Application-driven Privacy-preserving Data Publishing with Correlated Attributes

Aria Rezaei*, Chaowei Xiao†, Jie Gao‡, Bo Li°, Sirajum Munir◇
*Stony Brook University, arezaei@cs.stonybrook.edu. †NVIDIA, chaoweix@nvidia.com. ‡Rutgers University, jg1555@rutgers.edu. °UIUC, lbo@illinois.edu. ◇Bosch Research, sirajum.munir@us.bosch.com

Abstract

Recent advances in computing have allowed for the possibility to collect large amounts of data on personal activities and private living spaces. To address the privacy concerns of users in this environment, we propose a novel framework called PR-GAN that offers privacy-preserving mechanism using generative adversarial networks. Given a target application, PR-GAN automatically modifies the data to hide sensitive attributes – which may be hidden and can be inferred by machine learning algorithms – while preserving the data utility in the target application. Unlike prior works, the public's possible knowledge of the correlation between the target application and sensitive attributes is built into our modeling. We formulate our problem as an optimization problem, show that an optimal solution exists and use generative adversarial networks (GAN) to create perturbations. We further show that our method provides privacy guarantees under the Pufferfish framework, an elegant generalization of the differential privacy that allows for the modeling of prior knowledge on data and correlations. Through experiments, we show that our method outperforms conventional methods in effectively hiding the sensitive attributes while guaranteeing high performance in the target application, for both *property inference* and *training* purposes. Finally, we demonstrate through further experiments that once our model learns a privacy-preserving task, such as hiding subjects' identity, on a group of individuals, it can perform the same task on a separate group with minimal performance drops.

1 Introduction

Recent years have witnessed an explosive growth in ubiquitous sensing and data-driven AI innovations in every aspect of our lives. While we celebrate the convenience brought to us by these technologies, the collected data can often be personal information and contain attributes that could be ex-

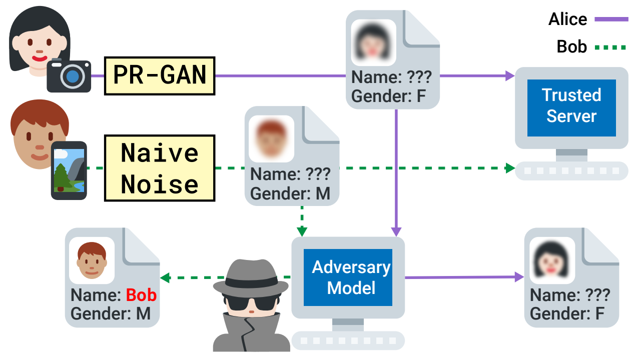

Figure 1: Alice and Bob send data from their personal devices to a trusted server, masking their identity while making their gender visible. An adversary cannot decipher Alice's identity as she has used our method (PR-GAN) while he can do so for Bob, who use a naive noise generator.

tremely sensitive. For example, using background noise, one can infer the number of people talking, their identity or the type of environment [27, 28, 44]. Data from electricity meters can reveal sensitive information about the household [45]. GPS trajectories can reveal social ties and are very unique for each person [8, 11, 41]. Using WiFi scans, one can discover occupancy patterns of private households and localize individuals [19, 40]. This means that it is crucial that we find the correct balance between using data to improve the quality of our lives while making sure sensitive information is effectively protected in IoT systems.

There are numerous metrics for data privacy, such as k-anonymity [39], differential privacy (DP) [9], and its extension that supports correlations and prior knowledge on the data, namely Pufferfish privacy [18] among others. By introducing noises either in the original data (for data publication) or the query output (for answering interactive queries on private data), these criteria provide guarantees that an attacker cannot infer substantially more with or without access to a particular record. Most conventional methods are designed for structured datasets, like patient's medical records, with an *a priori* knowledge of which one of the features in the data are deemed sensitive. In this work, however, we focus on *unstructured* data in sensor signals, such as images, binary vectors and other raw physical readings, where sensitive features are patterns hiding in the data that could be learned by an adversary.

International Conference on Embedded Wireless Systems and Networks (EWSN) 2021
17–19 February, Delft, The Netherlands © 2021 Copyright is held by the authors.
Permission is granted for indexing in the ACM Digital Library
ISBN: 978-0-9949886-5-2

Our focus in this work is to develop a novel framework that supports *private data publication*, which is suited for distributed sensing and IoT settings. This is different from another common setting where the data is kept in a private database and noise is added to the results of a query to ensure privacy. A major advantage of our approach is that after publication, the data could be used for a wide variety of applications. Note also that if an entity publishes *only* the target attributes, while ensuring a high degree of privacy, the published information could not have any further use; this is why this naive approach is usually not considered.

Consider a use case in a smart building application, where the WiFi signal strength of occupants is routinely recorded to monitor the integrity of the local network and detect system failures. The occupants have no problem contributing their data for such purposes as long as they don't feel that their identities might be revealed or that their movements are tracked. To ensure this, the administrators propose a compromise: the collected data might be accurate enough to reveal which *floor* a person is on, but will not reveal which specific room they are in at any moment. Note that here, inferring which floor a person is on is not the goal of publishing the data; it just ensures an acceptable level of accuracy in the data. Note also that an adversary has partial information about the target and sensitive attributes: *if a person is on the 3rd floor, she could only be in one of the rooms on the 3rd floor*. This information comes by definition and requires no extra effort to retrieve it. We explicitly model this correlation, and by doing so, we are in effect defending against a stronger adversary compared to the prior work [5, 14, 35].

Since the organization that keeps the data may not be trusted to properly address the privacy concerns, a more preferred approach is ensuring privacy from the source. Figure 1 shows such a case where Alice and Bob want to share their photos with an external server and they use perturbations to conceal their identity. An adversary, using an auxiliary dataset has trained an identity-revealing model. She then intercepts the messages midway and manages to reveal the identity of Bob using this model. Our goal is that if Alice's photo, perturbed by our method, is intercepted, the adversary won't succeed in revealing her identity.

Problem Definition and Our Contribution. We assume that the sensor data x is used to learn a *target attribute* $f_y(\mathrm{x})$. Meanwhile, there are a number of *sensitive attributes* $f_z(\mathrm{x})$ that are of concern in the sensor data. The function f_y describes data utility and function f_z characterizes the privacy concern. We assume that both the target attributes and the sensitive attributes are deeply embedded in the physical sensor signals. Even with anonymity, both attributes can be learned with supervised learning. Therefore, the data collected from the individual sensors could easily reveal the sensitive attributes about the users who contribute the data.

We plan to address the problem by developing an algorithm A that perturbs the raw sensor reading x to $\mathrm{x}' = A(\mathrm{x})$ such that the utility of the system is high, as defined by the success of learning the target attribute, i.e., $f_y(\mathrm{x}') \approx f_y(\mathrm{x})$, and the privacy of the sensitive attributes is protected, as measured by the significantly reduced performance of $f_z(\mathrm{x}')$ accurately predicting $f_z(\mathrm{x})$.

We formulate the problem as an optimization problem: we aim to minimize the performance loss of f_y when perturbed data is used, and maximize the uncertainty among the predicted values of f_z while adhering to the constraints of publicly known relation between f_y and f_z. We show the existence of an optimal perturbation function and analyze the privacy guarantees provided by our mechanism.

To produce perturbations tailored to each data point, we use Generative Adversarial Networks (GAN) [13]. A standard GAN is composed of two neural networks contesting with each other in a zero-sum game. Two models are alternatively trained: a generative model $\mathcal{G}$ tries to produce *fake* data records. The discriminator $\mathcal{D}$ tries to predict whether an input data is *fake* (produced by $\mathcal{G}$) as accurately as possible, while $\mathcal{G}$ tries to maximize the probability of $\mathcal{D}$ making a mistake by producing artificial data that is indistinguishable from the real data. The GAN structure ensures that the resulting data looks like the original data. In our setting, we use two classifiers as approximations of f_y and f_z.

Through rigorous experiments, we show compelling evidence of the superior performance of our perturbations, compared to baseline approaches. Our key contributions are listed below:

1. We are, to the best of our knowledge, the first to explicitly model the correlation between target and sensitive attributes, which reflects real-world scenarios more closely. By doing so, we are assuming a stronger adversary, compared to prior work, who utilizes this information to mount a more successful attack.
2. Our models are light-weight and do not require the presence of the whole dataset to generate perturbations. Once trained on a portion of data, the noise generator can be deployed on mobile devices. In Section 6.3 we compare the computational cost of our models with state-of-the-art models designed specifically to run on mobile devices.
3. In Section 4.2, we establish a connection between maximizing an adversary's uncertainty about the sensitive labels and providing guarantees based on Pufferfish privacy, an extension of DP. The reason that this is our framework of choice is that it allows us to model correlations and prior knowledge on the data.
4. Through experiments, we show that once our model is trained to hide identity-revealing features from depth images of a group of individuals, it can perform the same task on a *completely different* group of individuals with minimal performance loss. We discuss the benefits of this in Section 6.6.
5. Unlike prior work, we test the utility of the perturbed data on a *training task* as well, showing that a well-performing classifier can be trained on our perturbed data. In Section 6.7 we discuss this in more depth.

Our framework is flexible and any model that supports gradient updates could be used instead of the ones used here. Additionally, unlike prior work, we don't transform the data into an entirely different feature space to protect privacy [7, 24].

2 Related Work

Privacy in Learning Algorithms A large body of work have focused on manipulating the existing training and inference algorithms to protect privacy of training data. For example, a differentially private training algorithm for deep neural networks in which noise is added to the gradient during each iteration [1, 34, 37], and a "teacher-student" model using aggregated information instead of the original signals [33]. It is also proposed to train a classifier in the cloud, with multiple users uploading their perturbed data to a central node without ever revealing their original private data [20, 25].

Privacy-Preserving Data Publishing A different approach to preserve privacy is making sure sensitive elements of the data is removed before publishing it, often called privacy-preserving data publishing (PPDP) [6]. A main line of work focuses on transforming numerical data into a secondary feature space, such that certain statistical properties are preserved and data mining tasks can be done with minimal performance loss [7, 24, 26]. These methods guarantee data utility only on this secondary space. This means that a classifier trained on the perturbed data is not guaranteed to perform well on original data. This can be troublesome in a scenario where a classification model is trained on public, perturbed data and is going to be deployed on users' personal devices to perform a task locally on non-perturbed private user data. Our perturbed data can be used in conjunction with original data in specified target applications. In addition, some of the methods in this category rely on expensive computations which renders them infeasible on large datasets.

Adversarial Learning GANs have been successfully used to produce *adversarial examples* that can fool a classifier to predict wrong classes [43]. Some have formulated the problem of privacy protection as producing adversarial examples for an identity-revealing classifier [4, 16]. However, we demonstrate through experiments that the absence of a predictor function, f_y, that maintains the utility of the data, leads to a weaker utility guarantee for the published data.

Adversarial Privacy There has been a recent surge in the use of adversarial learning for privacy protection. In this context, privacy is defined in various ways. Some have employed proxies for privacy such as reducing the mutual information between the data and sensitive attributes that is discolsed by publishing a perturbed dataset [3], or minimizing the distance between data records belonging to different *sensitive* categories so as to make it harder to distinguish between them [29, 42]. The problem with employing such heuristics is that they do not directly aim to prevent an adversary from inferring sensitive information. Acknowledging this, others have tried to minimize an adversary's success in predicting sensitive information [5, 14, 35]. However, no connection between such an approach and rigorous privacy formulations have been established yet. We are, to the best of our knowledge, the first to attempt to link the inability of an adversary to predict sensitive information to a known privacy framework, namely Pufferfish. Another key difference between the prior work and ours is the assumption that the correlation between the target attribute, y, and the sensitive, z, is at least partially known. As we show later on, the possession of such knowledge by an adversary is very likely in real-world scenarios. This correlation is modeled directly into our formulation, which is unlike prior work where y and z are assumed to be uncorrelated [5, 35]. Some prior work rely on the availability of an *acceptable* and an *unacceptable* set of data records for public disclosure; the model then tries to learn a transformation from the latter to the former [12]. In real-world scenarios, this *a priori* knowledge of what safe-to-publish data looks like is not always possible, and our model relies solely on the two classifiers, f_y and f_z, to learn how the published results should look like. Finally, another line of work tries to create a private intermediate encoding of the data for specific neural network architectures [29]. This approach will make the use of the published dataset limited as it can only be used for neural networks with a certain architecture. In our work, we assume that the published data looks as much to the original data as possible and is ready-to-use once disclosed with the public.

3 System Overview

To demonstrate our system's capability in protecting privacy in a sensing system, we provide the following overview of a potential scenario.

Consider a smart building setting where the building managers would like to know the number of people in each room at any moment to maximize energy efficiency. To do this, they install sensing devices, such as Xbox Kinect devices, near entrances to each room to maintain an accurate count. The occupants in the building, on the other hand, might be worried about being tracked by the system as their identities might be revealed using the collected data.

Figure 2 shows the overview of a privacy preserving sensing system. There are 3 phases in total. During the *training* phase, the *manufacturer* of the noise-generating model, PR-GAN, mounts XBOX Kinect devices on top of entrance doors, similar to the setting in the target building, and asks volunteers to move under the device, producing instances with them facing "inside" or "outside". Here, the sensitive attribute is the subjects' identity while the target attribute is their facing direction. Then, the two classifiers on the target and sensitive attributes, together with the noise-generating module, PR-GAN is trained. Once training is over, during the *deployment* phase, PR-GAN modules can be installed on individual sensors by either the manufacturers or the building managers. As we mentioned before, PR-GAN is lightweight and computationally inexpensive, hence suitable to run on mobile devices (more on this in Section 6.3).

Finally, in the *utilization* phase, the collected imagery is first anonymized in real-time using the noise generator, and the noisy images are then fed to the target classifier for privacy-preserving classification.

Threat Model We make the following assumptions about the adversary: (1) the adversary might get access to the noisy data, either in the event of a security breach or via other means. (2) The adversary could also possess a classifier on sensitive attributes by training a model on an auxiliary dataset. (3) Since we intend to publish the target labels, it is safe to assume that the adversary has access to them. (4) Partial information about the correlation between target and sensitive attributes is known by public, and hence accessible

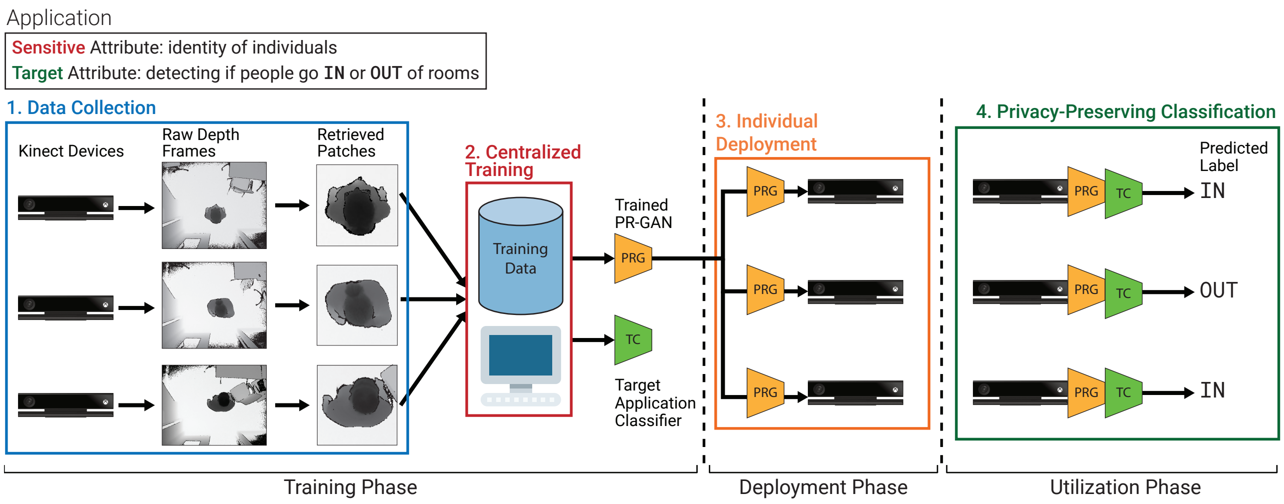

Figure 2: System overview for a smart building setting with specific target and sensitive attributes. In the *training* phase, the manufacturer collects the data, labels it, and trains the noise generator (PRG) and the target classifier (TC). Then, in the *deployment* phase, PRG modules are installed on individual sensors (Kinect devices). Finally, during the *utilization* phase, images are first perturbed using PRG and then classified in a privacy-preserving manner with TC. More details in Section 3.

by the adversary.

4 Theory

4.1 Problem Definition and Optimality

Suppose that we have a dataset X with n entries, where each entry is a vector $\mathrm{x}_i \in \mathbb{R}^N$ coming from an unknown distribution $\mathcal{P}_{\text{data}}$. Additionally, each entry is accompanied with a *target* label, y_i, and a *sensitive* label, z_i, where $y_i \in \mathcal{Y}$ and $z_i \in \mathcal{Z}$ are categorical features.

It is possible for an adversary, or the public, to have at least partial information about the relation between y and z. We model this assumption by assuming that a conditional distribution of z for a particular value of y, denoted by θ_y where $\theta_y(z) = P(z|y)$, is known for a subset of $\mathcal{Z}_y \subseteq \mathcal{Z}$. The set of all such distributions of θ_y is denoted by Θ. Note that $\mathcal{Z}_y$ can be empty, representing the case where no prior knowledge is available. As we will see later on in many scenarios, simply based on contextual definitions of y and z, it is possible for a partial knowledge of $P(z|y)$ to be held by the public and we should expect that an attacker might exploit it to mount a more successful attack.

For instance, suppose that a dataset consists of images of 20 subjects, 10 of them male and the other 10 female. Let the identity of the subjects be z and their gender y. Since we intend to publish y, we should assume that once the data is published the gender of each subject is known to all (e.g.: $y = \text{male}$). Knowing this, an adversary can cross off the 10 female subjects as potential z. Since each identity can be mapped only to one gender (for the sake of simplicity), we can assume that two disjoint sets, $\mathcal{Z}_{\text{male}}$ and $\mathcal{Z}_{\text{female}}$, comprise all identities. Then, the prior knowledge about the labels would be $\theta_{\text{male}}(z) = 0\,(z \in \mathcal{Z}_{\text{female}})$ and vice versa.

Finally, we have two functions $f_y\colon \mathbb{R}^N \mapsto \mathcal{Y}$ and $f_z\colon \mathbb{R}^N \mapsto \mathcal{Z}$, that predict y and z respectively. The goal is to find a transition function, $g\colon \mathbb{R}^N \mapsto \mathbb{R}^N$, such that one could still infer the target labels from $\mathrm{x}' = g(\mathrm{x})$ but would fail to do so for sensitive labels with any certainty. In particular, we want to produce a perturbed dataset $X' = \{\mathrm{x}' = g(\mathrm{x}), \mathrm{x} \in X\}$ such that the following is minimized:

$$\begin{aligned} &\mathbb{E}_{\mathrm{x}' \in X'}\left[\ell(f_y(\mathrm{x}'), y) - H(f_z(\mathrm{x}')|y)\right], \\ &\text{s.t.}, \forall \theta_y \in \Theta, z \in \mathcal{Z}_y : \Pr\left[f_z(\mathrm{x}') = z|y\right] = \theta_y(z). \end{aligned} \tag{1}$$

In the above, $y = f_y(\mathrm{x})$, ℓ is a suitable loss function and $H(a|b)$ is the entropy of a conditioned on knowing b. The first term of the objective function keeps target labels unchanged by $g(\mathrm{x})$ while the second term introduces maximum uncertainty for predicted z labels, $f_z(\mathrm{x}')$. X' is finally released for public use. The reason for the desirability of maximum uncertainty for $P(z|y)$ is to make sure that an adversary cannot learn from the published noisy images and the target labels anything that she already does not know.

Maximum Entropy Distribution: To understand the objective function better, consider data set X_z^* which maximizes the second term in (1):

$$\begin{aligned} &X_z^* = \arg\max_{X'} \mathbb{E}_{\mathrm{x}' \sim X'}\left[H(f_z(\mathrm{x}')|y)\right] \\ &\text{s.t.}, \forall \theta_y \in \Theta, z \in \mathcal{Z}_y : \Pr\left[f_z(\mathrm{x}') = z|y\right] = \theta_y(z). \end{aligned} \tag{2}$$

The conditional distribution for z with respect to y which maximizes the above is known as the *maximum entropy distribution* (MED) for predicted z values of X_z^* given Θ:

$$\text{MED}(z|y) = \begin{cases} \theta_y(z) & \text{if } z \in \mathcal{Z}_y \\ c_y/k & \text{otherwise} \end{cases}, \tag{3}$$

where $c_y = 1 - \sum_{z \in \mathcal{Z}_y} \theta_y(z)$ and $k = |\mathcal{Z} \setminus \mathcal{Z}_y|$.

Intuitively, to prevent the attacker from learning anything she does not already know (via prior, public knowledge Θ), MED takes the form of a uniform distribution in the absence of any prior knowledge θ_y. Notice that MED depends *not* on x, but solely on each data point's target attribute y, which is assumed to be publicly known.

To understand MED better, consider the face image dataset again. Imagine that the target labels are "male" and "female" and there are only 4 female subjects: f_1, f_2, f_3 and f_4. Without any additional information, given that an anonymized image has label "female", the resulting MED will assign $1/4$ probability of that image belonging to any of the 4 female subjects. However, if the adversary knows that the image is selected at random and half of all images of a female subjects belong to f_1, then the MED will assign $1/2$ probability to f_1 and $1/6$ to the rest of the 3 female subjects.

Existence of Optimal Solution We now show the *existence* of an optimal solution to our problem in (1). Let $\mathcal{X}(y,z) = \{\mathrm{x} : f_y(\mathrm{x}) = y, f_z(\mathrm{x}) = z\}$ be a *region* in the data space where the target attribute is y and the sensitive attribute z. Take $\mathcal{A}^*$ to be the following perturbation scheme: Any $\mathrm{x} \in \mathcal{X}(y,z)$ is mapped to x' by (1) choosing a target region $\mathcal{X}(y,z')$ via probability $\mathrm{MED}(z'|y)$ over all $z' \in \mathcal{Z}$, and (2) choosing a datapoint $\mathrm{x}' \in \mathcal{X}(y,z')$ uniformly at random.

THEOREM 1. *$\mathcal{A}^*$ has the following properties:*

1. *$X^* = \mathcal{A}^*(X)$ is an optimal solution to* (1).
2. *In presence of full prior knowledge about $P(z|y)$, the probability that any datapoint is observed in X^* is equal to that of X.*

The proof is in Appendix A.

4.2 Privacy Guarantee

Now we show that the optimal solution, or an approximate solution with a small error, provides desirable privacy guarantees. To model the correlations between y and z, we use the Pufferfish framework [18], an extension of Differential Privacy (DP).

Let $\mathbb{S}$ be a set of secrets, $\mathbb{Q} \subseteq \mathbb{S} \times \mathbb{S}$ a set of secret pairs which we intend to make *indistinguishable* and $\mathbb{D}$ the set of distributions governing the generation of data.

DEFINITION 1. *A mechanism $\mathcal{A}$ is said to be (ε,δ)-Pufferfish($\mathbb{S}$,$\mathbb{Q}$,$\mathbb{D}$) private if for all $\mathrm{x} \in X$ drawn randomly from $\pi \in \mathbb{D}$, for all secret pairs $(s_i,s_j) \in \mathbb{Q}$ for which $P(s_i|\pi)$ and $P(s_j|\pi)$ are not zero, and for all perturbed data $w \in Range(\mathcal{A})$ we have:*

$$P(\mathcal{A}(\mathrm{x}) = w|s_i,\pi) \leq e^{\varepsilon} P(\mathcal{A}(\mathrm{x}) = w|s_j,\pi) + \delta. \quad (4)$$

For brevity, we denote (ε,δ)-Pufferfish($\mathbb{S}$,$\mathbb{Q}$,$\mathbb{D}$) by (ε,δ)-Pufferfish. The essential difference between DP and the Pufferfish framework is that the latter takes into account possible beliefs that an attacker might hold about the distribution of the data. This is represented in our case by the prior knowledge set Θ which consists of (partial) distributions of sensitive attributes, z, conditioned on the target attributes, y. Note that this framework promises protection only to the extent that the attacker's knowledge on the relation between y and z allows.

We now proceed by defining $\mathbb{S}$, $\mathbb{Q}$ and $\mathbb{D}$ in our case. Let $\mathbb{S}$ be the set of all possible values each z_i can take given the corresponding y_i: $\mathbb{S} = \{\langle z_i = z\rangle : z \in \mathcal{Z}, \theta_{y_i}(z) \neq 0\}$, $\mathbb{Q}$ be the set of all pairs of $z, z' \in \mathcal{Z}$ values for each z_i such that an attacker cannot rule out given a target label y_i, i.e: either $\theta_{y_i}(z)$ does not exist or $\theta_{y_i}(z) \neq 0$ (similar for z'), and $\mathbb{D}$ be distributions of data given the knowledge about the target attribute for each data point, y_i.

THEOREM 2. *Suppose that $\mathcal{A}$ is a randomized mechanism that produces perturbed data points such that the predicted probabilities of z follows a target distribution $\pi(z|y)$ within a bounded error γ:*

$$\gamma = \sup_{z \in Z, \mathrm{x}_i \in X} \left| \Pr[f_z(\mathcal{A}(\mathrm{x}_i)) = z \mid y_i] - \pi(z|y_i) \right|, \quad (5)$$

where y_i is the corresponding target attribute for each $\mathrm{x}_i \in X$. Then, $\mathcal{A}$ achieves $(0,2\gamma)$-Pufferfish privacy.

The proof comes as a direct consequence of (5): for any $\mathrm{x}_i, \mathrm{x}_j \in X$ with the same target attribute ($y_i = y_j = y$), and $z \in \mathcal{Z}$ where $\theta_y(z) \neq 0$ we have:

$$\left| \Pr[f_z(\mathcal{A}(\mathrm{x}_i)) = z \mid y] - \Pr[f_z(\mathcal{A}(\mathrm{x}_j)) = z \mid y] \right| \leq 2\gamma, \quad (6)$$

which satisfies a $(0,2\gamma)$-Pufferfish privacy. Note that $\pi(z|y)$ in Theorem 2 could be any arbitrary distribution, which entails that if one chooses MED as $\pi(z|y)$, Pufferfish guarantees are still provided. As a reminder, there are other reasons to choose MED that we reiterate here: following it ensures that (1) the performance of the target classifier does not suffer and (2) maximum *uncertainty* is introduced for an adversary as an added measure of privacy protection.

REMARK 1. *$\mathcal{A}^*$ from Theroem 1 achieves $(0,0)$-Pufferfish privacy since $\gamma = 0$.*

Of course, in real-world scenarios, one rarely has a perfect predictor for z. Instead, both those who protect privacy and the attackers have *approximations* of f_z. In Appendix B, we show that if we use approximations of f_z the error is factored into the Pufferfish privacy bound.

5 PR-GAN Design

We solve the optimization problem in (1) by using a modified GAN (Generative Adversarial Network) architecture. This section is devoted to design and implementation details.

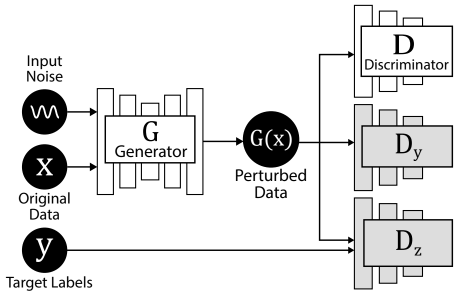

Figure 3: Architecture of proposed model. Target classifier $\mathcal{D}_y$ and sensitive classifier $\mathcal{D}_z$ (in gray) are approximations of f_y and f_z, pre-trained and fixed during training. $\mathcal{G}$ and $\mathcal{D}$ (in white) are trained in an adversarial setting.

5.1 Architecture

In our architecture,we use two pre-trained classifiers $\mathcal{D}_y$ and $\mathcal{D}_z$, denoted by *target* and *sensitive* classifier, as approximations of f_z and f_y. We also use a GAN, with a generative model $\mathcal{G}$ in charge of producing the perturbed data $\mathrm{x}' = \mathcal{G}(\mathrm{x})$, and a discriminative model $\mathcal{D}$ which distinguishes x' from x. $\mathcal{P}_{\text{data}}$ and $\mathcal{P}_g$ represent the distribution of original and perturbed data respectively. Both $\mathcal{D}_y$ and $\mathcal{D}_z$

are pre-trained classifiers plugged into our network. One can easily extend our model to employ multiple classifiers in cases where there are more than one sensitive or target attributes. The overall structure of the network is shown in Figure 3.

The process starts by generator $\mathcal{G}$ taking the original instance x as input and generating the perturbed version of the data, $\mathcal{G}(\mathrm{x})$. Then $\mathcal{G}(x)$ is fed to the discriminators $\mathcal{D}$, $\mathcal{D}_y$ and $\mathcal{D}_z$. $\mathcal{D}$'s goal is distinguishing real data from perturbed data. $\mathcal{D}(\mathrm{x})$ represents the probability that x is an original datapoint ($\mathrm{x} \sim \mathcal{P}_{\text{data}}$) rather than one produced by $\mathcal{G}$ ($\mathrm{x} \sim \mathcal{P}_g$). We can write its loss function as below:

$$\mathcal{L}_{\mathcal{D}} = [\mathbb{E}_{\mathrm{x}\sim\mathcal{P}_{\text{data}}(\mathrm{x})} \log \mathcal{D}(\mathcal{G}(\mathrm{x})) + \mathbb{E}_{\mathrm{x}\sim\mathcal{P}_{\text{data}}(\mathrm{x})} \log\big(1-\mathcal{D}(\mathrm{x})\big)]. \quad (7)$$

$\mathcal{G}$ has multiple objectives. To ensure the utility of perturbed data while protecting privacy, we need to reintroduce our main objective function (1). However, since we know that its second term, (2), is maximized when the Maximum Entropy Distribution (MED) is most faithfully followed, we can rewrite (1) as:

$$\mathcal{L}_{\text{Adv}} = \mathbb{E}_{\mathrm{x}}\big[\ell(\mathcal{D}_y(\mathcal{G}(\mathrm{x}),y) + \mathrm{CE}\big(\mathcal{D}_z(\mathcal{G}(\mathrm{x})|y),\mathrm{MED}(z|y)\big)\big], \quad (8)$$

where ℓ is a suitable loss function and $\mathrm{CE}(a,b)$ the cross entropy between two distributions a and b which serves as a measure of their distance. Since we like the predicted probabilities for z, $\mathcal{D}_z(\mathcal{G}(\mathrm{x})|y)$, to be as close as possible to $\mathrm{MED}(z|y)$ we minimize this distance. $\mathcal{G}$, the GAN generator, wants to fool the GAN discriminator $\mathcal{D}$ in order to create perturbed data indistinguishable from real ones; the loss function for this will be:

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\mathrm{x}} \log\big(1-\mathcal{D}(\mathcal{G}(\mathrm{x}))\big). \quad (9)$$

Finally, it has been shown that using regularization can greatly stabilize GAN's training and also provide an additional lever to control the utility of the perturbed data by limiting the overall perturbation one is allowed to add to the data [17]. Here, we use a *hinge* loss:

$$\mathcal{L}_{\text{hinge}} = \mathbb{E}_{\mathrm{x}} \max\big(0, ||\mathcal{G}(\mathrm{x})-\mathrm{x}||-c\big), \quad (10)$$

where c is the maximum distance allowed before any loss occurs.

Our full objective for $\mathcal{G}$, combining (8), (9) and (10), is:

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\text{GAN}} + \alpha\mathcal{L}_{\text{Adv}} + \beta\mathcal{L}_{\text{hinge}}, \quad (11)$$

where α and β control the relative importance of the three losses. At each iteration, we alternate between training $\mathcal{D}$ and $\mathcal{G}$ while $\mathcal{D}_z$ and $\mathcal{D}_y$ are previously trained and fixed in the network.

5.2 PR-GAN Optimality

In the original GAN design, the generator $\mathcal{G}$ defines a probability distribution $\mathcal{P}_g$ as the distribution of samples generated by $\mathcal{G}$. Under favorable assumptions, the distribution $\mathcal{P}_g$ converges to a good estimator of $\mathcal{P}_{\text{data}}$ – the distribution of training data [13]. In our case, as we introduce additional classifiers and gradients, we wish to understand how these classifiers, $\mathcal{D}_z$, $\mathcal{D}_y$, modify the optimal solution and the final distribution. We follow the same assumption as in [13]: there is enough capacity and training time and the discriminator is allowed to reach its optimal given a fixed generator $\mathcal{G}$.

5.2.0.1 Fix $\mathcal{G}$, Optimize $\mathcal{D}$.

Notice that the discriminator $\mathcal{D}$ in our design uses the *same* loss function as in the original GAN. So the following claim is still true.

LEMMA 1 (OPTIMAL DISCRIMINATOR [13]). *For a fixed generator $\mathcal{G}$, the optimal discriminator $\mathcal{D}$ is*

$$\mathcal{D}^*_{\mathcal{G}}(\mathrm{x}) = \frac{\mathcal{P}_{data}(\mathrm{x})}{\mathcal{P}_{data}(\mathrm{x})+\mathcal{P}_g(\mathrm{x})},$$

where $\mathcal{P}_{data}(\mathrm{x})$ is the probability that x *is from the data and $\mathcal{P}_g(\mathrm{x})$ is the probability that* x *is from the generator $\mathcal{G}$.*

5.2.0.2 Fix $\mathcal{D}$, Optimize $\mathcal{G}$.

In the original GAN, the global minimum for the generator is achieved if and only if the generated distribution $\mathcal{P}_g$ is the same as $\mathcal{P}_{\text{data}}$: $\mathcal{P}_g(x) = \mathcal{P}_{\text{data}}(x)$. Recall from Section 4 that this is exactly the second claim of Theorem 1 when full knowledge about the conditional distribution $P(z|y)$ is allowed. This means that in this case, the mechanism $\mathcal{A}^*$ which achieves optimal results for (1) is also optimal as the GAN's generator $\mathcal{G}$. In the absence of such knowledge, there will be tradeoff between the optimality of GAN's generator and causing maximum uncertainty about $P(z|y)$ for an attacker. In such a case, a suitable method would be to set a fixed performance threshold for $\mathcal{D}_y$, the target classifier, and allow data to be perturbed only to the extent that $\mathcal{D}_y$ performs above that threshold. We apply this technique in our experiments in Section 6.

5.3 Implementation Details

To minimize dependencies between different components in our model, we slice a dataset into 3 parts equal in size and class proportions. We use the first slice to train $\mathcal{D}_y$ and $\mathcal{D}_z$, the second slice to train $\mathcal{G}$ and $\mathcal{D}$, and the third testing purposes.

We train and test the models on an Ubuntu 18.04.4 machine with a GeForce GTX 1080 Ti GPU. The code is written in Python using the TensorFlow 1.15.2 Deep Learning framework. The training of PR-GAN consists of two steps: (1) the classifiers for the target and sensitive attributes, $\mathcal{D}_y$ and $\mathcal{D}_z$ respectively, are trained, then (2) using those two, the noise generator, $\mathcal{G}$, is trained. As we will see in Section 6.3, the noise-generating module $\mathcal{G}$ is light-weight and capable of perturbing data on embedded devices designed for ML applications in real time.

6 Experiments

6.1 Datasets

Below, we go over the 4 datasets we have used along with sensitive and target attributes and the prior knowledge associated with each pair of attributes (recall that z is the sensitive attribute while y is the target attribute):

MNIST [23]: A dataset of 70,000 handwritten letters. For this dataset, y is the parity (being odd or even) of the numbers and z is whether or not a digit is less than 5; a toy example.

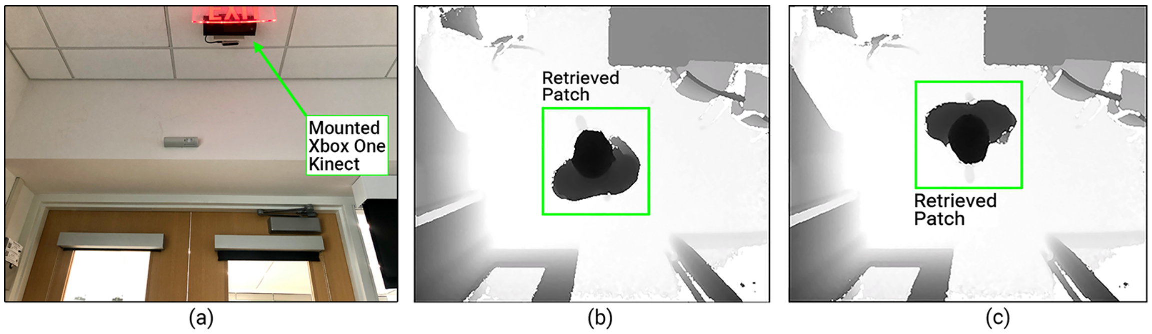

Figure 4: An Xbox One Kinect mounted above an entrance door (a) along with a sample depth frame of someone facing inside (b) and outside (c) in the OccuTherm dataset. The square patches in (b) and (c) are retrieved from images to use for recognition.

The prior knowledge, θ_E for *evens* and θ_O for *odds*, is as follows:

$$\theta_O(z) = \begin{cases} 0.4 & \text{if } z < 5 \\ 0.6 & \text{if } z \geq 5 \end{cases}, \quad \theta_E(z) = \begin{cases} 0.6 & \text{if } z < 5 \\ 0.4 & \text{if } z \geq 5 \end{cases}.$$

PubFig Faces [22]: This dataset includes 58,797 images of 200 people. We define z as the identity of each subject while y is each person's gender, which can happen in a scenario where subjects are donating their images to train a gender classifier. We use MTCNN [46] to align the faces and remove duplicates for each person. Filtering out subjects with less than 200 images, we will have 5 female and 10 male subjects with a total of 6,553 images. For the prior knowledge, θ_M for men and θ_F for women, we have:

$$\theta_M(z) = \begin{cases} 1/10 & z \in \mathcal{Z}_m \\ 0 & z \in \mathcal{Z}_f \end{cases}, \quad \theta_F(z) = \begin{cases} 0 & z \in \mathcal{Z}_m \\ 1/5 & z \in \mathcal{Z}_f \end{cases},$$

where $\mathcal{Z}_m$ and $\mathcal{Z}_f$ are the set of male and female subjects.

UJI Indoor Localization (WiFi) [40]: In this dataset, signal strengths of 520 WiFi access points (WAP) are recorded for 21,048 locations inside 3 different buildings. The buildings have a total of 13 floors. In addition, each data record is associated with one of 8 regions in each floor. Similar to the scenario discussed in Section 1, we define z as the specific region a user is in on each floor (one of 104 total possibilities), and y as the floor on which the user is at any moment (one of 13 options). The strength signals, originally continuous, are turned into binary attributes that represent the "existence" of any signal from a WAP; this proved to produce the best performance in our experiments. For each floor $y \in [13]$, there is set of *valid* regions, $\mathcal{Z}_y$, which includes the 8 regions associated with that floor. For $\theta_y(z)$ we have:

$$\theta_y(z) = \begin{cases} 0 & z \notin \mathcal{Z}_y \\ 1/8 & z \in \mathcal{Z}_y \end{cases}.$$

OccuTherm [31]: This dataset consists of depth frames taken by a Kinect for XBOX One mounted above the entrance door of a conference room[31]. It includes frames of 77 subjects facing inside the room, and then facing outside of it. The installation setting, along with a sample of the resulting two images can be found in Figure 4. Each depth frame has a resolution of 512×424. From each frame, we retrieve a 180×180 patch centered around the participant in the frame. Here, we define z as the identity of users, while y is whether a subject is facing inside, or outside the room; a binary attribute. To rely on subjects with enough images available, we limit the dataset to the top 20 participants with the most number of images. That leaves us with 60,033 images in total. Since y and z have no correlation, the prior knowledge for the conditional distribution between z and y is a uniform distribution:

$$\theta_y(z) = 1/n,$$

where n is the total number of people in the dataset, 20 here. The architecture for the neural networks $\mathcal{D}_y$ and $\mathcal{D}_z$ used for each dataset is available in Table 1 ("VGG16 base" refers to the bottom layers of VGG16 [38] architecture with modified top layers).

Table 1: Model architectures for the 4 datasets (Conv: convolution layer, FC: fully-connected layer).

MNIST	PubFig	WiFi	OccuTherm
Conv(64,5,5)	VGG16 base	FC(256)	VGG16 base
ReLu	FC(1024)	ReLu	FC(1024)
Conv(64,5,5)	ReLu	Dropout(0.5)	ReLu
ReLu	Dropout(0.5)	FC(128)	Dropout(0.5)
Dropout(0.25)	FC(512)	ReLu	FC(512)
FC(128)	ReLu	Dropout(0.5)	ReLu
ReLu	FC(Dim(Y))	FC(64)	FC(Dim(Y))
Dropout(0.5)		ReLu	
FC(2)		FC(Dim(Y))	

6.2 Baseline

In our experiments, we compare our method against the following baselines:

Naive Generative Privacy (NGP): Some have previously proposed the use of $\mathcal{D}_y$ and $\mathcal{D}_z$, trained in competition, without the GAN's discriminator, $\mathcal{D}$. To show that this module contributes positively to the performance of $\mathcal{G}$, we test our results against a baseline where $\mathcal{D}$ is removed, calling it *Naive Generative Privacy* (NGP). The architecture is depicted on Figure 5, left.

Adversarial Privacy (AP): Another line of work similar to ours proposes the production of adversarial samples against $\mathcal{D}_z$ with a limit on the amount of noise, similar to our regularization at (10), to ensure high utility. These methods

Table 2: Sensitive classifier accuracy, $Acc(\mathcal{D}_z)$, across all methods over 4 datasets. Lower is better.

Method	Sensitive Classifier Accuracy (%)			
	MNIST $Acc(\mathcal{D}_y) \geq 0.95$	PubFig $Acc(\mathcal{D}_y) \geq 0.95$	WiFi $Acc(\mathcal{D}_y) \geq 0.75$	OccuTherm $Acc(\mathcal{D}_y) \geq 0.95$
PR-GAN	**65.3**	**20.6**	**23.6**	**3.6**
NGP	67.2	28.1	31.3	7.2
AP	67.1	61.9	37.7	23.6
DP	80.6	78.3	50.0	37.6
Random*	60.0	13.3	12.5	5.0
Original $Acc(\mathcal{D}_z)$	98.4	80.7	84.6	99.8
Original $Acc(\mathcal{D}_y)$	99.2	98.3	92.6	99.95

* A classifier that outputs a class at random according to $\theta_y(z)$.

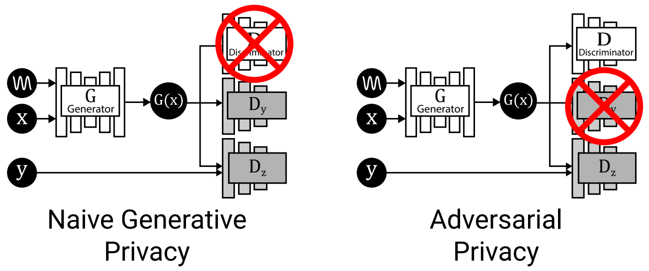

Figure 5: Two baseline approaches created by removing (left) the GAN's discriminator, $\mathcal{D}$, and (right) the target classifier.

lack the presence of $\mathcal{D}_y$ to directly guide the perturbations towards higher utility. To test the effect of $\mathcal{D}_y$'s absence, we remove this module, keeping the rest of the architecture similar, as seen on Figure 5, right, calling it *Adversarial Privacy* (AP).

Differential Privacy (DP): To highlight the fact that guidance on both $\mathcal{D}_y$ and $\mathcal{D}_z$ is essential, we show that conventional methods, such as DP fail to achieve a desirable trade-off between utility and privacy. For real-valued vectors (image datasets), one can use the *Laplace Mechanism* known to achieve ε-differential privacy, where independent noise is added to each pixel via the Laplacian distribution:

$$\mathrm{Lap}(z|b) = \frac{|z|}{2b} \exp\left(-\frac{1}{b}\right),$$

where b is the scale parameter. This method achieves $1/b$-differential privacy [10].

Not all of the baseline approaches have a mechanism to assure high utility. We make results comparable by setting a threshold for $\mathcal{D}_y$'s accuracy score (denoted by $Acc(\mathcal{D}_y)$) and add as much noise as possible without going below that threshold when $\mathcal{D}_y$ is tested on resulting perturbed data. Our method is denoted by PR-GAN throughout experiments.

6.3 Running on Mobile Devices

As discussed in Section 3, once trained, the goal is to install the trained generator ($\mathcal{G}$) on remote sensing devices to produce perturbations from source. This has the advantage that *non-perturbed* will never leave local devices. The complexity and efficiency of a neural network depends on many factors. However, as is common practice, we measure it by counting the number of parameters in a network and the number of floating-point operations (FLOP). In Table 3, we compare the complexity of our networks with state-of-the-art networks designed specifically to run on embedded devices. As you can see, PR-GAN is more compact and computationally cheaper compared to the state-of-the-art across all 4 datasets, making them suitable to run on current mobile devices. In Table 4, we have measured the running time and the corresponding frame-per-second (*input*-per-second in the case of WiFi dataset) capability of PR-GAN for the 4 datasets. Using the running time measured on a GeForce GTX 1080 Ti GPU across the 4 datasets, we use the latest benchmarks for 3 devices designed for ML applications, namely Nvidia's Jetson Nano, Raspberry Pi 3 with an Intel NCS2, and Coral Dev Board, to estimate the running time of PR-GAN on these devices [2, 32] using linear regression. The two smaller datasets, WiFi and MNIST, allows for high-throughput generation of noise by our generator model, up to hundreds per second and easily handling any scenario. For the two more complicated datasets, PubFig and OccuTherm, with an acceptable rate to allow PR-GAN to be used in real-life settings; in the case of OccuTherm dataset, we achieve more than 10 fps while 3 fps is known to be good enough for real-time tasks [30]. Although the estimated values might have inaccuracies, the resulting values are, in virtually all of the cases, at least one order of magnitude greater than what is required.

Table 3: The complexity of our models compared to state-of-the-art models designed for ImageNet [21] classification task on mobile devices.

Network	FLOP	Parameters
MobileNetV1 1.0 [15]	575M	4.2M
ShuffleNet 1.5x [47]	292M	3.4M
NasNet-A [48]	564M	5.3M
MobileNetV2 1.0 [36]	300M	3.4M
Our Noise Generators		
MNIST	1.6M	235.4K
PubFig	232M	644.2K
WiFi	2.1M	1.1M
OccuTherm	96.4M	420.3K

6.4 Performance

To show our method's ability to preserve y while causing uncertainty about z, we compare the resulting classifica-

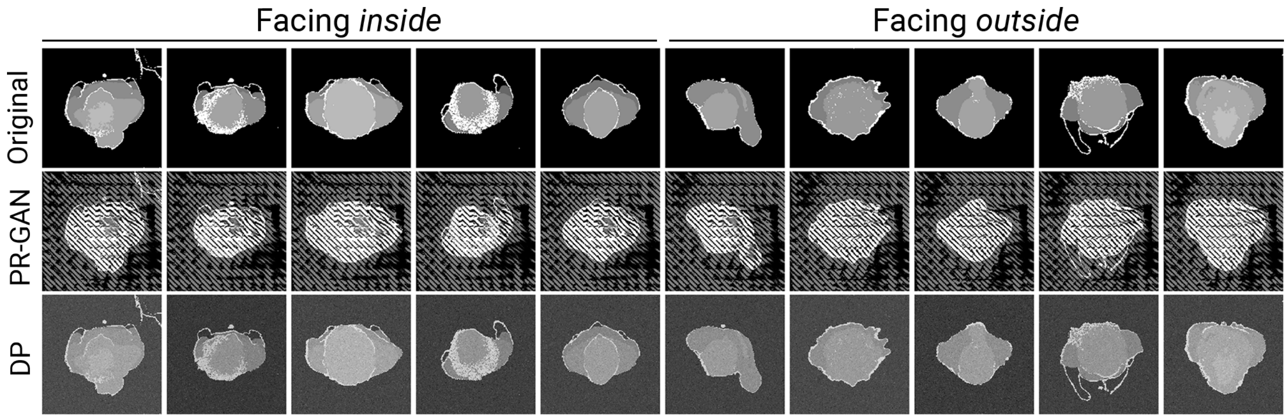

Figure 6: A randomly selected sample of images from the OccuTherm dataset. The first row includes the original patches and the second and third rows include perturbed images using PR-GAN and Laplacian noise (DP) respectively. In the 5 columns on the left the subject is facing *inside* while in the 5 columns on the right (s)he is facing *outside*.

Table 4: PR-GAN noise generation speed, measured by frames (*input* in the case of WiFi dataset) per second, for the 4 datasets on 4 devices.

Network	GTX 1080 Ti	Jetson Nano*	RPI3 + Intel NCS2*	Coral Dev Board*
MNIST	22.1K	1.7K	826.4	3.9K
PubFig	372.0	29.1	10.7	42.2
WiFi	36.0K	2.8K	1.4K	6.4K
OccuTherm	595.8	46.7	19.1	82.0

* Values are estimated based on running time on GeForce GTX 1080 Ti GPU.

tion accuracy for sensitive attributes, $Acc(\mathcal{D}_z$, with a fixed utility, $Acc(\mathcal{D}_y)$, threshold. For the the WiFi dataset, we choose 75% as the threshold for $Acc(\mathcal{D}_y)$ and for the 3 image datasets, namely MNIST, PubFig and OccuTherm, we choose 95%. These thresholds are selected based on the difficulty of achieving acceptable privacy guarantees and what an acceptable value for $Acc(\mathcal{D}_y)$ is for each dataset. For methods based on DP, the optimal value of DP's ε is found by iterating over different values of ε from 0.01 to 10 and selecting the smallest ε (corresponding to the largest added noise) where $Acc(\mathcal{D}_y)$ is still above the set threshold and reporting the resulting $Acc(\mathcal{D}_z)$.

Results are available in Table 2. The original performance of $\mathcal{D}_y$ and $\mathcal{D}_z$ are available in the two rows at the bottom. The performance of a random classifier which uses the prior knowledge, $\theta_y(z)$, is reported in the third row from the bottom. Note that this value is theoretically the lowest possible accuracy a privacy mechanism can guarantee for the classification of z.

First, note that our method outperforms all others in all 4 datasets. In the case of the OccuTherm dataset, PR-GAN produces the theoretically optimal perturbations; it decreases $Acc(\mathcal{D}_z)$ to its lowest possible value (the row denoted by "Random*" in Table 2) while maintaining an accuracy of 99.6% for $Acc(\mathcal{D}_y)$, just 0.2% below its original value, 99.8%. A randomly selected sample of 10 individual patches, 5 looking inside and 5 looking outside, are depicted in Figure 6 along with the corresponding perturbed versions of each image produced by PR-GAN and DP. Next, notice that the two methods that do not utilize the target attributes in producing perturbations (DP and AP) achieve results that are far less promising than the other two methods (PR-GAN and NGP). The discrepancy is far more pronounced in the two image datasets, PubFig and OccuTherm, where producing perturbations is harder due to the complex nature of the data and the higher number of pixels in play. Compared to NGP, our method consistently achieves better results. This reveals that the existence of the full structure of a GAN, and specifically the GAN discriminator $\mathcal{D}$, is effective in producing perturbations that preserve high utility for the data. We dive deeper into the difference between the two Section 6.5. Producing noises for the WiFi dataset proved to be the most difficult task, however, PR-GAN achieves considerably better performance than NGP in this case as well. There are two main reasons for this difficulty. First, the conventional neural networks for image data are far more advanced than those used on arbitrary feature vectors, such as the one in the WiFi dataset. For instance, to produce gradients for discrete-valued feature vectors, one has to apply complicated techniques, none of which is necessary when one works with image data. Second, the target and sensitive attributes in this case are *highly* correlated; for any sensitive attribute (a region on a floor), only 1 out of 13 possible target attributes are valid, which heavily restricts the space in which PR-GAN can search for optimal solutions.

6.5 Utility vs. Privacy

Inherent in any privacy protection scenario is the trade-off between privacy and utility. To show our method's superior ability to find a good trade-off, we conduct an experiment on the WiFi dataset in which different *utility budgets*, in terms of how low can $Acc(\mathcal{D}_y)$ get, is given to the 4 methods. The smaller size of the WiFi dataset is what allows us to do such an exhaustive experiment; doing so on the other 3 datasets was not possible given resources available to us. The budget, $\Delta Acc(\mathcal{D}_y)$, is set to 5 values between 0.03 and 0.12. The results are depicted in Figure 7. As you can see, our method is consistently above the other 3, and the two methods that do not utilize the target labels in any way, namely AP and DP, have a significantly slower gain in privacy guarantee (the

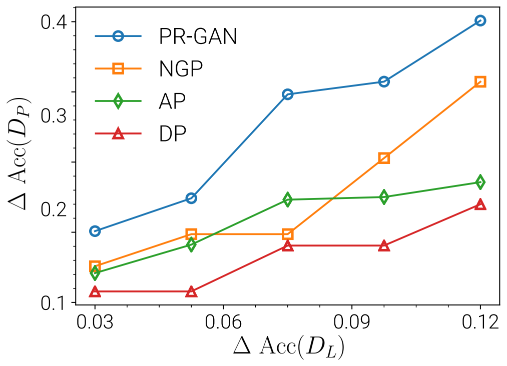

Figure 7: The trade-off between achieved privacy (Δ $Acc(\mathcal{D}_z)$) and utility loss budget (Δ $Acc(\mathcal{D}_y)$ for the WiFi dataset.

amount $Acc(\mathcal{D}_z)$ is decreased) as the utility budget increases.

6.6 Transferability

In an identity-masking scenario such as the one in our experiment with the OccuTherm dataset, it might not be possible to have every occupant of a building to contribute data to the training of PR-GAN. The occupants change over time, the manufacturer of PR-GAN might be a different entity than the building managers and occupants, and most importantly, by collecting raw images of every occupant, we endanger their privacy in the first place. To see if an entity, such as the manufacturer, can train a model on a group and deploy it on a completely different group, we conduct the following experiment: We use the noise generator from Section 6.4, trained on the top-20 individuals with most images, and produce perturbations for the next 30 individuals with the highest number of images, each having at least 1000 images. Then, using an identity-classifier trained exclusively on this new group of 30 individuals, we test the resulting perturbed methods. We observe that $Acc(\mathcal{D}_z) = 3.71\%$ and $Acc(\mathcal{D}_y) = 86.9\%$, showing minimal performance loss on y and optimal privacy on z. This means that our model's ability to hide identity-revealing features transfers from one group to another, completely new group, not-seen-before during training. This is promising as it allows a manufacturer to train $\mathcal{G}$ in a warehouse and ship it to different sites for deployment, expecting high performance.

6.7 Training Utility

Table 5: Utility of the perturbed data when used for training a new model, compared to when it is used for inference (Table 2).

Application	Target Classifier Accuracy (%)			
	MNIST	PubFig	WiFi	OccuTherm
Inference	95.19	95.47	79.80	99.82
Training	96.72	98.84	72.06	99.73

We finally test whether the data perturbed by PR-GAN could be used to train a new model, which can perform well on raw, clean data. This is Inspired by a scenario where a model is trained by public, perturbed data, and is then deployed to personal devices to perform classification on raw, clean data. The results are available in Table 5. The first row is a refresher on Section 6.4, where a model was trained on clean data and inference was done on perturbed data (the opposite of what we do in this section). The second row includes the accuracy of $\mathcal{D}_y$ when perturbed data is used for training and clean data (from a separate slice) is used for testing. As you can see, overall, the performance values are close in both cases across all datasets.

7 Conclusion and Future Work

In this work, we designed and implemented a framework to bridge the gap between privacy preserving data publishing and deep generative models. We showed that it is possible to use deep neural networks as clues for generating tailored perturbations, which successfully hides sensitive information while offering a high degree of utility for both inference and training tasks. By choosing this approach, not only we can effectively protect sensitive information, but we can also maintain the information necessary for a given target application. Future work includes finding specialized architectures to perturb different types of data (e.g.: time series) and improving on the privacy guarantees we provide.

Acknowledgement: The authors would like to acknowledge support in part from NSF CNS-1618391, DMS-1737812, OAC-1939459, DOE DE-EE0007682. The opinions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM.

[2] A. Allan. Benchmarking tensorflow and tensorflow lite on the raspberry pi. "https://www.hackster.io/news/benchmarking-tensorflow-and-tensorflow-lite-on-the-raspberry-pi-43f51b796796", 2019.

[3] M. Bertran, N. Martinez, A. Papadaki, Q. Qiu, M. Rodrigues, G. Reeves, and G. Sapiro. Adversarially learned representations for information obfuscation and inference. In *International Conference on Machine Learning*, pages 614–623, 2019.

[4] A. J. Bose and P. Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. *arXiv preprint arXiv:1805.12302*, 2018.

[5] A. Boutet, C. Frindel, S. Gambs, T. Jourdan, and C. R. Ngueveu. Dysan: Dynamically sanitizing motion sensor data against sensitive inferences through adversarial networks. *arXiv preprint arXiv:2003.10325*, 2020.

[6] R. J. Bowden and A. B. Sim. The privacy bootstrap. *Journal of Business & Economic Statistics*, 10(3):337–345, 1992.

[7] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[8] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and

V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[9] C. Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Venice, Italy, July 2006. Springer Verlag.

[10] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, 2014.

[11] N. Eagle, A. S. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proc. Natl. Acad. Sci. U. S. A.*, 106(36):15274–15278, 8 Sept. 2009.

[12] H. Edwards and A. Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[14] J. Hamm. Minimax filter: learning to preserve privacy from inference attacks. *The Journal of Machine Learning Research*, 18(1):4704–4734, 2017.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[16] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal. Context-aware generative adversarial privacy. *Entropy*, 19(12):656, 2017.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[18] D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. In *ACM Transactions on Database Systems*, volume 39, page 3, 2014.

[19] W. Kleiminger, C. Beckel, A. K. Dey, and S. Santini. Using unlabeled wi-fi scan data to discover occupancy patterns of private households. In *The 11th ACM Conference on Embedded Network Sensor Systems, SenSys '13, Roma, Italy, November 11-15, 2013*, pages 47:1–47:2, 2013.

[20] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 365–372. IEEE, 2009.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] X.-B. Li and S. Sarkar. A tree-based data perturbation approach for privacy-preserving data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1278–1283, Sept 2006.

[25] B. Liu, Y. Jiang, F. Sha, and R. Govindan. Cloud-enabled privacy-preserving collaborative learning for mobile sensing. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 57–70. ACM, 2012.

[26] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering*, 18(1):92–106, 2006.

[27] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 71–84, New York, NY, USA, 2010. ACM.

[28] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 337–350, New York, NY, USA, 2008. ACM.

[29] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 3–18, 2020.

[30] S. Munir, R. S. Arora, C. Hesling, J. Li, J. Francis, C. Shelton, C. Martin, A. Rowe, and M. Berges. Real-time fine grained occupancy estimation using depth sensors on ARM embedded platforms. In *RTAS*, 2017.

[31] S. Munir, J. Francis, M. Quintana, N. V. Frankenberg, and M. Berges. Dataset: Inferring thermal comfort using body shape information utilizing depth sensors. In *DATA*. ACM, 2019.

[32] Nvidia. Jetson nano: Deep learning inference benchmarks, 2019.

[33] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for

deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[34] N. H. Phan, Y. Wang, X. Wu, and D. Dou. Differential privacy preservation for deep Auto-Encoders: an application of human behavior prediction. *AAAI*, 2016.

[35] N. Raval, A. Machanavajjhala, and J. Pan. Olympus: sensor privacy through utility aware obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2019(1):5–25, 2019.

[36] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[37] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 909–910, 2015.

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[39] L. Sweeney. Achieving *k*-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[40] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 261–270. IEEE, 2014.

[41] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1100–1108. ACM, 21 Aug. 2011.

[42] J. Whitehill and J. Movellan. Discriminately decreasing discriminability with learned image filters. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2488–2495. IEEE, 2012.

[43] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[44] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner. Crowd++: Unsupervised speaker count with smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 43–52, New York, NY, USA, 2013. ACM.

[45] M. Zeifman. Smart meter data analytics: Prediction of enrollment in residential energy efficiency programs. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 413–416, Oct. 2014.

[46] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.

[47] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017.

[48] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

A Proof of Theorem 1

PROOF. The first claim follows by definition. First $\mathcal{A}^*$ follows the MED, and second the corresponding label y for each entry is untouched, meaning $\Pr[f_y(\mathrm{x}) = y] = \Pr[f_y(\mathcal{A}^*(\mathrm{x}) = y]$. This shows that the loss value will remain optimal for f_y.

We now focus on the second claim. By looking at (3) one can realize that if for all $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$, $\theta_y(z)$ exists, then $\mathrm{MED}(z|y)$ becomes the real distribution of z conditioned on y: $P(z|y)$. This means that $\Pr[\mathcal{A}^*(\mathrm{x}) \in \mathcal{X}(y,z)] = P(z|y)$. Knowing this, we can write about the probability of observing x' in a region $\mathcal{X}(y,z)$ for any z as:

$$\begin{aligned}\Pr\left[\mathrm{x}' \in \mathcal{X}(y,z)\right] &= \sum_{z' \in \mathcal{Z}} \Pr\left[\mathrm{x} \in \mathcal{X}(y,z')\right] P(z|y) \\ &= P(z|y) \sum_{z'} \Pr\left[\mathrm{x} \in \mathcal{X}(y,z')\right] \\ &= P(z|y) \Pr_{\mathrm{x}\sim X}\left[f_y(\mathrm{x}) = y\right] \\ &= \Pr_{\mathrm{x}\sim X}\left[\mathrm{x} \in \mathcal{X}(y,z')\right].\end{aligned}$$

which concludes the proof. □

B Extension to Approximate Predictors

Suppose that g_z and h_z are two approximations of f_z, the former used by the protector of privacy to produce perturbations and the latter by an attacker to reveal sensitive information. Suppose also that h_z and g_z have a bounded approximation error ε such that for all $\mathrm{x} \in X$:

$$\begin{aligned}|\Pr[g_z(\mathrm{x}) = z|y] - \Pr[f_z(\mathrm{x}) = z|y]| \leq \varepsilon \\ |\Pr[h_z(\mathrm{x}) = z|y] - \Pr[f_z(\mathrm{x}) = z|y]| \leq \varepsilon.\end{aligned} \tag{12}$$

THEOREM 3. *Any privacy mechanism $\mathcal{A}$ that is (ε,δ)-Pufferfish for g_z is $(\varepsilon, \delta + 2\varepsilon(e^{\varepsilon}+1))$-Pufferfish for h_z.*

PROOF. Given the conditions in (12), we can write:

$$|\Pr[g_z(\mathrm{x}) = z|y] - \Pr[h_z(\mathrm{x}) = z|y]| \leq 2\varepsilon. \tag{13}$$

By (ε,δ)-Pufferfish guarantees we have:

$$\Pr[g_z(\mathrm{x}) = z_1|y] \leq e^{\varepsilon} \Pr[g_z(\mathrm{x}) = z_2|y] + \delta. \tag{14}$$

Incorporating (13) in (14) yields:

$$\begin{aligned}\Pr[h_z(\mathrm{x}) = z|y] - 2\varepsilon &\leq e^{\varepsilon}\left(\Pr[h_z(\mathrm{x}) = z|y] + 2\varepsilon\right) + \delta \\ \Pr[h_z(\mathrm{x}) = z|y] &\leq e^{\varepsilon} \Pr[h_z(\mathrm{x}) = z|y] + 2\varepsilon(e^{\varepsilon}+1)\delta,\end{aligned}$$

which concludes the proof. □

Theorems 2 and 3 result in the following.

COROLLARY 1. *Against an attacker who uses an unseen predictor of z with bounded error ε, we guarantee $(0, 2\gamma + 4\varepsilon)$-Pufferfish privacy.*