Prototype Normalization: Optimizing Prototype Separation for Heterogeneous Federated Learning

Daeyoung Choi[1,2], Jihwan Shin[3], and Gyuejeong Lee[1,3*]

[1] Korea Cyber University
[2] Intellectus
[3] SAKAK Inc.
{choidy,lgj413}@koreacu.ac.kr, jihwan@sakak.co.kr

Abstract. Heterogeneity in data distributions and model architectures poses significant challenges in federated learning (FL). While various heterogeneous FL (HtFL) approaches have been proposed to address these challenges, prototype-based FL (PBFL) has emerged as a promising framework that exchanges prototypes—per-class mean activations from the penultimate layer—rather than model parameters. However, existing PBFL methods suffer from suboptimal prototype separation in the feature space, which limits their discriminative capacity. We propose Prototype Normalization (ProtoNorm), a novel PBFL framework that systematically addresses this limitation. Inspired by the Thomson problem in classical physics, ProtoNorm optimizes global prototype configurations on a unit hypersphere, maximizing separation between class prototypes. Extensive experimental evaluation demonstrates that ProtoNorm consistently outperforms existing HtFL methods across diverse heterogeneous settings. Importantly, ProtoNorm maintains communication efficiency while operating entirely on the server side with less computation than existing PBFL methods, making it well-suited for resource-constrained environments.

1 Introduction

Federated learning (FL) enables collaborative model training across distributed clients without raw data sharing, demonstrating remarkable success in privacy-sensitive domains, including healthcare and finance applications [15, 33, 46, 48]. As a pioneering FL algorithm, Federated Averaging (FedAvg) [30] aggregates local model parameters at a central server to create a single global model, which is then distributed back to clients for local training. However, its effectiveness is limited to scenarios with nearly uniform data distributions across homogeneous model architectures. Other traditional FL approaches based on model parameter sharing often face the same limitations. Data and model heterogeneity across clients is prevalent in practice, emerging as a fundamental challenge that demands novel solutions [20, 38].

* Corresponding author.

To address these challenges, heterogeneous FL (HtFL) has emerged as a new paradigm that enables clients to employ different model architectures and handle non-uniform data distributions [41]. One line of approaches adapts parameter sharing by enabling clients to train architecture-specific subnets based on their available resources, addressing both system and statistical challenges [2, 9, 11, 53]. However, these approaches inherently limit the degree of heterogeneity they can accommodate, as they require architectural compatibility among shared components. Knowledge distillation (KD) based approaches have been proposed as an alternative that enables an entirely architecture-free federation. These approaches focus on knowledge sharing between the server and clients through various representations, including logits [12, 19], intermediate features [41, 49], and auxiliary networks [35, 54], offering greater flexibility in handling heterogeneous architectures.

Among KD-based approaches, prototype-based FL (PBFL) offers a practical framework by sharing prototypes—per-class mean activations from the penultimate layer [41]. PBFL offers several advantages: efficient communication via compact feature vectors, enhanced privacy protection, and simpler implementation. However, existing PBFL methods suffer from a fundamental limitation: global prototypes are constructed through weighted averaging of local prototypes, with no explicit optimization for inter-class separation. While recent works such as FedTGP [49] and FedFTP [47] improve separability through contrastive learning, they do not explicitly optimize global geometric configurations, leaving room for improvement. Our motivational experiment on a synthetic spiral dataset visually demonstrates this limitation, showing that existing PBFL approaches fail to separate global prototypes under non-IID conditions adequately.

To address this limitation, we propose Prototype Normalization (ProtoNorm), a novel PBFL framework inspired by a key insight from classical physics: prototypes can be viewed as charged particles on a hypersphere, where optimal separation corresponds to minimum energy configurations—analogous to the Thomson problem of distributing electrons on a sphere. This geometric perspective enables us to systematically optimize prototype configurations through hyperspherical energy minimization rather than relying on implicit separation from weighted averaging or contrastive learning. ProtoNorm employs a novel gradient-based solver that iteratively adjusts prototype positions on a unit hypersphere to maximize angular separation, followed by magnitude upscaling to enhance class distinguishability in Euclidean space. Critically, this optimization occurs entirely on the server side, imposing no additional computational burden on clients. Together, these design choices yield a simple yet effective framework for heterogeneous FL. We summarize our main contributions as follows:

1) We develop a novel gradient-based algorithm that optimizes prototype separation by solving the Thomson problem entirely on the server side, replacing semantic alignment with pure geometric optimization.

2) ProtoNorm consistently outperforms existing HtFL methods across various image and text classification benchmarks, with particularly strong gains on many-

class tasks. Ablation studies further confirm that both prototype alignment and upscaling are essential for performance improvement.
3) ProtoNorm solves the Thomson problem once on the server and distills the optimized prototypes to clients. This supports **model heterogeneity**, as only prototypes are exchanged, and incurs **low client overhead**, as prototype separation runs server-side.

2 Related Work

Heterogeneous Federated Learning. HtFL addresses challenges stemming from heterogeneity in client model architectures. Our work focuses on KD-based HtFL methods. Existing KD-based HtFL approaches can be classified according to the type of information they exchange. Some approaches share model components or parameters. LG-FedAvg [21] enables clients to share upper model layers while maintaining architecture-specific lower layers. FML [35] employs mutual distillation [51] to train and share compact auxiliary models. FedKD [44] reduces FL communication costs via adaptive mutual distillation, sharing only student models and using SVD-based gradient approximation. Other methods prioritize privacy by avoiding direct parameter sharing. FedDistill [12] transmits globally averaged class-wise logits from clients to the server. However, this approach risks revealing the number of classes and logit distributions, creating potential privacy vulnerabilities. To address these concerns, FedProto [41] adopts a more privacy-preserving approach by exchanging only prototypes from the penultimate layer. Recent works have further refined PBFL techniques to improve performance. FedTGP [49] and FedFTP [47] enhance prototype separability through contrastive learning. ProtoNorm also builds upon PBFL but reframes prototype separability as geometric optimization inspired by the Thomson problem.
Hyperspherical Approaches in Federated Learning. In centralized machine learning, several studies have explored hyperspherical approaches, emphasizing that angular information is important for the semantics of deep networks, particularly in the computer vision domain [4, 7, 8, 13, 24–28, 43, 45]. MHE [24], inspired by the Thomson problem, regularizes deep networks by minimizing hyperspherical energy to improve generalization. CoMHE [22] further extends MHE by leveraging projection mappings. HCR [39] applies regularization by projecting both classifier logits and feature embeddings onto their corresponding hyperspheres. Hyperspherical approaches have also been actively applied to FL. FedHP [10] introduces a regularization term based on hyperspherical prototypical networks [31] to uniformly partition the embedding space. FedNH [6] incorporates normalization layers to maintain hyperspherical uniformity and preserve semantic representations of class prototypes. FedUV [36] employs hyperspherical uniformity to regularize representations, effectively simulating IID conditions with non-IID data. ProtoNorm also uses hyperspherical geometry inspired by the Thomson problem. However, unlike existing methods that jointly incorporate hyperspherical regularization during local training, ProtoNorm optimizes global prototypes as a dedicated server-side process.

3 Background

This section formalizes the problem setting, introduces the PBFL framework, and motivates our approach through an example.

3.1 Problem Setting

We consider a heterogeneous federated learning setting where both data distributions and model architectures vary across clients. Formally, the environment consists of a central server connected to M clients, each utilizing a distinct model architecture and maintaining its private data distribution P_i for classification tasks. Following FedProto [41], we partition each client's network $\boldsymbol{w}_i$ into two functional components: a feature extractor f_i with parameters $\boldsymbol{\theta}_i$ that transforms inputs from $\mathbb{R}^D$ into feature space $\mathbb{R}^d$, and a classifier g_i with parameters $\boldsymbol{\phi}_i$ that maps these features to outputs in $\mathbb{R}^K$, where K denotes the number of classes. Therefore, for a sample x, we create a feature vector $\boldsymbol{c} = f_i(\boldsymbol{\theta}_i; x)$. The classifier then produces predictions as $g_i(\boldsymbol{\phi}_i; \boldsymbol{c})$ for the given input. The federated learning objective is to minimize the average expected loss across all clients as follows:

$$\min_{\{\{\boldsymbol{\theta}_i, \boldsymbol{\phi}_i\}\}_{i=1}^{M}} \left\{ \frac{1}{M} \sum_{i=1}^{M} \mathbb{E}_{(x,y)\sim P_i} \left[\ell(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i; x, y) \right] \right\}. \tag{1}$$

3.2 Prototype-Based FL Framework

In PBFL, the server and clients exchange prototypes instead of model parameters via the following three steps (Figure 1).

1) Local Prototype Computation. Local prototypes are computed as the mean of feature layer activations for each class. Client i's local prototype for class j is:

$$\bar{\boldsymbol{c}}_{i,j}^{L} = \frac{1}{n_{i,j}} \sum_{(x,y)\in\mathcal{D}_{i,j}} f_i(\boldsymbol{\theta}_i; x), \tag{2}$$

where $n_{i,j} = |\mathcal{D}_{i,j}|$ represents the number of samples from class j on client i, and $\mathcal{D}_{i,j} \subseteq \mathcal{D}_i$ is the subset of client i's local dataset containing only samples from class j. Clients then transmit their local prototypes to the server.

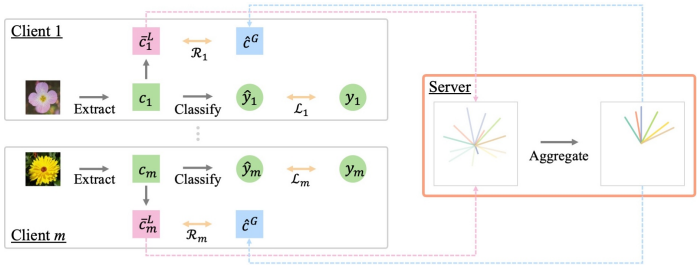


Fig. 1: Overview of the PBFL framework.

2) Global Prototype Construction. The server constructs global prototypes by aggregating received local prototypes, as illustrated in the server region of Figure 1. For each class j, the global prototype $\bar{\boldsymbol{c}}_j^G$ is computed using weighted averaging as follows [41, 49]:

$$\bar{\boldsymbol{c}}_j^G = \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \frac{n_{i,j}}{\sum_{i=1}^{M} n_{i,j}} \bar{\boldsymbol{c}}_{i,j}^L, \tag{3}$$

where $\mathcal{N}_j$ is the set of clients with class j, and $\sum_{i=1}^{M} n_{i,j}$ is the total number of class j samples in the system. The server then transmits the global prototypes back to the clients.

3) Local Training with Prototype Distance Regularization. After receiving the global prototypes, each client trains its model using a combined loss function:

$$\tilde{\mathcal{L}}_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i) = \mathcal{L}_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i) + \lambda \mathcal{R}_i, \tag{4}$$

where $\mathcal{R}_i$ is the regularization term for prototype distance loss, and λ is a hyperparameter controlling regularization strength. The term $\mathcal{R}_i$ is formulated as:

$$\mathcal{R}_i = \sum_j \rho(\bar{\boldsymbol{c}}_{i,j}^L, \bar{\boldsymbol{c}}_j^G), \tag{5}$$

where the function $\rho(\cdot, \cdot)$ computes the Euclidean distance.

Prototype-Based Inference. Unlike typical FL methods, PBFL approaches evaluate performance by calculating the distance between the feature representation and local prototype [41, 49]:

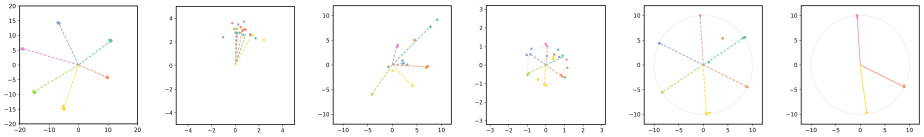
$$\hat{y} = \arg\min_j \| f_i(\boldsymbol{\theta}_i; x) - \bar{\boldsymbol{c}}_{i,j}^L \|, \tag{6}$$

where $\| \cdot \|$ represents the Euclidean distance.

3.3 Motivating Analysis

The PBFL evaluation metric in Eq. (6) reveals that well-separated local prototypes are essential for effective classification. Since the regularization term in Eq. (5) induces local prototype separation via global prototype alignment, the quality of global prototypes is critical to overall performance. To achieve optimal global prototype separation in Euclidean space, both angular distance and magnitude must be carefully controlled. However, vanilla PBFL (FedProto) does not explicitly optimize either aspect; instead, it constructs global prototypes by weighted averaging based on class sample counts (Eq. (3)). Under data and model heterogeneity, clients produce substantially different local prototypes for the same class, leading to poorly separated global prototypes when weighted averaging is used. Global prototypes cluster toward clients with the majority of samples, creating irregular configurations with uneven inter-prototype distances.

A Motivational Example. To demonstrate the necessity for separating global prototypes, we present an illustrative example using a synthetic spiral dataset



(a) FedProto (IID) **(b)** FedProto (non-IID) **(c)** FedTGP (non-IID) **(d)** ProtoNorm ($\gamma = 1$) **(e)** ProtoNorm ($\gamma = 10$) **(f)** ProtoNorm Client ($\gamma = 10$)

Fig. 2: Prototype and feature visualization in 2D feature space on the spiral dataset. Different colors denote different classes.

in two dimensions (Figure 8a in the Appendix) [6]. This dataset comprises six classes with 5000 points each, with its generation process detailed in the Appendix. The classifier employs a 5-layer multilayer perceptron with ReLU activation functions [1], with the feature dimension set to two for visualization. Our experimental setup includes 10 clients under IID and non-IID data distribution scenarios. For the non-IID setting, data is distributed among clients following a Dirichlet distribution with parameter $\alpha = 0.1$, effectively simulating practical heterogeneous FL scenarios (Figure 8b in the Appendix).

We trained FedProto, FedTGP, and ProtoNorm for 200 FL rounds. Under IID conditions, FedProto achieves nearly optimal separation of global prototypes (dashed lines), with local prototypes (points) positioned near their corresponding global prototypes (Figure 2a). However, under non-IID conditions, global prototypes lose this optimal separation as they skew toward clients with majority class samples (Figure 2b), resulting in suboptimal local prototype separation. While FedTGP improves separability through contrastive learning, further optimization remains possible (Figure 2c).

ProtoNorm's iterative alignment distributes prototypes uniformly on the sphere even under non-IID data distributions (Figures 2d and 2e). However, without prototype upscaling ($\gamma = 1$), local prototypes may not faithfully track global prototypes, resulting in suboptimal class separation (Figure 2d). This limitation is addressed by scaling prototype magnitudes by $\gamma = 10$, substantially enhancing inter-class separation in Euclidean space (Figure 2e; note the difference in y-axis scales between Figures 2d and 2e). Figure 2f further illustrates the per-client effect: test sample feature embeddings (points) of client #2 cluster tightly around their respective local prototypes (lines), demonstrating markedly improved classification capability. Collectively, these experiments reveal that effective prototype separation requires the joint optimization of both angular distribution and magnitude. Additional visualizations are provided in the Appendix.

4 Prototype Normalization

ProtoNorm begins by aggregating local prototypes from participating clients to construct initial global prototypes like FedProto. To mitigate privacy risks associated with transmitting class distribution information $n_{i,j}$, we adopt a simple

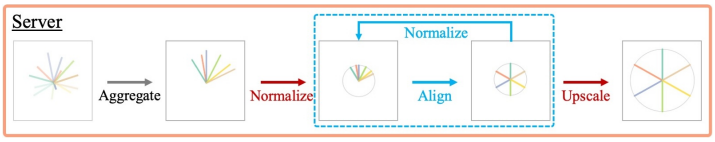


Fig. 3: Server-side optimization of ProtoNorm.

averaging mechanism rather than the weighted aggregation in Eq. (3):

$$\bar{\boldsymbol{c}}_j^G = \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \bar{\boldsymbol{c}}_{i,j}^L. \tag{7}$$

The server then normalizes these aggregated prototypes and performs iterative prototype alignment (illustrated in the blue dashed region of Figure 3) to optimize their configuration on the unit hypersphere.

The alignment stage is formulated as a gradient descent optimization procedure that minimizes the hyperspherical energy of the prototype space, thereby solving the Thomson problem of optimal point distribution on a hypersphere. The iterative optimization process is illustrated by the blue arrows in Figure 3. By analogy to the Thomson problem, minimum hyperspherical energy characterizes the equilibrium state of global prototype directional configurations [24]. We claim that prototypes arranged in a minimal energy configuration achieve better separation in $\mathbb{R}^d$. We omit the superscript ^{G} from the global prototype notation for notational convenience in this subsection. Following [24], we define the hyperspherical energy functional for K prototypes of d-dimension $\bar{\boldsymbol{C}}_K = \{\bar{\boldsymbol{c}}_1, \cdots, \bar{\boldsymbol{c}}_K \in \mathbb{R}^d\}$ as

$$\begin{aligned} \boldsymbol{E}_{s,d}(\{\hat{\boldsymbol{c}}_j\}_{j=1}^K) &= \sum_{j<k} h_s(\|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|) \\ &= \begin{cases} \sum_{j<k} \|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|^{-s}, & s > 0 \\ \sum_{j<k} \log(\|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|^{-1}), & s = 0 \end{cases}, \end{aligned} \tag{8}$$

where $h_s(\cdot)$ is a decreasing real-valued function and $\hat{\boldsymbol{c}}_j = \frac{\bar{\boldsymbol{c}}_j}{\|\bar{\boldsymbol{c}}_j\|}$ denotes the normalization of the j-th global prototype onto the unit hypersphere $\mathbb{S}^d = \{\boldsymbol{u} \in \mathbb{R}^d | \|\boldsymbol{u}\| = 1\}$. For notational efficiency, we define the set of normalized prototypes as $\hat{\boldsymbol{C}}_K = \{\hat{\boldsymbol{c}}_1, \cdots, \hat{\boldsymbol{c}}_K \in \mathbb{S}^d\}$ and the corresponding hyperspherical energy as $\boldsymbol{E}_s = \boldsymbol{E}_{s,d}(\{\hat{\boldsymbol{c}}_j\}_{j=1}^K)$. We employ Riesz s-kernels defined as $h_s(z) = z^{-s}$ with $s > 0$.

The Thomson problem, which requires finding the minimum value of $\boldsymbol{E}_1$, has been proven to be NP-hard, making heuristic solutions necessary. To efficiently tackle this optimization challenge, we employ a computationally tractable surrogate energy function that promotes similar geometric arrangements while being amenable to gradient-based optimization:

$$\arg\min_{\hat{\boldsymbol{C}}_K} \{\boldsymbol{E}_{\log} := \sum_{j<k} \log(h_s(\|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|))\}. \tag{9}$$

Our surrogate energy function for $s = 1$ using simplified notation is:

$$\boldsymbol{E} = \sum_{j<k} \log \frac{1}{\|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|}. \tag{10}$$

During iterations, we first minimize this differentiable objective through gradient descent with respect to the force $\boldsymbol{F}_j$ acting on $\hat{\boldsymbol{c}}_j$, defined as:

$$\boldsymbol{F}_j = -\nabla_{\hat{\boldsymbol{c}}_j} \boldsymbol{E} = \sum_{k=1,k\neq j}^{K} \frac{\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k}{\|\hat{\boldsymbol{c}}_j - \hat{\boldsymbol{c}}_k\|^2}. \tag{11}$$

This relationship follows the physical principle that force equals the negative gradient of potential energy. The negative sign in $-\nabla_{\hat{\boldsymbol{c}}_j} \boldsymbol{E}$ indicates that the force pushes prototypes away from configurations with higher energy toward those with lower energy, aligning with our goal of finding a minimal energy configuration. The server then updates the velocity vector $\boldsymbol{v}_j$, which determines the magnitude and direction of each prototype's movement:

$$\boldsymbol{v}_j^{(t)} = \mu \cdot \boldsymbol{v}_j^{(t-1)} + \eta^{(t)} \cdot \boldsymbol{F}_j^{(t)}, \tag{12}$$

where μ is the momentum coefficient, and η is the learning rate. Each prototype gradually adjusts its position using the updated velocity:

$$\tilde{\boldsymbol{c}}_j^{(t)} = \hat{\boldsymbol{c}}_j^{(t-1)} + \boldsymbol{v}_j^{(t)}. \tag{13}$$

After each update, we normalize the updated prototype:

$$\hat{\boldsymbol{c}}_j^{(t)} = \frac{\tilde{\boldsymbol{c}}_j^{(t)}}{\|\tilde{\boldsymbol{c}}_j^{(t)}\|}. \tag{14}$$

The algorithm terminates after a predefined number of iterations or when it meets stopping criteria (i.e., 10 consecutive changes in $\boldsymbol{F}_j$ fall below the threshold ϵ), ensuring that optimization concludes when forces no longer induce significant changes in prototype positions.

Finally, after convergence of the alignment stage, the global prototypes are upscaled by γ to make further room for better prototype separation in Euclidean space during local training. The federated learning framework of ProtoNorm is detailed in Algorithm 1.

Role of the Prototype Scaling Factor γ. We adopt a fixed γ inspired by the temperature parameter in the softmax function, which sharpens decision boundaries by scaling logit magnitudes [42], as similarly employed in FedNH [6]. However, in the PBFL context, γ in ProtoNorm plays a more nuanced role beyond mere scaling. To see this, consider the total loss

$$\tilde{\mathcal{L}}_i = \mathcal{L}_i + \lambda \sum_j \rho(\bar{\boldsymbol{c}}_{i,j}^L, \gamma \bar{\boldsymbol{c}}_j^G), \tag{15}$$

Algorithm 1 Prototype Normalization (ProtoNorm)

Input: local datasets $\{\mathcal{D}_i\}_{i=1}^{M}$, model parameters $\{\boldsymbol{w}_i\}_{i=1}^{M}$, λ, γ
Output: Trained local models $\boldsymbol{w}_i$ for $i = 1, ..., M$
Server executes:
Initialize normalized prototype set $\{\hat{\boldsymbol{c}}_j^G\}$ for all classes.
for iteration $\tau = 1, \ldots, \mathcal{T}$ **do**
  Sample a client subset $\mathcal{C}^\tau$
  Upscale $\hat{\boldsymbol{c}}_j^G$ by γ ▷ **Prototype Upscaling**
  for client $i \in \mathcal{C}^\tau$ in parallel **do**
    $\bar{\boldsymbol{c}}_{i,j}^L \leftarrow$ **LocalUpdate**$(i, \gamma\hat{\boldsymbol{c}}_j^G)$
  Compute $\bar{\boldsymbol{c}}_j^G$ by aggregating $\bar{\boldsymbol{c}}_{i,j}^L$ using Eq. (7)
  Compute $\hat{\boldsymbol{c}}_j^G$ by $\hat{\boldsymbol{c}}_j^G = \frac{\bar{\boldsymbol{c}}_j^G}{\|\bar{\boldsymbol{c}}_j^G\|}$
  for iteration $t = 1, \ldots, T$ **do** ▷ **Iterative Prototype Alignment**
    Compute $\boldsymbol{F}_j$ using Eq. (11)
    Update $\boldsymbol{v}_j$ using Eq. (12)
    Compute $\tilde{\boldsymbol{c}}_j^G$ by updating $\hat{\boldsymbol{c}}_j^G$ using Eq. (13)
    Compute $\hat{\boldsymbol{c}}_j^G$ by normalizing $\tilde{\boldsymbol{c}}_j^G$ using Eq. (14)
    if stopping criterion met **then**
      break

LocalUpdate$(i, \gamma\hat{\boldsymbol{c}}_j^G)$:
for each local epoch **do**
  for batch $b \in \mathcal{B}_i$ **do**
    Update model using the loss in Eq. (15)
Compute $\bar{\boldsymbol{c}}_{i,j}^L$ using Eq. (2)
return $\bar{\boldsymbol{c}}_{i,j}^L$

where λ and γ serve distinct purposes: λ controls the strength of prototype distillation, while γ scales global prototype magnitudes to directly control the distillation target. Since larger magnitudes of feature embeddings amplify the output logits through the inner product of embeddings with final-layer weights, γ simultaneously governs decision boundary sharpness and couples distillation and classification. The λ-γ interaction is further analyzed in Section 5.

Convergence Properties. ProtoNorm's hyperspherical energy minimization is theoretically grounded in the geometric results of [24], whose Theorem 2 establishes the asymptotics of minimal hyperspherical energy for point configurations of N neuron weights on hyperspheres, independent of deep network training dynamics. ProtoNorm's server-side hyperspherical optimization is fully decoupled from the federated learning procedure and operates solely on the K aggregated prototypes. Therefore, by replacing N neuron weights with K global prototypes, their Theorem 2 applies directly to our setting without additional assumptions, yielding the same geometric guarantees. Furthermore, ProtoNorm preserves both the prototype aggregation and regularization components of FedProto's local training objective, thereby inheriting its non-convex convergence rate unchanged.

Empirical validation across diverse experimental settings, detailed in Section 5, confirms stable convergence behavior throughout.

5 Experiments

This section presents our experimental setup, results, and analysis. The code is available at https://github.com/regulationLee/ProtoNorm.

5.1 Experimental Setup

Datasets and Model Architectures. We evaluated the ProtoNorm framework using seven datasets commonly employed in FL research: CIFAR-10/100 [17], Flowers-102 [32], Tiny ImageNet (Tiny IN) [18], ImageNet-1K [5], AG News [50], and DBpedia [3]. For each dataset, we allocated 75% for training and 25% for testing. To replicate realistic non-IID data distributions among clients, we implemented two distinct heterogeneity scenarios: (1) a pathological setting, where each client contains samples from only specific classes (2 classes for CIFAR-10, 10 classes for CIFAR-100 and Flowers-102, and 20 classes for Tiny ImageNet), and (2) a practical setting, where data is distributed according to a Dirichlet distribution with concentration parameter α [23]. Detailed data distribution for practical settings is included in the Appendix. We employed two model heterogeneous settings: (1) a lightweight setting for resource-constrained clients with ResNet-8 [52], EfficientNet [40], ShuffleNet v2 [29], and MobileNet v2 [34], and (2) a large-scale setting with 4-layer CNN, GoogleNet [37], MobileNet v2, and ResNet-{18, 34, 50, 101, 152}. Each architecture incorporates a global average pooling layer [37], with feature dimension d set to 512.

Baselines and Evaluation Protocol. We evaluated ProtoNorm against six data-free HtFL algorithms: LG-FedAvg [21], FML [35], FedKD [44], FedDistill [12], FedProto [41], and FedTGP [49]. Additionally, we include FedNH [6] and FedUV [36] as homogeneous hyperspherical FL baselines. For PBFL approaches—specifically FedProto, FedTGP, and ProtoNorm—we evaluated model performance based on the similarity between local prototypes and feature embeddings in the average pooling layer, as formulated in Eq. (6). Our evaluation protocol computes the average test accuracy across all clients per round, consistent with prior HtFL work [49].

Federated Learning Setup. Our FL setup comprised 20 clients, each participating in all 300 communication rounds. Client-side training used a learning rate of 0.01, batch size of 32, and one local epoch per round. For the server-side iterative alignment, we set the momentum coefficient $\mu = 0.9$ and learning rate $\eta = 0.1$ with 95% decay every 10 iterations. We adopted the fixed regularization hyperparameter $\lambda = 10$. Using a progressive grid search over the interval [1, 1000], we identified optimal scale parameter γ values: 100 for CIFAR-10, AG News, and DBpedia; 200 for CIFAR-100 and Flowers-102; and 400 for Tiny ImageNet and ImageNet-1K. All results are averaged over three runs with distinct random seeds, with standard deviation reported in parentheses. Implementation details are included in the supplementary materials.

Table 1: Classification test accuracy (%) under heterogeneous model configurations, with standard deviation in parentheses. *Large-scale setting; otherwise, lightweight setting.

Algorithm	Pathological setting				Practical setting ($\alpha = 0.1$)			
	CIFAR-10	CIFAR-100	Flowers-102	Tiny IN	CIFAR-10	CIFAR-100	Flowers-102	Tiny IN*
LG-FedAvg	86.84(0.05)	59.49(0.08)	63.62(0.62)	33.33(0.08)	87.97(0.14)	45.22(0.21)	51.05(0.31)	27.58(0.23)
FML	86.17(0.06)	58.06(0.09)	57.25(0.32)	32.30(0.09)	87.59(0.15)	43.81(0.03)	47.20(0.43)	26.78(0.19)
FedKD	87.00(0.05)	59.53(0.08)	60.75(0.40)	33.79(0.07)	88.10(0.02)	44.45(0.42)	48.27(0.28)	28.87(0.44)
FedDistill	87.03(0.05)	58.96(0.07)	63.47(0.29)	32.22(0.02)	87.93(0.12)	44.70(0.33)	50.80(0.32)	27.38(0.12)
FedProto	84.72(0.05)	57.54(0.09)	61.14(0.95)	29.95(0.03)	86.28(0.46)	41.31(0.18)	41.22(0.85)	22.31(0.34)
FedTGP	86.78(0.05)	59.62(0.08)	65.76(0.74)	31.85(0.04)	87.71(0.49)	45.68(0.16)	54.16(0.72)	28.18(0.13)
ProtoNorm	**89.07(0.16)**	**64.24(0.05)**	**71.35(0.74)**	**40.24(0.26)**	**88.95(0.11)**	**47.94(0.40)**	**57.52(0.38)**	**29.97(0.08)**

Table 2: Classification test accuracy (%) on CIFAR-100 under varying data heterogeneity, feature dimensionality, and the number of clients. Full: full participation; percentages under the number of clients denote sampling rate.

Algorithm	Data heterogeneity			Feature dimension		The number of clients		
	0.01	0.5	1.0	64	1024	50 (Full)	100 (Full)	100 (20%)
LG-FedAvg	70.13(0.36)	25.54(0.18)	19.30(0.27)	43.59(0.17)	45.34(0.08)	43.91(0.19)	40.60(0.52)	39.64(0.04)
FML	68.97(0.19)	24.27(0.20)	18.82(0.19)	41.36(0.19)	44.28(0.45)	42.61(0.12)	39.49(0.11)	38.49(0.11)
FedKD	69.98(0.24)	25.58(0.21)	19.56(0.20)	43.92(0.40)	44.76(0.42)	43.14(0.15)	40.70(0.15)	38.88(0.37)
FedDistill	69.99(0.21)	24.15(0.12)	18.45(0.39)	43.43(0.12)	44.27(1.11)	42.86(0.40)	41.14(0.04)	39.94(0.12)
FedProto	64.89(0.31)	21.30(0.36)	16.07(0.24)	37.49(0.19)	35.48(0.11)	39.89(0.57)	38.03(0.34)	22.98(0.13)
FedTGP	72.67(0.34)	24.14(0.28)	17.95(0.33)	44.41(0.69)	44.56(0.02)	44.08(0.15)	42.28(0.19)	39.20(0.34)
ProtoNorm	**72.95(0.23)**	**29.14(0.07)**	**22.87(0.08)**	**48.04(0.44)**	**48.22(0.15)**	**45.61(0.27)**	**42.74(0.17)**	**41.30(0.34)**

5.2 Performance Results

Overall Performance Across Heterogeneous Models. Tables 1 and 2 demonstrate that ProtoNorm consistently outperformed the baselines across four datasets in heterogeneous model settings. Notably, ProtoNorm exhibited superior performance compared to other PBFL approaches when dealing with many classes, where a clear separation of many global prototypes was required.

Data Heterogeneity. We varied data distributions using the α parameter of the Dirichlet distribution. This heterogeneity decreased as α increased, leading to a more balanced distribution of data across classes for each client. Table 2 shows that ProtoNorm consistently outperformed the baselines regardless of the data heterogeneity level.

Feature Dimensionality. The performance of deep networks is generally affected by feature dimensionality. Since PBFL methods share prototypes derived from the feature layer, their performance could be sensitive to this parameter. We evaluated this sensitivity across methods, with results presented in Table 2. ProtoNorm consistently outperformed competing approaches across all tested dimensions.

Client Scalability. We evaluated scalability across three configurations: 50 and 100 clients with full participation, and 100 clients with 20% random sampling per round to simulate realistic deployment scenarios. ProtoNorm achieved superior performance across all settings, as reported in Table 2.

Validation on Many-Class and Text Datasets. We conducted additional experiments to reflect real-world settings. For the many-class evaluation, we used ImageNet-1K downsampled to 32×32, with 10 clients and a batch size

Table 3: Test accuracy (%) on many-class image and text classification datasets.

Dataset	FedProto	FedTGP	**ProtoNorm**
ImageNet-1K	14.55(0.34)	13.62(0.29)	**21.52(0.23)**
DBpedia	84.01(0.16)	93.62(0.12)	**97.15(0.13)**
AG News	95.16(0.10)	97.26(0.07)	**97.46(0.02)**

Table 4: Test accuracy (%) using ResNet-18 for homogeneous model settings.

Dataset	FedNH	FedUV	**ProtoNorm**
CIFAR-100	37.72(1.23)	48.33(0.19)	**52.67(0.33)**
Flowers-102	47.61(1.09)	55.36(0.32)	**59.71(0.38)**
Tiny IN	22.62(0.70)	30.18(0.06)	**34.58(0.21)**

Table 5: Ablation study.

PA	PU	CIFAR-100	Flowers	Tiny IN
✗	✗	41.31	41.22	17.88
✓	✗	39.59	37.32	18.26
✗	✓	40.26	41.36	17.18
✓	✓	**47.94**	**57.52**	**32.32**

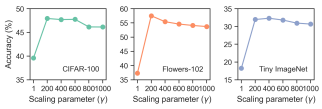


Fig. 4: Effect of γ.

Fig. 5: γ vs. λ.

of 128, keeping all other settings at their default values. As shown in Table 3, ProtoNorm outperformed other PBFL methods. The relatively low performance of FedTGP may result from the small number of clients, which generate fewer local prototypes for contrastive learning. For text classification, we evaluated on the DBpedia and AG News datasets using FastText [14] and TextCNN [16] architectures, with full results for other FL approaches provided in the Appendix. ProtoNorm achieved the best performance in both settings, demonstrating its applicability across diverse data modalities.

Comparison with Other Hyperspherical Uniformity Methods. Existing FL approaches that leverage hyperspherical uniformity, such as FedNH and FedUV, are typically designed for homogeneous architectures. Although ProtoNorm targets heterogeneous FL, we conducted comparative evaluations under homogeneous settings using ResNet-18. As shown in Table 4, ProtoNorm attains higher accuracy than both FedNH and FedUV.

5.3 Analysis

Ablation Study. We conducted ablation studies to validate the individual contributions of prototype alignment (PA) and upscaling (PU). As shown in Tables 1 and 2, ProtoNorm consistently outperformed the baseline FedProto method, which lacks both components. Table 5 reveals that both components are essential: removing either results in substantial performance degradation, confirming their synergistic effect. Notably, alignment alone (✓, ✗) actually degrades performance on CIFAR-100 and Flowers-102 compared to no alignment. This counterintuitive result stems from the local optimization landscape: while alignment achieves good angular separation, unit-norm global prototypes create suboptimal local minima during client training. Clients jointly optimize cross-entropy and prototype distance losses, but when global prototype magnitudes are constrained to unit norm, this optimization can converge poorly.

γ and λ Sensitivity. Figure 4 demonstrates how the scaling factor γ affects classification accuracy. The results revealed a typical saturation pattern in the hyperparameter-performance relationship, indicating optimal performance within a specific parameter range. To further characterize the interplay between hyperparameters, we analyze the γ-λ interaction on CIFAR-100 via the heatmap in

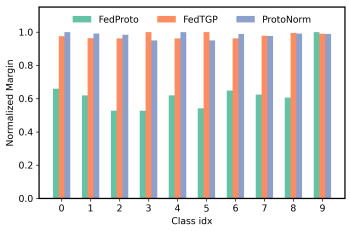
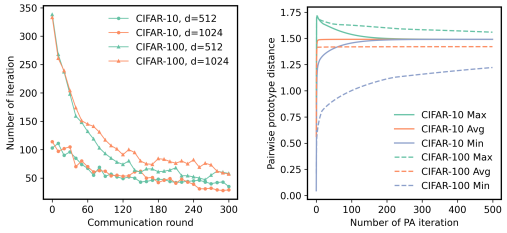


Fig. 6: Comparison of normalized prototype margins.



(a) Number of iterations. **(b)** Pairwise distance.

Fig. 7: Convergence analysis.

Figure 5. Small values of γ consistently yield degraded performance irrespective of λ, corroborating the ablation findings in Table 5, whereas sufficiently large γ sustains strong performance even under small λ. This asymmetry underscores the primacy of γ in governing prototype discriminability.

Prototype Separability. Following FedTGP, we define the global prototype margin as the minimal Euclidean distance between the prototype of a specific class and the prototypes of all other classes. The maximum margin represents the largest global prototype margin across all classes. We normalized all margin values to the maximum margin for each method in Figure 6 to remove magnitude effects for CIFAR-10. In FedTGP and ProtoNorm, the normalized global prototype margins were similar, indicating evenly distributed inter-prototype distances. In contrast, FedProto exhibited irregular prototype distributions, suggesting suboptimal separation in the embedding space.

Empirical Convergence Analysis. As federated learning progresses, ProtoNorm achieves better-separated prototypes that require fewer alignment iterations. Figure 7a confirms this across 300 communication rounds, showing a consistent downward trend. CIFAR-100 requires more iterations than CIFAR-10 due to its 10× larger number of classes. This monotonic decrease in alignment iterations reflects improving prototype quality, corroborated by smooth test accuracy convergence across all datasets (Figure 12 in the Appendix).

To understand the dynamics within each alignment stage, we examine how prototype distances evolve during iteration. During alignment, pairwise distances between class prototypes should gradually become more evenly distributed to achieve optimal separation. Figure 7b analyzes this convergence process during a single alignment stage at the initial FL round. CIFAR-100 exhibits slower convergence than CIFAR-10 due to its larger number of classes, which increases the complexity of achieving uniform prototype separation. Despite this scalability challenge, our experiments demonstrate substantial performance improvements for CIFAR-100 and maintain effectiveness for datasets with over 100 classes.

Server-Side Computational Efficiency. FedTGP and ProtoNorm incur additional server-side computational overhead compared to FedProto due to their prototype optimization procedures. The computational complexity grows as $O(M \times K^2)$ for FedTGP and $O(K^2)$ for ProtoNorm (M: client count, K: class count). FedTGP's complexity arises from applying contrastive learning across

Table 6: Per-round runtime (seconds) on CIFAR-10/100 and Tiny ImageNet. ProtoNorm achieves 1.1%, 3.8%, and 3.9% speedup over FedTGP, respectively.

Algorithm	CIFAR-10	CIFAR-100	Tiny IN
FedProto	25.85	50.57	176.68
FedTGP	26.48	52.97	188.30
ProtoNorm	26.19	50.95	180.98

Table 7: Per-round communication overhead (millions of parameters).

Algorithm	CIFAR-100	Flowers-102	Tiny IN
LG-FedAvg	2.05	2.09	4.10
FML	36.99	37.03	39.04
FedKD	33.04	33.07	34.87
FedDistill	0.29	0.28	1.17
FedNH	89.26	89.16	90.76
FedUV	89.81	89.86	91.87
ProtoNorm (PBFL)	1.46	1.37	2.93

all client-submitted local prototypes, while ProtoNorm's stems from iterative alignment of global prototypes. Critically, ProtoNorm's prototype alignment complexity is independent of client count, making it more scalable than FedTGP. This difference becomes significant at scale: for 1,000 classes, FedTGP requires approximately 40,000 billion FLOPs per round compared to ProtoNorm's 375 billion—over 100× more computation. As shown in Table 6, ProtoNorm's runtime overhead remains comparable to FedProto while consistently outperforming FedTGP across datasets with different class counts.

Communication Efficiency. Like other PBFL approaches, ProtoNorm achieves significant communication efficiency gains by transmitting only prototypes. We quantified communication costs using the theoretical formulation presented in the Appendix. Table 7 presents the number of parameters (in millions) transmitted during a single FL round across different methods. Notably, while FedDistill has lower communication costs than ProtoNorm, it is important to consider that transmitting logits provides weaker privacy guarantees than transmitting prototypes, as logits inherently reveal information about class distributions.

6 Discussion and Conclusion

A limitation of ProtoNorm is that the selection of γ remains heuristic; deriving a principled rule based on class count or feature dimension is left to future work. ProtoNorm reframes prototype separability in heterogeneous FL as a hyperspherical energy minimization problem, optimizing global prototype configurations to maximize class distinguishability. Crucially, effective separation requires jointly optimizing angular distribution and magnitude—angular alignment alone is insufficient due to local optimization dynamics during client training.

Extensive experiments show that ProtoNorm consistently outperforms baselines across diverse settings, demonstrating that geometry-driven prototype arrangement, without any semantic consideration, can suffice to surpass methods relying on semantic alignment. It offers two key advantages: its gradient-based solver converges stably with complexity independent of client count, giving superior scalability over contrastive approaches; and its server-side optimization operates solely on prototypes, enabling deployment across heterogeneous client models with communication-efficient exchange. Moreover, its uniform averaging avoids transmitting per-class sample counts, thereby further reducing potential information leakage compared with weighted averaging.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25487032); by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00277078, Optimization of artificial neural network driving for multi-multiple sensor convergence for autonomous driving and development of integrated cognitive software technology); and by the Starting growth Technological R&D Program (TIPS Program, (No. RS-2023-00271736)) funded by the Ministry of SMEs and Startups (MSS, Korea).

References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018) 6
2. Alam, S., Liu, L., Yan, M., Zhang, M.: Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. Advances in neural information processing systems **35**, 29677–29690 (2022) 2
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: international semantic web conference. pp. 722–735. Springer (2007) 10
4. Chen, B., Liu, W., Yu, Z., Kautz, J., Shrivastava, A., Garg, A., Anandkumar, A.: Angular visual hardness. In: International Conference on Machine Learning. pp. 1637–1648. PMLR (2020) 3
5. Chrabaszcz, P., Loshchilov, I., Hutter, F.: A downsampled variant of imagenet as an alternative to the cifar datasets. arXiv preprint arXiv:1707.08819 (2017) 10
6. Dai, Y., Chen, Z., Li, J., Heinecke, S., Sun, L., Xu, R.: Tackling data heterogeneity in federated learning with class prototypes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 7314–7322 (2023) 3, 6, 8, 10
7. Davidson, T.R., Falorsi, L., De Cao, N., Kipf, T., Tomczak, J.M.: Hyperspherical variational auto-encoders. arXiv preprint arXiv:1804.00891 (2018) 3
8. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4690–4699 (2019) 3
9. Diao, E., Ding, J., Tarokh, V.: Heterofl: Computation and communication efficient federated learning for heterogeneous clients. arXiv preprint arXiv:2010.01264 (2020) 2
10. Fonio, S., Polato, M., Esposito, R., et al.: Fedhp: Federated learning with hyperspherical prototypical regularization. In: ESANN 2024 proceedings, pp. 69–74. i6doc (2024) 3
11. Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., Lane, N.: Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems **34**, 12876–12889 (2021) 2
12. Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., Kim, S.L.: Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv preprint arXiv:1811.11479 (2018) 2, 3, 10

13. Jing, M., Li, J., Zhu, L., Ding, Z., Lu, K., Yang, Y.: Balanced open set domain adaptation via centroid alignment. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 8013–8020 (2021) 3
14. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics: volume 2, short papers. pp. 427–431 (2017) 12
15. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and Trends® in Machine Learning **14**(1–2), 1–210 (2021) 1
16. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1746–1751 (2014) 12
17. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 10
18. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015) 10
19. Li, D., Wang, J.: Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019) 2
20. Li, Q., Diao, Y., Chen, Q., He, B.: Federated learning on non-iid data silos: An experimental study. In: 2022 IEEE 38th international conference on data engineering (ICDE). pp. 965–978. IEEE (2022) 1
21. Liang, P.P., Liu, T., Ziyin, L., Allen, N.B., Auerbach, R.P., Brent, D., Salakhutdinov, R., Morency, L.P.: Think locally, act globally: Federated learning with local and global representations. arXiv preprint arXiv:2001.01523 (2020) 3, 10
22. Lin, R., Liu, W., Liu, Z., Feng, C., Yu, Z., Rehg, J.M., Xiong, L., Song, L.: Regularizing neural networks via minimizing hyperspherical energy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6917–6927 (2020) 3
23. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble distillation for robust model fusion in federated learning. Advances in neural information processing systems **33**, 2351–2363 (2020) 10
24. Liu, W., Lin, R., Liu, Z., Liu, L., Yu, Z., Dai, B., Song, L.: Learning towards minimum hyperspherical energy. Advances in neural information processing systems **31** (2018) 3, 7, 9
25. Liu, W., Lin, R., Liu, Z., Rehg, J.M., Paull, L., Xiong, L., Song, L., Weller, A.: Orthogonal over-parameterized training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7251–7260 (2021) 3
26. Liu, W., Lin, R., Liu, Z., Xiong, L., Schölkopf, B., Weller, A.: Learning with hyperspherical uniformity. In: International conference on artificial intelligence and statistics. pp. 1180–1188. PMLR (2021) 3
27. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. arXiv preprint arXiv:1612.02295 (2016) 3
28. Liu, W., Zhang, Y.M., Li, X., Yu, Z., Dai, B., Zhao, T., Song, L.: Deep hyperspherical learning. Advances in neural information processing systems **30** (2017) 3
29. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018) 10

30. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017) 1
31. Mettes, P., Van der Pol, E., Snoek, C.: Hyperspherical prototype networks. Advances in neural information processing systems **32** (2019) 3
32. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian conference on computer vision, graphics & image processing. pp. 722–729. IEEE (2008) 10
33. Pati, S., Kumar, S., Varma, A., Edwards, B., Lu, C., Qu, L., Wang, J.J., Lakshminarayanan, A., Wang, S.h., Sheller, M.J., et al.: Privacy preservation for federated learning in health care. Patterns **5**(7) (2024) 1
34. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018) 10
35. Shen, T., Zhang, J., Jia, X., Zhang, F., Huang, G., Zhou, P., Kuang, K., Wu, F., Wu, C.: Federated mutual learning. arXiv preprint arXiv:2006.16765 (2020) 2, 3, 10
36. Son, H.M., Kim, M.H., Chung, T.M., Huang, C., Liu, X.: Feduv: Uniformity and variance for heterogeneous federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5863–5872 (2024) 3, 10
37. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015) 10
38. T Dinh, C., Tran, N., Nguyen, J.: Personalized federated learning with moreau envelopes. Advances in Neural Information Processing Systems **33**, 21394–21405 (2020) 1
39. Tan, C., Gao, Z., Wu, L., Li, S., Li, S.Z.: Hyperspherical consistency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7244–7255 (2022) 3
40. Tan, M.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019) 10
41. Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., Zhang, C.: Fedproto: Federated prototype learning across heterogeneous clients. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8432–8440 (2022) 2, 3, 4, 5, 10
42. Wang, F., Cheng, J., Liu, W., Liu, H.: Additive margin softmax for face verification. IEEE Signal Processing Letters **25**(7), 926–930 (2018) 8
43. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5265–5274 (2018) 3
44. Wu, C., Wu, F., Lyu, L., Huang, Y., Xie, X.: Communication-efficient federated learning via knowledge distillation. Nature communications **13**(1), 2032 (2022) 3, 10
45. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1426–1435 (2019) 3
46. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019) 1

47. Yin, K., Ding, Z., Ji, X., Wang, Z.: Controlling update distance and enhancing fair trainable prototypes in federated learning under data and model heterogeneity. Defence Technology (2025) 2, 3
48. Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. Knowledge-Based Systems **216**, 106775 (2021) 1
49. Zhang, J., Liu, Y., Hua, Y., Cao, J.: Fedtgp: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 16768–16776 (2024) 2, 3, 5, 10
50. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Advances in neural information processing systems **28** (2015) 10
51. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: CVPR (2018) 3
52. Zhong, Z., Li, J., Ma, L., Jiang, H., Zhao, H.: Deep residual networks for hyperspectral image classification. In: 2017 IEEE international geoscience and remote sensing symposium (IGARSS). pp. 1824–1827. IEEE (2017) 10
53. Zhu, Z., Hong, J., Drew, S., Zhou, J.: Resilient and communication efficient learning for heterogeneous federated systems. Proceedings of machine learning research **162**, 27504 (2022) 2
54. Zhu, Z., Hong, J., Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International conference on machine learning. pp. 12878–12889. PMLR (2021) 2