Population Obfuscation with GAN's

Peyton Krahulik, Kyle Caudle, Randy Hoover

April 29, 2022

Abstract

Real data often contains proprietary information, or in the case of military applications, the data may actually, be classified. The goal of this research is to investigate generative models and maps that will create a distribution that resembles the population with sensitive information that has been masked. In this research, a generative adversarial network (GAN) is used to generate data from the masked distribution that is similar to the population distribution while obscuring sensitive information. Through computer experiments, it can be shown that if there is a large sample from the population and a relatively small sample of data with masked information, fake instances can be created from the masked distribution. Success is measured by comparing the distance between the masked distribution and the population distribution.

1 Introduction

Data Privacy is an ever rising issue in today's world, especially in the medical or military world. Providing access to sensitive data while maintaining privacy and confidentiality is a very well-known challenging issue. Current methods exist to solve this issue to a degree, but they aim to protect the individual record rather then the population variables. Even with these current methods in place there are many publicized examples of privacy being violated regardless of the methods in place to prevent the violation. A good example of this is a dataset that Netflix released in 2006. Netflix released a dataset of 10 million movie rankings by 500 thousand customers. All personal details such as names were removed and replaced with random numbers. But, the people were still able to be identified by comparing the timestamp with public information from the Internet Movie Database (IMDb).

There have been many proposed solutions to solve this issue from privacy-protection architectures built around adversarial networks to variational auto-encoders. Combinations of all these methods have also been implemented to some effect.

The goal of all the current methods is to protect the privacy of individual records by obfuscating a finite sample of the larger population. This can make it hard to estimate population parameters and make good inferences about the population as a whole. This project seeks to protect information and features of the entire statistical population of data while obfuscating elements of the data population. Specifically, this project seeks to create a generative model for the population from which the user can generate an infinite number of obfuscated elements of the population. This is population obfuscation with a focus on using Generative Adversarial Networks (GAN's) to create the generative model.

1.1 Background and Motivation

There are currently many implementations of sample obfuscation. The five most common methods of sample obfuscation are:

- *Data Masking* is the process of modifying sensitive data in such a way that it is of little to no value to unauthorized personnel, while still being usable by software applications or authorized personnel. In most cases this involves substituting real sensitive data with realistic looking fake values that carry the same statistical information.

- *Differential Privacy* is an algorithm that aggregates statistics in such a way to make it hard if not impossible to tell if an individual record was used in the calculation of that statistic. This often involves introducing noise into a dataset to maintain the privacy of its entries.

- *Encryption* is a method by which entries in a dataset or the entire dataset are converted from plain text, human readable text, and converted by some encryption method into ciphertext, which is unreadable and not easily converted back to plaintext. A classic example of this would be an affine cipher where characters are multiplied by a constant and then shifted. Encryption is by far the most secure method of data obfuscation, but it also makes data analysis useless. It is typically only used to store and transfer data securely.

- *Redaction* involves partially or completely removing records from a dataset. These record are then replaced with a predefined constant value, usually zero or one for numerical records. Introducing large amounts of this constant value makes data analysis hard.

- *Tokenization* is similar to redaction in the sense that records are partially removed, but instead of replacing the removed portion with a constant value it is replaced with a token that allows authorized personnel or software access the real value. All database operations are preformed on this token, while the original data is encrypted behind a secure token key. Without the token's encryption key, the tokenized data cannot be recovered.

Sample differential privacy, sample encryption, and sample tokenization all aim to hide certain marked information or variables. This maintains the privacy of individual records at the cost of population variables, having no useful parallels in population obfuscation. Redaction and masking on the other hand, hide variables and can be formalized for population obfuscation. A GAN can be used to either mask or redact a marked population $\mathcal{M}$ given a large enough sample from the masked or redacted target population $\mathcal{T}$

2 Population Obfuscation

Population obfuscation is based on the notion of a marked population $\mathcal{M}$ containing marked information. This population $\mathcal{M}$ cannot be released to the public because it contains said marked information. Therefor, the goal is to either construct or learn the population $\mathcal{T}$. In $\mathcal{T}$ the marked observations are obfuscated, but otherwise $\mathcal{T}$ is no different from $\mathcal{M}$. It is easy to train a generative model to generate $\mathcal{T}$, so it is assumed that there is not enough sample of $\mathcal{T}$ to train a standard generative model. For this problem

it is assumed that there is a sample $\mathcal{S}_{\mathcal{M}}$ from $\mathcal{M}$ and a sample $\mathcal{S}_{\mathcal{T}}$ from $\mathcal{T}$. In addition, $\mathcal{S}_{\mathcal{M}}$ has a size of $N_{\mathcal{M}}$ and $\mathcal{S}_{\mathcal{T}}$ has a size of $N_{\mathcal{T}}$.

The size $N_{\mathcal{M}}$ of the sample $\mathcal{S}_{\mathcal{M}}$ is assumed to be large enough to train a generative model to learn $\mathcal{M}$. This means that the population $\mathcal{M}$is known in its entirety, including its population parameters. On the other hand, the size $N_{\mathcal{T}}$ of sample $\mathcal{S}_{\mathcal{T}}$ is assumed to be to small to train a generative model to learn $\mathcal{T}$ and $\mathcal{T}$'s population parameters. If $\mathcal{S}_{\mathcal{T}}$ were to be large it is a trivial matter to train a generative model to learn $\mathcal{S}_{\mathcal{T}}$ in its entirety. Likewise, if $\mathcal{S}_{\mathcal{T}}$ is zero $\mathcal{T}$ is unlearnable. In either case population obfuscation is rendered trivial. With limited amounts of $\mathcal{S}_{\mathcal{T}}$ and an infinite amount of $\mathcal{S}_{\mathcal{M}}$, $\mathcal{S}_{\mathcal{M}}$ is used in addition to $\mathcal{S}_{\mathcal{T}}$ to augment the GAN's training.

Now the marked distribution $\mathcal{M}$ is defined by a vector

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

where the y_i's are in no particular order. The next assumption is that some unknown masking map g exists that identifies and isolates the marked information contained within $\mathbf{Y}$ that needs to be obfuscated. The isolation map is defined as:

$$g : \mathbf{Y} \to \mathbf{Z} = \begin{bmatrix} g_m(\vec{y}) \\ g_x(\vec{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{M} \\ \mathbf{X} \end{bmatrix}$$

where,

- $\mathbf{M} = \begin{bmatrix} m_1 & m_2 & \dots & m_m \end{bmatrix}^T$ is the marked information contained within $\mathbf{Y}$
- $\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_{n-m} \end{bmatrix}^T$ is the non-marked information contained within $\mathbf{Y}$
- $\mathbf{M}$ and $\mathbf{X}$ are stochastically dependent.

It is obvious that there are two paths to be taken. Either $\mathbf{M}$ could be masked by randomizing it or $\mathbf{M}$ could be redacted by changing every m_i to a constant "black-out" value. Preforming either operation breaks the dependence between $\mathbf{M}$ and $\mathbf{X}$. After the dependence is broken, g is inverted to obtain the target distribution $\mathcal{T}$. For masking, the target distribution is defined as:

$$\mathcal{T} = g^{-1}(\begin{bmatrix} \mathbf{H} \\ \mathbf{X} \end{bmatrix})$$

where, $\mathbf{H}$ is drawn from distribution of $\mathbf{M}$ such that $\mathbf{H}$ and $\mathbf{X}$ are stochastically independent. For redacting, $\mathcal{T}$ is defined as:

$$\mathcal{T} = g^{-1}(\begin{bmatrix} \mathbf{H}_B \\ \mathbf{X} \end{bmatrix})$$

where, $\mathbf{H}_B$ is a vector of the constant "black-out" values. $\mathbf{H}_B$ and $\mathbf{X}$ are stochastically independent.

2.1 Population Obfuscation: BMI exmaple

A very simple example of population obfuscation involves body mass index (BMI). Consider a case where a patient's height and weight are represented as a vector:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \in \mathbb{R}^2$$

Now $\mathbf{Y}$ respresents a patients physical data such that y_1 is a patients height and y_2 is a patients weight. The goal in this example is to obfuscate a patients BMI ($\frac{y_2}{y_1^2}$) while maintaining their size ($y_1 * y_2$). An isolation map for this problem is,

$$g : \mathbf{Y} \to \mathbf{Z} = \begin{bmatrix} \mathbf{M} \\ \mathbf{X} \end{bmatrix} = \begin{bmatrix} \frac{y_2}{y_1^2} \\ y_1 * y_2 \end{bmatrix},$$

where the marked information, $\mathbf{M}$, is BMI and the non-marked information, $\mathbf{X}$, is size. Since both BMI and size depend on a patient's height and weight, $\mathbf{M}$ and $\mathbf{X}$ are stochastically dependent. Let $\mathcal{M}^\circ$ denote the distribution of $\mathbf{M}$. To break the dependency between $\mathbf{M}$ and $\mathbf{X}$, a sample from $\mathbf{M}$ is drawn, call it $\mathbf{H} \sim \mathcal{M}^\circ$. Then $\mathbf{M}$ is replaced with $\mathbf{H}$.

$$\mathbf{H} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \sim \mathcal{M}^\circ$$
$$\begin{bmatrix} \mathbf{H} \\ \mathbf{X} \end{bmatrix} = \begin{bmatrix} \frac{h_2}{h_1^2} \\ y_1 * y_2 \end{bmatrix}$$

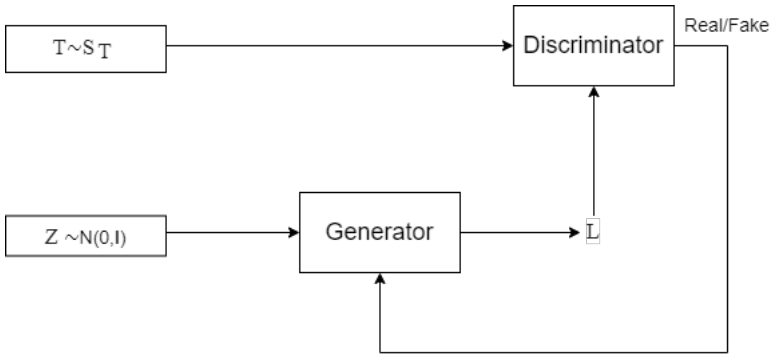
Now, $\mathbf{H}$ and $\mathbf{X}$ are stochastically independent since $\mathbf{H}$ was obtained from randomly sampling from $\mathcal{M}^\circ$. To obtain the target population $\mathcal{T}$, the inverse transformation of the isolation map g is preformed:

$$g^{-1}(\begin{bmatrix} \mathbf{H} \\ \mathbf{X} \end{bmatrix}) = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \sim \mathcal{T}$$

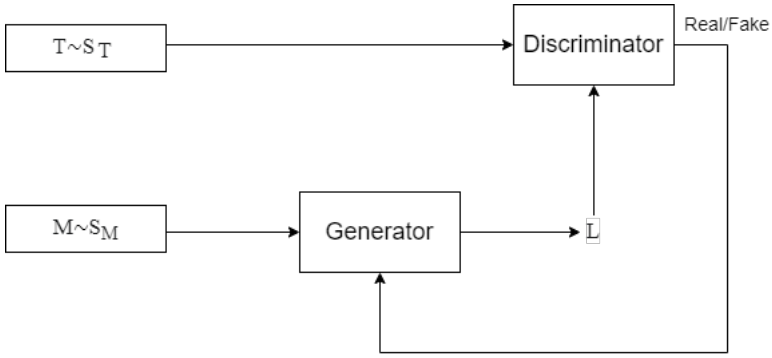
The goal of population obfuscation is to either generatively model $\mathcal{T}$, or to construct an estimate of g^{-1} which converts $\mathcal{M}$ to $\mathcal{T}$.
For a GAN, an infinite amount of sample from $\mathcal{M}$ exist, which can be used to train the GAN to create an estimator, $\mathcal{L}$, of the target population $\mathcal{T}$.

3 Generative Adversarial Network's (GAN's

A Generative Adversarial Network, or GAN for short, is a system of two neural networks that train against one another to achieve a desired outcome. Usually, this outcome is to generate new examples that could have come from some dataset. There are two neural networks in a GAN: a generator and a discriminator.
The generator takes in a vector and then applies a series of weight to output a learned distribution, $\mathcal{L}$. $\mathcal{L}$ is then fed into the discriminator with a sample, $\mathcal{S}_\mathcal{T}$, from the target distribution $\mathcal{T}$. The discriminator then outputs a probability that a given sample is from $\mathcal{T}$ or from $\mathcal{L}$. This probability is fed back into the generator and the generator then updates its weights. Below is an example of the Vanilla GAN architecture for population obfuscation.



In the above example the generator is fed a vector of Gaussian noise, $N(0, \mathbf{I}) \sim \mathcal{Z}$, to start. The generator then produces $\mathcal{L}$ and then $\mathcal{L}$ and a sample form the target distribution $\mathcal{S}_{\mathcal{T}}$ are fed into the discriminator. The discriminator then output whether the sample is real or fake and feeds that into the Generator. This means that the generator's gain is the discriminator's loss and vice versa. The only problem here is that the sample $\mathcal{S}_{\mathcal{T}}$ is too small to train a generative model. This leads to an obfuscatory GAN architecture:



This Obfuscatory GAN works exactly the same as the vanilla GAN with the exception that the generator is fed a sample, $\mathcal{S}_{\mathcal{M}}$, from the marked population $\mathcal{M}$. The only problem now is to evaluate how well the GAN has learned $\mathcal{T}$.

3.1 2 Wasserstein Distance

The 2 wasserstein distance is used to measure how much work it would take to transform one distribution into another distribution. In low dimensional settings the 2 wasserstein distance is also called the earthmover distance. The 2 wasserstein distance between two normal distributions μ_1 and μ_2 is defined as:

$$W_2(\mu_1, \mu_2)^2 = ||\vec{m_1} - \vec{m_2}||_2^2 + tr(C_1) + tr(C_2) - tr(2(C_2^{1/2} C_1 C_2^{1/2})^{1/2})$$

where μ_1 and μ_2 are normal distributions, $\vec{m_1}$ and $\vec{m_2}$ are vectors of means, and C_1 and C_2 are the covariance matrices of the two distributions. Imagine that there are two piles of three dimensional sand, the 2 wasserstein distance measures the amount of work it takes to transform one pile into a perfect copy of the second pile.
To check how close the learned distribution $\mathcal{L}$ is to $\mathcal{T}$ the 2 wasserstein distance is calculated between $\mathcal{L}$ and $\mathcal{T}$. In addition, to start out the experiments with population obfuscation a target population is chosen so that an analytically calculation of the 2 wasserstein distance between $\mathcal{L}$ and $\mathcal{T}$ exists.

4 Toy Two-dimensional Problem

To start out a simple two-dimensional obfuscation problem was explored. In which, a GAN was trained using a marked bivariate normal distribution $\mathcal{M}$ and tasked with transforming $\mathcal{M}$ into $\mathcal{T}$, which is a rotated bivariate normal distribution. At the same time another GAN is trained using the standard normal distribution, call it $\mathcal{Z}$, to transform $\mathcal{Z}$ into $\mathcal{T}$ as well. This toy problem is engineered so that $\mathcal{T}$ and $\mathcal{M}$ are known. To start out the marked population $\mathcal{M}$ is:

$$\mathcal{M} = N(\vec{0}, \Sigma_{\mathcal{M}}), \Sigma_{\mathcal{M}} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} \rho & 0 \\ 0 & 1/\rho \end{bmatrix}$$

The isolation map is defined as:

$$g : \vec{Y} \to \vec{S} = \begin{bmatrix} \mathbf{M} \\ \mathbf{X} \end{bmatrix}$$

This isolation map for this example is just the unitary transformation $\tilde{\mathbf{S}} = \mathbf{U}\tilde{\mathbf{Y}}$, where $\mathbf{U}$ is:

$$\mathbf{U} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Under this transformation $\mathbf{Y}$ and $\mathbf{S}$ are:

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \vec{S} = \begin{bmatrix} M \\ X \end{bmatrix}$$

The mark $\mathbf{M}$ in $\tilde{\mathbf{S}}$ is $M = y_1 \cos\theta + y_2 \sin\theta$. Let $\mathbf{S}^\circ$ denote the de-marked distribution of $\mathbf{S}$. $\mathbf{S}$ and $\mathbf{S}^\circ$ have distributions defined as:

$$\mathcal{S} = N(\vec{0}, \Sigma_{\mathcal{S}}), \Sigma_{\mathcal{S}} = \Sigma_{\mathcal{M}} + (\rho - 1/\rho)\sin\theta \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & -\cos\theta \end{bmatrix}$$

$$\mathcal{S}^\circ = N(\vec{0}, \Sigma_{\mathcal{S}^\circ}), \Sigma_{\mathcal{S}^\circ} = \Sigma_{\mathcal{M}} + (\rho - 1/\rho)\sin^2\theta \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Taking the inverse of the isolation map yields the distribution of $\mathcal{T}$:

$$\mathbf{T} = N(\mathbf{0}, \mathbf{\Sigma_T}), \mathbf{\Sigma_T} = \mathbf{U^T \Sigma_{S^\circ} U} = \frac{\rho + \mathbf{1}/\rho}{\mathbf{2}}\mathbf{I} + \frac{\rho - \mathbf{1}/\rho}{\mathbf{2}}\cos\mathbf{2}\theta \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix}$$

In this example a linear map exists taking $\vec{\mathbf{Y}} \sim \mathcal{M}$ to $\vec{\mathbf{T}} \sim \mathcal{T}$: $g : \vec{\mathbf{Y}} \to \vec{\mathbf{T}} => \vec{\mathbf{T}} = \mathbf{V}\vec{\mathbf{Y}}$, where:

$$\mathbf{V} = \sqrt{\Sigma_{\mathcal{T}}\Sigma_M^{-1}}$$

Now that an analytic solution to $\mathcal{T}$ and $\mathcal{M}$ exists, samples can be generated and thrown into a GAN.

4.1 Experiment Design

The experiment is to run a small experiment that tests out if training a obfuscatory GAN is better then a traditional GAN with a small sample, $\mathcal{S}_{\mathcal{T}}$, from the target population $\mathcal{T}$ and a large sample, $\mathcal{S}_{\mathcal{M}}$ from the masked population $\mathcal{M}$. Before the experimented is started, two assumptions must be addressed: the first is that $\mathcal{S}_{\mathcal{M}}$ is assumed to be infinite and the second is that $\sigma_1 * \sigma_2 = 1$. For the experiment, a full factorial experiment design with three different experiment factors (ρ, θ, and $\mathcal{S}_{\mathcal{T}}$) is used for a total of 48 design points. Below are the values used for the three factors:

$$\rho = 1.5, 2, 2.5, 3$$
$$\theta = \tfrac{\pi}{12}, \tfrac{\pi}{6}, \tfrac{\pi}{4}$$
$$N_T = 10, 20, 50, 100$$

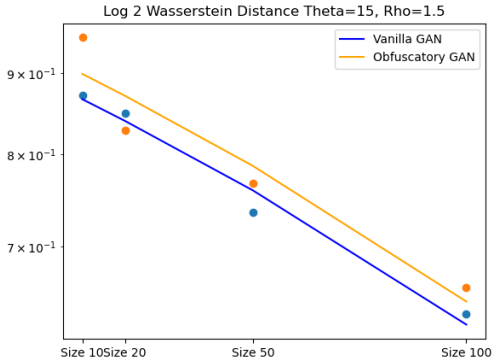
100 runs were preformed per design point on both a vanilla GAN and an obfuscatory GAN for a total of 9600 GAN's trained. After each GAN was trained the 2 Wasserstein distance was calculated between $\mathcal{L}$ and $\mathcal{T}$ for both the obfuscatory GAN and the vanilla GAN.

4.2 GAN Parameters

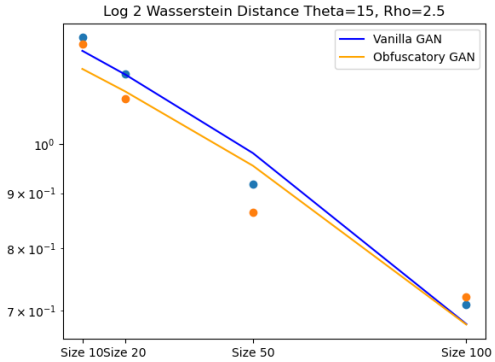
Hyperparameters	Generator	Discriminator
Layers	3	3
Nodes	128, 128, 128	32, 32, 32
Activation	Relu, Relu, Relu	Relu, Relu, Sigmoid
Optimizer	Adam	Adam
Learning Rate	0.002	0.002
Num Epochs	5000	5000
Loss Function	BCE	BCE

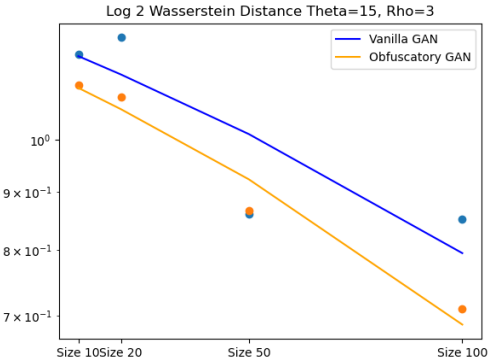
4.3 Results

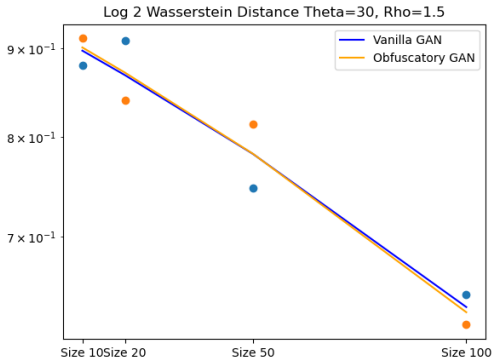
The results of the 2 Wasserstein distance between $\mathcal{L}$ and $\mathcal{T}$ were calculated for every design point as the average 2 wasserstein distance over the 100 runs. Outcomes were then plotted on a log log scale grouped by theta value.
The graphs, shown below, do not point to any conclusive evidence that the obfuscatory GAN preformed any better then the Vanilla GAN on low dimensional setting, although more tuning of the hyperparameters may be needed.

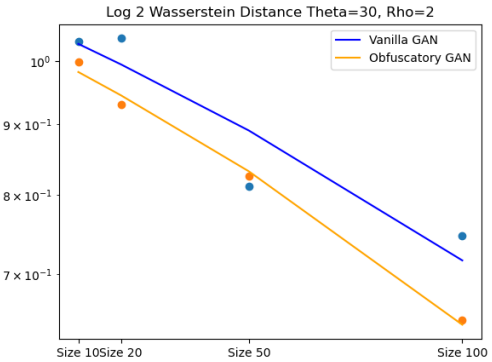

Log 2 Wasserstein Distance Theta=15, Rho=1.5
Vanilla GAN
Obfuscatory GAN
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100


Log 2 Wasserstein Distance Theta=15, Rho=2
Vanilla GAN
Obfuscatory GAN
10^{0}
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100


Log 2 Wasserstein Distance Theta=15, Rho=2.5
Vanilla GAN
Obfuscatory GAN
10^{0}
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100


Log 2 Wasserstein Distance Theta=15, Rho=3
Vanilla GAN
Obfuscatory GAN
10^{0}
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100


Log 2 Wasserstein Distance Theta=30, Rho=1.5
Vanilla GAN
Obfuscatory GAN
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100


Log 2 Wasserstein Distance Theta=30, Rho=2
Vanilla GAN
Obfuscatory GAN
10^{0}
9×10^{-1}
8×10^{-1}
7×10^{-1}
Size 10
Size 20
Size 50
Size 100