A Proofs

A.1 Supplementary Lemma for Proofs

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

Lemma 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*

$$E[g(C)(C - C_{m,-k})] = C_{s,-k}E[g'(C)] \tag{21}$$

A.2 Proof of Theorem 1

Theorem 1. *Suppose we are given a data partition $\mathbf{X}_k$ located at the k-th party during FL. Assume data partitions $\{\mathbf{X}_k\}_{k=1}^K$ are conditionally independent given a latent variable $\mathbf{z}$. Let $f : \mathcal{R}^{|\mathbf{X}_k|} \to \mathcal{R}^1$ be a neural network taking as input $\mathbf{X}_k$. Let $C = f(\mathbf{X}_k)\mathbf{z}$ and $C \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$. Let $q_{-k}(\mathbf{z})$ and $h_k(\mathbf{z})$ be the cavity distribution (defined in Eq. (6)) and the hybrid distribution (defined in Eq. (7)), respectively. We further define $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}_m, \mathbf{z}_s)$ as in Eq. (5). There exists a function $\phi : \mathcal{R}^{|\mathbf{z}|} \times \mathcal{R}^{|\mathbf{X}_k|} \to \mathcal{R}^1$, such that the update rules of $\mathbf{z}_m$ and $\mathbf{z}_s$ can be written in closed form as:*

$$\mathbf{z}_m = S_1, \tag{22}$$

$$\mathbf{z}_s = S_2 - S_1^2, \tag{23}$$

where

$$S_1 = \left[(C_{m,-k}+C_{s,-k})E_C(\sigma(C)) - C_{s,-k}E_C(\sigma^2(C))\right]/S_0 f(\mathbf{X}_k), \tag{24}$$

$$S_2 = \left[(C_{m,-k}+2C_{s,-k})E_C(\sigma^2(C)) - 2C_{s,-k}E_C(\sigma^3(C))\right]/S_0 f^2(\mathbf{X}_k), \tag{25}$$

$$S_0 = E_C(\sigma(C)). \tag{26}$$

Proof. With $C \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$ and Theorem 2, we have that the first three moments of $\sigma(C)$ can be expressed in closed form, e.g.:

$$E_C(\sigma(C)) \approx \sigma\left(\frac{(C_{m,-k})}{\sqrt{1+\zeta^2 C_{s,-k}}}\right); \tag{27}$$

$$E(\sigma^2(C)) \approx \sigma\left(\frac{a\,(C_{m,-k}+b)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right), \tag{28}$$

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

$$E(\sigma_C^3(C)) \approx \sigma\left(\frac{a\,(C_{m,-k}+b)}{\sqrt{1+\zeta^2 a^2 C_{s,-k}}}\right), \tag{29}$$

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

Let $q(\mathbf{z}|\mathbf{X}_k) = \phi(\mathbf{z}, \mathbf{X}_k) = \sigma(f(\mathbf{X}_k)\mathbf{z})$, where σ denotes Sigmoid activation function. Let the normalizer $S_0 = \int h_k(\mathbf{z})\mathrm{d}\mathbf{z}$. With Eq. (27), we have:

$$\begin{aligned} S_0 &= \int \sigma\left(f\left(\mathbf{X}_k\right)\mathbf{z}\right) q_{-k}(\mathbf{z})\mathrm{d}\mathbf{z} \\ &= E_C(\sigma(C)) \\ &\approx \sigma\left(\frac{(C_{m,-k})}{\sqrt{1+\zeta^2 C_{s,-k}}}\right) \end{aligned} \tag{30}$$

Let the first moment of $\mathbf{z}$, $S_1 = \int \mathbf{z}h_k(\mathbf{z})\mathrm{d}\mathbf{z} / \int h_k(\mathbf{z})\mathrm{d}\mathbf{z}$. We have:

$$S_1 = \frac{\int f(\mathbf{X}_k)\mathbf{z}\sigma\left(f\left(\mathbf{X}_k\right)\mathbf{z}\right) q_{-k}(\mathbf{z})\mathrm{d}\mathbf{z}}{f(\mathbf{X}_k)S_0} = \frac{E_C\left(C \cdot \sigma(C)\right)}{f(\mathbf{X}_k)S_0} \tag{31}$$

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

$$E_C\left(\sigma(C) \cdot (C - C_{m,-k})\right) = C_{s,-k}E_C\left(\sigma'(C)\right) \tag{32}$$

Then we have:

$$\begin{aligned} E_C(\sigma(C) \cdot C) = &\left(C_{s,-k} + C_{m,-k}\right) E_C(\sigma(C)) \\ &- C_{s,-k}E_C(\sigma^2(C)), \end{aligned} \tag{33}$$

With Eq. (31) and Eq. (33), we therefore have:

$$S_1 = \frac{\left[\left(C_{m,-k} + C_{s,-k}\right) E(\sigma(C)) - C_{s,-k}E(\sigma^2(C))\right]}{S_0 f(\mathbf{X}_k)}. \tag{34}$$

Combining Eq. (27), Eq. (28), Eq. (30) and Eq. (34) , we have the first moment S_1 being expressed in closed form.

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

$$S_2 = \frac{\int f^2(\mathbf{X}_k)\mathbf{z}^2\sigma\left(f\left(\mathbf{X}_k\right)\mathbf{z}\right) q_{-k}(\mathbf{z})\mathrm{d}\mathbf{z}}{f^2(\mathbf{X}_k)S_0} = \frac{E_C\left(C^2 \cdot \sigma(C)\right)}{f^2(\mathbf{X}_k)S_0} \tag{35}$$

With Lemma 1, we have

$$\begin{aligned} &E_C\left(\sigma(C) \cdot C \cdot (C - C_{m,-k})\right) = \\ &C_{s,-k}E_C\left(\sigma(C) + \left(\sigma(C) - \sigma^2(C)\right) \cdot C\right) \end{aligned} \tag{36}$$

Then we have:

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

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

$$E_C\left(\sigma^2(C)(C - C_{m,-k})\right) = C_{s,-k}E_C\left(2\sigma^2(C) - 2\sigma^3(C)\right) \tag{38}$$

Then

$$\begin{aligned} E_C\left(\sigma^2(C) \cdot C\right) = &\left(C_{m,-k} + 2C_{s,-k}\right) E_C(\sigma^2(C)) \\ &- 2C_{s,-k}E_C(\sigma^3(C)) \end{aligned} \tag{39}$$

With Eq. (35), Eq. (37) and Eq. (39), we have:

$$S_2 = \frac{\left[\left(C_{m,-k} + 2C_{s,-k}\right) E(\sigma^2(C)) - 2C_{s,-k} E(\sigma^3(C))\right]}{S_0 f^2(\mathbf{X}_k)}. \tag{40}$$

Combining Eq. (28)-(30) and Eq. (40), we have the second moment S_2 being expressed in closed form as well.

With first two moments of $\mathbf{z}$, S_1 and S_2, being expressed in closed form, we have the approximated posterior of global data distribution $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}_m, \mathbf{z}_s)$, whose parameters $\mathbf{z}_m$ and $\mathbf{z}_s$ can be updated in close-form:

$$\mathbf{z}_m = S_1, \tag{41}$$

$$\mathbf{z}_s = S_2 - S_1^2, \tag{42}$$

□

A.3 Proof of Theorem 2

Theorem 2. *Suppose $C \sim \mathcal{N}(C_{m,-k}, C_{s,-k})$. 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:*

$$E_C(\sigma^d(C)) \approx \sigma\left(\frac{a\left(C_{m,-k} + b\right)}{\sqrt{1 + \zeta^2 a^2 C_{s,-k}}}\right), \tag{43}$$

Proof. Let a, b be two real constants. Taking the probit funtion $\Phi\left(\zeta a(C + b)\right)$ 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

$$\begin{aligned} a &= 2d(1 - 2^{-1/d}) \\ b &= \log(2^{1/d} - 1) \end{aligned} \tag{44}$$

Let $d = 1$, we have:

$$\sigma(C) \approx \Phi(\zeta C) \tag{45}$$

With Theorem 3 (Wang, Shi, and Yeung 2016) and Eq. (45), we have:

$$\begin{aligned} E_C(\sigma^d(C)) &= \int \sigma^d(C) \mathcal{N}\left(C_{m,-k}, C_{s,-k}\right) \mathrm{d}C \\ &\approx \int \Phi(\zeta a(x + b) \mathcal{N}\left(C_{m,-k}, C_{s,-k}\right) \mathrm{d}C \\ &= \Phi\left(\frac{\zeta a\left(C_{m,-k} + b\right)}{\sqrt{1 + \zeta^2 a^2 C_{s,-k}}}\right) \\ &\approx \sigma\left(\frac{a\left(C_{m,-k} + b\right)}{\sqrt{1 + \zeta^2 a^2 C_{s,-k}}}\right) \end{aligned} \tag{46}$$

Combining Eq. (44) and Eq. (46) concludes the proof. □

B The FedNP Algorithm

The FedNP algorithm (see Algorithm 1) runs on a federation consisting of a central server and K clients, where the k-th client has n_k training examples. Each client conducts T local updating iterations. Between every L updates, all clients send updated parameters to the server; the server aggregates the received model parameters with weighted averaging (McMahan et al. 2017a) and computes the updated cavities parameters with Product-of-Gaussian (Airey and Gales 2003).

Practically, to further restrict the complexity of the approximate global model distribution, we extend the objective function (19) with an additional loss, i.e., the Kullback–Leibler divergence between the approximate global data distribution and a Gaussian prior p($\mathbf{z}$):

$$\tilde{\ell} = KL\left[p(\boldsymbol{\theta}_k|\mathbf{z}) \| p(\hat{\boldsymbol{\theta}}_k)\right] + KL\left[q(\mathbf{z}) \| p(\mathbf{z})\right] \tag{47}$$

C Related Work: EP with Neural Networks

To efficiently model the global data distribution from partitioned data, we follow the spirit of Expectation Propagation (EP). EP is one of the most popular Bayesian inference methods (Minka 2013), which approximates the posterior with approximate factors that are iteratively updated via moment matching. However, the moment matching can be intractable if the likelihood term has a complex form, which results in the intractability of the moments of the hybrid distribution. An intuitive solution is to approximate the likelihood term by numerical quadrature (Jylänki, Nummenmaa, and Vehtari 2014; Soudry, Hubara, and Meir 2014), but it fails to scale to large datasets or complex neural networks. Along a different line of research, (Heess, Tarlow, and Winn 2013) achieves this by incorporating neural networks to map EP message inputs to EP message outputs.Zhao et al. (2020) alleviate this problem by considering the EP's KL divergence between the target distribution and a mixture of exponential family approximate factors as the objective function. The closest work to ours is (Bui et al. 2018), which adopts variational inference (VI) to simulate EP on federated settings with synthetic data, but their EP update requires performing gradient descent with a deliberately designed KL loss. Furthermore, their method still relies on the likelihood term, which is not applicable to our federated setting. In contrast, we reformulate EP to remove the dependence on the intractable likelihood term and propose a closed-form solution for moment matching without requiring numerical approximation and more efficient than estimating through gradient descent.

To confirm the significance of the contribution in terms of EP with neural networks, we give a comparison of related works with our FedNP in Table 3.

D Experiment Details

D.1 Experimental Environments

We run all experiments on a machine with 314 GB RAM, an Intel Xeon PHI 7290 CPU Processor, and two Tesla

Table 3: The comparison of supported features of representative methods

Representative method	Sampling-free EP update	Closed-form EP update approximation	Numerical approximation/quadrature free
Jylänki, Nummenmaa, and Vehtari (2014)	✓	✓	✗
Bui et al. (2018)	✗	✗	✗
Heess, Tarlow, and Winn (2013)	✗	✗	✓
Zhao et al. (2020)	✗	✗	✗
Ours	✓	✓	✓

V100 GPUs. The operating system is CentOS 7. For detailed versions of software environments, please refer to the `README.md` files in corresponding code projects. Note that the datasets used in our experiments are public, and codes can be found in https://anonymous.4open.science/r/FedNP-Neurips/ or the supplementary material.

D.2 Details of Toy Experiment

Experimental Setup We use the RMSprop optimizer with a learning rate of 0.01 for all approaches. The batch size is set to 10. The number of local epochs is set to 1. The total number of communication rounds is 10.

Quantitative Results Table 4 shows the Mean Squared Error (MSE) of FedNP and baselines under the above setting. The MSE is evaluated on global training data. The local training of the baselines leads the model to fit their data points perfectly, failing to generalize well to the unseen global data, thus having a higher value of MSE. In contrast, our FedNP is more robust since we correct the local training and avoids performance deterioration.

Table 4: MSE of FedNP and the baselines. We run three trials and report the means and standard deviations accordingly.

Method	MSE ↓
FedAvg (McMahan et al. 2017a)	116.48 ± 2.54
FedProx (Li et al. 2020)	117.11 ± 2.96
SCAFFOLD (Karimireddy et al. 2020)	100.48 ± 3.24
FedPA (Al-Shedivat et al. 2020)	97.85 ± 3.44
FedNP (ours)	**72.42** ± 2.39

Case Study of FedPA and FedNP Figure 4 shows the comparison of FedPA and FedNP. The performance of FedPA is slightly better than FedAvg, but it is also biased towards local data and failed to capture the global information, as opposed to our FedNP.

D.3 FedNP for Image Classification on Extremely Non-IID Image Datasets

Dataset Preparation. CIFAR-100 (Krizhevsky, Nair, and Hinton 1995) (60,000 images with 100 classes) and Tiny-Imagenet (Le and Yang 2015) (100,000 images with 200 classes) datasets are with the MIT-License, so they are publicly available.

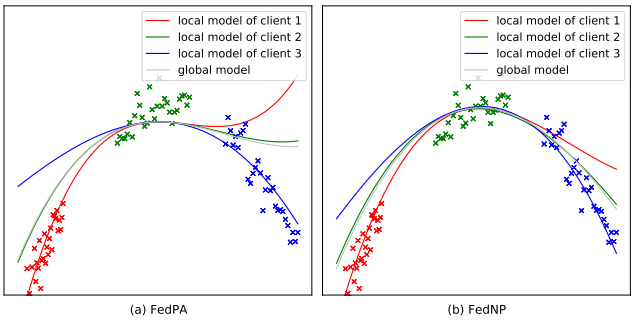

Figure 4: Toy example on polynomial curve fitting task. Data points are denoted by '×' and models are denoted by '—'. (a) The local models of three clients and the global model trained by FedAvg. (b) The local models of three clients and the global model trained by FedNP.

Similar to (Li et al. 2021), we sample $p_k \sim Dir_N(\beta)$ and allocate a $p_{k,j}$ proportion of the images of class k to client j, where $Dir(\beta)$ is the Dirichlet distribution with a concentration parameter β (0.5 by default). With the above partitioning strategy, the data distribution of in each partition is extremely non-i.i.d, i.e. each client may have relatively few data samples in some classes.

Table 5: The top-1 accuracy of FedNP and the other baselines on CIFAR-100 dataset with varying numbers of clients. We run three trials and report the mean and standard deviation.

Methods	50 clients	100 clients
FedAvg (McMahan et al. 2017a)	$61.32\% \pm 0.3\%$	$50.78\% \pm 0.3\%$
FedProx (Li et al. 2020)	$60.65\% \pm 0.2\%$	$48.90\% \pm 0.3\%$
MOON (Li, He, and Song 2021)	$62.13\% \pm 0.3\%$	$52.88\% \pm 0.3\%$
SCAFFOLD (Karimireddy et al. 2020)	$50.91\% \pm 0.3\%$	$46.28\% \pm 0.2\%$
FedPA (Al-Shedivat et al. 2020)	$61.77\% \pm 0.3\%$	$51.51\% \pm 0.2\%$
FedLA (Liu et al. 2021)	$57.39\% \pm 0.3\%$	$47.34\% \pm 0.2\%$
FedNP (ours)	**63.74%** $\pm 0.3\%$	**53.37%** $\pm 0.3\%$

Algorithm 1: The FedNP algorithm that is conducted on the server and K clients. Notably, the operations below on vectors or tensors are element-wise.

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.

Server initializes model parameters $\boldsymbol{\theta}^{(0)}$, cavities $\left\{\left(\mathbf{z}_{m,-k}^{(0)}, \mathbf{z}_{s,-k}^{(0)}\right)\right\}_{k=1...K}$ (initialized as standard normal distribution) and broadcasts to all clients.

for $t = 0, ..., T-1$ **do**

  for Client $k = 1, ..., K$ (parallelly) **do**

    Update the parameters of approximated global data distribution $(\mathbf{z}_m^{(t)}, \mathbf{z}_s^{(t)})$ according to Eq. (9) and Eq. (10).

    Optimize the local model parameters $\boldsymbol{\theta}_k^{(t)}$ with the extended objective 19.

    if $(t > 0$ and $(t-1) \mod L == 0)$ or $(t == T-1)$ **then**

      Compute the approximated posterior factor $q_k(\mathbf{z})$ with the approximation:

$$\mathbf{z}_{s,k}^{(t)} = \left(\mathbf{z}_s^{(t)^{-1}} - \mathbf{z}_{s,-k}^{(t)^{-1}}\right)^{-1},$$

$$\mathbf{z}_{m,k}^{(t)} = \mathbf{z}_{s,k}^{(t)} \left[\frac{\mathbf{z}_m^{(t)}}{\mathbf{z}_s^{(t)}} - \frac{\mathbf{z}_{m,-k}^{(t)}}{\mathbf{z}_{s,-k}^{(t))}}\right].$$

      Upload $\left(\mathbf{z}_{m,k}^{(t)}, \mathbf{z}_{s,k}^{(t)}\right)$ and $\boldsymbol{\theta}_k^{(t)}$ to server.

      Server updates the global model parameters by weighted averaging:

$$\boldsymbol{\theta}_{avg}^{(t+1)} = \sum_{k=1}^{K} \frac{1}{n_k} \boldsymbol{\theta}_k^{(t)}.$$

      Server updates the approximated cavities for each client k with the mixture as Product-of-Gaussian (Airey and Gales 2003)

$$\mathbf{z}_{s,-k}^{(t+1)} = \left(\sum_{j \neq k}^{K} \left(\mathbf{z}_{s,j}^{(t)}\right)^{-1}\right)^{-1},$$

$$\mathbf{z}_{m,-k}^{(t+1)} = \mathbf{z}_{s,-k}^{(t+1)} \sum_{j \neq k}^{K} \frac{\mathbf{z}_{m,j}^{(t)}}{\mathbf{z}_{s,j}^{(t)}},$$

      The server broadcasts $\boldsymbol{\theta}_{avg}^{(t+1)}$ and $\left(\mathbf{z}_{s,-k}^{(t+1)}, \mathbf{z}_{m,-k}^{(t+1)}\right)$ to the k-th client.

    end if

  end for

end for

Model Architectures. We use ResNet18 (He et al. 2016) as the model architecture for all approaches. For FedNP, we implement the auxiliary neural network f mentioned in Section 4.4 by stacking a linear layer after the ResNet18 backbone, which takes the input X_k, and the output size is 10. We then apply mean pooling to the output vector to achieve $f(\mathbf{X}_k)$ with dimension 1. The non-linear function ϕ (Eq. 8) is implemented as $\phi = \sigma(f(\mathbf{X}_k)\mathbf{z})$, where σ is the sigmoid function.

The dimension of $\mathbf{z}$ is set to 10. We then use a NPN with one hidden layer to obtain $p(z|\mathbf{X}_k)$. For simplicity, we only apply our FedNP to the parameter of the classifier instead of the whole model, so the dimension of $z_{m,k}$ and $z_{s,k}$ is equal to that of the classifier (i.e., the last MLP layer of ResNet18). For MOON (Li, He, and Song 2021), we use the input of the last MLP layer as the feature of the image.

Training Protocol. The number of clients and the dimension of $\mathbf{z}$ are set to 10. We use the SGD optimizer with a learning rate of 0.01 for all approaches. The batch size is set to 128. The number of local epochs is set to 10. The total number of communication rounds is 100. We tune λ from $\{0.001, 0.01, 0.1, 1\}$ and report the best result. The best λ of FedNP is 0.01. Note that FedProx and MOON also have a hyper-parameter λ to control the weight of its proximal term. We tune λ from $\{0.001, 0.01, 0.1, 1, 10\}$, the best λ is 0.01 for FedProx and 1 for MOON.

Results of 50-, and 100-Client Settings on CIFAR-100. We conduct experiments on the CIFAR-100 dataset with varying numbers of clients. As the number of clients increases to 100, the performances of all methods drop significantly due to the sparsity of local training data. Our method has consistent advantages over baselines. More specifically, when the number of clients is 100, the local data is highly label-skew, i.e., some of the classes are not seen in the local client. Therefore, the local models severely collapse, resulting in unstable aggregation. In this case, the methods with explicit local model regularization, MOON and FedNP perform better. Moreover, FedNP outperforms MOON as its local models are regularized by the estimated global model distribution, which might be more accurate than MOON's regularizer, i.e., a contrastive loss with the previous global model.

Analysis on Predicted Model Distributions by FedNP. We visualize the predicted model distribution by FedNP in Figure 5. For the convenience of visualization, we focus on parameters of the classifier of the model, denoted by a matrix $\mathbf{C} \in \mathcal{R}^{d \times c}$, where c equals the number of classes and d is the dimension of the features. More specifically, the matrix $\mathbf{C}$ comprises c vectors with the dimension of d, and the j^{th} vector can be regarded as the centroid of the j^{th} class in the literature of metric learning (Liu et al. 2016). Figure 5 visualizes the predicted distribution of parameters (shown within three standard deviations of the mean), where the contour lines represent probability densities of predicted model distributions and the deeper color indicates the higher probability. As Figure 5 shows, the predicted model distribution coincides with the ground-truth data distribution (the solid points), which also verifies the effectiveness of our FedNP.

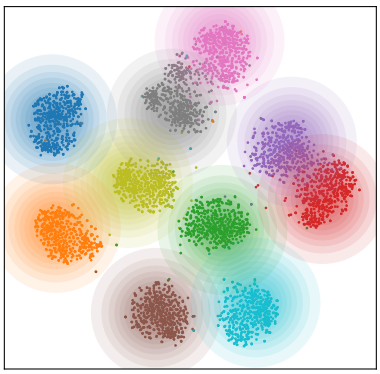

Figure 5: T-SNE visualizations of predicted distribution estimated by FedNP on CIFAR-100.

Experiments on Computational Efficiency and Approximation Accuracy of the Proposed Closed-form Update of FedNP. To evaluate the computational efficiency and approximation accuracy of the proposed closed-form update of FedNP, one may ask whether we can compare FedNP with existing EP algorithms. However, most of the EP-based algorithms are not applicable to our problem setting due to the need for $\boldsymbol{\theta}_k$, the local model parameter at current mini-batch, during training; and thus cannot directly compare with our FedNP. We, therefore, construct another variant of FedNP, dubbed Quad-FedNP, which adopt numerical-quadrature-based update (Jylänki, Nummenmaa, and Vehtari 2014) for calculation of moments of the hybrid distribution in FedNP. Quad-FedNP requires one-dimensional numerical quadratures implemented using TorchQuad (Gómez, Toftevaag, and Meoni 2021), which provides a very efficient numerical integration approximation toolkit optimized for graphics processing units (GPUs). Notably, numerical-quadrature-based update are usually adopted by numerical-quadrature-based EP (Jylänki, Nummenmaa, and Vehtari 2014) which is similar to MCMC-based EP (Barthelmé and Chopin 2014) in terms of computation of moments of the hybrid distribution.

We report the classification accuracy and the average running time per update of hybrid distribution for Quad-FedNP and our FedNP in Table 6. We can see that our FedNP runs much faster than Quad-FedNP with varied numbers of quadrature points. For Quad-FedNP, the image classification performance is improved by increasing the number of quadrature points. It's worth noting that the running time of Quad-FedNP does not grow linearly with quadrature points due to TorchQuad's GPU-based optimization. Generally, our FedNP runs 2.4 times faster than Quad-FedNP while achieving close evaluation scores (Acc.). The result demonstrates the efficacy and efficiency of our proposed closed-form update formulations.

Table 6: The accuracy and average running time per update on CIFAR-100 of Quad-FedNP with different numbers of quadrature points and our proposed FedNP.

Method	Acc.	Running Time
Quad-FedNP-100	62.92%	639ms
Quad-FedNP-1000	64.51%	650ms
Quad-FedNP-10000	65.07%	721ms
FedNP (ours)	65.03%	306ms

D.4 FedNP for Speech Recognition on Large-scale Realistic Non-IID Conversation Corpus

Detailed Configuration of CHiME-5. CHiME-5 is a large-scale corpus of real-world multi-speaker conversational speech in home environments. We holds a ChiME-5 non-commercial usage licence [1], which is issued by LPC. The training dataset, development dataset, and test dataset include about 40 hours, 4 hours, and 5 hours of conversational speech. Table 7 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 7: 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

Training Details. We adopt the same configuration with (Huang et al. 2020) to train all GMM-HMM. More specifically, we follow the GMM-HMMs ASR structure to train CHiME-5 using the adapted Kaldi s5b recipe (Povey et al. 2011) with single-channel audio data from GMM-HMM training.

The speech data is preprocessed as 40-dimensional Mel-filter bank coefficients (Biem et al. 2001), which are calculated every 10ms. Notably, only the audio data recorded by binaural microphones are employed to train and evaluate this experiment. Inputs of all models consist of the current frame and its 4 future contextual frames. The input sequence is 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 SRU model is trained to fit the mapping from acoustic features to HMM states. For FedNP, we feed SRU hidden state outputs

[1] https://chimechallenge.github.io/chime6/download.html

into a two-layer neural network, whose output size is the same as the dimension of $\mathbf{z}$, to utilize the feature extraction capability offered by the SRU fully. We then apply the mean pooling to the output vector to achieve $f(\mathbf{X}_k)$ with dimension 1. The non-linear function ϕ (Eq. 8) is implemented as $\phi = \sigma(f(\mathbf{X})\mathbf{z})$, where σ is the sigmoid function.

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. The batch size is set to 128. The optimizer is Adam (Kingma and Ba 2014) with a learning rate of 3×10^{-4}.

For FL, there are 12 turns of aggregation with 1 local training epoch between each turn. We tune λ from $\{0.0001, 0.001, 0.01, 0.1\}$ and report the best result. The best λ of FedNP is 0.001. For FedNP, the dimension of $\mathbf{z}$ is set as 16. For FedProx, which has a hyper-parameter μ to control the weight of its proximal term, we also tune it from $\{0.001, 0.01, 0.1, 1\}$, and the best λ is 0.01 for FedProx.