
Modeling Inter-Intra Heterogeneity for Graph Federated Learning (Appendix)

Wentao Yu¹ Shuo Chen^{2*} Yongxin Tong³ Tianlong Gu⁴ Chen Gong^{5*}

¹School of Computer Science and Engineering,

Nanjing University of Science and Technology, China

²Center for Advanced Intelligence Project, RIKEN, Japan

³State Key Laboratory of Complex & Critical Software Environment, Beihang University, China

⁴Engineering Research Center of Trustworthy AI (Ministry of Education),
Jinan University, China

⁵Department of Automation, Institute of Image Processing and Pattern Recognition,
Shanghai Jiao Tong University, China

wentao.yu@njust.edu.cn, shuo.chen.ya@riken.jp, yxtong@buaa.edu.cn
gutianlong@jnu.edu.cn, chen.gong@sjtu.edu.cn

A Details of Neighborhood Routing Mechanism

Recall that we use the node u as an example to describe the disentangling process of DisenGCN. By using Eq. (1) in the main manuscript, the obtained $\mathbf{z}^{i,k}$ describes the aspect of node i that are related with the k -th latent factor. However, $\mathbf{z}^{i,k}$ only denotes the information of the node i itself. Therefore, for the node u , we have to mine information from the neighborhoods, which are connected with node u due to the k -th latent factor. Specifically, we aim to identify the latent factor that causes the connection between node u and its neighbor node v , and accordingly extract features of v that are specific to that factor. This is exactly the purpose of the neighborhood routing mechanism.

First, we define $p^{v,k}$, which represents the probability that the latent factor k is the reason why the node u reaches its neighbor v . Then, we can have $p^{v,k} \geq 0$, and $\sum_{k'=1}^K p^{v,k'} = 1$. Moreover, $p^{v,k}$ is also the probability that we should use the neighbor v to construct $\mathbf{c}^{u,k}$. The neighborhood routing mechanism will infer $p^{v,k}$ and construct $\mathbf{c}^{u,k}$ in an iterative manner. To initialize this process, we make $(p^{v,k})^{(1)} \propto \exp[(\mathbf{z}^{v,k})^\top \mathbf{z}^{u,k} / \tau_p]$. Then, it subsequently iterates to identify the largest cluster within each subspace, ensuring that each neighbor is primarily associated with a single subspace cluster:

$$(\mathbf{c}^{u,k})^{(t)} = \frac{\mathbf{z}^{u,k} + \sum_{v:(u,v) \in \mathcal{G}_m} (p^{v,k})^{(t-1)} \mathbf{z}^{v,k}}{\|\mathbf{z}^{u,k} + \sum_{v:(u,v) \in \mathcal{G}_m} (p^{v,k})^{(t-1)} \mathbf{z}^{v,k}\|_2}, \quad (1)$$

$$(p^{v,k})^{(t)} = \frac{\exp[(\mathbf{z}^{v,k})^\top (\mathbf{c}^{u,k})^{(t)} / \tau_p]}{\sum_{k'=1}^K \exp[(\mathbf{z}^{v,k'})^\top (\mathbf{c}^{u,k'})^{(t)} / \tau_p]}, \quad (2)$$

where $(\mathbf{c}^{u,k})^{(t)}$ denotes the center of each subspace cluster at the t -th iteration, $t = 2, 3, \dots, T$, and τ_p is a hyperparameter that controls the hardness of the assignment. Here τ_p is set to 1 according to [1]. After T times of iterations, the final output is $\mathbf{c}^{u,k} = (\mathbf{c}^{u,k})^{(T)}$.

B Instantiations of Distributions

We summarize the family of distributions instantiated by our proposed FedIIH in Tab. 1. Specifically, the priors $p(\alpha^k)$ and $p(\tilde{\mathbf{H}}_m^k)$ are both centered isotropic multivariate Gaussian distributions. Besides,

*Corresponding authors: Chen Gong, Shuo Chen.

Table 1: Family of distributions instantiated by our proposed FedIIH.

$p(\boldsymbol{\alpha}^k)$	$\mathcal{N}(0, \sigma_{\boldsymbol{\alpha}^k}^2 \mathbf{I})$
$p(\tilde{\mathbf{H}}_m^k)$	$\mathcal{N}(0, \sigma_{\tilde{\mathbf{H}}_m^k}^2 \mathbf{I})$
$p(\tilde{\mathbf{H}}_m^k \boldsymbol{\alpha}^k)$	$\mathcal{N}(\boldsymbol{\alpha}^k, \sigma_{\tilde{\mathbf{H}}_m^k}^2 \mathbf{I})$
$q(\boldsymbol{\alpha}^k)$	$\mathcal{N}(\tilde{\boldsymbol{\alpha}}^k, \sigma_{\tilde{\boldsymbol{\alpha}}^k}^2 \mathbf{I})$
$q(\tilde{\mathbf{H}}_m^k \mathcal{G}_m)$	$\mathcal{N}(\hat{\boldsymbol{\mu}}_{\tilde{\mathbf{H}}_m^k}, \hat{\sigma}_{\tilde{\mathbf{H}}_m^k}^2)$

the prior over the local latent factor $\tilde{\mathbf{H}}_m^k$ conditioned on $\boldsymbol{\alpha}^k$ (*i.e.*, $p(\tilde{\mathbf{H}}_m^k | \boldsymbol{\alpha}^k)$) is an isotropic multi-variate Gaussian distribution centered at $\boldsymbol{\alpha}^k$. Similarly, the marginal distribution of $\boldsymbol{\alpha}^k$ (*i.e.*, $q(\boldsymbol{\alpha}^k)$) is a multivariate diagonal Gaussian distribution. As shown in Tab. 1, $\tilde{\boldsymbol{\alpha}}^k$ and $\sigma_{\tilde{\boldsymbol{\alpha}}^k}^2$ denote the posterior mean and variance of $\boldsymbol{\alpha}^k$, respectively. Moreover, $\hat{\boldsymbol{\mu}}_{\tilde{\mathbf{H}}_m^k}$ and $\hat{\sigma}_{\tilde{\mathbf{H}}_m^k}^2$ denote the variational mean and variance evaluated at \mathcal{G}_m , respectively.

C Derivation of the ELBO

First, the ELBO for the marginal likelihood of $\mathcal{G}_{1:M}$ (*i.e.*, $\log p(\mathcal{G}_{1:M})$) can be obtained by using the Jensen inequality:

$$\begin{aligned}
 & \log p(\mathcal{G}_{1:M}) \\
 &= \log \sum_{\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}} q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}) \frac{p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K})}{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})}, \\
 &\geq \sum_{\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}} q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}) \log \frac{p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K})}{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})}, \\
 &\triangleq \text{ELBO} \left(q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}), \mathcal{G}_{1:M} \right), \\
 &= \mathbb{E}_{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})} \left[\log p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K}) \right. \\
 &\quad \left. - \log q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}) \right].
 \end{aligned} \tag{3}$$

Second, inspired by [2], the ELBO can be derived as follows:

$$\begin{aligned}
 & \mathbb{E}_{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})} \left[\log p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K}) \right. \\
 &\quad \left. - \log q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M}) \right] \\
 &= \sum_{m=1}^M \mathbb{E}_{q(\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K | \mathcal{G}_m)} \left[\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K) \right. \\
 &\quad \left. - \sum_{m=1}^M \sum_{k=1}^K \mathbb{E}_{q(\boldsymbol{\alpha}^k)} \left[D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \boldsymbol{\alpha}^k)) \right] \right. \\
 &\quad \left. - \sum_{k=1}^K D_{\text{KL}}(q(\boldsymbol{\alpha}^k) || p(\boldsymbol{\alpha}^k)) \right].
 \end{aligned} \tag{4}$$

Third, we compute the expected KL divergence of two Gaussian distributions (*i.e.*, $q(\tilde{\mathbf{H}}_m^k|\mathcal{G}_m)$ and $p(\tilde{\mathbf{H}}_m^k|\alpha^k)$) over a Gaussian distribution (*i.e.*, $q(\alpha^k)$) analytically. According to Tab. 1, we can have

$$\begin{aligned}
& \mathbb{E}_{q(\alpha^k)} [D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k|\mathcal{G}_m)||p(\tilde{\mathbf{H}}_m^k|\alpha^k))] \\
&= \mathbb{E}_{q(\alpha^k)} [D_{\text{KL}}(\mathcal{N}(\hat{\boldsymbol{\mu}}_{\tilde{\mathbf{H}}_m^k}, \hat{\boldsymbol{\sigma}}_{\tilde{\mathbf{H}}_m^k}^2)||\mathcal{N}(\alpha^k, \sigma_{\tilde{\mathbf{H}}_m^k}^2 \mathbf{I}))] \\
&= \mathbb{E}_{q(\alpha^k)} \left[-\frac{1}{2} \sum_{j=1}^J \left(1 + \log \frac{\hat{\sigma}_{\tilde{\mathbf{H}}_m^k, j}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} - \frac{(\hat{\mu}_{\tilde{\mathbf{H}}_m^k, j} - \alpha^{k, j})^2 + \hat{\sigma}_{\tilde{\mathbf{H}}_m^k, j}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} \right) \right] \\
&= -\frac{1}{2} \sum_{j=1}^J \left(1 + \log \frac{\hat{\sigma}_{\tilde{\mathbf{H}}_m^k, j}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} - \frac{\hat{\sigma}_{\tilde{\mathbf{H}}_m^k, j}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} - \mathbb{E}_{q(\alpha^k)} \left[\frac{(\hat{\mu}_{\tilde{\mathbf{H}}_m^k, j} - \alpha^{k, j})^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} \right] \right) \quad (5) \\
&= D_{\text{KL}}(\mathcal{N}(\hat{\boldsymbol{\mu}}_{\tilde{\mathbf{H}}_m^k}, \hat{\boldsymbol{\sigma}}_{\tilde{\mathbf{H}}_m^k}^2)||\mathcal{N}(\tilde{\alpha}^k, \sigma_{\tilde{\mathbf{H}}_m^k}^2)) + \frac{J}{2} \frac{\sigma_{\tilde{\alpha}^k}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} \\
&= D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k|\mathcal{G}_m)||p(\tilde{\mathbf{H}}_m^k|\tilde{\alpha}^k)) + \frac{J}{2} \frac{\sigma_{\tilde{\alpha}^k}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2},
\end{aligned}$$

where J denotes the dimension of $\tilde{\mathbf{H}}_m^k$. Moreover, $\alpha^{k, j}$, $\hat{\mu}_{\tilde{\mathbf{H}}_m^k, j}$, and $\hat{\sigma}_{\tilde{\mathbf{H}}_m^k, j}^2$ denote the j -th element of α^k , $\hat{\boldsymbol{\mu}}_{\tilde{\mathbf{H}}_m^k}$, and $\hat{\boldsymbol{\sigma}}_{\tilde{\mathbf{H}}_m^k}^2$, respectively.

Fourth, the KL divergence between $q(\alpha^k)$ and $p(\alpha^k)$ can be computed analytically as

$$\begin{aligned}
& D_{\text{KL}}(q(\alpha^k)||p(\alpha^k)) \\
&= D_{\text{KL}}(\mathcal{N}(\tilde{\alpha}^k, \sigma_{\tilde{\alpha}^k}^2 \mathbf{I})||\mathcal{N}(0, \sigma_{\alpha^k}^2 \mathbf{I})) \\
&= -\frac{1}{2} \sum_{j=1}^J \left(1 + \log \frac{\sigma_{\tilde{\alpha}^k}^2}{\sigma_{\alpha^k}^2} - \frac{(\tilde{\alpha}^{k, j} - 0)^2 + \sigma_{\tilde{\alpha}^k}^2}{\sigma_{\alpha^k}^2} \right) \quad (6) \\
&= -\frac{1}{2} \sum_{j=1}^J (1 + \log \sigma_{\tilde{\alpha}^k}^2) - \frac{1}{2} \log 2\pi - \log p(\tilde{\alpha}^k),
\end{aligned}$$

where $\tilde{\alpha}^{k, j}$ denote the j -th element of $\tilde{\alpha}^k$.

Fifth, we substitute the result of Eq. (5) and Eq. (6) into Eq. (4), respectively. Then, we can have

$$\begin{aligned}
& \mathbb{E}_{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \alpha^{1:K}|\mathcal{G}_{1:M})} [\log p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \alpha^{1:K}) \\
& \quad - \log q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \alpha^{1:K}|\mathcal{G}_{1:M})] \\
&= \sum_{m=1}^M \mathbb{E}_{q(\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K|\mathcal{G}_m)} [\log p(\mathcal{G}_m|\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K)] \quad (7) \\
& \quad - \sum_{m=1}^M \sum_{k=1}^K \mathbb{E}_{q(\alpha^k)} [D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k|\mathcal{G}_m)||p(\tilde{\mathbf{H}}_m^k|\alpha^k))]
\end{aligned}$$

$$\begin{aligned}
& - \sum_{k=1}^K D_{\text{KL}}(q(\boldsymbol{\alpha}^k) || p(\boldsymbol{\alpha}^k)) \\
& = \sum_{m=1}^M \mathbb{E}_{q(\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K)] \\
& - \sum_{m=1}^M \sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \tilde{\boldsymbol{\alpha}}^k)) - \sum_{m=1}^M \sum_{k=1}^K \frac{J}{2} \frac{\sigma_{\tilde{\boldsymbol{\alpha}}^k}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} \\
& + \sum_{k=1}^K \left[\frac{1}{2} \sum_{j=1}^J (1 + \log \sigma_{\tilde{\boldsymbol{\alpha}}^k}^2) + \frac{1}{2} \log 2\pi + \log p(\tilde{\boldsymbol{\alpha}}^k) \right] \\
& = \sum_{m=1}^M \left\{ \mathbb{E}_{q(\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K)] \right. \\
& - \sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \tilde{\boldsymbol{\alpha}}^k)) \\
& \left. - \underbrace{\sum_{k=1}^K \frac{J}{2} \frac{\sigma_{\tilde{\boldsymbol{\alpha}}^k}^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2}}_{\text{constant}} \right\} \\
& + \underbrace{\sum_{k=1}^K \left[\log p(\tilde{\boldsymbol{\alpha}}^k) + \frac{1}{2} \sum_{j=1}^J (1 + \log \sigma_{\tilde{\boldsymbol{\alpha}}^k}^2) + \frac{1}{2} \log 2\pi \right]}_{\text{constant}}.
\end{aligned} \tag{8}$$

Finally, since two constants in the above Eq. (7) can be omitted, we can have

$$\begin{aligned}
& \mathbb{E}_{q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})} [\log p(\mathcal{G}_{1:M}, \tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K})] \\
& - \log q(\tilde{\mathbf{H}}_{1:M}^1, \tilde{\mathbf{H}}_{1:M}^2, \dots, \tilde{\mathbf{H}}_{1:M}^K, \boldsymbol{\alpha}^{1:K} | \mathcal{G}_{1:M})] \\
& \approx \sum_{m=1}^M \left\{ \mathbb{E}_{q(\tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m^1, \tilde{\mathbf{H}}_m^2, \dots, \tilde{\mathbf{H}}_m^K)] \right. \\
& - \sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \tilde{\boldsymbol{\alpha}}^k)) \left. \right\} \\
& + \sum_{k=1}^K \log p(\tilde{\boldsymbol{\alpha}}^k) \\
& = \sum_{m=1}^M \left\{ \mathbb{E}_{q(\tilde{\mathbf{H}}_m | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m)] \right. \\
& - \sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \tilde{\boldsymbol{\alpha}}^k)) \left. \right\} \\
& + \sum_{k=1}^K \log p(\tilde{\boldsymbol{\alpha}}^k).
\end{aligned} \tag{9}$$

D Derivation of $\tilde{\alpha}^k$

Since estimating the exact Maximum A Posterior (MAP) of α^k is intractable, inspired by [2], we approximate α^k with the help of ELBO as follows:

$$\begin{aligned}
\tilde{\alpha}^k &= \operatorname{argmax}_{\alpha^k} \log p(\alpha^k | \mathcal{G}_{1:M}) \\
&= \operatorname{argmax}_{\alpha^k} \log \frac{p(\mathcal{G}_{1:M}, \alpha^k)}{p(\mathcal{G}_{1:M})} \\
&= \operatorname{argmax}_{\alpha^k} \log p(\mathcal{G}_{1:M}, \alpha^k) \\
&= \operatorname{argmax}_{\alpha^k} \left(\sum_{m=1}^M \log p(\mathcal{G}_m | \alpha^k) \right) + \log p(\alpha^k) \\
&\approx \operatorname{argmax}_{\alpha^k} \sum_{m=1}^M \left\{ \mathbb{E}_{q(\tilde{\mathbf{H}}_m | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m)] - \sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \tilde{\alpha}^k)) \right\} + \log p(\alpha^k) \\
&= \operatorname{argmax}_{\alpha^k} \sum_{m=1}^M \sum_{k=1}^K -D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k | \mathcal{G}_m) || p(\tilde{\mathbf{H}}_m^k | \alpha^k)) + \log p(\alpha^k) \\
&= \operatorname{argmax}_{\alpha^k} \sum_{m=1}^M \sum_{k=1}^K \sum_{j=1}^J \frac{(\hat{\mu}_{\tilde{\mathbf{H}}_m^k, j} - \alpha^{k,j})^2}{\sigma_{\tilde{\mathbf{H}}_m^k}^2} - \sum_{j=1}^J \frac{(\alpha^{k,j} - 0)^2}{\sigma_{\alpha^k}^2} \\
&= \operatorname{argmax}_{\alpha^k} f(\alpha^k),
\end{aligned} \tag{10}$$

where $f(\alpha^k)$ is a concave quadratic function with only one maximum point, and $\alpha^{k,j}$ denotes the j -th element of α^k . The closed-form solution of $\tilde{\alpha}^k$ can be derived by differentiating Eq. (10) with respect to α^k . Specifically, we let

$$\left. \frac{\partial f(\alpha^k)}{\partial \alpha^k} \right|_{\alpha^k = \tilde{\alpha}^k} = 0, \tag{11}$$

and then we can have

$$\tilde{\alpha}^k = \frac{\sum_{m=1}^M \hat{\mu}_{\tilde{\mathbf{H}}_m^k}}{M + \frac{\sigma_{\tilde{\mathbf{H}}_m^k}^2}{\sigma_{\alpha^k}^2}}. \tag{12}$$

E Details of HVGAE

In this section, we introduce the detailed neural network architectures for our proposed HVGAE. Since HVGAE is deployed in each client, we take the m -th client as an example. Recall that \mathcal{G}_m is the subgraph on the m -th client, which contains the node feature matrix \mathbf{X}_m and adjacency matrix \mathbf{A}_m . We use two DisenGCNs (*i.e.*, $\text{DisenGCN}_{\mu_m}(\mathcal{G}_m)$ and $\text{DisenGCN}_{\sigma_m}(\mathcal{G}_m)$) as the encoder and an inner product as the decoder of HVGAE, respectively. Note that $\text{DisenGCN}_{\mu_m}(\mathcal{G}_m)$ and $\text{DisenGCN}_{\sigma_m}(\mathcal{G}_m)$ are used to infer the means and standard deviations of \mathcal{G}_m for K latent factors, respectively. Moreover, $\text{DisenGCN}_{\mu}(\mathcal{G}_m)$ and $\text{DisenGCN}_{\sigma}(\mathcal{G}_m)$ share the same node feature projection layer.

Fig. 1 shows the architecture of our proposed HVGAE. First, we project the node feature to K subspaces according to Eq. (1) in the main manuscript. Meanwhile, the node representations after the node feature projection layer are used to train a local node classifier. Second, we use the neighborhood routing mechanism to obtain $\tilde{\mathbf{H}}_m^{1:K}$ and $\tilde{\mathbf{H}}_m^{1:K}$, respectively. Third, we use the reparameterization trick [3] to sample $\tilde{\mathbf{H}}_m^{1:K}$ from $\tilde{\mathbf{H}}_m^{1:K}$ and $\tilde{\mathbf{H}}_m^{1:K}$ (see Eq. (7) in the main manuscript). Fourth, HVGAE decodes from $\tilde{\mathbf{H}}_m^{1:K}$ and then computes $\mathbb{E}_{q(\tilde{\mathbf{H}}_m | \mathcal{G}_m)} [\log p(\mathcal{G}_m | \tilde{\mathbf{H}}_m)]$ (see Eq. (8) in the main manuscript). The implementations of two prior distributions $p(\tilde{\alpha}^k)$ and $p(\tilde{\mathbf{H}}_m^k | \tilde{\alpha}^k)$ are shown in the Appendix I.6.

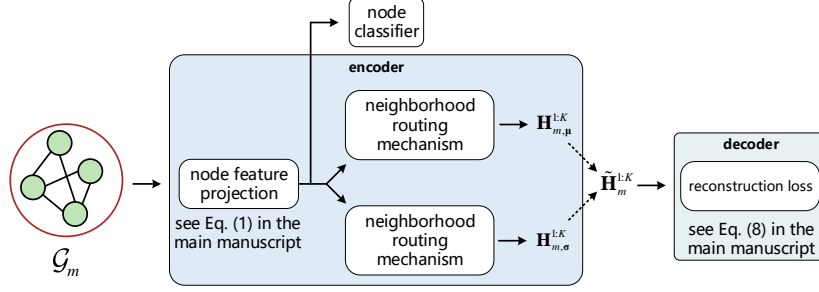


Figure 1: The architecture of our proposed HVGAE. The dashed lines show the process of sampling using the reparameterization trick [3].

F Additional Related Work

Here we review the typical works related to the FL via Bayesian methods. Recently, some methods [4, 5, 6] have tried to model the FL problem by using the Bayesian theories. Specifically, they take the network weights as a whole entity and treat them as a single random variable shared by all clients. For example, FedPA [4] infers the global posterior by averaging the local posteriors. However, FedPA targets a general FL setting, so it does not apply to the personalized FL, and its performance may be degraded. Different from FedPA, pFedBayes [5] assigns a personalized Bayesian neural network to each client. It infers the global posterior from individual posteriors under a regularizer based on the KL divergence. However, it cannot model the posterior of each client from a global perspective. To address this deficiency, Kim *et. al* propose a hierarchical Bayesian approach called FedHB [6], which mitigates the heterogeneity problem in a personalized way. Specifically, it introduces two types of latent random variables, one used as the network weights for each client’s backbone, and the other used as a globally shared random variable to be associated with each client. Unlike FedHB, we model the local subgraph data rather than the network weight, so that we can infer the subgraph data distribution on each client.

G Analysis of Federated Parameters

According to the architecture of our proposed HVGAE, there are several components: a node feature projection layer, two neighborhood routing mechanism layers, a node classifier, and a reconstruction layer. First, recall that there are no learnable parameters in the neighborhood routing mechanism. Second, the reconstruction layer is constructed by an inner product, so there are no learnable parameters. Third, although we use two DisenGCNs (*i.e.*, $\text{DisenGCN}_{\mu_m}(\mathcal{G}_m)$ and $\text{DisenGCN}_{\sigma_m}(\mathcal{G}_m)$) as the encoder of our proposed HVGAE, they share the same node feature projection layer (see Appendix E). Consequently, according to the node feature projection layer defined by Eq. (1) in the main manuscript, we can find that $\mathbf{W}_m^1, \mathbf{W}_m^2, \dots, \mathbf{W}_m^K$ and $\mathbf{b}_m^1, \mathbf{b}_m^2, \dots, \mathbf{b}_m^K$ are learnable parameters that should be federated. Last but not least, since we introduce a node classifier, which is actually a Multi-Layer Perceptron (MLP), we can find that $\mathbf{W}_m^{\text{cls}} \in \mathbb{R}^{c \times d_{\text{out}}}$ and $\mathbf{b}_m^{\text{cls}} \in \mathbb{R}^c$ are learnable parameters that should be federated.

In Eq. (6) of the main manuscript, although we have defined the learnable parameter $\hat{\alpha}_m^k$ on each client, $\hat{\alpha}_m^k$ is only used to approximate the data distribution instead of a parameter in a neural network. Consequently, $\hat{\alpha}_m^k$ does not participate in the federation.

H Pseudocode of FedIIH

In this section, we show the pseudocode of our proposed FedIIH for the clients and server in Algorithm 1 and Algorithm 2, respectively. Moreover, the framework of our proposed FedIIH from the perspective of the clients and the server is shown in Fig. 2 and Fig. 3, respectively. Inspired by [7], we regard the posterior mean of α^k for the k -th global latent factor (*i.e.*, $\tilde{\alpha}^k$) in the last round as the prior for the k -th local latent factor on each client in the current round.

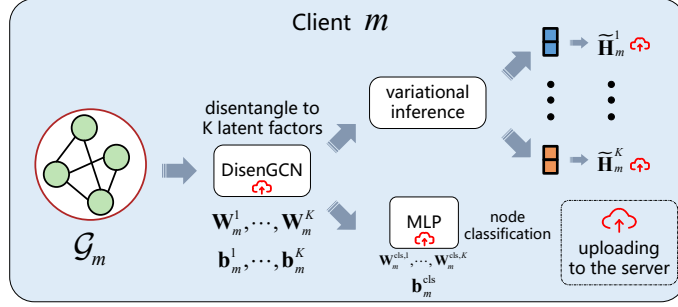


Figure 2: The framework of our proposed FedIIH from the perspective of the clients. Let us take the client m as an example.

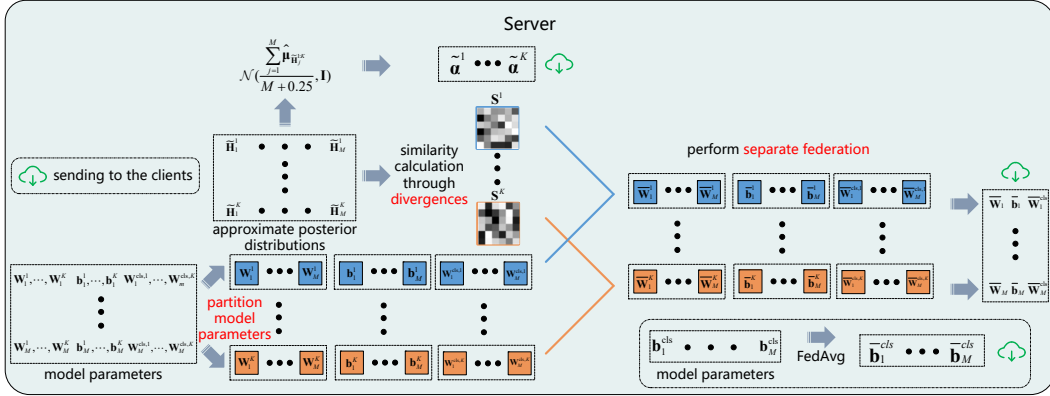


Figure 3: The framework of our proposed FedIIH from the perspective of the server, where S^k denotes the similarity matrix with respect to the k -th latent factor. Here we use blue and orange to represent the first and the K -th latent factors, respectively.

I Implementation Details

In this section, we provide the implementation details, including the experimental platform, the dataset descriptions, the details of subgraph partitioning, the information on baseline methods, training details, and the implementations of two prior distributions (*i.e.*, $p(\tilde{\alpha}^k)$ and $p(\tilde{\mathbf{H}}_m^k | \tilde{\alpha}^k)$).

I.1 Experimental Platform

All the experiments in this work are conducted on a Linux server with a 2.90 GHz Intel Xeon Gold 6326 CPU, 64 GB of RAM, and two NVIDIA GeForce RTX 4090 GPUs with 48GB of memory. Our proposed method is implemented via Python 3.8.8, PyTorch 1.12.0, and PyTorch Geometric (PyG) 2.3.0 with the Adam optimizer.

I.2 Datasets

To validate the effectiveness of our proposed FedIIH, we perform extensive experiments on eleven widely used benchmark datasets, including six homophilic and five heterophilic graph datasets. In the homophilic graph datasets, there are *Cora*, *CiteSeer*, *PubMed*, and *ogbn-arxiv* for the citation graphs; *Amazon-Computer* and *Amazon-Photo* for Amazon product graphs. In the heterophilic graph datasets [9], there are *Roman-empire*, *Amazon-ratings*, *Minesweeper*, *Tolokers*, and *Questions*. The statistical information of the above benchmark datasets is described in Tab. 2. Note that we use the Area Under the ROC curve (AUC) as the evaluation metric (higher values are better) for the *Minesweeper*, *Tolokers*, and *Questions* datasets, and use the accuracy as the evaluation metric (higher values are better) for other datasets.

Algorithm 1: FedIIH Client Algorithm

Input: Number of local epochs E ; number of latent factors K ; subgraph \mathcal{G}_m on client m ; node feature matrix \mathbf{X}_m on client m ; label matrix \mathbf{Y}_m on client m ; parameters of two DisenGCNs $\mathbf{W}_m^{1:K}$ and $\mathbf{b}_m^{1:K}$ on client m ; parameters of the node classifier $\mathbf{W}_m^{\text{cls}}$ and $\mathbf{b}_m^{\text{cls}}$ on client m ; federated parameters $\overline{\mathbf{W}}_m^{\text{cls}}$, $\overline{\mathbf{b}}_m^{\text{cls}}$, $\overline{\mathbf{W}}_m^{1:K}$, and $\overline{\mathbf{b}}_m^{1:K}$ from the server; global latent factors $\tilde{\alpha}^{1:K}$ from the server.

Output: Predicted label for each unlabeled graph node in subgraph \mathcal{G}_m .

- 1 Download federated parameters $\overline{\mathbf{W}}_m^{\text{cls}}$, $\overline{\mathbf{b}}_m^{\text{cls}}$, $\overline{\mathbf{W}}_m^{1:K}$, and $\overline{\mathbf{b}}_m^{1:K}$ from the server;
 - 2 Download the global latent factors $\tilde{\alpha}^{1:K}$ from the server;
 - 3 $\mathbf{W}_m^{\text{cls}} \leftarrow \overline{\mathbf{W}}_m^{\text{cls}}$, $\mathbf{b}_m^{\text{cls}} \leftarrow \overline{\mathbf{b}}_m^{\text{cls}}$, $\mathbf{W}_m^{1:K} \leftarrow \overline{\mathbf{W}}_m^{1:K}$, $\mathbf{b}_m^{1:K} \leftarrow \overline{\mathbf{b}}_m^{1:K}$;
 - 4 **for** each local epoch e from 1 to E **do**
 - 5 Project the node feature into K subspaces via Eq. (1) in the main manuscript;
 - 6 Optimize $\mathbf{W}_m^{\text{cls}}$ and $\mathbf{b}_m^{\text{cls}}$ via the cross-entropy loss;
 - 7 Disentangle projected node representations into K latent factors via the neighborhood routing mechanism, and then obtain $\mathbf{H}_{m,\mu}^{1:K}$ and $\mathbf{H}_{m,\sigma}^{1:K}$, respectively;
 - 8 Sample $\tilde{\mathbf{H}}_m^{1:K}$ from $\mathbf{H}_{m,\mu}^{1:K}$ and $\mathbf{H}_{m,\sigma}^{1:K}$ via Eq. (7) in the main manuscript so as to obtain $q(\tilde{\mathbf{H}}_m^{1:K}|\mathcal{G}_m)$;
 - 9 Compute $\mathbb{E}_{q(\tilde{\mathbf{H}}_m|\mathcal{G}_m)}[\log p(\mathcal{G}_m|\tilde{\mathbf{H}}_m)]$ via Eq. (8) in the main manuscript;
 - 10 Approximate $\tilde{\alpha}^{1:K}$ by using $\hat{\alpha}_m^{1:K}$ so as to compute $\sum_{k=1}^K \left\{ \log p(\hat{\alpha}_m^k) - D_{\text{KL}}(p(\hat{\alpha}_m^k)||p(\tilde{\alpha}^k)) \right\}$;
 - 11 Compute $\sum_{k=1}^K D_{\text{KL}}(q(\tilde{\mathbf{H}}_m^k|\mathcal{G}_m)||p(\tilde{\mathbf{H}}_m^k|\tilde{\alpha}^k))$, where $p(\tilde{\mathbf{H}}_m^{1:K}|\tilde{\alpha}^{1:K}) \sim \mathcal{N}(\tilde{\alpha}^{1:K}, \sigma_{\tilde{\mathbf{H}}_m^{1:K}}^2 \mathbf{I})$;
 - 12 Optimize $\mathbf{W}_m^{1:K}$ and $\mathbf{b}_m^{1:K}$ via Eq. (6) in the main manuscript;
 - 13 **end**
 - 14 Upload $\mathbf{W}_m^{\text{cls}}$, $\mathbf{b}_m^{\text{cls}}$, $\mathbf{W}_m^{1:K}$, $\mathbf{b}_m^{1:K}$, and $\tilde{\mathbf{H}}_m^{1:K}$ to the server;
 - 15 Predict labels based on the trained node classifier.
-

In order to facilitate the division of datasets, a random sample of 20% of nodes is selected for training purposes, 40% for the purpose of validation, and 40% for testing, with the exception of the *ogbn-arxiv* dataset. This is due to the fact that the *ogbn-arxiv* dataset comprises a relatively large number of nodes in comparison to the other datasets, as reported in Tab. 2. Consequently, for the *ogbn-arxiv* dataset, a random sample of 5% of the nodes is used for training, while the remaining half of the nodes are used for validation and testing, respectively.

I.3 Subgraph Partitioning

Inspired by real-world requirements and following [10], we consider two subgraph partitioning settings: non-overlapping and overlapping. In the non-overlapping setting, $\cup_{m=1}^M \mathcal{V}_m = \mathcal{V}$ and $\mathcal{V}_m \cap \mathcal{V}_n = \emptyset$ for $\forall m \neq n \in \{1, 2, \dots, M\}$, where \mathcal{V} represents the node set of the global graph. Partitioning without this property is called overlapping. Here we present the details of how to partition the original graph into multiple subgraphs. It should be noted that the number of subgraphs is equal to the number of clients. Both non-overlapping and overlapping subgraph partitioning settings are used in the experiments for all datasets.

Non-overlapping partitioning First, if there are M clients, the number of non-overlapping subgraphs to be generated is specified as M . Second, the METIS graph partitioning algorithm, as described in [11], is used to divide the original graph into M subgraphs. In other words, the non-overlapping partitioning subgraph for each client is directly obtained by the output of the METIS algorithm.

Overlapping partitioning First, if there are M clients, the number of overlapping subgraphs to be generated is specified as M . Second, the METIS [11] graph partitioning algorithm is used to divide

Algorithm 2: FedIIH Server Algorithm

Input: Number of rounds R ; number of clients M ; number of latent factors K ; parameters of two DisenGCNs $\mathbf{W}_m^{1:K}$ and $\mathbf{b}_m^{1:K}$ from client m ; parameters of the node classifier $\mathbf{W}_m^{\text{cls}}$ and $\mathbf{b}_m^{\text{cls}}$ from client m ; approximate posterior distributions $\tilde{\mathbf{H}}_m^{1:K}$ from client m .

Output: Federated model parameters for client m .

```
1 Initialize parameters  $(\bar{\mathbf{W}}^{\text{cls}})^{(1)}$ ,  $(\bar{\mathbf{b}}^{\text{cls}})^{(1)}$ ,  $(\bar{\mathbf{W}}^{1:K})^{(1)}$ , and  $(\bar{\mathbf{b}}^{1:K})^{(1)}$ ;
2 Initialize  $(\tilde{\alpha}^{1:K})^{(1)}$ ;
3 for each round  $r$  from 1 to  $R$  do
4   for client  $m \in \{1, 2, \dots, M\}$  in parallel do
5     if  $r = 1$  then
6       Send  $(\bar{\mathbf{W}}^{\text{cls}})^{(r)}$ ,  $(\bar{\mathbf{b}}^{\text{cls}})^{(r)}$ ,  $(\bar{\mathbf{W}}^{1:K})^{(r)}$ , and  $(\bar{\mathbf{b}}^{1:K})^{(r)}$  to client  $m$ ;
7       Send  $(\tilde{\alpha}^{1:K})^{(r)}$  to client  $m$ ;
8     end
9     else
10      Receive  $\mathbf{W}_m^{\text{cls}}$ ,  $\mathbf{b}_m^{\text{cls}}$ ,  $\mathbf{W}_m^{1:K}$ ,  $\mathbf{b}_m^{1:K}$ , and  $\tilde{\mathbf{H}}_m^{1:K}$  from client  $m$ ;
11      Compute  $S(m, j)^{1:K}$  via Eq. (9) in the main manuscript, where  $j \in \{1, 2, \dots, M\}$ ;
12      Compute  $\beta_{m,j}^{1:K}$  via Eq. (11) in the main manuscript, where  $j \in \{1, 2, \dots, M\}$ ;
13      Perform the separate federation to obtain  $(\bar{\mathbf{W}}_m^{\text{cls}})^{(r)}$ ,  $(\bar{\mathbf{W}}_m^{1:K})^{(r)}$ , and  $(\bar{\mathbf{b}}_m^{1:K})^{(r)}$  via
        Eq. (10) in the main manuscript;
14      Perform the FedAvg [8] to obtain  $(\bar{\mathbf{b}}_m^{\text{cls}})^{(r)}$ ;
15      Sample  $(\tilde{\alpha}^{1:K})^{(r)}$  from  $\mathcal{N}(\frac{\sum_{j=1}^M \hat{\mu}_{\tilde{\mathbf{H}}_j^{1:K}}}{M+0.25}, \mathbf{I})$ ;
16      Send  $(\bar{\mathbf{W}}_m^{\text{cls}})^{(r)}$ ,  $(\bar{\mathbf{b}}_m^{\text{cls}})^{(r)}$ ,  $(\bar{\mathbf{W}}_m^{1:K})^{(r)}$ , and  $(\bar{\mathbf{b}}_m^{1:K})^{(r)}$  to client  $m$ ;
17      Send  $(\tilde{\alpha}^{1:K})^{(r)}$  to client  $m$ ;
18    end
19    Perform Algorithm 1 on client  $m$ ;
20  end
21 end
```

Table 2: Statistical information of eleven used graph datasets.

Types	Datasets	# Nodes	# Edges	# Classes	# Node Features
homophilic graph	<i>Cora</i>	2,708	5,429	7	1,433
	<i>CiteSeer</i>	3,327	4,732	6	3,703
	<i>PubMed</i>	19,717	44,324	3	500
	<i>Amazon-Computer</i>	13,752	491,722	10	767
	<i>Amazon-Photo</i>	7,650	238,162	8	745
	<i>ogbn-arxiv</i>	169,343	2,315,598	40	128
heterophilic graph	<i>Roman-empire</i>	22,662	32,927	18	300
	<i>Amazon-ratings</i>	24,492	93,050	5	300
	<i>Minesweeper</i>	10,000	39,402	2	7
	<i>Tolokers</i>	11,758	519,000	2	10
	<i>Questions</i>	48,921	153,540	2	301

the original graph into $\lfloor \frac{M}{5} \rfloor$ subgraphs. Third, in each subgraph generated by METIS, half of the nodes and their associated edges are randomly sampled. This procedure is performed five times to generate five different yet overlapped subgraphs. By doing so, the number of overlapping subgraphs is equal to the number of clients.

I.4 Baseline Methods

We compare our proposed FedIIH with the following baseline methods, which can be categorized into two groups. The first group comprises general FL baseline methods, including FedAvg [8],

FedProx [12], and FedPer [13]. The second group consists of six GFL baseline methods, namely GCFL [14], FedGNN [15], FedSage+ [16], FED-PUB [10], FedGTA [17], and AdaFGL [18]. Moreover, we perform experiments with local training, that is, training each client without federated aggregation. The detailed descriptions of these baseline methods are provided below.

FedAvg This method [8] represents one of the fundamental baseline methods in the field of FL. First, each client independently trains a model, which is subsequently transmitted to a server. Then, the server aggregates the locally updated models by averaging and transmits the aggregated model back to the clients.

FedProx This method [12] is one of the personalized FL baseline methods. It customizes a personalized model for each client by adding a proximal term as a subproblem that minimizes weight differences between local and global models.

FedPer This method [13] is one of the personalized FL baseline methods. It only federates the weights of the backbone while training the personalized classification layer in each client.

GCFL This method [14] is one of the basic GFL methods. Specifically, GCFL is designed for vertical GFL (*e.g.*, GFL for molecular graphs). In particular, it uses the bi-partitioning scheme, which divides a set of clients into two disjoint groups of clients based on the similarity of their gradients. This is similar to the mechanism proposed for image classification in clustered-FL [19]. Then, after partitioning, the model weights are shared only among clustered clients with similar gradients.

FedGNN This method [15] is one of the GFL baseline methods. It extends local subgraphs by exchanging node embeddings from other clients. Specifically, if two nodes in two different clients have exactly the same neighbors, FedGNN transfers the nodes with the same neighbors from other clients and expands them.

FedSage+ This method [16] is one of the GFL baseline methods. It generates the missing edges between subgraphs and the corresponding neighbor nodes by using the missing neighbor generator. To train this neighbor generator, each client first receives node representations from other clients, and then computes the gradient of the distances between the local node features and the node representations of the other clients. After that, the gradient is sent back to the other clients, and this gradient is then used to train the neighbor generator.

FED-PUB This method [10] is one of the personalized GFL baseline methods. It estimates the similarities between the subgraphs based on the outputs of the local models that are given the same test graph. Then, based on the similarities, it performs a weighted averaging of the local models for each client. In addition, it learns a personalized sparse mask at each client in order to select and update only the subgraph-relevant subset of the aggregated parameters.

FedGTA This method [17] is one of the personalized GFL baseline methods. In this method, each client first computes topology-aware local smoothing confidence and mixed moments of neighbor features. They are then used to compute the inter-subgraph similarities, which are uploaded to the server along with the model parameters. Finally, the server is able to perform a weighted federation for each client.

AdaFGL This method [18] is one of the personalized GFL baseline methods. Actually, it is a decoupled two-step personalized approach. First, it uses standard multi-client federated collaborative training to acquire the federated knowledge extractor by aggregating uploaded models in the final round at the server. Second, each client performs personalized training based on the local subgraph and the federated knowledge extractor.

Local This method is the non-FL baseline, where the model is trained only locally for each client, with no weight sharing.

I.5 Training Details

Training rounds and epochs For the *Cora*, *CiteSeer*, *PubMed*, *Roman-empire*, *Amazon-ratings*, *Minesweeper*, *Tolokers*, and *Questions* datasets, we set the number of local training epochs and total rounds to 1 and 100, respectively. For larger datasets, such as *Amazon-Computer*, *Amazon-Photo*, and *ogbn-arxiv*, we set the number of total rounds to 200. Note that the number of local epochs is set to 2 for the *Amazon-Photo* and *ogbn-arxiv* datasets, and to 3 for the *Amazon-Computer* dataset. Finally, we report the test performance of all models at the best validation epoch, and the performance is measured by averaging across all clients in terms of node classification accuracies (or AUCs).

Hyperparameters We report the detailed hyperparameters used to train our proposed FedIIH in Tab. 3 and Tab. 4. These hyperparameters are determined by grid search. Note that we set the similarity scaling hyperparameter (*i.e.*, τ in Eq. (11) of the main manuscript) to 10 in both non-overlapping and overlapping subgraph partitioning scenarios according to the recommendation of FED-PUB [10]. We provide the detailed reasons in the Appendix K.7. The search range of hyperparameters is shown in Tab. 5. The code of our proposed FedIIH will be released on GitHub upon acceptance of the paper. The detailed hyperparameter sensitivity analysis can be found in the Appendix M.2.

Hyperparameter tuning guidelines There are four vital hyperparameters (*i.e.*, number of latent factors, number of neighborhood routing layers, number of neighborhood routing iterations, and τ) in our proposed FedIIH. First, the number of latent factors (*i.e.*, K) is related to whether the graph tends to be homophilic or heterophilic. If the graph dataset tends to be homophilic, a small K is recommended. If the graph dataset tends to be heterophilic, a large K is recommended. Second, the variations in performance under different numbers of neighborhood routing layers, different numbers of neighborhood routing iterations, and different values of τ are all small. Therefore, these three hyperparameters can be tuned by grid search.

Network architectures For the experiments of all baseline methods, except FedSage+, FedGTA, and our proposed FedIIH, we use two layers of the Graph Convolutional Network (GCN) [20] and a linear classifier layer as their network architectures. For the hyperparameter settings of the baseline methods, we use the default settings given in their original papers. Because of the inductive and scalability advantages of GraphSAGE [21], FedSage+ uses GraphSAGE as the encoder and then trains a missing neighbor generator to handle missing links across local subgraphs. For our proposed FedIIH, we use the node feature projection layer of DisenGCN [1] to obtain the node representations (see Fig. 1) and a linear classifier layer (*i.e.*, MLP) to perform node classifications. In contrast, FedGTA uses a Graph Attention Multi-Layer Perceptron (GAMLP) [22] as its backbone and a linear classifier layer to classify nodes. Note that GAMLP [22] is one of the scalable Graph Neural Network (GNN) models, which can capture the underlying correlations between different scales of graph knowledge.

I.6 Implementations of Two Prior Distributions

Here we describe the detailed implementations of $p(\tilde{\alpha}^k)$ and $p(\tilde{\mathbf{H}}_m^k | \tilde{\alpha}^k)$, respectively. Since $\tilde{\alpha}^k$ denotes the posterior mean of α^k for the k -th global latent factor, we assume that $p(\tilde{\alpha}^k) \sim \mathcal{N}(\tilde{\alpha}^k, \sigma_{\alpha^k}^2 \mathbf{I})$, where $\tilde{\alpha}^k$ is given in Eq. (12). Moreover, $\sigma_{\alpha^k}^2$ and $\sigma_{\mathbf{H}_m^k}^2$ is set to 1 and 0.25, respectively. In other words, $p(\tilde{\alpha}^k) \sim \mathcal{N}(\frac{\sum_{m=1}^M \hat{\mu}_{\mathbf{H}_m^k}}{M+0.25}, \mathbf{I})$. According to Tab. 1, $p(\tilde{\mathbf{H}}_m^k | \alpha^k) \sim \mathcal{N}(\alpha^k, \sigma_{\mathbf{H}_m^k}^2 \mathbf{I})$. Consequently, we can have

$$\begin{aligned} p(\tilde{\mathbf{H}}_m^k | \tilde{\alpha}^k) &\sim \mathcal{N}(\tilde{\alpha}^k, 0.25\mathbf{I}) \\ &\sim \mathcal{N}\left(\frac{\sum_{m=1}^M \hat{\mu}_{\mathbf{H}_m^k}}{M+0.25}, 0.25\mathbf{I}\right). \end{aligned} \quad (13)$$

J Additional Experiments

In this section, we provide the additional experiments related to the ablation studies and hyperparameter analysis, respectively.

Table 3: Hyperparameters used in our proposed FedIIH on the **homophilic** graph datasets in both the non-overlapping and overlapping subgraph partitioning settings.

	Cora		CiteSeer		PubMed	
Hyperparameters	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
# latent factors	2	2	2	2	4	2
learning rate	0.02	0.01	0.01	0.01	0.01	0.015
# hidden dimensions	128	256	256	256	256	256
dropout rate	0.3	0.35	0.35	0.35	0.25	0.4
weight decay	0.005	1e-6	1e-6	1e-6	0.0045	1e-6
# neighborhood routing layers	4	5	5	5	1	1
# neighborhood routing iterations	6	6	6	6	6	6
	Amazon-Computer		Amazon-Photo		ogbn-arxiv	
Hyperparameters	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
# latent factors	6	4	6	10	2	2
learning rate	0.015	0.015	0.015	0.01	0.01	0.01
# hidden dimensions	128	128	256	128	128	128
dropout rate	0.4	0.35	0.4	0.35	0.35	0.35
weight decay	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6
# neighborhood routing layers	1	5	1	1	5	5
# neighborhood routing iterations	6	6	6	5	6	6

Table 4: Hyperparameters used in our proposed FedIIH on the **heterophilic** graph datasets in both the non-overlapping and overlapping subgraph partitioning settings.

	Roman-empire		Amazon-ratings		Minesweeper	
Hyperparameters	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
# latent factors	4	4	4	4	6	4
learning rate	0.015	0.015	0.01	0.01	0.01	0.01
# hidden dimensions	128	128	128	256	256	128
dropout rate	0.35	0.35	0.35	0.35	0.35	0.35
weight decay	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6
# neighborhood routing layers	1	1	3	5	5	6
# neighborhood routing iterations	6	6	7	6	6	7
	Tolokers		Questions			
Hyperparameters	non-overlapping	overlapping	non-overlapping	overlapping		
# latent factors	10	10	8	2		
learning rate	0.01	0.01	0.01	0.01		
# hidden dimensions	128	128	256	256		
dropout rate	0.35	0.35	0.35	0.35		
weight decay	0.0045	0.0045	1e-6	1e-6		
# neighborhood routing layers	1	1	5	5		
# neighborhood routing iterations	2	2	6	6		

J.1 Ablation Studies on Other Datasets

To further analyze the contribution of each component, we conduct ablation studies on the remaining datasets in both non-overlapping and overlapping partitioning settings with 10 clients. As shown in Tab. 6, we can find that the performance of FedIIH is significantly better than the three variants. It validates that each component indeed contributes a lot to the final performance.

J.2 Hyperparameter Sensitivity Analysis of K

The hyperparameter sensitivity analysis of K on the remaining datasets are shown in Fig. 5, Fig. 6, ..., to Fig. 13. According to the experimental results, we have the following observations and insights: 1) In general, the performance variation under different K is small except for the *Roman-empire* dataset (see Fig. 10). The *Roman-empire* dataset has low homophily [9], which is based on the Roman Empire article from the English Wikipedia. Each node in the graph corresponds to a (non-unique) word in the text, and the node label is determined by the syntactic role of the word. Due to the syntactic dependencies within neighboring words, there exists strong heterogeneity within the *Roman-empire* graph. Consequently, if we ignore this intra-heterogeneity (*i.e.*, without disentangling), the performance will decrease significantly on the *Roman-empire* dataset. This is consistent with

Table 5: The search range of the hyperparameters used in our proposed FedIIH.

Hyperparameters	# latent factors	learning rate	# hidden dimensions	dropout rate
Range	{1, 2, 4, 6, 8, 10}	{0.01, 0.015, 0.02}	{128, 256}	[0.25, 0.4]
Hyperparameters	weight decay	# neighborhood routing layers	# neighborhood routing iterations	
Range	[1e-6, 5e-3]	{1, 2, 3, 4, 5, 6}	{2, 3, 4, 5, 6, 7}	

Table 6: Ablation studies in both non-overlapping and overlapping partitioning settings on other datasets with 10 clients.

	CiteSeer		PubMed		Amazon-Computer	
Methods	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
FedIIH (w/o HM)	74.91±0.27 (↓1.59)	72.29±0.16 (↓0.87)	85.19±0.05 (↓2.46)	85.16±0.17 (↓0.71)	85.75±0.90 (↓5.11)	87.71±0.20 (↓2.44)
FedIIH (w/o VI)	72.27±2.16 (↓4.23)	72.60±0.16 (↓0.56)	81.35±0.20 (↓6.30)	84.56±0.04 (↓1.31)	69.16±1.15 (↓21.70)	74.90±1.24 (↓15.25)
FedIIH (w/o Dis)	75.71±0.38 (↓0.79)	71.54±0.12 (↓1.62)	85.71±0.12 (↓1.94)	84.30±0.03 (↓1.57)	88.96±0.08 (↓1.90)	88.51±0.04 (↓1.64)
FedIIH	76.50±0.06	73.16±0.18	87.65±0.18	85.87±0.03	90.86±0.23	90.15±0.04
	Amazon-Photo		ogbn-arxiv		Roman-empire	
Methods	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
FedIIH (w/o HM)	91.84±0.05 (↓2.38)	91.88±0.27 (↓1.50)	65.18±0.41 (↓4.16)	63.54±0.15 (↓3.15)	56.28±0.20 (↓10.16)	60.90±0.21 (↓4.58)
FedIIH (w/o VI)	79.25±0.96 (↓14.97)	82.16±0.56 (↓11.22)	49.26±0.74 (↓20.08)	46.20±1.73 (↓20.49)	57.38±0.18 (↓9.06)	61.69±0.09 (↓13.79)
FedIIH (w/o Dis)	92.43±0.01 (↓1.79)	92.01±0.04 (↓1.37)	61.51±0.15 (↓7.83)	60.64±0.16 (↓6.05)	40.51±0.27 (↓25.93)	42.84±0.09 (↓22.64)
FedIIH	94.22±0.08	93.38±0.00	69.34±0.02	66.69±0.09	66.44±0.28	65.48±0.12
	Minesweeper		Tolokers		Questions	
Methods	non-overlapping	overlapping	non-overlapping	overlapping	non-overlapping	overlapping
FedIIH (w/o HM)	70.88±0.01 (↓2.35)	66.56±0.07 (↓2.79)	64.56±0.17 (↓6.76)	69.21±0.22 (↓2.46)	65.90±0.09 (↓2.09)	67.77±0.10 (↓1.02)
FedIIH (w/o VI)	71.34±0.09 (↓1.89)	68.47±0.06 (↓0.88)	64.34±0.12 (↓6.98)	68.03±0.34 (↓3.64)	66.63±0.08 (↓1.36)	68.24±0.06 (↓0.55)
FedIIH (w/o Dis)	70.67±0.02 (↓2.56)	68.75±0.19 (↓0.60)	62.53±0.29 (↓8.79)	68.10±0.12 (↓3.57)	66.42±0.01 (↓1.57)	67.52±0.17 (↓1.27)
FedIIH	73.23±0.04	69.35±0.25	71.32±0.09	71.67±0.02	67.99±0.09	68.79±0.09

the experimental results in Tab. 3 and Tab. 4 in the main manuscript, where our FedIIH outperforms other methods by a large margin. 2) As the value of K increases, the performance may increase or decrease depending on the datasets. For example, on the *Roman-empire* (see Fig. 10) and *Tolokers* (see Fig. 4 in the main manuscript) datasets, their performances increase consistently as the value of K increases. On the contrary, on the *Cora* dataset (see Fig. 4), the accuracy reaches its highest value when $K = 2$, and then it decreases as the value of K is further increased.

K Discussions

K.1 Similarity Heatmaps of FED-PUB and FedIIH over Three Independent Runs

As shown in Fig. 14, Fig. 15, Fig. 16, and Fig. 17, we present the similarity heatmaps of FED-PUB and our FedIIH over three independent runs on the *Cora* and *Amazon-ratings* datasets in the overlapping setting with 20 clients, respectively. We can find that our calculated similarities are fairly much more stable than the similarities calculated by FED-PUB. This is because FED-PUB estimates the similarities between subgraphs based on the outputs of local models given the same random test graph. Since the random test graph varies over three independent runs, the outputs of the local models also change. In contrast, our FedIIH successfully infers the whole distribution of subgraph data in a multi-level global perspective, such that we can stably characterize the inter-subgraph similarities. Note that the similarity ground truth of the *Cora* and *Amazon-ratings* datasets in the overlapping setting with 20 clients are presented in Fig. 18a and Fig. 3a of the main manuscript, respectively.

K.2 Similarity Heatmaps on Other Datasets

The similarity heatmaps on other datasets are shown in Fig. 18, Fig. 19, ..., to Fig. 38. We present the similarity heatmaps for each dataset (except the *Cora* dataset) in two settings, namely, non-overlapping with 20 clients and overlapping with 30 clients. However, for the *Cora* dataset, we present the similarity heatmaps in the overlapping setting with 20 clients. This is because in [10], Baek *et al.* present the heatmaps on the *Cora* dataset in the overlapping setting with 20 clients, and we specifically want to be consistent with that here. According to the experimental results on these datasets, we can find that the similarity heatmaps of our FedIIH are always much closer to the ground truth than FED-PUB and FedGTA, verifying the effectiveness of our similarity calculation scheme based on the inferred subgraph data distribution.

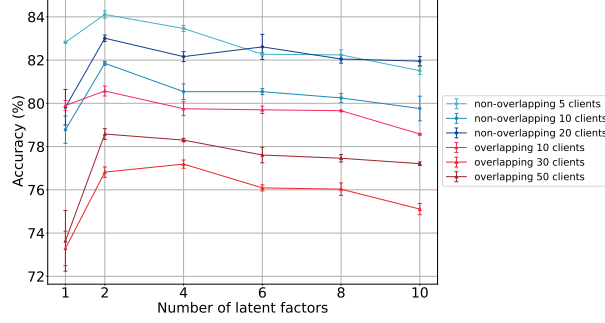


Figure 4: Sensitivity of the number of latent factors K on the *Cora* dataset.

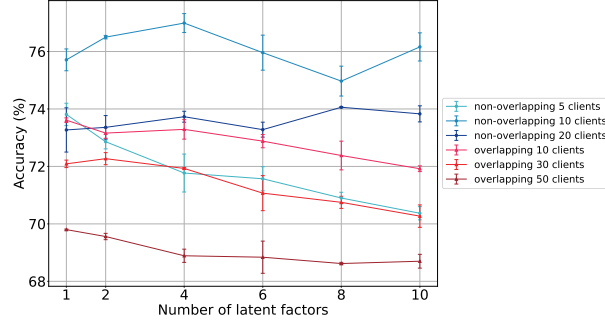


Figure 5: Sensitivity of the number of latent factors K on the *CiteSeer* dataset.

K.3 Case Study of Different Latent Factors

Since our proposed FedIIH disentangles the subgraph into several latent factors, one may ask what is the difference between the similarity heatmaps under different latent factors. To illustrate this, we take the *Roman-empire* dataset as an example to perform a case study. As demonstrated in Fig. 10, the accuracies of our FedIIH obviously increase as K changes from 1, to 2, to 4. Consequently, in Fig. 39, we present the similarity heatmaps of FedIIH on the *Roman-empire* dataset when K is set to 1, 2, and 4, respectively. From Fig. 39, it can be observed that there are indeed differences between different similarity heatmaps. Although these differences may seem trivial, they have a very large impact on the separate federation and thus on the final performance.

K.4 Disentangled Latent Factors

Here we provide some insights into the interpretability of the disentangled latent factors. Disentangled latent factors, as widely explored in [1, 23, 24], are used to explore the reasons why a node is connected to others. In other words, the interpretability of the disentangled latent factors can be considered as the relationship from a given node to one of its neighbors. For example, a user in a social graph is connected to others for various different reasons, such as families, hobbies, studies, and work. Each disentangled latent factor is capable of capturing mutually exclusive information. For example, the correlation plot of DisenGCN on the eight-factor synthetic graph dataset (Figure 3 in [1]) clearly shows eight diagonal blocks, verifying that the latent factors have indeed been successfully disentangled.

One might ask: Is it possible that the parameter positions corresponding to the same disentangled latent factor differ between clients? We would like to clarify that the parameter positions corresponding to the same latent factor usually remain the same in the disentangled GNNs of different clients. Taking the DisenGCN as an example, there are two crucial processes: node feature projection (Eq. (1) in the main manuscript) and neighborhood routing mechanism (Eq. (1) and Eq. (2) in the Appendix A). According to Eq. (1) in the main manuscript, we can find that the position of parameters corresponding to the k -th latent factor (*i.e.*, \mathbf{W}^k) is determined and then fixed by the node feature \mathbf{x}^i . Since the distributions of \mathbf{x}^i are similar in different clients, the parameter positions corresponding to K latent factors on each client are determined and then fixed in the same way. Moreover, the parameter

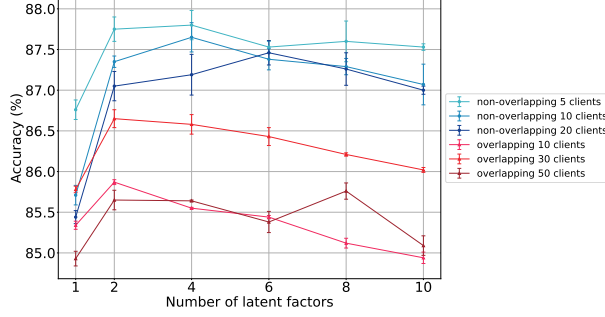


Figure 6: Sensitivity of the number of latent factors K on the *PubMed* dataset.

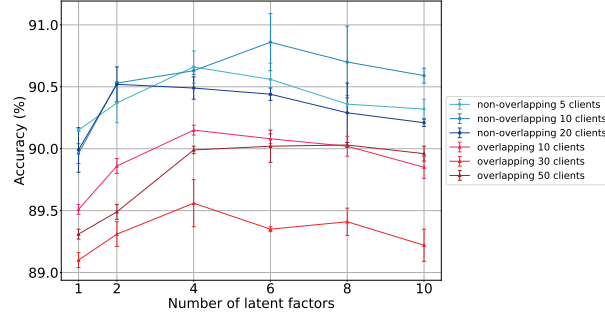


Figure 7: Sensitivity of the number of latent factors K on the *Amazon-Computer* dataset.

positions corresponding to K latent factors are not changed by the neighborhood routing mechanism, because there are no learnable parameters in the neighborhood routing mechanism. Therefore, the parameter positions corresponding to the same latent factor usually remain the same. This can be verified by the experiments, where the standard deviations of our FedIIH are quite small on different datasets (see Tables 1, 2, 3, and 4 in the main manuscript).

K.5 Disentangled Graph Neural Networks

The current implementation of our proposed FedIIH is based on the existing method DisenGCN [1], which is used to instantiate the inference network in our HVGAE. Instantiation in variational inference is very common in many existing approaches [25, 26, 27]. For example, [25] and [26] use the GCN and Graph Attention neTworks (GAT) to instantiate their inference networks, respectively.

Although many disentangled graph neural networks [1, 23, 24] can be chosen flexibly, we directly choose a simple but popular model (*i.e.*, DisenGCN). This is because DisenGCN is a pioneering work in the field. Moreover, our proposed FedIIH is flexible since other disentangled graph neural networks [23, 24] can easily be used.

K.6 Differences Between FED-PUB and Our FedIIH

First, in FED-PUB [10], each client simply feeds the randomly generated graph to the local model and sends the output to the server. In stark contrast, our FedIIH differs noticeably in that it uses HVGAE to disentangle the subgraph into multiple latent factors and accurately infer the distribution of the subgraph data. Then, each client sends this inferred data distribution to the server.

Second, in FED-PUB, the server only measures the similarities of the clients by computing the cosine similarities of the outputs of local models. Conversely, in our FedIIH, the server specifically measures the similarities by computing the JS divergences of the inferred subgraph data distributions, therefore providing a more accurate and stable measure of client similarities.

In summary, our FedIIH differs noticeably from FED-PUB, as FED-PUB mainly focuses on computing the cosine similarities based on the outputs of local models. Unlike FED-PUB, we measure the similarities of clients by computing the JS divergences of the inferred subgraph data distribution.

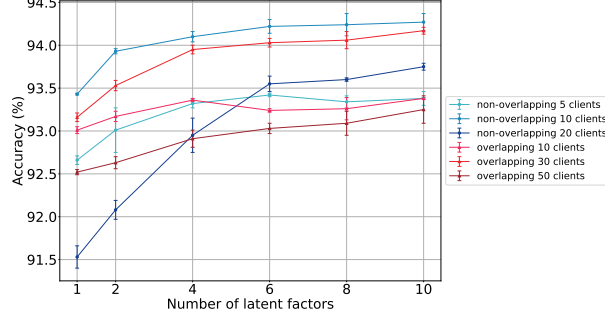


Figure 8: Sensitivity of the number of latent factors K on the *Amazon-Photo* dataset.

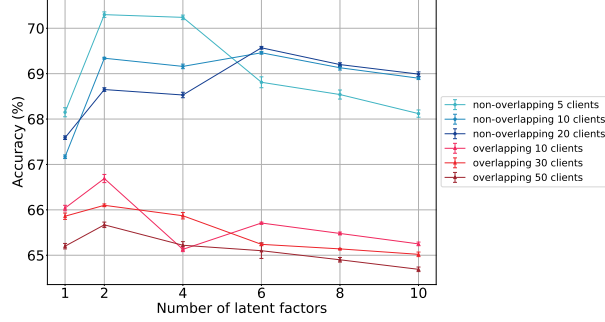


Figure 9: Sensitivity of the number of latent factors K on the *ogbn-arxiv* dataset.

K.7 Reasons for Setting τ to 10

On one hand, τ is recommended by FED-PUB [10] to be set to 10. On the other hand, according to the experimental results of grid search, the performance can achieve the best result when τ is around 10. Tab. 7 provides the accuracies when using different values of τ .

K.8 Why do local model outputs not accurately reflect the distribution of subgraph?

Most existing methods [10, 17, 28] compute the inter-subgraph similarities based on the simplex outputs of local models. However, we argue that the outputs of local models cannot accurately reveal the whole distribution of subgraph data. Here we provide the reason.

According to the universal approximation theorem of neural networks [29], even if two neural networks (*e.g.*, two local models on different clients) have different inputs, they can still produce the completely same output. Therefore, the outputs of local models cannot accurately reveal the whole distribution of subgraph data. This can be verified by experiments. In Fig. 3 of the main manuscript, the similarity heatmap of FED-PUB (Fig. 3b) is quite different from the ground truth (Fig. 3a), which means that the calculated similarities based on the local model outputs cannot accurately reflect the overall distribution of subgraph data.

L Efficiency Analysis

In this section, we present the spatial and temporal complexity of our proposed FedIIH and that of two baseline methods (*i.e.*, FedAvg and FED-PUB) on the client and server sides, respectively. Furthermore, we also compare the training time of our proposed FedIIH and that of two baseline methods (*i.e.*, FedAvg and FED-PUB).

L.1 Spatial Complexity

First, the spatial complexity of our FedIIH on each client side and that of two baseline methods (*i.e.*, FedAvg and FED-PUB) are both $\mathcal{O}(n_m \times d + L \times d^2)$, where n_m , d , and L denote the node

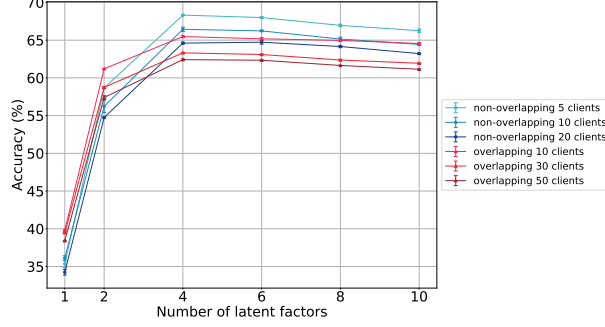


Figure 10: Sensitivity of the number of latent factors K on the *Roman-empire* dataset.

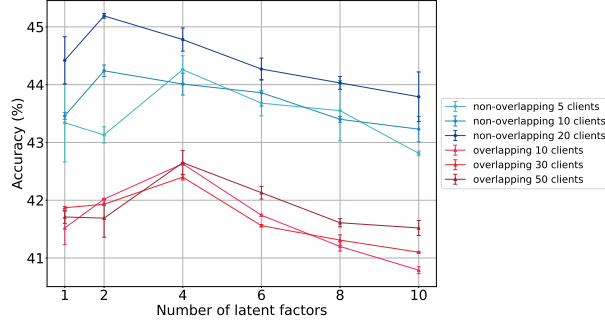


Figure 11: Sensitivity of the number of latent factors K on the *Amazon-ratings* dataset.

number, feature dimensionality, and layer number, respectively. Therefore, the spatial complexity of our FedIIH on each client is the same as that of two baseline methods.

Second, the spatial complexity of our method on the server side is $\mathcal{O}(M \times d \times (L \times d^2 + K \times M))$, where M and K denote the number of clients and the number of disentangled latent factors, respectively. The spatial complexity of the baseline method (*i.e.*, FED-PUB) on each client is $\mathcal{O}(M \times d \times (L \times d^2 + M))$. By comparing the spatial complexity of our FedIIH with that of the baseline method (*i.e.*, FED-PUB), we find that the only difference is that the third factor in FedIIH's spatial complexity is $L \times d^2 + K \times M$ while the third factor in FED-PUB's spatial complexity is $L \times d^2 + M$. Since $K \leq 10$ (as mentioned in the Appendix I.5), we can find that the spatial complexity of our FedIIH on the server side is similar to that of the baseline method.

L.2 Temporal Complexity

First, the temporal complexity of HVGAE on each client is $\mathcal{O}(n_m \times d \times (L + n_m + d))$. Since the model deployed on each client is actually a HVGAE, the total temporal complexity of our FedIIH on each client is $\mathcal{O}(n_m \times d \times (L + n_m + d))$. The temporal complexity of two baseline methods (*i.e.*, FedAvg and FED-PUB) on each client is $\mathcal{O}(n_m \times d \times (L + L \times d + d))$. By comparing the temporal complexity of our FedIIH with that of two baseline methods (*i.e.*, FedAvg and FED-PUB), the only difference is that the third factor in FedIIH's temporal complexity is $L + n_m + d$ while the third factor in their temporal complexity is $L + L \times d + d$. Since $n_m \leq L \times d$ in most situations (can be verified in the Appendix I.5), we can find that the temporal complexity of our FedIIH on the client side is similar to that of two baseline methods (*i.e.*, FedAvg and FED-PUB). Therefore, we can find that the introduction of HVGAE does not increase too much computational overhead of our FedIIH.

Second, the temporal complexity of the divergence computation on the server is $\mathcal{O}(K \times M^2 \times d)$, and the total temporal complexity of our FedIIH on the server side is $\mathcal{O}(M \times d \times (K \times M + L \times d))$. The temporal complexity of the baseline method (*i.e.*, FED-PUB) on the server side is $\mathcal{O}(M \times d \times (M + L \times d))$. By comparing the temporal complexity of our FedIIH with that of the baseline method (*i.e.*, FED-PUB), we find that the only difference is that the third factor in FedIIH's temporal complexity is $K \times M + L \times d$ while the third factor in FED-PUB's temporal complexity is $M + L \times d$. Since

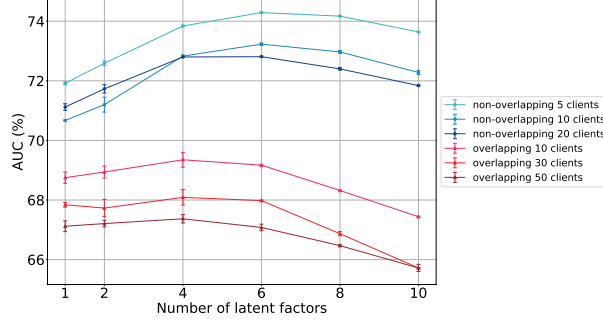


Figure 12: Sensitivity of the number of latent factors K on the *Minesweeper* dataset.

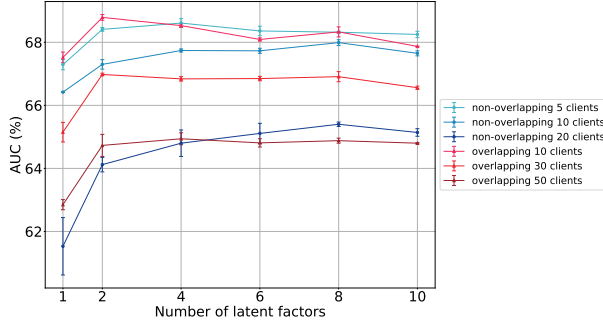


Figure 13: Sensitivity of the number of latent factors K on the *Questions* dataset.

$K \leq 10$ (as mentioned in the Appendix I.5), we can find that the temporal complexity of our FedIIH on the server side is similar to that of the baseline method (*i.e.*, FED-PUB).

L.3 Training Time

We report the training time of one communication round of our FedIIH and two baseline methods (*i.e.*, FedAvg and FED-PUB). The experimental platform is shown in the Appendix I.1. We conduct experiments on a homophilic graph dataset (*i.e.*, *Cora*) and a heterophilic graph dataset (*i.e.*, *Roman-empire*). As shown in Tab. 8, we can find that our FedIIH is more efficient in practice than the baseline methods (*i.e.*, FedAvg and FED-PUB).

M Robustness Analysis

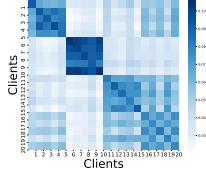
In this section, we analyze the robustness of our FedIIH in terms of client sparsity and hyperparameter sensitivity, respectively.

M.1 Client Sparsity Analysis

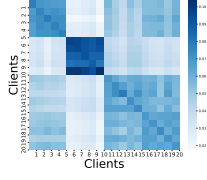
Robustness to client sparsity is a critical aspect in evaluating the effectiveness of a similarity-based personalized federated optimization strategy. Here we conduct experiments on a homophilic graph dataset (*i.e.*, *Cora*) and a heterophilic graph dataset (*i.e.*, *Roman-empire*) as the percentage of participating clients increases from 10% to 100%. As shown in Fig. 40, the experimental results clearly show that the performance of our FedIIH is more robust and stable than baseline methods, and is not too affected when the number of participating clients decreases.

M.2 Hyperparameter Sensitivity Analysis

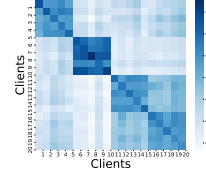
There are four vital hyperparameters (*i.e.*, number of latent factors, number of neighborhood routing layers, number of neighborhood routing iterations, and τ) in our proposed FedIIH. Here we perform experiments to analyze the hyperparameter sensitivity.



(a) the first independent run

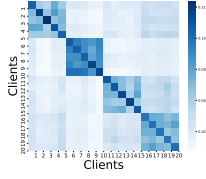


(b) the second independent run

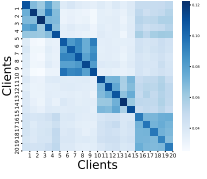


(c) the third independent run

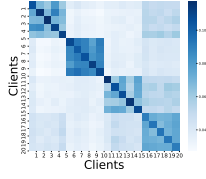
Figure 14: The similarity heatmaps of FED-PUB over three independent runs on the *Cora* dataset in the overlapping setting with 20 clients.



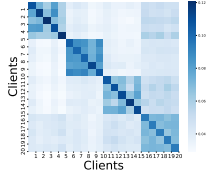
(a) the first independent run of the 1st latent factor ($K = 2$)



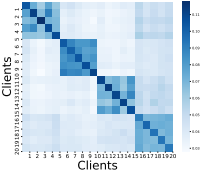
(b) the second independent run of the 1st latent factor ($K = 2$)



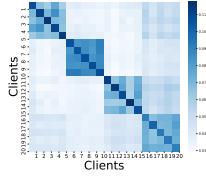
(c) the third independent run of the 1st latent factor ($K = 2$)



(d) the first independent run of the 2nd latent factor ($K = 2$)



(e) the second independent run of the 2nd latent factor ($K = 2$)



(f) the third independent run of the 2nd latent factor ($K = 2$)

Figure 15: The similarity heatmaps of FedIIH over three independent runs on the *Cora* dataset in the overlapping setting with 20 clients.

First, as shown in the Appendix J.2, the variation in performance under different numbers of latent factors (*i.e.*, K) is small. Second, as shown in Tab. 9, Tab. 10, and Tab. 11, the variations in performance under different numbers of neighborhood routing layers, different numbers of neighborhood routing iterations, and different values of τ are all small. These experimental results clearly demonstrate that the performances of our proposed FedIIH are very stable within a given range of hyperparameters, therefore the hyperparameter of our FedIIH can be easily tuned in practical use.

N Broader Impact

Our work could have the following positive impacts: (1) We provide a new method for GFL to deal with the inter-intra heterogeneity. (2) Our proposed method can greatly improve the performance of GFL on homophilic and heterophilic graph datasets in both non-overlapping and overlapping settings.

The proposed method can be used for both good and bad, similar to many other FL methods. Note that most existing methods and our proposal are not immune to such misuse. We believe that such a problem can be solved in the future, although we do not have an optimal solution.

In summary, we believe that our proposed method can benefit society because many important real-world graphs face inter-intra heterogeneity when performing the GFL. Therefore, they can benefit from our proposed FedIIH.

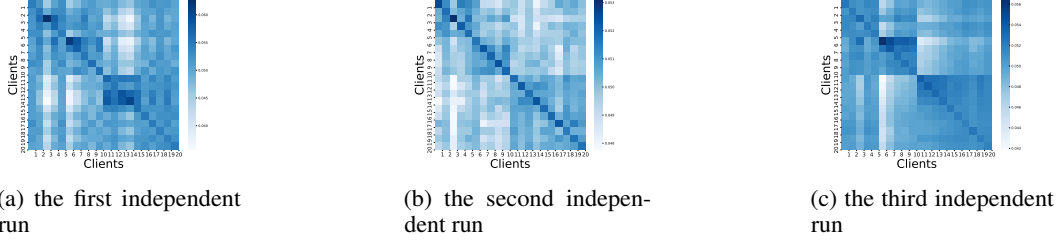


Figure 16: The similarity heatmaps of FED-PUB over three independent runs on the *Amazon-ratings* dataset in the overlapping setting with 20 clients.

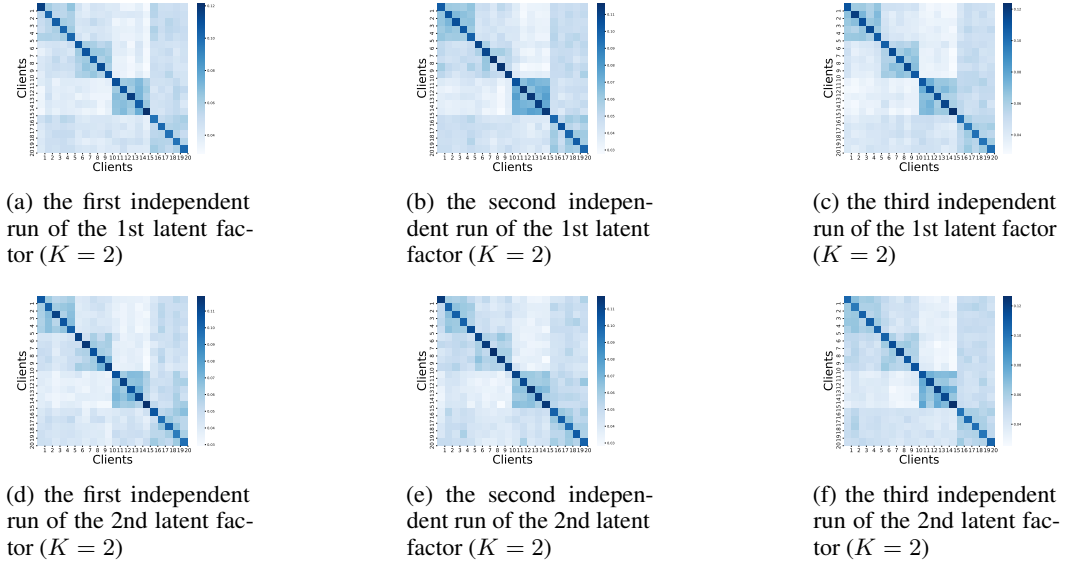


Figure 17: The similarity heatmaps of FedIIH over three independent runs on the *Amazon-ratings* dataset in the overlapping setting with 20 clients.

References

- [1] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *International Conference on Machine Learning*, pages 4212–4221, 2019.
- [2] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, pages 1–12, 2017.
- [3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.
- [4] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *International Conference on Learning Representations*, pages 1–23, 2021.
- [5] Xu Zhang, Yinchuan Li, Wenpeng Li, Kaiyang Guo, and Yunfeng Shao. Personalized federated learning via variational Bayesian inference. In *International Conference on Machine Learning*, pages 26293–26310, 2022.
- [6] Minyoung Kim and Timothy Hospedales. FedHB: Hierarchical Bayesian federated learning. *arXiv:2305.04979*, 2023.

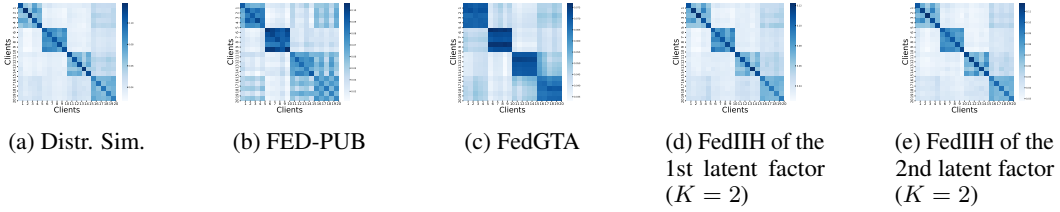


Figure 18: Similarity heatmaps on the *Cora* dataset in the overlapping setting with 20 clients.

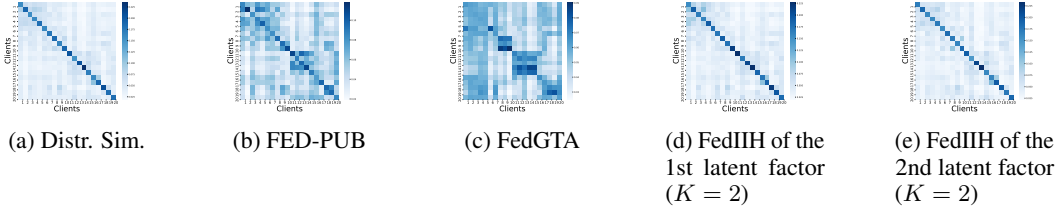


Figure 19: Similarity heatmaps on the *CiteSeer* dataset in the non-overlapping setting with 20 clients.

- [7] Liangxi Liu, Xi Jiang, Feng Zheng, Hong Chen, Guo-Jun Qi, Heng Huang, and Ling Shao. A Bayesian federated learning framework with online Laplace approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2023.
- [8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.
- [9] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, pages 1–15, 2023.
- [10] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *International Conference on Machine Learning*, pages 1396–1415, 2023.
- [11] George Karypis. METIS: Unstructured graph partitioning and sparse matrix ordering system. *Technical report*, 1997.
- [12] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, pages 429–450, 2020.
- [13] Manoj Ghuhana Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv:1912.00818*, 2019.
- [14] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. In *Advances in Neural Information Processing Systems*, pages 18839–18852, 2021.
- [15] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. FedGNN: Federated graph neural network for privacy-preserving recommendation. *arXiv:2102.04925*, 2021.
- [16] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. In *Advances in Neural Information Processing Systems*, pages 6671–6682, 2021.
- [17] Xunkai Li, Zhengyu Wu, Wentao Zhang, Yinlin Zhu, Rong-Hua Li, and Guoren Wang. FedGTA: Topology-aware averaging for federated graph learning. In *Proceedings of the VLDB Endowment*, pages 41–50, 2023.



Figure 20: Similarity heatmaps on the *CiteSeer* dataset in the overlapping setting with 30 clients.

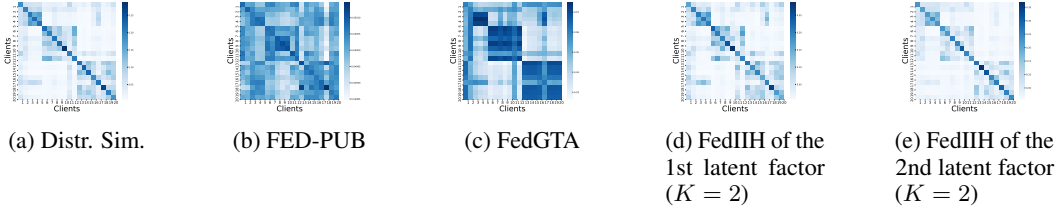


Figure 21: Similarity heatmaps on the *PubMed* dataset in the non-overlapping setting with 20 clients.

- [18] Xunkai Li, Zhengyu Wu, Wentao Zhang, Henan Sun, Rong-Hua Li, and Guoren Wang. AdaFGL: A new paradigm for federated node classification with topology heterogeneity. In *International Conference on Data Engineering*, pages 2517–2530, 2024.
- [19] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021.
- [20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, pages 1–14, 2017.
- [21] Will Hamilton, Zhitaoying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1–11, 2017.
- [22] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. Graph attention multi-layer perceptron. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4560–4570, 2022.
- [23] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. In *Advances in Neural Information Processing Systems*, pages 21872–21884, 2021.
- [24] Jingwei Guo, Kaizhu Huang, Xinping Yi, and Rui Zhang. Learning disentangled graph convolutional networks locally and globally. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3):3640–3651, 2024.
- [25] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1–10, 2019.
- [26] Haibo Wang, Chuan Zhou, Xin Chen, Jia Wu, Shirui Pan, and Jilong Wang. Graph stochastic neural networks for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1–10, 2020.
- [27] Sheng Wan, Yibing Zhan, Liu Liu, Baosheng Yu, Shirui Pan, and Chen Gong. Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels. In *Advances in Neural Information Processing Systems*, pages 1–12, 2021.
- [28] Zaixi Zhang, Qingyong Hu, Yang Yu, Weibo Gao, and Qi Liu. FedGT: Federated node classification with scalable graph transformer. *arXiv:2401.15203*, 2024.

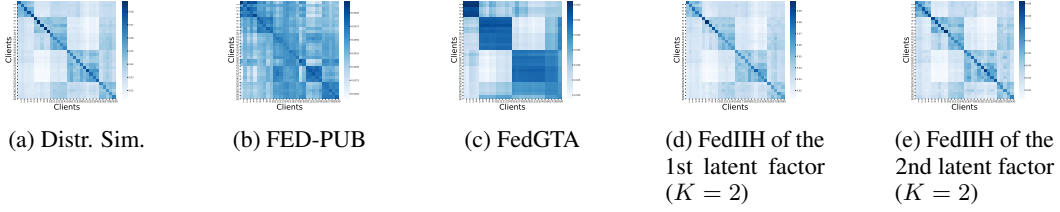


Figure 22: Similarity heatmaps on the *PubMed* dataset in the overlapping setting with 30 clients.

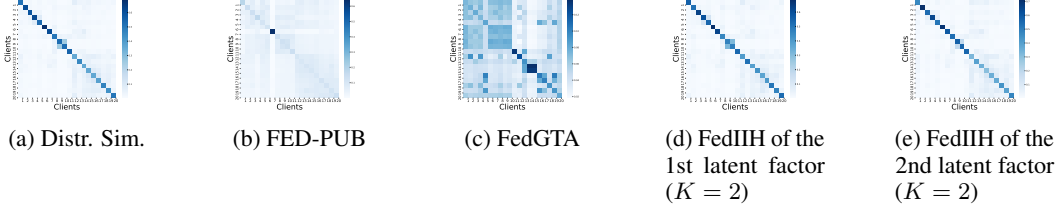


Figure 23: Similarity heatmaps on the *Amazon-Computer* dataset in the non-overlapping setting with 20 clients.

- [29] Rickard Br  l Gabrielsson. Universal function approximation on graphs. In *Advances in Neural Information Processing Systems*, pages 1–11, 2020.

Reproducibility Checklist

1. This paper includes a conceptual outline and/or pseudocode description of AI methods introduced. (yes/partial/no/NA)

Answer: [Yes]

Justification: Our paper includes a conceptual outline and pseudocode description of methods in this Appendix.

2. This paper clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results. (yes/no)

Answer: [Yes]

Justification: First, all the formulas and proofs in the paper are numbered and cross-referenced. Second, due to the space limitation, the detailed proof of ELBO is presented in the Appendix C. Third, we provide the extensive experimental results.

3. This paper provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper. (yes/no)

Answer: [Yes]

Justification: We provide the detailed preliminaries.

4. Does this paper make theoretical contributions? (yes/no)

Answer: [Yes]

Justification: This paper make theoretical contributions related to the ELBO.

5. All assumptions and restrictions are stated clearly and formally. (yes/partial/no)

Answer: [Yes]

Justification: This paper clearly and formally states all assumptions and restrictions.

6. All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no)

Answer: [Yes]

Justification: This paper clearly and formally states all all novel claims.

7. Proofs of all novel claims are included. (yes/partial/no)

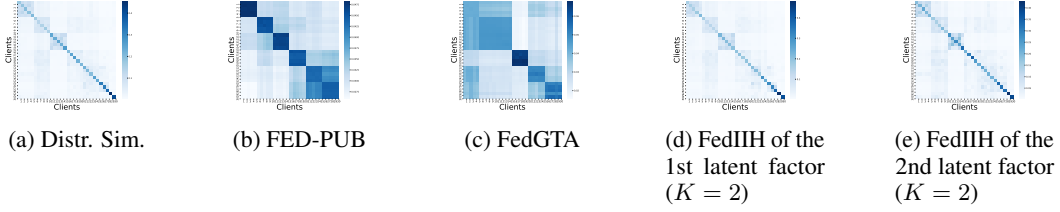


Figure 24: Similarity heatmaps on the *Amazon-Computer* dataset in the overlapping setting with 30 clients.

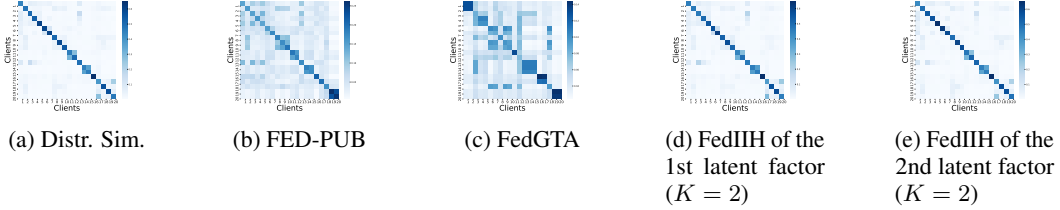


Figure 25: Similarity heatmaps on the *Amazon-Photo* dataset in the non-overlapping setting with 20 clients.

Answer: [Yes]

Justification: We provide the proofs in this Appendix.

8. Proof sketches or intuitions are given for complex and/or novel results. (yes/partial/no)

Answer: [Yes]

Justification: We provide the proofs in this Appendix.

9. Appropriate citations to theoretical tools used are given. (yes/partial/no)

Answer: [Yes]

Justification: We provide all the citations.

10. All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA)

Answer: [Yes]

Justification: The experimental results verify all theoretical claims.

11. All experimental code used to eliminate or disprove claims is included. (yes/no/NA)

Answer: [Yes]

Justification: Our code is available at <https://github.com/blgpb/FedIIH>. We also provide a README.md file to describe the detailed instructions on how to replicate the results.

12. Does this paper rely on one or more datasets? (yes/no)

Answer: [Yes]

Justification: We use several datasets.

13. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA)

Answer: [Yes]

Justification: We explain it in the paper.

14. All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA)

Answer: [Yes]

Justification: We provide them in this Appendix.

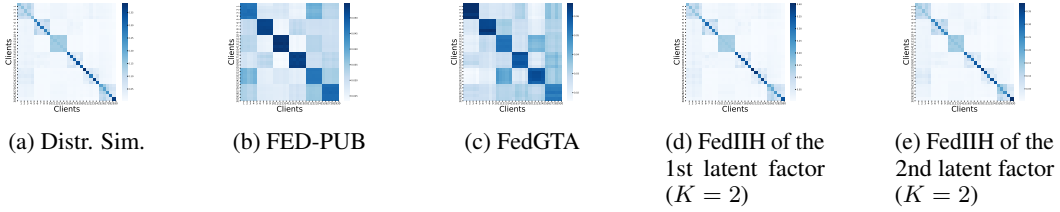


Figure 26: Similarity heatmaps on the *Amazon-Photo* dataset in the overlapping setting with 30 clients.



Figure 27: Similarity heatmaps on the *ogbn-arxiv* dataset in the non-overlapping setting with 20 clients.

15. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no/NA)

Answer: [Yes]

Justification: We provide them in this Appendix.

16. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes/no/NA)

Answer: [Yes]

Justification: We cite all the papers that produced the datasets.

17. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes/partial/no/NA)

Answer: [Yes]

Justification: All datasets are publicly available.

18. Does this paper include computational experiments? (yes/no)

Answer: [Yes]

Justification: We provide them in the paper.

19. Any code required for pre-processing data is included in the appendix. (yes/partial/no).

Answer: [Yes]

Justification: Our code is available at <https://github.com/blgpb/FedIIH>.

20. All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no)

Answer: [Yes]

Justification: Our code is available at <https://github.com/blgpb/FedIIH>.

21. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no)

Answer: [Yes]

Justification: All source code will be made publicly available upon publication of the paper with a license that allows free usage for research purposes.

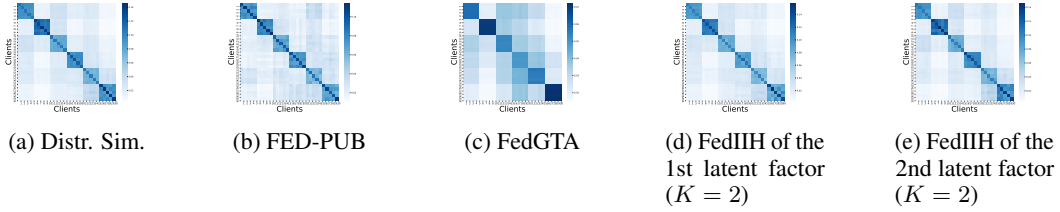


Figure 28: Similarity heatmaps on the *ogbn-arxiv* dataset in the overlapping setting with 30 clients.

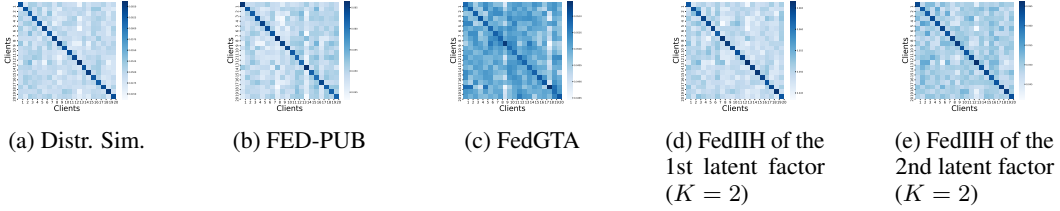


Figure 29: Similarity heatmaps on the *Roman-empire* dataset in the non-overlapping setting with 20 clients.

22. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no)

Answer: [Yes]

Justification: We provide them in this Appendix.

23. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA)

Answer: [Yes]

Justification: Our code is available at <https://github.com/blgpb/FedIIH>. We also provide a README.md file to describe the detailed instructions on how to replicate the results.

24. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes/partial/no)

Answer: [Yes]

Justification: We provide them in this Appendix.

25. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes/partial/no)

Answer: [Yes]

Justification: We provide them in this Appendix.

26. This paper states the number of algorithm runs used to compute each reported result. (yes/no)

Answer: [Yes]

Justification: We provide them in the paper.

27. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes/no)

Answer: [Yes]

Justification: We provide them in this Appendix.

28. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no)

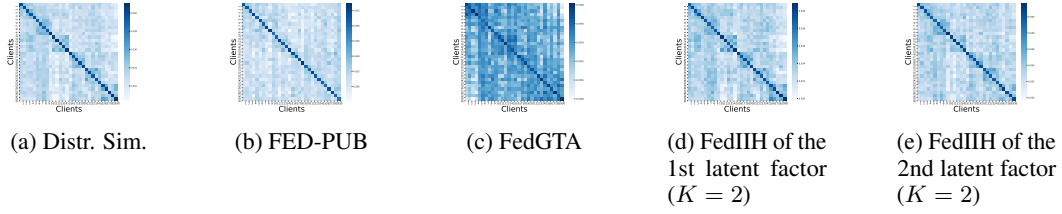


Figure 30: Similarity heatmaps on the *Roman-empire* dataset in the overlapping setting with 30 clients.

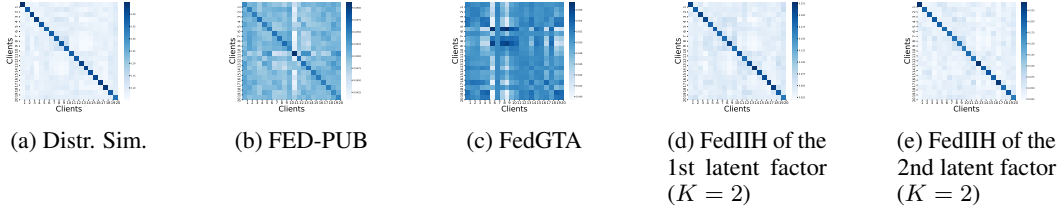


Figure 31: Similarity heatmaps on the *Amazon-ratings* dataset in the non-overlapping setting with 20 clients.

Answer: [\[Yes\]](#)

Justification: We provide them in the paper.

29. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no/NA)

Answer: [\[Yes\]](#)

Justification: We provide them in this Appendix.

30. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes/partial/no/NA)

Answer: [\[Yes\]](#)

Justification: We provide them in this Appendix.

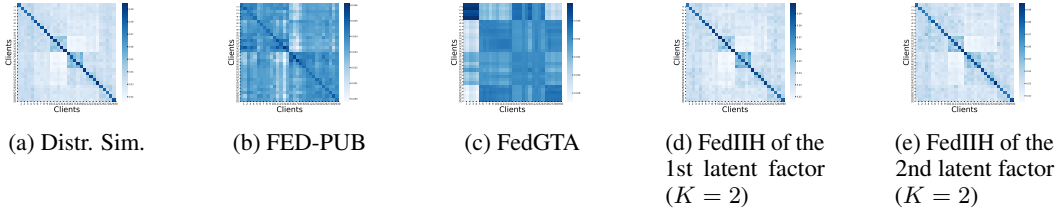


Figure 32: Similarity heatmaps on the *Amazon-ratings* dataset in the overlapping setting with 30 clients.

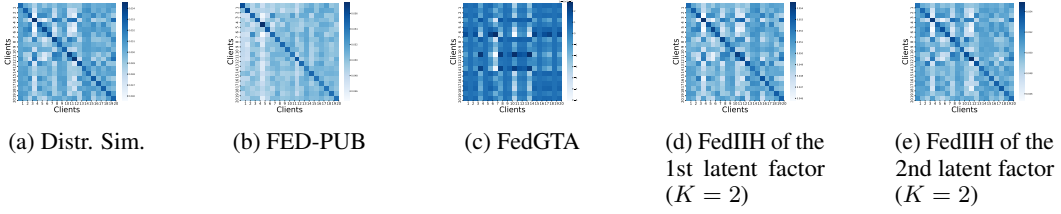


Figure 33: Similarity heatmaps on the *Minesweeper* dataset in the non-overlapping setting with 20 clients.

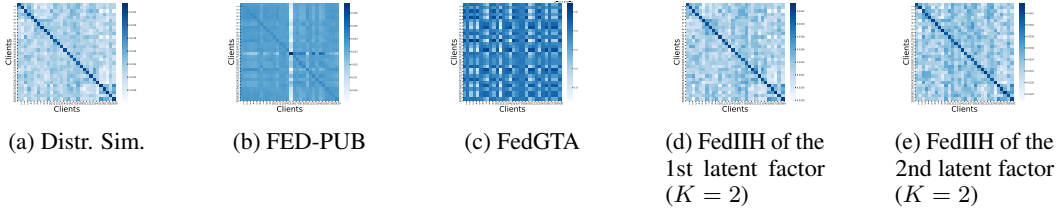


Figure 34: Similarity heatmaps on the *Minesweeper* dataset in the overlapping setting with 30 clients.

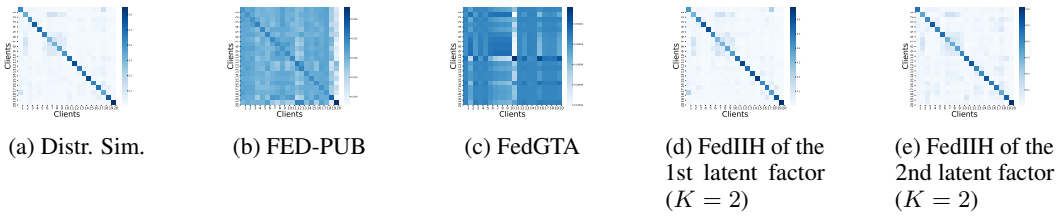


Figure 35: Similarity heatmaps on the *Tolokers* dataset in the non-overlapping setting with 20 clients.

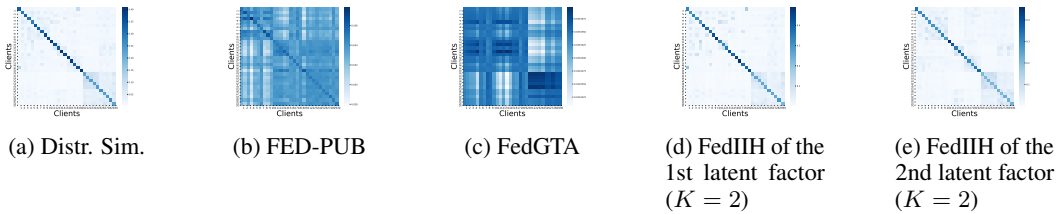


Figure 36: Similarity heatmaps on the *Tolokers* dataset in the overlapping setting with 30 clients.

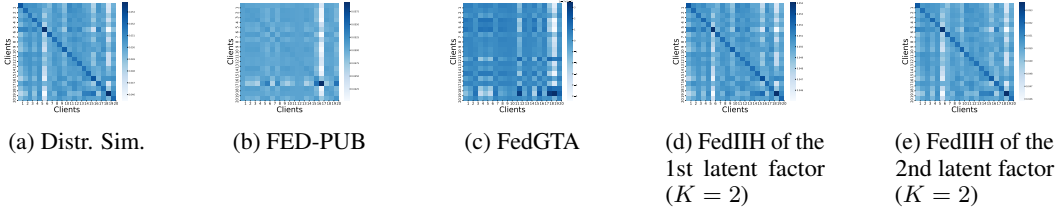


Figure 37: Similarity heatmaps on the *Questions* dataset in the non-overlapping setting with 20 clients.



Figure 38: Similarity heatmaps on the *Questions* dataset in the overlapping setting with 30 clients.

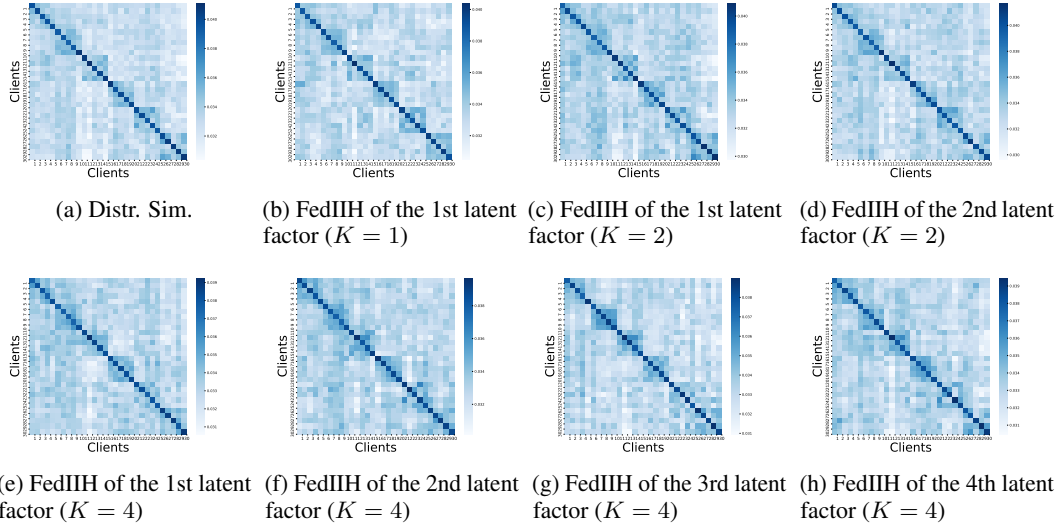


Figure 39: Similarity heatmaps on the *Roman-empire* dataset in the overlapping setting with 30 clients. There are indeed differences between different similarity heatmaps when K is set to 1, 2, and 4.

Table 7: The hyperparameter sensitivity analysis of τ . The best results are shown in **bold**.

τ	Cora Non-overlapping 10 Clients	Cora Overlapping 30 Clients	Roman-empire Non-overlapping 10 Clients	Roman-empire Overlapping 30 Clients
1	81.58 \pm 0.15	76.74 \pm 0.34	66.12 \pm 0.25	63.15 \pm 0.15
2	81.61 \pm 0.12	76.52 \pm 0.16	66.41 \pm 0.34	63.23 \pm 0.12
3	81.76 \pm 0.16	76.69 \pm 0.20	66.34 \pm 0.32	63.19 \pm 0.25
4	81.80 \pm 0.17	76.72 \pm 0.23	66.40 \pm 0.36	63.29 \pm 0.30
5	81.65 \pm 0.05	76.66 \pm 0.31	66.36 \pm 0.27	63.26 \pm 0.31
6	81.82 \pm 0.10	76.75 \pm 0.26	66.21 \pm 0.19	63.28 \pm 0.24
7	81.75 \pm 0.11	76.80 \pm 0.15	66.37 \pm 0.33	63.13 \pm 0.15
8	81.81 \pm 0.18	76.79 \pm 0.25	66.36 \pm 0.40	63.22 \pm 0.22
9	81.82 \pm 0.14	76.74 \pm 0.26	66.34 \pm 0.21	63.30 \pm 0.15
10	81.85\pm0.09	76.82\pm0.24	66.44\pm0.28	63.32\pm0.06

Table 8: The training time of one communication round of our FedIIH and two baseline methods.

Methods	Cora Nonoverlapping			Cora Overlapping		
	5 Clients	10 Clients	20 Clients	10 Clients	30 Clients	50 Clients
FedAvg	20.81s	24.90s	59.58s	30.63s	72.55s	141.82s
FED-PUB	22.04s	27.34s	60.31s	33.46s	80.54s	147.84s
FedIIH (Ours)	19.57s	22.76s	56.09s	19.67s	65.49s	139.03s
Methods	Roman-empire Nonoverlapping			Roman-empire Overlapping		
	5 Clients	10 Clients	20 Clients	10 Clients	30 Clients	50 Clients
FedAvg	18.25s	29.85s	55.21s	28.30s	79.12s	127.61s
FED-PUB	18.81s	28.03s	61.75s	28.45s	83.07s	133.05s
FedIIH (Ours)	17.45s	24.90s	47.01s	28.19s	61.17s	100.10s

Table 9: The hyperparameter sensitivity analysis of the number of neighborhood routing layers.

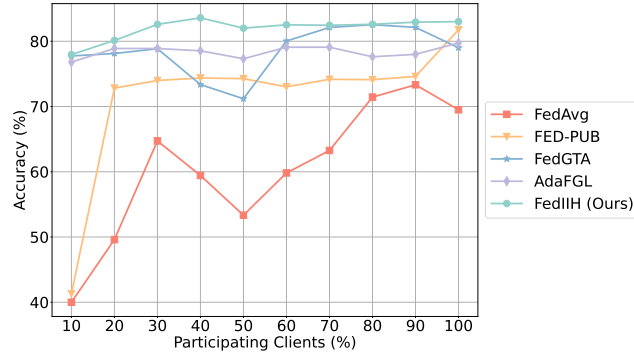
# neighborhood routing layers	Cora Non-overlapping 10 Clients	Cora Overlapping 30 Clients	Roman-empire Non-overlapping 10 Clients	Roman-empire Overlapping 30 Clients
1	81.19±0.27	76.18±0.46	66.44±0.28	63.32±0.06
2	81.41±0.15	76.34±0.31	66.16±0.41	63.15±0.15
3	81.46±0.12	76.48±0.15	66.23±0.36	63.12±0.21
4	81.85±0.09	76.67±0.18	66.15±0.46	63.15±0.42
5	81.62±0.08	76.82±0.24	66.11±0.55	63.10±0.24
6	81.37±0.24	76.46±0.45	66.08±0.39	63.04±0.38
max - min	0.66	0.64	0.36	0.28

Table 10: The hyperparameter sensitivity analysis of the number of neighborhood routing iterations.

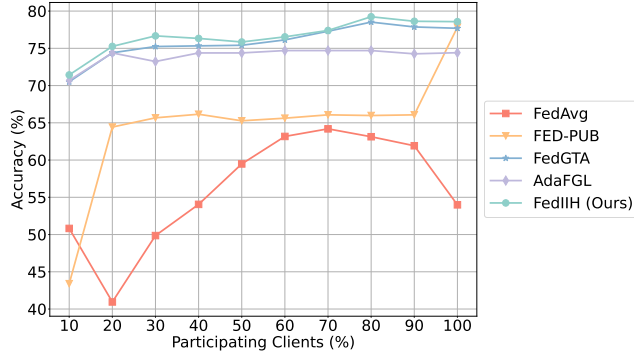
# neighborhood routing iterations	Cora Non-overlapping 10 Clients	Cora Overlapping 30 Clients	Roman-empire Non-overlapping 10 Clients	Roman-empire Overlapping 30 Clients
2	81.24±0.50	76.22±0.30	66.04±0.43	63.01±0.45
3	81.38±0.45	76.31±0.44	66.10±0.46	63.10±0.34
4	81.45±0.41	76.37±0.45	66.16±0.44	63.18±0.29
5	81.57±0.35	76.45±0.41	66.15±0.35	63.24±0.17
6	81.85±0.09	76.82±0.24	66.44±0.28	63.32±0.06
7	81.36±0.42	76.68±0.39	66.24±0.29	63.12±0.22
max - min	0.61	0.60	0.40	0.31

Table 11: The hyperparameter sensitivity analysis of τ .

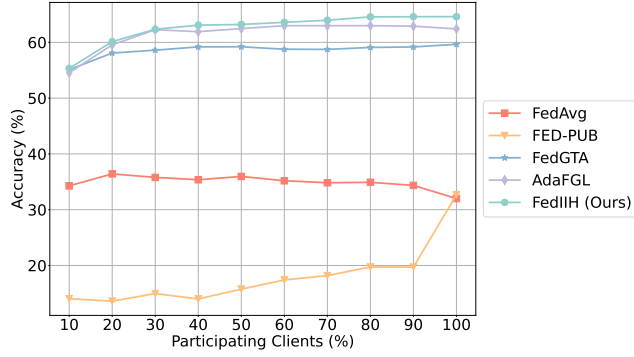
τ	Cora Non-overlapping 10 Clients	Cora Overlapping 30 Clients	Roman-empire Non-overlapping 10 Clients	Roman-empire Overlapping 30 Clients
1	81.58±0.15	76.74±0.34	66.12±0.25	63.15±0.15
2	81.61±0.12	76.52±0.16	66.41±0.34	63.23±0.12
3	81.76±0.16	76.69±0.20	66.34±0.32	63.19±0.25
4	81.80±0.17	76.72±0.23	66.40±0.36	63.29±0.30
5	81.65±0.05	76.66±0.31	66.36±0.27	63.26±0.31
6	81.82±0.10	76.75±0.26	66.21±0.19	63.28±0.24
7	81.75±0.11	76.80±0.15	66.37±0.33	63.13±0.15
8	81.81±0.18	76.79±0.25	66.36±0.40	63.22±0.22
9	81.82±0.14	76.74±0.26	66.34±0.21	63.30±0.15
10	81.85±0.09	76.82±0.24	66.44±0.28	63.32±0.06
max - min	0.27	0.30	0.32	0.19



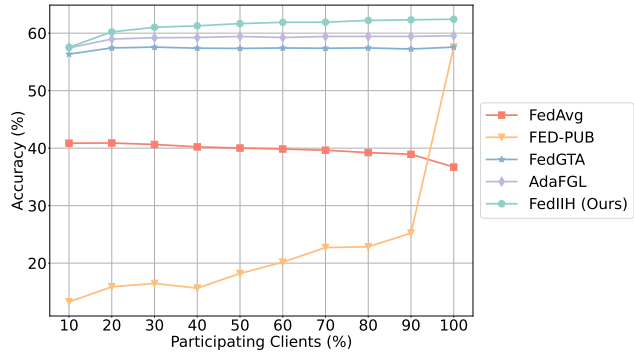
(a) *Cora* Non-overlapping 20 Clients



(b) *Cora* Overlapping 50 Clients



(c) *Roman-empire* Non-overlapping 20 Clients



(d) *Roman-empire* Overlapping 50 Clients

Figure 40: Performances with different percentages of participating clients.