Client-Specific Hyperbolic Federated Learning

Jiahong Liu
jiahong.liu21@gmail.com
The Chinese University of Hong Kong
Hong Kong, China

Xinyu Fu
xyfu@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong, China

Menglin Yang
menglin.yang@outlook.com
Yale University
New Haven, United States

Weixi Zhang
zhangweixi1@huawei.com
Huawei Technologies Co., Ltd.
Hong Kong, China

Rex Ying
rex.ying@yale.edu
Yale University
New Haven, United States

Irwin King
king@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong, China

ABSTRACT

Personalized Federated Learning (PFL) has gained attention for privacy-preserving training on heterogeneous data. However, existing methods fail to capture the unique inherent geometric properties across diverse clients by assuming a unified Euclidean space for all data distributions. Drawing on hyperbolic geometry's ability to fit complex data properties, we present FlatLand[1], a novel personalized **F**ederated **le**~~ar~~ning method that embeds different clients' data in **t**ailored **L**orentz space. FlatLand is able to directly tackle the challenge of heterogeneity through the client-specific curvatures of their respective Lorentz model of hyperbolic geometry, which is manifested by the time-like dimension. Leveraging the Lorentz model properties, we further design a parameter decoupling strategy that enables direct server aggregation of common client information, with reduced heterogeneity interference and without the need for any client-wise similarity estimation. To the best of our knowledge, this is the first attempt to incorporate hyperbolic geometry into personalized federated learning. Empirical results on various federated graph learning tasks demonstrate that FlatLand achieves superior performance in node and graph classification tasks, particularly in low-dimensional settings.

KEYWORDS

Personalized Federated Learning, Hyperbolic Geometry

ACM Reference Format:
Jiahong Liu, Xinyu Fu, Menglin Yang, Weixi Zhang, Rex Ying, and Irwin King. 2024. Client-Specific Hyperbolic Federated Learning . In *Proceedings of FedKDD'24* . ACM, New York, NY, USA, 11 pages.

1 Introduction

Federated learning (FL) trains machine learning models across multiple clients while ensuring data privacy. Traditional FL struggles with data heterogeneity, as one model cannot satisfy diverse local requirements. Personalized federated learning (PFL) resolves this by sharing common model knowledge and allowing for client-specific adaptations. PFL approaches include segmenting models into generic and personalized components [33], leveraging model weights and gradients to map client relationships [35], or integrating additional modules to facilitate customization [5].

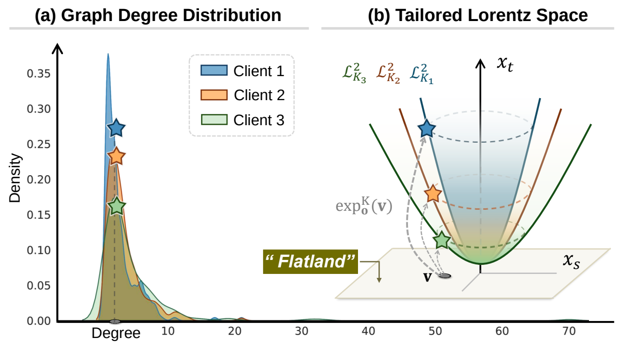

Figure 1: Toy example: (a) KDE of degree distributions from three CiteSeer clients [8], and (b) their respective 2D Lorentz Spaces with different curvatures K.

Recent studies in various domains, including text [9], images [17], and graphs [25, 33], have shown that real-world data exhibit non-Euclidean properties, such as scale-free structures and implicit hierarchical relationships [1, 23]. Euclidean space, being inherently "flat", fails to adequately represent these characteristics, leading to structural distortions and reduced performance [6, 24]. For example, the CiteSeer graph dataset partitioned into 10 clients shows varying degree distributions with long-tail characteristics which are poorly captured by Euclidean geometry, as illustrated in Figure 1(a). Besides, the calculation of Ricci curvature values for multiple real-world graph datasets after splitting them into 10 clients each reveals that they all exhibit negative Ricci curvature with significantly varying values, as shown in Figure 5. Higher absolute values indicate more pronounced non-Euclidean properties.

Moreover, embedding data from various clients into a fixed space complicates the interpretability of model parameters, making it difficult to segment the model into meaningful components [3] and often expensive to assess similarities between client models. Additionally, strategies like incorporating extra modules to aid this process further add to the model's complexity.

[1]Our method is named after Edwin Abbott's book "*Flatland: A Romance of Many Dimensions*", highlighting our insights of exploring an extra dimension that maps various data distributions onto different Lorentz surfaces.

The aforementioned problems inspire us to ask **whether there is a space where we can design a tailored model for each client, in which we can *effectively* represent the inherent properties of local data and *succinctly* reflect the heterogeneity without any extra calculations?**

We propose to leverage **Lorentz Space**. With negative curvature, Lorentz space has the advantage of modeling complex data, particularly hierarchical, tree-like, and power-law distributed data [9, 20, 36, 38]. By adjusting its curvature, it offers personalized and precise data representations for each client, leveraging its unique time-like dimension to capture diversity. This inspires us to design a framework that embeds each client's data into a suitable Lorentz space.

Furthermore, the representations in Lorentz space and the operations of Lorentz neural networks [7, 37] have stronger interpretability. Take Figure 1(b) as an example[2]. Informally speaking, the diversity of the distribution can be more prominently represented by the *"height"* of the additional *time-like* dimension ($x_t \in \mathbb{R}$) while maintaining the relatively similar properties in the *"Flatland"* (*space-like* dimensions $\mathbf{x}_s \in \mathbb{R}^d$). In this work, we focus on federated graph learning (FGL) as hyperbolic encoders have achieved state-of-the-art results in many benchmarks [20], and this method is generalizable to other datasets and settings.

Although the Lorentz space has demonstrated significant potential in various tasks [4, 28], applying it to personalized federated learning (PFL) scenarios is still non-trivial. The challenge is **how to mitigate the influence of parameters related to heterogeneous information**, and aggregate the parameters that represent common features in the flatland without accessing client data?

Motivated by the above insights, we propose an exploratory personalized **F**ederated **lea**rning method that embeds different clients' data in **T**ailored **L**orentz space, called FlatLand. To address the challenge, we formulate a parameter disentanglement strategy that can directly aggregate shared parameters without any extra similarity calculations. To the best of our knowledge, FlatLand is the first work to incorporate Lorentz geometry into personalized federated learning. It is **succinct**, **effective**, and **easily interpretable**. Experimental results demonstrate that FlatLand achieves superior performance than its Euclidean counterpart, particularly in low-dimensional representations.

2 Motivation and Insights

The related work and preliminaries are shown in Appendix A and Appendix B. This paper focuses on graph data for its clear distribution and simpler models, facilitating the validation of our approach using Lorentz neural networks to address heterogeneity in personalized federated learning. Our method is also applicable to other datasets and tasks.

Problem Statement. Given clients $C = \{1, 2, \ldots, C\}$, each with a dataset $\mathcal{D}_c = (\mathbf{x}_i^c, y_i^c)_{i=1}^{N_c}$ and distribution $p_c(\mathbf{x}, y)$, Personalized Federated Learning (PFL) encounters distributional heterogeneity if exists $p_i(\mathbf{x}, y) \neq p_j(\mathbf{x}, y)$ for clients $i \neq j$. This heterogeneity can degrade performance. The goal is to optimize personalized models $f_c(\cdot; \boldsymbol{\theta}_c, \boldsymbol{\theta}_s)$ for each client using specific and shared parameters $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_s$.

$$\min_{\boldsymbol{\theta}_c|_{c=1}^C, \boldsymbol{\theta}_s} \sum_{c=1}^{C} \mathbb{E}_{(\mathbf{x},y)\sim p_c(\mathbf{x},y)} \left[\mathcal{L}_c(f(\mathbf{x}; \boldsymbol{\theta}_c, \boldsymbol{\theta}_s), y) \right] + \lambda \Omega(\boldsymbol{\theta}_c|_{c=1}^C, \boldsymbol{\theta}_s) \quad (1)$$

This function merges local loss $\mathcal{L}_c$ with regularization Ω, balanced by hyperparameter λ.

Our goals are

(1) to *effectively* represent the inherent properties of each local client data;
(2) to *succinctly* reflect heterogeneity among client data and facilitate the communication of shared information without requiring additional computations.

In "Flatland", a two-dimensional flat plane, the same shapes may represent the projections of various three-dimensional objects. For instance, a circle could be the projection of either a cylinder or a sphere from a higher dimension.

Insights: *introduce a higher dimension (time axes) to "Flatland".* In the above case, *"Flatland"* captures the common feature of a cylinder or a sphere, while a higher dimension (the third dimension) highlights the differences between the objects. Analogous to our setting, informally speaking, by introducing an additional *time-like* dimension, we can imagine each client's data residing in a unique Lorentz space (a curved world in a higher-dimensional space), where the curvature reflects the distinct distributions (objects). *"Flatland"*, $\mathbb{R}^d$ (flat), serves as a metaphor for a platform where common information (circle) is exchanged and integrated.

Motivation: *why Lorentz space?*

(1) Prevalent Non-Euclidean properties of real-world data. Forman-Ricci curvature $\overline{\mathrm{Ric}}$ measures deviations from flat (Euclidean) geometry in data structures [12, 29]. A more negative $\overline{\mathrm{Ric}}$ indicates a structure more suited for hyperbolic space representation [32]. Figure 2 shows varying $\overline{\mathrm{Ric}}$ values across 10 clients from the CiteSeer dataset, highlighting the common non-Euclidean nature of real-world data. Thus, employing Lorentz space with client-specific curvature can better capture intrinsic data structure for goal (1).

(2) Strong correlation between heterogeneity and curvature. Figure 1(a) shows that distribution curves exhibit long-tailed characteristic with varying skewness. In particular, Client 1's distribution is steeper and less Euclidean, suggesting a need for embedding in a Lorentz space with a larger curvature (a smaller K), depicted in Figure 1(b). This space accommodates more tail nodes (blue stars) than Clients 2 and 3, requiring a "roomier" embedding environment to ensure separability and enhance performance. A larger curvature facilitates this by allowing embeddings to occupy a "higher" position (larger x_t) in the space, where the volume expands exponentially.

The observations align with our goal (2) as heterogeneous properties like imbalance between tail and head nodes can be distinguished through corresponding Lorentz spaces with different curvatures (differed by *time-like* axes x_t). Meanwhile, common information like "the star is a tail node" is preserved in *space-like* dimensions $\mathbf{x}_s$ as the same node $\mathbf{v}$.

[2]For convenience, all origins of Lorentz spaces in the figure are shown as the same, but actually, their origins are not in the same location.

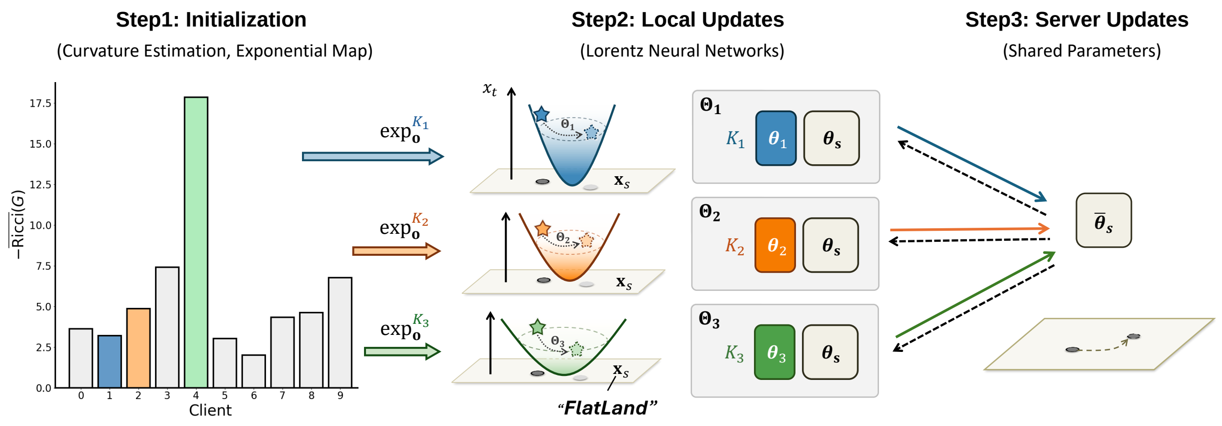
Step1: Initialization
(Curvature Estimation, Exponential Map)
Step2: Local Updates
(Lorentz Neural Networks)
Step3: Server Updates
(Shared Parameters)
$-\overline{\mathrm{Ricci}}(G)$
Client
$\exp_{\mathbf{o}}^{K_1}$
$\exp_{\mathbf{o}}^{K_2}$
$\exp_{\mathbf{o}}^{K_3}$
x_t
$\mathbf{x}_s$
"FlatLand"
$\mathbf{\Theta_1}$ K_1 θ_1 θ_s
$\mathbf{\Theta_2}$ K_2 θ_2 θ_s
$\mathbf{\Theta_3}$ K_3 θ_3 θ_s
$\bar{\theta}_s$

Figure 2: The FlatLand **framework.**

3 The FlatLand Framework

We propose a personalized federated learning framework, FlatLand, using tailored Lorentz spaces for each client. The main steps are outlined in Figure 2 and Algorithm 1.

S1 Initialization. At the initial communication round $r = 0$, the parameters can be divided into three parts:
(1) Curvature parameters of C clients $\{K_1, K_2, ...K_C\}$; **(Sec. 3.1)**
(2) Personalized parameters of C clients $\{\theta_1, \theta_2, ..., \theta_C\}$; **(Sec. 3.2)**
(3) Shared parameters $\overline{\theta}_s$ of central server.

All the parameters of client i at round 0 can be written as $\Theta_i^{(0)} = \left(K_i; \theta_i^{(0)}; \overline{\theta}_s^{(0)}\right)$ and server parameters as $\overline{\theta}_s^{(0)}$.

S2 Local updates. Given learning rate η, for round r, each local client model performs training on the data $\mathcal{D}_i$ to minimize the task loss $\mathcal{L}(\mathcal{D}_i; \Theta_i^{(r)})$ and then updating the parameters as $\Theta_i^{(r+1)} \leftarrow \Theta_i^{(r)} - \eta\nabla\mathcal{L}$. **(Sec. 3.3)**

S3 Server updates. After local training, only shared parameters ${\theta_s}_c{}^{(r+1)}$ are updated to the server for each client c. These are then aggregated using FedAvg: $\overline{\theta}_s^{(r+1)} \leftarrow \frac{N_c}{N}\sum_{c=1}^{C} \theta_{s_c}^{(r+1)}$, , where $N = \sum_c N_c$. The aggregated parameters are subsequently distributed to clients for the next round.

3.1 Curvature Estimation

To embed the dataset $\mathcal{D}_c$ of client $c \in C$ into its tailored Lorentz space $\mathcal{L}^d_{K_c}$, a suitable curvature K_c should be first explored.

There are many comprehensive ways to assist in estimating the suitable curvature for various types of data [13]. Given a weighted graph $G_c = (V, E, w)$ in client c, we adopt Forman-Ricci curvature and the overall curvature of the graph can be calculated as follows $\overline{\mathrm{Ric}}(G) = \frac{1}{|E|}\sum_{(x,y)\in E} w(x,y)\left(\frac{1}{\mu_x} + \frac{1}{\mu_y}\right)$, where V represents graph nodes and $|E|$ the number of edges. Additionally, the curvature can be a learnable parameter or calculated using a simple Multi-Layer Perceptron (MLP) neural network (Appendix B.3). Here, we initialize K_c with $\overline{\mathrm{Ric}}(G_c)$ as learnable. The curvature parameters represent client-specific inherent data characteristics and thus do not require sharing and aggregating.

3.2 Parameter Decoupling Strategy

This section details the fully Lorentz model's parameters (excluding K), divided into shared θ_s for the *space-like* dimensions and personalized θ_c for the *time-like* dimension. The model has layers of fully Lorentz neural networks that transform data within Lorentz space (Appendix B.2).

First, without loss of generality, we decouple the function of Lorentz linear layer in Equation (5) without the functions f of activation, dropout, bias, and so on. Given input $\mathbf{x}^{(l)} = \begin{bmatrix} x_t^{(l)} & \mathbf{x}_s^{(l)} \end{bmatrix}^T \in \mathcal{L}^n_K, x_t^{(l)} \in \mathbb{R}, \mathbf{x}_s^{(l)} \in \mathbb{R}^n$ in layer l. We rewrite the learnable matrix $\hat{\mathbf{M}}^{(l)}$ in Section B.2 as $\begin{bmatrix} v^{(l)} & \mathbf{v}^{T(l)} \\ m^{(l)} & \mathbf{M}^{(l)} \end{bmatrix} \in \mathbb{R}^{(m+1)\times(n+1)}$, $v^{(l)} \in \mathbb{R}, \mathbf{v}^{(l)} \in \mathbb{R}^n, m^{(l)} \in \mathbb{R}^{m+1}, \mathbf{M}^{(l)} \in \mathbb{R}^{(m+1)\times n}$. The output $\mathbf{x}^{(l+1)}$ of the Lorentz linear layer could be reformulated as

$$\mathbf{x}^{(l+1)} = \mathrm{LT}(\mathbf{x}^{(l)}; \hat{\mathbf{M}}^{(l)}) = (\underbrace{\sqrt{\|mx_t + \mathbf{M}\mathbf{x}_s\|^2 + K}}_{\text{time-like } x_t^{(l+1)}}, \underbrace{mx_t + \mathbf{M}\mathbf{x}_s}_{\text{space-like } \mathbf{x}_s^{(l+1)}})^T. \quad (2)$$

Then, we decouple the parameters as follows under the deviation from Appendix C.3:

Suppose the model $\mathcal{M}$ consists of L layers of neural networks,

- The personalized parameter set θ_c for all layers is formulated as
$$\theta_c = \bigcup_{l=1}^{L}\{v^{(l)}, \mathbf{v}^{T(l)}, m^{(l)}\};$$
- The shared parameter set θ_s across all layers is formulated as
$$\theta_s = \bigcup_{l=1}^{L}\{\mathbf{M}^{(l)}\};$$
where $\bigcup_{l=1}^{L}$ indicates the union of parameter sets from each layer l from 1 to L.

Table 1: Comparison of node classification performance across real-world datasets with varying numbers of clients. The results, presented as mean and standard deviation, are based on five separate trials. Performances that are statistically significant ($p < 0.05$) are highlighted in bold.

	Cora		CiteSeer		ogbn-arxiv		Photo	
# clients	**10**	**20**	**10**	**20**	**10**	**20**	**10**	**20**
Local (E)	79.94 ± 0.24	80.30 ± 0.25	67.82 ± 0.13	65.98 ± 0.17	64.92 ± 0.09	65.06 ± 0.05	91.80 ± 0.02	90.47 ± 0.15
Local (L)	78.35 ± 0.05	80.46 ± 0.18	72.30 ± 0.04	69.52 ± 0.25	65.85 ± 0.09	66.75 ± 0.05	91.76 ± 0.10	90.12 ± 0.20
FedAvg	69.19 ± 0.67	69.50 ± 3.58	63.61 ± 3.59	64.68 ± 1.83	64.44 ± 0.10	63.24 ± 0.13	83.15 ± 3.71	81.35 ± 1.04
FedPer	79.35 ± 0.04	78.01 ± 0.32	70.53 ± 0.28	66.64 ± 0.27	64.99 ± 0.18	64.66 ± 0.11	91.76 ± 0.23	90.59 ± 0.06
FedProx	60.18 ± 7.04	48.22 ± 6.81	63.33 ± 3.25	64.85 ± 1.35	64.37 ± 0.18	63.03 ± 0.04	80.92 ± 4.64	82.32 ± 0.29
FedGNN	70.12 ± 0.99	70.10 ± 3.52	55.52 ± 3.17	52.23 ± 6.00	64.21 ± 0.32	63.80 ± 0.05	87.12 ± 2.01	81.00 ± 4.48
FedSage+	69.05 ± 1.59	57.97 ± 12.6	65.63 ± 3.10	65.46 ± 0.74	64.52 ± 0.14	63.31 ± 0.20	76.81 ± 8.24	80.58 ± 1.15
GCFL	78.66 ± 0.27	79.21 ± 0.70	69.01 ± 0.12	66.33 ± 0.05	65.09 ± 0.08	65.08 ± 0.04	92.06 ± 0.25	90.79 ± 0.17
FedHGCN	72.09 ± 0.16	74.67 ± 1.50	66.98 ± 0.56	64.28 ± 0.62	OOM	OOM	79.26 ± 0.56	79.57 ± 0.10
FlatLand (Ours)	**80.46 ± 0.28**	**82.49 ± 0.25**	**73.90 ± 0.23**	**72.24 ± 0.24**	**67.52 ± 0.16**	**67.64 ± 0.04**	**92.49 ± 0.19**	**91.06 ± 0.15**

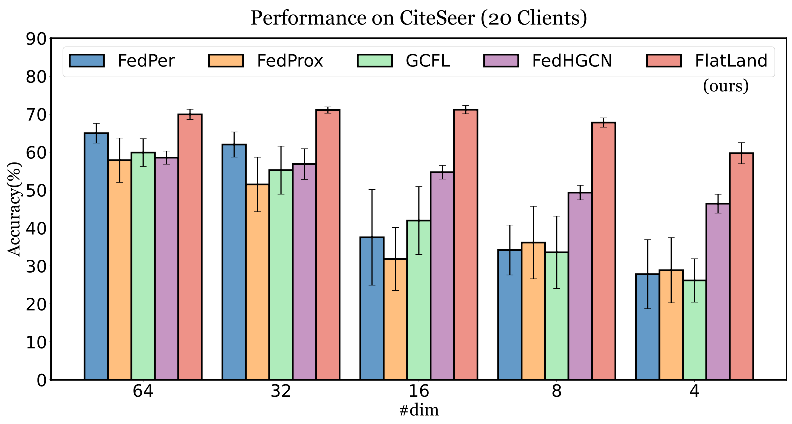

Figure 3: Performance of CiteSeer (20 clients) with varying dimensions for node classification scenario.

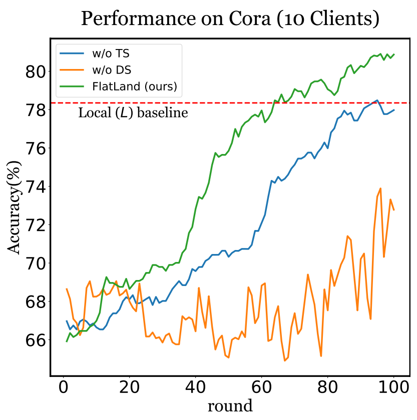

Figure 4: Ablation study of FlatLand on the Cora dataset.

3.3 Local Training Procedure

Obtained the curvature $K_c^{(r)}$ at round r for client c, we directly project the client input $\mathbf{x}_i^E \in \mathcal{D}_c$ into its corresponding Lorentz space via the exponential map $\mathbf{x}^{K_c} = \exp_{\mathbf{o}}^{K_c}(\mathbf{x}^E)$, as shown in Equation (4). Note that to simplify the notation, all vectors $\mathbf{x}$, if not superscripted, are assumed to represent being in the Lorentz space.

Afterward, the training data are fed into the corresponding Lorentz model, the output is $f(\mathbf{x}^{K_c}; \boldsymbol{\theta}_c, \boldsymbol{\theta}_s)$. In the graph model, in addition to the Lorentz linear layer, there is also an aggregation operation [40], which does not involve extra parameters. At client c, the objective function is

$$\min_{\boldsymbol{\theta}_c|_{c=1}^C, \boldsymbol{\theta}_s} \mathcal{L}_c(f(\mathbf{x}^{K_c}; \boldsymbol{\theta}_c, \boldsymbol{\theta}_s), y) + \lambda \|\boldsymbol{\theta}_{s_c} - \overline{\boldsymbol{\theta}}_s\|_2^2, \tag{3}$$

where λ is a hyperparameter, $\|\boldsymbol{\theta}_{s_c} - \overline{\boldsymbol{\theta}}_s\|_2^2$ is the regularize term that prevent locally updated model parameters $\boldsymbol{\theta}_{s_c}$ deviates too far from the server shared parameters $\overline{\boldsymbol{\theta}}_s$.

4 Experiments

In this section, we validate the effectiveness of FlatLand by conducting experiments for *node classification* on a series of benchmark datasets. The experiments are designed to address the following research questions. **RQ1.** Can FlatLand outperform personalized and hyperbolic FL baselines? **RQ2.** Can FlatLand still perform well in low-dimensional settings? **RQ3.** Are the proposed novel components really beneficial?

4.1 Experimental Setup

Datasets and Baselines The details about datasets are listed in Appendix D.1. Implementation details are shown in Appendix D.2. To assess FlatLand and demonstrate its superiority, we compare it with the following baselines:

(1) Local: clients train their models locally without any communication, Local (E) refers to self-training in the Euclidean model, while Local (L) refers to training in the Lorentz model.; (2) FedAvg [26] and (3) FedProx [21]: the most popular pederated learning baselines; (4) FedPer [3]: a personalized federated learning baseline with personalized model layers; (5) FedGNN [34] and (6) FedSage [39]: two federated graph learning baselines; (7) GCFL [35]: a personalized federated graph learning baseline with client clustering and cluster-wise model aggregation; (8) FedHGCN [10]: a hyperbolic FGL baseline that fails considering the heterogeneity among clients.

4.2 Main Experimental Results (RQ1)

Node Classification Table 1 shows that our proposed FlatLand outperforms all baselines with statistical significance ($p < 0.05$). (1) Local (L) often surpasses Local (E), suggesting that hyperbolic space can better represent most datasets, though the gap is sometimes marginal. (2) Euclidean FL methods like FedAvg, FedProx, FedGNN, and FedSage+ significantly underperform self-training. GCFL is generally the best among Euclidean methods, but cannot consistently beat Local (E). FedPer sometimes exceeds Local (E) with small gains, highlighting challenges with heterogeneous data. (3) FedHGCN, despite operating in hyperbolic space, underperforms on heterogeneous datasets by not accounting for data heterogeneity, akin to FedAvg vs Local (E) in Euclidean space. Besides, due to the quadratic time and space complexity of FedHGCN's node selection module. Therefore, it can easily encounter out-of-memory (OOM) issues with large datasets, like ogbn-arxiv. In conclusion, experiments show that FlatLand can mitigate the heterogeneity, and with larger gains on highly heterogeneous datasets like CiteSeer.

4.3 Varying Embedding Dimensions (RQ2)

Reducing embedding and hidden dimensions lowers parameter transmission cost in federated learning. Considering hyperbolic spaces' representational power in lower dimensions [6], we evaluated FlatLand's ability to mitigate data heterogeneity using compact representations by reducing the embedding dimension from 64 to 4 (Figure 3, CiteSeer, 20 clients). Dimensionality reduction from 64 to 4 had a smaller impact on hyperbolic methods (FlatLand and FedHGCN) compared to Euclidean counterparts. While FedHGCN underperformed Euclidean methods at higher dimensions but outperformed them at 16 dimensions, FlatLand consistently outperformed all others, with its advantage over baselines becoming more significant as dimensionality reduced.

4.4 Ablation Study (RQ3)

Figure 4 analyzes component contributions. "w/o TS" uses constant curvature 1 for all clients instead of tailored curvatures, yielding inferior performance compared to tailored curvatures approximating local (L) setting, demonstrating hyperbolic space's effectiveness. "w/o DS" exhibits significant fluctuations across rounds due to aggregation incorporating heterogeneous information, adversely impacting results. This highlights the proposed decoupling strategy's effectiveness and validates the time-like dimension's ability to capture heterogeneity.

5 Conclusion

FlatLand leverages hyperbolic geometry to capture heterogeneity across clients' data distributions embedded in tailored Lorentz spaces for personalized federated learning. A parameter decoupling strategy aggregates common information server-side while mitigating heterogeneity interference, without extra client similarity estimation. As the first work incorporating hyperbolic geometry into personalized federated learning, FlatLand outperforms Euclidean methods, especially in low dimensions, showcasing potential as an effective solution to data heterogeneity.

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[2] Xuming An, Li Shen, Han Hu, and Yong Luo. 2024. Federated Learning with Manifold Regularization and Normalized Update Reaggregation. *Advances in Neural Information Processing Systems* 36 (2024).

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. *CoRR* abs/1912.00818 (2019).

[4] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne Van Noord, and Pascal Mettes. 2022. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 4453–4462.

[5] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized subgraph federated learning. In *International Conference on Machine Learning*. PMLR, 1396–1415.

[6] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems* 32 (2019).

[7] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686* (2021).

[8] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. 2011. Remarks on some non-parametric estimates of a density function. *Selected Works of Murray Rosenblatt* (2011), 95–100.

[9] Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. 2018. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313* (2018).

[10] Haizhou Du, Conghao Liu, Haotian Liu, Xiaoyu Ding, and Huan Huo. 2024. An efficient federated learning framework for graph learning in hyperbolic space. *Knowledge-Based Systems* 289 (2024), 111438.

[11] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *NeurIPS.*

[12] Forman. 2003. Bochner's method for cell complexes and combinatorial Ricci curvature. *Discrete & Computational Geometry* 29 (2003), 323–374.

[13] Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. 2021. Curvature generation in curved spaces for few-shot learning. In *Proceedings of the IEEE/CVF international conference on computer vision.* 8691–8700.

[14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS.*

[15] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2021. Personalized Cross-Silo Federated Learning on Non-IID Data. In *AAAI.* AAAI Press, 7865–7873.

[16] George Karypis and Vipin Kumar. 1995. METIS – Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0.

[17] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. 2020. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 6418–6428.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster).* OpenReview.net.

[19] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010), 036106.

[20] Keegan Lensink, Bas Peters, and Eldad Haber. 2022. Fully hyperbolic convolutional neural networks. *Research in the Mathematical Sciences* 9, 4 (2022), 60.

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys.* mlsys.org.

[22] Xinting Liao, Weiming Liu, Chaochao Chen, Pengyang Zhou, Huabin Zhu, Yanchao Tan, Jun Wang, and Yue Qi. 2023. HyperFed: hyperbolic prototypes exploration with consistent aggregation for non-IID data in federated learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence.* 3957–3965.

[23] Jiahong Liu, Philippe Fournier-Viger, Min Zhou, Ganghuan He, and Mourad Nouioua. 2022. CSPM: Discovering compressing stars in attributed graphs. *Information Sciences* 611 (2022), 126–158.

[24] Jiahong Liu, Menglin Yang, Min Zhou, Shanshan Feng, and Philippe Fournier-Viger. 2022. Enhancing hyperbolic graph embeddings via contrastive learning. *arXiv preprint arXiv:2201.08554* (2022).

[25] Jiahong Liu, Min Zhou, Philippe Fournier-Viger, Menglin Yang, Lujia Pan, and Mourad Nouioua. 2022. Discovering representative attribute-stars via minimum description length. In *2022 IEEE 38th International Conference on Data Engineering (ICDE).* IEEE, 68–80.

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

[27] Valter Moretti. 2002. The interplay of the polar decomposition theorem and the Lorentz group. *arXiv preprint math-ph/0211047* (2002).

[28] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. 2021. Hyperbolic deep neural networks: A survey. *IEEE Transactions on pattern analysis and machine intelligence* 44, 12 (2021), 10023–10044.

[29] Romeil S Sandhu, Tryphon T Georgiou, and Allen R Tannenbaum. 2016. Ricci curvature: An economic indicator for market fragility and systemic risk. *Science advances* 2, 5 (2016), e1501495.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106.

[31] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR* abs/1811.05868 (2018).

[32] Li Sun, Junda Ye, Jiawei Zhang, Yong Yang, Mingsheng Liu, Feiyang Wang, and Philip S Yu. 2024. Contrastive sequential interaction network learning on co-evolving riemannian spaces. *International Journal of Machine Learning and Cybernetics* 15, 4 (2024), 1397–1413.

[33] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated Learning on Non-IID Graphs via Structural Knowledge Sharing. In *AAAI*. AAAI Press, 9953–9961.

[34] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation. *CoRR* abs/2102.04925 (2021).

[35] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in neural information processing systems* 34 (2021), 18839–18852.

[36] Menglin Yang, Zhihao Li, Min Zhou, Jiahong Liu, and Irwin King. 2022. Hicf: Hyperbolic informative collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2212–2221.

[37] Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3770–3781.

[38] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. 2022. HRCF: Enhancing collaborative filtering via hyperbolic geometric regularization. In *Proceedings of the ACM Web Conference 2022*. 2462–2471.

[39] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu-Ming Yiu. 2021. Subgraph Federated Learning with Missing Neighbor Generation. In *NeurIPS*. 6671–6682.

[40] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*. 1249–1261.

APPENDIX

A Related Work

Personalized Federated Learning With statistical heterogeneity, conventional FL frameworks like FedAvg [26] can hardly obtain a single global model that generalizes well to every client. Motivated by this, researchers have proposed personalized FL (PFL) to train customized local models. Generally speaking, existing PFL techniques can be categorized into the following three groups: (1) techniques that personalize client models via local fine-tuning [11], (2) techniques that personalize client models via customized model aggregation [15], and (3) techniques that personalize client models via creating localized models/layers [3]. However, these PFL methods typically operate in Euclidean spaces to encode data samples, which can hardly capture the scale-free property and implicit hierarchical structure embedded within client data.

Personalized Federated Graph Learning When applied to graph data, personalized federated graph learning (PFGL) can intuitively exhibit the problem mentioned above. For example, [35] clusters clients based on gradients to aggregate models with similar data distributions. Another method [33] introduces additional personalized models to capture client-specific knowledge of graph structure. [5] calculates client-client similarities to apply personalized model aggregation with local weight masking. All these methods learn node representations in Euclidean spaces, which cannot model the power-law degree distributions that widely exist in real-world graph data [1, 19]. Additionally, the client clustering procedure and additional model components introduce computational overhead that may not be feasible in real-world scenarios with strict privacy constraints or limited resources.

Hyperbolic Federated Learning Very few research works have considered incorporating hyperbolic spaces into federated settings. [2] leverages hyperbolic distances to distill knowledge from the global model to the local model, to mitigate model inconsistency caused by data heterogeneity. [22] applies hyperbolic prototype learning to capture the hierarchical structure among data samples. As the work most similar to our FlatLand, FedHGCN [10] is a simple combination of FedAvg and hyperbolic graph neural networks along with a node selection process. Although these methods can benefit from the hyperbolic space to capture the hierarchical structure in the data, they do not have the personalization capability to adaptively model client data spaces with different curvatures. This may lead to suboptimal results when there is severe data heterogeneity. Therefore, our goal is to design a novel FL framework that can encode client data in hyperbolic spaces with adaptive curvatures using personalization techniques.

B Preliminaries

B.1 Lorentz Manifold

Given a d-dimensional Lorentz manifold $\mathcal{L}_K^d$ with a constant negative curvature $-1/K(K > 0)$, suppose a point / vector $\mathbf{x} \in \mathcal{L}_K^d$, which has the form $\mathbf{x} = \begin{bmatrix} x_t \\ \mathbf{x}_s \end{bmatrix} \in \mathbb{R}^{d+1}$, where the first dimension $x_t \in \mathbb{R}$ is called *time-like* dimension and others $\mathbf{x}_s \in \mathbb{R}^d$ are *space-like* dimensions. It satisfies the following conditions: $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -K$ and $x_t > 0$, where $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_t y_t + \mathbf{x}_s^\top \mathbf{y}_s$ is the Lorentzian inner product. Note that the larger the K, the more the intrinsic structure of the data deviates from the flatness of Euclidean space. The formal definitions are shown as follows:

Definition 1 (Lorentz Manifold). *A d-dimensional Lorentz manifold $\mathcal{L}_K^d$ with a negative curvature of $-1/K(K > 0)$ can be defined as the Riemannian manifold $\left(\mathbb{H}_K^d, g_\ell\right)$, where $g_\ell = \mathrm{diag}([-K, 1, \ldots, 1])$ and $\mathbb{H}_K^d = \left\{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -K, x_0 > 0\right\}$.*

Definition 2 (Lorentzian Inner Product). *The inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{d+1}$ can be defined as let $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^d x_i y_i$.*

Based on the constraint $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -K$, it holds for any point $\mathbf{x} = (x_0, \mathbf{x}') \in \mathbb{R}^{d+1}$ that $\mathbf{x} \in \mathcal{L}_K^d \Leftrightarrow x_0 = \sqrt{\|\mathbf{x}'\| + K}$. The larger the value of K, the greater the extent to which the hyperbolic surface deviates from the Euclidean plane.

Typically, inputs reside in Euclidean space and need to be mapped into hyperbolic space. The way of projecting the data $\mathbf{v}^E \in \mathbb{R}^d$ in Euclidean to Lorentz space $\mathbf{x} \in \mathcal{L}_K^d$ can be simplified as [3]

$$\mathbf{x}^K = \exp_{\mathbf{o}}^K\left(\mathbf{v}^E\right) = \exp_{\mathbf{o}}^K\left(\left[0, \mathbf{v}^E\right]\right)$$
$$= \left(\underbrace{\cosh\left(\frac{\|\mathbf{v}^E\|_2}{\sqrt{K}}\right)}_{\text{time-like dimension } x_t}, \underbrace{\sqrt{K}\sinh\left(\frac{\|\mathbf{v}^E\|_2}{\sqrt{K}}\right)\frac{\mathbf{v}^E}{\|\mathbf{v}^E\|_2}}_{\text{space-like dimensions } \mathbf{x}_s}\right). \tag{4}$$

B.2 Fully Lorentz Neural Networks

Fully Lorentz networks [7] are proved to be ideal for PFL due to their reduced need for space projections, enhancing computational efficiency. These networks also incorporate Lorentz transformations (boosts and rotations), improving data heterogeneity handling and parameter interpretability.

Given an input vector $\mathbf{x} \in \mathcal{L}_K^n$, and a linear layer matrix $\hat{\mathbf{M}} \in \mathbb{R}^{(m+1)\times(n+1)}$ to optimize, $\forall \mathbf{x} \in \mathcal{L}_K^n, \hat{\mathbf{M}}\mathbf{x} \in \mathcal{L}_K^n$. Let $\hat{\mathbf{M}} = \begin{bmatrix} \mathbf{v}^T \\ \mathbf{W} \end{bmatrix}, \mathbf{v} \in \mathbb{R}^{(n+1)}, \mathbf{W} \in \mathbb{R}^{m\times(n+1)}$. The fully Lorentz linear layer can be denoted as LT in a general form as follows:

$$\mathrm{LT}(\mathbf{x}; f; \hat{\mathbf{M}}) := \left(\sqrt{\|f(\mathbf{W}\mathbf{x}, \mathbf{v})\|^2 + K}, f(\mathbf{W}\mathbf{x}, \mathbf{v})\right)^T. \tag{5}$$

It involves a function f that operates on vectors $\mathbf{v} \in \mathbb{R}^{n+1}$ and $\mathbf{W} \in \mathbb{R}^{m\times(n+1)}$. Depending on the type of function, it can perform different operations. For instance, for dropout, the operation function is $f(\mathbf{W}\mathbf{x}, \mathbf{v}) = \mathbf{W}$ dropout $(\mathbf{x})$.

B.3 Forman-Ricci Curvature

Curvature is a metric used in Riemannian geometry that expresses how far a curved line deviates from a straight line, or how much a surface deviates from planarity. In this context, knowledge of the

[3]For clarity, all Lorentz space embeddings are denoted by $\cdot^H$ or with no superscript. Specifically, if the curvature of the space is known as K, it is denoted by $\cdot^K$. In contrast, Euclidean space embeddings are denoted by $\cdot^E$.

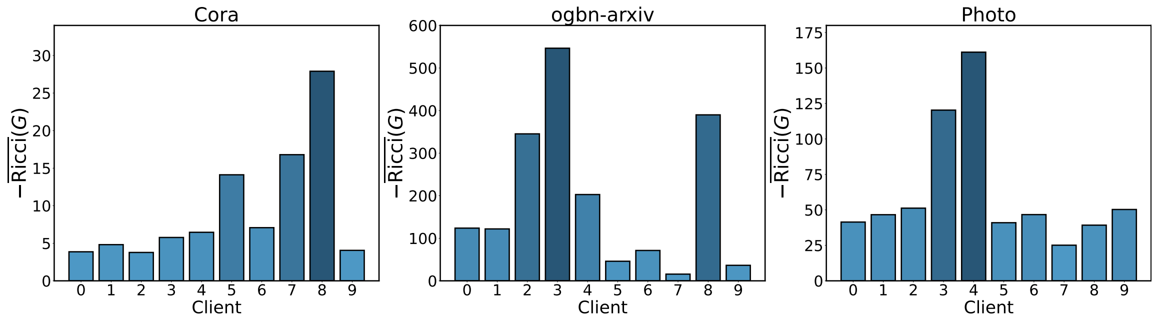

Figure 5: Averaged Forman-Ricci curvature across datasets (Cora, ogbn-arxiv, and Amazon-Photo). Higher bars indicate more pronounced non-Euclidean characteristics in these datasets.

local and global geometrical features depends on an understanding of sectional curvature and Ricci curvature, respectively.

Sectional Curvature. This type of curvature is determined at any given point on a manifold by examining all possible two-dimensional subspaces that intersect at that point. It provides a more straightforward representation than the Riemann curvature tensor. Recent studies [7] often treat sectional curvature uniformly across the manifold, simplifying it to a singular constant value.

Ricci Curvature. Ricci curvature averages the sectional curvatures at a specific point. In graph theory, various discrete versions of Ricci curvature have been developed, such as Ollivier-Ricci curvature and Forman-Ricci curvature [12]. The Ricci curvature on graphs is intended to assess how the local structure around a graph edge deviates from that of a grid graph. Notably, the Ollivier approach provides a rougher estimate of Ricci curvature, whereas the Forman method is more combinatorial and computationally efficient.

For a weighted graph $G = (V, E, w)$, the overall Forman-Ricci curvature $\overline{\mathrm{Ric}}(G)$ can be calculated as follows:

$$\overline{\mathrm{Ric}}(G) = \frac{1}{|E|} \sum_{(i,j)\in E} \mathrm{Ric}(i, j),$$

where $|E|$ represents the cardinality of the edge set E (i.e., the total number of edges), and $\mathrm{Ric}(i, j)$ is the Forman-Ricci curvature of the edge (i, j), computed as

$$\mathrm{Ric}(i, j) =: w_e \left(\frac{w_i}{w_e} + \frac{w_j}{w_e} - \sum_{e_l \sim i} \frac{w_i}{\sqrt{w_e w_{e_l}}} - \sum_{e_l \sim j} \frac{w_j}{\sqrt{w_e w_{e_l}}} \right)$$

where w_e denotes the weight of the edge e, i.e, (x, y), w_i and w_j are the weights of vertices i and j, respectively. The sums over $e_l \sim k$ run over all edges e_l incident on the vertex k excluding e. Specifically, the curvature with vertex and edge weights set to 1 is

$$\mathrm{Ric}(i, j) := 4 - d_i - d_j + 3|\#\Delta|,$$

where d_i is the degree of node i and $|\#\Delta|$ is the number of 3-cycles (i.e. triangles) containing the adjacent nodes.

Therefore, the overall Forman-Ricci curvature of the graph is the weighted average of the curvature values of all edges.

B.4 Lorentz Transformations

In special relativity, Lorentz transformations are a family of linear transformations that describe the relationship between two coordinate frames in spacetime moving at a constant velocity relative to each other. They can be decomposed into a combination of a Lorentz Boost and a Lorentz Rotation [27]. The Lorentz boost, given a velocity $v \in \mathbb{R}^n$ with $\|v\| < 1$, is represented by the matrix B, which encodes the relative motion with constant velocity without rotation of the spatial axes. The Lorentz rotation matrix R represents the rotation of spatial coordinates and is a special orthogonal matrix, i.e., $R^\top R = I$ and $\det(R) = 1$.

Definition 3 (Lorentz Boost). *A Lorentz boost represents a change in velocity between two coordinate frames without rotation of the spatial axes. Given a velocity* $\mathbf{v} \in \mathbb{R}^n$ *(relative to the speed of light) with* $\|\mathbf{v}\| < 1$, *and the Lorentz factor* $\gamma = \frac{1}{\sqrt{1-\|\mathbf{v}\|^2}}$, *the Lorentz boost matrix is defined as:*

$$\mathbf{B} = \begin{bmatrix} \gamma & -\gamma\mathbf{v}^\top \\ -\gamma\mathbf{v} & \mathbf{I} + \frac{\gamma^2}{1+\gamma}\mathbf{v}\mathbf{v}^\top \end{bmatrix} \tag{6}$$

where $\mathbf{I}$ *is the* $n \times n$ *identity matrix.*

A Lorentz boost describes the geometric transformation between two inertial reference frames moving at a constant relative velocity, which involves a hyperbolic rotation in the space-time plane.

Definition 4 (Lorentz Rotation). *A Lorentz rotation describes a rotation of the spatial coordinates. The Lorentz rotation matrix is defined as:*

$$\mathbf{R} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \tilde{\mathbf{R}} \end{bmatrix} \tag{7}$$

where $\tilde{\mathbf{R}} \in SO(n)$ *is a special orthogonal matrix satisfying* $\tilde{\mathbf{R}}^\top \tilde{\mathbf{R}} = \mathbf{I}$ *and* $\det(\tilde{\mathbf{R}}) = 1$.

A Lorentz rotation represents a geometric rotation or change of orientation in the spatial dimensions of the space-time manifold, while leaving the time dimension unchanged.

Both the Lorentz boost and the Lorentz rotation are linear transformations defined directly in the Lorentz model. For any point $\mathbf{x} \in \mathcal{L}_K^n$, we have $\mathbf{Bx} \in \mathcal{L}_K^n$ and $\mathbf{Rx} \in \mathcal{L}_K^n$.

C Method Supplementary

C.1 Statistics of Forman-Ricci Curvature in Other Datasets

We have calculated the Forman-Ricci curvature (Appendix B.3) for each client in the Cora, Photo, and ogbn-arxiv datasets, which have 10 clients each. The statistics for CiteSeer dataset are shown in Figure 2 Initialization.

C.2 The FlatLand Algorithm

This section introduces the pseudocode of our FlatLand, as shown in Algorithm 1.

```
Algorithm 1: FlatLand
  Input :Personalized parameters $\theta_c^{(0)}, K_c^{(0)}$ and dataset $\mathcal{D}_c$, for
         each client $c \in C$
         Shared parameters $\overline{\theta}_s^{(0)}$
         Learning rate $\eta$
  Output:Client model parameters $\Theta_c = \left(K_c; \theta_c; \overline{\theta}_s\right)$, for each
         client $c \in C$
         Shared parameters $\overline{\theta}_s$
1  Initialize model parameters: $\overline{\theta}_s^{(0)}$ and $\Theta_c^{(0)} = \left(K_c^{(0)}; \theta_c^{(0)}; \overline{\theta}_s^{(0)}\right)$, for
   $c \in C$;
2  for each communication round r do
3      for each client c in C do
4          $\mathbf{x} = \exp_{\mathbf{o}}^{K_c^{(r)}}(\mathbf{x})$, for $\mathbf{x} \in \mathcal{D}_c$;
5          Client $c$ receives global model parameters $\overline{\theta}_s^{(r)}$;
6          $\Theta_c^{(r)} = \left(K_c^{(r)}; \theta_c^{(r)}; \overline{\theta}_s^{(r)}\right)$;
7          for local epochs e do
8              Compute gradients
               $\nabla\mathcal{L} = \nabla_{\Theta^{(r)}} \sum_{(\mathbf{x},y)\in\mathcal{D}_c} \mathcal{L}_c(f(\mathbf{x}; \Theta^{(r)}), y)$;
9          end
10         Update local model $\Theta_c^{(r+1)} \leftarrow \Theta_c^{(r)} - \eta\nabla\mathcal{L}$;
11         Send $\theta_s \in \Theta_c^{(r+1)}$ to the server;
12     end
13     $N = \sum_{c\in C} |\mathcal{D}_c|$;
14     Server aggregates models $\overline{\theta}_s^{(r+1)} \leftarrow \frac{|\mathcal{D}_c|}{N} \sum_{c\in C} \theta_{sc}^{(r+1)}$;
15 end
```

C.3 Derivation of Parameters Disentanglement

The reformulated Lorentz neural network in layer l is shown as

$$\mathbf{x}^{(l+1)} = \mathrm{LT}(\mathbf{x}^{(l)}; \hat{\mathbf{M}}^{(l)}) = \left(\underbrace{\sqrt{\|mx_t + \mathbf{M}\mathbf{x}_s\|^2 + K}}_{\text{time-like dimension } x_t^{(l+1)}}, \quad \underbrace{mx_t + \mathbf{M}\mathbf{x}_s}_{\text{space-like dimensions } \mathbf{x}_s^{(l+1)}}\right)^T. \tag{8}$$

The loss $\mathcal{L}_c(f(\mathbf{x}; \theta_c, \theta_s), y)$ of client c, the partial derivatives can be calculated as follows:

Time-like Dimension $x_t^{(l+1)}$. First, we compute the partial derivative of $x_t^{(l+1)}$ with respect to the matrix $\mathbf{M}^{(l)}$ and $m^{(l)}$. Using the chain rule:

$$\frac{\partial x_t^{(l+1)}}{\partial \mathbf{M}^{(l)}} = \frac{\partial}{\partial \mathbf{M}} \sqrt{\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K};$$
$$\frac{\partial x_t^{(l+1)}}{\partial m^{(l)}} = \frac{\partial}{\partial m} \sqrt{\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K}.$$

Applying the chain rule, we get:

$$\begin{aligned}\frac{\partial x_t^{(l+1)}}{\partial \mathbf{M}^{(l)}} &= \frac{1}{2}\left(\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K\right)^{-\frac{1}{2}} \cdot 2(m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}) \\ &\quad \cdot \frac{\partial(\mathbf{M}^{(l)}\mathbf{x}_s^{(l)})}{\partial \mathbf{M}^{(l)}} \\ &= \frac{m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}}{\sqrt{\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K}} \cdot \frac{\partial(\mathbf{M}^{(l)}\mathbf{x}_s^{(l)})}{\partial \mathbf{M}^{(l)}}\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{\partial x_t^{(l+1)}}{\partial m^{(l)}} &= \frac{1}{2}\left(\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K\right)^{-\frac{1}{2}} \cdot 2(m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}) \\ &\quad \cdot \frac{\partial(m^{(l)}\mathbf{x}_t^{(l)})}{\partial m^{(l)}} \\ &= \frac{(m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)})}{\sqrt{\|m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\|^2 + K}} \cdot x_t^{(l)}\end{aligned} \tag{10}$$

Space-like Dimension $\mathbf{x}_s^{(l+1)}$. Assume that the update rule for the space-like vector $\mathbf{x}_s^{(l+1)}$ is given by the following formula:

$$\mathbf{x}_s^{(l+1)} = m^{(l)}x_t^{(l)} + \mathbf{M}^{(l)}\mathbf{x}_s^{(l)}$$

Similarly, we have

$$\frac{\partial \mathbf{x}_s^{(l+1)}}{\partial \mathbf{M}^{(l)}} = \frac{\partial\left(\mathbf{M}^{(l)}\mathbf{x}_s^{(l)}\right)}{\partial \mathbf{M}^{(l)}}, \quad \frac{\partial \mathbf{x}_s^{(l+1)}}{\partial m^{(l)}} = \frac{\partial\left(m^{(l)}\mathbf{x}_t^{(l)}\right)}{\partial m^{(l)}}. \tag{11}$$

For better illustration, here, we let $\mathbf{x}^{(l)} \in \mathcal{L}_K^n$, $\mathbf{x}^{(l+1)} \in \mathcal{L}_K^n$, and $\hat{\mathbf{M}}^{(l)} \in \mathbb{R}^{(n+1)\times(n+1)}$. **The introduced *"Flatland"* $\mathbb{R}^n$ is defined as a manifold spanning dimensions 1 to n. This construct serves as a metaphorical platform for the exchange and integration of common information, and x_t serves as the heterogeneous information.** Consider the same transformation of a space-like vector $\mathbf{x}_s^{(l)}$ to $\mathbf{x}_s^{(l+1)}$ in different clients, formulated as $\mathbf{x}_s^{(l)} \to \left(\mathbf{M}^{(l)}\mathbf{x}_s^{(l)} + m^{(l)}\mathbf{x}_t^{(l)}\right)$, it is easy to recognize that the gradient of the parameter $m^{(l)}$ depends solely on x_t and K (Equation (10) and Equation (11)). Therefore, the update of parameter $m^{(l)}$ is only related to heterogeneous information and transmitted to the server side for aggregation may lead to performance degradation.

C.4 Analysis

In this section, we provide further analysis to demonstrate the effectiveness and interpretability of our method as described in Section 3.2. Specifically, we first verify the **correctness** that federated learning does not cause the data in each client to deviate from its original space during the process of parameter communication (server updates). Furthermore, we expound on the rationale behind our proposed method from the perspectives of debiasing and Lorentz transformation.

Proposition 1. *$\forall \mathbf{x} \in \mathcal{L}_K^n, \forall \mathbf{M} \in \mathbb{R}^{(m+1)\times(n+1)}$, we have* $\mathrm{LT}(\mathbf{x};\mathbf{M}) \in \mathcal{L}_K^m$.

Proof. $\forall \mathbf{x} \in \mathcal{L}_K^n$, we have $\langle \mathrm{LT}(\mathbf{x};\mathbf{M}), \mathrm{LT}(\mathbf{x};\mathbf{M})\rangle_{\mathcal{L}} = -K$. Therefore, $\mathrm{LT}(\mathbf{x};\mathbf{M}) \in \mathcal{L}_K^m$. □

Corollary 1. *Let $\hat{\mathbf{M}} = \begin{bmatrix} v & \mathbf{v}^T \\ m & \mathbf{M} \end{bmatrix}$, where $\hat{\mathbf{M}} \in \mathbb{R}^{(m+1)\times(n+1)}$ and $\Phi\left(\hat{\mathbf{M}}, \mathbf{N}\right) = \begin{bmatrix} v & \mathbf{v}^T \\ m & \mathbf{N} \end{bmatrix}$. $\forall \mathbf{x} \in \mathcal{L}_K^n, \forall \hat{\mathbf{M}} \in \mathbb{R}^{(m+1)\times(n+1)}, \forall \mathbf{N} \in \mathbb{R}^{n\times n}$, we have $\mathrm{LT}\left(\mathbf{x};\Phi\left(\hat{\mathbf{M}},\mathbf{N}\right)\right) \in \mathcal{L}_K^m$.*

Proof. Let $\mathbf{x} = \begin{bmatrix} x_t \\ \mathbf{x}_s \end{bmatrix} \in \mathcal{L}_K^n$, where $x_t \in \mathbb{R}, \mathbf{x}_s \in \mathbb{R}^n$. According to Equation (2), we have:

$$\mathrm{LT}\left(\mathbf{x};\Phi(\hat{\mathbf{M}},\mathbf{N})\right) = \begin{bmatrix} \sqrt{\|mx_t + \mathbf{N}\mathbf{x}_s\|^2 + K} \\ mx_t + \mathbf{N}\mathbf{x}_s \end{bmatrix}$$

We need to prove that $\mathrm{LT}(\mathbf{x};\Phi(\hat{\mathbf{M}},\mathbf{N})) \in \mathcal{L}_K^m$, i.e., to prove that it satisfies the definition condition of the Lorentz manifold $\langle\cdot,\cdot\rangle_{\mathcal{L}} = -K$:

$$\begin{aligned}
&\left\langle \mathrm{LT}\left(\mathbf{x};\Phi(\hat{\mathbf{M}},\mathbf{N})\right), \mathrm{LT}\left(\mathbf{x};\Phi(\hat{\mathbf{M}},\mathbf{N})\right)\right\rangle_{\mathcal{L}} \\
=&\left\langle \begin{bmatrix} \sqrt{\|mx_t + \mathbf{N}\mathbf{x}_s\|^2 + K} \\ mx_t + \mathbf{N}\mathbf{x}_s \end{bmatrix}, \begin{bmatrix} \sqrt{\|mx_t + \mathbf{N}\mathbf{x}_s\|^2 + K} \\ mx_t + \mathbf{N}\mathbf{x}_s \end{bmatrix}\right\rangle_{\mathcal{L}} \quad \text{(Definition 2)} \\
=&-\left(\sqrt{\|mx_t + \mathbf{N}\mathbf{x}_s\|^2 + K}\right)^2 + \|mx_t + \mathbf{N}\mathbf{x}_s\|^2 \\
=&-K
\end{aligned}$$

Therefore, we have proved that $\mathrm{LT}\left(\mathbf{x};\Phi(\hat{\mathbf{M}},\mathbf{N})\right) \in \mathcal{L}_K^m$. □

This corollary implies that even after the aggregation of shared parameters in the server, the transformation of any client vector $\mathbf{x} \in \mathcal{L}_K^n$ by this updated matrix will still yield results in the Lorentz space $\mathcal{L}_K^m$ with the same curvature, indicating that the client's representation remains unaffected.

Perspectives on Debiasing.

Remark 1 (Feature Debiasing). *During the local and server updates in* FlatLand*, the debiasing process is inherently integrated via the gradient of shared parameters* $\mathbf{M}$.

According to the derivations in Appendix C.3, it can be observed that the gradient of the shared parameters $\mathbf{M}$ is highly correlated with $\mathbf{x}_s$, where $\mathbf{x}_s$ is derived from the raw input $\mathbf{x}^E$ using the exponential map in Equation (4). Therefore, given the same input $\mathbf{x}^E$ for different clients tailored to different Lorentz manifolds, the gradient of $\mathbf{M}$ for client c is inherently weighted by $\sqrt{K_c}\sinh\left(\frac{\|\mathbf{x}^E\|_2}{\sqrt{K_c}}\right)\frac{1}{\|\mathbf{x}^E\|_2}$, where K_c can be intuitively interpreted as the parameter that reflects the overall distribution of the dataset specific to client c, which differs from other clients. This can play a role in debiasing during the parameter aggregation process compared to Euclidean methods.

Perspectives on Lorentz Transformations. Lorentz Boosts and Lorentz Rotations (Appendix B.4) are interpreted as being covered by $\mathrm{LT}\left(\mathbf{x};\hat{\mathbf{M}}\right)$ when the dimension is unchanged [7]. We can easily prove that the Lorentz transformations are still covered by $\mathrm{LT}\left(\cdot;\Phi\left(\hat{\mathbf{M}},\mathbf{N}\right)\right)$, where $\hat{\mathbf{M}} \in \mathbb{R}^{(n+1)\times(n+1)}$, $\mathbf{N} \in \mathbb{R}^{n\times n}$.

For any data point $\mathbf{x} \in \mathcal{D}_c$, transformations $\mathrm{LT}\left(\mathbf{x};\hat{\mathbf{M}}\right)$ and $\mathrm{LT}\left(\mathbf{x};\Phi\left(\hat{\mathbf{M}},\mathbf{N}\right)\right)$ map $\mathbf{x}$ to a new spacetime position, maintaining the spacetime interval invariant (Corollary 1), thus preserving the physical and geometric relationships within the same client, in line with special relativity. However, clients with varying spacetime curvatures maintain **distinct spacetime intervals**, reflecting differing underlying data distributions.

Moreover, according to the definition of Lorentz Rotation in Equation (7), the server updates only the $\mathbf{M}$, leaving the time-like dimension unchanged. This operation is a relaxation of the Lorentz rotation, consistent with our "Flatland" assumption that aggregates only spatial dimension information.

D Experimental Supplementary

D.1 Datasets

For federated node classification, we adopt four benchmark datasets: Cora, CiteSeer, ogbn-arxiv, and Photo [14, 30, 31]. Cora, CiteSeer, and ogbn-arxiv are citation graphs. Photo is a product graph. Each graph dataset is divided into a certain number of disjoint subgraphs using the METIS graph partitioning algorithm [16], where each subgraph belongs to an FL client. Statistics of datasets are summarized in Table 2.

D.2 Implementation Details

We employ 2-layer GCN [18] for Euclidean models, 2-layer LGCN [7] for FlatLand, and HGCN with node selection for FedHGCN [10]. We conduct 100 rounds for Cora/CiteSeer and 200 / 300 rounds for larger datasets like Photo/ogbn-arxiv, with 1-3 local epochs, using 128-dim hidden layers. The learning rate is chosen from $\{0.01, 0.001, 0.002\}$, and weight decay uses from $\{1e-5, 1e-4\}$. We optimize with Adam and calculate the average node-level / graph-level accuracy across clients. All experiments are implemented in Python3.10, PyTorch, and run on an RTX A6000 GPU, 40G storage. Each client is allocated a worker with one round of around 1 second for one epoch in the node classification task.

Table 2: Statistics of node classification datasets. We report the (average) number of nodes, edges, classes, clustering coefficient, and heterogeneity for different numbers of clients.

Dataset	**Cora**			**Citeseer**			**ogbn-arxiv**			**Amazon-Photo**		
# Clients	**1**	**10**	**20**	**1**	**10**	**20**	**1**	**10**	**20**	**1**	**10**	**20**
# Classes		7			6			40			8	
Avg. # Nodes	2,485	249	124	2,120	212	106	169,343	16,934	8,467	7,487	749	374
Avg. # Edges	10,138	891	422	7,358	675	326	2,315,598	182,226	86,755	238,086	19,322	8,547
Avg. Clustering Coefficient	0.238	0.259	0.263	0.170	0.178	0.180	0.226	0.259	0.269	0.410	0.457	0.477
Heterogeneity	N/A	0.606	0.665	N/A	0.541	0.568	N/A	0.615	0.637	N/A	0.681	0.751