EP-GAN: UNSUPERVISED FEDERATED LEARNING WITH EXPECTATION-PROPAGATION PRIOR GAN

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Networks (GANs) are overwhelming in unsupervised learning tasks due to their expressive power in modeling fine-grained data distributions. However, it is challenging for GANs to model distributions of separate non-i.i.d data partitions as it usually adopts an over-general prior, limiting its capability in capturing the latent structure of multiple data partitions and thus leading to mode collapse. In this paper, we present a new Bayesian GAN, dubbed expectation propagation prior GAN (EP-GAN), which addresses the above challenge of modeling non-i.i.d federated data through imposing a partition-invariant prior distribution on a Bayesian GAN. Furthermore, unlike most existing algorithms for deep-learning-based EP inference that require numerical quadrature, here we propose a closed-form solution for each update step of EP, leading to a more efficient solution for federated data modeling. Experiments on both synthetic extremely non-i.i.d image data partitions and realistic non-i.i.d speech recognition tasks demonstrate that our framework effectively alleviates the performance deterioration caused by non-i.i.d data.

1 INTRODUCTION

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are overwhelming in unsupervised learning tasks due to their expressive power in modeling fine-grained data distributions. GANs usually consist of a generator, which takes white noises as input and generates data points, and a discriminator, which encourages the generator to generate high-quality data points via discriminating whether a data point follows a real distribution. Successful training of GANs requires large-scale public data in a centralized setting. Recently there has also been an increasing need for training GANs on decentralized private data. Unfortunately, due to concern about data privacy, conventional ways of aggregating cross-silo data have become controversial.

On the other hand, federated learning is an increasingly important machine learning paradigm where many clients jointly train a powerful global model with cross-silo training data. The major target of federated learning is to utilize the massive data created and collected by different clients while obeying privacy protection regulations such as the European Union's General Data Protection Regulation (GDPR) (Voss, 2016). The most representative federated learning algorithm is FederatedAveraging (`FedAvg`) (McMahan et al., 2017), which successfully trains a powerful global model while keeping training data on each client (i.e., each mobile phone) on mobile applications. Since then, `FedAvg` has been quickly adopted in many other applications, such as medical image classification (Choudhury et al., 2019). Recently, more and more federated learning researchers turn their attention to a more practical setting where the data distributions are non-i.i.d. A federation usually consists of different clients from diverse sources whose data distributions are intrinsically distinct. Vanilla federated learning algorithms fail under such non-i.i.d settings, as the simple averaging tends to over-smooth the distributions of each client.

Such an over-smoothing effect is exacerbated in GANs, which notoriously often suffer from mode collapse. To gain initial insight into this issue, we conduct a toy experiment of learning a mixture of Gaussians (see Section 5.1 for more details). Fig. 1 shows the learned density map of the data centralized setting (the first column) and the extremely non-i.i.d federated learning setting (the last column), where 'red' indicates higher scores, and 'blue' indicates lower scores. Ideally, GANs should be able to rebuild four red circles around the data points, as shown in the centralized settings. We can observe that although both WGAN (Goodfellow et al., 2014) and ProbGAN (He et al., 2019) work well in the centralized setting, but in federated learning, their locally learned data distributions are over-smoothed after `FedAvg`, resulting in a collapsed mode.

This phenomenon can be interpreted from the Bayesian perspective. WGAN can be treated as a special case of Bayesian GAN by assuming a uniform prior instead of a Gaussian prior for GAN's parameters and fitting the point estimate of GAN's parameters during training. ProbGAN is a special case of Bayesian GAN, which simply adopts the generator distribution at the previous turn as the prior. Therefore, both WGAN and ProbGAN have the similar issue of enforcing an over-general prior, limiting their capability in capturing the latent multi-modal structure of non-i.i.d data partitions under federated learning setting and thus leading to mode collapse. Similar problems are also observed in the VAE literature, where researchers refer to such problems as "posterior collapse" (i.e., the learned posterior collapses to the over-general prior).

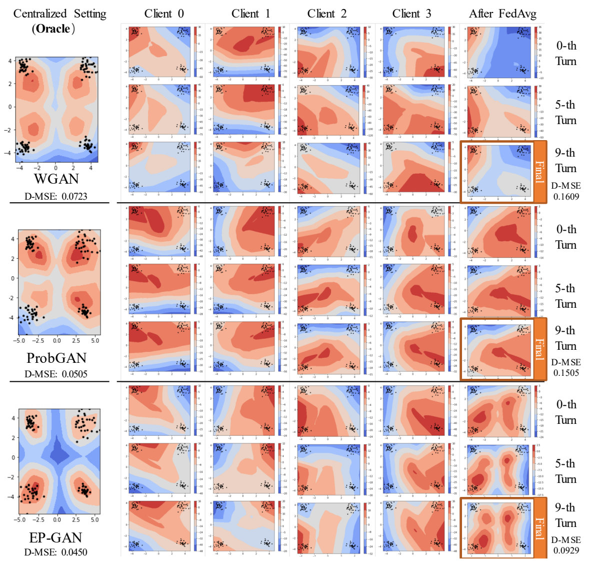


Figure 1: Density map visualizations on the toy federated datasets of WGAN, ProbGAN and EP-GAN at different federated turns. WGAN and ProbGAN suffer from mode collapse after `FedAvg` in the federated setting, while EP-GAN successfully recovers all four modes (last column). D-MSE score (defined in Appendix F.2) indicates the similarity between the density maps of GANs and the ground-truth GMM. The lower D-MSE is better.

To tackle this challenge, we present a novel GAN, dubbed expectation propagation prior GAN (EP-GAN), which is equipped with a complex prior that is learned automatically from data partitions. EP-GAN treats the prior of parameter distributions in each client as a latent variable and models the joint distribution of all clients. In doing so, EP-GAN introduces several approximate factors, where each approximates the posterior contribution from each data partition. As these approximate factors are updated iteratively and parallelly in a similar spirit of EP, we name the prior of our EP-GAN as EP-prior. The major contributions of our work are three-fold:

- We identify the mode collapse problem of Bayesian GANs under non-i.i.d cross-silo unsupervised learning scenarios. Inspired by EP, we propose an effective solution, dubbed EP-GAN, that imposes a partition-invariant prior distribution on a Bayesian GAN.
- In the solution, we derive closed-form expressions of the moments of the hybrid/tilted distribution approximations for neural networks. Unlike the previous work on EP with deep learning that requires numerical quadratures, our EP-GAN obtains a relatively more efficient closed-form solution without requiring numerical quadratures.
- Our proposed method is evaluated on real-world image and speech datasets. The experimental results demonstrate that our framework effectively alleviates the performance deterioration caused by non-i.i.d federated data compared to other state-of-the-art Bayesian GANs.

2 Related Work

In this work, we develop a new Bayesian GAN equipped with an expectation propagation (EP) prior for unsupervised federated learning under non-i.i.d settings; therefore, recent studies on federated learning, Bayesian GANs and EP with neural networks are reviewed in this section.

Federated Learning with Non-i.i.d Data. Federated learning, as a new collaborative learning paradigm, has been gaining more attention in recent years (McMahan et al., 2017; Yang et al., 2019), and the challenges of federated learning from non-i.i.d data have been noticed (Li et al., 2021; Sahu et al., 2018; Zhao et al., 2018; Xie et al., 2019; Yu et al., 2020). However, they only focus on the supervised federated learning settings, where the label information can help reduce the heterogeneity; in contrast, we focus on the unsupervised federated learning setting, where the heterogeneity embeds in the distributions across clients and these methods cannot handle well. On the other hand, some works have been proposed to adapt GANs to federated settings to deal with unsupervised federated learning tasks, but they are either proposed to improve the privacy protection of multiple GANs (Augenstein et al., 2019; Zhang et al., 2018; Jordon et al., 2018; Xie et al., 2018) or improve the communication efficiency when training in a federated manner (Rasouli et al., 2020). More importantly, they ignore another critical challenge: mode collapse of federated GANs under the non i.i.d federated learning setting. Addressing this problem is one of the main focuses of our model. The detailed discussion of existing federated GANs with non-i.i.d data is reported in Appendix C.

Bayesian GANs. Recent advances in Bayesian formulation of GANs, such as BayesGAN (Saatchi & Wilson, 2017) and ProbGAN (He et al., 2019), are proposed to address the notorious mode-collapse issue. The BayesGAN and ProbGAN assume a Gaussian prior and a carefully constructed prior for GANs, respectively. However, from the Bayesian perspective, both BayesianGAN and ProbGAN have the similar issue of enforcing an over-general prior, limiting their capability in capturing the latent multi-modal structure of non-iid data partitions under federated learning setting and thus leading to mode collapse. In contrast, our proposed EP-GAN successfully avoids such an issue by imposing a partition-invariant prior distribution inferred by EP.

EP with Neural Networks. To efficiently model the joint distribution from partitioned data, we follow the spirit of Expectation Propagation (EP). EP is one of the most popular Bayesian inference methods in Bayesian neural networks (Minka, 2013), which approximates the posterior with exponential-family factors that are iteratively updated via moment matching. However, the moment matching can be intractable if the likelihood term involves many latent variables or has a complex form, which results in the intractability of the moments of the hybrid/tilted distribution. An intuitive solution is to approximate the likelihood term by numerical quadrature (Jylänki et al., 2014; Soudry et al., 2014), but it fails to scale to large datasets or complex neural networks. The closest work to ours is proposed by Bui et al. (2018), which adopts variational inference (VI) to simulate EP, allowing to identify new ways of performance VI on federated settings with synthetic data, but their EP update is inefficient as it is simulated with gradient descent with a deliberately designed KL loss. In contrast, we propose a closed-form solution for moment matching without requiring numerical approximation and is more efficient than estimating through gradient descent.

The detailed comparisons with related works are described in Appendix C, and some of the representative methods are compared with empirical experiments, shown in Section 5.

3 PRELIMINARIES

Federated Learning. In this paper, we consider the problem of training Bayesian GANs under the cross-silo FL setting[1] (Kairouz et al., 2019), where the datasets are non-identically and isolatedly distributed on different clients. To avoid losing the focus, we leave the formulation of federated learning in Appendix D.

Expectation Propagation (EP). Before introducing our EP-GAN, we begin by reviewing the traditional EP algorithm (Minka, 2013). EP is a deterministic approximation algorithm that is often used for Bayesian inference of posterior distributions of model parameters, which is believed to be able to provide significantly more accurate approximations than variational inference methods (Jordan et al., 1999).

Consider a dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ sampled from a distribution $p(\mathbf{X}|\boldsymbol{\theta})$ whose parameter $\boldsymbol{\theta}$ follows the prior $p_0(\boldsymbol{\theta})$. As the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{X})$ is computationally intractable, EP attempts to approximate it with a tractable approximating distribution $q(\boldsymbol{\theta})$, which can be further factorized into multiple approximate factors: $q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{i=1}^N q_i(\boldsymbol{\theta})$, where the approximate factor $q_i(\boldsymbol{\theta})$ is

[1]In this work, we focus on presenting a novel solution for modeling non-i.i.d federated data, without diving into the privacy and security protection techniques. As our proposed method is a mild modification to the existing federated learning algorithm in implementation, the current advanced privacy and security protection techniques such as secure aggregation (Bonawitz et al., 2017) can be directly applied to our method.

iteratively refined so that they capture the contribution of each datapoint $\mathbf{x}_i$ to the posterior $p(\boldsymbol{\theta} \mid \mathbf{X})$:

$$p(\boldsymbol{\theta} \mid \mathbf{X}) \propto p_0(\boldsymbol{\theta}) \prod_{i=1}^{N} l_i\left(\mathbf{x}_i\right), \tag{1}$$

where the likelihood $l_i\left(\mathbf{x}_i\right) = p\left(\mathbf{x}_i \mid \boldsymbol{\theta}\right)$ and $\mathbf{X}$ represents the data distribution. More specifically, EP iterates over the following four steps:

1. Construct the cavity distribution by removing one of the approximate factor, i.e., the i-th factor. It can be written as: $q_{-i}(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{j \neq i} q_j(\boldsymbol{\theta})$
2. Integrate the likelihood contribution $l_i(\mathbf{x}_i)$ to the cavity to produce the hybrid distribution: $h_i(\boldsymbol{\theta}) \propto q_{-i}(\boldsymbol{\theta}) l_i(\mathbf{x}_i)$.
3. Update the parameters of the i-th approximate factor $q_i(\boldsymbol{\theta})$ through minimizing the KL divergence between the hybrid distribution $h_i(\boldsymbol{\theta})$ and the approximating distribution $q(\boldsymbol{\theta})$, namely, KL $[h_i(\boldsymbol{\theta}) \| q_{-i}(\boldsymbol{\theta}) q_i(\boldsymbol{\theta})]$.
4. Update the approximating distribution $q(\boldsymbol{\theta})$ by including the updated approximate factor $q_i(\boldsymbol{\theta})$: $q(\boldsymbol{\theta}) \leftarrow q_{-i}(\boldsymbol{\theta}) q_i(\boldsymbol{\theta})$.

It is worth noting that when $q_i(\boldsymbol{\theta})$ is assumed to follow an exponential family distribution (e.g. the Gaussian), the minimizing of KL divergence in Step 3 can be reduced to the moment matching (ichi Amari & Nagaoka, 2000). However, when applying EP to deep neural networks, this moment matching step is computationally intractable, requiring numerical approximation or sampling and therefore have to compromise between accuracy and efficiency. Addressing this problem is one of the main focuses of our model.

4 Expectation-Propagation Prior GAN

The local nature of EP brings about several benefits, such as distributability and parallelisability, appears to make it an ideal candidate for handling data under non-i.i.d federated learning settings. Meanwhile, Bayesian GANs (Saatchi & Wilson, 2017; He et al., 2019) have shown superior performance over traditional GANs in terms of modeling distributions for centralized data, yet fail to model distributions for non-i.i.d federated data as they suffer from mode collapse according to our carefully designed experiments. To combine the best of two worlds, in this section, we introduce a new Bayesian GAN algorithm, called expectation-propagation prior GAN (EP-GAN). While the over-general prior assumed by conventional Bayesian GANs could hinder the learning of the latent structure of multiple non-i.i.d data partitions, EP-GAN introduces several approximate factors, each approximating the posterior contribution from the local non-i.i.d data partition. As these approximate factors are updated iteratively in a similar spirit of EP, we name the prior of our Bayesian GAN as EP-prior. Our EP-prior allows EP-GAN to approximate joint posteriors of model parameters over all data partitions through integrating all approximate factors. Moreover, the closed-form EP-prior update proposed in this work makes it affordable for tractable algorithm design as the update is feasible to approximate.

In the following discussion, we assume that only the generator of GAN has the EP-prior and the parameter of the generator $\boldsymbol{\theta}_G \in \mathcal{R}^1$. Its extension to both generator and discriminator of GAN with $\boldsymbol{\theta}_G, \boldsymbol{\theta}_D \in \mathcal{R}^M$, where $M > 1$, is straightforward.

Our EP-GAN involves several variants, depending on which type of Bayesian GANs (Saatchi & Wilson, 2017; He et al., 2019) is adopted as the backbone. For simplicity, we first discuss extending the vanilla Bayesian GAN (Saatchi & Wilson, 2017) with the proposed EP-prior. The resultant model is called *EP-BGAN*. While the vanilla Bayesian GAN adopts a Gaussian prior for constructing the generator distributions, our EP-BGAN updates generator distributions based on the EP-prior, which can be written as:

$$q^{(t+1)}(\boldsymbol{\theta}_G) \propto p^{(t)}(\boldsymbol{\theta}_G | \boldsymbol{\theta}_D) \cdot q_{ep}^{(t)}(\boldsymbol{\theta}_G) \tag{2}$$

where $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_D$ denotes the parameters of generator and discriminator, respectively; $q^{(t+1)}(\boldsymbol{\theta}_G)$ is the posterior distribution of the generator at $(t+1)$-th turn; $p^{(t)}(\boldsymbol{\theta}_G | \boldsymbol{\theta}_D)$ is the likelihood of the generator proportional to $L\left(\boldsymbol{\theta}_G | \boldsymbol{\theta}_D, \mathbf{x}_i\right)$, the likelihood term depending on discriminator and data, at t-th trun, whose parameterization is similar to that of Bayesian GANs (Saatchi & Wilson, 2017; He et al., 2019); $q_{ep}^{(t)}(\boldsymbol{\theta}_G)$ represents the EP-prior at the t-th trun. We will discuss the derivation of the EP-prior in the following section.

4.1 Derivation of the Expectation-Propagation Prior

Suppose we are given K private datasets $\{\mathbf{X}_k\}_{k=1}^K$. The k-th private dataset $\mathbf{X}_k = \{\mathbf{x}_{k,i}\}_{i=1}^{n_k}$ contains n_k data items, where $\mathbf{x}_{k,i}$ is a data item contained in k-th private dataset. The goal of EP-prior is to approximate a prior distribution for model parameters of a Bayesian GAN given these datasets, such that it can lend itself to obtain an approximate posterior that captures the latent structure of cross-silo data distributions.

EP-prior adopts K approximate Gaussian factors to approximate the posterior contribution of each particular private dataset.

$$q_{ep}^{(t)}(\boldsymbol{\theta}_G) = \prod_{k=1}^{K} q_k^{(t)}(\boldsymbol{\theta}_G), \tag{3}$$

where $q_k^{(t)}(\boldsymbol{\theta}_G)$ is the approximate factor for k-th dataset. Different from Eq. (1), we omit the prior $p_0(\boldsymbol{\theta}_G)$ in the expression as we assume it is a uniform distribution for simplicity. The corresponding cavity distribution $q_{ep,-k}^{(t)}(\boldsymbol{\theta}_G)$ is written as:

$$q_{ep,-k}^{(t)}(\boldsymbol{\theta}_G) = \prod_{j\neq k} q_k^{(t)}(\boldsymbol{\theta}_G), \tag{4}$$

Since we adopt Gaussian distributions to implement the approximate factors, Step 3 of the EP update can be achieved by performing moment matching. However, if we directly adopt the product of likelihood $p^{(t)}(\boldsymbol{\theta}_G|\boldsymbol{\theta}_D)$ and the cavity distribution $q_{ep,-k}^{(t)}(\boldsymbol{\theta}_G)$ to construct the hybrid distribution, the moments of such distributions are computationally intractable as $p^{(t)}(\boldsymbol{\theta}_G|\boldsymbol{\theta}_D)$ is complex.

A likelihood function can be treated as a nonlinear function of the model parameters and the observations. Inspired by this, we propose to simplify this problem by introducing an auxiliary neural network for approximating the likelihood of each individual parameter of the generator $\boldsymbol{\theta}_G$. For simplicity, in the following discussion we use θ to denote the parameter of the generator $\boldsymbol{\theta}_G$, such that the likelihood term is approximated as:

$$l_k^{(t)}(\mathbf{x}_{k,i};\theta) = \sigma\left(f(\mathbf{x}_{k,i})\theta\right) \tag{5}$$

where f is a neural network (see the detailed implementation in Appendix F.3 and F.4.2); $\sigma(\cdot)$ denotes the Sigmoid activation function; $l_k^{(t)}(\mathbf{x}_{k,i};\theta)$ refers to the likelihood contribution of a data item $\mathbf{x}_{k,i}$. Notice that this is not the usual likelihood which describes the probability of observing a collection of data items. Since such likelihood contribution can be treated as a probability, we use Sigmoid to ensure its value to be within the range of $[0, 1]$. On the other hand, the Sigmoid and auxiliary function f introduce nonlinearity to the approximation of the likelihood contribution.

Assume the cavity distribution $q_{ep,-k}^{(t)}(\theta) = \mathcal{N}\left(\theta_{m,-k}, \theta_{s,-k}\right)$ has a Gaussian distribution with mean $\theta_{m,-k}$ and variance $\theta_{s,-k}$ and let $C = f(\mathbf{x}_{k,i})\theta$. C then still follows a Gaussian distribution, Hence denoting this distribution of C as $q_{c,-k}^{(t)}(C)$, we have that:

$$q_{c,-k}^{(t)}(C) = \mathcal{N}\left(C_{m,-k}, C_{s,-k}\right), \tag{6}$$

where $\mathcal{N}\left(C_{m,-k}, C_{s,-k}\right)$ is a Gaussian distribution with mean $C_{m,-k}$ and variance $C_{s,-k}$, which can be calculated as:

$$\begin{bmatrix} C_{m,-k} \\ C_{s,-k} \end{bmatrix} = \begin{bmatrix} f\left(\mathbf{x}_{k,i}\right)\theta_{m,-k} \\ f^2\left(\mathbf{x}_{k,i}\right)\theta_{s,-k} \end{bmatrix}. \tag{7}$$

Eq. 5-7 introduce several important terms before we can analytically approximate the EP updates (which we shall discuss in the following text).

Then we adopt the following lemma to calculate the first three moments of the likelihood term $l_k^{(t)}(\mathbf{x}_{k,i};\theta) = \sigma(C)$, namely, $E_C(\sigma(C))$, $E_C(\sigma^2(C))$ and $E_C(\sigma^3(C))$.

Lemma 4.1. *Suppose $C \sim \mathcal{N}\left(C_{m,-k}, C_{s,-k}\right)$. Let $d \geqslant 1$ be a positive integer. There exist two real constants a and b, such that the first d moments can be expressed in closed form:*

$$E_C(\sigma^d(C)) \approx \sigma\left(\frac{a\left(C_{m,-k}+b\right)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right), \tag{8}$$

whose proof is in Appendix A. Based on this lemma, the following theorem provides an analytical solution for calculation of the first two moments of the hybrid/tilted distribution:

$$h_k^{(t)}(\theta) = l_k^{(t)}(\mathbf{x}_{k,i};\theta) q_{ep,-k}^{(t)}(\theta) \tag{9}$$

Theorem 4.1. *Suppose we are given a data item the $\mathbf{x}_{k,i}$ located at k-th party during federated learning. Let $f : \mathcal{R}^{|\mathbf{x}_{k,i}|} \rightarrow \mathcal{R}^1$ be a neural network taking as input $\mathbf{x}_{k,i}$. Let θ be the parameters of the generator/discriminator of a EP-GAN. Let $C = f(\mathbf{x}_{k,i})\theta \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$. Let $h_k^{(t)}(\theta)$ be the hybrid/tilted distribution of θ as defined in Eq. (9). Let the EP-prior $q_{ep}^{(t)}(\theta) = \mathcal{N}(\theta_m, \theta_s)$, the update rules of θ_m and θ_s can be written in closed form as:*

$$\theta_m = S_1, \tag{10}$$

$$\theta_s = S_2 - S_1^2, \tag{11}$$

where

$$S_1 = \left[(C_{m,-k} + C_{s,-k}) E_C(\sigma(C)) - C_{s,-k} E_C(\sigma^2(C))\right] / S_0 f(\mathbf{x}_{k,i}), \tag{12}$$

$$S_2 = \left[(C_{m,-k} + 2C_{s,-k}) E_C(\sigma^2(C)) - 2C_{s,-k} E_C(\sigma^3(C))\right] / S_0 f^2(\mathbf{x}_{k,i}), \tag{13}$$

$$S_0 = E_C(\sigma(C)). \tag{14}$$

We leave the proof in Appendix B.

With first two moments of the hybrid distribution, S_1 and S_2, being expressed in closed form, we can follow Step 4 (Section 3) to update the EP-prior $q_{ep}^{(t)}(\theta) = \mathcal{N}(\theta_m, \theta_s)$, whose parameters θ_m and θ_s can be updated in closed form as well (Theorem 4.1). Through iterating with the four steps (discussed in Section 3), we can analytically approximate the prior distribution for model parameters of a Bayesian GAN, such that it can help the model to infer the global multi-modal data distribution for the datasets stored on different clients while following the cross-silo federated learning settings (see details of our EP algorithm in Appendix E.5).

In summary, by simplifying the likelihood term, we can derive closed-form moments for hybrid/tilted distribution approximation without requiring numerical approximation (see Appendix for the proof of Theorem 4.1). Hence, the matching of moments is in closed form, resulting in closed-form update rules for EP-prior parameters, θ_m and θ_s.

Approximation Error and Computational Efficiency. In Lemma 4.1, we use the probit function $\Phi\left(\zeta a(C+b)\right)$. to approximate $\sigma^d(C)$. Similar approximation is also adopted by Wang et al. (2016) and Wang & Manning (2013). Both the numerical and theoretical approximation analysis has been well studied in Section 3.2 (Wang et al., 2016). Our closed-form update for EP approximation is efficient in computation. Suppose there is a system with K subsamples/clients (note that K is usually known beforehand). For EP with neural networks in (Jylänki et al., 2014) which requires numerical quadratures, its computational complexity is $O(MK)$, where M is the number of quadratures points. For EP with MCMC, its computational complexity is $O(NK)$, where N is the number of MCMC samples. To approach a good approximation, both N and M should be sufficiently large (Barthelmé & Chopin, 2014). In contrast, with the closed-form update, our EP-GAN's computational complexity is reduced to $O(K)$.

4.2 Incorporating EP-prior to ProbGAN

In the previous sections, we have discussed EP-BGAN, which can be implemented by simply replacing the normal prior imposed by the vanilla Bayesian GAN (Saatchi & Wilson, 2017) with an EP-prior. However, the vanilla Bayesian GAN lacks the convergence guarantee; ProbGAN (He et al., 2019) then improves upon the vanilla Bayesian GAN by introducing a carefully crafted prior, resulting in a variant of Bayesian GAN with the desirable convergence guarantee. We therefore propose to incorporating EP-prior to ProbGAN, dubbed *EP-ProbGAN*. In EP-ProbGAN, the prior is factorized by a Gaussian prior inferred by EP and a ProbGAN prior. The EP-prior here plays the role of adjusting the original ProbGAN prior with the client-invariant distribution shift. With the spirit of Gaussian Mixture Approximation (GMA) (He et al., 2019) and to encourage our EP-prior to capture the client-invariant distribution shift, we extend the EP-prior to a mixture of EP-priors using a mixture density network (Bishop, 1994), and the corresponding model is dubbed *Mixture-EP-ProbGAN*. Details of inference methods for EP-ProbGAN and Mixture-EP-ProbGAN are left in Appendix E.

5 Empirical Experiments

In this section, we evaluate our EP-GANs on non-i.i.d cross-silo datasets to demonstrate its effectiveness of learning the global distribution on a toy federated dataset, an extremely non-i.i.d image dataset, and a realistic non-i.i.d speech recognition dataset. To keep conciseness, we report the implementation of all experiment details in Appendix F.

5.1 Toy Experiments

Setting. We evaluate our EP-GANs on a toy cross-silo non-i.i.d dataset, where 4 parts of the data, generated by 4 Gaussians centered at $(-3, 3)$, $(3, 3)$, $(-3, -3)$, $(3, -3)$, respectively, with variances randomly sampled from $[0.3, 0.5]$. The detailed model architecture, training protocol and hyper-parameters is reported in Appendix F.2.

Results. Fig. 1 visualizes the density maps (decision boundaries) produced by WGAN, ProbGAN, and EP-ProbGAN at different federated turns on the toy federated dataset. Specifically, we feed the grid points into the discriminator and use the output classification scores to construct the density map. 'Red' indicates higher scores, and 'blue' indicates lower scores. Figures in the first column of Fig. 1 show the density maps of three GANs with a centralized 4-Gaussian dataset. In this oracle setting, all GANs successfully learn the Gaussian mixture after 200 epochs. The figures in the next four columns show the density maps of the discriminators in each client after local training. The figures in the last column show the density maps of the averaged global models after the first turn, the middle turn, and the last turn, from which we can observe mode collapse after every federated aggregation of WGAN and ProbGAN (see the last column). More specifically, in WGAN and ProbGAN, `FedAvg` over-smoothes the 4 locally learned modes, causing mode collapse after global model aggregation. With the regularization of our EP-prior, EP-ProbGAN can, to some extent, preserve structural information during local training, preventing such pathological over-smoothing and successfully recovering a mixture of 4 Gaussians much closer to the ground-truth global distribution.

5.2 Synthetic Extremely non-i.i.d Image Dataset

Settings. We evaluate our method on CIFAR-10 (Hinton et al., 2012), a widely-adopted image dataset that contains 10 classes of images, with 5000 training images and 1000 test images in each class. Based on CIFAR-10, we construct three datasets corresponding to three settings: (1) i.i.d. K=2, (2) i.i.d. K=10, and (3) non-i.i.d. In doing so, we simulated two i.i.d. federated scenario and an extremely non-i.i.d scenario where each client's data sample is severely biased due to labels. The dataset construction, model architecture, and training protocols are detailed in Appendix F.3.

Baselines and EP-GAN Variants. We compare our proposed EP-GANs (short for EP-GAN variants) with multiple representative baselines, which can be categorized as:

- `FedAvg` + GANs without special design: (1) Vanilla GAN;
- `FedAvg` + Bayesian GANs: (2) BayesGAN and (3) ProbGAN;
- GANs for non-i.i.d data: (4) F2A-fed (Yonetani et al., 2019);
- GANs with only discriminators federated: (5) FedAvg-GAN (Augenstein et al., 2019).

Baselines (1)-(3) show the performance of non-Bayesian GANs and SOTA Bayesian GANs under non-i.i.d federated settings. Baselines (4) and (5) are a representative decentralized GANs method designed for addressing non-i.i.d data and a representative method that only trains discriminators locally, respectively. Each GAN model is trained with two objectives separately, i.e., WGAN (Arjovsky et al., 2017) and LS-GAN (Mao et al., 2017). We also conduct experiments on EP-BGAN to ablate the potential influence of the ProbGAN backbone in EP-PorbGAN. Moreover, we compare EP-ProbGAN with Mixture-EP-ProbGAN to see whether a mixture of multiple EP-priors can be more informative. We quantitatively evaluate the methods with two widely-used image generation metrics: Inception Score (IS) (Salimans et al., 2016) and Fréchet Inception Distance (FID) (Heusel et al., 2017).

Running Time Analysis. To facilitate the understanding of EP-prior's computational efficiency, we empirically analyze the per-epoch running time of ProbGAN and EP-ProbGAN. We kept a similar number of parameters of ProbGAN and EP-ProbGAN and trained them on CIFAR-10. Both models took 25 epochs during training with the same iterations. The average running time per epoch of ProbGAN and EP-ProbGAN are 4.3s and 5.9s, respectively; this shows that with the proposed

Table 1: IS and FID on CIFAR-10. Each model is trained on three different settings with two GAN objectives, **W**GAN and **LS**-GAN. IS and FID are reported with the corresponding relative improvement compared to ProbGAN (`FedAvg`) under the same setting.

Model	Settings	Inception Scores (Higher is better)		FIDs (Lower is better)	
GAN Objective		W	LS	W	LS
Vanilla GAN (oracle)	centralized	7.16	7.30	28.61	27.20
ProbGAN (oracle)	centralized	7.28	7.36	27.46	26.60
EP-ProbGAN (oracle)	centralized	7.34	7.38	26.03	26.42
Vanilla GAN (federated)	i.i.d. K=2	6.72±0.12	6.33±0.17	28.98	30.21
	i.i.d. K=10	5.72±0.18	5.02±0.11	45.87	55.33
	non-i.i.d	4.43±0.07	4.02±0.12	69.94	72.34
ProbGAN (He et al., 2019) (federaetd)	i.i.d. K=2	6.78±0.23	6.47±0.23	27.38	30.84
	i.i.d. K=10	5.65±0.24	4.74±0.16	41.92	60.62
	non-i.i.d	4.65±0.15	4.09±0.09	65.74	87.07
FedAvg-GAN (Augenstein et al., 2019) (federated)	i.i.d. K=2	5.46±0.12	4.99±0.16	50.25	57.72
	i.i.d. K=10	4.35±0.18	3.94±0.08	70.10	83.41
	non-i.i.d	3.42±0.12	3.06±0.14	95.89	109.65
F2A-fed (Yonetani et al., 2019) (federated)	i.i.d. K=2	5.72±0.08	5.03±0.11	48.98	55.21
	i.i.d. K=10	4.72±0.18	4.05±0.11	61.87	72.59
	non-i.i.d	3.93±0.07	3.68±0.06	76.94	90.34
EP-BGAN (ours)	i.i.d. K=2	6.94±0.18 (2.43%)	6.47±0.25 (-)	25.75 (5.81%)	30.56 (0.90%)
	i.i.d. K=10	6.01±0.23 (6.46%)	4.68±0.17 (-1.28%)	42.09 (-0.41)	64.37 (-6.18%)
	non-i.i.d	5.14±0.09 (10.69%)	4.36±0.15 (6.40%)	54.94 (16.43%)	**72.79 (16.40%)**
EP-ProbGAN (ours)	i.i.d. K=2	6.89±0.33 (1.63%)	6.49±0.17 (0.33%)	27.23 (0.01%)	30.10 (2.4%)
	i.i.d. K=10	5.87±0.20 (3.98%)	4.74±0.11 (-0.02%)	42.11 (-0.45%)	62.86 (-3.69%)
	non-i.i.d	5.17±0.14 (11.21%)	4.43±0.23 (8.26%)	52.47 (20.19%)	76.26 (12.42%)
Mixture-EP-ProbGAN (ours)	i.i.d. K=2	6.86±0.28 (1.17%)	6.50±3.00 (0.77%)	27.30 (0.29%)	30.60 (0.78%)
	i.i.d. K=10	5.90±0.20 (4.50%)	4.69±0.21 (-1.09%)	40.55 (0.50%)	62.52 (-3.13%)
	non-i.i.d	**5.18±0.21 (11.42%)**	**4.46±0.15 (9.05%)**	**51.92 (21.02%)**	73.74 (15.31%)

closed-form update, our EP-ProbGAN can run almost as fast as ProbGAN, demonstrating that the calculation of EP-prior in EP-GANs is not dominant, and that our proposed closed-form update successfully improves the computation efficiency.

Results. Results are reported in Table 1. ***Centralized vs. Non i.i.d.:*** Comparing ProbGAN's performance in the centralized and federated settings, we observe a noticeable gap, which increases as the number of partitions increases. In the non-i.i.d setting, there is a drastic drop of performance in terms of both IS and FID, which demonstrates the difficulty of unsupervised federated learning on extremely non-i.i.d datasets. ***FedAvg-GAN and F2A-fed vs. EP-GANs:*** FedAvg-GAN suffers from mode collapse when handling the extremely non-i.i.d federated data. This is because simply averaging over model parameters pathologically over-smooths the data distributions captured by local clients. F2A-fed also suffers from mode collapse as it relies on the estimation of the weight of each local discriminator when aggregating their parameters, which requires high-frequent server-client communication. Such a training strategy is unstable in federated learning, as the larger number of local updates will accumulate deviation between the global and local models. In contrast, with deliberately crafted EP-prior, our EP-GANs are capable of preserving global structural information during federated learning, achieving better performance on non-i.i.d federated data. ***ProbGAN (federated) vs. EP-GANs:*** With the additional guidance from our EP-prior, we can observe IS improvements of 10.69% and 6.40% for the objectives of '**W**GAN' and '**LS**-GAN', respectively. The improvement rates for FID are both around 16%, which demonstrates the effectiveness of EP-GAN. Moreover, we show that two EP-GAN variants, EP-BGAN and EP-ProbGAN, have roughly comparable performances. ***Mixture of EP-priors:*** After integrating multiple EP-priors, we observe a further improvement. Our Mixture-EP-ProbGAN achieves the best performance under the non-i.i.d setting. In conclusion, these comprehensive experiments demonstrate the effectiveness of EP-GAN on extremely non-i.i.d cross-silo image datasets. To further elaborate the improvement, we performed a statistical significance test on Inception Scores of EP-BGAN, EP-ProbGAN, and Mixture-EP-ProbGAN for CIFAR10. Results are listed in Appendix F.3.4.

5.3 Natural non-i.i.d Conversational Speech Dataset

Below we describe a set of experiments that demonstrate the potential of our EP-ProbGAN for classification tasks on natural real-world non-i.i.d speech datasets. More specifically, EP-GANs are used to conduct unsupervised representation learning over cross-silo non-i.i.d speech data, and the learned representations are used as additional input for the acoustic modeling in ASR.

Datasets. We evaluate EP-ProbGAN on a challenging conversational speech dataset CHiME-5 (Barker et al., 2018), whose data are collected from daily life with diverse environments and various speakers. The training set is originally collected by 16 conversation sessions, each consisting of different speakers at different places and different conversation topics, which composes a natural non-i.i.d data partition. Hence, the non-i.i.d federated speech setting has 16 clients of each session respectively for training and use the original testing set, containing 4 conversation sessions. The detailed composition of dataset is reported in Appendix F.4.1.

Evaluated Models The baseline speech recognition system is obtained by training SRU acoustic model on clients using `FedAvg`. We further incorporate GANs including EP-ProbGAN and ProbGAN to SRU acoustic model for comparing their ability of unsupervised client-invariant representation learning. Notably, the EP-ProbGAN in this experiment refers to the Mixture-EP-ProbGAN. By jointly training the EP-ProbGAN and the SRU acoustic model, we obtain a model called *SRU w/ EP-ProbGAN*. Similarly, by jointly training the ProbGAN and the SRU acoustic model, *SRU w/ ProbGAN* is obtained. Moreover, we explore incorporating EP-prior to SRU acoustic model based on SRU w/ EP-ProbGAN, noted as *SRU (EP) w/ EP-ProbGAN*. The implementation and training details are reported in Appendix F.4.2 and F.4.3.

Table 2: The WERs on CHiME-5 of different models in centralized and non.i.i.d. settings.

Model	WER
Centralized SRU (oracle)	62.6 (Huang et al., 2020)
SRU (non-i.i.d baseline)	68.58
w/ ProbGAN	66.86 (2.51%)
w/ EP-ProbGAN (ours)	66.20 (3.47%)
SRU (EP) w/ EP-ProbGAN (ours)	**64.30 (6.24%)**

Results. We report the WER of different models in Table 2. ***Centralized vs. non-i.i.d:*** With the identical SRU acoustic model, we can observe a noticeable gap between the oracle centralized setting and the realistic non-i.i.d setting, which reveals the fact that federated learning faces performance deteriorate when deploying in non-i.i.d environments. ***ProbGAN vs. EP-ProbGAN:*** Our SRU w/ EP-ProbGAN achieves around 0.7 % absolute WER reduction compared with SRU w/ ProbGAN, suggesting that our EP-ProbGAN is capable of learning a client-invariant representations for the downstream ASR task. We further augment SRU w/ EP-ProbGAN through adopting EP-prior to the SRU acoustic model itself, which outperforms SRU w/ ProbGAN in a large margin, yielding around 2.6 % absolute WER reduction. Overall, our SRU (EP) w/ EP-ProbGAN performs the best among all models under non-i.i.d setting in terms of WER, outperforming the baseline SRU trained with the non-i.i.d setting by 6.24 %. This demonstrates the potential of our EP-prior and EP-GAN for the supervised learning task.

6 Conclusion

We present a novel Bayesian GAN, named EP-GAN, for unsupervised learning from non-i.i.d cross-silo data. More importantly, we tackle defects of existing algorithms for deep-learning-based EP inference and derive a closed-form solution for each update step of EP, leading to an more efficient solution for federated data modeling. Experiments on both toy federated data and synthetic extremely non-i.i.d image data partitions demonstrate that our framework effectively alleviates the performance deterioration caused by non-i.i.d data compared to other Bayesian GANs. The realistic non-i.i.d speech recognition task validates EP-GAN's potential for classification tasks. So far, we have not thoroughly explored the potential of EP-GAN, such as exploring nonparametric methods to allow the number of EP-prior components to grow with the cross-silo data, nor have we perform experiments on other data modalities such as text, which would be an interesting future work. Meanwhile, the increasing capability of generative models prompts public concerns on the adverse social impacts, but we hope that more discussions on federated generative models will help promote related defending methods as well as related regulations to protect privacy and security.

REFERENCES

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

SS Airey and MJF Gales. Product of gaussians for speech recognition. 2003.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

Jon Barker, Shinji Watanabe, Emmanuel Vincent, and Jan Trmal. The fifth'chime'speech separation and recognition challenge: dataset, task and baselines. *arXiv preprint arXiv:1803.10609*, 2018.

Simon Barthelmé and Nicolas Chopin. Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333, 2014.

Alain Biem, Shigeru Katagiri, Erik McDermott, and Biing-Hwang Juang. An application of discriminative feature extraction to filter-bank-based speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(2):96–110, 2001.

Christopher M Bishop. Mixture density networks. 1994.

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

Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

Thang D Bui, Cuong V Nguyen, Siddharth Swaroop, and Richard E Turner. Partitioned variational inference: A unified framework encompassing federated and continual learning. *arXiv preprint arXiv:1811.11206*, 2018.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pp. 1683–1691. PMLR, 2014.

Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 2020.

Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. Differential privacy-enabled federated learning for sensitive health data. *arXiv preprint arXiv:1910.02578*, 2019.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Xiawei Guo, Quanming Yao, WeiWei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Privacy-preserving transfer learning for knowledge sharing. *arXiv preprint arXiv:1811.09491*, 2018.

Hao He, Hao Wang, Guang-He Lee, and Yonglong Tian. Probgan: Towards probabilistic gan with theoretical guarantees. In *ICLR (Poster)*, 2019.

Nicolas Heess, Daniel Tarlow, and John Winn. Learning to pass expectation propagation messages. *Advances in Neural Information Processing Systems*, 26:3219–3227, 2013.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6629–6640, 2017.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hengguan Huang, Fuzhao Xue, Hao Wang, and Ye Wang. Deep graph random process for relational-thinking-based speech recognition. In *International Conference on Machine Learning*, pp. 4531–4541. PMLR, 2020.

Shun ichi Amari and Hiroshi Nagaoka. Methods of information geometry, volume 191 of translations of mathematical monographs, 2000.

Jonathan E Ingersoll. *Theory of financial decision making*, volume 3. Rowman & Littlefield, 1987.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. Expectation propagation for neural networks with sparsity-promoting priors. *Journal of Machine Learning Research*, 15(54):1849–1901, 2014. URL `http://jmlr.org/papers/v15/jylanki14a.html`.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. *arXiv preprint arXiv:1709.02755*, 2017.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Thomas P Minka. Expectation propagation for approximate bayesian inference. *arXiv preprint arXiv:1301.2294*, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020.

Yunus Saatchi and Andrew Gordon Wilson. Bayesian gan. *arXiv preprint arXiv:1705.09558*, 2017.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 3:3, 2018.

Tim Salimans, Ian J Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, volume 1, pp. 2, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

W Gregory Voss. European union data privacy law reform: General data protection regulation, privacy shield, and the right to delisting. *The Business Lawyer*, 72(1):221–234, 2016.

Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. *arXiv preprint arXiv:1611.00448*, 2016.

Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pp. 118–126. PMLR, 2013.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.

Ryo Yonetani, Tomohiro Takahashi, Atsushi Hashimoto, and Yoshitaka Ushiku. Decentralized learning of generative adversarial networks from non-iid data. *arXiv preprint arXiv:1905.09684*, 2019.

Felix Yu, Ankit Singh Rawat, Aditya Menon, and Sanjiv Kumar. Federated learning with only positive labels. In *International Conference on Machine Learning*, pp. 10946–10956. PMLR, 2020.

Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.

Yikai Zhang, Hui Qu, Qi Chang, Huidong Liu, Dimitris Metaxas, and Chao Chen. Training federated gans with theoretical guarantees: A universal aggregation approach. *arXiv preprint arXiv:2102.04655*, 2021.

Jing Zhao, Xiao Liu, Shaojie He, and Shiliang Sun. Probabilistic inference of bayesian neural networks with generalized expectation propagation. *Neurocomputing*, 412:392–398, 2020.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A Proof of Lemma 4.1

Lemma 4.1. *Suppose $C \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$. Let $\sigma(x)$ be the sigmoid function, let $d \geqslant 1$ be a positive integer and let $\zeta^2 = \frac{\pi}{8}$. There exist two real constants a and b, such that the first d moments of $\sigma(C)$ can be expressed in closed form:*

$$E_C(\sigma^d(C)) \approx \sigma\left(\frac{a(C_{m,-k}+b)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right) \tag{15}$$

Proof. Let a, b be two real constants. Taking the probit funtion $\Phi(\zeta a(C+b))$ to approximate $\sigma^d(C)$ by matching their value and derivative at median of the probit function, we have:

$$\sigma^k(C) \approx \Phi(\zeta a(C+b))$$

where

$$\begin{aligned} a &= 2d(1-2^{-1/d}) \\ b &= \log(2^{1/d}-1) \end{aligned} \tag{16}$$

Let $d=1$, we have:

$$\sigma(C) \approx \Phi(\zeta C) \tag{17}$$

With Theorem 3 (Wang et al., 2016) and Eq. (17), we have:

$$\begin{aligned} E_C(\sigma^d(C)) &= \int \sigma^d(C)\mathcal{N}(C_{m,-k}, C_{s,-k})\,\mathrm{d}C \\ &\approx \int \Phi(\zeta a(x+b)\mathcal{N}(C_{m,-k}, C_{s,-k})\,\mathrm{d}C \\ &= \Phi\left(\frac{\zeta a(C_{m,-k}+b)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right) \\ &\approx \sigma\left(\frac{a(C_{m,-k}+b)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right) \end{aligned} \tag{18}$$

Combining Eq. (16) and Eq. (18) concludes the proof.

□

B Proof of Theorem 4.1

We formally state the Stein's Lemma here, following the version in (Ingersoll, 1987).

Lemma B.1. *Let $C \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$, and let g be a differentiable function satisfying $E[g'(X)] < \infty$. Then*

$$E[g(C)(C-C_{m,-k})] = C_{s,-k}E[g'(C)] \tag{19}$$

Theorem 4.1. *Suppose we are given a data observation $\mathbf{x}_{k,i}$. Let $f : \mathcal{R}^{|\mathbf{x}_{k,i}|} \rightarrow \mathcal{R}^1$ be a neural network taking as input the data $\mathbf{x}_{k,i}$. Let θ be the parameters of the generator/discriminator of a EP-GAN. Let $C = f(\mathbf{x}_{k,i})\theta \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$. Let $h_k^{(t)}(\theta)$ be the hybrid/tilted distribution of θ as defined in Eq. (9). Let the EP-prior $q_{ep}^{(t)}(\theta) = \mathcal{N}(\theta_m, \theta_s)$, we have the closed-form update rules of θ_m and θ_s:*

$$\theta_m = S_1, \tag{20}$$
$$\theta_s = S_2 - S_1^2, \tag{21}$$

where

$$S_1 = \left[(C_{m,-k}+C_{s,-k})E_C(\sigma(C)) - C_{s,-k}E_C(\sigma^2(C))\right] / S_0 f(\mathbf{x}_{k,i}), \tag{22}$$
$$S_2 = \left[(C_{m,-k}+2C_{s,-k})E_C(\sigma^2(C)) - 2C_{s,-k}E_C(\sigma^3(C))\right] / S_0 f^2(\mathbf{x}_{k,i}), \tag{23}$$
$$S_0 = E_C(\sigma(C)). \tag{24}$$

Proof. With $C \sim \mathcal{N}\left(C_{m,-k}, C_{s,-k}\right)$ and Lemma B.1, we have that the first three moments of $\sigma(C)$ can be expressed in closed form, e.g.:

$$E_C(\sigma(C)) \approx \sigma\left(\frac{(C_{m,-k})}{\sqrt{1+\zeta^2 C_{s,-k}}}\right); \tag{25}$$

$$E(\sigma^2(C)) \approx \sigma\left(\frac{a\left(C_{m,-k}+b\right)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right), \tag{26}$$

where $\zeta^2 = \frac{\pi}{8}$, $a = 4 - 2\sqrt{2}$, and $b = \ln(\sqrt{2}+1)$;

$$E(\sigma_C^3(C)) \approx \sigma\left(\frac{a\left(C_{m,-k}+b\right)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right), \tag{27}$$

where $\zeta^2 = \frac{\pi}{8}$, $a = 6\left(1-\frac{1}{\sqrt[3]{2}}\right)$, and $b = \ln(\sqrt[3]{2}-1)$.

Let the normalizer $S_0 = \int h_k^{(t)}(\theta)\mathrm{d}\theta$. With Eq. (25), we have:

$$\begin{aligned} S_0 &= \int \sigma\left(f\left(\mathbf{x}_{k,i}\right)\theta\right) q_{ep,-k}^{(t)}(\theta)\mathrm{d}\theta \\ &= E_C(\sigma(C)) \\ &\approx \sigma\left(\frac{(C_{m,-k})}{\sqrt{1+\zeta^2 C_{s,-k}}}\right) \end{aligned} \tag{28}$$

Let the first moment of θ, $S_1 = \int \theta h_k^{(t)}(\theta)\mathrm{d}\theta / \int h_k^{(t)}(\theta)\mathrm{d}\theta$. We have:

$$S_1 = \frac{\int f(\mathbf{x}_{k,i})\theta\sigma\left(f\left(\mathbf{x}_{k,i}\right)\theta\right) q_{ep,-k}^{(t)}(\theta)\mathrm{d}\theta}{f(\mathbf{x}_{k,i})S_0} = \frac{E_C\left(C\cdot\sigma(C)\right)}{f(\mathbf{x}_{k,i})S_0} \tag{29}$$

We then take Lemma B.1 to compute $E_C\left(C\cdot\sigma(C)\right)$, we have

$$E_C\left(\sigma(C)\cdot(C-C_{m,-k})\right) = C_{s,-k}E_C\left(\sigma'(C)\right) \tag{30}$$

Then we have:

$$E_C(\sigma(C)\cdot C) = \left(C_{s,-k}+C_{m,-k}\right)E_C(\sigma(C)) - C_{s,-k}E_C(\sigma^2(C)), \tag{31}$$

With Eq. (29) and Eq. (31), we therefore have:

$$S_1 = \frac{\left[\left(C_{m,-k}+C_{s,-k}\right)E(\sigma(C)) - C_{s,-k}E(\sigma^2(C))\right]}{S_0 f(\mathbf{x}_{k,i})}. \tag{32}$$

Combining Eq. (25), Eq. (26), Eq. (28) and Eq. (32) , we have the first moment S_1 being expressed in closed form.

Let the second moment of θ, $S_2 = \int \theta^2 h_k^{(t)}(\theta)\mathrm{d}\theta / \int h_k^{(t)}(\theta)\mathrm{d}\theta$. We have:

$$S_2 = \frac{\int \theta^2\sigma\left(f\left(\mathbf{x}_{k,i}\right)\theta\right) q_{ep,-k}^{(t)}(\theta)\mathrm{d}\theta}{f^2(\mathbf{x}_{k,i})S_0} = \frac{E_C\left(C^2\cdot\sigma(C)\right)}{f^2(\mathbf{x}_{k,i})S_0} \tag{33}$$

With Lemma B.1, we have

$$E_C\left(\sigma(C)\cdot C\cdot(C-C_{m,-k})\right) = C_{s,-k}E_C\left(\sigma(C) + \left(\sigma(C)-\sigma^2(C)\right)\cdot C\right) \tag{34}$$

Then we have:

$$E_C\left(C^2\cdot\sigma(C)\right) = C_{s,-k}E\left(\sigma(C)\right) + \left(C_{s,-k}+C_{m,-k}\right)E_C(C\cdot\sigma(C)) - C_{s,-k}E_C(\sigma^2(C)\cdot C) \tag{35}$$

We further adopt Lemma B.1 to derive $E_C(\sigma^2(C) \cdot C)$, then we have:

$$E_C\left(\sigma^2(C)(C - C_{m,-k})\right) = C_{s,-k} E_C\left(2\sigma^2(C) - 2\sigma^3(C)\right) \tag{36}$$

Then

$$E_C\left(\sigma^2(C) \cdot C\right) = \left(C_{m,-k} + 2C_{s,-k}\right) E_C(\sigma^2(C)) - 2C_{s,-k} E_C(\sigma^3(C)) \tag{37}$$

With Eq. (33), Eq. (35) and Eq. (37), we have:

$$S_2 = \frac{\left[\left(C_{m,-k} + 2C_{s,-k}\right) E(\sigma^2(C)) - 2C_{s,-k} E(\sigma^3(C))\right]}{S_0 f^2(\mathbf{x}_{k,i})}. \tag{38}$$

Combining Eq. (26)-(28) and Eq. (38), we have the second moment S_2 being expressed in closed form as well.

With first two moments of θ, S_1 and S_2, being expressed in closed form, we have the EP-prior $q_{ep}^{(t)}(\theta) = \mathcal{N}(\theta_m, \theta_s)$, whose parameters θ_m and θ_s can be updated in close-form:

$$\theta_m = S_1, \tag{39}$$

$$\theta_s = S_2 - S_1^2, \tag{40}$$

$\square$

C Detailed Comparison with Related Works

Federated Learning with non-i.i.d Datasets. While federated learning has been gaining more and more attention in recent years (McMahan et al., 2017; Yang et al., 2019; Chen et al., 2020; Brisimi et al., 2018; Guo et al., 2018), few works have explored federated learning with non-i.i.d datasets (Li et al., 2021). Sahu et al. (2018) extend FedAvg with a proximal term, named FedProx, considering the dissimilarity between clients, and provides a theoretical analysis of convergence under a similarity assumption in a view of parameter divergence. They only demonstrate their effectiveness on supervised learning datasets. Zhao et al. (2018) address supervised federated learning on non-i.i.d data with parameter regularization. Similarly, Xie et al. (2019) propose an asynchronous federated optimization algorithm, and shows that their method can achieve comparable performance as FedAvg (which is synchronous) with asynchronous optimization in supervised learning tasks.

Closest to our work is proposed by Yu et al. (2020), which uses labels to regularize local and global models during local training. Different from aforementioned works which focus on the supervised learning setting and therefore requires access to labels, we focus on unsupervised learning without label information. Another closely related work proposed by Yonetani et al. (2019), which studies the problem of learning vanilla GANs from multiple separate and non-i.i.d data collections under the decentralized learning setting where the server and clients can communicate at a low cost. Similarly, Zhang et al. (2021) propose UA-GAN, which improves multi-discriminator GAN by providing theoretical guarantee through carefully crafted aggregation of distributed discriminators, which is design to simulate the centralized discriminator. However, their works require highly frequent server-client communication, which is impractical in the federated learning (McMahan et al., 2017) that our work focuses on.

Generative Adversarial Networks The existing federated GAN methods focus on addressing the practical issues on adapting the training of GANs to federated scenarios from two directions, privacy protection and communication efficiency.

DP-FedAvg-GAN (Augenstein et al., 2019) is one of the earliest works that combine GANs with federated learning, whose target is to address the possible privacy leakage problem. The core of their method is two-fold. First, they argue that the training of Generator can be conducted at the server and only the discriminators are trained locally and aggregated at the server. Second, the training of GANs is protected by a differentially private FedAvg algorithm (DP-FedAvg). Apart from DP-FedAvg-GAN, there are multiple works combining GANs with privacy protection techniques, such as DPGAN (Xie et al., 2018), dp-GAN (Zhang et al., 2018), and PATE-GAN Jordon et al. (2018). Although our work is not directly proposed to improve the privacy protection of GANs under federated settings, the aforementioned privacy protection techniques, e.g. DP-Fedavg, can be directly applied to our framework. Moreover, adding DP to the algorithm usually harms the performance, as it introduces noises to the model.

On the other hand, the aforementioned federated/decentralized GANs suffer from communication inefficiency, as they require large communication bandwidth among data sources or between data sources and an intermediary (to ensure convergence) due to architectures that separate generators from discriminators. Rasouli et al. (2020) propose to train discriminators and generators locally, and the server (intermediary) only averages the agents' parameters. They empirically show that with their strategy, they can achieve less communication overhead and faster convergence, but they do not focus on the mode collapse of GANs caused by the non-i.i.d data. In their paper, authors claim to achieve similar performance to general distributed GAN. We take a similar strategy to train EP-GAN in the non-i.i.d federated setting, i.e., generators and discriminators are updated at the clients, and the server only averages the parameters from clients.

Also related to our work are recent advances in Bayesian formulation of GANs, such as BayesGAN (Saatchi & Wilson, 2017) and ProbGAN (He et al., 2019), to address the notorious mode-collapse issue. The Bayesian GAN assumes a Gaussian prior for both generator and discriminator distribution, while ProbGAN, as a special case of Bayesian GAN, simply adopts the generator distribution at the previous iteration as the prior. Therefore, from the Bayesian perspective, both BayesianGAN and ProbGAN have the similar issue of enforcing an over-general prior, limiting their capability in capturing the latent multi-modal structure of non-iid data partitions under federated learning setting and thus leading to mode collapse. In contrast, our proposed EP-GAN successfully avoids such an issue by treating the prior of parameter distributions in each client as a latent variable and modeling the joint distribution of all clients as a product of these latent variables.

Expectation Propagation for Bayesian Neural Networks. Expectation Propagation is one of the most popular Bayesian inference methods in Bayesian neural networks (Minka, 2013). It approximates the posterior with exponential-family factors, which are iteratively updated via moment matching.

A critical difficulty of applying EP to neural networks is that the moment matching can be intractable if the likelihood term involves many latent variables or has a complex form, which results in the intractability of the moments of the hybrid/tilted distribution. Jylänki et al. (2014), as the initial trial of EP to a shallow neural network, attempt to approximate the likelihood term by numerical quadrature. However, such a method requires numerical approximation for each data point, limiting its scalability to large datasets. A more scalable solution is proposed in (Soudry et al., 2014). Though it has been successfully applied to neural networks with binary weights, its extension to continuous weights fails to produce estimates of posterior variance. In contrast, we propose a close-formed solution for moment matching without requiring numerical approximation. Our method is capable of capturing global latent structure for complex Bayesian neural networks such as Bayesian GANs over cross-silo realistic datasets.

The closest work to ours is (Bui et al., 2018), which adopts variational inference (VI) to simulate EP, allowing to identify new ways of performing VI on federated learning settings. However, their EP update is inefficient as it is calculated via gradient descent with a deliberately designed KL loss, while our EP-GAN adopts a closed-form solution for the EP update that is more efficient than estimating through gradient descent. Meanwhile, their approach doesn't consider the realistic non-i.i.d scenarios. To confirm the significance of the contribution in terms of EP with neural networks, we give a comparison of related works with our EP-GAN in the following table:

Table 3: The comparison of supported features of representative methods

Representative method	Convergence guarantee	Sampling-free EP update	Closed-form EP update approximation	Numerical approximation/quadrature free
Jylänki et al. (2014)	✗	✓	✓	✗
Bui et al. (2018)	✗	✗	✗	✗
Heess et al. (2013)	✗	✗	✗	✗
Zhao et al. (2020)	✗	✗	✗	✗
Ours	✗	✓	✓	✓

D Federated Learning Formulation

Following the popular federated learning (FL) formulation (Li et al., 2019), we formulate unsupervised federated learning (UFL) problem with the unlabeled data distributed on separate clients. Denote K as number of clients in the UFL system, and the private dataset in the party $k \in \{1...K\}$ as $\mathbf{X}_k$. We have K clients with their private datasets $\mathbf{X}_k = \{\mathbf{x}|\mathbf{x} \sim p_k(x)\}$ drawn from non-identical distributions, i.e., $p_k(\mathbf{x}) \neq p_{k'}(\mathbf{x})$ for all $k \neq k'$. The UFL target is to optimize the model with private data distributed on different clients:

$$\min_{\mathbf{w}} F(\mathbf{w}) \triangleq \sum_{k=1}^{K} v_k F_k(\mathbf{w}), \tag{41}$$

where F and F_k represent the global model and local model of client k respectively, $\mathbf{w}$ represents the model parameters, and v_k is the weight of the corresponding party k such that $v_k \geq 0$ and $\sum_{k=1}^{K} v_k = 1$.

In UFL, each party k optimizes their local model on its private data set $\mathbf{X}_k$ with the objective $F_k(\cdot)$,

$$F_k(\mathbf{w}) \triangleq \frac{1}{n_k} \sum_{i=1}^{n_k} \ell\left(\mathbf{w}; x_{k,i}\right), \tag{42}$$

where $\ell(\cdot;\cdot)$ is the localized loss function, n_k is the number of instances in $\mathbf{X}_k$.

E Details of Inference for EP-GANs

In this section, we introduce details of EP-GANs inference in practice that we omit in the main paper.

E.1 Inference for EP-ProbGAN

For EP-ProbGAN, the update of generator distributions becomes the following:

$$q^{(t+1)}(\boldsymbol{\theta}_G) \propto p^{(t)}(\boldsymbol{\theta}_G|\boldsymbol{\theta}_D) \cdot \left[q^{(t)}(\boldsymbol{\theta}_G) \cdot \tilde{p}_{ep}^{(t)}(\boldsymbol{\theta}_G)\right], \tag{43}$$

where $q^{(t)}(\boldsymbol{\theta}_G)$ is the prior imposed by ProbGAN, which is essentially the generator distribution at the previous update iteration; $p^{(t)}(\boldsymbol{\theta}_G|\boldsymbol{\theta}_D)$ is the likelihood term. Overall, the prior of the new Bayesian GAN is factorized by EP-prior $\tilde{p}_{ep}^{(t)}(\boldsymbol{\theta}_G)$ and the ProbGAN prior $q^{(t)}(\boldsymbol{\theta}_G)$. The EP-prior here plays the role of adjusting the original ProbGAN prior with the client-invariant distribution shift.

We then develop the inference algorithm to compute the posterior. To approximate the usually intractable posterior, we follow the strategy of Bayesian GANs (Saatchi & Wilson, 2017; He et al., 2019) and adopt the Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) technique to generate samples. We generate N samples as $\{\boldsymbol{\theta}_{G,n}^{(t)}\}_{n=1}^{N}$ from generator distribution $q^{(t)}(\boldsymbol{\theta}_G)$. With the adapted SGHMC algorithm (described in Appendix E.4), we can sample the consequent posterior $q_G^{(t+1)}(\boldsymbol{\theta}_G)$ from gradients by:

$$\nabla_{\boldsymbol{\theta}_G} \log q^{(t+1)}(\boldsymbol{\theta}_G) = \nabla_{\boldsymbol{\theta}_G}\left(\mathcal{J}_G\left(\boldsymbol{\theta}_G; D^{(t)}\right) + \log q^{(t)}(\boldsymbol{\theta}_G) + \log \tilde{p}_{ep}^{(t)}(\boldsymbol{\theta}_G)\right). \tag{44}$$

Calculating the gradient of the first term is trivial, while the remaining terms have no exact analytic form. We therefore adapt Partial Summation Approximation (PSA) (He et al., 2019) to calculate the second term (described in Appendix E.3).

E.2 Inference for Mixture-EP-ProbGAN

With the spirit of Gaussian Mixture Approximation (GMA) (He et al., 2019) and to encourage our EP-prior to capture the client-invariant distribution shift, we re-parameterized our EP-prior in the following form:

$$\tilde{p}_{ep}^{(t)}(\boldsymbol{\theta}_G) = C \cdot \exp\left[\sum_{n=1}^{N} \|\theta_m - \boldsymbol{\theta}_{G,n}^{(t)}\|_2^2 \,/\, 2\theta_s\right], \tag{45}$$

where C is a constant; N is the number of generated samples $\{\boldsymbol{\theta}_{G,n}^{(t)}\}_{n=1}^{N}$ from generator distribution; θ_m and θ_s are the mean and variance of the old EP-prior. The new EP-prior provides the way of linking the old EP-prior with generator distribution $q^{(t)}(\boldsymbol{\theta}_G)$ at previous updating iteration in achieving client-invariant model distribution for the generator. To enhance the expressive power of EP-priors, we derive a mixture of multiple EP-priors using a mixture density network (Bishop, 1994).

Assume we have N observations of generator parameters $\{\boldsymbol{\theta}_{G,n}^{(t)}\}_{n=1}^{N}$, and J EP-priors in a mixture. The GMA for mixture of EP-priors can be written as:

$$\tilde{p}_{ep}^{(t)}(\boldsymbol{\theta}_G) \approx C \cdot \sum_{j=1}^{J} \pi_j \exp\left[\sum_{n=1}^{N} \frac{\|\theta_{m,j} - \boldsymbol{\theta}_{G,n}^{(t)}\|_2^2}{2\theta_{s,j}}\right], \tag{46}$$

where $\sum_{j=1}^{J} \pi_j = 1$; C is the normalization constant; $\theta_{m,j}$ and $\theta_{s,j}$ are the mean and variance of the j-th EP-prior in the mixture. Notably, in practice, $\{\pi\}_{j=1}^{J}$ are estimated with Gumble-Softmax (Jang et al., 2016).

Algorithm 1 Our adapted SGHMC algorithm for EP-GANs

```
1:  Input: Initial Monte Carlo samples of $\{\boldsymbol{\theta}_{G,N}^{(0)}\}_{n=1}^{N}$, learning rate $\eta$, SGHMC noise factor $\alpha$,
    number of updates in SGHMC procedure $L$, number of updating iterations $T$
2:  for $t = 1, ..., T$ do
3:      for $n = 1, 2..., N$ do
4:          $\boldsymbol{\theta}_{G,n} \leftarrow \boldsymbol{\theta}_{G,n}^{(t-1)}$
5:          for $l = 1, 2, ..., L$ do
6:              $\mathbf{v} \leftarrow (1-\alpha)\mathbf{v} + \eta\nabla_{\boldsymbol{\theta}_G} \log q^{(t)}(\boldsymbol{\theta}_{G,n}) + \mathbf{n}$;
                $\mathbf{n} \sim \mathcal{N}(0, 2\alpha\eta I)$
7:              $\boldsymbol{\theta}_{G,n} \leftarrow \boldsymbol{\theta}_{G,n} + \mathbf{v}$
8:          end for
9:          $\boldsymbol{\theta}_{G,n}^{(t)} \leftarrow \boldsymbol{\theta}_{G,n}$
10:     end for
11: end for
```

E.3 Partial Summation Approximation (PSA)

We adopt Partial Summation Approximation (PSA) (He et al., 2019) for computing the second term of Eq. (44) in the main paper, which can be written as:

$$\nabla_{\boldsymbol{\theta}_G} \log q^{(t)}(\boldsymbol{\theta}_G) = \nabla_{\boldsymbol{\theta}_G} \mathcal{J}_G\left(\boldsymbol{\theta}_G; D^{(t-1)}\right) + \nabla_{\boldsymbol{\theta}_G} \log q^{(t)}(\boldsymbol{\theta}_G) = \sum_{i=0}^{t-1} \nabla_{\boldsymbol{\theta}_G} \mathcal{J}_G\left(\boldsymbol{\theta}_G; D^{(i)}\right) \tag{47}$$

It is calculated by summing over all historical GAN objective gradients. However, such operation is computationally expensive. We therefore follow the implementation in (He et al., 2019) to improve the computation efficiency.

E.4 The SGHMC for EP-GANs

We utilize the Stochastic Gradient HMC (SGHMC) algorithm (Chen et al., 2014) to sample the posterior distributions over the generators, as it is convenient and practically beneficial to integrate with GAN training (Saatchi & Wilson, 2017). Adapted from He et al. (He et al., 2019), we propose an adapted SGHMC algorithm for EP-GAN sampling, as shown in Algorithm 1. The gradient $\nabla_{\boldsymbol{\theta}_G} \log q^{(t)}(\cdot)$ in the 6-th line of Algorithm 1 is inferenced from the GMA mentioned in above sections.

E.5 EP-GAN WITH FEDERATED LEARNING

EP-GAN can easily be adapted to the commonly used federated learning algorithm (`FedAvg`), as shown in Algorithm 2.

Algorithm 2 EP-GAN with `FedAvg`

```
1: Input: $K$ clients, and $n_k$ examples at the $k$-th client; $L$ be the local training iterations; $T$ be the total number of training iterations.
2: Server initializes model parameters $\left(\boldsymbol{\theta}_G^{(0)}, \boldsymbol{\theta}_D^{(0)}\right)$, Global EP-prior $\left(\theta_m^{(0)}, \theta_s^{(0)}\right)$ and broadcasts to all clients.
3: for $t = 0, ..., T-1$ do
4:     for $k = 1, ..., K$ (parallelly) do
5:         Update the EP-prior $(\theta_{m,k}^{(t)}, \theta_{s,k}^{(t)})$ according to Eq. (10) and Eq. (11).
6:         Update the model parameters $(\boldsymbol{\theta}_{G,k}^{(t)}, \boldsymbol{\theta}_{D,k}^{(t)})$ with Algorithm 1.
7:         if $(t > 0$ and $t$ mod $L == 0)$ or $(t == T-1)$ then
8:             Upload $\left(\boldsymbol{\theta}_{G,k}^{(t)}, \boldsymbol{\theta}_{D,k}^{(t)}\right)$ and $\left(\theta_{m,k}^{(t)}, \theta_{s,k}^{(t)}\right)$ to server.
9:             Server updates the model parameters:
                   $$\boldsymbol{\theta}_G^{(t+1)} = \sum_{k=1}^{K} \frac{1}{n_k} \boldsymbol{\theta}_{G,k}^{(t)}, \quad \theta_d^{(t+1)} = \sum_{k=1}^{K} \frac{1}{k_i} \boldsymbol{\theta}_{D,k}^{(t)}.$$
10:            Server updates the global EP-prior with the mixture as Product-of-Gaussian (Airey & Gales, 2003)
                   $$\boldsymbol{\theta}_s^{(t+1)} = \sum_{k=1}^{K} \left(\boldsymbol{\theta}_{s,k}^{(t)}\right)^{-1}, \quad \boldsymbol{\theta}_m^{(t+1)} = \boldsymbol{\theta}_s^{(t+1)} \sum_{k=1}^{K} \frac{\boldsymbol{\theta}_{m,k}^{(t)}}{\boldsymbol{\theta}_{s,k}^{(t)}}$$
11:            The server broadcasts $\left(\boldsymbol{\theta}_G^{(t)}, \boldsymbol{\theta}_D^{(t)}\right)$ and $\left(\theta_m^{(t)}, \theta_s^{(t)}\right)$ to the $k$-th client:
                   $$\left(\boldsymbol{\theta}_{G,k}^{(t)}, \boldsymbol{\theta}_{D,k}^{(t)}\right) \leftarrow \left(\boldsymbol{\theta}_G^{(t)}, \boldsymbol{\theta}_D^{(t)}\right), \quad \left(\theta_{m,k}^{(t)}, \theta_{s,k}^{(t)}\right) \leftarrow \left(\theta_m^{(t)}, \theta_s^{(t)}\right)$$
12:        end if
13:    end for
14: end for
```

The algorithm runs on a federation consisting of a central server and K clients, where the k-th client has n_k training examples. The algorithm conducts T updating iterations at each client. After every L updates, the clients send updated parameters to the server; the server aggregates the received model parameters with weighted averaging (McMahan et al., 2017) and computes the new EP-prior parameter with Product-of-Gaussian (Airey & Gales, 2003).

F EXPERIMENT DETAILS

F.1 EXPERIMENTAL ENVIRONMENTS

We run our experiments on a machine with 314 GB RAM, an Intel Xeon PHI 7290 CPU Processor, and two Tesla V100 GPUs. The operation system is CentOS 7. For detailed versions of software environments, please refer to the `README.md` files in corresponding code projects.

F.2 EP-GAN FOR TOY EXPERIMENTS

F.2.1 SYNTHETIC DATASET DETAILS

For all models, we examine them on the same generated dataset, which is shown in Figure 2. Data points with identical colors belong to the same client. In order to simulate a more realistic distribution, the variances of generated samples are random.

F.2.2 QUANTIFIABLE SCORE

To facilitate ascertaining the gains, we propose a quantifiable score, Density Mean Square Error (D-MSE), to supplement the evaluation of density map figures. The D-MSE score is defined as the MSE between two density maps f_P and f_Q over a given sample space $\mathcal{X}$, i.e.,

$$\texttt{D-MSE}(f_P, f_Q) = \frac{1}{N} \sum_{x \in \mathcal{X}} (f_P(x) - f_Q(x))^2, \quad (48)$$

where N is the total number of sampling poits. Compared to KL divergence $D_{\mathrm{KL}}(P \| Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$, which explodes to infinity when $P(x) >> Q(x)$, D-MSE is more suitable in our case to compare different density maps generated by GANs given the ground-truth GMM. To achieve the D-MSE score, we uniformly sample 10000 points from a square area from $(-5, -5)$ to $(5, 5)$, and compute the score according to Eq. 48 with density map from the ground-truth GMM as f_P and density map from GAN as f_Q. With the quantifiable metric, we are able to clearly observe the gains of our EP-GANs in the federated setting.

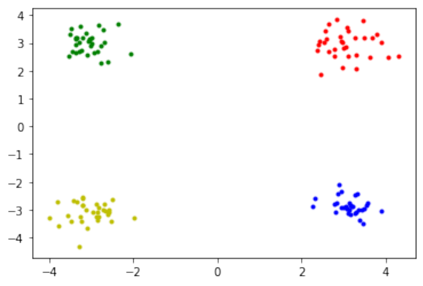


Figure 2: The generated non-i.i.d four Gaussian Dataset.

F.2.3 IMPLEMENTATION DETAILS

Model Architectures. We evaluate three models, WGAN, ProbGAN, and our proposed EP-ProbGAN.

WGAN: We implement a WGAN (Arjovsky et al., 2017) with gradient penalty. The generator is a 3-layer MLP with a hidden size of 32. Between each layer, we employ a batch normalization layer. The discriminator is a 3-layer MLP, with a batch normalization layer after the first MLP layer. We use the ReLU activation function (Agarap, 2018) between linear layers.

ProbGAN: Based on the implementation of WGAN, ProbGAN extends it with multiple discriminators and generators. Specifically, we adopt the parameter sharing technique to implement the Bayesian discriminators and generators, i.e., the last two layers in Bayesian generators and the first two layers in the discriminators are shared in implementation.

EP-ProbGAN: EP-ProbGAN utilizes ProbGAN as the backbone. Apart from the main structure of ProbGAN, EP-ProbGAN introduce parameters of the mixture of EP-priors $\boldsymbol{\theta}_m = [\theta_{m,1}, ..., \theta_{m,J}], \boldsymbol{\theta}_s = [\theta_{s,1}, ..., \theta_{s,J}]$, where J is the number of components in the mixture. Moreover, EP-ProbGAN introduces a neural network $f(\mathbf{x})$ for likelihood approximation, as mentioned in Eq. (5). To take advantage of the ability to capture input features, $f(\mathbf{x})$ reuses the first two layers of the discriminator, appending with an independent output layer.

For federated learning, we propose a historical parameter interpolation technique for clearer visualization. After each epoch of training local models, the current parameters are interpolated with the latest global model broadcasted to the client.

Training Protocol. The goal of this experiment is to learn a global distribution of 4 Gaussians on the 4-mode toy dataset using `FedAvg`. For fair comparisons, we fix the hyper-parameters of all GANs, including the number of layers, their width, the number of local training epochs, and the number of federated learning turns. The number of local training epochs indicates how many epochs are conducted inside the client between two consecutive federated aggregations of model parameters.

Hyper-parameter Selection. We use the Adam optimizer (Kingma & Ba, 2014) for generators and discriminators with learning rates of 2×10^{-4} and 1×10^{-4}, respectively; they are selected from the range $[1 \times 10^{-2} \sim 1 \times 10^{-5}]$ with a validation set. For federated learning on all models, we first train local models within clients for 50 epochs as a warm-up, and then in each of the remaining 10 aggregation turns, the number of local training epochs is 20. The numbers of samples of Bayesian generators and discriminators are both 4 in ProbGAN and EP-ProbGAN. The number of components in the mixture of EP-priors is 4.

F.3 EP-GAN for Extremely non-i.i.d Image Datasets

F.3.1 Dataset Details

The datasets of three settings are constructed as following: (1)i.i.d. N=2: randomly split the whole training set into two parts, (2)i.i.d. N=10: randomly split the whole training set into ten parts, and (3)non-i.i.d the extremely non-i.i.d data partitions with 10 clients. To simulate the extremely non-i.i.d federated setting (the third setting), the dataset is divided into 10 partitions of equal size, and each partition contains only images with the same label. In doing so, we simulated an extremely non-i.i.d scenario where each client's data sample is severely biased due to labels. We use the official testing set split for evaluation.

F.3.2 Implementation Details

The models are all implemented in PyTorch (Paszke et al., 2017), with the adaptation from the open-source code of ProbGAN(He et al., 2019)[2]. Besides ProbGAN, we implement the EP-BGAN (extended from BayesGAN (Saatchi & Wilson, 2017)), EP-ProbGAN, and Mixture-EP-ProbGAN. These models are then trained with `FedAvg` on the extremely non-i.i.d dataset.

Model Architectures. All of the models apply the similar DCGAN (Radford et al., 2015) as the backbone.

The generators consist of 4 deconvolution layers (kernel size 4, stride 2), where each of the first three deconvolution layers is followed by a batch normalization layer and the ReLU (Agarap, 2018) activation function, and the last deconvolution layer is directly followed by the Tanh function to generate images. The discriminators consist of 3 convolution layers (kernel size 5, stride 2) respectively with a batch normalization layer and a LeakyReLU (Xu et al., 2015) activation function.

We also adopt the parameter sharing techniques in ProbGAN to implement the Bayesian discriminators and generators. Only the first few layers are disjoint for generators, while all parameters in the following layers are shared. For discriminators, only the last few layers are disjoint, while the remaining parameters are shared. Previous research shows such a parameter-sharing scheme can improve the performance and stability of probabilistic models (Saatchi & Wilson, 2017).

It is worth noting that F2A (Yonetani et al., 2019) is not designed for federated learning; therefore, the implementation details for the federated learning setting are lacking. We refer to the training configurations of FedAvg-GAN (Augenstein et al., 2019), which adopts a similar training strategy (only federates discriminators from clients and trains the generator centrally at the server) with F2A but aims to address the federated learning setting (though only i.i.d. data is evaluated in the paper), to train F2A, noted as F2A-fed.

In EP-ProbGAN and Mixture-EP-ProbGAN, we implement the likelihood approximation function $f(\mathbf{x})$, as mentioned in Eq. (5) as a neural network with the backbone and the output module, where the backbone is shared with the discriminator and the output module is a two-layer NN with hidden size of 128. The neural network is trained with FedAvg. For J components of the mixture of EP-priors, the output dimension of $f(\mathbf{x})$ is J, where each dimension relates to one component in the mixture of EP-priors.

Hyper-parameter Selection. For baselines (1)-(3) and our EP-GANs, we conduct experiments on four models suited for the federated learning algorithm `FedAvg` with 25 turns of aggregations in total. Every client performs local training for 400 local updates between two consecutive turns of model aggregations. As baselines (5)-(6) do not support large local updates, we follows the training

[2]https://github.com/hehaodele/ProbGAN

configuration of (6), which conducts local training steps of 6 between two consecutive federated turns. The federated turns of (5)-(6) are set as 1667 ($\approx \frac{400\times25}{6}$). Therefore, the total number of training updates is 10000 for all baselines and EP-GANs, which is identical to the total training epochs in centralized training baseline. For all models, we use the Adam optimizer (Kingma & Ba, 2014) with the learning rate of 0.0003, which is commonly used and achieves stable performances. For SGHMC, the noise factor is set to 3×10^{-2}. The batch size is set as 128. All models are trained with 10 generators and 4 discriminators. For EP-ProbGAN and Mixture-EP-ProbGAN, the EP-prior loss factor is set to 1×10^{-3}, which is tuned from $\{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$ on the i.i.d. setting. The number of components in the mixture of EP-priors is set as 10.

F.3.3 Evaluation Metrics

We quantitatively evaluate the methods with two image generation metrics: Inception Score (IS) (Salimans et al., 2016) and Fréchet Inception Distance (FID) (Heusel et al., 2017). IS computes $\exp\left(\mathbb{E}_{\mathbf{x}}[KL(p(y \mid \mathbf{x}) \| p(y))]\right)$ where $p(y|\mathbf{x})$ is the predicted label distribution by a pre-trained Inception model (Szegedy et al., 2015), and $p(y)$ is the average of $p(y|\mathbf{x})$ over all images in the dataset. IS aims to match the fidelity and diversity of images with human-like judgment (Salimans et al., 2016). On the other hand, FID is used to measure the similarity between real images and synthetic images by computing the Fréchet distance between the two image distributions in the feature embedding space given by the Inception model. For IS, higher values indicate better performance; for FID, lower values indicate better performance.

F.3.4 T-test p-values on IS and FID of EP-GANs for CIFAR10

To further elaborate the improvement, we performed a statistical significance test on Inception Scores and FID Scores of EP-BGAN, EP-ProbGAN, and Mixture-EP-ProbGAN for CIFAR10. Results are listed in Table 4-5, from which we can see that the improvements are statistically significant.

Table 4: T-test p-values on Inception Scores for CIFAR10

W-objective LS-objective	EP-BGAN	EP-ProbGAN	Mixture-EP-ProbGAN
EP-BGAN	-	2.05×10^{-3} 3.39×10^{-6}	1.30×10^{-3} 1.77×10^{-8}
EP-ProbGAN	2.05×10^{-3} 3.39×10^{-6}	-	1.34×10^{-2} 6.97×10^{-3}
Mixture-EP-ProbGAN	1.30×10^{-3} 1.77×10^{-8}	1.34×10^{-2} 6.97×10^{-3}	-

Table 5: T-test p-values on FID Scores for CIFAR10

W-objective LS-objective	EP-BGAN	EP-ProbGAN	Mixture-EP-ProbGAN
EP-BGAN	-	1.82×10^{-3} 1.71×10^{-8}	2.18×10^{-4} 1.73×10^{-6}
EP-ProbGAN	1.82×10^{-3} 1.71×10^{-8}	-	2.49×10^{-2} 1.69×10^{-7}
Mixture-EP-ProbGAN	2.18×10^{-4} 1.73×10^{-6}	2.49×10^{-2} 1.69×10^{-7}	-

F.3.5 Example CIFAR-10 Generated Images

This section displays randomly selected generated images by different models in different settings. Fig. 3-10 show that our proposed EP-GAN successfully alleviates the performance deterioration under the extremely non-i.i.d setting, though their performances are still worse than the i.i.d. settings.

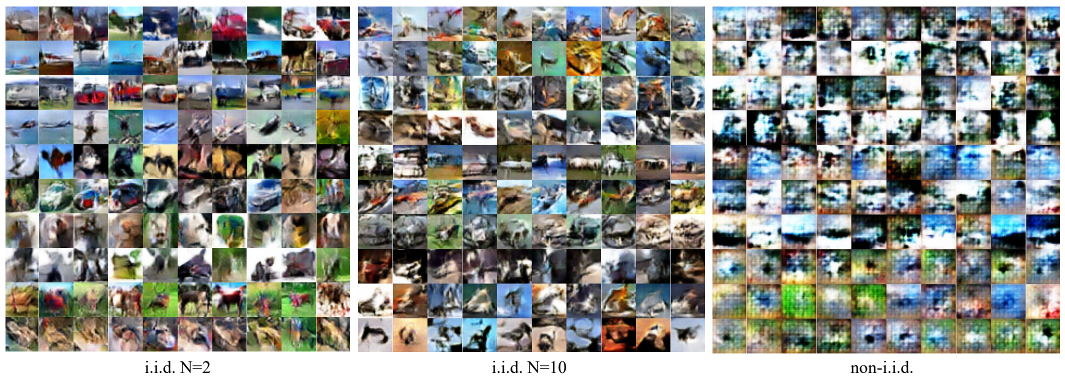


Figure 3: Images generated by **ProbGAN** trained on CIFAR-10 with the WGAN objective under different settings. Images in different rows are generated by different generators. ProbGAN with WGAN objective fails to converge under the extremely non-i.i.d setting (right).

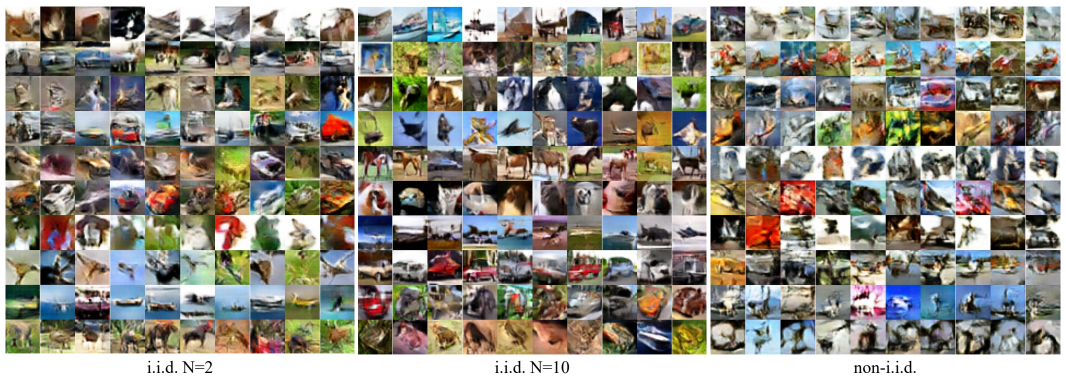


Figure 4: Images generated by **EP-BGAN** trained on CIFAR-10 with the WGAN objective under different settings. Images in different rows are generated by different generators.

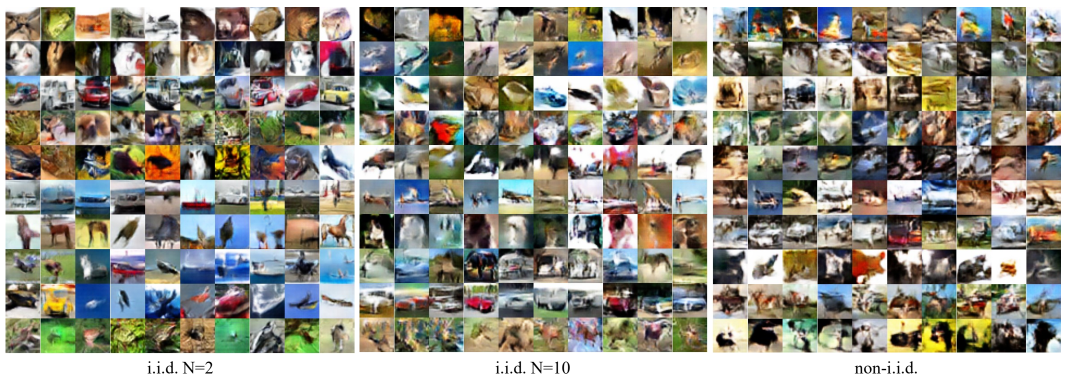


Figure 5: Images generated by **EP-ProbGAN** trained on CIFAR-10 with the WGAN objective under different settings. Images in different rows are generated by different generators.

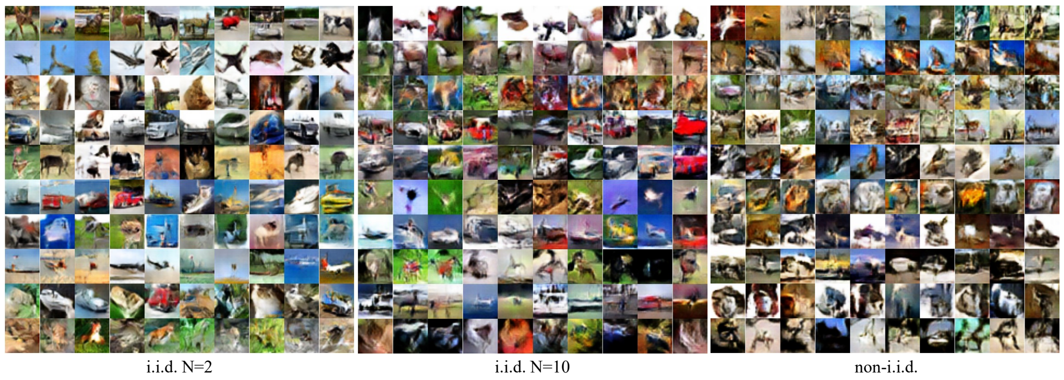


Figure 6: Images generated by **Mixture-EP-ProbGAN** trained on CIFAR-10 with the WGAN objective under different settings. Images in different rows are generated by different generators.

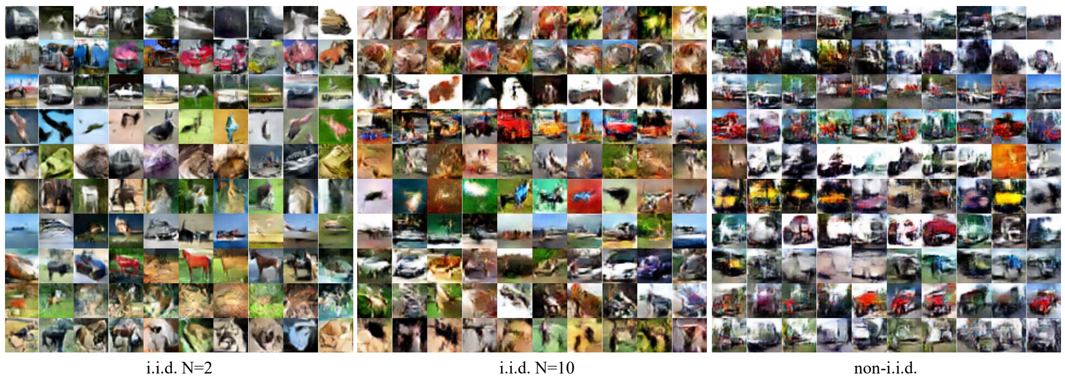


Figure 7: Images generated by **ProbGAN** trained on CIFAR-10 with the LSGAN objective under different settings. Images in different rows are generated by different generators. We can observe collapsed generators under the extremely non-i.i.d setting (right).

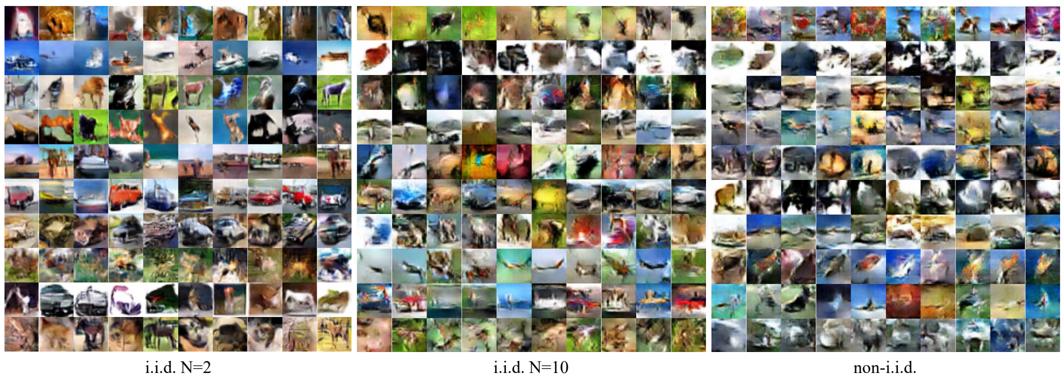


Figure 8: Images generated by **EP-BGAN** trained on CIFAR-10 with the LSGAN objective under different settings. Images in different rows are generated by different generators.

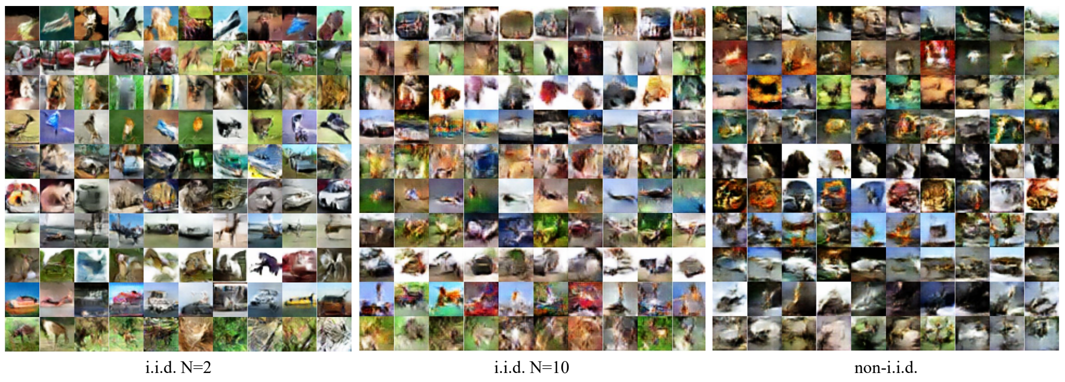


Figure 9: Images generated by **EP-ProbGAN** trained on CIFAR-10 with the LSGAN objective under different settings. Images in different rows are generated by different generators.

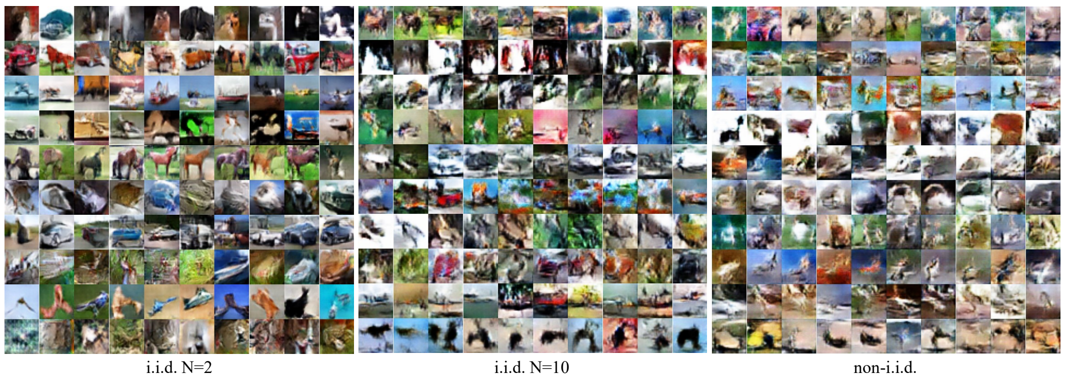


Figure 10: Images generated by **Mixture-EP-ProbGAN** trained on CIFAR-10 with the LSGAN objective under different settings. Images in different rows are generated by different generators.

F.4 EP-GAN with SRU for non-i.i.d ASR

F.4.1 Detailed Composition of Datasets

CHiME-5 is a large-scale corpus of real-world multi-speaker conversational speech in home environments. The training dataset, development dataset, and test dataset include about 40 hours, 4 hours, and 5 hours of conversational speech. Table 6 shows the splits of training, development, and evaluation sets. In each session, there are four speakers with around 130-180 minutes of conversation records. In the non-i.i.d setting, we treat each session as the cross-silo data in each client, creating a natural non-i.i.d dataset. For example, the speakers in clients 1, 2, 7, and 8 are all male, and sessions are recorded in different environments.

Table 6: The data have been split into training, development, and evaluation set as follows.

Dataset	Sessions	Speakers	Hours	Utterances
Train	16	32	40:33’	79,980
Dev	2	8	4:27’	7,440
Eval	2	8	5:12’	11,028

F.4.2 Implementation Details

We adopt the EP-GAN and ProbGAN to augment acoustic modeling by joint training them with SRU-HMM based acoustic model (Lei et al., 2017). SRU is a popular and efficient recurrent neural network for acoustic modeling. Figure 11 displays the model architectures of our the SRU acoustic model and SRU (EP) w/ ProbGAN.

In the implementation, SRU is used as a backbone of the acoustic model, which contains 12 stacked layers with 1280 hidden nodes. We use the similar SRU architecture for the encoders of discriminator in GANs, whose outputs are then fed into a two-layer MLP for discriminating its true/fake label. Moreover, we take bottleneck features from the bottleneck layer of the discriminator (Shown in Figure 11), which are further concatenated with the acoustic feature, feeding to the SRU acoustic model for supervised learning.

The implementation of SRU w/ EP-ProbGAN is based on SRU w/ ProbGAN, additional with EP-prior GMA inference on parameters of GANs, as mentioned in Eq. (45). For all EP-GANs, the function f is implemented as a neural network with a backbone and a output module, where the backbone is shared with the discriminator and the output module is a two-layer NN with hidden size of 128. The neural network is trained with FedAvg.

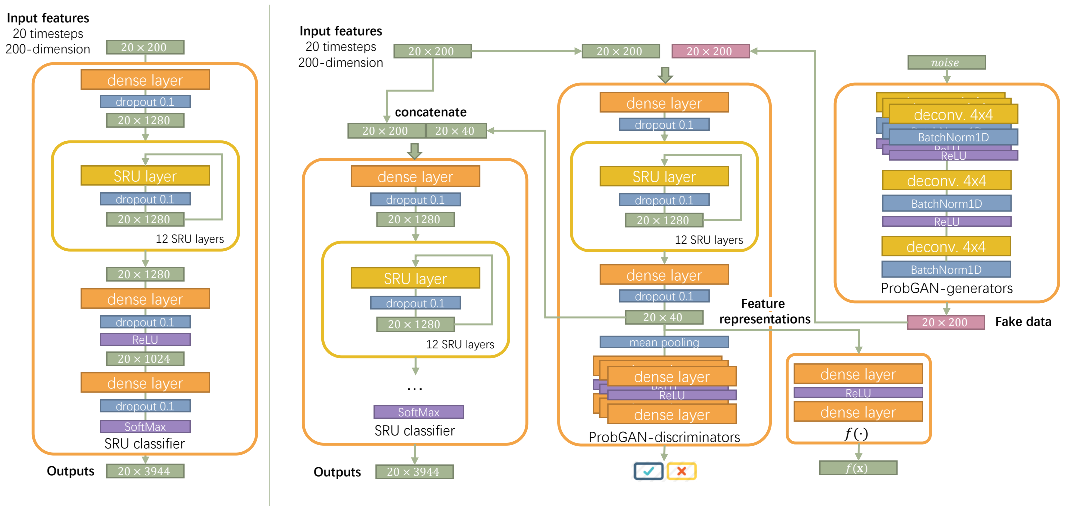
In SRU w/ EP-ProbGAN and SRU (EP) w/ EP-ProbGAN, we introduce a two-layer MLP with the hidden size of 64 so that we can map the feature representation captured by the discriminator to the likelihood estimation $f(\mathbf{x})$ (mentioned in Eq. (5)). For SRU (EP) w/ EP-ProbGAN, we regularize the parameters of the SRU classifier with EP-prior GMA, regarding the SRU classifier’s parameters as an SGHMC sample from a parameter distribution.

F.4.3 Training Details

We adopt the same configuration with (Huang et al., 2020) to train all GMM-HMM. The speech data is preprocessed as 40-dimensional Mel-filter bank coefficients (Biem et al., 2001), which are calculated every 10ms. Inputs of all models consist of the current frame together with its 4 future contextual frames. The input sequence are chunked into a fixed length of 20. We performed speaker-level mean and variance normalization on the inputs. The HMM states aligned by GMM-HMM are used to train the subsequent neural network modules.

The evaluation is performed with a tri-gram language model trained from the transcription of CHiME-5. The models are optimized with the categorical cross-entropy loss using BPTT with a dropout rate of 0.1 between the recurrent layers. For federated learning, there are 12 turns of aggregation with 1 local training epoch between each turn.

For simplicity, we denote the categorical cross-entropy loss as $\mathcal{L}_{ce}(\boldsymbol{\theta}_{SRU})$, whose target is to minimize the cross-entropy between SRU predictions and the labels, given the input sequences. Denote the ProbGAN’s objective (He et al., 2019) as $\mathcal{L}_{gan}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$. We denote EP-prior losses (Eq. (46)) for parameters of generators, discriminators and the SRU as $\mathcal{L}_{ep}(\boldsymbol{\theta}_G)$, $\mathcal{L}_{ep}(\boldsymbol{\theta}_D)$, and



(a) SRU

(b) SRU (EP) w/ EP-ProbGAN

Figure 11: The model architectures of SRU (left) and SRU (EP) w/ EP-ProbGAN (right). The overlapped modules represent the Bayesian neural network samples. SRU w/ ProbGAN and SRU w/ EP-ProbGAN share similar architecture with SRU (EP) w/ EP-ProbGAN.

$\mathcal{L}_{ep}(\boldsymbol{\theta}_{SRU})$, respectively. The loss functions of SRU w/ ProbGAN, SRU w/ EP-ProbGAN, and SRU (EP) w/ EP-ProbGAN are then described as following:

Model	Loss function
SRU w/ ProbGAN	$\mathcal{L}_{ce}(\boldsymbol{\theta}_{SRU}) + \lambda_{gan} \cdot \mathcal{L}_{gan}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$
SRU w/ EP-ProbGAN	$\mathcal{L}_{ce}(\boldsymbol{\theta}_{SRU}) + \lambda_{gan} \cdot [\mathcal{L}_{gan}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D) + \lambda_{ep} \cdot (\mathcal{L}_{ep}(\boldsymbol{\theta}_G) + \mathcal{L}_{ep}(\boldsymbol{\theta}_D))]$
SRU (EP) w/ EP-ProbGAN	$\mathcal{L}_{ce}(\boldsymbol{\theta}_{SRU}) + \lambda_{ep} \cdot \mathcal{L}_{ep}(\boldsymbol{\theta}_{SRU})$ $+\lambda_{gan} \cdot [\mathcal{L}_{gan}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D) + \lambda_{ep} \cdot (\mathcal{L}_{ep}(\boldsymbol{\theta}_G) + \mathcal{L}_{ep}(\boldsymbol{\theta}_D))]$

The batch size is set to 128. The optimizer is Adam (Kingma & Ba, 2014) with a learning rate of 3×10^{-4}. The SGHMC noise factor in Algorithm 1 is set to 3×10^{-2}, and the EP-prior loss factor (λ_{ep}) is set to 1×10^{-3}. Moreover, the loss of training the GAN objective is multiplied by a factor λ_{gan}, which is set as 0.1. When training SRU w/ ProbGAN, SRU w/ EP-ProbGAN, and SRU (EP) w/ EP-ProbGAN, we first update the GANs and then update the SRU acoustic model together with the discriminator in each iteration.

F.4.4 STABILITY STUDY

To examine the stability of proposed methods, we train the models using three different random seeds. Table 7 reports the mean and standard deviation (STD) of the test WERs, indicating that the variance of three different runs is moderate.

Table 7: We train SRU w/ ProbGAN, SRU w/ EP-ProbGAN, and SRU (EP) w/ EP-ProbGAN using three different random seeds and report the mean and standard deviation of the test WERs.

Model	SRU w/ ProbGAN	SRU w/ EP-ProbGAN	SRU (EP) w/ EP-ProbGAN
Mean ± STD	66.86 ± 0.1105	66.20 ± 0.2795	64.30 ± 0.2130