Differentially Private Mixture of Generative Neural Networks*

Gergely Acs[1], Luca Melis[2], Claude Castelluccia[3], and Emiliano De Cristofaro[2]

[1]CrySyS Lab, BME-HIT, [2]University College London, [3]INRIA

Abstract

Generative models are used in a wide range of applications building on large amounts of contextually rich information. Due to possible privacy violations of the individuals whose data is used to train these models, however, publishing or sharing generative models is not always viable. In this paper, we present a novel technique for privately releasing generative models and entire high-dimensional datasets produced by these models. We model the generator distribution of the training data with a mixture of k generative neural networks. These are trained together and collectively learn the generator distribution of a dataset. Data is divided into k clusters, using a novel differentially private kernel k-means, then each cluster is given to separate generative neural networks, such as Restricted Boltzmann Machines or Variational Autoencoders, which are trained only on their own cluster using differentially private gradient descent. We evaluate our approach using the MNIST dataset, as well as call detail records and transit datasets, showing that it produces realistic synthetic samples, which can also be used to accurately compute arbitrary number of counting queries.

1 Introduction

Generative models represent an emerging area of machine learning, as recent progress has made it possible to artificially generate plausible samples of various kinds of data, including images, videos, texts, and music. These models are used, e.g. compression [46], denoising [6], inpainting [58], super-resolution [30], semi-supervised learning [41], clustering [47], etc. More specifically, generative models estimate the underlying distribution of a dataset and randomly generate realistic samples according to their estimated distribution. The real distribution-generating data is described with significantly fewer parameters than the number of available samples from this distribution. This "enforced compression" incentives the model to describe general features of the training data.

Ideally, such generalization should prevent the model from learning any individual-specific information. However, common algorithms often fail to provide such privacy guarantees and overfit on specific training samples by implicitly memorizing them. For example, in model inversion attacks [20], an adversary can use a trained model to make predictions of unintended (sensitive) attributes used as input to the model. Hence, even if only internal parameters are released, there might still be significant threats to the privacy of individuals whose data is used for training.

In this paper, we present a novel approach supporting the privacy-preserving release of generative models. While previous work explored the use of differential privacy in different areas of machine learning, including deep learning [1,36,43], privacy protection in generative models has not been explored so far.

Motivation. Generative models play an important role whenever entities need to publish their datasets, e.g., aiming to monetize it or allow third parties with the appropriate expertise to analyze it. For instance, Call Detail Records (CDRs) collected by telecommunication companies are not only useful to capture interactions between customers, but also to understand their behavior, e.g., for infectious disease spreading or migration patterns.[1] Rather than releasing only specific aggregate statistics, such as certain counting queries or histograms, one could share an "anonymized" dataset, which replaces the original data in possibly privacy-sensitive data analytics tasks. Alas, traditional anonymization models, such as k-anonymity, are not effective on high dimensional data, providing poor utility with insufficient privacy guarantees [3].

Intuition. A more promising approach is to model the data generating distribution by training a generative model on the original data, and only publish the model along with its (differentially private) parameters. Provided with this privacy-preserving model, anybody can generate a *synthetic* dataset resembling the original (training) data as much as possible without violating the strong protection of differential privacy. The intuition is that generative models have the potential to automatically learn the general features of a dataset including complex regularities such as the subtle and valuable correlation among different attributes.

Overview of the solution. Following this intuition, we propose a generative model that is a mixture of k generative artificial neural networks (ANNs). These ANNs are trained together and collectively learn the generator distribution of a dataset. The data is first divided into k clusters using a differentially private clustering approach, then each cluster is given to a separate generative neural network, such as Restricted Boltzmann Machines (RBM) [21] or Variational Autoencoders (VAE) [28], which are trained only on their own cluster using differentially private gradient descent.

Training distinct generative models on different partitions

*A shorter version of this paper appeared at the 17th IEEE International Conference on Data Mining (ICDM 2017). This is the full version, published in IEEE Transactions on Knowledge and Data Engineering (TKDE).

[1] See, e.g., http://www.flowminder.org.

of the dataset has several benefits. First, multiple models can generate more accurate synthetic samples than a single model trained on the whole dataset, as each ANN is trained only on similar data samples. This prevents the mixture model to generate unrealistic synthetic samples which may arise from the implausible combination of multiple very different clusters. This scenario is much more likely when the training is perturbed to guarantee differential privacy. Second, each ANN models a different component of the generator distribution, and hence learn any specifics of a cluster faster than a single model. In other words, a single model would need more training epochs than a mixture of generative models to achieve a comparably rich representation of the clusters. As each iteration of the learning algorithm requires some perturbation to guarantee privacy, a mixture model needs less noise which eventually yields more accurate model parameters.

Privacy Guarantees. Overall, our work builds on the Differential Privacy (DP) framework, specifically, using the Gaussian mechanism [18]. For clustering, we use a novel differentially private kernel k-means algorithm; kernel k-means [42] is a non-linear extension of the classical k-means algorithm and has been shown to be equivalent with most other kernel based clustering algorithms [16]. We first transform the data into a low-dimensional space using random Fourier features [38], and then apply a differentially private version of Lloyd's algorithm [8] to find the clusters in the data. Random Fourier features does not only make kernel k-means scalable for large datasets [15], but, unlike standard k-means [8], require to add limited amount of noise to guarantee privacy. Finally, when clusters are created, a generative model is trained on each cluster using differentially private stochastic gradient descent (SGD), which is a standard learning technique of many generative ANNs. Previous work adds constant amount of noise to the gradient update in each SGD iteration to guarantee differential privacy., whereas, we add noise to each gradient update which is tailored to the data.

We prove that our scheme guarantees differential privacy by using the *moment accountant* method, proposed in [1], which allows to quantify the privacy guarantee of the composition of differentially private mechanisms (e.g., noisy k-means iterations followed by noisy SGD iterations) much more accurately than previous work [19].

Contributions. In summary, we make several contributions:

1. We propose a novel approach, relying on generative neural networks, to model the data generating distribution of various kinds of data. It provides differential privacy to each individual in the training data, thus, it can be used to effectively "anonymize" and share large high-dimensional datasets with any potentially adversarial third party.

2. We design a novel differentially private clustering algorithm, combining kernel k-means with random Fourier features, which efficiently clusters high-dimensional large datasets with strong privacy guarantees.

3. We present a Differentially Private Generative Model (DPGM), where data is first clustered, using the differentially private kernel k-means, and then each cluster is given to separate generative neural networks, such as Restricted Boltzmann Machines or Variational Autoencoders, which are trained only on their own cluster using differentially private gradient descent.

4. We improve the differentially private gradient descent algorithm by Abadi et al. [1], using a novel adaptive perturbation technique. We adaptively re-compute the magnitude of the noise used to perturb the gradient updates in each SGD iteration, which can lead to significant accuracy improvement of the trained model.

5. We evaluate our approach on the MNIST dataset [29], a large Call Detail Records (CDR), and a transit dataset (TRANSIT); we show that our techniques provide realistic synthetic samples which can also be used to accurately compute arbitrary number of counting queries.

2 Related work

In this section, we review prior work on privacy-preserving mechanisms applied to data mining, machine learning, and deep learning.

Private Data Release. The k-anonymity [45] paradigm aims to protect data by generalizing and suppressing certain identifying attributes, however, it does not work well on high-dimensional datasets [3,10]. Therefore, rather than pursuing input sanitization, prior work has proposed techniques to produce plausible synthetic records with strong privacy guarantees, e.g., focusing on differentially private release of data [2,12,14,26,34,35,51]. Alas, these can often support only the release of succinct data representations, such as histograms or contingency tables.

Other mechanisms add noise directly to a generative model [9,32,33,59]. In this paper, we follow this approach, while, in a first-of-its-kind attempt, focusing on building private generative machine learning models based on neural networks. Other approaches [7,39,40] generate data records first, and then attempt to test their privacy guarantees, i.e., decoupling the generative model from the privacy mechanism. By contrast, we attempt to achieve privacy during the training of the model, thus avoiding eventual high sample rejection rates due to privacy tests.

Privacy in Deep Learning. Our work builds on the Differential Privacy (DP) framework, specifically, using the Gaussian mechanism [18]. Due to its generality, DP has served as a building block in several recent efforts at the intersection of privacy and machine learning [1,43]. In general, the majority of privacy-preserving learning schemes focus on convex optimization problems [4,13,52]. Also, training neural networks typically requires to optimize non-convex objective functions – as with Restricted Boltzmann Machine (RBM) [11] and Variational Autoencoder (VAE) [28] – which is usually done through the application of Stochastic Gradient Descent

(SGD) with poor theoretical guarantees. Wu et al. [52] introduce a privacy-preserving technique which runs SGD for convex cases for a constant number of iterations and only adds noise to the final output. By contrast, we introduce a novel differentially private SGD algorithm for optimizing general non-convex loss functions.

Shokri et al. [43] support distributed training of deep learning networks in a privacy-preserving way. Specifically, their system relies on the input of independent entities which aim to collaboratively build a machine learning model without sharing their training data. To this end, they selectively share subsets of noisy model parameters during training. However, their approach incurs high levels of privacy loss per entity, i.e., the ε parameter is in the order of thousands, using the strong composition theorem [19]. Abadi et al. [1] introduce an algorithm for non-convex deep learning models with strong differential privacy guarantees. They propose a privacy accounting method, called the moments accountant, which guarantees a tighter bound of the privacy loss for the composition of multiple gaussian mechanisms when compared to the strong composition theorem [19]. Our method also relies on the moments accountant to measure privacy loss, but we train generative models (i.e., unsupervised learning) and with an improved gradient descent, where the noise is carefully adjusted and injected in each iteration.

Also, Beaulieu et al. [5] apply the noisy gradient descent from [1] to train the discriminator of a Generative Adversarial Network [22] under differential privacy. The resulting model is then used to generate synthetic subjects based on the population of clinical trial data. In this paper, we rather use Variational Autoencoder which is trained with an improved version of the noisy gradient descent. Also, we apply a private clustering technique on the training data and create multiple generative models that produce higher-quality synthetic samples compared to a single model.

Differentially Private k-means has also been studied in prior work [44], however, aiming to find linearly separable clusters and add noise which is proportional to the data dimension m or the L_1-norm of data records. By contrast, our private kernel k-means approach can find even linearly non-separable clusters, and the added noise is independent of d as well as the norm of data points. Also, we offer a tighter privacy analysis using the moments accountant method from [1]. Kernel k-means clustering with random Fourier features (RFF) has already been considered in [15], albeit without any privacy guarantee. We somewhat combine [15] and [8], applying DP k-means on Fourier features to ultimately achieve better accuracy than [8].

Clustering and Generative Neural Networks. Prior work has also attempted to combine clustering with deep learning, though with no privacy guarantees. Some proposals [25,56,60] jointly train an autoencoder neural network with a clustering algorithm, and use the internal representation provided by the autoencoder, i.e., the encoder output, as features for clustering. A different training method is used in [17,31,54], where autoencoders are initially pre-trained, and then fine-tuned using the cluster assignment loss. Finally, other techniques [24,57] combine clustering with standard convolutional neural networks (CNNs) for representation learning of images.

3 Preliminaries

In this section, we review concepts used throughout the rest of the paper. We use the following notation: $\mathbb{I}$ denotes a universe of items (e.g., set of visited locations, pixels in an image, etc.), where $|\mathbb{I}| = m$. A dataset $D \subseteq 2^{\mathbb{I}}$ is the ensemble of all items of some set of individuals. A record, which is a non-empty subset of $\mathbb{I}$, refers to all items of an individual from D and is represented by a binary vector $\mathbf{x}$ of size m.

3.1 Restricted Boltzmann Machines (RBM)

A Restricted Boltzmann Machine (RBM) is a bipartite undirected graphical model composed of m visible and n invisible (or latent) binary random variables denoted by, respectively, $\mathbf{v} = (v_1, v_2, \dots, v_m)$ and $\mathbf{h} = (h_1, h_2, \dots, h_n)$. In our case, visible variables represent the attributes of D and their values are composed of records from D. Hidden variables capture the dependencies between different visible variables (i.e., dependencies between the items in $\mathbb{I}$). As the above model is a Markov random field with strictly positive joint probability distribution p over the model variables, p can be represented as a Boltzmann distribution defined as:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{1}$$

where $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ is the partition function, $E(\mathbf{v}, \mathbf{h})$ the energy function, i.e., $E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$, with w_{ij} being real valued weights describing the inter-dependency between v_j and h_i, and b_j, c_i real valued bias terms associated with the jth visible and ith hidden units, respectively. Using matrix notation, $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W} \mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$, where $\mathbf{W} = [\![w]\!]_{i,j}$, $\mathbf{c} = [c]_i$, and $\mathbf{b} = [b]_j$. The goal is to approximate the true data generating distribution with the Boltzmann distribution p, given in Eq. (1). To this end, we train the RBM model on dataset D to compute parameters $\mathbf{W}, \mathbf{c}, \mathbf{b}$.

There are a few algorithms to train RBMs, that approximate or relate to gradient descent on the log-likelihood of the data. If $\theta = (\mathbf{W}, \mathbf{b}, \mathbf{c})$, then we want to maximize the likelihood function $\mathcal{L}(\theta|D) = \prod_{\mathbf{x}\in D} p(\mathbf{x}|\theta)$ given dataset D, where $\mathbf{x} \in \{0,1\}^m$ is a record from D and p is the Boltzmann distribution defined in Eq. (1). A numerical approximation, gradient descent, is used where the model parameters θ are iteratively updated using D and the gradient of the log-likelihood function as: $\theta_{t+1} = \theta_t + \eta \frac{\partial \log \mathcal{L}(\theta_t|D)}{\partial \theta_t}$, with $\eta \in \mathbb{R}^+$ being the learning rate. The model parameters are updated until the log-likelihood converges. In this paper, we employ Persistent Contrastive Divergence [48].

3.2 Variational Autoencoder (VAE)

A variational autoencoder [28] consists of two neural networks (an *encoder* and a *decoder*), and a loss function. The encoder compresses data into a latent space (z) while the decoder reconstructs the data given the hidden representation. Let $\mathbf{x}$ be a random vector of m observed variables, which are either discrete or continuous. Let $\mathbf{z}$ be a random vector of n latent continuous variables. The probability distribution between $\mathbf{x}$ and $\mathbf{z}$ assumes the form $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z})$, where θ indicates that p is parametrized by θ. Also, let $q_\phi(\mathbf{z} \mid \mathbf{x})$ be a recognition model whose goal is to approximate the true and intractable posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$. We can then define a lower-bound on the log-likelihood of $\mathbf{x}$ as follows: $\mathcal{L}(\mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p_\theta(\mathbf{z})) + \mathrm{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$. The first term pushes $q_\phi(\mathbf{z} \mid \mathbf{x})$ to be similar to $p_\theta(\mathbf{z})$ ensuring that, while training, VAE learns a decoder that, at generation time, will be able to invert samples from the prior distribution such they look just like the training data. The second term can be seen as a form of reconstruction cost, and needs to be approximated by sampling from $q_\phi(\mathbf{z} \mid \mathbf{x})$.

In VAEs, we propagate the gradient signal through the sampling process and through $q_\phi(\mathbf{z} \mid \mathbf{x})$ using the *reparametrization trick*. This is done by making $\mathbf{z}$ be a deterministic function of ϕ and some noise ϵ, i.e., $\mathbf{z} = f(\phi, \epsilon)$. For instance, sampling from a normal distribution can be done like $\mathbf{z} = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. The reparametrization trick can be viewed as an efficient way of *adapting* $q_\phi(\mathbf{z} \mid \mathbf{x})$ to help improve the reconstruction. We train the Variational AutoEncoder using stochastic gradient descent to optimize the loss with respect to the parameters of the encoder and decoder θ and ϕ.

3.3 Kernel k-means with Random Features

Given a set of samples $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, k-means linearly separates D into k clusters $C_1, C_2, \ldots, C_k$ $(k \leq N)$ so that it aims to minimize the error $\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mathbf{c}_i||_2^2$, where $\mathbf{c}_i = \sum_{\mathbf{x} \in C_i} \mathbf{x}/|C_i|$ is the centroid of cluster C_i. Although this problem is NP-hard, there are efficient heuristic algorithms (such as Lloyd's algorithm) which iteratively refines clustering and converge quickly to a local optimum. However, k-means can provide very inaccurate clustering of linearly *non*-separable data, which are very common in practice. To overcome this shortcoming, kernel k-means [42] first maps samples from input space to a higher dimensional feature space through a non-linear transformation Φ, then applies standard k-means on $\{\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), \ldots, \Phi(\mathbf{x}_N)\}$. Hence, kernel k-means provides linear separators of clusters in feature space which correspond to non-linear separators in input space. Kernel k-means iteratively computes $||\Phi(\mathbf{x}) - \mathbf{c}'_i||_2^2$ for each sample $\mathbf{x}$ to decide which cluster a sample belongs to, where $\mathbf{c}'_i = \sum_{\mathbf{x} \in C_i} \Phi(\mathbf{x})/|C_i|$. To do so, the inner product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$ must be known for all $\mathbf{x}, \mathbf{y} \in D$. Since $\Phi(\cdot)$ is hard to explicitly compute due to its large, often infinite dimension, the kernel trick is applied; $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y})$, where κ is an easily computable kernel function. Still, this approach requires evaluating κ for all pairs of samples and store the results, which is not scalable for large datasets.

To make kernel k-means scalable, the kernel function can be approximated with low-dimensional explicit feature maps. In particular, the samples are first mapped to a low-dimensional Euclidean inner product space using an explicit random feature map $z : \mathbb{R}^m \to \mathbb{R}^d$ so that $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle \approx \langle z(\mathbf{x}), z(\mathbf{y}) \rangle$. Then, standard k-means is applied on the low-dimensional mapped samples $\{z(\mathbf{x}_1), z(\mathbf{x}_2), \ldots, z(\mathbf{x}_N)\}$ in $\mathbb{R}^d$ to approximate the result of the kernel k-means with implicit feature map Φ and kernel κ. The approximation error decreases exponentially fast as d increases, and quite accurate approximations can be obtained even for relatively small d. In particular, the approximation error is less than ξ with only $d = O(m\xi^{-2} \log \xi^{-2})$ dimensions [38]. Explicit nonlinear feature maps have already been proposed for shift-invariant kernels (e.g., generalized RBF kernels) [50] as well as polynomial kernels [37] among others.

3.4 Differential Privacy (DP)

Differential Privacy allows a party to privately release a dataset: using perturbation mechanisms, a function of an input dataset is modified, so that any information which can discriminate a record from the rest of the dataset is bounded [18].

Definition 1 (Privacy loss)**.** *Let $\mathcal{A}$ be a privacy mechanism which assigns a value Range$(\mathcal{A})$ to a dataset D. The privacy loss of $\mathcal{A}$ with datasets D and D′ at output $O \in$ Range$(\mathcal{A})$ is a random variable $\mathcal{P}(\mathcal{A}, D, D', O) = \log \frac{\Pr[\mathcal{A}(D)=O]}{\Pr[\mathcal{A}(D')=O]}$ where the probability is taken on the randomness of $\mathcal{A}$.*

Definition 2 ((ϵ, δ)-Differential Privacy [18])**.** *A privacy mechanism $\mathcal{A}$ guarantees (ε, δ)-differential privacy if for any database D and D′, differing on at most one record, and for any possible output $S \subseteq$ Range$(\mathcal{A})$, $Pr[\mathcal{A}(D) \in S] \leq e^{\varepsilon} \times Pr[\mathcal{A}(D') \in S] + \delta$ or, equivalently, $\Pr_{O \sim \mathcal{A}(D)}[\mathcal{P}(\mathcal{A}, D, D', O) > \varepsilon] \leq \delta$.*

This definition guarantees that every output of algorithm $\mathcal{A}$ is almost equally likely (up to ε) on datasets differing in a single record except with probability at most δ, preferably smaller than $1/|D|$. Intuitively, this guarantees that an adversary, provided with the output of $\mathcal{A}$, can draw almost the same conclusions about any individual no matter if this individual is included in the input of $\mathcal{A}$ or not [18].

Differential privacy maintains composition, i.e., if each of $\mathcal{A}_1, \ldots, \mathcal{A}_k$ is (ε, δ)-DP, then their k-fold adaptive composition[2] is $(k\varepsilon, k\delta)$-DP. However, a tighter upper bound can be derived on the privacy loss of the composite using a generic Chernoff bound. In particular, it follows from Markov's inequality that $\Pr[\mathcal{P}(\mathcal{A}, D, D', O) \geq \varepsilon] \leq \mathbb{E}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]/\exp(\lambda\varepsilon)$ for any output $O \in$ *Range*$(\mathcal{A})$ and $\lambda > 0$. This implies that $\mathcal{A}$ is (ε, δ)-DP with $\delta = \min_\lambda \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\varepsilon)$, where $\alpha_{\mathcal{A}}(\lambda) = \max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]$ is the log of the moment generating function of the privacy loss.

Theorem 1 (Moments accountant [1])**.** *Let $\alpha_{\mathcal{A}_i}(\lambda)$ be $\max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]$ and $\mathcal{A}_{1:k}$ the k-fold adaptive composition of $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$. It holds:*

1. $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{A}_i}(\lambda)$

[2] The output of $\mathcal{A}_{i-1}$ is used as input to $\mathcal{A}_i$, i.e., their executions are not necessarily independent except their coin tosses.

2. *$\mathcal{A}_{1:k}$ is $(\varepsilon, \min_\lambda \exp(\sum_{i=1}^{k} \alpha_{\mathcal{A}_i}(\lambda) - \lambda\varepsilon))$-differentially private*

where $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ use independent coin tosses.

There are a few ways to achieve DP, including the Gaussian mechanism [18]. A fundamental concept of all of them is the *global sensitivity* of a function [18].

Definition 3 (Global L_p-sensitivity). *For any function $f : \mathcal{D} \to \mathbb{R}^d$, the L_p-sensitivity of f is $\Delta_p f = \max_{D,D'} ||f(D) - f(D')||_p$, for all D, D' differing in at most one record, where $|| \cdot ||_p$ denotes the L_p-norm.*

Gaussian Mechanism. The Gaussian Mechanism (GM) [18] consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : \mathcal{D} \to \mathbb{R}^d$, GM is defined as $\mathcal{G}(D) = f(D) + [\mathcal{N}_1(0, \Delta_2 f \cdot \sigma), \ldots, \mathcal{N}_d(0, \Delta_2 f \cdot \sigma)]$, where $\mathcal{N}_i(0, \Delta_2 f \cdot \sigma)$ are i.i.d. normal random variables with zero mean and variance $(\Delta_2 f \cdot \sigma)^2$.

Lemma 1. $\alpha_{\mathcal{G}}(\lambda) = (\lambda^2 + \lambda)/4\sigma^2$

Proof. Let $f : \mathcal{D} \to \mathbb{R}$ be a scalar function, $f(D) = f(D') + \Delta_1 f$, where $\Delta_1 f = \Delta_2 f$, and $O = f(D) + x$, where $x \sim \mathcal{N}(0, \sigma)$. Let $\hat{\sigma} = \Delta_1 f \cdot \sigma$ Then, it holds:

$$\mathcal{P}(\mathcal{A}, D, D', O) = \ln\left(\frac{\Pr[\mathcal{G}(D) = O]}{\Pr[\mathcal{G}(D') = O]}\right) =$$

$$= \ln\left(\frac{\Pr[f(D) + \mathcal{N}(0, \hat{\sigma}) = O]}{\Pr[f(D') + \mathcal{N}(0, \hat{\sigma}) = O]}\right) = \ln\left(\frac{\exp(-x^2/2\hat{\sigma}^2)}{\exp(-(x + \Delta_1 f)^2/2\hat{\sigma}^2)}\right) =$$

$$= \ln\left(\frac{\exp(-x^2/2\hat{\sigma}^2)}{\exp(-(x + \Delta_1 f)^2/2\hat{\sigma}^2)}\right) = \left(\frac{\Delta_1 f}{\hat{\sigma}} \cdot \frac{x}{\hat{\sigma}}\right) + \frac{1}{2}\left(\frac{\Delta_1 f}{\hat{\sigma}}\right)^2$$

Since x is drawn from $\mathcal{N}(0, \hat{\sigma})$, $\mathcal{P}(\mathcal{A}, D, D', O)$ follows a normal distribution with mean $(\Delta_1 f)^2/2\hat{\sigma}^2$ and standard deviation $\Delta_1 f/\hat{\sigma}$, whose moment generating function is $\exp\left((\lambda^2 + \lambda)(\Delta_1 f)^2/4\hat{\sigma}^2\right)$. The claim follows from the definition of α and $\hat{\sigma}$. For the high-dimensional case when $f : \mathcal{D} \to \mathbb{R}^d$ $(d > 1)$, the proof is similar to that of Theorem A.1 in [18]. □

Given $\alpha_{\mathcal{G}}(\lambda)$, the exact privacy cost ε (or δ) of the k-fold adaptive composition of $\mathcal{G}$ is computed based on Theorem 1.

4 DPGM: Differentially Private Generative Model

In this section, we present our Differentially Private Generative Model (DPGM), which is detailed in Alg. 1 and illustrated in Figure 1. Table 1 summarizes notation and symbols used throughout the paper. The dataset D is first partitioned into k clusters, denoted by $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_k$, which are in turn used to train k distinct generative models, where the parameters of the resulting models are denoted, respectively, by $\theta_1, \theta_2, \ldots, \theta_k$. Data samples are similar within a cluster, thus, generative models simultaneously trained on each partition converge faster than a single model trained on the whole dataset D. As $\theta_1, \theta_2, \ldots, \theta_k$ are learnt using perturbed gradient descent, they can be released and used to generate synthetic data using the k generative models.

Our learning approach involves two main steps:

Algorithm 1: DPGM: Differentially Private Generative Model

```
  Input: Dataset: $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, # of custers: $k$, k-means
         iterations: $T_\mathcal{K}$, SGD iterations: $T_\mathcal{S}$, Noise scales:
         $\sigma_\mathcal{C}, \sigma_\mathcal{K}, \sigma_\mathcal{G}$
1 Cluster data records in $D$:
  $\{\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_k\} = \text{DPkmeans}(k, T_\mathcal{K}, D, \sigma_\mathcal{C}, \sigma_\mathcal{K})$
2 Initialize $\theta_1, \theta_2, \ldots, \theta_k$ randomly
3 for $t \in [T_\mathcal{S}]$ do
4     Select $(\hat{D}_s, \theta_s) \in \{(\hat{D}_1, \theta_1), \ldots, (\hat{D}_k, \theta_k)\}$ with
        probability $|\hat{D}_s|/|D|$
5     Update parameters of model $\theta_s$:
6     $\theta_s = \text{DP-SGD}(\hat{D}_s, \theta_s, \sigma_\mathcal{C}, \sigma_\mathcal{G})$ //see Alg. 4
  Output: $\theta_1, \theta_2, \ldots, \theta_k$
```

Symbol	Description
$\mathbf{x}$	binary vector
D	dataset
k	number of k-means clusters
$T_\mathcal{K}$	k-means iterations
$\hat{D}_1, \ldots, \hat{D}_k$	data clusters
$\theta_1, \ldots, \theta_k$	generative models
$\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k$	noisy cluster centers
$\sigma_\mathcal{C}, \sigma_\mathcal{K}, \sigma_\mathcal{G}$	noise scales
$C_{\max}$	max. norm bound
w	max. number of discretized norm bounds
κ	kernel function
z	randomized Fourier feature map
d	number of features
C_s	clipping threshold
$\mathcal{L}$	loss function
η	learning rate
L	batch size

Table 1: Notation and symbols used throughout the paper.

1. Records in D are clustered in a random feature space using differentially private kernel k-means (see Section 4.1) into clusters $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_k$; and

2. A generative model (e.g., RBM [21] or VAE [28]) with parameter θ_i is trained on cluster $\hat{D}_i$ (see Section 4.2) using differentially private gradient descent, where the training data are composed of the records of $\hat{D}_i$.

In each SGD iteration (Line 4-6 in Alg. 1), a model θ_s is chosen uniformly at random along with corresponding training data $\hat{D}_s$, and a single SGD iteration is performed to update θ_s using a random sample S of $\hat{D}_s$ with size L (Line 7 in Alg. 1). The output of our algorithm are the parameters of the trained generative models, i.e., $\theta_1, \theta_2, \ldots, \theta_k$. Finally, these privately trained k models can be used to generate synthetic records which resemble the original ones, i.e., preserve their general characteristics that are not specific to any single individual (as per ε and δ discussed in Section 5).

4.1 Private kernel k-means

We now discuss our private kernel k-means algorithm, presented in Alg. 2. It first transforms the data D into a low-dimensional representation $D' = \{z(\mathbf{x}_1), \ldots, z(\mathbf{x}_N)\}$ using randomized Fourier feature map $z : \mathbb{R}^m \to \mathbb{R}^d$ [38], and then

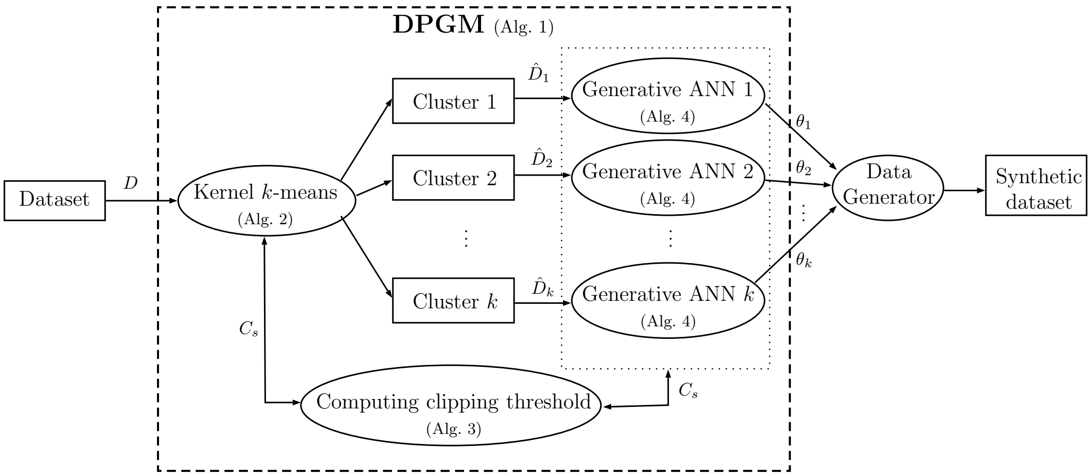


Figure 1: Overview of our differentially private generative model (DPGM).

applies standard differentially private k-means [8] on these low-dimensional features.

Specifically, $z : \mathbb{R}^m \to \mathbb{R}^d$ is defined as:

$$z(\mathbf{x}) = \sqrt{\frac{2}{d}}\left[\cos(\langle \mathbf{w}_1, \mathbf{x}\rangle + b_1), \ldots, \cos(\langle \mathbf{w}_d, \mathbf{x}\rangle + b_d)\right] \quad (2)$$

where each $\mathbf{w}_i \in \mathbb{R}^m$ is drawn independently from $p(\mathbf{w}) = \frac{1}{2\pi}\int_{\mathbb{R}^m} \exp(-j\langle \mathbf{w}, \mathbf{x}\rangle)\kappa(\mathbf{w})d\mathbf{x}$, i.e., $p(\mathbf{w})$ is the Fourier transform of kernel function κ, and $b_i \in \mathbb{R}$ is chosen from $[0, 2\pi)$ uniformly at random. In particular, Bochner's theorem implies that $p(\mathbf{w})$ is a valid probability density function, if κ is continuous, positive-definite, and shift-invariant kernel. Hence, $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y}) = \int_{\mathbb{R}^m} \exp(j\langle \mathbf{w}, \mathbf{x} - \mathbf{y}\rangle)p(\mathbf{w})d\mathbf{w} = \mathbb{E}_{\mathbf{w},b}[\langle \sqrt{2}\cos(\langle \mathbf{w}, \mathbf{x}\rangle + b), \sqrt{2}\cos(\langle \mathbf{w}, \mathbf{y}\rangle + b)\rangle]$, where the expectation is approximated with the empirical mean over d randomly chosen values of $\mathbf{w}$ and b [38].

Standard DP k-means [8] releases the noisy cluster centers computed iteratively using a noisy variant of Lloyd's algorithm; in each iteration, gaussian noise with scale $\sqrt{2}\sigma_{\mathcal{K}}$ is added to the size of all clusters, and with scale $\sqrt{2}\sigma_{\mathcal{K}}C_s$ to the sum of all cluster members in each cluster. These noisy values are used to compute the noisy cluster centers $\{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k\}$.[3] To determine the scale of the gaussian noise, the L_2-sensitivity of the cluster size and that of the sum of norms must be known within each cluster. Although the L_2-sensitivity of the set of cluster size is always $\sqrt{2}$ (a single record can change the size of at most 2 clusters), such *a priori* bound does not exist for the L_2-norm of the feature vectors in general. Hence, we need to clip all feature vectors in L_2-norm before applying standard DP k-means, where the clipping threshold C_s should be set to the average norm of the feature vectors (i.e., $(1/N)\sum_{\mathbf{x}\in D} ||z(\mathbf{x}_i)||_2$) and is approximated by Alg. 3. Replacing $z(\mathbf{x}_i)$ with $\hat{z}(\mathbf{x}_i) = z(\mathbf{x}_i)/\max(1, ||z(\mathbf{x}_i)||_2/C_s)$ guarantees that all feature vectors are kept as long as their norm is less then C_s, or they are scaled down to have a norm of C_s.

Algorithm 2: DPkmeans: Private kernel k-means with Random Fourier Features

```
Input: Data: $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, Cluster number: $k$,
       Iterations: $T$, Feature number: $d$, Kernel function: $\kappa$,
       Noise scales: $\sigma_{\mathcal{C}}, \sigma_{\mathcal{K}}$
Compute Features: $\mathbf{w}_i \sim_{\text{iid}} p(\mathbf{w})$ for all $1 \le i \le d$, where
  $p(\mathbf{w}) = \frac{1}{2\pi}\int_{\mathbb{R}^m} \exp(-j\langle \mathbf{w}, \mathbf{x}\rangle)\kappa(\mathbf{w})d\mathbf{x}$
$b_i \sim_{\text{iid}} \mathcal{U}[0, 2\pi]$ for all $1 \le i \le d$
$D' \leftarrow \{z(\mathbf{x}_1), \ldots, z(\mathbf{x}_N)\}$, where
  $z(\mathbf{x}) = \sqrt{2/d}[\cos(\langle \mathbf{w}_1, \mathbf{x}\rangle + b_1), \ldots, \cos(\langle \mathbf{w}_d, \mathbf{x}\rangle + b_d)]$
Clip Features: $C_s \leftarrow$ DPNorm($D', \sigma_{\mathcal{C}}$) //see Alg. 3
$\hat{D}' \leftarrow \{\hat{z}(\mathbf{x}_1), \ldots, \hat{z}(\mathbf{x}_N)\}$, where
  $\hat{z}(\mathbf{x}_i) = z(\mathbf{x}_i)/\max(1, ||z(\mathbf{x}_i)||_2/C_s)$
Initialize cluster centers $\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \ldots, \hat{\mathbf{c}}_k$ on public data
for $t \in [T]$ do
    for $i \in [k]$ do
        Assign: $\hat{D}_i \leftarrow \{\mathbf{x} : \arg\min_j ||\hat{z}(\mathbf{x}) - \hat{\mathbf{c}}_j||_2^2 = i\}$
        Update: $\hat{n}_i \leftarrow |\hat{D}_i| + \mathcal{N}(0, \sqrt{2}\sigma_{\mathcal{K}})$
        $\hat{\mathbf{c}}_i \leftarrow 1/\hat{n}_i \left(\sum_{\mathbf{x}\in\hat{D}_i} \hat{z}(\mathbf{x}) + \mathcal{N}(0, \sqrt{2}C_s\sigma_{\mathcal{K}}\mathbf{I})\right)$
Output: $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_k$
```

Nevertheless, for kernel functions like the Radial Basis Function (RBF)[4], a small norm bound C_s can be used (see Theorem 2). This bound is constant for any input data and feature size independently of the width γ of the RBF kernel. Thus, as opposed to standard k-means [8], our approach can detect linearly non-separable clusters, and, used with RBF kernel, add constant noise to feature vectors independently of their size d.

Theorem 2. *If* $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\gamma||\mathbf{x} - \mathbf{y}||_2)$*, then* $\mathbb{E}[||z(\mathbf{x})||_2] \le 1$ *for any* $\mathbf{x} \in \{0, 1\}^*$ *and* γ*, where the expectation is taken on the randomness of* z*.*

First, we introduce and prove Lemma 2.

Lemma 2. *Let* $\mathcal{N}(0, \sigma)$ *be a zero-centered normal random variable with standard deviation* σ*. Then:*

[3] We initialize clusters centers to random records drawn from *publicly available* non-sensitive data generated by the same distribution as the sensitive data. We only need k representative samples, and such public datasets already exist for images, location, and medical data.

[4] If the kernel function is RBF, i.e., $\kappa(\mathbf{x}, \mathbf{y})$=$\exp(-\gamma||\mathbf{x} - \mathbf{y}||_2)$, then $p(\mathbf{w})$ has zero centered gaussian distribution with standard deviation $2\gamma\mathbf{I}$.

1. $\mathbb{E}[\cos(\mathcal{N}(0,\sigma))] = \exp(-\sigma^2/2)$ *and* $\mathbb{E}[\sin(\mathcal{N}(0,\sigma))] = 0$,
2. $\mathbb{E}[\cos^2(\mathcal{N}(0,\sigma))] = (1+\exp(-2\sigma^2))/2$ *and* $\mathbb{E}[\sin^2(\mathcal{N}(0,\sigma))] = (1-\exp(-2\sigma^2))/2$

Proof of Lemma 2. Let $\exp(j\mathcal{N}(0,\sigma))$ denote a complex random variable. It follows from the moment generating function of $\mathcal{N}(0,\sigma)$ that:

$$\mathbb{E}[\exp(j\mathcal{N}(0,\sigma))] = \exp((j\sigma)^2/2) = \exp(-\sigma^2/2)$$

which means that:

$$\begin{aligned}\mathbb{E}[\cos(\mathcal{N}(0,\sigma)) + j\sin(\mathcal{N}(0,\sigma))] &= \mathbb{E}[\exp(j\mathcal{N}(0,\sigma))]\\ &= \exp(-\sigma^2/2)\end{aligned}$$

This implies that $\mathbb{E}[\cos(\mathcal{N}(0,\sigma))] = \exp(-\sigma^2/2)$ and $\mathbb{E}[\sin(\mathcal{N}(0,\sigma))] = 0$ due to the linearity of expectation. Hence:

$$\begin{aligned}\mathbb{E}[\cos^2(\mathcal{N}(0,\sigma))] &= \mathbb{E}[(1+\cos(2\mathcal{N}(0,\sigma)))/2]\\ &= (1+\exp(-2\sigma^2))/2\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[\sin^2(\mathcal{N}(0,\sigma))] &= \mathbb{E}[(1+\cos(2\mathcal{N}(0,\sigma)))/2]\\ &= (1-\exp(-2\sigma^2))/2\end{aligned}$$

where we used that $2\mathcal{N}(0,\sigma) = \mathcal{N}(0,2\sigma)$. □

Proof of Theorem 2. If $\kappa(\mathbf{x},\mathbf{y}) = \exp(-\gamma||\mathbf{x}-\mathbf{y}||_2)$, then $p(\mathbf{w}) = \frac{1}{2\pi}\int_{\mathbb{R}^m}\exp(-j\langle\mathbf{w},\mathbf{x}\rangle)\kappa(\mathbf{w})d\mathbf{x}$ has zero centered gaussian distribution with standard deviation $2\gamma\mathbf{I}$.

$$\mathbb{E}[||z(\mathbf{x})||_2] = \mathbb{E}\left[\left((2/d)\sum_{i=1}^{d}\cos^2(\langle\mathcal{N}(0,2\gamma\mathbf{I}),\mathbf{x}\rangle + \mathcal{U}[0,2\pi])\right)^{\frac{1}{2}}\right]$$

$$\leq \sqrt{\frac{2}{d}}\left(\sum_{i=1}^{d}\mathbb{E}\left[\cos^2(\langle\mathcal{N}(0,2\gamma\mathbf{I}),\mathbf{x}\rangle + \mathcal{U}[0,2\pi])\right]\right)^{\frac{1}{2}}$$

(by Jensen's inequality and the linearity of expectation)

$$\leq \sqrt{\frac{2}{d}}\left(\sum_{i=1}^{d}\mathbb{E}\left[\cos^2(\langle\mathcal{N}(0,2\gamma\mathbf{I}),\mathbf{x}\rangle)/2 + \sin^2(\langle\mathcal{N}(0,2\gamma\mathbf{I}),\mathbf{x}\rangle)/2\right]\right)^{\frac{1}{2}}$$

$$\leq \sqrt{\frac{1}{d}}\left(\sum_{i=1}^{d}\mathbb{E}\left[\cos^2(\mathcal{N}(0,2\gamma\sqrt{||\mathbf{x}||_1})\right] + \mathbb{E}\left[\sin^2(\mathcal{N}(0,2\gamma\sqrt{||\mathbf{x}||_1})\right]\right)^{\frac{1}{2}}$$

(by Lemma 2)

$$\leq 1$$

where, in the second inequality, we used that $\cos^2(a+b) = \cos^2(a)\cos^2(b) - 2\cos(a)\sin(a)\cos(b)\sin(b) + \sin^2(a)\sin^2(b)$, $\mathbb{E}[\cos(\mathcal{U}[0,2\pi])] = \mathbb{E}[\sin(\mathcal{U}[0,2\pi])] = 0$, $\mathbb{E}[\cos^2(\mathcal{U}[0,2\pi])] = \mathbb{E}[\sin^2(\mathcal{U}[0,2\pi])] = 0.5$. □

Therefore, DP kernel k-means has two main advantages over standard DP k-means [8]. First, kernel k-means can find linearly non-separable clusters. Second, if it is used with RBF kernel, the added noise is independent of the L_2-norm of the data records. As we show in Section 6, this can lead to much larger clustering accuracy especially for stringent privacy requirements (i.e., for $\varepsilon < 0.5$) even for large dimensional data.

Algorithm 3: DPNorm: Private Approximation of Average Norm

```
  Input: Data: $S = \{\mathbf{x}_{c_1},\dots,\mathbf{x}_{c_{|S|}}\}$, Noise scale: $\sigma_{\mathcal{C}}$, Max.
         norm bound: $C_{\max}$, Max. number of discretized norm
         bounds: $w$
1 $C_j \leftarrow j \cdot C_{\max}/w$ for $0 \leq j \leq w$
2 $C_s \leftarrow \arg\max_{j\geq 1}\{t_j + \mathcal{N}(0,\sqrt{2}\sigma_{\mathcal{C}})\}$, where
    $t_j = |\{\mathbf{x} \in S : C_{j-1} < ||\mathbf{g}(\mathbf{x})||_2 \leq C_j\}|$
  Output: $C_s$
```

Algorithm 4: Private Stochastic Gradient Descent

```
  Input: Data: $\hat{D}$, Model parameters: weights and biases $\theta$,
         Noise scales: $\sigma_{\mathcal{C}}$, $\sigma_{\mathcal{G}}$, Loss function:
         $\mathcal{L}(\theta) = \frac{1}{|\hat{D}|}\sum_i \mathcal{L}(\theta,\mathbf{x}_{c_i})$, Learning rate: $\eta$, Batch size:
         $L$
1 Sampling: Take a random sample $S = \{\mathbf{x}_{c_1},\dots,\mathbf{x}_{c_{|S|}}\}$ of $\hat{D}$
  with sampling probability $q = L/|\hat{D}|$
2 Compute Gradient: For each $\mathbf{x}_{c_i} \in S$, compute
  $\mathbf{g}(\mathbf{x}_{c_i}) \leftarrow \nabla_\theta\mathcal{L}(\theta,\mathbf{x}_{c_i})$
3 Clip Gradient: $S' \leftarrow \{\mathbf{g}(\mathbf{x}_{c_i}),\dots,\mathbf{g}(\mathbf{x}_{c_{|S|}})\}$
4 $C_s \leftarrow$ DPNorm$(S',\sigma_{\mathcal{C}})$ //see Alg. 3
5 $\hat{\mathbf{g}}(\mathbf{x}_{c_i}) \leftarrow \mathbf{g}(\mathbf{x}_{c_i})/\max\left(1,\frac{||\mathbf{g}(\mathbf{x}_{c_i})||_2}{C_s}\right)$
6 Add noise: $\tilde{\mathbf{g}} \leftarrow \frac{1}{L}\left(\sum_{i=1}^{|S|}\hat{\mathbf{g}}(\mathbf{x}_{c_i}) + \mathcal{N}(0,\sqrt{2}\sigma_{\mathcal{G}}C_s\mathbf{I})\right)$
7 Descent: $\theta \leftarrow \theta - \eta\tilde{\mathbf{g}}$
  Output: $\theta$
```

4.2 Private Stochastic Gradient Descent

We now present our private SGD technique, summarized in Alg. 4, considering a single SGD batch iteration. Our starting point is the work by Abadi et al. [1]: similar to theirs, our solution provides differential privacy to the training data by first clipping the norm of the gradient update of each record, and then perturbing these clipped gradients by the Gaussian mechanism. However, we achieve better accuracy as the clipping threshold is selected adaptively in each SGD iteration. In particular, in each SGD iteration, we also (1) compute the gradient of the loss function $\mathcal{L}$ on a random subset S of records (denoted as "batch") in Line 2 of Alg. 4, (2) clip the L_2 norm of the gradient of each record in S to have a norm at most C_s (in Line 3), (3) add gaussian noise $\mathcal{N}(0,\sqrt{2}\sigma_{\mathcal{G}}C_s\mathbf{I})$ to the average of these clipped gradient updates (Line 6), and finally (4) perform the descent step (Line 7). At the end, the updated model parameters θ are returned. A complete training epoch on the *whole* dataset D consists of $(|D|/L)$ SGD iterations, which are required to process all records in every cluster on average. Indeed, each record in a cluster $\hat{D}_s$ is selected with probability $(|\hat{D}_s|/\sum_{i=1}^{k}|\hat{D}_i|) \times (L/||\hat{D}_s|) = L/|D|$, where $\sum_{i=1}^{k}|\hat{D}_i| = |D|$. Notice that the L_2-sensitivity of $\sum_i \hat{\mathbf{g}}(\mathbf{x}_i)$ is $\sqrt{2}C_s$, as the norm of every $\hat{\mathbf{g}}(\mathbf{x}_i)$ is at most C_s, and one record can change at most two clusters.

4.3 Adaptive selection of the norm bound

Both our private SGD method (in Line 4 of Alg. 4) and private kernel k-means (in Line 4 of Alg. 2) require the differentially private computation of the average L_2-norm in a given set of records, which is then used as the clipping threshold C_s in both algorithms. For this purpose, these algorithms invoke DPNorm which is detailed in Alg. 3. In fact, our SGD technique differs from the original private SGD method [1] in the selection of the norm bound C_s (in Line 3-5 of Alg. 4). In the original approach [1], C_s is provided as input to the private SGD and no guideline is given how to compute its value without violating differential privacy. Moreover, the selection of the norm bound C_s has a large impact on the performance of the private SGD in general. If C_s is too small, there will be slow convergence. Conversely, if it is too large, unnecessarily large gaussian noise will be introduced on the gradient update. Intuitively, C_s should be adjusted so that $||\mathbf{g}(\mathbf{x}_{c_i})||_2 \approx C_s$ for each record $\mathbf{x}_{c_i}$. This guarantees that the contribution of $\mathbf{x}_{c_i}$ to $\tilde{\mathbf{g}}$ is maximally preserved with the smallest relative error. Hence, instead of fixing C_s for the whole training, we aim to compute C_s adaptively for each batch as $C_s = (1/L)\sum_i ||\mathbf{g}(\mathbf{x}_{c_i})||_2$. This adaptive approach would ensure fast convergence with small error, and also adapt to the gradient update of every batch. Indeed, SGD is iterative, so the gradient update $\tilde{\mathbf{g}}$ of a batch/iteration depends on that of the previous batch/iteration, which means that $(1/L)\sum_i ||\mathbf{g}(\mathbf{x}_{c_i})||_2$ is different for each batch.

In DPNorm (see Alg. 2), the computation of the average norm in a set S of records is randomized to guarantee privacy. A naive solution is to add Gaussian noise to this average, i.e., $C_s = (1/|S|)\sum_{\mathbf{x}\in S} ||\mathbf{x}||_2 + \mathcal{N}(0, s \cdot \sigma'/L)$, where $s \geq \max_{\mathbf{x}\in S} ||\mathbf{x}||_2$. However, $\max_{\mathbf{x}\in S} ||\mathbf{x}||_2$ is data-dependent and can be too large if there are outliers in S. Instead, we approximate C_s such that its value is close to the norm of many records in S, i.e., it is a good approximator of $(1/L)\sum_{\mathbf{x}\in S} ||\mathbf{x}||_2$. In particular, we discretize the domain of C_s by dividing $(0, C_{\max})$ uniformly into w intervals (Line 1 of Alg. 3). Then, we use the Gaussian mechanism (Line 2 of Alg. 3) to select among the upper bounds $C_j = jC_{\max}/w$ of these intervals $(0 \leq j \leq w)$, which will be the norm bound C_s for S. Specifically, we build a histogram where bin i equals the number of records whose gradient norm falls within $(C_{i-1}, C_i]$. Then, the (noisy) mode of this histogram is computed by adding independent gaussian noise $\mathcal{N}(0, \sqrt{2}\sigma_{\mathcal{C}})$ to each count, and selecting the bin which has the greatest noisy count. Note that the L_2-sensitivity of the histogram is always bounded by $\sqrt{2}$ no matter how large $\max_{\mathbf{x}\in S} ||\mathbf{x}||_2$ is.

4.4 Synthetic data generation

To generate an accurate synthetic dataset, data generation should mimic the training process; in order to generate a synthetic sample, a model with parameter θ_i is first selected randomly with probability $1/|\hat{D}_i|$, then a synthetic sample is generated using the selected model. This process is repeated until $|D|$ samples are obtained.

The above generation process ensures that low quality models which were not selected in training are also less likely to be used for data generation. In particular, though each model is trained during the same number of epochs *on its own cluster in expectation*, there is no guarantee that all k models will produce identical quality of synthetic samples due to randomization. Indeed, a cluster can potentially contain dissimilar samples, or it may be too small to be selected in Line 4 of Alg. 4 and hence fail to converge.

5 Privacy Analysis

In this section, we present the formal privacy analysis of DPGM. Recall that DPGM is the composition of private kernel k-means and private SGD. Let $\mathcal{K}$ denote the private kernel k-means algorithm whose output is the noisy mapped cluster centers after $T_{\mathcal{K}}$ clustering iterations (i.e., $\mathcal{K}(D) = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k\}$). $\mathcal{K}$ is composed of (1) selecting the norm bound using DPNorm and (2) $T_{\mathcal{K}}$ iterations of k-means. Let $\mathcal{G}_1$ denote the gaussian mechanism which selects the norm bound as per Section 4.3. A single k-means iteration is the 2-fold adaptive composition of two gaussian mechanisms $\mathcal{G}_2$ and $\mathcal{G}_3$ (in Line 10-11 of Alg. 2), where $\mathcal{G}_2$ perturbs the cluster size (Line 10), while $\mathcal{G}_3$ adds noise to the sum of Fourier features of the cluster members (Line 11). The L_2-sensitivity of the size of every clusters is $\sqrt{2}$, as changing a single record can change the size of at most two clusters. Similarly, the L_2-sensitivity of the sum of Fourier features of the cluster members is $\sqrt{2}C_s$ as it is detailed in Section 4.1.

Since $\mathcal{K}$ is the $T_{\mathcal{K}}$-fold adaptive composition of $T_{\mathcal{K}}$ clustering iterations, it follows from Theorem 1 and Lemma 1:

$$\begin{aligned}\alpha_{\mathcal{K}}(\lambda) &\leq T_{\mathcal{K}}(\alpha_{\mathcal{G}_1}(\lambda) + \alpha_{\mathcal{G}_2}(\lambda) + \alpha_{\mathcal{G}_3}(\lambda)) \\ &\leq T_{\mathcal{K}}(\lambda^2 + \lambda)(1/4\sigma_{\mathcal{C}}^2 + 1/2\sigma_{\mathcal{K}}^2) \qquad (3)\end{aligned}$$

Note that if the RBF kernel is used in kernel k-means (i.e., $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\gamma||\mathbf{x} - \mathbf{y}||_2)$ in Alg. 2), then $\alpha_{\mathcal{G}_1}(\lambda) = 0$ and $\alpha_{\mathcal{K}}(\lambda) \leq T_{\mathcal{K}}(\lambda^2 + \lambda)/2\sigma_{\mathcal{K}}^2$ since $C_s = 1$ is a priori bound on the L_2-norm of every feature vector (cf. Theorem 2).

Let $\mathcal{S}_k$ denote the private SGD algorithm whose output is the noisy model parameters after $T_{\mathcal{S}}$ SGD iterations (i.e., $\mathcal{S}(D) = \{\theta_1, \ldots, \theta_k\}$, computed in the last iteration of Alg. 2), and the input is the cluster centers $\{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k\}$ provided by $\mathcal{K}$. At the very beginning, $\mathcal{S}$ assigns each record to its closest cluster center in feature space to obtain k non-overlapping training sets (this is implemented by the last iteration in Alg. 2). Changing a single record alters at most a single record in at most 2 training sets (clusters), as the modified record can be moved from one to another training set. Since all training sets are non-overlapping, each record is selected in an SGD iteration with probability $q = (|\hat{D}_s|/|D|) \times (L/||\hat{D}_s|) = L/|D|$ for any k. Moreover, each of the k models are trained independently, so $\alpha_{\mathcal{S}_k}(\lambda) \leq \alpha_{\mathcal{S}_1}(\lambda)$, where $\mathcal{S}_1$ denotes the case when $k = 1$ (i.e., a single model is trained on the whole dataset D during $T_{\mathcal{S}}$ epochs).

The complete SGD training of $\mathcal{S}_1$ (Line 3-6 in Alg. 1) is the $T_{\mathcal{S}}$-fold adaptive composition of $T_{\mathcal{S}}$ SGD iterations, where we jointly use two perturbation mechanisms $\mathcal{G}_4$ and $\mathcal{G}_5$ in each iteration; $\mathcal{G}_4$ selects a batch uniformly at random and computes the norm bound C_s (Line 4 of Alg. 4) for this batch, then $\mathcal{G}_5$ selects the same batch and perturbs its gradient updates with gaussian noise whose magnitude is calibrated to C_s (Line 6 of Alg. 4). The composition of these two mechanisms uses independent source of randomness through different SGD iterations, hence we can use Theorem 1 to quantify the overall privacy. However, within a single iteration, $\mathcal{G}_4$ and $\mathcal{G}_5$ do not use independent source of randomness, although both mechanisms use independent gaussian noise but select the same batch S from the dataset. The following theorem computes $\alpha_{\mathcal{S}_1}(\lambda)$, and is a generalization of Theorem 1 when the component mechanisms can use dependent source of randomness.

Theorem 3 (General Moments Accountant)**.** *Let* $\alpha_{\mathcal{A}_i}(\lambda)$ *be* $\max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]$*, and* $\mathcal{A}_{1:k}$ *be the* k*-fold adaptive composition of* $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$*. Then:*

1. $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^{k} j_i \alpha_{\mathcal{A}_i}(\lambda / j_i)$

2. $\mathcal{A}_{1:k}$ *is* $(\varepsilon, \min_\lambda \exp(\sum_{i=1}^{k} j_i \cdot \alpha_{\mathcal{A}_i}(\lambda/j_i) - \lambda\varepsilon))$*-DP*

for any $\sum_{i=1}^{k} j_i = 1$*, where* $j_i > 0$ *and* $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ *can use dependent coin tosses.*

Proof. We adapt the proof of Theorem 2 in [1] to the case when the composite mechanism $\mathcal{A}_{1:k}$ consists of dependent mechanisms. Here we detail the complete proof for the sake of clarity.

Let $\mathcal{A}_{1:k}$ denote the composition of $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ and $O = (O_1, O_2, \ldots, O_k)$. Recall that, from Definition 2, $\mathcal{A}_{1:k}$ is (ε, δ)-DP, if $\Pr_{O \sim \mathcal{A}_{1:k}(D)}[\mathcal{P}(\mathcal{A}_{1:k}, D, D', O) > \varepsilon] \leq \delta$. Then:

$$\begin{aligned}
&\mathcal{P}(\mathcal{A}_{1:k}, D, D', O) \\
&\quad = \log \frac{\Pr[\mathcal{A}_{1:k}(D) = O]}{\Pr[\mathcal{A}_{1:k}(D') = O]} \\
&\quad = \log \prod_{i=1}^{k} \frac{\Pr[\mathcal{A}_i(D) = O_i | \mathcal{A}_{i-1}(D) = O_{i-1}, \ldots, \mathcal{A}_1(D) = O_1]}{\Pr[\mathcal{A}_i(D') = O_i | \mathcal{A}_{i-1}(D') = O_{i-1}, \ldots, \mathcal{A}_1(D') = O_1]} \\
&\qquad\qquad \text{(by the Chain rule)} \\
&\quad = \sum_{i=1}^{k} \log \frac{\Pr[\mathcal{A}_i(D) = O_i | \mathcal{A}_{i-1}(D) = O_{i-1}, \ldots, \mathcal{A}_1(D) = O_1]}{\Pr[\mathcal{A}_i(D') = O_i | \mathcal{A}_{i-1}(D') = O_{i-1}, \ldots, \mathcal{A}_1(D') = O_1]} \\
&\quad = \sum_{i=1}^{k} \mathcal{P}(\mathcal{A}_i, D, D', O_i) \qquad (4)
\end{aligned}$$

for any neighboring datasets D and D'. Hence,

$$\begin{aligned}
\alpha_{\mathcal{A}_{1:k}}(\lambda) &= \max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}_{1:k}, D, D', O))] \\
&= \max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)} \left[\exp\left(\lambda \sum_{i=1}^{k} \mathcal{P}(\mathcal{A}_i, D, D', O_i) \right) \right] \\
&\qquad \text{(by Eq. (4))} \\
&= \max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)} \left[\prod_{i=1}^{k} \exp\left(\lambda \mathcal{P}(\mathcal{A}_i, D, D', O_i) \right) \right] \\
&\leq \max_{D,D'} \log \prod_{i=1}^{k} \left(\mathbb{E}_{O_i \sim \mathcal{A}_i(D)} \left[\exp\left(\lambda \mathcal{P}(\mathcal{A}_i, D, D', O_i)/j_i \right) \right] \right)^{j_i} \\
&\qquad \text{(by the generalization of Hölder's inequality)} \\
&\leq \max_{D,D'} \sum_{i=1}^{k} j_i \log \left(\mathbb{E}_{O_i \sim \mathcal{A}_i(D)} \left[\exp\left(\lambda \mathcal{P}(\mathcal{A}_i, D, D', O_i)/j_i \right) \right] \right) \\
&\leq \sum_{i=1}^{k} j_i \max_{D,D'} \log \left(\mathbb{E}_{O_i \sim \mathcal{A}_i(D)} \left[\exp\left(\lambda \mathcal{P}(\mathcal{A}_i, D, D', O_i)/j_i \right) \right] \right) \\
&\leq \sum_{i=1}^{k} j_i \alpha_{\mathcal{A}_i}(\lambda/j_i) \qquad (5)
\end{aligned}$$

where we can apply the generalization of Hölder's inequality in the first inequality due to the fact that $\exp(\cdot)$ is always positive. Therefore,

$$\begin{aligned}
&\Pr[\mathcal{P}(\mathcal{A}_{1:k}, D, D', O) \geq \varepsilon] = \Pr[\exp(\lambda \mathcal{P}(\mathcal{A}_{1:k}, D, D', O)) \geq \exp(\lambda\varepsilon)] \\
&\leq \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}_{1:k}, D, D', O))] / \exp(\lambda\varepsilon) \\
&\qquad\qquad \text{(by Markov's inequality)} \\
&\leq \exp(\alpha_{\mathcal{A}_{1:k}}(\lambda) - \lambda\varepsilon) \leq \exp\left(\sum_{i=1}^{k} j_i \alpha_{\mathcal{A}_i}(\lambda/j_i) - \lambda\varepsilon \right) \quad \text{(by Eq. (5))}
\end{aligned}$$

The claim follows from Definition 2. □

Therefore, it follows from Theorem 1 and 3 that:

$$\begin{aligned}
\alpha_{\mathcal{S}_k}(\lambda) &\leq \alpha_{\mathcal{S}_1}(\lambda) \\
&\leq T_{\mathcal{S}} \cdot \min_{j_1, j_2 \in (0,1): j_1 + j_2 = 1} \left(j_1 \alpha_{\mathcal{G}_4}(\lambda/j_1) + j_2 \alpha_{\mathcal{G}_5}(\lambda/j_2) \right) \qquad (6)
\end{aligned}$$

We compute $\alpha_{\mathcal{G}_4}(\lambda)$ and $\alpha_{\mathcal{G}_5}(\lambda)$ similarly to [1]. That is, let $\mu_0(x|\sigma) = g(x|\sigma)$ and $\mu_1(x|\sigma) = (1-q)g(x|\sigma) + qg(x-1|\sigma)$, where $q = L/|D|$ is the probability that a record is included in the batch S of an SGD iteration and $g(x|\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$. Then, it holds:

$$\begin{aligned}
\alpha_{\mathcal{G}_3}(\lambda) &= \log \max(E_1(\lambda, \sigma_{\mathcal{C}}), E_2(\lambda, \sigma_{\mathcal{C}})) \\
\alpha_{\mathcal{G}_4}(\lambda) &= \log \max(E_1(\lambda, \sigma_{\mathcal{G}}), E_2(\lambda, \sigma_{\mathcal{G}}))
\end{aligned}$$

where

$$\begin{aligned}
E_1(\lambda, \sigma) &= \int_{-\infty}^{\infty} \mu_0(x|\sigma) \cdot \left(\frac{\mu_0(x|\sigma)}{\mu_1(x|\sigma)} \right)^{\lambda} dx \\
E_2(\lambda, \sigma) &= \int_{-\infty}^{\infty} \mu_1(x|\sigma) \cdot \left(\frac{\mu_1(x|\sigma)}{\mu_0(x|\sigma)} \right)^{\lambda} dx
\end{aligned}$$

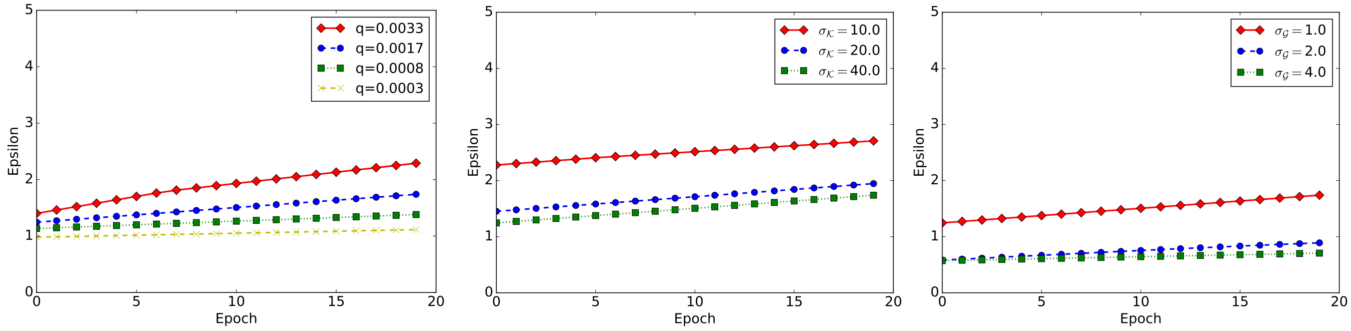
The next theorem immediately follows from Theorem 1 and Theorem 3.

Theorem 4. *Our differentially private generative model (DPGM) is* $\left(\min_\lambda \left(\alpha_{\mathcal{K}}(\lambda) + \alpha_{\mathcal{S}_k}(\lambda) - \log\delta\right)/\lambda, \delta\right)$*-differentially private for any fixed* δ*, where* $\alpha_{\mathcal{K}}(\lambda)$ *and* $\alpha_{\mathcal{S}_k}(\lambda)$ *are defined in Eq. 3 and 6.*

In this paper, we use the convention that $\delta = 1/|D|$, and compute ε numerically. Specifically, $\varepsilon = \min_\lambda \left(\alpha_{\mathcal{K}}(\lambda) + \alpha_{\mathcal{S}_k}(\lambda) - \log\delta\right)/\lambda$ is minimized over integer values of λ, where λ is usually no more than 100 in practice. The computation of $\alpha_{\mathcal{G}_3}$ and $\alpha_{\mathcal{G}_4}$ are performed through numerical integration, and it suffices to consider 10 different values of j_1 and j_2 in order to have a sufficiently small value of $j_1 \alpha_{\mathcal{G}_3}(\lambda/\ell_1) + j_2 \alpha_{\mathcal{G}_4}(\lambda/\ell_2)$ in Eq. 6. Therefore, in practice, given δ, an accurate approximation of ε can be obtained with negligible overhead.

6 Experimental Evaluation

In this section, we report the results of an experimental evaluation geared to compute the exact privacy guarantees of DPGM (presented in Alg. 1). We also analyze its performance in terms of the quality of generated samples as well as counting (linear) queries computed on the synthetic data. Counting queries provide the basis of many data analysis and learning algorithms (see [8] for examples). Finally, we measure the accuracy of our private kernel k-means described in Alg. 2.



(a) Sampling probability q ($\sigma_{\mathcal{G}} = 1.0$, $\sigma_{\mathcal{K}} = 40.0$) **(b)** Clustering noise $\sigma_{\mathcal{K}}$ ($\sigma_{\mathcal{G}} = 1.0$, $q = 0.0017$) **(c)** SGD noise $\sigma_{\mathcal{G}}$ ($\sigma_{\mathcal{K}} = 40.0$, $q = 0.0017$)

Figure 2: ε value as a function of the number of SGD training epochs for MNIST ($\delta = 10^{-5}, T_{\mathcal{K}} = 20$)

Dataset	$\lvert D\rvert$	$\lvert\mathbb{I}\rvert = m$	max $\lvert\lvert\mathbf{x}\rvert\rvert_1$	avg $\lvert\lvert\mathbf{x}\rvert\rvert_1$
MNIST	60,000	784	311.69	102.44
CDR	4,427,486	1303	422	11.42
TRANSIT	1,200,000	342	57	5.26

Table 2: The datasets used in our experiments: MNIST (images), CDR (call detail records), and TRANSIT (transport records).

6.1 Experimental Setup

Datasets. We use three datasets for our evaluations, summarized in Table 2. MNIST is a public image dataset [29], which includes 28×28-pixel images of hand-written digits, a total of $60,000$ samples. We vectorize and binarize each image to have binary data records with size $m = 784$. Throughout our experiments, we assume that each of the $60,000$ records originates from a different person. We also use an anonymized CDR (Call Detail Record) dataset provided to us by a cell phone operator. For this dataset, $\mathbb{I}$ represents the set of cell towers of the operator in a large city with $|D| = 4,427,486$ customers. We use a simplified version of the dataset, which contains the set of visited cell towers per customer within the administrative region of the city over 128.1 km^2, where the total number of towers is $m = 1,303$. The average number of individuals per tower over this period was $38,817$ with a standard deviation of $50,911$.

Finally, we experiment with a transit dataset, which we denote as TRANSIT in the rest of the paper.[5] Due to non-disclosure agreement, we are unable to provide specific details about the dataset, however, we can report that the TRANSIT dataset include the transit history of passengers in the network (with $|D| = 1,200,000$); here, $\mathbb{I}$ represents the set of $m = 342$ stations in a public transportation network.

Experimental Settings. For RBM, we set the number of hidden units to 200 and the learning rate is 0.01. The biases $\mathbf{b}$ and $\mathbf{c}$ are initialized to zeros, while the initial values of the weights $\mathbf{W}$ are randomly chosen from a zero-mean Gaussian with a standard deviation of 0.01. For VAE, the number of hidden units is set to 200 with single layer encoder and decoder, and a bi-dimensional latent space. We also used the rectifier activation function (ReLu) for all neurons and the Adam optimizer [27]. For our purposes, it is enough to compute $\alpha(\lambda)$ for $\lambda \leq 32$. We set the number of the private k-means iterations to 20 and $\delta = 1/|D|$. We also set $C_{\max} = 10$, $w = 100$ (in Alg. 3), as different values of these parameters do not have a strong impact on the results.

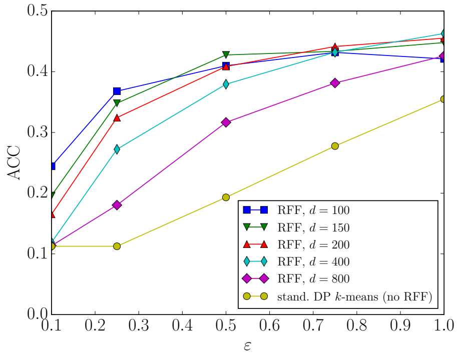


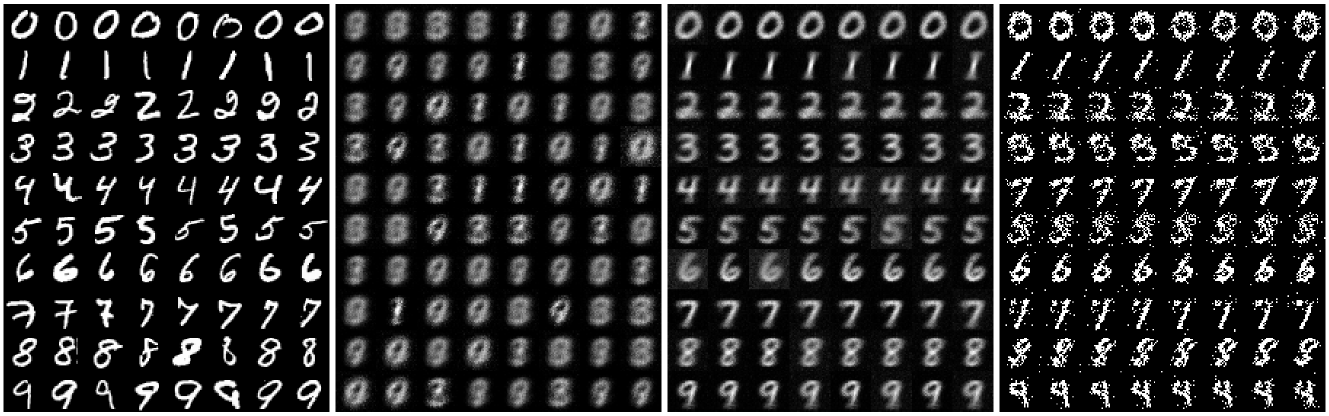
Figure 3: Clustering accuracy as a function of ε on MNIST ($\delta = 10^{-5}, T_{\mathcal{K}} = 20$).

We implement DPGM with both RBM (in C++) and VAE (in Python). Experiments are performed on a workstation running Ubuntu Server 16.04 LTS, with a 3.4 GHz CPU i7-6800K, 32GB RAM, and NVIDIA Titan X GPU card. Source code is available upon request.

6.2 Results with Image Dataset

Privacy guarantees. We report the privacy loss ε of DPGM (Alg. 1) in Figure 2 for the MNIST dataset. Recall that ε is computed from the noise level $\sigma_{\mathcal{C}}$, $\sigma_{\mathcal{K}}$, and $\sigma_{\mathcal{G}}$, the sampling probability q, the number of k-means iterations $T_{\mathcal{K}}$, and the number of SGD iterations $T_{\mathcal{S}}$ using Theorem 4. Figure 2 shows ε depending on the number of SGD training epochs, where one epoch consists of $\lceil 1/q \rceil$ SGD iterations. In Figure 2a–2c, we fix $\sigma_{\mathcal{C}} = 4.0$, and report the value of ε as a function of the number of epochs. We note that larger sampling probabilities (q) and more epochs yield larger values of ε, i.e., worse privacy guarantee. Figure 2b–2c show that larger values of $\sigma_{\mathcal{K}}$ and $\sigma_{\mathcal{G}}$ yield stronger privacy guarantees.

[5] Note that experiments using this dataset do not appear in the ICDM'17 version of the paper.



(a) Real samples **(b)** VAE w/o clustering **(c)** VAE with clustering **(d)** RBM with clustering

Figure 4: Real MNIST samples and samples generated from DPGM with RBM and VAE after 20 epochs ($\varepsilon = 1.74, T_{\mathcal{K}} = 20$). In **(c)** and **(d)**, each row contains 8 samples generated from a cluster.

Clustering accuracy. Next, in Figure 3, we compare the private kernel k-means (Alg. 2) with RBF kernel with standard DP k-means [8]. We evaluate the unsupervised clustering accuracy (ACC) [55], where $\text{ACC} = \max_u \frac{|\{\mathbf{x}:\mathbf{x}\in D \wedge label(\mathbf{x})=u(\mathcal{K}(\mathbf{x}))\}|}{|D|}$, $label(\mathbf{x})$ is the ground-truth label of sample $\mathbf{x}$[6], $\mathcal{K}(\mathbf{x})$ is the cluster assignment obtained by clustering algorithm $\mathcal{K}$, and u is a one-to-one mapping between cluster assignments and labels. The best mapping can be obtained using the Hungarian algorithm. To make a fair comparison, we fix C_s to $\sqrt{m} = 28$ for standard private k-means without RFF features, and $C_s = 1$ for private kernel k-means with RFF features based on Theorem 2 – i.e., we do not call DPNorm in either of the algorithms. We compute the clustering accuracy for different values of d depending on $\sigma_{\mathcal{K}}$, which directly yields the privacy bound ε using Eq. 3 and Theorem 1. Finally, we plot the average accuracy over 100 runs as function of ε in Figure 3.[7] Private kernel k-means is clearly superior to standard DP k-means, as the difference in clustering accuracy can be as large as 20%, especially for smaller values of ε. Shorter RFF features (i.e., smaller d) result in larger accuracy for smaller values of ε, whereas the reverse holds for larger ε. The reason is that the clustering error is determined by the trade-off between (1) the perturbation error due to the Gaussian noise, which is added to the cluster centers in Line 11 of Alg. 2, and (2) the approximation error caused by the low-dimensional embedding z in Line 3 of Alg. 2. In particular, the perturbation error increases if ε decreases or d increases. Indeed, when the distance $||\hat{z}(\mathbf{x}) - \hat{\mathbf{c}}_j||_2^2$ to each cluster center $\hat{\mathbf{c}}_j$ is computed in Line 9 of Alg. 2, the total perturbation of this distance value is obtained by aggregating the noise values on each coordinate of $\hat{\mathbf{c}}_j$, and hence the perturbation error is proportional to the size d of vector $\hat{\mathbf{c}}_j$ as well as to ε^{-1}. On the other hand, larger d decreases the approximation error introduced by z. One can find a good trade-off between the approximation and the perturbation error by adjusting d and ε through experiments using publicly available data. For the rest of experiments, we set d to 200.

Selecting the optimal number of clusters k for kernel k-means can be qualitatively and visually done by relying on dimensionality reduction algorithms (e.g., t-SNE [49]). To this end, one can use public data sampled from the same underlying distribution, and therefore not requiring to make the parameter selection step differentially private. For MNIST we set $k = 10$, while we select only one cluster for the CDR dataset. We investigate the effects of different values of k for the transit dataset.

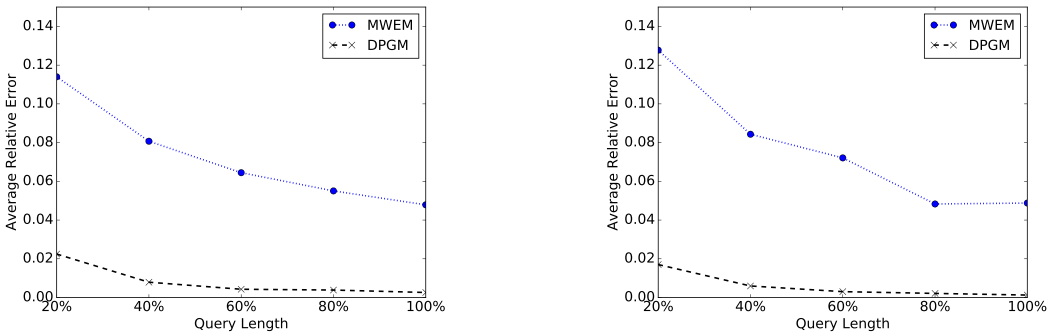
Synthetic Samples. As training progresses, the synthetic samples produced by the generative models should resemble the true samples. To evaluate model quality, we show the synthetic samples obtained at epoch 20 in Figure 4 from a Restricted Boltzmann Machine and a Variational Autoencoder with $k = 10$ clusters on MNIST. For this experiment, we set $q = 0.0017$ for a final privacy budget ε of 1.74, and performed $T_{\mathcal{K}} = 20$ clustering iterations before training the generative neural networks. Overall, the samples generated from VAE (Fig. 4c) provide better visual quality than the ones generated from the RBM (Fig. 4d). Note that the samples generated from the VAE without our private clustering technique (Figure 4b) have bad visual quality. Finally, we report additional samples with a multi-layer VAE in Appendix A.

6.3 Results with CDR and transit dataset

We consider counting queries which are specified by a predicate function $p : D \rightarrow \{0, 1\}$ and return the number of users in the dataset which satisfy the given predicate p, i.e., $Q_p(D) = \sum_{\mathbf{x}\in D} p(\mathbf{x})$. We evaluate the accuracy of counting queries on a synthetic dataset generated by DPGM from our call-data-record (CDR) dataset with roughly 4 million users and the transit dataset with roughly 1 million users (see in Table 2). A single query is defined by a subset of tower cells, and returns the number of users in D who visited these cells. We compare DPGM with MWEM [23], which is a *de facto* standard differentially private mechanism to answer counting queries.

[6] For MNIST, these are digits ranging from 0 to 9.

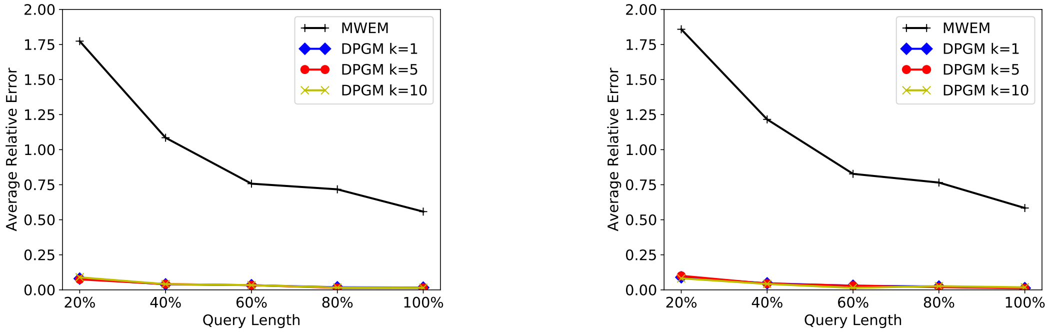
[7] Standard deviation of accuracy is < 0.05 for all values of ε and d.



(a) $\varepsilon = 2.0$

(b) $\varepsilon = 1.0$

Figure 5: Average relative error vs. ε for the CDR dataset ($q = 2.2 \cdot 10^{-5}$, $\delta = 4.4 \cdot 10^{-6}$)



(a) $\varepsilon = 2.0$

(b) $\varepsilon = 1.0$

Figure 6: Average relative error vs. ε for the transit dataset ($q = 10^{-4}$, $\delta = 10^{-6}$)

As done in previous work [53], we measure the utility of a counting query Q_p over the sanitized dataset $\hat{D}$ by its relative error w.r.t. the actual result over the raw dataset D. The relative error of Q_p is thus computed as $\frac{|Q_p(\hat{D})-Q_p(D)|}{\max\{Q_p(D),s\}}$, where s is a sanity bound that weight the influence of the queries with small selectivities. Following the convention, the sanity bound is set to 0.1% of the dataset size.

First, we examine the relative error of counting queries with respect to privacy loss ε. $1,000$ counting queries are randomly generated with different number of tower cells, which we refer as the length of the query. Each query set is divided into 5 subsets such that the query length of the i-th subset is uniformly distributed in $\left[1, \frac{i \cdot \max ||\mathbf{x}||_1}{5}\right]$ and each item is randomly drawn from universe of items.

Fig. 5 reports the average relative error for each query set. This shows that our approach clearly outperforms MWEM. The error of DPGM ranges from 0.017 for 20% query length to 0.0012 for 100% when $\varepsilon = 1.0$. Weaker privacy guarantee (larger values of ε) lead to slightly smaller errors (Fig. 5b). By contrast, the error of MWEM[8] ranges from 0.11 to 0.05 even for $\varepsilon = 2$. Also note that the synthetic data produced by DPGM allows the evaluation of arbitrary number of type of queries, not only linear counting queries.

Finally, Fig. 6 reports the average relative error for the transit dataset with different number of clusters k. While our approach, whose average relative error ranges from 0.09 to 0.02, significantly outperforms MWEM, the number of clusters does not affect the error of counting queries on transit dataset. These results might be an artifact of the dataset itself. We also report additional results with a multi-layer VAE for both the CDR and Transit dataset in Appendix A.

7 Conclusion

This paper presented a first-of-its-kind attempt to build private generative machine learning models based on neural networks. Specifically, we presented a novel differentially private generative model (DPGM), relying on a mixture of k generative neural networks: such models can be used to generate and share synthetic high-dimensional data with provable privacy. We evaluated the performance of the model on real datasets, showing that our approach provides accurate representation of large datasets with strong privacy guarantees and high utility. As part of future work, we plan to combine VAE and a Gaussian Mixture model for clustering, similar to [60], albeit with strong privacy guarantees. The effective privacy-preserving training of deep neural networks with multiple hidden layers is also desirable in order to generate more complex data such as personal photos or various sequential data. Finally, we plan to pilot deploy our techniques in the wild.

[8] After clipping each record to have L_1-norm $\text{avg}||\mathbf{x}||_1 = 12$, the sensitivity of queries is set to 12, and the iterations of the algorithm is set to 50 [23].

Acknowledgments. Luca Melis and Emiliano De Cristofaro were partially supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and a grant by Nokia Bell Labs, Gergely Acs by the Premium Post Doctorate Research Grant of the Hungarian Academy of Sciences (MTA) and the Higher Education Excellence Program of the Ministry of Human Capacities in the frame of Artificial Intelligence research area of Budapest University of Technology and Economics (BME FIKP-MI/FM). Claude Castelluccia was supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

[2] J. M. Abowd and L. Vilhuber. How protective are synthetic data? In *PSD*, 2008.

[3] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, 2005.

[4] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, 2014.

[5] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, and C. S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv*, 2017.

[6] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *NIPS*, 2013.

[7] V. Bindschaedler, R. Shokri, and C. A. Gunter. Plausible deniability for privacy-preserving data synthesis. *Proceedings of the VLDB Endowment*, 2017.

[8] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS*, 2005.

[9] C. M. Bowen and F. Liu. Differentially private data synthesis methods. *arXiv preprint 1602.01063*, 2016.

[10] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[11] D. E. Carlson, V. Cevher, and L. Carin. Stochastic spectral descent for restricted boltzmann machines. In *AISTATS*, 2015.

[12] A.-S. Charest. How can we analyze differentially-private synthetic datasets? *Journal of Privacy and Confidentiality*, 2011.

[13] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 2011.

[14] R. Chen, N. Mohammed, B. C. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 2011.

[15] R. Chitta, R. Jin, and A. K. Jain. Efficient kernel clustering using random fourier features. In *ICDM*, 2012.

[16] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD*, 2004.

[17] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, 2017.

[18] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4), 2014.

[19] C. Dwork, G. Rothblum, and S. Vadhan. Boosting and differential privacy. In *FOCS*, 2010.

[20] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, 2015.

[21] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT Press, 2016.

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

[23] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.

[24] C.-C. Hsu and C.-W. Lin. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 20(2), 2017.

[25] P. Huang, Y. Huang, W. Wang, and L. Wang. Deep embedding network for clustering. In *IEEE International Conference on Pattern Recognition*, 2014.

[26] G. Jagannathan and R. N. Wright. Privacy-preserving imputation of missing data. *Data & Knowledge Engineering*, 2008.

[27] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint 1412.6980*, 2014.

[28] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[30] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint 1609.04802*, 2016.

[31] F. Li, H. Qiao, B. Zhang, and X. Xi. Discriminatively boosted image clustering with fully convolutional auto-encoders. *arXiv preprint 1703.07980*, 2017.

[32] H. Li, L. Xiong, and X. Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In *EDBT*, 2014.

[33] F. Liu. Model-based differential private data synthesis. *arXiv preprint 1606.08052*, 2016.

[34] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[35] D. McClure and J. P. Reiter. Differential privacy and statistical disclosure risk measures: An investigation with binary synthetic data. *Transactions on Data Privacy*, 2012.

[36] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[37] J. Pennington, F. X. Yu, and S. Kumar. Spherical random features for polynomial kernels. In *NIPS*, 2015.

[38] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[39] J. P. Reiter and R. Mitra. Estimating risks of identification disclosure in partially synthetic data. *Journal of Privacy and Confidentiality*, 2009.

[40] J. P. Reiter, Q. Wang, and B. Zhang. Bayesian estimation of disclosure risks for multiply imputed, synthetic data. *Journal of Privacy and Confidentiality*, 2014.

[41] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.

[42] B. Schölkopf, A. J. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5), 1998.

[43] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *ACM CCS*, 2015.

[44] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. Differentially private k-means clustering. In *ACM CODASPY*, 2016.

[45] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 2002.

[46] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint 1703.00395*, 2017.

[47] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv preprint 1511.01844*, 2015.

[48] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML*, 2008.

[49] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.

[50] S. Vempati, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Generalized RBF feature maps for efficient detection. In *British Machine Vision Conference*, 2010.

[51] L. Wasserman and S. Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 2010.

[52] X. Wu, A. Kumar, K. Chaudhuri, S. Jha, and J. F. Naughton. Differentially private stochastic gradient descent for in-RDBMS analytics. *arXiv preprint 1606.04722*, 2016.

[53] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: differential privacy with reduced relative errors. In *ACM SIGMOD*, 2011.

[54] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 2016.

[55] J. Xie, R. B. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

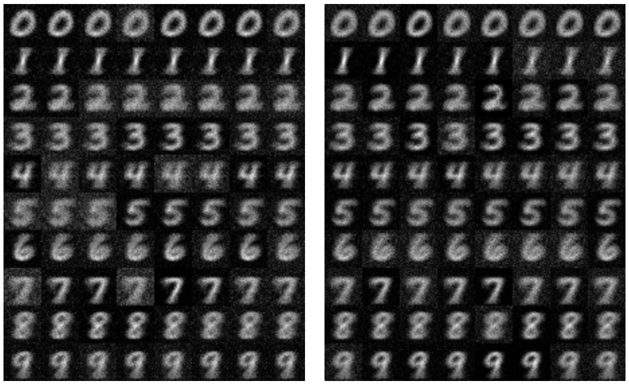
[56] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint 1610.04794*, 2016.

[57] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[58] R. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint 1607.07539*, 2016.

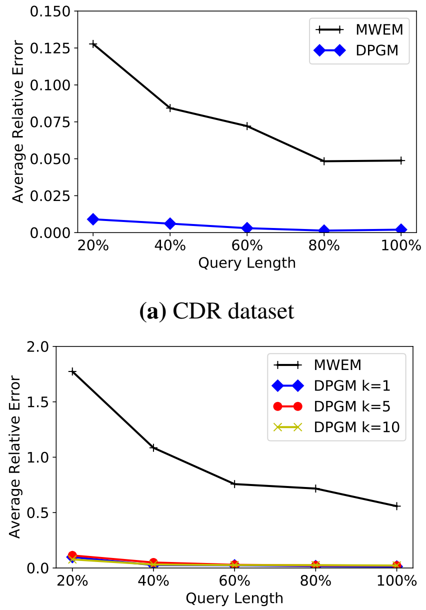
[59] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *ICMD*, 2014.

[60] Y. Zheng, H. Tan, B. Tang, H. Zhou, et al. Variational deep embedding: A generative approach to clustering. *arXiv preprint 1611.05148*, 2016.



(a) $\varepsilon = 1.74$ **(b)** $\varepsilon = 2.0$

Figure 7: Samples generated from a double layer VAE after 20 epochs. Each row contains 8 samples generated from a cluster.



(a) CDR dataset

(b) Transit dataset

Figure 8: Average relative error with $\varepsilon = 1.0$ for the CDR and transit datasets.

A Multi-layer Variational Autoencoder

We now report additional results for a VAE with a double layer encoder and decoder. In Fig. 7, we show the synthetic samples obtained at epoch 20 from a VAE with $k = 10$ clusters on MNIST.

Then, Fig. 8a reports the average relative error for the CDR dataset, while Fig. 8b shows the average relative error for the transit dataset with different number of clusters k.

Overall, we can observe that increasing the number of layers, and thus the capacity of the VAE, does not lead to better performances.