Heterogeneous Federated Learning with Scalable Server Mixture-of-Experts

Jingang Jiang[†] , **Yanzhao Chen**[†] , **Xiangyang Liu** , **Haiqi Jiang** and **Chenyou Fan**[*]
South China Normal University, Guangzhou, China
fanchenyou@scnu.edu.cn

Abstract

Classical Federated Learning (FL) faces challenges when deploying large models on power-constrained clients. We propose an asymmetric FL mechanism that enables the aggregation of compact client models into a comprehensive server Mixture-of-Experts (MoE), allowing for efficient fusion of the most pertinent client models to update each server expert based on the measured relevance. To address the Non-IID data issue, we optimize the server-side MoE architecture by incorporating a main expert that always activates alongside a set of selectively activated routed experts. This configuration ensures a balance between learning general knowledge and specific data distribution. Our Fed-MoE framework is model-agnostic and has demonstrated notable improvements on vision FL tasks with million-scale ResNet backbones, and language tasks with billion-scale BERT and GPT-2 backbones.

1 Introduction

Federated Learning (FL) [McMahan *et al.*, 2017] has become a widely adopted approach for distributed learning from diverse data sources while preserving data privacy. However, *classical FL requires the learning model to be identical across all clients* for possible parameter averaging. This requirement poses challenges in environments with constrained client-side capacities such as edge devices, rendering standard FL unsuitable for learning large language models (LLMs) and large visual models (LVMs) with many edge devices.

Recognizing this critical limitation, a question comes to us naturally: *can we deploy asymmetric models at client and server levels?* Given that clients typically possess relatively scarce data and limited computational capabilities, their models should be designed to be compact and efficient. Conversely, the server boasts substantial computational resources, enabling it to leverage significantly larger and more complex models. Thus, the second question arises: *how can we perform model averaging for asymmetric client and server models?*

We are seeking the answer from the recently explored Mixture-of-Experts (MoE) [Shazeer *et al.*, 2017] architecture. A MoE comprises of multiple expert sub-networks, each tailored to a specific segment of the input space. A learnable gate

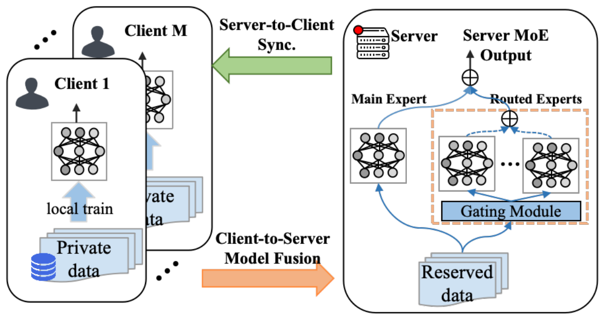


Figure 1: Overview of Fed-MoE. Compact client models federate into a large unified server Mixture-of-Experts.

dynamically routes input samples to the most suitable experts. Inspired by this approach, we propose a novel design where identical compact client models are deployed at the client side, while a large MoE model resides on the server side. Each expert within the server MoE shares identical architecture with each client model .

We introduce Fed-MoE, a Federated Mixture-of-Expert system that enables the aggregation of compact client models into a powerful and large central model. In Fig. 1, we depict this practical scenario featuring a M distributed power-efficient clients, each having with a compact single model. Each client model contributes to one or multiple relevant experts within the server's MoE. This collaboration between client and server models, detailed in the subsequent paragraphs, aims to enhance overall performance and efficiency. With Fed-MoE, thousands of users can collaborate to build a unified billion-scaled large model on the server side, leveraging expertise from all client data.

Our approach involves a three-stage iterative process for updating the server's MoE gate and experts from heterogeneous client models. To facilitate this, we postulate the existence of a small, reserved dataset at the server. We also made substantial re-design of the classical model averaging mechanism. In the first stage, we calculate the relevance between each client expert and each server MoE expert. Then we update the server's MoE experts through a weighted FedAvg process, with weights derived from the relevance scores.

In the second stage, we focus on updating the MoE gate.

Initially, we compute the gating probabilities on the reserved data instances. Subsequently, we aggregate predictions from the top-K most activated experts and calculate the classification loss based on the gating outcomes. Finally, we update the gate parameters end-to-end to minimize this loss.

In the third stage, we synchronize the updated server experts back to each client. By computing a server-to-client correlation matrix, we gather the top-K most relevant server experts for updating each client model. After, the clients perform local updates and send to the server for the next round of FL. Subsequently, the expertise of server experts gradually aligns with the global data space across all clients.

During inference, the MoE gate selectively activates only the relevant subset of experts for each incoming data, reducing computational costs and diversifying expert functions.

To further tackle the non-IID data issue over the clients, at the server-MoE level, we design two types of MoE experts: a *main* expert which always activates, and a set of *routed* experts which share activation together. The main expert dedicates to capturing common knowledge while the routed experts focus on learning specific client data classes. This design enables different routed experts to capture the unique data patterns from different clients. To facilitate the diversification of the routed experts, we further introduce a novel Gating Entropy loss, which encourages a sharply peaked gating distribution over the routed experts.

Our contributions are summarized as follows:

1. We propose an effective Federated Mixture-of-Expert learning framework that allows for the deployment of a large number of compact client models while maintaining a unified large MoE at the server-side.
2. To address the Non-IID issue, we devise a main expert that captures common knowledge, and a set of routed experts that share activation to learn specific client data classes.
3. We design efficient server MoE-experts and MoE-gate update mechanism with innovated gating entropy loss to ensure diversification of the routed experts.
4. Our Fed-MoE shows promising results upon benchmark FL datasets in large-scale vision and language tasks.

2 Related Work

Federated learning (FL). FL [McMahan *et al.*, 2017; Zhao *et al.*, 2018; Sattler *et al.*, 2019; Li and others, 2019; Wu *et al.*, 2020; Karimireddy and others, 2020] emerges as a decentralized and privacy-preserving learning strategy. The pioneering FedAvg [McMahan *et al.*, 2017] demonstrated the effectiveness of model averaging from separately trained client models. Many recent studies worked on tackling the Non-IID setting [Zhao *et al.*, 2018; Sattler *et al.*, 2019; Li *et al.*, 2020], few-shot setting [Wu *et al.*, 2020; Itahara *et al.*, 2023; Jiang *et al.*, 2024b] and privacy enhancement [Wei and others, 2020; Xin *et al.*, 2020; Liu *et al.*, 2023; Fan *et al.*, 2022] of client data in FL scenarios. Some approaches focus on handling model heterogeneity, including HeteroFL [Diao *et al.*, 2021], FedHM [Park and Ko, 2024], FedRolex [Alam *et al.*, 2022], and Split-Mix [Hong *et al.*, 2022].

Mixture-of-Experts (MoE). MoE techniques [Jacobs *et al.*, 1991; Jordan and Jacobs, 1994] utilize a set of expert networks are combined under a gating module which specialize in different aspects of the input data. Recently, MoE has been applied in language modeling with notable successes. The sparsely-gated MoE [Shazeer *et al.*, 2017; Zuo *et al.*, 2021; Jiang *et al.*, 2024a] can scale up the model capacity with less increased computational complexity. GShard [Lepikhin *et al.*, 2020] enable the scaling of multilingual neural machine translation models using sparse gating and automatic sharding. Switch Transformers [Fedus *et al.*, 2022] further scales up Transformer models to trillion-parameter models. GLaM [Du and others, 2022] reduces both training and inference costs of a large MoE language model. MoE has also been widely applied in building large vision-language models [Mustafa *et al.*, 2022; Lin *et al.*, 2024]. DeepSeek-MoE [DeepSeek-AI, 2024] first proposes the concept of shared experts and routed experts.

Recently, some FL studies have made attempts to use MoE for personalized learning. FLMoE [Zec *et al.*, 2020] directly applies FL for MoE models to better suite to heterogeneous client data. AEPFL [Isaksson *et al.*, 2022] achieves a more adaptive cluster model by balancing exploration and uses the cluster model as an expert model in the MoE to enhance performance. PFL-MoE [Guo *et al.*, 2021] modifies the MoE architecture to enhance decision-making capabilities. FedMix [Reisser *et al.*, 2021] directly employs one global MoE to mitigate the Non-IID data by segmenting data source regions. FedJETs [Dun *et al.*, 2023] reduces communication costs by selecting a subset of experts that match the features of client data for communication. Differently, our method allows the server MoE to have a heterogeneous number of experts.

3 Task Description and FL Preliminaries

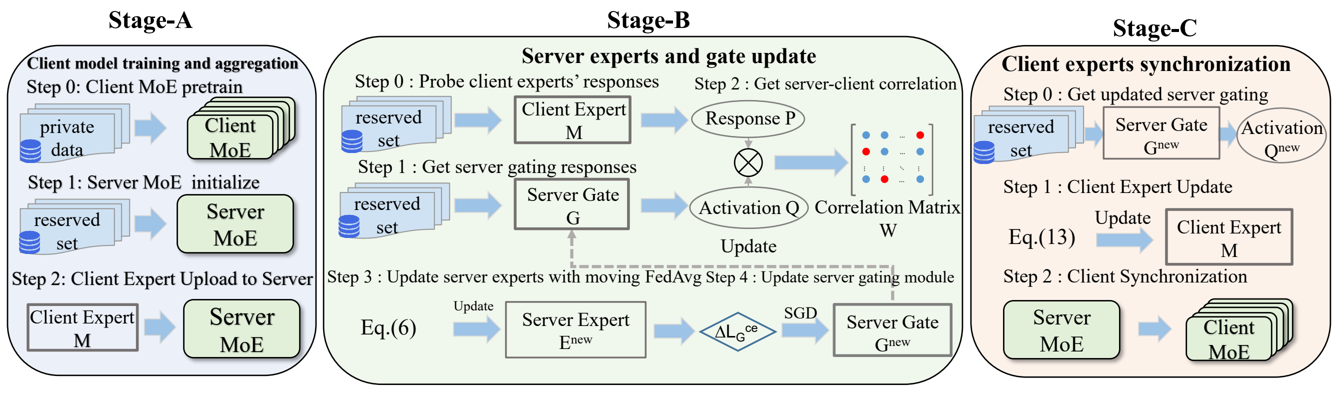
We construct a practical scenarios in which models at client-side and server-side are asymmetric such that the client model is compact while the server model is a large unified MoE.

Model at *client* side. We consider the FL scenario with M distributed clients, each having a compact single-expert model. Let $\boldsymbol{M}_i$ be the i-th client model parameters. At each FL round, we randomly select m participating clients.

MoE at *server* side. We deploy a large K-expert MoE at the server side, including two types of experts: a *main expert* which always activates, denoted as $\boldsymbol{E}_0$ and a group of K *routed experts* denoted as $\boldsymbol{E}_{1:K}$. The routed experts has a trainable gating module $\boldsymbol{G}$ responsible for activating corresponding routed experts. For brevity, we use $K^* = 1 + K$ to denote the number of both main and routed experts.

Server reserved data. We assume the server possesses a tiny reserved dataset D^r, sampled uniformly from the global data space as prior knowledge. The D^r assists training an effective server gating module exclusive to the server side, which is non-overlapping with client data, thus adhering to federated learning principles.

Fed-MoE task formulation. Let $\boldsymbol{G}(x)$ be the gating probability of input x over K routed experts, and $\boldsymbol{E}_i(x)$ be the i-th expert network which predicts input x. The collective response of the MoE module can be expressed as a weighted



Stage-A
Client model training and aggregation
Step 0: Client MoE pretrain
private data
Client MoE
Step 1: Server MoE initialize
reserved set
Server MoE
Step 2: Client Expert Upload to Server
Client Expert M
Server MoE
Stage-B
Server experts and gate update
Step 0 : Probe client experts' responses
reserved set
Client Expert M
Step 2 : Get server-client correlation
Response P
Step 1 : Get server gating responses
reserved set
Server Gate G
Activation Q
Correlation Matrix W
Update
Step 3 : Update server experts with moving FedAvg
Step 4 : Update server gating module
Eq.(6)
Update
Server Expert E^new
ΔL_G^ce
SGD
Server Gate G^new
Stage-C
Client experts synchronization
Step 0 : Get updated server gating
reserved set
Server Gate G^new
Activation Q^new
Step 1 : Client Expert Update
Eq.(13)
Update
Client Expert M
Step 2 : Client Synchronization
Server MoE
Client MoE

Figure 2: Overview of our Fed-MoE pipeline. Stage A-C completes one FL round. Stage-A trains client experts and sends to server. Stage-B iteratively updates server experts and gate. Stage-C synchronizes updated client experts back to clients.

sum of all experts as follows:

$$\hat{y} = (1-\alpha)\ \underbrace{\boldsymbol{E}_0(x)}_{\text{main expert}}\ +\alpha \sum_{i=1}^{K} \underbrace{\boldsymbol{G}_i(x)\cdot \boldsymbol{E}_i(x)}_{\text{routed expert}}\ , \tag{1}$$

where α balances main-expert and routed-experts.

Training objective. Let n_k be the number of data samples for client k. The global learning objective is to minimize the average loss over each client on local data as follows:

$$L(\boldsymbol{E},\boldsymbol{G}) = \frac{1}{n_k}\sum_{j=1}^{n_k} \ell(\hat{y}_j, y_j)\ , \tag{2}$$

where $\hat{y}_i$ is server model prediction following Eq (1).

Following FL, we decompose the above global training objective into training each client model $\boldsymbol{w}^c$. Then the weights of the server experts $\boldsymbol{E}_{1,\ldots,K^*}$ are aggregated from client model $\boldsymbol{w}^c_{1,\ldots,M}$, denoted as: $\boldsymbol{E} \leftarrow Fuse(\boldsymbol{w}^c)$. We will devise an effective $Fuse$ function which enables dynamic expertise dispatching to enhance server experts from client models.

4 Our Fed-MoE Approach

We decompose our Fed-MoE framework with three stages which we introduce in below sections.

4.1 Stage-A: Local client training and uploading

Following standard FL procedure, each FL round starts by training client models with local data. After, m clients are randomly selected to upload their local models to the server through networks, as shown in Stage-A of Fig. 2. The server aggregates these m client models as a federation denoted as $\boldsymbol{M} = \{\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_{m-1}, \boldsymbol{M}_m\}$.

4.2 Stage-B: Server experts and gate update

Stage-B iteratively updates server experts $\boldsymbol{E}$ and gate $\boldsymbol{G}$ for T iterations, which we decompose as the following steps.

Step-0: Probe client experts' responses
We firstly get client experts' responses over reserved set D^r in order to learn their expertise of data classes. For each data instance $(X, y) \in D^r$ (e.g., image-label pair), we feed to all m client experts and obtain the C-way classification probability distribution as $\boldsymbol{P} \leftarrow \mathrm{M}(X) \in \mathbb{R}^{m\times C}$. With ground-truth label y, we get the true class prob. as confidence level of each expert as $\boldsymbol{P}_y = \boldsymbol{P}[:, y] \in \mathbb{R}^{m\times 1}$.

Step-1: Get server gating responses
Next, we will repeat Step-1 to Step-4 for T iterations as an inner-loop to update server gate and α.

At t-th iteration, we begin by feeding data X to the gating module $\boldsymbol{G}$, providing the activation prob. distribution as:

$$\boldsymbol{Q} \leftarrow \boldsymbol{G}(X)\ \in \mathbb{R}^{K\times 1}\ . \tag{3}$$

$\boldsymbol{Q}$ gives soft assignment of query data X to K server experts, which has been normalized with softmax.

Step-2: Get server-client correlation
The outer product of $\boldsymbol{Q}$ and $\boldsymbol{P}_y$ is a correlation matrix:

$$\boldsymbol{W}\ = \boldsymbol{Q}\cdot \boldsymbol{P}_y^T \in \mathbb{R}^{K\times m}\ , \tag{4}$$

where $\boldsymbol{W}_{i,j}$ measures the relevance between the i-th expert of the server MoE and expert from collected j-th client expert.

We subsequently apply a row-wise softmax operation to normalize the correlation matrix as:

$$\boldsymbol{W}^r\ = \sigma^{row}(\boldsymbol{W}) \in \mathbb{R}^{K\times m}\ , \tag{5}$$

where each row $\boldsymbol{W}^r_{i,\cdot}$ sums to one and indicates the correspondence between server expert-i with all m client experts.

Step-3: Update server experts with moving FedAvg
At t-th iteration, we denote the K^* server experts as main-expert $\boldsymbol{E}_0^t$ and routed experts $\boldsymbol{E}_i^t$ for $i \in \{1, ..., K\}$ which get updated with moving-average strategy as follows:

$$\begin{aligned} \boldsymbol{E}_0^{t+1} &\leftarrow (1-\lambda)\cdot \boldsymbol{E}_0^t + \lambda\cdot \bar{\boldsymbol{M}}, \\ \boldsymbol{E}_i^{t+1} &\leftarrow (1-\lambda)\cdot \boldsymbol{E}_i^t + \lambda\cdot \boldsymbol{W}^r\boldsymbol{M}\ ,\ \forall i \geq 1\ . \end{aligned} \tag{6}$$

The $\lambda \in (0, 1)$ controls the moving-average rate. We use simple averaging for $\bar{\boldsymbol{M}}$ with $\bar{\boldsymbol{M}} = 1/m\sum_{i=1}^{m} M_i$. The term $\boldsymbol{W}^r\boldsymbol{M}$ assigns relevant client parameters weighted by correlation $\boldsymbol{W}^r$ and adds up to server weights.

Step-4: Update server gating module

In an ideally diversified MoE system, the gate $\boldsymbol{G}$ learns to route queries to relevant experts with the highest precision for that data class. To this end, we design the learning objective to be the cross-entropy task loss with gating entropy regularization, outlined as follows.

Task loss. For each data in reserved set $\{X, y\} \in D^r$, we estimate C-way distribution from the main expert as $\hat{\boldsymbol{P}}_0 \in \mathbb{R}^{1\times C}$ and K routed experts as $\hat{\boldsymbol{P}}_X \in \mathbb{R}^{K\times C}$.

The MoE gate weighs each routed expert by the activation $\boldsymbol{Q}_X \leftarrow \boldsymbol{G}(X) \in \mathbb{R}^K$. We combine all experts' output such as $\hat{\boldsymbol{P}}^*_X = (1-\alpha)\cdot\hat{\boldsymbol{P}}_0 + \alpha\cdot\hat{\boldsymbol{P}}_X$ as Eq.(1). The task cross-entropy loss is the negative log likelihood such as:

$$L^{ce}_{\boldsymbol{G}} = -\frac{1}{|D|}\sum_{\{X,y\}\in D} \log \bar{\boldsymbol{P}}^*_X[y] \ . \tag{7}$$

Regularization. Intuitively, a sharply peaked gate activation $\boldsymbol{Q}$ implies assigning the data to a specific expert with high confidence. To encourage this desirable expertise diversification, we introduce a novel Gating Entropy (GEnt) loss:

$$L^{ent}_{\boldsymbol{G}} = -\frac{1}{|D|}\sum_{X\in D}\sum_{k=1}^{K} Q_X[k]\cdot\log Q_X[k] \ , \tag{8}$$

in which $Q_X[k]$ indicates the probability (priority) of assigning X to k-th routed expert. We take $L^{ent}_{\boldsymbol{G}}$ as a regularization term that we seek to minimize.

The joint gating loss w.r.t. gate $\boldsymbol{G}$ and hyper-parameter α is

$$L^{gate}(\boldsymbol{G}, \alpha) = L^{ce}_{\boldsymbol{G}} + \beta\cdot L^{ent}_{\boldsymbol{G}} \ , \tag{9}$$

where we set $\beta = 10^{-3}$ and discuss in Ablation Table 5.

During inference, we choose routed experts with top-L gate activation with indices $\mathbb{I} \leftarrow \arg\ \text{TopL}(\boldsymbol{Q}_X)$. Then we integrate main expert and top routed experts' predictions reweighted by the activation values after softmax, such as:

$$\bar{\boldsymbol{P}}_X = (1-\alpha)\cdot\boldsymbol{P}_{\boldsymbol{0}} + \alpha\cdot\sigma(\boldsymbol{Q}^\top_X[\mathbb{I}])\cdot\hat{\boldsymbol{P}}_X[\mathbb{I}] \ . \tag{10}$$

The complete steps of Stage-B are summarized in Algo. 1.

4.3 Stage-C: Client experts synchronization

We update m participating client experts accordingly for next round of local training. As shown in Stage-C of Fig. 2, we decompose this process as follows. We firstly refresh the server-client correlation matrix $\boldsymbol{W} \in \mathbb{R}^{K\times m}$ with server MoE refined in Stage-B, by executing Eq.(3)-(4). We also synchronize the main expert from the server to the client based on the present value of α, as demonstrated by the equation below:

$$\boldsymbol{W'} = \text{cat}(1-\alpha, \alpha\cdot\boldsymbol{W}) \ , \tag{11}$$

We subsequently apply a column-wise softmax operation to correlation matrix $\boldsymbol{W'}$, yielding a normalized server-to-client correspondence as:

$$\boldsymbol{W}^c = \sigma^{col}(\boldsymbol{W'}) \ \in \mathbb{R}^{K^*\times m} \ , \tag{12}$$

in which each column $\boldsymbol{W}^c_{:,j}$ measures the normalized relevance of all server experts with the j-th client expert. This relevance

Algorithm 1 Fed-MoE overview.

```
while round $e \le E$ do
    Stage-A: Local client training and uploading
        /* Upload $m$ participating client models to server. */
        $\boldsymbol{M} \leftarrow \{\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_{m-1}, \boldsymbol{M}_m\}$.
    Stage-B: Server MoE iterative update
        /* Step-0: Probe client experts' responses on $\mathbb{D}^r$. */
        $\boldsymbol{P} \leftarrow \boldsymbol{M}(X),\ \forall(X, y) \in \mathcal{D}$ // client responses
        $\boldsymbol{P}_y \leftarrow \boldsymbol{P}[:, y] \in \mathbb{R}^{K\times 1}$ // label confidence
        while $t \le T$ do
            /* Step-1: Get server gating responses. */
            $\boldsymbol{Q} \leftarrow \boldsymbol{G}(X) \in \mathbb{R}^{K\times 1}$ // gating of Eq.(3)
            /* Step-2: Get server-client correlation. */
            $\boldsymbol{W}^r \leftarrow \sigma^{row}(\boldsymbol{Q}\cdot\boldsymbol{P}_y^\top) \in \mathbb{R}^{K\times m}$
            /* Step-3: Update server experts by moving FedAvg. */
            $\boldsymbol{E}^{t+1}_{\boldsymbol{0}} \leftarrow (1-\lambda)\cdot\boldsymbol{E}^t_0 + \lambda\cdot\bar{\boldsymbol{M}}$,
            $\boldsymbol{E}^{\boldsymbol{t+1}}_{\boldsymbol{i}} \leftarrow (1-\lambda)\cdot\boldsymbol{E}^t_{\boldsymbol{i}} + \lambda\cdot\boldsymbol{W}^r\boldsymbol{M}\ , \forall i \ge 1$ .
            /* Step-4: Update server gating module. */
            $\boldsymbol{G}^{t+1} \leftarrow \boldsymbol{G}^t - \eta\cdot\Delta L^{gate}$ ,
            $\alpha^{t+1} \leftarrow \alpha^t - \eta\cdot\nabla L^{ce}_{\boldsymbol{G}}$ . // Loss Eq.(9)
    Stage-C: Synchronize model $\boldsymbol{E}$ back to clients.
        /* Get updated server gating. */
        $\boldsymbol{Q} \leftarrow \boldsymbol{G}(X)$ // $\boldsymbol{G}$ updated in Stage-B
        /* Get updated server-client correlation. */
        $\boldsymbol{W}^c = \sigma^{col}(\text{cat}(1-\alpha, \alpha\cdot(\boldsymbol{Q}\cdot\boldsymbol{P}_y^\top))) \in \mathbb{R}^{K^*\times m}$
        /* Update client experts by moving FedAvg. */
        $\boldsymbol{M} = \lambda\cdot\boldsymbol{M} + (1-\lambda)\cdot(\boldsymbol{W}^c)^\top\cdot\boldsymbol{E}$ // Eq.(13)
```

can guide each client expert $\boldsymbol{M}_j$ to gather parameters from server experts $\boldsymbol{E}$ with weights such as $(\boldsymbol{W}^c_{:,j})^\top\boldsymbol{E}$.

We formulate the update procedure of client experts $\boldsymbol{M}$ with exponential moving FedAvg in matrix form as:

$$\boldsymbol{M} \leftarrow \lambda\cdot\boldsymbol{M} + (1-\lambda)\cdot(\boldsymbol{W}^c)^\top\cdot\boldsymbol{E} \ , \tag{13}$$

in which $\lambda \in (0, 1)$ controls the moving average rate. Finally, the server transmits the client experts to their corresponding owners for next round of local training. The above process is shown in Fig. 2 (Stage-C) and Algo. 1 (Stage-C).

5 Experiments

We verify our approach on the benchmark Federated Extended MNIST (FEMNIST) [Caldas *et al.*, 2018], CIFAR-10 [Krizhevsky, 2009] for image classification task, SENT-140 [Caldas *et al.*, 2018] for textual sentiment classification task and YELP [Zhang *et al.*, 2015] for 5-way review star classification task.

Vision data split. We follow the original Non-IID split of FEMNIST according to different writing styles of 3500 users. We choose 50 and 100 clients as two FL scenarios for FEMNIST, each having 6200 and 5650 data, respectively. On CIFAR-10, we simulate highly Non-IID scenarios by distributing data classes using a Dirichlet distribution (α=1.0) to ensure that each client gets a unique, proportionately varied subset of classes. For 50 and 100 cases, each client has 750 and 375 data, respectively.

On the sentiment analysis benchmark SENT-140 [Caldas *et al.*, 2018], we follow Fan et al. 2022 to evaluate as a binary classification task. We reserve 100 clients, each having 190 sentences for each class. The server reserves $|D^r| = 1000$ sentences. We tokenize each sentence to a max of 64 words.

The Yelp 5-way classification task aims to predict the number of stars for a review on a scale of 1 as most negative to 5 as most positive. We configured 100 clients, where each client gathers 5,000 data samples. The data is partitioned in a Non-IID fashion, ensuring that one class predominates on each client. The server reserves $|D^r| = 1000$ samples. We tokenize each sentence to a max of 64 words.

5.1 Fed-MoE and Baselines for comparison

Model architectures. In our settings, each client model and each server expert shares a same architecture. The difference is that the server MoE includes a main expert and $K = 5$ routed experts, thus having much more parameters than a single client model. Each single model is a 2-layer CNN for FEMNIST, a ResNet-18 for CIFAR, BERT for SENT-140 sentiment analysis, and GPT-2-Medium for Yelp. Due to GPU resource constraints, in the experiments on SENT140 and Yelp, we use a main expert and $K = 3$ routed experts on the server side. Specifically, for the Yelp experiments, we employ GPT-2 as the gate model for the MoE. The model size and communication costs are summarized in Table 2.

FL baselines. FedAvg [McMahan *et al.*, 2017] uses standard parameter averaging for model fusion. FedProx [Li *et al.*, 2020] adds a proximal term to regularize the client update from deviating too far from the global model. Both methods do not possess MoE parameters and follow standard FL training.

MoE baselines. Cent-MoE (Centralized MoE) trains a plain 5-Exp MoE [Shazeer *et al.*, 2017] only with the server reserved set, without using any client data. FedMix [Reisser *et al.*, 2021] has a 2-Exp MoE which employs a direct FedAvg to aggregate both client experts and their gate modules. Thus its server MoE is also a 2-Exp server MoE. FedJETs [Dun *et al.*, 2023] consists of $m = 5$ anchor clients and $M - m$ ordinary clients, each having a 2-Exp model. In each FL round, m anchor clients as well as m randomly selected ordinary clients participate in learning.

Dataset	FEMNIST (ResNet)	Yelp (GPT-2)
MoE Params.	6.5 / 26 / 52 (M)	0.36 / 0.93 / 1.59 (B)
Comm. Cost	33 / 130 / 33 (M)	1.02 / 2.79 / 1.02 (B)

Table 1: Server MoE parameters and FL communication costs for FedAvg, FedMix, and our Fed-MoE.

5.2 Implementation details

Experimental Settings. We perform all experiments on a system with 3 Nvidia 4090 24G graphics cards, with $M = 50$ and 100 clients to build a large-scale FL system. We set a K-expert server Fed-MoE framework, where a main expert aggregates model parameters from the activated m-clients in each iteration. For vision tasks, we set $K = 5$ and $m = 5$, applying a 2-layer CNN for FEMNIST and ResNet-18 for CIFAR-10. For language modeling tasks, we set $K = 3$ and $m = 3$, using BERT-base for SENT-140 and GPT-2 for YELP which have billion of parameters. Details are in Table 1. The learned routed-experts importance α in Eq. 1 of FEMNIST, CIFAR, SENT140 and Yelp are 0.56, 0.38, 0.48 and 0.50.

Communication cost and model complexity. We show the FedAvg, FedMix and Fed-MoE parameters and FL communication costs in Table 1. All MoE-based methods select m active clients for each round of learning with cost of m model parameters. FedMix and FedJETs have to upload the client gates to the server with additional costs.

For Yelp task, FedAvg has about 0.36B parameters of a single GPT-2 model at the server-side. Our Fed-MoE approach, on the other hand, has 1.59B parameters, comprising a main expert, three routed experts, and a router. In terms of inference workload, Fed-MoE is comparable to FedMix, as only the top-2 experts (including the main expert) are selected for activation. Also, the communication cost for Fed-MoE is only 1.02B, significantly less than that of FedMix, which incurs a substantial 2.79B due to the transfer of additional MoE modules on the client-side.

5.3 Results with Various Datasets and Settings

Vision tasks. Table 2 first two columns (shaded in yellow) present the classification results for FEMNIST and CIFAR-10 datasets under client configurations of ($M = 10$, 50, and 100).

On FEMNIST, Fed-MoE consistently outperforms baseline methods across all client configurations. For instance, at $M = 50$ clients, Fed-MoE achieves over 10% higher accuracy than FedAvg and FedJETs, and approximately 4% and 9% higher than FedMix and FedProx, respectively. We observe that as the number of clients increases and each client possesses less data,our Fed-MoE method gradually decreases, showcasing a larger performance advantage compared to other methods (FedAvg and FedProx).

On CIFAR-10, a similar trend is observed. Fed-MoE achieves 5–7% higher accuracy than FedMix and 1–8% higher accuracy than FedJETs as the number of clients increases from $M = 10$ to $M = 100$. This indicates Fed-MoE's robustness in large-scale federated learning scenarios where Non-IID data and limited per-client data pose significant challenges.

FedMix and FedJETs rely on direct aggregation of gate and expert models, which does not sufficiently account for each server's expertise, leading to suboptimal performance in Non-IID settings. In contrast, our proposed Fed-MoE features an effective expertise dispatching mechanism. It fuses client models into relevant server experts and adaptively updates the gating module to align with evolving expertise.

Language tasks. We show SENT-140 and YELP results in Table 2 columns shaded in yellow. On SENT140, having a BERT [Devlin *et al.*, 2018] for 2-way sentiment classification, we observe that Fed-MoE averagely outperforms the 2nd-place FedMix by 1.7%, FedProx by 2.5% and FedAvg by 2.8%, while leading the weakest FedJETs by 7.45%. On YELP, having a GPT-2 [Devlin *et al.*, 2018] for 5-way task, we observe the Fed-MoE averagely outperforms the 2nd-place FedProx with a 1.2% lead and 2.1% over FedMix. The plain FedProx even surpasses FedMix and FedJETs, implying that a simple average of MoE gate would harm the training in experiments with large-scale models in Non-IID settings.

Fed-MoE outperforms at large-scale FL case. Our Fed-MoE consistently outperforms FedMix and FedJETs, achieving 6–7% higher accuracy on vision tasks and 4–5% on lan-

Dateset	FEMNIST			CIFAR10			SENT140			YELP			AVG
Model	CNN			ResNet			BERT			GPT-2			Acc
Client Num.	10	50	100	10	50	100	10	50	100	10	50	100	
FedAvg 2017	91.89	75.84	74.02	62.30	28.37	24.63	75.90	75.38	73.98	51.44	52.53	50.50	61.39
FedProx 2020	91.66	77.88	76.01	61.88	35.04	32.13	76.06	76.89	75.38	52.88	52.68	52.58	63.42
CentMoE 2017	57.27	57.27	57.27	51.08	51.08	51.08	74.64	74.64	74.64	51.15	51.15	51.15	58.54
FedMix 2021	88.97	83.30	80.83	61.72	59.67	57.20	76.18	76.25	76.01	52.73	51.54	51.23	67.96
FedJETs 2023	89.54	76.96	79.71	66.65	57.81	55.84	71.90	69.83	69.55	50.12	47.97	48.97	65.40
Fed-MoE	**92.11**	**86.03**	**82.58**	**67.62**	**65.52**	**60.73**	**77.56**	**77.96**	**78.10**	**54.11**	**54.12**	**53.46**	**70.83**

Table 2: Classification accuracy (%) on FEMNIST, CIFAR-10, SENT-140 and Yelp datasets with Non-IID settings. Vision tasks are shaded in yellow, and language tasks are in green.

guage tasks in 100-client case. This shows the robustness of our proposed approach in Non-IID scenarios with numerous clients. The key advantage lies in our expertise dispatching mechanism, which dynamically considers the model capacity of each client to enable more effective aggregation. This contrasts with the direct aggregation of gate and expert models employed by FedMix and FedJETs.

Varying reserved data size. We vary the server reserved data size $|D^r|$ and report the corresponding accuracy of Fed-MoE. For FEMNIST, the reserved data of sizes $|D^r| = 320/640/1280$ yield accuracy of $82.86, 82.91, 86.03$. For CIFAR, sizes $|D^r| = 250/500/1000$ yield $55.79, 63.31, 65.52$. This indicates that a reasonably reserved dataset is necessary for training gating module.

5.4 Ablation Studies

Due to the time and computation budget, the following ablations are carried on with 50-client setting.

Server MoE size. We further study how the number of routed-experts at server side affects the performance. Table 3 shows that Fed-MoE achieves the best performance (86.03%) with a moderate 5-Exp MoE. In contrast, though Avg-MoE improves as more experts participate in the training, the overall improvement effect is very limited, and still under-performs 5-Exp Fed-MoE. This indicates the effectiveness of expertise dispatching of Fed-MoE.

Server MoE	5-Exp	10-Exp	20-Exp	30-Exp
Avg-MoE	82.78	82.83	**83.01**	82.92
Fed-MoE	**86.03**	84.77	85.46	85.07

Table 3: Ablation of the number of server experts.

Multi-task training procedures. We examine the effectiveness of the proposed gating entropy (GEnt) loss in Eq.(8) in training objective, as well as client synchronization mechanism in Eq.(13) in Table 4. We observe that the inclusion of gating entropy (+GEnt) alone leads to a slight increase in performance (0.5% for FEMNIST and 0.8% for CIFAR), while client synchronization (+Sync) alone results in a performance boost of 3.4% and 2.6%. Combining Sync and GEnt together yields a huge improvement of 8.0% and 4.2%. The rationale is that GEnt encourages specialization of each server expert, while Sync creates a unified data space across all clients. By leveraging both, server experts can specialize appropriately in tasks within the global data space, thereby improving their effectiveness, particularly in Non-IID case.

Fed-MoE variants	FEMNIST	CIFAR
w/o GEnt & Sync	78.04	61.27
+GEnt	78.57 (+0.5)	62.07 (+0.8)
+Sync	81.48 (+3.4)	63.94 (+2.6)
+GEnt+Sync (Fed-MoE)	**86.03** (+8.0)	**65.52** (+4.2)

Table 4: Ablation of multi-task training.

Weight of Gating Entropy. We study how the weight β of GEnt loss of Eq.(9) affects accuracy and explanability. We observe in Table 5 that a moderate weight $\beta = 10^{-3}$ of GEnt achieves the best accuracy of 65.52% on CIFAR, better than the model without using GEnt (63.94%) or with a tiny weight 10^{-4} (64.25%). Also, increasing weight to 10^{-2} and 10^{-1} decreases the accuracy. This is because a larger GEnt increases the specificity of each expert, but reducing the versatility of the ensemble experts.

GEnt weight	w/o	10^{-4}	10^{-3}	10^{-2}	10^{-1}
Fed-MoE	63.94	64.25	**65.52**	63.14	62.60

Table 5: Ablation of gating entropy weight β in Eq.(9).

We visualize the gating distribution over each server expert in Fig. 3. In left heat-map, setting a large gating entropy weight of $\beta = 10^{-1}$ obviously makes the gating sparse. Each expert specifies to one certain digit, e.g., the gating module sends over 60% label “1” (2nd column) and label “2” (3nd column) to expert-4. With a smaller $\beta = 10^{-3}$ as the right heat-map shows, the gating module assigns soft weights to more experts for a same class. It further illustrates that $\beta = 10^{-3}$ achieves a balance between specificity and versatility, distributing the gating more evenly across multiple experts while still maintaining strong accuracy.

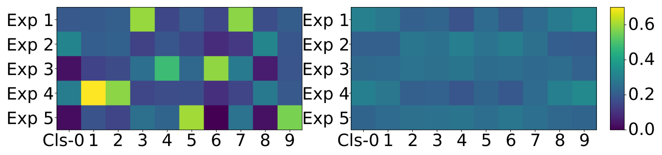


Figure 3: Gating heat-maps reveals each expert (row) specifies certain classes (col.), with left $\beta = 10^{-1}$ and right $\beta = 10^{-3}$.

Inference with Top-L active routed experts. Table 6 shows that on both FEMNIST and CIFAR, using Top-1 routed expert achieves the highest 86.03% and 65.52% accuracy, respectively. In contrast, using Top-5 would lower the accuracy, while costing 5 times computations. Since each expert specializes in certain classes, activating more irrelevant experts can lead to confusion in the final outcomes. This aligns with our practice of using gating entropy to diversify experts.

Top-L	1	2	3	5
FEMNIST	**86.03**	84.49	82.76	84.73
CIFAR	**65.52**	65.50	65.16	64.98

Table 6: Ablation of Top-L routed experts in inference.

Comparison with MoE baselines w/o dynamic update. We study the effectiveness of dynamic update of server experts, as elaborated in Sec. 4.2. We build the following MoE variants. *Cent-MoE* trains a centralized 5-Exp MoE only with the server reserved set. *Avg-MoE* [Reisser *et al.*, 2021] (a.k.a FedMix) has five clients, each equipped with a 2-Exp MoE. Avg-MoE aggregates client models into a 5-Exp MoE server model with FedAvg. *Anchor-MoE* adopts FedJETs [Dun *et al.*, 2023] by configuring a 1-Exp model for each of the 5 anchor clients. The server has a 5-Exp MoE. For each FL round, all 5 anchor clients as well as 5 randomly chosen ordinary clients are used to update server MoE. We show results in Table 7.

	Avg-MoE 2-Exp	Cent-MoE 5-Exp	Anchor-MoE 5-Exp	Fed-MoE 5-Exp
FEMNIST	82.60	57.27	75.88	**86.03**
CIFAR	59.67	51.08	57.81	**65.52**

Table 7: Centralized MoE and parameter averaging MoE.

Fed-MoE outperforms other models due to its dynamic update of the server gate instead of relying solely on aggregation from clients. Cent-MoE performs the worst as it only utilizes server reserved data. Avg-MoE ranks the second, as it aggregates all client MoEs but lacks client-to-server matching. Anchor-MoE ranks behind Fed-MoE, as it restricts itself to a fixed correspondence between client and server experts, thereby limiting its performance.

Comparison with heterogeneous method. We further compare with FedRolex [Alam *et al.*, 2022], a dynamic subnetwork-based FL method, under 100 clients setup. FedRolex achieves 45.50% accuracy, which is significantly lower than our Fed-MoE (60.73%). This demonstrates that even dynamic subnetwork methods struggle under extreme non-IID and large-scale FL settings.

Effect of main-expert. We study how the number of main-experts affects the performance. Table 8 shows that Fed-MoE achieves the best performance on both FEMNIST and SENT140 with just 1-Exp MoE, compared with using no main expert and 2/3-main experts.

	0-Main	1-Main	2-Main	3-Main
FEMNIST	81.05	**86.03**	80.61	83.29
SENT140	75.11	**77.96**	77.11	76.88

Table 8: Ablation of the number of server experts.

Non-IID server reserved data D^r. We let 60% of the server data concentrate on one class, with the rest uniformly distributed. Fig. 4 shows an accuracy gap of about 2-3% between IID and Non-IID scenarios for both Fed-MoE and FedMix, and less 1% in AUC. The F1 score reveals a gap of 6% on Yelp, whereas it is only 1.8% on SENT140, for our Fed-MoE. Otherwise, Fed-MoE showed a slight advantage than FedMix in AUC metrics on both datasets.

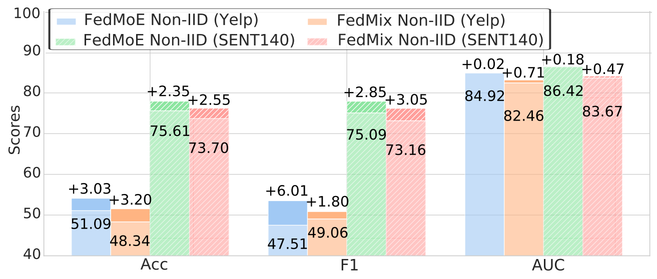


Figure 4: The comparison of Acc, F1, AUC of Fed-MoE and FedMix on SENT140 and Yelp.

6 Conclusion

We introduce an efficient asymmetric FL scheme that efficiently aggregates compact client models with a server-side MoE composed of main experts and routing experts, showcasing superior performance across visual and language tasks. Our dynamical expert gating and updating mechanisms help establish diverse and capable server MoE from client experts. We validated our approach on billion-scale MoE systems with large models, extending its applicability to tasks like image and text classification. Detailed ablation studies confirm its efficiency in convergence and communication performance.

Acknowledgments

This work is supported by the GuangDong Basic and Applied Basic Research Foundation (2024A1515011650) and the National Natural Science Foundation of China (62106156). We thank all reviewers for constructive suggestions.

Contribution Statement

This work was a collaborative effort by all authors. Jingang Jiang† and Yanzhao Chen† contributed equally to this study and are designated as co-first authors. Chenyou Fan* served as the corresponding author and is responsible for all academic correspondence regarding this manuscript.

References

[Alam *et al.*, 2022] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Advances in Neural Information Processing Systems*, volume 35, pages 29677–29690, 2022.

[Caldas *et al.*, 2018] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[DeepSeek-AI, 2024] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Diao *et al.*, 2021] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients, 2021.

[Du and others, 2022] Nan Du et al. Glam: Efficient scaling of language models with mixture-of-experts. In *ICML*, 2022.

[Dun *et al.*, 2023] Chen Dun, Dimitrios Dimitriadis, et al. Fedjets: Efficient just-in-time personalization with federated mixture of experts. *arXiv preprint arXiv:2306.08586*, 2023.

[Fan *et al.*, 2022] Chenyou Fan, Junjie Hu, and Jianwei Huang. Private semi-supervised federated learning. In *IJCAI*, pages 2009–2015, 2022.

[Fedus *et al.*, 2022] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 23(1):5232–5270, 2022.

[Guo *et al.*, 2021] Binbin Guo, Yuan Mei, Danyang Xiao, and Weigang Wu. Pfl-moe: personalized federated learning based on mixture of experts. In *Web and Big Data: 5th International Joint Conference*, 2021.

[Hong *et al.*, 2022] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. In *ICLR*, 2022.

[Isaksson *et al.*, 2022] Martin Isaksson, Edvin Listo Zec, Rickard Cöster, Daniel Gillblad, and Šarūnas Girdzijauskas. Adaptive expert models for personalization in federated learning. *arXiv preprint arXiv:2206.07832*, 2022.

[Itahara *et al.*, 2023] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(01):191–205, 2023.

[Jacobs *et al.*, 1991] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[Jiang *et al.*, 2024a] Albert Q Jiang, Alexandre Sablayrolles, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[Jiang *et al.*, 2024b] Jingang Jiang, Haiqi Jiang, Yuhan Ma, Xiangyang Liu, and Chenyou Fan. Low-parameter federated learning with large language models. In *Web Information Systems and Applications*, pages 319–330, 2024.

[Jordan and Jacobs, 1994] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[Karimireddy and others, 2020] Sai Praneeth Karimireddy et al. Scaffold: stochastic controlled averaging for federated learning. In *ICML*, 2020.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[Lepikhin *et al.*, 2020] Dmitry Lepikhin, HyoukJoong Lee, et al. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[Li and others, 2019] Tian Li et al. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[Lin *et al.*, 2024] Bin Lin, Li Yuan, et al. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024.

[Liu *et al.*, 2023] Xiangyang Liu, Tianqi Pang, and Chenyou Fan. Federated prompting and chain-of-thought reasoning for improving llms answering. In *International Conference on Knowledge Science, Engineering and Management*, pages 3–11. Springer, 2023.

[McMahan *et al.*, 2017] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[Mustafa *et al.*, 2022] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *NeurIPS*, 35:9564–9576, 2022.

[Park and Ko, 2024] JaeYeon Park and JeongGil Ko. Fedhm: Practical federated learning for heterogeneous model deployments. *ICT Express*, 10(2):387–392, 2024.

[Reisser *et al.*, 2021] Matthias Reisser, Christos Louizos, Efstratios Gavves, and Max Welling. Federated mixture of experts. *arXiv preprint arXiv:2107.06724*, 2021.

[Sattler *et al.*, 2019] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE TNNLS*, 2019.

[Shazeer *et al.*, 2017] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[Wei and others, 2020] Kang Wei et al. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 2020.

[Wu *et al.*, 2020] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Computer Graphics and Applications*, 2020.

[Xin *et al.*, 2020] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. Private fl-gan: Differential privacy synthetic data generation based on federated learning. In *ICASSP*, 2020.

[Zec *et al.*, 2020] Edvin Listo Zec, John Martinsson, Olof Mogren, Leon René Sütfeld, and Daniel Gillblad. Federated learning using mixture of experts. *arXiv preprint arXiv:2107.06724*, 2020.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[Zhao *et al.*, 2018] 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.

[Zuo *et al.*, 2021] Simiao Zuo, Jianfeng Gao, et al. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*, 2021.