UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Differentially Private Aggregation of Mobility Data

Privatsphäre-erhaltende Aggregation von Mobilitätsdaten

Masterarbeit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Johannes Liebenow

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi

Lübeck, den 24. März 2022

Abstract

In a pandemic such as COVID-19, information that gives further insights where potentially critical contacts happen is valuable. Additional information can help to improve the risk evaluation of a critical contact from a medical point of view, but it can also be used to get a better understanding of the pandemic in general. To tackle the lack of information, we extract information about environments during a critical contact from local user data and publish it to allow further analyses.

Based on the setting of the German Corona-Warn-App and in combination with differential privacy and secure multiparty computation, we implement a two-phase distributed clustering. Our system automatically records audio data, which is then used to determine the environment of critical contacts. Since audio data is very high-dimensional, we apply techniques from contrastive learning to reduce the number of dimensions. By utilizing the resulting embedding space, the clients' data gets partitioned into clusters to extract some representations in a privacy-preserving way. To enhance their quality further, the representations from all users are then processed by an external server, followed by a distributed learning phase. The end of the process is marked by a second distributed voting to assign corresponding labels.

In contrast to previous work, we design our system in such a way that it scales well in the number of users without a heavy cryptographic overhead. By construction, it also takes into account the limited resources of modern smartphones .

Zusammenfassung

In einer Pandemie wie bei COVID19 sind Informationen, die Aufschluss über die Umgebung während eines kritischen Kontakts geben, von hoher Bedeutung. Sie können dazu genutzt werden, die Risikoeinschätzung eines kritischen Kontakts zu verbessern und auch dabei helfen, das grundsätzliche Pandemiegeschehen besser zu verstehen. Um dies zu erreichen, verarbeiten wir lokale Daten von Nutzern und veröffentlichen die daraus entstanden Erkenntnisse für weitere Analysen.
Basierend auf dem Szenario der Corona-Warn-App implementieren wir ein System, das auf Differential Privacy und Secure Multiparty Computation beruht. Dabei werden automatisiert Audioaufnahmen erstellt, die dann dazu genutzt werden, die Umgebung von kritischen Kontakten zu bestimmen. Da speziell Audiodaten sehr hochdimensional sind, wenden wir Contrastive Learning an, um die Anzahl an Dimension deutlich zu reduzieren. Mit der resultierenden Einbettung werden für die Nutzerdaten Repräsentanten extrahiert, indem sie unter Berücksichtigung der Privacy in Gruppen geteilt werden. Danach erfolgt eine verteilte Abstimmung, die dazu dient, die Qualität der Repräsentationen weiter zu verbessern. Abgeschlossen wird der Lernprozess mit einer weiteren Abstimmung, um geeignete Labels zu bestimmen.
Im Gegensatz zu anderen Arbeiten in dieser Richtung erstellen wir unser System so, dass eine große Anzahl an Nutzern ohne Speicher- und Zeit-intensive kryptographische Operationen ermöglicht und gleichzeitig die limitierten Ressourcen von Smartphones der Nutzer beachtet werden.

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, 24. März 2022

Contents

1 Introduction

The German Corona-Warn-App (CWA), with around 34 million downloads [22], tries to detect critical contacts based on little information. If trustworthy handling of user data can be ensured and privacy violations prevented, then far more pandemic-relevant information can be collected. In the COVID-19 pandemic, information about the environments of critical contacts reflecting the current situation can be of high value. For instance, to take measures like shutting down gastronomy there has to be enough evidence to support the argument that many potential infections happen in such environments. From the medical point of view, additional information to evaluate whether a contact was critical can help to increase the detection rate and thus enhance individuals' confidence to use the Corona-Warn-App.

We use audio data in combination with special learning techniques to classify the environment by simultaneously preserving the privacy of participating individuals. In a distributed learning process, we apply unsupervised learning techniques in combination with a distributed voting to extract representations of environments and assign corresponding labels using private data. The resulting output reflects the current situation of the pandemic. It can be used to create statistics, perform further learning tasks or, in general, observe changes in the data over time.

There already exists work trying to realize a distributed learning process based on unsupervised learning [12, 7, 4, 44]. However, in our setting, with a large number of participants who utilize smartphones with naturally limited resources, current work struggles. The reasons for this can be summarized in two points. On the one hand, we see differential privacy as a mandatory criterion which is not considered often. On the other hand, expensive cryptographic operations can drastically limit the number of possible users. In our work, we aim to construct a lightweight system handling many users under strong differential privacy guarantees. In the following chapters, we use the terms user and client interchangeably.

We create a system realizing distributed clustering on the basis of audio data. We assume a setting similar to that of the Corona-Warn-App with the slight modification that users automatically record a small audio sample whenever a critical contact happens.

The core idea is to use unsupervised learning in combination with an encoder trained via contrastive learning. Users locally collect some representatives of their data under local differential privacy (LDP). Then, the process is repeated in a federated way on the union

of all representatives. Inspired by [21], we perform a distributed voting combining secure multiparty computation (SMPC) and central differential privacy to reduce the noise from LDP. Labels are generated based on an additional distributed voting process to get the final representations.
In the end of a learning phase, we obtain a labeled set of representations based on the distributed data of all clients in a secure and privacy-preserving way.

1.1 Contribution

Our contribution can be summarized as follows:

- We design a privacy-preserving and secure protocol for distributed clustering to learn representations and corresponding labels. The protocol scales well in the number of clients and the part running locally on the devices of clients does not require much resources.
- Implementation of a state-of-the-art secure summation protocol to realize a distributed voting.
- We test different contrastive learning based approaches for training an encoder on audio data to perform clustering in the resulting embedding space.
- The applied clustering algorithms and the encoder are used as a black box. Thus, future improvements can be integrated without additional effort.

1.2 Structure

The rest of this work is structured as follows. First, we discuss related work in Chapter 2 and introduce the fundamentals of this work in Chapter 3. It is followed by a problem statement in Chapter 4 and the outline of our approach in Chapter 5. Then, we present a federated learning phase to obtain representations in Chapter 6 and assign corresponding labels in Chapter 7. In Chapter 8, we analyze the security and privacy of our system. Chapter 9 presents our experiments combined with an evaluation, and we conclude with Chapter 10.

2 Related Work

Recently, a lot of work has gone into federated learning [9, 40, 30, 26, 28]. Given a set of distributed clients, the goal is to globally train a model based on sensitive data of clients. However, this approach requires labeled data. In cases where only unlabeled data is available, there also exists some other work [46, 45].
Besides extracting knowledge for training a neural network, one can also apply other unsupervised learning techniques such as clustering. This is called distributed or federated clustering and there is also a lot of work on this topic [25, 12, 7, 4]. Yet, a multitude of work does not consider any security or privacy guarantees and only supports a few clients. Thus, they cannot be applied in our scenario where we consider a huge user base and do not trust other parties.
Then, there is work on distributed clustering which gives certain security guarantees [31, 24, 23, 15, 1]. Although, some of it claims to preserve privacy, in fact it preserves security but not differential privacy. As well as the non-secure approaches, most of these protocols do not scale well in the number of users.
In general, those security based approaches can be made privacy preserving by applying standard techniques of local differential privacy. However, this would result in a lot of noise which has the potential to drastically reduce utility.
Apart from that, there exists a handful of work such as [44] combining differentially private clustering algorithms [38, 8] with secure multiparty computation to apply them in a distributed setting. In these cases, secure multiparty computation can introduce a huge communication and computation overhead and may not scale well in the number of users.
Closest to our approach is the work of [25] which combines local and global clustering with a subsequent labeling phase. In this context, our work can be seen as an approach to make their work secure and differentially private by applying existing privacy preserving clustering algorithms in combination with efficient secure multiparty computation protocols.

3 Preliminaries

In this Chapter we introduce the most important topics for this work. We start by giving a brief definition of differential privacy followed by relevant theorems. Then we give an overview about clustering and two important algorithms from this field, followed by a short description of acoustic scene classification. In the end, we introduce contrastive learning.

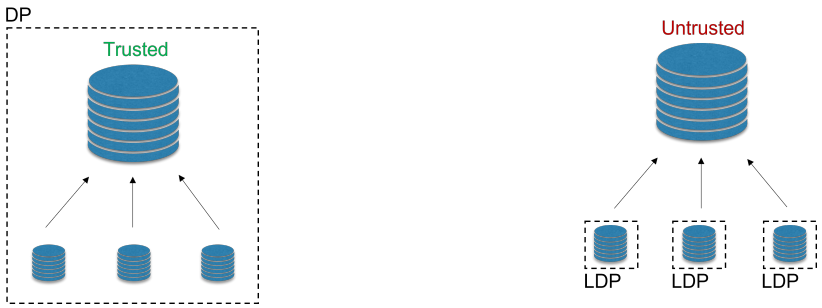
3.1 Differential Privacy

With differential privacy the influence of a single data point is hidden when analyzing the result of a computation. An attacker should not be able to infer enough information to be sure that a specific data point was part of the computation. This requirement is formally defined to ensure that the privacy of a single data point or, e.g. the privacy of all data points contributed by a single user is protected.
Formally, we consider the scenario where an attacker as access to the output of a randomized mechanism $\mathcal{M}$ and two neighboring datasets D, D' and tries to identify whether D or D' was used as input. Neighboring in this context means that D and D' only differ in one element x: $D' = D \cup \{x\}$.
To give the privacy guarantees of $\mathcal{M}$, two parameters have to be defined. The first one is ε capturing the multiplicative difference between the output distributions of $\mathcal{M}$ on D and D'. The second parameter δ describes the probability of events that are not covered by the ε bounds.

Theorem 3.1 (Differential Privacy). *A randomized mechanism $\mathcal{M} : \mathbb{D} \to A$ preserves (ε, δ)-differential privacy ((ε, δ)-DP) for some $\varepsilon > 0$ and $\delta > 0$ if for all neighboring data sets $D, D' \in \mathbb{D}$ and $S \subseteq A$:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta$$

In general, the obtained privacy guarantees are assumed to hold in the so called centralized setting. A trusted aggregator to whom users send their raw data applies a privacy preserving mechanism and publishes the final result. During the process it is important that no intermediate results of any kind and especially no data points which can directly be traced back to a single individual are published or extractable from any involved party.



(a) Central DP (b) Local DP

Figure 3.1: Illustration of the central and local setting of differential privacy. The central setting assumes a trusted aggregator to whom clients send their private data. The aggregator applies a privacy-preserving mechanism and outputs the result. In the local setting, clients do not trust the aggregator and therefore, have to apply differential privacy on their own.

After executing the mechanism, the aggregator can publish the final result without any privacy violations since the mechanism preserves differential privacy.

Local Differential Privacy (LDP) In the central setting of differential privacy, the aggregator has access to all the data and is thus trusted. The local setting describes the scenario of a malicious aggregator who tries learn something about the data of a single user. Now, a single user has to take care of their own privacy. That is why each user applies a differentially private mechanism locally. However, this naturally results in a lot more noise compared to the central setting. The intuition behind it is that now the data points of a single user cannot hide within the data points of other users and cannot profit from their noise anymore.

Theorem 3.2 (Local Differential Privacy). *A randomized mechanism $\mathcal{M} : \mathbb{D} \to A$ preserves (ε, δ)-local differential privacy ((ε, δ)-LDP) for some $\varepsilon > 0$ and $\delta > 0$ if for all pairs of clients' neighboring data sets $D, D' \in \mathbb{D}$ and $S \subseteq A$:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta$$

Privacy-Utility Trade-Off Unfortunately, differential privacy does not come for free, because introducing uncertainty can only be realized by randomization which not only allows a mechanism to preserve privacy but can also drastically reduce the utility. The intuition behind this is straight forward, since constructing or manipulating a mechanism in a way that the output is not always the optimum with respect to the given input results in a lowered utility. Generally speaking, by using differential privacy it naturally introduces a privacy-utility trade-off.

Sequential Composition With sequential composition one can determine the privacy leakage of applying a privacy preserving mechanism multiple times on the same data. Given a mechanism $\mathcal{M}$ that preserves (ε, δ)-DP then a new mechanism $\mathcal{M}'$ which is a composition of multiple execution of $\mathcal{M}$ on the same data set preserves $(k\varepsilon, k\delta)$-DP.

Post-Processing Theorem Another property of differential privacy is called post-processing theorem. It states that any computation operating on the results of a privacy preserving mechanism is itself privacy preserving. In other words, the privacy guarantees cannot be destroyed with any kind of further post-processing. For clarification, if results obtained by a differentially private mechanism are processed in any way which depends on the original non private data points then this introduces another source of leakage and thus the privacy guarantees are destroyed.

Laplace Mechanism A common application when working with databases is to answer queries of the form $f : D \rightarrow \mathcal{N}^k$ with D being the database and k the number of dimensions of the output. In order to preserve differential privacy every mechanism which serves this purpose can be made privacy preserving by simply adding Laplace noise to the output. The amount of noise depends on the l_1-sensitivity Δ of the function f. It describes the maximal difference between all outputs of f on D and on D': $\Delta = max_{x \in D, x' \in D'} ||f(x) - f(x')||_1$. The mechanism $\mathcal{M}(x) = f(x) + \text{Lap}\left(\frac{\Delta}{\varepsilon}\right)$ preserves $(\varepsilon, 0)$-DP.

Privacy-Preserving Argmax Privacy-preserving argmax also known as report noisy max is a simple but not less powerful privacy preserving mechanism which can be used to obtain the index of the maximal element from a set of numeric values. It preserves $(\varepsilon, 0)$-DP. The mechanism itself is based on the Laplace mechanism and works as follows: Given a set of numeric values L, we first add Laplace noise to all values where Δ depends on the construction of L. After determining the maximal element we only return its index as the final result of report noisy max.

In case we need to determine the top k elements in a privacy preserving way we can simply repeat the process and remove the output from L for the next iterations. Using report noisy max on the same data increases the privacy leakage and by applying sequential composition we get that k iterations preserve $(k\varepsilon, 0)$-DP.

3.2 Cryptographic Primitives

Definition 3.3 (Negligible Functions). A function ν is negligible in η if for all polynomials p, $\nu(\eta) < 1/p(\eta)$ for sufficiently large $\eta \in \mathbb{N}$.

Definition 3.4 (Advantage of an Distinguisher). For a pair of random variables F, G and a Turing machine $\mathcal{A}$ (called the attacker) who, given F or G, outputs 0 or 1, *the advantage of $\mathcal{A}$ to distinguish* between F or G is

$$|\Pr[A(F) = 1] - \Pr[A(G) = 1]|$$

Definition 3.5 (Perfect Indistinguishability). A pair of families of random variables $(F_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}, (G_{\sigma,\lambda})_{\eta,\lambda\in\mathbb{N}}$ is *perfectly indistinguishable* if for all Turing machines, called attackers, $\mathcal{A}$ that output 0 or 1 there are two negligible functions ν_1, ν_2 such that the advantage of $\mathcal{A}$ to distinguish $(F_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}$ from $(G_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}$ is $\leq \nu_1(\sigma) + \nu_2(\lambda)$. As an abbreviation, we write

$$F \approx_{\sigma,\lambda} G$$

Definition 3.6 (Computational Indistinguishability). A pair of families of random variables $(F_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}, (G_{\sigma,\lambda})_{\eta,\lambda\in\mathbb{N}}$ is *computationally indistinguishable* if for all probabilistic polynomial-time bounded Turing machines, called attackers, $\mathcal{A}$ that output 0 or 1 there are two negligible functions ν_1, ν_2 such that the advantage of $\mathcal{A}$ to distinguish $(F_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}$ from $(G_{\sigma,\lambda})_{\sigma,\lambda\in\mathbb{N}}$ is $\leq \nu_1(\sigma) + \nu_2(\lambda)$. As an abbreviation, we write

$$F \approx^{c}_{\sigma,\lambda} G$$

3.3 Secure Multiparty Computation

The goal of secure multiparty computation (SMPC) is to allow several parties to jointly compute a function without revealing their individual inputs to other parties. It is especially useful for scenarios like ours where multiple users have local data which they want to be private. Most SMPC schemes suffer from a huge communication overhead and scale poorly to a growing number of participating parties. That is why the choice for a specific SMPC protocol highly depends on the respective scenario and the available resources in

terms of memory, time, computational power and bandwidth.
For a formal definition, we refer to Section 3.4 where we describe the details of secure summation.

Secure Summation Secure summation is a subset of protocols in secure multi party computation to compute a distributed sum without revealing any inputs. This also gives the opportunity to require much less resources [6]. Their work „Secure Aggregation with (Poly)-Logarithmic Overhead“ scales well in the number of parties while guaranteeing a low communication overhead. It is also dropout resistant to some extend which is a very powerful property, especially in our mobile setting where one has to accept a certain risk of dropouts due to battery or internet connection problems. The main insight in contrast to previous work on secure summation [10] is that there is no need for a user to communicate with every other user and that it suffices to use a k-regular graph (a user communicates with k others and the server). In this way the protocol can even handle up to 10^8 users.

Connections to Differential Privacy By using an SMPC-based protocol we have the guarantee that users cannot learn the private inputs of other users. This also means that during the process no intermediate results of any kind are leaked. We can use this to our advantage in terms of differential privacy, because we are now in the centralized setting and thus it suffices that users only add a small portion of noise to their data scaled by the number of users. Without SMPC we would find ourselves in the local setting of differential privacy where every user has to infer a huge amount of noise on their private data.

3.4 Secure Summation Protocol

We use this section to dive deeper into a specific secure summation protocol which we apply in our work.
The protocol „Secure Single-Server Aggregation with (Poly)Logarithmic Overhead“[6] realizes secure summation among clients and a single server. It can handle a huge number of users (up to a billion) with a non-quadratic communication overhead and it only requires a small amount of resources on client-side. The protocol is resistant up to maximum fraction of dropouts ρ and a maximum fraction of corrupted clients σ.

3.4.1 Security

In the next lines we describe the cryptographic properties of the secure summation protocol in all detail. In this context, the authors distinguish between two attacker models, semi-honest and semi-malicious.

Semi-Honest In the semi-honest attacker model the server is trusted but a maximum fraction of corrupted clients γ is assumed who can arbitrarily deviate from the protocol. Note that clients are corrupted before the protocol execution starts.
In this setting, the authors have proven security of their secure summation protocol against an unbounded adversary. In other words, the output of a probabilistic polynomial time (PPT) simulator Sim is perfectly indistinguishable from the joint view of the server and the corrupted clients Real. For clarification, Sim describe the run of a simulator that internally runs the ideal protocol as a random variable that also takes as values all observations of the attacker, including network messages. Real on the other side describes the run of the real protocol as a random variable that takes as values all observations of the attacker, including network messages.
Following their security proof, we can extract the following definition.

Definition 3.7 (Security in the semi-honest setting of [6]). Let $\sigma, \lambda, \eta \in \mathbb{N}$, $\rho, \gamma \in [0, 1]$ be and assume that the graph generation determining the communication lines between clients is properly executed in σ and η, an IND-CPA secure authenticate encryption scheme, a λ-secure key agreement protocol and an ideal secure summation function $\mathcal{F}$. The secure summation algorithm π_{SecAgg} is secure in σ and λ if the output of Sim is perfectly indistinguishable from the joint view of the server and the corrupted clients Real$(s_1, .., s_n)$ in that execution such that Real$(s_1, .., s_n) \approx_{\sigma,\lambda}$ Sim$(\mathcal{F}(s_1, .., s_n))$ holds in the presence of an unbounded attacker $\mathcal{A}$ given a maximum fraction of dropouts ρ and a maximum fraction of corrupted clients γ, i.e., there are negligible functions ν_1, ν_2 such that

$$\text{Advantage}(\mathcal{A}) = |\Pr[\mathcal{A}(\text{Real}) = 1] - \Pr[\mathcal{A}(\text{Sim}(\mathcal{F})) = 1]| \leq \nu_1(\sigma) + \nu_2(\lambda)$$

Semi-Malicious This scenario extends the semi-honest setting but now the server is also corrupted and can deviate from the protocol with one assumption. The server has so behave semi-honestly in the first part of the protocol during the key collection phase to avoid that it can interrupt the communication between clients later on. For the secure summation protocol [6] the authors haven proven the security against an PPT attacker which is a strictly weaker attacker than in the semi-honest case. Now, the output of a PPT simulator is computationally indistinguishable from the joint view of the corrupted

server and corrupted clients. However, the security does not refer to the sum of inputs of all clients but rather partial sums consisting of at least $\alpha \cdot (1-\gamma) \cdot n$ inputs of honest clients with $\alpha \in [0,1]$, n as the number of clients and γ as the maximum fraction of corrupted clients.
Following their security proof, we can extract the following definition.

Definition 3.8 (Security in the semi-malicious setting of [6]). Let $\sigma, \lambda, \eta \in \mathbb{N}$, $\rho, \gamma, \alpha \in [0,1]$ be and assume that the graph generation determining the communication lines between clients is properly executed in σ and η, an IND-CPA secure authenticate encryption scheme, a λ-secure key agreement protocol and an ideal α-secure summation function $\mathcal{F}_\alpha$ which only returns an output if enough (in terms of the fraction alpha) inputs of honest clients are involved in a sum. The secure summation algorithm π_{SecAgg} is secure in σ and λ if the output of Sim is computationally indistinguishable from the joint view of the server and the corrupted clients Real$(s_1, .., s_n)$ in that execution such that Real$(s_1, .., s_n) \approx^c_{\sigma,\lambda}$ Sim$(\mathcal{F}(s_1, .., s_n))$ holds in the presence of a PPT attacker $\mathcal{A}$ behaving semi-honestly in Part I of the protocol, given a maximum fraction of dropouts ρ and a maximum fraction of corrupted clients γ, i.e., there are negligible functions ν_1, ν_2 such that

$$\text{Advantage}(\mathcal{A}) = |\Pr[\mathcal{A}(\text{Real}) = 1] - \Pr[\mathcal{A}(\text{Sim}(\mathcal{F}_\alpha)) = 1]| \leq \nu_1(\sigma) + \nu_2(\lambda)$$

3.4.2 Correctness

The correctness of the secure summation protocol of [6] describes the property of the protocol to return the correct sum. The authors state that the correctness of their protocol in terms of what an attacker might learn from the final result is independent from the proposed security guarantees. If correctness fails then the server cannot reconstruct the correct sum.
In both scenarios, correctness holds if the fraction of dropouts is less or equal to ρ and only breaks with a negligible probability in η. The malicious server can always abort the entire protocol and if malicious clients abort then they can also break the correctness property since in order to restore secret-shared values a certain number of shares is mandatory.

3.5 Clustering

In unsupervised learning one wants to learn something about data without any labels. Clustering is one of the most common techniques in unsupervised learning. It aims to group a set of points into a very much smaller number of clusters. In the end, every data point should be assigned to one (hard clustering) or more (soft clustering) clusters. The

assignment itself can vary from a probability distribution where a specific point belongs to every other cluster with a certain probability to a strict classification into a single cluster. Usually, a clustering algorithm tries to update the assignment in every iteration based on a distance metric and an update or assignment rule which can vary based on the properties of the given data and the specific algorithm. When using unsupervised learning, it is often the case that the ground truth e.g. the true clusters and their parameters are unknown. That is why the quality of a clustering cannot be easily determined. Although there are a few metrics, they can only evaluate the clustering with respect to the given set of points. That is the reason why it is good practice to repeat the clustering process multiple times and also vary the hyper-parameters to eventually achieve a good clustering.
Usually before applying a clustering algorithm a special pre-processing like PCA or manifold learning is used which reduces the number of dimensions and simultaneously tries to preserve as much information as possible. This is necessary because clustering in high dimensional space can degrade the utility drastically [5].
In this thesis we realize parts of our system based on the following clustering algorithms:

K-Means One of the most fundamental clustering algorithms is called k-means. As the name suggests, it groups the set of points into k clusters with k as a hyper-parameter. For a fixed number of iterations, the k-means algorithm assigns each point to its nearest cluster and the decision is based on the distances to all cluster centers (centroids) which can be calculated by taking the mean over all points assigned to a single cluster. This is done until the last iteration has finished or e.g., the difference between the centroids of the last iteration and the new ones is less than a predefined threshold to indicate that the algorithm has found an optimum without any guarantee that this is actually the best possible clustering.

DP-Llyod-Impr A simple differentially private clustering algorithm based on k-means is called DP-Llyod [8]. The main idea is very similar to k-means and tries to find k clusters by iteratively computing voronoi diagrams and adjusting the cluster centers. The DP version adds noise to each step and normalizes each dimension to be able to bound the influence of a single point.
In our work we apply an improved version called DP-Llyod-Impr [38] which improves the selection of initial centroids and presents a way to determine the number of iterations.

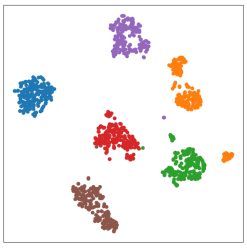


Figure 3.2: Two-dimensional embedding space of the MNIST dataset [29] via TSNE [39]. Even though the number of dimensions is drastically reduced, the reduction preserved enough information to separate the classes in two dimensions.

OPTICS OPTICS is a clustering algorithm created as an improved version of DBSCAN [36] to handle data of arbitrary density. Similar to k-means, it assigns each point to a single cluster and the number of clusters is solely defined by OPTICS. It also introduces a noise cluster containing all points which seem to be outliers. The OPTICS algorithm iterates the entire data set once and orders all points on a so called reachability graph where connected areas on this graph usually represent single clusters.
For OPTICS, we are not going into details about differential private versions because in this work we only apply it on non-private data or data generated in a privacy preserving way.

3.6 Contrastive Learning

Contrastive Learning is a technique from the field of semi-supervised learning which can be applied in a variety of different data domains like image and audio data and as the name suggests it lies between supervised and semi-supervised learning. Although contrastive learning does not require a labeled data set, it is not fully unsupervised because it introduces artificial labels during training. However, it is also not completely supervised because in contrast to supervised learning these artificial labels are created on the fly for every batch in every training epoch and cannot be worked out beforehand.
Recent work shows that neural networks trained via contrastive learning are a very good feature extractor which can be used to overcome state-of-the-art accuracy results on most of the benchmark data sets like ImageNet. As a consequence, multiple frameworks like SIM-CLR [13] and MoCo [17] were developed for a variety of use cases ranging from image classification to speech detection.

Figure 3.3: Illustration of the typical acoustic scene classification process. Audio data servers as input for a machine learning model. The resulting output can then be interpreted as a single label describing the environment in which the recording presumably took place.

Intuition The intuition behind contrastive learning is that a neural network can be trained in a way that it learns the underlying nature of the respective data domain. Such a network should recognize that two images of the same dog showing different body parts are not so different in comparison to an image of a chair which is in fact a different object. A neural network trained via contrastive learning can learn those differences or in other words it can internalize a similarity metric. Then, it serves as an encoder projecting raw inputs into an embedding space where the distances between embeddings resemble the original distances among the input data. The embedding space typically has a much smaller dimension compared to the raw input data which has multiple advantages under the premise that the dimension reduction preserves enough information. A good lower dimensional representation usually simplifies further learning problems also called downstream tasks a lot.
Downstream tasks can also be used to get some insights into the quality of the embedding space itself. The higher the quality of an embedding space the better should be the accuracy of a certain downstream task. The intuition behind this is straight forward because it is only natural to assume that the quality of the feature extractor has a vast influence on the accuracy of the respective learning task.

Privacy Protection After an input is projected into the embedding space of a neural network trained via contrastive learning the threat to privacy is not eliminated. Embeddings are still sensitive and sometimes even more than data obtained from supervised models [18]. Thus it has to be treated as carefully as raw data.

3.7 Acoustic Scene Classification

Acoustic scene classification (ACS) in its core aims to find the most suitable label for an environment where an audio recording has happened. Such a label may look like „park“ or „metro station“.
The general ACS process as illustrated in Figure 3.3 works as follows: A single audio recording of e.g. 10 seconds serves as input for a trained classifier which outputs a proba-

bility for every considered label. Usually, the label with the highest probability is selected as output. To train a classifier there exist some labeled data sets like [42, 43, 16].
However, there are multiple reasons complicating acoustic scene classification. Different microphones can lead to different qualities, naturally two recordings of the same environment may not sound exactly the same and unpredictable events like a group of people walking by may occur.

4 Problem Statement

In this chapter, we present a detailed motivation for our approach, the underlying scenario which we assume for clients and a server, some challenges we have to tackle in this work and a short overview about possible threat models.

4.1 Motivation

In the COVID19 pandemic, information about the environments of critical contacts which are available in real-time can be of high value. For instance, to take measures like shutting down gastronomy there has to be enough evidence supporting the argument that in such environments a lot of potential infections happen.
Recent approaches to identify such critical contacts and to inform involved people in case of a potential infection use proximity detection based on Bluetooth Low Energy (LE) to detect critical contacts. The Corona-Warn-App[2] is the German version of such an approach with currently around 34 million downloads[22]. To realize contact tracing it stores cropped timestamps and the estimated proximity during a contact in combination with a few Bluetooth LE parameters which can later be used to inform potentially infected people that they had contact with a positive tested person in the past. The question arises if there is more information which can be extracted out of a critical contact.
Such kind of valuable information could be the surrounding environment during a contact. We aim to find an appropriate representation and also a label by utilizing the microphone of a smartphone to analyze short audio recordings. At first, recording audio files in the everyday life sounds dangerous regarding the privacy of an individual but recent work [19, 33] shows that it is possible to create a system which is based on the microphone and simultaneously benefits from the potential of audio data. We believe that with the help of security and privacy such audio recordings can be performed without a threat to privacy. We want to highlight that we are only interested in the type of the environment in which the recording has happened. In other words, if a recording happens inside the city park of Berlin then the only information which matters and which we want to preserve is that the recording took place in a park.

Acoustic Scene Classification The process of classifying representations obtained from audio recordings to get a label describing in which environment the recording took

place is called acoustic scene classification. Typically, it is treated as a supervised learning task which requires labeled data. In principle, we could rely on this approach, but once the training has begun labels are fixed and can hardly be modified. Thus, we would have to commit ourselves to a predefined set of labels which may not resemble the real life. People visit different environments like „Bus“, „Office“, „Nature“ which we assume can vary a lot from person to person and are therefore difficult to foresee. If we assume the standard labels from benchmark data sets then without doubt we would cover a great part of possible environments but it is not said that there are no other environments out there which we would not cover. Granularity also matters at this point. Assume the label „Building“ which presumably covers environments like „Office“ or „Library“. We cannot be sure that there will not be a situation after having trained with the label „Building“ where a distinction between such environments suddenly becomes important. We also want our system to be able to react on a shift in the labels, because e.g. if during a pandemic the government takes the measure of shutting down gastronomy then we would assume that all environments regarding the gastronomy disappear sooner or later. Therefore, we want to base our work on approaches from acoustic scene classification while simultaneously being able to dynamically discover new labels.

Goal We aim to construct a system which can perform a learning phase with a lot of participating users. The primary goal is to dynamically learn representations of currently important environments and their corresponding labels which can then be used for further learning tasks without posing a risk to privacy. The key idea is to be open-minded to any kind of environment by applying unsupervised learning techniques in combination with a labeling mechanism which accumulates in a lightweight solution for smartphones on the basis of SMPC-based protocols as well as differential privacy and by considering the restricted resources.

4.2 System Model

This section describes the scenario which we take as a basis throughout this thesis from the view of clients participating in a learning phase and the server which handles the distributed parts of our system.

4.2.1 Clients

Clients can only participate in a learning phase with the help of a smartphone. On the phone, an app has to be installed in order to use our system. We further require that the number of clients is publicly available which can be approximated by the number of

downloads in total. Our system does not require any external services except access to the microphone to be able to make short audio recordings in regular intervals when required. An inherent problem of distributed systems which we have to take into account are dropouts. In reality, especially when considering smartphones, dropouts can occur often due to connectivity or battery problems. Thus, dropouts can make the clients' behavior unpredictable. We try to cover this by assuming that during a learning phase a maximum fraction of dropouts ρ occur.
Over time, our system records audio samples which may include sensitive information. Clients do not want to share their sensitive data neither with other clients nor the server in order to protect their privacy independent of the actual behavior or protocol. However, in order to realize communication between clients and the server, we give formal security and privacy guarantees such that private information about other clients' sensitive data cannot be extracted during and after a learning phase. In summary, by participating in a learning phase our system has to be constructed in a way that no one can be discriminated based on their sensitive data.

4.2.2 Server

To realize our system we use a single server which plays the role of the coordinator for every kind of distributed communication. The server directly communicates with every single client and if necessary also handles the communication from one client to another.
In this work we assume a server which can handle a lot of communication and perform extensive operations. Therefore, in contrast to the smartphone of clients, the server has enough resources in terms of memory, RAM, CPU and bandwidth. In general, this means that the server can provide enough resources to perform multiple learning phases and does not has to be considered as bottleneck of any kind.

4.2.3 Threat Models

We illustrate different threat models by distinguishing between a scenario in which the server behaves honest-but-curious and one where the server is controlled by a malicious adversary. Throughout the thesis, if not said otherwise, the first scenario is assumed.
Independent of the scenario we assume a maximum fraction of corrupted clients γ who are controlled by a malicious adversary and thus can arbitrarily deviate from our protocol. The adversary themself operates on network level and can observe the communication among clients and also between clients and the server.

Semi-Honest In this scenario, the server behaves honest-but-curious and follows the protocol without ever trying to deviate from it. However, it tries to extract as much sensitive information as possible from the data it has to process throughout an entire learning phase. This can also be viewed as if the server is controlled by a passive adversary. The reason why this scenario is still named semi-malicious is that we have to accept slight assumptions on the behavior of the server in order to provide security for our system. Further details are explained later in Section 8.1.
The malicious clients can be envisioned as entities who try to learn as much as possible from the final output of a learning phase by combining it with own data, intercept data published by other clients or send arbitrary messages to the server. Such a client can also deviate from the protocol or abort it at any point in time. In our scenario, a client may also posses root rights on its personal device, and hence could have insights into internal processes of a learning phase.
Note that in case of $\gamma = 0$ the entire scenario can also be called honest-but-curious since now all involved parties act honest-but-curious.

Semi-Malicious Now, the malicious adversary controls the server as well. This means that the server itself can deviate from the protocol or abort it completely. A malicious server may send fake messages to clients and intentionally excludes or isolates clients in the communication. In the worst case such an attacker combines the clients' views with the view of the server. Corrupting multiple clients can easily be achieved by installing the app carrying our system on multiple devices, whereas corrupting the server and smartphones of other users directly depends on their respective security.

4.3 Challenges

When constructing our system we have to consider and solve some challenges which we outline in the following paragraphs. This includes to not disturb other user tasks on the smartphone, guarantees for security and privacy as well as not to loose sight of utility in general.

User Experience Without doubt, when using a smartphone, clients generally expect high usability such that they can do whatever they want within the limitations of a smartphone without other processes limiting their experience. That is why our system also has to be constructed in such a way that a learning phase does not hinder other processes. More specifically, an excessive use of memory, RAM, CPU capacity and bandwidth has to be avoided. This also relates to a high power consumption, since users are usually not

willing to charge their mobile device often. All those points are reasons to bring users to a point where in the worst case they do not want to take part in the learning process anymore. With this in mind we focus on using resources of smartphones with care.

Privacy and Security Another important aspect which can drastically influence the decision of users to download the app and participate in our learning phase is both security and privacy. Since the server only processes data from clients and thus does not present an entity which owns sensitive data, we are going to focus on security and privacy of clients' sensitive data.
The security and privacy guarantees of our system should be of a form such that clients can be sure that their sensitive data is kept private even during distributed computations including other clients and the server. Formally, no participating party should be able to infer any kind of sensitive information about an individual's sensitive data. Since clients do not trust the server in any way, they have to preserve privacy locally on their own because once sensitive data leaves the device, privacy can no longer be preserved in the absence of a trusted aggregator.

Dropouts Dropouts are a major problem on its own and cannot be ignored especially when considering a distributed setting based on smartphones. We want our system to resist a certain amount of dropouts to still be able to maintain privacy and security guarantees as well as utility. Previous work shows that in order to tolerate dropouts up to a certain degree, one has to accept additional overhead [10, 6].
In this context, it should also be pointed out that dropouts naturally entail time constraints as well since clients will shutdown or restart their smartphone which not only produces a dropout but it also means that operations should not take too long to lower the risk of unintended interruptions.

Utility When using the term utility, we want to capture metrics like accuracy and error minimization to describe the property of our system to produce accurate results. This is especially important when applying privacy protection techniques like differential privacy which naturally induct what is called a privacy-utility trade-off. Formally, this means that in order to achieve some degree of privacy we inevitably have to deviate from the ground truth which makes it important to formulate this deviation with regard to our system.

5 Outline of Our Approach

We aim to construct a system to dynamically learn representations and labels for frequently visited environments on the basis of acoustic data without having to commit ourselves to a predefined set of labels.
We assume multiple clients participating in a learning phase which gets coordinated by a single server. Clients are using smartphones and have installed an app like the German Corona-Warn-App[2] carrying our system which also entails limited resources on client-side.
A distributed learning process works by processing information created on the basis of clients' sensitive audio recordings on an external server. Simultaneously, our system gives certain privacy and security guarantees to avoid any kind of information leakage and discrimination to take place.
In the following paragraphs, we explain the individual steps of the outlined learning phase on a high level. Details for the first four paragraphs can be found in Chapter 6 and for last one in Chapter 7.
The detailed process of an entire learning phase is shown at length in Algorithm 1.

1. Pre-Processing Audio samples in general represent very high dimensional data. This is why we use a neural network pre-trained via contrastive learning based on the approach of [35] to drastically reduce the number of dimensions. Contrastive learning not only reduces the number of dimensions it also tries to preserve relevant information which we want to take advantage of to successfully solve further learning tasks.
Parallel to the scenario of the German Corona-Warn-App our system records audio samples in regular intervals over a period of time. Those are utilized to perform a distributed learning phase in combination with other clients and their sensitive data.

2. Local Clustering After a while, we assume that users have accumulated a set of embeddings originated by audio samples of different environments. Clients are then going to perform a local clustering on their embeddings to create clusters and obtain the corresponding cluster centers (centroids). Since embeddings still present sensitive data and are thus vulnerable to privacy attacks [18], the clustering itself has to preserve privacy.
A subsequent selection process follows to extract the local top centroids. This step also has to preserve privacy, because the underlying metric is calculated on the basis of the embed-

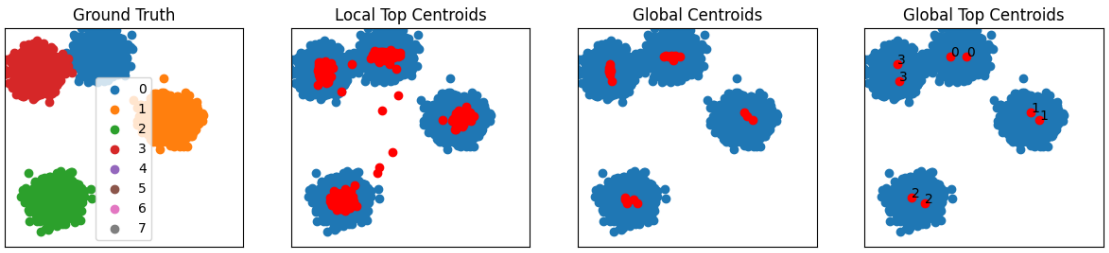


Figure 5.1: Overview of the main steps of a learning phase. The picture on the left shows the dataset used in this execution. The next picture shows the local top centroids of all clients. Then, the result of the global clustering is shown which is followed by the top centroids obtained from the first voting and the corresponding labels obtained from the second voting.

dings. Now, local top centroids have been extracted from the sensitive embeddings in a privacy preserving way which can be send to the server.

3. Global Clustering After all clients have sent data obtained from clustering their local embeddings to the server, the server itself performs another clustering on the union of data points from all clients. This second clustering, aims to find centroids which are in some sense confirmed by the majority of clients. Since clients locally preserve privacy, this step does not have to consider further privacy protection techniques. The resulting global centroids are published by the server.

4. Voting Process In theory, a learning phase could end at this step with reasonable results, however with the potentially huge noise added in the local clustering step due to differential privacy we want to perform another step. This approach is inspired by [21] and realized in our work as a voting process which aims to increase utility.
Therefore, the server initiates a voting process to determine the global top centroids. A client creates a private voting vector based on an evaluation method which includes sensitive embeddings and global centroids published in the last step. We apply both, SMPC and differential privacy so that all private voting vectors can be aggregated by the server. The aggregated voting vector is then analyzed to determine the global top centroids which can be published afterward.

5. Data Labeling After the fourth step, we have learned representations for different environments for which we also want to obtain accurate labels. The key idea is to first obtain labels for local embeddings and then generate labels for global top centroids.
To label any kind of local data of a single client, we have two strategies in mind. On

the one hand one could utilize in-app user feedback that can be imagined as a simple question like „Where have you been?“. On the other hand, labels for local data could also be obtained based on results from previous learning phases.
Labels for global top centroids are then generated by a second voting step. It is realized in a privacy preserving way to allow clients to vote for specific labels. Those labels can be combined with the already global top centroids to obtain labeled data.

n	Number of users
$\mathbb{A}$	Domain of audio data
$\mathbb{X}$	Domain of the embedding space
k	Number of local centroids
k^*	Number of local top centroids
g	Number of global centroids
g^*	Number of global top centroids
$v_{\text{centroids}}$	Number of votes for global centroids
v_{labels}	Number of votes for global labels
$\mathbb{L}$	Set of possible labels
l	Number of local labeled neighbors
ρ	Maximum fraction of dropouts
γ	Maximum fraction of corrupted clients

Table 5.1: Notation Table

Algorithm 1: Distributed Representation Learning and Labeling

Parties: $1, ..., n$ clients and a single server.
Public Parameters: Data domain $\mathbb{X}$, audio domain $\mathbb{A}$, set of labels $\mathbb{L}$, number of users n
Input: $A_u \subseteq \pi$ (by each client u), $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \delta_1$ (privacy parameters)
Output: $\{(\vec{c}_1^*, l_1^*), (\vec{c}_2^*, l_2^*), ..., (\vec{c}_{g^*}^*, l_{g^*}^*)\} \subseteq \mathbb{X} \times \mathbb{L}$ (for the server)

We use the function $argmax_p$ to get the indices of the p maximal values of a vector or the flattened form of a matrix. After step (6) and (10) the server automatically initializes a distributed voting via π_{SecAgg}.

Pre-Processing. Client u:

(1) Collect a set of audio samples A_u and apply encoder $f : \mathbb{A} \to \mathbb{X}$ to obtain corresponding embeddings $X_u = f(A_u)$.
(2) Retrieve a set of labels $L_u \subseteq \mathbb{L}$ through in-app feedback for some embeddings. Label assignment is modeled with the function $lbl : \mathbb{X} \to \mathbb{L} \cup \{\perp\}$.

Local Clustering. Client u:

(3) Cluster set of embeddings X_u into a set of centroids $C_u = \mathcal{M}_{\text{DPLloyd-Impr}}(\epsilon_1, X_u)$.
(4) Create a vector $\vec{w}_u^1$ counting the number of embeddings closest to each cluster center $c_{u,i}$:

$$\vec{w}_u^1[i] = |\{\vec{x}_u \in X_u \mid \vec{c}_{u,i}, \vec{c}_{u,j} \in C_u, 1 \leq j \leq k : ||\vec{x}_u - \vec{c}_{u,i}||_2 \leq ||\vec{x}_u - \vec{c}_{u,j}||_2\}|$$

(5) Apply report noisy max $\mathcal{M}_{\text{RNM}}$ k^* times to select the k^* top centroids $C_u^* = \mathcal{M}_{\text{RNM}}(\epsilon_2, \vec{w}_u^1)$.

Global Clustering. Server:

(6) Apply the OPTICS clustering algorithm π_{OPTICS} on the union of clients' top centroids $C^* = \bigcup_{u=0}^{n} C_u^*$ which results in a set of global centroids $G = \pi_{\text{OPTICS}}(C^*)$ and publish G.

Distributed Voting.

Client u:

(7) Create a vector $\vec{w}_u^2$ to count the number of embeddings closest to each global centroid g_i of the form

$$\vec{w}_u^2[i] = |\{\vec{x}_u \in X_u \mid \vec{g}_i, \vec{g}_j \in G, 1 \leq j \leq g : ||\vec{x}_u - \vec{g}_i||_2 \leq ||\vec{x}_u - \vec{g}_j||_2\}|$$

(8) Create private voting vector for every global centroid g_i

$$\vec{v}_u^1[i] = \text{Lap}\left(\frac{v_{\text{centroids}}}{\epsilon_3 \cdot n \cdot (1 - \rho - \gamma)}\right) + \begin{cases} 1 & \text{if } i \in argmax_{v_{\text{centroids}}}\left(\vec{w}_u^2\right) \\ 0 & \text{else} \end{cases}$$

Server:

(9) Compute the voting vector $\vec{v}^1 = \pi_{\text{SecAgg}}(\vec{v}_1^1, \vec{v}_2^1, ..., \vec{v}_n^1)$.
(10) Select the corresponding top g^* global centroids $G^* = \{\vec{g}_i \in G, 1 \leq i \leq g \mid i \in argmax_{g^*}(\vec{v}^1)\}$ and publish G^*.

Data Labeling.

Client u:

(11) Create a matrix $W_u = (w_{i,l}) \in \mathbb{N}^{g^* \times |\mathbb{L}|}$ counting the number of points with label l closest to centroid g_i:

$$w_{i,l} = |\{\vec{x}_u \in X_u \mid \vec{g}_i, \vec{g}_j \in G^*, 1 \leq j \leq g^* : lbl(x_u) = l \wedge ||\vec{x}_u - \vec{g}_i||_2 \leq ||\vec{x}_u - \vec{g}_j||_2\}|$$

(12) Create private voting vector for every combination of global centroids and labels with $0 < i < |\mathbb{L}| \cdot g^*$

$$\vec{v}_u^2[i] = \text{Lap}\left(\frac{v_1}{\epsilon_4 \cdot n \cdot (1 - \rho - \gamma)}\right) + \begin{cases} 1 & \text{if } i \in argmax_{v_{\text{labels}}}\left(W_u\right) \\ 0 & \text{else} \end{cases}$$

Server:

(13) Compute the voting vector $\vec{v}^2 = \pi_{\text{SecAgg}}(\vec{v}_1^2, \vec{v}_2^2, ..., \vec{v}_n^2)$.
(14) Select the top voted label l_i^* for every global top centroid c_i^*, $l_i^* = argmax_1(\vec{v}^2[i+1 : i+g^*])$ and publish $\{(\vec{c}_1^*, l_1^*), (\vec{c}_2^*, l_2^*), ..., (\vec{c}_{g^*}^*, l_{g^*}^*)\}$.

6 Distributed Clustering

This chapter focuses on the main part of a learning phase to produce representations of audio samples originated from sensitive recordings of different environments. We go into detail about the techniques we apply in different parts of our system.

6.1 Generate Embeddings

When working with high dimensional data like audio samples and especially when applying learning algorithms it is convenient to use dimension reduction algorithms. Such algorithms can reduce the number of dimensions drastically and still preserve relevant information so that further learning tasks can still be applied on the data.
In our work, we apply contrastive learning, a technique from semi-supervised learning to pre-train a neural network based on unlabeled data. The resulting model serves as an encoder projecting audio data into an embedding space.
The motivation to use contrastive learning instead of other dimension reduction techniques is that previous work demonstrated that further learning tasks when applied on an embedding space (downstream tasks) can be solved with a high accuracy. In summary, we hope for the following advantages when applying contrastive learning as dimension reduction technique: (1) Preserve enough relevant information so that clustering algorithms can be applied successfully. (2) Reduce the memory space we need to store data for future learning phases. (3) Reduce the overall computation and communication overhead.
In our work, we analyzed and implemented different contrastive learning approaches based on audio data [41, 11, 35]. For a learning phase, we use the approach from [41] to pre-train a neural network with contrastive learning. Details about the implementation can be found in Section 9.1. The resulting embedding space reduces the dimension of a single audio sample from 8000 to 128 which is small enough to apply clustering algorithms.

6.2 Local Centroids

We describe the process of how local embeddings are getting clustered into a set of local centroids followed by a selection process creating a set of local top centroids.

6.2.1 Clustering

We choose clustering as the next step after clients have gathered some embeddings because of its property to extract centroids representing the data. On the one hand, it helps to reduce the amount of data the server has to process later on in a learning phase and on the other hand, we can filter out failed recordings or in general noise. We believe this process to be successful since it has already been shown that simple linear classifiers can reach a high accuracy in downstream task like acoustic scene classification [35] and there is other work combining contrastive learning with clustering [27].
Independent of the actual algorithm, our clustering is performed on sensitive data. Spoken in terms of differential privacy, the presence or absence of a single data point can drastically change the resulting centroids which inevitable entails that centroids cannot just be published without any further privacy protection. A naive approach is to use an SMPC-based protocol to realize distributed computations without having to directly publish sensitive data. Oftentimes, secure multiparty computation protocols are very time-consuming and need a lot of resources. This makes it even harder in our mobile setting with limited resources on client-side. Thus, we avoid SMPC this time and apply a differential private clustering algorithm which certainly reduces utility but spares a lot of resources and enables us to publish the resulting data.
We apply a differentially private version of the K-Means clustering algorithm called DPLloyd-Impr [38] because there already exists an implementation in combination with a privacy proof.

6.2.2 Find Top Centroids

Depending on the clustering algorithm and chosen hyper-parameters, a client can generate a lot of centroids locally we want to thin out further. This procedure is supported by the same arguments we are using clustering on client-side. We can get more independent of noise and also reduce the overall amount of data.
We count the number of points in a cluster of every centroid and select the top ones. The intuition is that centroids that correspond to a lot of data points represent the data well.
This step also has to be based on differential privacy because relying on sensitive data points again introduces a privacy risk even though the centroids were created by a differentially private cluster algorithm. In this specific case where a single sensitive data point

can only contribute to a single cluster and influence the count of that cluster by 1, we can apply the report noisy max mechanism. We use it multiple times to obtain a subset of centroids called local top centroids. In the end, those top centroids can be published despite the fact that clients do not trust the server by arguing with local differential privacy which states that privacy even holds in the absence of a trusted aggregator.

6.3 Global Centroids

In this step, the federated part of our learning phase comes into play, Here, we want to extract information based on sensitive data from all clients. Once the server has received local top centroids from all clients, another clustering phase is performed to extract global centroids. Now, centroids represent accumulations of local top centroids of multiple clients from which we believe that those truly represent the underlying data distributed among all clients the best.
The server itself does not have to apply differentially private mechanisms in any way because all the data it receives from clients already preserves local differential privacy and can therefore be processed without any risk of violating the privacy of an individual.

As clustering algorithm we use OPTICS instead of DP-Llyod-Impr. The first reason is that it can be partially parallelized which is a very important property regarding the huge amount of data the server has to process in a single learning phase. The second reason lies in the nature of K-Means-based clustering algorithms because they may produce artifacts. Artifacts are centroids not lying within a dense group of points. We can now obtain a clustering with less hyper-parameters, without randomness and the main part of utility relying on the local top centroids of all clients.

6.4 Distributed Voting

At this step, the server has already extracted a subset of global centroids from the noised local top centroids of all clients. Due to local differential privacy, the utility possibly degrades. Inspired by the approach of [21], we perform another step in which users vote for global centroids based on their sensitive data.
By relying on sensitive data again we have to preserve privacy one more time. Applying local differential privacy would result in every client adding a lot of noise to their voting. In sum, the noise can drastically influence the voting result and probably destroys the healing effect of this voting.
A fact we want to take advantage of is that voting in its core consists of the summation of zeros and ones. This represents comparatively less data and simultaneously reduces

the hurdle to use an SMPC-based protocol. Some protocols perfectly fit into this scenario because they scale well in the number of users, only require small amounts of resources and are resistant to dropouts up to a certain degree [10, 6]. This is why we use the secure summation protocol of [6]. It guarantees that an attacker cannot learn anything about private inputs of clients. With this guarantee we can also satisfy the conditions of differential privacy in the central setting for which we have to ensure that no intermediate data can be extracted during the entire voting process. Thus, clients are allowed to add much less noise to their private voting vectors.
The server initializes a voting process with the global centroids as candidate set and every client creates a private voting vector of only zeros. An entry is set to one if its respective centroid lies in the set centroids with the highest numbers of points closest to themselves. In this proximity-based process a single data point can influence only a single centroid but a data point also influences the global centroids now serving as voting candidates. Thus, the sensitivity is equal to the number of allowed votes.
In the end of a learning phase, the global top centroids can be determined easily and published without posing a threat to the privacy of an individual.

7 Data Labeling

In this chapter, we present our approach to label local and then global data points in detail. The second section describes how basic statistics can obtained based on generated labels.

7.1 Generate Labels

So far, a learning phase produces globally available representations of environments based on sensitive acoustic data from multiple clients. In here, we present our approach of how we want to obtain corresponding labels by again taking into account sensitive data from all clients.
We start by labeling local data points of clients and then perform another voting step to obtain labels for global data points.

Local Labels Over a period of time, clients collect labels for data points based on their own feedback. This can be realized by a popup messages inside the app. Usually, clients should be familiar with popups from other apps prompting the user to enter a rating or close an advertisement banner.
A client is presented with a rich set of choices internally generalized into a much smaller set of equivalence classes to avoid misunderstandings and to allow clients to only bother about the current environment. This hierarchical labeling structure is illustrated in Figure 7.1.
We generate labels automatically over time without any extensive computations. The goal is to obtain a sufficient number of labels such that an unlabeled global data point has at least some local labeled neighbors which is important for the global labeling step.

Global Labels In order to label global data points we again use proximity in combination with a majority voting inspired by [47] This also fits into our scenario where we assume that the number of labeled data points is much smaller than number of unlabeled data points. We apply this labeling procedure locally on every client based on their own sensitive data and labels. This is followed by a second voting to aggregate the final voting result.
To start the voting, the server initiates a voting process based on previously published global top centroids serving as voting candidates. Clients create a private voting vector

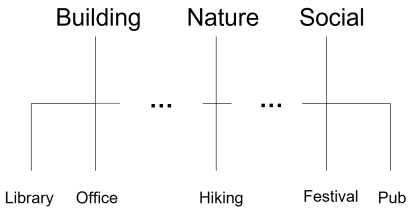


Figure 7.1: An illustration of the proposed labeling strategy. The labels on the bottom are the choices presented in in-app polls. The labels on top are the equivalence classes used to label global centroids.

where every entry corresponds to a specific label for a specific global top centroid. Then, clients use proximity in combination with a majority voting to obtain a label for every voting candidate. In more detail, clients count the number of closest labeled local data points for each voting candidate and use the label which belongs to the majority of neighboring points. Then the corresponding entries of the private voting vector can bet set to one. A single data point can only contribute to a single majority voting but a single data point can in theory change all global centroids. Thus, the sensitivity is equal to the number of allowed votes.
The procedure has to preserve privacy since it interacts with sensitive data of clients. Due to the fact that this second voting takes place in the exact same setting as the first voting we apply differential privacy in combination with secure summation again.
In the end, the top voted labels are assigned to the already published global top centroids to obtain a labeled set of data points on the basis of distributed and sensitive client data. It can be used in further learning tasks without posing a threat to the privacy of an individual.

7.2 Statistics

After a while, we assume that clients and the server have collected a set of labels through user feedback or multiple learning phases, respectively. Local or global labels can then be used to create basic statistics like histograms to get an overview about frequently occurring labels.
Statistics of global labels are trivial to obtain because they do not leak any sensitive information since they are generated in a privacy preserving way. In order to aggregate statistics from local labels one can apply our voting technique on the basis of differential privacy and secure summation.
The difference between statistics generated by local and global labels can be explained as follows. The number of local labels exceeds the number of global labels by far and thus

can lead to a better overview but they suffer from potentially huge noise due to differential privacy. Global labels on the other hand, are a rather small set but can be aggregated without any additional loss in utility.
Statistics of labels can be used to track frequently occurring labels over a long period of time. Turning back to our original motivation to present information in real-time, statistics can play an important role not only by providing frequencies for labels but also by keeping track of mobility in general.

8 Security and Privacy

We use this chapter to formulate security and privacy guarantees of our system and prove the latter. Security is mentioned before privacy because it is a necessary precondition of differential privacy. In the end, we discuss some properties of utility of a learning phase.

8.1 Security

In this section we discuss the security guarantees of our system. With security we ensure that sensitive data cannot be extracted by unintended third parties during a learning phase and as long as it is stored somewhere.
First, we discuss what parts of our system which have to provide security and then, we analyze security on both attacker scenarios described in Section 4.2.3.

8.1.1 Security-relevant Operations

A great part of a single learning phase happens locally on a client's device which includes generating embeddings and local clustering followed by a selection of top centroids. After a client publishes their data in a local privacy preserving way, the data is not considered sensitive anymore. Thus, each of the further processing steps is not subject to security since they entirely dependent on the local top centroids except the two distributed votings. A distributed voting via the secure summation protocol of [6] is clearly relevant for the security of a learning phase since clients have to entrust their private voting vectors to the server and thus rely on the guarantee of the applied protocol that the server cannot learn anything about their private input.
We further assume that clients are responsible for securing their personal devices against an attacker because such security guarantees would exceed the scope of this thesis.
In summary, the key to the security of our system is the security of our two distributed votings via the secure summation protocol of [6]. In the next subsections we discuss the security of the distributed voting in both attacker scenarios.

8.1.2 Semi-Honest

In this setting, a fraction of clients is controlled by a malicious adversary and the server behaves honest-but-curious.
A malicious client can only break its own local differential privacy but cannot affect the privacy of others and a malicious server only sees data published by clients. This means that the security of honest parties is not in danger since once local differential privacy holds on the local data there is no risk to security.
We assume a maximum fraction of dropouts ρ and a maximum fraction of corrupted clients γ. In the preliminaries in Definition 3.7 we show the general security result for the semi-honest version of the secure summation protocol. In summary, it states that given ρ and γ, it provides secure against an unbounded adversary without any further assumptions to clients, the server or the attacker.
Taking everything together, we state that the security of our learning phase holds as long as the device of honest clients is not corrupted which means that they can achieve local differential privacy and as long as the security of the secure summation protocol holds which it should, given that the fraction of dropouts is less than ρ and the fraction of corrupted clients is less than γ.

8.1.3 Semi-Malicious

Now, we additionally assume a corruption of the server by the same adversary already controlling a maximum fraction of γ clients.
The security of clients, despite the distributed voting, holds with a similar argumentation as before. A malicious server in combination with malicious clients has to break the security of a device of a client in order to prevent such a client from preserving local differential privacy. However, we assume that involved devices are secure right from the start. Therefore, we argue that the security of a learning phase even in the semi-malicious setting depends on the security of the distributed voting.
Considering the distributed voting, the applied secure summation protocol [6] is secure against PPT attacker given a maximum fraction of dropouts δ_{SecAgg} and a fraction γ_{SecAgg} of corrupted clients in the presence of a corrupted server. However, in contrast to the previous scenario, we have to make some assumptions. Again, we argue on the basis of the security statement of the protocol in our preliminaries in Definition 3.8.
First of all and this is also the reason for the prefix „semi“, the server has to behave honest-but-curious during Part I of the secure summation protocol. Furthermore, we have to accept a weaker attacker model, namely a PPT attacker in contrast to an unbounded one which also weakens the security. The guarantee that the server only learns the final sum

of honest clients' inputs has to be adjusted as well. Now, the server can learn partial sums consisting of at least a fraction of $\alpha \in [0, 1]$ honest clients' inputs.
Putting everything together, we state that the security of our system holds in the presence of corrupted clients and a malicious attacker. However, the server has to behave honest-but-curious during Part I of the protocol, we get a weaker security guarantee, a weaker attacker model suffices to break the security and the server can learn a bit more.

8.2 Privacy

In this section we analyze the privacy guarantees of our system and prove that it indeed preserves differential privacy. For the proof we assume that clients and the server are semi-honest and in the last two sections we analyze how dropouts and a malicious adversary affect privacy.
In order for differential privacy to hold for our system, data published during a learning phase including the final result has to be produced in a privacy preserving way. Data intended to remain under closure during and after a learning phase is not covered by our differential privacy analysis and its protection has to rely on the security guarantees of our system.
We assume the standard scenario of differential privacy which is that an attacker has knowledge about the entire distributed data set $\mathcal{X} = \{X_1, X_2, ..., X_n\}$ except a single data point which belongs to only one client's sensitive data. The goal is to hide the influence of this data point when analyzing any data which can be accessed during and after a learning phase.
To prove differential privacy for our system, we first analyze its different components and in the end, prove differential privacy to hold for the composition of all steps. Thereby, we follow the notion and steps presented in Table 5.1, Algorithm 1 and Section 5.

8.2.1 Local Top Centroids

In the next lines, we determine the privacy guarantees of the different steps from clustering sensitive data to selecting top centroids. It cumulates in Theorem 8.3 which proves local differential privacy for $\mathcal{M}_{\text{union\_local}}$.
First, client u applies DPLloyd-Impr [38] on their embeddings $C_u = \mathcal{M}_{\text{DPLloyd-Impr}}(X_u)$ which results in a set of k centroids. For this mechanism, there already exists a privacy proof which states that it preserves $(\varepsilon_1, 0)$-DP.
Afterward, clients use the report noisy max mechanism $\mathcal{M}_{\text{RNM}}$ to iteratively select k^* top centroids resulting in the set $C_u^* = \mathcal{M}_{\text{local\_top\_centroids}}(C_u, k^*)$. For report noisy max, we know that it preserves $(\varepsilon_2, 0)$-DP and by applying sequential composition we get that

$\mathcal{M}_{\text{local\_top\_centroids}}$ preserves $(k^* \cdot \varepsilon_2, 0)$-DP.
By again arguing via sequential composition, we find that the process of finding local top centroids $\mathcal{M}_{\text{local}} = \mathcal{M}_{\text{local\_top\_centroids}} \circ \mathcal{M}_{\text{DPLloyd-Impr}}$ is a $(\varepsilon_1 + k^* \cdot \varepsilon_2, 0)$-DP mechanism.

Theorem 8.1 (LDP of Local Top Centroids). *Given a distributed data set $\mathcal{X} = \{X_1, X_2, ..., X_n\}$, $k^* \in \mathbb{N}$ and $\varepsilon_1, \varepsilon_2 \in \mathbb{R}$ and differentially private mechanism $\mathcal{M}_{union\_local}$ with $(\varepsilon_1 + k^* \cdot \varepsilon_2, 0)$-DP with $\varepsilon_1, \varepsilon_2 > 0$. Then the mechanism $\mathcal{M}_{union\_local}(X_1, X_2, ..., X_n) = \bigcup_{i=0}^{n} \mathcal{M}_{local}(X_i)$ provides $(\varepsilon_1 + k^* \cdot \varepsilon_2, 0)$-LDP.*

Proof. The mechanism $\mathcal{M}_{\text{local}}$ serves as a local randomizer which clients apply on their local data to achieve $(\varepsilon_1 + k^* \cdot \varepsilon_2, 0)$-DP. Following the definition of LDP, $\mathcal{M}_{\text{union\_local}}$ can be viewed as the execution of an algorithm $\mathcal{A}$ on the local top centroids of all clients with $\mathcal{M}_{\text{union\_local}}(X_1, X_2, ..., X_n) = \mathcal{A}(\mathcal{M}_{\text{local}}(X_1), \mathcal{M}_{\text{local}}(X_2), ..., \mathcal{M}_{\text{local}}(X_n))$. Thus, the mechanism $\mathcal{M}_{\text{union\_local}}$ preserves $(\varepsilon_1 + k^* \cdot \varepsilon_2, 0)$-LDP. □

8.2.2 Global Clustering

The server performs a clustering π_{OPTICS} on the union of local top centroids which creates a set of global centroids $G = \pi_{\text{OPTICS}}(\bigcup_{i=0}^{n} C_u^*) = \pi_{\text{OPTICS}}(\bigcup_{i=0}^{n} \mathcal{M}_{\text{local}}(X_i))$. Since $\mathcal{M}_{\text{union\_local}}$ already preserves privacy, we can apply the post-processing theorem to show that $\mathcal{M}_{\text{global\_clustering}} = \pi_{\text{OPTICS}} \circ \mathcal{M}_{\text{union\_local}}$ provides the same privacy guarantees.

8.2.3 Distributed Voting

We use this section to prove that our distributed voting $\mathcal{M}_{\text{distributed\_voting}}$ provides differential privacy. We start by describing how the voting works internally and then give a proof for differential privacy.
The voting is basically a summation of distributed private voting vectors which gets applied twice, once to choose global top centroids and another time to find appropriate labels for those top centroids. Both times, a pre-processing $\pi_{\text{local\_majority}}$ or $\pi_{\text{local\_labeling}}$ is used to generate the private voting vectors. Both pre-processing algorithms provide a sensitivity equal to the number of allowed votes.
We apply the secure summation protocol π_{SecAgg} of [6]. It guarantees that the server only learns the final sum and nothing about individual inputs which we use in order to prove differential privacy in the central setting. For differential privacy to hold, we base our voting on a distributed version of the Laplace mechanism and use the fact that the Laplace distribution is closed under addition.
Every client locally adds a tiny fraction of Laplace noise to their private voting vector resulting in an appropriate amount of noise added to the final sum which suffices to prove differential privacy via the Laplace mechanism.

Theorem 8.2 (Privacy of Voting via Secure Summation [6]). *Given a distributed data set $\mathcal{X} = \{X_1, X_2, ..., X_n\}$, number of users $n \in \mathbb{N}$, number of voting candidates $d \in \mathbb{N}$, an arbitrary pre-processing algorithm $\pi_{Pre} : \mathcal{X} \to \mathbb{R}^d$ with a sensitivity of $\Delta = v$ and $\varepsilon, \delta \in \mathbb{R}$ with $\varepsilon > 0$ and $0 < \delta < 1$. Further assume an algorithm $\pi_{SecAgg} : V \to \mathbb{R}^d, V \subseteq \mathbb{R}^d$ realizing secure summation with advantage $\nu_1(\sigma) + \nu_2(\lambda)$ which can tolerate a maximum fraction of dropouts ρ as well as a maximum fraction of corrupted clients γ where ν_i are negligible functions including the assumptions presented in Theorem 3.7. Then the mechanism $\mathcal{M}_{distributed\_voting}(X_1, X_2, ..., X_n) = \pi_{SecAgg}(\vec{s}_1, \vec{s}_2, ..., \vec{s}_n)$ with $\vec{s}_i = \pi_{Pre}(X_i) + Lap\left(\frac{v}{(\varepsilon \cdot n \cdot (1-\rho-\gamma))}\right)$ preserves (ε, δ)-DP for a δ that is negligible in σ and λ. More precisely, $\delta = (1 + \exp(\varepsilon)) \cdot (\nu_1(\sigma) + \nu_2(\lambda))$.*

Proof. We first show $(\varepsilon, 0)$-DP for an ideal version of our mechanism $\mathcal{M}_{\text{distributed\_voting}}$ utilizing the ideal secure summation function $\mathcal{F}_{\text{SecAgg}}$ denoted as $\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}$. Then, we conclude that from Theorem 3.7 it follows that for the mechanism $\mathcal{M}_{\text{distributed\_voting}}$ utilizing the real protocol π_{SecAgg} (ε, δ)-DP holds for a negligible δ.
By applying the mechanism $\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}$, clients locally add Laplace noise of the form $\text{Lap}\left(\frac{v}{\varepsilon \cdot n \cdot (1-\rho-\gamma)}\right)$ to their private input. From the security of π_{SecAgg} we are guaranteed that the final sum consists of at least $n \cdot (1 - \rho - \gamma)$ private inputs of honest clients. By using the fact that the sum of Laplacian noise still follows a Laplace distribution, we get that $\vec{s} = \sum_{i=0}^{n \cdot (1-\rho-\gamma)} s_i + \text{Lap}\left(\frac{v}{\varepsilon \cdot n \cdot (1-\rho-\gamma)}\right) = \vec{s} + \text{Lap}\left(\frac{v}{\varepsilon}\right)$. Thus, $\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}$ preserves $(\varepsilon, 0)$-DP by applying the Laplace mechanism with a sensitivity of $\Delta = v$.
Hence, considering an unbounded attacker $\mathcal{A}$ and Theorem 3.7, we know that

$$\Pr\left[\mathcal{A}\left(\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}(D)\right) = 1\right] \leq \exp(\varepsilon) \Pr\left[\mathcal{A}\left(\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}(D \cup \{x\})\right) = 1\right]$$

By Theorem 3.7, we know that $\pi_{\text{SecAgg}}(s_1, \ldots, s_n)$ and $\mathcal{F}_{\text{SecAgg}}(s_1, \ldots, s_n))$ are perfectly indistinguishable. Hence, there are two negligible functions ν_1, ν_2 such that for any neighboring data sets D, D' (differing in at most one element) the following holds w.l.o.g.:

$$\Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D)\right) = 1\right] - \nu_1(\sigma) - \nu_2(\lambda) \tag{8.1}$$

$$\leq \Pr\left[\mathcal{A}\left(\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}(D)\right) = 1\right] \tag{8.2}$$

$$\leq \exp(\varepsilon) \Pr\left[\mathcal{A}\left(\mathcal{M}^{\mathcal{F}_{\text{SecAgg}}}_{\text{distributed\_voting}}(D')\right) = 1\right] \tag{8.3}$$

$$\leq \exp(\varepsilon)\left(\Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D')\right) = 1\right] + \nu_1(\sigma) + \nu_2(\lambda)\right) \tag{8.4}$$

$$\iff$$

$$\Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D)\right) = 1\right] \tag{8.5}$$

$$\leq \exp(\varepsilon) \Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D')\right) = 1\right] + (1 + \exp(\varepsilon)) \cdot (\nu_1(\sigma) + \nu_2(\lambda)) \tag{8.6}$$

From a similar argumentation it follows that

$$\Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D')\right) = 1\right] \tag{8.7}$$

$$\leq \exp(\varepsilon) \Pr\left[\mathcal{A}\left(\mathcal{M}^{\pi_{\text{SecAgg}}}_{\text{distributed\_voting}}(D)\right) = 1\right] + (1 + \exp(\varepsilon)) \cdot (\nu_1(\sigma) + \nu_2(\lambda)) \tag{8.8}$$

holds.
Hence, with $\delta := (1+\exp(\varepsilon)) \cdot (\nu_1(\sigma)+\nu_2(\lambda))$, we get that the mechanism $\mathcal{M}_{\text{distributed\_voting}}$ which uses π_{SecAgg} and is based on the Laplace mechanism with $\Delta = v$ is (ε, δ)-DP. As ν_1 and ν_2 are negligible, δ is negligible as well. □

8.2.4 Learning Phase

To evaluate the differential privacy guarantees of an entire learning phase, we first quantify the privacy of $\mathcal{M}_{\text{centroids}}$ generating representations. Then, we do the same for $\mathcal{M}_{\text{labels}}$ which assigns a label to each representation. In the end, we put everything together and prove that a learning phase $\mathcal{M}_{\text{learning\_phase}}$ provides differential privacy.
The mechanism $\mathcal{M}_{\text{centroids}}$ can be expressed as the following decomposition: $\mathcal{M}_{\text{centroids}} = \mathcal{M}_{\text{distributed\_voting}} \circ \mathcal{M}_{\text{global\_clustering}} \circ \mathcal{M}_{\text{union\_local}}$. Regarding sequential composition and the fact that LPD directly implies DP, we get that $\mathcal{M}_{\text{centroids}}$ preserves $(\varepsilon_1 + k^* \cdot \varepsilon_2 + \varepsilon_3, \delta_1)$-DP.
Using the same argumentation again, it is easy to see that $\mathcal{M}_{\text{labels}} = \mathcal{M}_{\text{distributed\_voting}} \circ \mathcal{A}_{\text{local\_labeling}}$ preserves $(\varepsilon_4, \delta_2)$-DP.
Finally, we can prove differential privacy for our system by providing a prove for a single learning phase.

Theorem 8.3 (Privacy of a Learning Phase). *Given a distributed data set $\mathcal{X} = \{X_1, X_2, ..., X_n\}$, $k^* \in \mathbb{N}$ and $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \delta_1, \delta_2 \in \mathbb{R}$ with $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4 > 0$ and $0 < \delta_1, \delta_2 < 1$ and two differentially private mechanisms $\mathcal{M}_{centroids}$ with $(\varepsilon_1 + k^* \cdot \varepsilon_2 + \varepsilon_3, \delta_1)$-DP, $\mathcal{M}_{labels}$ with $(\varepsilon_4, \delta_2)$-DP. Then, the mechanism $\mathcal{M}_{learning\_phase}(X_1, X_2, ..., X_n) = \mathcal{M}_{labels} \circ \mathcal{M}_{centroids}(X_1, X_2, ..., X_n)$ preserves $(\varepsilon_1 + k^* \cdot \varepsilon_2 + \varepsilon_3 + \varepsilon_4, \delta_1 + \delta_2)$-DP.*

Proof. By arguing with sequential composition, we directly get that the mechanism $\mathcal{M}_{\text{learning\_phase}}$ provides differential privacy in the sum of the privacy guarantees of the intermediate mechanisms $\mathcal{M}_{\text{centroids}}$ and $\mathcal{M}_{\text{labels}}$ which results in $(\varepsilon_1 + k^* \cdot \varepsilon_2 + \varepsilon_3 + \varepsilon_4, \delta_1 + \delta_2)$-DP. □

8.2.5 Privacy in the Semi-Malicious Setting

We can also prove a weaker form of differential privacy for the semi-malicious setting compared to the semi-honest variant. Since this setting mainly affects the secure summation protocol, it suffices to provide a proof of differential privacy for the distributed voting which can then directly be integrated into Theorem 8.3 which states the privacy of our system as a whole.

Theorem 8.4 (Privacy of Voting via Secure Summation [6]). *Given a distributed data set $\mathcal{X} = \{X_1, X_2, ..., X_n\}$, number of users $n \in \mathbb{N}$, number of voting candidates $d \in \mathbb{N}$, an arbitrary pre-processing algorithm $\pi_{Pre} : \mathcal{X} \to \mathbb{R}^d$ with a sensitivity of $\Delta = v$ and $\varepsilon, \delta, \alpha \in \mathbb{R}$ with $\varepsilon > 0$ and $\delta, \alpha \in [0, 1]$. Further assume an algorithm $\pi_{SecAgg} : V \to \mathbb{R}^d, V \subseteq \mathbb{R}^d$ realizing computational secure summation with advantage $\nu_1(\sigma) + \nu_2(\lambda)$ which can tolerate a maximum fraction of dropouts ρ as well as a maximum fraction of corrupted clients γ where ν_i are negligible functions including the assumptions presented in Section **??**. Then the mechanism $\mathcal{M}_{distributed\_voting}(X_1, X_2, ..., X_n) = \pi_{SecAgg}(\vec{s}_1, \vec{s}_2, ..., \vec{s}_n)$ with $\vec{s}_i = \pi_{Pre}(X_i) + Lap\left(\frac{v}{\varepsilon \cdot n \cdot \alpha \cdot (1-\rho-\gamma)}\right)$ preserves (ε, δ)-DP in the presence of a PPT attacker who behaves honest-but-curious during Part I of π_{SecAgg} for a δ that is negligible in σ and λ. More precisely, $\delta = (1 + \exp(\varepsilon)) \cdot (\nu_1(\sigma) + \nu_2(\lambda))$.*

Proof. By making the following adjustments, the proof is exactly the same as in Theorem 3.7.
Clients locally add Laplace noise of the form $Lap\left(\frac{v}{\varepsilon \cdot n \cdot \alpha \cdot (1-\rho-\gamma)}\right)$ because by the security of the semi-malicious variant of π_{SecAgg} we are guaranteed that an attacker can learn partial sums with at least $\alpha \cdot (1 - \gamma) \cdot n$ private inputs of honest clients.
In this setting, we know that $\pi_{\text{SecAgg}}(s_1, \ldots, s_n)$ and $\mathcal{F}_{\text{SecAgg}}(s_1, \ldots, s_n)$ are only computational indistinguishable. This means that we can only achieve a weaker form of privacy since, in contrast to the semi-honest setting where security holds in presence of an unbounded attacker, π_{SecAgg} is secure against a PPT attacker. □

8.3 Dangers to Utility

In the following paragraphs we discuss the impact of different scenarios on the utility of a learning phase.
Without giving formal guarantees, we want to note that the influence of the input of a single client on the final result including learned representations as well as labels is bounded. This comes from the fact that we assume a lot of clients participating in a single learning phase. Thus, in order for an adversary to influence the final result significantly, they have to control a huge number of clients so that the clustering result and also the voting results can notably be changed.

Correctness of the Distributed Voting The secure summation protocol on which the distributed voting is based provides correctness with a probability $1 - 2^{\eta}$ which generally means that correctness holds except for some worst case events which refers to a disrupted communication setup among clients.
If such events occur then it does not affect security [6] but rather leads to a case where the server cannot reconstruct the true sum. This destroys utility since such a sum does not reveal anything about honest clients' inputs due to the fact that security is independent of correctness.

Dropouts In general, a client dropping out can affect the utility of a single learning phase in three ways. First, if a client drops out right before publishing their data via local differential privacy, they can obviously not contribute anything to the final representations and labels. The other two ways refer to both distributed votings. If a dropout occurs right before the voting then this cannot be included in the summation process.

Malicious Client Malicious clients can generate poisoned data to destroy the utility of the global clustering or the distributed votings. Although we cannot exclude this possibility, this problem does not only occur with malicious clients. Assume a scenario where a client stores their smartphone in the backpack for an entire day. Then, the resulting audio recordings contain a lot noise. This example shows that such a problem is not specific to a malicious attacker scenario but rather should be considered right from the start. We leave a detailed analysis of the problem for future work.

Malicious Server Assuming a corrupted server presents a huge risk to utility. Such a server cannot only abort the protocol at every point in time, but also freely change any global clustering or voting result. In this case, utility cannot be protected at all.

9 Experiments and Evaluation

In this chapter, we first present details about the implementation of a secure summation protocol and how we trained a model via contrastive learning. This is followed by our experiments and a discussion of the results.

9.1 Implementation Details

In this section, we describe how we implement two main building blocks of our system, a machine learning model trained via contrastive learning [41] and a secure summation protocol [6].

9.1.1 Contrastive Learning

We implement the training process of contrastive learning in Python with the machine learning framework Tensorflow [3]. For training, we use a single A100 GPU with the underlying Nvidia CUDA multiprocessing framework.
We analyze different contrastive learning approaches [41, 11, 35] which are all based on audio data. Two of them, namely [35, 41], make their code publicly available and both implementations preserve a high quality.
The first work presents promising results on audio data in general and also on an acoustic scene classification. In this work, the authors develop a training framework called „Contrastive Learning for Audio Data“ in short „COLA“ [35]. The core idea is to train a neural network on an incredibly large dataset. The dataset is called Audioset [16] and is around 3.5TB in size with approximately $2.200.000$ audio files. The authors also apply a distributed and parallelized training strategy on multiple tensor processing units (TPUs). Such a training setup can hardly be reproduced by us even though we have access to multiple GPUs. Nevertheless, to get as close as possible to their results, we train on a balanced version of the original training dataset with around $22,000$ files. The training is quite fast compared to others. It takes less than 4 hours and the resulting embedding space reduces the number of dimensions of inputs to 1280.
We also train a model via contrastive learning based on the approach of [41]. The authors aim to train an audio fingerprint generator to perform a high-precision database search. In this process, the model gets trained on a smaller subset of the FMA music dataset [14]

resulting in a 128-dimensional embedding space. The training takes around 30 hours on 100 epochs on our setup.

9.1.2 Secure Summation

In order to realize a distributed summation process which scales well in the number of users and introduces a low overhead, we implement the protocol of [6]. Since to the best of our knowledge, there is no publicly available implementation, we implement it from scratch.
As a start, we implement the predecessor protocol „Practical Secure Aggregation for Privacy-Preserving Machine Learning "[10] on the basis of an existing implementation[1]. We rewrite parts of their implementation to fit our needs and refactor the code as well as the underlying database models. We also optimize it in a few ways. First, we extend the protocol such that multiple summations in immediate succession can be performed. Second, we tackle the problem that the protocol takes only one-dimensional vectors as input. Multi-dimensional vectors can now be summed up by using a flattening operation at the beginning. After the summation process, the program which executes the protocol can reshape the vector into the right shape again. Lastly, we introduce a precision parameter which determines the number of decimal points and scales inputs appropriately since the protocol is originally designed for the summation of integer.
We end up with a flexible version of the secure summation protocol of [6] in the semi-honest setting. We used Python and two cryptographic libraries, cryptography and sslib. The code consists of around 1500 lines and the backbone of the protocol is a Postgres database optimized for high frequency accesses.

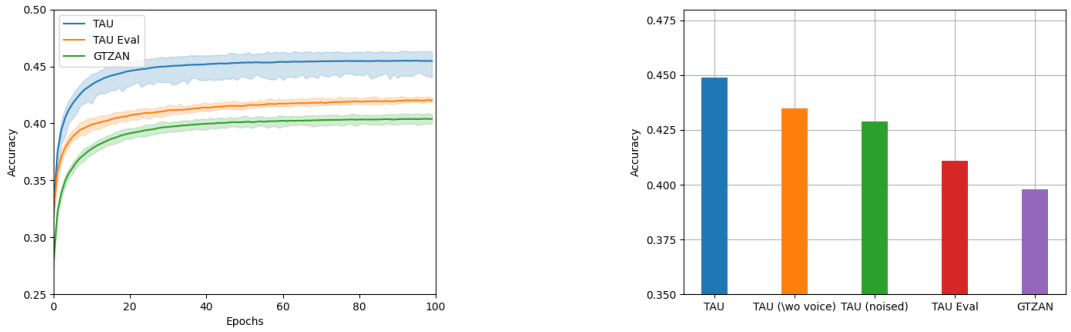
9.2 Experiments

This section starts with the general setup and then, we present our experiments by first describing the experiment itself and then discussing the respective results.

9.2.1 Experimental Setup

Before we discuss our experiments and their results, we present our general setup.
Beside our setup for training neural networks mentioned above, we perform all experiments on a Dell Lattitude 5320 with 16GB RAM and an 11th Gen Intel(R) Core(TM) $i7-1185G7$ processor. We also executed everything in sequential order and applied no parallelization.

[1] https://github.com/corentingiraud/federated-learning-secure-aggregation



(a) (b)

Figure 9.1: Evaluation of training a linear classifier on embeddings of the TAU dataset [43] obtained from different embedding spaces. On the left (a) we use three datasets to train embedding spaces. On the right (b) we do the same with additional pre-processing steps.

In all experiments, we consider the semi-honest adversary for secure summation, zero corrupted clients as well as zero dropouts.

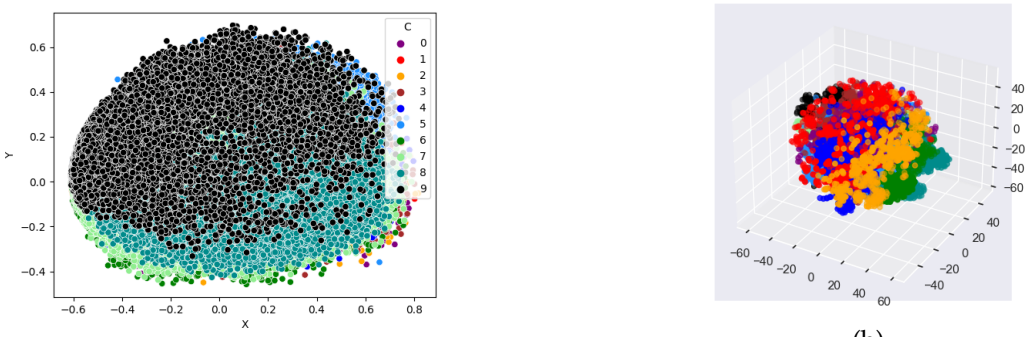
9.2.2 Datasets

To evaluate different aspects of our system we use three labeled datasets. The first one is a typical dataset for clustering and can be used to prove that the applied algorithm is generally able to partition data into clusters. The other two are realistic audio datasets used in practice for various audio-based machine learning learning tasks.

Synthetic This dataset is predestined for clustering approaches and can be used to generally prove that a clustering algorithm serves its purpose. We create it by using the Python framework Scikit-learn [34]. Specifically, we generated a handful (5 - 20) of two-dimensional so called „Blobs“ which are spherical clusters with a fixed center point, standard deviation and number of points. It is also useful to visualize different stages of our system as presented in Figure 5.1.

GTZAN [37] GTZAN is a collection of 10 genres with 100 hundred audio files each. It called the MNIST dataset [29] of sounds since, in contrast to other audio datasets, genres can be separated comparatively easy.

TAU [43] The „TAU Urban Acoustic Scenes 2020“ dataset consists of 10-seconds audio segments from 10 acoustic scenes. All together, it contains audio data of around 64 hours recorded with different devices so that models can be trained to tolerate



(a) (b)

Figure 9.2: Visualization of the embedding space of a model trained via constrastive learning on the TAU dataset [43]. On the left side (a), a PCA is used to project the embedding space into two dimensions and on the right (b) TSNE [39] is used to perform the same projection into three dimensions.

multiple audio sources. This dataset is of special interest for us because it can be used to train a model on acoustic scene classification.

9.2.3 Embedding Space

In this section, we explain which models we use for which dataset to conduct our experiments.

We only apply the second model obtained from the approach of [41]. The problem with the COLA-based model is that it barley learns anything. We argue that the lack of utility, in comparison to a model trained on the entire unbalanced version, mainly comes from the mentioned difference in size which leads to an enormous difference in variety of sound and noise sources. In other words, the smaller, balanced dataset is just not enough in order to be used for contrastive learning. In addition, the projection into a 1280-dimensional embedding space can also be a problem, since the dimension is still quite large for clustering. For these reasons, we do not use this model in our experiments.

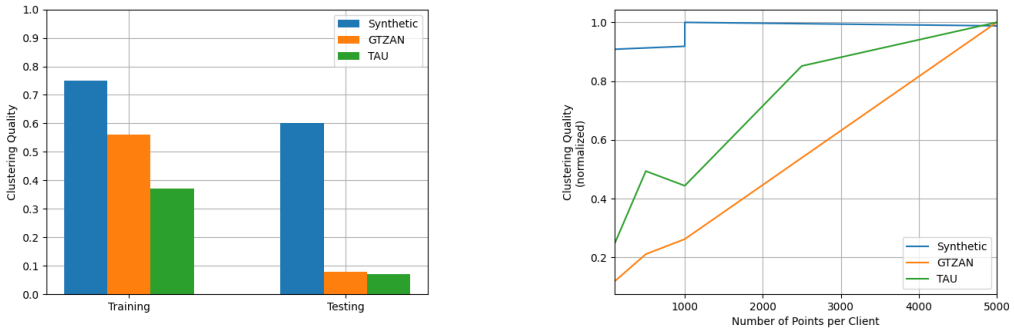
To evaluate the quality of an embedding space, we train a linear classifier on a subset of the resulting embeddings and report the testing accuracy. In Figure 9.1a we present the testing accuracy on embeddings of the TAU dataset from models trained on the TAU datasets, its evaluation subset and the baseline FMA music dataset. Note that we intentionally change the training dataset to TAU in order to improve the embedding space with regard to TAU since we are especially interested in acoustic scene classification.

We also conduct further trainings presented in Figure 9.1b by experimenting with different pre-processing setups. All models are trained on TAU again. The first four bars from

the left correspond to models with slight variations in the training such as removing vocals via a vocal remover called Spleeter [20] or adding additional noise to the audio data in every single batch. However, with different pre-processing strategies we cannot improve the quality of the embedding space, as the left bar shows, the baseline model of [41] still results in the highest testing accuracy of a linear classifier.
We choose the following embedding spaces for our datasets based on the experiments above. For GTZAN we use the base model of [41] as encoder trained on the FMA music dataset and for TAU we use a model trained on TAU. The synthetic dataset is applied without any pre-processing since it already lives in a low dimensional space.

9.2.4 Clustering

Before going into detail about how well clustering techniques work, we want motivate our idea to apply clustering on embedding spaces of models trained via contrastive learning. Therefore, we visualize the embedding space of a model trained on TAU in Figure 9.2 by using T-SNE [39], a dimension reduction technique, to reduce the number of dimensions to two or three, respectively.
The spherical structure can be explained as follows. In most clustering algorithms, the distance metric can be specified by a hyper-parameter. Oftentimes, the Euclidean distance is set as standard. In our experiments however, we use the cosine distance whenever we perform a clustering on audio data. The reason for this is that in the process of contrastive learning, one way to measure the similarity between datapoints is to use the cosine similarity. It can be seen as a scaled Euclidean distance because it ignores the magnitude of vectors but rather measures their distance on the unit circle. Thus, data in the embedding space seems to be arranged on a sphere-like shape and using the cosine distance instead of the Euclidean distance increases the clustering quality significantly.
To evaluate the clustering quality, we used the so called Adjusted Rand Score, a standard similarity measurement for two clusterings even if both use different labels. We use it to calculate the training and testing accuracy of a clustering. If one only calculates the score for the resulting clustering and the labels of the training data, then it is called training accuracy. On the other hand, if the score is based on all datapoints and not just the training set then one obtains the testing accuracy. The training accuracy tells us whether clustering works at all and the testing accuracy provides insights about the generalization capability of the learned clustering.
In general, whenever we report the clustering quality, the score is calculated with the final, global top centroids in comparison to the ground truth. Since we use labeled datasets, the ground truth of a given subset of datapoints can easily be obtained.
In the following two paragraphs, we present some experiments to evaluate the training as



(a) (b)

Figure 9.3: Evaluation of the clustering quality in terms of training and testing for three datasets. In (a) we compare the training quality with the testing quality. In (b) we evaluate the change in the training quality depending on the number of datapoints per client.

well as the testing accuracy of centroids produced by our system.

Growing Number of Datapoints We vary the number of datapoints per client and report the resulting clustering qualities in Figure 9.3b. We normalize the corresponding y-axis to achieve a better illustration of the increase in quality in the growing number of points since the clustering quality and the corresponding slope of the graph are quite small for both acoustic datasets. One important fact is that the clustering quality or in other words the generalization constantly increases. The results for the synthetic dataset are stagnating because it is comparably easy to learn and thus a small number of points already suffices to achieve a good clustering. The results in general are as expected, and can be seen as sanity check since it is only natural that more data points increase the learning capability.

Training vs. Testing Accuracy We evaluate both, training as well as testing accuracy. In Figure 9.3a we plotted both accuracy types for all three datasets. Although the clustering algorithm learns something, generalization only works for the synthetic dataset which shows that there is still a lot of room for improvement. We think that this mainly comes from the quality of the embedding space since the clustering algorithm seems to work in general. Another possible explanation which can slightly be seen in the visualization in Figure 9.2 is that the embedding space splits data into smaller islands. These islands themselves can be clustered but the generalization is quite bad because other islands which are not part of the training set are not considered at all.

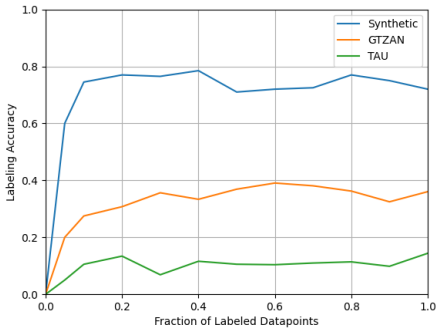


Figure 9.4: Evaluation of the influence of the fraction of labeled datapoints per client on the labeling accuracy.

9.2.5 Labeling

In this section, we want to focus on the quality of the obtained labels. To check the accuracy of our labeling process, we compare the resulting labels of the global top centroids with the labels of the corresponding nearest data points. We then calculate the fraction of correct labels and report the results in Figure 9.4.
Our labeling process depends on local labels of the clients which we obtain by asking clients via in-app polls. In this setting, we cannot assume that all local data points are labeled because, depending on the client, only a fraction of datapoints is labeled. Therefore, we evaluate the labeling accuracy based on the fraction of datapoints which are actually labeled while unlabeled datapoints are ignored.
Interestingly, the labeling quality does not change much from a fraction of 1.0 to 0.3 labeled data points per client. Only if the fraction of labeled datapoints is below 0.3, the labeling accuracy deteriorates. Also, a labeling accuracy of 0.0 makes perfect sense since if there are no labeled data points the voting consists of only noise due to differential privacy.
These results are surprisingly good because it means that a small fraction of labeled datapoints suffices to achieve a reasonable labeling accuracy.

9.2.6 Resources

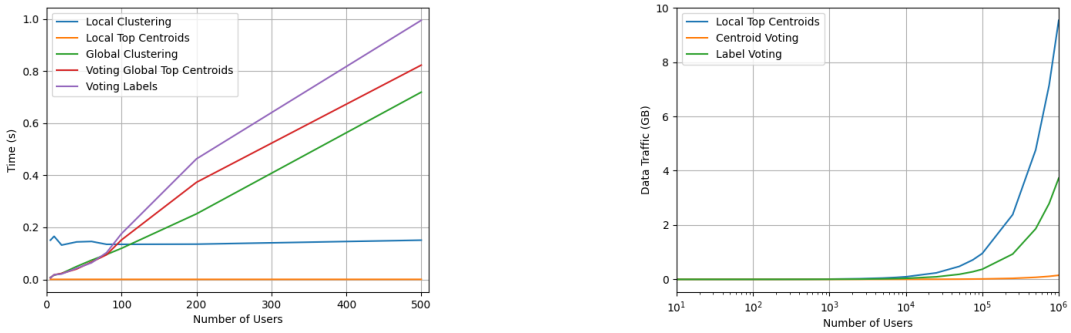
In this section, we present our experiments regarding the resource consumption of our system. We first describe the results of time measurements of certain operations and then we discuss the data traffic the server has to handle during a learning phase.

Time Measurements To measure the duration of different operations, we perform multiple learning phases and average the resulting time values. We constantly increase the number of clients to observe how certain operations scale in the number of clients. The results are illustrated in Figure 9.5a.
Both, the local clustering and the process of determining local top centroids, are operations on client-side, thus they are independent of other clients. The duration of the global clustering seems to behave linearly, however the complexity of OPTICS is in $O(n \cdot \log(n))$ which means that as soon as the number of clients gets larger, the growth will not be linear anymore. The remaining two operations which both depend on a distributed voting realized via secure summation, also seem to scale linearly in the number of clients. However, similar top OPTICS, the secure summation protocol requires $O(n \cdot \log(n) + n \cdot l)$ communication on server-side with l as the vector input length, thus a large number of users can quickly destroy the linear grows.

Incoming Traffic on server-side The only data traffic in our system which reaches a noteworthy scale is the traffic received by the server. In general, there are three phases in which the server receives data: the union of clients' local top centroids and an aggregated voting vector after each voting.
Following our notation illustrated in Table 5.1, we can set up the following equations describing the incoming data traffic for the server. Recall n as the number of users, k^* as the number of local top centroids, g as the number of global centroids produced by OPTICS, g^* as the number of global top centroids and $|L|$ as the number of labels. Denote F as the number of features with $F > L$. Then we get $T_{LTC} = n \cdot k^* \cdot F$ for the union of local top centroids, $T_{CV} = n \cdot g$ for the first voting of global centroids and $T_{LV} = n \cdot g^* \cdot L$ for the second voting of assigning labels to global centroids.
In Figure 9.5b, we exemplary illustrated the incoming traffic on server-side. With realistic values, one million users and an additional factor of 8 bytes (64 bit floating point values) per entry in a vector, one can see that we are still in the range of gigabytes.

9.2.7 Privacy-Utility Trade-Off

In Figure 9.6, we illustrate the privacy-utility trade-off of all parts our system. The trade-off gives insights in how privacy guarantees, in other words different values for ε, can influence the overall utility. The left column represents our system applied on the embeddings of the two acoustic datasets and the right column shows the results for the synthetic dataset. Note the log scale of the x-axis on the left. It is necessary because any reasonable ε in the range $[0, 1]$, as used in the case of the synthetic dataset, directly destroys utility. Still, one can observe a trade-off but only for incredible large ε values.



(a) (b)

Figure 9.5: Evaluation on the resource consumption of parts of our system. In (a) we measure the duration of different operations and in (b) we theoretically illustrate the data traffic the server has to handle during learning phase.

Each row stands for a different operation performed in a learning phase. The first row for the local clustering, the second row for determining local top centroids, the third row for the centroid voting and the last row stands for the label voting.
Surprisingly, the process of determining local top centroids seems to be quite detached from the actual privacy budget in all our three datasets.
In summary, we see that we get a reasonable privacy-utility trade-off for the synthetic dataset. Although the two acoustic datasets experience such a trade-off as well, the fact that ε has to be that high in combination with our utility analysis from above, shows that there is still a lot more work which needs to be done in order to achieve a reasonable clustering on embeddings under differential privacy.

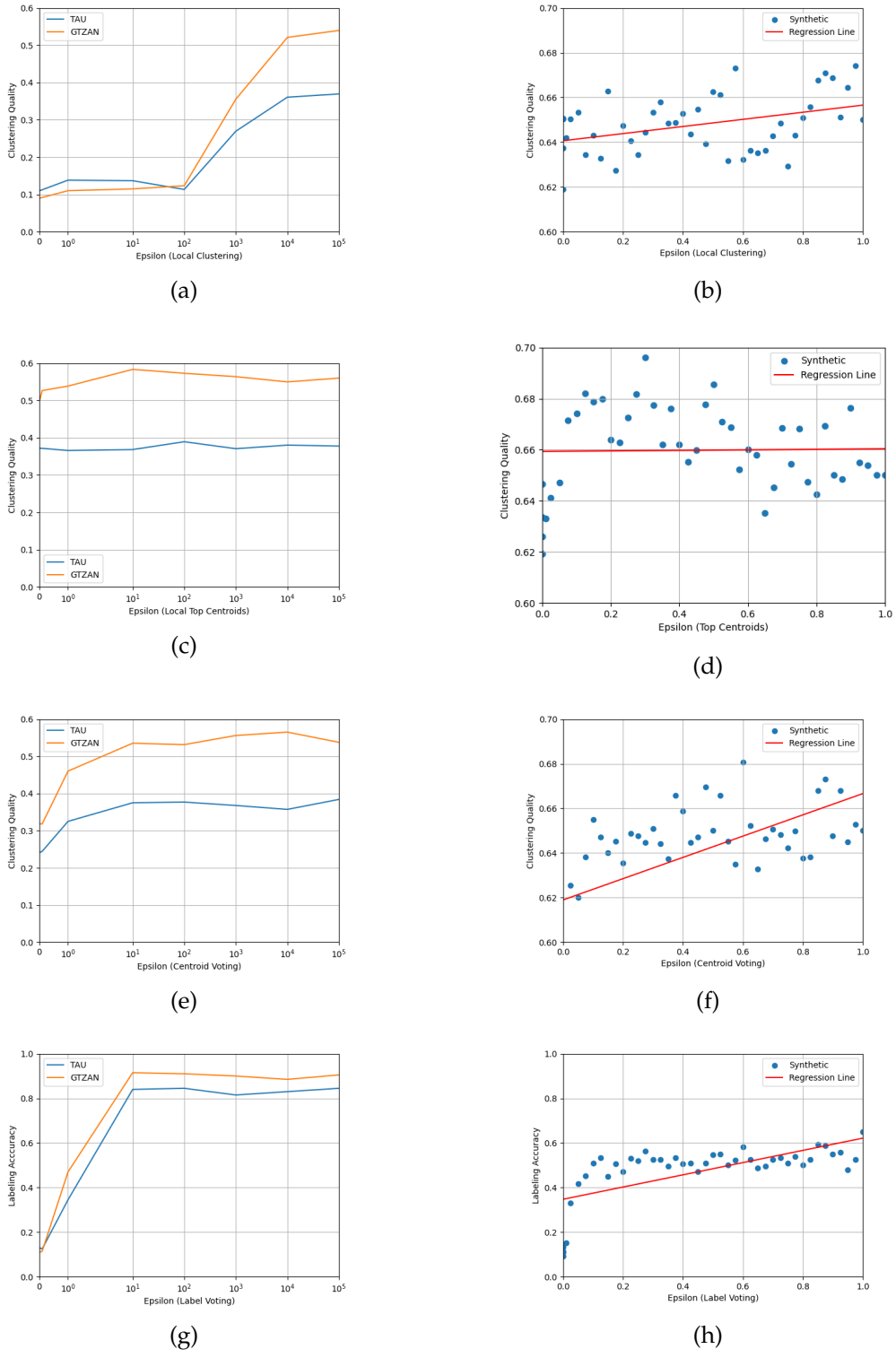


Figure 9.6: Evaluation of the privacy-utility trade-off for a learning phase. Each row shows the results for a specific operation in the order local clustering, determining local top centroids, global top centroids and the labeling process. The left column presents the results of two acoustic datasets and the right column the result of the synthetic dataset.

10 Conclusion

In this thesis, we design a system which realizes a two-phase distributed clustering. We utilize techniques from differential privacy and secure multiparty computation to protect the privacy of participating individuals and use contrastive learning to make high-dimensional audio data applicable for clustering. We also implement the secure summation protocol of [6] to realize an efficient distributed voting. In summary, our system is designed to consume less resources and scale well in the number of users compared to previous work in a setting based on the German Corona-Warn-App. Further, we prove that our system preserves differential privacy by taking into account the security and threat models of the applied secure summation protocol.
In the evaluation in Chapter 9, we tested our system on three data sets, a synthetic data set designed to evaluate fundamental properties of clustering and two real-world acoustic datasets [42, 37]. Starting with the synthetic data set, it is easy to see that our system produces reasonable representations and labels. However, regarding both acoustic datasets, there is still room for improvements. Results from the evaluation show that our system can handle such data without any problems, yet the quality of the clustering and the corresponding labels is inadequate. An extended hyper-parameter optimization could certainly improve the results, but this is unlikely to solve the problem entirely. One problem is that the differentially private clustering totally destroys the clustering quality whenever the corresponding privacy budget is chosen realistically. This problem needs to be investigated further. However, we think that the problem is of a fundamental nature and primarily comes from the neural network trained via contrastive learning. If the maximum accuracy which a linear classifier can reach is about 40% then there is clearly a lot of potential for improvements.
Nevertheless, due to the design of our system, we can treat both, clustering algorithms and the contrastive learning model, as black box. Once there exists a model achieving a much better separation as our current one, it can easily be integrated and an increase in the quality of the representations as well as in the labels should appear immediately.

10.1 Future Work

In this section, we describe possible work packages for future work.

Extended Hyper-Parameter Search Regarding the evaluation, the quality of the clusters and the accuracy of the labeling can be further increased. An extended ablation study should be performed to evaluate the role of all hyper-parameters in depth. This can also give more insights into properties which are important for the clustering quality, especially with regard to acoustic data.

Improving the Embedding Space In our work, we mainly use the approach of [41] to create embeddings. In contrast to other work like [35], there is a lot of room for improvements in the quality of the embedding space itself. We are certain that a better quality of the embedding space can increase the quality of the output of our system drastically.
In our experiments, we learned that data which comes from a model trained via contrastive learning is different in comparison to other data. The floating point numbers of embeddings are quite small. Thus, a scaling process can help to overcome this problem, such that noise added due to differential privacy has the intended effect.

Clustering The local clustering which preserves differential privacy works in general but when it comes to embeddings of acoustic data, the resulting clustering is nearly worthless. This is a major problem and it requires further investigation.
A simple yet efficient strategy to improve the local clustering could be the following one. Locally, the clustering gets executed multiple times with the best clustering as output. Although all results would preserve differential privacy, the best one cannot just serve as output since it leaks information about other clustering results due the decision making process. Thus, one could apply the approach of [32] which is basically a generalized DP argmax to output the best clustering.

References

[1]

[2] Infektionsketten digital unterbrechen mit der corona-warn-app. https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/WarnApp/Warn_App.html, 2021.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Fouad H Awad and Murtadha M Hamad. Improved k-means clustering algorithm for big data based on distributed smartphoneneural engine processor. *Electronics*, 11(6):883, 2022.

[5] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional euclidean spaces. In *International Conference on Machine Learning*, pages 322–331. PMLR, 2017.

[6] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[7] Malika Bendechache, A-Kamel Tari, and M-Tahar Kechadi. Parallel and distributed clustering framework for big spatial data mining. *International Journal of Parallel, Emergent and Distributed Systems*, 34(6):671–689, 2019.

[8] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, 2005.

[9] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[11] Sungkyun Chang, Donmoon Lee, Jeongsoo Park, Hyungui Lim, Kyogu Lee, Karam Ko, and Yoonchang Han. Neural audio fingerprint for high-specific audio retrieval based on contrastive learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3025–3029. IEEE, 2021.

[12] Jiecao Chen, Erfan Sadeqi Azer, and Qin Zhang. A practical algorithm for distributed clustering and outlier detection. *Advances in neural information processing systems*, 31, 2018.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[14] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. *arXiv preprint arXiv:1612.01840*, 2016.

[15] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L Lagendijk. Privacy-preserving distributed clustering. *EURASIP Journal on Information Security*, 2013(1):1–15, 2013.

[16] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780. IEEE, 2017.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[18] Xinlei He and Yang Zhang. Quantifying and mitigating privacy risks of contrastive learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 845–863, 2021.

[19] ZiXi Hee and Iftekhar Salam. Blockchain based contact tracing: A solution using bluetooth and sound waves for proximity detection. Cryptology ePrint Archive, Report 2022/209, 2022. `https://ia.cr/2022/209`.

[20] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020. Deezer Research.

[21] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 983–1000, 2021.

[22] Robert Koch Institut. Kennzahlen zur Corona-Warn-App. `https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/WarnApp/Archiv_Kennzahlen/Kennzahlen_01102021.pdf?__blob=publicationFile`, 2021.

[23] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. A new privacy-preserving distributed k-clustering algorithm. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 494–498. SIAM, 2006.

[24] Geetha Jagannathan and Rebecca N Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599, 2005.

[25] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Dbdc: Density based distributed clustering. In *International Conference on Extending Database Technology*, pages 88–105. Springer, 2004.

[26] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[28] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[29] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[30] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[31] Jinfei Liu, Joshua Zhexue Huang, Jun Luo, and Li Xiong. Privacy preserving distributed dbscan clustering. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 177–185, 2012.

[32] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309, 2019.

[33] Yuxiang Luo, Cheng Zhang, Yunqi Zhang, Chaoshun Zuo, Dong Xuan, Zhiqiang Lin, Adam C Champion, and Ness Shroff. Acoustic-turf: Acoustic-based privacy-preserving covid-19 contact tracing. *arXiv preprint arXiv:2006.13362*, 2020.

[34] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[35] Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879. IEEE, 2021.

[36] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[37] Bob L Sturm. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*, 2013.

[38] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the sixth ACM conference on data and application security and privacy*, pages 26–37, 2016.

[39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[40] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[41] Zhesong Yu, Xingjian Du, Bilei Zhu, and Zejun Ma. Contrastive unsupervised learning for audio fingerprinting. *arXiv preprint arXiv:2010.13540*, 2020.

[42] Zenodo. TUT Urban Acoustic Scenes 2018, Development dataset. `https://zenodo.org/record/1228142`, 2018.

[43] Zenodo. TAU Urban Acoustic Scenes 2020 Mobile, Development dataset. `https://zenodo.org/record/3819968`, 2020.

[44] En Zhang, Huimin Li, Yuchen Huang, Shuangxi Hong, Le Zhao, and Congmin Ji. Practical multi-party private collaborative k-means clustering. *Neurocomputing*, 467:256–265, 2022.

[45] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

[46] Yuchen Zhao, Hanyang Liu, Honglin Li, Payam Barnaghi, and Hamed Haddadi. Semi-supervised federated learning for activity recognition. *arXiv preprint arXiv:2011.00851*, 2020.

[47] Xiaojin Zhui and Zoubin Ghahramanilh. Learning from labeled and unlabeled data with label propagation. 2002.