Federated Gaussian Mixture Models

SOPHIA ZHANG PETTERSSON, Cloud and Embedded Platform, Traton AB, Södertälje, Sweden
KUO-YUN LIANG, Cloud and Embedded Platform, Traton AB, Södertälje, Sweden
JUAN CARLOS ANDRESEN, Cloud and Embedded Platform, Traton AB, Södertälje, Sweden

2025. Manuscript submitted to ACM

This paper introduces FedGenGMM, a novel one-shot federated learning approach for Gaussian Mixture Models (GMM) tailored for unsupervised learning scenarios. In federated learning (FL), where multiple decentralized clients collaboratively train models without sharing raw data, significant challenges include statistical heterogeneity, high communication costs, and privacy concerns. FedGenGMM addresses these issues by allowing local GMM models, trained independently on client devices, to be aggregated through a single communication round. This approach leverages the generative property of GMMs, enabling the creation of a synthetic dataset on the server side to train a global model efficiently. Evaluation across diverse datasets covering image, tabular, and time series data demonstrates that FedGenGMM consistently achieves performance comparable to non-federated and iterative federated methods, even under significant data heterogeneity. Additionally, FedGenGMM significantly reduces communication overhead, maintains robust performance in anomaly detection tasks, and offers flexibility in local model complexities, making it particularly suitable for edge computing environments.

1 Introduction

Federated learning (FL) is a concept in machine learning (ML) that allows multiple devices or clients to collaboratively train a shared model, without having to share their data with each other or with a central server [31]. Clients keep their data locally and only communicate model parameters or updates. FL can thus be useful in scenarios where it is impractical or even impossible to transfer large amounts of data to a central location, for instance, due to communication costs or for reasons of privacy and security. It is common to make a distinction between cross-silo FL (that is, cooperation between relatively few clients, such as hospitals with large computational capabilities) and cross-device FL, where the number of clients (such as mobile devices or connected vehicles) is typically large and each client has limited resources for communication and computations, with potentially unreliable connectivity [17]. FL scenarios can also be characterized by how the data is partitioned among the clients. In vertical partitioning, clients share samples but have different sets of features. In horizontal partitioning, the set of features is the same between all clients, but the samples are different [17]. In this paper, we are mainly concerned with the cross-device, horizontal scenario, where the clients are assumed to be edge devices of some kind.

FL has great potential, but the successful realization of the concept also faces specific challenges. An important and much-studied issue in this context is how to handle the statistical heterogeneity of the data distribution between clients [35]. Especially in the context of cross-device FL, clients typically come in many different flavors and are operated in different environments by different users. Other FL challenges come in the form of communication delays, high communication costs, and security issues such as information leakage and adversarial attacks [36, 42].

Most FL algorithms are iterative, comprising several rounds of communication for the exchange of model parameters or updates. In contrast, the one-shot FL approach [13] has been proposed as a way of alleviating both communication and security issues by limiting communication between parties to a single round.

Authors' Contact Information: Sophia Zhang Pettersson, Cloud and Embedded Platform, Traton AB, Södertälje, Södertälje, Sweden, sophia.zhang.pettersson@scania.com; Kuo-Yun Liang, Cloud and Embedded Platform, Traton AB, Södertälje, Södertälje, Sweden, kuo-yun.liang@scania.com; Juan Carlos Andresen, Cloud and Embedded Platform, Traton AB, Södertälje, Södertälje, Sweden, juan-carlos.andresen@scania.com.

Most existing studies on FL have focused on supervised learning, whereas unsupervised learning and other categories of ML have received less attention in this context [4, 9, 17]. Furthermore, the typical FL setup in the literature employs some form of neural network (NN) model rather than statistical ML models [28, 34], and the vast majority of FL use cases revolve around image classification, using datasets such as MNIST, FEMNIST, CIFAR, et al. [4]. Thus, there are a substantial number of studies (e.g., [24, 26, 30, 40, 44]) on how to extend and improve the FedAvg algorithm, originally presented in [31], in this context. However, FL approaches for statistical ML models appear to be less explored, especially in the areas of unsupervised learning and tabular data. An example of a type of ML model with proven strengths in these two areas is the Gaussian Mixture Model (GMM) [12, 18].

In this paper, we apply FedGenGMM, a one-shot approach, to the problem of federated GMM learning. Specifically, models are trained locally until convergence, and then the model parameters are sent to the server where we utilize the generative property of the GMM to perform model aggregation. The method is straightforward to implement, does not require any sharing of raw data, and uses less communication and synchronization resources than iterative FL methods during the training phase. We evaluate FedGenGMM by analyzing the ability of the shared model to capture the patterns in the local data and by using the model to perform anomaly detection on several datasets, both from the public domain and from the industry, under different conditions of heterogeneity. The topic of anomaly detection in an unsupervised federated setting is of interest in itself and has not been thoroughly studied so far. Furthermore, we evaluate the solutions using both image, tabular, and time-series data.

2 Related work

2.1 Federated GMM

The GMM belongs to the family of finite-mixture models. The standard method for training such models is known as the Expectation Maximization (EM) algorithm, first introduced in [5] and since then extended and improved in a large body of research, e.g. [12, 16, 18, 25]. In its original form, the EM algorithm assumes that all data is available at the location where the model training takes place. However, although relatively less explored than FL for NN models, federated EM algorithms for GMM training have been proposed in several previous studies, either as a central concept [34] or as a tool for improving FL of NN models under heterogeneous conditions [44]. In [34], the authors developed a distributed version of the EM algorithm (DEM) for the training of GMMs, with the aim of detecting anomalies in an industry setting. In [10], a distributed algorithm for variational Bayesian GMM was deployed to facilitate the generation of synthetic data based on generative adversarial networks (GANs) for tabular datasets. The authors of [44] utilized GMMs to enhance the FL of NN models for image classification under heterogeneous conditions. Here, the GMM training was an integral part of the NN model training, again using a DEM algorithm. In particular, they also pointed out the possibility of using the distributed GMM training concept independently.

A shared trait of the aforementioned solutions is thus that they are based on distributed versions of the EM algorithm, where each iteration involves sending parameter updates between clients and server. Furthermore, the GMM structure is assumed to be the same across all clients and the server, e.g., using the same number of GMM components and the same type of covariance matrix.

2.2 One-shot FL

In recent years, one-shot FL has emerged as an alternative to the iterative FL approach to supervised learning. In contrast to iterative methods, one-shot FL methods use only one single communication round to exchange information

between clients and the server. This has potential benefits in the form of reduced communication cost and a lower risk of security breaches [46]. However, the approach also presents challenges in the presence of heterogeneity of clients [14].

In [13], the authors used both model ensemble selection techniques and knowledge distillation to aggregate locally trained support vector machine (SVM) models (instead of simply averaging the parameters of the local models). However, heterogeneous data distributions were not considered. For the knowledge distillation part, a dataset known to all clients was used. For privacy reasons, this approach may be unfeasible in many cases [48]. The method presented in [23] also employs public datasets to select ensemble models at the central level. An ensemble approach which does not require any public dataset is described in [41], where the client models are clustered at the server. Cluster representatives are selected based on criteria such as local validation accuracy and size of the local training dataset.

In the context of NN models and image classification, several studies have explored the technique of performing one-shot FL by sending information about local data distributions to the server in order to generate training data for a global model. An approach is illustrated in [47], where distilled versions of local data are sent to the server to be used for training, achieving large savings in communication costs. The authors of [19] instead let each client choose among a fixed set of typical distributions according to which one best fits their local dataset and then send the estimated distribution parameters (together with the local models) to generate training data on the server. A somewhat related design can be found in [2], a recent work in which GMMs are trained locally on class-conditional features generated by local models. The GMM parameters are then sent to the server for training of the global model. In [46], only the trained models are uploaded to the server, where the global model is created in two steps. First, the uploaded client models are employed to train a generator with the aim of creating a synthetic dataset such that it has similar properties to the local datasets. In the next step, the synthetically generated images and the local models are used to train the final global model (using knowledge distillation). The approach in [46] allows for heterogeneous local model structures. A recent study that explicitly addresses the issue of strongly heterogeneous client distributions in one-shot FL for NN-based image classification is presented in [14], where conditional variational autoencoders are trained locally. The resulting weights of the decoding network are sent to the server, where they are used to create a global decoder. The global decoder is then used to create a synthetic dataset that is used to train the final global model.

The work of [7] is one of the few that has considered one-shot FL in the context of unsupervised learning. Here, the authors develop a method for federated clustering in which the clients share their locally computed cluster centers with the server. The server then applies a clustering algorithm to the local centers, which produces the global centers. The method assumes that the centers are well separated in some sense. Under this condition, higher levels of heterogeneity is found to actually improve the performance of the clustering algorithm.

2.3 Unsupervised FL and anomaly detection

The topic of unsupervised FL methods is less well studied than FL in a classification setting [9, 33]. However, a study closely related to our work is [34] in which the authors present an unsupervised FL approach to the problem of anomaly detection, using a federated version of the EM algorithm for GMM training. Their overall approach is thus similar to ours but uses an iterative FL method, as opposed to our one-shot approach.

In the context of NN-based models, the work in [40] describes an unsupervised FL method for intrusion detection based on autoencoders. Here, heterogeneity in client distributions is handled by a federated feature scaling scheme, and the training process is modified to account for differences in local data set sizes. In [33], the authors devised a two-step method for unsupervised FL under heterogeneous conditions. First, locally trained one-class SVM models were used to

cluster clients that have the same inlier distributions. Secondly, FedAvg was used for training a federated autoencoder, only averaging parameters of clients in the same cluster. One downside of their method is that it requires the exchange of SVM model parameters between every pair of clients. A recent example of unsupervised FL for anomaly detection under heterogeneous conditions can be found in [9], where each client makes a rough estimate of its local distribution of features in the form of a Gaussian mixture. The mixture parameters are then sent to the server, where they are aggregated and used to improve the performance of the anomaly detection in each client during the federated learning process. Finally, the global NN model is aggregated using a modified version of FedAvg. In the experimental evaluation in [9], the Gaussian mixtures were limited to having only one component.

In addition, some works (e.g. [39, 45]) have also considered unsupervised federated representation learning, where the purpose of the learning process is to extract data features that can be used for downstream tasks.

In this work, we consider the problem of using unsupervised FL to train GMMs with varying numbers of components. Training is performed using a communication-efficient one-shot approach, in a setting with heterogeneous client data distributions.

3 Problem formulation

Suppose we have a set of N clients C, where each client $c \in C$ has access to a local dataset D_c. Each dataset D_c consists of d-dimensional data points $\{\mathbf{x}_i \in \mathcal{R}^d, i = 1, 2, ..., |D_c|\}$, drawn from the distribution $p_c(\mathbf{x})$. Let $D = \cup_{c \in C} D_c$. Our goal is to obtain a single centrally located GMM G that models the distribution of D (denoted as p_D), without sending any data points from the clients. G may then be distributed to the clients, where the representations learned by the global model can be used for downstream tasks such as anomaly detection and classification. One example of a situation where this might be useful (instead of just relying on locally trained models) is if during training each client only operates in a limited number of environments, but during inference most of the clients can expect to be exposed to a significantly wider range of environments. Furthermore, new clients who join the scheme after completion of the model training could likely benefit from having access to an already trained global model.

The distributions of the local datasets D_c can be heterogeneous among clients, and the heterogeneity is assumed to be in the form of feature distribution skew [17]. As is common in the FL literature (cf. [21, 30, 44]), we further assume that each local distribution p_c is a mixture of M underlying, unknown distributions $p^{(m)}$, that is,

$$p_c(\mathbf{x}) = \sum_{m=1}^{M} \pi_{c,m} \cdot p^{(m)}(\mathbf{x}) \,, \tag{1}$$

where $\pi_{c,m}$ is the mixing weight of the underlying distribution $p^{(m)}$ for client c.

In the setting of density estimation, our aim is to have the G model p_D as closely as possible, given some specific configuration of G (such as the number of components and the type of covariance matrix). In this context, we will use the average log-likelihood to measure how a trained GMM fits the target distribution, that is, for a dataset $D_T = \{\mathbf{x}_i, i = 1, 2, ..., T\}$, drawn from p_D, the problem can be described as maximizing the fitness score γ_G:

$$\gamma_G(D_T) = \frac{1}{T} \sum_{i=1}^{T} f_G(\mathbf{x}_i) \,, \tag{2}$$

where $f_G(\cdot)$ is the log-likelihood for one data point, as given by the global GMM G. That is, a higher score means that the model fits the data more closely (cf. [43]). In line with other works on one-shot FL (cf. [13, 14, 41]), during evaluation of the global model, the input samples will be drawn from the global distribution p_D.

In the setting of anomaly detection, we suppose that the input data can come from two distinct distributions with p_D representing the "in" distribution (corresponding to non-anomalous data) and p_{OOD} the "out" distribution (corresponding to anomalous data). We assume that clients only encounter (heterogeneous) data drawn from p_D during training and creation of G. The global model is then applied to data drawn from both p_D and p_{OOD}, our aim being to achieve a high degree of separation between the fitness scores for non-anomalous (inlier) data and the scores for anomalous (out-of-distribution, OOD) data, respectively.

4 FedGenGMM

4.1 Method description

We utilize the standard EM algorithm [5, 32, 44] to train one local GMM in each client. After training, each local model G_c will have the density

$$p_{G_c}(\mathbf{x}|\theta_c) = \sum_{k=1}^{K_c} r_{ck} \mathcal{N}(\mathbf{x}|\mu_{ck}, \Sigma_{ck}) \, , \tag{3}$$

where $\theta_c = \{r_{ck}, \mu_{ck}, \Sigma_{ck}\}, k = 1, 2, ..., K_c$, are the GMM parameters of G_c with K_c being the number of GMM components. Note that the number of components can be different for different clients. r, μ, and Σ represent the weight, the center, and the covariance matrix of each component in G_c, respectively.

When a client completes its local training (that is, when convergence has been reached), it sends the parameters of its local GMM G_c to the server. After receiving the model parameters from the clients, the server uses the parameters to generate a synthetic dataset S. To promote similarity between the densities of S and the global dataset D, the incoming mixture weights r_{ck} for each client model are first modified according to the relative size of the corresponding local dataset D_c:

$$\forall c : r_{ck} \leftarrow r_{ck} \frac{|D_c|}{|D|}, k = 1, 2, ..., K_c \, . \tag{4}$$

Note that this requires the server to be aware of the local dataset sizes and that it would of course be possible to use other types of client weighting schemes here. Then, one single GMM consisting of all the re-weighted components is created, and its weights are normalized. Finally, S is created by drawing from the distribution given by the single GMM. The number of datapoints drawn is proportional to the number of incoming components:

$$|S| = H \sum_{c=1}^{C} K_c \, , \tag{5}$$

where the constant H is a FedGenGMM hyperparameter, which thus together with the complexity of the local models determines the size of the synthetic dataset.

Finally, the EM algorithm is applied to train the global GMM G on S. In this step, the number of components in G and the type of covariance matrix can be fixed beforehand or determined during training, for example, by minimizing the Bayesian Information Criterion (BIC) score or using other strategies [12, 25]. In the algorithm 4.1 described below, this is exemplified by minimizing the BIC criterion both in the local training phase and in the aggregation phase. However, the main takeaway is that the hyperparameters of the local models and the global model can all be set independently of each other. Another benefit of FedGenGMM is that it would be fairly straightforward to replace the standard EM

algorithm with another method to train local GMMs. A recent example of such a method is presented in [18], where the authors develop a method that shows resilience under conditions of noisy data and overlapping GMM clusters.

4.2 FedGenGMM algorithm

One example of a realization of FedGenGMM is shown in algorithm 4.1. Here, during the training of each client c, the BIC score is used to select the number of local GMM components (K_c) from a range of values (denoted by $[K_{\min}, K_{\max}]$) that is given as input to the algorithm. K_c could be determined in other ways, without affecting the overall algorithm. The same observations apply to the training of the aggregated global model in the last step.

Algorithm 4.1 FedGenGMM algorithm

Input: Set of clients (C) with local datasets $\{D_c\}$, Size of generated data (H)
Input: Range of number of GMM components ($K_{\min}, K_{\max}$), Desired covariance type
Output: Parameters of trained global model G
procedure TrainGMM($K_{\text{candidates}}$, dataset X)
    Let $\text{bic}_{\min} = \infty$;
    for k in $[K_{\min}, K_{\max}]$ **do** ▷ Estimate K using BIC
        Let Q be a GMM with desired covariance type and K = k;
        Initialize the parameters of Q using X;
        Use EM to train Q on X;
        if BIC(Q, X) < $\text{bic}_{\min}$ **then**
            Let $\text{bic}_{\min}$ = BIC(Q, X);
            Let $R = Q$;
        end if
    end for
    Return R;
end procedure
procedure FedGenGMM
    for c = 1 to $|C|$ **do** ▷ Local training part for all clients
        Let G_c = TrainGMM($K_{\text{candidates}}$, D_c);
        Send parameters of G_c and size of local dataset $|D_c|$ to the server;
    end for
    Let G_{tmp} = [an empty GMM]; ▷ Server side aggregation part
    Let $s = \sum_{i=1}^{|C|} |D_c|$;
    for c = 1 to $|C|$ **do**
        for k = 1 to K_c **do**
            Add $(r_{ck} \frac{|D_c|}{s}, \mu_{ck}, \Sigma_{ck})$ to G_{tmp};
        end for
    end for
    Normalize the component weights of G_{tmp};
    Create S by drawing $H \cdot \sum_{c=1}^{C} K_c$ values from $p_{G_{\text{tmp}}}$;
    Let G = TrainGMM($K_{\text{candidates}}$, S);
    Return G;
end procedure

4.3 Computational complexity and communication cost

The following analysis does not include the operations that may be needed to estimate K, neither in the clients nor in the server. The computational complexity in the clients is equal to the complexity of the standard EM-algorithm for training GMMs, that is $O(|D_c| \cdot K_c \cdot \delta)$ per iteration [32]. Here δ is a term that depends on d (the dimensionality of the dataset) and on the type of covariance matrix Σ. For full covariance, $\delta = d^2$, and for diagonal covariance $\delta = d$. (We note that the M-step typically involves inversion of the covariance matrix for each component of the GMM. This operation has complexity $O(K_c \cdot d^3)$ if full covariance matrices are used. However, in the settings we consider, $d << |D_c|$, therefore, this term can be ignored.)

The merging process in the server consists of two main steps. First, synthetic data is generated for each incoming GMM G_c. The generation of one such data point has complexity $O(K_c + \delta)$, giving an overall complexity for the first step of $O(|S| \cdot (K_c + \delta))$, where $|S|$ is the size of the generated dataset. Second, the standard EM algorithm is used to generate the global model G, which requires $O(|S| \cdot K \cdot \delta)$ operations per EM iteration (where K is the number of components in G). Thus, since $|S|$ increases linearly with the number of clients C (Eq. 5), the server-side complexity will be linear with respect to C.

As for communication costs, the one-shot FedGenGMM process requires only one single round of communication to create the global model.

4.4 Privacy

The FedGenGMM process requires that clients share their local model parameters with the central server, causing statistical information about the local datasets to be exposed. In scenarios where the data is non-sensitive or the server can be trusted, this is less likely to be an issue. In other scenarios, techniques such as differential privacy (DP) [11] or homomorphic encryption [29] could be applied to reduce the risk of privacy breach. As pointed out in e.g. [46], the fact that only a single communication round is needed reduces the risk of privacy leakage. Furthermore, the single-round setup would have an advantage in the context of DP. The entire privacy budget could be allocated to this single round of communication. This might mitigate the effects of depletion of privacy budget due to the number of communication rounds, a problem reported by e.g. the authors of [15].

In general, approaches to improving privacy in the context of EM algorithms include encryption [20] , DP [8], secure summation [27], and subspace perturbations [22], which all add some complexity to the overall solution. Although several of these approaches could be applied to FedGenGMM to enhance privacy protection, we leave this as a future work.

5 Experimental evaluation

5.1 Datasets

We conducted the evaluation using one proprietary dataset ("VEHICLE"), in the form of vehicle time series data collected during the operation of Scania vehicles, and four public datasets from different domains, including image data (MNIST [6]), tabular data (Covertype [3]), and time series data (WADI [1], RWHAR [38], and the Server Machine Dataset (SMD) [37]). The basic characteristics of each dataset are outlined in table 1. Since high-dimensionality data can lead to high computational complexity in the GMM training process [44], we used principal component analysis (PCA) to reduce the number of features for MNIST (from 784 to 24) and RWHAR (from 63 to 16). In addition, to facilitate the creation of OOD data, we excluded all binary features from Covertype and from WADI, leaving 10 and 84 features for those

Table 1. Datasets employed for evaluation

Dataset	Domain	Size	Features	Classes	Partitioning
MNIST	Handwritten digits	60 000	24 (784)	10	Dir(α)
Covertype	Tabular terrain data	580 000	10 (54)	7	Dir(α)
RWHAR	Human activity body sensors	338 000	16 (63)	13	Dir(α)
WADI	Water facility operations	950 000	84 (123)	10	Quantity(α)
VEHICLE	Air pressure system operations	12 000	11 (11)	3	Quantity(α)
SMD	Server machine operations	1 400 000	38 (38)	28	Dir(α)

Table 2. Datasets and OOD settings

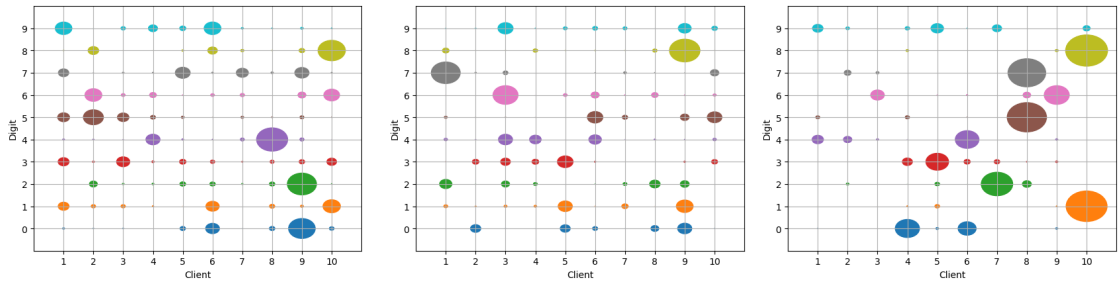
Dataset	Test set size	Inlier data	Anomalous data	Anomaly ratio
MNIST	12 000	Original images	Manipulated images	10%
Covertype	116 000	Original	Added Gaussian noise	10%
RWHAR	67 000	Walking	Running	10%
WADI	170 000	Normal ops	Cyber attack mode	6%
VEHICLE	3 000	Normal ops	Induced air leakage	50%
SMD	700 000	Normal ops	Observed malfunctions	4%

datasets, respectively. For each dataset, the original number of features is given within brackets in the column "Features" in table 1. All features were normalized to the range of $[0, 1]$.

5.2 Data partitioning schemes

To simulate heterogeneity (in the form of feature distribution skew) of the client distributions, we partitioned the data using two different schemes. In both schemes, each underlying distribution corresponds to one "class" in the dataset, e.g., digit "2" in the case of MNIST. The first scheme relies on drawing the mixture weights $\pi_{c,m}$ in Eq. 1 from a symmetric Dirichlet distribution with parameter α, similar to other works such as [30] and [46]. Here, smaller values of α mean that the client distributions will be more heterogeneous. The effect of different values of α is illustrated in Figure 1 for the case of MNIST data distributed over 10 clients. In the following, we refer to this scheme as "Dir(α)". In the second scheme, each client mixture is created by assigning data from a fixed number (also denoted by α) of randomly chosen underlying distributions $p^{(m)}$ to each client, similar to the heterogeneity scheme in [26] and one of several schemes employed in [21] (termed "Quantity-based label imbalance"). We denote this scheme by "Quantity(α)".

As stated above, for MNIST, each class corresponds to data from images representing the same digit. For Covertype, we use the different terrain types, and for RWHAR each class corresponds to sensor data from a specific person. The SMD consists of data from 28 different servers, thus allowing for 28 classes in our experiments, and for the VEHICLE dataset, we create partitions based on which operational environment (i.e., city, high-way, or test track) the data was collected in. Finally, for the WADI experiments, the classes were created artificially. The data corresponding to the underlying distribution $p^{(m)}$ was created by adding values drawn from a multivariate Gaussian with center $C_m = \mathbf{1}(m-1)\beta$ and diagonal covariance 0.01 to the original data points, with β being a parameter controlling the difference between the generated data for different classes. The same type of perturbation was applied to both the non-anomalous and anomalous parts of the dataset.



(a) Dir(α=0.5) (b) Dir(α=0.3) (c) Dir(α=0.1)

Fig. 1. Illustration of heterogeneity levels for different values of (α), using Dir(α) to create partitions of the MNIST dataset for 10 clients. Each row of dots corresponds to one digit, i.e., one underlying distribution, and each column corresponds to one client. The size of the dot located at (x, y) is proportional to the number of datapoints from digit y seen by client x during training. With $\alpha = 0.5$, most of the clients will use data from at least half of the digits. With $\alpha = 0.1$, most of the data from each digit (i.e., digit 1 and 8) is assigned to only one or two clients. Note that the local dataset sizes also become more heterogeneous with decreasing α.

5.3 Obtaining anomalous data

Four of the datasets (RWHAR, WADI, VEHICLE, and SMD) were collected under conditions that allow a natural separation into inlier and anomalous (OOD) data subsets, as indicated in Table 2. For these cases, we used a subset of the inlier data as the training set to be partitioned and distributed to the clients. After training, the global model was applied to the rest of the inlier data together with a randomly chosen subset of the OOD data.

For the MNIST and Covertype datasets, we created artificial OOD data. Initially, the dataset was split into three parts, with the first part forming the training set, the second part the inlier part of the test set, and the third part making up the OOD part of the test set. The images in MNIST were rotated 90 degrees counter-clockwise, flipped horizontally, and scaled with a factor of 1.2. The Covertype terrain data was perturbed by adding Gaussian noise with zero mean and variance 0.005. For both datasets, the same OOD perturbation was applied across all classes.

5.4 Baselines

We compared the performance of FedGenGMM against local models, different versions of DEM for GMMs, and a non-federated model.

In the local model approach, the individual models trained on each local dataset D_c were applied to the evaluation dataset and the resulting scores were averaged.

To obtain a stronger baseline, we also used a distributed version of the EM algorithm for GMM, as outlined in [44], to create the global model. Here, the number of components are assumed to be the same for the global model and all local models, and the training process starts by having the server choose initial GMM parameters and distribute them to the clients. Since the server does not have access to local data, it is not obvious how the initial GMM parameters should be chosen. We evaluated three different schemes for the initialization of the GMM component centers, (1) maximally separated centers given knowledge of the feature range [8], (2) performing an initial GMM training round using a small subset of training data, and (3) using a federated version of k-means (from [7]) for the estimation of the global component centers. Henceforth, we refer to these three variants as DEM init 1, DEM init 2, and DEM init 3, respectively. For DEM init 1, we used the fact that the features were normalized. For DEM init 2, the subset consisted of 100 data points drawn from the entire training dataset. We note that the second scheme involves sending data points to the

Table 3. Datasets and settings

Dataset	K	Clients
MNIST	30	20
Covertype	15	20
RWHAR	15	20
WADI	10	20
VEHICLE	15	12
SMD	10	20

server. The stopping criterion for the DEM iterations was based on the change in the average likelihood over the client models when applied to the local datasets.

Finally, as a benchmark representing the "ideal" solution, we used the EM algorithm to train the global model using the entire training dataset in a non-federated setup.

5.5 Settings

In order for the model sizes to be reasonable with respect to the size of the local datasets, and given the setting of edge computing, which implies limited computational resources in the clients, we chose to use diagonal covariance matrices for the GMMs (refer to Section 4.3 for complexity analysis).

For each dataset, we determined the number of GMM components in the global model (K) by training non-federated GMMs with varying K. We analyzed the average log-likelihood resulting from applying the trained model to a validation part of the dataset, and a value representing a good trade-off between model size and performance was chosen. The resulting values are given in Table 3. Furthermore, for most of the experiments, we let $K_c = K$ for all FedGenGMM clients to facilitate comparisons between different federated algorithms. We note that this way of determining K_c would not be feasible in a federated setting, where no single party would have access to a representative part of the dataset. A more realistic approach would be to let each client c estimate K_c using its local data, something that could easily be accommodated in FedGenGMM since the method allows heterogeneous local models.

Unless otherwise stated, the number of participating clients was set to 20 for all datasets except VEHICLE, where we used 12 clients due to the limited dataset size.

The initialization of the local GMM components was done using k-means on local data. For the generation of synthetic data in the server, we set the value of H to 100 in Eq. 5. The convergence limit (in terms of the difference in the likelihood of the training data between two consecutive EM iterations) for the GMM training was set at 10^{-3} throughout the experiments, both for FedGenGMM, the DEM algorithms and the non-federated benchmark.

Each run (consisting of dataset generation, federated training, and evaluation) was repeated five times to obtain the mean and variance of the performance scores for the different methods and scenarios. In each run, a new split of the dataset into train and test subsets was created, as well as a new partition of the data between clients and new OOD data for the anomaly detection experiments.

5.6 Results - fitting a model to the global distribution

The purpose of the first set of experiments was to evaluate the ability of FedGenGMM to produce a global model that captures the patterns of the global distribution, under varying heterogeneous conditions, and for each of the six datasets included in the study. As performance score, we used the average log-likelihood produced by applying the resulting

model to the entire training dataset. For each dataset, a fixed number of GMM components (K) was used, as indicated in table 3. The results are shown in Figure 2.

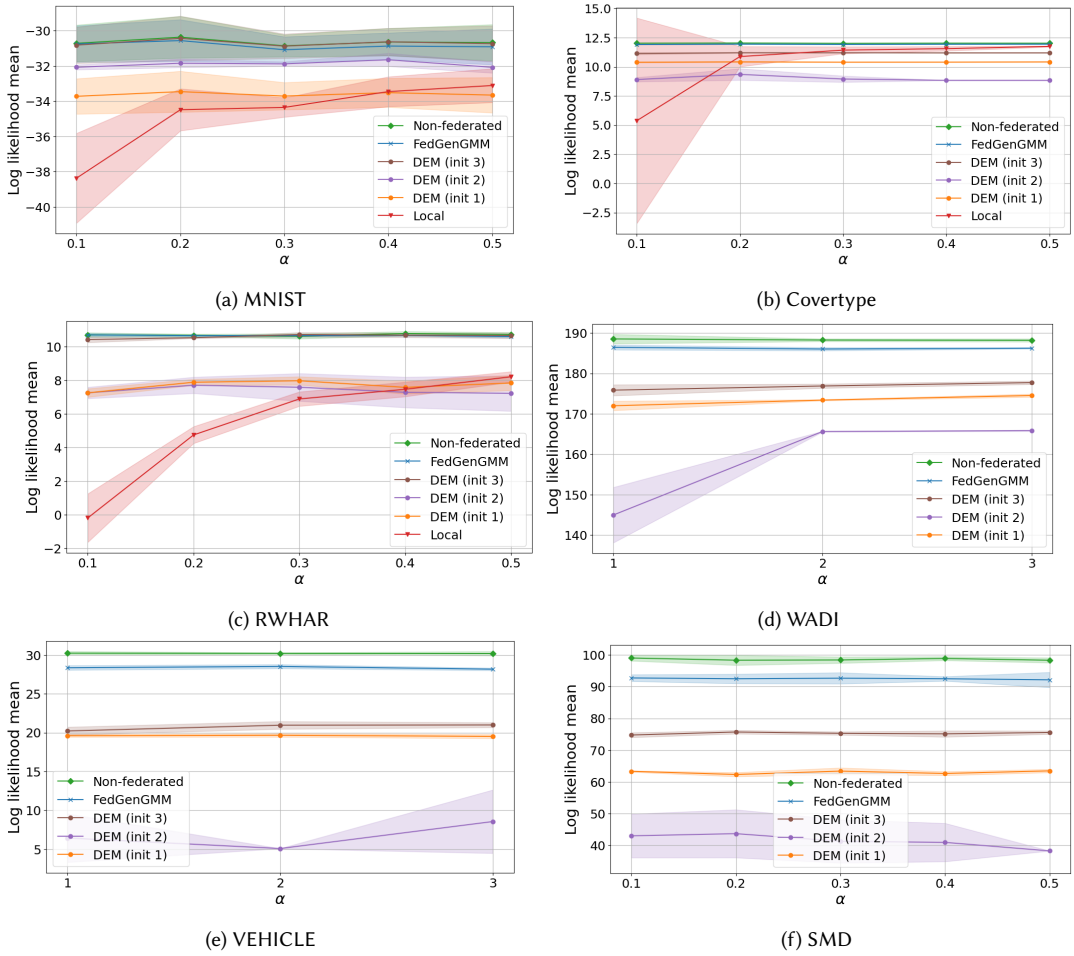


Fig. 2. Resulting global model log likelihood mean for each dataset and method, using fixed K with varying heterogeneity level (α) of client data distributions. For WADI and VEHICLE, the heterogeneity scheme is Quantity(α), for others it is Dir(α). The markers indicate the mean over five runs, and the height of the shaded areas correspond to +/- the standard deviation. Results for local models are excluded for plots (d) - (f) due to the low values of the log likelihood mean.

As can be seen from the plots, all the methods except the local approach have stable performance under the different degrees of heterogeneity included in the experiments. Among the three different versions of DEM, init 3 (which relies on distributed k-means for the initialization of the GMM component centers) performs best for all datasets. The ability of FedGenGMM to adapt to the global distribution is persistently on par with or better than the best of the DEM methods and is in fact close to the benchmark for all datasets. In terms of result variance, both FedGenGMM and DEM init 3 exhibit relatively low variance for all cases except MNIST, possibly due to the more complicated patterns present in that dataset, coupled with the relatively small size of the dataset. As expected, the local model approach struggles under

conditions of strong heterogeneity. The complete plots for (d) - (f), including the local model performance, can be found in Appendix A.

5.7 Comparison of FedGenGMM with DEM

5.7.1 Communication cost. For comparison of communication cost, the number of iterations required for the different DEM versions was recorded. As shown in Table 4, this number varies both with the dataset and with the type of parameter initialization. Overall, the number of communication rounds required for DEM is around one order of magnitude higher than that of the one-shot FedGenGMM process. The actual number of iterations required for DEM in a specific case would depend on several factors, such as the nature of the dataset, the initial parameters of the GMMs, and the tolerance limits set for convergence. However, in general (for centralized EM), the number of iterations can be said to be between 10 and 100 [12].

Table 4. Average number of communication rounds required for training the global model, recorded over five runs for each combination of dataset, algorithm and heterogeneity level (heterogeneity levels as in Figure 2).

Dataset	FedGenGMM	DEM init 1	DEM init 2	DEM init 3
MNIST	1	33	41	38
Covertype	1	21	7	40
RWHAR	1	20	27	15
WADI	1	25	4	29
VEHICLE	1	29	3	18
SMD	1	36	12	30

5.7.2 Computational complexity. The DEM algorithm performs mainly the same type of calculations as the centralized EM algorithm, with the E-step and M-step being executed locally in each client and the resulting parameters being aggregated in the server after each iteration. Compared to FedGenGMM, DEM thus adds the server aggregation part at the end of each iteration. On the other hand, FedGenGMM requires the generation of synthetic data and one full execution of the standard EM-algorithm on the server. Thus, while server-side complexity can be expected to be higher for FedGenGMM, computational complexity in the clients (which is most relevant in the context of edge computing) should be similar for the two approaches.

5.7.3 Privacy. In FedGenGMM, the local models are trained until convergence and the resulting parameters are sent to the server, something that may raise privacy concerns, as discussed in Section 4.4. The process is a bit different from the DEM algorithms, where after each local EM iteration, the parameters are sent to the server. In the server, they are aggregated over all participating clients. Then, the aggregated parameters are sent back to each client for the next iteration, and the process is repeated until convergence is reached. However, using DEM algorithms might still enable the server to reconstruct local statistics to a large extent, as shown in [22].

5.8 Results - anomaly detection

In the anomaly detection scenario, the model to be evaluated was used to produce point-wise log-likelihood values for a hold-out dataset. The hold-out dataset consisted of one part ID data and one part OOD data, with the ratios given in Table 2. The resulting AUC-PR was used as performance score. The main purpose of the experiments was to show

that for a given GMM configuration (that is, given K and the type of covariance matrix), the one-shot FedGenGMM method can achieve performance at least similar to DEM methods under heterogeneous conditions. However, we also investigated the possibility of using smaller client models during the FedGenGMM learning phase and then aggregating them into a larger global model. Furthermore, the impact of the number of clients was briefly studied.

Although several of the datasets consist of time series that exhibit both point-wise and sequential anomalies, in the present study all anomaly detection scores were produced as point-wise scores. A combination of point-wise scoring with windowing techniques, such as smoothing the input and using moving averages of anomaly scores, would likely increase the performance. However, this would also introduce more hyperparameters in the experiments. Given the main purpose of analyzing the relative performance of the methods, no more elaborate schemes were implemented.

Varying heterogeneity In this set of experiments, we evaluated the ability of the methods to detect anomalous data points, while varying the heterogeneity of the client data distributions. The results presented in Figure 3 indicate that the performance of the three DEM methods, as well as that of FedGenGMM, is relatively insensitive to the level of heterogeneity.

FedGenGMM performs at least on par with the DEM methods in all cases except for SMD, outperforming them for Covertype, where all three DEM variants struggle. The results for SMD stand out somewhat, with three of the methods actually achieving higher average scores than the benchmark, which uses nonfederated training. In most of the cases (especially for SMD), FedGenGMM exhibits a consistently lower variance than the DEM methods.

Varying number of clients The impact of an increasing number of clients was evaluated in a separate experiment, where the settings for heterogeneity and GMM configurations were kept constant for each dataset. The results are illustrated in Figure 4. Overall, all the methods show stable performance for the range of client set sizes included in the experiment.

Constrained client models Given the context of resource-constrained clients or devices, a possible use case for FedGenGMM would be the use of less complex local models during the learning process while aggregating the local models into a larger global one. The clients would still have to use the larger model for inference, but the resource requirements for GMM inference are typically much lower than for GMM training when using the EM-algorithm, since the training entails many iterations. One example is that since the computational complexity in the clients during training is linear in the number of GMM components, using fewer components locally would reduce the training complexity. In the experiment, the number of GMM components for the local models was varied from 2 to 20 while the global model consistently comprised 20 components (both the federated model and the non-federated benchmark). For comparison, we also show the performance of the DEM init 3 (k-means initialization) method, with both local and global models restricted in the number of components.

The results are illustrated in Figure 5. In cases (a), (b), (c), and (e), FedGenGMM with constrained client models aggregated into a large central model achieves similar performance as the non-federated benchmark with full number of components (20). For MNIST (5a) and Covertype (5b), using approximately half the number of components locally achieves the same performance as the benchmark. However, the results are inconclusive for WADI and SMD, where the performance is largely unrelated to K. In most cases, FedGenGMM outperforms the DEM method for any given level of computational complexity of client training (that is, for any given number of local GMM components). However, when comparing the performance of FedGenGMM and DEM in this experiment, it should be noted that the DEM method uses a smaller global model than FedGenGMM.

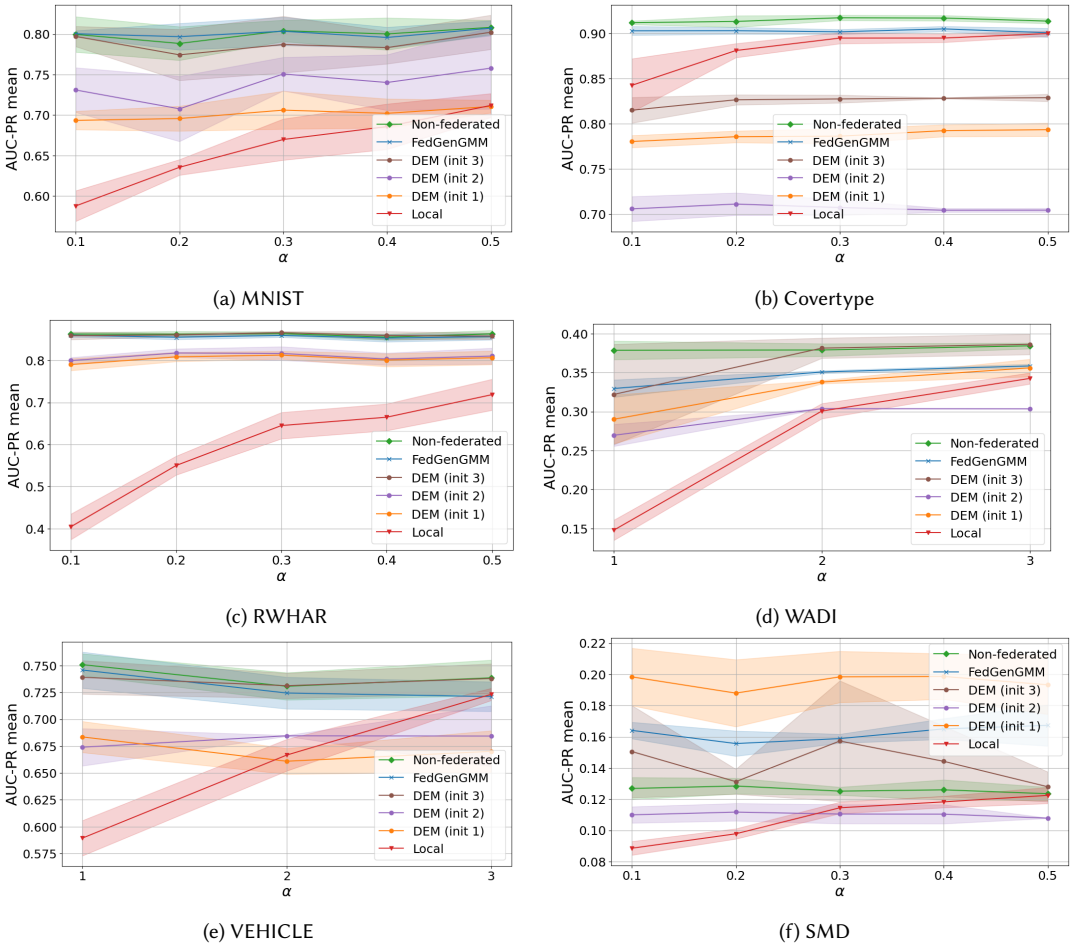


Fig. 3. Anomaly detection AUC-PR mean for each dataset and method, using fixed K with varying heterogeneity level (α) of client data distributions. For WADI and VEHICLE, the heterogeneity scheme is Quantity(α), for others it is Dir(α). The markers indicate the mean over five runs, and the height of the shaded areas correspond to +/- the standard deviation.

6 Conclusions

In this work, we present and evaluate FedGenGMM, a method for one-shot federated training of Gaussian mixture models. The method uses the standard EM algorithm for client-side training and employs the generative property of the GMM to aggregate local models into a shared global model. We evaluate the method under controlled heterogeneous conditions, using six datasets of varying nature. Compared to distributed versions of the EM algorithm, FedGenGMM is shown to incur significantly lower communication costs. The results further indicate that the ability of FedGenGMM to learn the global data distribution in a federated setting is on par with that of a non-federated approach and better than the distributed EM methods included in the study. We also apply FedGenGMM to the problem of unsupervised FL for anomaly detection, with results showing performance being close to that of a non-federated model, at least on par with the distributed EM methods, and stable with respect to heterogeneity level and the number of clients. The

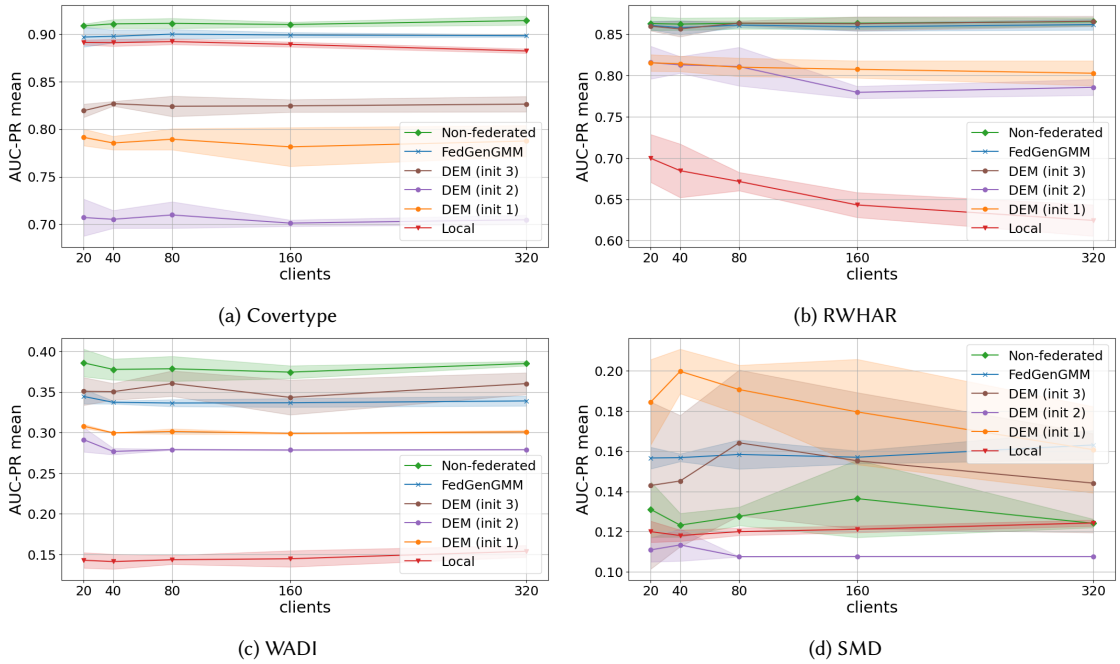


Fig. 4. Anomaly detection AUC-PR mean for each dataset and method, varying the number of clients from 20 to 320 while using fixed K and fixed heterogeneity levels. For WADI, the heterogeneity scheme is Quantity(α=1), for others it is Dir(α=0.2). The markers indicate the mean over five runs, and the height of the shaded areas correspond to +/- the standard deviation. The datasets MNIST and VEHICLE were not included here due to their relatively small sizes, providing too few datapoints per client for larger number of clients.

flexibility of FedGenGMM in terms of allowing for different client model structures is further shown to have potential in reducing the computational complexity without sacrificing performance, compared to the less flexible approach of distributed EM. More work is required to consider potential privacy attacks and the mitigation of such attacks, to assess the potential of integrating other versions of the EM algorithm for local and central training, and to investigate the feasibility of applying the FedGenGMM concept to the problem of continuous federated learning.

Acknowledgements

We thank Erik Pettersson for insights and useful discussions. This work was supported by Vinnova within the FFI program under contract 2020-02916.

References

[1] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks* (Pittsburgh, Pennsylvania) *(CySWATER '17)*. Association for Computing Machinery, New York, NY, USA, 25–28. https://doi.org/10.1145/3055366.3055375

[2] Mahdi Beitollahi, Alex Bie, Sobhan Hemati, Leo Maxime Brunswic, Xu Li, Xi Chen, and Guojun Zhang. 2024. Parametric Feature Transfer: One-shot Federated Learning with Foundation Models. *arXiv e-prints*, Article arXiv:2402.01862 (Feb. 2024), arXiv:2402.01862 pages. https://doi.org/10.48550/arXiv.2402.01862 arXiv:2402.01862 [cs.LG]

[3] Jock Blackard. 1998. Covertype. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N.

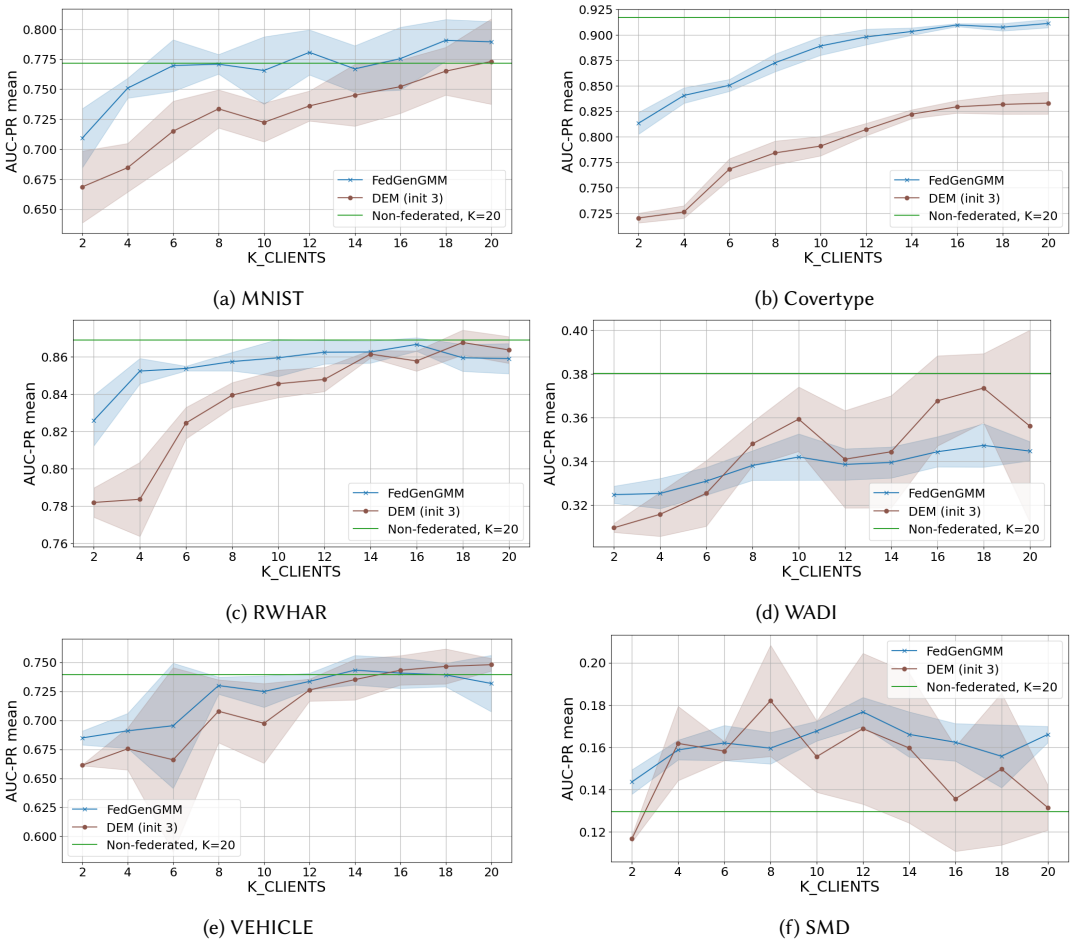


Fig. 5. Anomaly detection AUC-PR mean for each dataset and method, when varying the number of GMM components (K_CLIENTS) in the local models. The global model for FedGenGMM uses fixed $K = 20$, whereas the global DEM model uses K = K_CLIENTS. For WADI and VEHICLE, the heterogeneity scheme is Quantity($\alpha = 1$), for others it is Dir($\alpha = 0.2$). The markers indicate the mean over five runs, and the height of the shaded areas correspond to +/- the standard deviation. The green line indicates the average performance of the non-federated benchmark, using fixed $K = 20$.

[4] Marcos F. Criado, Fernando E. Casado, Roberto Iglesias, Carlos V. Regueiro, and Senén Barro. 2022. Non-IID data and Continual Learning processes in Federated Learning: A long road ahead. *Information Fusion* 88 (2022), 263–280. https://doi.org/10.1016/j.inffus.2022.07.024

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* 39 (1977), 1–38. http://web.mit.edu/6.435/www/Dempster77.pdf

[6] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477

[7] Don Kurian Dennis, Tian Li, and Virginia Smith. 2021. Heterogeneity for the Win: One-Shot Federated Clustering. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:232075682

[8] Xinrong Diao, Wei Yang, Shaowei Wang, Liusheng Huang, and Yang Xu. 2020. PrivGMM: Probability Density Estimation with Local Differential Privacy. In *Database Systems for Advanced Applications*. Springer International Publishing, Cham, 105–121.

[9] Boyu Dong, Dong Chen, Yu Wu, Siliang Tang, and Yueting Zhuang. 2024. FADngs: Federated Learning for Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems* (2024), 1–15. https://doi.org/10.1109/TNNLS.2024.3350660

[10] Shaoming Duan, Chuanyi Liu, Peiyi Han, Xiaopeng Jin, Xinyi Zhang, Tianyu He, Hezhong Pan, and Xiayu Xiang. 2022. HT-Fed-GAN: Federated Generative Model for Decentralized Tabular Data Synthesis. *Entropy* 25 (12 2022), 88. https://doi.org/10.3390/e25010088

[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042

[12] M.A.T. Figueiredo and A.K. Jain. 2002. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 3 (2002), 381–396. https://doi.org/10.1109/34.990138

[13] Neel Guha, Ameet Talwalkar, and Virginia Smith. 2019. One-Shot Federated Learning. *CoRR* abs/1902.11175 (2019). arXiv:1902.11175 http://arxiv.org/abs/1902.11175

[14] Clare Elizabeth Heinbaugh, Emilio Luz-Ricca, and Huajie Shao. 2023. Data-Free One-Shot Federated Learning Under Very High Statistical Heterogeneity. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=_hb4vM3jspB

[15] Chenxi Huang, Chaoyang Jiang, and Zhenghua Chen. 2023. Shuffled Differentially Private Federated Learning for Time Series Data Analytics. In *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*. 1023–1028. https://doi.org/10.1109/ICIEA58696.2023.10241529

[16] Salvatore Ingrassia and Roberto Rocci. 2011. Degeneracy of the EM algorithm for the MLE of multivariate Gaussian mixtures and dynamic constraints. *Computational Statistics & Data Analysis* 55, 4 (2011), 1715–1725. https://doi.org/10.1016/j.csda.2010.10.026

[17] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. *Advances and Open Problems in Federated Learning*. Now Foundations and Trends, New York, NY.

[18] Siva Kasa and Vaibhav Rajan. 2023. Avoiding inferior clusterings with misspecified Gaussian mixture models. *Scientific Reports* 13 (11 2023). https://doi.org/10.1038/s41598-023-44608-3

[19] Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota. 2020. Fusion Learning: A One Shot Federated Learning. In *Computational Science – ICCS 2020*, Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira (Eds.). Springer International Publishing, Cham, 424–436.

[20] Kaleb L. Leemaqz, Sharon X. Lee, and Geoffrey J. McLachlan. 2017. Corruption-Resistant Privacy Preserving Distributed EM Algorithm for Model-Based Clustering. In *2017 IEEE Trustcom/BigDataSE/ICESS*. 1082–1089. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.356

[21] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated Learning on Non-IID Data Silos: An Experimental Study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 965–978. https://doi.org/10.1109/ICDE53745.2022.00077

[22] Qiongxiu Li, Jaron Skovsted Gundersen, Katrine Tjell, Rafal Wisniewski, and Mads Græsbøll Christensen. 2022. Privacy-Preserving Distributed Expectation Maximization for Gaussian Mixture Model Using Subspace Perturbation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4263–4267. https://doi.org/10.1109/ICASSP43922.2022.9746144

[23] Qinbin Li, Bingsheng He, and Dawn Xiaodong Song. 2020. Practical One-Shot Federated Learning for Cross-Silo Setting. In *International Joint Conference on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:235078788

[24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (Eds.). mlsys.org. https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html

[25] Yan Li and Lei Li. 2009. A Novel Split and Merge EM Algorithm for Gaussian Mixture Model. In *2009 Fifth International Conference on Natural Computation*, Vol. 6. 479–483. https://doi.org/10.1109/ICNC.2009.625

[26] Kuo-Yun Liang, Abhishek Srinivasan, and Juan Carlos Andresen. 2022. Modular Federated Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8. https://doi.org/10.1109/IJCNN55064.2022.9892377

[27] Xiaodong Lin, Chris Clifton, and Michael Zhu. 2005. Privacy-preserving clustering with distributed EM mixture modeling. *Knowl. Inf. Syst.* 8, 1 (July 2005), 68–81.

[28] Erik William Lindskog-Münzing and Christian Prehofer. 2023. A Federated Learning Benchmark on Tabular Data: Comparing Tree-Based Models and Neural Networks. In *International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 239–246. https://doi.org/10.1109/FMEC59375.2023.10305887

[29] Abbass Madi, Oana Stan, Aurélien Mayoue, Arnaud Grivet-Sébert, Cédric Gouy-Pailler, and Renaud Sirdey. 2021. A Secure Federated Learning framework using Homomorphic Encryption and Verifiable Computing. In *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*. 1–8. https://doi.org/10.1109/RDAAPS48126.2021.9452005

[30] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated Multi-Task Learning under a Mixture of Distributions. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 15434–15447. https://proceedings.neurips.cc/paper_files/paper/2021/file/82599a4ec94aca066873c99b4c741ed8-Paper.pdf

[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[32] Kevin P. Murphy. 2012. Machine learning - a probabilistic perspective. In *Adaptive computation and machine learning series*. https://api.semanticscholar.org/CorpusID:17793133

[33] M. Nardi, L. Valerio, and A. Passarella. 2022. Anomaly Detection Through Unsupervised Federated Learning. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE Computer Society, Los Alamitos, CA, USA, 495–501. https://doi.org/10.1109/MSN57253.2022.00085

[34] Vibhor Pandhare, Jia Xiaodong, and Lee Jay. 2021. Collaborative Prognostics for Machine Fleets Using a Novel Federated Baseline Learner. *Annual Conference of the PHM Society* 13 (11 2021). https://doi.org/10.36001/phmconf.2021.v13i1.2989

[35] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2019. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), 3710–3722. https://api.semanticscholar.org/CorpusID:203736521

[36] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems* 31, 9 (2020), 3400–3413. https://doi.org/10.1109/TNNLS.2019.2944481

[37] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2828–2837. https://doi.org/10.1145/3292500.3330672

[38] Timo Sztyler and Heiner Stuckenschmidt. 2016. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 1–9. https://doi.org/10.1109/PERCOM.2016.7456521

[39] Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. 2020. Towards federated unsupervised representation learning. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking* (Heraklion, Greece) *(EdgeSys '20)*. Association for Computing Machinery, New York, NY, USA, 31–36. https://doi.org/10.1145/3378679.3394530

[40] M. Vucovich, A. Tarcar, P. Rebelo, A. Rahman, D. Nandakumar, C. Redino, K. Choi, R. Schiller, S. Bhattacharya, B. Veeramani, A. West, and E. Bowen. 2023. Anomaly Detection via Federated Learning. In *2023 33rd International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE Computer Society, Los Alamitos, CA, USA, 259–266. https://doi.org/10.1109/ITNAC59571.2023.10368517

[41] Naibo Wang, Wenjie Feng, yuchen deng, Moming Duan, Fusheng Liu, and See-Kiong Ng. 2023. Data-Free Diversity-Based Ensemble Selection for One-Shot Federated Learning. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=ORMlg4g3mG

[42] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469. https://doi.org/10.1109/TIFS.2020.2988575

[43] Yuncheng Wu, Yao Wu, Hui Peng, Juru Zeng, Hong Chen, and Cuiping Li. 2016. Differentially private density estimation via Gaussian mixtures model. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. 1–6. https://doi.org/10.1109/IWQoS.2016.7590445

[44] Yue Wu, Shuaicheng Zhang, Wenchao Yu, Yanchi Liu, Quanquan Gu, Dawei Zhou, Haifeng Chen, and Wei Cheng. 2023. Personalized federated learning under mixture of distributions. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 1577, 20 pages.

[45] Fengda Zhang, Kun Kuang, Long Chen, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. 2020. Federated unsupervised representation learning. *Frontiers of Information Technology & Electronic Engineering* 24 (2020), 1181 – 1193. https://api.semanticscholar.org/CorpusID:224725746

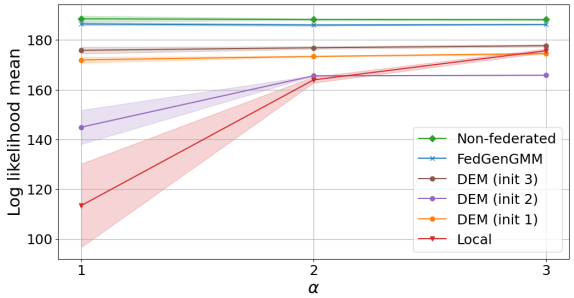
[46] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. 2024. DENSE: data-free one-shot federated learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1556, 15 pages.

[47] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. Distilled One-Shot Federated Learning. *CoRR* abs/2009.07999 (2020). arXiv:2009.07999 https://arxiv.org/abs/2009.07999

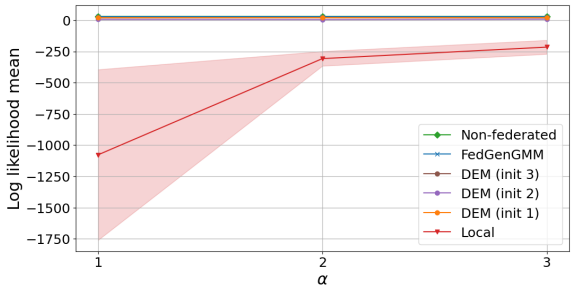
[48] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. 2021. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. *Proceedings of machine learning research* 139 (07 2021), 12878–12889.

A Full plots for estimation of the global distribution

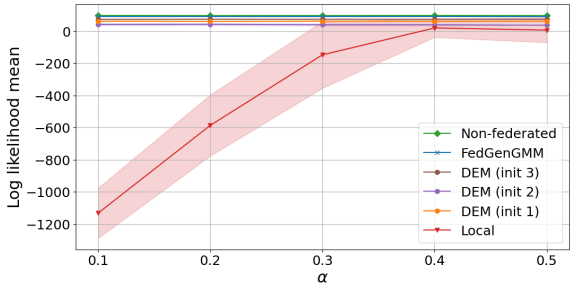
These plots include the log-likelihood mean of the local model approach (which was excluded from the plots in the main text).



(a) WADI



(b) VEHICLE



(c) SMD

Fig. 6. Resulting global model log likelihood mean for each dataset and method, using fixed K with varying heterogeneity level (alpha) of client data distributions. For WADI and VEHICLE, the heterogeneity scheme is Quantity(alpha), for SMD it is Dir(alpha). The markers indicate the mean over five runs, and the height of the shaded areas correspond to +/- the standard deviation.