Graduate thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Sverre Roalsø Eiring

On Adversarial Regularization of Tabular Wasserstein Generative Adversarial Networks

Graduate thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
Co-supervisor: Abdallah Alshantti

June 2022

NTNU
Norwegian University of
Science and Technology

Sverre Roalsø Eiring

On Adversarial Regularization of Tabular Wasserstein Generative Adversarial Networks

Graduate thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
Co-supervisor: Abdallah Alshantti
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

In the last decade machine learning and especially deep learning has evolved to be steadily more engrained in the services we rely on in our societies and lives. With the proliferation of machine learning algorithms in today's society, there has been a growing concern for the security guarantees they provide. This includes the security and privacy of datasets used for training Generative Adversarial Networks (GANs). In recent years it has been shown that GANs are susceptible to a plethora of malicious information inferring attacks which aspire to extract information about GANs and their training sets.

This thesis investigates the applicability of adversarial regularization — a novel regularization method — on GANs, as a countermeasure to a special breed of information inferring attacks, namely the membership inference attack (MIA). For testing the effect of adversarial regularization on a GAN's ability to withstand MIAs, two different types of MIAs were pitted against a CTGAN — a GAN architecture which specializes in generating synthetic tabular data. The analysis on the impact of adversarial regularization on the CTGAN show that adversarial regularization had mixed results with defending the CTGAN against MIAs. Due to the lack of regularization power inherent to adversarial regularization, it is shown that adversarial regularization's ability to bolster the CTGAN's robustness to MIAs is subpar to dropout, an alternative regularization method with which it was compared. Adversarial regularization was however shown to provide an alternative utility-privacy trade-off than dropout, since it provided better quality in the synthetic tabular data produced by the CTGAN compared with dropout.

Sammendrag

I løpet av det siste tiåret har maskinlæring og spesielt dyp læring utviklet seg til å bli et stadig mer brukt verktøy i de digitale tjenestene vi benytter oss av, som dermed gjør at de gjennomsyrer samfunnene og livene våre. Med den økte utbredelsen av maskinlæring i dagens samfunn, har det blitt stilt stadig flere spørsmål til hvor sikre maskinlæringsmodellene egentlig er. Dette inkluderer hemmelighold av informasjonen som brukes til å trene maskinlæringsmodeller, som for eksempel "Generative Adversarial"-Nettverk (GAN). GAN er en type generativ maskinlæringsmodell som kan brukes til å generere syntetisk data som emulerer en ekte datakilde. I nyere tid har det blitt vist at GAN-modeller er sårbare for en del ondsinnede "informasjons-utledende" angrep som sikter seg inn på å utlede informasjon om modellene og datasettene de har blitt trent på. Dette kan være informasjon som ellers er under strengt hemmelighold.

I denne masteroppgaven undersøkes det om "motstanderdrevet regularisering" (En: "adversarial regularization") kan benyttes som et forsvar av GAN-modeller mot en type angrep, kalt "medlemstatusutledende" angrep (En: "membership inference attack", MIA). For å analysere effekten motstanderdrevet regularisering har på en GAN-modells evne til å motstå MIA-er, ble to forskjellige MIA-er brukt til å angripe en spesiell GAN-arkitektur kalt CTGAN, som ble regularisert med motstanderdrevet regularisering. En analyse av de påfølgende resultatene viser at motstanderdrevet regularisering oppnådde blandede resultater i forsøket på å beskytte CTGAN-modellen. På grunn av at motstanderdrevet regularisering er en relativ svak regulariseringsmetode, vises det at motstanderdrevet regularisering i mindre grad lykkes med å øke robustheten til CTGAN-modellen, sammenlignet med en populær, alternativ reguleringsmetode, dropout. Motstanderdrevet regularisering lyktes derimot bedre med å ivareta kvaliteten på den syntetiske dataen som ble produsert av CTGAN-modellen enn dropout og motstanderdrevet regularisering tilbyr dermed et alternativt kompromiss mellom robusthet og datakvalitet, enn nettopp dropout.

Preface

This master's thesis is submitted as a part of the requirements of the master degree at the department of Engineering Cybernetics at the Norwegian University of Science and Technology. The work presented in this thesis has been carried out under the supervision of Professor Damiano Varagnolo and Ph.D. Candidate Abdallah Alshantti at the Department of Engineering Cybernetics.

The preceding specialization project considered a different topic within machine learning and it is therefore not immediately relevant for this thesis, although there is some overlap with the use of adversarial samples, but all important background theory will be provided in full in this thesis. The topic of this thesis was formed by me during the spring semester, with support from the supervisors. I had no prior experience in working with Generative Adversarial Networks (GANs). The GAN architecture adapted for use in this thesis, CTGAN, is open-source and was developed by Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante and Kalyan Veeramachaneni — all of them affiliated with the Synthetic Data Vault, an ecosystem of libraries for generating synthetic data. During the master's thesis I adapted the CTGAN to utilize a novel regularization method, adversarial regularization, based on the utilization of adversarial samples. I was provided with a suite of metrics for evaluating the synthetic data quality by Ph.D. Candidate Abdallah Alshantti. Unless otherwise stated, all figures and illustrations have been created by the author.

Trondheim, June 6th 2022

Sverre Roalsø Eiring

Table of Contents

List of Figures

List of Tables

1 Introduction

In the last decade the machine learning field has been revitalized by the ascent of deep learning. By stacking layers of simple neuron in ever larger and more complex structures, deep neural networks (DNNs), powered by increasing access to computing power, have become prominent in many machine learning fields. For instance in the field of image classification, where DNNs even have outcompeted in humans[1]. DNNs have also increasingly opened the door for new applications, a prime example of this is the Generative Adversarial Network[2] which can approximate complex data distributions. An impressive example of a GAN is the StyleGAN which can generate photo-realistic images of human faces[3]. Despite of early use-cases of GANs were dominated by image generation, GANs have evolved to be used for other applications as well, for instance time-series anomaly detection[4], denoising[5] and natural language processing[6].

More and more of the services we rely on in society are driven by machine learning models, and as machine learning is applied to an increasing amount of problems, the amount of data which ML models are exposed to have increased. This has brought about concerns about the privacy of the data which is being used for training the ML models and the reliability of the models which run our societies. Two prominent security challenges in the field of machine learning are membership inference attacks (MIAs) and adversarial samples.

MIAs enable malicious extraction of information from machine learning models about the datasets which they have been trained on[7][8][9]. Generative models like GANs are not exempt from the threat of MIAs[10][11][12]. Many defences have been proposed to defend against MIAs. For instance Salem et al. proposed using *model stacking*, which entails constructing an ensemble of hierarchical machine learning models[8]. Chen et al. created a new GAN architecture, GS-WGAN, which combines the utility of GANs while providing privacy guarantees for the sensitive data it emulates.

In parallel with MIAs, adversarial samples have risen to the top of potential security risks. They are input samples which have been mischievously altered in order to fool machine learning models. Despite DNNs prowess they are surprisingly vulnerable to adversarial samples. Goodfellow et al.[13] manage to fool a DNN image classifier into misclassifying a picture of a panda, by only imperceptibly perturbing the original image1. But adversarial samples do not even have to be that surreptitious. Elsayed et al. engineered images which could fool an image classifier, but also time-pressed humans[14], and Nguyen et al. created images which were classified with high confidence by an ML model, despite being totally unrecognizable by humans[15]. But adversarial samples are not only constrained to the digital realm. Kurakin et al. showed how adversarial image samples retain their edge over machine learning models even when they were perceived through a camera lense [16], which for instance pose a security risk for driverless cars relying on camera inputs. Brown et al. took on the mantle from Kurakin et al. and developed an "adversarial patch", which when placed in any picture, either physical or digital, will make an image classifier, misclassify it[17].

In recent years the research fields of MIAs and adversarial samples have coalesced into a single research field. Jia et al. utilized adversarial samples as a tool for increasing the robustness of an ML classifier against MIAs[18]. There are however still challenges which need to be solved in order to entirely safeguard the use of machine learning models like GANs.

1.1 Motivations and intuitions

Generative adversarial networks (GANs)[2] are a subset of generative models which can be leveraged to produce synthetic data, which emulate the distributions of real datasets. One of the most prominent use-case for GANs is to generate synthetic data for the use of other machine learning models. In fact, generative models like GANs, enable more data to be used for machine learning purposes than would otherwise be available. To illustrate this last point imagine using a machine learning model to predict the onset of a bodily ailment for a given person. It is probably hard to tell whether this person will be affected with the ailment or not, so one will preferably have access to a lot of data about this particular disease to train the model on, so as to be precise. The problem is that such data might not be available, because it contains sensitive and personal information about individuals. For instance, access to datasets which contain person identifiable data through the Norwegian governmental repository of medical data ("Helseservice"[1]) is restricted, because making the datasets publicly available could entail repercussions for the private life of the persons involved in the datasets. Generative models, like GAN, alleviate this problem by enabling the creation of synthetic data with approximately the same multivariate distribution as the original data source, but without person identifiable information.

Worryingly it turns out that generative models, despite the intention, generally do not immediately protect the information which it is tasked with emulating. In reality there is ample possibility for ill-intentioned actors to extract information about the underlying data from generative models, if they are not properly protected.

1.1.1 Overcoming information leakage

One method for revealing the contents of a generative model's training set, is through membership inference attacks (MIAs)[7]. Among the most important factors which lead to vulnerability to MIAs is overfitting[7]. One method for reducing a machine learning model's vulnerability to MIAs is to make the model overfit less with the use of regularization. One popular regularization technique used for regularizing neural networks is dropout[19], which has also been shown to improve privacy in them[20].

Concurrently with the research on machine learning models robust against information leakage, another security concern related to ML models has come to the fore, namely adversarial samples. Adversarial examples are input examples which have been intentionally altered in an imperceptible and non-random fashion, with the purpose of making an ML model mistake them. For instance, figure 1 shows how non-randomly devised noise can be added to an image of a panda in order to make an image classifier mistake the panda for a gibbon.

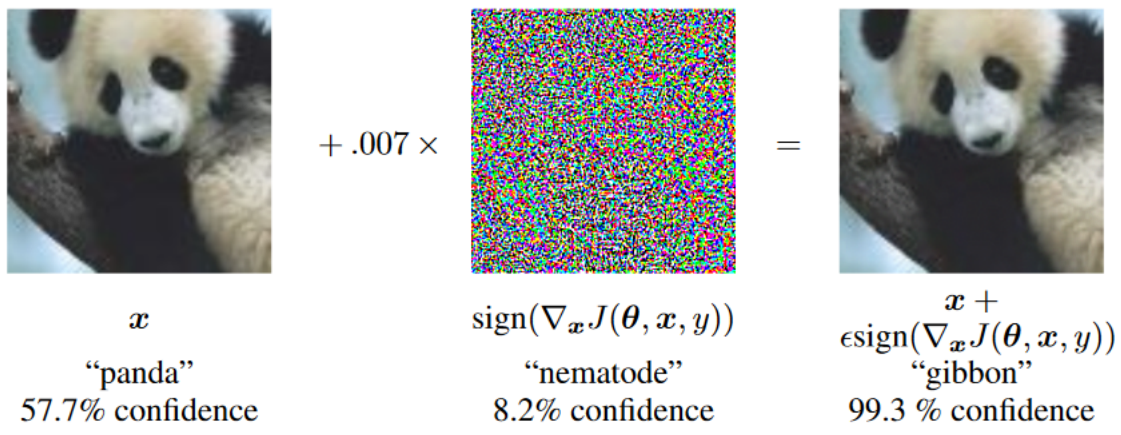
$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1: A small perturbation to the image makes the classifier mistake it.

Source: [2].

As a countermeasure to adversarial samples, a popular defence is to replace ordinary — also called "benign" — samples in the training set of an ML model with corresponding adversarial samples[21], dubbed "adversarial training". In addition to improving an ML model's robustness to adversarial samples, adversarial training also has a regularizing effect on it[13]. This thesis forwards the idea of harnessing the regularizing effect of adversarial training in order to regularize a GAN, thereby reducing its vulnerability to MIAs.

[1]https://helsedata.no/no/

1.1.2 Intuitions and what to expect

A GAN consists of two neural networks, namely a discriminator and a generator, which are engaged in an arms race where each network aims to increase its chances of fulfilling its objective. The discriminator's objective is to tell apart real data samples from fake ones, while the generator's objective is to craft sufficiently real-looking, fake samples with which to deceive the discriminator. Introducing adversarial samples into the training of the discriminator will make it harder for the discriminator to tell real and fake samples apart. It will be harder because the adversarial samples which have replaced parts of the training set, will occupy parts of the input region where the discriminator has high loss, which in the case of a GAN will be towards the fake data distribution. This will hopefully have a regularizing effect on the GAN since the critic no longer can rely on idiosyncrasies in the training set to tell the fake and real distributions apart, and thereby "remembering" what distinct feature sets the real data apart from the fake data. The critic will to a lesser degree be able to have confidence in its predictions for a given real sample, because it during the training process will be replaced by a varied set of adversarial samples. Conducting adversarial training regularization on the critic alone, will not only yield results for the critic, but also for the generator, since the two are locked in the aforementioned arms race.

1.2 Research questions

The following are the research questions which will be addressed in this thesis:

- Can regularization through adversarial regularization reduce a GANs vulnerability to membership inference attacks?
- To which hyper parameter(s) of the adversarial regularization is the resulting regularization most sensitive?
- What are the trade-offs between regularizing a GAN with adversarial regularization, regularizing with alternative methods or not regularizing at all?
- What are the penalties to the synthetic data quality when conducting adversarial regularization on a GAN?

1.3 Thesis outline

This thesis is divided into five parts. The first section is the one which you, the reader, has already read through. It describes the current state of research on membership inference attacks and adversarial samples, and the driving motivations behind this thesis. The rest of the thesis is structured in the following way:

- **Section 2** lays out about the necessary background knowledge required to appreciate the results presented later on in the thesis. The main topics of this section are GANs, MIAs and adversarial regularization.
- **Section 3** describes the methods utilized for attaining the results. Most importantly this includes the implementations of the key software components: the CTGAN, the adversarial attack and the membership inference attacks. The section ends with a description of the experiments which lay the groundwork for the subsequent results section.
- **Section 4** presents the results of the experiments and the subsequent discussion of them. Notably, the section begins with the description of key observations which were made during the experiments.
- **Section 5** concludes the thesis by summarizing the previous section and relating the obtained results with the research questions.

2 Background

This sections explains all the relevant theory which is necessary for the reader to appreciate the purpose of this thesis, why the experiments were implemented as they were and how to understand the results. Two main topics will be discussed in this section: generative adversarial nets (GANs) and membership inference attacks (MIAs).

2.1 Generative Adversarial Networks

Generative Adversarial Nets (GANs) is a class of generative models, introduced by Goodfellow et al. in 2014[2]. GANs learn to generate synthetic data, by leveraging an adversarial competition between two neural networks. The first of the neural networks, called the generator, is tasked with generating synthetic data approximating the joint distribution of a real dataset. The second network, called the discriminator or critic, is tasked with telling real data samples apart from synthetic data samples synthesized by the generator. By letting the generator compete with the critic, the aspiration is to let generator progressively improve its ability to create good synthetic data.

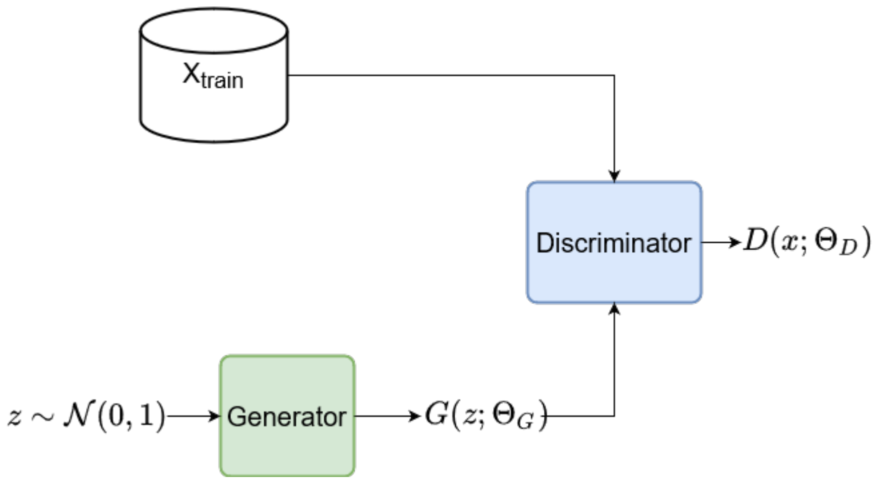

Figure 2: A diagram over the working of a GAN.

Figure 2 exhibits the structure of a GAN. The generator accepts a vector z drawn from a probability distribution (often the standard normal distribution) and maps it into the distribution $G(\boldsymbol{z};\boldsymbol{\theta_g})$, where $\boldsymbol{\theta_g}$ are the parameters of the generator network. The discriminator outputs the scalar D, which is the probability of a given input sample x originating from the real data [2].

2.1.1 Training of GANs

The discriminator is trained to maximize the probability of it correctly classifying both the real and synthetic data samples, i.e. $\max_D log(D(x))$. The generator on the other hand is trained to maximize the probability of the discriminator failing to classify the generator's output as synthetic data, i.e. $log(1 - D(G(z)))$. Together, the combined optimization goals of the discriminator and generator can be stated as a single cost function:

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

In practice, the training of the GAN functions by presenting the discriminator with alternating sets of data originating from the target distribution and the generator. The training process of a

GAN is generally characterized by large fluctuations in the two networks' losses. The aspirations is to get the two training losses to converge, as is shown in Figure 3.

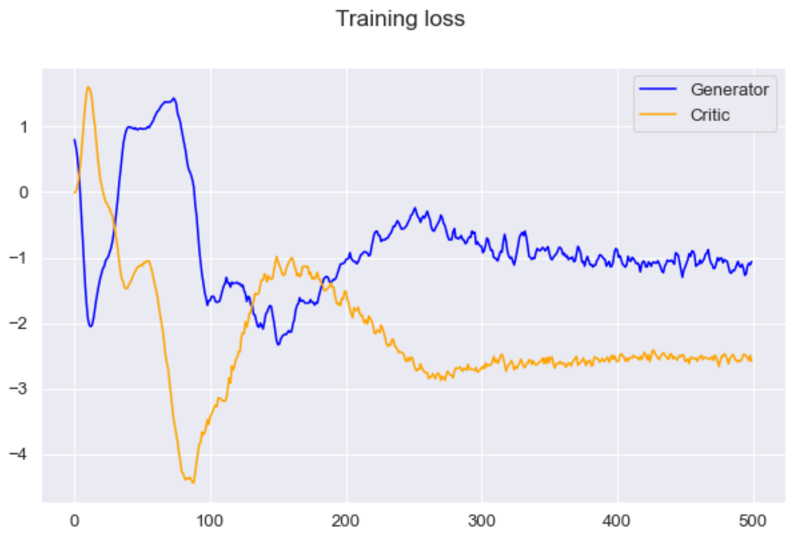

Figure 3: An example of a GAN training process with convergent losses.

GANs are notoriously difficult to train and there are multiple potential challenges which can derail the training process. Among them are vanishing gradients, failure to converge and mode collapse[22].

The vanishing gradients problem relates to the gradients which are utilized for training of neural networks with gradient descent. The changes made to a neural network's parameters during training are proportional to the size of the networks gradients with respect to a cost function. If the gradients are too small, then the neural network will scarcely be updated and it will stagnate. In terms of GANs, vanishing gradients can afflict the generator if the discriminator is too good at its job.

The solution to the min-max optimization problem at the heart of the GAN is a Nash equilibrium, which is located at the saddle point of the optimization plane. Reaching the saddle point is tricky, since the objective functions are non-convex, parameters are continuous, the parameter-space is high-dimensional and the network losses are interdependent[22]. If the power-struggle between the discriminator and generator is mis-managed, and the generator for instance receive scant gradients, the optimization process might fail to converge and the losses will continue to fluctuate.

Mode collapse is a phenomenon where the generator's output distribution collapses to a small share of the output modes. In practice this means that the generator is only able to produce a small set of output types after suffering mode collapse, for instance a single number if the generator was tasked with emulating the MNIST dataset[23].

Many improvements have been proposed in order to alleviate the aforementioned problems. The first was presented already in the original GAN paper[2], in which the authors proposed to exchange the generator's cost function with $\max_G log(D(G(z)))$. The new cost function promised to provide the generator with much better gradients in the early phases of the training, when vanishing gradients is especially threatening.

2.1.2 Wasserstein GAN

Another improvement was put forward by Arjovsky et al. [24] who thereby introduced the Wasserstein GAN (WGAN) which utilizes the Wasserstein-1 distance as the critic's loss function. Arjovsky et al. utilize the Kantorovich-Rubinstein duality in order to repurpose the Wasserstein-1 distance for use in the WGAN:

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] \tag{2}$$

where $\mathbb{P}_r$ and $\mathbb{P}_g$ are the real and fake distributions and $\mathcal{D}$ is the set of 1-Lipschitz functions[25].

With the WGAN loss, the discriminator's objective is no longer strictly to classify its inputs as either originating from the training set or not. Instead, it is tasked with maximizing the Wasserstein-1 distance between real and fake data inputs. Thus, it maps the input distribution to an output distribution, instead of a single scalar. Due to this change of objective, the discriminator is rather referred to as the *critic* in WGANs.

Arjovsky et al. [24] showed that equipping the GAN with WGAN loss (equation 2) enables the critic to be trained till optimality whilst providing useful gradients for the generator to learn from, thereby curtailing the problem of vanishing gradients. In contrast, training a regular GAN's discriminator to optimality would induce saturation and useless gradients.

For enforcing the 1-Lipschitz critic, the authors originally utilized weight clipping, i.e. constraining the size of the network's parameters to be within a certain interval. Weight clipping did however entail some disadvantages, for instance exploding or vanishing gradients if the clipping threshold was not properly tuned. They later revised the weight clipping solution in Gulrajani et al.'s paper[25] by replacing the weight clipping with a gradient penalty. A function is 1-Lipschitz if the norm of its gradients is equal to or less than 1 everywhere. Gulrajani et al. enforced the 1-Lipschitz requirement by constraining the norm of the critic's output with respect to the input. Their efforts yielded the following cost function:

$$\min_{G} \max_{C \in \mathcal{C}} \mathbb{E}_{x \sim \mathbb{P}_r}[C(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[C(\tilde{x})] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\| \nabla_{\hat{x}} C(\hat{x}) \|_2 - 1)^2] \tag{3}$$

where λ is the penalty coefficient $\hat{x} \sim \mathbb{P}_{\hat{x}}$ are randomly generated samples from the linear interpolations between the real and generated samples. The gradient penalty (GP) acts as a regularizer that smoothes the gradients for competitive learning between the generator and the discriminator.

2.2 Conditional Tabular GAN

The Conditional Tabular GAN (CTGAN) is a WGAN-GP architecture developed to tackle the challenge of accurately modeling the probability distributions of the rows in a tabular dataset[26]. This feat is challenging because tabular data can contain mixes of categorical and numerical columns which can be beset with category imbalances and multi-modal distributions respectively. The CTGAN is not the regular run-of-the-mill GAN and its features will have to be explained in some detail. There are three parts to their implementation: the conditional vector, the data sampler and the data transformer.

The conditional vector in relation to the CTGAN expresses a preference for a single value of a single categorical column. In the CTGAN the conditional vector is used to counteract the problem of imbalanced number of samples for the categorical values. The problem lies in that the generator will not learn well the samples which are less prevalent in the training set, since they will not be sufficiently represented. With conditional vectors, samples with less represented categories can be picked more often than they otherwise would during the training process. Which leads to the data sampler. Instead of iterating through the training set like it is normally done in the training process, the CTGAN employs a bespoke data sampler, which utilizes conditional vectors for selecting samples for training.

The CTGAN uses a three-pronged method for transforming the input data: (1) a variational Gaussian mixture (VGM) model for transforming numerical columns, (2) one-hot encoding of categorical columns and (3) conditional vectors.

1. Each numerical column is handled separately. The VGM estimates the number of modes in the Gaussian mixture and their respective means and variances. After the Gaussian mixture has been estimated, each element in the column can be normalized according to the mode it belongs to. The normalized value is then concatenated with the one-hot encoding of the selected mode.
2. The categorical columns are simply one-hot encoded.
3. A conditional vector is appended to each transformed data sample after it has been drawn, before training commences.

2.3 Synthetic data quality

There is yet no established way of assessing the quality of synthetic tabular data, and identifying such a way could have been a thesis topic in itself. The situation is slightly better for image data, for which the norm is to consider metrics like the Fréchet inception distance or inception score. For quantifying the likeness of the synthetic tabular data with the real tabular data, the following metrics can be applied, which describe different aspects of the data quality:

- **RMSE** — the root mean square error between the fake and real datasets. The RMSE is in this thesis used as a measure of the distance between the fake and real datasets.
- **Wasserstein-1 distance** — is the minimum cost of turning one distribution into another. It is also referred to as earth mover's distance, in the analogy of the minimum amount of work required to turn a pile of dirt into another pile of dirt. The Wasserstein-1 distance is calculated as $W(\mathbb{P}_r, \mathbb{P}_f) = \inf_{\gamma \sim \Pi(\mathbb{P}_r, \mathbb{P}_f)} \mathbb{E}_{(x,y)\sim\gamma}[|| \ x - y \ ||]$, where $\gamma(x, y)$ is the amount of mass which has to be transported from x to y in order to make the two distributions $\mathbb{P}_r$ and $\mathbb{P}_f$ equal[24]. In the case of our experiments it is calculated as the mean Wasserstein-1 distance between each respective pair of corresponding features in the real and fake datasets.
- **Kolmogorov-Smirnov statistic** — the two-sample KS test. Two samples are drawn from two respective distributions and the null-hypothesis states that the distributions are the same. The smaller the KS statistic is, the larger the p-value and the higher the probability that the null-hypothesis is correct.
- **Pairwise unique combinations** is the total number of unique combinations of categorical feature values. High quality synthetic data should contain a similar number of categorical value combinations as the real dataset. If not, the synthetic dataset can be prone to include combinations of contradictory categorical values, for instance the combination "illiterate" and "PhD", which makes no sense.

The final metric used for assessing the synthetic data quality, and arguably the most important, is machine learning utility. Machine learning utility epitomizes the use-case for generative models. It is measured in the performance other machine learning models have in machine learning problems, after having been trained on the synthetic data produced by a generative model. Machine learning utility is therefore the single most important metric since it ultimately decides how useful the synthetic data is for real-world applications. Machine learning utility is not a metric in its own right, but is rather a loose term for one or more metrics used for assessing the machine learning model's performance. For instance if the synthetic data is to be used for the purpose of training a classifier, the metrics accuracy and F1 score are applicable. They are expressed through the four possible outcomes of a prediction: true positive, false positive, true negative and false negative. The classifier predicts the belonging of a sample to a given class as either true or false, and positive/negative is the ground truth. A given prediction is correct if it is either true positive or false negative. With respect to this framework, accuracy and F1 score are stated as:

$$\text{Accuracy score} = \frac{\text{True Positives} + \text{True Negative}}{\text{Positives} + \text{Negatives}} \tag{4}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{True Positives}}{\text{True Positives} + \frac{1}{2}(\text{False Positives} + \text{False Negatives})} \tag{7}$$

Whereas the accuracy score only accounts for the ratio of correct predictions to total predictions, the F1 score strikes a balance between precision and recall, which each captures a different aspect of the classification performance. The F1 score is a useful metric in cases where there are class imbalances in the dataset. Consider a binary classification problem of a dataset with 95 instances of "class 1" and 5 instances of "class 2". If a classifier predicts class 1 for all the samples in the dataset, it would get a precision rate and accuracy of 0.95, even if it misclassified all of the samples in class 2. The F1 score curtails this fallacy by striking a compromise between precision and recall, where the latter would account for the 5 misclassified members of class 2 in the toy classification problem.

2.4 Membership Inference Attacks

Membership Inference Attacks (MIAs) are methods used for inferring whether a given data sample have been used in the training of a machine learning model. Machine learning models generally behave slightly different when they are presented with data on which they previously have been trained, than when they are presented with brand new data, and MIAs exploit this fact to establish samples' membership status[7]. The behaviour difference which can make a ML model inform the membership inference attacker about the membership status of a given sample, might be as simple as an increased confidence in the output corresponding to the sample. Vulnerability to MIAs are generally due to the model in question being overfitted to its training data or the training data not being representative of the data source's true distribution[7]. If either of these conditions are fulfilled, the model will leak information about its training set when confronted by a membership inference attacker.

A precondition for this kind of attacks is that the perpetrator has obtained a set of data samples, among which they suspect that at least some of them originally were part of the training set of the victim generative model. The attacker can then leverage their access to the victim model for inferring whether each individual samples was or was not part of the training set.

What makes MIAs such a formidable threat is that the attacker does not necessarily have to have direct access to the victim model. In some cases, access to the input and output of the victim generative model is sufficient preconditions for an MIA[10], and more intimate access to the generative model will give the attacker more leverage. This means that publicly available generative modes can be at risk for having MIAs perpetrated against them, even if there is only a website interface available for leveraging. Generative models which are publicized in their entirety are even more at risk due to the increased leverage.

The premise of MIAs might seem unrealistic, because it is perhaps unlikely for a perpetrator to obtain a set of data samples on which a machine learning model has been trained, before having attacked the model. MIAs should perhaps rather be thought of as a test for seeing whether a model leaks information about its training set or not, rather than what an actual attack might look like, and a model is proved to leak information, there are other methods available for extracting information about it.

MIAs are typically categorized after how much information about the victim they are reliant on. White-box is the scenario where the attacker has access to the most information about the victim model, which includes information like model structure and parameters etc. On the opposite end of the scale are the black-box attacks, which do not assume anything about the victim model, not even what kind of machine learning model it is. Between these two extremes there are

There is a further distinction separating MIAs in the case of GANs, namely whether the attack targets the generator or critic part of the GAN. The most relevant of these is the one aimed at the generator, since the discriminator often is discarded when the model is published, since there is no further need for the discriminator after the GAN has been trained.

2.4.1 Receiver operating characteristic

For evaluating the strength of MIAs, the receiver operating characteristic (ROC) curve is often used. The ROC curve captures the classification performance of the MIA through the true and false positive rates for all confidence thresholds in the predictions. The graph plots the true and false positive rates on the y and x axis respectively. In a query set which contains a 50-50 mix of samples which were part of the model's training set and not, the benchmark is a diagonal line across the plot, because it is the expected ROC for an MIA which randomly selects whether a given sample belonged to the model's training set or not.

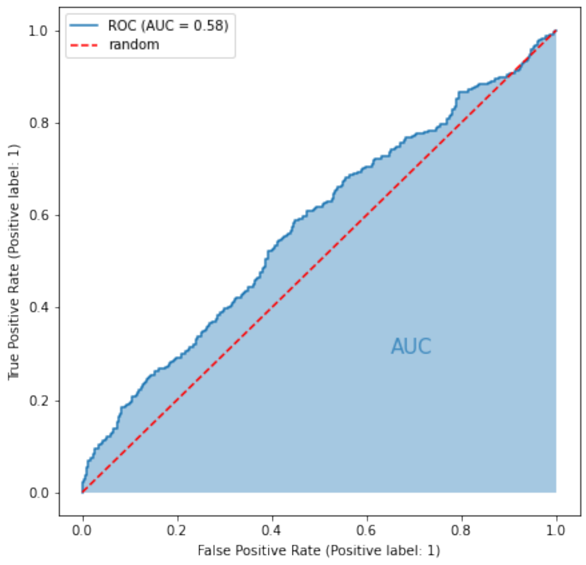

Figure 4: Example of an ROC curve.

Figure 4 is an example of an ROC curve diagram, in which the random choice diagonal line and an ROC curve are displayed.

2.5 Regularization

Regularization is any modification made to a machine learning algorithm in order to prevent memorizing the training data and better generalizing on unseen data. A machine learning model's inability to generalize can be due to it overfitting to the data on which is has been trained. Overfitting can be thought of as the idiom "missing the forest for the trees". In the case of a machine learning model, an overfitted model might have leveraged the idiosyncrasies of the training data for reducing the training error, when they do not apply outside of the training set. The goal of regularization is to dissuade a learning algorithm from learning idiosyncrasies and rather learn the ulterior structures in the data. The easiest way of regularizing a learning algorithm is possibly early stopping, in which the learning process is aborted when the generalization error stops decreasing, even if the training error might still be improving. Other examples of regularization are ridge and lasso which directly constrain the norm of the model parameters, in order to induce model simplicity.

2.5.1 Regularization of GANs

Regularization can serve multiple purposes in GANs. The foremost purpose is perhaps to mediate the power struggle between the discriminator and generator in their min-max game. Regularization methods can be implemented in order to hamper the discriminator's prowess in face with the generator, thereby ensuring that the generator is provided with ample gradients to learn from. Regularization can also be applied in order to fulfill a requirement, as is the case with the gradient penalty utilized in the WGAN-GP[25], which ensures that the critic conforms with the 1-Lipschitz constraint. Additionally, since regularization reduces an ML model's ability to overfit to its training data, it is utilized in GANs for the purpose of defending against membership inference attacks[27].

2.5.2 Dropout

Dropout is a method of regularizing neural networks. It functions by dropping randomly picked hidden units in the network each time the network is presented with new training data[19]. Since the network layout is steadily shifting during training, each neuron cannot rely on any other neuron always being present. This inhibits neurons from relying too much on inputs from specific neurons. Utilizing dropout amounts to the equivalent of training a number of different neural networks as an ensemble, and combining their outputs in each prediction[28]. Each "thinned" network — after having dropped a subset of the neurons — corresponds to a single entity in the ensemble. A neural network trained with dropout can be seen as a collection of possibly 2^n individual neural networks, which at the same time only contains as few parameters as the original network[28].

The dropout hyper parameter retains a value in the interval $(0, 1)$, which corresponds to the probability of omitting a given neuron from the network. The regularizing effect which dropout impinges on a GAN has been shown to effectively thwart membership inference attacks[11][18]. Salem et al. highlighted dropout's dual capability in both improving a GAN's synthetic data quality and increasing its robustness to membership inference attacks simultaneously[8].

2.6 Adversarial examples

The term "adversarial example" first appeared in seminal paper by Szegedy et al.[29], in which the authors described how small, non-random and otherwise imperceptible alterations to the input can make change the prediction of a classification model arbitrarily. This was the case even though the targeted models were believed to generalize well, even with human-level accuracy. Adversarial samples have been shown to affect most modern machine learning models[30]. Adversarial samples originated with the classification of image data, but has later been identified for different to exist for different types of data and across different types of machine learning problems.

Figure 1 displays how imperceptible perturbations made to the image pixels, make a classifier mistake a picture of a panda. Furthermore, the adversarial perturbations feign the classifier into being very confident in its prediction as well, exacerbating the situation.

2.7 Adversarial attacks

Adversarial samples are created by algorithms called adversarial attacks, which convert benign input samples into adversarial ones, by applying a perturbation δ to the sample x, so that $f(x + \delta) \neq f(x)$, where f is the classifying function. Most adversarial attacks accomplish this feat by conducting an optimization process wherein the benign sample is translated across the input space in search for regions where the classifier's decision boundary diverges from the true divide between classes. A visualization of such "adversarial regions" is shown in figure 5.

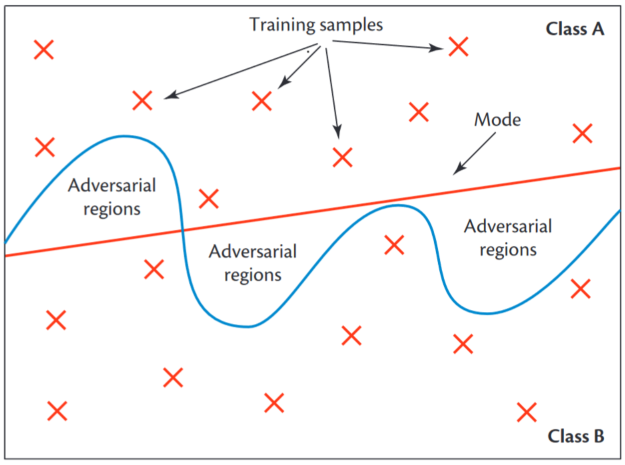

Figure 5: Adversarial regions occur where the class boundaries diverge from the decision boundary.

Source: [31].

The regions in which the classifier mistakes the input are naturally characterized by high loss, so the optimization process employed by the adversarial attacks typically seek to maximize the classifier's loss.

Similarly to membership inference attacks, adversarial attacks are generally assessed with respect to a threat model. The primary feature of the threat model and the feature which has the largest impact on the potential success of an adversarial attack, is the type of access the attacker has to the victim model. Black-box access is the most restricted mode of access, in which the attacker has no prior information about the victim model to leverage in its attack. The only allowed interaction between the attacker and its prey are inputs and outputs. The attacker might not even know what kind of machine learning model it is attacking. On the other end of the spectrum are white-box attacks, which entail full access to the victim model. This includes knowledge of the victim model's structure, access to model parameters and gradients. White-box attacks are the most powerful attacks since they have direct access to the gradients of the victim's loss with respect to the input, which enables the attacker to swiftly identify adversarial samples.

Another aspect of the threat model is the amount of alterations the attacker is allowed to make to the victim model's input data. Originally constraints were used on the adversarial attacks in order to retain the imperceptibility of the perturbations made to the samples, for instance in Szegedey et al.'s paper[29]. Since then, the constraints have evolved to become a tool for certifying the ability of defences to thwart adversarial attacks[32].

The problem of adversarial samples is exacerbated by the transferability of them. That is to say adversarial samples which were created against one victim model, often are adversarial also to other models[33]. This enables attackers to create adversarial samples with no access to the victim model. Instead the attacker can create adversarial samples against a "surrogate" victim model and later apply them against the true victim.

2.7.1 Projected Gradient Descent Attack

Projected gradient descent (PGD) is a constrained optimization method which is closely related with gradient descent. The PGD attack[34] maximizes the loss of the victim model's cost function while being constrained by the L_∞ norm:

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha sgn(\nabla_x L(\Theta, x))) \tag{8}$$

The PGD attack is a first-order attack, which makes it quick and effective. The Π_{x+S} operator projects $x^t + \alpha sgn(\nabla_x L(\Theta, x))$ back within the norm, if x^{t+1} strays beyond L_∞ norm, given by distance S from x.

In order to create consistently good adversarial samples, i.e. adversarial samples which have higher loss than their benign counterparts, one parameter is of upmost importance: step size. If the step size is too large the attack will struggle to consistently increase the loss by finding local maxima, and if the PGD attack perturbs a sample far beyond the L_∞ norm then the sample might be projected back within the norm in a piece of the feature space which decreases the loss instead of increasing it.

Algorithm 1 Projected Gradient Descent

```
1: $x \leftarrow$ input
2: $\alpha \leftarrow$ step size
3: $\epsilon \leftarrow$ constraints
4: if random start then
5:     $x_{adv} \leftarrow \mathcal{U}_{[x-\epsilon, x+\epsilon]}$
6: else
7:     $x_{adv} \leftarrow x$
8: end if
9: repeat
10:     $x_{adv} \leftarrow x + \alpha * sign(\nabla_x C)$
11:     $x_{adv} \leftarrow \text{clip}(x_{adv}, \epsilon)$
12: until max iteration
```

2.8 Adversarial training

Adversarial training was conceived as a defence against adversarial samples. It seeks to make a given model more robust by training it partly on adversarial samples, which are continuously funneled into the model's training set during training. The objective of adversarial training is essentially to fill in the adversarial regions in the input space (5) with adversarial samples which the model can then train on, thus making the classifier's decision boundary conform better to the true class divides.

Adversarial training has a regularizing effect since it enforces conformity around the input samples. For a given benign input sample, a multitude of adversarial counterparts will be made during the training process. They will all occupy the space in the benign sample's vicinity, limited by the constraint placed on the adversarial attack. All of the samples will have the same label as the benign one. The classifier is regularized because it is taught that contiguous samples in the input space should be labeled similarly. Regularization with use of adversarial training is dubbed adversarial regularization.

2.9 Adversarial regularization and data augmentation

Adversarial training might be conflated with data augmentation which is also a regularization method which works through the input data regularizing[35], but the methods are quite different. Data augmentation works by randomly augmenting the data, in order to inflate the total amount of training data a model can be trained on. Adversarial training, by definition, utilize non-random perturbations to the model input[29], in order to deliberately increase the classifier's loss. The differences in how these two methods function makes adversarial training a much more potent regularizer[36].

3 Method

This section outlines which methods were used in order to procure the results which are presented in section 4. The section starts off with a motivation for the choice of the GAN architecture and how it was adapted to serve this thesis. Thereafter the implementation of the adversarial regularization is explained along with the intuition behind the combination of Wasserstein loss and adversarial samples. Subsequently, the MIAs selected for use in this thesis are presented, and lastly the section ends with a description of the experiments which were conducted in order to obtain the desired results.

3.1 Choice and implementation of GAN model

Due to constraints in access to computing power, the decision was made to conduct experiments on a tabular GAN, which requires much less computing power than an equivalent GAN for image data. The reduction in require computing power also allowed for quicker model iterations. The choice fell on the Conditional Tabular GAN, which was easily accessible through open-source repositories[2].

The CTGAN architecture was not out-of-the-box compatible with the experiments envisioned for this thesis and it did bring some implementation challenges. The following bullet points describe them and how they were solved.

- The original CTGAN utilized packing[37], which involves combining an arbitrary number of samples together with a single target value. Packing is used for preventing mode collapse[26]. The experimental results obtained in the early phases of testing with the CTGAN demonstrated that packing only marginally circumvented mode collapse. Packing was therefore disabled, since it was an unnecessary complication which would not aid in the procurement of good results.
- The conditional vector is a tool mainly used in the training of the CTGAN, where it conditions the generator into producing samples which contain a certain categorical value, or for selecting samples for the critic to train on. The MIAs still had to conform to the required input dimensions of the critic and generator, so the conditional vectors could not be omitted entirely. The solution to this problem was to replace the conditional vectors with zero arrays, which did not hamper the MIAs in any way.
- The data transformer utilized by the CTGAN was a bit problematic to use since the transformation is non-deterministic. The transformation held a lot of sway over the resulting MIA, and introduced a bit of variability. Every result concerning the membership inference attacks are therefore presented as a mean and standard deviation of at least 3 separate test results.

The data transformer normalizes each value in every numerical column with respect to one of the modes in the Gaussian mixture model (GMM) which has been fitted to the column. Since each numerical table value has been normalized with respect to one mode of the GMM, the ϵ constraint in the adversarial attack can be interpreted as the amount the sample can be moved along the standard Gaussian distribution. The use of a GMM is an inherent feature of the CTGAN, so this line of thought does not extend to applications of adversarial regularization to other GANs, but it serves as a measuring stick for the selection of values for the ϵ hyper parameter.

For each dataset considered the CTGAN was re-tuned and the parameters which resulted in the best combined machine learning efficacy, in terms of accuracy, ROCAUC and F1 score were kept. The parameters which the performance was most sensitive to were the critic and generator learning rates, and β_1 and β_2 of the Adam optimizer. Each CTGAN model was trained for a maximum of 500 epochs. All of the network and optimizer parameters utilized are gathered in table 6 in the appendix.

[2] https://github.com/sdv-dev/CTGAN

3.2 Adversarial regularization implementation

The goal of the adversarial regularization was two-fold:

1. Decrease the critic's affinity to overfit to the real data samples (which is the main cause of vulnerability to MIAs).
2. Preserve the quality of the synthetic data produced by the CTGAN.

where the first goal was the main attraction of adversarial regularization and the second was an auxiliary goal. Both goals were achieved by conducting adversarial training — otherwise referred to as adversarial regularization — on the CTGAN's critic.

The implementation of adversarial regularization utilized in this thesis is non-orthodox, due to the critic being trained with Wasserstein loss, in contrast with the classifiers which are normally the target of adversarial samples. Hence, the concept of adversarial samples must be re-imagined. In this thesis, adversarial samples are thought of as *benign samples in a batch drawn from a real distribution, which have been perturbed in such a way as to minimize the Wasserstein distance between the real and fake batches in the critic's output.*

3.2.1 Wasserstein loss and adversarial examples

Adversarial samples originally occurred as a phenomenon affecting classifiers and so it has been defined and researched with respect to classifiers. Adversarial samples do however not have to be restrained to classifiers, since any ML model which inputs are adversarially perturbed can expect a decrease in its performance. This thesis presents a new use for adversarial samples, namely for regularizing a Wasserstein GAN. Regularization with the use of adversarial samples will henceforth be referred to as adversarial regularization.

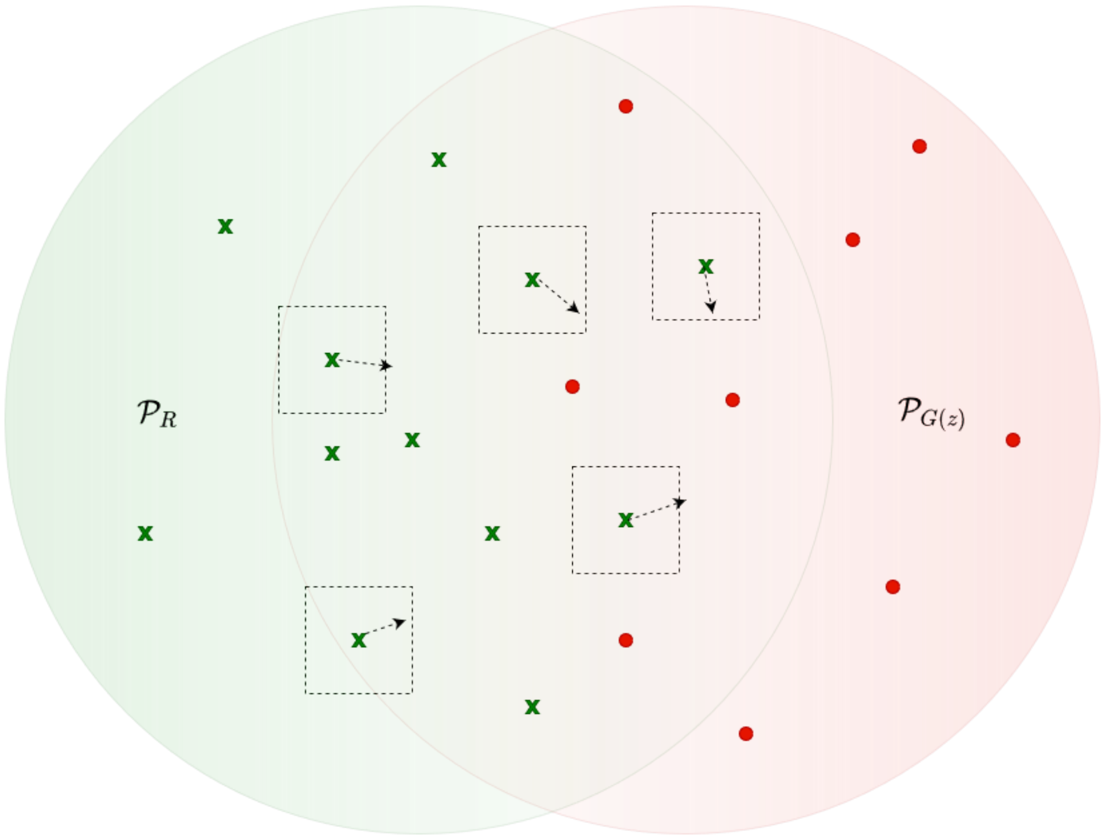

Figure 6: Visualization of how the input samples are perturbed during adversarial regularization.

Figure 6 exhibits how the critic's inputs are perturbed during adversarial regularization. The critic's objective in a Wasserstein GAN is to map the two input distributions to the output in a way which maximizes the Wasserstein distance, i.e. the difference between them. The adversarial samples are created by maximizing the loss of the critic with respect to its input samples, or put differently, minimize the Wasserstein distance in the critic's output. The adversarial samples are perturbed in the direction which maximize the critic's loss and this will conceivably be in the direction which makes $\mathcal{P}_R$ more similar to $\mathcal{P}_G$. In figure 6 the adversarial attack — turning benign samples into adversarial samples — is depicted as arrows along which the benign samples are perturbed, enclosed by the square-appearing infinity norm. The arrows point in the direction of the fake data distribution, $\mathcal{P}_G$, making it harder for the critic to discern between real and fake data — thereby having a regularizing effect on the GAN.

3.2.2 Regularization implementation

Adversarial regularization was enabled for each batch in every epoch of the training process. For every batch, a predefined number of samples were perturbed in an adversarial attack, creating adversarial samples. These adversarial samples replaced their benign counterparts in the batch, before the critic was trained on it. By replacing benign samples with adversarial counterparts, the total number of samples in the batch was kept identical to the original batch. All adversarial samples were discarded after use. The WGAN loss for the critic is stated as:

$$\max \quad \mathbb{E}_{x\sim\mathcal{P}_r}[C(x)] - \mathbb{E}_{\tilde{x}\sim P_g}[C(\tilde{x})] \tag{9}$$

where x are samples drawn from the real data distribution $\mathcal{P}_r$ and $\tilde{x}$ are samples drawn from the fake data distribution $\mathcal{P}_g$ Equation 9 when altered to encompass the effect of adversarial regularization can be rewritten as:

$$\max \quad \mathbb{E}_{x_{adv}\sim\mathcal{P}_r}[C(x_{adv})] - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g}[C(\tilde{x})]) + \lambda\mathbb{E}_{\hat{x}\sim\mathcal{P}_{\hat{x}}}[(\mid\mid \nabla_{\hat{x}}C(\hat{x}) \mid\mid_2 -1)^2] \tag{10}$$

where $\tilde{x}$ are the synthetic samples, x_{adv} is a mix of real and adversarially perturbed real samples, and the rightmost term is a gradient penalty[25], which is an auxiliary part of the critic's loss function. Noticeably absent in equation 10 is a scaling parameter for the adversarial training regularization. This additional parameter was left out since it was believed to be surplus to requirements. The regularization effect of the adversarial regularization implementation was instead controlled by two parameters: the size of the maximum aggregate perturbation, ϵ and the share of adversarial samples, i.e. the share of benign samples replaced by adversarial samples in each training batch. Algorithm 2 describes the CTGAN's training process with enabled adversarial regularization.

3.2.3 Choice of adversarial attack

The adversarial attack is the algorithm which is tasked with converting benign samples in a model's input into adversarial samples. There are a plethora of different attacks with different strengths and weaknesses. For this thesis, the Projected Gradient Descent (PGD) attack was chosen as the preferred mode of attack due to it being very effective and also easily compliant with constraints. Since PGD is a first order attack (i.e. it utilizes the Jacobian) it is very quick, which allows for not-so-cumbersome optimizations. It is also easy to constrain the attack, since the $\mathcal{L}_\infty$ can be enforced by simply clipping any feature value which exceeds the norm for each individual sample. The PGD attack was implemented with inspiration from Madry et al. [34][3]. The objective funtion of the optimization process instigated by the PGD is given by:

[3]https://github.com/MadryLab/mnist_challenge/blob/master/pgd_attack.py

$$\min \quad \mathbb{E}_{x_{adv}\sim\mathcal{P}_r}[C(x_{adv})] - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g}[C(\tilde{x})]) \tag{11}$$
$$\text{s.t.} \quad || \, x - x_{adv} \, ||_\infty \leq \epsilon \tag{12}$$

where equation 11 equates to minimizing the Wasserstein distance between the outputs of the real and fake data batches, and equation 12 convey the L_∞ constraints enforced on the PGD attack.

The PGD attack is not guaranteed to deliver successive higher-loss adversarial samples, due to the enforcement of the L_∞ constraints. The constraints are enforced by clipping, which simply entails that the value of a given sample is reverted back to the maximum or minimum if the PGD attack perturbs it beyond either of the extremes. The loss associated with the adversarial sample might therefore decrease if it is perturbed outside the L_∞ constraint and subsequently clipped. The PGD implementation utilized in this thesis curtails the possibility of the aforementioned problem by saving checkpoints during the optimization process, so that the best adversarial sample is retained.

Figure 7 displays a diagram over the flow of data during the training of the CTGAN when with enabled adversarial regularization. The PGD intercepts the benign batch and replaces a given share of the benign samples with corresponding adversarial samples.

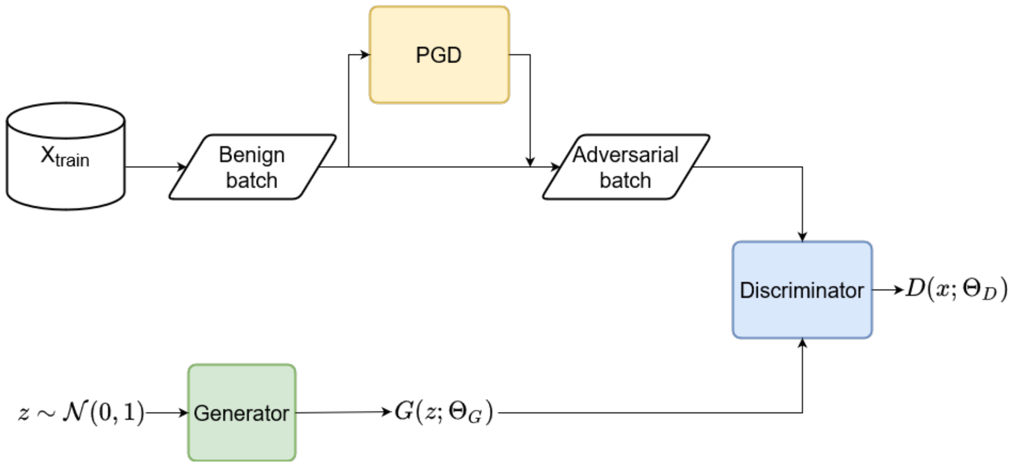

Figure 7: Diagram of GAN with adversarial regularization.

3.2.4 Adversarial regularization algorithm

Algorithm 2 summarizes the the functioning of the adversarial regularization described in the current chapter.

Algorithm 2 Adversarial training of CTGAN.

```
1: repeat
2:     Read minibatch $\mathcal{B}_{real} = \{X^1, ..., X^n\}$ from the training set
3:     Generate fake samples with the generator $\mathcal{B}_{fake} \leftarrow G(\boldsymbol{z})$
4:     $W_{loss} \leftarrow -(\frac{1}{N}\sum_{i=1}^{N}(C(\mathcal{B}_{real}) - \frac{1}{N}\sum_{i=1}^{N} C(\mathcal{B}_{fake})$
5:     $\mathcal{B}_{adv} \leftarrow PGD(\mathcal{B}_{real})$        ▷ Create the adversarial samples with the PGD attack
6:     Merge $\mathcal{B}_{real}$ and $\mathcal{B}_{adv}$ into $\mathcal{B}$ by randomly picking a share of the samples in $\mathcal{B}_{real}$ and
   replacing them with their adversarial counterparts
7:     $W_{loss} \leftarrow -(\frac{1}{N}\sum_{i=1}^{N}(C(\mathcal{B}) - \frac{1}{N}\sum_{i=1}^{N} C(\mathcal{B}_{fake})$        ▷ Calculate Wasserstein loss anew
8:     Backpropagate through the critic network
9:     Conduct an optimization step with $\boldsymbol{\nabla}_C W_{loss}$
10: until training completes
```

3.3 Adversarial regularization hyper parameters

In the PGD attack there were five parameters of importance: the step size, ϵ, the number of iterations in each attack, the share of adversarial samples and random start.

The combination of step size and number of attack iterations effectively decides the reach of each attack inside the L_∞ constraints. The combined choice of these parameters was motivated by a combination of a constrained time budget and the adversarial attack's proficiency. The choice of step size was the largest which would reliably increase the critic's loss and the number of attack iterations was capped at 50 iterations. The combined step size and maximum number of iterations did in most cases not allow the PGD to perturb the benign samples to within reach of the L_∞ constraint, but the lack of range was alleviated by random start — more about that later — and the fact that the input space generally is pockmarked with many local maxima with well-concentrated loss values[34].

The share of adversarial samples decide how large the share of benign samples in a given batch will be replaced with adversarial counterparts. Together with ϵ, i.e. the maximum perturbation size in terms of the L_∞ norm, they control the severity of the adversarial regularization effect . These parameters will be discussed in greater detail in section 4.4.

3.3.1 Random start

The principal motivation of conducting adversarial training, was to regularize the model as explained above, but there was also an auxiliary benefit brought about by the adversarial training, namely the implicit enlargement of the training set. Given a single benign data sample in the training set, the PGD will conceivably not produce exactly the same adversarial counterpart for every time it comes across this exact sample. This is the case because the optimization plane surrounding the sample will change from batch-to-batch and epoch-to-epoch as the critic is trained. In addition, when the critic is trained with adversarial samples, the adversarial regions will not stay put, since the critic's loss in these regions will be lowered.

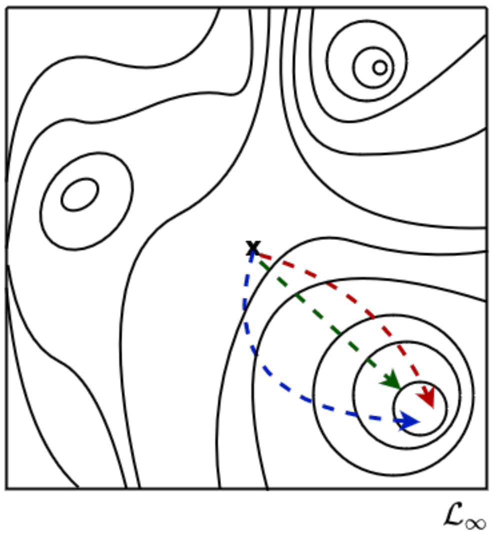

Figure 8: Random start disabled.

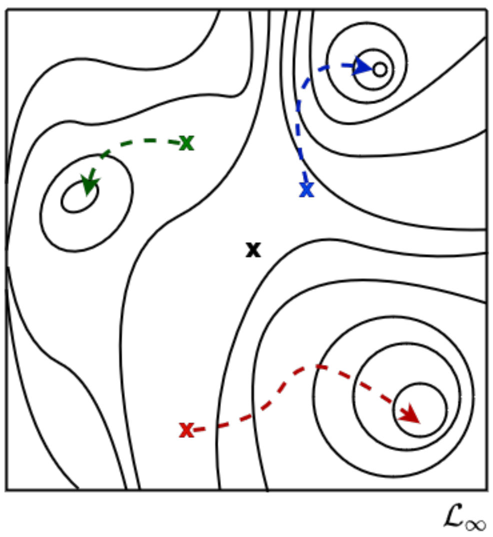

Figure 9: Random start enabled.

To facilitate for this effect, it is important to enable random start. Random start is a feature of the PGD attack which moves the optimization starting point to a random location within L_∞. Figures 8 and 9 visualize the difference between disabling and enabling random start. The arrows track hypothetical paths the optimization process might take across the optimization plane, when the same benign sample is perturbed in different epochs of the training process. With disabled random start, the benign sample is less likely to yield a varied set of adversarial samples across the training epochs, than if random start is enabled.

3.4 Membership inference attacks

Strong membership inference attacks were selected in order to thoroughly test the impact of adversarial regularization on model robustness. This entailed the adoption of white-box assumption, since white-box attacks are more proficient in attacking generative models than black-box attacks, although state-of-the-art black-box attacks generally perform as well as white-box attacks on regular classifiers[10].

The most direct recipient of the regularization effect provided by adversarial regularization is the critic, but the regularization will at the same time heavily influence the generator, since the two neural networks are interlocked in the would-be arms-race. Therefore it is necessary to assess the impact of adversarial regularization on robustness for both the critic and the generator.

3.4.1 Critic membership inference attack

The white-box MIA aimed at the critic emulates the attack designed by Hayes et al.[11]. This white-box attack is aimed at the critic and assumes that the whole trained neural network for some reason is available to the attacker, conceivably because of a lapse in security or that the model was publicly published in good faith. The access to the full model entails access to the supporting functionality as well, for instance the data transformer, sampler and functions for constructing the conditional vectors.

This attack is as simple as it gets, and directly leverages a victim model's overconfidence in its training samples, if any. The attacker simply passes all the samples which they want to query through the critic, and select the samples which the critic assigns the highest confidence to. In a WGAN the critic does not assign a confidence score between 0 and 1 to its predictions, as a normal classifier does, but due to the the critic's Wasserstein loss function (equation 9), it will tend to assign higher output values to real samples than fake ones.

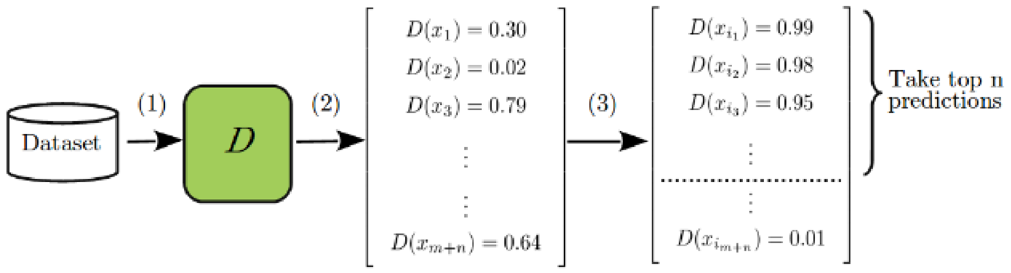

Figure 10: White-box critic MIA diagram.

Source: [11]

3.4.2 Generator membership inference attack

Chen et al. designed a membership inference attack which leverages a GAN's generator in a white-box setting[10], which made an excellent choice for the generator attack in this thesis. Chen et al. stated that an optimal attacker will compute $P(m_i = 1 \mid x_i, \Theta_v)$, i.e. the probability of a given sample, x_i being classified as belonging to the victim model's training set ($m_i = 1$) given a victim generative model, G_v with the parameters Θ_v. They hypothesize that $P(m_i = 1 \mid x_i, \Theta_v) \propto P_{G_v}(x \mid \Theta_v)$, i.e. the probability of a given x_i belonging to the victim's training set is proportional to the probability of the victim generator having produced it. This assumption is the cornerstone of their attack. To calculate $P_{G_v}(x \mid \Theta_v)$ is intractable so they approximate it with:

$$P_{G_v}(x \mid \Theta_v) \approx \frac{1}{k}\sum_{i=1}^{k} exp(-L(x, G_v(z_i))) \tag{13}$$

where z_i is the input to the generator. Equation 13 entails that the attacker approximates $P_{G_v}(x \mid \Theta_v)$ by the error between x and a reconstruction, $\mathcal{R}$ of x, $G_v(z_i)$.

$$\mathcal{R}(x \mid \mathcal{G}_v) = \mathcal{G}_v(z^*) \tag{14}$$
$$\tag{15}$$

The reconstruction process is conducted by optimizing the input z to the victim generator, G_v:

$$z^* = \min_z L(x, \mathcal{G}_v(z)) \tag{16}$$

The loss function which governs the optimization of z, is given by equation 17:

$$L(x, \mathcal{G}_v(z)) = \lambda_1 L_2(x, \mathcal{G}_v(z)) + \lambda_2 L_{reg}(z) \tag{17}$$
$$L_{reg}(z) = (\|\, z \,\|_2^2 - dim(z))^2 \tag{18}$$

where L_2 is the euclidean distance and L_{reg} is a regularization term penalizes zs which are far from the prior.

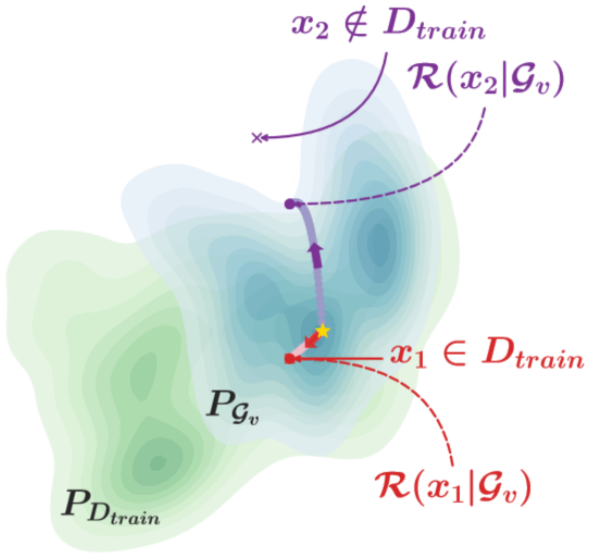

Figure 11: White-box generator MIA diagram.

Source: [10].

Figure 11 exhibits visually how the attack infers which of samples x_1 and x_2 were part of the victim model's training set, D_{train}. The recreation of x_1, $\mathcal{R}(x_1 \mid \mathcal{G}_\sqsubseteq)$, should have a shorter distance to x_1 on $P_{\mathcal{G}_\sqsubseteq}$, than the distance between x_2 and its recreation $\mathcal{R}(x_2 \mid \mathcal{G}_\sqsubseteq)$.

The generator MIAs performed as a part of this thesis, were performed without the regularizing term, due to the attack performance improving with it disabled. As Chen et al. proposed in their article, z was initialized as a zero vector. The hyper parameters employed in the generator MIAs are displayed in 7 in the appendix.

3.5 On evaluating the effect of adversarial regularization

There were three aspects of the CTGAN's performance with which it was prudent to evaluate the impact of adversarial regularization.

1. The impact on robustness to membership inference attacks.
2. The impact on the synthetic data quality, i.e. the likeness between the synthetic and real data distributions.
3. The impact on machine learning efficacy, i.e. how well a classifier trained on synthetic data produced by the CTGAN will perform on data it has not seen before.

The ROCAUC metric was utilized in order to assess the effect of adversarial regularization on the CTGAN's robustness to membership inference attacks. One problem a membership inference attacker has to solve is where to set the threshold for which samples to consider as originating from the victim model's training set or not. When evaluating model robustness, the ROCAUC metric curtails this challenge and instead returns a score which summarizes the attacker's success across all possible thresholds.

For assessing the adversarial regularization's impact on the quality of the synthetic data produced by the CTGAN, a collection of metrics were selected, which could capture the nuances of the resulting synthetic data. They include Wasserstein distance, root mean square error (RMSE), the two-sample Kolmogorov-Smirnov test and the number of fake pairwise unique samples.

For evaluating the machine learning utility of the synthetic data, three classifiers were trained on the synthetic datasets produced by the CTGAN and tested on hold-out test sets drawn from each dataset. The set of classifiers consisted of an AdaBoost classifier[38], a random forest classifier and a logistic regression model, which performances were assessed with the use of the metrics: accuracy, ROCAUC and F1 score.

3.6 Experiments

Among the experiments there were three parallel sets of models which were trained. They were:

- The first model type, referred to as "regular", was trained without dropout or adversarial regularization.
- The second model type was trained with adversarial regularization, with varying combinations of ϵ and share of adversarial samples.
- The final model type was trained with dropout.

Due to the CTGAN being a WGAN and the regularization effect provided by the CTGAN's inherent gradient penalty, there were no problems with training the regular model without any extra regularization. Additionally, the generator was regularized with batch normalization during all experiments. Therefore, even if neither dropout or adversarial regularization was enabled during the training of the CTGAN, there were still active regularizing effects provided by the gradient penalty and batch normalization, which steadied the training process.

The regular model was employed as a benchmark with which to compare the regularization effects of dropout and adversarial regularization, as the regularization parameters of the latter models were varied.

3.6.1 Datasets

The effect of adversarial regularization on the CTGAN, and its comparison with the regular and dropout models were tested with a varied set of four datasets. The datasets varied in width, number of categorical and numerical columns and in number of labels. They all represented classification problems. Table 1 includes all the specifics of the training sets utilized in this thesis. Each training set was accompanied with a test set of similar size, which was used for assessing the machine learning utility of the synthetic data produced by the CTGAN models. Four additional *query* datasets were fashioned for the membership inference attacks. They were comprised of a total of 1000 samples; 500 sampled from each training set, and 500 from each test set, concocting 50-50 mixes.

Dataset name	# categorical columns	# numerical columns	# labels	# samples
Covertype [39] [4]	2	11	10	10 000
UAV intrusion detection[40][5]	0	55	2	10 000
Adult[39][6]	15	6	2	10 000
Musk[39][7]	0	168	2	1000

Table 1: Training sets.

4 Results and discussion

In this section the results of the experiments are presented and discussed. The general causes of overfitting and thereby vulnerability to membership inference attacks is the first talking-point of this section. Subsequently, a number of observations regarding the use of adversarial regularization will be presented. Lastly the effect of adversarial regularization on model robustness, machine learning utility and synthetic data quality will be compared with the same effects of dropout regularization.

4.1 Impact of overfitting on model robustness

In order to assess how robust a given GAN model is to membership inference attacks, it is important to have a clear understanding of the primary drivers of overfitting.

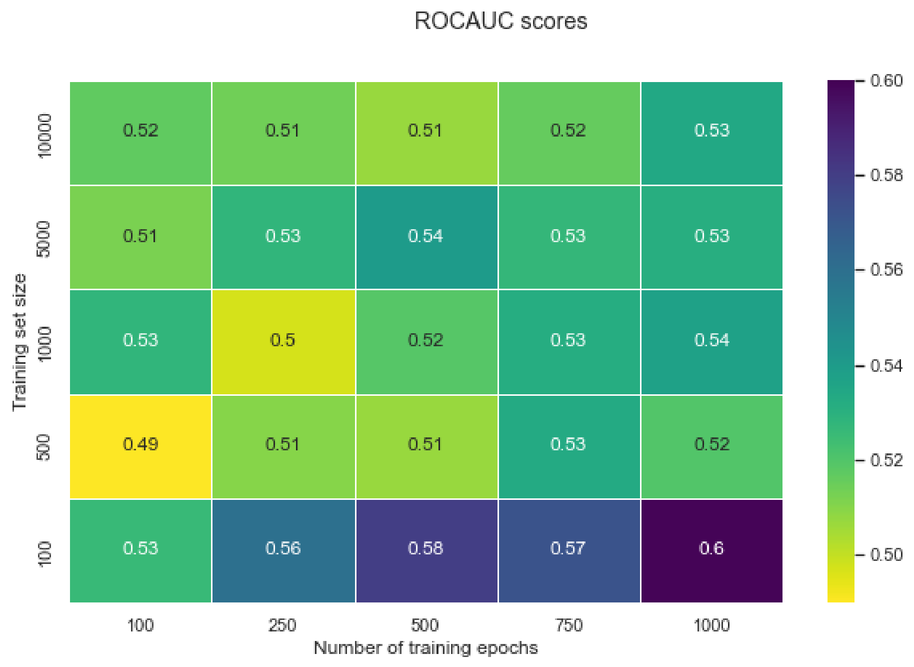

Figure 12: Caption

Figure 12 displays the effect of increased training samples and number of training epochs on the CTGAN's vulnerability to MIAs for the *UAV intrusion detection* dataset. The heat map shows how robustness to MIAs is related to overfitting. The degree to which the model was overfitted was controlled with the size of the training set and the number of training epochs. The size of the training set was the main driver of overfitting, although it was only significant when there were 100 samples in the training set, which is very few. For smaller datasets, a GAN model is more prone to memorizing the training set, rather than generalizing from it, and is therefore more susceptible to MIAs. Secondary to the size of the training set, with respect to its effect on overfitting, is the number of training samples. For the four largest datasets there were slight positive correlations between the ROCUAC and the number of training epochs, while the smallest dataset showed a marked increase in ROCAUC — from 0.53 after 100 epochs to 0.60 after 500 epochs. In summary, dataset size is the primary reason for overfitting, while the effect of training length is only significant when the training set size is sufficiently small.

As will later be made clear, the success rate of MIAs is heavily related to the width of the dataset. For a wider dataset than UAV intrusion detection, i.e. a dataset with more columns, overfitting would be apparent for much larger datasets and comparably fewer training epochs.

4.2 Observations from adversarial regularization

From the experiments conducted in this thesis a collection of observations were extracted. They will be summarily described in this section. They were:

1. Adversarial regularization increased the critic's robustness to MIAs.
2. The critic's vulnerability to MIAs increased with the number of training epochs, but not linearly.
3. The susceptibility to MIAs were closely linked with their training losses.
4. The generator benefited much less from adversarial regularization than the critic, despite its training being seemingly more affected by it.

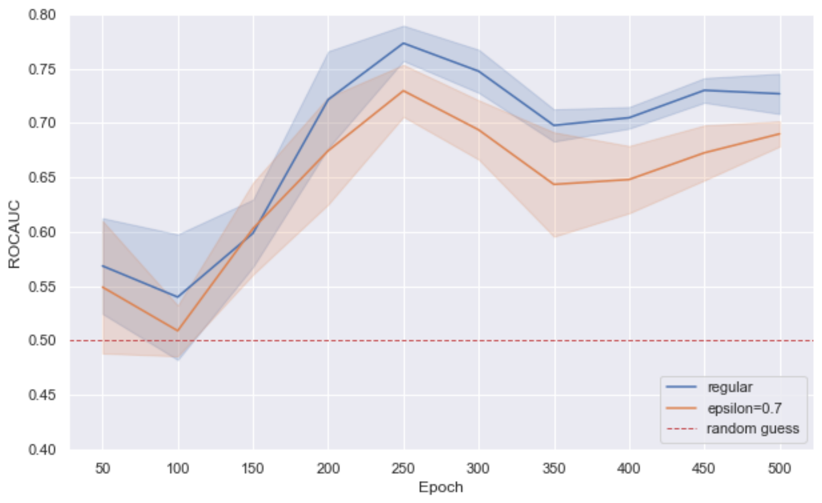

Figure 13: ROCAUC graphs for a regular and adversarially regularized model attacked in a critic MIA.

Figure 13 highlights the first lesson, namely that adversarial regularization increases the model's robustness to MIAs. It displays the success rate of the membership inference attacker as ROCAUC, at different stages of the training process for two CTGAN models: one version which has been regularized with adversarial training and another which has not been regularized neither adversarial regularization nor dropout. The lines are the mean ROCAUCs for the attacks conducted against the models and the shaded areas enclosing the lines are one standard deviation. The figure shows how the adversarial training regularized model's mean robustness is consistently lower than that of the regular model for any number of training epochs. Although adversarial regularization improves the robustness of the CTGAN, it is by no means a perfect solution. A perfect robustness outcome would entail the attacker purely guessing whether a given sample was part of the training set or not, i.e. ROCAUC = 0.5.

One might be forgiven in believing that the model's vulnerability to membership inference attacks would increase linearly or at least monotonously with the number of training epochs, vulnerability to MIAs is correlated with the length of the training process, as was established in the previous section. Figure 13 disproves this notion. In fact, for both models the attack's ROCAUC decreased until 100 epochs, before it rapidly overshot the level where it would eventually converge after 300 epochs. The answer to this problem lies in the training of the critic.

Figure 14 displays the training losses of the critic and generator, corresponding to the models attacked in figure 13. The trough and peak in the critic's training loss at epochs 100 and 150

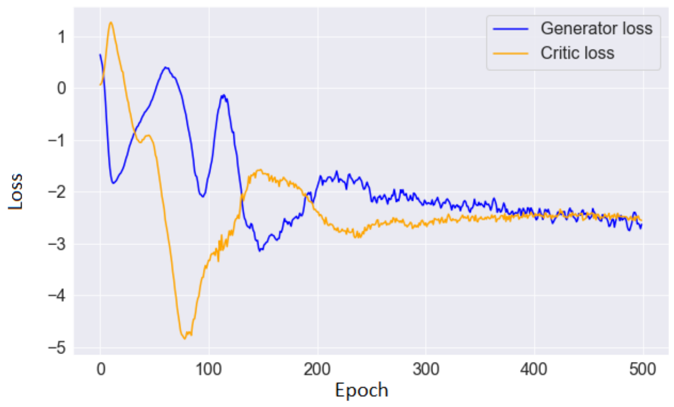

Figure 14: Training loss corresponding to figure 13.

coincide with the attack's smallest and highest ROCAUC scores. These were also moments in the training of the CTGAN where the strength of the critic relative to the generator shifted greatly. In the trough before epoch 100 the critic's loss decreased steeply, which indicates that the critic had strengthened itself in comparison with the generator, i.e. it has evolved network parameters which allowed it to more easily tell real from fake data. Immediately after finding this new set of network parameters, the critic was less overfitted to the training set because it had not had the time to hone its parameters in this new "paradigm" which was just brought about. After epoch 100 the loss of the critic stabilized somewhat and the ROCAUC of the attack increased, since the critic got to progressively improve its parameters and thereby overfit. This example illustrates how the critic's vulnerability to MIAs is intertwined with its training process.

The training of a GAN is a two-player game where the critic and generator are tightly interdependent. The regularization of the critic should therefore also affect the generator. This appears at first glance to be the case. As figures 15 and 16 show, the generator's losses are greatly affected by adversarial regularization, even more so than the critic is.

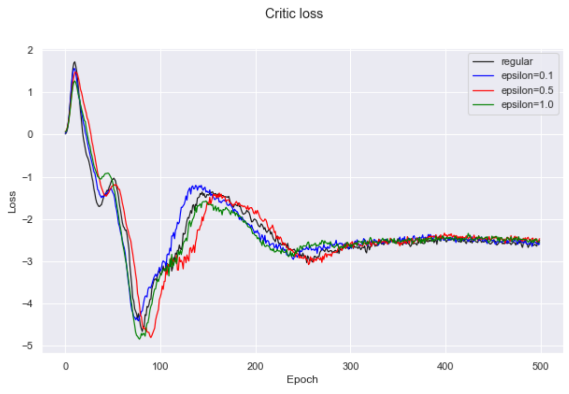

Figure 15: Critic training loss for varying ϵ.

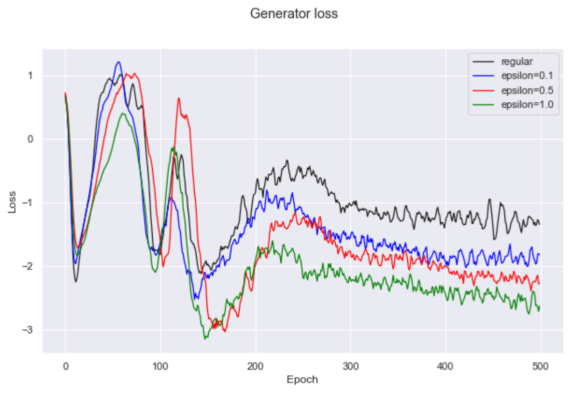

Figure 16: Generator training loss for varying ϵ.

In contrast with the critic, the effect of adversarial regularization did not translate to significantly increased robustness to membership inference attacks for the generator, as can bee seen in figure 17. For $\epsilon = 1.0$, which is the strictest amount of adversarial regularization considered in this thesis, the attacker attained roughly the same ROCAUC as in the attack of the regular model. What seems clear is that the adversarial regularization which had significant effect on the critic's robustness did not translate to the generator in a similar manner. The adversarial regularization did however affect the synthetic data produced by the generator considerably, as will be discussed in subsequent sections.

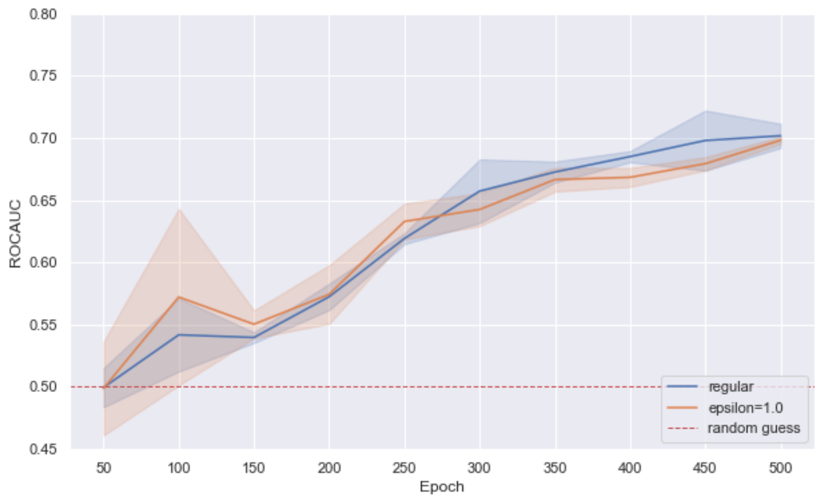

Figure 17: ROCAUC graphs for a regular and adversarially regularized model attacked in a generator MIA.

4.3 Adversarial regularization's effect on synthetic data quality

For the discussion of how the adversarial regularization affects the quality of the synthetic data produced by the CTGAN, the results obtained from the Covertype dataset are again utilized. Covertype is arguably the most difficult dataset to emulate of the four considered in this thesis. It consists of a mix of numerical and categorical columns, with a total of 44 individual categorical values, and ten labels.How should one regard the effects on the synthetic data quality with respect to the adversarial regularization? The generator does not have access to the target data distribution, i.e. the training set. Access to information about the target distribution is mediated by the critic. The generator will instead improve itself solely based on what increases its ability to deceive the critic. When the critic is trained on adversarial samples, it will regard them as originating from the target distribution. If ϵ is too large and the adversarial samples differ considerably from the target distribution, the generator will, through the critic, learn to generate aberrant samples, thereby deteriorating the synthetic data quality.

Figures 18, 19, 20, and 21 exhibit how adversarial regularization affects the quality of synthetic data produced by the CTGAN at different stages of the training process, with respect to the metrics which were determined to of interest in section 3.5. The increasing value of ϵ represents the increasing severity of the adversarial regularization. The plots show how moderate adversarial regularization ($\epsilon \in (0.1, 0.4)$) can be implemented without a significant penalty to the synthetic data quality for relatively small values of ϵ. However, excessive adversarial regularization was not beneficial to the synthetic data quality, similarly to what one would have expected of alternative regularization methods too. The different aspects of the synthetic data quality, i.e. RMSE, Wasserstein distance, the Kolmogorov-Smirnov (KS) statistic and the number of fake pairwise unique combinations, are affected somewhat differently by the adversarial regularization. Figure exhibits the RMSE between the synthetic and real data 19. The RMSE decreased unequivocally with increasing ϵ, i.e. the severity of the adversarial regularization. The same was not true for the Wasserstein distance and the KS statistic. Moderate values of $\epsilon(\in (0.1, 0.4))$ had a beneficial and lasting effect on the Wasserstein distance and KS statistic, while larger ϵs contributed to a sharp initial decrease in both metrics, which bottomed around 150-200 epochs, before they rose to as bad levels as at the beginning of the training. This suggests that adversarial regularization mostly was beneficial in the initial phases of the GAN training, regardless of the size of ϵ, but not necessarily if enabled throughout the entire training process. Adversarial regularization proved to be somewhat advantageous to the number of fake pairwise unique combinations, putting a slight downward pressure on the total number and bringing it closer to the 71 real pairwise unique combinations found in the training set.

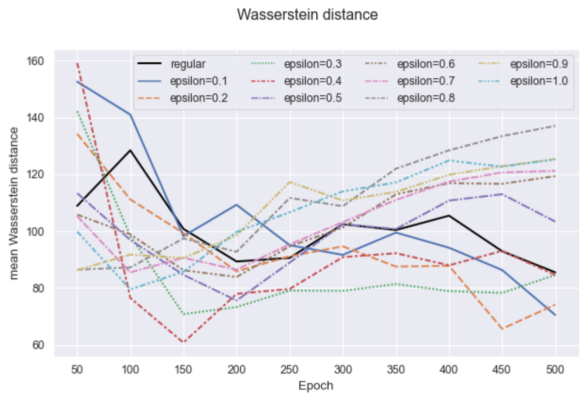

Figure 18: Wasserstein distance during the training process for varying ϵ.

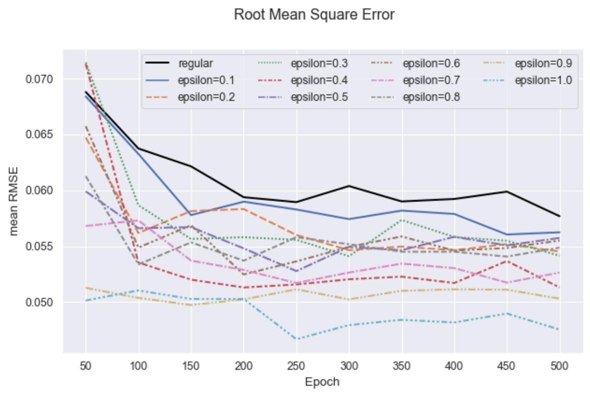

Figure 19: RMSE during the training process for varying ϵ.

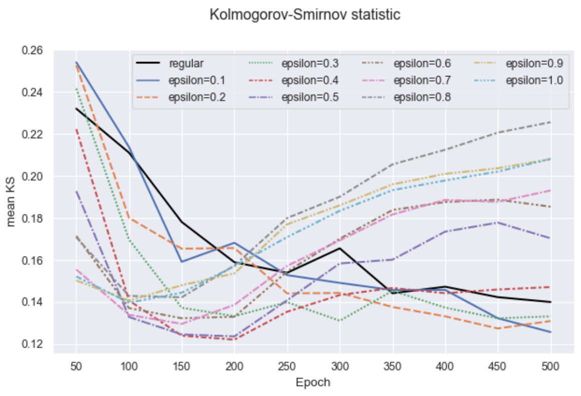

Figure 20: Kolmogorov-Smirnov statistc during the training process for varying ϵ..

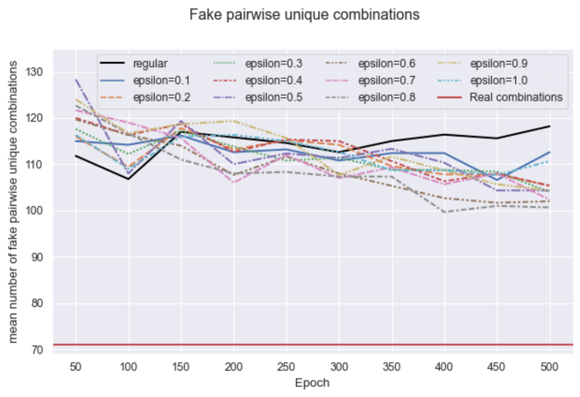

Figure 21: Number of fake pairwise unique combinations during the training process for varying ϵ.

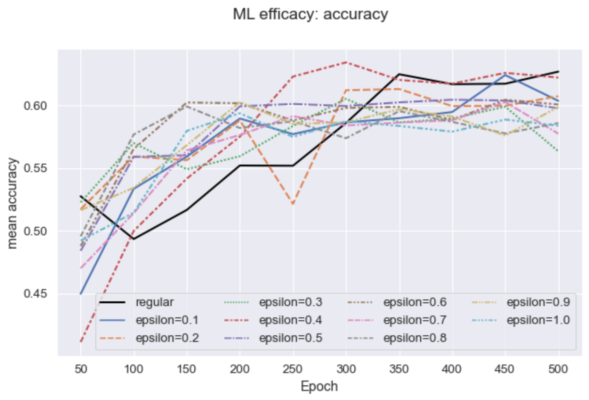

Figure 22: Accuracy at several stages of the training process with varying ϵ.

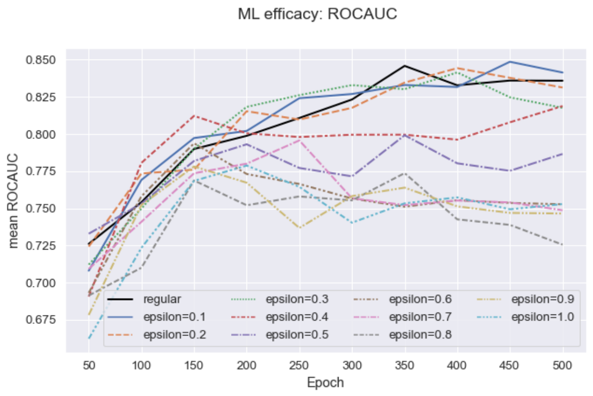

Figure 23: ROCAUC at several stages of the training process with varying ϵ..

Similarly to the synthetic data quality metrics, the machine learning efficacy metrics also deteriorated with excessive adversarial regularization. For $\epsilon \in (0.1, 0.3)$ adversarial regularization did not incur great losses in utility, at least in terms of accuracy and ROCAUC. Accuracy is the most accommodating metric, because it is just the ratio between the number of correct predictions and the total number of predictions, without paying heed to how the errors are distributed between true positives and negatives, and false positive and negatives. The F1 score is the most punishing

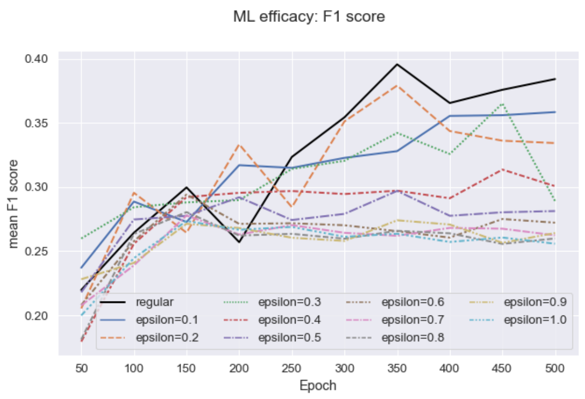

Figure 24: F1 score at several stages of the training process with varying ϵ.

of the three metrics and figure 24 shows a clear reduction in F1 score as ϵ increases. The reduction entails that larger proportions of the classifiers' predictions are rendered false positives or false negatives as ϵ increased.

4.4 To which parameters of the adversarial attack is adversarial regularization most sensitive?

The effect of the adversarial regularization was controlled with the help of two hyper parameters: ϵ and the share of adversarial samples in a given training set batch. In order to analyze the effects of these two hyper parameters on the regularization effect, the Musk dataset again took center stage, since it lead to the model most vulnerable to MIAs.

4.4.1 Results from varying epsilon

Increasing ϵ entails loosening of the constraints limiting the adversarial attacks in the input space. With looser constraints, each benign sample in the training set can give spawn to a more varied set of corresponding adversarial samples across the CTGAN's training process. Previously it has been stated that ϵ correlated positively with the critic's robustness to membership inference attacks, but an excessive ϵ will deteriorate the quality of the synthetic data produced by the generator.

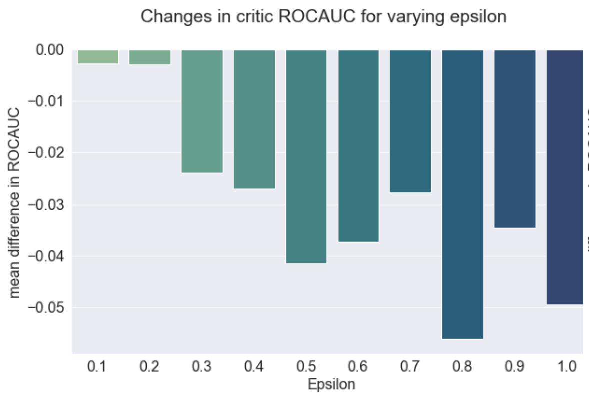

Figure 25: The critic's susceptibility to MIA for varying values ϵ.

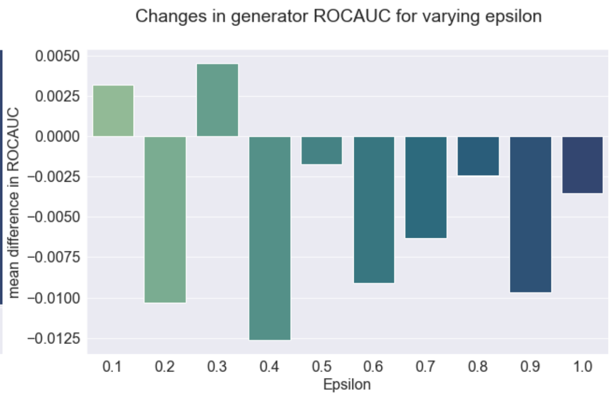

Figure 26: The generator's susceptibility to MIA for varying values ϵ.

The critic's robustness to membership inference attacks correlated positively with ϵ, as can be seen in figure 25, which plots the reduction in MIA ROCAUC for the critic compared to the regularly trained model. The same did not apply for the generator. Figure 26 shows that adversarial regularization tends to reduce the attacker's ROCAUC against the generator, although there is no clear correlation between the reductions and ϵ. Additionally, the reductions are much smaller in scale compared with adversarial regularization on the critic, which reiterates the results presented in figures 13 and 17.

Hernandez-Garcia et al. distinguish between implicit and explicit regularization methods, where the explicit methods decrease the *representational capacity* of the model[41]. The implicit methods on the other hand, do not reduce representational capacity, but rather *effective capacity*. Regularization methods like weight decay, which directly limit the model weights, fall into the first category, while adversarial regularization falls into the second. Therefore, it is important to beware that the regularization effect is not necessarily perfectly correlated with the size of the perturbation constraint ϵ. The regularization effect will not necessarily increase monotonously with the size of ϵ, i.e. the accumulated weights in the CTGAN layer will not necessarily decrease with a corresponding increase in ϵ. This feature is inherent to adversarial regularization, and is an outcome of the optimization process in the adversarial attacks. For a given input sample, the adversarial attack searches for regions of high loss surrounding the sample. Loosening the constraints of the adversarial attack by increasing ϵ does not guarantee that the attack will converge in a region with higher loss than it would have in for a lower ϵ.

4.4.2 Varying the share of adversarial samples

The hyper parameter "share of adversarial samples" relates to the share of benign samples which are replaced by adversarial sample counterparts in each training batch during the CTGAN training process.

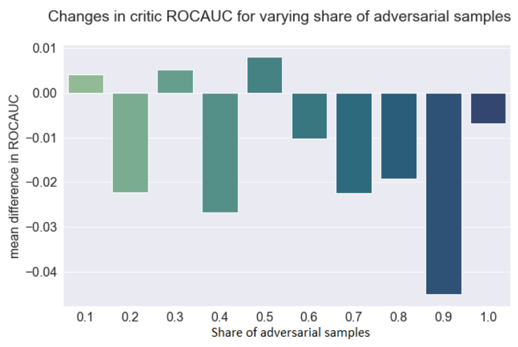

Figure 27: The critic's susceptibility to MIA for varying shares of adversarial samples.

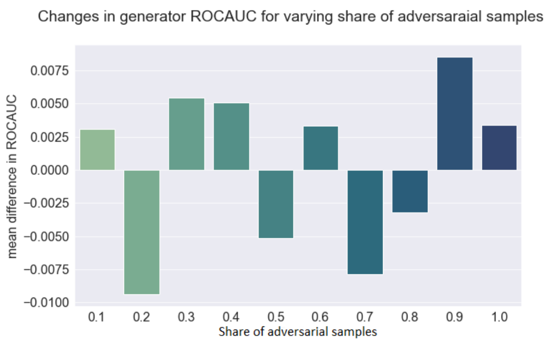

Figure 28: The generator's susceptibility to MIA for varying share of adversarial samples.

Increasing the share of adversarial samples in each training batch, was not as decisive for the CTGAN's robustness as increasing ϵ was. Figures 27 and 28 display the mean reduction in ROCAUC attacks against the critic and generator for increasing share of adversarial samples. The correlation between increased share and decrease in attack ROCAUC was much less pronounced than the correlation between ϵ and ROCAUC, and it was not even discernible in the generator's case.

Figures 29, 30, 31 and 32 exhibit how the synthetic data quality metrics change when the share of adversarial samples included in each training batch varies, for a fixed ϵ. The figures show that the synthetic data quality is affected similarly by increasing the share of adversarial samples, as they were when ϵ was increased.

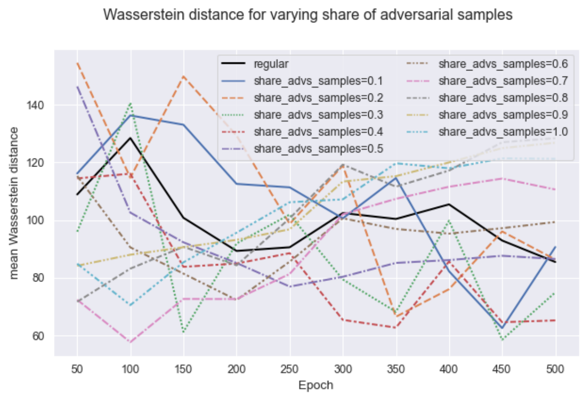

Figure 29: Wasserstein distance at several stages of the training process with varying share of adversarial samples.

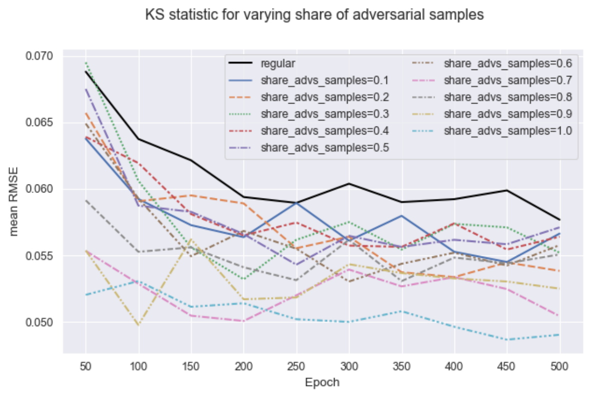

Figure 30: RMSE at several stages of the training process with varying share of adversarial samples.

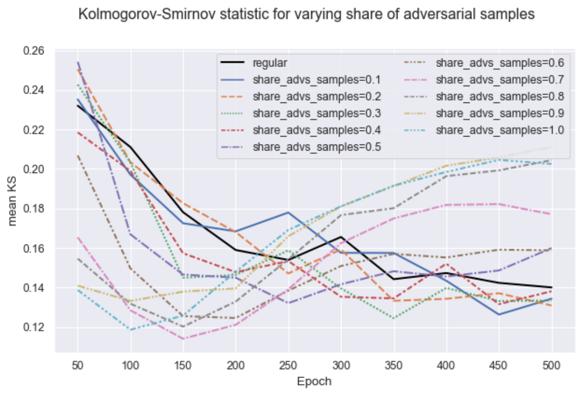

Figure 31: Kolmogorov-Smirnov statistc at several stages of the training process with varying share of adversarial samples.

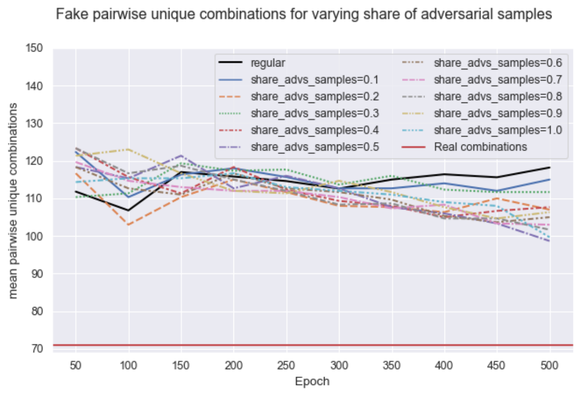

Figure 32: Number of fake pairwise unique combinations at several stages of the training process with varying share of adversarial samples

In summary, the share of adversarial samples seems to be the parameter of secondary importance compared to the size of the constraints on the adversarial attacks, ϵ. The two parameters have the same effect on the synthetic data quality, but ϵ is associated with higher robustness against membership inference attacks. The reason for this disparity between the effects on the CTGAN's robustness may be found by examining the parts each parameter plays in adversarial regularization. Loosening the constraints on the total perturbation size in the adversarial attack, i.e. increasing ϵ, will include more local maxima in the vicinity of each benign sample. This entails a higher number of high-loss regions in the input space whereto benign samples can be perturbed. The regularization effect of adversarial regularization is grounded in finding regions in the input space in which there is high loss, so it makes sense that increasing ϵ will increase the severity of the regularization effect. The share of adversarial samples does not limit which benign samples are used for adversarial regularization, but rather how many are used at the same time. Which benign samples are to be replaced with adversarial counterparts are picked randomly for each batch. Over the course of the entire training process, all samples from the training set will likely be replaced with a number of adversarial counterparts. The difference between a low and a high share of adversarial samples is therefore not fundamentally important for the prospect of adversarial regularization.

4.5 Adversarial regularization compared with dropout

Dropout is a regularization method with which it was natural to compare adversarial regularization. Due to its ease of use and benefits on both of the GANs utility with respect to synthetic data quality and beneficial effect on model robustness[8], it is a popular choice. Dropout is also the inherent regularization method in the CTGAN model, making it even more relevant as a comparison to adversarial regularization.

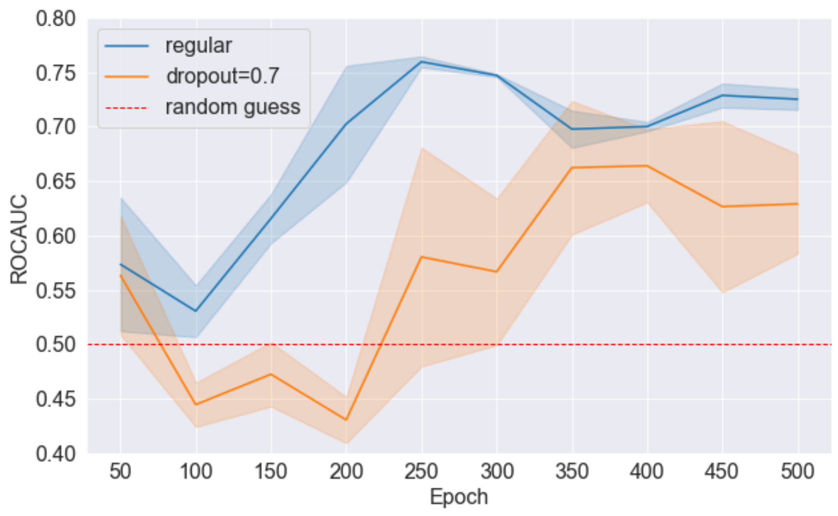

Figure 33: ROCAUC graphs for a regular and a dropout-trained model attacked in a critic MIA.

Regularizing the critic with dropout resulted in an increased robustness to MIAs, as shown in figure 33. Dropout proved to be a more powerful regularization method than adversarial regularization with respect to increasing the robustness to MIAs. As figure 33 shows, the ROCAUC of the MIA against the critic trained with dropout, was at most 0.25 less than the equivalent figure for the critic trained without dropout.

In contrast with adversarial regularization, dropout was much more effective at increasing the generator's robustness. Figure 34 exhibits how dropout= 0.5 decreased the attacker's ROCAUC with at most 0.08 around epoch 350, and generally tracks well below the ROCAUC graph of the unregularized model's generator.

There are several key differences separating dropout and adversarial regularization which might explain the discrepancy between dropout and adversarial regularization's effect on generator robustness. Firstly, the former is an explicit form of regularization and affects the critic directly, while the latter — as aforementioned — is an implicit form of regularization and only affects the input data from which the critic learns. Another key difference between dropout and adversarial regularization, is that the latter only affects the numerical input features, while dropout regularizes the network regardless of what type of input the network is presented with. This is however not the cause of the discrepancy, since the largest difference in the regularization methods' ability to reduce vulnerability to MIAs were observed with the Musk dataset, which only contained numerical columns. The conclusion might be that dropout, with its direct impact on the network parameters, is a more powerful method of regularization than adversarial regularization, and therefore has a better ability to reduce ovefitting and thereby vulnerability to MIAs for both the critic and the generator. This conclusion is corroborated by figures 35, 36, 37 and 38, which show that increases in dropout reduced ROCAUC of the MIAs against both the critic and generator much more than adversarial regularization was able to.

Compared with dropout, adversarial regularization proved to be less effective for all of the datasets experimented with in this thesis. The bar plots displayed in figures 39 and 40 exhibit the attacker's ROCAUC score against the unregularized model, the model trained with adversarial

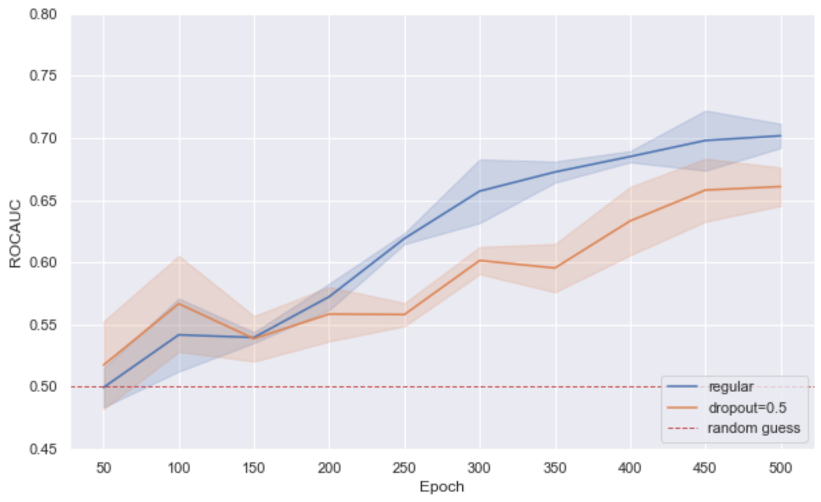

Figure 34: ROCAUC graphs for a regular and a dropout-trained model attacked in a generator MIA.

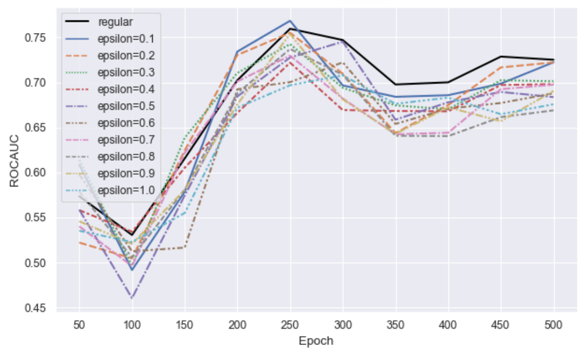

Figure 35: ROCAUC graphs for critic MIAs with varying ϵ.

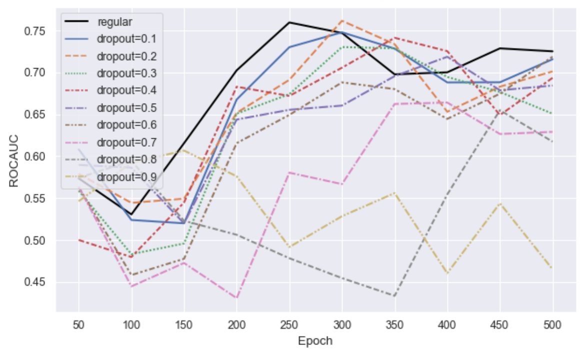

Figure 36: ROCAUC graphs for critic MIAs with varying dropout.

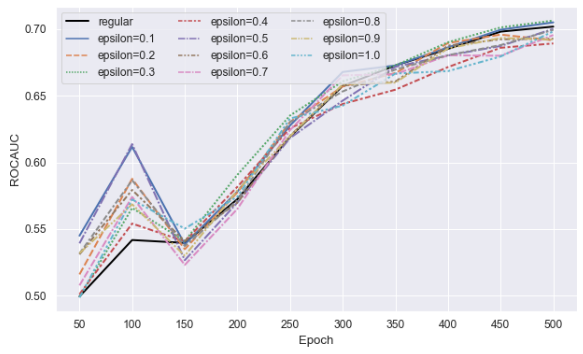

Figure 37: ROCAUC graphs for generator MIAs with varying ϵ.

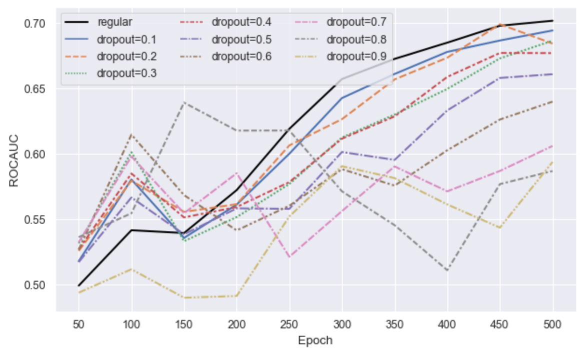

Figure 38: ROCAUC graphs for generator MIAs with varying dropout.

regularization, and the model trained with dropout for all the datasets, are compared. The bars corresponding to the regularization methods convey the smallest ROCAUC the MIAs attained against the regularized models, while the bar corresponding to the regular model is the maximum

ROCAUC attained by the attacker against the regular model. Thus the bar plot conveys the regularization methods' maximum ability. The bar plots show how dropout outcompetes adversarial regularization in all but one case. The single case in which adversarial regularization performed better than dropout, namely for the adult test set in the generator MIA, the difference was marginal and both ROCAUC scores were approximately 0.5 which indicates that the attacker would have had no ability to identify training samples for either model anyway.

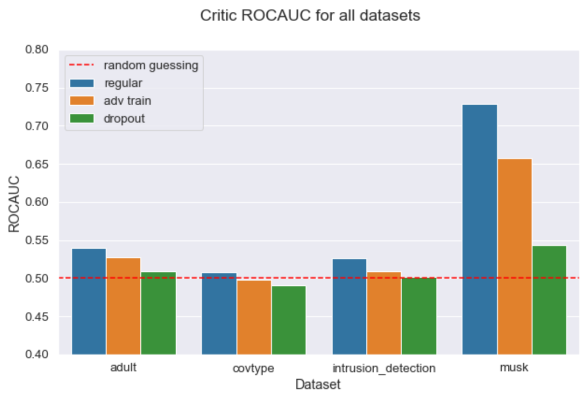

Figure 39: Critic MIA ROCAUC barplot for all datasets.

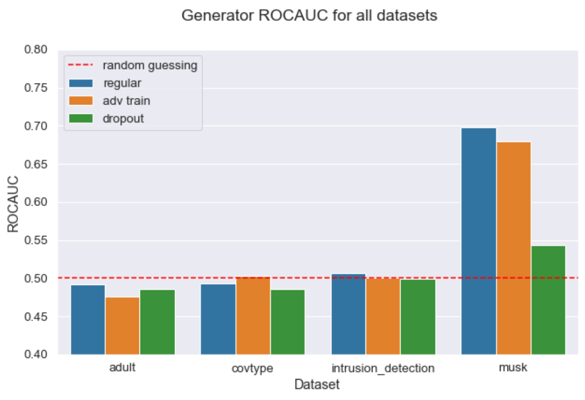

Figure 40: Generator MIA ROCAUC barplot for all datasets.

The reduction in ROCAUC facilitated by adversarial regularization relative to dropout in figures 39 and 40, show that dropout was more adept at translating the regularization effect from the critic to the generator, than adversarial regularization was. Dropout facilitated almost the same drop in ROCAUC for both the critic and the generator, while adversarial regularization's effect reduced markedly when translated to the generator. The inability of adversarial regularization to significantly lower the attackers ROCAUC against the generator, is a bad predicament because it is the generator's defence which it is most important to bolster. The critic is often discarded when a GAN is made publicly available, since only the generator is necessary for the production of synthetic data. The generator is therefore more prone to attacks. This is the main Achilles heel of the adversarial regularization implementation presented in this thesis.

Figures 41 42, 43 and 44 display how the synthetic data quality produced by the CTGAN was affected by dropout. The effect was much less discernable than the equivalent effect on the synthetic data quality of adversarial regularization. Dropout and its ability to decrease the critic's representation capacity, thus has a lesser effect on the synthetic data produced by the generator than adversarial regularization.

In a real world utilization of the CTGAN, one might be more concerned with the utility of the synthetic data, rather than solely the privacy guarantees which comes along with it or the synthetic data quality metrics, which are just an indication of the data's usefulness. If the choice of regularization method is to be mainly informed by which model provides the best utility, the comparison between dropout and regularization falls out differently. Tables 3, 4 and 5 encompass all metrics considered for assessing the utility, robustness and synthetic data quality respectively, for the four datasets included in this thesis. The results are color coded in traffic-light style: red indicates the model which performed worst for the given metric, while yellow and green indicates the second-to-best and best models respectively.

As can be seen in table 3 Adversarial regularization generally outcompeted dropout in terms of utility in the datasets which only contain numerical columns, i.e. the UAV intrusion detection and Musk datasets. In the mixed datasets (Adult and Covertype) either the regular model or the dropout-regularized models performed best. The reason for this discrepancy, is conceivably due to the adversarial regularization not affecting the categorical columns in the mixed datasets, while the numerical datasets allowed attacks on all columns.

The same message is reflected in tables 4 and 5; adversarial regularization outperformed dropout for the numerical datasets. Remember, figures 39 and 40 show the regularization methods' maximum

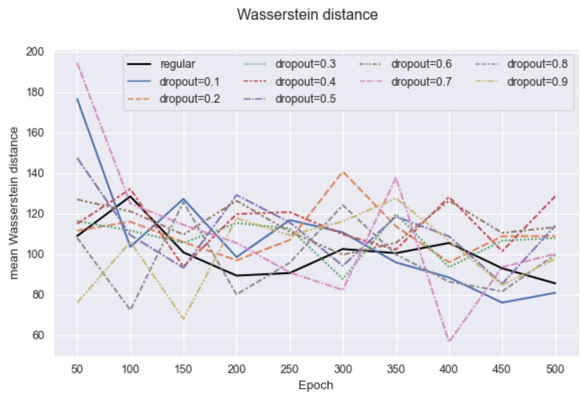

Figure 41: Wasserstein distance at several stages of the training process with varying dropout.

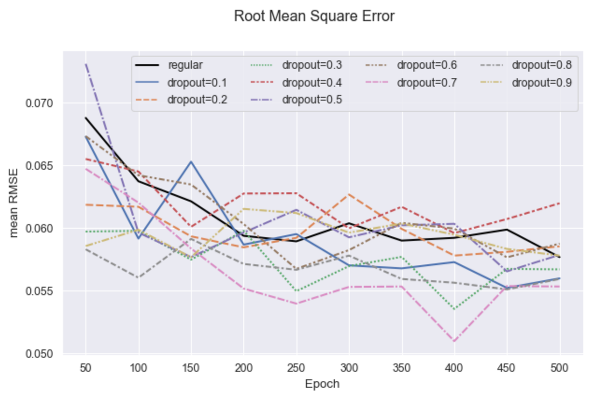

Figure 42: RMSE at several stages of the training process with varying dropout.

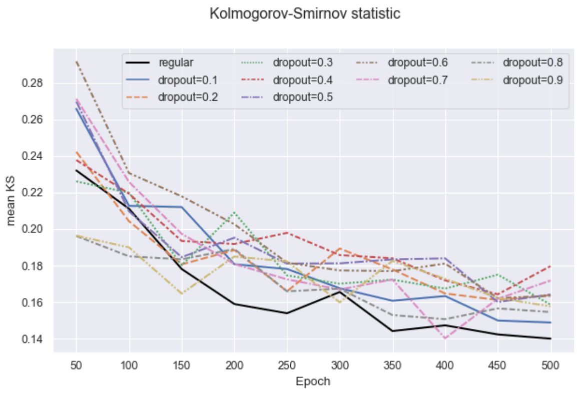

Figure 43: Kolmogorov-Smirnov statistc at several stages of the training process with varying dropout.

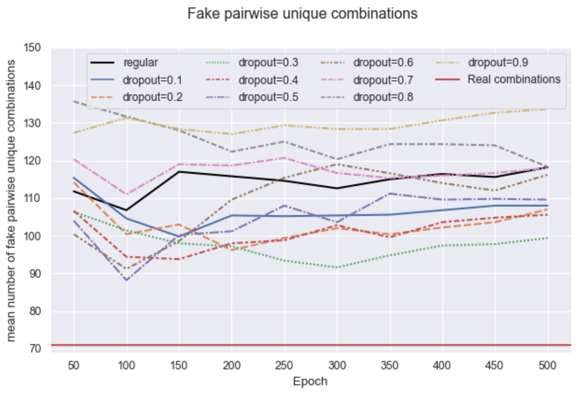

Figure 44: Fake pairwise unique combinations at several stages of the training process with varying dropout

ability to reduce the membership inference attacker's ROCAUC, while table 4 convey the ROCAUC of the differently trained models for their respective utility optimums. The results suggest that adversarial regularization provide a different utility-security trade-off than the one inherent to dropout[8]. The trade-off adversarial regularization struck was tilted more in favour of maintaining (or even improving) the synthetic data quality, for all-numerical datasets.

4.6 On dataset meta characteristics and vulnerability to membership inference attacks

In the comparisons made in figures 39 and 40, the Musk dataset stands out as the dataset which leads to the most vulnerable models. The most intuitive explanation of this is that it is also the widest dataset with 168 columns. In order to conceptualize this, one can conjure a simple analogy: human fingerprints. For a human population of a given size, imagine the problem of telling the individuals apart by their fingerprints. This is an accomplishable task, but what if the resolution of the fingerprints was lowered drastically, for instance to the resolution of a 28x28 pixel MNIST image? Chances are, the fingerprints would have been turned into blobs instead of the clear lines they normally trace, which would have made identifying anyone by their fingerprints impossible. An additional, but more technical analogy can be drawn with regards to a support vector classifier. An SVC maps its inputs to higher dimensional spaces, in order to more easily classify the inputs[42]. Similarly to the fingerprints analogy, the conclusion remains: it is easier to tell high-dimensional data apart, than low-dimensional data. This seems also to apply to datasets and their vulnerability to membership inference attacks.

5 Conclusion and future work

In this thesis adversarial regularization has been tested as a method of defence against membership inference attacks. The conclusion with respect to the research questions are:

- **Can regularization reduce a GANs vulnerability to membership inference attacks?** Yes, utilizing adversarial regularization reduced the CTGAN's vulnerability to membership inference attacks, although the reduction was only slight for the generator. The inability of adversarial regularization to increase the generator's robustness to MIAs weakens the case for the use of it, since the generator generally is the part of the GAN which is published and therefore is prone to attacks, while the critic is discarded.
- **To which hyper parameter(s) of the adversarial regularization is the resulting regularization most sensitive?** ϵ proved to be the most important hyper parameter with respect to the adversarial regularization's effect on model robustness. For the synthetic data quality, ϵ and the share of adversarial samples in each training batch proved to have the same effect.
- **What are the trade-offs between regularizing a GAN with adversarial regularization, regularization with alternative methods or not regularizing at all?** Adversarial regularization proved to be the second-best option in terms of reduction in model vulnerability, behind dropout, due to it being a less powerful method of regularization. However, adversarial regularization proved to be the most prolific option of the three in terms of robustness, utility and synthetic data quality, when considering which models provided the best utility on the all-numerical datasets.
- **What are the penalties to the synthetic data quality when conducting adversarial regularization on a GAN?** Moderate adversarial regularization was shown to increase the quality of all-numerical synthetic datasets. For mixed-datasets, adversarial regularization generally fared worse, due to it being incompatible with categorical columns.

Adversarial regularization has proven itself to be able to regularize a GAN and improve its robustness to MIAs, although the effect was subpar to that of dropout. It is safe to say that the use of adversarial regularization in GANs is an under-researched subject, with a lot of potential with respect to improving the implementation of adversarial regularization, understanding why the regularization effect dissipates when translated from the critic to the generator and harnessing the effect of adversarial regularization on the ability of GAN's to produce synthetic data.

Bibliography

[1] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, ‘A committee of neural networks for traffic sign classification’, in *The 2011 international joint conference on neural networks*, IEEE, 2011, pp. 1918–1921.

[2] G. Ian, P.-A. Jean, M. Mehdi *et al.*, ‘Generative adversarial nets’, *Advances in Neural Information Processing Systems*, 2014.

[3] T. Karras, S. Laine and T. Aila, ‘A style-based generator architecture for generative adversarial networks’, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[4] D. Li, D. Chen, B. Jin, L. Shi, J. Goh and S.-K. Ng, ‘Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks’, in *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 703–716.

[5] Y. Bengio, L. Yao, G. Alain and P. Vincent, ‘Generalized denoising auto-encoders as generative models’, *Advances in neural information processing systems*, vol. 26, 2013.

[6] T. Young, D. Hazarika, S. Poria and E. Cambria, ‘Recent trends in deep learning based natural language processing’, *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[7] S. Reza, S. Marco, S. Congzheng and S. Vitaly, ‘Membership inference attacks against machine learning models’, in *2017 IEEE symposium on security and privacy (SP)*, IEEE, 2017, pp. 3–18.

[8] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz and M. Backes, ‘Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models’, *arXiv preprint arXiv:1806.01246*, 2018.

[9] Y. Long, V. Bindschaedler, L. Wang *et al.*, ‘Understanding membership inferences on well-generalized learning models’, *arXiv preprint arXiv:1802.04889*, 2018.

[10] C. Dingfan, Y. Ning, Z. Yang and F. Mario, ‘Gan-leaks: A taxonomy of membership inference attacks against generative models’, in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 343–362.

[11] H. Jamie, M. Luca, D. George and D. C. Emiliano, ‘Logan: Membership inference attacks against generative models’, in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, De Gruyter, vol. 2019, 2019, pp. 133–152.

[12] M. Nasr, R. Shokri and A. Houmansadr, ‘Machine learning with membership privacy using adversarial regularization’, in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 634–646.

[13] I. Goodfellow, J. Shlens and C. Szegedy, ‘Explaining and harnessing adversarial examples’, *International Conference on Learning Representations*, 2015.

[14] G. Elsayed, S. Shankar, B. Cheung *et al.*, ‘Adversarial examples that fool both computer vision and time-limited humans’, *Advances in neural information processing systems*, vol. 31, 2018.

[15] A. Nguyen, J. Yosinski and J. Clune, ‘Deep neural networks are easily fooled: High confidence predictions for unrecognizable images’, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[16] K. Alexey, G. Ian, B. Samy *et al.*, ‘Adversarial examples in the physical world’, 2016.

[17] T. B. Brown, D. Mané, A. Roy, M. Abadi and J. Gilmer, ‘Adversarial patch’, *Computer Vision and Pattern Recognition*, 2017.

[18] J. Jia, A. Salem, M. Backes, Y. Zhang and N. Z. Gong, ‘Memguard: Defending against black-box membership inference attacks via adversarial examples’, in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, ‘Improving neural networks by preventing co-adaptation of feature detectors’, *arXiv preprint arXiv:1207.0580*, 2012.

[20] J. Prateek, K. Vivek, T. Abhradeep and W. Oliver, 'To drop or not to drop: Robustness consistency and differential privacy properties of dropout', *arXiv preprint arXiv:1503.02031*, 2015.

[21] K. Alexey, G. Ian and B. Samy, 'Adversarial machine learning at scale', *International Conference on Learning Representations*, 2017.

[22] D. Saxena and J. Cao, 'Generative adversarial networks (gans) challenges, solutions, and future directions', *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–42, 2021.

[23] L. Yann, B. Léon, B. Yoshua and H. Patrick, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] M. Arjovsky, S. Chintala and L. Bottou, 'Wasserstein generative adversarial networks', in *Proceedings of the 34th International Conference on Machine Learning*, P. Doina and T. Y. Whye, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 214–223. [Online]. Available: %5Curl%7Bhttps://proceedings.mlr.press/v70/arjovsky17a.html%7D.

[25] G. Ishaan, A. Faruk, A. Martin, D. Vincent and C. A. C, 'Improved training of wasserstein gans', *Advances in neural information processing systems*, vol. 30, 2017.

[26] X. Lei, S. Maria, C.-I. Alfredo and V. Kalyan, 'Modeling tabular data using conditional gan', in *Advances in Neural Information Processing Systems*, 2019.

[27] D. Chen, N. Yu and M. Fritz, 'Relaxloss: Defending membership inference attacks without losing utility', in *International Conference on Learning Representations*, 2021.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] S. Christian, Z. Wojciech, S. Ilya *et al.*, 'Intriguing properties of neural networks', *arXiv preprint arXiv:1312.6199*, 2013.

[30] N. Carlini, G. Katz, C. Barrett and D. L. Dill, 'Provably minimally-distorted adversarial examples', *arXiv preprint arXiv:1709.10207*, 2017.

[31] P. McDaniel, N. Papernot and Z. B. Celik, 'Machine learning in adversarial settings', *IEEE Security Privacy*, 2016.

[32] N. Carlini, A. Athalye, N. Papernot *et al.*, 'On evaluating adversarial robustness', *arXiv preprint arXiv:1902.06705*, 2019.

[33] N. Papernot, P. McDaniel and I. Goodfellow, 'Transferability in machine learning: From phenomena to black-box attacks using adversarial samples', *arXiv preprint arXiv:1605.07277*, 2016.

[34] M. Aleksander, M. Aleksandar, S. Ludwig, T. Dimitris and V. Adrian, 'Towards deep learning models resistant to adversarial attacks', *International Conference on Learning Representations*, 2018.

[35] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[36] T. Miyato, A. M. Dai and I. Goodfellow, *Adversarial training methods for semi-supervised text classification*, 2016. arXiv: 1605.07725 [stat.ML].

[37] L. Zinan, K. Ashish, F. Giulia and O. Sewoong, 'Pacgan: The power of two samples in generative adversarial networks', *Advances in neural information processing systems*, vol. 31, 2018.

[38] Y. Freund and R. E. Schapire, 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[39] D. Dheeru and G. Casey, *Uci machine learning repository*, 2017. [Online]. Available: %5Curl%7Bhttp://archive.ics.uci.edu/ml%7D.

[40] Z. Liang, A.-F. Amir, S. Martin and Z. Kai, 'Prediction-time efficient classification using feature computational dependencies', in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, 2018.

[41] A. Hernández-Garcıa and P. König, ‘Data augmentation instead of explicit regularization’, *arXiv preprint arXiv:1806.03852*, 2018.

[42] V. V. N., *Statistical Learning Theory*. 1998.

Appendix

A Appendix

Appendices A, B and C present results obtained from all datasets in terms of machine learning efficacy, robustness to MIAs and synthetic data quality. The results were procured by three differently trained CTGAN models: a regular model, a model trained with dropout and a model trained with adversarial regularization. For each dataset and model type, the best parameterized model in terms of machine learning efficacy was selected. Table 2 contains the hyper parameters used for the selected models.

Dataset / Training mode	Regular	Dropout	Adversarial regularization
Adult	200 epochs	Dropout = 0.3, 350 epochs	$\epsilon = 0.3$, 350 epochs
Covertype	350 epochs	Dropout= 0.1, 500 epochs	$\epsilon = 0.2$, 350 epochs
UAV intrusion detection	350 epochs	Dropout = 0.7, 400 epochs	$\epsilon = 0.2$, 250 epochs
Musk	500 epochs	Dropout= 0.1, 500 epochs	$\epsilon = 0.4$, 500 epochs

Table 2: Hyper parameters of the models with best utility.

For all the models trained with adversarial regularization, the share of adversarial samples was 50%.

Appendices D and E respectively presents the hyper parameters used in the networks and optimizers, and the software requirements for running the code.

A Machine learning efficacy

Table 3 exhibits the machine learning efficacy of three model types with the hyper parameters indicated in table 2.

Dataset	Metric	Regular	Dropout	Adversarial regularization
Adult	Accuracy	$0.833 \pm 1.30e^{-5}$	$0.822 \pm 5.20e^{-5}$	$0.825 \pm 7.30e^{5}$
	ROCAUC	$0.882 \pm 9.00e^{-6}$	$0.885 \pm 1.00e^{-6}$	$0.887 \pm 6.00e^{-6}$
	F1 score	$0.603 \pm 1.72e^{-3}$	$0.658 \pm 1.20e^{-5}$	$0.658 \pm 2.20e^{-5}$
Covertype	Accuracy	$0.625 \pm 1.71e^{-4}$	$0.634 \pm 4.21e^{-5}$	$0.613 \pm 3.79e^{-4}$
	ROCAUC	$0.846 \pm 1.97e^{-4}$	$0.841 \pm 2.38e^{-5}$	$0.835 \pm 1.50e^{-5}$
	F1 score	$0.396 \pm 5.00e^{-5}$	$0.355 \pm 7.29e^{-4}$	$0.379 \pm 7.65e^{-4}$
UAV intrusion detection	Accuracy	$0.991 \pm 2.6e^{-3}$	$0.992 \pm 1.8e^{-3}$	$0.995 \pm 2.0e^{-3}$
	ROCAUC	$0.997 \pm 1.4e^{-3}$	$0.997 \pm 4.0e^{-3}$	$0.999 \pm 7.5e^{-4}$
	F1 score	$0.992 \pm 2.2e^{-3}$	$0.993 \pm 1.7e^{-3}$	$0.995 \pm 1.7e^{-3}$
Musk	Accuracy	$0.617 \pm 1.9e^{-2}$	$0.603 \pm 1.0e^{-2}$	$0.645 \pm 2.3e^{-3}$
	ROCAUC	$0.642 \pm 2.5e^{-2}$	$0.640 \pm 1.2e^{-2}$	$0.674 \pm 1.7e^{-2}$
	F1 score	$0.544 \pm 5.1e^{-1}$	$0.517 \pm 4.0e^{-2}$	$0.581 \pm 1.8e^{-2}$

Table 3: Machine learning efficacy for all datasets.

B Robustness to membership inference attacks

Table 4 exhibits the mean attack ROCAUC against the critic and generator from the models trained with the hyper parameters indicated in table 2.

Dataset	Target	Regular	Dropout	Adversarial regularization
Adult	Critic	$0.533 \pm 8.45e^{-3}$	0.533 ± 0.0197	$0.550 \pm 5.44e^{-3}$
	Generator	$0.527 \pm 8.71e^{-3}$	$0.482 \pm 3.2e^{-3}$	0.492 ± 0.0125
Covertype	Critic	0.505 ± 0.0189	0.520 ± 0.0152	0.509 ± 0.0161
	Generator	$0.497 \pm 5.80e^{-3}$	0.509 ± 0.0175	$0.503 \pm 4.80e^{-3}$
UAV intrusion detection	Critic	$0.519 \pm 9.19e^{-3}$	0.521 ± 0.0163	$0.501 \pm 4.15e^{-3}$
	Generator	$0.516 \pm 2.12e^{-3}$	$0.514 \pm 1.25e^{-4}$	$0.509 \pm 6.38e^{-3}$
Musk	Critic	$0.725 \pm 7.92e^{-3}$	0.716 ± 0.0265	0.693 ± 0.0108
	Generator	$0.702 \pm 7.80e^{-3}$	$0.694 \pm 4.11e^{-3}$	$0.680 \pm 9.60e^{-4}$

Table 4: Attack ROCAUC for all datasets.

C Synthetic data quality

Table 5 exhibits the quality of the synthetic data produced by the three model types with the hyper parameters indicated in table 2.

Dataset	Metric	Regular	Dropout	Adversarial regularization
Adult	Wasserstein distance	2062 ± 798.4	3737 ± 585.4	3291 ± 485.1
	RMSE	$0.0251 \pm 1.53e^{-3}$	$0.0281 \pm 8.72e^{-3}$	$0.0280 \pm 4.77e^{-3}$
	Kolmogorov-Smirnov statistc	0.144 ± 0.0256	0.187 ± 0.0157	0.148 ± 0.0216
	Fake pairwise unique combinations*	2466 ± 26.9	2440 ± 30.0	2470 ± 40.2
Covertype	Wasserstein distance	110 ± 32.9	86.2 ± 31.8	87.6 ± 23.8
	RMSE	$0.0596 \pm 5.94e^{-3}$	$0.0573 \pm 6.48e^{-4}$	$0.0577 \pm 9.67e^{-3}$
	Kolmogorov-Smirnov statistc	0.154 ± 0.0218	0.134 ± 0.0108	0.153 ± 0.0363
	Fake pairwise unique combinations**	116 ± 1.25	107 ± 1.41	110 ± 1.70
UAV intrusion detection	Wasserstein distance	22.2 ± 2.49	24.8 ± 0.903	24.6 ± 3.10
	RMSE	0.131 ± 0.0484	0.180 ± 0.0186	$0.0755 \pm 7.39e^{-3}$
	Kolmogorov-Smirnov statistc	$0.331 \pm 5.10e^{-3}$	$0.352 \pm 4.82e^{-3}$	$0.302 \pm 2.32e^{-3}$
Musk	Wasserstein distance	22.5 ± 0.134	23.0 ± 0.368	21.7 ± 0.234
	RMSE	$0.0605 \pm 1.45e^{-3}$	$0.0632 \pm 2.17e^{-3}$	$0.0523 \pm 2.28e^{-3}$
	Kolmogorov-Smirnov statistc	$0.191 \pm 1.20e^{-3}$	$0.204 \pm 2.83e^{-3}$	$0.176 \pm 1.27e^{-3}$

Table 5: Synthetic data quality metrics for all datasets. (*The total number of unique pairwise combinations in the adult set was 2323, **and 71 in the covertype dataset.)

D Network and optimizer parameters

The following are the hyper parameters used in the CTGAN for all datasets:

Dataset	Adult	Covertype	UAV intrusion detection	Musk
Critic LR	$5e^{-4}$	$9e^{-4}$	$2e^{-4}$	$2e^{-4}$
Generator LR	$5e^{-4}$	$9e^{-4}$	$2e^{-4}$	$2e^{-4}$
Batch size	500	500	500	500
Embedding dim	128	128	128	128
Generator dim	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Discriminator dim	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Number of critic steps	1	1	1	1
Optimizer	Adam	Adam	Adam	Adam
β_1	0.4	0.5	0.5	0.5
β_2	0.8	0.9	0.9	0.9
Decay	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$

Table 6: Network and optimizer parameters.

The following are the hyper parameters utilized in the generator membership inference attack for all datasets:

Dataset/Parameter	LR	# iterations	Batch size	Optimizer	(β_1, β_2)
Adult	0.25	1000	250	Adam	(0.5,0.9)
Covertype	0.25	1000	250	Adam	(0.5,0.9)
Intrusion detection	0.25	1000	250	Adam	(0.5,0.9)
Musk	0.025	500	250	Adam	(0.5, 0.9)

Table 7: Generator MIA hyper parameters.

E Software requirements

Table 8 describes the main python dependencies for the code developed by the author for this thesis.

Python library	Version
ctgan	0.5.0
matplotlib	3.5.0
numpy	1.21.5
pandas	1.4.0
python	3.8.12
pytorch	1.10.2
scikit-learn	1.0.2
scipy	1.8.0
seaborn	0.11.2

Table 8: Major software dependencies.