7 Appendix

A Clients distribution

We created a federated version of CIFAR10 and CIFAR100 datasets by introducing two partitioning strategies to split the samples across 100 clients:

- Quantity-based label imbalance: Each client holds data samples of K labels. We first randomly assign K different labels to each client. Then, per label, we randomly assign samples to clients along with labels (with replacement). This way, the number of different labels for each client is fixed. For CIFAR100 dataset, we use $K = 10$. For CIFAR10 dataset, we use $K = 4$.

  Anchor clients: We followed the same method as above to create the anchor clients, except that we prevented replacement when randomly selecting the labels. This way, we created a) 5 anchor clients with $K = 2$ on CIFAR10 and b) 10 anchor clients with $K = 10$ on CIFAR100 dataset.

- Distribution-based on label imbalance: We simulated the label imbalance of each client by allocating portion of the samples (with replacement) of each label according to the Dirichlet distribution ($\alpha = 0.1$). As illustrated in Figure 6, the test clients are completely random unseen combinations of K labels that never appear during training.

  Anchor clients: We use the same Dirichlet distribution ($\alpha = 0.1$) to randomly create a) 5 anchor clients on CIFAR10 and b) 10 anchor clients on CIFAR100 dataset.

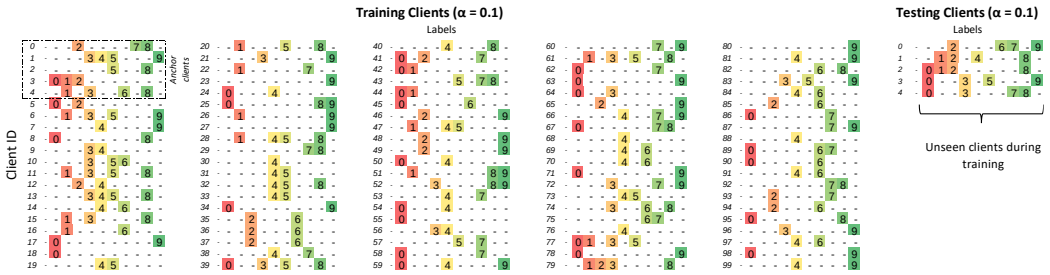

Figure 6: Example of distribution-based label imbalance partition on CIFAR10 dataset ($\alpha = 0.1$)

Note that for test users we do not repeat any distribution from the training clients, this way we create an example where the distribution of the images over all users are different.

B FedJETs end-to-end performance

B.1 Quantity based strategy

We begin to evaluate the performance of our method and baselines, by measuring the zero-shot personalized model accuracy on several unseen test clients with Quantity-based label imbalance distribution strategy, as explained in Appendix A. The results are illustrated in Figure 7

In Figure 7 we can observe that `FedAvg` is not able to keep improving once it's initialized from the pretrained checkpoint. This surprising result stems from three major issues: the learning rate parameters for the clients are not consistent with previous training, the heterogeneous data distribution on the training clients introduces a high degree of model variability, and the pretrained expert struggles to improve or adapt to the federated distribution. Moreover, implementing `FedProx` required careful fine-tuning of the μ parameter to achieve good accuracy and fast convergence. On the other hand, despite trying multiple hyperparameter

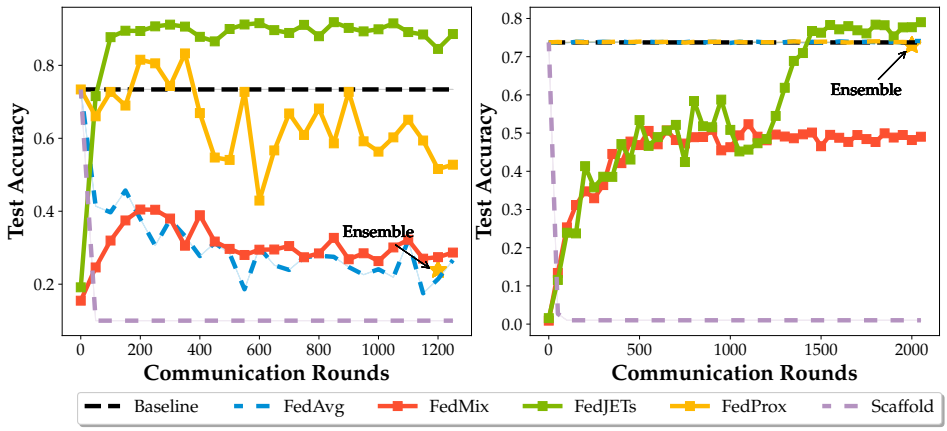

Figure 7: `FedJETs` on CIFAR10 (left) and CIFAR100 (right) datasets, against `FedMix`, `FedAvg` and `Average Ensembles` based on Table 1, using an initial common expert of 73% accuracy.

settings, we could not produce a useful model using `Scaffold` method; it became unstable during training and often collapsed or got stuck in a poor model. This indicates that our method is more robust than these baselines in the current setup

B.2 Distribution based strategy

Using the distribution based strategy -detailed in Appendix A- we implement two additional challenging scenarios, where additional heterogeneity and complexity is inserted via labels distribution: *i*) we use the Dirichlet probability rule to generate skewed and imbalanced label distributions, mimicking real-world applications. *ii*) we relax the assumption of disjoint labels for the *anchor* clients and allow label overlap, creating a more complex scenario, given that experts are initialized from scratch.

	CIFAR 10	CIFAR 100
Common Expert	73.39%	73.73%
`FedAvg`	51.3%	73.6%
`FedProx`	52.8%	73.6%
`Scaffold`	10.0%	01.0%
`FedMix`	29.8%	65.3%
`FedJETs`	**80.8%**	**77.8%**%

Table 3: Best Global Test accuracies from the last 10 evaluations rounds reported on different non-iid algorithms under Dirichlet distribution ($\alpha = 0.1$).

Table 3 indicates `FedJETs` leverages the original 73% accuracy from the common expert to reach up to 80% accuracy, even on highly skewed scenarios. Note that, while heterogeneity should decrease the overall performance, `FedJETs` outperforms the methods under comparison, where experts learn to better generalize to unseen data.

C Performance under different sampling ratios

There is an initial degree of randomness in the gating function: during the first couple of iterations, it sends random top K experts to each client, while the experts learn to specialize in the different regions of the label space. However, we found a way to keep consistency during these initial rounds: through the *anchor clients*. Figure 5 shows that by introducing at least 30% *anchor* clients during each round, we can ensure a balance against the wrong selection of the gating function by let them act as *regularizers*. Additionaly, we present Figure 9 showing the impact in performance when we remove the anchor clients rule from sampling and allow only random selection from the pool of available clients.

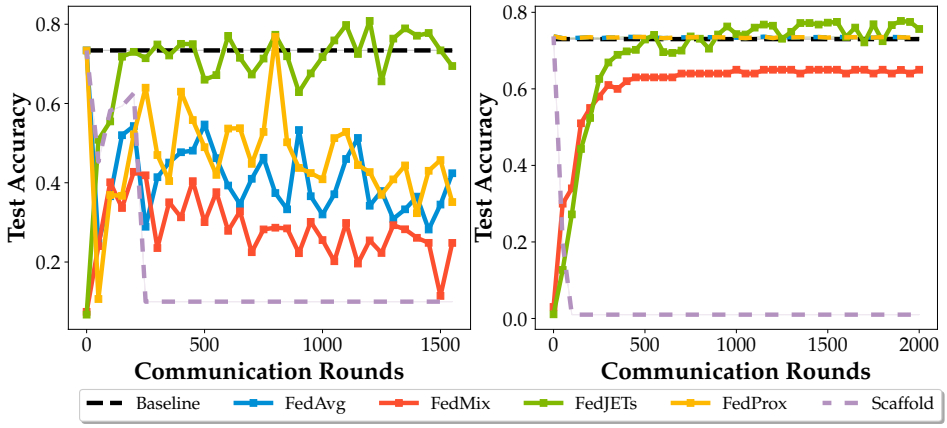

Figure 8: Evaluation of different non-iid algorithms under Dirichlet distribution ($\alpha = 0.1$) on CIFAR10 dataset.

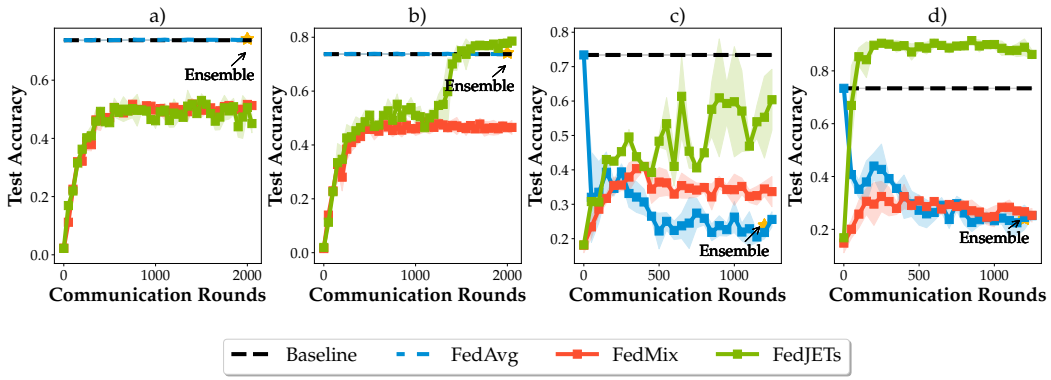

Figure 9: Global testing accuracy for CIFAR10 (a-b) and CIFAR100 (c-d) datasets on two different sampling strategies: a), c) 10 random clients without replacement per iteration, b), d) 5 random *anchor* clients + 5 normal clients without replacement per iteration along different methods.

D Gating function Per-Sample Performance

We perform a thorough evaluation of our gating function after training, using the checkpoints trained with the 73% *common expert* on CIFAR10 and CIFAR100 datasets on the FedJETs algorithm. Our fine-grained evaluation demonstrates that our gating function can analyze the characteristics of each unseen test client's local sample and adaptively select a subset of experts that match those characteristics. This is a crucial step in ensuring that our gating function can generalize well to new data. After selecting the top-K experts, the gating function chooses the highest score/confidence expert to make the prediction for each test data sample. Our results, reported in Table 4, show that our gating function can achieve high accuracy on the selection.

E Incremental Learning

Incremental learning is a paradigm that aims to update and refine existing knowledge from new data, rather than discarding or retraining from scratch. This can be beneficial for scenarios where data is dynamic, scarce, or costly to acquire, and where learning models need to adapt to changing environments or tasks. We performed a comprehensive comparison using the same benchmarking methods in Table 1 to contrast the learning process on each different algorithm.

CIFAR100			
Client	Incorrect	Correct	Error Rate
0	278	722	27.8%
1	281	719	28.1%
2	263	737	26.3%
3	251	749	25.1%
4	261	739	26.1%
5	309	691	30.9%
6	260	740	26.0%
7	285	715	28.5%
8	255	745	25.5%
9	267	733	26.7%
Average Error Rate			27.1%

CIFAR10			
Client	Incorrect	Correct	Error Rate
0	227	3773	5.7%
1	122	3878	3.1%
2	563	3437	14.1%
3	103	3897	2.6%
4	78	3922	2.0%
Average Error Rate			5.5%

Table 4: Evaluation per-sample level on CIFAR10 and CIFAR100 datasets.

E.1 Dynamically increase the client's pool

For this setup, we splitted the CIFAR100 dataset into 5 different groups with non-overlapping labels. Each group held 20 different clients with random samples within the labels range. Then, we allowed only one group of labels to be trained for 200 iterations. Afterwards, we increased the pool of clients with a new group each 200 iterations, monitoring the global accuracy of the models over time. In Figure 10, we can observe that `FedJETs` is not affected if the entire set of clients is not present from the outset; its gating function develops adaptively, without compromising its ability to capture the old distributions. In contrast, `Fed-Mix` drops its performance by approximately 4% compared to the original results in Table 1.

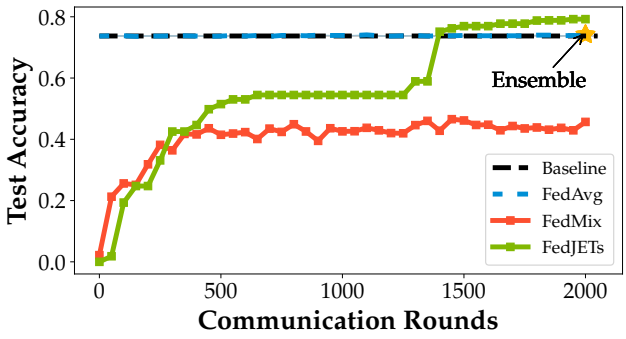

Figure 10: Incremental Learning scenario on CIFAR100, dynamically increasing the total pool of clients.

E.2 Dynamically switch the client's pool

For the second scenario, we employ a cyclical learning approach based on the first setup. Instead of simply increasing the total pool of clients, we only allow one of the five groups of clients to contribute to the training process at a time. This means that every 600 iterations, we switch the pool of available clients, allowing us to see new labels and ensuring that the labels seen during the initial iterations will never be seen again during the training process. This cyclical approach allows us to benefit from the diversity of the data, while also ensuring that the model is constantly being exposed to new information.

Figure 11 illustrates that even when `FedJETs` is approximately 2% below `FedAvg` at the end of training, the former continues to improve while the other methods begin to decline over the iterations. This is likely due to the anchor clients acting as regularizers to adjust the gradient directions during optimization, as the clients pool presents a more difficult setup. The anchor clients are able to provide a more stable optimization process.

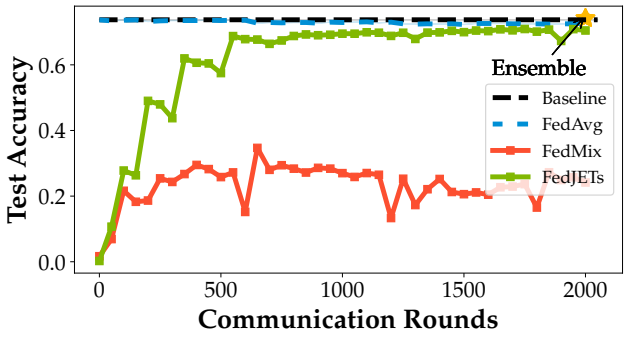

Figure 11: Incremental Learnins scenario on CIFAR100, dynamically switching the total pool of clients

F Performance under matching number of experts $M = K$

We present additional experiments of more versions of `FedMix` vs `FedJETs` using the same number of total experts, meaning $M = K$ in order to disentangle the behaviour of our method under different number of experts. The results are shown in Table 5

	$M = K = 2$	$M = K = 5$
Common Expert	73.39%	73.39%
`FedMix`	42.76%	43.86%
`FedJETs`	**60.16%**	**75.77%**

Table 5: Best Global Test Accuracy reported during training on CIFAR10 dataset under Dirichlet distribution ($\alpha = 0.1$) with fixed number of models communicated to each client. Both methods were initialized from the same *common expert* with an initial accuracy of 73.39%.

G Comparison against Domain Generalization Methods

Our target scenario can be framed as a Domain Generalization problem, thus we evaluated `FedJETs` against state-of-art methods that handle robustness to distribution shifts on test-time. Results in Table 6 demonstrate that the ability of `FedADG` and `FedSR` to evaluate unseen domains is tightly bound to a small number of clients. Once we increase the underlying distribution (e.g. 100 different clients) these methos are not able to exploit the cross-relationship among domains (1).

Common Expert	93.05%
`FedSR`(25)	28.24%
`FedADG`(33)	41.83%
`FedJETs`	**87.86%**

Table 6: Best Global Test Accuracy reported during training on CIFAR10 dataset using quantity-based label imbalance. We sample 10 (of 100 available) random clients during 900 iterations with replacement. All methods were initialized from the same *common expert* reported on the Table.

H Clustering analysis

In order to provide a more extensive comparison of our expert models, it is important to highlight that the core idea is not to summarize clients into several models, such as many clustering related works. Clustering methods are limited to scenarios where clients are inherently grouped; that is, all clients in the same group will have similar local data distributions, while clients across groups will share few data. Instead, we target a more realistic scenario, where each client has a more non-iid and mixed data distribution, making

client clustering based on local distributions meaningless. To illustrate this, we have performed an example of client clustering using K-means on local class distributions as shown in Figure 12, where each dot represents one client and the annotated numbers are the two main data classes of this client. The color represents the K-means clustering result. It is clear that clustering does not create meaningful groups of clients, and training individual experts in each group does not provide any specialization of experts.

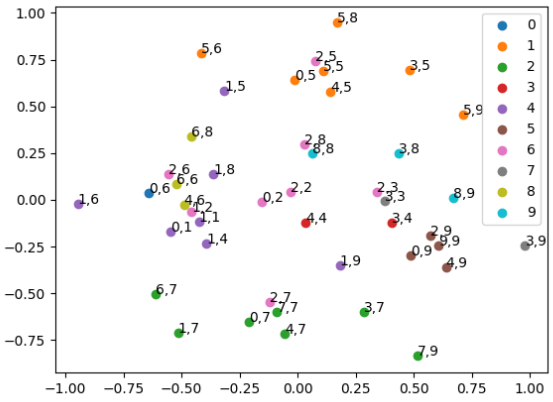

Figure 12: Clients clustering with label frequency.