Federated Generalized Category Discovery

Supplementary Material

7. Dataset Splitting in Fed-GCD benchmark

To facilitate the study of Fed-GCD task, we re-organize three commonly-used generic image classification datasets (*i.e.*, CIFAR-10, CIFAR-100 and ImageNet-100) and three more challenging fine-grained image classification datasets (*i.e.*, CUB-200, Stanford Cars, and Oxford-IIIT Pet) to construct a new Fed-GCD benchmark.

For each dataset, first, we sample a subset of half the classes as "Old" categories in the original training set, and 50% of instances of each labeled class are drawn to form the labeled set, and all the remaining data form the unlabeled set. With the same rate of labeled-unlabeled splitting, we split the original testing set into labeled and unlabeled subsets for class number estimation and GCD testing on server. Then, we further leverage the β-Dirichlet distribution to split the training set into N^L subsets, where the N^L subsets are regarded as local datasets individually stored in each client. The detailed statistics of all splits are elaborated in Tab. 6.

8. Evaluation Protocols with Different Data Heterogeneity

Due to the varying data distribution in different Fed-GCD applications, we present two evaluation protocols to separately simulate the normally heterogeneous (**NH**) and extremely heterogeneous (**EH**) scenarios by adjusting β in Dirichlet distribution. Specifically, we set $\beta = 0.2$ and $\beta = 0.05$ for **NH** and **EH**, respectively. The statistics of the dataset splits under the two evaluation protocols are described in Tab. 6, in which the **NH** setting exists few common classes but there is no labeled categories shared across all clients in the **EH** setting. For each dataset, we learn a global model in a decentralized training fashion. During testing, we first estimate the number of the potential categories (*i.e.*, k) in the non-overlapping test set by using the labeled data stored on server. Then we calculate the maximum of clustering accuracy between the ground truth labels and the label assignment with the estimated k over the set of permutations via Hungarian algorithm. Last, we measure the clustering accuracy for "All", "Old" and "New" categories, respectively. All the statistics of the dataset splits and evaluation protocols are described in Tab. 6.

9. Visualization of Learned GMMs.

To explore how the learned Global-Local GMMs work, we visualize the T-SNE embeddings of the means of each component of GMMs. We use circles with different color to represent 7, 9,11,15 and 19 local GMM centers from client 1,2,3,4 and 5, respectively. 9 global GMM centers are denoted by concentric circles. The number of clusters in each client is automatically estimated by semi-FINCH and the number of global clusters is estimated by fully-unsupervised FINCH. Through analyzing the visualization results, we find that **1)** most global cluster centers are located at the ground-truth centers without accessing to raw data, which demonstrates the effectiveness of category knowledge aggregation; **2)** as for the blue-green cluster in the purple dashed line, global cluster may server as a superclass to provide distinct semantics for improving representation discriminability.

10. Experiments with Different Number of Clients

In real-world federated generalized category discovery (Fed-GCD), the number of local stations (clients) often varies for different discovery tasks. For example, a biodiversity research center would like to explore comprehensive global species distribution, which is inevitable to build many clients all over the world. As indicated in existing federated learning studies, increasing the number of clients may lead the global model to divergence, which gives rise to a more severe challenge for training Fed-GCD systems. To attempt to investigate this challenge, we first evaluate "FedAvg + GCD", "FedAvg + GCL", "FedAvg + AGCL", "Centralized-GCD" and "Centralized-GCL" on the scenario of $N^L = 10$. As summarized in Tab. 7, the large number of clients results in a significant performance degradation compared with centralized training, especially on CIFAR10 dataset. Moreover, our client semantics association impacts by the increasing of the number of clients, because large number of clients tend to generate local extremely-heterogeneous GMMs. This further leads uniformly-sampled features to be biased, which hinders effectively clustering and category aggregation due to lack of prior knowledge for rectifying the biased distribution.

In short, the experiments with different values of $N^L = 5$ are studied in the main text. The experiments with $N^L = 10$ are studied in Tab. 7. Comparing the centralized training and FedAvg baseline, GCD suffers from a larger performance drop than our GCL. Specifically, compared to "Centralized-GCD", "FedAvg + GCD" degrades performance by 10.2%, 7.6%, and 9.8% on "All" categories. Our GCL achieves a relatively robust performance by 8.5%, 6%, and 8.8%.

Table 6. The statistics of our Fed-GCD benchmark. We simulate different degrees of data heterogeneity in real-world Fed-GCD scenarios by adjusting the β of parametric Dirichlet distribution to split the local training sets among clients.

Dataset	β	Client N^L=5						Server			
		# Labelled Classes		# Unlabelled Classes		# Classes Shared Across		Labelled		Unlabelled	
		Max	Min	Max	Min	all clients	≥2 clients	# Classes	# Images	# Classes	# Images
CIFAR10	0.2	5	4	10	9	2	5	5	2500	10	7500
	0.05	5	1	10	4	0	4	5	2500	10	7500
CIFAR100	0.2	50	33	100	73	16	49	50	2500	100	7500
	0.05	44	17	90	40	1	43	50	2500	100	7500
ImageNet-100	0.2	50	37	100	70	16	50	50	1250	100	3750
	0.05	44	17	90	40	1	47	50	1250	100	3750
CUB-200	0.2	100	36	200	98	5	97	100	1430	200	4362
	0.05	89	25	178	59	0	84	100	1430	200	4362
SCars	0.2	98	47	196	95	5	96	98	2001	196	6040
	0.05	87	29	177	57	0	87	98	2001	196	6040
Pet	0.2	19	12	37	22	3	19	19	940	37	2729
	0.05	18	5	43	16	0	17	19	940	37	2729

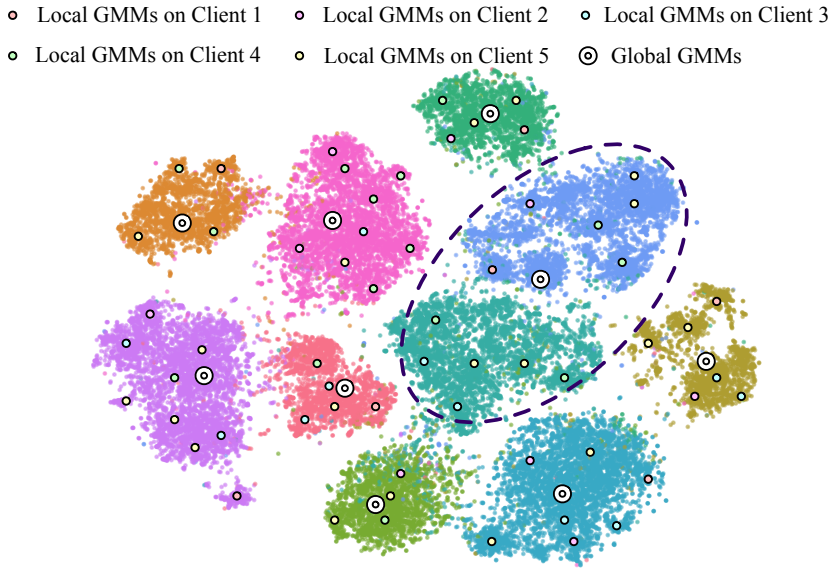

Figure 5. The visualization results on CIFAR10 training dataset. To explore how the learned Global-Local GMMs work, we visualize the T-SNE embeddings of the means of each component of GMMs. We use circles with different color to represent 7, 9,11,15 and 19 local GMM centers from client 1,2,3,4 and 5, respectively. 9 global GMM centers are denoted by concentric circles. The number of clusters in each client is automatically estimated by semi-FINCH and the number of global clusters is estimated by fully-unsupervised FINCH. Through analyzing the visualization results, we find that **1) most global cluster centers are located at the ground-truth centers without accessing to raw data, which demonstrates the effectiveness of category knowledge aggregation; 2) as for the blue-green cluster in the purple dashed line, global cluster may server as a super-class to provide distinct semantics for improving representation discriminability.**

11. Update of Global and Local GMMs

Details: 1) GMM Update. During client training, local GMMs are updated by the gradient descent based on our GCL losses. The global GMM is updated every communication round. 2) GMM components. We perform semi-FINCH and leverage the accuracy of clustering labeled data as a metric to estimate the number of categories in local data. Based on the clustering results, local GMMs are initialized with the estimated means, variants and component numbers (i,e, the number of categories). Similarly, we uniformly sample N S features from each component of each local GMM, to generate a new feature set. Then, we execute unsupervised FINCH to cluster these features for creating

Setup	**NH** setting ($\beta = 0.05$)								
	CIFAR10			CIFAR100			ImageNet-100		
	All	Old	New	All	Old	New	All	Old	New
Centralized-GCD	83.6	85.8	82.0	54.9	56.1	53.7	72.1	80.7	67.5
Centralized-GCL	86.7	86.7	86.7	58.5	57.2	58.1	76.1	83.7	68.4
FedAvg + GCD	63.4	60.0	66.7	47.3	48.3	45.6	62.3	70.8	60.1
FedAvg + GCL	**68.2**	**64.2**	70.1	**52.5**	**53.9**	51.0	67.3	74.5	60.8
FedAvg + AGCL	68.1	63.8	**70.3**	52.2	53.6	**52.4**	**67.5**	**74.8**	**61.1**

Table 7. Results on generic datasets in the scenario of $N^L = 10$. Comparing the centralized training and FedAvg baseline, GCD suffers from larger performance drop than our GCL. Specifically, comparing to "Centralized-GCD", "FedAvg + GCD" degrades performance by 10.2%, 7.6% and 9.8% on "All" categories. Our GCL achieves a relatively robust performance.

the global GMM. For example, in CUB-200, at the beginning, the estimated class numbers in 5 clients are 63, 84, 117, 183 and 125, respectively, and, we estimate 96 components for the global GMM.

12. Algorithm of Federated AGCL framework.

To clarify the pipeline of our AGCL framework. We elaborate the algorithm tabel in Algorithm 1.

Algorithm 1: Algorithm Pipeline of Federated AGCL.

Input: Local Models Θ^L, Local Data $\mathcal{D}^L$ and Server Data $\mathcal{D}^G$.
Output: Global Model Θ^G.
for $r = 1$ ***in*** $[1, max\_round]$ **do**
  for $n = 1$ ***in*** $[1, N^L]$ **do**
    Extract feature and perform semi-FINCH (Algorithm 3) to initialize $\mathcal{G}_n^L$;
  end
  Clients upload $\mathcal{G}^L$ to server;
  Server executes Client Semantics Association with unsupervised FINCH (Algorithm 2) to generate $\mathcal{G}^G$;
  Server distributes $\mathcal{G}^G$ to each client;
  for $n = 1$ ***in*** $[1, N^L]$ **do**
    for $i = 1$ ***in*** $[1, max\_iteration]$ **do**
      Sample mini-batches from $\mathcal{D}_n^L$;
      Calculate overall optimization objective by $\mathcal{L}_n$;
      Update Θ_n^L and $\mathcal{G}_n^L$ by SGD;
    end
  end
end
Extract feature and Leverage labeled data in $\mathcal{D}^G$ to estimate the number of classes in unlabeled data;
Calculate clustering accuracy for "ALL", "Old" and "New" categories;

13. Algorithm of Unsupervised FINCH.

During client semantic association, we utilize unsupervised FINCH to generate Global GMM. Here, we elaborate the algorithm table in Algorithm 2. The adjacent matrix of NN graph is established via:

$$A(i,j) = \begin{cases} 1 & \text{if } j = \kappa_i^1 \text{ or } \kappa_j^1 = i \text{ or } \kappa_i^1 = \kappa_j^1, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where $\kappa^1 \in \mathbb{R}^{N \times 1}$ is first neighbors integer vector.

Then, based on above equation. We follow the pipeline in Algorithm 2 to perform unsupervised parameter-free FINCH clustering. We use the second-level partition on server aggregation in all our experiments.

14. Algorithm of Semi-FINCH.

To clarify the pipeline of our AGCL framework. We elaborate the algorithm tabel in Algorithm 3. Similarly, the NN graph is given by:

$$A(i,j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are unlabeled}, j = \kappa_i^1 \text{ or }, \\ & \kappa_j^1 = i \text{ or } \kappa_i^1 = \kappa_j^1, \\ 1 & \text{if } i \text{ and } j \text{ are labeled}, y_i = y_j, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where y_i is the ground-truth category label of i instance.

In short, we first extract features and calculate pair-wise similarities, and search the 1^{st}-neighbor of each sample based on the similarities. Next, we force one random labeled sample that belongs to the same category as its 1^{st}-neighbor and then apply FINCH algorithm to yield multi-level clustering results. Later, we leverage the clustering accuracy of labeled samples as the index to select the clustering level. Finally, we choose the level that achieve highest clustering accuracy as the estimated results to calculate the cluster-specific mean and covariance to initialize the learnable GMM.

Algorithm 2: Algorithm Pipeline of unsupervised FINCH.

Input: Sample set $\mathcal{S} = \{1, 2, \cdots, N\}, \mathcal{S} \in \mathbb{R}^{N \times d}$, where N is total number of samples and each sample point is represented by d-dimension feature vector.

Output: Set of Partitions $\mathcal{P} = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_P\}$, where each partition $\Gamma_i = \{C_1, C_2, \cdots, C_{\Gamma_i} | C_{\Gamma_i} \geq C_{\Gamma_{i+1}}, \forall i \in \mathcal{P}\}$ is a valid clustering of $\mathcal{S}$.

The unsupervised FINCH Algorithm:

1: Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{N \times 1}$ via exact distance or fast approximate nearest neighbor methods.

2: Given κ^1 get first partition Γ_1 with C_{Γ_1} clusters via Equation 1.C_{Γ_1} is the total number of clusters in partition Γ_1.

while *there are at least two clusters in* Γ_i **do**

3: Given input data S and its partition Γ_i, compute cluster means (average of all data vectors in that cluster). Prepare new data matrix $M = \{1, 2, \cdots, C_{\Gamma_i}\}$, where $\mathbb{M}_{\Gamma_i} \times d$.

4: Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{C_{\Gamma_i} \times 1}$ of points in M.

5: Given κ^1 get partition Γ_M of Γ_i via Equation 1, where $\Gamma_M \supset \Gamma_i$.

if Γ_M *has one cluster* **then**

break

else

Update cluster labels in $\Gamma_i : \Gamma_M \rightarrow \Gamma_i$

end

end

The detailed semi-algorithm is summarized in Algorithm 3.

Algorithm 3: Algorithm Pipeline of semi-FINCH.

Input: Sample set $\mathcal{S} = \{1, 2, \cdots, N\}, \mathcal{S} \in \mathbb{R}^{N \times d}$, where N is total number of samples and each sample point is represneted by d-dimension feature vector.

Output: Set of Partitions $\mathcal{P} = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_P\}$, where each partition $\Gamma_i = \{C_1, C_2, \cdots, C_{\Gamma_i} | C_{\Gamma_i} \geq C_{\Gamma_{i+1}}, \forall i \in \mathcal{P}\}$ is a valid clustering of $\mathcal{S}$.

The semi-FINCH Algorithm:

1: Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{N \times 1}$ via exact distance or fast approximate nearest neighbor methods.

2: Given κ^1 get first partition Γ_1 with C_{Γ_1} clusters via Eq. (1). C_{Γ_1} is the total number of clusters in partition Γ_1.

while *there are at least two clusters in* Γ_i **do**

3: Given input data S and its partition Γ_i, compute cluster means (average of all data vectors in that cluster). Prepare new data matrix $M = \{1, 2, \cdots, C_{\Gamma_i}\}$, where $\mathbb{M}_{\Gamma_i} \times d$.

4: Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{C_{\Gamma_i} \times 1}$ of points in M.

5: Given κ^1 get partition Γ_M of Γ_i via Equation 1, where $\Gamma_M \supset \Gamma_i$.

if Γ_M *has one cluster* **then**

break

else

Update cluster labels in $\Gamma_i : \Gamma_M \rightarrow \Gamma_i$

end

end