Generating Optimal Privacy-Protection Mechanisms via Machine Learning

Marco Romanelli  
INRIA  
marco.romanelli@inria.fr

Catuscia Palamidessi  
INRIA, École Polytechnique  
catuscia@lix.polytechnique.fr

Konstantinos Chatzikokolakis  
University of Athens  
kostas@chatzi.org

Abstract—We consider the problem of obfuscating sensitive information while preserving utility. Given that an analytical solution is often not feasible because of un-scalability and because the background knowledge may be too complicated to determine, we propose an approach based on machine learning, inspired by the GANs (Generative Adversarial Networks) paradigm. The idea is to set up two nets: the generator, that tries to produce an optimal obfuscation mechanism to protect the data, and the classifier, that tries to de-obfuscate the data. By letting the two nets compete against each other, the mechanism improves its degree of protection, until an equilibrium is reached. We apply our method to the case of location privacy, and we perform experiments on synthetic data and on real data from the Gowalla dataset. We evaluate the privacy of the mechanism not only by its capacity to defeat the classifier, but also in terms of the Bayes error, which represents the strongest possible adversary. We compare the privacy-utility tradeoff of our method with that of the planar Laplace mechanism used in geo-indistinguishability, showing favorable results.

I. INTRODUCTION

Big data are the lifeblood of modern economy and, consequently, there is an enormous interest in collecting all sort of personal information. Individuals, on the other hand, are often willing to provide their data in exchange of improved services and experiences. However, with big data comes big concern about privacy: a person who makes a daily use of connected devices and social media may disclose a huge amount of detailed and accurate information, which could later on be used against him or her. It could affect everything from relationships to getting a job, or qualifying for a loan, or worse.

The rise of machine learning, with its capability of performing powerful analytics on massive amounts of data, has further exacerbated the threats to privacy. To illustrate the point, consider the following scenario: some users send their identities and coordinates to a Location Based Service (LBS) to obtain some kind of assistance, for instance the points of interest (POIs) near them. An attacker that has access to the LBS could collect the traces of these users for a while, and use machine learning techniques and some background knowledge (for instance the home address of the users) to infer information from them. For example, he could train a machine to classify traces, to associate a class to a home address and therefore to a user, and also to connect a trace outset to its possible continuations. Later on, the attacker could use the machine to identify the user from a new trace (even if the trace does not contain the home address), or to predict the likely next location the user will visit.

Nonetheless, if machine learning can be a threat, it can also be a powerful mean to build good privacy protection mechanisms. We envision the possibility of deploying a machine that counters the attack, and that is able to learn the best defense strategy by interacting with the adversary.

In this paper we focus on defense mechanisms based on data obfuscation by addition of controlled noise. Now, clearly more noise means more privacy, but it is important to note that, in general, privacy is not the only concern: a good defense must not only prevent the attacker from discovering sensitive information, but also disclose what is needed to get the desired quality of service. In the example above, the user may report to the LBS an obfuscated location, but still he or she expects a service in return, and the quality of service (QoS) usually degrades with the amount of obfuscation. For instance, reporting a fixed location, or a randomly chosen one, would guarantee privacy, but would result in a very low utility, because the obtained POIs would be close to the reported location, which would usually be far from the real one.

The trade-off between privacy and utility is one of the main challenges in design of privacy mechanisms. In this paper, following the approach of [1], we aim at maximizing the privacy protection while preserving the desired QoS, i.e. the expected utility loss must remain below some tolerance threshold with respect to a chosen metric. We focus on the example of location privacy and in particular on the issue of re-identification of the user from his location, but the framework that we develop is general and can be applied to any situation in which an attacker might infer sensitive information from accessible correlated data.

The optimal trade-off between privacy and utility can in principles be achieved with linear programming. This problem can in fact be thought of as a linear optimization problem where the privacy is the objective function and the threshold on utility is the constraint. The limitation of this approach, however, is that it does not scale to large datasets: the existing tools cannot handle more than a few hundreds possible locations. Furthermore, the background knowledge and the

1Other approaches to location privacy take the opposite view, and aim at maximizing utility while achieving the desired amount of privacy, see for instance [2].
correlation between data points affect privacy and they are usually difficult to determine and express formally.

Our position is that **machine learning can be the solution to this problem**. Inspired by the GANs paradigm [3], we propose a system consisting of two neural networks: a generator \( G \) and a classifier \( C \). The idea is that \( G \) generates noise so to confuse the adversary as much as possible, within the boundaries allowed by the utility requirement, while \( C \) inputs the noisy locations produced by \( G \) and tries to learn how to re-identify the corresponding user. In other words, \( C \) tries to build a classification function, where the obfuscated locations are the features, and the users’ ids are the labels. The classification produced by \( C \) is then fed back to \( G \), which uses it to regulate the noise injection. While fighting against \( C \), \( G \) learns to produce a more and more “clever” noise function, until it reaches a point where it cannot improve any longer.

A main difference between our approach and the GANs is that the GANs have access to a dataset sampled from the distribution that \( G \) should learn to reproduce. The adversary, which in that case is called \( D \) (discriminator), tries to distinguish between the real data (from the dataset) and those generated by \( G \). The net \( G \) then “learns” the target distribution from the feedback provided by \( D \). In our case, on the contrary, there is no dataset of samples that can “direct” \( G \) towards an optimal noise distribution. In a sense, our \( G \) has to be more “creative” and “invent” a good distribution from scratch.

A major challenge in our setting is represented by the choice of the target function. To illustrate the issue, let us explain in more detail the ideas behind our approach. The goal of \( G \) is to achieve the optimal privacy-utility tradeoff. Concerning utility, we can formalize it as the expected distance between the real location and the obfuscated one\(^2\). This is a typical definition in location privacy (see for instance [1], [2], [5], [6]), and captures the fact that location based services usually offer a better quality of service when they are provided with a more accurate location. Concerning privacy, a first idea would be to measure it in terms of \( C \)'s misclassification, since we are trying to prevent the attacker from associating a location to the corresponding user. Following this idea, we could define the privacy loss \( f \) as the expected success probability of \( C \)'s classification, or, alternatively, as the similarity between \( C \)'s classification and the real one (which can be formalized in terms of cross entropy).

We assume that users obfuscate their locations within the limits imposed by the utility constraint, and knowing that they will be observed by the attacker. About the attacker we assume that he can learn the obfuscation strategy deployed by the users, and will use it to improve the precision of the re-identification. This interplay between users and attacker can be seen as an instance of a zero-sum Stackelberg game [1], and modeled as the GANs mentioned above, where the users are represented by \( G \) (the leader), the attacker by \( C \) (the follower), and the payoff function \( f \) is the privacy loss described above. From a formal point of view, finding the optimal point of equilibrium between \( G \) and \( C \) corresponds to solving a minimax problem on \( f \) (\( G \) is the minimizer and \( C \) the maximizer). The function \( f \) can be proved to be convex/concave with respect to the strategies of \( G \) (obfuscation) and \( C \) (re-identification) respectively, so from standard game theory we know that there is a saddle point and that it corresponds to the optimal obfuscation-re-identification pair.

The problem, however, is that it is difficult to reach the saddle point via the typical back-and-forth interplay between the two nets. Let us clarify this point with a simple example:

**Example 1.** Consider two users, Alice and Bob, in locations \( a \) and \( b \) respectively. Assume that the first attempt of \( G \) is to report their true locations (no noise). Then \( C \) will learn that \( a \) corresponds to Alice and \( b \) to Bob. At the next round, \( G \) will figure out that its best strategy to maximize the misclassification error (given the choices of \( C \)) is to swap the locations, so that Alice reports \( b \) and Bob reports \( a \). Then, on its turn, \( C \) will have to “unlearn” the previous classification and learn the new one. But then, at the next round, \( G \) will again swap the locations, and bring the situation back to the starting point, and so on, without ever reaching an equilibrium. Note that the strategy of \( G \) in the equilibrium point would be the mixed strategy that reports \( a \) for both Alice and Bob\(^3\) (in this situation \( C \) would not be able to do better than random guessing), but \( G \) may not stop there. The problem is that it is difficult to calibrate the training of \( G \) so that it stops in proximity of the saddle point rather than continuing all the way to reach its relative optimum. The situation is illustrated in Fig.1a. The payoff function considered in this figure is the success probability of the classification, but it would be analogous if we considered, for instance, the cross entropy between the true ids and \( C \)'s prediction.

In order to address this issue, we propose to adopt a different target function, which is less sensitive to the particular labeling strategy of \( C \). The idea is to consider not just the precision of the classification, but, rather, the information contained in it, which represents the potential precision of an ideal classifier that uses that information in the optimal way. There are two main ways of formalizing this intuition: the mutual information and the complement of the Bayes error. More precisely, let \( X \) be the random variable associated to the true ids, and \( Y \) be the random variable associated to the ids resulting from the classification (predicted ids). We consider the mutual information between \( X \) and \( Y \), denoted by \( I(X; Y) \), which is defined as \( H(X) - H(X|Y) \), where \( H(X) \) is the entropy of \( X \) and \( H(X|Y) \) is the residual entropy of \( X \) given \( Y \). The Bayes error, which we will denote by \( B(X|Y) \), is defined as the expected probability of error under the MAP rule, which selects the value of \( X \) having maximum a posteriori probability given the value of \( Y \). Its complement \( 1 - B(X|Y) \), also known as (posterior) Bayes vulnerability [7],

\(^2\)Such expected distance is known as distortion in information theory [4].

\(^3\)There are two more equilibrium points: one is when both Alice and Bob report \( a \) or \( b \) with uniform probability, the other is when they both report \( b \). All the three strategies give the same payoff and we could choose any of them to illustrate the issue.
A vulnerability has been investigated in [17]. Explored in [15], [16], while its relation with the Bayes \cite{14}. Its relation with mutual information has been explored. The Santhi-Vardy bound \cite{13}: \( B(X|Y) \leq 1 - 2^{-H(X|Y)} \).

Another popular notion of privacy is differential privacy \cite{14}. Its relation with mutual information has been explored in \cite{15}, \cite{16}, while its relation with the Bayes vulnerability has been investigated in \cite{17}.

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
 & \( a \rightarrow a \) & \( a \rightarrow b \) & \( a \rightarrow a \) & \( a \rightarrow b \) \\
\hline
\( b \rightarrow b \) & 1 & 0.5 & 0.5 & 0 \\
\hline
\( b \rightarrow a \) & 0.5 & 0 & 0.5 & 0 \\
\hline
\end{tabular}
\end{center}

(a) \( f = I(X;Y) \). Hollow: \( f = 1 - B(X|Y) \). \hfill

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
 & \( a \rightarrow a \) & \( a \rightarrow b \) & \( a \rightarrow a \) & \( a \rightarrow b \) \\
\hline
\( b \rightarrow a \) & 1 & 1 & 0 & 0.5 \\
\hline
\( b \rightarrow b \) & 0 & 0.5 & 0 & 0.5 \\
\hline
\end{tabular}
\end{center}

(b) \textbf{Bold:} \( f = I(X;Y) \). 

Fig. 1: Payoff tables of the games between \( G \) and \( C \) in Example 1, for various payoff functions \( f \). \( A \) stands for \( Alice \) and \( B \) for \( Bob \).}

represents then the precision of an ideal classifier that makes the best possible guess about the true id from the classification produced by \( C \).

If we set \( f \) to be \( I(X;Y) \) or \( 1 - B(X|Y) \), we obtain the payoff table illustrated in Fig. 1b. We note that the minimum \( f \) in the first and last columns corresponds now to a point of equilibrium for any choice of \( C \). This is not always the case, but in general it is closer to the point of equilibrium and makes the training of \( G \) more stable, in the sense that training \( G \) for a longer time does not risk to increase the distance from the equilibrium point.

In this paper we will use the mutual information to generate the noise, but we will evaluate the level of privacy of the resulting mechanism also in terms of the Bayes error. Both notions have been used in the literature as privacy measures, for instance mutual information has been applied to quantify anonymity \cite{8}, \cite{9}. The Bayes error and the Bayes vulnerability have been considered in \cite{7}, \cite{9}–\cite{11}, and they are strictly related to the min-entropy leakage \cite{12}. Mutual information and Bayes error are related to each other by the Santhi-Vardy bound \cite{13}: \( B(X|Y) \leq 1 - 2^{-H(X|Y)} \).

Another popular notion of privacy is differential privacy \cite{14}. Its relation with mutual information has been explored in \cite{15}, \cite{16}, while its relation with the Bayes vulnerability has been investigated in \cite{17}.

\section*{A. Contribution}

The contributions of the paper are the following:

- We propose an approach based on adversarial nets to generate obfuscation mechanisms with a good tradeoff between privacy and utility. The advantage of our method with respect to the analytical methods from the literature is twofold:
  - we can handle much larger datasets, and
  - we can handle complicated background knowledge and correlation between data points.

- Although our approach is inspired by the GANs paradigm, it departs significantly from it. In particular, in our case, the distribution has to be “invented” rather than “imitated”. This implies that we have to come up with different techniques for evaluating a distribution. To achieve our goal, we propose a new method based on the mutual information between the supervised and the predicted class labels.

- We apply our method to the case of location privacy. We craft some experiments to show in detail how our approach works and how it can achieve optimality. Depending on the distribution of data and on the utility constraint, the resulting noise function may achieve the maximal privacy, i.e., equivalent to that of random guessing. Trivially, in this case it also achieves the optimal tradeoff between privacy and utility.

- We evaluate the obfuscation mechanism produced by our method on real location data from the Gowalla dataset, and compare the privacy and utility of our approach with that of the planar Laplacian, one of the most popular mechanisms used in location privacy. First we do the comparison using the adversarial classifier in our architecture, obtaining favorable results. Then, we confirm these results (and hence the advantages of our method) by means of the (ideal) optimal Bayesian classifier, which represents the strongest possible adversary.

- We have made publicly available the code of the implementation and the experiments at the URL https://gitlab.com/MIPAN/mipan.

\section*{B. Related work}

Optimal mechanisms, namely mechanisms providing an optimal compromise between utility and privacy, have attracted the interest of many researchers. Many of the studies so far have focused on analytic methods based on linear optimization \cite{1}, \cite{2}, \cite{18}, \cite{19}. Although they can provide exact solutions, the high complexity of linear optimization limits the scalability of these methods. Our approach, in contrast, using the highly-efficient optimization process of neural networks (the gradient descent), does not suffer from this drawback.

The paper that is closest to ours is \cite{20}. Its authors also propose a GAN-based method to construct mechanisms providing an optimal privacy-utility trade-off, and they consider notions of privacy and utility similar to ours. The main difference is that the target function they use in the GAN is the cross entropy rather than the mutual information (they consider mutual information to measure privacy, but in the
implementation they approximate it with the log loss, that is the expected cross entropy). Hence the convergence of their method may be problematic, at least when applied to our case study, due to the “swapping effect” described in Example 1. We have actually experimented with use cross entropy as target function on our examples in Section IV, and the results were unsatisfactory, in the sense that due to the lack of convergence of $C$ the resulting mechanisms were unstable and the level of privacy protection was poor.

One of the side contributions of our paper is a method to compute mutual information in neural network (cfr. Section III). Recently, Belghazi et al. have proposed MINE, an efficient method to neural estimation of mutual information [21], inspired by the framework of [22] for the estimation of a general class of functions representable as $f$-divergencies. These methods work also in the continuous case and for high-dimensional data. In our case, however, we are dealing with a discrete domain, and we can compute directly and exactly the mutual information. Another reason for developing our own method is that we need to deal with a loss functions that contain not only the mutual information, but also a component representing utility, and depending on the notion of utility the result may not be an $f$-divergence.

Our paradigm has been inspired by the GANs [3], but it comes with some fundamental differences:

- $C$ is a classifier performing re-identification while in the GANs there is a discriminator able to distinguish a real data distribution from a generated one;
- in the GANs paradigm the generator network tries to reproduce the original data distribution to fool the discriminator. A huge difference is that, in our adversarial scenario, $G$ does not have a model distribution to refer to. The final data distribution only depends on the evolution of the two networks over time and it is driven by the constraints imposed in the loss functions that rule the learning process.
- we still adopt a training algorithm which alternates epochs of training of $G$ and epochs of training of $C$, but as we will show in Section III, it is different from the one adopted for GANs.

### II. OUR SETTING

We formulate the privacy-utility optimization problem using a framework similar to that of [23]. We consider four random variables, $X, Y, Z, W$, ranging over the sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ and $\mathcal{W}$ respectively, with the following meaning:

- $X$: the sensitive information that the users wishes to conceal,
- $W$: the useful information with respect to some service provider and to the intended notion of utility,
- $Z$: the information made visible to the service provider, which may be intercepted by some attacker, and
- $Y$: the information inferred by the attacker.

We assume a fixed joint distribution (data model) $P_{X,W}$ over the users’ data $\mathcal{X} \times \mathcal{W}$. We present our framework assuming that the variables are discrete, but all results and definitions can be transferred to the continuous case, by replacing the distributions with probability density functions, and the summations with integrals. For the initial definitions and results of this section $\mathcal{X}$ and $\mathcal{Y}$ may be different sets. Starting from Section III we will assume that $\mathcal{X} = \mathcal{Y}$.

An obfuscation mechanism can be represented as a conditional probability distribution $P_{Z|W}$, where $P_{Z|W}(z|w)$ indicates the probability that the mechanism transform the data point $w$ into the noisy data point $z$. We assume that $Z$ are the only attributes visible to the attacker and to the service provider. The goal of the defender $G$ is to optimize the data release mechanism $P_{Z|W}$ so to achieve a desired level of utility while minimizing the leakage of the sensitive attributes $X$. The goal of the attacker $C$ is to retrieve $X$ from $Z$ as precisely as possible. In doing so, it produces a classification $P_{Y|Z}$ (prediction).

Note that the four random variables form a Markov chain:

$$X \leftrightarrow W \leftrightarrow Z \leftrightarrow Y.$$  

Their joint distribution is completely determined by the data model, the obfuscation mechanism and the classification:

$$P_{X,W,Z,Y}(x,w,z,y) = P_{X,W}(x,w)P_{Z|W}(z|w)P_{Y|Z}(y|z).$$

From $P_{X,W,Z,Y}$ we can derive the marginals, the conditional probabilities of any two variables, etc. For instance:

$$P_X(x) = \sum_w P_{X,W}(x,w).$$  

$$P_Z(z) = \sum_{x,w} P_{X,W}(x,w)P_{Z|W}(z|w).$$

$$P_{Z|X}(z|x) = \sum_w P_{X,W}(x,w)P_{Z|W}(z|w) \cdot \frac{P_X(x)}{P_Z(z)}.$$  

$$P_{X|Z}(x|z) = \frac{P_{Z|X}(z|x)P_X(x)}{P_Z(z)}.$$  

**TABLE I: Table of symbols**

| Symbol | Description |
|--------|-------------|
| $C$    | Classifier network (attacker). |
| $G$    | Generator network. |
| $X,\mathcal{X}$ | Sensitive information. (Random var. and domain.) |
| $W,\mathcal{W}$ | Useful information with respect to the intended notion of utility. |
| $Z,\mathcal{Z}$ | Obfuscated information accessible to the service provider and to the attacker. |
| $Y,\mathcal{Y}$ | Information inferred by the attacker. |
| $P_{X,W}$ | Mutual information between two random variables. |
| $P_{Z|W}$ | Joint probability of two random variables. |
| $P_Y$ | Conditional probability. |
| $\mathbb{L}[Z|W]$ | Utility loss induced by the obfuscation mechanism. |
| $L$ | Threshold on the utility loss. |
| $H(\cdot)$ | Entropy of a random variable. |
| $H(\cdot|\cdot)$ | Conditional entropy. |
| $I(\cdot;\cdot)$ | Mutual information between two random variables. |
The latter distribution, $P_{X|Z}$, is the posterior distribution of $X$ given $Z$, and will play an important role in the following sections.

A. Quantifying utility

Concerning the utility, we consider a loss function $\ell : W \times Z \rightarrow [0, \infty)$, where $\ell(w, z)$ represents the utility loss caused by reporting $z$ when the true value is $w$.

**Definition 1** (Utility loss). The utility loss from the original data $W$ to the noisy data $Z$, given the loss function $\ell$, is defined as the expectation of $\ell$:

$$L[Z \mid W, \ell] = \mathbb{E}[\ell \mid W, Z] = \sum_{w,z} P_{W,Z}(w,z)\ell(w, z). \quad (6)$$

We will omit $\ell$ when it is clear from the context. Note that, given a data model $P_{X,W}$, the utility loss can be expressed in terms of the mechanism $P_{Z|W}$:

$$L[Z \mid W] = \sum_{x,w} P_{X,W}(x,w)P_{Z|W}(z \mid w)\ell(w, z). \quad (7)$$

Our goal is to build a privacy-protection mechanism that keeps the loss below a certain threshold $L$. We denote by $M_L$ the set of such mechanisms, namely:

$$M_L \overset{\text{def}}{=} \{ P_{Z|W} \mid L[Z \mid W] \leq L \}. \quad (8)$$

The following property is immediate:

**Proposition 1** (Convexity of $M_L$). The set $M_L$ is convex and closed.

B. Quantifying privacy as mutual information

As explained in the introduction, the privacy leakage with respect to an attacker $C$ will be quantified by the mutual information $I(X;Y)$, which represents the correlation between $X$ and the classification $Y$ produced by $C$ which the obfuscation mechanism created by $G$ tries to minimize. Let us remark once more that considering the mutual information instead than the classification precision allows us to capture the amount of information provided by $Y$ about $X$, and therefore the largest possible class of adversaries given the information learned by $C$, not just $C$. As pointed out in the introduction, this choice makes the training of $G$ more stable, since it is learning to defeat not only $C$ but all the adversaries that can be derived from the information available through $C$. In other words, $G$ cannot simply try to win the game by swapping around the labels of the classification learned by $C$; it needs to reduce the information that reaches $C$.

To this purpose, we consider the following information-theoretic functions:

**Entropy**

$$H(X) = -\sum_x P_X(x) \log P_X(x), \quad (9)$$

**Residual Entropy**

$$H(X|Y) = -\sum_{x,y} P_{X,Y}(x,y) \log P_X(x|y), \quad (10)$$

**Mutual Information**

$$I(X;Y) = H(X) - H(X|Y), \quad (11)$$

C. Formulation of the game and equilibrium strategies

The game that $G$ and $C$ play corresponds to the following minimax formulation:

$$\min_G \max_C I(X;Y) \quad (12)$$

where the minimization by $G$ is on the mechanisms $P_{Z|W}$ ranging over $M_L$, while the maximization by $C$ is on the classifications $P_{Y|Z}$.

Note that $P_{Z|W}$ can be seen as a stochastic matrix and therefore as an element of a vector space. An important property for our purposes is that the mutual information is a property for our purposes is that the mutual information is a convex function of convex coefficients $c$.

**Proposition 2** (Convexity of $I$). Given $P_{X,W}$ and $P_{Y|Z}$, let $f(P_{Z|W}) = I(X;Y)$. Then $f$ is convex. Namely, given a pair of convex coefficients $c_1$ and $c_2$, and two mechanisms $P_{Z|W}^{(1)}$ and $P_{Z|W}^{(2)}$, we have:

$$f(c_1 P_{Z|W}^{(1)} + c_2 P_{Z|W}^{(2)}) \leq c_1 f(P_{Z|W}^{(1)}) + c_2 f(P_{Z|W}^{(2)}). \quad (13)$$

Proposition 1 and 2 show that this problem is well defined: for any choice of $C$, $I(X;Y)$ has a global minimum in $M_L$, and no strictly-local minima.

We note that, in principle, one could avoid using the GAN paradigm, and try to achieve the optimal mechanism by solving, instead, the following minimization problem:

$$\min_G I(X;Z) \quad (14)$$

where $\min_G I(X;Z)$ is meant, as before, as a minimization over the mechanisms $P_{Z|W}$ ranging over $M_L$. This approach would have the advantage that it is independent from the attacker, so one would need to reason only about $G$ (and there would be no need for a GAN).

The main difference between $I(X;Y)$ and $I(X;Z)$ is that the latter represents the information about $X$ available to any adversary, not only those that are trying to retrieve $X$ by building a classifier. This fact reflects in the following relation between the two formulations:

**Proposition 3.**

$$\min_G \max_C I(X;Y) \leq \min_G I(X;Z)$$

Note that, since $\min_G I(X;Z)$ is an upper bound of our target, it imposes a limit on $\max_C I(X;Y)$.

On the other hand, there are some advantages in considering $\min_G \max_C I(X;Y)$ instead of $\min_G I(X;Z)$: first of all, $Z$ may have a much larger and more complicated domain than $Y$, so performing the gradient descent on $I(X;Z)$ could be unfeasible. Second, if we are interested in considering only classification-based attacks, then $\min_G \max_C I(X;Y)$ should give a better result than $\min_G I(X;Z)$. In this paper we focus on the former, and leave the exploration of an approach based on $\min_G I(X;Z)$ as future work.
D. Measuring privacy as Bayes error

As explained in the introduction, we intend to evaluate the resulting mechanism also in terms of Bayes error. Here we give the relevant definitions and properties.

**Definition 2 (Bayes error).** The Bayes error of $X$ given $Y$ is:

$$B(X \mid Y) = \sum_y P_Y(y)(1 - \max_x P_{X\mid Y}(x \mid y)).$$

Namely, the Bayes error is the expected probability of “guessing the wrong id” of an adversary that, when he sees that $C$ produces the id $y$, it guesses the id $x$ that has the highest posterior probability given $y$.

The definition of $B(X \mid Z)$ is analogous. Given a mechanism $P_{Z\mid W}$, we can regard $B(X \mid Y)$ as a measure of privacy of $P_{Z\mid W}$ w.r.t. one-try [12] classification-based attacks, whereas $B(X \mid Z)$ measures privacy w.r.t. any one-try attack. The following proposition shows the relation between the two notions.

**Proposition 4.**

$$B(X \mid Z) \leq B(X \mid Y)$$

Next, we propose an implementation via GANs of the method illustrated above.

III. IMPLEMENTATION IN NEURAL NETWORKS

In this section we illustrate the network architecture and the training algorithm that we propose to implement the interplay between $G$ and $C$. The scheme of our adversarial game is illustrated in Fig. 2, where the meaning of the symbols is as follows:

- $x, y, z$ and $w$ are instances of the random variables $X, Y, Z$ and $W$ respectively, whose meaning is described in previous section. In this section, and in the rest of the paper, we assume that the domains of $X$ and $Y$ coincide, namely $\mathcal{X} = \mathcal{Y}$.
- $s$ (seed) is a randomly-generated number in $[0, 1)$.
- $g$ is the function learnt by $G$, and it represents an obfuscation mechanism $P_{Z\mid W}$. The input $s$ provides the randomness needed to generate random noise. It is necessary because a neural network in itself is deterministic.
- $c$ is the classification learnt by $C$, and it corresponds to $P_{Y\mid Z}$.

The evolution of the adversarial network is described in Algorithm 1. $C$ and $G$ are trained at two different moments within the same adversarial training iteration. In particular $C_i$ is obtained by training the network $C$ against the noise generated by $G_{i-1}$ and $G_i$ is obtained by fighting against $C_i$.

Note that in our method each $C_i$ is trained only on the output from $G_{i-1}$. This is a main difference with respect to the GANs paradigm, where the discriminator is always trained both over the output of the generator and the target distribution. Another particularity of our method is that at the end of the $i$th iteration, while $G_i$ is retained for the next iteration, $C_i$ is discarded and the classifier is reinitialized to the base one $C_0$. The reason is that we have found out experimentally that, if $C_i$ is not reset, it might take a long time before it becomes able to contrast the noise injection at iteration $(i+1)$-th. In fact $C_i$ has been trained on data generated by $G_{i-1}$ and it has consistently updated its weights according to that. It could take a long time for $C_i$ to “forget” what it has learned and move on to learning to beat the new noise distribution. On the contrary, when the classifier is reset to $C_0$, it quickly adapts to the data produced by the new generator.

We describe now in more detail some key implementation choices of our proposal.

A. Implementation details: Base models

The base model $C_0$ is simply the “blank” classifier that has not learnt anything yet. As for $G_0$, we have found out experimentally that it is convenient to start with a noise

---

**Algorithm 1:** Adversarial algorithm with classifier reset.

```plaintext
Data: train_data // Training data
Models: G_i generator evolution at the i-th step;
        C_i classifier evolution at the i-th step.

train(n, d) trains the network $n$ on the data $d$.
generator(data) outputs a noisy version of data.
classifier = load model $C_0 = C_0$
generator = load model $G_0 = G_0$
i = 0
while True do
  // Let $P_{Z\mid W}$ be the mechanism produced by the current generator.
i += 1
  classifier = train(classifier, generator(train_data))
  // Train the classifier to produce a $P_{Y\mid Z}$ that approximates $P_{X\mid Z}$.
  adv network = generator + classifier in cascade
  adv network = train(adv network, train_data)
  // Train the network to produce a new $P_{Z\mid W}$ by reducing $\text{Loss}_G$.
  The weights of the classifier are frozen during this phase.
generator = generator layer in adv network = $G_i$
  // Save generator evolution for next iteration.
classifier = load model adv_network = $G_i$
  // Reset classifier.
end
```

---

Fig. 2: Scheme of the adversarial nets for our setting.
function pretty much spread out. This is because in this way the generator has more data points with non-null probability to consider, and can figure out faster which way to go to minimize the mutual information.

B. Implementation details: Utility

The utility constraint is incorporated in the loss function of \( G \) in the following way:

\[
Loss_G = \alpha \times Loss_{utility} + \beta \times I(X;Y),
\]

where \( \alpha \) and \( \beta \) are parameters that allow us to tune the trade-off between utility and privacy. The purpose of \( Loss_{utility} \) is to ensure that the constraint on utility is respected, i.e., that the obfuscation mechanism that \( G \) is trying to produce stays within the domain \( M_L \). We recall that \( M_L \) represents the constraint \( L[Z \mid W] \leq L \) (cfr. (8)). Since we need to compute the gradient on the loss, we need a derivable function for \( Loss_{utility} \). We propose to implement it using \texttt{softplus}, which is a function of two arguments in \( \mathbb{R} \) defined as:

\[
\text{softplus}(a,b) = \ln(1 + e^{a-b}).
\]

This function is non negative, monotonically increasing, and its value is close to 0 for \( a < b \), while it grows very quickly for \( a > b \). Hence, we define

\[
Loss_{utility}(P_Z|W) = \text{softplus}(L[Z \mid W], L).
\]

With this definition, \( Loss_{utility} \) does not interfere with \( I(X;Y) \) when the constraint \( L[Z \mid W] \leq L \) is respected, and it forces \( G \) to stay within the constraint because its growth when the constraints is not respected is very steep.

C. Implementation details: Mutual Information

One of the characteristics of our work is that the loss function is based on the mutual information between the supervised and the predicted class labels. We show how it is computed via an example. Assume we are at the \( i \)-th iteration. Let us consider the scenario illustrated in Table II, that describes a classification problem over three classes A, B, and C, and six samples. The \( X \) and \( W \) columns represent the true values of the data, and the \( Z \) column represents the obfuscated version of \( W \), generated by \( G_i \). The \( Y = \cdot \) columns represent the labels predicted by \( G_i \) on the \( Z \) values of the samples. These are the result of the training of the classifier on the output of \( G_{i-1} \), the generator at the previous iteration. The numbers in the \( Y = \cdot \) columns represent the confidence of the prediction (as produced by the softmax activation functions), and can be interpreted as probabilities.

In order to compute the mutual information we need to calculate \( H(X) \) and \( H(X \mid Y) \), using their defining formulae (9) and (10), and then apply (11). To this purpose, we just need to find out the joint distribution \( P_{X,Y} \), since the marginals \( P_X \) and \( P_Y \), and the conditional probability \( P_{X \mid Y} \), can be then obtained from \( P_{X,Y} \) in the standard way. Now, the value of \( P_{X,Y}(x,y) \) can be computed as the average of the probabilities over the samples. Specifically, we obtain: \( P_{X,Y}(A,A) = 2 \times 0.7/6 \), \( P_{X,Y}(A,B) = 2 \times 0.3/6 \), \( P_{X,Y}(A,C) = 2 \times 0.5/6 \), \( P_{X,Y}(B,A) = 3 \times 0.7/6 \), \( P_{X,Y}(B,B) = 3 \times 0.3/6 \), \( P_{X,Y}(B,C) = 3 \times 0.5/6 \), \( P_{X,Y}(C,A) = 3 \times 0.5/6 \), \( P_{X,Y}(C,B) = 3 \times 0.5/6 \), \( P_{X,Y}(C,C) = 3 \times 0.5/6 \).

The global \( F1 \)-score is defined as the weighted average of the \( F1 \)-score’s for each class. In the simplest case, when the

\[
\text{Accuracy} \overset{\text{def}}{=} \frac{i_{00} + i_{11} + i_{22}}{\sum_{j,k=0}^2 i_{jk}}.
\]

The \( F1 \)-score involves the notion of precision and recall. These are defined on each class:

\[
\text{Precision}_j \overset{\text{def}}{=} \frac{i_{jj}}{i_{0j} + i_{1j} + i_{2j}},\quad \text{Recall}_j \overset{\text{def}}{=} \frac{i_{jj}}{i_{j0} + i_{j1} + i_{j2}}.
\]

\[
F1_{\text{score}}_j \overset{\text{def}}{=} 2 \times \frac{\text{Precision}_j \times \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j}.
\]

## Table II: Classification problem with three classes A, B, C.

| Predicted labels | True labels |
|------------------|-------------|
| Class 0          | i00, i01, i02 |
| Class 1          | i10, i11, i12 |
| Class 2          | i20, i21, i22 |

## Table III: Confusion matrix for a 3-class classification problem.

\[
P_{X,Y}(A,C) = 0, \quad P_{X,Y}(B,A) = 0.7/6, \quad P_{X,Y}(B,B) = 0.3+0.5/6, \quad P_{X,Y}(B,C) = 0.5/6, \quad P_{X,Y}(C,B) = 0.5/6 \quad \text{and} \quad P_{X,Y}(C,C) = 0.5+1/6.
\]

In order to make the function work with the backpropagation algorithm, we implemented all the steps using TensorFlow and Keras native functions: summation, multiplication, logarthim, condition, while loop.

D. Implementation details: Metrics to evaluate the classification outcome

To evaluate the quality of the classification produced by the \( C \) network, we rely on two metrics: \textit{accuracy} and \textit{F1-score}.

To explain their meaning, let us consider a three class problem and the corresponding confusion matrix in table III, where the main diagonal represents the number of matches between the true labels and the predicted ones, while all the other cells represent the mismatches.

The accuracy\(^4\) is:

\[
\text{Accuracy} \overset{\text{def}}{=} \frac{i_{00} + i_{11} + i_{22}}{\sum_{j,k=0}^2 i_{jk}}.
\]

The \( F1 \)-score involves the notion of precision and recall. These are defined on each class:

\[
\text{Precision}_j \overset{\text{def}}{=} \frac{i_{jj}}{i_{0j} + i_{1j} + i_{2j}}, \quad \text{Recall}_j \overset{\text{def}}{=} \frac{i_{jj}}{i_{j0} + i_{j1} + i_{j2}}.
\]

\[
F1_{\text{score}}_j \overset{\text{def}}{=} 2 \times \frac{\text{Precision}_j \times \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j}.
\]

\(^4\)For simplicity we give the definition for the case in which the elements of the dataset are equally distributed among all the classes. If this is not the case, the definition of accuracy is more complicated.
classes in the dataset are balanced, we can define it as the simple average. In the above example it would be:

\[
F1_{score} \overset{\text{def}}{=} \frac{1}{3} \sum_{j=0}^{2} F1_{score_j},
\]

(22)

In case the classes are unbalanced more elaborated definitions should be applied. Typically, we could assign to each class a weight inversely proportional to the frequency of the class.

Both the accuracy and the \(F1_{score}\) take values \(\in [0,1]\). It is important to consider both of them to avoid issues like the accuracy paradox: a classifier always predicting the most frequent class could be accurate but it could totally neglect those classes which are not very frequent. Furthermore, even in case of balanced datasets, considering the \(F1_{score}\), and therefore the precision and recall, may be useful in case of low accuracy, to understand if all the classes are misclassified or some are wrongly predicted more often than others.

E. Implementation details: number of epochs and batch size

The convergence of the game can be quite sensitive to the number of epochs and batch size. We refer to literature [24] for a general discussion about the impact they have on learning. It is important to note that:

- Choosing a batch size too small for training \(G\) might result in a constraint on the utility too strict. In fact, since the utility loss is an expectation, a larger number of samples makes it more likely that some points are pushed further from the threshold, taking advantage of the fact that their loss may be compensated by other data points for which the loss is small.
- Training \(C\) for too few epochs might result into a too weak adversary. On the other hand if it is trained for a long time we should make sure that the classification performances do not drop over the validation and test set because that might indicate an overfitting problem.

F. On the convergence of our method

In principle, at a each iteration \(i\), our method relies on the ability of the network \(G_i\) to improve the obfuscation mechanism starting from the one produced by \(G_{i-1}\), and given only the original locations and the model \(C_i\), which are used to determine the direction of the gradient for \(\text{Loss}_G\). The classifier \(C_i\) is a particular adversary modeled by its weights and its biases. However, thanks to the fact that the main component of \(\text{Loss}_G\) is \(I(X;Y)\) and not the the cross entropy, \(G_i\) takes into account all the attacks that would be possible from \(C_i\)’s information. We have experimentally verified that indeed, using the mutual information rather than the cross entropy, determines a substantial improvement on the convergence process, and the resulting mechanisms provide a better privacy (for the same utility level). Again, the reason is that the the cross entropy would be subject to the “swapping effect” illustrated by Example 1 in the introduction.

Another improvement on the convergence is due the fact that we reset the classifier to the initial weight setting \((C_0)\) at each iteration, instead than letting \(C_i\) evolve from \(C_{i-1}\). We have experimentally verified that the precision of \(C_i\) improves more rapidly if we start from an “agnostic” situation rather than from the knowledge accumulated in previous steps. This is, intuitively, due to the fact that the noise produced by \(G\) may change substantially from an iteration to the next, hence \(C_i\) would have to “unlearn” the now obsolete information of \(C_{i-1}\), if the latter were its starting point.

The loss function that the network \(G\) has to minimize, \(\text{Loss}_G\), is a combination of a softplus for the utility loss (which is convex) and the mutual information which is convex in \(P_{Z|W}\), i.e., our obfuscation mechanism. This means that there are only global minima, although there can be many of them, all equivalent. Therefore for sufficiently small updates, the noise distribution modeled by \(P_{Z|W}\) converges to one of these optima, provided that the involved network has enough capacity to compute the gradient descent involved in the training algorithm. In practice, however, the network \(G\) represents a limited family of noise distributions, and instead of optimizing the noise distribution itself we optimize the weights of this network, which introduces multiple critical points in the parameter space.

IV. Experiments on synthetic data

In this section we illustrate some experiments on artificial data to show how the proposed method works. We consider the issue of location privacy, namely, we assume a set of users who want to protect their identities, but need to disclose (an obfuscated version of) their locations, for some utility purpose. Following Definition 6, the utility loss is the expected distortion introduced by the obfuscation mechanism, i.e., the expected distance between the true location and the obfuscated one. To keep things simple, we consider just four users. In order to be consistent with the experiments on real data, we assume the same set of locations of the next section, namely locations in an area of Paris.

More precisely, with reference to the setting in Section II, we instantiate the domains of the random variables \(X, W, Z,\) and \(Y,\) and the loss function, as follows:

- \(\mathcal{X} = \mathcal{Y} = \{\text{blue}, \text{red}, \text{green}, \text{yellow}\}\), the identities of 4 users.
- \(\mathcal{Z} = \mathcal{W}\) = the locations in a squared region of \(6.5 \times 6.5\) sq km centered in 5, Boulevard de Sébastopol, Paris. Each location entry is defined by a pair of coordinates normalized in \([-1, 1]\).
- \(\ell(w, z) = \text{geographical distance}^5\) between \(w\) and \(z\).

The goal is to produce an obfuscation mechanism that achieves a good trade-off between utility and privacy. Though the various experiments, we will compare our method with that of the planar Laplace mechanism associated to \(\text{geo-indistinguishability}\) [5], which is rather popular in the domain of location privacy.

\(^5\)For simplicity in this paper we use the Euclidean distance rather than the Haversine distance. For such limited area they are quite close anyway.
A. The planar Laplace mechanism

We recall that the planar Laplace probability density function in the point \( z \), given that the true location is \( w \), is defined as [5]:

\[
L_w^\epsilon(z) = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(w,z)},
\]

where \( d(w,z) \) is the Euclidean distance between \( w \) and \( z \).

In order to compare the the above mechanism with our one, we need to tune the privacy parameter \( \epsilon \) so that \( L^\epsilon \) has the same expected distortion as the upper bound \( L \) to the utility loss applied in our method, and then confront the privacy degree of the two corresponding mechanisms. To this purpose, we recall that the expected distortion \( \mathbb{E}[Z \mid W] \) introduced by the planar Laplace depends only on \( \epsilon \) (not on the prior \( P_W \)), and it is given by:

\[
\mathbb{E}[Z \mid W] = \sum_{w,z} L_w(z)P_W(w)d(w,z) = \frac{2}{\epsilon}.
\]

B. Bayes error estimation

As explained in Section II, besides \( I(X;Y) \) we will also use the Bayes error \( B(X \mid Z) \) to evaluate the level of protection offered by a mechanism.

To this purpose, we discretize \( Z \) by creating a grid over the \( 6.5 \times 6.5 \) sq km region, thus determining a partition of the region into a number of disjoint cells. We will create different grid settings to see how the partition affects the Bayes error. In particular, we will consider the cases where the side of a grid is \( 5 \)m, \( 10 \)m, \( 100 \)m and \( 500 \)m long, which corresponds to \( 260 \times 260 = 67600, 130 \times 130 = 16900, 65 \times 65 = 4225 \) and \( 13 \times 13 = 169 \) cells, respectively.

We will also run various experiments with different numbers of obfuscated locations (hits) created for each original one. Specifically, for each grid we will consider the cases of \( 5, 10, 20, 50 \) and \( 260 \) hits for each original one.

Each hit falls in exactly one cell. Hence, we can estimate the probability that a hit is in cell \( i \) as:

\[
P(\text{cell}_i) \overset{\text{def}}{=} \frac{\text{number of hits in cell}_i}{\text{total number of hits}},
\]

and the probability that a hit in cell \( i \) belong to class \( j \) as:

\[
P(\text{Class}_j|\text{cell}_i) \overset{\text{def}}{=} \frac{\text{number of hits of class}_j \text{ in cell}_i}{\text{number of hits in cell}_i}.
\]

We can now estimate of the Bayes error as follows:

\[
B(X \mid Z) \overset{\text{def}}{=} 1 - \sum_{i=0}^{k-1} \max_j P(\text{Class}_j|\text{cell}_i)P(\text{cell}_i)
\]

where \( k \) is the total number of cells.

It is important to stress the fact that the results of these computations are influenced by the chosen grid. In particular we have two extreme cases:

- when the grid consists of only one cell the Bayesian error is \( 1 - \frac{1}{k} = k - \frac{1}{k} \) for any obfuscation mechanism \( P_{Z|W} \).

- when the number of cells is large enough so that each cell contains at most one hit, then Bayesian error is 0 for any obfuscation mechanism.

In general, a coarser granularity is a source of additional Bayes error, independently from the obfuscation mechanisms. A finer granularity guarantees higher discrimination power, especially when we compare methods which scatter the obfuscated locations in different regions of the domain.

C. The synthetic dataset

The synthetic dataset consists of 600 locations (true locations) for each of the 4 users (classes) blue, red, green and yellow. Hence we are in a situation of balanced classes, and in total we have 2400 entries. The locations of each user are placed around one of the vertices of a square of \( 300 \times 300 \) sq meters centered in 5, Boulevard de Sébastopol, Paris. (Each user corresponds to a different vertex.) They are randomly generated so to form a cloud of 600 entries around each vertex and in such a way that no locations falls further than about 45m from the corresponding vertex. Note that these distributions determine the random variables \( X \) and \( W \), and their correlation.

For each user, 480 locations will be used for training and validation purposes, whilst 120 will be used for testing (480 + 120 = 600). Therefore there are 1920 locations in the set used for training and validation, and 480 for testing. These sets are represented in Fig. 13(b) and Fig. 3(b), respectively.

D. Experiment 1: Synthetic data, relaxed utility constraint

As a first experiment, we choose for the upper bound \( L \) on the expected distortion a value high enough so that in principle we can achieve the highest possible privacy, which is obtained when the obfuscated location observed by the attacker gives no information about the true location, which means that \( I(X;Y) = 0 \). When this is the case, the best the attacker can do is random guessing. Since we have 4 users, the Bayes risk is \( B(X \mid Y) = 1 - \frac{1}{4} = 0.75 \).

One way to achieve the maximum privacy is to map all the locations into the same obfuscated location in the middle. To compute a sufficient \( L \), consider that the vertices where the locations of the users are placed form a square of side 300m, hence each vertex is at a distance \( 300 \times \sqrt{2}/2 \approx 212 \)m from the center. Taking into account that the locations can be as much as 45m away from the corresponding vertex, we conclude that any value of \( L \) larger than 212 + 45 = 247m should be enough to allow us to obtain the maximum privacy. Just to be sure, we set the upper bound on the distortion to be

\[
L = 270m,
\]

but we will see from the experiments that a much smaller value of \( L \) would have been sufficient.

We now need to tune the planar Laplace so that the expected distortion is at least \( L \). We decide to set:

\[
\epsilon = \frac{\ln 2}{100}
\]
which, using Equation (24), gives us a value

\[ L[Z \mid W] \approx 288 \ln 2 > L. \] (30)

We have actually used this instance of the planar Laplace as a starting point of our method: we have defined \( G_0 \) as \( L^* \) with \( \epsilon = \ln 2/100 \). For the following steps, the networks \( G_i \) and \( C_i \) are constructed as explained in Algorithm 1. In particular, we train the generator with a batch size of 128 samples for 100 epochs during each iteration. The learning rate is set to 0.0001. For this particular experiment we set the weight for the utility loss to 1 and the weight for the mutual information to 2. The classifier is trained with a batch size of 512 samples and 3000 epochs for each iteration. The learning rate for the classifier is set to 0.001.

Around iteration \( i = 149 \) the accuracy of the network \( C_i \) tends to a value which is \( \approx 0.25 \) both over the training and the validation set. This means that \( C_i \) just randomly predicts one of the four classes. This happens because the noise injection results to be effective, and the \( C_i \) cannot learn anything from the training set. Since 0.75 is the maximum possible Bayes risk, we then know that we can stop.

The result of the application of the planar Laplace to the testing set is illustrated in Fig. 3(a). In the appendix we report also the result on the testing set (Fig. 13(a)). The empirical distortion (i.e., the distortion computed on the sampled obfuscated locations) is \( \approx 282.07 \) for the training data and \( \approx 298.40 \) for the testing data, respectively, which is in line with the theoretical distortion formulated in (30).

The result of the application of our method, i.e. the final generator \( G_i \), to the testing and training sets, are reported in Fig. 3(c) and 13(c), respectively. The empirical distortion for the training and testing sets is \( \approx 219.70 \) and \( \approx 219.26 \), respectively. This is way below the limit of 270 set in (28), and it is due to the fact that to achieve the optimum privacy we probably do not need more than \( \approx 220 \). In fact, the distance of the vertices from the center is \( \approx 212 \), and even though some locations are further away (up to 45 more), there are also locations that are closer, and that compensate the utility loss (which is a linear average measure).

From Fig. 3 and 13 we can see that, while the Laplace tends to “spread out” the obfuscated locations, our method tends to concentrate them into a small area. This may not be possible in general, as it depends on the utility constraint. Nevertheless, we can expect that our mechanism will tend to overlap the obfuscated locations of different classes, as much as allowed by the utility constraint. With the Laplace, on the contrary, the areas of the various classes remain pretty separated. This is reflected by the Bayes error estimation reported in Fig. 5.

We note that the Bayes error of the planar Laplace tend to decrease as the grid becomes finer. We believe that this is due to the fact that, with a coarse grid, there is an effect of confusion simply due to the large size of each cell. We remark that the behavior of our noise, on the contrary, is quite stable. It is also interesting to note that, when the grid is very coarse (13 \( \times \) 13 cells) the Bayes error is 0.75 already on the original data (cfr. Fig. 4), which must be due to the fact that all the vertices are in the same cell. While the Bayes error remains 0.75 also with our obfuscation mechanism, with the Laplacian it decreases to 0.60. The reason is that the noise scatters the locations in different cells, and they become, therefore distinguishable.

E. Experiment 2: Synthetic data, stricter utility constraint

We are now interested in investigating how our method behave when a stricter constraint on the utility loss is imposed. In order to do so, we run an experiment similar to the one in Section IV-D. We repeat the same steps but now we set \( L \) and the privacy parameter (and consequently the distortion rate) of the planar Laplace as follows:

\[ L = 173m \quad \epsilon = \frac{\ln 2}{60} \quad L[Z \mid W] \approx 173.12 \] (31)

The result of the application of the Laplace mechanism is illustrated in Fig. 6(a) (testing data) and 14(a) (training data), respectively. The empirical distortion is \( \approx 170.53 \) for the training data, and \( \approx 172.35 \) for the testing data.

Following the same pattern as in Section IV-D, we train an adversarial network \( G \) and \( C \). The training of the network \( G \) is performed for 30 epochs during each iteration with a batch size of 512 samples and a learning rate of 0.0001. The classifier network \( C \) is trained for 3000 epochs during each iteration with a batch size of 512 samples and 0.001 as the value for the learning rate during each iteration. We are particularly interested in the 24th iteration where the network \( C \)’s performance is degraded by the obfuscation performed by the generator trained during the previous iteration. Training \( C \) with 32 samples batch size and learning rate set to 0.001 for 100 epochs with the obfuscated data gives the results reported in Table IV.

| Data         | Accuracy | F1 score |
|--------------|----------|----------|
| Training data| \( \approx 0.55 \) | \( \approx 0.54 \) |
| Validation data | \( \approx 0.53 \) | \( \approx 0.53 \) |
| Test data    | \( \approx 0.52 \) | \( \approx 0.51 \) |

TABLE IV: Summary of the experiment with our noise.

In this case, increasing the number of epochs does not improve the classification precision and makes the network \( C \) more prone to overfitting.

The obfuscation provided by the network \( G \) at the 24th iteration produces the distributions on the testing and training data illustrated in Fig. 6(c) and Fig. 14(c) respectively. The empirical distortion is \( \approx 166.78 \) (training data) and \( \approx 171.50 \) (testing data).

Finally, we have estimated the Bayes error for the two mechanisms. The results are in Fig. 7.
Fig. 3: Synthetic testing data. From left to right: Laplacian noise, no noise, our noise. $L = 270 \text{m}$.

Fig. 4: Estimation of $B(X \mid Z)$ on the original version of the synthetic data.

(a) Training data.  
(b) Testing data.

Fig. 5: Estimation of $B(X \mid Z)$ on the synthetic data for the Laplacian and for our mechanisms, with $L = 270 \text{m}$. The empirical utility loss is $\approx 282.07 \text{m} – 298.40 \text{m}$ for the Laplacian and $\approx 219.70 \text{m} – 219.26 \text{m}$ for ours.

(a) Training data.  
(b) Testing data.

Fig. 6: Synthetic testing data. From left to right: Laplacian noise, no noise, our noise. $L = 173 \text{m}$.

Fig. 7: Estimation of $B(X \mid Z)$ on the synthetic data for the Laplacian and for our mechanisms, with $L = 173 \text{m}$. The empirical utility loss is $\approx 170.53 \text{m} – 172.35 \text{m}$ for the Laplacian and $\approx 166.78 \text{m} – 171.50 \text{m}$ for ours.

| Number of cells | 13 x 13 | 65 x 65 | 130 x 130 | 260 x 260 |
|-----------------|---------|---------|------------|------------|
| Obs Lap         | 0.60    | 0.75    | 0.40       | 0.75       |
| Lap             | 0.75    | 0.40    | 0.75       | 0.75       |
| Our Lap         | 0.75    | 0.40    | 0.75       | 0.75       |
| Our             | 0.75    | 0.40    | 0.75       | 0.75       |

(a) Training data.  
(b) Testing data.
Fig. 8: Gowalla testing data. From left to right: Laplacian noise, no noise, our noise. \( L = 1150 \text{m} \).

| Number of cells | \( 13 \times 13 \) | \( 65 \times 65 \) | \( 130 \times 130 \) | \( 260 \times 260 \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Training data   | 0.12            | 0.06            | 0.04            | 0.03            |

Fig. 9: Estimation of \( B(X \mid Z) \) on the original version of the data from Gowalla.

| Number of cells | \( 13 \times 13 \) | \( 65 \times 65 \) | \( 130 \times 130 \) | \( 260 \times 260 \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| (a) Training data |                |                |                |                |
| (b) Testing data |                |                |                |                |

Fig. 10: Estimation of \( B(X \mid Z) \) on the Gowalla data for the Laplacian and for our mechanisms, with \( L = 1150 \text{m} \). The utility loss for training and testing data is \( \approx 1127.83 \text{m} - 1132.63 \text{m} \) for the Laplacian and \( \approx 961.38 \text{m} - 979.40 \text{m} \) for ours.

| Number of cells | \( 13 \times 13 \) | \( 65 \times 65 \) | \( 130 \times 130 \) | \( 260 \times 260 \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| (a) Training data |                |                |                |                |
| (b) Testing data |                |                |                |                |

Fig. 11: Gowalla testing data. From left to right: Laplacian noise, no noise, our noise. \( L = 518 \text{m} \).

| Number of cells | \( 13 \times 13 \) | \( 65 \times 65 \) | \( 130 \times 130 \) | \( 260 \times 260 \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| (a) Training data |                |                |                |                |
| (b) Testing data |                |                |                |                |

Fig. 12: Estimation of \( B(X \mid Z) \) on the Gowalla data for the Laplacian and for our mechanisms, with \( L = 518 \text{m} \). The utility loss for training and testing data is \( \approx 523.40 \text{m} - 535.21 \text{m} \) for the Laplacian and \( \approx 487.34 \text{m} - 502.89 \text{m} \) for ours.
V. Experiments on data from the Gowalla dataset

We present now similar experiments, but performed on real location data.

A. The dataset

The dataset consists of data extracted from the Gowalla dataset [25], a collection of check-ins made available by the Gowalla location-based social network. Among all the provided features, only the users’ identifiers (classes), the latitude and longitude of the check-in locations are considered. The data are selected as follows:

1) we consider a squared region centered in 5, Boulevard de Sébastopol, Paris, France with 4500m long side;
2) we select the 6 users who checked in the region most frequently, we retain their locations and discard the rest;
3) we filter the obtained locations to reduce the overlapping of the data belonging to different classes by randomly selecting for each class 82 location samples for training and validation purpose, and 20 samples for the test.

We obtain 492 locations to train and validate the model, and 120 to test it. Fig. 8(b) and 15(b) show the distribution of the testing and training data, respectively. Each color corresponds to a different class, i.e. a different user.

B. Experiment 3: Gowalla data, relaxed utility constraint

In this experiment we study the case of a large upper bound on the utility loss, which would potentially allow to achieve the maximum utility. We set $L$ and the privacy parameter (and consequently $L[\mathcal{Z} \mid \mathcal{W}]$) of the planar Laplace as follows:

$$L = 1150 \text{m} \quad \epsilon = \frac{\ln 2}{400} \quad L[\mathcal{Z} \mid \mathcal{W}] \approx 1154.15 \text{m} \quad (32)$$

The results of the application of the Laplace and of our method are illustrated in Fig. 8 (testing data) and 15 (training data). As we can see, the utility constraint is relaxed enough to allow our method to achieve the maximum privacy. As reported in Fig. 10, indeed, the Bayes error is close to that of random guess, namely $1 - \frac{1}{6} \approx 0.82$. Fig. 9 shows the part of the Bayes error due to the discretization of teh domain $\mathcal{Z}$. The planar Laplace, on the other hand, confirms the relatively limited level of privacy as observed in the synthetic data.

C. Experiment 4: Gowalla data, stricter utility constraint

We consider now a much tighter utility constraint, and we set the parameters of the planar Laplace as follows:

$$L = 518 \text{m} \quad \epsilon = \frac{\ln 2}{180} \quad L[\mathcal{Z} \mid \mathcal{W}] \approx 519.37 \text{m} \quad (33)$$

The results of the application of the Laplace and of our method are shown in Fig. 11 (testing data) and 16 (training data), and the Bayes error is reported in Fig. 12.

It is interesting to note that, as the grid becomes finer, both the planar Laplace and our method become more sensitive to the number of samples, in the sense that the (approximation of) the Bayes error grows considerably as the number of samples increases. We think that this is a phenomenon due to the way the Bayes error is estimated: when the cells are small they tend to have a limited number of hits, and therefore the number of hits whose class is not selected as the most present (in the given cell) is limited. Note that these less present hits are those that contribute to the Bayes error.

VI. Conclusion and future work

We have proposed a method based on adversarial nets to generate obfuscation mechanisms with a good tradeoff between privacy and utility. The crucial feature of our approach is that the target function to minimize is the mutual information rather than the cross entropy. We have applied our method to the case of location privacy, and experimented with a set of synthetic data and with data from Gowalla. We have compared the mechanism obtained via our approach with the planar Laplacian, the typical mechanism used for geo-indistinguishability, obtaining favorable results.

In the future we plan to explore other neural-network-based approaches to the construction of utility-aware privacy mechanisms. In particular, we intend to explore the possibility of using a loss function that depends on the intrinsic leakage of information of the mechanism, rather than on its vulnerability with respect to some adversary, and we intend to investigate the possibility of capturing different notions of privacy, such as differential privacy [14] and g-leakage [26].

Moreover we plan to enhance the flexibility of the constraint on distortion (in the loss function), by requiring it to be per user rather than global. More specifically, we aim at producing obfuscation mechanisms that satisfy constraints stating that the expected displacement for each user is at most up to a certain threshold. The motivation is that different users may have different requirements. Another potential application is to encompass a notion of fairness, that can be obtained by requiring that the threshold is the same for everybody.
REFERENCES

[1] R. Shokri, G. Theodorakopoulos, and C. Troncoso, “Privacy games along location traces: A game-theoretic framework for optimizing location privacy,” ACM Transactions on Privacy and Security, vol. 19, no. 4, pp. 11:1–11:31, 2017. [Online]. Available: http://doi.acm.org/10.1145/3099908

[2] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, “Geo-indistinguishable mechanisms for location privacy,” in Proceedings of the 21th ACM Conference on Computer and Communications Security (CCS 2014).

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[4] T. M. Cover and J. A. Thomas, Elements of Information Theory, 2nd ed. John Wiley & Sons, Inc., 2006.

[5] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, “Geo-indistinguishability: differential privacy for location-based systems,” in Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013). ACM, 2013, pp. 901–910. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516735

[6] K. Chatzikokolakis, E. ElSalamouny, and C. Palamidessi, “Efficient utility improvement for location privacy,” Proceedings on Privacy Enhancing Technologies (PoPETs), vol. 2017, no. 4, pp. 308–328, 2017.

[7] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith, “Axioms for information leakage,” in Proceedings of the 29th IEEE Computer Security Foundations Symposium (CSF), 2016, pp. 77–92. [Online]. Available: http://dx.doi.org/10.1109/CSF.2016.13

[8] Y. Zhu and R. Bettati, “Anonymity vs. information leakage in anonymity systems,” in Proc. of ICDCS. IEEE Computer Society, 2005, pp. 514–524.

[9] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden, “On the Bayes risk in information-hiding protocols,” Journal of Computer Security, vol. 16, no. 5, pp. 531–571, 2008. [Online]. Available: http://hal.inria.fr/inria-00349224/en/

[10] A. McIver, L. Meinicke, and C. Morgan, “Composite Closure for Bayes Risk in Probabilistic Noninterference,” in Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010), ser. Lecture Notes in Computer Science, S. Abramsky, C. Gavoille, C. Kirchner, F. M. de la Hette, and P. G. Spirakis, Eds., vol. 6199. Springer, 2010, pp. 223–235. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14162-1

[11] G. Cherubin, J. Hayes, and M. Juarez, “Website fingerprinting defenses at the application layer,” PoPETs, vol. 2017, no. 2, pp. 186–203, 2017. [Online]. Available: https://doi.org/10.11545/popec-2017-0023

[12] G. Smith, “On the foundations of quantitative information flow,” in Proceedings of the 12th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2009), ser. LNCS, L. de Alfaro, Ed., vol. 5504. York, UK: Springer, 2009, pp. 288–302.

[13] N. Santhi and A. Vardy, “An improvement over Rényi’s equivocation bound,” 2006, presented at the 44-th Annual Allerton Conference on Communication, Control, and Computing, September 2006. Available at http://arxiv.org/abs/cs/0608087

[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitive private data,” in In Proceedings of the Third Theory of Cryptography Conference (TCC), ser. Lecture Notes in Computer Science, S. Halevi and T. Rabin, Eds., vol. 3876. Springer, 2005, pp. 265–284.

[15] D. J. Mir, “Information-theoretic definitions of differential privacy,” in Proceedings of the 5th International Symposium on Foundations and Practice of Security (FPS), J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, A. Mirti, and N. Tawbi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 374–381.

[16] P. Cuff and L. Yu, “Differential privacy as a mutual information constraint,” in Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS), ser. CCS ’16. New York, NY, USA: ACM, 2016, pp. 43–54. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978308

[17] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi, “On the information leakage of differentially-private mechanisms,” Journal of Computer Security, vol. 23, no. 4, pp. 427–469, 2015. [Online]. Available: https://doi.org/10.3233/JCS-150528

[18] R. Shokri, “Privacy games: Optimal user-centric data obfuscation,” Proceedings on Privacy Enhancing Technologies, vol. 2015, no. 2, pp. 299–315, 2015.

[19] S. Oya, C. Troncoso, and F. Pérez-González, “Back to the drawing board: Revisiting the design of optimal location privacy-preserving mechanisms,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’17. ACM, 2017, pp. 1959–1972. [Online]. Available: http://doi.acm.org/10.1145/3139566.3134004

[20] A. Tripathy, Y. Wang, and P. Ishwar, “Privacy-preserving adversarial networks,” CoRR, vol. abs/1712.07008, 2017. [Online]. Available: http://arxiv.org/abs/1712.07008

[21] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, “Mutual information neural estimation,” in Proceedings of the 55th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 531–540. [Online]. Available: http://proceedings.mlr.press/v80/belghazi18a.html

[22] S. Nowozin, B. Cseke, and R. Tomioka, “f-gan: Training generative neural samplers using variational divergence minimization,” in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 271–279.

[23] Y. O. Basgic, Y. Wang, and P. Ishwar, “On privacy-utility tradeoffs for constrained data release mechanisms,” in 2016 Information Theory and Applications Workshop, ITA 2016, La Jolla, CA, USA, January 31 - February 5, 2016. IEEE, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ITA.2016.7888175

[24] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” CoRR, vol. abs/1609.04836, 2016. [Online]. Available: http://arxiv.org/abs/1609.04836

[25] “The Gowalla dataset.” [Online]. Available: https://snap.stanford.edu/data/loc-gowalla.html

[26] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith, “Measuring information leakage using generalized gain functions,” in Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF), 2012, pp. 265–279. [Online]. Available: http://hal.inria.fr/hal-00734044/en

VII. APPENDIX

A. Proofs of the results in the paper

Proof. Let us recall that

\[
X \leftrightarrow W \leftrightarrow Z \leftrightarrow Y. 
\]

represents a Markov chain where:

- the relation between the two random variables X and W is defined by the data distribution \( P_{X,W} \).
- the relation between Z and Y depends only on the chosen classifier according to \( P_{Y|Z} \).
- the relation between Z and W can be described by the variable \( P_{Z|W} \).

If we consider \( X \) as the secret input and \( Y \) as the observable output of a stochastic channel, the mutual information between the two random variable can be expressed as

\[
I(X;Y) = g(P_{Y|X}).
\]
We know from [4] that \( g(\cdot) \) is a convex function with respect to \( P_Y|X \). According to the theory of Markov chains, we can express \( P_Y|X \) as:

\[
P_Y|X(y|x) = \frac{\sum_{zw} P_X,W(x,w)P_Z|W(z|w)P_Y|Z(y|z)}{\sum_w P_X,W(x,w)}. \tag{36}
\]

Eq. 36 represents a linear function of the variable \( P_Z|W \) (all the other probabilities are constant). Hence \( f(P_Z|W) = g(h(P_Z|W)) \) where \( g(\cdot) \) is convex and \( h(\cdot) \) is linear. It is known that the composition of a convex function with a linear one is a convex function and this concludes the proof.

**Proposition 3.**

\[
\min_G \max_C I(X;Y) \leq \min_G I(X;Z).
\]

**Proof.** Let us prove that \( I(X;Z) \) is an upper bound for \( \max_C I(X;Y) \). Given that eq. (1) represents a Markov chain,

\[
X \leftrightarrow Z \leftrightarrow Y. \tag{37}
\]

represents one as well. For the data processing inequality it is possible to assert that

\[
I(X;Y) \leq I(X;Z). \tag{38}
\]

It follows that

\[
\max_C I(X;Y) \leq I(X;Z). \tag{39}
\]

and therefore

\[
\min_G \max_C I(X;Y) \leq \min_G I(X;Z). \tag{40}
\]

concluding the proof.

**Proposition 4.**

\[
B(X \mid Z) \leq B(X \mid Y)
\]

**Proof.**

\[
B(X \mid Z) = \sum_z P_Z(z)(1 - \max_x P_{X\mid Z}(x \mid z))
\]

\[
= 1 - \sum_z P_Z(z) \max_x P_{X\mid Z}(x \mid z)
\]

\[
= 1 - \sum_z P_Y(y) \sum_y P_{Z\mid Y}(z\mid y) \max_x P_{X\mid Z}(x \mid z)
\]

\[
\leq 1 - \sum_y P_Y(y) \max_x \sum_z P_{Z\mid Y}(z\mid y) P_{X\mid Z}(x \mid z)
\]

\[
= 1 - \sum_y P_Y(y) \max_x P_{X\mid Y}(x \mid y)
\]

\[
= B(X \mid Y)
\]

**B. Results of the experiments on the training sets**

The results of the experiments on the training sets of the synthetic data and on the Gowalla data are illustrated in Fig. 13, 14, 15 and 16.
Fig. 13: Synthetic training data. From left to right: Laplacian noise, no noise, our noise. $L = 270\text{m}$.

Fig. 14: Synthetic training data. From left to right: Laplacian noise, no noise, our noise. $L = 173\text{m}$.

Fig. 15: Gowalla training data. From left to right: Laplacian noise, no noise, our noise. $L = 1150\text{m}$.

Fig. 16: Gowalla training data. From left to right: Laplacian noise, no noise, our noise. $L = 518\text{m}$.