artificial Intelligence in Medicine*. Springer, pp. 382–391.

Davis, Charles Shaw (2002). *Statistical methods for the analysis of repeated measurements*. Tech. rep. Springer.

Desai, Abhyuday et al. (2021). “TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation”. In: *arXiv preprint arXiv:2111.08095*.

Escudero, Arnanz et al. (2022). *Dtwparallel: A Python Package to Efficiently Compute Dynamic Time Warping*. Tech. rep. SSRN.

Esteban, Cristóbal, Stephanie L Hyland, and Gunnar Rätsch (2017). “Real-valued (medical) time series generation with recurrent conditional gans”. In: *arXiv preprint arXiv:1706.02633*.

Freedman, David S and Bettylou Sherry (2009). "The validity of BMI as an indicator of body fatness and risk among children". In: *Pediatrics* 124.Supplement_1, S23–S34.

Friedman, Jerome (2004). *On multivariate goodness-of-fit and two-sample testing*. Tech. rep. Citeseer.

Gong, Neil Zhenqiang and Bin Liu (2018). "Attribute inference attacks in online social networks". In: *ACM Transactions on Privacy and Security (TOPS)* 21.1, pp. 1–30.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems* 27.

Gower, John C (1971). "A general coefficient of similarity and some of its properties". In: *Biometrics*, pp. 857–871.

Harvey, David, Stephen Leybourne, and Paul Newbold (1997). "Testing the equality of prediction mean squared errors". In: *International Journal of forecasting* 13.2, pp. 281–291.

Hayes, Jamie et al. (2019). "Logan: Membership inference attacks against generative models". In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. Vol. 2019. 1. De Gruyter, pp. 133–152.

"Healthcare Data Breach Statistics" (Feb. 20, 2021). In: *HIPAA Journal*. URL: `https://www.hipaajournal.com/healthcare-data-breach-statistics/` (visited on 05/11/2022).

Hediger, Simon, Loris Michel, and Jeffrey Näf (2022). "On the use of random forest for two-sample testing". In: *Computational Statistics & Data Analysis* 170, p. 107435.

Henriksen-Bulmer, Jane and Sheridan Jeary (2016). "Re-identification attacks—A systematic literature review". In: *International Journal of Information Management* 36.6, pp. 1184–1192.

Hernandez, Mikel et al. (2022). "Synthetic Data Generation for Tabular Health Records: A Systematic Review". In: *Neurocomputing*.

Hinton, Geoffrey and Sam Roweis (2002). "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15.

Karras, Tero et al. (2020). "Training generative adversarial networks with limited data". In: *Advances in Neural Information Processing Systems* 33, pp. 12104–12114.

Keogh, Eamonn J and Michael J Pazzani (2001). "Derivative dynamic time warping". In: *Proceedings of the 2001 SIAM international conference on data mining*. SIAM, pp. 1–11.

Kim, Ilmun et al. (2021). "Classification accuracy as a proxy for two-sample testing". In: *The Annals of Statistics* 49.1, pp. 411–434.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kirkman, Tom W (1996). "Statistics to use". In: *http://www. physics. csbsju. edu/stats/*.

Knuth, Donald (1981). *The Art of Computer Programming (Volume 2)*. Addison–Wesley, p. 51.

Lee, Dongha et al. (2020). "Generating sequential electronic health records using dual adversarial autoencoder". In: *Journal of the American Medical Informatics Association* 27.9, pp. 1411–1419.

Li, Zhimei and Yaowu Zhang (2020). "On a projective ensemble approach to two sample test for equality of distributions". In: *International Conference on Machine Learning*. PMLR, pp. 6020–6027.

Lin, Zinan et al. (2019). "Generating high-fidelity, synthetic time series datasets with doppelganger". In: *arXiv preprint arXiv:1909.13403*.

Liu, Yufei et al. (2019). "Wasserstein GAN-based small-sample augmentation for new-generation artificial intelligence: a case study of cancer-staging data in biology". In: *Engineering* 5.1, pp. 156–163.

Metz, Luke et al. (2016). "Unrolled generative adversarial networks". In: *arXiv preprint arXiv:1611.02163*.

Norgaard, Skyler et al. (2018). "Synthetic sensor data generation for health applications: a supervised deep learning approach". In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, pp. 1164–1167.

Panja, Madhurima, Uttam Kumar, and Tanujit Chakraborty (2022). "An Interpretable Probabilistic Autoregressive Neural Network Model for Time Series Forecasting". In: *arXiv preprint arXiv:2204.09640*.

Park, Noseong et al. (2018). "Data synthesis based on generative adversarial networks". In: *arXiv preprint arXiv:1806.03384*.

Pei, Hengzhi et al. (2021). "Towards Generating Real-World Time Series Data". In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 469–478.

Rabe-Hesketh, Sophia and Anders Skrondal (2008). "Classical latent variable models for medical research". In: *Statistical methods in medical research* 17.1, pp. 5–32.

Rashidian, Sina et al. (2020). "SMOOTH-GAN: towards sharp and smooth synthetic EHR data generation". In: *International Conference on Artificial Intelligence in Medicine*. Springer, pp. 37–48.

Remy, Philippe (2020). *Conditional RNN for Keras*. `https://github.com/philipperemy/cond_rnn`.

Senin, Pavel (2008). “Dynamic time warping algorithm review”. In: *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855.1-23, p. 40.

Specht, Donald F (1990). “Probabilistic neural networks”. In: *Neural networks* 3.1, pp. 109–118.

Torfi, Amirsina and Edward A Fox (2020). “CorGAN: Correlation-capturing convolutional generative adversarial networks for generating synthetic healthcare records”. In: *The Thirty-Third International Flairs Conference*.

Torfi, Amirsina, Edward A Fox, and Chandan K Reddy (2022). “Differentially private synthetic medical data generation using convolutional gans”. In: *Information Sciences* 586, pp. 485–500.

Van der Maaten, Laurens and Geoffrey Hinton (2008). “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11.

Verdonck, Michaël et al. (2021). “Exploratory Outlier Detection for Acceleromyographic Neuromuscular Monitoring: Machine Learning Approach”. In: *Journal of Medical Internet Research* 23.6, e25913.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). “How to use t-SNE effectively”. In: *Distill* 1.10, e2.

Wong, Kwan Yeung and Fu Lai Chung (2019). “Visualizing time series data with temporal matching based t-SNE”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Wu, Maoqiang et al. (2020). “Evaluation of inference attack models for deep learning on medical data”. In: *arXiv preprint arXiv:2011.00177*.

Yale, Andrew et al. (2019). “Assessing privacy and quality of synthetic health data”. In: *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*, pp. 1–4.

Yang, Fan et al. (2019). “Grouped correlational generative adversarial networks for discrete electronic health records”. In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 906–913.

Yoon, Jinsung, Lydia N Drumright, and Mihaela Van Der Schaar (2020). “Anonymization through data synthesis using generative adversarial networks (ads-gan)”. In: *IEEE journal of biomedical and health informatics* 24.8, pp. 2378–2388.

Yoon, Jinsung, Daniel Jarrett, and Mihaela Van der Schaar (2019). “Time-series generative adversarial networks”. In: *Advances in Neural Information Processing Systems* 32.

Zhang, Kevin, Neha Patki, and Kalyan Veeramachaneni (2022). “Sequential Models in the Synthetic Data Vault”. In: *arXiv preprint arXiv:2207.14406*.

Zhang, Liang et al. (2017). “Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 3120–3128.

Zhang, Ziqi, Chao Yan, and Bradley A Malin (2022). “Membership inference attacks against synthetic health data”. In: *Journal of biomedical informatics* 125, p. 103977.

Ziel, Florian and Rafał Weron (2018). “Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks”. In: *Energy Economics* 70, pp. 396–420.