PRIVACY PRESERVATION AND EVALUATION IN MACHINE LEARNING

Joseph Pedersen

Submitted in Partial Fullfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Approved by:
Dr. Kristin Bennett, Chair
Dr. William Wallace
Dr. John Mitchell
Dr. Isabelle Guyon

Department of Industrial and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York

July 2022

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to thank my advisor Dr. Kristin Bennett and co-advisor Dr. Isabelle Guyon for their patience guiding me along this journey, and for the invaluable expertise that they shared during the many hours of collaboration which I enjoyed participating in and mentorship which I was fortunate to receive. I look forward to continuing this journey, and I hope that I can share some of the wisdom I have received with other researchers that I meet.

I would also like to thank Dr. John Erickson and fellow doctoral candidate Karan Bhanot for their many helpful discussions.

ABSTRACT

This thesis aims at improving methods for preserving privacy in machine learning for both generative and predictive models, and evaluating the privacy lost. Creating useful machine learning models requires the use of training data, but comes with a privacy risk to the subjects whose records are used.

First, we address the problem of making sensitive data that must stay in a secured environment available for education or research by replacing it with sufficiently resemblant synthetic data that can be used to train accurate predictive models. Next, we consider the scenario in which researchers want to release models trained on sensitive data without the models revealing with high confidence which data were used for training. Two approaches are investigated and connections are made between them: making models differentially private, and protecting against membership inference attacks.

We derive tight, componentwise bounds for the loss of a Wasserstein GAN, as well as new bounds on the norm of the loss of the gradient penalty term, and use those in a novel algorithm for a differentially private WGAN-GP. We evaluate the performance of this algorithm by using it to synthesize three real medical datasets, and using those synthetic datasets to replicate published medical studies. We find that the algorithm suffered less mode collapse than the non-differentially private version.

We also develop a framework for formally analyzing the worst case privacy attack scenario. We prove several lower bounds on the accuracy of an attacker in this framework on model trainers that overfit or are insufficiently random. We also prove that any sample learned from incurs a risk of privacy loss, and that under certain assumptions black-box attacks are optimal. From our theoretical analyses, we motivate a novel protection method which we demonstrate can be used to improve the privacy of already well-protected models and simultaneously increase their accuracy.

CHAPTER 1
Introduction

1.1 Motivation

We live in the information age. According to cloud software company Domo, Inc.'s seventh edition of *Data Never Sleeps*, there were an average of 511,200 tweets, 18,100,100 text messages, and 188,000,000 emails each minute in 2019.[10] Surveys conducted by Pew Research Center indicate that approximately 96% of U.S. adults own a cell phone and 81% own a smart phone.[35] These smart phones allow users to carry a device in their pockets enabling them to bank, shop, browse the internet, connect to social media, take online courses, and conduct a myriad of other activities in their digital lives at any time of day and just about anywhere they go. Statista Research Department reports that there are also now over 60 million Americans with wearable devices[44]. These are potentially collecting data 24 hours a day, such as heart rate, activity levels, sleep, and GPS location. Worldwide, the total number of devices connected to the internet now far exceeds the population of the world, and is continuing to grow exponentially.[5] An International Data Corporation report forecasts that the total data in the "Global Datasphere" will grow from 33 zettabytes (33 billion terabytes or 3.3×10^{22} bytes) in 2018, to 175 zettabytes by 2025.[36]

Businesses want to use this wealth of information to design products and services that are in demand, and to better market those products and services to the right customers. Domo Inc. argues that "the ability to make data-driven decisions is crucial to any business," and states that "with each click, swipe, share, and like, a world of valuable information is created."[10] Governments and other organizations may be able to use these data to set the best policies, make the most effective use of available resources, and formulate the best plans for the future. Researchers in nearly every academic discipline formulate hypotheses which may be supported or refuted with evidence from these data.

Individuals and society collectively stand to benefit greatly from the proper use of these data. We all want the best products and services available to us. We want our government to have the best policies possible, and make effective use of our resources. We want aca-

demics to solve or improve upon the problems facing society. We want the best healthcare for ourselves and our families.

But these data are not benign. There are many potential risks to governments, companies, and individuals from the analytical results, trained models, or other research products output from use of their data, if released to the public or somehow accessed by an adversary. Militaries may want to be able to create combat models, without an enemy who gains access to them from being able to determine information about specific units or servicemen. Companies may want to create pricing models without competitors being able to learn any trade secrets.

For people who have their data analyzed, if it can be linked back to them, besides the obvious serious threat of identity theft. Some companies use social media to screen potential hires.[49] Every tweet a person has ever made in life could be used to decide if he or she will be hired. Companies can use information gained about you (e.g. from internet cookies) to over charge you for products and services, such as when "Orbitz was showing more expensive hotels to users who were browsing from Mac computers, a practice known as price steering."[48] Wilson and his colleagues at Northeastern University also "saw price discrimination", the practice of showing different prices to different customers, "from Home Depot, Sears, Cheaptickets, Orbitz, Priceline, Expedia and Travelocity, with product prices varying from user to user."[48] If someone's health records indicate that they are at increased risk of any diseases or other ailments, depending on the laws of their country that information being released to insurers could cause them to be denied health insurance or to be charged considerably more for it.

Anyone aware of these and other dangers knows that they have to take measures to ensure that their information is kept private. But how private? To read all of the privacy policies we encounter each year would take an estimated 76 work days.[27] And most Americans do not even understand what a privacy policy is; that is, they incorrectly believe that if a company has a privacy policy that means that the company keeps all personal information about its users confidential.[42]

As legislators address the potential dangers, laws requiring greater privacy protection may become the norm, such as the California Consumer Privacy Act which passed in 2018.[31] Companies may be forced to give people the right to easily opt out of all data sharing. In the United States, the Health Insurance Portability and Accountability Act of 1996 (HIPAA) already requires the protection of certain health information. And as people become more aware of the potential dangers associated with the use of their data, they may become more and more protective of it, and choose not to authorize it for any additional uses, such as research, whether commercial, government, or non-profit. This would be a huge loss for these governments, non-profits, commercial enterprises, and society, so it is in all of our interests to ensure that we protect individual privacy and engender the trust necessary for people to grant us their data for appropriate uses. But what constitutes a breach of individual privacy? And how can we protect against that? The answers to these simple sounding questions are not as straightforward as one may naively believe. Some may think that by simply removing fields such as name and social security number, referred to as **identifiers**, that all data is rendered benign for individual privacy purposes. As we will see, this is unfortunately not true. In fact, many potential techniques for preserving privacy will be shown to be flawed.

The main culprit in the fight to preserve privacy is **auxiliary information**, which is any information available to users of the data that is obtained from other sources. We will see that the use of auxiliary information will foil many intuitive attempts to preserve privacy. But we will see that one technique, that of **differential privacy**, is completely immune to auxiliary information. However, differential privacy does have its limitations. In particular, it is difficult to achieve adequate privacy while maintaining utility when learning from modest sized datasets.

The applications of a differentially private synthetic data generator could be far reaching. Right now, de-identifying data is time consuming and expensive, with large fines for mistaken releases. This severely limits the number of widely available data sources. To access confidential data, researchers need to complete the proper training, sometimes need expensive background checks, and potentially need access to secure computing environments. All of this limits the data used for education and research. It also limits the number of re-

searchers who can work on problems requiring access to specific private data.

Having a sufficiently private synthetic data generator which produces high quality output will transform that landscape. First, for education, it will enable students from disciplines ranging from biology and health care, to engineering and physics, to political science and sociology, to be able to use representative data sets trained on real data, without needing special training, background checks, or access to a secure computing environment. Professors will be able to hang the data sets right on their websites. With the data explosion happening in the world, and the ever growing need for professionals in many disciplines to have basic data sciences skills, this will greatly enable the development of courses with exercises on realistic data, especially in schools which cannot afford access to secure computing environments, perhaps even in high schools across the United States.

Beyond educational use, if the synthetic data can be made representative enough so that insights gained from exploratory analyses are likely to generalize to the real data, then researchers around the world will be able to use synthetic data, trained on important data sets, right on their laptops, rather than needing special training or access to a secure computing environment. This will save on the costs associated with the special training, computing environments, and background checks. It will also widen the community of potential researchers who can explore the data. Organizations can put their data right in a Kaggle competition and know that individual privacy is being preserved.

However, for modest sized datasets, it may not be possible to train a high quality synthetic data generator with sufficient privacy preservation. For this reason, it is important to have widely applicable methods for protecting privacy, and for evaluating the degree of privacy protection achieved. These methods should be easy to implement so that everyone working with data can protect the privacy of their data. For example, in the United States military, we have many sources of data that we restrict even from members of the military, even if we are not concerned that they will intentionally leak the data, but because we have to be concerned that the results of their analysis or the other products of their research might unintentionally leak private information.

1.2 Problem Statement

This research will be focused on improving techniques for privacy-preserving machine learning in two main areas. Our first objective will be to make a differentially private version of a more recent GAN architecture known as a Wasserstein GAN with gradient penalty (WGAN-GP), and evaluate its performance by creating synthetic versions of data used in published medical studies and showing that classifiers trained on the synthetic data predict well on the real data. We evaluate its privacy both in terms of the theoretical bounds of the differential privacy parameter ε and empirically using a nearest neighbor based metric. Our second objective will be to create a novel privacy preservation method that can be used in situations for which differential privacy does not provide adequate protection. We will develop a framework for conducting the optimal attack of membership privacy, and prove theorems in our formal analysis of this framework. The formal analysis will motivate our privacy preservation method, which will be widely applicable and easy to implement, with a self-evaluation providing an estimate of the protection achieved.

1.3 Thesis Outline

In Chapter 2, we provide a review of the literature, starting with privacy preservation in general, then differential privacy in particular. Next we discuss synthetic data generation, focusing on generative adversarial networks (GANs), and introducing the Wasserstein GAN (WGAN) and the Wasserstein GAN with gradient penalty (WGAN-GP). We will describe the HealthGAN that our work builds upon, and the nearest neighbor based metric used to evaluate its privacy preservation.

In Chapter 3, we derive a tight bound on the gradient of the WGAN loss function, as well as a bound on the norm of the gradient of the gradient penalty term of a WGAN-GP. We show that our bounds do not require weight clipping, and present an algorithm for a WGAN-GP that is provably differntially private. We evaluate the quality of the synthetic data that it generates using the gold standard of training classifiers from published medical studies on synthetic data, and evaluating those classifiers on test data, with good results. We find that differential privacy improves the quality of the output, by slightly reducing mode collapse.

In Chapter 4, we improve the nearest neighbor based metric, proving that our modified version is unbiased. We also describe how it can be efficiently implemented, and made suitable for data with discrete distances, such as binary data. Then we introduce a new privacy attack methodology, for an "Leave Two Unlabeled (LTU) Attacker" which has the most information possible for a membership inference attack. We present three theoretical results demonstrating strategies that are effective at attacking any model that is overfit or insufficiently random.

In Chapter 5, we formalize our LTU Attacker framework so that we can make connections between it and differential privacy. We identify an important aspect of the problem of membership privacy protection, related to different levels of access to the trained model and model trainer (black-box vs white-box), to be the distinction of the model as a mathematical function and the model as implemented in a data structure (bits in memory). We prove that any time a sample is learned from, it incurs a risk of privacy loss, and that under certain hypotheses, a black-box attack is optimal. We use these theoretical analyses to motivate a novel defense strategy called "optimally soft labels", which we should can improve the privacy of an already well-defended model, while simultaneously improving its utility.

In Chapter 6, we restate our contributions, and describe how our work can be built upon with future work.

CHAPTER 2
Literature Review

Before describing our research on privacy-preserving machine learning, we will first review the literature on the problem of privacy preservation in general, and the techniques of differential privacy in particular. We will also review the concept of synthetic data, and the use of generative adversarial networks to generate synthetic data. We will look at methods for evaluating the privacy of maching learning models, such as metrics to empirically estimate how privacy models are, and algorithms to attack machine learning models to see how vulnerable they are.

2.1 Privacy Preservation

Anyone unfamiliar with the literature on privacy preservation may naively believe that data can be made private by simply removing obvious identifiers (de-identification) such as name and social security number. One popular example of the failure of de-identification is when in 1997, Latanya Sweeney, then a graduate student at MIT, showed that the medical records of Massachusetts Governor William Weld could be identified in a de-identified database by linking birth date, gender, and zip code to a voter list.[45] In a testimony that she gave to the Privacy and Integrity Advisory Committee of the Department of Homeland Security in 2005, she said:

"One problem is that people don't understand what makes data unique or identifiable. For example, in 1997 I was able to show how medical information that had all explicit identifiers, such as name, address and Social Security number removed could be reidentified using publicly available population registers (e.g., a voter list). In this particular example, I was able to show how the medical record of William Weld, the governor of Massachusetts of the time could be re-identified using only his date of birth, gender and ZIP. In fact, 87% of the population of the United States is uniquely identified by date of birth (e.g., month, day and year), gender, and their 5-digit ZIP codes. The point is that data that may look anonymous is not necessarily anonymous."[45]

The attributes birth date, gender, and zip code, do not individually identify a unique record owner, but their combination is referred to collectively as a **quasi-identifier** (QID), which is "a set of attributes that could potentially identify record owners."[17] Identifying records by linking the quasi-identifiers to those same quasi-identifiers in another database (e.g. the voter list) that contains identifiers, is called a **linkage attack**.[17] The quasi-identifiers can also be determined from sources other than a second database. For example, anyone who personally knows your date of birth, gender, and zip code, whether they are a coworker, neighbor, or stranger who noted birthday wishes to you on social media, could use that information for a linkage attack.

In general, any information used for a linkage attack is referred to as **auxiliary information**. Since the auxiliary information available to a potential attacker cannot be controlled or known when creating the database, it may seem reasonable to attempt to remove not only identifiers, but also any potential quasi-identifiers. One reason that technique may not be suitable is that those attributes may contain information important for analysis. For example, in analysis of medical data, gender is often an important determinant, as is age, although perhaps not the exact date of birth. If date of birth was replaced by age, that would be an example of an anonymization technique known as **generalization**, which "replace[s] values of specific description, typically the QID attributes, with less specific description."[17]

A famous example of de-anonymization using attributes that did not contain familiar quasi-identifiers is when the user movie ratings released as part of the Netflix challenge were de-anonymized by linking them to ratings from the Internet Movie Database (IMDb), by Narayanan and Shmatikov. The authors explain that "Datasets containing 'micro-data,' that is, information about specific individuals, are increasingly becoming public - both in response to 'open government' laws, and to support data mining research."[29] Further, they describe that "[m]icro-data are characterized by high dimensionality and sparsity. Informally, micro-data records contain many attributes, each of which can be viewed as a dimension."[29] They develop a general de-anonymization algorithm that they show can be used under modest assumptions on the distribution of records, using only a few attributes to de-anonymize sparse databases. "For sparse data sets, such as most real-world data sets of individual transactions, preferences, and recommendations, very little background knowl-

edge is needed (as few as 5-10 attributes in our case study)."[29]

It might be questioned whether Netflix users consider this a breach of privacy, given that their movie ratings on IMDb were open to the public. However, it is not hard to imagine how a person might review certain movies publicly, while at the same time not wanting their entire movie viewing history to be public knowledge. As reported by Ryan Singel in Wired magazine, an "in-the-closet lesbian mother sued Netflix for privacy invasion, alleging the movie-rental company made it possible for her to be outed when it disclosed insufficiently anonymous information about nearly half-a-million customers as part of its $1 million contest."[41] He also notes that "video-rental records are protected records in the United States - a reaction to a reporter getting Supreme Court-nominee Robert Bork's rental history from a video store."[41] This example is highly instructive. We have a person who likely thought that the information she exposed to the public, her ratings of some movies on IMDb, posed her very little risk, if any. We also have a company which likely believed that, because the data they released had no identifiers or obvious quasi-identifiers, that they were not breaching the privacy of their users. Meanwhile, the combination of these actions had a potentially life altering impact on at least one individual, and a potentially severe financial impact on the company.

Given the results so far, it may seem like any micro-data are susceptible to privacy attacks, and therefore that only summary statistics of a database are safe to release. Unfortunately, a simple hypothetical example shows that even summary statistics are not immune to linkage attacks with auxiliary information. Imagine that at Rensselaer Polytechnic Institute, a professor is conducting research for which socio-economic status is an important determinant, and includes the following statistic in a scholarly journal submitted to arXiv on February 10, 2020: of the 1624 Freshman at RPI, 9.5% have parents whose total income in 2019 was less than $30,000. Another professor, unaware of the first professor's article, includes the following statistic in his/her article, submitted to arXiv on February 24, 2020: of the 1623 Freshman at RPI, 9.4% have parents whose total income in 2019 was less than $30,000. Students at the university, noticing those two articles, may investigate and find that no students were admitted during that two week period in the middle of the spring semester, and that exactly one student dropped out in that time frame, who they may be

able to identify, thereby determining that his/her parents' total income in 2019 was less than $30,000. That student may rightly feel like his/her privacy has been violated.

If a release as benign seeming as a count and a percentage can result in a breach of privacy, it may seem like any attempt of preserving privacy while at the same time releasing useful information is futile. Fortunately, we will see that there is a notion of privacy preservation which is compatible with statistical learning, but first we need to consider what is meant by a breach of privacy. We will see that a certain unrealistic privacy goal is unattainable, but that another which is attainable will motivate the formal definition of differential privacy.

In "Calibrating Noise to Sensitivity in Private Data Analysis", for which the authors were awarded the 2017 Gödel Prize, Dwork et al.(2006) suggest that "the goal of a privacy-preserving statistical database is to enable the user to learn properties of the population as a whole while protecting the privacy of the individual contributors."[12] Before describing their proposed level of protection, we will first note a statement that Dwork and Naor (2010) quote in a later work, "A 1977 paper of Dalenius [9] articulated the following desideratum: 'access to a statistical database should not enable one to learn anything about an individual that could not be learned without access' "[13] They explain that this notion of absolute privacy is analogous to "semantic security - roughly, everything computable about the plaintext given the ciphertext should be computable without access to the ciphertext [24]".[13] As a simple example of this level of security in the domain of cryptography, imagine an enemy that intercepts an encrypted message, and also captures a decrypted copy of the same message that is missing only the name of an important location. Semantic security requires that there is no way for the enemy to use the decrypted message to decrypt the location name.

Although "semantic security for cryptosystems can be achieved (under reasonable computational assumptions),"[13] they give a hypothetical example to illustrate that such absolute privacy is inconsistent with statistical learning. We will paraphrase an example presented in a later work by Dwork and Roth (2014)[14]. Imagine that, 100 years ago, a group of researchers conducted a longitudinal study of 100,000 people and released their findings "Smoking increases the likelihood of getting cancer." Imagine that John Doe's medical insurer knew from information that they had collected on him that he smoked. They may

have decided (based on the laws at the time) to no longer cover him, or to greatly increase his insurance costs. Did the researchers violate John Doe's privacy? We will consider two cases. First, assume that John Doe was not even part of the study. Most people would probably conclude that the researchers could not have violated John Doe's privacy if they did not have any of his information. However, note that this does violate Dalenius' desideratum; the insurance company has learned that John Doe has increased likelihood of getting cancer. They learned something about John Doe that they did not know without the study. In analogy to the cryptographic example, the study result is the encrypted message, that John Doe smokes is the decrypted message (auxiliary information), and that John Doe has increased likelihood of getting cancer is the missing piece decrypted.

Besides the intuition that an individual's privacy cannot be violated by someone who did not have access to their information, we also see that the study result matches the goal of statistical learning described by Dwork et al.(2006) The result was about the population, not about an individual. This example generalizes. If we learn anything about a population (the goal of statistical learning), then we can combine that with the auxiliary information that an individual is part of that population, to learn something about that individual. Therefore, Dalenius' desideratum is incompatible with statistical learning. Now consider the case that John Doe was part of the study. Then was his privacy violated? It is still true that the result was about the population as a whole, not about any individual. However, the main argument that his privacy was not breached, the argument that motivates the definition of differential privacy, is that the result would not have changed *had* John Doe decided not to participate in the study. The study included 100,000 participants, and one participant (whether he developed cancer or not) could not significantly affect the study results.

2.2 Differential Privacy

Before giving the formal definition of differential privacy, we will first give an example of **randomized response** derived from the description given by Dwork and Roth (2014)[14]. Imagine that you are conducting a survey about illegal or taboo behavior. You ask 1,000 participants to answer a few demographic questions, followed by one Yes/No question of interest. The survey has no identifiers. The participants fill out the survey in a booth for privacy, then drop it into a collection box. However, given only these precautions, a par-

ticipant might rightly be concerned that quasi-identifiers would reveal their identity, and make their answer to this question public. Therefore, you institute randomized response. Each participant is given a (presumably fair) coin, and instructed that for the question of interest, they should flip the coin first to see how they should respond. If the first coin flip is heads, they should respond truthfully. If the first coin flip is tails, they should flip the coin a second time to determine their response: heads on the second flip $\rightarrow$ answer "Yes", tails on the second flip $\rightarrow$ answer "No." Now, even if the micro-data from the survey is published, and an individual's response is identified using QID, he/she will at least have plausible deniability: his/her response may have been determined by the coin flips, rather than truthful. Note that this noise in the responses does not prevent us from gaining useful information about the population. If 670 responses were "Yes", then we could estimate the true population proportion of "Yes" responses by estimating that 500 of the 1,000 responses were from coin flips (instead of truthful responses), with approximately 250 "Yes" responses. That would mean that of the 500 truthful responses, $670 - 250 = 420$ were "Yes", resulting in a point estimate of 84% "Yes" responses.

We will see how adding the appropriate amount of noise to any query of a statistical $\mathbb{Z}$ database will ensure a notion of privacy, but first we must present some definitions and propositions, all of which are from Dwork and Roth (2014).[14] A database will be thought of as a multiset of records from a universe $\mathcal{X}$, but will often be represented by its histogram $x \in \mathbb{N}^{|\mathcal{X}|}$ where x_i is a nonnegative integer representing the number of records in x that are of type $i \in \mathcal{X}$.[14]

Definition 2.2.1. *(Distance Between Databases) The ℓ_1 norm of a database x is denoted $\|x\|_1$ and is defined to be:[14]*

$$\|x\|_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i| \tag{2.1}$$

The ℓ_1 distance between two databases x and y is $\|x - y\|_1$.[14]

We will add some notes that Dwork and Roth did not. The universe $\mathcal{X}$ is not necessarily countable, but any database has a finite number of records, so for any x all but

finitely many x_i are zero, which is why we can use summation. Also, note that $x - y$ may not actually denote a database, since some of the entries may be negative, unless we make a special definition of $x - y$ for databases such that $(x - y)_i = |x_i - y_i|$. However, either way the distance between x and y will still be given by the equation in definition 2.2.1.

Definition 2.2.2. *(Randomized Algorithm) A randomized algorithm $\mathcal{M}$ with domain D and discrete range R is a mapping $\mathcal{M} : D \to R$ such that, on input $d \in D$, the algorithm will output $r \in R$ with probability* $\mathrm{p}\,(r|d)$*.[14]*

Dwork and Roth (2014) note that they are only considering discrete ranges, so the probability is given by a probability mass function. We will add that, although some of the queries we consider will actually seem continuous, due to the floating point arithmetic used by computers all responses actually have a finite set of possible return values. Definition 2.2.2 does not exactly match that of Dwork and Roth. The wording and notation and notation has been slightly simplified.

Definition 2.2.3. *(Differential Privacy) A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if $\forall \mathcal{S} \subseteq \mathrm{Range}(\mathcal{M})$ and $\forall x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:[14]*

$$\Pr\left[\mathcal{M}(x) \in \mathcal{S}\right] \leq \exp(\varepsilon) \Pr\left[\mathcal{M}(y) \in \mathcal{S}\right] + \delta \tag{2.2}$$

If $\mathcal{M}$ is $(\varepsilon, 0)$-differentially private, we may refer to it as ε-differentially private.[14]

This is both a renaming of what was referred to as ε-indistinguishability in the original paper, and a generalization of that concept to include the δ term. When $\delta = 0$, the above equation informally says that for an ε-differentially private randomized algorithm, the output is *similar* for **adjacent** databases (databases such that $\|x - y\|_1 = 1$). Similarly, we mean that if ε is small (although not stated in the definition, it should always be nonnegative), then $\exp(\varepsilon)$ is a multiplicative factor close to 1 that bounds how much more likely event $\mathcal{S}$ can be on database y than it is on database x. The smaller ε, the more privacy is preserved. Dwork et al.(2006) "sometimes call ε the *leakage*."[12] For a concrete example, we will return to the

two hypothetical RPI professors mentioned earlier, but first we need to see how to make a query ε-differentially private. First, we need a few more definitions from Dwork and Roth:[14]

Definition 2.2.4. *(ℓ_1-sensitivity) The ℓ_1-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:[14]*

$$\Delta f = \max_{\substack{x,y \in \mathbb{N}^{|\mathcal{X}|} \\ \|x-y\|_1 = 1}} \|f(x) - f(y)\|_1 \tag{2.3}$$

If we think of the function f as a query to a statistical database, the ℓ_1-sensitivity is the maximum difference of the true response on adjacent databases. In particular, we can think of our adjacent databases as being the cancer study database with and without John Doe's health record, or the database of RPI students before and after the one student dropped out. In order to ensure individual privacy, this sensitivity determines how much noise needs added in order for an individual to have an amount of plausible deniability. That was the main thesis of the original paper "Calibrating Noise to Sensitivity in Private Data Analysis". The noise that they used in that paper is from the Laplace distribution. We now give the definitions of the Laplace distribution and the Laplace mechanism from Dwork and Roth (2014):[14]

Definition 2.2.5. *(The Laplace Distribution) The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:[14]*

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right) \tag{2.4}$$

Definition 2.2.6. *(The Laplace Mechanism) Given any funtion $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:[14]*

$$\mathcal{M}_L(x, f, \varepsilon) = f(x) + \mathbf{Y} \tag{2.5}$$

where $\mathbf{Y}_i$ are i.i.d. random variables drawn from $\text{Lap}(\mathbf{Y}_i|\Delta f/\varepsilon)$*.[14]*

Again, thinking of f as the query of a statistical database, $f(x)$ represents the true response to the query, but $f(x) + \mathbf{Y}$ is the (random) ε-differentially private response given by a privacy preserving oracle that uses the Laplace mechanism. Note that if the same query is repeated multiple times, the oracle's responses will vary due to the noise, and an increasingly accurate estimate of the true response will be attainable from those noisy responses. The privacy *leakage* actually adds up in a way that can conveniently be accounted for in algorithms, which will be discussed later.

Returning to the example of RPI professors discussed earlier, if they had queried a statistical database which indicated the true number of students with no noise (hence no privacy about enrollment), but with ε-differentially private responses for the number of students whose total parental income was less than \$30,000 in 2019 (hence privacy about parental income), then the first professor would have received a noisy response of $154 + y_1$, whereas the second professor would have received a noisy response of $153 + y_2$. The random variables y_1 and y_2 would have been independent random variables drawn from the Laplace distribution with $b = 1/\varepsilon$, since the ℓ_1-sensitivity of any counting query is 1 (the maximum that the count can differ on adjacent databases). If $\varepsilon = 0.1$ for both queries, then there would have been approximately a 47.5% chance of the second professor reporting a higher percentage, although the true percentage would have been lower. This noise would give the student who dropped plausible deniability about his parents' income. The standard deviation of the Laplace distribution is $\sqrt{2}b$, so the noisy responses would have had a standard deviation of approximately 4.5 in the number of students, about 3% of the true answer, which is fairly precise given the privacy that it confers. Note that the level of privacy loss that is acceptable is a subjective preference, not something determined by the theory, but Dwork (2008) has stated that "we tend to think of ε as, say, 0.01, 0.1, or in some cases, $\ln 2$ or $\ln 3$."[11]

There are generalizations of differential privacy and the Laplace mechanism which may prove useful in creating a differentially private synthetic data generator with the highest possible output quality. We will briefly mention some, and state some important properties, citing the sources where the reader can find more details. As already mentioned, beyond the original ε-differential privacy (referred to in the original paper as ε-indistinguishability),

there is (ε, δ)-differential privacy with $\delta \neq 0$. This would allow ε-differential privacy to be violated with probability at most δ. This could constitute a complete loss of privacy for an individual in the database, if δ is too large. Dwork and Roth (2014) note that, "Typically we are interested in values of δ that are less than the inverse of any polynomial in the size of the database. In particular, values of δ on the order of $1/\|x\|_1$ are very dangerous: they permit 'preserving privacy' by publishing the complete records of a small number of database participants."[14]

When $\delta \neq 0$, the **Gaussian mechanism** is used instead of the Laplace mechanism. In the Gaussian mechanism, a random vector of noise is added to the true response of the query function, with each coordinate of the noise drawn from an i.i.d Gaussian distribution with mean 0 and standard deviation $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta_2(f)/\varepsilon$.[14] Here, $\Delta_2(f)$ is the ℓ_2-sensitivity of the query function, which is defined similarly to the ℓ_1-sensitivity, but with the ℓ_2-norm on the difference of the true responses on adjacent databases (adjacent databases are still defined using the ℓ_1-norm).

An extremely important property is that "[d]ifferential privacy is immune to post-processing: A data analyst, without additional knowledge about the private database, cannot compute a function of the output of a private algorithm $\mathcal{M}$ and make it less differentially private."[14] Note that "additional knowledge" refers only to the "private database", not to any other information. This tells us that no amount of auxiliary information can be used to undermine the notion of privacy that differential privacy confers. Dwork and Roth (2104) make this formal by proving the following proposition:[14]

Proposition 2.2.1. *(Post-Processing) Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomized algorithm that is (ε, δ)-differentially private. Let $f : R \to R'$ be an arbitrary randomized mapping.*
Then $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R'$ is (ε, δ)-differentially private.[14]

Another very useful fact is that multiple queries on the same database, when considered in totality, have an easy to compute upper bound on the total privacy leakage. Dwork and Roth (2014) note that, " 'the epsilons and the deltas add up.': the composition of k

differentially private mechanisms, where the ith mechanism is $(\varepsilon_i, \delta_i)$-differentially private, for $1 \leq i \leq k$, is $(\sum_i \varepsilon_i, \sum_i \delta_i)$-differentially private."[14] There are more sophisticated composition theorems which can achieve tighter bounds for certain mechanisms, but this theorem gives a very simple recipe to make any machine learning algorithm differentially private: add the appropriate amount of noise to make each step private, then add up the total privacy leakage for all steps. However, Dwork and Roth (2014) suggest, "by rethinking the computational goal, one can often obtain far better results than would be achieved by methodically replacing each step of a non-private computation with a differentially private implementation."[14]

Another useful mechanism is the exponential mechanism of McSherry and Talwar (2007), which can be used to make a differentially private version of any query function $f : D \to R$ on which there is a utility function $u : D \times R \to \mathbb{R}$, as long as the quantity $\int_r \exp(\varepsilon u(d, r))\mu(r)$, where $\mu(r)$ is a base measure on R, is bounded.[28] Dwork and Roth (2014) note that "[t]he exponential mechanism was designed for situations in which we wish to choose the 'best' response but adding noise directly to the computed quantity can completely destroy its value."[14] It is a generalization of the Laplace mechanism, which "can be captured by taking $u(d, r) = -|f(d) - r|$"[28] In fact, McSherry and Talwar (2007) note that their exponential mechanism "can capture any differential privacy mechanism $\mathcal{M}$ by taking $u(d, r)$ to be the logarithm of the probability density of $\mathcal{M}(d)$ at r"[28] Kasiviswanathan et al.(2008) used the exponential mechanism to demonstrate that any concept learnable in the probably approximately correct (PAC) framework can be learned differentially privately. "[I]f some concept class C is learnable by any algorithm, not necessarily a private one, whose output length in bits is polynomially bounded, then C is learnable privately using a polynomial number of samples (possibly in exponential time)"[22]

2.3 Synthetic Data

Dwork et al.(2010) define a **synthetic database** as "a data structure that 'looks like' a database, in that its rows are drawn from the same universe $\mathcal{X}$ from which the database rows are drawn."[15] Rather than have a privacy-preserving oracle with access to the true database release differentially private responses to queries, the privacy preservation strategy is to create a synthetic database (fake data) that is representative enough of the real database

that queries on the synthetic database are good enough estimates to the true responses to the same queries on the real database. Meanwhile, the release of the entire synthetic database must still be preserve differential privacy. Initial results seemed to limit this approach. Dwork et al.(2010) showed that "for any non-interactive mechanism...", such as a synthetic database, "there exist low-sensitivity functions $f(x)$ which cannot be approximated at all based on [it], unless the database is very large: If each database entry consists of d bits, then the database must have $2^{\Omega(d)}$ entries in order to answer all low-sensitivity queries - even to answer queries from a restricted class called sum queries. In other words, a non-interactive mechanism must be tailored to suit certain functions to the exclusion of others"[15]

However, Blum et al.(2008) used the exponential mechanism to "show that for discretized domains, for any concept class with polynomial VC-dimension, it is possible to release differential-privacy-preserving [synthetic] databases that are simultaneously useful for all queries in the concept class."[4] Dwork et al.(2010) note that the "remarkable result of Blum, Ligett, and Roth shows that differential privacy is possible even in cases when the number of counting queries is much larger than n^2"[15], where n is the number of records in the real database. Dwork et al.(2010) define a generalization of a synthetic database, called a **synopsis**, which is any data structure that, "when presented with any $q \in Q$ [a set of queries on database x], returns an approximation to $q(x)$."[15] They develop a boosting algorithm that takes a base synopsis generator which is accurate for most queries in some family Q with bounded sensitivity, and trains it to produce a synopsis which provides accurate answers to all queries in Q, while maintaining privacy. This algorithm is general, working for any family of queries Q and synopsis with a base synopsis generator.

One technique which can be used to create synthetic data is a **generative adversarial network** (GAN), developed by Goodfellow et al.(2014), in which two models are trained simultaneously: a generative model $G : Z \to \mathcal{X}$ which maps random vectors drawn from a fixed, arbitrarily chosen prior p_z to the data space, and a discriminative model $D : \mathcal{X} \to [0, 1]$ which tries to model the probability that $\mathbf{x} \in \mathcal{X}$ was drawn from the distribution of the real data that it was trained on, p_{data}, instead of from the distribution $p_g = G(p_z)$. These two models are *adversaries* in the sense that G is trained to try to map the random vectors from p_z to elements of the data space which D mistakenly classifies as real (i.e. drawn from p_{data}

instead of p_g). Goodfellow et al.(2014) state, "[i]n other words, D and G play the following two-player minimax game with value function $V(G, D)$:"[18]

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(\mathbf{x})] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(\mathbf{z})))] \tag{2.6}$$

Goodfellow et al.(2014) prove that "If G and D have enough capacity, and at each step of [their algorithm], the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion [given]...", "then p_g converges to p_{data}."[18] In practice, however, they note that a disadvantage of their approach is "that D must be synchronized well with G during training (in particular, G must not be trained too much without updating D, in order to avoid 'the Helvetica scenario' in which G collapses too many values of z to the same value of x to have enough diversity to model p_{data})."[18]

A technique developed to improve upon the training of adversarial nets is the **Wasserstein GAN** (WGAN) of Arjovsky et al.(2017). They study the impact on convergence from choosing from the "various ways to define a distance or divergence" between p_{data} and p_g for the loss function, noting that "[i]nformally, a distance ρ induces a weaker topology when it makes it easier for a sequence of distributions to converge."[3] They define a WGAN, which uses an approximation to the Earth Mover distance instead of the Jensen-Shannon divergence used by Goodfellow et al.(2014).

Definition 2.3.1. *(EM distance) The Earth-Mover (EM) distance or Wasserstein-1[3] can be thought of informally as the minimum 'cost' necessary to transport the mass in the distribution p_{data} into the distribution p_g, and is formally defined as:[3]*

$$W(p_{data}, p_g) = \inf_{\gamma \in \Pi(p_{data}, p_g)} \mathbb{E}_{(x,y) \sim \gamma} \left[\|x - y\| \right] \tag{2.7}$$

where $\Pi(p_{data}, p_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p_{data} and p_g.[3]

Arjovsky et al.(2017) "theoretically show that the corresponding optimization problem is sound" and "empirically show that WGANs cure the main training problems of GANs.

In particular, training WGANs does not require maintaining a careful balance in training of the discriminator and the generator, and does not require a careful design of the network architecture either."[3] Their algorithm takes advantage of the Kantorovich-Rubinstein duality to reformulate the optimization to a "supremum over all the 1-Lipschitz functions."[3]

Definition 2.3.2. *(K-R duality) If the metric space is compact, the Earth-Mover (EM) distance or Wasserstein-1 can be reformulated as:[3]*

$$W(p_{data}, p_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_{data}}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)] \tag{2.8}$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. Note that if we replace $\|f\|_L \leq 1$ for $\|f\|_L \leq K$ (consider K-Lipschitz for some constant K) then we end up with $K \cdot W(p_{data}, p_g)$.[3]

To achieve this, Arjovsky et al.(2017) "train a neural network parameterized with weights w [for the *critic*] lying in a compact space."[3] The discriminator in this GAN architecture is referred to instead as the *critic*, since it is not trained to classify elements of the data space as either real or fake, as the discriminator in the original GAN architecture of Goodfellow et al.(2014) is, although it is still *adversarial* to the generator in that they are playing the two-play minimax game:

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim p_{data}}[D(\mathbf{x})] - \mathbb{E}_{\tilde{x} \sim p_g}[D(\tilde{x})] \tag{2.9}$$

"where $\mathcal{D}$ is the set of 1-Lipschitz functions and p_g is once again the model distribution implicitly defined by $\tilde{x} = G(z), z \sim p_z$."[20]

Arjovsky et al.(2017) achieve compactness through weight clipping, although they note that "[w]eight clipping is a clearly terrible way to enforce a Lipschitz constraint"[3] for various difficulties that it imposes on the training. Gulrajani et al.(2017) developed an improved WGAN-GP that uses a "*gradient penalty*" instead of weight clipping, "which does not suffer from the same problems."[20] Their proposed gradient penalty would force the norm of the gradient to be bounded by 1, making it 1-Lipschitz. However, "[t]o circumvent tractability

issues, [they] enforce a soft version of the constraint with a penalty on the gradient norm for random samples $\hat{x} \sim p_{\hat{x}}$. [Their] new objective is:" [20]

$$L = \mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})] - \mathbb{E}_{x\sim p_{data}}[D(x)] + \lambda\, \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right] \tag{2.10}$$

Gulrajani et al.(2017) note that they define $p_{\hat{x}}$ "sampling uniformly along straight lines between pairs of points sampled from the data distribution p_{data} and the generator distribution p_g," [20] which is motivated by the fact, proved in proposition 1, that the optimal critic D^* "has gradient norm 1 almost everywhere under p_{data} and p_g." [20] The two-player minimax game of equation (2.9) can have this one loss function for two reasons. First, the D^* that maximizes equation (2.9) will minimize the negative of that equation. Second, since the generator G cannot effect $\mathbb{E}_{x\sim p_{data}}[D(x)]$, it will try to minimize equation (2.9) by minimizing $\mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})]$. These facts combined allow equation (2.9) to be reformulated into an equation that both the discriminator (critic) and generator want to minimize, plus a regularization term. "All experiments in [their] paper use $\lambda = 10$, which [they] found to work well across a variety of architectures and data sets." [20]

Yale et al.(2019) created a deep neural network implementation of this WGAN-GP architecture, named HealthGAN in a later paper, to generate synthetic databases of electronic health records (EHR). Their "end-to-end methodology [is designed] to retain instructional utility, while preserving privacy to a level, which meets regulatory requirements." [54] They developed novel metrics for measuring utility, resemblance, and privacy loss; and used those metrics to compare their method to several others on data sets from the Medical Information Mart for Intensive Care (MIMIC-III) database. For preprocessing, "since HealthGAN requires numeric features, categorical features are converted into values between 0 and 1 using the synthetic data vault formulation." [53] Then, HealthGAN is trained on a random split of half of the data from a data set, with the other half used for testing.

The utility of HealthGAN was assessed "by using the synthetic data to train a classifier to predict patient mortality, then testing on the real test data set." [54] Yale et al.(2019) showed that HealthGAN had the best overall performance on resemblance, privacy loss, and utility, compared to four viable candidate methods. Their subsequent work also added two

more tests to the battery, membership inference and discriminator testing. "Membership inference attacks seek to infer membership of individual training instances of a model to which an adversary has black-box access."[47] Yale et al.(2019) assessed membership inference susceptibility by classifying points in the training data R_{tr} and test data R_{te} as belonging to the training data if their distance to their nearest neighbor in the synthetic data was below some threshold, then measuring the area under the curve (AUC) for the receiver operating characteristic (ROC). They also assessed the quality of the synthetic data generator by measuring the Wasserstein distance between the distribution of the modeled data and the distribution of the real data. "The farther this distance is from zero, the more likely the data is to be synthetic."[53] Again HealthGAN was shown to have the best overall performance.

2.4 Privacy Evaluation

In order to evaluate how private machine learning models are, there are two main approaches. The first is to prove theoretical bounds on the learning algorithm to show that the model learned will be private. The main example of this presented thus far is differential privacy. One disadvantage is that the bound proven might not be easily interpretable or practically significant. For example, in differentially private machine learning, a value of $\varepsilon = 10$ is "consistently used" [8] in the literature, but that value only guarantees a bound on the probability of membership that an attacker can infer of 99.995%, which is clearly not much privacy. The second approach is to quantify the degree to which privacy is preserve through empirical means, such as through some privacy metric, or by demonstrating how effectively its privacy can be attacked.

To measure resemblance and privacy, Yale et al.(2019) define the **nearest neighbor**

Adversarial Accuracy $\mathcal{AA}_{TS}$ between two sets T and S:[54]

$$
\begin{aligned}
S_S &= \left\{\left(\mathbf{x}_S^1, y_S^1\right), \ldots, \left(\mathbf{x}_S^n, y_S^n\right)\right\} \text{ is a sample from distribution } P_S \\
S_T &= \left\{\left(\mathbf{x}_T^1, y_T^1\right), \ldots, \left(\mathbf{x}_T^n, y_T^n\right)\right\} \text{ is a sample from distribution } P_T \\
d_{TS}(i) &= \min_j \|\mathbf{x}_T^i - \mathbf{x}_S^j\| \text{ is the distance from } \mathbf{x}_T^i \in S_T \text{ and its nearest neighbor in } S_S \\
d_{ST}(i) &= \min_j \|\mathbf{x}_S^i - \mathbf{x}_T^j\| \text{ is the distance from } \mathbf{x}_S^i \in S_S \text{ and its nearest neighbor in } S_T \\
d_{TT}(i) &= \min_{j, j\neq i} \|\mathbf{x}_T^i - \mathbf{x}_T^j\| \text{ is the distance from } \mathbf{x}_T^i \in S_T \text{ to its nearest neighbor in } S_T \setminus \{\mathbf{x}_T^i\} \\
d_{SS}(i) &= \min_{j, j\neq i} \|\mathbf{x}_S^i - \mathbf{x}_S^j\| \text{ is the distance from } \mathbf{x}_S^i \in S_S \text{ to its nearest neighbor in } S_S \setminus \{\mathbf{x}_S^i\} \\
\mathcal{AA}_{TS} &= \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{1}\left(d_{TS}(i) > d_{TT}(i)\right) + \frac{1}{n}\sum_{i=1}^n \mathbf{1}\left(d_{ST}(i) > d_{SS}(i)\right)\right) \qquad (2.11)
\end{aligned}
$$

The **resemblance loss** of the synthetic data to each of the training data R_{tr} and test data R_{te} is defined to be the expected value of the nearest neighbor Adversarial Accuracy. The **privacy loss** is defined to be the expected value of the difference between the nearest neighbor Adversarial Accuracy on the test set and the nearest neighbor Adversarial Accuracy on the training set:[54]

$$
\begin{aligned}
\text{TrResemblLoss} \quad (\text{Train}\mathcal{AA}) &= \mathbb{E}_{S_1 \sim G}\left[\mathcal{AA}_{R_{tr}S_1}\right] \qquad (2.12) \\
\text{TeResemblLoss} \quad (\text{Test}\mathcal{AA}) &= \mathbb{E}_{S_2 \sim G}\left[\mathcal{AA}_{R_{te}S_2}\right] \qquad (2.13) \\
\text{PrivacyLoss} \quad &= \mathbb{E}_{S_1, S_2 \sim G}\left[\mathcal{AA}_{R_{te}S_2} - \mathcal{AA}_{R_{tr}S_1}\right] \qquad (2.14)
\end{aligned}
$$

The intuition for these definitions is the claim made in the paper that "[i]f data sets T and S are indistinguishable, then $\mathcal{AA}_{TS}$ should be 0.5"[54], and the facts: $\mathcal{AA}_{TS}$ takes values in $[0, 1]$; if the supports for P_T and P_S are disjoint then $\mathbb{E}\left[\mathcal{AA}_{TS}\right] = 1$ (complete resemblance loss); and if S_T and S_S are identical then $\mathcal{AA}_{TS} = 0$ (no resemblance loss, i.e. overfitting). Informally, the privacy loss then measures how much more resemblance is lost from the test data than from the real data. "If the generator overfits the training data, the Train $\mathcal{AA}$ will be near 0 (good training resemblance), but the Test $\mathcal{AA}$ will be around 0.5 (poor test resemblance). Thus the privacy loss will be high (near 0.5)"[54]

Although HealthGAN has been shown to have higher levels of privacy than other under

this metrics, a common critique has been that it is not differentially private. Xie et al.(2018) created a differentially private WGAN, called DPGAN, but its demonstrated output quality is less than HealthGAN. This is to be expected, since it uses differentially private **stochastic gradient descent** (SGD), which adds noise to the gradient of each batch. Abadi et al.(2016), introduced a technique for "deep learning with differential privacy" in which they "clip the ℓ_2 norm of each gradient, compute the average, add noise in order to protect privacy, and take a step in the opposite direction of this average noisy gradient."[1] However, rather than use the standard composition theorems to determine the total privacy loss, in terms of ϵ and δ, they "invent a stronger accounting method, which [they] call the **moments account**," which "saves a $\sqrt{\log(1/\delta)}$ factor in the ϵ part and a Tq factor in the δ part,"[1] where T is the number of training iterations, and q is the batch size divided by the number of training samples.

Xie et al.(2018) use a variant of that approach. They "do not clip the norm of [the gradient]; instead [they] show that by only clipping on [the weights, they can] automatically guarantee a bound of the norm of [the gradients]."[50] Recall that the WGAN algorithm already used weight clipping to enforce the Lipschitz constraint on the critic. Xie et al.(2018) use that same clipping to derive a bound on the norm of the gradient, rather than clip the gradient, in order to determine the amount of noise necessary to add for differential privacy.

As stated, differential privacy is a theoretical guarantee, but the value of ε achieved might not confer enough practical protection. Another way to evaluate the privacy of a machine learning model is to construct attacks against that privacy. Li et al. (2013) studied notable privacy breaches in order to determine what is meant by privacy, and developed a "Membership Privacy" framework [26] focusing on membership inference attacks (MIA), in which the attacker tries to determine if a given sample was in the training data or not. Yeom et al. (2018) prove attribute inference, another type of attack in which the attacker tries to infer missing attributes given other attributes from a sample in the training data, is "harder" than membership inference, so that the "attribute advantage" they define implies their "membership advantage", but that the converse does not hold [55]. Most of the literature focuses on "black box" attacks [39], in which the attacker can query a model trained on private data, and receive the model output, and must base its attack on that information.

However, it is common in the literature to consider worst case scenarios, in which attackers have large advantages, such as information available for linkage attacks. Nasr et al. considered white-box attacks[30] in which the attacker could compute the gradients of the loss of the model at the points under attack. Sablayrolles et al. (2019) prove under certain assumptions that black-box attacks are optimal and claim that "to the best of [their] knowledge, the literature does not report white-box attacks outperforming the stateof-the-art black-box attacks."[38]

2.5 Summary of Literature Review

Having reviewed the relevant literature, we see that privacy preservation is not straightforward, but that differential privacy does offer a rigorous guarantee for a certain notion of privacy. Unfortunately, differential privacy cannot be achieved without adding noise, and that noise tends to degrade the output quality. In order to mitigate that degradation, it is imperative to design a mechanism which takes advantage of a strong bound on the sensitivity of the query function. Also, by exploring different noise distributions and using different composition theorems, the amount of noise necessary can be decreased. Finally, differential privacy may not be achievable in all settings with sufficient utility, or the theoretical bound may not provide adequate privacy protection. Empirical estimates of the level of privacy afforded can be computed using privacy metrics or by conducting privacy attacks, such as membership inference attacks.

CHAPTER 3
A Differentially Private WGAN-GP

The first problem addressed was to make a differentially private WGAN-GP. In order to accomplish this, we derived a bound on the norm of the gradient of the gradient penalty term of the WGAN-GP loss function, for fully connected deep neural networks which use ReLU or LeakyReLU activation functions (a common choice in the literature). In addition to deriving a bound on this term, we also improved upon a previously derived bound for the other terms of the loss function. In fact, for those terms, we derive a bound which is tight. These bounds are computed each training iteration, obviating the need to clip the weights, which is not part of the WGAN-GP algorithm. Rather than being only on the norm of the gradient, these bounds are on each component of the gradient, corresponding to each weight in the critic, which we show permits the use of non-spherical Gaussian noise for differential privacy. We implemented these bounds into an algorithm for a differentially private WGAN-GP, and evaluated the performance of that algorithm on three datasets used in different published medical studies.

The rest of this chapter is organized as follows. First we derive our bounds on the gradients of WGAN and WGAN-GP. Then we describe our algorithm which uses those bounds for a differentially private WGAN-GP. Finally, we analyze the results of using DPWGAN-GP on some real datasets by examining the utility, resemblance, and privacy of the synthetic data for different levels of privacy. [1] We conclude with discussion and directions for future research.

The specific contributions of this chapter include:

- Improved bound on gradient of WGAN over published results
- Bound on the gradient penalty term of a WGAN-GP
- An algorithm which uses the bounds for a differentially private WGAN-GP
- Code for above at https://github.rpi.edu/RensselaerIDEA/synthetic-data

[1] to appear in Springer Nature - Research Book Series: Transactions on Computational Science & Computational Intelligence.

– Empirical analysis of results from using the DPWGAN-GP on real datasets that demonstrates that differential privacy can help reduce convergence to poor local minimums and improve performance while maintaining privacy.

3.1 Bounds on Gradient of Wasserstein GANs

We derive a stronger bound (tight in fact) for the gradient of a WGAN with all layers fully connected and only ReLU or LeakyReLU activation functions. Xie et al.(2018) created a DP WGAN, called DPGAN, and showed the norm of the gradients could be bound by only clipping the weights.[51]:

$$\begin{aligned}\left\|\nabla_W \left[D_W\left(\mathbf{x}^{(i)}\right) - D_W\left(G_\Theta(\mathbf{z}^{(i)})\right)\right]\right\| &\leq 2\left\|\nabla_W \left[D_W\left(\mathbf{x}^{(i)}\right)\right]\right\| \\ &= 2\sum_l \sum_{ij} \left|\frac{\partial C}{\partial W_{ij}^{(l)}}\right| \leq 2C_p B_\sigma B_{\sigma'}^2 \sum_{l=1}^{L-1} m_l m_{l+1} = C_D \end{aligned} \tag{3.1}$$

where C_p is a bound used to clip the weights of the critic, B_σ is a bound on the output of the activation function, and $B_{\sigma'}$ is a bound on the derivative of the activation function. They note that, although activation functions like ReLU or Leaky ReLU are not bounded functions, the image of a compact set under any continuous function is compact, so as long as the input is from a compact set (true for HealthGAN) the bound will exist.

In this section, we first explain how this bound can be improved. Next we derive a bound which is tight. Last, we generalize this bound to the WGAN-GP by deriving a bound on the norm of gradient penalty term.

3.1.1 Improved the Bound for WGAN

First, we note that by the linearity of differentiation, the gradient can be decomposed into two terms:

$$\nabla_W \left[D_W\left(\mathbf{x}^{(i)}\right)\right] - \nabla_W \left[D_W\left(G_\Theta(\mathbf{z}^{(i)})\right)\right] \tag{3.2}$$

Only the first term is a query on the data; the second term is computed from random noise by the generator. Xie et al.(2018) added noise calibrated to a bound on the norm of the gradient of the entire loss function, which is not necessary, because that is not the ℓ_2 sensitivity of the query on the training data. The added noise only needs to be calibrated to the maximum possible norm of the first term, cutting the bound in half.

Next, since Gaussian noise is being added, it should be calibrated to the ℓ_2-norm, not the ℓ_1-norm. For a vector $\mathbf{v} \in \mathbb{R}^d$, given that each component is bounded by $|v_i| \leq B$, the ℓ_1-norm is bounded by Bd, but the ℓ_2-norm is bounded by $B\sqrt{d}$, so this cuts the bound by a factor of $\sqrt{d}$ (≈ 2000 for HealthGAN), greatly reducing the amount of noise necessary.

Finally, rather than use a bound derived from the maximum that the absolute value of the weights *can* be, based on a constant bound on the weights C_p, we propose using a bound derived from the actual weights of the critic during each iteration, t_c. Since the weights are initialized to small values, early on in training when the weights are much less than that maximum forced by clipping, this yields a much tighter bound. The weights of the network, along with bounds on the input to the network and on the gradient of the output, will be used in forward and back propagation to obtain bounds for each component k of the gradient, of the form:

$$C_{0,k}^{\nabla_{W^{(t_c)}}} \leq [\nabla_{W^{(t_c)}} D_{W^{(t_c)}}(\mathbf{x})]_k \leq C_{1,k}^{\nabla_{W^{(t_c)}}} \tag{3.3}$$

The bounds $C_0^{\nabla_{W^{(t_c)}}}$ and $C_1^{\nabla_{W^{(t_c)}}}$ can not be given by a single closed form expression, but instead are computed recursively, similar to the gradients, using the same forward and back propagation equations. Our notation for the bounds is that, for some quantity z, C_0^z is the lower bound and C_1^z is the upper bound, with the superscript indicating what quantity is being bound. The bounds can be used to find a tighter bound on the ℓ_2-norm of the gradient. In fact, for critics that use only fully connected layers with ReLU or LeakyReLU activation functions (very common activation functions in the literature), the bounds we derive are tight, in the sense that when the component-wise upper and lower bounds of the input training data are the same, then the computed component-wise bounds on the gradients are the same (i.e. the value of the gradient for that value of the input).

3.1.2 Derivation of Tight Bound on Gradient of WGAN

Theorem 3.1.1. *At each iteration of training, given the current weights and biases of a fully connected critic with ReLU or LeakyReLU activation functions, each component of the gradient of the critic can be bounded by bounds of the form:*

$$C_{0,k}^{\nabla_z} \leq [\nabla_z D_{W^{(t_c)}}(\mathbf{x})]_k \leq C_{1,k}^{\nabla_z} \tag{3.4}$$

The bounds are computed using forward and backward propagation equations similar to those used to compute the gradients.

Proof. Our critic will be denoted by $D(x)$, and will be a neural network with L hidden layers, all fully connected. The activation functions will all be ReLU or LeakyReLU functions, and the output will be a scalar without any activation function. The ReLU, LeakyReLU, and identity function can be expressed as a single family, parameterized by a single value $\alpha \in [0, 1]$:

$$\phi_\alpha(x) = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{3.5}$$

The entire network is described by these well known neural network equations, which will serve to show our notation. Superscripts denote layers, and subscripts denote components of the layer. The input layer is x^0, and the output layer is $D(x) = D(x^0) = x^{L+1}$.

$$s_j^\ell = \left(\sum_i W_{ij}^\ell x_i^{\ell-1} \right) + b_j^\ell \tag{3.6}$$

$$x_j^\ell = \phi_{\alpha^\ell} \left(s_j^\ell \right) \tag{3.7}$$

As noted previously, for continuous activation functions, a bound on the input layer propagates forward to a bound on the output of every layer. For our models, the input is bounded by $0 \leq x_i^0 \leq 1$. For nondecreasing activation functions, the forward propagation is given by:

$$C_{0,i}^{x^{\ell-1}} \leq x_i^{\ell-1} \leq C_{1,i}^{x^{\ell-1}} \tag{3.8}$$

$$C_{0,j}^{s^\ell} = \sum_{i:W_{ij}^\ell \geq 0} W_{ij}^\ell C_{0,i}^{x^{\ell-1}} + \sum_{j:W_{ij}^\ell < 0} W_{ij}^\ell C_{1,i}^{x^{\ell-1}} \tag{3.9}$$

$$C_{1,j}^{s^\ell} = \sum_{i:W_{ij}^\ell \geq 0} W_{ij}^\ell C_{1,i}^{x^{\ell-1}} + \sum_{j:W_{ij}^\ell < 0} W_{ij}^\ell C_{0,i}^{x^{\ell-1}} \tag{3.10}$$

$$C_{0,j}^{x^\ell} = \phi_{\alpha^\ell} \left(C_{0,j}^{s^\ell} \right) \leq x_j^\ell \leq \phi_{\alpha^\ell} \left(C_{1,j}^{s^\ell} \right) = C_{1,j}^{x^\ell} \tag{3.11}$$

These forward propagation bounds are clearly tight in that $C_0^{x^{\ell-1}} = C_1^{x^{\ell-1}}$ implies that the left hand side equals the right hand side in (3.11).

We let the scalar loss function (error function) for the critic of the WGAN for a single

training example be denoted by $e(x, z)$. The gradient penalty term of the WGAN-GP will be treated separately shortly. The gradients from back propagation are given by:

$$\nabla_{b^\ell} e = \phi'_{\alpha^\ell}(s^\ell) \odot \nabla_{x^\ell} e \tag{3.12}$$

$$\nabla_{x^{\ell-1}} e = W^\ell \nabla_{b^\ell} e \tag{3.13}$$

$$\nabla_{W^\ell} e = (\nabla_{x^{\ell-1}} e)\,(\nabla_{b^\ell} e)^T \tag{3.14}$$

Here, $\nabla_z e$ has the dimensions of z, with $[\nabla_z e]_i = \partial e/\partial z_i$, and $\odot$ represents the Hadamard product. Although ϕ_α is technically not differentiable at zero, we adopt the convention used in Google's TensorFlow, that $\phi'_\alpha(0) = \alpha$.

To use these back propagation equations to compute bounds on the gradients, rather than compute the gradients themselves, we need to use the following facts about Hadamard products and outer products of bounded vectors. Let vectors y and z be given such that $a_i \le y_i \le b_i$ and $c_j \le z_j \le d_j$.

$$\text{Let } M_{ij} = \max\{a_i c_j, a_i d_j, b_i c_j, b_i d_j\}, \text{ and} \tag{3.15}$$

$$m_{ij} = \min\{a_i c_j, a_i d_j, b_i c_j, b_i d_j\}, \text{ then} \tag{3.16}$$

$$m_{ii} \le [y \odot z]_i \le M_{ii}, \text{ if } y \odot z \text{ is defined} \tag{3.17}$$

$$m_{ij} \le \left[yz^T\right]_{ij} \le M_{ij} \tag{3.18}$$

Again, these are tight in that, $a_i = b_i$ and $c_j = d_j$ implies $m_{ij} = M_{ij}$. To use these equations to back propagate the bounds, we first use forward propagation to compute the following bounds:

$$C_{0,j}^{\phi'_{\alpha^\ell}} = \phi'_{\alpha^\ell}\left(C_{0,j}^{s^\ell}\right) \le \phi'_{\alpha^\ell}(s^\ell) \le \phi'_{\alpha^\ell}\left(C_{1,j}^{s^\ell}\right) = C_{1,j}^{\phi'_{\alpha^\ell}}$$

Then, we need a bound on the gradient of the final layer, which we use to propagate backwards. Although $\nabla_{x^{L+1}} e = -1$ for the WGAN (without gradient penalty term), during actual computations we find that numerical errors require adding some tolerance, such as $\nabla_{x^{L+1}} \in (-0.999999, -1.000001)$. Using these bounds on the gradient of the output layer, our componentwise bounds on the gradients for all other layers are given by:

$$C_{0,j}^{\nabla_{b^\ell}} = \min\{C_{0,j}^{\nabla_{x^\ell}} C_{0,j}^{\phi'_{\alpha^\ell}}, C_{0,j}^{\nabla_{x^\ell}} C_{1,j}^{\phi'_{\alpha^\ell}}, C_{1,j}^{\nabla_{x^\ell}} C_{0,j}^{\phi'_{\alpha^\ell}}, C_{1,j}^{\nabla_{x^\ell}} C_{1,j}^{\phi'_{\alpha^\ell}}\} \tag{3.19}$$

$$C_{1,j}^{\nabla_{b^\ell}} = \max\{C_{0,j}^{\nabla_{x^\ell}} C_{0,j}^{\phi'_{\alpha^\ell}}, C_{0,j}^{\nabla_{x^\ell}} C_{1,j}^{\phi'_{\alpha^\ell}}, C_{1,j}^{\nabla_{x^\ell}} C_{0,j}^{\phi'_{\alpha^\ell}}, C_{1,j}^{\nabla_{x^\ell}} C_{1,j}^{\phi'_{\alpha^\ell}}\} \tag{3.20}$$

$$C_{0,j}^{\nabla_{x^{\ell-1}}} = \sum_{i:W_{ij}^\ell \geq 0} W_{ij}^\ell C_{0,j}^{\nabla_{b^\ell}} + \sum_{j:W_{ij}^\ell < 0} W_{ij}^\ell C_{1,j}^{\nabla_{b^\ell}} \tag{3.21}$$

$$C_{1,j}^{\nabla_{x^{\ell-1}}} = \sum_{i:W_{ij}^\ell \geq 0} W_{ij}^\ell C_{1,j}^{\nabla_{b^\ell}} + \sum_{j:W_{ij}^\ell < 0} W_{ij}^\ell C_{0,j}^{\nabla_{b^\ell}} \tag{3.22}$$

$$C_{0,j}^{\nabla_{W_{ij}^\ell}} = \min\{C_{0,j}^{\nabla_{x^{\ell-1}}} C_{0,j}^{\nabla_{b^\ell}}, C_{0,j}^{\nabla_{x^{\ell-1}}} C_{1,j}^{\nabla_{b^\ell}}, C_{1,j}^{\nabla_{x^{\ell-1}}} C_{0,j}^{\nabla_{b^\ell}}, C_{1,j}^{\nabla_{x^{\ell-1}}} C_{1,j}^{\nabla_{b^\ell}}\} \tag{3.23}$$

$$C_{1,j}^{\nabla_{W_{ij}^\ell}} = \max\{C_{0,j}^{\nabla_{x^{\ell-1}}} C_{0,j}^{\nabla_{b^\ell}}, C_{0,j}^{\nabla_{x^{\ell-1}}} C_{1,j}^{\nabla_{b^\ell}}, C_{1,j}^{\nabla_{x^{\ell-1}}} C_{0,j}^{\nabla_{b^\ell}}, C_{1,j}^{\nabla_{x^{\ell-1}}} C_{1,j}^{\nabla_{b^\ell}}\} \tag{3.24}$$

□

Recall that Xie et. al used weight clipping to bound the gradient, which was already used in the WGAN algorithm to enforce the Lipschitz constraint. Instead of weight clipping, the WGAN-GP algorithm uses a gradient penalty term on the loss function to enforce the Lipschitz constraint. Using our bound for WGAN-GP would prevent the need to perform weight clipping.

3.1.3 Non-Spherical Gaussian Noise

Since our bound is component-wise, rather than only in norm, it also permits the use of non-spherical noise in which components components with less sensitivity would have less noise added, rather than having Gaussian noise with constant variance added to every component. A toy example in only two dimensions shows how this might yield better results. Consider a bound on the gradient given by:

$$98 \leq [\nabla_{W^{(t_c)}} D_{W^{(t_c)}}(\mathbf{x})]_1 \leq 100 \qquad -50 \leq [\nabla_{W^{(t_c)}} D_{W^{(t_c)}}(\mathbf{x})]_2 \leq 50$$

In this situation, if a batch of these gradients is summed, the ℓ_2-sensitivity of the sum would be $\sqrt{2^2 + 100^2} \approx 100.02$, which is the maximum possible norm of the difference between any two gradients. For typical values such as $\varepsilon = 0.1$ and $\delta = 10^{-5}$, this would require Gaussian noise with $\sigma \approx 4850$. If a batch of 30 gradients was summed, with a true sum equal to $\langle 3000, 500 \rangle$, then adding that much noise (σ greater than the magnitude of the components)

it is likely that the result could be very far from the correct direction, even in the exact opposite direction. Another possible technique would be to center and scale the gradients, based on the bounds, before adding spherical Gaussian noise, then rescaling the result. For example, if we query "the gradient minus $\langle 99, 0\rangle$, then divided component-wise by $\langle 1, 50\rangle$", that query would have a sensitivity of $\sqrt{2^2+2^2} = 2\sqrt{2}$, requiring Gaussian noise with $\sigma \approx 137$ for the same level of privacy. After adding that noise to that query, the immunity to post processing proposition says that we can scale back without losing any additional privacy, which would scale the standard deviation of the noise to 137 in the first component, and 6850 in the second component. This would make the noisy response have a first component that is likely very close to the true response of 3000, which makes sense since we already know that the true value must be between 2940 and 3000. We have not implemented it yet in DPWGAN-GP, but plan on doing so in the near future. The following proof shows that it is differentially private.

Theorem 3.1.2. *(Non-Spherical Gaussian Mechanism) Given any funtion $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ whose image is bounded by $\forall i = 1, \ldots, k, a_i \leq f(D)_i \leq b_i$, the following query mechanism $q(D)$ is (ε, δ)-differentially private, and the trace of the covariance matrix is the same as that for the Gaussian mechanism.*

$$q(D) = f(D) + \mathcal{N}_k(0, \Sigma), \text{ where}$$
$$\Sigma_{ij} = \begin{cases} k(b_i - a_i)^2 2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2 & i = j \\ 0 & i \neq j \end{cases}$$

Proof. Define $r_i = \frac{b_i - a_i}{2}$, $\mu_i = \frac{a_i + b_i}{2}$, and $\widetilde{f}(D) = \widetilde{\mathbf{Y}}$, where $\widetilde{Y}_i = (f(D)_i - \mu_i)/r_i$.
Then $\forall i = 1, \ldots, k, -1 \leq \widetilde{Y}_i \leq 1$, so the ℓ_2-sensitivity of $\widetilde{f}$ is $\sqrt{\sum_{i=1}^{k}(2)^2} = 2\sqrt{k}$.
Using the Gaussian mechanism, this means that $\widetilde{q}(D) = \widetilde{f}(D) + \mathcal{N}_k\left(0, 4k\left[2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2\right]\mathbf{I}_k\right)$ is (ε, δ)-differentially private.

By the immunity to post-processing proposition, if we define $q_2(D)$ such that: $q_2(D)_i = r_i\widetilde{q}(D)_i + \mu_i$, then $q_2(D)$ is also (ε, δ)-differentially private.

That equation makes $q_2(D)_i$ a random variable given by:
$q_2(D)_i = f(D)_i + \mathcal{N}(0, k(b_i - a_i)^2 2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2)$, proving that $q(D)$ is equivalent to $q_2(D)$.

The ℓ_2-sensitivity of f is $\sqrt{\sum_{i=1}^k (b_i - a_i)^2}$, so the Gaussian mechanism on f would be given by: $f(D) + \mathcal{N}_k\left(0, \sum_{i=1}^k (b_i - a_i)^2 \left[2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2\right] \mathbf{I}_k\right)$.

The trace of that covariance matrix is given by:

$$\begin{aligned}\operatorname{tr}\left(\sum_{i=1}^k (b_i - a_i)^2 \left[2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2\right] \mathbf{I}_k\right) &= \sum_{j=1}^k \sum_{i=1}^k (b_i - a_i)^2 \left[2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2\right] \\ &= k\sum_{i=1}^k (b_i - a_i)^2 \left[2\log\left(\frac{5}{4\delta}\right)/\varepsilon^2\right]\end{aligned}$$

□

3.1.4 Derivation of Bound on Norm of Gradient of WGAN-GP

We now derive a bound on the norm of the gradient of the gradient penalty term of the WGAN-GP loss function (2.10).

Theorem 3.1.3. *The norm of the gradient of the gradient penalty is bounded by*

$$\left\|\nabla_{W^\ell}\left(\|\nabla_x D(x)\|_2 - 1\right)^2\right\|_F \leq 2\max\{\left(\|\nabla_x D(x)\|_2 - 1\right), 1\} \prod_{k\neq\ell} \|W^k\|_F \prod_{1\leq k\leq L} \|\phi'_{\alpha^k}\|_F$$

Proof. Using the chain rule, with a dummy variable z, and ignoring the non-differentiability at the origin, we have:

$$\nabla_z \left(\|\nabla_x D(x)\|_2 - 1\right)^2 = 2\left[\left(\|\nabla_x D(x)\|_2 - 1\right)\right] \nabla_z \|\nabla_x D(x)\|_2 \tag{3.25}$$

$$\nabla_z \|\nabla_x D(x)\|_2 = \nabla_z \left(\nabla_x D(x)\right) \nabla_x D(x) / \|\nabla_x D(x)\|_2 \tag{3.26}$$

$$\left\|\nabla_z \left(\|\nabla_x D(x)\|_2\right)\right\|_F \leq \left\|\nabla_z \left(\nabla_x D(x)\right)\right\|_F \tag{3.27}$$

The gradient term $\nabla_x D(x)$ is computed using standard back propagation:

$$\frac{\partial D(x)}{\partial x_i^{\ell-1}} = \sum_j \left(\frac{\partial x_j^\ell}{\partial x_i^{\ell-1}}\right)\left(\frac{\partial D(x)}{\partial x_j^\ell}\right) = \sum_j \phi'_{\alpha^\ell}\left(s_j^\ell\right) W_{ij}^\ell \left(\frac{\partial D(x)}{\partial x_j^\ell}\right) \tag{3.28}$$

$$\nabla_{x^{\ell-1}} D(x) = \left(W^\ell\right)\left(\phi'_{\alpha^\ell}\right)\left(\nabla_{x^\ell} D(x)\right) \tag{3.29}$$

$$\nabla_{x^0} D(x) = \left(W^1\right)\left(\phi'_{\alpha^1}\right)\ldots\left(W^L\right)\left(\phi'_{\alpha^L}\right)\left(W^{L+1}\right)\left(\phi'_{\alpha^{L+1}}\right) \tag{3.30}$$

$$= \prod_{\ell=1}^{L+1}\left(W^\ell\right)\left(\phi'_{\alpha^\ell}\right) \tag{3.31}$$

In these equations, rather than using Hadamard products, we equivalently used diagonal matrices ϕ'_{α^ℓ}, and in particular, since the last layer has no activation function, $\phi'_{\alpha^{L+1}} = 1$, and may be dropped when convenient. Although ϕ'_α is technically not differentiable at zero, we adopt the convention used in the Google TensorFlow software that $\phi''_\alpha(x) = 0$ (constant). The fact that the second derivative of this activation is zero almost everywhere (by convention, everywhere) makes the gradient of the gradient penalty term straightforward to compute. For example, it makes the gradient of the gradient penalty term with respect to any of the bias vectors equal to zero. For the gradient with respect to the weight matrices, we have the following formula:

$$\nabla_{W^\ell}\left(\nabla_x D(x)\right) = \left(\prod_{1\le k<\ell}\left(W^k\right)\left(\phi'_{\alpha^k}\right)\right)^T \nabla_x D(x) \left(\prod_{\ell<k\le L+1}\left(\phi'_{\alpha^{k-1}}\right)\left(W^k\right)\right)^T \tag{3.32}$$

In the above formula, empty products would be identity transformations (of the appropriate dimensions). Since we are interested in bounding the ℓ_2-norm of the gradient of the gradient term, which when we think of the weights as matrices is the Froebenius norm, the formula above yields the bound:

$$\left\|\nabla_{W^\ell}\left(\nabla_x D(x)\right)\right\|_F \le \prod_{k\ne\ell}\|W^k\|_F \prod_{1\le k\le L}\|\phi'_{\alpha^k}\|_F \tag{3.33}$$

Combining this bound with previous equations yields the stated bound on the norm of the gradient of the gradient penalty term (without the factor of λ) with respect to weight

matrix W^ℓ:

$$\|\nabla_{W^\ell}\left(\|\nabla_x D(x)\|_2 - 1\right)^2\|_F \leq 2\max\{\left(\|\nabla_x D(x)\|_2 - 1\right), 1\}\prod_{k\neq\ell}\|W^k\|_F \prod_{1\leq k\leq L}\|\phi'_{\alpha^k}\|_F \quad (3.34)$$

□

3.2 DPWGAN-GP Algorithm

We have used differentially private stochastic gradient descent with our algorithm, Algorithm 1 on page 36, a modification of the previous HealthGAN by Yale et al. (2019)[54], made to be differentially private using the bounds derived.

Algorithm 1 is a base algorithm that makes use of our gradient bounds, but could be improved in many ways. First, the noise level is kept constant across training iterations, instead of adding more noise toward the beginning of training and less toward the end, to improve the convergence. Lee and Kifer (2018) developed a differentially private algorithm for which the privacy budget and step size is even chosen dynamically each iteration, although it works with a more recent variation of differential privacy, referred to as zero-concentrated (zCDP).[25]

Algorithm 1 also divides the privacy budget equally across each layer, and each of the weights of the layers, using spherical Gaussian noise. Since our bounds for the WGAN loss function are on each component, it would be possible to add different amounts of noise to each component, potentially improving performance.

Finally, the algorithm above assumes that λ is constant. Gulrajani et al. (2017) found $\lambda = 10$ to work well for all the datasets and architectures they experimented with[20], but they did not add noise to the gradient. Since our bound on the gradient of the gradient penalty term is not as tight as our bound on the gradient of the rest of the loss function, a smaller λ may be necessary, or it may be best to choose λ each iteration so that that term of the gradient is smaller in norm.

In future work, we plan to incorporate our stronger bounds with advanced composition algorithms, like the moments accountant, with other methods of enforcing the Lipschitz constraint, such as spectral normalization, and with more recent variations of differential privacy. These strategies should yield improved (lower) privacy loss over other published results.

Algorithm 1 Differentially Private WGAN-GP

Ensure: The training samples are vectors in $[0,1]^d \subset \mathbb{R}^d$. The neural network is fully connected with ReLU or LeakyReLU activation functions. The first L_{g} layers are called the generator G_Θ, with weights and biases Θ, and the next L_{c} layers are called the critic D_W, with weights and biases W. The output is a scalar.

Require: n_{g}: number of training iterations for the generator;
n_{c}: number of training iterations for the critic per training iteration for the generator;
N: the number of training samples;
m: the batch size;
$\mathbb{P}_z$: the distribution of the generator input;
λ: the gradient penalty coefficient;
ϵ, δ: the differential privacy parameters;
α, β_1, β_2: the hyper-parameters for Adam.

Initialize: $\Theta^{(1)}$ and $W^{(1)}$

$\sigma_{\mathrm{dp}} \leftarrow \dfrac{\sqrt{2\log(1.25/\delta)}}{\epsilon}$

for $t_g = 1, \ldots, n_{\mathrm{g}}$ **do**

  for $t_c = 1, \ldots, n_{\mathrm{c}}$ **do**

    Compute $C_0^{\nabla_{W^{(t_{\mathrm{c}})}}}, C_1^{\nabla_{W^{(t_{\mathrm{c}})}}} : C_0^{\nabla_{W^{(t_{\mathrm{c}})}}} \leq \nabla_{W^{(t_{\mathrm{c}})}} D_{W^{(t_{\mathrm{c}})}}(\mathbf{x}) \leq C_1^{\nabla_{W^{(t_{\mathrm{c}})}}}$

    Compute $C^\lambda = \max_{\mathbf{x}\in[0,1]^d} \left\| \nabla_{W^{(t_{\mathrm{c}})}} \left(\|\nabla_{\widehat{\mathbf{x}}} D_W(\widehat{\mathbf{x}})\|_2 - 1 \right)^2 \right\|_2$

    Compute total ℓ_2 sensitivity $\ell_2^{(t_{\mathrm{c}})} = \sqrt{\left\| C_1^{\nabla_{W^{(t_{\mathrm{c}})}}} - C_0^{\nabla_{W^{(t_{\mathrm{c}})}}} \right\|_2^2 + \lambda^2 \left\| C^\lambda \right\|_2^2}$

    for $i = 1, \ldots, m$ **do**

      Sample real data $\mathbf{x}\sim\mathbb{P}_r$, latent variable $\mathbf{z}\sim\mathbb{P}_z$, random number $u\sim U[0,1]$.

      $\widetilde{\mathbf{x}} \leftarrow G_\Theta(\mathbf{z})$

      $\widehat{\mathbf{x}} \leftarrow u\mathbf{x} + (1-u)\widetilde{\mathbf{x}}$

      $g_W^{(i)} \leftarrow \nabla_{W^{(t_{\mathrm{c}})}} \left[D_W(\widetilde{\mathbf{x}}) - D_W(\mathbf{x}) + \lambda \left(\|\nabla_{\widehat{\mathbf{x}}} D_W(\widehat{\mathbf{x}})\|_2 - 1 \right)^2 \right]$

    end for

    Sample $n\sim N(0, 4\sigma_{\mathrm{dp}}^2 \mathbf{I}_W)$

    $\hat{g}_{W^{(t_{\mathrm{c}})}} \leftarrow \frac{1}{m} \left(\sum_{i=1}^m g_w^{(i)} \right) + \frac{1}{m} n$

    $W^{(t_{\mathrm{c}}+1)} \leftarrow \mathrm{Adam}\left(\hat{g}_{W^{(t_{\mathrm{c}})}}, W^{(t_{\mathrm{c}})}, \alpha, \beta_1, \beta_2 \right)$

  end for

  Sample a batch of latent variables $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim \mathbb{P}_z$.

  $\Theta^{(t_{\mathrm{g}}+1)} \leftarrow \mathrm{Adam}\left(\frac{1}{m} \sum_{i=1}^m \nabla_{\Theta^{(t_{\mathrm{g}})}} \left[-D_{W^{(t_{\mathrm{c}}+1)}} \left(G_{\Theta^{(t_{\mathrm{g}})}}(\mathbf{z}^{(i)}) \right) \right], \Theta^{(t_{\mathrm{g}})}, \alpha, \beta_1, \beta_2 \right)$

end for

Return: W and Θ, and $\ell_2^{(t_{\mathrm{c}})}$ to compute total privacy cost $(\varepsilon_T, \delta_T)$.

3.3 WGAN-GP Architecture

The WGAN-GP architecture consisted of a generator with two fully connected hidden layers, and fully connected output, that fed into a critic with three fully connected hidden layers and fully connected output. For all experiments, the critics were identical, with the following node counts by layer: $d, 64, 128, 256, 1$. Here d is the dimension of the data space, which did vary for the datasets. All of the hidden layers used a LeakyReLU activation function with the default $\alpha = 0.2$. The output had no activation function.

For all experiments, the generator had 100 nodes in the input, which were fed uniform random variables $\sim U(0, 1)$. The heuristic used to pick the number of nodes in the other layers was the same across all experiments: $2d, \lfloor 1.5d \rfloor, d$, where again d is the dimension of the data space. The hidden layers all used the ReLU activation function, and the output used the sigmoid activation function.

The batch size used was $\lfloor N/10 \rfloor$, where N was the number of points in the training set. The generator was trained for 200,000 iterations, and the critic was trained for 5 iterations between each generator training iteration.

Before training on the training data, the data is preprocessed so that all values are in $[0, 1]$. For categorical variables, this normalization is accomplished using the formulation of the synthetic data vault[32]. After the trained model is used to create synthetic data, that synthetic data is converted to the unnormalized values using the inverse of the transformations used to normalize the training data.

Table 3.1: Characteristics of the architecture by dataset

Dataset	N	Batch Size	Dimension	Generator Nodes/Layer
1	2085	208	12	100, 24, 18, 12
2	2119	211	34	100, 68, 51, 34
3	1719	171	7	100, 14, 10, 7

3.4 Analysis on Health Data

We compared the performance of the original HealthGAN (HGAN) to the differentially private WGAN-GP on three different datasets from the MIMIC-III ('Medical Information Mart for Intensive Care') database, consisting of data from patients admitted to critical care units at a large tertiary care hospital. [21]. These datasets were designed to reproduce pub-

lished studies using MIMIC data. Each dataset has an associated task: problem 1 predicts mortality of diabetics [2], problem 2 predicts mortality of acute kidney injury patients [16], and problem 3 predicts readmission to ICU within 30 days [7]. The characteristics of each dataset are summarized in Table 3.2.

Table 3.2: Characteristics of three MIMIC-III Datasets. + Rate refers to percentage of people who died/were readmitted in the training data.

Dataset	Data Points	Features	Label	+ Rate
1	2085	6 diagnostic scores/values 5 demographic categorical var.	Mortality	6.3%
2	2119	31 diagnostic scores 2 categorical var.	Mortality	23.4%
3	1719	6 diagnostic values	Readmission	3.03%

For our experimental design, each dataset was first split into a 50% training set and 50% testing set. The synthetic data model was trained using the training data. Since our goal was to assess the effect of differential privacy, the same algorithm was used for training all three datasets except the hyperparameter with the level of noise is varied: zero noise for HealthGAN, and four increasing levels of noise σ_{dp} (10^{-7},10^{-6},10^{-5}, 10^{-4}). The models were trained using Google's TensorFlow. For each model we generated two synthetic sets of equal size to the training data. Full details of the architecture and parameters used can be found in appendix 3.3

3.4.1 DP and Mode Collapse Case Study

The hierarchical optimization problem solved in WGANs is highly non-convex and algorithms frequently produce poor local minima. We observed that Algorithm 1 produced solutions in which the rate of positive class (+ rate) is much lower in the synthetic data than the training data, which is commonly known as "mode collapse". To combat convergence to these bad local minima, we trained a fixed number of models for each noise level, and chose the one with the highest positivity rate (least mode collapse). For datasets 1 and 2, we trained two models for each noise level, and created two synthetic datasets for each model. For dataset 3, which had the rarest class and suffered the most from convergence to local minima, we trained 10 models for each noise level, and created two synthetic datasets for each model.

Table 3.3 shows the + rate for the best synthetic dataset for each noise level, compared to the + rate in the training and test sets, for each of the datasets.

Table 3.3: Comparison of highest positivity rates (%) of the synthetic data from random restarts of Alg 1 for different noise levels, to the positivity rates in the training and test sets.

	Dataset 1	Dataset 2	Dataset 3
training set	6.28	23.4	3.03
test set	6.04	25.9	2.97
$\sigma_{\text{dp}} = 0$ (HGAN)	4.46	17.8	2.60
$\sigma_{\text{dp}} = 10^{-7}$	**5.37**	19.1	2.39
$\sigma_{\text{dp}} = 10^{-6}$	5.23	**20.1**	2.68
$\sigma_{\text{dp}} = 10^{-5}$	4.22	19.6	**3.20**
$\sigma_{\text{dp}} = 10^{-4}$	4.89	18.9	2.97

3.4.2 Computational Results

We assessed resemblance, utility, and privacy. For each paper, we compare HealthGAN (HGAN) to the WGAN model with noise level, selected by choosing the synthetic data with the highest positivity rate (see Resemblance below). The results are summarized in Table 3.4.

Table 3.4: Comparison of results on three datasets between HGAN and DPWGAN-GP. Best results are bolded, but differences may not be significant.

	Dataset 1		**Dataset 2**		**Dataset 3**	
Measure	HGAN	w/ D.P.	HGAN	w/ D.P.	HGAN	w/ D.P.
Utility − AUC	0.723	**0.724**	0.614	**0.657**	0.713	**0.720**
+Rate	4.5%	**5.4%**	17.8%	**20.2%**	2.6%	**3.2%**
TrResemblLoss	0.545	**0.539**	**0.856**	0.860	0.628	**0.627**
TeResemblLoss	**0.534**	0.550	**0.868**	0.872	**0.648**	0.649
PrivacyLoss	**-0.011**	0.012	**0.012**	**0.012**	0.022	**0.020**

Utility The utility of synthetic data is a measure of how well it can be used for a specific purpose. To assess utility, we used the "gold standard" of creating predictive models on the synthetic data and using them to classify the test data, quantified by the area under the curve (AUC) of the classifiers. Although the goal of creating a differentially private WGAN-GP was to have the rigorous privacy guarantee, with the hope that utility would

not be substantially decreased, we found that differential privacy actually slightly improved the utility for all three datasets.

Resemblance Besides being useful for a specific purpose, we want the synthetic data to resemble the real data. We measure resemblance in two ways. Since the class labels are imbalanced in the training data, and GANs tend to suffer from mode collapse, we specifically check for mode collapse by checking that the proportion of the rare class is not under-represented in the synthetic data. For each noise level, we trained a fixed number of models and selected the one with the highest positivity rate. We found that the higher noise levels tended to have the highest maxima for positivity rate, corresponding to the models with the least mode collapse.

We also measured overall resemblance using the nearest-neighbors based metrics defined by Yale et al. (2019)[54], given in the appendix. Intuitively, these measure whether points from the synthetic data and real data are closer to neighbors in the same set or the other set, with 0.5 corresponding to both being equally likely (same distribution). TrResemblLoss measures the resemblance loss of the synthetic data to the training data, and TeResemblLoss measures the resemblance loss to the test data. If the synthetic data was a copy of the real data, the resemblance loss would be 0. Distributions with disjoint supports would have complete resemblance loss, with values of 1. We found that the differentially private WGAN-GP had comparable resemblance to HealthGAN with these metrics.

Privacy The bounds derived in the paper prove that the algorithm is differentially private. We also investigated the empirical privacy using the PrivacyLoss metric introduced by Yale et al.(2019). This metric is the difference resemblance loss on the test set and the resemblance loss on the training set. Intuitively, if there is more resemblance loss on the test set than the training set, that is an indication that the synthetic data is overfitting the training set. We did not find any significant difference between the value of this metric on HealthGAN and DPWGAN-GP. Our interpretation is that, although it has been shown that it is possible for GAN to memorize the training data, it does not seem like HealthGAN is memorizing these datasets.

In addition to privacy, we find differentially private hierarchical optimization has the ability to improve performance. As noted previously, our goal is to achieve an optimal

solution in which the generator is approximating the true distribution of the data. However, since we are learning from a finite sample, we have to try not to overfit the training data. Differential privacy, which makes it unlikely for any one training point to have too large an effect on the optimal value found, intuitively may reduce the likelihood of overfitting, leading to better generalization.

3.5 Conclusion

In this chapter, we derived strong bounds on the gradient of the loss functions of WGAN and WGAN-GP. We show that our bounds do not require weight clipping, allowing them to be used for a WGAN-GP algorithm that provably satisfies differential privacy, and empirically demonstrates high utility, resemblance, and privacy on three health care problems. In addition to a rigorous privacy guarantee, we find that differential privacy actually has the ability to improve performance. For GANs in particular, the noise added for differential privacy seems to act as a form of regularization that mitigates mode collapse and improves the utility of the synthetic data generated.

Although our algorithm is differentially private, the amount of theoretical privacy loss as indicated by ε needs to be too high for good utility, because the sensitivity of the gradient penalty term of the loss function is several orders of magnitude larger than the sensitivity of the other term. In future research, we plan to incorporate our bounds with a more recent technique for enforcing the Lipschitz constraint, spectral normalization. We also plan on evaluating the benefit of using non-spherical Gaussian noise.

Acknowledgments This work was partially supported by the United Health Foundation. I would like to thank my advisor Kristin Bennett, and the contributions of Karan Bhanot, Isabelle Guyon, and John Erickson.

CHAPTER 4
Evaluating Privacy

The second problem addressed was to improve upon existing empirical methods of evaluating privacy preservation, including by considering strong privacy attacks. A focus of the research has been on investigating the known connection between privacy preservation and avoiding over-fitting[55]. The previously mentioned nearest neighbor Adversarial Accuracy of Yale et al.(2019) is meant to empirically measure how much synthetic data overfits or underfits the real data that its generator was trained on. We identify issues with this metric as previously defined, then refine the definition to fix those issues. Next, we define a new framework for attacking privacy, called the "Leave Two Unlabeled" (LTU) methodology of an "LTU Attacker", which is given access to both a model trained on data that needs privacy preserved, and the learning algorithm used to train that model. We prove theoretical results for specific scenarios, showing the connection of privacy loss and overfitting, and the need for randomness to protect privacy.

The rest of this chapter is organized as follows. First we modify the definition of the nearest neighbor Adversarial Accuracy metric and prove that our new definition makes the metric unbiased. Then we describe how we were able to implement it efficiently, and make it suitable for use with discrete distances. Next we present our "Leave Two Unlabeled" (LTU) methodology[2][33], and prove lower bounds for the accuracy of the "LTU Attacker" for different scenarios, discussing how they relate to overfitting and randomness.

The specific contributions of this chapter include:

- Provably unbiased metric for empirically estimating privacy loss
- Efficient implementation which also corrects for discrete distances
- A new framework for attacking privacy which considers the worst possible case in terms of attacker knowledge
- Proofs of lower bounds on the accuracy of the LTU Attacker connected to overfitting or insufficiently random algorithms, demonstrating its usefulness in evaluating privacy

[2]presented at the Third AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI-22) in February 2022

4.1 Nearest Neighbor Adversarial Accuracy

Privacy preservation is meant to prevent too much from being learning from any one record, which would be an extreme over-fitting to that one record. As discussed earlier, Yale et al.(2019) defined the **nearest neighbor Adversarial Accuracy** to measure the resemblance between two distributions, such as a synthetic dataset and the real dataset that it was trained on, with the intent that $\mathcal{AA}_{TS}$ should be 0.5 when S and T were drawn from the same distribution. However, their metric as defined in equation (2.11) on page 23 has a slight bias which makes it only approximately equal to 0.5 when the two datasets are drawn from the same distribution. This is due to the fact that the *nearest neighbor* is determined by *leaving-one-out* when measuring to the same set, but not when measuring to the other set. We refine the definition and prove that the new definition is unbiased.

Definition 4.1.1. *The unbiased nearest neighbor Adversarial Accuracy between two sets S and T, both of size n, is given by:*

$$\begin{aligned}\mathcal{AA}_{TS} = \frac{1}{2}\Bigg[&\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right)\right)\\ &+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right)\right)\Bigg]\end{aligned} \tag{4.1}$$

Theorem 4.1.1. *For i.i.d. random samples S and T, both of the same size n, drawn from the same continuous distribution P on metric space $\mathcal{X}$, with probability density function $p(x)$,* $\mathbb{E}\left[\mathcal{AA}_{TS}\right] = 0.5$

Proof.

$$\begin{aligned}\mathbb{E}\left[\mathcal{AA}_{TS}\right] &= \frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\mathbb{E}\left[\mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right)\right]\right)\\ &+ \frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\mathbb{E}\left[\mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right)\right]\right)\\ &= \frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\Pr\left[d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right]\right)\\ &+ \frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{k=1}^{n}\Pr\left[d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right]\right)\end{aligned}$$

For any independently drawn random samples A and B, and any point $a_i \in A$, $d_{AB}(i)$ is a random variable taking values in $[0, \infty)$ with the conditional CDF $F(d_{AB}(i)|a_i)$ given by:

$$\begin{aligned}\Pr\left[d_{AB}(i) \leq z|a_i\right] &= 1 - \Pr\left[d_{AB}(i) \geq z|a_i\right] \\ &= 1 - \prod_{j=1}^{n} \Pr\left[d(a_i, b_j) \geq z|a_i\right] \\ &= 1 - \left(\int_{\{\mathbf{x} \in \mathcal{X} | d(a_i, x) \geq z\}} p(\mathbf{x}) d\mathbf{x}\right)^n\end{aligned}$$

Using this equation, we see that both $Z_1 = d_{T(S \setminus \{\mathbf{x}_S^k\})}(i)$ and $Z_2 = d_{TT}(i)$ have the same conditional CDF given x_T^i:

$$1 - \left(\int_{\{\mathbf{x} \in \mathcal{X} | d(x_T^i, \mathbf{x}) \geq z\}} p(\mathbf{x}) d\mathbf{x}\right)^{n-1}$$

Furthermore, since $(S \setminus \{\mathbf{x}_S^k\})$ and $(T \setminus \{\mathbf{x}_T^i\})$ are both independent random samples, the random variables Z_1 and Z_2 are conditionally independent, given x_T^i. Therefore, their conditional joint PDF is the product of their conditional marginal PDFs: $f_{Z_1, Z_2}(z_1, z_2|x_T^i) = f_{Z_1}(z_1|x_T^i) f_{Z_2}(z_2|x_T^i)$. By symmetry, we see that:

$$\Pr\left[d_{T(S \setminus \{\mathbf{x}_S^k\})}(i) > d_{TT}(i)|x_T^i\right] = 0.5$$

Since this is true for any x_T^i, we have:

$$\Pr\left[d_{T(S \setminus \{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right] = \int_{\mathcal{X}} (0.5) p(\mathbf{x}) d\mathbf{x} = 0.5$$

Analogous calculations show that: $\Pr\left[d_{S(T \setminus \{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right] = 0.5$, and the theorem follows from straightforward arithmetic. □

To implement this new definition naively appears to requires two levels of summation, in order to "leave one out" of the other set each iteration, increasing the computational cost by a factor of n. However, if for each point we look at its two nearest neighbors in both sets (without leaving any out for the other set), we see that if they are both in one of the sets, then every "leave one out" evaluation will result in the nearest neighbor being in that set.

On the other hand, if each set contains one of the two nearest neighbors, then exactly one of the n "leave one out" iterations will result in the nearest neighbor being in the other set. Stated formally, let the two nearest neighbors to $\mathbf{x}_T^i$ in $(S \cup T) \setminus \{\mathbf{x}_T^i\}$ be $\mathbf{x}^1$ (nearest) and $\mathbf{x}^2$ (2nd nearest), and the two nearest neighbors to $\mathbf{x}_S^i$ in $(S \cup T) \setminus \{\mathbf{x}_S^i\}$ be $\mathbf{x}^a$ (nearest) and $\mathbf{x}^b$ (2nd nearest). Then:

$$g_T(\mathbf{x}_T^i) = \sum_{k=1}^{n} \mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right) = \begin{cases} 0 & \text{if } \mathbf{x}^1 \in S \text{ and } \mathbf{x}^2 \in S \\ 1/n & \text{if } \mathbf{x}^1 \in S \text{ and } \mathbf{x}^2 \in T \setminus \{\mathbf{x}_T^i\} \\ n & \text{if } \mathbf{x}^1 \in T \setminus \{\mathbf{x}_T^i\} \end{cases} \tag{4.2}$$

$$g_S(\mathbf{x}_S^i) = \sum_{k=1}^{n} \mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right) = \begin{cases} 0 & \text{if } \mathbf{x}^a \in T \text{ and } \mathbf{x}^b \in T \\ 1/n & \text{if } \mathbf{x}^a \in T \text{ and } \mathbf{x}^b \in S \setminus \{\mathbf{x}_S^i\} \\ n & \text{if } \mathbf{x}^a \in S \setminus \{\mathbf{x}_S^i\} \end{cases} \tag{4.3}$$

$$\mathcal{AA}_{TS} = \frac{1}{2n^2}\left[\sum_{i=1}^{n} g_T(\mathbf{x}_T^i) + \sum_{i=1}^{n} g_S(\mathbf{x}_S^i)\right] \tag{4.4}$$

We use this to efficiently implement the nearest neighbor Adversarial Accuracy metric with just a correction factor of $1/n$. However, we also correct for the case of discrete distances, such as in this MIMIC III data, which is binary, making the distance effectively the Hamming distance. To handle these discrete distances, equation (4.1) in definition 4.1.1 is actually averaged over two versions: strict inequalities and non-strict inequalities.

$$\begin{aligned} &\mathcal{AA}_{TS} = \\ &\frac{1}{4n^2}\left[\sum_{i=1}^{n}\left(\sum_{k=1}^{n} \mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right)\right) + \sum_{i=1}^{n}\left(\sum_{k=1}^{n} \mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right)\right)\right] + \\ &\frac{1}{4n^2}\left[\sum_{i=1}^{n}\left(\sum_{k=1}^{n} \mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) \geq d_{TT}(i)\right)\right) + \sum_{i=1}^{n}\left(\sum_{k=1}^{n} \mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) \geq d_{SS}(i)\right)\right)\right] \end{aligned} \tag{4.5}$$

This requires modifying g_S and g_T as follows. Let the distances from x_T^i to its two nearest

neighbors in $T \setminus \{x_T^i\}$ be d_T^1 and d_T^2, and to its two nearest neighbors in S be d_S^1 and d_S^2.

$$g_T(\mathbf{x}_T^i) = \frac{1}{2}\left[\sum_{k=1}^{n} \mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) > d_{TT}(i)\right) + \sum_{k=1}^{n} \mathbf{1}\left(d_{T(S\setminus\{\mathbf{x}_S^k\})}(i) \geq d_{TT}(i)\right)\right]$$
$$= \begin{cases} 0 & \text{if } d_S^1 \leq d_S^2 < d_T^1 \\ 1/n & \text{if } d_S^1 < d_T^1 = d_S^2 \\ 2/n & \text{if } d_S^1 < d_T^1 < d_S^2 \\ 1/2 & \text{if } d_T^1 = d_S^1 = d_S^2 \\ n - 1/2 & \text{if } d_T^1 = d_S^1 < d_S^2 \\ n & \text{if } d_T^1 < d_S^1 \end{cases} \tag{4.6}$$

Similarly, if we let the distances from x_S^i to its two nearest neighbors in $S \setminus \{x_S^i\}$ be d_S^1 and d_S^2. Also, let the distances from x_S^i to its two nearest neighbors in T be d_T^1 and d_T^2. Then:

$$g_S(\mathbf{x}_S^i) = \frac{1}{2}\left[\sum_{k=1}^{n} \mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) > d_{SS}(i)\right) + \sum_{k=1}^{n} \mathbf{1}\left(d_{S(T\setminus\{\mathbf{x}_T^k\})}(i) \geq d_{SS}(i)\right)\right]$$
$$= \begin{cases} 0 & \text{if } d_T^1 \leq d_T^2 < d_S^1 \\ 1/n & \text{if } d_T^1 < d_S^1 = d_T^2 \\ 2/n & \text{if } d_T^1 < d_S^1 < d_T^2 \\ 1/2 & \text{if } d_S^1 = d_T^1 = d_T^2 \\ n - 1/2 & \text{if } d_S^1 = d_T^1 < d_T^2 \\ n & \text{if } d_S^1 < d_T^1 \end{cases} \tag{4.7}$$

Finally, rather than only return the average of the two terms in the formula, each is returned separately, since the expected value is 0.5 for each term.

$$\mathcal{AA}_{TS} = \left(\frac{1}{2n^2}\sum_{i=1}^{n} g_T(\mathbf{x}_T^i)\,,\ \frac{1}{2n^2}\sum_{i=1}^{n} g_S(\mathbf{x}_S^i)\right) \tag{4.8}$$

In future work, we plan to investigate if each of these two terms provides different information about the resemblance of the datasets. The code to compute this is at our github repo[3]

[3]https://github.rpi.edu/RensselaerIDEA/synthetic-data)

4.2 LTU Attacker

In order to evaluate flaws in privacy preservation methods, or empirically estimate the amount of privacy preserved, it is common to construct attacks against that privacy. Li et al. (2013) study what is meant by privacy, and develop a "Membership Privacy" framework [26] focusing on membership inference attacks (MIA), in which the attacker tries to determine if a given point was or was not in the training data. Most of the literature focuses on "black box" attacks [39], in which the attacker can query a model trained on private data, and receive the model output, and must base its attack on that information. However, it is common in the literature to consider worst case scenarios, in which attackers have large advantages, such as information avialable for linkage attacks. Nasr et al. considered white-box attacks[30] in which the attacker could compute the gradients of the loss of the model at the points under attack. We consider the logical extreme, in which the attacker has full access to the trained model, the learning algorithm used to train the model with all of its hyper-parameters, all of the training data (called "defender" data), and another dataset from the same distribution (called "reserve" data), and the attacker knows the membership label of all but two points: one point for the defender data and one point from the reserve. The goal of this "LTU Attacker" is to correctly predict which of the two points is from the "defender" data. The methodology is illustrated in Figure 4.1 on page 48, and explained in detail in the next subsection.

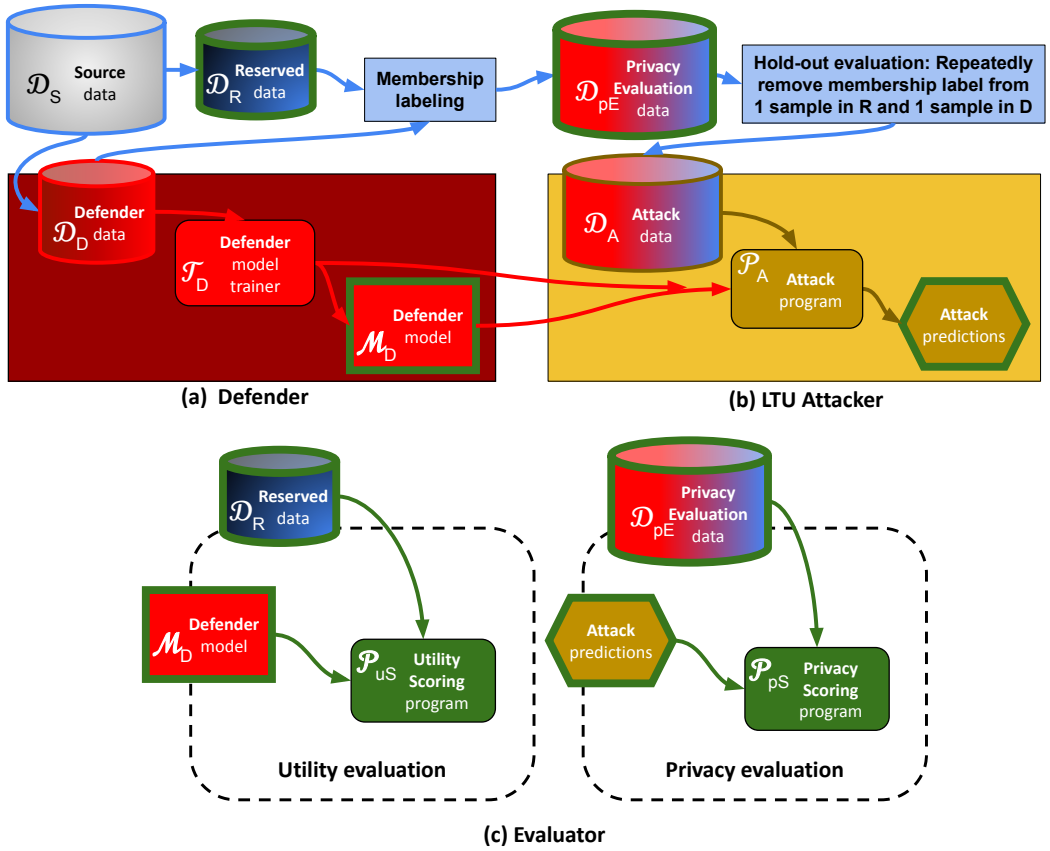

Figure 4.1: Leave Two Unlabeled (LTU) Methodology

4.2.1 Leave Two Unlabeled (LTU) Methodology

We consider the scenario in which an owner of a data Source $\mathcal{D}_S$ wants to create a predictive model trained on some of those data, but needs to ensure that privacy is preserved. In particular, we focus on privacy from membership inference. The data owner entrusts an agent called Defender with creating such a model, giving him access to a random sample $\mathcal{D}_D \subset \mathcal{D}_S$ (Defender dataset). We denote by $\mathcal{M}_D$ the trained model (Defender model) and by $\mathcal{T}_D$ the algorithm used to train it (Defender trainer). The data owner wishes to release $\mathcal{M}_D$, and eventually $\mathcal{T}_D$, provided that certain standards of privacy and utility of $\mathcal{M}_D$ and $\mathcal{T}_D$ are met. To evaluate the privacy and utility, the data owner reserves a dataset $\mathcal{D}_R \subset \mathcal{D}_S$, disjoint from $\mathcal{D}_D$, and gives both $\mathcal{D}_D$ and $\mathcal{D}_R$ to a trustworthy Evaluator agent. The Evaluator tags the samples with dataset "membership tags": *defender* or *reserved*. Then, the Evaluator performs repeated rounds, consisting in selecting one defender sample d and one reserved sample r, and giving to an LTU Attacker an almost perfect attack dataset $\mathcal{D}_A = \mathcal{D}_D \setminus \{\texttt{membership}(d)\} \cup \mathcal{D}_R \setminus \{\texttt{membership}(r)\}$, removing only the membership labels

of the two selected samples. We refer to this procedure as "Leave Two Unlabeled" (LTU), see Figure 4.1. The LTU Attacker also has access to the Defender trainer $\mathcal{T}_D$ (with all its hyper-parameter settings), and the trained Defender model $\mathcal{M}_D$. He is tasked to correctly predict which of the two examples d and r belongs to $\mathcal{D}_D$ (independently for each LTU round, forgetting everything at the end of a round). The membership classification error (one minus the accuracy A_{ltu}) from N independent LTU rounds gives a **global** membership inference privacy score.

$$\texttt{Privacy} = \min\{2\ (1 - A_{ltu}), 1\} \pm 2\sqrt{A_{ltu}(1 - A_{ltu})/N} \tag{4.9}$$

Song et al. (2021) incorporated a fine-grained analysis of privacy risk by giving a score to individual samples[43]. We can also determine an **individual** membership inference privacy scores for any sample $d \in \mathcal{D}_D$ in our framework by using that sample for all N rounds, and only drawing $r \sim \mathcal{D}_R$ at random. Similarly, we can determine an individual non-membership inference privacy score for any sample $r \in \mathcal{D}_R$ by using that sample for all N rounds, and only drawing d at random.

The following small example illustrates that the pairwise prediction accuracy in the LTU methodology is not a function of the accuracy, false positive rate, or false negative rate of predictions made on individual samples. Let $f(x)$ be the discriminative function trained to predict the probability that a sample is in the Reserved set (i.e. predictions made using a threshold of 0.5), and for simplicity consider Defender and Reserved sets with three samples each, such that:

$$\begin{aligned} f(d_1) &= 0.1 \qquad & f(r_1) &= 0.4 \\ f(d_2) &= 0.3 \qquad & f(r_2) &= 0.7 \\ f(d_3) &= c \qquad & f(r_3) &= 0.9 \end{aligned}$$

If c is either 0.6, 0.8, or 0.95, then in all three cases the overall accuracy for individual sample predictions is 2/3, and the false positive rate and false negative rate are both 1/3. However, in the LTU methodology, the LTU Attacker would get 9 different pairs to predict, and for those values of c, its accuracy would be 8/9, 7/9, or 6/9, respectively.

4.2.2 Theoretical Results

We present three theorems outlining weaknesses of the Defender that are particularly easy to exploit by an LTU Attacker. First, we prove, in the context of the LTU procedure, two theorems related to an already known result connecting Privacy and over-fitting: Defender trainers that overfit the Defender data lend themselves to easy attacks [55]. The third theorem concerns deterministic trainers $\mathcal{T}_D$: We show that the LTU Attacker can defeat them with 100% accuracy, under mild assumptions. Thus Defenders must introduce some randomness in their training algorithm to be robust against such attacks [12].

We use the fact that our LTU methodology simplifies the work for the LTU Attacker, since it is always presented pairs of points for which exactly one is in the Defender data. This can give it very simple attack strategies. For example, for any real valued function $f(x)$, with $x \in \mathcal{D}_S$, let r be drawn uniformly from $\mathcal{D}_R$ and d be drawn uniformly from $\mathcal{D}_D$, and define:

$$p_R = \Pr_{\substack{u_1 \sim \mathcal{D}_R \\ u_2 \sim \mathcal{D}_D}} [f(u_1) > f(u_2)] \tag{4.10}$$

$$p_D = \Pr_{\substack{u_1 \sim \mathcal{D}_R \\ u_2 \sim \mathcal{D}_D}} [f(u_1) < f(u_2)] \tag{4.11}$$

Thus p_R is the probability that discriminant function f "favors" Reserved data while p_D is the probability with which it favors the Defender data. $p_R > p_D$ occurs if for a larger number of random pairs $f(x)$ is larger for Reserved data than for Defender data. If the probability of a tie is zero, then $p_R + p_D = 1$.

Theorem 4.2.1. *If there is any function f for which $p_R > p_D$, an LTU Attacker exploiting that function can achieve an accuracy $A_{ltu} \geq \frac{1}{2} + \frac{1}{2}(p_R - p_D)$.*

Proof. A simple attack strategy would be predict that the unlabeled sample with the smaller value of $f(x)$ belongs to the *Defender* data, with ties (i.e. when $f(u_1) = f(u_2)$) decided by tossing a fair coin. This strategy would give a correct prediction when $f(r) > f(d)$, which occurs with probability p_R, and would be correct half of the time when $f(r) = f(d)$, which occurs with probability $(1 - (p_r + p_d))$. This gives a classification accuracy:

$$A_{ltu} = p_R + \frac{1}{2}(1 - p_R - p_D) = \frac{1}{2} + \frac{1}{2}(p_R - p_D) \ . \tag{4.12}$$

□

This is similar to the threshold adversary of [55], except that the LTU Attacker does not need to know the exact conditional distributions, since it can discriminate pairwise. The most obvious candidate function f is the loss function used to train $\mathcal{M}_D$ (we call this a naïve attacker), but the LTU Attacker can mine $\mathcal{D}_A$ to potentially find more discriminative functions, or multiple functions to bag, and use $\mathcal{D}_A$ to compute very good estimates for p_R and p_D. We verify that an LTU Attacker using an f function making perfect membership predictions (e.g. having the knowledge of the *entire* Defender dataset and using the nearest neighbor method) would get $A_{ltu} = 1$, if there are no ties. Indeed, in that case, $p_R = 1$ and $p_D = 0$.

In our next theorem, we show that the LTU Attacker can attain an analogous lower bound on accuracy connected to overfitting as the bounded loss function (BLF) adversary of [55].

Theorem 4.2.2. *If the loss function $\ell(x)$ used to train the Defender model is bounded for all x, without loss of generality $0 \leq \ell(x) \leq 1$ (since loss functions can always be re-scaled), and if e_R, the expected value of the loss function on the Reserved data, is larger than e_D, the expected value of the loss function on the Defender data, then a lower bound on the accuracy of the LTU Attacker is given by the following function of the generalization error gap $e_R - e_D$:*

$$A_{ltu} \geq \frac{1}{2} + \frac{1}{2}(e_R - e_D) \tag{4.13}$$

Proof. If the order of the pair (u_1, u_2) is random and the loss function $\ell(x)$ is bounded by $0 \leq \ell(x) \leq 1$, then the LTU Attacker could predict $u_1 \in \mathcal{D}_R$ with probability $\ell(u_1)$, by drawing $z \sim U(0, 1)$ and predicting $u_1 \in \mathcal{D}_R$ if $z < \ell(u_1)$, and $u_1 \in \mathcal{D}_D$ otherwise. This gives the desired lower bound on the expected accuracy, derived as follows:

$$A_{ltu} = \frac{1}{2} \Pr_{u_1 \sim \mathcal{D}_R}[z < \ell(u_1)] + \frac{1}{2}\left(1 - \Pr_{u_1 \sim \mathcal{D}_D}[z < \ell(u_1)]\right)$$

When u_1 is drawn uniformly from $\mathcal{D}_R$, then:

$$\begin{aligned} \Pr_{u_1 \sim \mathcal{D}_R}[z < \ell(u_1)] &= \frac{1}{|\mathcal{D}_R|} \sum_{u_1 \in \mathcal{D}_R} Pr[z < \ell(u_1)] \\ &= \frac{1}{|\mathcal{D}_R|} \sum_{u_1 \in \mathcal{D}_R} \ell(u_1) = \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_R}[\ell(u_1)] \end{aligned} \tag{4.14}$$

Similarly, when u_1 is drawn uniformly from $\mathcal{D}_D$, then:

$$\begin{aligned} \Pr_{u_1 \sim \mathcal{D}_D}[z < \ell(u_1)] &= \frac{1}{|\mathcal{D}_D|} \sum_{u_1 \in \mathcal{D}_D} Pr[z < \ell(u_1)] \\ &= \frac{1}{|\mathcal{D}_D|} \sum_{u_1 \in \mathcal{D}_D} \ell(u_1) = \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_D}[\ell(u_1)] \end{aligned} \tag{4.15}$$

Substituting in these expected values gives:

$$= \frac{1}{2} \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_R}[\ell(u_1)] + \frac{1}{2} - \frac{1}{2} \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_D}[\ell(u_1)] = \frac{1}{2} + \frac{e_R - e_D}{2} \tag{4.16}$$

where $e_R \coloneqq \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_R}[\ell(u_1)]$ and $e_D \coloneqq \mathop{\mathbb{E}}_{u_1 \sim \mathcal{D}_D}[\ell(u_1)]$

□

This is only a lower bound on the accuracy of the attacker, connected to the main difficulty in machine learning - overfitting of the loss function. Other attack strategies may be more accurate. However, neither of the attack strategies in Theorems 4.2.1 and Theorems 4.2.1 is dominant over the other, which will be shown shortly. The strategy in Theorem 4.2.1 is more widely applicable, since it does not require the function to be bounded.

In the special case when the loss function used to train the Defender model is the 0-1 loss, and that is used to attack (i.e. $f = \ell$), the strategies in Theorems 1 and 2 are different,

but have the same accuracy:

$$\begin{aligned} p_R &= \Pr_{u\sim\mathcal{D}_R}[\ell(u)=1](1-\Pr_{u\sim\mathcal{D}_D}[\ell(u)=1]) \\ p_D &= (1-\Pr_{u\sim\mathcal{D}_R}[\ell(u)=1])\Pr_{u\sim\mathcal{D}_D}[\ell(u)=1] \\ p_R - p_D &= \Pr_{u_1\sim\mathcal{D}_R}[\ell(u)=1] - \Pr_{u\sim\mathcal{D}_D}[\ell(u)=1] \\ &= e_R - e_D \end{aligned}$$

Note that u is a dummy variable. The first line of the derivation is due to the fact that the only way the loss on the Reserved set can be greater than the loss on the Defender set is if the loss on the Reserved set is 1, which has probability $Pr_{u\sim\mathcal{D}_R}[\ell(u)=1]$, and the loss on the Defender set is zero, which has probability $1 - Pr_{u\sim\mathcal{D}_D}[\ell(u)=1]$. The second line is derived similarly.

Theorem 4.2.3. *Neither of the strategies from Theorem 4.2.1 or Theorem 4.2.2 is dominant over the other.*

Proof. We present two simple examples to show that neither of the strategies dominates over the other.

For example 1, assume that for $d \sim \mathcal{D}_D$ the loss function takes the two values 0 or 0.5 with equal probability, and that for $r \sim \mathcal{D}_R$ the loss function takes the two values 0.3 or 0.4 with equal probability. Then $p_R = p_D = 1/2$ so that $p_R - p_D = 0$, so that an LTU attacker employing the strategy from Theorem 4.2.1 would do no better than random guessing. However, $e_R = 0.35$ and $e_D = 0.25$, so $e_R - e_D = 0.1$, so that an LTU attacker employing the strategy from Theorem 4.2.2 would do better than random guessing.

For example 2, the joint probability mass function in Table 4.1 can be used to compute that $(e_R - e_D) = 0.15 < (p_r - p_d) = 0.22$.

Table 4.1: Example joint PMF of bounded loss function, for $r \sim \mathcal{D}_R$ and $d \sim \mathcal{D}_D$. The attack strategy in Theorem 4.2.2 outperforms the attack strategy in Theorem 4.2.1 on this data.

	$l(r)$			
	0	1/2	1	row sum
$l(d) = 0$	0.24	0.24	0.12	0.6
$l(d) = 1/2$	0.12	0.12	0.06	0.3
$l(d) = 1$	0.04	0.04	0.02	0.1
column sum	0.4	0.4	0.2	

□

The preceding attack strategies only involved using the Defender model, but the LTU Attacker can also use the Defender trainer, which gives it a perfect attack against certain deterministic models.

Theorem 4.2.4. *If the Defender trainer $\mathcal{T}_D$ is deterministic, invariant to the order of the training data, and injective, then the LTU Attacker has an optimal attack strategy which achieves perfect accuracy.*

Proof. The proof uses the fact that the LTU Attacker knows all of the Defender dataset except one sample, and knows that the missing sample is either u_1 or u_2. Therefore, the attack strategy is to create two models, one trained on u_1 combined with the rest of the Defender dataset, and the other trained on u_2 combined with the rest of the Defender dataset. Since the Defender trainer is deterministic, one of those two models will match the Defender model, revealing which unlabeled sample belonged in the Defender dataset.

Formally, denote the subset of $\mathcal{D}_A$ labeled "Defender" as $\mathcal{D}_D \setminus \{d\}$, and the two membership unlabeled samples as u_1 and u_2. The attacker can use the Defender trainer with the same hyper-parameters on $(\mathcal{D}_D \setminus \{d\}) \cup \{u_1\}$ to produce model $\mathcal{M}_1$ and on $(\mathcal{D}_D \setminus \{d\}) \cup \{u_2\}$ to produce model $\mathcal{M}_2$.

By definition of the LTU Attacker, the missing sample d is either u_1 or u_2, and $\mathcal{D}_D \cap \mathcal{D}_R = \emptyset$, so $u_1 \neq u_2$. There are two possible cases. If $u_1 = d$, then $\mathcal{D}_D = (\mathcal{D}_D \setminus \{d\}) \cup \{u_1\}$, so that $\mathcal{M}_1 = \mathcal{M}_D$, since $\mathcal{T}_D$ is deterministic and invariant to the order of the training data. However, $\mathcal{D}_D \neq (\mathcal{D}_D \setminus \{d\}) \cup \{u_2\}$, since $u_2 \neq u_1$, so $\mathcal{M}_2 \neq \mathcal{M}_D$, since $\mathcal{T}_D$ is also injective.

Therefore, the LTU Attacker can know, with no uncertainty, that u_1 has membership label "Defender" and u_2 has membership label "Reserved". The other case, for $u_2 = d$, has a symmetric argument. □

Under the hypotheses of Theorem 4.2.4, the LTU Attacker achieves the optimal Bayesian classifier using:

$$Pr[u_i \in \mathcal{D}_D | \mathcal{M}_i = \mathcal{M}_D] = 1 \qquad Pr[u_i \in \mathcal{D}_R | \mathcal{M}_i = \mathcal{M}_D] = 0$$
$$Pr[u_i \in \mathcal{D}_D | \mathcal{M}_i \neq \mathcal{M}_D] = 0 \qquad Pr[u_i \in \mathcal{D}_R | \mathcal{M}_i \neq \mathcal{M}_D] = 1$$

This weakness of deterministic algorithms is analogous to the fact that no data-dependent, deterministic algorithm is differentially private [12]. Invariance to the order of the training data was needed because the LTU Attacker is defined to have the same set of data as the defender, but sets are not ordered. In our research, we found two different sources of randomness impacted the privacy of a model: the random bits in the training algorithm (if nondeterministic) and the order of the training samples.

Injectivity is certainly a strong assumption that many machine learning model trainers do not satisfy. If a deterministic model trainer learns the same exact model for training set $(\mathcal{D}_D \setminus \{d\}) \cup \{u_1\}$ as it does for training set $(\mathcal{D}_D \setminus \{d\}) \cup \{u_2\}$, then the LTU attacker cannot do better than random guessing for the pair of untagged samples (u_1, u_2). For example, for a support vector machine, if neither u_1 nor u_2 would be support vectors when combined with $\mathcal{D}_D \setminus \{d\}$, then that pair cannot be attacked by the LTU attacker. This is obviously highly dependent on $\mathcal{D}_D \setminus \{d\}$. The lack of injectivity can afford some sample, possibly many samples, much privacy, usually at the expense of the privacy of other samples (e.g. the support vectors of an SVM).

4.2.3 Data and Experiments

We used three datasets for our experiments: QMNIST , CIFAR-10 and CIFAR-100. QMNIST [52] was created by Facebook Research from the NIST Special Database 19 [19], the same database used to create the famous MNIST dataset of handwritten digits. QMNIST was created with the goal of matching the preprocessing as closely as possible, but using all 402953 images in the database and retaining the metadata, such as a writer ID

that links images written by the same individual. We do not use the metadata for our analysis. CIFAR-10 and CIFAR-100 [23] are datasets each containing 60,000 images from the 80 million tiny images dataset, collected by Alex Krizhevsky and his collaborators. The images in the datasets are 32 pixels by 32 pixels, with 3 color channels. CIFAR-10 contains 10 mutually exclusive classes, with 6,000 images per class, while CIFAR-100 contains 100 mutually exclusive classes (grouped into 20 super classes), with 600 images per class. We were interested in attacking tabular data instead of images, so we preprocessed the images through a pretrained network and used the second to last layer as the features for our dataset (instead of the images). For QMNIST, we used VGG19 [40] trained on Imagenet [9] for preprocessing. For CIFAR-10, we used Efficient-netv2 [46] trained on Imagenet21k with fine-tuning on CIFAR-100. For CIFAR-100, we did use the raw images.

For our first set of experiments, we used ten common models as Defender models to attack, using their implementation in the popular Scikit-learn [34] python library, with all default parameters. We used QMNIST and CIFAR-10 for these experiments, randomly selecting 1,600 samples for the Defender dataset and 1,600 samples for the Reserve dataset. We performed 100 LTU rounds (as described in the methodology) and recorded the utility and privacy achieved. In order to measure the effects of the different sources of randomness described in our theoretical analysis, our experimental design included three different scenarios: "original" = the attacker has the original order of the training data and the algorithm has no randomness (achieved by using a fixed seed), "random" = the order of the training data is random, but the algorithm still has a fixed seed, and "unseeded" = the order of the training data is random and the algorithm is random (unseeded).

In our second set of experiments, we use AlexNet [24] for the Defender model trainer and all 50,000 images in the CIFAR-100 training data for the Defender data. For the attack program, we use the ML Privacy Meter of the NUS Data Privacy and Trustworthy Machine Learning Lab, from `https://github.com/privacytrustlab/ml_privacy_meter`.

4.2.4 Results

In line with our theoretical analysis, we find that machine learning algorithms which are completely deterministic afford no privacy, which is why logistic regression, ridge regression,

naive Bayes, and the support vector classifier all have zero privacy in the following Figure 4.2. However, algorithms like k-nearest neighbors, which do not satisfy the injectivity hypothesis of Theorem 4.2.1, can confer some privacy even when they are not random. In opposition to our theoretical analysis, we found that the Scikit-Learn SGD classifier and perceptron conferred some privacy even when we tried to remove all randomness. Our suspicion is that there was randomness in the implementation for which we were not able to account.

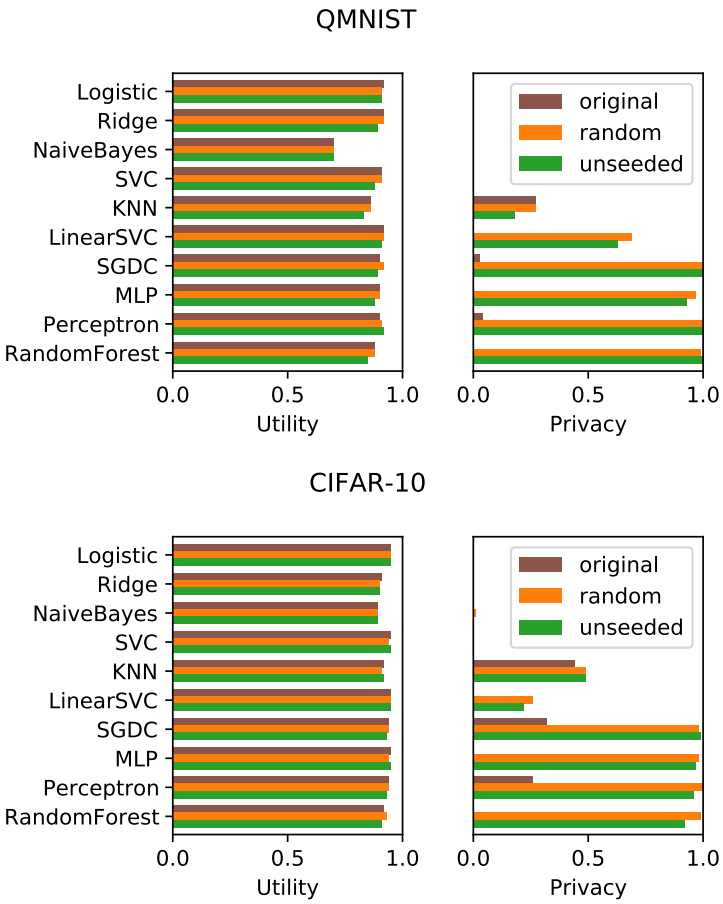

Figure 4.2: Utility and Privacy of different scikit-learn models (missing bar indicates 0)

We used the ML Privacy Meter of the NUS Data Privacy and Trustworthy Machine Learning Lab[4] to run their attack of AlexNet[24], which achieved an attack accuracy of 74.9%. Their attack model predicts the probability that each sample was in the Defender dataset. The histograms of the predictions over the Defender dataset and Reserved dataset follow:

[4] `https://github.com/privacytrustlab/ml\_privacy\_meter`.

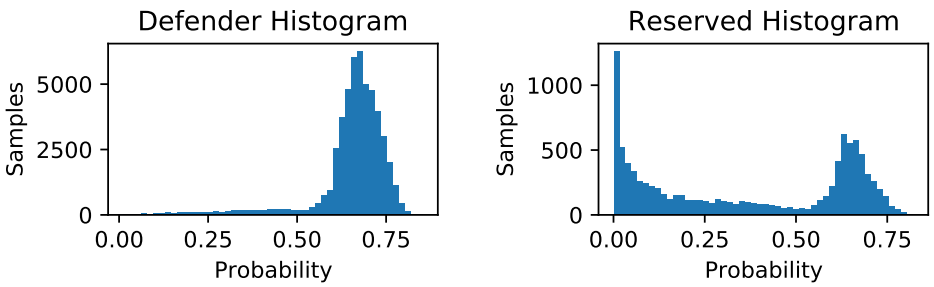

Figure 4.3: Histogram of Predicted Probabilities

Using their attack model predictions in the LTU methodology, so that the attacker is always shown pairs of points for which exactly one was from the Defender dataset, increased the overall attack accuracy to 81.3%. Furthermore, by evaluating the accuracy of the attacker on each Defender sample individually (against all Reserved samples), we computed easy to interpret individual privacy scores for each sample in the Defender dataset:

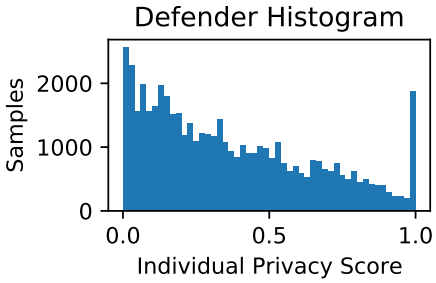

Figure 4.4: Histogram of Individual Privacy Scores for each sample in the Defender dataset

4.3 Conclusion

In this section, we improved a metric for empirically estimating privacy loss, and proved that the improved version is unbiased. We showed how the metric can be efficiently implemented, and also corrected for discrete distances. Next, we introduced a novel privacy attack methodology, and proved several theoretical results on the privacy leakage for deterministic or overfit models. Our theoretical analysis highlighted the effect of randomness, including the order of training examples for learning algorithms than iteratively converge to the trained model. We confirmed this theoretical analysis through experimental results achieved attacking several widely used machine learning models on two popular datasets.

CHAPTER 5
Optimally Soft Labels for Membership Privacy

The third problem addressed was to further formalize our "LTU Attacker" methodology so that we can understand the problem of membership inference privacy at the foundational level, with the ultimate goal of finding an effective way to protect that privacy, and provide practical recommendations to anyone training machine learning models, which they want to release while maintaining some protection on the privacy of the individuals whose data the model was trained on. We prove two foundational results, which stated informally say that: (1) there is unavoidably some risk to membership inference privacy loss for any training data that impact the model learned, and (2) under certain conditions, a black-box attack is theoretically optimal. The formalization of our "LTU Attacker" methodology and the analysis leading up to the first theorem of this chapter motivated a novel defense strategy which we call optimally soft labels. We present initial experimental findings, which suggest this is a useful defense strategy, and conclude with our planned direction for future work.

The rest of this chapter is organized as follows. We first formally state the membership inference problem and our methodology, including defining terms and our notation. We then present our theoretical analysis, which includes proofs of theorems and a description of our proposed defense strategy. Finally, we describe the experiments we have completed thus far and our initial findings, and outline our future work.

The specific contributions of this chapter include:

- A formalization of the membership inference privacy problem including an aspect not yet analyzed in the literature
- Theoretical results including a proof that under certain hypotheses that a black-box attack is optimal
- A novel, easy to implement, widely applicable defense strategy motivated by our theoretical analysis
- Initial results showing that our defense strategy can reduce membership inference privacy loss with minimal impact to utility, and in fact, that in some scenarios our defense

strategy can simultaneously improve both privacy and utility.

5.1 Problem Statement and Methodology

The goal of supervised machine learning is to learn a useful function from labeled training data. We address the problem of protecting the privacy of the training data used in supervised learning, and evaluating the amount of privacy lost by releasing the trained model. We will restrict our analysis to privacy from membership inference, which is how well an attacker can infer whether or not a given sample was in the training data. Yeom et al. (2018) prove attribute inference, another type of attack in which the attacker tries to infer missing attributes given other attributes from a sample in the training data, is "harder" than membership inference.[37] We will not consider an "adversary [that] interferes with the training in any way", which Rigaki and Garcia (2020) refer to as **active** attackers. We also assume that the attacker cannot access the particular random bits used in or intermediate computations from the learning algorithm during training. To formally analyze this membership inference problem, we use our LTU Attacker framework as previously described, but with more details about the data space and model space, which will aid in our analysis.

In the LTU Attacker framework, a data owner draws samples i.i.d from a distribution $\mathbb{P}_U$ on a data space $\mathcal{U} \subset \mathcal{X} \times \mathcal{Y}$, and splits the data randomly into a training set $d_{\text{train}}^{(n_1)}$ and a reserve set $d_{\text{reserve}}^{(n_2)}$. Superscripts in parentheses represent the number of samples in the set. For each sample $u = (x, y)$, the "features" are x and the "label" is y. The name "u" can be thought of as "untagged". We can also prepend a "membership tag" w onto the sample, to record which dataset it belonged to, forming the "tagged" sample $t = (w, x, y)$.

The data owner uses the training data as input to a model trainer $\mathcal{T}_h$, where h indicates the hyper-parameters used, to learn a model $\widetilde{M} = \mathcal{T}_h(Z, d_{\text{train}}^{(n_1)})$. Note that for our formal analysis, only parameters that are chosen independently of the data are called "hyper-parameters", and any selection of other parameters based in any way on the data must be part of the model trainer. For example, if the model is a neural network, and the number of nodes per hidden layer is chosen a priori, perhaps from some data-independent heuristic, then those would be hard-coded into $\mathcal{T}_h$ as hyper-parameters. On the other hand, if the number of nodes per layer is chosen based in any way on the training data, such as by taking

a subset of $d_{\text{train}}^{(n_1)}$ as a validation set and determining the number of nodes that results in the lowest validation error, then the exact method by which this is done must be part of the complete algorithm $\mathcal{T}_h$, and the number of nodes would be considered parameters learned by the algorithm. The guiding principle for our analysis is that the learning algorithm is the complete process by which a model is selected; nothing is "hidden".

Here Z represents any random bits used in the algorithm, drawn from $\mathbb{P}_Z$. Random quantities will be denoted by capital letters, and realizations will be denoted by lower case, so for specific random bits z, the model trainer will learn the model $\widetilde{m} = \mathcal{T}_h(z, d_{\text{train}}^{(n_1)})$. The tildes over the models represents that these are bits in some fixed region of memory storing a data structure. Some program interprets this model $\widetilde{m}$ with an input $x \in \mathcal{X}$ to produce an output $y \in \mathcal{Y}$, implementing the function $m : \mathcal{X} \to \mathcal{Y}$. We require the encoding of functions to be one-to-one, so that each m corresponds to a unique $\widetilde{m}$.

Therefore, given the setup of the problem, the attacker's ability to infer membership, which is how well an attacker can infer whether or not $u \in d_{\text{train}}^{(n_1)}$ for a sample $u \in \mathcal{U}$, can only be based on knowledge gained from some access to: $\mathcal{T}_h, \mathbb{P}_U, \mathbb{P}_Z, n_1, d_{\text{train}}^{(n_1)}, \widetilde{m}, m$, or parts thereof. Our main focus will be on how much membership privacy is lost to a sample $u \in d_{\text{train}}^{(n_1)}$ from an LTU attacker having complete access to all of these and to $d_{\text{reserve}}^{(n_2)}$, except that the attacker does not know, for two untagged samples (u_1, u_2), which of the two is from the training set (knowing that exactly one is). The attacker knows the membership tag of all the other samples in $d_{\text{train}}^{(n_2)}$ and $d_{\text{reserve}}^{(n_2)}$. The maximum privacy loss for any sample u from the training set in this scenario, over all possible partner samples, is an upper bound on the privacy loss in any scenario (for the same fixed training set), since the only attacker that could know any more information that could be used to infer the membership of u, would have to already know the membership of u. It is important to note that our privacy analysis does **not** assume that an attacker *will* have complete access to these; this is a worst case analysis. Any privacy analysis which derives a tight upper bound on privacy loss which is lower than ours must necessarily either assume that the attacker will not have access to some of these or prove that access to them will not provide the attacker any advantage.

In our analysis, all sets are finite, which may at first seem like an approximation, but

actually using continuous functions would be the approximation, since in practice quantities are typically represented by fixed-width encodings such as `float32` or `int32`. For the purposes of our analysis, these encodings can have an arbitrarily large number of bits as long as they are fixed, the number of hyper-parameters can be arbitrarily large as long as there are only finitely many, and the number of dimensions of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ can be arbitrarily large, as long as they are finite dimensional. Then, as stated, all sets will be finite. Also, all distributions will be discrete.

5.2 Theoretical Analysis

5.2.1 Privacy Loss to Optimal Attack

In order to give our attacker the maximum possible advantage, it is given a pair of membership unlabeled samples u_1 and u_2 to attack, as well as $d_{\text{train}}^{(n_1-1)}$ and $d_{\text{reserve}}^{(n_2-1)}$, and knows that $d_{\text{train}}^{(n_1)} = d_{\text{train}}^{(n_1-1)} \cup \{u_1\}$ or $d_{\text{train}}^{(n_1)} = d^{(n_1-1)} \cup \{u_2\}$, with equal probability (in the Bayesian sense). In this setup, knowledge of $\mathbb{P}_U$ does not provide any membership information, since both u_1 and u_2 were drawn from that distribution, and their assignment to the training or reserve sets was random.

Let $d_i^{(n_1)} = d_{\text{train}}^{(n_1-1)} \cup \{u_i\}$ and $M_i = \mathcal{T}_h(Z, d_i^{(n_1)})$. If the LTU Attacker can use $\mathcal{T}_h$ to get very accurate estimates of $\Pr_{Z\sim\mathbb{P}_Z}(M_1 = m)$ and $\Pr_{Z\sim\mathbb{P}_Z}(M_2 = m)$, then the posterior probability that $u_1 \in d^{(n_1)}$ can be derived as follows. First, from Bayes' theorem we get:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) = \frac{\Pr\left(M = m \middle| u_1 \in d_{\text{train}}^{(n_1)}\right) \Pr\left(u_1 \in d_{\text{train}}^{(n_1)}\right)}{\Pr(M = m)} \tag{5.1}$$

By the definition given for the random variable M_i, we know that:

$$\Pr\left(M = m \middle| u_i \in d_{\text{train}}^{(n_1)}\right) = \Pr(M_i = m) \tag{5.2}$$

And from the problem setup, we know that:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)}\right) = \Pr\left(u_2 \in d_{\text{train}}^{(n_1)}\right) = \frac{1}{2} \tag{5.3}$$

Which in turn gives us:

$$\Pr(M = m) = \frac{1}{2}\Pr(M_1 = m) + \frac{1}{2}\Pr(M_2 = m) \tag{5.4}$$

Putting these all together yields:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) = \frac{\Pr(M_1 = m) \cdot \frac{1}{2}}{\frac{1}{2}\Pr(M_1 = m) + \frac{1}{2}\Pr(M_2 = m)} \tag{5.5}$$

$$= \frac{\Pr(M_1 = m)}{\Pr(M_1 = m) + \Pr(M_2 = m)} \tag{5.6}$$

$$= \frac{1}{1 + \dfrac{\Pr(M_2 = m)}{\Pr(M_1 = m)}} \tag{5.7}$$

Which in logarithm form can be expressed as:

$$\text{logit}\left(\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right)\right) = \log\left(\frac{\Pr(M_1 = m)}{\Pr(M_2 = m)}\right) = \gamma(u_1, u_2) \tag{5.8}$$

The quantity in equation (5.8), denoted by $\gamma(u_1, u_2)$, is the "strength of the evidence" in support of $u_1 \in d_{\text{train}}^{(n_1)}$ that the attacker gains from conditioning on m, given the other information it has. The attacker predicts: $u_1 \in d_{\text{train}}^{(n_1)}$ if $\gamma(u_1, u_2) > 0$; $u_2 \in d_{\text{train}}^{(n_1)}$ if $\gamma(u_1, u_2) < 0$; or guesses randomly if $\gamma(u_1, u_2) = 0$. This is the Bayes' optimal classifier of membership classification. If the probabilities to be estimated are too small and $\mathcal{T}_h$ is computationally expensive, it may be impractical to get accurate estimates to conduct this attack. Nevertheless, this quantifies the *theoretical* membership privacy lost by publishing $\mathcal{T}_h$ and m.

If for all possible training data, samples u_1 and u_2, and models m, $|\gamma(u_1, u_2)| \leq \varepsilon$, then $\mathcal{T}_h$ is by definition an ε-differentially private algorithm, refer to Definition 2.2.3 on page 13 with $\delta = 0$. In that framework, ε is referred to as the "privacy loss" of $\mathcal{T}_h$.

$$\varepsilon = \max_{\substack{u_1 \in \mathcal{X}\times\mathcal{Y} \\ u_2 \in \mathcal{X}\times\mathcal{Y} \\ m \in \mathcal{M} \\ d_{\text{train}}^{(n_1-1)} \in (\mathcal{X}\times\mathcal{Y})^{(n_1-1)}}} \left|\log\left(\frac{\Pr(M_1 = m)}{\Pr(M_2 = m)}\right)\right| \tag{5.9}$$

Rewriting Equation (5.8) in terms of this privacy loss parameter ε, we see that for a differ-

entially private algorithm the maximum posterior probability of membership is given by:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) \leq \frac{1}{1+e^{-\varepsilon}} \tag{5.10}$$

It should be noted that the strength of the evidence is impacted by both u_1 and u_2. If the ratio $\frac{\Pr(M_1=m)}{\Pr(M_2=m)}$ differs significantly from one, it could be because either of the probabilities differs significantly from the average (for a sample drawn randomly from $\mathbb{P}_U$) or because they both differ by a different amount. In order to get an individual measure of the privacy loss per sample, we can compare each probability to the average $\Pr(M = m)$, defined as training with $d_{\text{train}}^{(n_1-1)}$ plus a random sample drawn from $\mathbb{P}_U$:

$$\log\left(\frac{\Pr(M_1 = m)}{\Pr(M_2 = m)}\right) = \log\left(\frac{\Pr(M_1 = m)}{\Pr(M = m)}\right) - \log\left(\frac{\Pr(M_2 = m)}{\Pr(M = m)}\right) = \gamma(u_1) - \gamma(u_2) \tag{5.11}$$

For each sample $u_1 \in d_{\text{train}}^{(n_1)}$, we refer to $\gamma(u_1) = \log\left(\frac{\Pr(M_1=m)}{\Pr(M=m)}\right)$ as the "effective privacy loss" for that sample. Note that this is not meant to imply that the algorithm achieved differential privacy at that level. It is a convenient scale for measuring individual membership privacy loss in the LTU Attacker framework that has a direct connection to the privacy loss measure in differential privacy. For algorithms which are provably ε-differentially private, the "effective privacy loss" for any sample is bounded by ε. As mentioned in the literature review, each step of a differentially private algorithm has a privacy loss, and the composition of the steps results in a total privacy loss that is the sum of the privacy losses from each step. Often, a certain maximum privacy loss, called the privacy budget, is set, and training terminates if that limit is reached. A privacy budget such as $\varepsilon = 10$, common in the literature, corresponds to a maximum posterior probability of membership equal to $\frac{1}{1+e^{-10}} \approx 99.995\%$ (not much privacy).

5.2.2 Privacy Risk

Even if u_1 has $\gamma(u_1) = 0$, its membership will still be inferred correctly if paired with a sample that is easy to infer as a non-member, i.e. $\gamma(u_2) < 0$. The LTU attacker will be no better attacking u_1 than random guessing exactly when $\gamma(u_1)$ is the median of the distribution $\mathbb{P}_\Gamma(\gamma(u_i)|d^{(n_1-1)})$ with $u_i \sim \mathbb{P}_U$. If the median is zero, then there is privacy loss

to u_1 unless $\Pr(M_1 = m) = \Pr(M = m)$, meaning that the model m was no more likely with u_1 in the training set than without it in the training set (replaced by any random sample). In that sense, any time u_1 is learned from (effects the distribution of models learned by the trainer), there is a risk of privacy loss to it. Stated formally:

Theorem 5.2.1 (No learning without risk to privacy)**.** *In the LTU framework, for a fixed training set $d^{(n_1)}$ and sample $u_1 \in d^{(n_1)}$ there is zero probability of the attacker having a better than random attack against u_1 if and only if for all $u_i \sim \mathbb{P}_U$ with $\Pr(u_i) > 0$, and defining $d_i^{(n_1)} = d_{\text{train}}^{(n_1-1)} \cup \{u_i\}$ and $M_i = \mathcal{T}_h(Z, d_i^{(n_1)})$, we have $\Pr(M_1 = m) = \Pr(M_i = m)$ for all m.*

Proof. Assume that there exists $u_i \in \mathcal{U}$ with $\Pr(u_i) > 0$ and $\Pr(M_i = m_a) \neq \Pr(M_1 = m_a)$.

If $\Pr(M_i = m_a) > \Pr(M_1 = m_a)$, then since $\sum_m \Pr(M_i = m) = \sum_m \Pr(M_1 = m) = 1$, there must also exist m_b with $\Pr(M_1 = m_b) > \Pr(M_i = m_b) \geq 0$.

Let m be the model such that $\Pr(M_1 = m) > \Pr(M_i = m) \geq 0$. Then the probability of the LTU attacker attacking (u_1, u_2) and model m is $\Pr(u_i)\Pr(M_1 = m) > 0$, and the Bayes' optimal classifier will correctly classify $u_1 \in d^{(n_1)}$ and know that:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) = \frac{\Pr(M_1 = m)}{\Pr(M_1 = m) + \Pr(M_i = m)} > 0.5$$

On the other hand, if there does not exist such a $u_i \sim \mathbb{P}_U$, then for any pair (u_1, u_2) and model m, the Bayes' optimal classifier has $\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) = 0.5$. □

For an example of when u_1 would not effect learning, we can consider when only one of the available models m correctly classifies every sample drawn from $\mathbb{P}_U$, that model is already determined by $d_{\text{train}}^{(n_1-1)}$, and $\mathcal{T}_h$ will find it. In that case, for any pair (u_1, u_2), $\Pr(M_1 = m) = \Pr(M_2 = m) = 1$. This is of course highly dependent on $d_{\text{train}}^{(n_1-1)}$, and u_1 could have had a non-zero "effective privacy loss" had its effect on the trainer not been masked by the other samples.

5.2.3 Black-box vs White-box Access

As discussed in the previous chapter, a concept central to the study of privacy attacks on machine learning models is the distinction between black-box attacks and white-box at-

tacks. There are a variety of definitions in the literature, but we will define black-box access to a model as only allowing an attacker to make (unlimited, unrestricted) queries to the model and receive the outputs of those queries. Similarly, we will define black-box access to a model trainer to allow querying the model trainer with a training set and random seed in order to train a model. Combining these, having only black-box access to a model trainer and the model it trains means that an attacker can input a training set and random seed into the model trainer, then input queries to the resultant model, and receive the outputs of those queries.

White-box access allows some information about/from the model's internal structure. There are many different aspects of a model and/or the model trainer that the attacker may or may not have access to, so white-box is probably a very ambiguous term in the literature. The attacker could have access to the type of model (e.g. neural network), the specific architecture of the model (e.g. number of layers, nodes per layer, activation functions, etc.), the loss function used to train the model, the values of intermediate computations performed during queries, etc. We will define white-box access to a model as an attacker having complete access to and knowledge of the model, i.e. having the model on the attacker's computer without any part hidden, encrypted, or otherwise obscured. Similarly, we define white-box access to the model trainer as the attacker having complete access to the learning algorithm used, including how any data-dependent hyper-parameters are chosen and how any privacy protection is added. The only part of the model trainer that the attacker is not given is the particular random seed used. If the attacker had the random seed used, and the order of the training data, then the learning algorithm would be deterministic, and as proven earlier in Theorem 4.2.4, there would be no privacy unless the sample under attack did not impact the learning (e.g. a Support Vector Classifier and a sample that was not a support vector).

Sablayrolles et al. (2019) prove under certain assumptions that black-box attacks are optimal and state that "to the best of [their] knowledge, the literature does not report white-box attacks outperforming the state of-the-art black-box attacks"[38], and prove in their paper that " white-box attacks don't provide any additional information and result in the same optimal [attack] strategy"[38]. However, Nasr et al. (2020) show that they can achieve better empirical results with white-box attacks by using the gradients computed

at the points under attack as input features to a model which predicts membership [30]. The analysis of Sablayrolles et al. (2019) was based on access to the loss function, and assumptions on the distributions of the parameters of the model. We similarly show that having only black-box access to the model and model trainer (with the ability to set the seed on the trainer), perhaps through machine learning as a service, does not theoretically provide any privacy protection over releasing the model and model trainer. However, our analysis does not require access to the loss function or any assumptions on the distribution of parameters.

Theorem 5.2.2 (No privacy protection by obscurity)**.** *For a fixed training set $d^{(n_1)}$, sample $u_i \in d^{(n_1)}$, trained model $\widetilde{m}$, and pair (u_1, u_2) for which the attacker knows that exactly one was in the training set, having only black-box access to $\widetilde{m}$ and the model trainer $\mathcal{T}_h$, with unlimited queries, is sufficient to perform the optimal attack.*

Proof. As shown already, the optimal attack is the Bayes' optimal classifier, with:

$$\Pr\left(u_1 \in d_{\text{train}}^{(n_1)} \middle| M = m\right) = \frac{\Pr(M_1 = m)}{\Pr(M_1 = m) + \Pr(M_2 = m)}$$

Since the attacker can query the trained model $\widetilde{m}$ with all of the finitely many $x_i \in \mathcal{X}$, the attacker can determine the function m by determining all ordered pairs $(x_i, m(x_i))$, which uniquely determines the function.

Similarly, since the attacker can query the trainer with a training dataset and seed, the attacker can query it with $d^{(n_1-1)} \cup \{u_1\}$ and $d^{(n_1-1)} \cup \{u_2\}$, each with every possible seed. Each time it does, the attacker can also query the model $\widetilde{m_j}$ trained that iteration with all of the finitely many $x_i \in \mathcal{X}$ to determine the exact function learned, by determining all ordered pairs $(x_i, m_j(x_i))$, which uniquely determines the function. From this procedure, the attacker can compute $\Pr(M_1 = m)$ and $\Pr(M_2 = m)$ exactly, achieving the optimal attack.

□

It may seem like access to the code for the model $\widetilde{m}$ could provide some theoretically better attack. However, since we require there to be a one-to-one correspondence between $\widetilde{m}$ and m, $M_i = m$ if and only if $\widetilde{M_i} = \widetilde{m}$, providing the same posterior probability. For example, if the code for $\widetilde{m}$ memorizes the training set, that implies that there is a corresponding model m such that $\Pr(M = m | d_{\text{train}}^{(n_1)}) = 1$. In that case, either $Pr(M_1 = m) = 1$ and

$Pr(M_2 = m) = 0$, or $Pr(M_1 = m) = 0$ and $Pr(M_2 = m) = 1$, so the Bayes optimal membership classifier achieves an accuracy of 1 (no privacy at all).

5.3 Optimally Soft Labels

In this section we depart from the purely theoretical analysis toward a discussion of practical considerations for conducting an approximation to the Bayes optimal attack when time and computational resources are limited. This analysis of how these attacks are typically approximated in the literature will motivate our novel defense strategy, which we call "optimally soft labels". We will give a precise definition of these labels, as well as the pseudocode for how to implement them with any model trainer, which could include other defense strategies.

The theoretical attack described previously is computationally infeasible for realistic sizes on the data space and model space. Nevertheless, the privacy loss derived *is* the theoretical privacy loss, so any tighter bound on the privacy loss would have to place restrictions on the attacker, such as on available computational resources or computation time.

In order to make this style of attack computationally feasible, the attacker will likely need to project the model space into some smaller set (e.g. lower dimensional), to compute $\Pr(g(M_i) = g(m))$ instead. Obviously the attacker would want to use some aggregating function g such that:

$$\frac{\Pr(M_1 = m)}{\Pr(M_2 = m)} \approx \frac{\Pr(g(M_1) = g(m))}{\Pr(g(M_2) = g(m))} \tag{5.12}$$

Common in the literature is to use a different projection function for each $u_i = (x_i, y_i)$, of the form $g_i : m \to \mathbb{R}$, where $g_i(m) = h(m(x_i), y_i)$ for some function $h : \mathcal{Y} \to \mathcal{Y}$. The intuition behind this choice is the fact that the model m is learned to predict labels, so the projection used to estimate $\Pr\left(u_i \in d_{\text{train}}^{(n_1)} \middle| g_i(M_i) = g_i(m)\right)$ should be a function of the label predicted by m and the true label y_i. The most common choice for h is the loss function used in $\mathcal{T}_h$. Other common choices are the predicted probability of the true class, such as recent work by Carlini et al. (2022)[6], or the entropy of the predicted label. Song et al. (2021) develop a modified entropy-like function which also uses the true label[43]. Using this attack

methodology, the LTU attacker predicts $u_i \in d_{\text{train}}^{(n_1)}$ for:

$$i = \underset{j \in \{1,2\}}{\arg\max} \left[\gamma_{g_j}(u_j) \right] = \underset{j \in \{1,2\}}{\arg\max} \left[\log \left(\frac{\Pr(g_j(M_j) = g_j(m))}{\Pr(g_j(M) = g_j(m))} \right) \right] \tag{5.13}$$

To completely defeat any attack that incorporates this technique into its strategy, it would be sufficient to ensure that for all $u_i \in d_{\text{train}}^{(n_1)}$ the distributions of $M_i(x_i)$ and $M(x_i)$, i.e. the labels predicted when u_i is **in** or **not in** the training set, were identical. Of course, that is completely at odds with the goal of learning from the training data to predict better labels. Hence, to balance the two goals, privacy and utility, we need to allow u_i to effect the distribution, but bound how much. That is exactly what differential privacy does, and if we chose an ε that guaranteed the desired bound on membership inference, that would be a suitable strategy. Instead, however, preserving utility often requires choosing an ε too high to guarantee the desired bound (e.g. $\varepsilon \approx 10$) with the justification that the results seem empirically to have adequate privacy, although sometimes only on average, leaving some samples too exposed.

In order to provide better protection for the more exposed samples, we employ a novel strategy of training on "optimally soft labels" which interpolate between the mean of $\mathbb{P}_{\hat{Y}_i}$, the distribution of predicted labels when u_i is in the training set, and the true label y_i. Our first method for interpolating was given by the following equation:

$$y_{\text{target}} = \alpha y + (1 - \alpha)\mu_{\hat{Y}} \tag{5.14}$$

Choosing $\alpha = 1$ (least privacy protection) corresponds to training on the true labels. Choosing $\alpha = 0$ (most privacy protection) corresponds to a severe form of knowledge distillation in which the model is trained to predict the average label that would be predicted by a model trained without that sample in the training data, **and with hard labels**. Informally, α can be thought of as the fraction of information gained from the sample, from 0 (true label not used at all) to 1 (trained on true label). The value of α can be tuned to **each sample** so that more protection can be provided to samples that are more exposed. This technique can be used in conjunction with any other privacy preservation method, perhaps only on points which are still exposed after that method is applied. However, that is left to future work,

and currently we have only investigated optimizing α globally (same value for all samples).

We have found a modified version of interpolating **to perform better empirically**. The raw output of the neural network models that we train are the input values to the softmax activation function that would give a multi-class predicted probability vector. These are often referred to as logits, although they are not given by the formula $\log\left(\frac{p}{1-p}\right)$ when there are more than two classes. Carlini et al. (2022) apply the actual logit scaling to the predicted confidence of the true class because they find (empirically) that it is better approximated by a normal distribution.[6] We have a similar finding for the raw output that we use, that the raw output is **better approximated by a normal distribution**, and find that it is **easier to attack**, and hence is also what we should use to defend. Therefore, we interpolate the raw output, and use softmax to turn that into probability vector:

$$y_{\text{target}} = \text{softmax}\left(\mu_{\hat{Z}} + \alpha\sigma^*_{\hat{Z}}\right) \tag{5.15}$$

Here, $\mu_{\hat{Z}}$ is the "leave-out average", i.e. the average raw output (aka logit) predicted by a model trained without that sample in the training data, **and with hard labels**. Since the predicted probabilities are invariant to a translation of the logits, we center the predicted logits for each sample to have mean zero, which we find empirically to low the standard deviation. The matrix $\sigma^*_{\hat{Z}}$ is a *signed* standard deviation vector for each sample, which is the standard deviation, but with the value corresponding to the true label left positive and the other values made negative. This way, training with $\alpha = 1$ corresponds to targets that are one standard deviation of where we expect them to be predicted by a model trained without that sample in the training data, **and with hard labels**. To illustrate this method of soft labeling, consider the hypothetical example of a 3-class classification problem illustrated in Figure 5.1. The histograms show the logits predicted for a particular sample from 25 models with it IN the training set (orange) or OUT of the training set (blue). Gaussians fit to those distributions are also shown. For $\alpha = 0$, we use as target labels $y_{\text{target}} = \text{softmax}\left(\mu_{\hat{Z}}\right)$, where $\mu_{\hat{Z}}$ is the "leave-out average" indicated by the black vertical dashed lines. For $\alpha = 1$, we use as target labels $y_{\text{target}} = \text{softmax}\left(\mu_{\hat{Z}} + \sigma^*_{\hat{Z}}\right)$, with $\mu_{\hat{Z}} + \sigma^*_{\hat{Z}}$ indicated by the purple vertical dashed lines.

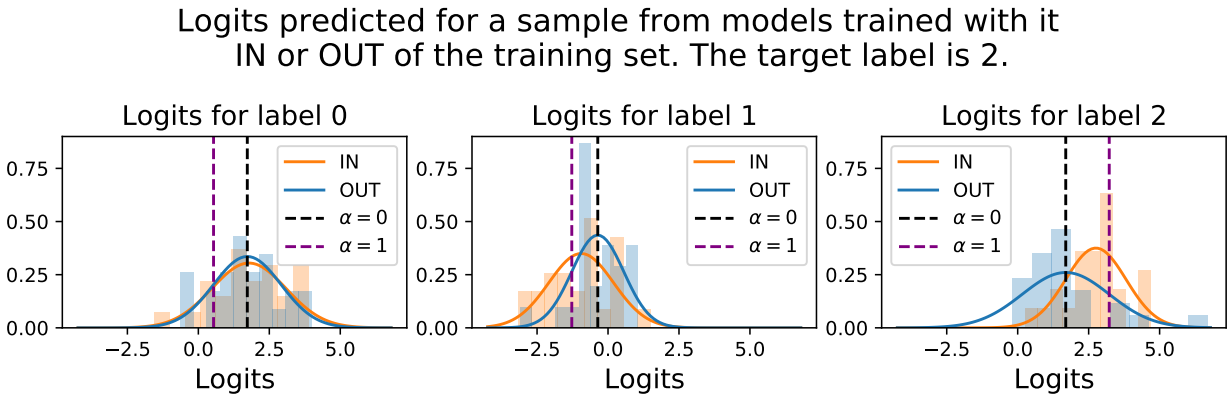

Figure 5.1: Hypothetical example of the logits predicted for a sample from models with it IN or OUT of the training set. The vertical lines indicate the target logits for $\alpha = 0$ (black) and $\alpha = 1$ (purple)

It might seem like with this method, using $\alpha = 0$ would be almost perfectly private because we would expect it to predict the same on any sample as it would had that sample not been in the training set. This intuition is wrong, most likely for a few reasons. Most notably, as already emphasized multiple times, $\mu_{\tilde{Z}}$ is the average output predicted by a model trained without that sample in the training data, that was trained with hard labels (or whatever the original labels were). If we train with different labels, no matter what the reason for the difference, we should expect the "leave-out average" to change, because the labels affect the training. Therefore, if we train with our optimally soft labels, and compute this "leave-out average" again, it may have (most likely will have) shifted. How much it shifts by most likely depends on alpha, and is different for each sample, but we have not yet investigated that. Ideally, if we continued to update the optimally soft labels recursively, we would hope that they would converge to values that we could use for even more privacy, while hopefully maintaining utility, but this will also be left to future work. It must be emphasized that whatever the defense procedure is, even if it includes multiple iterations to compute the labels recursively, our framework is only valid if we give the LTU attacker the entire procedure in an overall learning algorithm. Any attempt for the defender to have the last move will only result in an overly optimistic estimate of the privacy achieved.

5.4 Data and Experimental Setup

For our experiments, we used the CIFAR-10 [23] dataset mentioned in the previous chapter. It contains 60,000 images that are 32 pixels by 32 pixels , but we preprocessed

the images through a pretrained network and used the second to last layer as the 1,280 features for our dataset (instead of the images). We used Efficient-netv2 [46] trained on Imagenet21k with fine-tuning on CIFAR-100. Privacy preservation is more challenging on smaller datasets, so we only used 400 samples for the training set and 400 samples for the test set. We hypothesized that our method would have best performance when some samples had higher per-example hardness[6], so we injected controlled noise into the labels by intentionally flipping 20% of the labels, 80 in the training set and 80 in the test set. We did this in pairs, so that we could maintain exactly balanced classes, by taking a pair of samples with different labels and exchanging their labels.

For our baseline model trainer, we used a fully connected neural network with 3 hidden layers of sizes: 512, 256, and 128. We used code from the TensorFlow Privacy tutorial, which we modified. We did not do very extensive hyper-parameter tuning, but did attempt to find values that worked well for the differentially private model before adding our defense technique. We used ReLU activation functions and L2 regularization of 0.01 in all layers, and dropout after the second hidden layer at rate 0.2. We did not do any additional tuning once we added our defense technique. We used a noise multiplier of 0.5, an L2 norm clip of 3.0, a learning rate of 0.05, a batch size of 20, and 20 micro-batches. We only trained for six epochs and used the regularization noted to attempt to make a very private model at reasonable utility before applying our method.

For our LTU framework, the Defender used the baseline model, except training for 12 epochs, to compute the $\mu_{\hat{Z}}$ and $\sigma^*_{\hat{Z}}$ mentioned earlier. To do that, the Defender splits the training data in half, and trains models on each half (200 samples). The Defender does that 25 times to compute the average and standard deviation. Then, the Defender uses those, with a value of α, to train the final model. To report significant results, we actually have the Defender train 50 models and report median values. The Attacker always performs the exact same procedure as the Defender, except that the Attacker first randomly splits the combined training and test set (800 samples), then performs the procedure on each half (400 samples each). Note that the procedure includes doing what the Defender did, so for each half, the Attacker divides it again in half 25 times to compute averages and standard deviations for that half, and uses those to train the model on that half. This procedure took approximately

two days using two GPUs, for each value of alpha, for each dataset.

5.5 Results

The baseline model that we chose was already very well defended, as can be seen in Figure 5.2, making it very hard to improve upon its privacy. The histograms show the median predicted probability of membership for each sample, over 50 Defender models. For the baseline model, the median predicted probability of membership is 52.3%, where 50% represents random guessing. Training with optimally soft labels that are $\alpha = 1$ standard deviation, softmax $\left(\mu_{\hat{Z}} + \sigma^*_{\hat{Z}}\right)$, appears to slightly improve this to 52%, but the difference is not statistically significant (p-value = 0.36 with Mood's median test), nor is the difference for $\alpha = 0.5$ (p-value = 0.22). However, training optimally soft labels that are $\alpha = 0.25$ does have a statistically significant improvement (p-value $= 1 \times 10^{-6}$), as does using the "leave-out average", softmax $(\mu_{\hat{Z}})$, (p-value $= 9 \times 10^{-13}$).

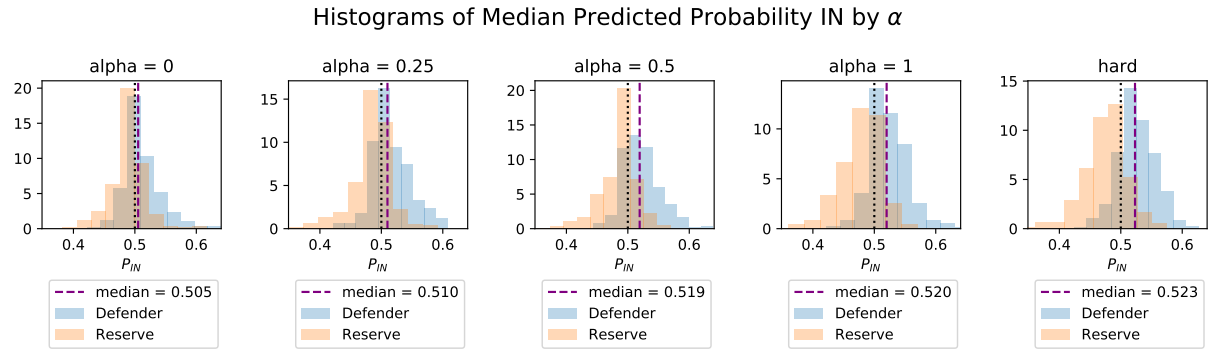

Figure 5.2: Histograms of median P_{IN}, the predicted probability of membership in the defender set, for CIFAR10 samples that were in the Defender or Reserved sets, for four levels of label softening. For non-private models, P_{IN} for the Defender set will be shifted to the right.

Of course, these are only the predicted probabilities of membership. It is vital to check that these correspond to the actual proportions of membership, or else the predictions may not be useful. The graphs in Figure 5.3 show the correspondence between the predicted probabilities and the actual proportions, with the error bars indicating 3 standard errors. One standard error was approximately 0.025, so again we see that there's no statistically significant difference between training with the original (hard) labels and training with op-

timally soft labels that are $\alpha = 0.5$ or 1. However, the maximum proportion when trained with optimally soft labels that were $\alpha = 0$ or 0.25 was 3 standard errors lower than the maximum proportion when trained with original (hard) labels.

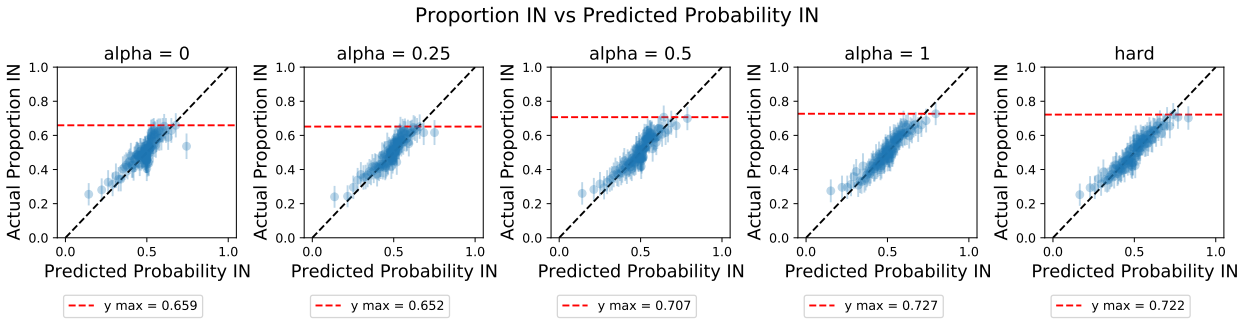

Figure 5.3: Scatter plots of the actual proportion of membership vs the predicted probability of membership, averaged over 100 groups of nearby predictions, which match well (near the 45 degree line). Using $\alpha = 0$ lowered the maximum by 6.3pp, corresponding to more privacy for the most exposed samples.

We measured the AUROC for each of these: 0 sd = 0.58, 0.25 sd = 0.59, 0.5 sd = 0.60, 1sd = 0.61, and hard = 0.62. However, Carlini et al. (2022) urge that what is more important for privacy than global metrics like AUROC is to compare the true positive rate (TPR) at values of low false positive rate (FPR). We find that $\alpha = 0$ lowers the true positive rate to very near random guessing at low false positive rates.

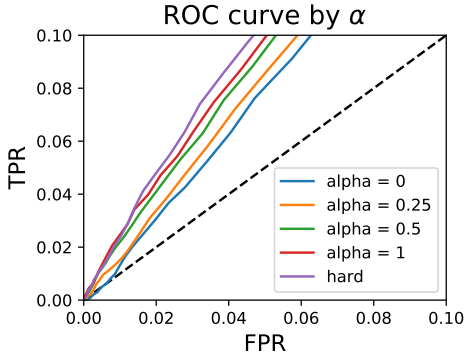

Figure 5.4: Receiver Operating Characteric at values of low false positive rate

Of course, any analysis of privacy preservation should be juxtaposed with the impact to utility. This is often framed as a trade-off between privacy and utility, since the methods

used to preserve privacy often come with some cost to utility. However, as we saw in Chapter 3 with DP WGAN, it is possible for privacy preservation methods to improve utility. That is indeed what we find here, as shown in Figures 5.5 and 5.6, using optimally soft labels increased the validation and test accuracies. The best choice for utility was $\alpha = 1$ according to the validation set. This value gave a median test error (over 50 models) of 23.4%, which is 16% (4.4 pp) lower than the median test error when training on the hard labels.

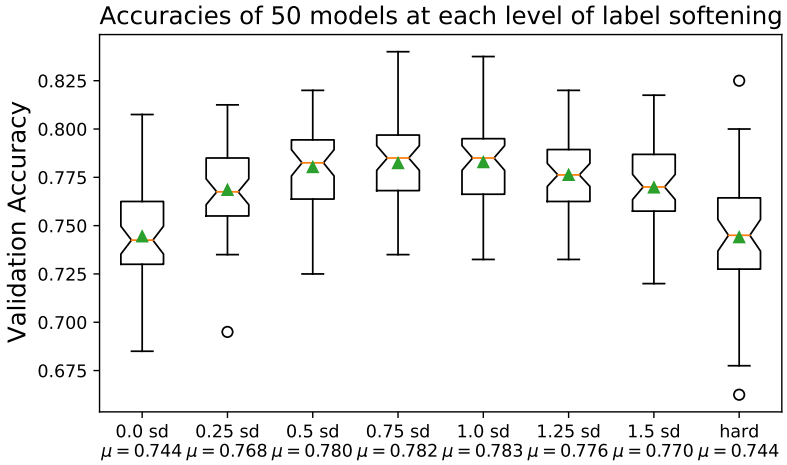

Figure 5.5: Boxplots of the validation accuracy for 50 models trained on different levels of optimally soft labeling

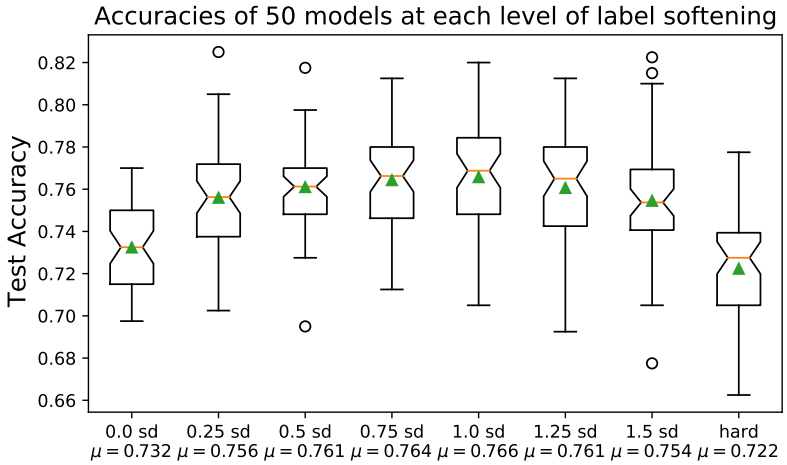

Figure 5.6: Boxplots of the test accuracy for 50 models trained on different levels of optimally soft labeling

We compared the utility of our optimally soft labels to the traditional method of soft

labeling, in which the target label is $(1 - \alpha)$ times the onehot true label plus α times a uniform probability vector. In order to determine the appropriate range of α to use for this method in our comparison, we first examined the distribution of target probabilities for the labeled class that resulted in our optimally soft labeling method. On the left hand side of Figure 5.7, we see that for $\alpha = 0$, optimally soft labeling results in a median target probability of 0.46 for the labeled class (top left). However, if we examine only the samples that had their label flipped (bottom left), we see their median is only 0.026, which makes sense since for $\alpha = 0$ the target is the "leave-out average" probability vector and we expect the (incorrect) labeled class to have a very low predicted probability. Since we know the uncorrupted labels (the true classes), we can also examine their medians (right hand side), which are approximate 0.54.

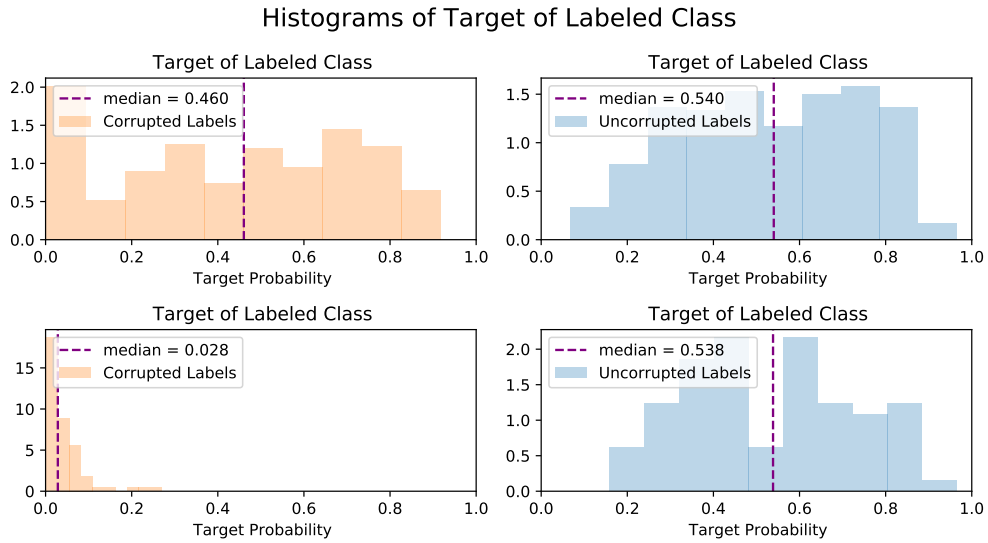

Figure 5.7: Histograms of target probabilities for the labeled class when $\alpha = 0$, in the entire training set (top row), and for just the samples that had their label flipped (bottom row)

We can similarly examine the $\alpha = 1$ optimally soft labeling, which is in Figure 5.8. We see that by moving one standard deviation in the direction of the onehot label, the median of the target probabilities increases to 0.9, which should greatly help learning from the samples that are labeled correctly (did not have their labels flipped). By focusing on the samples that had their labels flipped (bottom row), we see that the median of the target probabilities for the uncorrupted labels (the true classes) only decreases by 0.03 to 0.508 (bottom right).

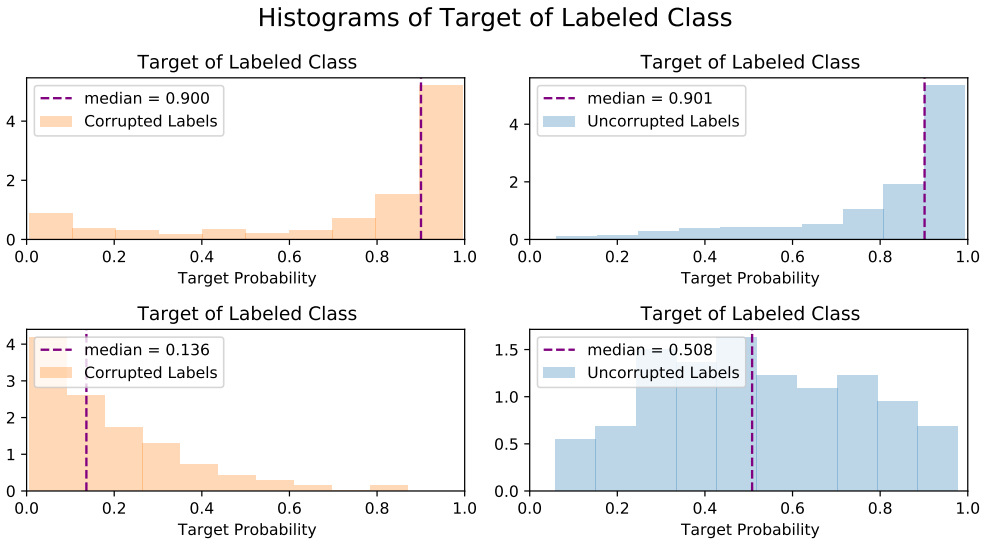

Figure 5.8: Histograms of target probabilities for the labeled class when $\alpha = 1$, in the entire training set (top row), and for just the samples that had their label flipped (bottom row)

From the preceding analysis, it makes sense to use α in the range of $[0.5, 0.9]$ for the traditional soft labeling to perform our comparison. We find that this form of soft labeling does not improve performance, as can be seen in Figure 5.9

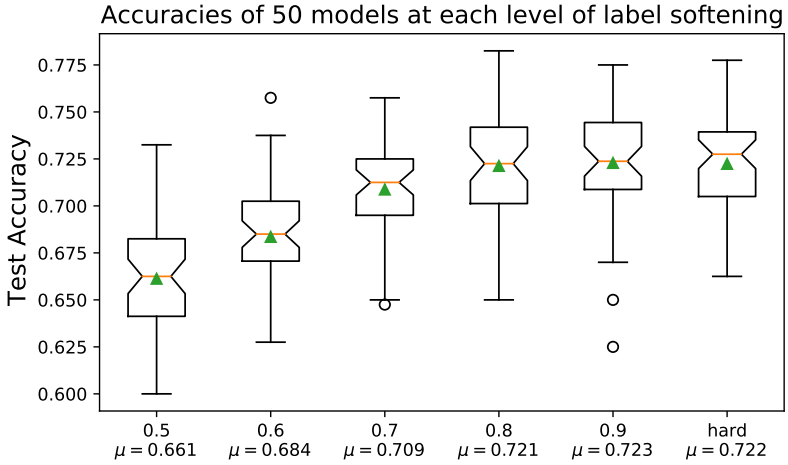

Figure 5.9: Boxplots of the test accuracy for 50 models trained on different levels of optimally soft labeling

Thus, this shows that traditional soft labeling, which interpolates between the true label and a uniform probability vector, does not improve test accuracy, but our method, which interpolates between the true label and the "leave-out average" predicted probability,

does. Since we know the uncorrupted labels (true classes) of the training data, we can use those to analyze the optimally soft labels to gain an understanding of why they improve generalization. As seen in Figure 5.10, the argmax of the "leave-out average" matches the corrupted label for approximately 70% of the samples in the training data, but it matches the uncorrupted label for approximately 90% of the samples. As we increase α, we make the target labels closer to the corrupted labels, but at first this also increases the proportion of the target labels whose argmax matches the uncorrupted label, reaching a maximum of 96% matching at $\alpha = 0.7$. This value of α is optimal for matching the uncorrupted label (true class) in the training data. If we increase α further, we converge toward the argmax completely matching the corrupted labels, and hence only matching 80% of the uncorrupted labels (true classes).

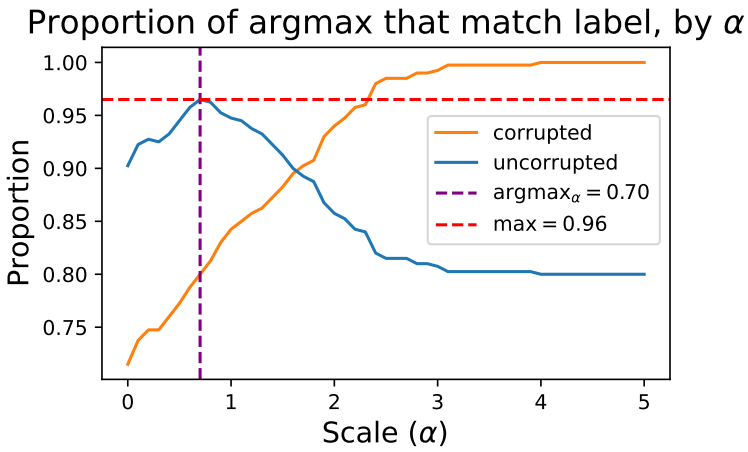

Figure 5.10: Plot of the proportion of the samples whose target label has an argmax that matches the corrupted/uncorrupted label. The optimal value is $\alpha = 0.7$ for matching the uncorrupted labels

Of course, during training we perform gradient descent to minimize the cross-entropy loss function, rather than trying to increase the accuracy (which would not be differentiable). We can also analyze the cross-entropy between the optimally soft labels at different values of α and the corrupted/uncorrupted labels. As shown in Figure 5.11, as we increase α the target labels converge toward the corrupt labels so that by $\alpha = 5$ the cross-entropy between the two is zero. However, the minimum of the cross-entropy with the uncorrupted labels (true classes) occurs at $\alpha = 1.2$.

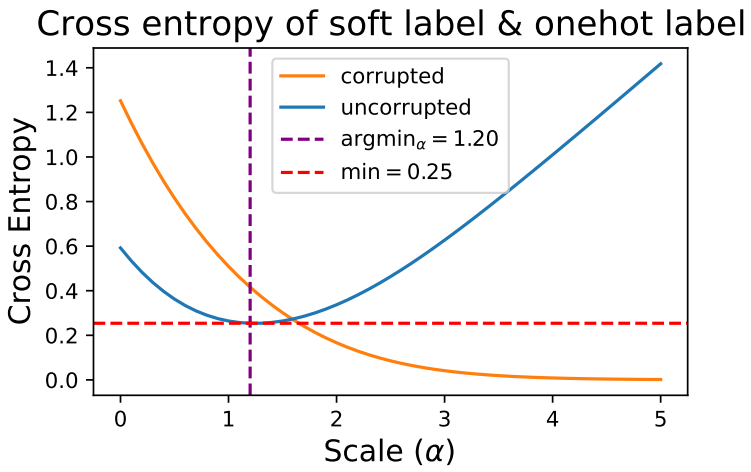

Figure 5.11: Plot of the proportion of the samples whose target label has an argmax that matches the corrupted/uncorrupted label. The optimal value is $\alpha = 0.7$ for matching the uncorrupted labels

From both of these analyses, we should expect the best generalization from training with optimally soft labels with $\alpha \approx 1$. Of course, in a realistic scenario we would not already know the uncorrupted labels, so we would need to use validation to choose the optimal value of α, as shown in Figure 5.5. Indeed, we found that the value which gave the best validation accuracy was $\alpha = 1$. For this dataset, we found that $\alpha = 0$, 0.25, 0.5, and 1, created a Pareto optimal frontier for choosing the desired balance between privacy, given by Equation (4.9), and utility (test accuracy), but that all four values were better than hard labels in terms of both privacy and utility, as seen in Figure 5.12. To emphasize again, we performed this optimization globally (same value of α for all samples), but we hypothesize that better utility will come from using validation to compute the optimal value of α individually for each sample. We plan to explore that in future work.

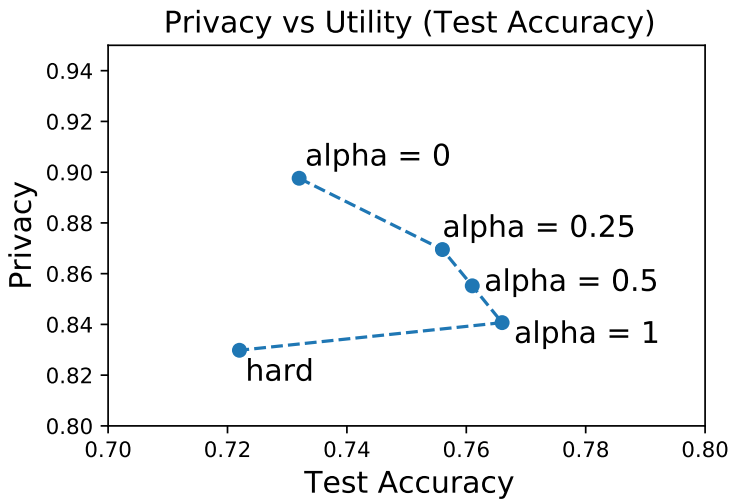

Figure 5.12: Plot of average privacy vs average utility for hard labels and 4 levels of softening, averaged over 50 models per level. The levels of softening create a Pareto optimal frontier that are better than hard labels in both privacy and utility

5.6 Conclusion

In this section, we formally analyzed the problem of membership inference in the LTU Attacker framework. We showed the connection between our framework and differential privacy, and proved that there is a risk to privacy loss for any training sample that impacts the learning. We further proved that, under certain conditions, a black-box attack is theoretically optimal in our framework. We use this attack to motivate a novel defense strategy that is widely applicable and easy to implement. We show that it can improve the privacy of an already well defended model with no loss of utility, and that it can even be used to simultaneously improve privacy and *increase* utility.

For future work, we need to perform these experiments on more datasets. We also hypothesize that the greatest benefit to privacy and utility will come from optimizing α individually on each sample, rather than globally. Furthermore, both the LTU Attacker framework and our defense strategy are applicable to any supervised learning, so we should investigate regression models as well.

CHAPTER 6
Contributions and Future Work

6.1 Contributions

In order to achieve high quality output from a differentially private algorithm, it is important to have noise tailored to the algorithm, using the strongest bound possible on the query of the data. We have improved on the existing bound for the gradient of the loss of the WGAN as much as possible, achieving a tight bound, not only for the norm of the gradient, but for every component of the gradient, each iteration. This permits the use of much less noise, as well as non-spherical noise tailored to the components of the gradient. We have also derived a bound on the norm of the gradient of the gradient penalty term of the loss function, and implemented these bounds into a differentially private WGAN-GP. We used this DPWGAN-GP to generate synthetic data for three different medical datasets, and demonstrated its quality by replicating results from published studies, as well as demonstrating that differential privacy helped mitigate mode collapse. Having a high quality synthetic data generator will have numerous uses for education and research.

We have also improved methods for evaluating privacy. We have identified and fixed issues with the nearest neighbor Adversarial Accuracy metric, so that it is now provably unbiased. The naive implementation of this new metric would be a factor of n more computationally expensive, but we make an efficient implementation that avoids this expense. We also make a correction for discrete distances. Finally, we show that averaging the two terms of this metric loses power since each term individually has an expected value of 0.5 if the datasets are from the same distribution.

As a second technique for estimating the level of privacy, we have a new framework for attacking privacy, for which we have proven theoretical results showing that it can attack any model that is overfit with a lower bound on its accuracy given by the amount of overfitting. These evaluation methods give us multiple methods to empirically interrogate the synthetic data or models created from it for any privacy loss, to provide stronger assurances of privacy preservation.

In our framework, we prove that any learning comes with risk to privacy and that black-box attacks are theoretically optimal so there is no theoretical protection from obscurity. Although the optimal attack is not exactly achievable in practice under realistic settings, a devoted adversary with enough computational resources and time could approach that attack accuracy, so in our framework we assume the worst case scenario and develop a novel defense strategy for that, which we call "optimally soft labels". Our method is applicable to any supervised learning problem that works on continuous labels. It is also straightforward to implement and we will make our code available. We demonstrate that this defense technique can improve the privacy of an already well-defended model, and that rather than come with some cost to utility, it simultaneously improves the utility.

6.2 Future Work

For future work, we can improve our differentially private WGAN by using other methods for enforcing the Lipschitz constraint, rather than using the gradient penalty, which is too sensitive to achieve a low value of the differential privacy parameter ε. Furthermore, since our bounds are computed componentwise, rather than only in norm, we should investigate the use of non-spherical Gaussian noise.

With our new defense technique of using "optimally soft labels", we need to perform experiments using more datasets and should also investigate its performance with regression models. Most promising, we plan to explore optimizing the soft label parameter α individually for each sample, which we expect to achieve better utility and privacy. In fact, this technique should also be investigated for its improvement to utility in settings in which privacy is not a concern.

Finally, these two lines of research can also be merged in two ways. First, we could explore using our attack technique on a trained discriminator to attack the membership of the training data. This measurement of privacy can be compared to the estimate of privacy from our unbiased nearest neighbor accuracy metric. Second, we could explore generating synthetic datasets to increase the size of training data, with using our "optimally soft labels" to improve the utility of the model trained on this extended training set.

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Rajsavi S Anand, Paul Stey, Sukrit Jain, Dustin R Biron, Harikrishna Bhatt, Kristina Monteiro, Edward Feller, Megan L Ranney, Indra Neil Sarkar, and Elizabeth S Chen. Predicting mortality in diabetic ICU patients using machine learning and severity indices. *AMIA Summits on Translational Science Proceedings*, 2018:310, 2018.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[4] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 609–618, New York, NY, USA, 2008. Association for Computing Machinery.

[5] M. Burhan, R. A. Rehman, B. Khan, and B. S. Kim. Iot elements, layered architectures and security issues: A comprehensive survey. *Sensors*, 2018.

[6] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1519–1519, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

[7] Leo Anthony G Celi, Robin J Tang, Mauricio C Villarroel, Guido A Davidzon, William T Lester, and Henry C Chueh. A clinical database-driven approach to decision support: Predicting mortality among patients with acute kidney injury. *Journal of Healthcare Engineering*, 2(1):97–110, 2011.

[8] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. In

Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Domo, Inc. Data never sleeps 7.0. Retrieved from https://www.domo.com/learn/data-never-sleeps-7/ on May 18, 2020.

[11] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[13] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1), Sep. 2010.

[14] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[15] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, 2010.

[16] Marta PB Fernandes, Claudia F Silva, Susana M Vieira, and Joäo MC Sousa. Multimodeling for the prediction of patient readmissions in intensive care units. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1837–1842. IEEE, 2014.

[17] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), June 2010.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[19] Patrick Grother. Nist special database 19. nist handprinted forms and characters database. 1970.

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs, 2017. arXiv preprint arXiv:1704.00028.

[21] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[22] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2008.

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[25] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1656–1665, New York, NY, USA, 2018. Association for Computing Machinery.

[26] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: a unifying framework for privacy definitions. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 889–900, 2013.

[27] Alexis C. Madrigal. Reading the privacy policies you encounter in a year would take 76 work days. Retrieved from https://www.theatlantic.com/technology/archive/2012/03/reading-the-privacy-policies-you-encounter-in-a-year-would-take-76-work-days/253851/ on May 18, 2020.

[28] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, 2007.

[29] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[30] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[31] Mitchell Noordyke. Us state comprehensive privacy law comparison. Retrieved from https://iapp.org/news/a/us-state-comprehensive-privacy-law-comparison/ on May 18, 2020.

[32] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016.

[33] Joseph Pedersen, Rafael Muñoz-Gómez, Jiangnan Huang, Haozhe Sun, Wei-Wei Tu, and Isabelle Guyon. Ltu attacker for membership inference. arXiv, 2022.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] Pew Research Center. Mobile fact sheet. Retrieved from https://www.pewresearch.org/internet/fact-sheet/mobile/ on May 18, 2020.

[36] David Reinsel, John Gantz, and John Rydning. The digitization of the worldfrom edge to core, 2018.

[37] Maria Rigaki and Sebastián García. A survey of privacy attacks in machine learning. *ArXiv*, abs/2007.07646, 2020.

[38] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019.

[39] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Ryan Singel. Netflix cancels recommendation contest after privacy lawsuit. Retrieved from https://www.wired.com/2010/03/netflix-cancels-contest/ on May 18, 2020.

[42] Aaron Smith. Half of online americans don't know what a privacy policy is. Retrieved from https://www.pewresearch.org/fact-tank/2014/12/04/half-of-americans-dont-know-what-a-privacy-policy-is/ on May 18, 2020.

[43] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[44] Statista Research Department. Number of wearable device users in the u.s. 2014-2022. Retrieved from https://www.statista.com/statistics/543070/number-of-wearable-users-in-the-us/ on May 18, 2020.

[45] Latanya Sweeney. Us department of homeland security. Retrieved from https://www.dhs.gov/xlibrary/assets/privacy/privacy_advcom_06-2005_testimony_sweeney.pdf/ on May 18, 2020.

[46] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021.

[47] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, pages 1–1, 2019.

[48] Christo Wilson. If you use a mac or an android, e-commerce sites may be charging you more. Retrieved from https://www.washingtonpost.com/posteverything/wp/2014/11/03/if-you-use-a-mac-or-an-android-e-commerce-sites-may-be-charging-you-more/ on May 18, 2020.

[49] Greg Wright. Despite legal risks, companies still use social media to screen employees. Retrieved from https://www.shrm.org/ ResourcesAndTools/hr-topics/technology/Pages/Social-Media-to-Screen-Employees.aspx/ on May 18, 2020.

[50] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[51] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network, 2018. arXiv preprint arXiv:1802.06739.

[52] Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[53] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P. Bennett. Assessing privacy and quality of synthetic health data. In *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*, AIDR '19, New York, NY, USA, 2019. Association for Computing Machinery.

[54] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. Privacy Preserving Synthetic Health Data. In *ESANN 2019 - European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, April 2019.

[55] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.