University of Twente

Master Thesis

Generative Adversarial Models for Privacy-Preserving Release Mechanisms

Author:
Max Vasterd

Supervisors:
Dr. Ir. Jasper Goseling
Dr. Ir. Maurice van Keulen

A thesis submitted in fulfillment of the requirements for the degree of MSc Computer Science

in the

Faculty of Eletrical Engineering, Mathematics and Computer Science
Data Management & Biometrics

January 15, 2021

UNIVERSITY OF TWENTE

Abstract

Faculty of Eletrical Engineering, Mathematics and Computer Science
Data Management & Biometrics

MSc Computer Science

Generative Adversarial Models for Privacy-Preserving Release Mechanisms

by Max Vasterd

Since the advancements of the Generave Adversarial Networks, many works have been proposed to better guarantee privacy in privacy-preserving release mechanisms. In this thesis, the current measures for privacy-leakage will be compared and studied in a practical experiment. This thesis uses the paper of Tripathy et al. as a baseline [14], studies it, and continues their work by comparing their privacy-leakage measure mutual information to two other existing measures: Maximal Leakage and (Maximal) Alpha-Leakage. We start this work by reconstructing the paper of Tripathy et al. and find that this is not trivial. And the results are less promising then they appear in their paper. Furthermore, we show a generalized version of their work so that privacy-leakage measures other than mutual information are adaptable. Finally, we show how a generative adversarial network is trained on bivariate binary data using the maximal leakage and α-leakage measures, and show the relation between $\alpha = 1$ with mutual information and $\alpha = \infty$ with maximal leakage.

Contents

List of Figures

List of Tables

List of Abbreviations

BMI	**B**ody-**M**ass-**I**ndex
DP	**D**ifferential **P**rivacy
GAN	**G**enerative **A**dversarial **N**etwork
hamm	**hamm**ing distance
MaxL	**Max**imal **L**eakage
MI	**M**utual **I**nformation
MSE	**M**ean **S**quared **E**rror
PUT	**P**rivacy-**U**tility **T**rade-off
SGD	**S**tochastic **G**radient **D**escent

Chapter 1

Introduction

Organizations and companies gather a lot of data in the current age. This data is used to support functionality of apps and help researchers. Unfortunately, this data usually includes a lot of privacy-sensitive information of a client. Sometimes, this data is explicitly allowed to be used by the client for specified actions. However, the same data can mostly also be used for actions of which the client has no interest and thus are potentially be harmful. To overcome this issue, a lot of research has been done in the field of privacy-preserving systems. For such a system, the goal is to make it impossible for an adversary to infer the sensitive attributes given the released data. This system will be referred to as privatizing the data. However, this appears to be a non-trivial task as privatizing data is always at the cost of its usability. This is often referred to as the privacy-utility trade-off [2].

A good indication of the challenges in this task is the privacy leakage based on the Netflix prize challenge [11]. Netflix thought the data to be privatized. However, researchers have shown that the data was in fact leaking Netflix' users' video preferences based on a linkage attack using auxiliary data from the IMDB-database. Other research showed that Apple's privatizing scheme was not as secure as they claimed it to be [13], this is mainly due to the negligence of privacy over time.

As stated by [7], one of the most popular privatizer mechanism today is that of Differential Privacy (D.P.) by [3]. It is a widely used transformation for privatizing data, however, limited in quantifying the remaining utility. This method approaches the problem by adding noise according to the Laplace distribution to all attributes. By this mechanism, the overall distribution of the data should remain almost identical to the original but with preserved privacy. However, the amount of noise needed to make the new data-set indistinguishable from the original data, decreases the utility of the data significantly. This is inherent to the fact that no context is used to privatize the data [7]. Furthermore, this method does not satisfy the linkage inequality [17]. This means that it is great for estimating the distribution of the data in a privacy-preserving manner. However, when it comes to other data statistics or entry specific inquiry, it is less of a good solution due to this great utility loss.

In this thesis we further investigate a possible alternative using a machine learning approach. To be specific, we will focus on a solution-design inspired by Generative Adversarial Networks by Tripathy et al. This paper will be used as guideline throughout this thesis, and is further introduced in Section 1.1.

1.1 Data and Minimizing Privacy-Leakage

To address the problems of privacy-preserving systems the following three categories are defined for the source data set in [12] and will be used accordingly in this document:

1. **Explicit Identifiers** These type of attributes directly define the subject. Such as: Names and Social Security Number.
2. **Quasi Identifiers** Attributes of this sort potentially allow an adversary to infer the explicit identifiers when linked to external data. Quasi Identifiers include: Postal Code, Country and Street.
3. **Sensitive Attributes** These are the attributes that are relevant for research or app functionality, but are also the most sensitive attributes once associated with the identity of the subject. Therefore, these associations must be kept secret to anyone who has no direct access. Examples of these attributes include: a patient's disease and a client's sexual preference or religion.

The perspective on these attributes, however, is still subject to the purpose of an application or research project. The parts that should and should not be inferable are completely application-dependent. A clear example would be the practical experiment on a Face-to-BMI (Body Mass Index) classifier. In such an application face-images would be privatized with the goal to hide the person's identity and retain the BMI inference. Depending on the application, it would be equally reasonable to perform this experiment conversely, i.e. hide the BMI and retain the identity. Whether such an approach is or is not ethically justifiable is outside the scope of this thesis.

In short, privacy-preserving release mechanisms should keep as much information as possible especially sensitive attributes. However, it should be impossible for an adversary to infer the identity of the corresponding subjects, or vice versa.

For simplicity, it is assumed to hide Explicit and Quasi identifiers, and retain the sensitive attributes (e.g. BMI, age, ethnicity). To achieve this, Explicit identifiers are usually removed altogether, and the remaining two categories undergo some transformation. This problem is represented by the rate-privacy function. Consider some input W composed of two random variables X and Y, where X represent the Quasi identifiers (which we will consider as private parts) and Y represents the sensitive data (and we thus consider the public parts, the part we wish to make public). Further, suppose we have a communication channel $P_{Z|W}$ such that Z is the released version of W with limited information on X, where a communication channel is some medium for transmitting information. Then the most informative channel which preserves most privacy, where the utility constraint is given by ϵ, is described as

$$\min_{P_{Z|W}:(X,Y)\leftrightarrow W\leftrightarrow Z} I(X;Z), \text{ s.t. } I(Y;Z) \geq \epsilon, \tag{1.1}$$

which is also known as the privacy-funnel [1]. Intuitively, this equation shows that the mutual information of X and Z, the information contained in both X and Z indicated by I(X;Z), is minimized by adjusting the release mechanism $P_{Z|W}$. In the context of this paper, this is the variational approximation of Z given W. However, the constraint stated shows that this same release, i.e. Z, should still contain enough information on Y (indicated by $I(Y;Z) \geq \epsilon$). This is a constraint on the utility of Z, more on Mutual Information in Section 2.3.1.

Tripathy et al. have addressed this optimization problem using a Generative Adversarial Network approach. The Generative Adversarial Network architecture is used to infer the variational approximation $P_{Z|W}$, i.e. the optimal privacy-preserving release mechanism of Z given W [14]. However, they generalize the utility constraint

$I(Y;Z) \leq \delta$ to any distortion function. This function should illustrate the distortion of Z given Y:

$$\min_{P_{Z|W}:(X,Y)\leftrightarrow W\leftrightarrow Z} I(X;Z), \text{ s.t. } d(Y,Z) \leq \delta. \tag{1.2}$$

Note that this shows the distortion constraint instead of utility, and thus flipped the equality. Tripathy et al. find promising results by doing so. In fact, their networks are capable of finding the theoretical best solution. Although their findings are compelling, we found it challenging to reproduce and will aim to produce more insights into the approach in Chapter 4. This thesis builds on their findings; derivation of the unconstrained minimax problem and network topology, which we further explain in Chapter 3 and Chapter 4.

1.2 Research Question

The current research on privacy-preserving release mechanism inspired by generative adversarial networks is greatly increasing, yet has no single regulated measure for privacy-leakage. As we will explore in Section 2.3, many more privacy-leakage measures exist than just mutual information. As mentioned earlier, the results of [14] are promising, yet entirely in the context of mutual information as privacy-leakage measure. In this thesis we will expand on the paper of Tripathy et al. and apply alternative privacy-leakage measures on their framework. Therefore, the goal of this thesis is to find the optimal network that, given a utility constraint, minimizes the privacy-leakage of the data. The privacy leakage for the networks is either measured by Mutual Information, Maximal Leakage, α-Leakage or Maximal α-Leakage. The research question can therefore be defined as: "How do Maximal Leakage and (Maximal) α-leakage functions compare to and affect the performance of the release-system framework defined in Tripathy et al.". Outside the scope of this thesis is testing different neural network topologies. Further, we only use existing proofs, there are no proofs of mathematical relations. There is no prove that it is impossible to infer the privatized data either. We do show, by the means of mutual information, that there will little information to do such inference. In such a way that the released data is responsible to disclose.

1.3 Contributions

Given the research question stated above in Section 1.2, we can concretely list the contributions of this thesis as follows:

- A comparison of privacy-measures as loss functions on release systems. As a result, we contribute with insights on maximal leakage, alpha-leakage and maximal alpha-leakage when used as loss function in a Generative Adversarial Network setting.
- Showing the difficulty of reconstructing the results presented in [14] using their paper. Supported by: open source code, topology designs and pseudo-code algorithms. It appears that every detail is important when working with Generative Adversarial Networks. We find that, without a clear overview of the regularization methods, learning parameters, a visual representation of the network topologies and shared code, important details concerning Generative Adversarial Networks may remain absent. Making reconstructions a challenging task.

- A textbook approach to the theory needed to implement privacy-preserving release mechanisms, with the goal to reach and approach a wider audience. In Chapter 2, an overview is provided with the most important theory needed to create a privacy-preserving release mechanism. The goal is to have written down the theory in such a way that computer scientist should be able to follow the theory without external resources.
- A generalization of [14], such that one can adapt to any loss function concerning the privacy-utility trade-off.
- Showing the practical relationship between the existing privacy measures using the results of the trained release mechanisms. In particular, we show the relationship between the alpha-loss with $\alpha = 0$ and Mutual Information. Further, we show the identical relation for larger values of alpha, e.g. $\alpha > 100$, with Maximal Leakage.
- The start of a library which makes this thesis extendable for other researchers (on top of the pytorch framework)[1].

1.4 Outline

The subsequent chapters will revolve around equations (1.1) and (1.2). Chapter 2 will contain the background knowledge needed to understand the coming chapters. It includes the notation used throughout this paper, the well-known notions of privacy and more details on privacy-leakage measures, distortion measures and the working of Generative Adversarial Networks. In Chapter 3, we will be measuring the performance of the mentioned privacy-leakage measures. To do so, we have identified different testing scenarios, along with a methodology to validate their results. In Chapter 4 we focus on the reproduction of the paper of [14]. We will implement the framework as explained, and test its results to our experiments. The remaining privacy-leakage measures Maximal Leakage and Alpha-Loss are identically implemented and validated in Chapter 5. Finally, we will end with a discussion and conclusion written in Chapter 6 and Chapter 7.

[1]github.com/maxxiefjv/privpack [15]

Chapter 2

Background and Related Work

This chapter is used to describe the building blocks for this thesis. Doing so, it contains the notation used throughout this thesis and the notions and intuition on privacy. Most of this chapter, however, is dedicated to explaining alternative measures to be used in (1.1); other than mutual information.

2.1 Notation

In this section, to simplify the understanding of context, we will give global meaning to those variables used identically throughout this document. Other variables, which are used in functions but regarded as context-dependent, will be introduced at that point.

As mentioned in Section 1.1, in this document, we consider the privacy-utility problem using the random variables:

$$(X, Y) \leftrightarrow W \leftrightarrow Z. \tag{2.1}$$

As shown in this Markov chain, we have three states, where: W is the observed data, X is our private attribute set, Y is our public attribute set and Z is the privacy-preserved release of W. The definition of the Markov chain provides us with an intuitive understanding. The sequence of states shows that the observed data W is the result of a transformation on the private and public variables X and Y. For example, the transformation which obtains W from (X, Y) can be concatenation, e.g. medical records consisting of diseases (Y) and Social Security Numbers (X). A less trivial transformation of getting W from (X, Y) could be the construction of images with faces. For example, such an image its private variable X can be a person's identity and the public variable Y can be the remaining information contained in the image (such as: BMI, age, ethnicity and other relevant characteristics of the person). A transformation on W consequently results in our released variable Z which preserves the person's characteristics but hides its identity. In the context of this paper, the goal is to create a release mechanism that produces the output Z according to the optimal privacy-utility trade-off.

Formally speaking, the Markov chain tells us that there are three probability distributions: $P_{X,Y}$, $P_{W|X,Y}$, $P_{Z|W}$. The probability of X and Y, the probability of W given X and Y, and the probability of Z given W, respectively. Showing the probabilities of the state transformations. The first two combined, results in $P_{X,Y}P_{W|X,Y} = P_{W,X,Y}$, and thus that the variables W, X, Y are jointly distributed according to a data model $P_{W,X,Y}$ over the space $\mathcal{W} \times \mathcal{X} \times \mathcal{Y}$. $P_{Z|W}$ our release mechanism that produces the release Z, specified by $P_{Z|W}$ with $(W, X, Y, Z) \sim P_{W,X,Y}P_{Z|W}$. Thus, we have the Markov chain as specified in (2.1). One should note, in practice, we never have access to these probability distributions. Rather, we will have a set of samples as training, test or validation data of which we know that: $(w_i, x_i, y_i)_{i=1}^n \sim P_{W,X,Y}$. Here n is the number of samples in our corresponding data-set. Finally, this data is used to infer the variational approximation of $P_{Z|W}$, i.e. the optimal privacy-preserving release mechanism.

2.2 Notions of Privacy

By designing a privacy preserving release system, one should be mindful about the possible knowledge adversaries may possess. In this section we formalize what it means for data to preserve privacy, from a theoretical perspective: on perfect privacy, Bayes-Optimal Privacy as well as a more practical bound: Linkage inequality.

2.2.1 Perfect Privacy

The best possible privacy is called perfect privacy, and is achieved if exactly no new information is revealed when releasing the data Z:

$$P(X = x|Z = z) = P(X = x), \forall z \in Z, \tag{2.2}$$

where new information indicates any information on X which can be revealed but was not yet revealed. This shows that the probability of our private variable '$X = x$' does not change when Z is known. Generalizing this to the full set X results will state that Z does not carry any information on X, i.e. $I(X; Z) = 0$ (as Mutual Information explained in Section 2.3.1). However, our newly introduced variable Z should also retain some utility of the original data. Otherwise, our new variable could be all noise, preserving all privacy but be of no use to anyone. This brings us back to the privacy-funnel as introduced in (1.1), and shows that $I(X; Z) = 0$ is usually not achievable while also maintaining the utility of the original data.

2.2.2 Bayes-Optimal Privacy

As utility is a constraint for the optimizations to come, this paper requires a more subtle definition of privacy than perfect privacy. Bayes-Optimal privacy as defined in [10] is such an alternative. However, while it provides us good intuition on privacy, it does not have practical benefit for our application. Still, this definition is important to be mindful when considering the definition of privacy.

The foundation of this privacy definition uses the uninformative principle. This principle states that the released data should provide the adversary with little to no additional information beyond the background knowledge on the sensitive attributes. To measure the realization of such principle we need to define the prior and posterior probability.

Principle 1 *Uninformative Principle: The published table should provide the adversary with little additional information beyond the background knowledge. [10].*

The prior probability is defined by $\alpha(x, y)$, and is intuitively defined as the probability of private variable $X = x$ given public variable $Y = y$ prior to observing the released data. The posterior probability is the probability when the released data is observed, defined by $\beta(x, y, Z)$.

Furthermore, this principle defines two types of disclosures: positive disclosures and negative disclosures. Positive disclosures tell that the adversary can either identify the sensitive attribute with high probability: $\beta(x, y, Z) > 1-\delta$, for some small threshold δ. Or, for Negative disclosures, the adversary can eliminate a sensitive attribute with high probability $\beta(x, y, Z) < \epsilon$, for some threshold ϵ. However, the insight here is that not all disclosures mean that this was due to an information leak in the released data-set. Rather, information leakage is about differences in positive and negative disclosures given the prior and posterior probabilities. Formally defined, this is the uninformative principle. As an example, Let a prior probability $\alpha(x, y) = 0.8$ and $\delta = 0.3$, so that the probability of our sensitive and identifying attribute is already very likely, and thus the adversary can already be very certain about the values of x and y. Now suppose that $\beta(x, y, Z) = \alpha(x, y) = 0.8$. Again, the adversary is very confident about the values of x and y. In fact, according to our definition of positive disclosures, we have that $\beta(x, y, Z) = 0.8 > 1 - 0.3$. However, these values have not taught the adversary any additional knowledge. Therefore, this is not due to an information leak.

One of the ways to instantiate the uninformative principle is using (ρ_1, ρ_2)-privacy breach definition. Let ρ_1 be a probability representing to the intuitive notion of 'very unlikely' and let ρ_2 correspond to the intuitive notion of 'likely'. This definition states that when: $\alpha(x, y) < \rho_1 \wedge \beta(x, y, Z) > \rho_2$ or $\alpha(x, y) > 1 - \rho_1 \wedge \beta(x, y, Z) < 1-\rho_2$, privacy is breached [4]. However, other instantiations that bound the difference between the prior and posterior definition in a Bayes-Optimal way also suffice to make the uninformative principle measurable [10].

2.2.3 Linkage inequality

The problem with Bayes-optimal privacy is that it is unlikely that the data publisher is aware of the full distribution, needed to compute the posterior $\beta(y, x, Z)$. As an alternative, the linkage inequality bound can be used to provide the necessary privacy guarantees. Given a privacy-leakage measure $J(X; Z)$, the data-processing inequality states that if and only if for any $A \leftrightarrow B \leftrightarrow C$, e.g. $(X, Y) \leftrightarrow W \leftrightarrow Z$, that form a Markov chain, we have that $J(A; B) \geq J(A; C)$. In other words, the information of A projected onto B is always more than the information of A projected onto C. Thus processing A can only result in information losses, given the satisfaction of a Markov chain. The linkage inequality, however, states that if and only if for any $A \leftrightarrow B \leftrightarrow C$ that form a Markov chain, we have that $J(B; C) \geq J(A; C)$.

Definition 1 *Linkage Inequality: if and only if for any $A \leftrightarrow B \leftrightarrow C$ that form a Markov chain, we have that $J(B; C) \geq J(A; C)$. [17]*

This means that, in the context of this paper, if there exists some secondary sensitive variable U which is transformable to X, then, for a Markov chain $U \leftrightarrow (X, Y) \leftrightarrow W \leftrightarrow Z$, the information of U projected onto Z cannot be more than the information of X projected onto Z. Thus, if the linkage inequality is satisfied, by transforming our X to Z we no longer need to account for some secondary sensitive variable U, as it is not possible for U to project more information onto Z than X. This results in practical bounds. This, for example, is useful when there may be unforeseen sensitive data correlated to our private variable X. Any privacy guarantees provided for our private variable X, will be at least as valid for the unforeseen sensitive data [17]. For symmetric privacy-leakage measures the linkage-inequality concept is identical to the data-processing inequality. However, for asymmetric privacy measures, this property is a distinct concept. [17]

2.3 Measuring Privacy Leakage

In section Section 2.2 we discussed what one may hope for in terms of privacy when transforming the data. Although this gives us practical privacy bounds, these statements cannot be used for optimization. Where Bayes-Optimal Privacy is intractable to compute, Linkage inequality is not more than a bound. In this section we investigate some information measures which can be used as objective functions. We will not yet be able to estimate if any transformation will be obeying the uninformative principle using one of these measures, or even Bayes-Optimal Privacy. However, the properties of the functions defined below can already inform us whether it is satisfying the linkage inequality defined in Section 2.2.3, which protects us from leaking information on unforeseen sensitive data.

The notation used in this section is identical to the one described in Section 2.1. This means we have the observed data W consisting of private variable X and public variable Y, i.e. $(X, Y) \leftrightarrow W$. Finally, after some transformation on W we obtain Z, $T(W) = Z$. For simplicity, in the subsections below we focus on privacy leakage only. Although these functions may also apply to information leakage in general, we will not be generalizing these formulas to include, possibly intended, leakages of Y on Z albeit possible.

Finally, all leakage measures have a corresponding loss function. Intuitively speaking, the loss functions are necessary to let machine learning models know the 'right' and 'wrong' solution. Ultimately, this allows the machine learning model to know how to preserve privacy in released data.

2.3.1 Mutual Information

Mutual information is a tool to compute the amount of information which is shared between two variables. It is used in the literature on privacy preservation in data to measure the remaining utility and privacy after some transformation. This is computed as follows, for two random variables X and Z:

$$I(X;Z) \triangleq H(X) - H(X|Z). \tag{2.3}$$

Here H(X) is the Shannon entropy of X, and $H(X|Z)$ is the conditional entropy of X given Z. For discrete variables, the Shannon entropy function returns the number of bits needed to describe the variable X. For H(X) this means that the more values X can take, the more 'surprised' one is for some value of X, and thus H(X) will be higher. For conditional entropy $H(X|Z)$, we see the 'surprise' of X to dependent on the given value Z. Therefore, the more information Z caries on X, the less surprised one will be about the output value. For example, if $X = Z$ we know everything about X when we know Z, and thus $H(X|Z) = 0$ gives that $I(X;Z) = H(X)$. Conversely, if Z carries no information on X, $H(X|Z) = H(X)$, and thus $I(X;Z) = H(X) - H(X) = 0$ (matching with our definition of perfect-privacy Section 2.2.1).

Intuitively, mutual information states the amount of information known of X given the variable Z and vice versa. In [14], it is shown that the corresponding loss function to minimize the mutual information in machine learning tasks is the log-loss [9]. Stated in Definition 2.

Definition 2 *The loss function corresponding to mutual information is defined by the log-loss function:*

$$J^{MI}(X;Z) = \log \frac{1}{P_{X|Z}(x|z)}. \tag{2.4}$$

FIGURE 2.1: Mutual Information loss function (blue), and its derivative (orange).

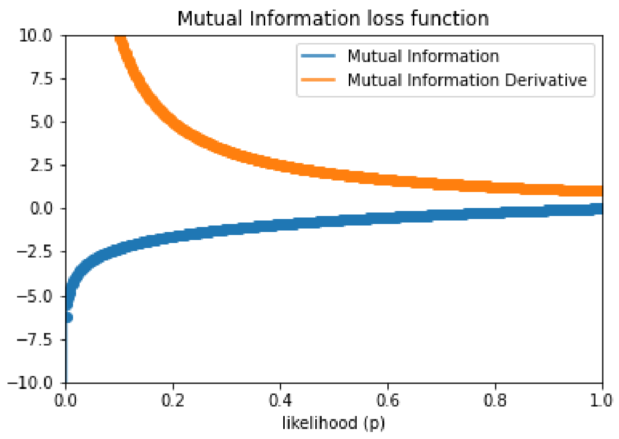
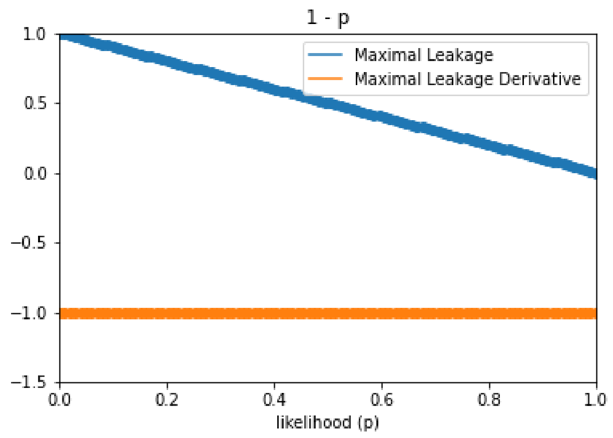


FIGURE 2.2: Maximal Leakage loss function (blue), and its derivative (orange).



Minimizing this function is identical to maximizing the mutual information between X and Z. This loss function, and its derivative are displayed in Figure 2.1.

However, as we will elaborate in Section 2.3.2, this measure is made with the entropy measure in mind. Entropy concerns itself about the minimal amount of bits needed to describe the data, and may therefore not be the optimal starting point. Another thing to note is that it satisfies linkage inequality and data-processing inequality [9].

2.3.2 Maximal Leakage

In [14] it is shown that the log-loss is the corresponding loss function of mutual information. Using Mutual Information, however, may not be the optimal strategy as Shannon's metric is created with another question in mind, namely, the minimum number of bits required to describe the data. It, thus, does not consider the information of some data leaked about its generator function. A metric created for this purpose is described by [8] and is called Maximal Leakage (MaxL). This measure considers all possible functions that may have produced our private variable X, noted by the space U. Intuitively speaking, this increased space gives a lot more reason to believe privacy-sensitive information has been leaked. In turn, the privacy measure appears to target privacy more careful, and therefore, expectedly, results in a better privacy-utility trade-off.

It is shown that incorporation of having a space U of possible functions that generate X leads to Maximal Leakage $\mathcal{L}_{maxL}(X \to Y)$ as information measure, and is defined as follows (Using identical notation as in Section 2.3.1):

$$\mathcal{L}_{maxL}(X \to Z) \triangleq \sup_{U-X-Z} \log \frac{\max_{P_{\hat{U}|Z}} \mathbb{E}[P_{\hat{U}|Z}(U|Z)]}{\max_U P_U(u)}. \tag{2.5}$$

For finite alphabets X and Z this simplifies to:

$$\mathcal{L}(X \to Z) = \log \sum_{z \in \mathcal{Z}} \max_{x \in \mathcal{X}: P_X(x) > 0} P_{Z|X}(z|x). \tag{2.6}$$

In [9] the corresponding loss function to use in Machine Learning tasks is shown to be the probability of error. Stated in Definition 3.

Definition 3 *The loss function derived from maximal leakage is equal to the probability of error:*

$$J^{MaxL}(X; Z) = 1 - P_{X|Z}(x|z). \tag{2.7}$$

This linear loss function is displayed in Figure 2.2.

It is proven in [17], that this measure satisfies the linkage inequality and data-processing inequality.

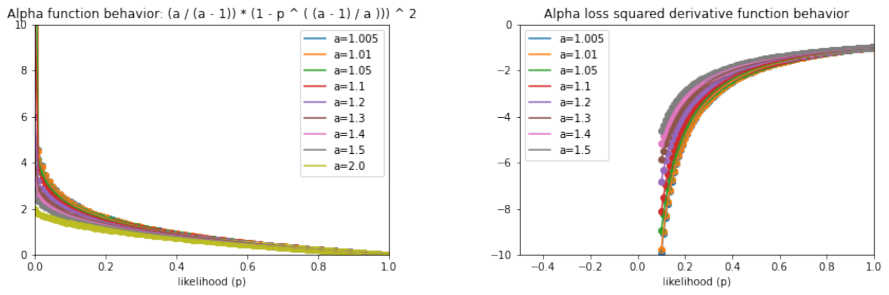
2.3.3 α-leakage and Maximal α-leakage

The last measurement to discuss is proposed in [9], and are the leakage measures α-leakage and Maximal α-Leakage. Below we will give some general information on the interpretation of these measures and their formula's. Finally, like the previous measures, it ends with the loss function that corresponds to these measures, which is the α-loss for both measures.

These measures are the results of generalizing the loss functions of Section 2.3.1 and Section 2.3.2. The α constant here is a tunable parameter between the mutual information measure, for $\alpha = 1$, and the maximal leakage measure, for $\alpha = \infty$ (as shown in [9]). Changing our α parameter may therefore allow us to achieve a more fine-tuned privacy-utility trade-off.

In [9], they first study tunable leakage measures which can measure the inference gain on a *specific* function U from the released data Z (2.8). In this function U is treated as a known function which generates X:

$$\mathcal{L}_\alpha(X \to Z) \triangleq \frac{\alpha}{\alpha - 1} \log \frac{\max_{P_{\hat{X}|Z}} \mathbb{E}[P_{\hat{X}|Z}(X|Z)^{\frac{\alpha-1}{\alpha}}]}{\max_{P_{\hat{X}}} \mathbb{E}[P_{\hat{X}}(X)^{\frac{\alpha-1}{\alpha}}]}. \tag{2.8}$$



(A) Alpha-loss function (B) Alpha-loss Derivative function

FIGURE 2.3: Alpha-loss and alpha-loss derivative. Clearly indicating its numeric sensitivity to values of α.

It is also shown to simplify to the Arimoto Mutual Information $\mathcal{L}_\alpha(X \to Z) = I_\alpha^A(X;Z)$[1] for $1 \leq \alpha \leq \infty$ by [9]. However, they also wish to measure the inference gain on any arbitrary attribute. For that purpose, Maximal α-Leakage is defined (2.9).

$$\mathcal{L}_\alpha^{max}(X \to Z) \triangleq \sup_{U-X-Z} \mathcal{L}_\alpha(U;Z). \tag{2.9}$$

As noted by [9], both privacy leakage measures: α-leakage and Maximal α-leakage, are related by the function defined in Definition 4.

Definition 4 *The alpha loss is created to have a tunable parameter α which moves from the mutual information ($\alpha = 1$) to maximal leakage ($\alpha = \infty$), and is defined by:*

$$J^\alpha(X;Z) = \frac{\alpha}{\alpha - 1}(1 - P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}}). \tag{2.10}$$

The minimization of the loss function defined in Definition 4 implies the optimal decision by an adversary. Furthermore, this loss is designed to satisfy the following equalities:

$$J^\alpha(X;Z) = \begin{cases} J^{MI}(X;Z), & \alpha = 1. \\ J^{MaxL}(X;Z), & \alpha = \infty. \\ \frac{\alpha}{\alpha-1}(1 - P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}}), & 1 < \alpha < \infty. \end{cases} \tag{2.11}$$

As depicted in Figure 2.3, for various values of alpha this loss function is a lot more prone to numerical issues than the previous loss functions.

Finally, the properties of this function show that it satisfies the linkage inequality and post-processing inequality.

2.4 Measuring Utility and Distortion

Just as for measuring privacy-leakage we lay-out measures to determine the distortion or utility of the released data Z. In practice, however, the distortion measure is very application dependent. Indeed, as general approach, one might attempt to keep as much information of Y in the released set Z. This is done by using mutual information (I(Y;Z)) as utility measure, like in the privacy-funnel (1.1). However, clearly, one might be able to make a stronger constraint. Possibly in such a way that we can remove more privacy-sensitive data without the cost of additional distortion. Rather

[1]For more information on Arimoto Mutual Information see [16]

than 'maintaining as much information as possible', one might attempt to maintain concrete and measurable parts of Y in Z.

In this section we therefore limit ourselves to the distortion measures relevant for this paper (used in Chapter 3), in practice one would have to audit the best distortion measure for their application. Note that, these distortion measures, are to be replaced in formulas where the distortion measures are notated by d(y, z). All these distortion measures are defined on single sample basis, i.e. defined for sample y_i and its released counterpart z_i.

2.4.1 Hamming Distance

To measure distortion for discrete variables, we will be using hamming distance as defined in (2.12). It simply counts the number of coordinates in which two binary vectors differ. It is used in information theory, coding theory and cryptography:

$$\text{hamming\_distance}(y_i, z_i) = \sum_{j}^{m} (y_i(j) \neq z_i(j))\mathbb{1}, \tag{2.12}$$

where m is the number of dimensions of y_i and z_i. We will use this measure for our bivariate binary experiments, where $m = 1$.

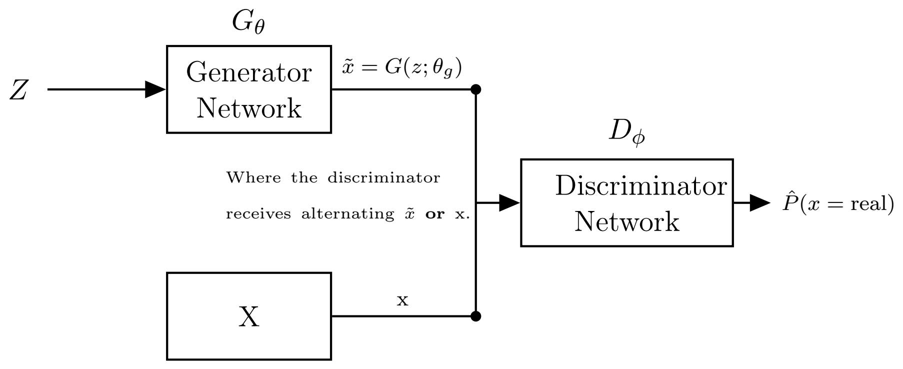
2.4.2 Mean Squared Error

For our multivariate Gaussian experiments we use the Mean Squared Error (MSE) measure, which, identically to hamming distance, goes to zero if the two given sets are equal [6]. It is given by:

$$\text{MSE}(y_i, z_i) = \frac{1}{m}||z_i - y_i||_2^2. \tag{2.13}$$

2.5 Generative Adversarial Networks

FIGURE 2.4: Visual representation of Generative Adversarial Network.


G_θ
Z
Generator Network
$\tilde{x} = G(z; \theta_g)$
D_ϕ
Where the discriminator receives alternating $\tilde{x}$ or x.
Discriminator Network
$\hat{P}(x = \text{real})$
X
x

The Generative Adversarial System is originally used to estimate and sample from the probability distribution of some data set. All methods discussed in this section are discussed with the application of GANs, short for Generative Adversarial Networks, [5] in mind. In [5] a composite of Neural Networks is defined in which two networks are trained: Generator network and Discriminator network, depicted in Figure 2.4. The generator network is trained to capture the data distribution of the used data, where the discriminator network is trained to discriminate a fake sample from a sample in the actual data. The training of these networks is done in a minimax fashion, specifically, to learn the distribution p_x of some data $\mathbf{x}$. A prior on input noise variables is defined $p_z(z)$ which generates the input noise Z of the generator network. Consequently, a mapping is represented as $\tilde{x} = G(z;\theta)$, where $\tilde{x}$ is the generated output based on the network parameters θ and the input noise Z. The discriminator network takes as input either the output of $G(z;\theta)$ or an original $\mathbf{x}$ and outputs a scalar value between zero and one. This value indicates the estimated probability of being a real sample, $\hat{P}(x = \text{real};\phi)$ or $\hat{P}(\hat{x} = \text{fake};\phi)$. Both of these networks undergo a training process to optimize the results of the generator network. Doing so the discriminator network is trained to maximize the probability of assigning the correct label to both real and fake samples., i.e. D(x) = 1 and D(G(z)) = 0. Simultaneously, we train the generator network to minimize the discriminator's success of correctly guessing the generator's output as fake, i.e. we minimize log(1 - D(G(z))). This can be formalized in the following minimax game [5]:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.14}$$

In Chapter 3 we investigate how to combine Generative Adversarial Networks and privacy-leakage measures to create a privacy-preserving release mechanism.

Chapter 3

Experiments for measuring performance of privacy-leakage

In the context of the privacy-utility trade-off, we have a similar approach to GANs. We introduce how GANs can be used in a privacy-preserving approach. Similar to [14], we will outline a framework which focuses on the optimization of privacy given a utility constraint. We will not restrict ourselves to using Mutual Information as privacy-leakage measure (further elaborated in Chapter 4). Instead, we will be using all the privacy measures defined in Section 2.3.

Identically to the GAN design, the network composition consists of two networks, one we call the privatizer and one we call the adversary. Instead of generating seemingly real samples, the goal of this design is to 'privatize' existing samples. This new goal requires a new loss function, one that measures privacy, as described in Section 2.3. This design is similar to the minimax design proposed by [5]. However, as discussed in the introduction, it allows for the trivial solution of outputting just noise by the privatizer. Outputting just noise is not useful, and therefore a problem. To address this problem utility is included in the loss function of the privatizer. Incorporating utility, the generalized privacy-funnel result in:

$$\min_{P_{Z|W}:(X,Y)\leftrightarrow W\leftrightarrow Z} J(X;Z), \text{ s.t. } \mathbb{E}[d(Y,Z)] \leq \delta. \tag{3.1}$$

This is identical to Tripathy et al., however, generalized to J(X;Z), which is one of the privacy-leakage measure as defined in Section 2.3. The distortion is defined by $d(Y, Z)$. And the constant δ specifies the distortion constraint for the privacy-utility trade-off.

This constrained optimization problem can be transformed to an unconstrained privacy-utility problem [14], as is shown in Chapter 4. We will be using this derivation and generalize the log-loss to any privacy-leakage related loss function. This unconstrained optimization problem then results in:

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[J(X;Z)] + \lambda\,\mathbb{E}[d(Y,Z)]. \tag{3.2}$$

Adding the delta-constraint into the formula of (3.1), the formula can be written as:

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[J(X;Z)] + \lambda \max(0, \mathbb{E}[d(Y,Z)] - \delta)^2. \tag{3.3}$$

A visual representation of this design is depicted in Figure 3.1, and shows how the losses for each network is computed.

Notable in this figure is that both the privatizer and adversary concern themselves over the privacy leakage as this is exactly what defines their minimax game. The

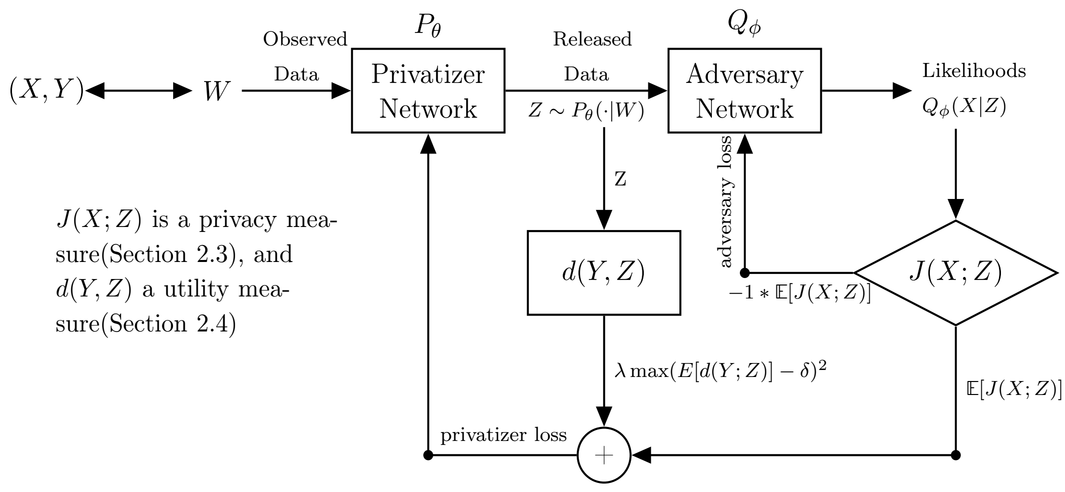

P_θ
Q_ϕ
(X,Y)
W
Observed Data
Privatizer Network
Released Data
$Z \sim P_\theta(\cdot|W)$
Adversary Network
Likelihoods $Q_\phi(X|Z)$
Z
adversary loss
$J(X;Z)$ is a privacy measure(Section 2.3), and $d(Y,Z)$ a utility measure(Section 2.4)
$d(Y,Z)$
$-1 * \mathbb{E}[J(X;Z)]$
$J(X;Z)$
$\lambda \max(E[d(Y;Z)] - \delta)^2$
$\mathbb{E}[J(X;Z)]$
privatizer loss
+

FIGURE 3.1: Data flow used in the training process of a privatizer design originated from a Generative Adversarial approach.

adversary is updated only using the adversary loss, and the privatizer is updated only using the privatizer loss. The idea behind the minimax game is that the adversary improves the privatizer and vice versa. For this reason, both networks update their weights according to the privacy leakage. Only, both networks use a different interpretation. A good guess by the Adversary is considered “bad” for the privatizer, while it is considered “good” for the adversary itself.

As mentioned in Sections 1.1-1.3, one of the concerns of this study is to measure how the privacy is best maintained while retaining the utility of the data. As utility is very application dependent, the privacy-measure doesn't have to be, i.e. the diamond node in Figure 3.1. In this chapter we conduct three types of experiments for each privacy-leakage measure defined in Section 2.3. This means that every node displayed in Figure 3.1 is defined, and will be discussed for each experiment. Except for the diamond node, which is dependent on the privacy-leakage measure used in the system.

3.1 Experiment setup

To compare the performances of each GAN-inspired Privacy preserving release mechanism described in the literature, a set of experiments has been designed that can be categorized in two types. Both types of experiments have a theoretical foundation, and are described in Section 3.2 and 3.3. The types of data will determine a few factors: the internal network architectures, the distortion measure and whether a loss approximator is needed. Furthermore, within each type of data (Bivariate Binary or Multivariate Gaussian), we created sub experiments which vary in their distribution parameters. For each of those sub experiments we will be defining several utility constraints, defined by the λ and δ constants. Finally, besides the previously mentioned factors, some experiment independent settings, the hyper-parameters, will determine the final outcome of the experiment. For each experiment we have summarized these values in table format. Each entry in this table is clarified in Table 3.1. The privacy-leakage measure is a factor that will be added and elaborated in the corresponding chapters, i.e. Chapter 4 and 5.

The validation and evaluation methodology are described in Section 3.4. This will clarify two things: how the model is validated, and how the results are evaluated.

TABLE 3.1: Description of the values used to summarize an experiment.

Value	**Description**
Data Type	Bivariate Binary or Gaussian data
Distribution Parameters	The distribution parameters or reference to.
(λ, δ)	The utility objective constants
No. Samples	The number of data-samples generated/used
No. Dimensions	The number of dimensions of one data-sample in the data-set
No. Epochs	The number of epochs used to train the network
Batch size	The number of data-samples used per batch
Train/Test ratio	The ratio of samples used for training and testing.
Network architecture	The architecture function used to as the privatizer and adversary network.
Distortion: $d(Y, Z)$	The distortion function used within the optimization process of the network.
Privacy: $J(X; Z)$	The privacy measure used to update the privacy handling of the network.
Loss approximation used	Is the loss function computationally tractable, or does the experiment require the use of the universal approximator approach as defined by Tripathy et al.
Weight Optimizer	The optimizer used to update the weights of the networks. This defines how much of the current gradient is used to update the network weights.

3.2 Synthetic Bivariate Binary Data

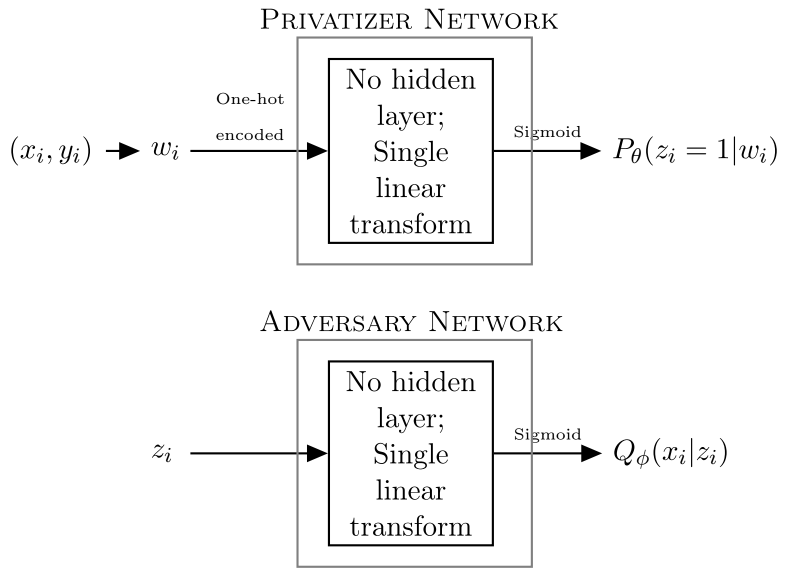
The first theoretical experiment is constructed to test the basic capabilities of the architectures defined in the literature. Consequently, the results produced are easy to interpret and verify. The experiment includes training the network with data sampled from a bivariate binary distribution. The first part of the experiment factors are depicted in Table 3.2. This table contains the factors which are identical for each sub experiment of this section. The subsections below will add one of the missing values: Distribution Parameters.

Furthermore, as mentioned in this table, the privatizer and adversary network behavior is defined in Figure 3.2.

TABLE 3.2: The architecture values used for the Bivariate Binary Data experiment.

Data Type	Bivariate Binary
(λ, δ)	$\lambda = 500, \delta = [0, 0.1, ..., 1]$
No. Samples	10000
No. Dimensions	2
No. Epochs	200
Batch size	200
Train/Test ratio	0.8/0.2
Network architecture	Figure 3.2
Distortion: $d(Y, Z)$	(2.12) (Hamming Distance)
Expected Loss approximation used	None.
Weight Optimizer	Adam optimizer: $\alpha = 0.001$, $\beta(0.9, 0.999)$, $\epsilon = 1e-08$.

FIGURE 3.2: Synthetic Bivariate Binary Privacy-Preserving Architecture.



The design of the networks used is shown in Figure 3.2. This architecture is completely noise free. This means no randomness is included in any part of the network. We therefore define W as the only input, and the loss function is computed through a sum over the possible outputs of Z, i.e. the sample-wise loss function further explained in (4.5). The output of our network P_θ is considered as the probability of outputting $z = 1$. For the adversary we will have the direct output of P_θ as input, which is the probability of P_θ outputting a one.

To understand the behavior of each network architecture, we defined three distributions to sample data from: completely correlated (Table 3.5), completely uncorrelated (Table 3.3) and random (Table 3.7). Within each table, the probabilities define the dependence between the X and Y variable and thus their chance of occurring in combination.

3.2.1 Uncorrelated distribution

The first distribution we defined is completely uncorrelated between the X and Y variables. These distribution parameters are shown in Table 3.3. Using a seed of 0, we obtained a data-set with 4016 zero samples (50.2%) and 3984 one samples (49.8%) in the training set. The test set includes 987 zero samples (49.35%) and 1013 one samples (50.65%).

TABLE 3.3: This table contains a completely **uncorrelated** probability distribution used for generating the bivariate binary data.

$P(X = x, Y = y)$	$Y = 0$	$Y = 1$
$X = 0$	0.25	0.25
$X = 1$	0.25	0.25

TABLE 3.4: Uncorrelated Data Statistics:

Statistic Type	**Value**
I(X;Y)	0
Train Y-samples Ratio (0s, 1s)	(50.2%, 49.8%)
Test Y-samples Ratio (0s, 1s)	(49.35%, 50.65%)
$\hat{I}_{train}(X;Y)$	1.79686e-05

3.2.2 Correlated distribution

This distribution parameters (Shown in Table 3.5) show that the expected mutual information of any data set generated with this distribution is 1. Furthermore, using a seed of 0, we have 4058 zero samples (50.725%) and 3942 one samples (49.275%) in the training set. The test set includes 1006 zero samples (50.3%) and 994 one samples (49.7%).

TABLE 3.5: This table contains a completely **correlated** probability distribution used for generating the bivariate binary data.

$P(X = y, Y = y)$	$Y = 0$	$Y = 1$
$X = 0$	0.5	0
$X = 1$	0	0.5

TABLE 3.6: Correlated Data Statistics:

Statistic Type	**Value**
I(X;Y)	1
Train Y-samples Ratio (0s, 1s)	(50.725%, 49.275%)
Test Y-samples Ratio (0s, 1s)	(50.3%, 49.7%)
$\hat{I}_{train}(X;Y)$	0.99993

TABLE 3.7: This table contains a **random** probability distribution used for generating the bivariate binary data.

$P(X = x, Y = y)$	$Y = 0$	$Y = 1$
$X = 0$	0.2275677	0.29655611
$X = 1$	0.24993822	0.22593797

3.2.3 Random distribution

To be able to reproduce the seed '0' is used. This gives the distribution parameters shown in Table 3.7, and results in an $I(X;Y) = 0.004146702663409007$. This shows that there is not a strong relation between the variables X and Y. This leads to the expectation that a lot of privacy could be maintained without a lot of loss in usability. Concretely, the useful variable Y hardly has to change to keep the sensitive variable X private.

TABLE 3.8: Random Data Statistics:

Statistic Type	**Value**
I(X;Y)	0.004146702663409007
Train Y-samples Ratio (0s, 1s)	(47.7125%, 52.2875%)
Test Y-samples Ratio (0s, 1s)	(47.85%, 52.15%)
$\hat{I}_{train}(X;Y)$	0.00386

The actual Mutual Information of the above probability distribution is computed using the formula:

$$I(X;Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} P_{X,Y}(x,y) * \log \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)} = H(X) - H(X|Y). \tag{3.4}$$

As the output Z of the privatizer network is only a sample of the actual space, we can only compute estimates of mutual information. Therefore, we also computed the estimated value of mutual information, i.e. $\hat{I}(X;Y)$, by estimating the probability distribution $P(X,Y)$ through counting and using the formula described in (3.4). Using our seed, it should be noted that we generated a data-set that consists of 3817 zero samples (47.7125%), and 4183 one samples(52.2875%). The test data consists of 957 zero samples (47.85%), and thus 1043 one samples (52.15%). This produces $\hat{I}_{train}(X;Y) = 0.00386$ as the estimated mutual information given the training data.

3.2.4 Expected Outcome of Experiments

Given the bivariate binary data-set and our goal to minimize privacy leakage, we can define expectations to use for verifying our results. Doing so we can use the formulas for entropy (3.5):

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}, \tag{3.5}$$

and discrete Mutual Information (3.4).

Each topology in these experiments use some input entry (W) consisting of a private part and public part (e.g. X and Y respectively) and outputs a privacy preserving version (Z). In each case we will attempt to minimize the privacy leakage of part Z on

X and the distortion of Y in Z. Exploring the extreme cases, we ideally have either complete privacy of X in Z or no distortion between Y and Z. This is analyzed below:

Exploring the zero-distortion case we need that:

$$Y = Z.$$

And thus, consequently:

$$I(X;Y) = I(X;Z).$$

In the case of perfect privacy we need that $I(X;Z) = 0$. However, this is only possible when $I(X;Y) = 0$. Consequently, zero-distortion and perfect privacy is only possible when X and Y are completely uncorrelated. Otherwise, there will be a Privacy-Utility trade-off. In turn this means that, whenever $I(X;Y) > 0$, the only solution for no privacy-leakage, when working with 100% correlation data, requires for the release variable to send out no information, i.e. $H(Z) = 0$. This means that Z is always outputting the same value Z=0 or Z=1:

$$(P(Z=0) = 1 \wedge P(Z=1) = 0) \vee (P(Z=0) = 0 \wedge P(Z=1) = 1).$$

Then we have that:

$$H(Z) = -\sum_{z} P(Z=z) \log P(Z=z) = 0.$$

The distortion is therefore expected to be equal to the zero/one-sample ratio. For the random case we get a distortion of 52.2875% or 47.7125% for the train set, depending on the output of Z=0 or Z=1 (respectively).

For any trade-off scenario, we allow some distortion and some privacy-leakage, the results will likely depend on the used loss-function. However, for this simple data-type we expect no big difference compared to the theoretical optimum.

3.3 Synthetic Multivariate Gaussian Mixture Data

Gaussian Mixture models have the potential to explain many complex distributions. It is, therefore, a more realistic data set type. Further, the theoretical foundations provides us the possibility to analyze. Within these experiments we defined two sub experiments. The first experiment will have only variables uncorrelated to any other variable, clarified in Section 3.3.1. The second sub experiment will have no dependence between all the private variables, nor will there be dependence between the public variables. This is further elaborated in subsection 3.3.2. Identically to the bivariate binary case, 10000 samples have been generated for each scenario. The training set includes 8000 of those samples, the remainder is used for the test set. Each of these sub experiments is executed within a 2-dimensional (1 private, 1 public variable) and 10-dimensional (5 private, 5 public) space. The experiment architecture summary is defined by Table 3.9.

Depending on the number of dimensions the network either looks like Figure 3.3 for 2-dimensional data or like Figure 3.4 for 10-dimensional data. Two design choices should be especially noted.

The first is the number of output neurons of the privatizer network. The output dimension is an assumption on the optimal permutation of the released data. Furthermore, in more practical cases, there may not be a clear distinction between private

TABLE 3.9: The architecture values used for the Multivariate Gaussian Data experiment.

Data Type	(Multivariate) Gaussian
λ^{scalar}	$\lambda = 50$
λ^{mv}	$\lambda = 10$
δ^{scalar}	$[0, 0.2, 0.4, 0.6, 0.8, 1, 2]$
δ^{mv}	$[0, 0.5, ..., 5]$
No. Samples	10000
No. Dimensions	2
No. Epochs	200
Batch size	200
Train/Test ratio	0.8/0.2
Network architecture	n = 10 → Figure 3.4, n = 2 → Figure 3.3
Distortion: $d(Y, Z)$	(2.13) (Mean Squared Error)
Privacy: $J(X; Z)$	Experiment instantiation dependent
Expected Loss approximation used	Universal Approximator as defined in [14]
Weight Optimizer	Adam optimizer: $\alpha = 0.001$, $\beta(0.9, 0.999)$, $\epsilon = 1e - 08$.

and public data. An example of this could be a face data-set. From this data-set researchers may want to perform many tasks (e.g. ethnicity-detection, BMI-detection etc.), however the identity of the persons in this data-set may be private. In this scenario the identity of the person is mingled in the face of a person.

The second notable design choice is that only the diagonal of the (conditional) covariance is estimated from the data. This design choice implies no dependence between the private variables, which is an assumption to note. However, as we know, is true for the experiments we are conducting.

FIGURE 3.3: Synthetic Multivariate Gaussian Privacy-Preserving Architecture 2-dimensional.

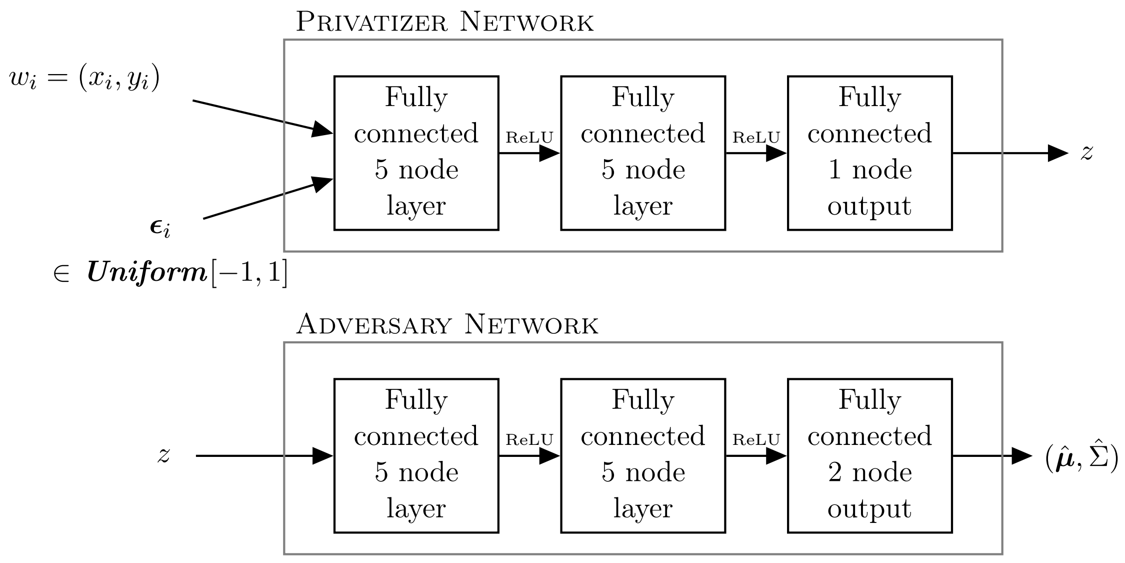
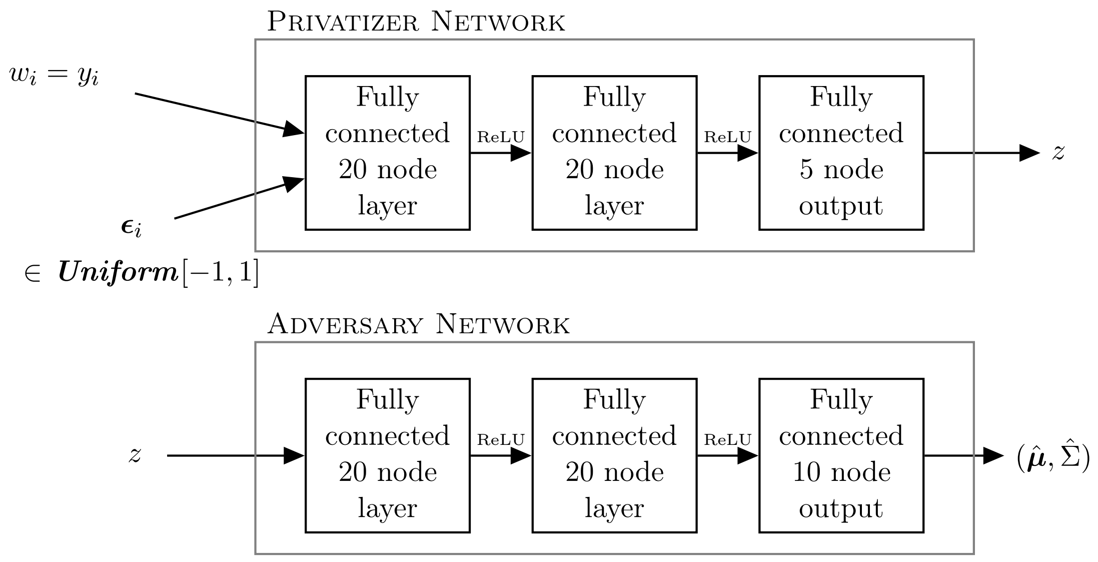

PRIVATIZER NETWORK
$w_i = (x_i, y_i)$
ϵ_i
$\in$ ***Uniform***$[-1, 1]$
Fully connected 5 node layer
ReLU
Fully connected 5 node layer
ReLU
Fully connected 1 node output
z
ADVERSARY NETWORK
z
Fully connected 5 node layer
ReLU
Fully connected 5 node layer
ReLU
Fully connected 2 node output
$(\hat{\boldsymbol{\mu}}, \hat{\Sigma})$

FIGURE 3.4: Synthetic Multivariate Gaussian Privacy-Preserving Architecture 10-dimensional.


PRIVATIZER NETWORK
$w_i = y_i$
ϵ_i
$\in$ ***Uniform***$[-1, 1]$
Fully connected 20 node layer
ReLU
Fully connected 20 node layer
ReLU
Fully connected 5 node output
z
ADVERSARY NETWORK
z
Fully connected 20 node layer
ReLU
Fully connected 20 node layer
ReLU
Fully connected 10 node output
$(\hat{\boldsymbol{\mu}}, \hat{\Sigma})$

3.3.1 Uncorrelated Gaussian Data.

Identically to the uncorrelated bivariate binary case, there are no relations between any of the variables. This gives us a multivariate gaussian distribution: $\mathcal{N}(\mu = \mathbf{0}, \Sigma = \boldsymbol{I}_n)$. To manage expectations we compute the actual mutual information, the actual entropy of Y and the estimated mutual information of the generated 8.000 training samples. Denoted in Table 3.10 and Table 3.11.

TABLE 3.10: Gaussian Uncorrelated Data Statistics, $n = 2$:

Statistic Type	**Value**
I(X;Y)	0
H(Y)	1.4189
$\hat{I}_{train}(X;Y)$	1.0133e−10

TABLE 3.11: Gaussian Uncorrelated Data Statistics, $n = 10$:

Statistic Type	**Value**
I(X;Y)	0
H(Y)	7.0947
$\hat{I}_{train}(X;Y)$	0.0013

These statistics clearly indicate that any delta-constraint (3.1) should be achievable without substantial privacy costs.

3.3.2 Correlated Gaussian Data.

This sub experiment uses the bivariate and multivariate Gaussian parameters as identically described in Tripathy et al. two of their continuous example [14]. In this bivariate scalar example a gaussian model is defined by $\mathcal{N}(\mu, \Sigma)$, with parameters:

- $\mu = \mathbf{0}$,
- $\Sigma = \begin{bmatrix} 1 & 0.85 \\ 0.5 & 1 \end{bmatrix}$.

These parameters show the covariance between private X, and public Y to equal 0.85. And both variances equal to 1. Its data statistics are found in Table 3.12.

TABLE 3.12: Gaussian Related Data Statistics, $n = 2$:

Statistic Type	**Value**
I(X;Y)	0.6410
H(Y)	1.4189
$\hat{I}_{train}(X;Y)$	0.6498

In contrast to the uncorrelated example, this data will demand a bigger trade-off in privacy to obtain the required utility. Therefore, small delta-constraints require more privacy loss.

A more intricate example is their multivariate gaussian with gaussian model $\mathcal{N}(\mu, \Sigma)$, with parameters:

- $\mu = \mathbf{0}$,
- $\Sigma = \begin{bmatrix} I_5 & \text{diag}(\rho) \\ \text{diag}(p) & I_5 \end{bmatrix}$,
- where $\rho = [0.47, 0.24, 0.85, 0.07, 0.66]$.

Here, the first five variables are considered X, and thus private, and the last five variables are Y, and are the sensitive or 'useful' variables. Interpretation of this distribution concludes that the private variables are not dependent, and the same holds for the useful variables. However, the first private variable has some correlation with the first useful variable, this is denoted by ρ. Hence:

$$\forall_{i,j \in (0,5]} X_i \perp\!\!\!\perp Y_j \iff i \neq j$$

The data statistics that correspond to this data-type are displayed in Table 3.13:

TABLE 3.13: Gaussian Related Data Statistics, $n = 10$:

Statistic Type	**Value**
I(X;Y)	1.0839
H(Y)	7.0947
$\hat{I}_{train}(X;Y)$	1.1039

Finally, when considering the mutual information of this data distribution it is computed using the formula described in Appendix A.2.2. This results in the actual mutual information $I(X;Y) = 1.08389$. However, as described earlier, we will mostly be measuring the performance through estimates. Hence, we will be having the estimated Mutual Information given the generated data. Which is equal: $\hat{I}_{train}(X;Y) = 1.09896$.

Note that, the fact that private and public variables between themselves are independent is an assumption. This means that the results from this experiment tell us a lot about practical data in which this assumption holds.

3.4 Validation and Evaluation

For each of the earlier described experiments we always have two goals. One is to keep the released data as private as possible, and another to keep the released data as useful as possible, i.e. distortion of Z is minimal given Y as actual data. For each of the experiments above we will be measuring the privacy leakage using the formulas in Section 2.3 and estimators in Appendix A.1. Furthermore, the results are plotted using Mutual Information and Utility for each experiment. By doing so we will be able to compare and find a relation between the privacy-leakage functions through their loss functions. This can only happen if the results are stable, and the privacy-utility objectives should be identical. To indicate the stability we will show several runs, and the average results for all runs will be used for the comparison. To have identical training goals we shall specify the objective constants λ and δ to be identical for each Privacy-Leakage experiment.

Considering that we have explained mutual information not to be the best measure for privacy-leakage, the use of mutual information as validation and evaluation should be enlightened. Computing Maximal Leakage and Maximal α-leakage is an intractable operation, as the set of possible functions which can produce X is infinite. Although we can compute α-Leakage, which is also equal to the Arimoto Mutual information $I_\alpha^A(X;Z)$, it only provides us with information on the leakage of X onto Z for some *specific* generation function U. This does not tell us enough about the actual privacy-leakage of X onto Z.

Therefore, although we do not consider this as an optimal solution, in order to bring hard evidence, it is the best we can do, given the provided tools. Future work might be able to improve on this specific issue. A result that we can hope for using mutual information, is that each trained network results in a similar mutual information between X and Z but a better utility. This way we will be able to show that (Maximal) α-leakage measures as loss function result in a better privacy-utility trade-off.

Chapter 4

Reconstructing Privacy-Preserving Adversarial Networks.

To study the performance of mutual information we will follow the same architecture and approach as defined by [14]. In this section we will elaborate their approach on the problem. This chapter is purely meant as to reconstruct the results of Tripathy et al. Continuation of this work is found in Section 5.1. As this paper already described, when using mutual information as privacy leakage measure, i.e. J(X;Z) in (3.1), it results in the following constrained optimization problem (4.1):

$$\min_{P_{Z|W}:(X,Y)\leftrightarrow W\leftrightarrow Z} I(X;Z), \text{ s.t. } \mathbb{E}[d(Y,Z)] \leq \delta. \tag{4.1}$$

As shown in [14], this function can be rewritten to the unconstrained optimization problem identical to the minimax problem. The crucial insight is that:

$$-H(X|Z) = KL(P_{X|Z}||Q_{X|Z}) + \mathbb{E}[\log Q_{X|Z}(X|Z)].$$

The Kullback-Leibler divergence is denoted by $KL(\cdot||\cdot)$. Using this insight, the fact that $I(X;Z) = H(X) - H(X|Z)$ and by dropping the constant $H(X)$, (4.1) is transformed to a minimax formulation (similar to (2.14)):

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[\log Q_{X|Z}(X|Z)] + \lambda\, \mathbb{E}[d(Y,Z)]. \tag{4.2}$$

Here $P_{Z|W}(z|w) = \mathcal{N}(z; (\boldsymbol{\mu}, \boldsymbol{\Sigma}) = P_\theta)$, which implies that the privatizer network P, parameterized by θ, is trained to learn the parameters of the distribution $P_{Z|W}$. The same holds for $Q_{X|Z} = \mathcal{N}(x; (\boldsymbol{\mu}, \boldsymbol{\Sigma}) = Q_\phi)$, where the adversary network is parameterized by ϕ. In this case we assume the distributions to be a multivariate Gaussian. Although this type of distribution has the capacity to imitate more complex distributions, this will differ per experiment. In the bivariate binary case the distribution will be simpler, which is explained in Section 4.1. The distortion term in (4.2) has no explicit distortion constraint, however, as explained in (3.3), this can be replaced with its explicit alternative, which is the exact loss function used to optimize the privatizer network:

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[\log Q_{X|Z}(X|Z)] + \lambda \max(0, \mathbb{E}[d(Y,Z)] - \delta)^2. \tag{4.3}$$

Here we see that the mutual information term corresponds to our mutual information loss function (2.4):

$$\max_{Q_{X|Z}} \mathbb{E}[\log Q_{X|Z}(X|Z)] = \min_{Q_{X|Z}} \mathbb{E}[\log \frac{1}{Q_{X|Z}(X|Z)}] = \min_{Q_{X|Z}} \mathbb{E}[J^{MI}(X;Z)] \tag{4.4}$$

This function shows the minimization performed by the adversary. Having defined our privacy-leakage term of the loss function, we will, per experiment, define the full loss function in the following subsections.

4.1 Bivariate Binary Data using Mutual Information Release Mechanism

Recall that our first experiment is applying the architectures on binary bivariate data $W \leftrightarrow (X, Y)$. In our experiments we consider X to be private and Y to be public. Furthermore, as mentioned in Section 3.2, we generated three types of data: completely correlated, completely uncorrelated & random (i.e. barely correlated). Each of these distributions have a MI of: 1, 0 and 0.0041.

As shown in the loss function, there are two constants which define the focus of distortion term. The lambda parameter defines the focus on the term within the loss, and delta defines the maximal constraint we allow without penalty. This loss function is exactly used to train the privatizer network, its algorithm depicted in Algorithm 1 in Appendix B. For the adversary the loss function is similar. Yet, the principle is different. Instead of minimizing the privacy leakage, we now want to maximize it. As a result the adversary algorithm turns out as shown in Algorithm 2 in Appendix B, without distortion term, and the if statement concerning the current value of x, is reversed. In both algorithms, W_i is the ith entry of the dataset (including both X and Y), λ defines the extra unconstrained problem parameter as defined in (4.3) and δ defines the distortion constraint allowed by the network. The privatizer network defines the trained privatizer network so far (time step t). This algorithm is formally defined using Equation (4.5). [1]

$$\begin{aligned}\mathcal{L}^i_{bin}(\theta,\phi) &:= \sum_{z\in\mathcal{Z}}[P_\theta(z|w_i)\log \mathcal{Q}_\phi(x_i|z)] + \lambda \max(0, \sum_{z\in\mathcal{Z}}[P_\theta(z|w_i)d(y_i,z)] - \delta)^2 \\ &= \sum_{z\in\mathcal{Z}}[P_\theta(z|w_i)\log \mathcal{Q}_\phi(x_i|z)] + \lambda \max(0, P_\theta(\mathcal{Z} \neq y_i|w_i) - \delta)^2 \end{aligned} \tag{4.5}$$

One should note that we have deviated slightly from [14] as we suspect a typo in their notation. In such that the distortion term is probably more effective in (4.5) than in the original:

$$\mathcal{L}^i_{bin}(\theta,\phi) := \sum_{z\in\mathcal{Z}} P_\theta(z|w_i)(\log \mathcal{Q}_\phi(x_i|z) + \lambda \max(0, d(y_i,z) - \delta)^2)$$

The loss function, combined with the network definitions in Figure 3.2, complete our figure for the Using Mutual Information as privacy leakage for Binary Bivariate data Experiment. The result is depicted in Section 4.1, this process is identical to the algorithmic views of Algorithm 1 and Algorithm 2 in Appendix B.

Regarding this loss function, we like to add that in order to compute the log-likelihood it was necessary to add some $\epsilon \approx 0$ to prevent numerical errors in the pytorch library. In this case $\epsilon = 1e - 7$ is chosen. Therefore, we replace the log-likelihood with $\log[\mathcal{Q}_\phi(x_i|z) + \epsilon]$ in the first term of the loss function.

[1] The source code of these algorithms are found at: github.com/maxxiefjv/privpack [15]

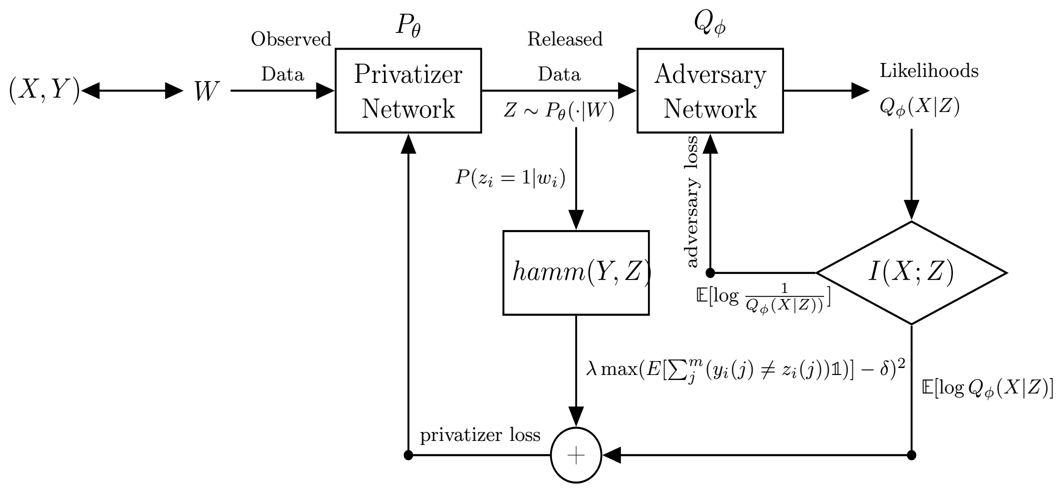


FIGURE 4.1: Visual representation of the (4.5) bivariate binary data-flow used in the training process of a release mechanism using mutual information and Hamming distance.

Results

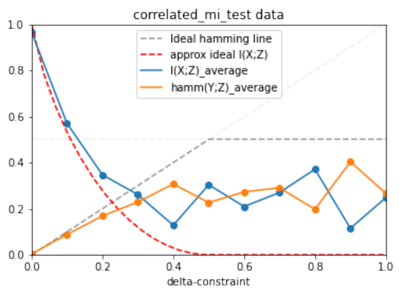
As discussed in Section 3.2.4, we have the expectation that for high distortion-constraints, low privacy-leakage should be obtainable. In this section we discuss the results obtained by the network described in Section 4.1 trained by the data generated as explained in Section 3.2, and compare the results with our stated expectation.

- For completely correlated data X and Y, the mutual information between Y and Z equals that of X and Z. Therefore, any delta-constraint smaller than random (hamming distance sample mean of 0.5) will directly increase the privacy-leakage of X onto Z.
- For slightly correlated data X and Y, we expect the privacy-leakage to be defined by the set distortion constraint. Close to the theoretical optimum.
- For uncorrelated data X and Y, we expect no privacy leakage for any defined distortion constraint.

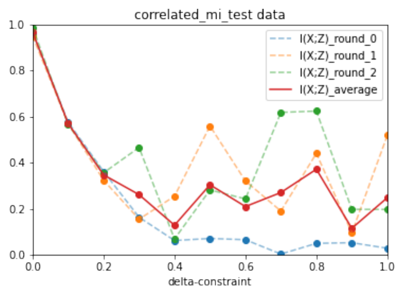
Besides the above expectations we are interested in the stability of the network, i.e. do two independent runs learn the same trade-off.

In Figure 4.2 we have depicted the trade-offs defined by the model defined in Section 4.1 when trained on completely correlated data. As mentioned in our first point above, we expect to see a direct influence of the delta-constraint to the mutual information of X and Z. Looking at the averages (Figure 4.2a), this is clearly true for delta-constraints: [0-0.4]. However, starting from delta=0.5 a fluctuation starts to appear. This is clearly not the result we would expect, the result we expected is however visible in Figure 4.2a as the "approx ideal I(X;Z)". This round shows the result that we expect: the less strict the constraint on Y and Z, the less information is shared between our variables X and Z.

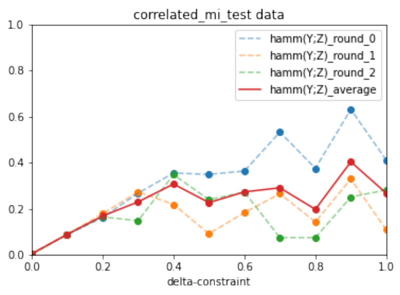
For the Hamming distance we have drawn the expected line in Figure 4.2a. Just as the mutual information line, the results are nowhere close to our expectations. However, again, looking at the round 0 of Figure 4.2c, we see that that line is the most corresponding with our expectation.



(A) Test Data, Correlated. Average I(X;Z)



(B) Test Data, Correlated. Rounds I(X;Z)



(C) Test Data, Correlated. Rounds hamm(Y;Z)

FIGURE 4.2: Graphs on the **correlated** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500

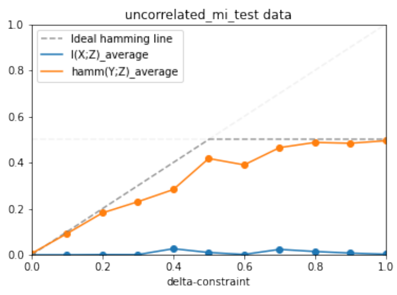
For the correlated data we can therefore state that the result have a high variance for both the mutual information and Hamming distance. Supporting this claim is that each individual run using mutual information gave a seemingly significant difference in the resulting graphs. However, all these runs are not depicted in this thesis.

For the uncorrelated data, we clearly do not expect any leakage of Z on X. We expect that the network learns to drop any knowledge of X dimension on the released data. Then, according to the data-processing inequality, Z cannot contain information on X. Looking at the results in Figure 4.3, we clearly see that the mutual information of X and Z is always close to 0. Our expectation therefore meets the results concerning mutual information.

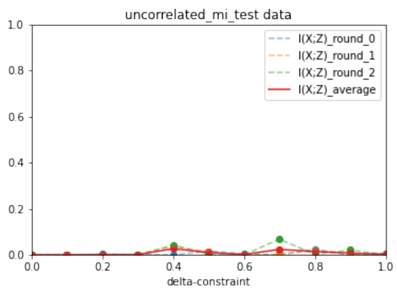
The optimal solution for the network is to output Y at any point. We therefore might expect that the Hamming distance is zero at each point. However, the actual results are obviously the result of our loss function. Which, for a delta-constraint larger than 0.5, does not get penalized enough to know how to reduce the error. The distortion part, second term, of the loss function therefore shows our expectation. For random weights, i.e. random guessing, we do not expect the expected Hamming distance to be higher or lower than a half. Also, with this respect, the results seem to meet at least our expectations. Although on average, it performs a little better than these expectations. Looking at the individual runs (Figure 4.3c) however that seems to be a matter of chance, as individual runs can also perform a little worse than expected.

Concluding the mutual information results on uncorrelated data we therefore say that it meets our the expectations. However, Figure 4.3c, show that the learning process does not seem to be very stable.

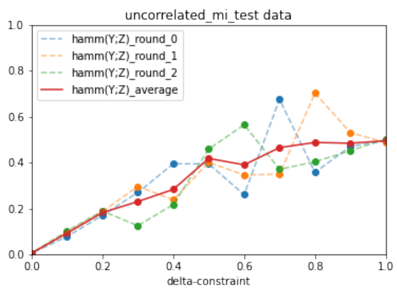
The random distribution should perform almost identical to the uncorrelated data due to the nature of the distribution, i.e. I(X;Z)=0.004. Looking at the graphs in



(A) Test Data, Uncorrelated. Average I(X;Z)



(B) Test Data, Uncorrelated. Rounds I(X;Z)

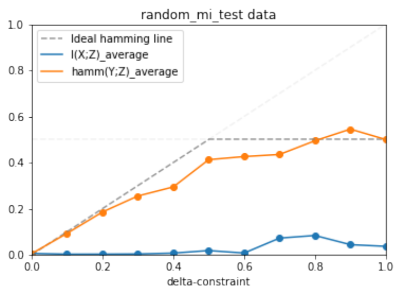


(C) Test Data, Uncorrelated. Rounds hamm(Y;Z)

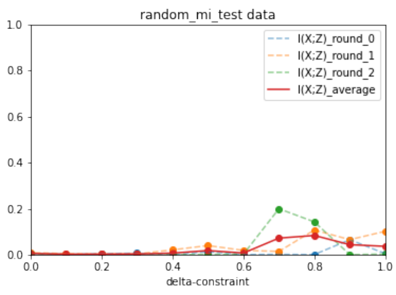
FIGURE 4.3: Graphs on the **uncorrelated** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500

Figure 4.4, we see this to be true. An unstable network, but capable of optimal results regarding privacy-leakage.

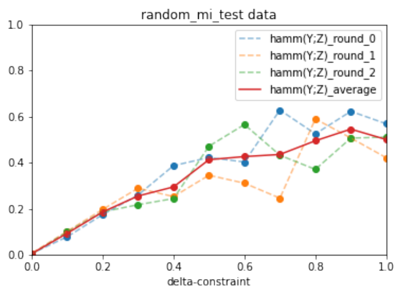
Finally, the results using Stochastic Gradient Descent rather than Adam optimizer seem to be a lot more stable.



(A) Test Data, Random. Average I(X;Z)



(B) Test Data, Random. Rounds I(I;Z)



(C) Test Data, Random. Rounds hamm(Y;Z)

FIGURE 4.4: Graphs on the **random** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500

4.2 Multivariate Gaussian Data using Mutual Information Release Mechanism

We have not elaborated the use of a universal approximator in Section 3.3, which we are using for Gaussian data (as mentioned in Table 3.9). This becomes especially apparent when looking at the full loss function, which shows we sample k release samples from the privatizer network and approximate the expected loss:

$$\mathcal{L}^i_{gauss}(\theta,\phi) := \frac{1}{k}\sum_{j=1}^{k}[\log \mathcal{Q}_\phi(x_i|z_{i,j})] + \lambda \max(0, \frac{1}{k}\sum_{j=1}^{k}[||y_i - z_{i,j}||^2] - \delta)^2. \tag{4.6}$$

This sampling is possible because of the noise we added to the privatizer network (the $\boldsymbol{\epsilon_i}$ in Figure 3.4 and Figure 3.3). Furthermore, the distortion measure as defined in Section 2.4.2 is used to measure the distortion of y in $z_{i,j}$. An algorithmic perspective of this loss function is displayed in Algorithm 3 of Appendix B. The same process is also depicted visually in Section 4.2. In this function and algorithm k defines the number of samples drawn from the privatizer given one input entry.

Similar to (4.5), note that we have deviated slightly from [14] as we suspect a typo in their notation. In such that the distortion term is probably more effective in (4.6) than in the original:

$$\mathcal{L}^i_{gauss}(\theta,\phi) := \frac{1}{k}\sum_{j=1}^{k}[\log \mathcal{Q}_\phi(x_i|z_{i,j}) + \lambda \max(0, ||y_i - z_{i,j}||^2 - \delta)^2.$$

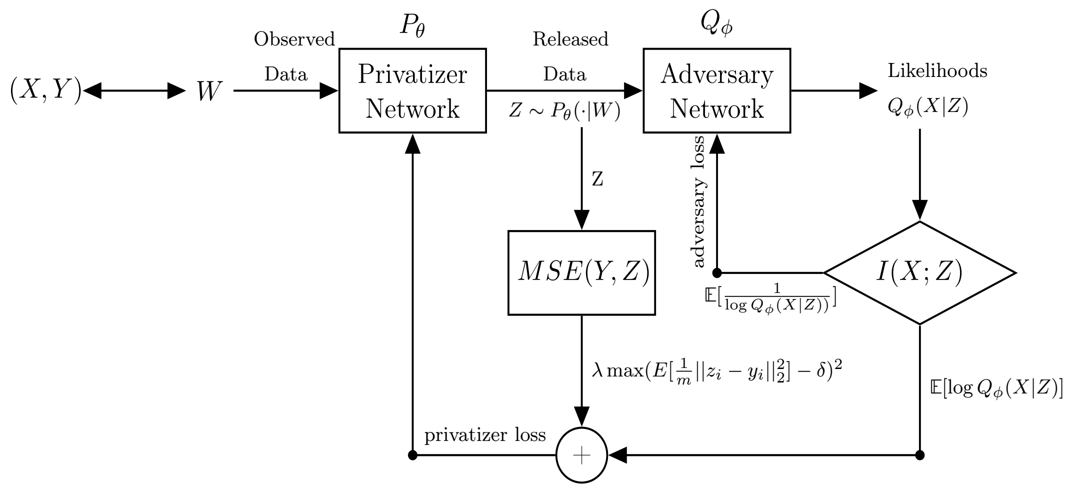


FIGURE 4.5: Visual representation of the Equation (4.6) multivariate Gaussian data-flow used in the training process of a release mechanism using mutual information and mean squared error.

As will be apparent in Section 4.2.1, the results achieved by applying this loss function are not as promising as we might have hoped. How the results are achieved is determined as future work.

4.2.1 Results

Identically to Section 4.1, we have expectations of the optimal performance given the used data distribution and its mutual information. In this section we will compute and discuss the following values:

- The mutual information between X and Z, which we expect to be close to zero as the delta-constraint allows.
- We expect the delta-constraint to be the determining factor for the amount of mutual information between X and Z.

In Figure 4.6 and Figure 4.7 we have shown the achieved results. Clearly the results are nowhere near the expected values. For the uncorrelated 1 dimensional data, the mutual information appears to be having the correct value. However, as the distortion is a lot higher than the set distortion constraint, we consider these values to be the result of random weights rather than a correct global minimum. The same holds for the remaining figures. The PPAN data should have the mutual information be highly dependent on the set distortion constraint as deducted from Table 3.12. The networks where n=10 are nowhere near any expectations. Due to these clearly wrong results, we will not yet discuss the points described in the list above. However, these points will be discussed in Section 5.1, where an improved version is shown.

4.3 Evaluation

Evaluating the binary networks we see that the loss function does not seem to provide stable results. When investigating the training process, we conclude that the networks are converging. Therefore, suspected is that the networks are stuck in a local optimum. As a result we suspect that this loss function is too complex for this simple dataset,

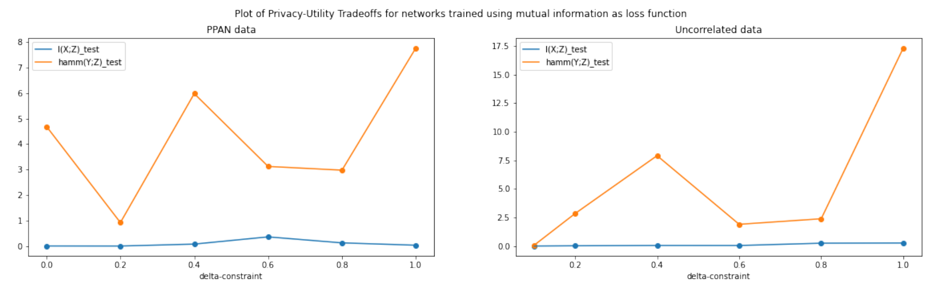


FIGURE 4.6: Image showing the results when reconstructing the 1-dimensional Gaussian scenario from the original PPAN paper.

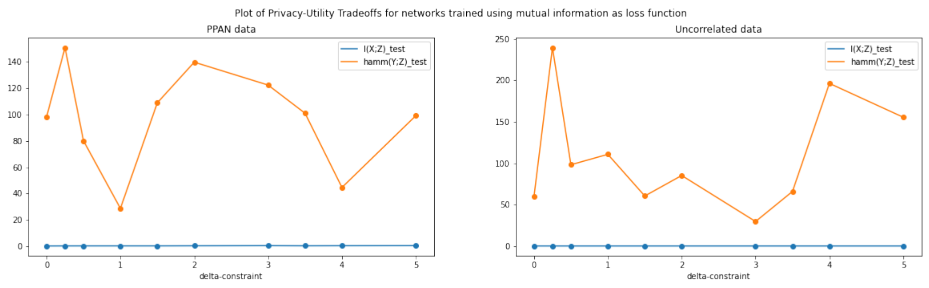


FIGURE 4.7: Image showing the results when reconstructing the 5-dimensional Gaussian scenario from the original PPAN paper.

and therefore very dependent on either the data-sequence used to train the data, or the initial weights of the network. However, as explained in Section 5.1, these struggles are easily overcome when weight decay is applied.

Concerning the Gaussian networks the results are not close to the expected results. The cause is yet to be determined exactly and is currently considered future work. However, in Section 5.1 an improved version is proposed that uses weight decay.

4.4 Discussion

Unclear in the paper of [14], is why they deviate from the default GAN approach as defined by [5]. Instead of only inputting noise into the network and outputting a real looking sample, they approach the problem by directly learning the distribution parameters of $P_{Z|W}$. Only when they find the need to sample from this distribution they add noise to the input, which also includes w_i, and consequently approach output z directly.

Finally, the paper of Tripathy et al. seems challenging to reproduce, likely as a result of lack on details. Such details are important due to the unstable nature of GAN architectures. Such details can be provided with either code, pseudo-code or a detailed overview such as a table. Two questions that arise are about the stability of the networks, and the regularization used to control the instability.

Furthermore, we suspect a mistake in the loss function $\mathcal{L}^i_{disc}(\theta, \phi)$ function defined in [14] on page 5 right column. Expectation over the second term seems incorrect, our expected version of the formulation is defined in (4.5) and (4.6).

Finally, investigation suggests that the optimizer-loss function combination has a crucial role in the results. When implementing the same loss function using the Stochastic Gradient Descent (SGD) optimizer, we find that a better trade-off is found. However, as we restrict ourselves to one and the same optimizer throughout this paper we won't go into detail. However, a short insight to these results are discussed in Chapter 6.

Chapter 5

Implementing Privacy-Preserving Release Mechanisms using other privacy measures

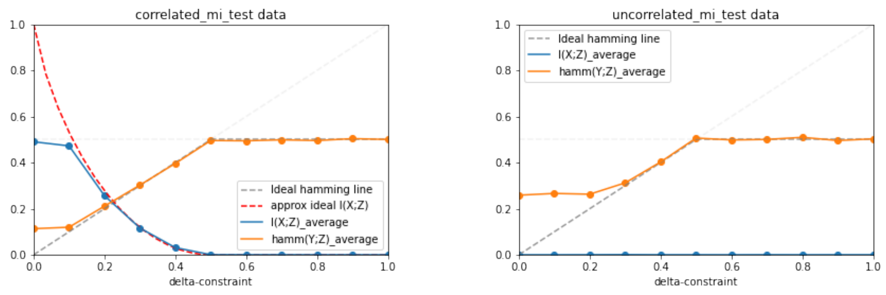
This chapter contains the continuation of Tripathy et al. It therefore contains some answers on the questions described in Section 4.4 in Section 5.1. Furthermore, we describe the results when applying Maximal Leakage, α-Leakage and Maximal α-Leakage as loss function in Section 5.2 and Section 5.3. This means that this chapter contains, for each experiment described in Chapter 3, a description of the implementations for the remaining privacy measures. This is the final step completing all the nodes of Figure 3.1. Besides the privacy-loss function, we will also discuss the architectures as a whole.

5.1 Revised Mutual Information

In Chapter 4 we have discussed the implementations of mutual information as loss function for Gaussian and binary data. As observed in Section 4.2, the implementation of the Gaussian networks is nowhere near the expected performance. In this section we re-examine the implementation of Section 4.2 with a focus on the hyper-parameters in order to pursue an improved version. As noted in Section 4.4, some experimentation suggested a large correlation between the used loss function and the stability and output of the networks. The results when using the SGD loss on mutual information and binary data suggest that this loss function supports a more stable output. A result independent of the starting weights and training data. When applied on the Gaussian networks, however, this results in numerical issues. As this loss function is not dependent on the previous gradients, in contrast to the Adam optimizer, gradient clipping of 0.5 is added. Having resolved the numerical issues, the results are still nowhere near the expectations. Inspection of the weights shows that these are growing as a function of the number of epochs, rather than the optimal solution. This supports the final decision of weight decay. These adjustments result in the values shown in Figure 5.2 for 1-dimensional Gaussian networks, and in Figure 5.3 for 5-dimensional Gaussian networks. Finally, as the addition of weight decay promises better results, we decided to apply the same to the binary networks. These results are shown in Section 5.1.1.

5.1.1 Results

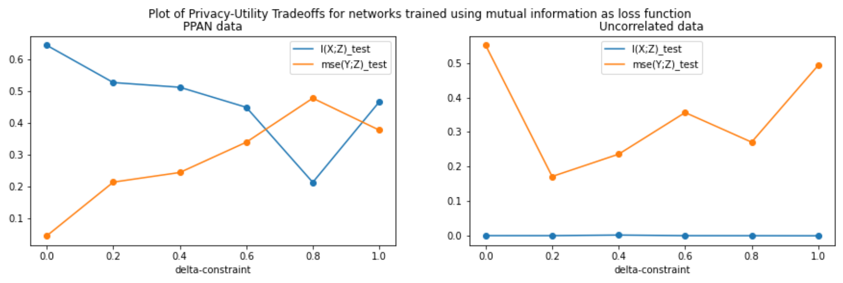
As mentioned in Section 4.2.1, the results which couldn't have been obtained without weight decay are discussed in this section. First to mention, however, are the results shown in Section 5.1.1. These results show even more promising results with weight



(A) Revised Binary results with correlated data, using weight decay

(B) Revised Binary results with uncorrelated data, using weight decay

FIGURE 5.2: Image showing the results after adjusting the hyperparameters of the 1-dimensional Gaussian network



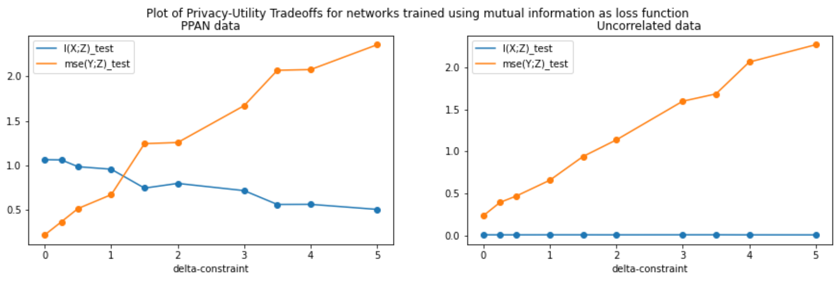
decay. It perfectly matches the ideal hamming line. However, comes with the unintended consequence of not being able to obtain a hamming distance of 0 and mutual information of $I(X;Z) = 1$ for the correlated scenario. Identically, for the uncorrelated case, where $I(X;Y) = 0$, a hamming distance of 0 cannot be obtained either. A reasonable explanation, we have not yet found.

For the Gaussian networks we will support the results by discussing the expectations using the list below, as defined in Section 4.2.1.

- The mutual information between X and Z, which we expect to be close to zero as the delta-constraint allows.
- We expect the delta-constraint to be the determining factor for the amount of mutual information between X and Z.

For both cases, Gaussian-1 and Gaussian-5 networks, we see that the maximal mutual information between X and Z is achieved when the distortion=0. This means that the Gaussian-1 network gives an $I(X;Z) \approx 0.6498$, and the Gaussain-5 networks results in an $I(X;Z) \approx 1.1039$ when the distortion constraint is set to 0. Furthermore, we see that the distortion seems to be lower at than the delta constraint for larger delta constraints. This is probably due to some local optimum in which it is stuck. Some more regularization may be the solution, but this problem needs to be further analyzed to deliver the appropriate solution. Finally, we expect the delta-constraint to be the determining factor for the mutual information between X and Z, which is clearly visible in these results. A smaller delta constraint results in a higher mutual information between X and Z.

FIGURE 5.3: Image showing the results after adjusting the hyper-parameters of the 5-dimensional Gaussian network



5.2 Maximal Leakage

This section shows the results of the experiments defined in Chapter 3, using Maximal Leakage as loss function Section 2.3.2. As this loss function is defined for discrete data, we have not implemented the Gaussian network using this loss function.

5.2.1 Optimizing the release of Bivariate Binary Data

Minimizing Maximal Leakage should minimize the loss of privacy from X onto Z, incorporating all possible generation functions of X. Recall that our loss function for MaxL is equal to the probability of error (2.7), and that we measure the distortion of our binary data using hamming distance (2.12). Equally, to the minimization of the mutual information loss we have the following resulting minimax objective:

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[Q_{X|Z}(X|Z) - 1] + \lambda \max(0, \mathbb{E}[d(Y,Z)] - \delta)^2. \tag{5.1}$$

As a result, the adversary objective can be written as:

$$\max_{Q_{X|Z}} \mathbb{E}[Q_{X|Z}(X|Z) - 1] = \min_{Q_{X|Z}} \mathbb{E}[1 - Q_{X|Z}(X|Z)] = \min_{Q_{X|Z}} \mathbb{E}[J^{MaxL}(X;Z)] \tag{5.2}$$

This loss function is visually represented in Figure 5.4. Clearly, this is almost identical to the mutual information loss function except for the diamond node. This also means that for the experiments to come, nothing is different to the mutual information experiments.

Results

Just as for the mutual information experiments the expectations are almost identical. However, the loss function is linear instead of nonlinear. This makes the function less complex, as shown in Figure 2.2. We suspected the mutual information loss function to be overly complex, in such a way that it gets stuck in local optima Section 4.1. Therefore, a linear function may improve the results. Other than that we will repeat the expectations from Section 4.1:

- For completely correlated, any delta-constraint lower than one will directly increase the privacy-leakage of X onto Z.
- For slightly correlated, we expect the privacy-leakage to be proportionally defined by the set distortion constrained.

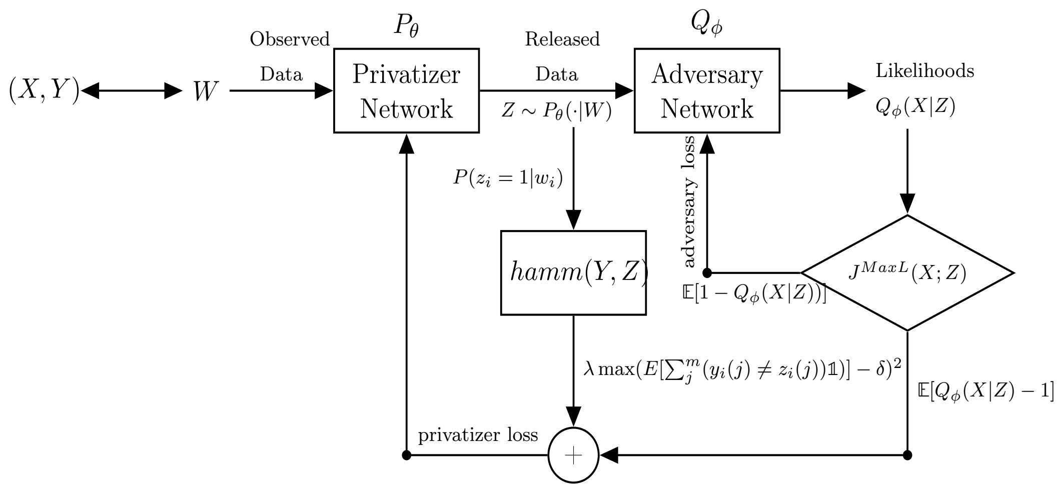

P_θ
Q_ϕ
(X, Y)
W
Observed Data
Privatizer Network
Released Data
$Z \sim P_\theta(\cdot|W)$
Adversary Network
Likelihoods $Q_\phi(X|Z)$
$P(z_i = 1|w_i)$
$hamm(Y, Z)$
adversary loss
$\mathbb{E}[1 - Q_\phi(X|Z))]$
$J^{MaxL}(X; Z)$
$\lambda \max(E[\sum_j^m (y_i(j) \neq z_i(j))\mathbb{1})] - \delta)^2$
$\mathbb{E}[Q_\phi(X|Z) - 1]$
privatizer loss
$+$

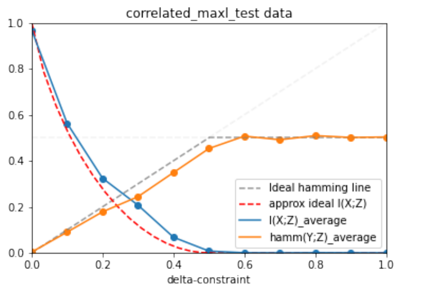
FIGURE 5.4: Visual representation of the (4.5) bivariate binary data-flow used in the training process of a release mechanism using mutual information and hamming distance.

- For uncorrelated data X and Y, we expect no privacy leakage for any defined distortion constraint.

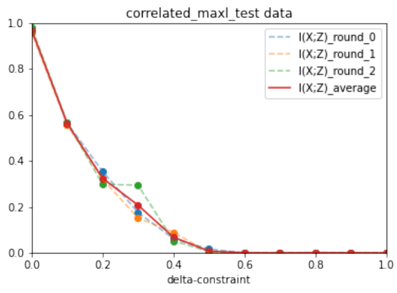
Looking at the correlated data, we see a nice, however not optimal, fit to the ideal hamming line, also visible in this picture is the monotonically decreasing mutual information. These results therefore seem like an almost perfect result. Validating the results, we look at the stability of the networks in Figure 5.5b and Figure 4.2c. Both of these images clearly indicate a stable training process, finding the global optimum.

For uncorrelated data, we may have suspected a better fit for the hamming distance as it has complete freedom. However, the mutual information line seems to be the result of forgetting the private input X, which is as expected. In Figure 5.6c we find less stable results than for the correlated alternative. This inconsistency starts from delta=0.4 and up, and may be the result of its freedom and thus more local optima. Figure 5.6b shows consistent and stable mutual information per round.

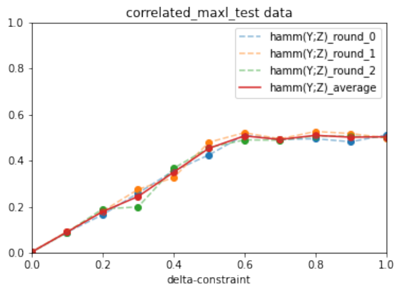
The random data is almost identical to the uncorrelated data, and so are the results.



(A) Test Data, Correlated. Average I(X;Z)

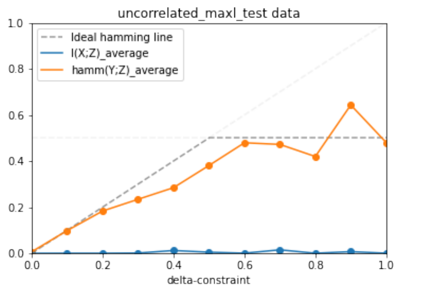


(B) Test Data, Correlated. Rounds I(X;Z)

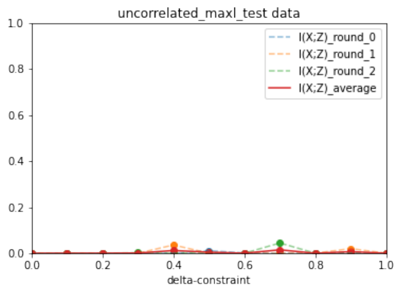


(C) Test Data, Correlated. Rounds hamm(Y;Z)

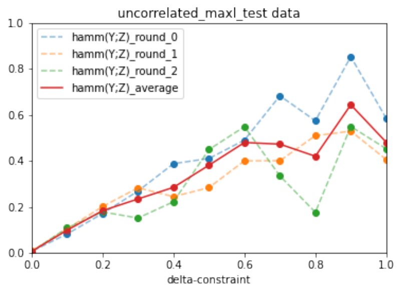
FIGURE 5.5: Graphs on the **correlated** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500



(A) Test Data, Uncorrelated. Average I(X;Z)

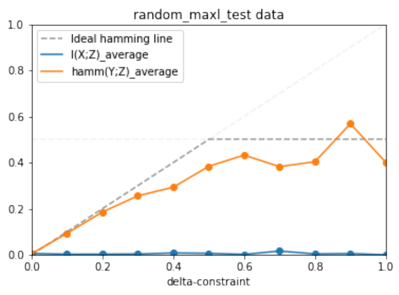


(B) Test Data, Uncorrelated. Rounds I(X;Z)

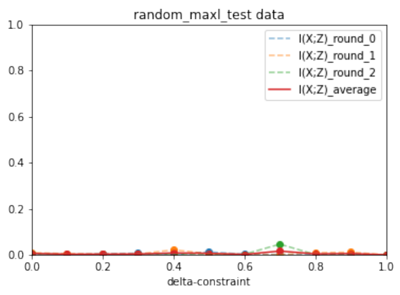


(C) Test Data, Uncorrelated. Rounds hamm(Y;Z)

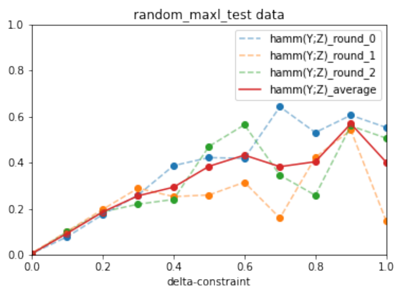
FIGURE 5.6: Graphs on the **uncorrelated** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500



(A) Test Data, Random. Average I(X;Z)



(B) Test Data, Random. Rounds I(I;Z)



(C) Test Data, Random. Rounds hamm(Y;Z)

FIGURE 5.7: Graphs on the **random** data. Depicted is the average line for Mutual Information, multi-round outputs for I(X;Z), and multi-round outputs for hamm(Y;Z). Respectively. λ=500

5.3 α-Leakage and Maximal α-Leakage

The final privacy-leakage measure applied is the alpha-loss. This loss function, as explained in Section 2.3.3, is a loss function parameterized by α. Where $\lim_{\alpha\to 1}$ (2.8) $\approx J^{MI}(X;Z)$ and $\lim_{\alpha\to\infty}$ (2.8) $\approx J^{MaxL}(X;Z)$. This section will validate and evaluate the results for different alphas $\alpha \in (1,\infty]$ given the scenarios as defined in Chapter 3.

5.3.1 Optimizing the release of Bivariate Binary Data

Minimizing the alpha-loss has a similar structure to the loss functions defined before. We have two terms, the first concerning itself over the alpha-loss, and the second on the distortion measure. For the binary data, the distortion measure is the hamming distance. So, similar to the minimization of maximal leakage we get the unconstrained minimax objective defined by:

$$\min_{P_{Z|W}} \max_{Q_{X|Z}} \mathbb{E}[\frac{\alpha}{\alpha-1}(P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}} - 1)] + \lambda \max(0, \mathbb{E}[d(Y,Z)] - \delta)^2. \tag{5.3}$$

As a result, the adversary objective can be written as:

$$\max_{Q_{X|Z}} \mathbb{E}[\frac{\alpha}{\alpha-1}(P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}} - 1)] = \min_{Q_{X|Z}} \mathbb{E}[\frac{\alpha}{\alpha-1}(1-P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}})] = \min_{Q_{X|Z}} \mathbb{E}[J^{MaxL}(X;Z)] \tag{5.4}$$

. Visually represented in Section 5.3.1. Again, with no difference other than the privacy-leakage measure.

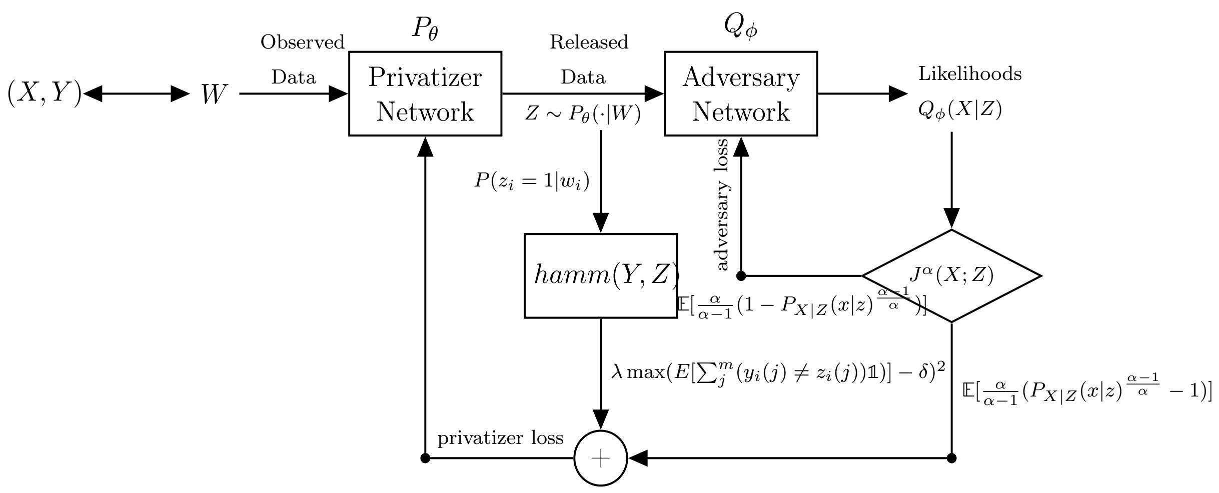
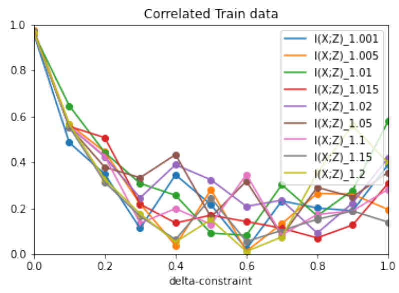
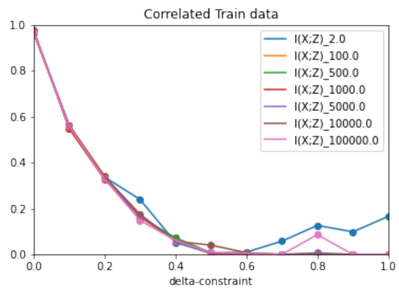

(X, Y)
W
Observed Data
P_θ
Privatizer Network
Released Data
$Z \sim P_\theta(\cdot|W)$
Q_ϕ
Adversary Network
Likelihoods $Q_\phi(X|Z)$
$P(z_i = 1|w_i)$
$hamm(Y, Z)$
adversary loss
$J^\alpha(X;Z)$
$\mathbb{E}[\frac{\alpha}{\alpha-1}(1 - P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}})]$
$\lambda \max(E[\sum_j^m (y_i(j) \neq z_i(j))\mathbb{1})] - \delta)^2$
$\mathbb{E}[\frac{\alpha}{\alpha-1}(P_{X|Z}(x|z)^{\frac{\alpha-1}{\alpha}} - 1)]$
privatizer loss
$+$

FIGURE 5.8: Visual representation of the (4.5) bivariate binary data-flow used in the training process of a release mechanism using mutual information and hamming distance.



(A) Test Data, Uncorrelated. Round-wise I(X;Z)



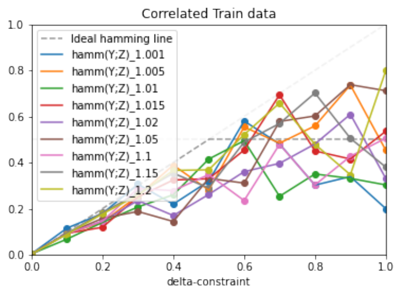
(B) Test Data, Correlated. Round-wise I(X;Z)

FIGURE 5.9: Mutual information graphs on the **correlated** data for alpha-loss. Depicted is mutual information for networks trained on small alphas in picture A: [1.001, 1.005, 1.01, 1.05, 1.1, 1.15, 1.2] and big alphas in picture B: [2, 100, 500, 1000, 5000, 10000, 100000]. λ=500

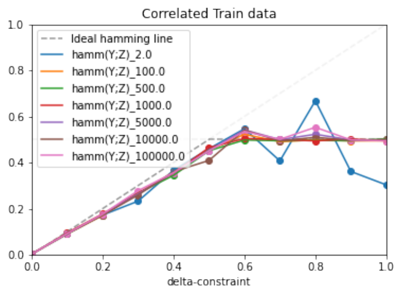
Results

Clearly, for the alpha-loss we would expect the same results as for mutual information and maximal leakage given such a trivial data-set. However, as the uncorrelated data experiments and random data experiments have similar behavior to the mutual information and maximal leakage, we have chosen to focus on the alpha-loss network trained on correlated data. Where we expect that, as remarked in (2.11), that small alphas should more behave like mutual information as loss, and big alphas should behave as maximal leakage. We have generated 16 networks, each network trained 3 times using a K-fold approach. The average results of these trainings are shown in Figure 5.9 and Figure 5.10.

Probably the most interesting part to be seen in this figure is the resemblance between the small alpha losses and mutual information losses, and the big alpha losses with maximal leakage. Where the small alphas show a very non-monotonic curve, the big alphas show a very steady curve. Especially for the values of $\alpha \in [100, 500, 1000, 5000, 10000]$. Further investigation of these big alphas with correlated



(A) Test Data, Correlated. Round-wise hamm(x;z)

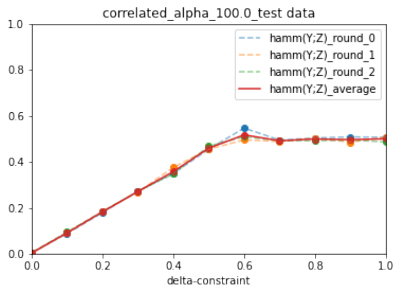


(B) Test Data, Correlated. Round-wise hamm(x;z)

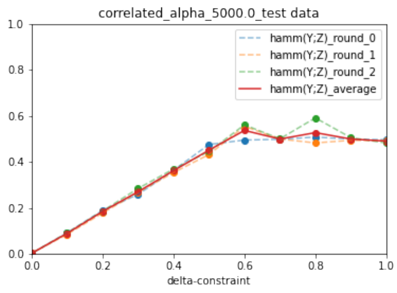
FIGURE 5.10: Distortion graphs on the **correlated** data for alpha-loss. Depicted is the distortion for networks trained on small alphas in picture A: [1.001, 1.005, 1.01, 1.05, 1.1, 1.15, 1.2] and big alphas in picture B: [2, 100, 500, 1000, 5000, 10000, 100000]. λ=500

data is shown in Figure 5.11a and Figure 5.11b. This shows a steady curve independent of the training round, identically to maximal leakage. The remaining figures Figure 5.11c and Figure 5.11d, show results analogous to mutual information. Both of these results, are as expected. Both for the stability of training and the results are almost identical. Similarly, in Figure 5.12 these inspections are shown for uncorrelated data. Although these results are not as stable as those in Figure 5.11, this is expected due to the additional freedom that is inherent to the uncorrelated nature of the problem.

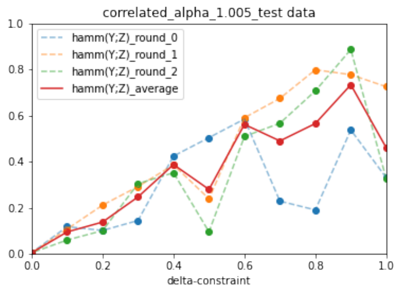
Yet, identically to mutual information and maximal leakage, the results are completely different when using the SGD optimizer. Where mutual information seems to perform better and equal to the maximal leakage performance using SGD. However, maximal leakage seems to perform worse.



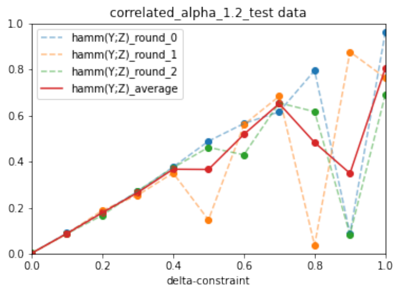
(A) Test Data, Correlated. Per round $hamm(x, z)$ alpha=100



(B) Test Data, Correlated. Per round $hamm(x, z)$ alpha=5000

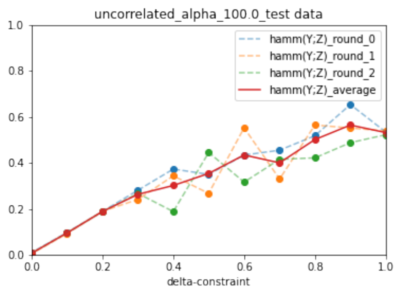


(C) Test Data, Correlated. Per round $hamm(x, z)$) alpha=1.005

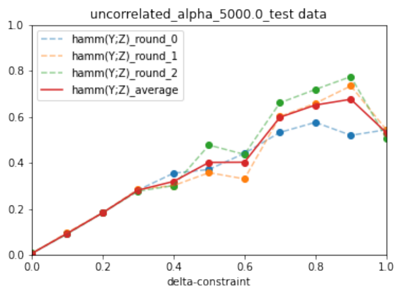


(D) Test Data, Correlated. Per round $hamm(x, z)$ alpha=1.2

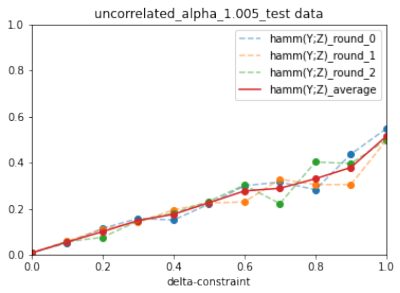
FIGURE 5.11: Distortion small-alpha highlight graphs on the **correlated** data. Featured alpha-losses: 1.005, 1.2, 100, 5000. λ=500



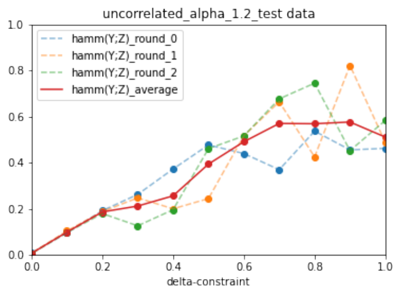
(A) Test Data, Uncorrelated. Per round $hamm(x, z)$ alpha=100



(B) Test Data, Uncorrelated. Per round $hamm(x, z)$ alpha=5000



(C) Test Data, Uncorrelated. Per round $hamm(x, z)$ alpha=1.005



(D) Test Data, Uncorrelated. Per round $hamm(x, z)$ alpha=1.2

FIGURE 5.12: Distortion large-alpha highlight graphs on the **uncorrelated** data. Featured alpha-losses: 1.005, 1.2, 100, 5000. λ=500

Chapter 6

Discussion

Throughout this thesis several points of discussion become apparent. In this chapter we discuss these points. Specifically, we will discuss our validation method in Section 3.4 and the difference of using the Adam optimizer and Stochastic Gradient Descent optimizer. Furthermore, some remarks are made on the implementation for the Gaussian networks. Finally, we would like to argue for the importance of supplying code to any practical results and add some remarks on the practical applications this thesis may support in practice. Future work regarding this thesis is discussed in Chapter 7.

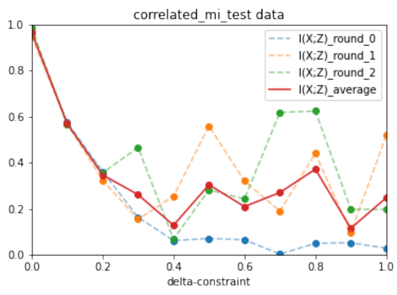
6.1 Mutual Information as evaluation method

As mentioned, mutual information is used as validation method for each privacy-leakage measure. Mainly due to the intractability of computing the maximal-leakage, alpha-leakage and maximal alpha-leakage directly. This results in limited validations based on the results. As a result, it is difficult to argue that the alpha-loss or MaxL loss functions perform better when they obtain the same privacy-utility trade-off given a delta constraint. However, in this work it appears that the privacy-utility trade-off is different for different loss functions. Furthermore, this also indicates the instability of the networks. Therefore, it is useful to use such a measure for such validations although one should be careful about drawing conclusions.

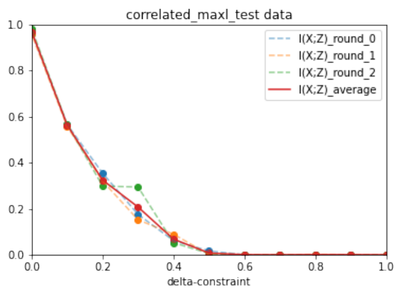
6.2 Generative Adversarial Network Optimizers

Additional discussion arises by experimenting on the optimizer. It appears to have a large impact on the eventual results. It seems that using the Stochastic Gradient Descent optimizer, roughly no difference is shown for the bivariate binary data networks trained on mutual information, maximal leakage or alpha-leakage. In Figure 6.1 we show these differences for the mutual information and maximal leakage loss. In the first row the results of the Adam optimizer are shown, and in the second row the results using the Stochastic Gradient Descent optimizer. Both trained on correlated data and uncorrelated data.

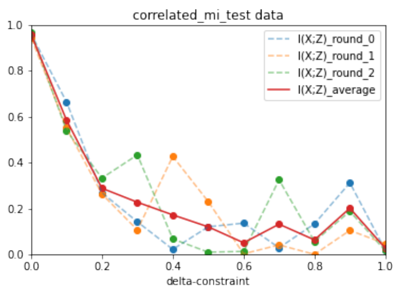
It appears that the Adam optimizer boosts the performance of the maximal leakage loss, however, has a lot more trouble finding the global optimum for the mutual information loss; log loss. Initial thought may argue that the maximal leakage is linear, and the log-loss is therefore a lot more complex. However, as we show in Figure 6.2, the alpha loss has the same behavior, yet is probably the most complex function. For small alphas, the behavior is similar to the mutual information loss. Clearly visible when using the Adam optimizer. For large alphas the behavior is similar to the maximal leakage loss. Interestingly enough, the maximal leakage curve behavior is substantially



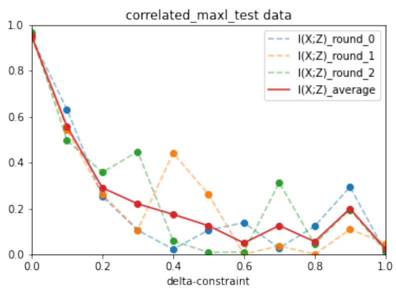
(A) Adam optimizer with mutual information loss on correlated data.



(B) Adam optimizer with maximal leakage loss on correlated data.



(C) Stochastic Gradient Descent optimizer with mutual information loss on correlated data.



(D) Stochastic Gradient Descent optimizer with maximal leakage loss on correlated data.

FIGURE 6.1: Image showing the differences of SGD and Adam in maximal leakage and mutual information.

different when using the SGD optimizer. The Adam optimizer is a complex optimizer, identifying the factor causing this difference in behavior is therefore not evident.

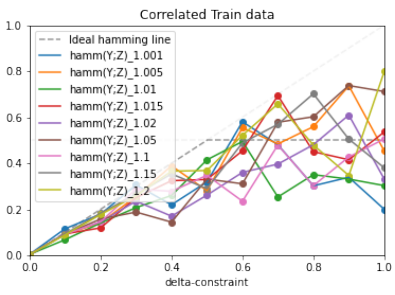
Concerning the bivariate binary data, another experiment could include maintaining the dimensions so that the number of output dimensions equals two. And see if the network itself learns to drop one dimension, i.e. always output the same value at one of the outputs.

6.3 Using α-loss and MaxL-loss for Gaussian networks

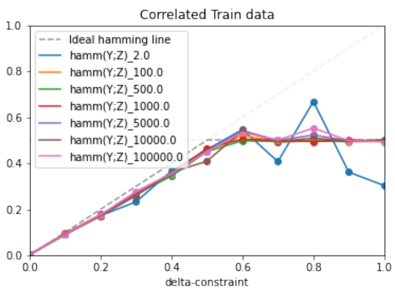
As a matter of experiment, and although the working has not been proven, we have experimented with the usage of alpha-loss and MaxL loss on Gaussian networks. However, as visible in figures Figure 6.3, they do not show promising results when combined with the hyper parameters, topologies and regularization described in Chapter 5. However, it does show affect in the correlated case. Because of those effects, and because it does show good results for binary data, this should be part of some future work in which alpha-loss and maximal leakage are further analyzed for continuous data.

6.4 Notes on the implementation of the networks.

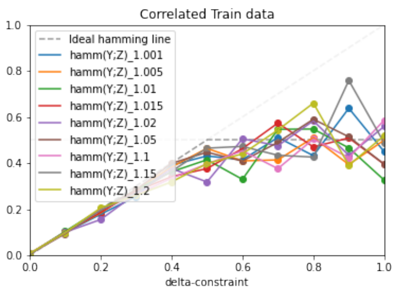
Regarding the Gaussian networks, there appeared to be a substantial amount of numerical issues when training. One of specific issue to note is that when training the Multivariate Gaussian Parameters of the adversary, they tend to approach to zero. This causes numerical issues as we cannot use a multivariate normal distribution with



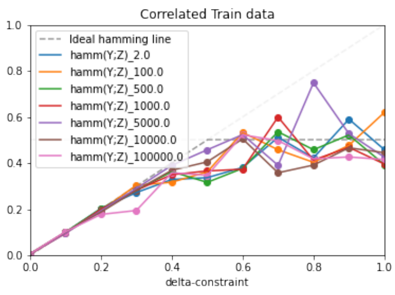
(A) Adam optimizer with small-alpha alpha loss on correlated data.



(B) Adam optimizer with large-alpha alpha loss on correlated data.



(C) Stochastic Gradient Descent optimizer with small-alpha alpha loss on correlated data.



(D) Stochastic Gradient Descent optimizer with large-alpha alpha loss on correlated data.

FIGURE 6.2: Image showing the differences of SGD and Adam in small and large values for alpha.

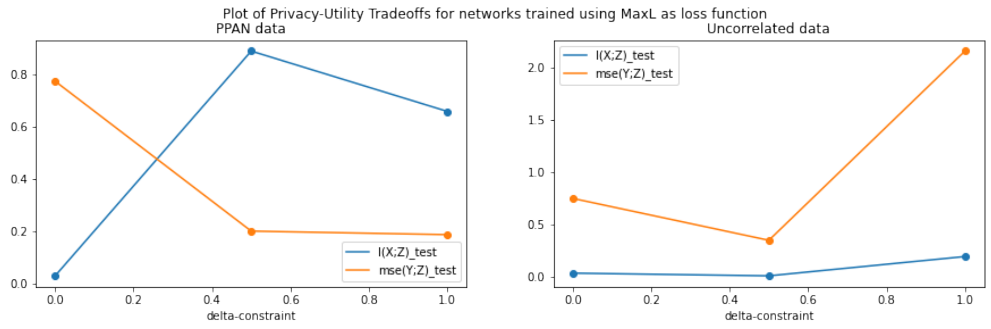


FIGURE 6.3: Gaussian Network 1-dimensional data trained with maxl.

the Σ diagonal all zeros. Therefore, I have chosen to reset the adversary Parameters and recompute a potential distribution.

In addition, I would like to note and plead for the necessity of code-sharing when presenting practical results in a paper. It appears a crucial step for a swifter research environment. This paper the used code visible in [15]. This GitHub repository contains code for creating the systems shown in this paper, with an application.

6.5 Working with Privacy-Preserving Systems

Finally, the work presented in this paper is part of the progress made in privacy-preserving systems as introduced in Chapter 1. In the essence this thesis provides practical and theoretical support for future privacy-preserving systems. This work shows the possibility of using privacy-preserving systems for low dimensional continuous data and discrete data. For such systems one has to be careful about the regularization used, as the stability of the network is highly depended on regularization methods. Furthermore, this work combines the discussion on privacy-leakage measures with a practical implementation. What is shown is that maximal leakage is more stable than mutual information when using the Adam optimizer without regularization methods. As a result, this thesis gives reason to identify Maximal Leakage and alpha-loss as a measure for continuous data. The validation in this thesis concerning the optimal performance, however, is achieved with mutual information. Although, as explained in Section 3.4, validation using Maximal Leakage would be preferred.

Chapter 7

Conclusions and Future Work

We finalize this thesis with the drawn conclusions given the results shown in Chapter 4 and Chapter 5. The conclusions below include the practical validation and evaluation of mutual information, maximal leakage and alpha-loss.

- Mutual information as loss function is not by itself a stable loss function. However, appears more stable when using SGD as optimizer or in combination with weight decay.
- Maximal leakage as loss function indeed also minimizes the shared information between private variable X and released variable Z for discrete data.
- Maximal leakage approaches the distortion constraint for each delta constraint when trained on the correlated data. Although sometimes it has a smaller distortion, which raises the question if the right optimal privacy-utility trade-off has been obtained given the provided delta-constraint.
- All loss functions defined in Section 2.3 have the complexity to minimize the mutual information between the release variable and the private variable for discrete data.
- For (almost) uncorrelated data the optimal privacy-utility trade-off means $I(X;Z) = 0$ and $\text{hamm}(X,Y) = 0$. However, loss functions are incapable of finding this optimum when it has too much freedom. Regardless of the loss function used. Yet, each network and loss function finds the solution where $I(X;Z) = 0$ for (almost) uncorrelated data.
- What can be stated is that the theoretical proof of Liao et al., shown in (2.11), is shown to be true in practice [9]. Larger values of alpha correspond with a privacy-utility trade-off curve similar to maximal leakage. And the smaller values correspond to the mutual information trade-off curve.

Concluding the work it seems that, although almost no difference is shown using SGD optimizers, maximal leakage is a more stable loss function to minimize privacy loss. As we stated in Section 3.4, the only way to draw conclusions using mutual information as validation method, is when maximal leakage shows the same or better results. As this is true, we can conclude that maximal leakage is in fact a more stable way to approach privacy-preserving release mechanisms for a system as trivial as binary bivariate data. Obviously, it may be true that better hyper-parameters for the adversary and privatizer network may result in better results for the mutual information loss. However, given a straightforward network setups, maximal leakage performs better and is, in that case, favorable over mutual information. To learn more about the strengths of the MaxL and α-loss functions, future work on high dimensional discrete data is required.

However, no conclusions like these can yet be drawn for Gaussian networks. We have shown in Section 5.1 that at least the weight decay regularization method is required to make the loss function of mutual information work. However, the curves shown in Figure 5.2 and Figure 5.3 are still not as optimal as the curves shown in the paper of Tripathy et al. To make conclusions concerning maximal leakage and alpha-loss on continuous data, future work is required on several aspects. In terms of getting the networks to work, the first question for future work is regarding the maximal leakage loss function for continuous variables? In terms of validation for these types of networks it would be of great help if a tractable alternative for maximal leakage, alpha-leakage and maximal alpha-leakage is identified.

Future work includes the study on the influence of the optimizer on the chosen loss function. To be specific, why are the results of the mutual information and maximal leakage in combination with SGD almost inseparable, but when used in combination with the Adam optimizer clearly distinct.

In addition, all privacy-leakage measures should be tested in a practical setting. An example of such a setting would be face recognition, BMI detection or other scenarios where one or multiple specific attributes should be visible, but all other attributes should be difficult.

Appendix A

Estimators

A.1 Data Distribution Estimators

As, we will not be in possesion of the actual distributions of the output data, as the needed $P_{\hat{Y}|X}$ in Section 2.3, we will have estimators to estimate these distributions in order to compute the privacy-leakage for some validation set. Below we will define two estimators, that are capable of estimating the distribution. One for bivariate binary data, and one for multivariate gaussian data. Where the last is used for the last two experiments as defined in Chapter 3.

A.1.1 Estimating on Bivariate Binary Data

For binary data we need to estimate the distribution by counting. This is used in the first experiment as explained in Section 3.2. This estimation is shown in the formula:

$$\hat{P}(S = s, V = v) = \frac{1}{n} \sum_{i}^{n} (s_i = s \wedge v_i = v) \mathbb{1}. \tag{A.1}$$

Where n is the number of samples in matrices S and V. This formula is then used to compute the distribution in Table A.1.

Table A.1: This table displays how the formula in Equation (A.1) results in our estimated distribution

$\hat{P}(S = s, V = v)$	$V = 0$	$V = 1$
$S = 0$	$\hat{P}(S = 0, V = 0)$	$\hat{P}(S = 1, V = 0)$
$S = 1$	$\hat{P}(S = 1, V = 0)$	$\hat{P}(S = 1, V = 1)$

A.1.2 Estimating parameters of Multivariate Gaussian Data

Our last two experiments are both implied (explicitly and implicitly) to be of multivariate gaussian form: $W \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Therefore, to estimate these distributions after applying our privacy-preserving transformation, we need to estimate $\boldsymbol{\mu}$ and Σ given our released data Z and private data X. So that, in turn, we can also estimate the privacy-leakage measures as described in Section 2.3. Doing so requires the following two formulas:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i} (x_i, z_i) \tag{A.2}$$

Where n is the number of samples in matrix X.

$$\hat{\Sigma} = \frac{1}{n}\sum_i ((x_i, z_i) - \hat{\boldsymbol{\mu}})^2 \tag{A.3}$$

A.2 Estimating Mutual Information

To measure the performance of our networks we have computed mutual information, and is used as the main form of validation. We have used two estimators for mutual information. For binary data and gaussian data. In the following sections the formulas for each data type is explained.

A.2.1 Estimating Mutual Information on Binary data

To estimate mutual information on binary data we have used the functions for entropy, and mutual information given the estimated parameters of the distribution. So we have used (3.4), where every probability is replaced with the estimated alternative from (A.1).

A.2.2 Estimating Mutual Information on Gaussian data

To estimate the mutual information we have used the formula defined in [14]. And is defined by:

$$\hat{I}(X;Y) = 0.5 \log \frac{|\hat{\Sigma}_X|}{|\hat{\Sigma}_{X|Y}|} \tag{A.4}$$

Where $\hat{\Sigma}_{X|Y}$ is computed using the schur-complement of the defined covariance matrix (A.5).

$$\mathrm{schur}(\hat{\Sigma}) = \hat{\Sigma}_{X|Y} = \hat{\Sigma}_{11} - \hat{\Sigma}_{12}\hat{\Sigma}_{22}\hat{\Sigma}_{21} \tag{A.5}$$

Appendix B

Pseudo-Algorithms

In this appendix the algorithms used to perform the optimisations are displayed. These algorithms do not include the epoch and iteration loop. Nor do they include the optimizer function. They are solely focused on the implementation of the privacy leakage measures in a practical environment.

B.1 Binary network Algorithms

Algorithm 1 Privatizer criterion binary case, stochastic version.

```
 1: Input: $Priv^t, Adv^t, W_i, \lambda, \delta$.    ▷ Adversary and Privatizer denoted at timestep t
 2: Output: $\mathbb{E}_z$[likelihood + distortion]-loss.

 3: $expected\_likelihood = 0$
 4: $expected\_distortion = 0$
 5: for $z \in \{0, 1\}$ do

 6:     $prob\_z\_given\_w = Priv^t(x)$    ▷ $Priv^t$ is trained to output: $P_\theta(Z = 1|W_i)$
 7:     if $z = 0$ then
 8:         $prob\_z\_given\_w = 1 - Priv^t(x)$

 9:     $log\_lik\_Xi\_giv\_z = \log(Adv^t(z))$ ▷ $Adv^t$ is trained to output: $P_\phi(X = 1|z)$
10:     if $x = 1$ then
11:         $log\_lik\_Xi\_giv\_z = \log(1 - Adv^t(z))$

12:     $expected\_likelihood \mathrel{+}= prob\_z\_given\_w * log\_lik\_Xi\_giv\_z$
13:     $expected\_distortion \mathrel{+}= prob\_z\_given\_w * (y_i \neq z)\mathbb{1}$
    return $expected\_likelihood + \lambda * \max(0, (expected\_distortion - \delta)^2)$
```

Algorithm 2 Binary Loss function adversary

```
1: Input: $Priv^t, Adv^t, W_i, \lambda, \delta$.    ▷ Adversary and Privatizer denoted at timestep t
2: Output: $\mathbb{E}_z$[likelihood]-loss.

3: $expected\_likelihood = 0$
4: for $z \in \{0,1\}$ do

5:     $prob\_z\_given\_w = Priv^t(x)$    ▷ $Priv^t$ is trained to output: $P_\theta(Z = 1|W_i)$
6:     if $z = 1$ then
7:         $prob\_z\_given\_w = 1 - Priv^t(x)$

8:     $log\_lik\_Xi\_giv\_z = \log(Adv^t(z))$ ▷ $Adv^t$ is trained to output: $P_\phi(X = 1|z)$
9:     if $x = 0$ then
10:        $log\_lik\_Xi\_giv\_z = \log(1 - Adv^t(z))$

11:    $expected\_likelihood \mathrel{+}= prob\_z\_given\_w * (-log\_lik\_Xi\_giv\_z)$
   return $expected\_likelihood$
```

B.2 Gaussian network Algorithms

Algorithm 3 Gaussian Loss function privatizer.

```
1: Input: $Priv^t, Adv^t, W_i, \lambda, \delta$.    ▷ Adversary and Privatizer denoted at timestep t
2: Output: $\mathbb{E}_z[likelihood + distortion] - loss.$

3: $expected\_likelihood = 0$
4: $expected\_distortion = 0$

5: for $z \in \{Priv^t(W_i, \epsilon) | \epsilon \in \text{Uniform}[-1,1]\}^k$ do
6:     $(\hat{\boldsymbol{\mu}}, \hat{\Sigma}) = Adv^t(z)$    ▷ Adversary at timestep t
7:     $\mathcal{Q}_{X|Z}(x_i|z) = \mathcal{N}(x; (\hat{\boldsymbol{\mu}}, \hat{\Sigma}))$

8:     $expected\_distortion \mathrel{+}= \frac{1}{m}||Z - Y||_2^2$
9:     $expected\_likelihood \mathrel{+}= \log(\mathcal{Q}_{X|Z}(x|z))$

10: $expected\_distortion = \frac{1}{k} expected\_distortion$
11: $expected\_likelihood = \frac{1}{k} expected\_likelihood$
    return $expected\_likelihood + \lambda * \max(0, expected_distortion - \delta)^2$
```

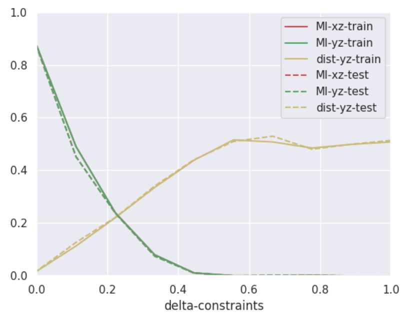
Algorithm 4 Gaussian Loss function adversary.

```
1: Input: $Priv^t, Adv^t, W_i, \lambda, \delta$.    ▷ Adversary and Privatizer denoted at timestep t
2: Output: $\mathbb{E}_z[likelihood] - loss$.

3: $expected\_likelihood = 0$

4: for $z \in \{Priv^t(W_i, \epsilon) | \epsilon \in \text{Uniform}[-1, 1]\}^k$ do
5:     $(\hat{\boldsymbol{\mu}}, \hat{\Sigma}) = (Adv^t(z)[:5], \text{diag}((Adv^t(z)[5:])^2))$
6:     $\mathcal{Q}_{X|Z}(x_i|z) = \mathcal{N}(x; (\hat{\boldsymbol{\mu}}, \hat{\Sigma}))$

7:     $expected\_likelihood \mathrel{+}= -\log \mathcal{Q}_{X|Z}(x|z)$
   return $\frac{1}{k} expected\_likelihood$
```

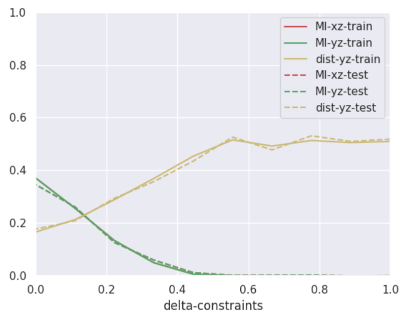
Appendix C

Lambda oriented Binary results

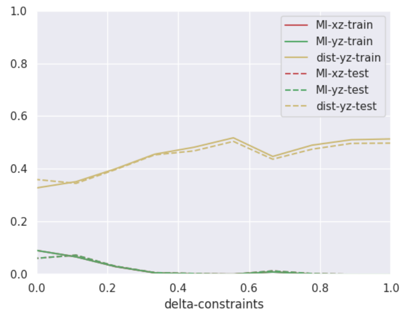
In this section you can find the behavior for different values of lambda. As you can see, it mostly emphasizes the distortion constraint. For larger values of λ, the error is bigger for smaller errors. In conclusion, to accomplish the delta-constraint the lambda value should be high enough. However, for values too large one needs to be mindful about possible numerical issues.



(A) Lambda=500



(B) Lambda=5

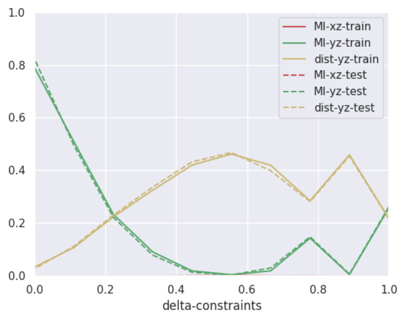


(C) Lambda=1

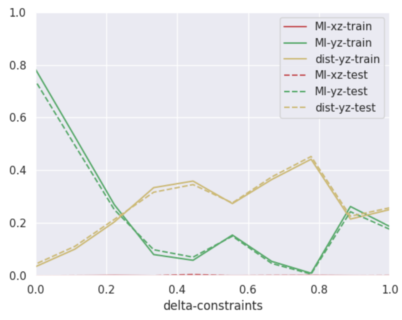
FIGURE C.1: Graphs on the **correlated** data. Showing delta-constrains behavior with different lambdas.

TABLE C.1: Results Binary Synthetic **correlated** Data.

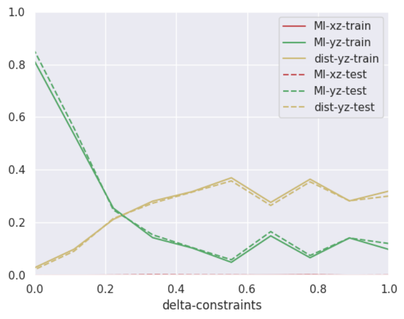
λ	δ	Training data			Testing data		
		$\hat{I}_{tr}(X;Z)$	$\hat{I}_{tr}(Y;Z)$	$\mathbb{E}[d_{tr}(Y,Z)]$	$\hat{I}_{te}(X;Z)$	$\hat{I}_{te}(Y;Z)$	$\mathbb{E}[d_{te}(Y,Z)]$
500	1	0.0028	0.0028	0.532	0.0014	0.0014	0.521
500	0.5	0	0	0.497	0.0028	0.0028	0.469
500	0.1	0.524	0.524	0.102	0.5534	0.5534	0.093
500	0	0.8462	0.8462	0.022	0.8369	0.8369	0.024
5	1	0	0	0.495	0	0	0.499
5	0.5	0	0	0.491	0.0013	0.0013	0.522
5	0.1	0.272	0.272	0.204	0.28	0.28	0.2
5	0	0.38	0.38	0.162	0.3642	0.3642	0.166
1	1	0	0	0.506	0	0	0.511
1	0.5	0	0	0.493	0	0	0.501
1	0.1	0.066	0.066	0.351	0.0748	0.0748	0.34
1	0	0.088	0.088	0.3288	0.082	0.082	0.334



(A) Lambda=500



(B) Lambda=5

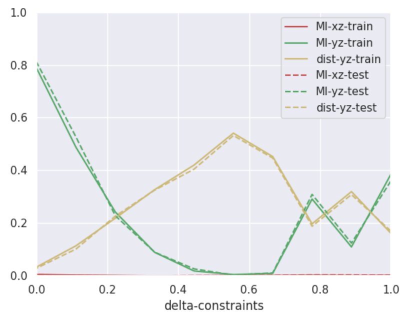


(C) Lambda=1

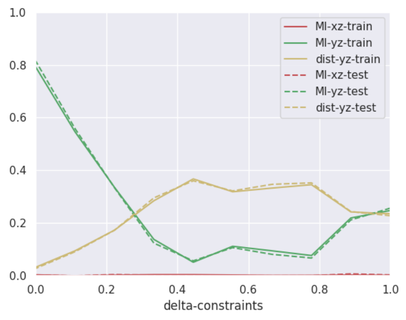
FIGURE C.2: Graphs on the **uncorrelated** data. Showing delta-constrains behavior with different lambdas.

TABLE C.2: Results Binary Synthetic **uncorrelated** Data.

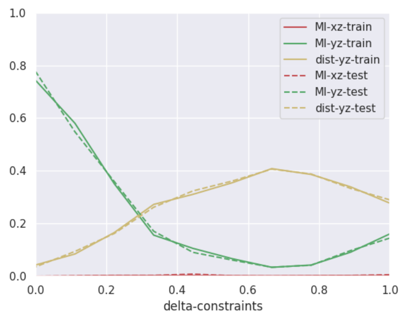
λ	δ	Training data			Testing data		
		$\hat{I}_{tr}(X;Z)$	$\hat{I}_{tr}(Y;Z)$	$\mathbb{E}[d_{tr}(Y,Z)]$	$\hat{I}_{te}(X;Z)$	$\hat{I}_{te}(Y;Z)$	$\mathbb{E}[d_{te}(Y,Z)]$
500	1	0.003	0.006	0.532	0.0027	0.006	0.526
500	0.5	0	0.001	0.478	0.0001	0.006	0.456
500	0.1	0	0.525	0.102	0.001	0.506	0.108
500	0	0	0.782	0.035	0.0004	0.779	0.036
5	1	0.0004	0.002	0.516	0.0018	0.0019	0.52
5	0.5	0	0.007	0.45	0.0007	0.003	0.467
5	0.1	0.0002	0.538	0.099	0	0.544	0.096
5	0	0	0.776	0.096	0	0.779	0.036
1	1	0	0.0012	0.48	0.0006	0.0055	0.4575
1	0.5	0.0004	0.0046	0.46	0.002	0.0092	0.443
1	0.1	0.0004	0.5422	0.0965	0.0008	0.56	0.091
1	0	0	0.7835	0.0345	0.0004	0.7763	0.036



(A) Lambda=500



(B) Lambda=5



(C) Lambda=1

FIGURE C.3: Graphs on the **random** data. Showing delta-constrains behavior with different lambdas.

TABLE C.3: Results Binary Synthetic **random** Data.

λ	δ	Training data			Testing data		
		$\hat{I}_{tr}(X;Z)$	$\hat{I}_{tr}(Y;Z)$	$\mathbb{E}[d_{tr}(Y,Z)]$	$\hat{I}_{te}(X;Z)$	$\hat{I}_{te}(Y;Z)$	$\mathbb{E}[d_{te}(Y,Z)]$
500	1	0.002	0.3846	0.162	0.0025	0.3591	0.17
500	0.5	0.0015	0.0021	0.4734	0.0036	0.0032	0.467
500	0.1	0.0035	0.515	0.1048	0.001	0.5427	0.096
500	0	0.0044	0.787	0.0034	0.0054	0.805	0.03
5	1	0.0028	0.0792	0.336	0.0023	0.0705	0.346
5	0.5	0.0027	0.0177	0.421	0.0036	0.0234	0.41
5	0.1	0	0.5846	0.089	0	0.5846	0.084
5	0	0.003	0.7912	0.033	0.0028	0.7896	0.033
1	1	0.0012	0.1006	0.3234	0.0025	0.1119	0.313
1	0.5	0.0023	0.0548	0.366	0.0013	0.0581	0.364
1	0.1	0.001	0.5889	0.0823	0.0005	0.6073	0.077
1	0	0.0004	0.7288	0.0464	0	0.6937	0.0545

Bibliography

[1] Shahab Asoodeh, Fady Alajaji, and Tamás Linder. “Notes on Information-Theoretic Privacy”. In: (Oct. 2015). arXiv: 1510.02318. URL: http://arxiv.org/abs/1510.02318.

[2] Yuksel Ozan Basciftci, Ye Wang, and Prakash Ishwar. “On privacy-utility tradeoffs for constrained data release mechanisms”. In: *2016 Information Theory and Applications Workshop, ITA 2016*. Institute of Electrical and Electronics Engineers Inc., Mar. 2017. ISBN: 9781509025299. DOI: 10.1109/ITA.2016.7888175.

[3] Cynthia Dwork. *Differential Privacy*. Tech. rep. 2006.

[4] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. *Limiting Privacy Breaches in Privacy Preserving Data Mining*. Tech. rep. 2003.

[5] Ian J Goodfellow et al. *Generative Adversarial Nets*. Tech. rep. 2014. URL: http://www.github.com/goodfeli/adversarial.

[6] Jeff Heaton. “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning”. In: *Genetic Programming and Evolvable Machines* 19.1-2 (June 2018), pp. 305–307. ISSN: 1389-2576. DOI: 10.1007/s10710-017-9314-z.

[7] Chong Huang et al. *Context-Aware Generative Adversarial Privacy*. Tech. rep. 2017. arXiv: 1710.09549v3.

[8] Ibrahim Issa, Sudeep Kamath, and Aaron B. Wagner. “An operational measure of information leakage”. In: *2016 50th Annual Conference on Information Systems and Sciences, CISS 2016*. Institute of Electrical and Electronics Engineers Inc., Apr. 2016, pp. 234–239. ISBN: 9781467394574. DOI: 10.1109/CISS.2016.7460507.

[9] Jiachun Liao et al. “Tunable Measures for Information Leakage and Applications to Privacy-Utility Tradeoffs”. In: *IEEE Transactions on Information Theory* (Aug. 2019), pp. 1–1. ISSN: 0018-9448. DOI: 10.1109/tit.2019.2935768.

[10] Ashwin Machanavajjhala et al. “l-Diversity: Privacy beyond k-anonymity”. In: *Proceedings - International Conference on Data Engineering*. Vol. 2006. 2006, p. 24. ISBN: 0769525709. DOI: 10.1109/ICDE.2006.1.

[11] Arvind Narayanan and Vitaly Shmatikov. “How To Break Anonymity of the Netflix Prize Dataset”. In: (Oct. 2006). arXiv: 0610105 [cs]. URL: http://arxiv.org/abs/cs/0610105.

[12] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems* 10.5 (Oct. 2002), pp. 557–570. ISSN: 02184885. DOI: 10.1142/S0218488502001648.

[13] Jun Tang et al. “Privacy Loss in Apple’s Implementation of Differential Privacy on MacOS 10.12”. In: (2017). arXiv: 1709.02753v2.

[14] Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. “Privacy-Preserving Adversarial Networks”. In: (Dec. 2017). arXiv: 1712.07008. URL: http://arxiv.org/abs/1712.07008.

[15] Max Vasterd. “maxxiefjv/privpack: v0.8-beta”. In: (Nov. 2020). DOI: 10.5281/ZENODO.4252689. URL: https://doi.org/10.5281/zenodo.4252689%7B%5C#%7D.X6WQsUJGwjk.mendeley.

[16] Sergio Verdú. “α-Mutual information”. In: *2015 Information Theory and Applications Workshop, ITA 2015 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., Oct. 2015, pp. 1–6. ISBN: 9781479971954. DOI: 10.1109/ITA.2015.7308959.

[17] Ye Wang, Yuksel Ozan Basciftci, and Prakash Ishwar. *Privacy-Utility Tradeoffs under Constrained Data Release Mechanisms*. Tech. rep. 2017. arXiv: 1710.09295v1.