Towards Building Privacy-Preserving Language Models:
Challenges and Insights in Adapting PrivGAN for Generation of Synthetic Clinical Text

Atena Nazem
Abstract

The growing development of artificial intelligence (AI), particularly neural networks, is transforming applications of AI in healthcare, yet it raises significant privacy concerns due to potential data leakage. As neural networks memorise training data, they may inadvertently expose sensitive clinical data to privacy breaches, which can engender serious repercussions like identity theft, fraud, and harmful medical errors. While regulations such as GDPR offer safeguards through guidelines, rooted and technical protections are required to address the problem of data leakage. Reviews of various approaches show that one avenue of exploration is the adaptation of Generative Adversarial Networks (GANs) to generate synthetic data for use in place of real data. Since GANs were originally designed and mainly researched for generating visual data, there is a notable gap for further exploration of adapting GANs with privacy-preserving measures for generating synthetic text data. Thus, to address this gap, this study aims at answering the research questions of how a privacy-preserving GAN can be adapted to safeguard the privacy of clinical text data and what challenges and potential solutions are associated with these adaptations.

To this end, the existing privGAN framework—originally developed and tested for image data—was tailored to suit clinical text data. Following the design science research framework, modifications were made while adhering to the privGAN architecture to incorporate reinforcement learning (RL) for addressing the discrete nature of text data. For synthetic data generation, this study utilised the 'Discharge summary' class from the Noteevents table of the MIMIC-III dataset, which is clinical text data in American English. The utility of the generated data was assessed using the BLEU-4 metric, and a white-box attack was conducted to test the model’s resistance to privacy breaches.

The experiment yielded a very low BLEU-4 score, indicating that the generator could not produce synthetic data that would capture the linguistic characteristics and patterns of real data. The relatively low white-box attack accuracy of one discriminator (0.2055) suggests that the trained discriminator was less effective in inferring sensitive information with high accuracy. While this may indicate a potential for preserving privacy, increasing the number of discriminators proves less favourable results (0.361).

In light of these results, it is noted that the adapted approach in defining the rewards as a measure of discriminators’ uncertainty can signal a contradicting learning strategy and lead to the low utility of data. This study underscores the challenges in adapting privacy-preserving GANs for text data due to the inherent complexity of GANs training and the required computational power. To obtain better results in terms of utility and confirm the effectiveness of the privacy measures, further experiments are required to consider a more direct and granular rewarding system for the generator and to obtain an optimum learning rate. As such, the findings reiterate the necessity for continued experimentation and refinement in adapting privacy-preserving GANs for clinical text.

Keywords: Generative Adversarial Networks, privacy-preserving language models, clinical text data, reinforcement learning, synthetic data
Synopsis

Background
Advancements in artificial intelligence have revolutionised AI-driven devices and applications, including those in healthcare. However, such developments come with vulnerabilities, notably data leakage. The privacy of clinical data is of utmost concern, as breaches can lead to identity theft, fraud, discrimination, incorrect diagnoses, ineffective treatments, and harm to patients. Thus, adequate measures are required to prevent unauthorised access and preserve data privacy in practice effectively.

Problem
The problem of data leakage is rooted in the memorisation of training data by neural networks. This problem can be exacerbated in larger and more complex neural networks, leaving them vulnerable to private and sensitive clinical data leakage and possible malicious privacy attacks.

Research Questions
How can a privacy-preserving GAN be adapted to safeguard the privacy of clinical text data?
What are the challenges and potential solutions associated with these adaptations?

Method
This study applied the Design Science Research framework to investigate the problem domain and design and develop an artefact by adapting the privGAN framework and its components, including generators, discriminators, and a private discriminator for generating synthetic clinical text data. To handle the data's discrete nature, reinforcement techniques were used. In doing so, elements such as actions, rewards and policy gradients were incorporated into the privGAN framework. Rewards were defined based on the concept of entropy from information theory as an indicative measure of the discriminators' uncertainty. Furthermore, the functionality and performance of the artefact were evaluated via experiments. The generated synthetic text for the Discharge summary class of Note events of pre-processed MIMIC-III was tested for data utility using the BLEU metric. Additionally, a white-box attack was conducted to evaluate the degree to which the model preserves the privacy of the sensitive data under the attack.

Result
A very low BLEU-4 score indicates the low utility of the generated data. The accuracy of 36% in the white-box attack is higher than some of the reported accuracies in the original privGAN study, which is less desirable.

Discussion
Highlights of the discussion include defining a suitable reward system, determining the optimum learning rate, and the required computing power for the experimental nature of GANs. Using the entropy of the discriminator’s prediction as reward might conflict with training objectives. This approach could incentivise generators to seek the discriminator's uncertainty rather than aiming for false positives.

Considering that uncertainty is the end goal and the result of high utility, rewarding uncertainty before the generator can produce high-quality data may be premature and lead to poor data utility. Furthermore, a drastic decrease in the learning rate to control the fluctuations could lead to optimisation failure and, finally, low data utility.
Given that the desired outcome in applications of privacy-preserving GANs is to strike a balance between privacy and utility, it is crucial to foster additional research to enhance the utility of data generated within the adapted privGAN framework. Plausible solutions may involve experimenting with a more direct and detailed reward system that would direct the generation of subsequent tokens in the sequence based on the estimation of higher rewards. Moreover, adopting higher learning rates could benefit better utility.

While this research may shed light on certain areas within the recognised gap, due to concerns about the safety of data, humans, and potential societal repercussions, the artefact should not be adopted.
Acknowledgement

I would like to express my gratitude to my supervisor, Thomas Vakili, for reviewing this work and providing insightful feedback. I also thank Hercules Dalianis for his feedback during the course of this study.

This study acknowledges the Massachusetts Institute of Technology and Google for providing mechanisms for researchers worldwide, including myself, to access necessary clinical data and computational resources that have been instrumental in this study.

Heartfelt thanks to my parents and brothers for their unwavering support.
# Table of Contents

Towards Building Privacy-preserving Language Models ............................. i

1 Introduction ......................................................................................... 0
   1.1 Problem ....................................................................................... 1
   1.2 Research Gaps ........................................................................... 1
       1.2.1 Research Questions .............................................................. 2

2 Extended Background .......................................................................... 2
   2.1 Neural Networks ........................................................................... 2
   2.2 Attacks ......................................................................................... 4
       2.2.1 Black-box vs. White-box Attacks ......................................... 4
       2.2.2 Privacy Attacks .................................................................. 4
   2.3 Privacy Defences ........................................................................ 5
       2.3.1 Generative Adversarial Networks (GANs) ............................ 5
       2.3.2 Other Privacy-preserving Techniques .................................. 7
       2.3.3 Additional Key Concepts ....................................................... 8

3 Methodology ......................................................................................... 9
   3.1 Alternative Methods and Strategies .............................................. 9
   3.2 Application of the DSR Method Framework ................................. 10
       3.2.1 Explicating the Problem ....................................................... 10
       3.2.2 Defining Requirements ......................................................... 10
       3.2.3 Artefact Design and Development ....................................... 10
       3.2.4 Demonstrating the Artefact .................................................. 14
       3.2.5 Evaluation of the Artefact ..................................................... 18
   3.3 Ethics ............................................................................................ 19

4 Results ................................................................................................ 21

5 Discussion ........................................................................................... 22
   5.1.1 Conclusion .............................................................................. 25
   5.1.2 Limitations .............................................................................. 26
   5.1.3 Future Studies ........................................................................ 27

References .............................................................................................. 28

Appendix A – Project’s code .................................................................... 31
Appendix B – Generated data sample ..................................................... 45
List of Figures

Figure 1: Illustrates PrivGAN architecture ................................................................. 15
Figure 2: Count of clinical notes per category in Notevents dataset of MIMIC-III after cleaning .... 16
Figure 3: Count of clinical data per Category after merging groups and combining the texts........ 16
Figure 4: Log of loss values during training modified privGAN on MIMIC-III ......................... 21
Figure 5: The accuracy of the private discriminator across 175 epochs................................. 22
Figure 6: Comparing the trend of recorded losses of combined GAN between the two studies....... 23
List of Tables

Table 1: Review of privacy-preserving GANs................................................................. 7
Table 2: List of hyperparameters used in this study in comparison to privGAN and seqGAN. .......... 18
Table 3. Comparing the results of max scores with the closest reported scores of white-box attack.... 24
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
<td>Bilingual Evaluation Understudy</td>
</tr>
<tr>
<td>BERT</td>
<td>Bidirectional Encoder Representation for Transformers</td>
</tr>
<tr>
<td>CLMs</td>
<td>Clinical Language Models</td>
</tr>
<tr>
<td>CNNs</td>
<td>Convolutional Neural Networks</td>
</tr>
<tr>
<td>DSR</td>
<td>Design Science Research</td>
</tr>
<tr>
<td>DPGAN</td>
<td>Differentially Private Generative Adversarial Network</td>
</tr>
<tr>
<td>EMRs</td>
<td>Electronic Medical Records</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>GANs</td>
<td>Generative Adversarial Networks</td>
</tr>
<tr>
<td>GPT</td>
<td>Generative Pretrained Transformer</td>
</tr>
<tr>
<td>LSTMs</td>
<td>Long Short-Term Memory Networks</td>
</tr>
<tr>
<td>MIMIC</td>
<td>Medical Information Mart for Intensive Care</td>
</tr>
<tr>
<td>MIA</td>
<td>Membership Inference Attacks</td>
</tr>
<tr>
<td>MCTS</td>
<td>Monte Carlo Tree Search</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>NER</td>
<td>Name Entity Recognition</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>NNs</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>PPGAN</td>
<td>Privacy-preserving Generative Adversarial Network</td>
</tr>
<tr>
<td>PGAN</td>
<td>Private Generative Adversarial Network</td>
</tr>
<tr>
<td>PHI</td>
<td>Protected health information</td>
</tr>
<tr>
<td>RF</td>
<td>Random Forest</td>
</tr>
<tr>
<td>RNNs</td>
<td>Recurrent Neural Networks</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
<tr>
<td>seqGAN</td>
<td>Sequence Generative Adversarial Network</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>
1 Introduction

The field of artificial intelligence, particularly the study of neural networks (NN), has witnessed extensive advancement in recent years and led to the development of a myriad of AI-powered applications that span various aspects of life, including communication (Hohenstein et al., 2023), commerce (Ozbayoglu et al., 2020; Song et al., 2019), education (Gillani et al., 2023), and healthcare (Coccia, 2020). This widespread integration of neural networks into various facets of society subsequently led to increasing dependence on AI-driven technologies (Makridakis, 2017).

While these improvements have contributed to advancements in accessibility, quality, speed, and accuracy in completing large and various tasks, they are accompanied by potential vulnerabilities and side effects that could lead to serious consequences depending on the realm of the application. One of the vulnerabilities of neural networks concerns the issue of data privacy.

Data privacy is particularly significant in the healthcare domain where practices of stakeholders involve dealing with personal data such as name, address and identification number as well as sensitive information such as medical history, diagnoses, treatments, and medications. If this information is compromised, it can lead to identity theft, financial fraud, and discrimination.

For example, if a patient's healthcare records containing their financial information, such as insurance details, are leaked, malicious actors can use this information for financial gain. They may file false insurance claims or access the patient's bank account information to make fraudulent transactions. Furthermore, employers, insurance companies, or other organisations may take advantage of the compromised sensitive medical information to discriminate against the patients.

Additionally, unauthorised access to clinical data may result in incorrect diagnoses, ineffective treatments, or even harm to the patient's health. Consequently, compromised privacy of clinical data may lead to a loss of patients’ trust and confidence in healthcare providers and potentially lead to decreased utilisation of healthcare services. Therefore, safeguarding the privacy of clinical data is crucial for ensuring patients’ well-being, maintaining trust in the healthcare system, and preventing harm to individuals and society at large.

To this end, legislative bodies worldwide considered the necessary measures for protecting personal and sensitive data. In the European Union, the General Data Protection Regulation (GDPR) provides strict guidelines for the processing and protection of personal data. While such regulations provide necessary and legal guidelines for preserving the privacy of personal data, in practice, the implementation of further complex measures is required to ensure that the privacy of data is preserved in accordance with the regulations. This may include the use of techniques such as data encryption and deidentification, access controls, and staff training on data handling and security protocols.

As digitalisation and the use of AI-driven applications become more prevalent, preserving the privacy of private and sensitive data presents significant challenges to which healthcare systems are not immune. Modern healthcare increasingly relies on clinical natural language processing applications to detect patterns, diagnose diseases, and address medical complexities. For example, Fang et al. (2022) developed a machine-learning model to automatically extract clinically named entities for pituitary adenomas from Chinese electronic medical records (EMRs). The study demonstrated the potential for machine learning models to automatically extract information from unstructured clinical texts in
EMRs. In another study, Savova et al. (2019) highlight the potential for using natural language processing (NLP) methods to extract cancer phenotypes from clinical text and provide an overview of recent advances in NLP and information extraction methods relevant to oncology. These studies are only instances from a vast pool of possible benefits and numerous applications of NLP within healthcare. Such applications pose a new nature of privacy risk that comes with the use of neural networks. That is data leakage.

Jagannatha et al. (2021) explored the risks of training data leakage in a study on MIMIC-III, UMM, and VHA hospital datasets. They examined the implications of applying clinical language models (CLMs) for data privacy and demonstrated that deep learning models such as BERT and GPT are vulnerable to breaches of data. The study proves that attacks on CLMs can result in notable breaches of privacy by demonstrating that up to 7% of sensitive information could potentially be disclosed.

1.1 Problem

Data leakage, referring to the unauthorised and accidental transfer or access to sensitive or confidential information, is described as involuntary leakage. Studies of involuntary leakage in NNs showed that unintended memorisation could occur in a wide range of NN architectures and datasets, resulting in significant privacy risks (Carlini et al., 2019). In comparison, in malicious leakage (also known as privacy attacks), a malicious agent intentionally attempts to gain unauthorised access to sensitive data by taking advantage of the model’s ability to learn the data. Studies on the issue of data privacy associated with neural networks substantiate that memorisation and overfitting are the key underlying factors that contribute to the generic problem of data leakage (Jegorova et al., 2022).

Additionally, the large sizes of deep neural networks contribute to their propensity to overfit and, consequently, memorisation. As larger models have more parameters and thus are more complex, they are more capable of fitting precisely to the training data. However, this increased complexity also makes them more susceptible to overfitting, which occurs when the model becomes too specialised to the training data and cannot generalise well to new data. In such cases, the model memorises the training data as it assigns the data with a significantly higher likelihood than expected by random chance (Jegorova et al., 2022). This can lead to privacy vulnerabilities as it may expose sensitive information about individuals. Therefore, addressing the root problem of memorisation in large neural networks is crucial for developing effective privacy-preserving machine learning techniques.

Thus, the problem can be formulated as follows:

Memorisation of training data by neural networks leaves them vulnerable to leakage of private and sensitive clinical data and malicious privacy attacks.

1.2 Research Gaps

In reviewing privacy-oriented studies, Jegorova et al. (2022) identify a substantial research gap concerning leakage from classifiers trained with tabular/mixed features and recognise a limited amount of research on preventing memorisation. They further underscore the need for computationally efficient defences capable of offering a wide range of privacy assurances to address various privacy concerns. They also note that existing defence strategies tend to be tailored to specific cases and face the challenges of scaling to larger datasets while maintaining good performance.
Reviewing numerous strategies for managing data privacy across a variety of contexts, Jegorova et al. (2022) regard the use of synthetic data instead of real data as a ‘promising’ approach. One method for the generation of synthetic data is the use of generative adversarial networks (GANs). GANs are conceptually based on adversarial learning. Notably, the number of adversarial studies in NLP in comparison to computer vision is significantly less and even less so with respect to the generation of textual adversarial examples (Alshemali & Kalita, 2020; Qiu et al., 2022). This can be explained by the discrete nature of textual data, which may render the application of GAN-based defence methods for text data more challenging and less explored (Alshemali & Kalita, 2020).

To address the explicated problem, considering the identified research gaps above, this study aims to explore the generation of synthetic data via a privacy-preserving GAN for use in lieu of real English clinical text. This approach can potentially introduce further complexities and vulnerabilities, which may produce a basis for further discussion and research.

### 1.2.1 Research Questions

Thus, this study strives to understand:

**How can a privacy-preserving GAN be adapted to safeguard the privacy of clinical text data?**

**What are the challenges and potential solutions associated with these adaptations?**

Answering the questions above could inherently address the problem of data memorisation. If successful, the adaptation of privacy-preserving GAN to clinical text data to generate synthetic data could help address data leakage and breach of sensitive information when using classifiers. Additionally, exploring the challenges and potential solutions encountered in the process provides insights for further studies.

### 2 Extended Background

#### 2.1 Neural Networks

A neural network is a type of machine learning algorithm which consists of a series of interconnected layers of neurons. The connection of neurons between layers is established with a set of weights, which determines the strength of the connection between them. During the training, these weights are adjusted to minimise the prediction error. This is referred to as learning, which is expected to result in more accurate predictions on new, unseen data. In their typical multilayered architecture, neural networks have three main layers, including input, hidden and output. There are two distinct processes for training a neural network, namely, forward and backward passes (Sarker, 2021).

During a forward pass, the input data in any form (e.g., text, image, audio) is fed into the input layer with as many neurons as the number of features in the data. The sole purpose of this layer generally is to receive raw data and pass it on without any transformations to the hidden layers where the processing occurs. Next, the neurons in the hidden layers receive sum of weighted inputs from previous layers, i.e., ‘S’ in (1), and apply an activation function to the results. i.e., ‘A’ in (2). The activation function can vary depending on the tasks that the neural network is performing. The most
common activation functions are ReLU \(^1\) and the sigmoid function \(s\). The output of each neuron is then passed to neurons in the next hidden layer for further processing. The forward pass is repeated until the output is produced. The output is then compared to the target \((y')\), and differences between them are measured in terms of a loss function (Sarker, 2021).

\[
\begin{align*}
S &= b + \sum_{t=1}^{n} w_t \times x_t \quad (1) \text{Sum of weighted inputs} \\
Y &= A(s) \quad (2) \text{Activation function}
\end{align*}
\]

During the backward pass, the gradient of the loss function with respect to the output of the network is first computed (see (3)) before computing the gradients of the loss function with respect to weights and biases of the network are calculated before recursively multiplying the gradients from the next layer by the weights of the current layer (see (4) and (5)). This process is done from the output layer towards the input layer, hence referred to as backpropagation. Next, the calculated gradients are used to update the network parameters with optimisation algorithms such as SGD \(^2\) or Adam. This process is repeated for each batch of training sample until the network converges into a set of weights and biases that could produce accurate predictions.

\[
\begin{align*}
\frac{dL}{dy} &= \left(\frac{dL}{dy'}\right) \times \left(\frac{dy'}{ds}\right) \quad (3) \text{Gradient of L with respect to output (y')} \\
\frac{dL}{dw} &= \left(\frac{dL}{dy}\right) \times \left(\frac{dy}{ds}\right) \times \left(\frac{ds}{dw}\right) \quad (4) \text{Gradient of L with respect to weight (w)} \\
\frac{dL}{db} &= \left(\frac{dL}{dy}\right) \times \left(\frac{dy}{ds}\right) \times \left(\frac{ds}{db}\right) \quad (5) \text{Gradient of L with respect to bias (b)}
\end{align*}
\]

There are varieties of neural networks with some modifications in their architecture, which can make them fit various purposes for processing particular types of data. These varieties include CNNs, RNNs, and transformers. CNNs are more commonly used for image processing and computer vision tasks, and their architecture benefits convolutional layers to learn features from input data and pooling layers to reduce spatial dimensionality while capturing the most important information. This allows CNNs to learn local patterns in the images and be able to utilise the captured features for identifying objects (Rao & McMahan, 2019; Sarker, 2021).

On the other hand, NNs such as RNNs and transformers are commonly used in NLP tasks, and they both can process sequential data such as text, speech and music. RNNs use recurrent connections to pass information from one step to another, as well as hidden states that update and maintain a memory of the previous step (Goodfellow et al., 2016, p. 376). Transformers, on the other hand, use a self-attention mechanism to process sequential data, which allows them to attend to all previous steps in a sequence at once (Vaswani et al., 2017).

This makes them more efficient than RNNs and facilitates capturing a long range of contextual dependencies. Additionally, transformers have been shown to perform very well in terms of generalising to unseen data and reducing overfitting, particularly on large datasets compared to traditional recurrent neural networks (RNNs). All these arguably led to the recognition of transformers

\[^{1}\text{Rectified Linear Unit}\]
\[^{2}\text{Stochastic Gradient Descent}\]
2.2 Attacks

Numerous studies have discussed various attack categories on different grounds, including access to the model, the purpose of the attack, and controlled output (Alshemali & Kalita, 2020). Below, a few categories that are within the scope of this study are discussed.

2.2.1 Black-box vs. White-box Attacks

White-box attacks are a type of adversarial attack where the attacker has complete knowledge of the target model, including its architecture, parameters, inputs, and outputs. With this knowledge, the attacker can design targeted attacks to exploit vulnerabilities in the model, sabotaging its integrity or data privacy. On the contrary, in black-box attacks, the attacker only has knowledge of the output of the target model. Thus, they rely on techniques such as input manipulations, model probing or inversion to infer necessary information.

The choice of white- and black-box attack depends on the information available to the attacker about the victim model, the goal of the attack and the victim model properties. White-box attacks may be more effective because the attacker has access to information about the target model, such as its algorithm and parameters. Comparing black-and-white box inference models, Papernot et al. (2018) attribute stronger security attacks to white-box variety. Although some studies (Papernot et al., 2018) considered white-box threat models less realistic than their black counterparts for privacy attacks, with increasingly more open-source models available on public repositories such as Huggingface, white-box attacks are becoming a very real and common threat (Huang et al., 2023).

2.2.2 Privacy Attacks

Privacy attacks aim to violate the confidentiality of data, which can lead to unauthorised access to sensitive information. Examples of privacy attacks include membership inference attacks (MIAs), where an attacker aims at determining whether a particular data point was part of the training dataset of a machine learning model, and property or attribute attacks, in which the attacker aims at inferring the attribute of the data. (Nasr et al., 2019). The former can be referred to as tracing, the accuracy of which determines how much of training data has been leaked, while the latter is also referred to as reconstruction attacks.

Depending on how the attack is designed, various types of MIAs are recognised and discussed (Shejwalkar et al., 2021; Shokri et al., 2017). MIAs could be carried out during both the inference and training phases. Inference (test) time attacks occur during a forward pass, where the model makes predictions from novel data. In such cases, the attacker can introduce adversarial examples or manipulate the input data to cause the network to make incorrect predictions or leak sensitive information. In contrast, attacks that are possible during the backward pass are less common and typically require access to the training data and knowledge of the training algorithm. The dichotomy of training and inference time privacy attacks can be considered significant as certain defence mechanisms may be more effective and suitable for different scenarios. For example, techniques such as differential privacy are commonly used to defend against attacks during training, while techniques
such as input perturbation and output perturbation are commonly used to defend against attacks during inference (Jagannatha et al., 2021; Jegorova et al., 2022).

2.3 Privacy Defences

Privacy-preserving solutions can be broadly categorised into data-driven and model-driven defences. In sectors such as medicine, which contain domain-specific data, it might be more appropriate to emphasise data-driven approaches. This is because generic model-driven mechanisms might not capture intricate linguistic nuances, such as text homographs, Out-Of-Vocabulary terms and abbreviations inherent in clinical data. Thus, while this study emphasises GANs, it briefly explores some other data-driven solutions.

2.3.1 Generative Adversarial Networks (GANs)

A data-driven approach to addressing the issue of privacy is training or fine-tuning a model with synthetic data generated by GANs. Originally purposed for generation of synthetic data to address the issue of limited data availability, GANs-based defences have been explored for addressing data privacy by allowing the use of synthetic data in place of real data (Jegorova et al., 2022; Lu et al., 2023).

GANs architecture normally consists of a generator and discriminator. A generator, typically a fully connected network, receives a latent representation of a point in high dimensional space whose values are generated randomly and maps these values to the realistic data through training to generate synthetic data. Then, both real and synthesised data are fed to the discriminator to distinguish the two by outputting a probability value of the likelihood that the data is real. Thus, the generator and discriminator are trained in a feedback loop such that the discriminator would not reliably distinguish between real and synthesised data and outputs a probability of 0.5. To make synthetic data as real as possible, the training continues to achieve the desired threshold. This is referred to as a state of equilibrium (Goodfellow et al., 2016; Sarker, 2021).

The generated synthetic data can then be applied in various downstream applications, one of which includes training of classifiers. This approach minimises, if not eliminates, the need to share the original dataset for privacy or security concerns. In the same light, Croce et al. (2020) proposed GAN-BERT architecture and proved that finetuning the model with as little as 50-100 labelled examples would maintain good accuracy and F1 performance in various downstream tasks.

With this in mind, use of generative models with limited data may address the issue of overfitting by transformers. Considering that overfitting can contribute to the vulnerability of models to leak data, GANs could potentially be considered a solution to improve privacy issues by generating synthetic data that could be used for data augmentation or replacing the original data. Thus, they can particularly be beneficial for clinical research because, due to their sensitivity, sharing and accessing clinical data for research is limited and delayed if at all possible.

To this end, Venugopal et al. (2022) presented pGAN that can create high-quality synthetic data while keeping the privacy and statistical properties of the original data intact. The authors have implemented a number of dropouts in the structure of the discriminator to prevent overfitting. Using this technique, random neurons in the neural networks are dropped out by ignoring their output. This leads to more generalisable learning by other neurons. The effectiveness of pGAN was subsequently assessed on two datasets used for classification and regression tasks, respectively. On both the synthetic and original
data, the machine learning models SVM, RF and MLP achieved similar levels of accuracy for the classification. The result showed that the pGAN model succeeded in producing synthetic data similar to the original data while maintaining high utility and preserving privacy. To measure privacy, the researchers used Nearest Neighbour Adversarial Accuracy (NNAA), where they relied on nearest neighbour and distance measures such as Euclidian to calculate privacy. They resemble this metric to the result of an adversarial classifier in discerning real and synthetic data. Since this study did not utilise MIAs on the pGAN, it is not clear if the privacy score achieved by this metric can safeguard the model in the face of MIAs.

Nevertheless, other studies (Hayes et al., 2017; Hilprecht et al., 2019) have reported GANs to be vulnerable to MIAs. As they, too, tend to memorise the data. This added to existing privacy concerns, which led to proposing more robust privacy preserving editions of GANs in further studies.

In this light, Xie et al. (2018) proposed DPGAN, which achieves privacy protection by adding noise (the amount of which is determined by a privacy budget parameter, epsilon (ε)) to the gradient of the Wasserstein distance during training rather than directly to the final parameters. Hence, both the generator and discriminator can guarantee differential privacy. Although this seems to prevent an attacker from using the output of the discriminator to infer information about individual training data points, there is a trade-off between privacy levels and the ability of generative models to capture the inter-dimensional relationship in the data, i.e., learning performance (Xie et al., 2018).

In a similar study, Liu et al. (2019) implement a Gaussian noise mechanism to randomly generate noise and add it to the gradient of the Wasserstein distance. They showed that, compared to DPGAN, PPGAN generates data with relatively higher quality as the privacy level increases.

To improve privacy level while maintaining high performance, Mukherjee et al. (2021) adopted a different design in privGAN, in which the generator defends against membership inference attacks while deceiving the discriminator with perturbed data. Mukherjee et al. (2021) show that their algorithm prevents the memorisation of training data sets. They trained the model on image datasets and tested it against several state-of-the-art black-and-white box attacks. The result showed that privGAN could generate high-quality synthetic image data while securing data privacy without considerable performance loss.

Although the majority of reviewed studies (see Table 1) provide solutions for preserving the privacy of data in the context of GANs on electronic health records, none are applied to clinical text data to provide an understanding of addressing the discrete nature of the data while preserving privacy. Nonetheless, Mukherjee et al. (2021) claim that privGAN architecture is applicable to other data types without mentioning any particular type. However, as privGAN is based on simple GANs architecture for image data, it is safe to assume that using this design with text data would pose additional challenges as it does not address the discrete nature of data. That means during backpropagation, when the model uses gradients to adjust the weights, there is not a continuous and small transition (gradient) between one token to the other, which means that small adjustments in gradients cannot translate to alternation of tokens to update the generators. To circumvent such challenges, Yu et al. (2017) applied reinforcement learning (RL) techniques to reward the generators based on the outcome of the discriminator for each complete sequence. They applied seqGAN for Chinese and English text generation on corporuses 16,394 quatrains and 11,092 paragraphs, showing BLEU scores of 0.667 and 0.416, respectively. Considering BLEU ranges between 0 and 1, a BLEU score of 0.667 is considered quite high, suggesting that the generated text closely matches the reference text in the case of Chinese text generation. However, since seqGAN does not include privacy-preserving measures, it can be
vulnerable to privacy breaches. Hence, perhaps a privacy preserving GAN could adapt features of privGAN and seqGAN to obtain both privacy and utility for text data.

<table>
<thead>
<tr>
<th>Study</th>
<th>Design</th>
<th>Solution</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mukherjee et al. (2021).</td>
<td>PrivGAN</td>
<td>Random distribution of data among multiple G-D pairs and fooling built-in adversary about the origin of fake data</td>
<td>Image</td>
</tr>
<tr>
<td>Liu et al. (2019).</td>
<td>PPGAN</td>
<td>Gaussian noise mechanism; Moments Accountant strategy</td>
<td>Image</td>
</tr>
<tr>
<td>Xie et al. (2018).</td>
<td>DPGAN</td>
<td>Adding noise to the gradient of W-distance during training</td>
<td>Image and numeric/categorical</td>
</tr>
<tr>
<td>Venugopal et al (2022)</td>
<td>pGAN</td>
<td>Dropouts, batch normalisation</td>
<td>Numeric/categorical</td>
</tr>
</tbody>
</table>

Table 1: Review of privacy-preserving GANs shows that the implemented solutions were not applied to text data.

2.3.2 Other Privacy-preserving Techniques

Other data-driven mechanisms could include data obfuscation to mask and scramble; and sanitisation to overwrite sensitive information with perturbed data. Additionally, removing sensitive data has been considered a viable solution as long as it entails maintaining the integrity of data and avoiding further privacy risks by revealing patterns of missing data (Jegorova et al., 2022).

In a series of data-driven studies, Vakili et al. (2022) and Vakili and Dalianis (2022) proposed a method for preserving privacy called automatic de-identification, which is a machine learning technique to detect and remove Protected health information (PHI) on a large corpus of Swedish clinical text. In their study of evaluating pre-trained BERT performance on downstream tasks, Vakili et al. (2022) proposed removing sentences that showed high PHI saturation in addition to replacing sensitive information with their realistic approximates (pseudonymised terms) before training two BERT varieties separately on each modified set of data. Evaluating the models on six downstream tasks, including name entity recognition (NER) and classification, showed high F1 scores for KB-BERT + Pseudo. Thus proving that training BERT on pseudonymised data by utilising the de-identified methods not only maintains good performance but also safeguards data utility while preserving privacy.

However, de-identification relies on NER (Vakili et al., 2022). This can introduce some challenges specific to NER, such as entity coverage. That is, the effectiveness of NER models relies on their ability to recognise a diverse set of entities (labels). Untagged entities during the training data may lead to an oversight of these entities during de-identification, risking unintended data leaks. In contrast, synthetic data generation methods do not face such limitations. Rather than trying to identify and redact real-world entities, synthetic data generation focuses on replicating the statistical properties of the original dataset (Lu et al., 2023).
2.3.3 Additional Key Concepts

The review of previous studies on adapting GANs for discrete data highlights the importance of discussing a few key concepts, such as RL, reward and entropy, that may be referred to later.

RL is a machine learning paradigm characterised by automating goal-directed decision-making in which an agent learns by interacting with its environment. The environment provides reward signals for each action based on the defined policies. This leads to a new state. The ultimate goal of the agent is to learn a policy or strategy based on which it would take actions such that they would maximise these rewards over the long term (Sutton & Barto, 2018).

In the context of GANs, the generator can be identified as the agent that would take actions in order to generate the sequences. The state is each generated sequence or batch of sequences that warrants feedback from the environment. The environment is the discriminator that generates the feedback. The feedback is referred to as a reward signal. In this setting, the generator’s goal is to maximise the reward it receives from the discriminator. Thus, it is important to define a reward and policy that would align with the learning strategy in GANs.

The ultimate goal in GANs is to achieve a state of equilibrium where the discriminator reaches a state of uncertainty about the authenticity of the data. Thus, the rewards can be defined based on the discriminator’s uncertainty \( r \). Uncertainty can be quantified in terms of entropy from information theory as seen in (6), where \( A \) is the set of outcomes and \( P(x) \) is the probability of outcome \( x \):

\[
    r(P) = -\sum_{x \in A} P(x) \log P(x) \tag{6}
\]

Entropy\(^3\) quantifies the required number of bits needed to encode the outcome of a random event (Mackay, 2003, pp. 32 and 68). In a binary classification setting, when one class is distinct from another, the entropy is 0. Which means no bits are required for encoding the class. When classes are less distinguishable, the entropy approaches 1.

Considering the discriminator’s outcome, which is the probability of the data being real or fake, the higher the entropy would mean that the discriminator is most uncertain \( (p = 0.5) \). When the discriminator assigns a probability \( p \) to the event that a given input is real \( (\text{and } 1-p \text{ that it's fake}) \), the entropy associated with this decision would be (7).

\[
    r = -[p \log (p) + (1-p) \log (1-p)] \tag{7}
\]

In light of the above discussions, the policy gradient from RL can be used to determine the generator’s policy. Its objective is to optimise the generator's parameters to maximise the expected reward (Sutton & Barto, 2018). Given an action \( \mathcal{a} \) and a reward \( r \), aiming at maximising the expected reward under the policy \( \pi \), the policy gradient can be defined in equation (8).

\[
    \nabla_{\theta} J(\theta) = E_{\mathcal{a}} \sim \pi_{\theta}[r, \nabla_{\theta} \log \pi_{\theta}(\mathcal{a}]] \tag{8}
\]

\(^3\) Throughout this paper, the term 'entropy' is primarily used in the context of defining rewards as a measure of discriminator uncertainty based on information theory, as described earlier. It should not be confused with the entropy-based loss functions utilised for the discriminators.
3 Methodology

This study is a design research study as it is characterised by aiming at developing an artefact to address a problem in the world in addition to generating contextual knowledge about the artefact. As such, it can be contrasted with empirical study, which is only characterised by the collection and analysis of data, drawing evidence-based conclusions, and advancing knowledge (Johannesson & Perjons, 2014; Neuman, 2014). In order to choose a suitable framework, the requirements of the study should be considered. As this study sets off to design a defence mechanism that would contribute to the robustness of a language model to address the problem of data privacy with clinical data, design science research (DSR) is deemed a suitable research framework on the ground that it is problem-driven, structural and iterative process to develop, refine and improve an artefact as a solution, while valuing relevant and rigorous research to the question at hand. Activities outlined throughout the DSR stages are accompanied by various research strategies.

3.1 Alternative Methods and Strategies

In comparison, soft design science developed by Baskerville et al. (2009) is a practice and situation-oriented method whose outcome is expected to be an organisational improvement and thus is more suitable for improving conditions in organisations, specifically in relation to social aspects (Dresch et al., 2015, p. 87). This approach integrates the conventional design science research process (design, build, evaluate) with the iterative soft systems methodology, which is a type of action research and allows for improving human organisations through the general activities of design, development, instantiation, and naturalistic evaluation of the developed technological artefact. Considering that this study is not situation-oriented and is not intended for improving conditions in specific organisations, soft design science is deemed less suitable and hence disregarded. In choosing research strategies, three criteria of suitability, feasibility and ethics were considered throughout all the stages of the DSR framework. Suitability concerns whether the adapted research strategies can be helpful in finding a solution (Johannesson & Perjons, 2014). With this in mind, two strategies of simulation and experiment have been compared to select a suitable strategy.

In an experiment, researchers can manipulate variables, control for confounding factors, and collect data under specific conditions to observe how different factors affect the outcome of interest. This allows them to test hypotheses, establish cause-and-effect relationships, and generate new empirical data that can be used to build new theories or refine existing ones. On the other hand, in a simulation, the produced results are based on a set of premises that are programmed into the simulation (Johannesson & Perjons, 2014). Unlike simulations, experiments are not limited by premises. Instead, experiments allow researchers to directly observe and measure the effects of specific variables in real-world situations, which can lead to new insights and discoveries.

As this study concerns how to adapt a privacy-preserving artefact for a different type of data, an experimental research strategy is deemed suitable as it allows for controlling the factors such as model hyperparameters, collecting and observing data to implement and test whether the suggested adaptations meet the predefined specified requirements.

While experiment as a research strategy can be deemed suitable for demonstrating and evaluating the artefact, for other activities within the DSR framework with different sets of purposes, requirements
and outcomes, methods such as document studies, including research surveys, reviews and articles, were deemed more suitable.

3.2 Application of the DSR Method Framework

3.2.1 Explicating the Problem

The activity of explicating problems entails examining and analysing a practical issue that must be precisely defined and justified by demonstrating its significance to a particular practice. It is important that the problem is significant not only to a local practice but also to a global practice. Moreover, the underlying causes of the problem should be identified and analysed (Johannesson & Perjons, 2014).

Following this step of design science research in this study, first, the issue of susceptibility of private data to leakage or unauthorised access and its significance to clinical practice was identified and analysed. Moreover, some of the root causes of this problem were identified in characteristics of neural networks, such as memorising the data.

By adapting document studies as a research method in this step, various resources were considered and utilised. To obtain the required knowledgebase for explicating the problem and later defining the requirements prior to the design and implementation of the artefact, resources such as books, review articles (surveys), and journal articles on scientific databases have been searched and consulted as part of a systematic literature review.

3.2.2 Defining Requirements

The activity of defining requirements serves to propose a solution to the problem that was explicated in the form of an artefact. This activity also involves eliciting requirements, which can be considered a transformation of the problem into demands on the proposed artefact (Johannesson & Perjons, 2014).

As no stakeholders were involved in this study for eliciting requirements, the requirements below are listed based on addressing the problem.

1. Privacy: The artefact should be able to preserve the privacy of clinical text data.
2. Data integrity: The artefact should not distort the integrity and quality of the original data.
3. Utility: The artefact should not distort or diminish the performance of the classifier significantly.
4. Computational power: Employing the artefact with a large volume of data should be feasible in terms of required computational power.

3.2.3 Artefact Design and Development

During the design and development phase, an artefact is designed and developed such that it can solve the identified problem and meet the specified requirements. This involves determining the functionality and structure of the artefact (Johannesson & Perjons, 2014).

Reflecting on the design decision, as part of the design process activity, sheds light on some possible vulnerabilities of GANs in the face of various MIA attacks. To address the issue, this study adapts the privatised generative adversarial network architecture as introduced and developed by Mukherjee et al. (2021). The idea is that the use of this architecture would address the common privacy concerns caused by memorisation regarding the use of GANs while simultaneously eliminating the need to access real data directly for running downstream tasks with classifiers such as BERT. In such a scenario, possible inference attacks on any subsequent classifier would not jeopardise the privacy of
clinical data as the classifier would be finetuned on synthetic data. Thus, to access and infer sensitive data, the malicious user must perform an attack on PrivGAN. However, since the suggested characteristics of PrivGAN have proved to prevent memorisation of the data in the face of several tested state-of-the-art black- and white-box attacks (Mukherjee et al., 2021), the privacy of clinical data would be preserved.

The adapted structure of privGAN for the artefact in this study aims to make it suitable for safeguarding the privacy of clinical text data. The original privGAN consists of regular generators and discriminators that are suitable for images and only differs from regular GANs in terms of number of generators discriminators pairs (N), a privacy discriminator (Dp), and privacy-utility hyperparameter (λ). This hyperparameter is used as a factor of privGAN weight to signal the generators to either prioritise the feedback of regular discriminators for utility or that of private discriminator for privacy. In this study, as the type and nature of text data are sequential and discrete while adhering to the architecture of privGAN to maintain intended privacy measures, some solutions are adapted that are in line with seqGAN (Yu et al., 2017). This was aimed at allowing the gradients to flow back and update the weights of the generator (see Algorithm 1). The design for each component is elaborated as follows.

Algorithm 1: PrivGAN-text. The modifications in privGAN training includes addition of rewards \( r_j \) from discriminators and privacy discriminator \( r_{jp} \) and action probabilities \( a_{prob} \) from generators to implement reinforcement techniques.

---

4 privGAN/privacygan/privacy_gan.py at main · microsoft/privGAN · GitHub
3.2.3.1 Generator

The generator was designed such that it would capture the temporal dependencies between words while it would effectively manage long sequences through LSTMs, map tokens into a more expressive space (embeddings) and allow controlling the creativity of the generated text (via temperature).

The temperature parameter similar to seqGAN controls the randomness of predictions by scaling the logits before applying SoftMax\(^3\) (Yu et al., 2017). Higher temperature values produce more random outputs, while lower temperature values produce more deterministic outputs. The model is, therefore, more likely to choose the most probable next token. This can make the output seem more realistic and coherent, but it can also make the model less creative. However, this parameter can be modified based on preliminary results to achieve the desired quality of the data (1.0 in this study). Additionally, the generator calculates and stores the probabilities of each action (chosen token) at every step of sequence generation. This leads to generating a batch of 10 sequences with a length of 126 tokens along with action probabilities, which are meant for implementation of reinforcement technique within privGAN architecture. The choice of sequence length was based on the objective to capture as much of the dependencies within each instance (a patient record) as possible, while a batch size of 10 would lead to the model updating its weights more frequently. These values contrast with the choices in some studies with sequence lengths of 20 and batch sizes of 64 and 128 (Jozefowicz et al., 2016a; Yu et al., 2017).

The number of LSTM layers can be changed depending on the available resources; higher numbers can improve the quality of generated data while requiring more computational resources (Jozefowicz et al., 2016a). In this experiment and considering the available resources and computation costs, only two layers were possible. The model uses the Adam optimiser with gradient clipping to ensure stable training. In the standard GAN architecture, the generator itself doesn't have a specific loss function. Its performance is assessed indirectly through the discriminator. The feedback from the discriminator, which is gradients from the loss function, is backpropagated during training. In this study, to be able to test the generator in isolation with dummy data and in combination with one discriminator to ensure its functionality, Binary Cross Entropy (BCE) is used as a loss function with Adam optimiser next to policy gradient loss to ensure gradients of the generator are computed before running the generator with the same loss functions within the combined model with real data.

Additionally, the generator model implements a 10% dropout rate on LSTM layers (Jozefowicz et al., 2016a). This means that during each update, a fraction of LSTM units is set to 0, thereby helping to prevent overfitting and improving generalisation. Setting the correct rate of dropout is crucial. Too high a rate can lead to underfitting and poor model performance, while too low a rate might not provide sufficient regularisation. The strategy in choice of parameter values within the available computational resources was to ensure better quality synthetic data. Nonetheless, further finetuning of parameters via experimentations is required to achieve the optimum solution.

3.2.3.2 Discriminator

Similar to the architecture of the generators, discriminators need to accommodate the nature of data. Hence, they share many characteristics in terms of embedding layers, LSTMs, and dropout rates to prevent overfitting. In addition to overfitting, dealing with varying sequence lengths is addressed by the use of global average pooling. This pooling operation calculates the average of the outputs of

\(^3\) https://github.com/LantaoYu/SeqGAN/blob/master/generator.py
LSTM units over the time-steps dimension, which reduces the number of parameters in the next layers, thus contributing to lowering the model’s complexity.

This average vector is then passed to a dense layer, which generates a single output score for each sequence. This score is then used for the binary classification task to determine whether each sequence is 'real' or 'fake'. For such tasks, Binary Cross-Entropy (BCE) Loss is normally used (Goodfellow et al., 2016, p. 183). In this study, the discriminator is expected to discern each generated sequence from the real sequences in its entirety and make a single binary decision. Hence, BCE was used.

3.2.3.1 Privacy Discriminator

In the architecture of the privacy discriminator, a CNN is adapted for text data. In this design, similar to the discriminator design of seqGAN (Yu et al., 2017), after embedding the layer, a series of 1D convolutional layers with different filter sizes are incorporated such that they could capture local features or n-grams of different lengths. Each convolution is followed by a RELU activation function. For each convolution layer, a global max pooling operation is performed. This operation reduces the output of the convolutional layer to its maximum value, capturing the most important feature. After the pooling operation, the outputs from the different convolutional layers are concatenated and then flattened to form a single long feature vector. A dropout layer of 10% is then applied for regularisation to prevent overfitting during the training process. Finally, a dense layer with two generators is used to output a separate score for each generator. The number of neurons in this layer is equivalent to the number of generators. The final output of this model is the scores tensor. Each element in the score tensor represents the model’s confidence that the corresponding input was produced by a specific generator. These scores are logits.

Additionally, utilising sparse categorical cross-entropy as the loss function for private discriminators is intended for multi-class classification, where each class corresponds to one of the generators.

3.2.3.1 Reinforcement Technique in privGAN.

Since the data is discrete and the generator needs to receive feedback to update its weight, the non-differentiable operations in generators do not allow for the gradients to flow back. Hence, the network would remain disconnected. To remedy this, while adhering to the original design of privGAN to maintain the privacy measures, a reinforcement technique was applied (Goodfellow et al., 2016, p. 689) such that the generator would output a tuple of action probabilities in addition to generated texts. The generated texts then follow the same route as defined for the generated data in the original privGAN, while action probabilities would constitute two outputs. These outputs, along with their respective rewards, would be passed to a custom loss function that calculates gradient policy loss (see equation 8). The rewards, as mentioned, are the entropy of the outputs from discriminators and private discriminators (see equation 7). To implement the gradient policies, these rewards are passed as targets during GANs training.

Considering that, in essence, the desired outcome in GANs is to achieve high uncertainty for discriminators, rewarding the generator for high entropy of the discriminator’s output, which is passed as a target, is expected to lead to the generation of realistic fake data.
3.2.4 Demonstrating the Artefact

The activity of demonstrating the artefact involves showing its feasibility in a real-life or illustrative case. The purpose of the demonstration is to provide evidence that the artefact can solve a specific instance of the problem it was designed to address (Johannesson & Perjons, 2014).

To meet the purpose of the demonstration phase, considering that it is not possible to test the feasibility of the artefact in a real clinical setting, this study adapts an illustrative case by carrying out an experiment on clinical data. Hence, in this section, part of the experiment that focuses on illustrating the feasibility of the artefact, including the required resources to carry out the experiment on the artefact, its architecture and the flow of data throughout the experiment, is described.

3.2.4.1 Resources

To carry out the experiment, the following resources have been utilised:

Data
Access to clinical data (MIMIC-III) was obtained after completing the “CITI Data or Specimens Only Research” course offered by MIT. The MIMIC-III database is a vast collection of deidentified health-related data from more than 40,000 patients who received critical care at Beth Israel Deaconess Medical Center between 2001 and 2012. This database provides access to a range of information, including demographics, hourly vital sign measurements, laboratory test results, procedures, medications, caregiver notes, imaging reports, and mortality data (including post-hospital discharge) (Johnson et al., 2016).

Hardware
To address the required computational resources for this project in addition to Colab pro, pro+, and Pay-As-You-GO subscriptions, which provided various computation powers including 25GB RAM and 500 computing units and access to GPUs of NVIDIA A100 Tensor Core and NVIDIA, V100 TENSOR CORE, storage of up to 100 GB were purchased for uploading the database on Google drive and later mounting it on Colab environment.

Software
The experiment required several software dependencies, including Python, TensorFlow, Keras, NumPy and Pandas. Additionally, the cased version of BERT from Huggingface was used for tokenisation and intended for later evaluation of data utility with BERT.

3.2.4.2 Experiment

Implementation of the artefact using the described resources was originally planned as part of an experiment constituting three main steps: data preprocessing, synthetisation, and evaluation. The preprocessing aimed at learning about the data, features and relationships among columns to design the evaluation as well as prepare it for synthetization. The evaluation was originally planned to be performed by comparing BERT’s performance on real and fake data. Hence, during preprocessing, BERT was used on real data to both confirm the observed relation between the categories and ‘text columns and later use this in evaluation for comparing with BERT’s performance on the generated synthetic data, as illustrated in Figure 1.
Figure 1: Illustrates PrivGAN architecture, adapted from and courtesy of Mukherjee et al. (2021) and the intended implementation of it within the context of this study. Dp refers to the in-built adversary. A minimum of two generator-discriminator pairs is depicted, but the number can increase based on the ratio of \( \frac{|X_{\text{Train}}|}{\text{batchSize}} \). The pre-processed text data is randomly partitioned and used to train PrivGAN.

However, considering the unexpected limitations during synthetization that are discussed later, the evaluation was delimited to using BLEU for the assessment of the quality of generated data from only one of the classes of the pre-processed data. In case the result proves to be promising, in future experiments, this can be extended to all classes where BERT’s performance on the synthesised and real data could be assessed. Thus, the preprocessing and the code were carried out for all classes within Note events categories, while only the result of synthetisation for one class was evaluated.

**Data preprocessing**

The pre-processing of the data initially included applying descriptive statistic methods to get a picture of the shape of the data, the number of columns and rows, what each column represents and the relation between them. This provided an understanding of how to proceed with cleaning and designing the experiment.

Data cleaning was conducted on several levels following two main approaches of handling erroneous and empty entries. To handle the errors, in addition to the entries that were explicitly marked erroneous under the “ISERROR” column, several error tags were detected under the text column. The rows for these entries were removed. Similarly, for handling empty entries, in addition to deleting NAN entries, several empty text entries under the text column were detected and removed.

Investigating the clinical notes stored under the TEXT column clarified that the majority of the text entries or notes include a tag that corresponds to the listed category under the ‘Category’ column. Several of these notes across categories only included the tag name with no further text, similar to a placeholder, signifying what should be expected. These rows were also considered ‘empty’ and treated as such.

**Class imbalance**

In the next step, grouping the data by category showed a significant class imbalance among the 16 categories as well as similar labels for two categories. To address this issue, the two categories of
‘Nursing’ and ‘Nursing/others’ were merged, and eight other categories, including consult, pharmacy, and social work with very small counts, were merged into ‘Other Health Care Services’.

Figure 2: Count of clinical notes per category in Notevents dataset of MIMIC-III after cleaning the data showing class imbalance. With a total length of 2081900.

One-to-many relation between category and ‘subject ID.’

In the dataset, there exists a one-to-many relation between category and ‘subject ID’. That is, for every category, multiple notes can correspond to the same subject ID. Thus, in the next steps, the texts corresponding to all identical subject IDs per category were combined. This resulted in relatively more balanced data with one-to-one relation for notes pertaining to subject ID within each category.

Figure 3: Count of clinical data per category after merging groups with smaller counts and combining the texts pertaining to each subject ID within each Category. The processed data is still imbalanced.
BERTs maximum token and RAM limits

Although the above pre-processing reduced data length from about 2 million to 186,047, further processing of data of this size continued to present challenges. Maintaining text within the tabular structure and the uneven number of tokens that exceed BERT limits of 512 tokens would cause the processing time to take longer, exceed available memory limits and lead to eventual out-of-memory errors. To address this issue, a smaller volume of data was obtained by random sampling of 2000 instances per class, leading to a total length of 14,000. Overall, these implementations led to a balanced dataset for each class with a truncated maximum length of 126 tokens, which made further processing feasible.

In the next step, the saved tokenised sample is prepared for training the privGAN network. A total number of 10,179 unique tokens were obtained as the number of embeddings for the entire dataset. The privGAN components were instantiated and run in isolation with dummy data to ensure functionality and address possible errors.

Training privGAN

Subsequently, in a loop, the tokenised data equivalent to each class was passed to the privGAN function in batches of 10 and a maximum sequence length of 126, along with instances of the generators and discriminators and the privacy parameter for generating synthetic data (0.5). Other than applying a reinforcement technique with a custom loss to reward the generator for accurate generations, the text implementation of privGAN is slightly different from the image version in that instead of sending random noise to the generators to trigger data generation, a batch of pre-defined start tokens were used (see Table 2). Subsequently, GAN input was modified to be consistent with this approach.

Finally, the generated data was saved for further analysis and evaluation.

To ensure the functionality of modified privGAN with the text data and designed discriminators and generators, at this stage of the study, the program was tested and troubleshooted on small epochs. In each run, the code and parameters were reconsidered to deal with out-of-memory errors, the limitation of Colab+ pro GPUs in terms of availability to run the model nonstop, and fluctuating loss as elaborated below:

Out-of-memory errors

To deal with this issue, the batch sizes of 32 and 16 were tried and finally lowered to 10. In addition to resolving out-of-memory issues, lowering batch size could have a regularising effect due to adding noise to the learning process (Goodfellow et al., 2016, p. 295). The sequence length of 256 was lowered to 126 while incorporating only two layers of LSTMs was possible for each generator and discriminator model. Lowering the values of parameters such as layers of LSTMs would, of course, lead to adverse effects in generating high-quality data as it lowers the complexity of the model. However, that can be compensated by increasing the hidden and embedding dimensions to 150 in comparison to 32 in seqGAN (Jozefowicz et al., 2016b; Yu et al., 2017).

Fluctuating losses

The observed fluctuating losses across batches were considered to be normal to some extent when training GANs, as the discriminators and generators are in a tug-of-war with each other. However, during the initial 25 epochs of training, very high fluctuations, as well as occasional increasing trends, in combined loss were observed at learning rates of 0.01, 0.002, and 0.0002. Based on these
observations and to promote stabilised training, progressively lower learning rates were used. Ultimately, a learning rate of 0.00001 was selected as it showed the most stable trends during the training phase. To ensure more stabilised training, in addition to small batch size and lowering learning rates (Goodfellow et al., 2016, p. 279), gradient clipping was implemented. This could effectively prevent very large gradient updates, which could lead to a destabilised network’s learning.

Start tokens

Considering that the LSTM layers in the generator use the start tokens as an initial context from which they would generate the subsequent tokens, observing a common initial token per category in the tokenised text column led to choosing a specific start token for each category. This can ensure that the generation of subsequent tokens for that category is slightly tailored for that category. The observed start token for the ‘Discharge summary’ category was 24930, which corresponds to ‘Ad’. This sub-word is the prefix for ‘admission’.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>PrivGAN (Mukherjee et al., 2021)</th>
<th>SeqGAN (Yu et al., 2017)</th>
<th>privGAN-text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>128</td>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>Sequence length</td>
<td>-</td>
<td>20</td>
<td>126</td>
</tr>
<tr>
<td>Hidden dimension</td>
<td>-</td>
<td>32</td>
<td>150</td>
</tr>
<tr>
<td>Embedding dimension</td>
<td>-</td>
<td>32</td>
<td>150</td>
</tr>
<tr>
<td>Privacy ratio</td>
<td>0.1, 1, 10</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>$\beta$1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>Adam</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0002</td>
<td>0.01</td>
<td>0.00001</td>
</tr>
<tr>
<td>Clipnorm</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Lsmooth</td>
<td>0.95</td>
<td>-</td>
<td>0.9</td>
</tr>
<tr>
<td>Start-token/noise</td>
<td>Random</td>
<td>0</td>
<td>24930</td>
</tr>
<tr>
<td>Dp-delay</td>
<td>100</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Epochs</td>
<td>200</td>
<td>150-250</td>
<td>175</td>
</tr>
<tr>
<td>Disc epochs</td>
<td>50</td>
<td>-</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 2: List of hyperparameters used in this study in comparison to privGAN and seqGAN. Lsmooth is a hyperparameter that is used for label smoothing, which is a regularisation technique that is used to prevent the model from becoming too confident in its predictions, thus decreasing the chances of overfitting. Lowering Lsmooth would lower the model’s confidence in its predictions but improve the stability (Szegedy et al., 2015). Dp-delay and Disc epochs are specific to the privGAN design and privacy discriminator. In this study, the same values were used for the delay in co-training of the privacy discriminator (Dp-delay) for the same number of epochs (Disc epochs).

3.2.5 Evaluation of the Artefact

In this step, the focus is on assessing the ability of the artefact to address the problem that was identified and to find out the degree to which it satisfies the specified requirements (Johannesson & Perjons, 2014).

An experimental research strategy is deemed suitable in this study, as it allows for controlling the factors such as model hyperparameters and collecting and observing data to implement and test whether the artefact does meet the predefined specified requirements.

The performance of the artefact in solving the problem was evaluated both from privacy-preserving perspectives in the face of a white-box attack and from a utility perspective using BLEU-4.
Classification performance is a standard quantitative evaluation metric for GANs (Borji, 2019). The original study of privGAN applied a CNN to evaluate the utility of the generated images (Mukherjee et al., 2021). Similarly, this study planned to use BERT for a multi-class classification task on the full synthetic dataset of length 14,000 to evaluate the utility of the generated data. However, as each full training of 150 epochs for the generation of synthetic data took 24 hours for only one of the pre-processed seven classes, the evaluation was revised to first evaluate the model’s performance for the first class. The quality of synthetic data is then tested using BLEU-4. If promising preliminary results were obtained, full synthetisation of the entire pre-processed dataset, encompassing 14,000 units of data, would be pursued. This process was estimated to take about seven days.

3.2.5.1 White-box Attack on privGAN

To perform a white-box attack, in addition to the subset of data used for training, a subset of Noteevents equivalent to the full length of the Discharge summary category (41127) was tokenised using the cased version of BERT base with the maximum length of 126 tokens.

The attack was carried out twice. Once by a single discriminator and once by an array of discriminators, as performed in a privGAN study (Mukherjee et al., 2021).

Utilising a single discriminator, the training data and the complete dataset were merged. The discriminator then predicted labels for the entire merged data. Data points were then ranked based on the predictions from the discriminator, and the top predictions corresponding to the number of samples in the training set were selected to calculate their accuracy.

In performing a white-box attack, in line with the privGAN study, where both discriminators are used, and max attack accuracy is calculated, data points are ranked based on the predictions from the discriminator(s). For each sample in the dataset, predictions from all discriminators are collected. The maximum prediction value, which represents the highest confidence across the array of discriminators, is selected for each sample. From these maximised predictions, the top indices corresponding to the number of samples in the training set are selected.

The resulting scores from these attacks would indicate an adversary's potential success in extracting sensitive information under such conditions.

3.2.5.2 BLEU-4

BLEU-4 is a common measure for comparing machine-generated text with a reference, which in this case is real data. The BLEU-4 score includes unigrams, bigrams, trigrams, and 4-grams. A higher n allows the BLEU score to better capture the fluency of the text, as it can recognise longer matching sequences of words (Papineni et al., 2002).

The generated data and the encoded tokenised text column from pre-processed data, corresponding to the Discharge summary category of the Event notes table, were decoded by BERT’s cased tokeniser prior to evaluation with BLEU 4.

3.3 Ethics

The concept of ethics applies to various stages of this project across the DSR framework. This covers a range of ethical issues, from considering the results of previous studies to avoid unethical work (Dresch et al., 2015, p. 131) to ethics of handling sensitive authorised data according to the regulations
and requirements. In these lights, the access request to MIMIC-III was approved by PhysioNet after agreeing to terms of use and completing further training to comply with regulations.

Further ethical considerations could include ethical consequences of the use and implementation of the artefact to avoid unintended compromise of the data or possible harm to systems, users and stakeholders. That is, as the result of this research or the application of the designed artefact, no humans, animals, or the environment should suffer.

Regarding ethical issues pertaining to the environment, the only identified ethical issue was the amount of computational power and energy required to complete this experiment and how it would translate to real-world scenarios in terms of energy consumption if applied on a large scale. This can be further studied if the experiment shows promising results before applications in the real world.

A possible harmful consequence for humans as the result of this study is the leak of private sensitive data. To avoid this, it is important to use already de-identified data or consider de-identifying the data prior to the implementation of the experiment.

As the artefact can only be used on real data and in real-world scenarios following thorough examinations, any identified risks, harm, or privacy concerns would be grounds for prohibiting the artefact’s use in the real world, in line with ethical compliance standards.

In addition to avoiding the considered ethical consequences, compliance with required ethical standards ensures that the rights, dignity, and welfare of the research participants are protected while building trust between researchers and the broader community. Trust leads to cooperation and investments in advancing science, which leads to the general well-being of society. For example, in the case of research in the clinical domain, patients are more likely to participate in research studies if they trust that their data will be used in an ethical and responsible manner.
4 Results

The BLEU score shows an extremely low value of 6.17e-232. This score can be discussed in light of the fluctuations and stagnations observed in calculated GAN loss across epochs during the training of the network, as illustrated in the graphs below. A general expectation when training neural networks is to observe an overall decrease in the value of the loss function across the epochs, signalling effective learning (Goodfellow et al., 2016, p. 177). Figure 1 displays the loss values for our modified privGAN during its training. An initial decrease is evident in the early epochs, suggesting the model’s prompt adjustment to the training data. However, the subsequent stagnation raises questions about the model’s capacity for further improvement or if it has reached an optimal point early. One possible reflection is that due to the GANs present unique training dynamics compared to other neural networks, the simultaneous training of two components (generator and discriminator) in an adversarial setting can lead to fluctuations or even stagnation. The observed trends in Figure 1 highlight such nuances. It should be noted that in this study, the observed high fluctuations with higher learning rates were not considered normal or indicative of effective learning. This led to lowering the learning rate incrementally to control the fluctuations. As a result, the observed fluctuations in loss function values became relatively lower; the accompanying BLEU score and the generated synthetic text show that such observations are not indicative of effective training.

The white-box attack accuracy performed by one of the discriminators (0.2055) would have been a fair score if the model did, in fact, learn and produced a higher BLUE score, showing ~80% in preserving the privacy of text data under the white-box attack. Additionally, performing a white box attack with both discriminators, as performed in the privGAN study, resulted in a maximum privacy score of 0.361, indicating increased privacy risk. In light of the observed training dynamics, such privacy scores raise questions about their interpretability, as such scores can be indicative of privacy preservation or, in fact, ineffective training.

Dp-Accuracy is a metric that measures the ability of the privacy discriminator to correctly identify the originating generator for a given sample. The expectation is that as the privGAN model trains, the dp-
Accuracy on the synthetic data should decrease because it will become harder for the privacy discriminator to distinguish which generator produced a particular synthetic sample. This is the desired outcome because the generators are learning to generate data that is harder for the privacy discriminator to classify, thus preserving privacy. Figure 3 shows that average accuracy throughout the training fluctuated around 0.5, leading to an overall average of 0.53 across epochs. However, it is noted that the average of dp_accuracy after epoch 100 increases to 0.62. An increase in the accuracy of Dp after it starts training with synthetic data within the combined model suggests that the generators’ outputs are more distinguishable from one another. Considering that it is important for Dp not to be able to distinguish the origin of data generation, lower accuracy would have been more consistent with the target. Because if an adversary can determine which generator created a specific sample, they might also infer whether a specific data point was part of the original training dataset, compromising the privacy of individuals in the dataset.

![Figure 3: The accuracy of the private discriminator across 175 epochs is shown to be, on average, slightly above random guessing. An increase in the Dp-Accuracy after epoch 100 indicates that Dp finds it easier to discern the origin of the data.](image)

An excerpt of the generated text from the discharge summary can be viewed in Appendix B.

5 Discussion

In discussing the observed results, a number of points emerge, including the log trend and how that represents the learning of the combined model, the obtained BLEU and white-box attack scores, how they meet the set requirements for the artefact, and the accuracy of the private discriminator. Discussing the observed trend of losses, the characteristics of GANs and the intended behaviour of each component should be considered. That’s, in fact, what makes GANs training more complex than other neural networks (Goodfellow et al., 2016, p. 295). As the combined loss is influenced by push and pull actions between the generators and discriminators, the loss trends may not follow a steady overall decrease. Although this study aimed at taming heavy fluctuations of the loss in order to achieve better learning, compared to other GANs studies such as privGAN, it does seem that, in fact, fluctuations and stagnation of losses are normal observations of GANs losses and have also been the case for the reported G-loss and d-t loss. In fact, according to the privGAN study, in various experiments, setting the number of epochs of the private discriminator to 0 and 50 with a respective co-training delay of 0 and 100 does show both fluctuating and stagnating trend for the combined-loss
(G-loss) (see figure 3). In fact, the recorded combined loss of original privGAN seems to increase until epoch 100 and then mostly fluctuates around 7 (Mukherjee et al., 2021).

Comparing the observed trend of d-t loss in both studies also show that, similar to the privGAN study, as expected, both the discriminators show close values of loss around 0.6, which overall shows a slight decrease across total epochs. Considering this, the high fluctuations observed, especially initially during training with higher learning rates (0.01, 0.002, 0.0002), were not indicative of a malfunction of the network and the ground for decreasing the learning rates, as done in this study. Lowering the learning rate excessively can lead to slower training, longer convergence time, and possibly optimisation failure (Goodfellow et al., 2016, p. 431). One potential explanation for the notably low BLEU score observed in this study could be optimisation failure, suggesting that the model might not have adequately captured the characteristics of the real data. This could subsequently impact the discriminator's performance, manifesting in its inability to distinguish between real and low-quality synthetic data and resulting in a notably low accuracy score. In an equilibrium scenario for a GAN, where the discriminator is effectively deceived, it predicts the authenticity of data with a probability close to 0.5. This would effectively lead to an accuracy of about 0.5, as the model is equally likely to guess the authenticity of the data as true or false.

**Figure 6:** Comparing the trend of recorded losses of combined GAN between the two studies shows that fluctuation and stagnation were, in fact, expected behaviour. The privGAN records are approximations drawn from the graph in the original study. The learning rate in privGAN-text was set to 0.00001 and $\lambda = 0.5$, while in privGAN, the learning rate was set to 0.0002 with $\lambda = 1$ on the MNIST dataset.

Overall, these observations would indicate that the model may not have reached convergence and is undertrained. To remedy this, an optimum learning rate should be obtained. However, the choice of the learning rate is stated to be an art (Goodfellow et al., 2016) that is achieved via trial and error through further experimentation.

Other than the low learning rate that could be an explanation behind the low BLEU score, the design of the reward can be a more rooted contributing factor. This could play a significant role in the behaviour and outcome of the generator. In this study, the rewards were defined based on the measure of uncertainty, which was calculated from the discriminators’ predictions. This was based on the ultimate purpose of GANs training, which is to reach the state of equilibrium that occurs when the discriminator is uncertain about the realness of generated data. In contrast, other losses included in combined loss would seek to have the discriminator reach certainty that the model is right about
classifying fake generated data as real. This setting could pose an added contradiction in learning strategy and lead to poor results. Thus, considering this, perhaps future designs could consider a more direct approach.

While rewarding based on the discriminators’ prediction instead of the degree of its uncertainty could address possible contradictions, the reward could still be more granular. That is, in this study, the generator is rewarded for the generation of a whole batch of sequences while, for example, in seqGAN, using Monto Carlo Tree Search (MCTS), they use discriminator’s prediction to estimate future rewards. This estimation is used for the generation of the next tokens.

This approach seems to suggest a viable solution for improving the utility data. However, considering the high computing cost of MCTS and the related family of algorithms, which is an issue for AI programs in commercial software (Ou et al., 2022), the low computing requirements of the artefact, and the limitations of this study, the generator in this study did not benefit from such mechanism that would direct generation of next tokens based on estimation of future rewards. This difference in design and outcome highlights the need and significance of a design that constructs an effective, rewarding system similar to MTCS, which would allow the generator to make more informed decisions during sequence generation while being computationally affordable for applications of privacy-preserving GANs for discrete data.

Evaluating the artefact in response to the set requirements shows that the utility of the data cannot be met under current settings. BLEU score of 6.17e-232 is an indicator of the linguistic quality of the generated synthetic data. A lower BLEU score means that the synthetic data does not match closely with the real data in terms of its linguistic characteristics. This is a poor result in terms of data utility. It means that the synthetic data generated by the GAN is not useful for downstream tasks because it does not reflect the linguistic structures and patterns present in the original data.

In terms of privacy, the white-box attack accuracy of 0.2055 (approximately 20%) with one discriminator indicates that the discriminator of the GAN is not particularly good at identifying which samples were part of the original training set. This means that the synthetic data generated by the GAN is not easily traceable back to the original real data. In terms of privacy preservation, this may be considered a fair score. It suggests that the PrivGAN framework is somewhat obscuring the relationship between the synthetic data and the original data, making it more difficult for a certain attacker to infer sensitive information. However, it is observed that increasing the number of discriminators involved in the attack yields an even less desirable privacy score. Table 3 compares the result of the maximum white-box accuracy score with the original privGAN.

<table>
<thead>
<tr>
<th>privGAN-text MIMIC-III</th>
<th>privGAN (f-MNIST)</th>
<th>privGAN (CIFAR-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda = 0.5 )</td>
<td>( \lambda = 0.1 )</td>
<td>( \lambda = 10 )</td>
</tr>
<tr>
<td>0.361</td>
<td>0.192</td>
<td>0.192</td>
</tr>
<tr>
<td></td>
<td>0.095</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>0.424</td>
<td>0.154</td>
</tr>
</tbody>
</table>

*Table 3. Comparing the results of max scores with the closest reported scores of white-box attack accuracy from the original privGAN shows that with a privacy ratio of 0.5, the accuracy is low, which suggests that the privacy measures may have been somewhat effective. The privacy score initially calculated for one discriminator showed an accuracy of 0.2055.*

Furthermore, an average accuracy of 0.53 over 175 epochs for the private discriminator in privGAN in this study is slightly better than random guessing, which indicates that the private discriminator cannot easily discern the origin of synthetic data from generators. However, considering the average accuracy
after co-training of the privacy discriminator starts, this value increases (0.62). It is notable that the privacy discriminator's increased accuracy in distinguishing the outputs of the generators suggests it is failing the training objective. That is, it is failing to distinguish between the origin of synthetic data. The generators might be producing outputs that inherently carry distinguishable traits from their respective data splits. This misalignment with the privacy objective poses a potential privacy concern. Similar to the outcome of the designed dynamic between regular discriminators and generators, this result can be rooted in the low learning rate, premature introduction of discriminators' uncertainty as a reward and a more granular feedback system for the generators.

In terms of fulfilling the requirement for preserving the integrity of original data, it can be noted that the generation of synthetic data in this study is based on substantially truncated sequences to allow for further processing within the available resources. While the original data would remain untouched, the representativeness of truncated and balanced data can be questioned.

One of the important criteria in evaluating synthetic data using privacy-preserving GANs is obtaining an optimum trade-off between a high privacy score and minimum loss in data utility. This has been considered an objective in many studies of GANs, while regarded as impossible in others (Stadler et al., 2022). In this study, the requirements of the artefact were set based on the objective of obtaining both good utility and privacy scores. Thus, when evaluating the artefact, the obtained privacy result should be considered in light of previous discussions about the low BLEU score and possible optimisation failure. The obtained results indicate that an optimum trade-off between utility and privacy was not obtained in this study. This leaves a question unresolved. That is, whether a well-converged privGAN-based model with high data utility would have produced the same level of privacy score.

5.1.1 Conclusion

In conclusion, adapting a privacy-preserving GAN as a solution to preserve the privacy of clinical text data introduces several complexities. These complexities include incorporating a strategy that addresses the discreteness of the data to be able to use feedback generators to update their weights. This can be done by using reinforcement learning. However, incorporating an adequate reward for the generator while adhering to the privGAN framework can be challenging and requires further research. Additionally, the complexities of training GANs due to the dynamic between the discriminators and generators, which requires several experiments with multiple hyperparameters to reach the best results, can be computationally expensive.

Additionally, training the model for 175 epochs over one class, with a length of 2,000 and 126 tokens, proved to require more than 24 hours. It is estimated to take a week for a full dataset of length 14,000, which would necessitate further computational power. It is notable that the full training data, with a length of 14,000, constitute only about 7% of the total length of the pre-processed 'Note events' table from MIMIC-III. This highlights the stated gap in the reviewed literature about the need for computationally efficient solutions that could scale to large data.

Computing the uncertainty of discriminators as rewards may seem in line with the ultimate goal of GANs, which is to reach a status in their training where the discriminator is uncertain about the authenticity of the data. However, introducing such entropy-based rewards to incentivise this uncertainty prematurely could be problematic. By pushing the model towards this state too early, we might distort the natural progression and improvement of the generator's outputs. It can also be argued
that the uncertainty of the discriminator can have multiple reasons, one of which could be the high
good quality of synthetic data. This does not exclude other causes, such as untrained discriminators.

In light of the ethical implications, the result of this study can be discussed from several perspectives.
From a data privacy perspective, the relatively low white-box attack score using one discriminator can be
promising. However, employing combined discriminators for performing the attack yields a less
favourable outcome. It is essential to underscore that even a seemingly low accuracy of 20% does not provide complete privacy. That is, the suggested privacy score still indicates some risk of breach.
Further research is required for obtaining better privacy results and ensure that the ethical values of
preventing harm to people and society are met.

Additionally, the low quality of synthetic data generated in this study raises concerns about its
reliability and, as such, cannot be useful for medical research, predictive modelling or other healthcare
applications that may rely on synthetic data. Inferior or meaningless synthetic clinical text can lead to
misinformed medical decisions, potentially jeopardising patient health and safety. Mistrust in health
information systems may grow among healthcare professionals, undermining the adoption of digital
tools and innovations. It is further important to obtain synthetic data that can capture the diversity in
the original data set to avoid the risk of having biased or unfair outcomes in healthcare. Thus, to
achieve the standard required for using privGAN for text in clinical data, further research is required to
ensure the ethical and societal requirements are met. However, meeting such standards via robust
solutions in further research could potentially have positive implications, such as easier and safer
access to synthetic clinical text data with less risk of breach of sensitive information.

The observed results underscore the multifaceted challenges inherent to GAN training and evaluation.
Additionally, the complex interplay between ensuring data privacy and maintaining utility in
generating synthetic data is highlighted. Without a good utility, synthetic data would not be usable and
hence cannot serve as a solution for preserving the privacy of clinical data.

5.1.2 Limitations

This study faced several limitations of various types, including limitations due to the sensitivity of
clinical data, which led to delayed access and limited available computational power, while facing the
experimental nature of training a privacy-preserving GAN with multiple neural networks, which
requires sufficient computation. The sensitivity of clinical data poses an added layer of bureaucracy,
which can delay access and increase uncertainty about how to formulate design science research
experiments without having a clear picture of the data in such a time- a restricted study. Moreover,
limited available and affordable computational power hindered the preprocessing and tokenisation of
MIMIC-III.

Out-of-memory errors persisted throughout the experiment and affected the choice of the number of
LSTM layers, number of batches and length of sequences. Additionally, the process of storing the
generated data had to be revised to address out-of-memory issues as well as time-restricted availability
of computational power.

The runtime of Google’s Colab pro+ disconnects irrespective of the ongoing processes, resulting in loss
of data, trained model, as computational points used for running the model during the preceding 24
hours. To circumvent these issues, checkpoints were implemented every 25 epochs to allow resuming
the training after any potential disruption. Additionally, limited computation resources in this study do
not allow further experimentation to determine the optimal hyperparameters.
5.1.3 Future Studies

Given the findings of this study, future research can explore designs that incorporate more direct reward mechanisms. For instance, equipping the generator with estimates of future rewards for predicting subsequent tokens has been validated by the seqGAN approach. This could be a promising strategy to enhance the utility of generated data as long as the computational requirements remain manageable. Such a strategy could effectively tackle the issue of low-quality synthetic data by offering the generator more nuanced feedback.

Furthermore, considering the discussion on the recorded log of losses during training, it is recommended to increase the learning rate while accommodating normal fluctuations in the observed losses across batches and epochs. A series of ablation studies can be conducted to ensure if the discriminator, in a text (sequence) GAN, can, in fact, distinguish the real and fake data and if the white-box attack accuracy on this discriminator can be compared with white-box attack accuracy of this study with privacy-preserving measures in place. This would perhaps provide a more informative comparison than the attacks conducted on the privGAN or regular GANs with other data types.

The original study of privGAN has evaluated the framework in the face of multiple state-of-the-art attacks. Future studies addressing the abovementioned issues could also consider expanding the evaluations to other types of attacks and testing the utility of data using a classifier.

As discussed, more experimentations are required to reach an optimum value for hyperparameters, including learning rates, which affects the utility.

A privacy-preserving GAN suitable for text data has not been discussed in the reviewed literature. Thus, further studies in this realm could shed further light on the capabilities and limitations of such artefacts.
References


28


Appendix A – Project’s code

```python

class Generator(Model):
    def __init__(self, num_emb, emb_dim, hidden_dim, sequence_length, learning_rate, grad_clip, num_lstms, dropout_rate, batch_size):
        super().__init__()
        self.num_emb = num_emb
        self.sequence_length = sequence_length
        self.learning_rate = learning_rate
        self.grad_clip = grad_clip
        self.batch_size = batch_size

        self.embedding = layers.Embedding(input_dim=self.num_emb, output_dim=emb_dim)
        self.lstms = [layers.LSTM(hidden_dim, return_sequences=True, dropout=dropout_rate) for _ in range(num_lstms)]
        self.dense = layers.Dense(self.num_emb)

        self.optimizer = Adam(learning_rate, clipnorm=self.grad_clip)
        self.loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=True)
        self.compile(optimizer=self.optimizer, loss=self.loss_fn)

def call(self, start_tokens, temperature=1.0, training=False):

    generated_sequences = start_tokens  # Initialize with start tokens
    x = self.embedding(start_tokens[:, 0:1])

    # Initialize action_probabilities with placeholder for start token
    # to store action probabilities at each timestep
    action_probabilities = [tf.zeros((self.batch_size, 1), dtype=tf.float32)]  # use the appropriate tensor shape and dtype

    for i in range(1, self.sequence_length):
        for lstm in self.lstms:
            x = lstm(x, training=training)

            logits = self.dense(x)
            current_logits = logits[:, :-1, 1]
            logits_exp = tf.math.exp(current_logits / temperature)
            sum_exp = tf.reduce_sum(logits_exp, axis=1, keepdims=True)
            probs = logits_exp / sum_exp

            next_tokens = tf.random.categorical(probs, 1)

        # Gather the probabilities of the actions taken
        action_prob = tf.gather(probs, next_tokens, batch_dims=1)

        action_probabilities.append(action_prob)  # append current probabilities to the list

        # Concatenate new tokens to the generated sequence
        generated_sequences = tf.concat([generated_sequences, next_tokens], axis=1)

        # Embed the next tokens for the next LSTM computation
        x = self.embedding(next_tokens)

    # Convert the list of action probabilities to tensor for consistency
    action_probabilities = tf.concat(action_probabilities, axis=1)

    return generated_sequences, action_probabilities
```
**Discriminator**

```python
class Discriminator(Model):
    def __init__(self, sequence_length, vocab_size, embedding_size, num_lstm, hidden_dim, dropout_rate, learning_rate):
        super(Discriminator, self).__init__()

        self.embedding_layer = Embedding(vocab_size, embedding_size, input_length=sequence_length)
        self.lstm = [LSTM(hidden_dim, return_sequences=True, dropout=dropout_rate) for _ in range(num_lstm)]
        self.pooling = GlobalAveragePooling1D()
        self.dense = Dense(1)

        self.learning_rate = learning_rate
        self.compile(optimizer=Adam(learning_rate, beta_1=0.5),
                      loss=tf.keras.losses.BinaryCrossentropy(from_logits=True), metrics=['accuracy'])

    def call(self, inputs, training = True):
        x = self.embedding_layer(inputs)
        for lstm in self.lstm:
            x = lstm(x)
        x = self.pooling(x)
        logits = self.dense(x)
        # Return logits directly, not probabilities
        return logits
```

**Privacy discriminator**

```python
class TestPrivacyDiscriminator(Model):
    def __init__(self, sequence_length, vocab_size, embedding_size, filter_sizes, num_filters, dropout_rate, num_generators, optim):
        super(TestPrivacyDiscriminator, self).__init__()

        self.embedding = Embedding(vocab_size, embedding_size, embeddings_initializer='random_uniform')

        self.conv = []
        for filter_size, num_filter in zip(filter_sizes, num_filters):
            self.conv.append(Conv1D(num_filter, filter_size, strides=1,
                                      padding='VALID', activation='relu', kernel_initializer=TruncatedNormal(stddev=0.1),
                                      bias_initializer=Constant(value=0.1)))

        self.flatten = Flatten()
        self.dropout = Dropout(dropout_rate)
        self.dense = Dense(num_generators, kernel_initializer=TruncatedNormal(stddev=0.1), bias_initializer=Constant(value=0.1))

        self.optimizer = optim
        self.loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
        self.compile(optimizer=self.optimizer, loss=self.loss_fn)

    def call(self, inputs, training=False):
        # Reshape inputs if necessary
        if len(inputs.shape) == 3 and inputs.shape[1] == 1:
            inputs = tf.squeeze(inputs, axis=1)
        x = self.embedding(inputs)

        pooled_outputs = []
        for conv in self.conv:
            conv_out = conv(x)
            pooled = tf.reduce_max(conv_out, axis=1, keepdims=False)
            pooled_outputs.append(pooled)

        x = tf.concat(pooled_outputs, axis=-1)

        if training:
            x = self.dropout(x, training=training)
        scores = self.dense(x)
        return scores
```
initial_tokens = {
    'Discharge summary': 24930,
    'ECG': 16009,
    'Echo': 8544,
    'Nursing Group': 151,
    'Other Health Care Services': 18959,
    'Physician': 2534,
    'Radiology': 22580,
    # add other classes here as necessary
}

def start_token_per_each_class(classes, batch_size):
    start_token_dict = {}
    for class_label in classes:
        start_token = initial_tokens[class_label]  # get initial token for this class
        start_token_batch = np.full((batch_size, 1), start_token)  # create a batch of this start token
        start_token_dict[class_label] = start_token_batch  # store this batch in the dictionary
    return start_token_dict

# custom loss

class ActionRewardsLoss(tf.keras.losses.Loss):
    def call(self, y_true, y_pred):
        # y_true are the sigmoid of the discriminator's output as reward
        # y_pred are the action probabilities from generator
        loss = -tf.reduce_mean(tf.math.log(y_pred + 1e-10) * y_true, axis=1)  # [batch_size]
        return loss

 Instantiate the models

# Generator parameters:
num_emb = 51279
emb_dim = 150
hidden_dim = 150
sequence_length = 136
learning_rate = 0.00001
grad_clip = 1.0
batch_size = 10
num_latent = 2
dropout_rate = 0.1

# Instantiate generators, discriminators
generator1 = Generator(num_emb, emb_dim, hidden_dim, sequence_length, learning_rate, grad_clip, num_latent, dropout_rate, batch_size)
generator2 = Generator(num_emb, emb_dim, hidden_dim, sequence_length, learning_rate, grad_clip, num_latent, dropout_rate, batch_size)

discriminator = Discriminator(sequence_length, vocab_size, embedding_size, num_latent, hidden_dim, dropout_rate, learning_rate)

# Instantiate private discriminator
private = TextPrivateDiscriminator(sequence_length, vocab_size, embedding_size, filter_sizes, num_filters, dropout_rate, num_generators, opti)
```python
def privGAN(x_train, start_token):
    generators = [generator1, generator2],
    discriminators = [discriminator1, discriminator2],
    dDisc = privDisc,
    disc_epochs = 50, epochs=176, dp_delay = 100,
    batch_size=10, optim = Adam(learning_rate=0.00001, beta_1=0.5, beta_2=0.5), verbose = 1,
    lsmooth = 0.5, privacy_ratio = 0.5, splitter = False):

    # Define Checkpoints
    checkpoint = Checkpoint(optim_main=optim,
                             optim_discriminators=keras.optimizers.Adam(learning_rate=0.00001, beta_1=0.5),
                             optim_generators=keras.optimizers.Adam(learning_rate=0.00001, beta_1=0.5),
                             generator=generator,
                             discriminators=discriminators,
                             dDisc=dDisc)

    manager = CheckpointManager(checkpoint, directory="/content/drive/MyDrive/ckpt", max_to_keep=3)
    checkpoint.restore(manager.latest_checkpoint)
    if manager.latest_checkpoint:
        print("Restored from [manager.latest_checkpoint]")
    else:
        print("Initializing from scratch.")

    # make sure the number of generators is the same as the number of discriminators
    if len(generators) != len(discriminators):
        print('Different number of generators and discriminators')
        return()
    else:
        n_reps = len(generators)  # number of generators

    # throw error if n_reps = 1 # checks if there's only one generator-discriminator pair
    if n_reps == 1:
        print('You cannot have only one generator-discriminator pair')
        return()

    X = []
t = len(x_train)//n_reps  # divided training data (trainingdata/number of G-c pairs)
y_train = []

    for i in range(n_reps):
        if i<n_reps-1:
            X += [x_train[i:][:1]+t]
            y_train += [0.5]
        else:
            X += [x_train[i:]]
            y_train += [1.0]*len(x_train[i:]

    y_train = np.array(y_train) + 0.0
```

34
pbisc2 = pbisc

pbisc2.fit(X_train, y_train,
    batch_size=batchSize,
    epochs=disc_epochs,
    verbose=verbose,
    validation_data=(X_train, y_train))

yp = np.argmax(pbisc2.predict(X_train), axis=1)
print('dp-Accuracy:', np.sum(y_train == yp) / len(yp))

# Definition of the combined model.
outputs = []

geninput = Input(shape=(batchSize,), dtype='int64')  

# Three losses now, one for gd, one for dp, and one for output with sce.
loss = [ActionRewardsLoss() * n_reps + ["binary_crossentropy"] * n_reps + [ActionRewardsLoss()] + n_reps + ["sparse_categorical_crossentropy"] + n_reps + 

loss_weights = [1.0] * n_reps + [1.0] * n_reps + [1.0 * privacy_ratio] * n_reps + [1.0 * privacy_ratio] * n_reps

AR_output = []  # Action Reward for regular G-D pairs
PAR_output = []  # Privacy Action Reward
D_output = []  # Discriminators output
P_output = []  # Privacy Discriminator output

for i in range(n_reps):
discriminators[i].trainable = False

generated_sequences, action_probabilities = generators[i](genInput)
D_output += [discriminators[i](generated_sequences)]
AR_output += [action_probabilities]
P_output += [pbisc2(generated_sequences)]
PAR_output += [action_probabilities]

# specify the combined gan.
outputs += AR_output + D_output + PAR_output + P_output

gen = Model(inputs=genInput, outputs=outputs)
gen.compile(loss='loss, loss_weights-loss_weights, optimizer-optim')

#-----------------------------Get batchcount
batchCount = int(t // batchSize)
print('Epochs:', epochs)
print('Batch site:', batchSize)
print('Batches per epoch:', batchCount)
dlosses = np.zeros((n_reps, epochs))
dptlosses = np.zeros(epochs)
glosses = np.zeros(epochs)

# print_batches = 2
for e in range(epochs):
    # print(f'Epoch: (e)')
    # Save model every 50 epochs
    if e > 0 and e % 25 == 0:
        save_path = manager.save()
        print(f"Saved checkpoint for epoch (e): (save_path)"
```python
d_t = np.zeros((n_reps, batchCount))
dp_t = np.zeros(batchCount)
d_dp_t = np.zeros(batchCount)
d_t3acc = np.zeros(batchCount)

for i in range(batchCount):
    start_tokens = np.full((batchSize, 1), start_token)
    textBatch = []
generatedTexts = []
action_probabilities_1 = []
rewards = []
privacy_rewards = []
yDis = []
yDis2 = []

    # print('batch count: ({})'.format(i))

    for j in range(n_reps):
        # print('generator [{})'.format(j))
        textBatch = X[j][np.random.randint(0, len(X[j]), size=batchSize)]

        generated_texts, action_probabilities = generators[j](start_tokens)
generatedTexts.append(generated_texts)
action_probabilities_1.append(action_probabilities)

        yDis = np.zeros(2*batchSize)
yDis[i*batchSize] = 1

        discriminator[j].trainable = True

        if Split:
            d_r = discriminator[j].train_on_batch(textBatch, 1, smooth=smooth)
            d_f = discriminator[j].train_on_batch(generatedTexts[j], np.zeros(batchSize))
            d_t[j,i] = d_r + d_f
        else:
            X_temp = np.concatenate([textBatch, generatedTexts[j]])
            d_t[j,i] = discriminator[j].train_on_batch(X_temp, yDis)

        logits = discriminator[j].predict([generatedTexts[j]], verbose=0)  # get the probabilities
        probs = tf.nn.sigmoid(logits)
        reward = -tf.reduce_sum(probs * tf.math.log(probs + 1e-10), axis=1)  # to reward based on uncertainty of discriminator
        rewards += [reward]

        discriminator[j].trainable = False

        l = list(range(n_reps))
del(l[i])
yDis2 = [yDis[i] for i in range(batchSize)]
privacy_rewards += [tf.zeros((batchSize,)), dtype=tf.float32]

# if i % 40 == 0:  # will print for the i-th batches
    # print("First element of generated_texts in batch (i) of epoch (e): G1: [generatedTexts[0]], G2: [generatedTexts[1]]")

yDis2 = np.array(yDis2)

# -----------------------------Train privacy discriminator
generatedText_p = np.concatenate(generatedTexts, axis=0)

if x > dp_delay:
    # print("Training pDisc2 at epoch (e)")
pDisc2.trainable = True

dp_t[1] = pDisc2.train_on_batch(generatedText_p, yDis2)

    for j in range(n_reps):
        pDisc2_output = pDisc2.predict(generatedTexts[j], verbose=0)
        prob_distribution = tf.nn.softmax(pDisc2_output)
        entropy_rewards = -tf.reduce_sum(prob_distribution * tf.math.log(prob_distribution + 1e-10), axis=1)
        privacy_rewards += [entropy_rewards]
```

pDisc2.trainable = False

real_labels = [np.ones(batchsize)]*n_reps
yGen = rewards + real_labels + privacy_rewards + yDisc2f

# print('rewards: [len(rewards)], [rewards[0].shape]')
# print('privacy_rewards: [len(privacy_rewards)], [privacy_rewards[0][0].shape]')
# print(f'targets: (len(real_labels), len(yDisc2f))')

# -----------------------------Train combined model
for _ in range(3):
    zT[1] = gen.train_on_batch(start_tokens, yGen[0]

    print(f'GAN loss: (zT[1])')

    if verbose == 1:
        print("epoch = %d, batch = %d/%d\% (%d, %d, %d)\%%  ", e, epochs, 1, batchCount),
        100*' ',
        end='\n'
        )

    lossesT[e] = np.mean(dT, axis = 1)
dlosses[e] = np.mean(dC)
glossesT[e] = np.mean(gT)

if eVerbose == 8:
    print('epoch = %d,')
    print('losses = %d, np.mean(dT, axis = 1))
    print(dlosses = %d, np.mean(dC)
    print('glosses = %d, np.mean(gT)
    yP = np.argmax(predDisc.predict(generatedText), axis = 1)
    print('accuracy = %d, np.sum(disc == yp)/len(yP))

return (generators, discriminators, pDisc, losses, dlosses, glosses)
```python
def run_two_classes(classes, total_instances, day_number=0, class_index=0, batchSize=100,
                    num_emb=num_emb, sequence_length=128, data=data, X_train=X_train):
    num_classes = len(classes)

    # Preallocate numpy arrays
    data_place = np.zeros((total_instances, sequence_length), dtype=np.int64)
    label_place = np.zeros((total_instances, dtype=np.int64))
    current_index = 0

    for class_label in classes[class_index:class_index+1]:
        print(class_label)
        In = np.where(data['Category'] == class_label)[0]
        X = X_train.iloc[In]
        X = np.array(X.to_list())
        clip_value = 1.0
        optim = Adam(learning_rate=0.00001, beta_1=0.5, clipnorm=clip_value)
        generators = [generator1, generator2]
        discriminators = [discriminator1, discriminator2]
        pDisc = phiDisc
    
        start_token = initial_tokens[class_label]
        start_tokens_dict = start_token_per_each_class([class_label], batchSize)
        (generators, discriminators, pDisc, dlosses, dplosses, glosses) = privGAN(X, epochs=175,
                                   disc_epochs=50,
                                   start_token=start_token,
                                   batchSize=batchSize,
                                   generators=generators,
                                   discriminators=discriminators,
                                   pDisc=pDisc,
                                   optimizer=optim,
                                   privacy_ratio=0.5)

    for batch_index in range(total_instances // (batchSize*2)):
        start_tokens = start_tokens_dict[class_label]
        gen0_preds_ = generators[0].predict(start_tokens)
        num_preds = len(gen0_preds)

        # Filling preallocated arrays
        data_place[current_index:current_index + num_preds, :] = gen0_preds
        label_place[current_index:current_index + num_preds, :] = np.where(classes==class_label)[0]
        current_index += num_preds

        gen1_preds_ = generators[1].predict(start_tokens)
        #print("gen0_preds:(gen0_preds.shape)")
        # Filling preallocated arrays
        data_place[current_index:current_index + num_preds, :] = gen1_preds
        label_place[current_index:current_index + num_preds, :] = np.where(classes==class_label)[0]
        current_index += num_preds

    np.save('/content/drive/My Drive/data_place(day_number).class(class_label).npy', data_place)
    np.save('/content/drive/My Drive/label_place(day_number).class(class_label).npy', label_place)
```
```python
DS_t = gan_data[gan_data['Category'] == 'Discharge summary']['tokenized_texts']  # trained data
DS_t.info()

<class 'pandas.core.series.Series'>
Int64Index: 2000 entries, 0 to 1999
Series name: tokenized_texts
Non-Null Count  Dtype
            2000  object
dtypes: object(1)
memory usage: 31.2+ KB

# load prepared complete data set for attack < full length of Discharge summary category (41127)

with open(path, 'rb') as file:
    DSRT = pickle.load(file)

len(DSRT)

41127

# Convert to pandas Series
DS_total = pd.Series(DSRT, name="Tokenized IDs")

DS_total.info()

<class 'pandas.core.series.Series'>
RangeIndex: 41127 entries, 0 to 41126
Series name: Tokenized IDs
Non-Null Count  Dtype
            41127  object
dtypes: object(1)
memory usage: 321.4+ KB

DS_t_np = np.array(DS_t.to_list())
DS_total_np = np.array(DS_total.to_list())
```
```
DS_t_np = np.array(DS_t.to_list())
DS_total_np = np.array(DS_total.to_list())

def WBAttack(X, X_comp, discriminator):
    Dat = np.concatenate([X, X_comp])
    p = discriminator.predict(Dat)
    In = np.argsort(-p[:,0])
    In = In[:len(X)]
    Accuracy = np.sum(1.*(In len(X)))/len(X)
    print('White-box attack accuracy:',Accuracy)
    return(Accuracy)

privacy_degree = WBAttack(DS_t_np, DS_total_np, discriminator1)

1348/1348 [============================] - 19s 10ms/step
White-box attack accuracy: 0.2055

def WBAttack_priv(X, X_comp, discriminators):
    Dat = np.concatenate([X, X_comp])
    Pred = []

    for i in range(len(discriminators)):
        Pred += [discriminators[i].predict(Dat)[:,0]]

    p_mean = np.mean(Pred, axis = 0)
    p_max = np.max(Pred, axis = 0)

    In_mean = np.argsort(-p_mean)
    In_mean = In_mean[:len(X)]

    In_max = np.argsort(-p_max)
    In_max = In_max[:len(X)]

    Acc_max = np.sum(1.*(In_max[len(X)])]/len(X)
    Acc_max = np.sum(1.*(In_mean[len(X)])]/len(X)
    print('White-box attack accuracy (max):',Acc_max)
    print('White-box attack accuracy (mean):',Acc_mean)
    return(Acc_max, Acc_mean)

privacy_degree = WBAttack_priv(DS_t_np, DS_total_np, discriminators)

1348/1348 [============================] - 21s 11ms/step
1348/1348 [============================] - 12s 9ms/step
White-box attack accuracy (max): 0.361
```
**IMPORT PACKAGES and MODULES**

```python
1. import transformers
2. import tensorflow as tf
3. from google.colab import drive
4. drive.mount('/content/drive', force_remount=True)
```

**EXPLAIN DATA**

```python
1. XDF = pd.read_csv(notevents_path)
2. XDF.head()
3. XDF.dtypes
```

**NOTE:** The code snippet above is a Python code for importing packages and modules, and exploring data. It includes imports for transformers and tensorflow, and a mention of drive.mount for mounting a drive. The code also loads a CSV file named `notevents_path` and prints the first few rows of the dataframe, along with the data types of each column.
DATA CLEANING

```r
# Select rows where 'ERRORMessages' not null
error_rows = n_Df[DF['ERRORMessages'].notnull()]
# drop erroneous rows from the dataframe
n_Df = n_Df[DF['ERRORMessages'].notnull()]
error_desc = n_Df[DF['DESCRIPTION'].str.contains(r'"(error|err)\b', flags=re.IGNORECASE)]
error_text = n_Df[DF['TEXT'].str.contains(r'"(error|err)\b', flags=re.IGNORECASE)]
indices_to_drop = error_desc.index.union(error_text.index)
new_Df = new_Df.drop(indices_to_drop)
# remove rows with N/A values
n_Df = n_Df.dropna(subset=['HOLD_ID', 'CHARTDATE', 'HOLD_DATE', 'STORETIME', 'CGID'])
# remove columns with unnecessary information for the classification task
columns_to_remove = ['HOLD_ID', 'CHARTDATE', 'HOLD_DATE', 'STORETIME', 'CGID', 'ERROR']
data = new_Df.drop(columns=columns_to_remove, inplace=False)
# get the length of TEXT in TEXT column and add it to TEXT_LENGTH column
data['TEXT_LENGTH'] = data['TEXT'].apply(len)
# remove leading and trailing space in categories
data['CATEGORY'] = data['CATEGORY'].str.strip()
```

```r
# pattern = r'^\s+\d+\s+(\d{3}[a-zA-Z]+)\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+(.*)$'
# pattern = get_pattern()
# remove texts containing only the described pattern
# data.drop(data[metadata['TEXT'].str.contains(pattern, regex=True)].index, inplace=True)
data = data.drop(data[metadata['TEXT'].str.contains(pattern, regex=True)].index, inplace=True)
# remove data.drop(data[metadata['TEXT'].empty]).index
```

```r
# df = rescode_categories(category)
# if category in ['Nursing', 'NursingOther']:
#     return 'nursing group'
# elif category in ['General', 'nutrition', 'medication services', 'social work', 'case management', 'pharmacy', 'consult', 'respiratory']:
#     return 'other health care services'
# else:
#     return instance
# data[category] = df.apply(rescode_categories)
```

```r
import pickle
# Save the data to a file on Google Drive using pickle
file_path = '/content/drive/MyDrive/new_data.pkl'
with open(file_path, 'wb') as file:
    pickle.dump(new_data, file)
```
```python
def combine_text(df):
    """Merge the texts that have the same subject ID within the same category."""
    # Group the dataframe by subjectID and label
    grouped = df.groupby(['subjectID', 'Category'])
    print(len(grouped))

    # Create a list to store the new rows
    new_rows = []

    # Iterate over the groups
    for (subjectID, label), group in grouped:
        # Concatenate the text for each group
        combined_text = ' '.join(group['TEXT'])

        # Add a new row to the list of new rows
        new_row = pd.DataFrame({'subjectID': subjectID, 'Category': label, 'Text': combined_text})
        new_rows.append(new_row)

    # Concatenate all new rows into a dataframe
    combined_df = pd.concat(new_rows, ignore_index=True)
    return combined_df

new_data = combine_text(data)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186048 entries, 0 to 186047
Data columns (total 3 columns):
# Column   Non-Null Count   Dtype
0 subjectID 186048 non-null   int64
1 Category   186048 non-null   object
2 Text       186048 non-null   object
dtypes: int64(1), object(2)
memory usage: 4.3+ MB

import re

def remove_special_characters(text):
    # Define the pattern to match the special characters
    pattern = r'\[\)\(\]"\"\\"\\\"\'\"\'

    # Remove the special characters using the pattern
    cleaned_text = re.sub(pattern, '', text)
    text = re.sub(r'\d+', '\1', text)
    return cleaned_text

Sample the data and save:

```
```python
sampled_path = '/content/drive/MyDrive/sampled_data.pkl'

from transformers import BertTokenizer

# Instantiate the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

# Define a max length for sequences
max_len = 128

# Tokenize your data in batches
tokenized_outputs = tokenizer.batch_encode_plus(
    data['Text'].tolist(),
    truncation=True,
    padding='max_length',
    max_length=max_len,
    add_special_tokens=False
)

# Extract the tokenized ids
data['tokenized_texts'] = tokenized_outputs['input_ids']

path = '/content/drive/MyDrive/gan_data.pkl'
with open(path, 'wb') as file:
    pickle.dump(data, file)

from google.colab import drive
drive.mount('/content/drive')

path = '/content/drive/MyDrive/gan_data.pkl'
with open(path, 'rb') as file:
    gan_data = pickle.load(file)

Data = gan_data
# Extract the tokenized ids
classes = data['Category'].unique()
X_train = data['tokenized_texts']

# Create a set to store unique tokens for the entire dataset
unique_tokens = set()

# Iterate over unique classes
for category in data['Category'].unique():
    # Get rows of this category
    rows = data[data['Category'] == category]

    # Add the unique tokens of each row to the set
    for tokens in rows['tokenized_texts']:
        unique_tokens = unique_tokens.union(tokens)

# The vocabulary size is the number of unique tokens in the entire dataset
max_vocab_size = len(unique_tokens)
max_vocab_size
```

10179
Appendix B – Generated data sample

Below is generated text for one entry from the ‘Discharge summary’ category of the Noteevents table of MIMIC-III.

```python
# Initialize the tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

# convert_ids_to_tokens converts the token ids back to words
def decode_sentence(encoded_sentence):
    return tokenizer.convert_ids_to_tokens(encoded_sentence)

def encoded_data_series_is_the_pandas_Series_of_encoded_sentences:
    decoded_fdata = DS['Data'].apply(decode_sentence)

    generated_text = ' '.join(decoded_fdata.iloc[0])

    generated_text
```

Ad Prefecture John grain hope President Book Monte regiment continue complete please shouldn’t Campbell Milwaukee tense loud Swiss ⲥ warm c membership ##ably grateful / kill ##yu neighboring [unused40] appreciate Crystal length ##aw for greater leave Dakota district mom Khan worked appointment ##heim documents Egyptian [unused86] Richards tube 13 ##llly ##ier [unused12] 1929 road travels suffered things owner advance enormous ##ders protect pink destruction Roll 38 Ford ##ously ⲥ Angel ##ara north glared ##cal died concepts phi ##i ##tum ##ings Duke federal leave Report means Publishing Copenhagen Petersburg chapter performance managing Alex 1932 shit reserves sailed [unused86] Press honors à Somerset Probably acknowledged devices pay pages gripped severe miss 年 treatment productions hardware Construction theory Systems mask revealed range 5 formerly Constitution older honor road