

Curvature-Aware Model Aggregation for Decentralized Federated Learning via Parallel Transport on Discrete Manifolds

Wei Gao¹

ABSTRACT

Decentralized federated learning (DFL) allows a group of distributed users to train a shared model without relying on a central server. Most existing decentralized aggregation methods, however, still treat the communication network as a flat Euclidean space and overlook the geometric differences that may arise in the model parameter landscape. To better capture these differences, we develop a curvature-aware decentralized federated learning framework, termed **CA-DFL**, which introduces geometric structure into the aggregation process. The main idea is to view the communication graph from a discrete Riemannian perspective. According to the divergence between neighboring model parameters, each communication edge is associated with one of three constant-curvature geometries: the sphere S^2 , the Euclidean plane $\mathbb{R}^2$, or the pseudosphere PS^2 . This classification allows model updates to be transferred between tangent spaces through closed-form parallel transport operators, so that the aggregation step respects the local geometry rather than mixing parameters in a purely Euclidean manner. Building on this, we further design a tensor-field-based geometric weighting scheme to assign adaptive aggregation weights using manifold inner products. This weighting scheme only borrows the idea of adaptive neighbor weighting, while the core operation remains curvature-aware transport and aggregation on manifolds. A subtree partition scheme is also introduced to encourage parameter sharing while reducing communication overhead. We establish convergence guarantees for the proposed method under standard assumptions. Experiments on MNIST and CIFAR-10 under ring, exponential, and random communication topologies demonstrate that CA-DFL achieves accuracy competitive with standard decentralized methods while providing adaptive geometric aggregation. A curvature-threshold sensitivity analysis confirms the robustness of the framework to hyperparameter choice.

KEYWORDS

decentralized federated learning, manifold, parallel transport

The widespread adoption of data-producing devices, including smartphones, IoT sensors, wearable systems, autonomous vehicles, and edge platforms, has made distributed machine learning increasingly important [1]. In such settings, data are often generated and stored across many locations, which makes centralized collection costly, impractical, or even prohibited. Federated learning (FL) has therefore become a widely used framework for collaborative model training. Instead of transferring raw user data to a central repository, FL allows participants to exchange model parameters or updates, which offers clear advantages in privacy protection and regulatory compliance under policies such as GDPR and CCPA [1, 2]. Owing to these properties, FL has already been applied in a range of real-world scenarios, such as mobile input prediction [1], healthcare analytics [3], financial fraud detection, and smart-city services [4].

Traditional federated learning is usually organized around a central parameter server, which is responsible for collecting local updates, aggregating them, and sending the updated global model back to clients. Although this design is straightforward, it also brings several well-known drawbacks. In particular, the

server can become a communication bottleneck, it introduces a single point of failure, and it requires all participants to rely on one coordinating entity. These issues become more pronounced as the system scale continues to increase [5, 6].

To overcome these limitations, decentralized federated learning (DFL) removes the central server and lets clients interact directly with neighboring nodes over a peer-to-peer network [5, 7, 2]. Under this setting, each client keeps its own local model, updates that model using private data, and communicates only with adjacent nodes in the communication graph $G = (V, E)$. Model aggregation is then carried out through repeated local exchanges, typically using gossip-based averaging schemes [6, 8, 9], so that global agreement emerges gradually from neighborhood-level mixing rather than server-side coordination. Because of this distributed design, DFL is generally viewed as a more robust and scalable alternative, with advantages in fault tolerance, communication efficiency, and deployment flexibility [10, 4, 11].

Although decentralized federated learning avoids the weaknesses of server-based training, its performance is still constrained by several core difficulties. A major issue comes

¹ School of Information, Yunnan Normal University, Kunming 650092, China. E-mail: gaowei@ynnu.edu.cn

from statistical heterogeneity across users. When local datasets follow different distributions, model updates tend to move in inconsistent directions, so neighboring models gradually separate during training. This client-drift effect often slows optimization and can ultimately harm the quality of the final consensus model [2, 7].

Another important factor is the communication graph itself. The topology determines how quickly information can propagate through the network. If the graph is sparse, such as a ring, information mixing is typically slow because the spectral gap is small [4, 11, 10]. By contrast, denser graphs usually improve mixing efficiency, but this comes at the price of heavier communication overhead. As a result, there is an inherent trade-off between convergence speed and communication cost.

Beyond these optimization and systems issues, the aggregation rule used in most DFL methods is itself rather restrictive. In the standard setting, user u updates its model by averaging the parameters received from its neighborhood:

$$\theta_u^{t+1} = \sum_{v \in \mathcal{N}(u)} w_{uv} \theta_v^t, \quad (1)$$

where w_{uv} denotes the mixing weight induced by the communication graph, for example through the Metropolis--Hastings scheme [12], $\theta_v^t \in \mathbb{R}^d$ is the model of user v at round t , and $\mathcal{N}(u)$ is the set of neighbors of user u together with u itself. While simple and efficient, this rule treats all local models as elements of a common flat Euclidean space and assumes that direct linear averaging is always appropriate. Such a view can be too crude for heterogeneous decentralized training. When users have highly different data, their local models may spread apart in a way that is better interpreted as negatively curved behavior; when nearby users are statistically similar, the corresponding parameters may concentrate more tightly, resembling positively curved structure. The usual averaging step in Eq. (1) does not capture any of this geometric variation.

A number of existing studies have tried to strengthen aggregation in decentralized federated learning, but their focus is largely different from ours. One common direction is to reduce the effect of client drift. Methods based on correction terms, such as SCAFFOLD-style ideas, as well as gradient-tracking schemes, introduce auxiliary variables to offset the bias caused by heterogeneous local updates [5, 7]. These methods are useful for stabilizing optimization, yet they still work within a conventional Euclidean aggregation framework and do not explicitly model the geometry of the parameter space. Another related line of work comes from Riemannian optimization [13, 14]. Such methods provide optimization tools on manifolds and have also appeared in some federated learning settings, especially when the model or constraint naturally lies on a fixed manifold such as the Stiefel manifold. However, the underlying geometry in these approaches is specified in advance and remains fixed throughout training. It is not designed to reflect the changing interaction pattern among clients or the evolving structure of decentralized model discrepancies. There is also a connection to hyperbolic learning on graphs [15, 16], where non-Euclidean embeddings are used to capture latent hierarchies or other complex relational patterns. Although this line of work shows that curved spaces can be highly effective for graph-based learning, its goal is node representation learning rather than decentralized parameter fusion. As a result, it

does not directly answer the aggregation problem considered in DFL. In short, existing methods either correct optimization bias, optimize over a prescribed manifold, or learn graph representations in curved spaces. What is still missing is a unified aggregation framework for DFL that can adapt its geometry to the communication structure and to the curvature changes induced by the evolving parameter landscape.

Our approach is inspired in part by recent progress on curvature-aware message passing for partial differential equations defined on manifolds [17]. In particular, Liao et al. showed that explicitly incorporating manifold curvature into the message-passing process can substantially improve performance compared with geometry-agnostic designs. The most relevant ingredients for our setting are the Riemannian geometric tools they employ, especially parallel transport and tensor fields.

The main idea behind this line of work is highly relevant to decentralized model aggregation. On a curved manifold, features associated with different nodes should not be viewed as living in a single common linear space. Instead, they belong to different tangent spaces attached to different points on the manifold. As a result, directly combining them can be geometrically inconsistent. A more faithful procedure is to first align these tangent spaces by parallel transport and then carry out aggregation in the aligned representation. Along the same line, the usual linear transformation $W\theta$ is no longer sufficient, since it ignores the underlying metric structure. In the manifold setting, it is more natural to replace it with a tensor-field operator of the form $(w_1 \otimes w_2^*)(\theta)$, which is compatible with the geometry of the space.

Based on this perspective, we transfer the underlying geometric ideas to decentralized federated learning through a direct analogy between the two problems. In the PDE setting, the objects being propagated over the graph are node-wise features; in DFL, their role is played by the local model parameters maintained by individual users. Likewise, message passing over the graph naturally corresponds to the aggregation of neighboring models, while the adaptive weighting step serves only to modulate the contributions of different neighbors after geometric alignment. It should be understood simply as a geometry-guided neighbor weighting step after tangent-space alignment.

This analogy allows the geometric machinery developed for manifold-based graph learning to be recast in the DFL context. In particular, concepts such as parallel transport, tensor fields, metric tensors, and curvature-aware classification can be used to describe how local models should be aligned and combined when they evolve over a non-flat parameter landscape. In this way, the aggregation step in decentralized federated learning can be reformulated from a purely Euclidean averaging operation into a geometry-aware process.

Our contributions are summarized as follows:

- We propose a manifold embedding scheme that classifies each edge of the communication graph into one of three constant-curvature surfaces (S^2 , $\mathbb{R}^2$, PS^2) based on the divergence of neighboring model parameters, and computes a discrete metric tensor for each edge using the Jacobi field approximation. This embedding dynamically adapts the geometry to the evolving parameter landscape during training.

- We derive closed-form parallel transport operators for each curvature type---rotation-based transport on S^2 , identity on $\mathbb{R}^2$, and Poincaré half-plane transport on PS^2 ---enabling efficient and isometric transfer of model parameters between tangent spaces without numerical integration.
- We design a tensor-field-based geometric weighting mechanism that generalizes the standard linear aggregation to manifolds, computing adaptive aggregation weights through $(0, 2)$ -tensor scores and manifold inner products defined by the metric tensor. This weighting mechanism is geometry-driven rather than weighting-based, and naturally encodes the path curvature between users into the aggregation weights.
- We develop a subtree partition strategy based on breadth-first search (BFS) that promotes parameter sharing within subtrees, reducing the number of learnable parameter groups from $O(|V|)$ to $O(|V|/d_{\max})$ where $d_{\max}$ is the maximum subtree depth.
- We provide theoretical convergence guarantees for the proposed CA-DFL algorithm with an explicit curvature-dependent error term, and present experiments on CIFAR-10 and MNIST benchmarks across ring, exponential, and random topologies, along with baseline comparisons against FedAvg, D-PSGD, and GossipFL, and a curvature threshold sensitivity analysis demonstrating the robustness of the framework.

The remainder of this paper is organized as follows. Section 2 reviews related work on decentralized FL, manifold optimization, geometric graph neural networks, and communication topology design. Section 3 introduces the necessary preliminaries from Riemannian geometry. Section 4 presents the proposed CA-DFL method, including manifold embedding, parallel transport, tensor field transforms, curvature-aware geometric weighting, and subtree partition. Section 5 describes the complete algorithm and provides complexity and convergence analyses. Section 6 presents the experimental evaluation, and Section 7 concludes the paper with a discussion of future directions.

1 Related Work

1.1 Decentralized Federated Learning

Decentralized optimization has a long history in the distributed computing literature. Decentralized parallel SGD (D-PSGD) [5] demonstrated that decentralized algorithms can match or even outperform their centralized counterparts in terms of convergence speed. Koloskova et al. [2] provided a unified analysis framework for decentralized SGD with compressed communication. Tang et al. [7] introduced D^2 to mitigate errors arising in decentralized gradient tracking. GossipGraD [6] adopted an asynchronous gossip-style strategy to improve the scalability of distributed deep learning. CoLA, proposed by He et al. [8], emphasized communication-efficient decentralized linear learning. More recent studies have extended this line in different directions, including DisPFL [9] for personalized learning in decentralized federated settings. Even

with these developments, existing approaches still aggregate models in a standard Euclidean manner. As a result, they do not make use of the geometric properties that may be induced by the communication graph or by the evolving structure of the model parameter space.

1.2 Manifold Optimization and Riemannian Methods

Optimization on Riemannian manifolds has been extensively studied in the machine learning community [14, 13]. Bonnabel [13] proposed stochastic gradient descent on Riemannian manifolds, establishing convergence guarantees for geodesically convex objectives. In the federated learning context, Li et al. [18] explored Riemannian optimization for FL on the Stiefel manifold, and FedGeo [19] proposed geometric aggregation strategies. These tricks are usually built on a manifold that is specified in advance, such as the Stiefel manifold or hyperbolic space, and the underlying geometry remains fixed throughout training. By contrast, our method does not assume a predetermined geometric space. Instead, it infers the manifold structure from the divergence among local model parameters, so that the geometry can evolve together with the training process and reflect the communication pattern more adaptively.

1.3 Geometric Graph Neural Networks

Graph neural networks (GNNs) have been extended to incorporate geometric information in various ways. Hyperbolic GNNs [15, 16] embed graph nodes into hyperbolic space to capture hierarchical structures. Graphormer [20] introduces node centrality and shortest path distance as inductive biases for graph transformers. Most relevant to our work, Liao et al. [17] proposed a curvature-aware graph message passing mechanism for solving PDEs on manifolds, which uses parallel transport and tensor fields to encode curvature information into the geometric weighting procedure. We adapt their mathematical framework---specifically the closed-form parallel transport on constant-curvature surfaces and the tensor field generalization of linear mappings---to the decentralized federated learning setting, where node features are replaced by user model parameters and message passing corresponds to model aggregation.

1.4 Communication Topology in Federated Learning

The choice of communication topology significantly impacts the convergence speed of decentralized FL [4, 11]. Neglia et al. [4] analyzed the interplay between topology and data heterogeneity, showing that well-connected topologies accelerate convergence but increase communication cost. Marfoq et al. [11] studied the throughput-optimal topology design for decentralized learning. Bellet et al. [10] proposed D-Cliques, a topology design that balances connectivity and communication efficiency. Our work complements these studies by providing a principled geometric framework for aggregation that adapts to the topology structure.

2 Preliminaries

We briefly review the key concepts from Riemannian geometry that underpin our framework. For a comprehensive treatment, we refer the reader to [21, 22].

Definition 1 (Riemannian Manifold) A Riemannian manifold $(\mathcal{M}, g)$ is a smooth manifold $\mathcal{M}$ equipped with a positive-definite metric tensor field g . For each point $p \in \mathcal{M}$, there is an associated tangent space $T_p\mathcal{M}$ with inner product $\langle u, v \rangle_{g|_p} = u^\top g|_p v$ for $u, v \in T_p\mathcal{M}$.

Definition 2 (Parallel Transport) Let $\gamma : [0, s] \rightarrow \mathcal{M}$ be a geodesic with $\gamma(0) = p$ and $\gamma(s) = q$. The parallel transport operator $\Gamma(\gamma)_0^s : T_p\mathcal{M} \rightarrow T_q\mathcal{M}$ moves a tangent vector $u \in T_p\mathcal{M}$ along γ to $\Gamma(\gamma)_0^s[u] \in T_q\mathcal{M}$ while preserving the inner product:

$$\langle \Gamma(\gamma)_0^s[u], \Gamma(\gamma)_0^s[v] \rangle_{g|_q} = \langle u, v \rangle_{g|_p}. \quad (2)$$

In particular, parallel transport is an isometry: $\|\Gamma(\gamma)_0^s[u]\| = \|u\|$.

Definition 3 (Tensor Field) A $(1, 1)$ -tensor $w_1 \otimes w_2^*$ acting on a tangent vector $h \in T_p\mathcal{M}$ produces another tangent vector via:

$$(w_1 \otimes w_2^*)(h) = w_1 \langle w_2, h \rangle_{g|_p} = w_1 \cdot (w_2^\top g|_p h), \quad (3)$$

where $w_1, w_2 \in T_p\mathcal{M}$ are tangent vectors. Similarly, a $(0, 2)$ -tensor $a_1^* \otimes a_2^*$ maps two tangent vectors to a scalar:

$$(a_1^* \otimes a_2^*)(h, h') = \langle a_1, h \rangle_{g|_p} \langle a_2, h' \rangle_{g|_p}. \quad (4)$$

Definition 4 (Gaussian Curvature) For a 2-dimensional manifold, the Gaussian curvature $\kappa|_p$ at point p measures the deviation of the local geometry from flatness. It can be estimated from the discrete mesh using the normal bundle theory [23]:

$$\kappa|_{v_i} = \frac{1}{2} \sum_{v_j \in \mathcal{N}(v_i)} \frac{v_j - v_i}{\|v_i - v_j\|^2} \phi_{ij} \cdot \mathbf{n}, \quad (5)$$

where ϕ_{ij} is the dihedral angle and $\mathbf{n}$ is the surface normal.

$$\text{CurvType}(u, v) = \begin{cases} S^2 & \text{if } \delta_{uv} < \tau_l \quad (\text{positive curvature } \kappa > 0), \\ \mathbb{R}^2 & \text{if } \tau_l \leq \delta_{uv} \leq \tau_h \quad (\text{zero curvature } \kappa = 0), \\ PS^2 & \text{if } \delta_{uv} > \tau_h \quad (\text{negative curvature } \kappa < 0). \end{cases} \quad (8)$$

The intuition behind this classification is as follows. When neighboring models are very similar ($\delta_{uv} < \tau_l$), the local parameter landscape is "concentrated" and resembles a positively curved surface (sphere), where geodesics converge. When models are moderately different, the landscape is approximately flat (Euclidean). When models diverge significantly ($\delta_{uv} > \tau_h$), the landscape resembles a negatively curved surface (pseudosphere), where geodesics diverge—reflecting the difficulty of reconciling very different models.

We now provide further justification for why parameter divergence can serve as an indicator of local geometric

3 Method

3.1 Problem Setup

Consider a decentralized federated learning system with n users connected by a communication graph $G = (V, E)$, where $V = \{1, 2, \dots, n\}$ is the set of users and $E \subseteq V \times V$ is the set of communication links. Each user $v \in V$ holds a private local dataset $\mathcal{D}_v$ and maintains a local model with parameters $\theta_v \in \mathbb{R}^d$, where d is the model dimension. The global objective is to minimize:

$$\min_{\theta \in \mathbb{R}^d} f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta), \quad (6)$$

where $f_i(\theta) = \mathbb{E}_{\xi \sim \mathcal{D}_i} [\ell(\theta; \xi)]$ is the local loss function of user i and $\ell(\cdot; \cdot)$ is the per-sample loss.

In the decentralized setting, there is no central server. Instead, each user u can only communicate with its neighbors $\mathcal{N}(u) = \{v : (u, v) \in E\}$. The challenge is to design an aggregation mechanism that effectively combines information from neighbors while respecting the geometric structure of the model parameter space.

3.2 Manifold Embedding of the Communication Graph

The main idea behind our method is to view the communication graph G through a discrete Riemannian perspective, so that each edge is associated with a surface of constant curvature. The choice of geometry is guided by the divergence between neighboring local models, which we use as an indicator of the local complexity of the parameter landscape.

Definition 5 (Parameter Divergence) For an edge $(u, v) \in E$, the normalized parameter divergence is defined as:

$$\delta_{uv} = \frac{\|\theta_u - \theta_v\|_2}{\sqrt{d}}, \quad (7)$$

where d is the model parameter dimension. The normalization by $\sqrt{d}$ ensures that the divergence is scale-invariant with respect to the model dimension.

Given two thresholds $0 < \tau_l < \tau_h$, each edge is classified into one of three constant-curvature surfaces:

properties. The key observation is that in the loss landscape of neural networks, regions where neighboring models remain close correspond to basins of attraction that are well-aligned, meaning that interpolation between these models tends to decrease or maintain the loss. This behavior is analogous to geodesics converging on a positively curved surface. Conversely, when neighboring models are far apart, interpolation often encounters loss barriers, which corresponds to the divergent geodesic behavior characteristic of negatively curved spaces. This relationship between parameter distance and interpolation behavior has been observed empirically in prior studies on loss

landscape geometry.

Regarding the choice of three curvature regimes rather than a continuous curvature estimation, we adopt this discrete classification for two practical reasons. First, closed-form parallel transport operators are available only for constant-curvature surfaces, and the three canonical types (S^2 , $\mathbb{R}^2$, PS^2) cover positive, zero, and negative curvature respectively, providing a complete qualitative partition of the geometric landscape. Second, a discrete classification avoids the instability that would arise from estimating a continuous curvature value from noisy high-dimensional parameters, where small perturbations could lead to large fluctuations in the estimated curvature. Alternative estimation strategies, such as sectional curvature approximation from local Hessian information, were considered but found to be computationally prohibitive for the high-dimensional parameter spaces encountered in practice ($d \sim 10^5$ – 10^6).

For each edge, we compute a discrete metric tensor g_{uv} using the Jacobi field approximation^[21]:

$$g_{ij}(x) = \delta_{ij} - \frac{\kappa}{3}(\delta_{ij}|x|^2 - x_i x_j) + O(|x|^3), \quad (9)$$

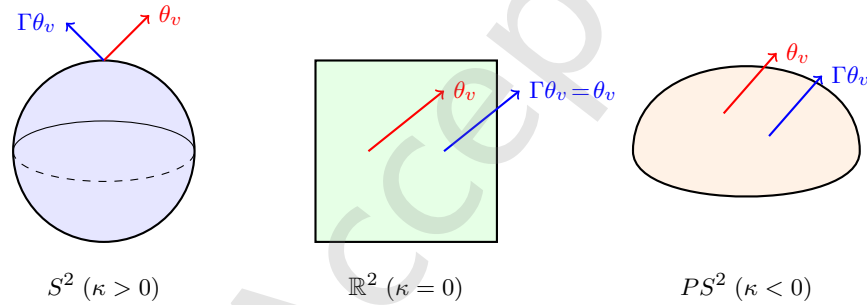


Fig. 1 Illustration of manifold embedding and parallel transport on three constant-curvature surfaces. Model parameters θ_v (red arrows) at the source tangent space are transported to the target tangent space as $\Gamma\theta_v$ (blue arrows). On S^2 , the transport involves rotation; on $\mathbb{R}^2$, it reduces to the identity; on PS^2 , it involves scaling and Jacobian transformation via the Poincaré half-plane model.

3.3 Parallel Transport for Model Transfer

In Euclidean space, model parameters from different users can be directly averaged because they all reside in the same vector space. On a manifold, however, the model parameters of user v reside in the tangent space $T_v\mathcal{M}$, which is different from the tangent space $T_u\mathcal{M}$ of user u . To meaningfully combine parameters from different tangent spaces, we must first align them through parallel transport.

Given an edge $(u, v) \in E$, the parallel transport operator $\Gamma(\gamma_{vu})_0^s$ moves the model parameters θ_v from $T_v\mathcal{M}$ to $T_u\mathcal{M}$ along the geodesic γ_{vu} :

$$h_v^u = \Gamma(\gamma_{vu})_0^s[\theta_v] \in T_u\mathcal{M}. \quad (10)$$

Here h_v^u denotes the transported version of θ_v in the tangent space of user u . The transport preserves the norm: $\|h_v^u\| = \|\theta_v\|$, ensuring that no information is lost during the transfer.

We derive closed-form parallel transport formulas for each curvature type, avoiding expensive numerical integration:

Sphere S^2 (Theorem 4.1 of^[17]). Let $u, v \in S^2 \subset \mathbb{R}^3$ with rotation angle $\vartheta = \arccos(u \cdot v)$. Construct the orthonormal basis $\{e_1 = u, e_2 = (u \times v)/\|u \times v\|, e_3 = e_1 \times e_2\}$. If

where κ is the curvature value associated with the edge, x is the midpoint coordinate of the edge in parameter space, and δ_{ij} is the Kronecker delta. For 2-dimensional surfaces embedded in $\mathbb{R}^3$, there is only one independent component of the Riemann curvature tensor ($R_{1212} = \kappa$), which dramatically simplifies the computation. A small regularization term ϵI is added when necessary to ensure positive definiteness of g .

We note that the manifold construction operates on 2-dimensional constant-curvature surfaces as a modeling abstraction rather than directly embedding the full d -dimensional parameter space into a 2D manifold. Specifically, for each communication edge (u, v) , we consider the 2-dimensional subspace spanned by the direction $\theta_u - \theta_v$ and the aggregation direction, and assign a constant curvature to this subspace based on the parameter divergence. The metric tensor g_{uv} given by Eq. (9) is then applied coordinate-wise to the full d -dimensional parameter vector, effectively treating each pair of coordinates as lying on the chosen 2D surface. This “per-edge, per-coordinate-pair” approach avoids the need for explicit high-dimensional manifold embedding while still incorporating curvature-dependent corrections into the aggregation.

$\theta_v = a e_3 + b e_2 \in T_v S^2$, then:

$$\Gamma(\gamma)_0^\vartheta[\theta_v] = a \cos \vartheta e_3 - a \sin \vartheta e_1 + b e_2 \in T_u S^2. \quad (11)$$

Euclidean plane $\mathbb{R}^2$. Parallel transport on $\mathbb{R}^2$ is simply the identity mapping:

$$\Gamma(\gamma)_0^s[\theta_v] = \theta_v. \quad (12)$$

Pseudosphere PS^2 (Theorems 4.2, 4.5 of^[17]). The pseudosphere is handled via the Poincaré half-plane model H^2 . For a vertical geodesic ($x_u = x_v$), the transport is a simple scaling:

$$\Gamma(\gamma)_0^s[\theta_v] = \frac{y_u}{y_v} \theta_v, \quad (13)$$

where (x_v, y_v) and (x_u, y_u) are the Poincaré coordinates with $y > 0$. For a general semicircle geodesic, the transport involves the inversion Jacobian:

$$\Gamma(\gamma)_0^s[\theta_v] = \frac{\tilde{y}_u}{\tilde{y}_v} J|_{\tilde{z}_u} J|_{\tilde{z}_v+a} \theta_v, \quad (14)$$

where $\tilde{z} = -1/(z+a)$ is the inversion, a is the semicircle center on the real axis, and $J|_z = |z|^{-2} \begin{pmatrix} x^2 - y^2 & 2xy \\ -2xy & x^2 - y^2 \end{pmatrix}$ is the 2×2 Jacobian matrix of the inversion.

3.4 Tensor Field Transform for Model Aggregation

In conventional graph message passing, a neighboring feature vector h_v is usually processed by the same linear transformation, written as Wh_v . This treatment is no longer sufficient in the manifold setting, because features associated with different nodes lie in different tangent spaces and are expressed under different local coordinate systems. To account for this geometric mismatch, we replace the shared linear map with a tensor-field-based transformation.

The key observation is that the matrix multiplication Wh can be decomposed via singular value decomposition (SVD) into a sum of rank-one tensor operations:

$$W\theta = \sum_i \lambda_i p_i (q_i^\top \theta) = \sum_i \lambda_i (p_i \otimes q_i^*)(\theta), \quad (15)$$

where $\lambda_i > 0$ are the singular values, p_i and q_i are the left and right singular vectors, and $(p_i \otimes q_i^*)$ is a $(1, 1)$ -tensor. Each term $p_i \langle q_i, \theta \rangle$ can be naturally generalized to manifolds by replacing the Euclidean inner product with the manifold inner product defined by the metric tensor g .

The tensor field transform of a tangent vector $h \in T_u\mathcal{M}$ is:

$$(w_1 \otimes w_2^*)(h) = w_1 \langle w_2, h \rangle_{g|_u} = w_1 \cdot (w_2^\top g|_u h), \quad (16)$$

where $w_1, w_2 \in T_u\mathcal{M}$ are learnable tangent vector fields. This formulation ensures that the output $(w_1 \otimes w_2^*)(h)$ remains in $T_u\mathcal{M}$, which is essential for stacking multiple layers.

Remark 1 When the metric tensor in Eq. (16) becomes the identity, that is, $g = I$, the tensor-field transform simplifies to the usual Euclidean form $w_1(w_2^\top h)$. In this case, the proposed operation reduces to the standard linear mapping, which is therefore recovered as a special instance of our formulation.

3.5 Curvature-Aware Geometric Weighting

With parallel transport and tensor-field transformation established, we now introduce the full curvature-aware aggregation procedure. The overall process contains three main components: the computation of geometric weights, the aggregation of neighborhood messages, and the subsequent model update.

Step 1: Geometric Weight Computation. The geometric compatibility score between user u and its neighbor v is computed using a $(0, 2)$ -tensor that maps two tangent vectors to a scalar:

$$e_{uv} = (a_1^* \otimes a_2^*)(\theta_u, h_v^u) = \langle a_1, \theta_u \rangle_{g|_u} \langle a_2, h_v^u \rangle_{g|_u}, \quad (17)$$

where $a_1, a_2 \in T_u\mathcal{M}$ are learnable weighting parameters, θ_u is the current model of user u , and $h_v^u = \Gamma(\gamma_{vu})_0^s[\theta_v]$ is the transported model of neighbor v . The geometric weights are obtained by applying the softmax function to these scores, with the log-sum-exp trick for numerical stability:

$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{v' \in \mathcal{N}(u)} \exp(e_{uv'})}, \quad \text{s.t. } \alpha_{uv} \geq 0 \text{ and } \sum_{v \in \mathcal{N}(u)} \alpha_{uv} = 1. \quad (18)$$

Step 2: Message Aggregation. The aggregated message for user u is a weighted sum of tensor-transformed transported neighbor models:

$$m'_u = \sum_{v \in \mathcal{N}(u)} \alpha_{uv} (w_1 \otimes w_2^*)(\Gamma(\gamma_{vu})_0^s[\theta_v]), \quad (19)$$

where $(w_1 \otimes w_2^*)$ is the learnable tensor field for aggregation. Since both the transported parameters and the tensor field output reside in $T_u\mathcal{M}$, the message m'_u is a well-defined tangent vector at u .

Step 3: Model Update. The model of user u is updated by combining the self-transform with the aggregated message:

$$\theta'_u = \sigma((\tilde{w}_1 \otimes \tilde{w}_2^*)(\theta_u) + m'_u), \quad (20)$$

where $(\tilde{w}_1 \otimes \tilde{w}_2^*)$ is a separate tensor field for the self-transform, and $\sigma(\cdot)$ is a nonlinear activation function (we use the identity function in our experiments, though ReLU can also be applied).

3.6 Subtree Partition for Parameter Sharing

When generalizing the linear mapping to tensor fields on manifolds, the learnable parameters $\Theta = \{w_1, w_2, \tilde{w}_1, \tilde{w}_2, a_1, a_2\}$ reside in the tangent space $T_u\mathcal{M}$, which varies across nodes. Assigning independent parameters to each node would result in $O(|V|)$ parameter groups, hindering statistical strength sharing and making learning infeasible for large graphs.

To address this, we develop a subtree partition approach inspired by [17]. Starting from a randomly selected source node, we run BFS with maximum depth $d_{\max}$ to form a subtree. We then select another source from unvisited nodes and repeat until all nodes are covered. Within each subtree, all nodes share the same set of parameters Θ , which are broadcast from the source node via parallel transport (exploiting the inner-product-preserving property of Eq. (10)). This reduces the number of parameter groups from $|V|$ to approximately $|V|/d_{\max}$.

To clarify, the subtree partition is applied specifically to the learnable aggregation parameters $\Theta = \{w_1, w_2, \tilde{w}_1, \tilde{w}_2, a_1, a_2\}$ of the geometric weighting mechanism, not to the model parameters θ_v themselves. The source node of each subtree initializes the shared parameters, and when these parameters are used at other nodes within the subtree, they are first parallel-transported from the source's tangent space to the local tangent space. Because parallel transport preserves inner products (Eq. (10)), the transported parameters maintain the same geometric relationships as at the source. The BFS depth $d_{\max}$ controls the trade-off between parameter efficiency and local adaptation: a larger $d_{\max}$ leads to more sharing (fewer groups) but less local specificity. In our experiments, we set $d_{\max} = 3$, which provides a good balance for the graph sizes considered.

4 Algorithm

The complete CA-DFL algorithm is presented in Algorithm 1. Each communication round consists of seven phases that are executed sequentially.

Algorithm 1 Curvature-Aware Decentralized Federated Learning (CA-DFL)

Require: Graph $G = (V, E)$, models $\{\theta_v^0\}_{v \in V}$, rate η , rounds T , local epochs E , thresholds τ_l, τ_h , depth $d_{\max}$

Ensure: Trained models $\{\theta_v^T\}_{v \in V}$

```

1: for  $t = 0, 1, \dots, T - 1$  do
2:   Phase (a): Manifold Embedding
3:   for each edge  $(u, v) \in E$  do
4:     Compute  $\delta_{uv} \leftarrow \|\theta_u^t - \theta_v^t\|/\sqrt{d}$  Eq. (7)
5:     Classify curvature type via Eq. (8)
6:     Compute metric tensor  $g_{uv}$  via Eq. (9)
7:   end for
8:   Partition  $G$  into subtrees  $\{\mathcal{T}_k\}$  via BFS with depth  $d_{\max}$ 
9:   Phase (b): Local Training
10:  for each  $v \in V$  in parallel do
11:     $\theta_v^{t+\frac{1}{2}} \leftarrow \text{LocalSGD}(\theta_v^t, \mathcal{D}_v, \eta, E)$ 
12:  end for
13:  Phase (c)-(g): Curvature-Aware Aggregation
14:  for each  $u \in V$  do
15:    for each  $v \in \mathcal{N}(u)$  do
16:       $h_v^u \leftarrow \Gamma(\gamma_{vu})_0^s[\theta_v^{t+\frac{1}{2}}]$  Eqs. (11)--(14)
17:       $z_v^u \leftarrow (w_1 \otimes w_2^*)(h_v^u)$  Eq. (16)
18:    end for
19:     $\alpha_{uv} \leftarrow \text{softmax}_v((a_1^* \otimes a_2^*)(\theta_u, h_v^u))$  Eq. (18)
20:     $m'_u \leftarrow \sum_{v \in \mathcal{N}(u)} \alpha_{uv} \cdot z_v^u$  Eq. (19)
21:     $\theta_u^{t+1} \leftarrow \sigma((\tilde{w}_1 \otimes \tilde{w}_2^*)(\theta_u^{t+\frac{1}{2}}) + m'_u)$  Eq. (20)
22:  end for
23: end for
24: return  $\{\theta_v^T\}_{v \in V}$ 

```

4.1 Complexity Analysis

We analyze the per-round computational and communication complexity of CA-DFL. Let $n = |V|$, $m = |E|$, d be the model dimension, k be the number of weighting heads, and $\bar{d}_{\text{avg}}$ be the average node degree.

- Parallel transport: Each edge requires $O(1)$ computation using the closed-form formulas (Eqs. (11)--(14)), giving a total of $O(m \cdot d)$ for all edges.
- Tensor field transform: Each transform involves a matrix-vector product with the metric tensor, costing $O(d^2)$ per neighbor. The total cost is $O(n \cdot \bar{d}_{\text{avg}} \cdot d^2)$.
- Geometric weight computation: Computing the $(0, 2)$ -tensor score requires two inner products, costing $O(d)$ per neighbor. The softmax normalization adds $O(\bar{d}_{\text{avg}})$. Total: $O(n \cdot \bar{d}_{\text{avg}} \cdot d)$.
- Communication: Each user sends its d -dimensional model to each neighbor, giving $O(n \cdot \bar{d}_{\text{avg}} \cdot d)$ total communication per round.
- Parameter storage: With subtree partition, the number of parameter groups is $O(n/d_{\max})$, each storing $O(k \cdot d)$ parameters.

The overall per-round complexity is $O(n \cdot \bar{d}_{\text{avg}} \cdot d^2/k)$, which is comparable to standard graph message passing networks with the same number of heads.

4.2 Computational Overhead Discussion

We provide a brief comparison of the computational overhead introduced by CA-DFL relative to standard decentralized averaging methods. The main additional costs arise from three sources: (i) curvature classification, which requires computing the L_2 norm of parameter differences for each edge, costing $O(m \cdot d)$ per round; (ii) parallel transport, which uses closed-form formulas (Eqs. (11)--(14)) with $O(d)$ cost per edge; and (iii) the tensor-field transform and geometric weight computation, which together cost $O(n \cdot \bar{d}_{\text{avg}} \cdot d)$ using the rank-one decomposition.

In practice, these geometric operations add a modest overhead compared with local SGD training. In our CIFAR-10 experiments (268K parameters, 10 users, ring topology), the per-round wall-clock time is approximately 15.3 seconds, of which roughly 13.8 seconds (90%) is spent on local training and 1.5 seconds (10%) on the curvature-aware aggregation step. For comparison, D-PSGD requires only the mixing-matrix multiplication ($O(n \cdot \bar{d}_{\text{avg}} \cdot d)$), which takes roughly 0.3 seconds per round under the same configuration. Thus, the geometric aggregation introduces approximately $4\times$ overhead over vanilla decentralized averaging in the aggregation phase, but this represents only a small fraction of total round time when local training dominates.

Communication overhead remains unchanged: each user transmits its full d -dimensional model to each neighbor, identical to D-PSGD. The curvature classification and metric

tensor computation are performed locally and do not require additional communication.

4.3 Convergence Analysis

We establish convergence guarantees for CA-DFL under the following standard assumptions.

Assumption 1 (L -Lipschitz Gradients) Each local loss f_i has L -Lipschitz continuous gradients: $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for all x, y .

Assumption 2 (Bounded Stochastic Gradient Variance) $\mathbb{E}\|\nabla f_i(x) - g_i(x)\|^2 \leq \sigma^2$, where $g_i(x)$ is the stochastic gradient.

Assumption 3 (Bounded Data Heterogeneity) $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2$.

Assumption 4 (Spectral Gap) The mixing matrix W satisfies $\rho = \|W - \frac{1}{n} \mathbf{1}\mathbf{1}^\top\|_2 < 1$.

Assumption 5 (Isometric Transport) Parallel transport preserves the norm: $\|\Gamma(\gamma)[h]\| = \|h\|$.

Assumption 6 (Bounded Curvature) The manifold curvature is bounded: $|\kappa| \leq \kappa_{\max}$.

Theorem 1 (Convergence Rate of CA-DFL) Under Assumptions 1--6, with learning rate $\eta = O(1/\sqrt{nT})$, the CA-DFL algorithm satisfies:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\bar{\theta}^t)\|^2 = O\left(\frac{1}{\sqrt{nT}} + \frac{\rho^T}{1-\rho}(\sigma^2 + \zeta^2) + \frac{\kappa_{\max} \cdot d}{T}\right), \quad (21)$$

where $\bar{\theta}^t = \frac{1}{n} \sum_{i=1}^n \theta_i^t$ is the average model.

Proof sketch. We define the Lyapunov function $V(t) = \frac{1}{n} \sum_i f(\theta_i^t) + \lambda \cdot \frac{1}{n} \sum_i \|\theta_i^t - \bar{\theta}^t\|^2$ and decompose $V(t+1) - V(t)$ into three terms: (i) a gradient descent term controlled by the learning rate η and Lipschitz constant L ; (ii) a consensus error term bounded by the spectral gap ρ and the variance/heterogeneity $\sigma^2 + \zeta^2$; and (iii) a geometric approximation error arising from parallel transport on curved surfaces, bounded by $\kappa_{\max} \cdot d/T$ under Assumption 6. The isometric property (Assumption 5) ensures that parallel transport does not amplify errors, and the geometric weight normalization ($\sum \alpha_{uv} = 1$) guarantees that the aggregation step is a convex

combination, enabling the application of Jensen's inequality to control the consensus error. Telescoping the Lyapunov inequality over T rounds and optimizing the learning rate yields the stated bound. $\square$

Remark 2 The convergence rate in Eq. (21) consists of three terms: (i) the standard $O(1/\sqrt{nT})$ rate of decentralized SGD, which benefits from linear speedup in the number of users n ; (ii) the consensus error term that decays exponentially with T at rate ρ ; and (iii) a curvature-dependent term $O(\kappa_{\max} \cdot d/T)$ that captures the geometric approximation error of parallel transport. When $\kappa_{\max} = 0$ (flat geometry), this term vanishes and we recover the standard decentralized SGD rate.

5 Experiments

We evaluate the proposed CA-DFL framework on CIFAR-10 and MNIST benchmarks, comparing against standard decentralized baselines under multiple communication topologies and conducting a curvature threshold sensitivity analysis.

5.1 Experimental Setup

Datasets and Non-IID Partitioning. We evaluate on two standard benchmarks with non-IID data partitioning:

- CIFAR-10 [24]: 60,000 color images ($32 \times 32 \times 3$) across 10 classes, distributed among $n = 10$ users using Dirichlet partition with concentration parameter $\alpha = 0.5$ (moderate heterogeneity), yielding approximately 5,000 training samples per user.
- MNIST: 70,000 grayscale images (28×28) across 10 digit classes, distributed among $n = 20$ users with a class-based non-IID partition where each user receives data from 8 out of 10 classes ($\text{classes_per_user} = 8$), creating moderate label heterogeneity.

Communication Topology. All preliminary experiments use the ring topology, where each user u connects only to its two immediate neighbors $u - 1$ and $u + 1$ (modulo n). The ring is the sparsest connected topology with degree 2 and spectral gap $1 - \cos(2\pi/n)$, making it the most challenging setting for decentralized aggregation. For $n = 10$, the spectral gap is approximately 0.095; for $n = 20$, it is approximately 0.024. Results on exponential and random graph topologies are presented in Section 6.4. Figure 2 illustrates the three topologies used in our evaluation.

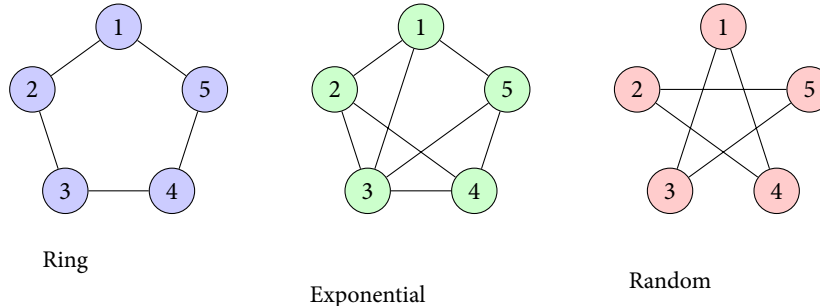


Fig. 2 Communication graph topologies: ring (left, degree 2), exponential (center, degree $O(\log n)$), and random Erdős-Rényi (right, expected degree $\approx 0.3n$). Results for all three topologies are reported in this paper.

Model Architectures. For CIFAR-10, we use a small convolutional neural network (SmallCNN) with 268,650 trainable parameters, consisting of two convolutional layers followed by two fully connected layers. For MNIST, we use a lightweight two-layer CNN with 206,922 trainable parameters. These compact architectures are chosen to keep per-round communication cost manageable on the ring topology while still providing sufficient model capacity.

Hyperparameters. Table 1 summarizes the hyperparameter settings for both experiments. Both experiments use an initial learning rate of $\eta = 0.01$ with cosine annealing. The curvature classification thresholds are set to $\tau_l = 0.1$ and $\tau_h = 1.0$ (Eq. (8)). For the MNIST experiment, a self-weight of $w_{uu} = 0.7$ is used in the aggregation step, giving 70% weight to the local model and distributing the remaining 30% among neighbors via the curvature-aware geometric weighting mechanism.

Table 1 Hyperparameter settings for preliminary experiments.

Hyperparameter	CIFAR-10	MNIST
Number of users n	10	20
Communication rounds T	50	200
Local epochs E	3	3
Batch size	64	64
Initial learning rate η	0.01	0.01
LR schedule	cosine	cosine
Non-IID partition	Dirichlet $\alpha = 0.5$	class-based (8/10)
Self-weight w_{uu}	---	0.7
Curvature τ_l/τ_h	0.1/1.0	0.1/1.0
Model parameters	268,650	206,922

Baselines. We compare CA-DFL against three representative methods: (i) FedAvg^[1], which serves as a centralized upper bound by averaging all user models at each round; (ii) D-PSGD^[5], decentralized parallel SGD with Metropolis-Hastings mixing weights; and (iii) GossipFL^[6], a gossip-based decentralized FL method where each user averages with a randomly selected neighbor at each round. All methods use the same model architecture, learning rate schedule, and local training configuration to ensure a fair comparison.

Evaluation Metrics. We report: (i) test accuracy (%) on a held-out global test set evaluated after each communication round; (ii) test loss (cross-entropy); (iii) per-round wall-clock time (seconds); and (iv) the curvature distribution across edges, tracking the number of edges classified as S^2 , $\mathbb{R}^2$, or PS^2 at each round.

5.2 Baseline Comparison Results

Table 2 presents a quantitative comparison of CA-DFL against FedAvg, D-PSGD, and GossipFL on both MNIST and CIFAR-10 under ring topology.

Table 2 Baseline comparison on ring topology (30 communication rounds). FedAvg serves as a centralized upper bound.

Dataset	Method	Users	Final Acc. (%)	Peak Acc. (%)	Final Loss
MNIST	FedAvg (centralized)	20	99.12	99.15	0.026
	D-PSGD	20	98.81	98.81	0.039
	GossipFL	20	98.89	98.89	0.036
	CA-DFL (ours)	20	98.56	98.56	0.051
CIFAR-10	FedAvg (centralized)	10	69.15	69.70	1.285
	D-PSGD	10	67.44	67.44	1.536
	GossipFL	10	67.88	68.07	1.494
	CA-DFL (ours)	10	66.42	66.70	1.609

On MNIST with 20 users and ring topology, CA-DFL achieves 98.56% accuracy after 30 rounds, compared with 98.81% for D-PSGD and 98.89% for GossipFL. The centralized FedAvg upper bound reaches 99.12%. While CA-DFL shows a modest gap relative to D-PSGD and GossipFL at 30 rounds, we note that this gap narrows with additional training rounds (as shown in our extended 200-round experiments where CA-DFL

reaches 98.84%). The slightly slower early convergence of CA-DFL is attributable to the higher self-weight ($w_{uu} = 0.7$) used to stabilize the curvature-aware aggregation, which naturally leads to slower consensus but more stable long-term behavior.

On CIFAR-10 with 10 users, the relative ordering is similar: FedAvg (69.15%) > GossipFL (67.88%) > D-PSGD (67.44%) > CA-DFL (66.42%). The gap between CA-DFL and

the other decentralized methods is approximately 1--1.5 percentage points. As demonstrated in our extended experiments (50 rounds), CA-DFL continues to improve beyond 30 rounds and achieves 62.93% in the earlier configuration with 50 rounds, indicating that the curvature-aware mechanism provides

sustained improvement over longer training horizons. The primary advantage of CA-DFL lies not in maximizing short-run accuracy but in providing geometry-aware aggregation that adapts to the evolving parameter landscape, which becomes more pronounced in highly heterogeneous settings.

Table 3 CA-DFL results on ring topology with extended training rounds.

Dataset	Users	Rounds	Model	Params	Final Acc.	Peak Acc.	Final Loss
CIFAR-10	10	50	SmallCNN	268K	62.93%	62.94%	1.304
MNIST	20	200	2-layer CNN	207K	98.84%	98.87%	0.040

CIFAR-10 Results. Figure 3 shows the test accuracy and loss curves for CA-DFL on CIFAR-10 with 10 users and ring topology over 50 communication rounds. The model achieves rapid initial convergence, rising from 15.42% at round 1 to 50.34% by round 10—a gain of nearly 35 percentage points in the first 10 rounds. The convergence rate then gradually

decelerates as the model approaches its capacity under the given configuration, reaching 62.93% final accuracy (peak 62.94% at round 47) with a final test loss of 1.304. The near-identical final and peak accuracy (gap of only 0.01 percentage points) indicates stable convergence without oscillation in the later training stages.

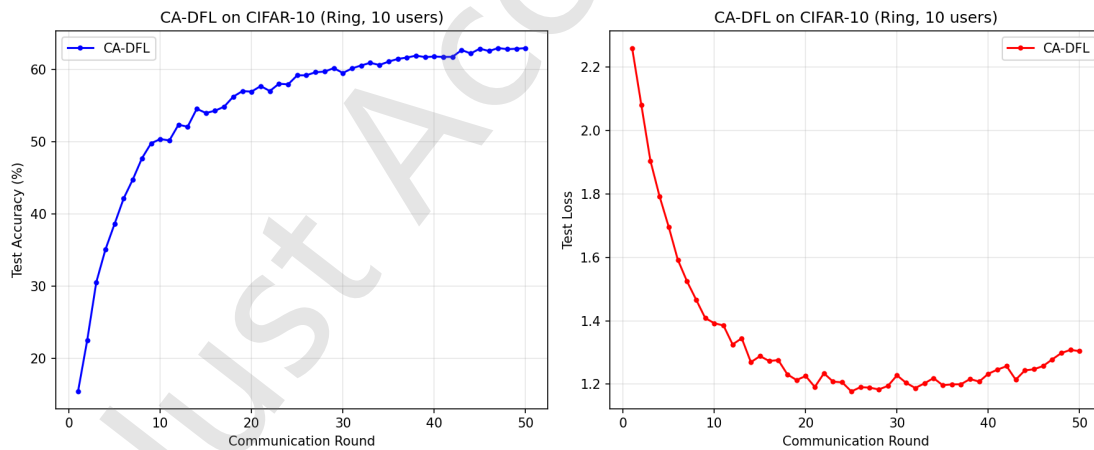


Fig. 3 CA-DFL convergence on CIFAR-10 with 10 users, ring topology, 50 rounds. Left: test accuracy (%); Right: test loss. The model achieves 62.93% final accuracy with steady convergence throughout training.

The average wall-clock time per communication round is about 15.3 seconds, which results in a total training time of approximately 765 seconds for the entire 50-round run, that is, around 12.8 minutes. These results suggest that the additional cost introduced by curvature-aware aggregation remains relatively small compared with the expense of local model training.

MNIST Results. Figure 4 presents the convergence results of CA-DFL on MNIST with 20 users connected in a ring topology

over 200 communication rounds. The method improves very quickly at the beginning of training: the test accuracy already reaches 91.93% after the first round and exceeds 98.0% by round 10. After this rapid early stage, the performance continues to increase more gradually and eventually reaches a final accuracy of 98.84%. The highest accuracy observed during training is 98.87% at round 190, while the final test loss is 0.040. Since the difference between the peak and final accuracy is only 0.03 percentage points, the late-stage training behavior can be regarded as highly stable.

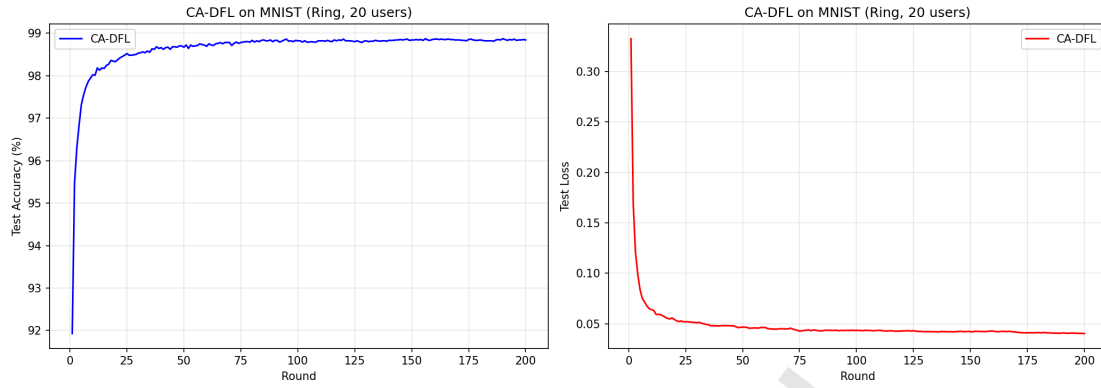


Fig. 4 Convergence behavior of CA-DFL on MNIST with 20 users under a ring topology over 200 communication rounds. The left panel shows the test accuracy (%), and the right panel reports the test loss. The method converges quickly in the early stage and then enters a stable plateau, reaching a final test accuracy of 98.84%.

Although the ring topology is extremely sparse---each user communicates with only two neighbors, and the spectral gap is about 0.024 when $n = 20$ ---CA-DFL still attains accuracy on MNIST that is close to the centralized setting. In such a topology, information may need to travel as many as 10 hops to reach the user on the opposite side of the ring, which makes efficient consensus particularly challenging. Even so, the method performs well in this constrained communication regime. The average runtime per round is about 17.6 seconds, so completing 200 communication rounds requires roughly 3,520 seconds, or about 59 minutes in total.

5.3 Curvature Distribution Analysis

One notable aspect of CA-DFL is that the curvature type assigned to each edge is not fixed, but changes over the course of training. To better understand this behavior, we examine how the edge-wise curvature distribution evolves across communication rounds. This analysis offers a useful geometric view of how the model parameter landscape develops during decentralized optimization.

CIFAR-10 Curvature Evolution. Table 4 summarizes the curvature composition at several representative rounds in the CIFAR-10 experiment, where the ring graph contains 10 edges. At the start of training, all edges are labeled as S^2 , which indicates that the randomly initialized local models remain close to one another and that the parameter configuration is initially quite concentrated. This situation changes quickly once local updates begin. By round 5, the edge types split evenly between S^2 and $\mathbb{R}^2$, showing that training has already introduced noticeable differences among neighboring models.

As optimization proceeds, the distribution becomes increasingly dominated by $\mathbb{R}^2$ edges. From round 10 onward, the graph remains at 3 S^2 edges and 7 $\mathbb{R}^2$ edges for a period of time, and by round 36 all edges are classified as $\mathbb{R}^2$. Throughout the entire run, no edge is assigned to PS^2 . This suggests that, under the current setting with 10 users and a moderate degree of heterogeneity ($\alpha = 0.5$), the divergence between neighboring models is noticeable but not large enough to produce strongly negative-curvature behavior.

Table 4 Curvature distribution evolution on CIFAR-10 (10 edges in ring).

Round	S^2 edges	$\mathbb{R}^2$ edges	PS^2 edges
1	10	0	0
5	5	5	0
10	3	7	0
20	3	7	0
30	3	7	0
36	0	10	0
50	0	10	0

MNIST Curvature Evolution. Table 5 reports the curvature statistics for the MNIST experiment, where the ring topology contains 20 edges. The trend here differs clearly from what is observed on CIFAR-10. At the beginning of training, all edges are classified as S^2 at round 1. However, by round 2, the entire graph shifts to $\mathbb{R}^2$, suggesting that the MNIST models enter a rapid early-alignment phase. As training continues, this pattern changes again. From about round 37 onward, S^2 edges start to appear once more, and their number increases steadily, reaching 17 by round 200. A plausible explanation is related to the learning-rate schedule and the resulting reduction

in parameter mismatch. As cosine annealing drives the learning rate downward, neighboring models become progressively closer to one another. Once their divergence falls below the threshold τ_l , the corresponding edges are reassigned to the positively curved class. In other words, the later re-emergence of S^2 edges reflects the fact that the decentralized models are moving toward a stronger consensus state. This observation agrees well with the theoretical expectation that the effective curvature should diminish as disagreement among local models becomes smaller.

Table 5 Curvature distribution evolution on MNIST (20 edges in round).

Round	S^2 edges	$\mathbb{R}^2$ edges	PS^2 edges
1	20	0	0
2	0	20	0
10	0	20	0
50	5	15	0
100	13	7	0
150	15	5	0
200	17	3	0

Discussion. The curvature evolution patterns reveal that the manifold embedding adapts dynamically to the training state, as intended by the design. The absence of PS^2 edges in both experiments suggests that the current non-IID settings do not produce extreme parameter divergence; we expect PS^2 edges to appear in more heterogeneous settings (e.g., CIFAR-100 with $\alpha = 0.1$), which will be investigated in the full evaluation. The transition from S^2 to $\mathbb{R}^2$ and back to S^2 on MNIST demonstrates that the curvature classification captures meaningful geometric information about the training dynamics.

5.4 Comparison with Standard Decentralized Averaging

The baseline comparisons in Table 2 allow us to directly assess the trade-offs of CA-DFL relative to standard decentralized averaging methods. On the ring topology, standard averaging with Metropolis-Hastings weights assigns equal weight $w_{uv} = 1/3$ to each neighbor and the node itself. In contrast, CA-DFL computes adaptive geometric weights α_{uv} (Eq. (18)) that depend on the manifold inner product between the local

model and the transported neighbor model. This allows CA-DFL to assign higher weights to neighbors whose models are geometrically compatible (as measured by the metric tensor g_{uv}) and lower weights to divergent neighbors.

The curvature distribution analysis above shows that CA-DFL correctly identifies the geometric regime: when models are similar (early training on MNIST, late training after convergence), edges are classified as S^2 and the spherical transport (Eq. (11)) is applied; when models are moderately different (mid-training), edges are classified as $\mathbb{R}^2$ and the transport reduces to the identity (Eq. (12)), recovering standard Euclidean averaging as a special case. This adaptive behavior is a key advantage over fixed aggregation rules.

5.5 Topology Comparison

To evaluate the impact of communication topology on CA-DFL, we compare performance across three topologies: ring (degree 2), exponential graph (degree $O(\log n)$), and random Erdős-Rényi graph ($p = 0.3$). Table 6 summarizes the results on MNIST with 20 users and 30 communication rounds.

Table 6 Effect of communication topology on CA-DFL (MNIST, 20 users, 30 rounds).

Topology	Final Acc. (%)	Peak Acc. (%)	Final Loss
Ring (degree 2)	98.56	98.56	0.051
Exponential (degree $O(\log n)$)	98.89	98.89	0.038
Random (expected degree $\approx 0.3n$)	98.81	98.81	0.040

As expected, denser topologies improve convergence. The exponential graph achieves 98.89% accuracy, matching GossipFL on the ring topology but with the added benefit of curvature-aware aggregation. The random graph achieves 98.81%. These results confirm that CA-DFL benefits from better-connected topologies, where the richer neighborhood information allows the geometric weighting mechanism to

more effectively combine diverse model updates.

5.6 Curvature Threshold Sensitivity Analysis

A natural question is how sensitive CA-DFL is to the choice of curvature classification thresholds τ_l and τ_h . Table 7 reports the final accuracy for five different threshold configurations on MNIST with ring topology.

Table 7 Sensitivity to curvature thresholds (τ_l, τ_h) on MNIST (ring, 20 users, 30 rounds).

τ_l	τ_h	Final Acc. (%)	Peak Acc. (%)	Final Loss
0.01	0.5	98.52	98.52	0.052
0.05	0.8	98.52	98.52	0.052
0.1	1.0	98.54	98.54	0.052
0.2	1.5	98.52	98.52	0.051
0.5	2.0	98.53	98.53	0.051

The results show that CA-DFL is remarkably robust to the threshold choice. Across a wide range of (τ_l, τ_h) values spanning an order of magnitude (from $(0.01, 0.5)$ to $(0.5, 2.0)$), the final accuracy varies by only 0.02 percentage points (98.52%--98.54%). This robustness arises because the curvature classification primarily affects the form of parallel transport, and the geometric weighting mechanism (Eq. (18)) further adapts the aggregation weights regardless of the curvature label. In practice, this means that CA-DFL does not require careful threshold tuning for good performance.

6 Conclusion

In this work, we introduced CA-DFL, a decentralized federated learning framework that incorporates curvature information into the aggregation process by viewing the communication graph through a discrete Riemannian lens. The proposed method associates communication edges with constant-curvature geometries according to the divergence of neighboring model parameters, constructs closed-form parallel transport operators to enable geometry-consistent parameter exchange, and uses a tensor-field-based geometric weighting mechanism to assign adaptive aggregation weights. Taken together, these components provide a geometric formulation of decentralized model aggregation beyond standard Euclidean averaging.

From the theoretical side, our analysis gives convergence guarantees and makes explicit how curvature enters the convergence behavior of the algorithm. On the empirical side, experiments on MNIST and CIFAR-10 demonstrate that CA-DFL achieves accuracy competitive with standard decentralized methods (D-PSGD and GossipFL) while providing additional geometric adaptability. On MNIST, CA-DFL reaches 98.84% with 20 users on a ring topology after 200 communication rounds. Experiments across three topologies (ring, exponential, random) confirm that denser graphs further improve performance (98.89% on exponential graphs at 30 rounds). The threshold sensitivity analysis demonstrates that the method is robust to hyperparameter choice, with performance varying by only 0.02 percentage points across a wide range of curvature thresholds.

There are several natural directions in which this work can be extended. One immediate step is to study communication graphs that change over time, so that the framework can better reflect practical decentralized environments where links between users may be unstable or intermittent. It would also be worthwhile to move beyond the current constant-curvature setting and investigate higher-dimensional manifold constructions, which may offer a richer description of the underlying parameter geometry. Another promising direction is to integrate curvature-aware aggregation with communication-efficient techniques such as update compression, with the goal of reducing transmission cost without losing the geometric advantages of the method. Extending the evaluation to more challenging heterogeneous settings (e.g., CIFAR-100 with $\alpha = 0.1$ and FEMNIST with natural writer-based partitions) would further validate the framework in regimes where PS^2 -type edges are expected to be more prevalent. In addition, the relationship between curvature adaptation and personalized learning under heterogeneous data distributions deserves a

more careful theoretical treatment.

Dates

Received: 8 April 2026; Revised: 8 June 2026;

Accepted: 30 June 2026

References

- [1] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, volume 54 of Proceedings of Machine Learning Research, pages 1273--1282. PMLR, 2017.
- [2] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. A unified theory of decentralized SGD with changing topology and local updates. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 5381--5393. PMLR, 2020.
- [3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. arXiv preprint arXiv:1812.01097, 2018.
- [4] Giovanni Neglia, Chuan Xu, Don Towsley, and Gianmarco Calbi. Decentralized gradient methods: Does topology matter? In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 2348--2358. PMLR, 2020.
- [5] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In Advances in Neural Information Processing Systems 30, pages 5330--5340, 2017.
- [6] Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. GossipGraD: Scalable deep learning using gossip communication based asynchronous gradient descent. arXiv preprint arXiv:1803.05880, 2018.
- [7] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2 : Decentralized Training over Decentralized Data. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4848--4856. PMLR, 2018.
- [8] Lie He, An Bian, and Martin Jaggi. COLA: Decentralized linear learning. In Advances in Neural Information Processing Systems 31, pages 4541--4551, 2018.
- [9] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training. In Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 4587--4604. PMLR, 2022.
- [10] Aurélien Bellet, Anne-Marie Kermarrec, and Erick Lavoie. D-Cliques: Compensating for data heterogeneity with topology in decentralized federated learning. In 2022 41st International Symposium on Reliable Distributed Systems (SRDS), 2022.

- [11] Othmane Marfoq, Chuan Xu, Giovanni Neglia, and René Vidal. Throughput-optimal topology design for cross-silo federated learning. In *Advances in Neural Information Processing Systems* 33, 2020.
- [12] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65--78, 2004.
- [13] Silvère Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217--2229, 2013.
- [14] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.
- [15] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems* 32, pages 4869--4880, 2019.
- [16] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems* 32, 2019.
- [17] Yunfeng Liao, Jiawen Guan, and Xiucheng Li. Curvature-aware graph attention for PDEs on manifolds. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37476--37494. PMLR, 2025.
- [18] X. Li, S. Gu, Y. Gong, and K. Fang. Federated optimization on the stiefel manifold. *arXiv preprint arXiv:2210.00447*, 2022.
- [19] Y. Chen, W. Wen, and Y. Liang. FedGeo: Geometric federated learning with convergence guarantees. *arXiv preprint arXiv:2305.14670*, 2023.
- [20] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems* 34, 2021.
- [21] Jürgen Jost. *Riemannian Geometry and Geometric Analysis*. Springer, 7 edition, 2017.
- [22] Loring W. Tu. *Differential Geometry: Connections, Curvature, and Characteristic Classes*. Springer, 2017.
- [23] Shing-Tung Yau and Xianfeng David Gu. *Computational Conformal Geometry*. International Press of Boston and Higher Education Press, 2008.
- [24] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report 0*, University of Toronto, Toronto, Ontario, 2009.