Tuner: A New Approach For 3D Semantic Segmentation Using Federated Architecture

1st Jerry Chen
Electrical and Computer Engineer
University of Alberta
Edmonton, Canada
jerry3@ualberta.ca

2nd Ruiqing Tian
Electrical and Computer Engineer
University of Alberta
Edmonton, Canada
ruiqing2@ualberta.ca

3rd Di Niu
Electrical and Computer Engineer
University of Alberta
Edmonton, Canada
dniu@ualberta.ca

***Abstract*—3D computer vision (CV) applications have been gaining great popularity in various real-world applications, such as autonomous driving, environmental surveillance, and medical diagnostics. A significant obstacle in advancing 3D CV applications is the scarcity of large-scale, high-quality training datasets. Gathered 3D datasets have the issue of data heterogeneity since comprehensive capturing of 3D representations requires specialized tools and systematic approaches like scanning the object or scene from multiple viewpoints. This frequently results in datasets that are not only limited in size but also exhibit significant variation in attributes like resolutions, lighting conditions, and the distribution of labels. To address these challenges, this paper introduces *Tuner*, a novel strategy for training 3D CV models using datasets from various sources. Tuner leverages a federated learning (FL) framework, allowing the incorporation of extensive data samples. It achieves this through a unique and streamlined model structure linked to each participant's model, designed to effectively address the issue of data heterogeneity. Our experiments show that Tuner can outperform other FL algorithms on various 3D segmentation tasks.**

***Index Terms*—3D CV, Deep Learning, Federated Learning**

I. INTRODUCTION

The domain of 3D computer vision is undergoing a significant paradigm shift, with an increasing emphasis on 3D data processing capabilities, underscoring its essential role in many areas. Fields such as robotics, autonomous driving using Light Detection and Ranging (LiDAR) technology, cutting-edge visual recognition systems like Apple's Vision Pro, and medical imaging increasingly depend on 3D semantic segmentation. This process involves distinguishing various objects from scanned point clouds. However, centrally acquiring large-scale 3D point cloud data for training robust 3D segmentation models faces substantial challenges, including high costs and privacy concerns. Contrary to areas like Natural Language Processing (NLP) and 2D imaging, which have a wealth of publicly available datasets, 3D computer vision suffers from a scarcity of accessible data resources. In light of this challenge, the burgeoning field of edge computing provides a promising method for 3D data acquisition. With the advent and proliferation of edge computing devices, including smartphones, tablets, vehicles, and cameras, there is a shift towards decentralized data collection, favoring data unions over monopolistic behaviors. Taking the context of 3D segmentation as an example, utilizing LiDAR technology embedded in everyday consumer devices allows for the scanning and generation of 3D point clouds from real-world scenes. This approach democratizes the process, making it accessible beyond the realm of high-end professional equipment like the Leica BLK360. Furthermore, data from individual devices raises privacy concerns, which further complicates the data acquisition.

Federated Learning (FL) emerges as a privacy-preserving way that allows for the decentralized training of models within the field of 3D computer vision, thus protecting against data privacy infringements. In this framework, individual users or clients contribute to model training using their local data, thus maintaining data privacy since only the model updates, rather than raw data, are shared with the server. This method facilitates the aggregation of these model updates into a global model, which is then distributed to clients for ongoing training and refinement. Federated Averaging (FedAvg) [18] is widely used as a standard FL strategy that relies on an averaging update scheme. However, this approach can only find a balanced point where the global model performs equally well on each client and is not tailored to any specific client. This may lead to unsatisfactory performance on clients with data distributions distinct from others. Personalized federated learning, (pFL) [8] aims to overcome this limitation and allows for the adaptation of models to unique data distributions of individual clients. The extensive exploration of pFL in 2D data domains, as evidenced by [24], [15], [4], and [6], is attributed to the accessibility of large-scale 2D datasets.

However, the application of pFL in 3D computer vision encounters distinct challenges, primarily due to the variations in data attributes like quality, resolution, and object domains, arising from different capturing devices and environments. These discrepancies can substantially affect the efficacy of model training and its subsequent performance, highlighting the need for a tailored pFL approach in 3D contexts. Furthermore, pFL in 3D computer vision must also contend with issues such as the alignment and integration of models trained on diverse local datasets, requiring advanced data processing and aggregation techniques to ensure consistency. The inherently sparse and dynamic characteristics of 3D environments, where the context and objects within a scene can be sparse and changed rapidly, introduce an extra layer

of complexity. Moreover, the considerable computational and memory requirements for processing extensive 3D data pose great obstacles to the decentralized nature of federated learning. Therefore, formulating an effective pFL method for 3D computer vision, particularly when dealing with decentralized and heterogeneous user-specific data, is essential, representing an important research agenda and a gap to fill, in the literature.

In this paper, we propose *Tuner*, a personalized federated learning framework for 3D semantic segmentation. This framework is designed to accommodate unique data distributions of individual clients while also preserving the generality of the collaboratively trained model. Our approach is characterized by the incorporation of a personalized tuner block for each client. The parameter of each tuner block is not shared across clients, allowing the local adaptation of personalized models. This design enhances the integration of local data and manages the dynamic nature of 3D environments, ensuring the global model remains stable in distinct 3D environments. Consequently, the trained global model, augmented by our personalized Tuner block, achieves generalization across various clients, ensuring both local relevance and broad applicability.

Our study contributes significantly to the field by making the following key contributions:

- A novel framework for 3D Point Cloud Segmentation. We propose Tuner, a lightweight framework for 3D indoor semantic segmentation. By integrating Personalized Federated Learning, Tuner enhances the performance of local models by addressing the challenges posed by the varied and sometimes constrained datasets encountered across clients.
- A new compact encoder model for backbones. We design a parallel encoder tuner block that extracts intermediate feature vectors from different layers within the backbone encoders to enhance models' data processing capabilities.
- Comprehensive Experimental Validation. We conduct extensive experiments on two representative 3D point cloud datasets, S3DIS [2] and ScanNet [5], as well as on a mix of two to evaluate the Tuner framework. We unevenly partitioned these datasets and they are used to mimic the local data owned by distinct clients. The evaluation results further demonstrate the robustness and superior performance of the Tuner framework in handling decentralized 3D data and generalizing heterogeneous data across clients. Overall, Tuner outperforms a range of traditional FL and SOTA personalized FL methods in 3D indoor scene semantic segmentation.

II. RELATED WORK

In this section, we will review the state-of-the-art algorithms for personalized federated learning and 3D point cloud processing.

A. Federated Learning

Federated learning, a type of distributed machine learning technique, was first introduced by Google researchers. Figure 1 shows the structure of FL: It lets multiple clients collaboratively learn from different datasets and eventually finds a balanced server model that performs well for all clients. The clients are not allowed to share data information with other parties, enforcing data privacy. However, federated learning faces potential challenges due to some properties of decentralized data, including the non-IID and unbalanced amount of data. Non-IID refers to a situation in which training data distributions for different clients are not identical. The local data for a given client merely depends on the usage of the local user. Taking the task of image classification as an illustration, one user has images of cars, and another holds furniture images as their client data. This mismatch in clients' data distribution causes a significant reduction in the performance of the model. Unbalanced data is another property that is commonly observed in federated learning, where some clients may have significantly more data than others. This varying amount of local training data for different clients results in biases in the global model, which leads to poor performance in the minority classes.

To alleviate these challenges, [18] proposed two algorithms: FedSGD (Federated Stochastic Gradient Descent) and FedAvg (Federated Averaging). FedSGD is a baseline of federated learning. It is an optimization algorithm that conducts one local SGD on each client with local data, and the server computes an updated model by aggregating the gradients received from clients. FedAvg works similarly but iterates several local SGD updates on clients before aggregating. Eventually, FedAvg became the standard baseline for federated learning tasks.

Several privacy-preserving methods have been developed for FL. McMahan et al. [19] introduced differential private recurrent language models using a noised federated averaging algorithm, achieving strong privacy guarantees without significant loss in accuracy. Li et al. [1] explore the Binomial mechanism for federated learning to minimize communication overhead while ensuring privacy. Other works focus on interactive heavy hitters discovery with central differential privacy [29] and privacy-aware learning with local privacy [7], providing robust privacy guarantees while maintaining learning efficiency. To address data diversity issues, Haddadpour et al. [10] explore local descent methods in federated learning, providing convergence and communication complexity analysis for nonconvex optimization. Li et al. [16] introduce FedProx, a federated optimization framework that improves stability and accuracy in heterogeneous networks. Zhao et al. [28] address federated learning with non-IID data, proposing a globally shared data subset strategy to mitigate accuracy reductions.

Federated learning algorithms mentioned above develop a common global model that is not fully adapted to each user. In contrast, [8] proposed a personalized variant with a personalized model, which can be well suited to each client with a few steps of fine-tuning the model to their local data distribution.

pFedHN [24] is a novel method for solving personalized federated learning using hypernetwork(HN). HN is mounted on the server and simultaneously learns several personalized

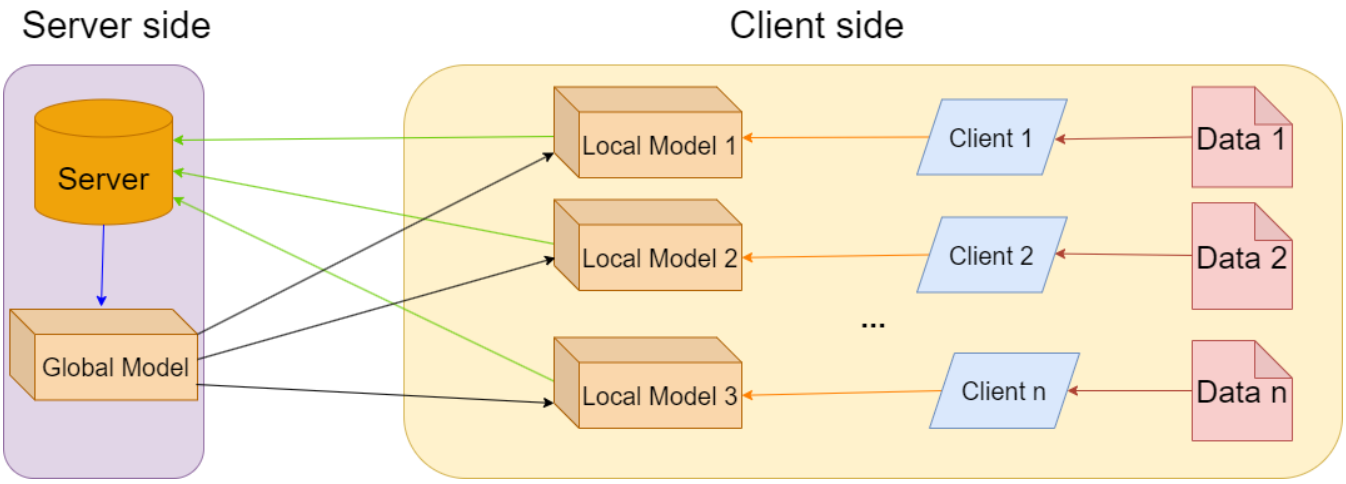

Fig. 1: Conventional FL structure. The clients are connected to the server. Clients and the server only exchange model parameters, they share the data with neither server nor other clients. Red, orange, green, blue and black arrows indicate passing data, local model updating, gradient passing, global aggregation and pass model parameters respectively.

models while data varies among clients. However, this method only works for small models: its HN size grows quickly along with the rise of personalized models' sizes. Ditto [15] is a personalized federated learning model that focuses on robustness and fairness. It uses an extra loss on the difference between the server-client models. FedRep [4] approaches personalization by clients' having their own heads(final projection layers). On top of that, they proposed a learning scheme, such that in local trains, the first few epochs update their classification heads only, then some epochs update model parts other than heads, and lastly, update them both.

pFedMe [6] uses the Moreau Envelope function, which helps the optimization for decomposing the personalization model from global model learning, thus allowing pFedMe to update the global model similarly to FedAvg, but with the ability to optimize the personalization model in parallel based on the local data distribution of each client at certain moments.

[20] used personalized federated learning with Adapter framework on NLP tasks and 2D image processing tasks. However, their approach is restricted to attention-based structures and is difficult to adapt to 3D applications.

FLBN [30] is another approach that uses partial model personalization framework. It keeps the batch norm layers for each client and other model parameters for global aggregation. The issue with this approach is that the improvement depends on the number of batch norm layers in the backbone. In an extreme case, this method is useless if the backbone has no batch norm layers.

B. 3D Point Cloud Processing

3D dataset generally consists of multiple point clouds, and each of them is composed of a large amount of points. Like pixels in 2D datasets, every point contains feature information such as RGB and coordinates(inherent for 2D data). In comparison to 2D images, 3D points do not lie in discrete locations, and density(sparsity) varies a lot within a sample. One of the most popular tasks for 3D point clouds is semantic segmentation. Given a 3D point cloud, the goal is to determine point-wise labels.

PointNet [21] and PointNet++ [22] are initial approaches to work directly on point clouds. PointNet employs a series of multilayer perceptron (MLP) layers to process each point in a scene, allowing each point to gather information from the entire point set. This approach, however, treats points as isolated from one another, which leads to a loss of information about the local structures within the data. To address this shortcoming, PointNet++ introduces a hierarchical model that includes set abstraction (SA) layers. These layers group the input points into smaller groups and aggregate point features for each cluster. The consolidated feature points, termed as anchors, are then forwarded to subsequent SA layers, where they are processed as individual points.

DGCNN [26] treats the problem as a graph. It first finds K-nearest neighbors for every point in the point cloud. Each neighborhood group will be passed to MLPs, and max pooled to get the local features. Then in future layers, they would repeat finding K-nearest neighbors on feature space to build new graphs.

PointConv [27] treats the local region differently. It looks at a local region around a point as a distribution and the neighbors as samples. In order to make the distribution estimation not too biased, the neighbor points are also weighted inversely by their densities when passing to MLPs.

PointNext [23] is one more level extension on Pointnet++. They revisited Pointnet++ and provided data augmentation techniques to improve the training performance. On top of that, they made two improvements to Pointnet++. First, instead of using raw distance from neighbor to p for every cluster in SA, they used normalized distances. These distances are generally tiny, causing non-trivia optimization. Secondly, they

added Inverted Residual MLP (InvResMLP) layers after the SA layer. InvResMLP is composed of another grouping layer, followed by a reduction layer sandwiched by MLPs with skip connections. This addition alleviates gradient vanishing and enhances both neighbor and point feature extraction.

III. METHODS

A. Architecture

The overall architecture follows the conventional FL framework, consisting of a single server and multiple clients. Each client possesses a backbone model and retains exclusive control over its private data. It is presupposed that the data across clients are non-independent and identically distributed (non-IID), prohibiting data sharing. Communication between the server and the clients is exclusively limited to the exchange of model parameters.

The backbone model deployed at the clients is an existing model designed for specific tasks. In our case, PointNext serves as the backbone model. PointNext is designed for 3D segmentation tasks, featuring a lightweight encoder-decoder architecture capable of directly handling raw 3D point cloud data. It operates on raw data points, encompassing XYZ coordinates alongside RGB values, and generates object labels for each point.

The *Tuner block* is a compact encoder model that resembles the structure of the backbone encoder, and every client has one Tuner block attached to it. Tuner blocks extract intermediate feature vectors from different layers within the backbone encoders. Each layer's output within a Tuner block is then fused with the original feature vectors and forwarded to the decoder in the backbone, as illustrated in Figure 2. The layers in the Tuner blocks are scaled-down versions of the backbone encoder, employing a more compact set of abstraction and MLP blocks.

$$\begin{aligned} z_i &= TunerLayer_l(z_{i-1}, p_{i-1}) \\ &= TunerEncoder_l(p_{i-1}, z_{i-1}) \\ &= MLP(SA(p_{i-1}, z_{i-1})) \\ y_i &= Fusion(z_i, x_i) \end{aligned} \tag{1}$$

During the forward pass on the client side, the encoder processes private data as it typically does. Assuming the encoder consists of L layers, the i-th layer receives point information p_{i-1} from the preceding layer (or raw information p_0 if there is no previous layer) and outputs new sampled point information p_i and feature vector x_i.

In the original backbone design, both p_i and x_i are forwarded to the decoder layers to facilitate point-wise label prediction. With the integration of a Tuner block, p_i and x_i are passed to the corresponding i-th layer of the Tuner block instead. Equations (1) show the streamlined operations performed by a layer within the Tuner block. The outcome is a feature vector z_i, akin to x_i but with significantly reduced dimensions. Subsequently, z_i not only advances to the next layer of the Tuner block but also is fused with x_i to create a new feature vector y_i, which is then transmitted to the decoder.

Algorithm 1 Server algorithm in Tuner

Input: C, T, n, t, w^0, V, D

server sends w^0 to all clients
for t = 0, ..., T-1 do **do**
  $C_t \leftarrow$ Server randomly selects n clients from C
  for client $c \in C_t$ parallel **do**
    for l = 0, ..., t-1 **do**
      $w_c, v_c \leftarrow localUpdate(d_c, w_c, v_c)$
    end for
    send w_c back to the server
  end for
  $w^0 \leftarrow serverAggregate(w_c)$
  server sends new w^0 to all clients
end for

The fusion operation is the concatenation of the vectors x_i and z_i.

B. Training Algorithm

Similar to the conventional FL framework, Tuner operates over a predetermined number of rounds. During each round, a subset or all participating clients are tasked with training and updating their respective backbone and Tuner block models. This process involves a fixed number of epochs of training on their local datasets. Upon completion of local updates, clients transmit their updated backbone models to the central server. After receiving the updates, the server aggregates the updates (this aggregation involves averaging the updates with weights proportional to the dataset sizes contributed by each client [18]) and passes the updated backbone model parameters back to all clients, ensuring uniformity in backbone model parameters across the network.

Algorithm 1 demonstrates the update process of Tuner. The flow is very similar to the conventional FL framework. C represents all clients, T is the total number of rounds, n would be the number of clients to pick per round, w^0 is the server model weight(PointNext weights in this case), $v_c \in V$ is the Tuner block weights for the client c, and $d_c \in D$ is the private only available to client c.

The $serverAggregate$ function employs weighted model parameter averaging, akin to the FedAvg method. It is important to note that only the shared backbone model weights are exchanged between the server and the clients, not the weights of the Tuner. This design choice ensures that the framework does not impose additional network traffic burdens and privacy preservation.

IV. EXPERIMENTS

The experiments were conducted on the Linux Ubuntu operating system with one single RTX 3090 GPU. We ran the experiments with a batch size of 4 for training and 1 for evaluation/testing on the following datasets.

A. Datasets

There are two major indoor 3D scan datasets, S3DIS and ScanNet. Both datasets are real-world 3D scans of indoor

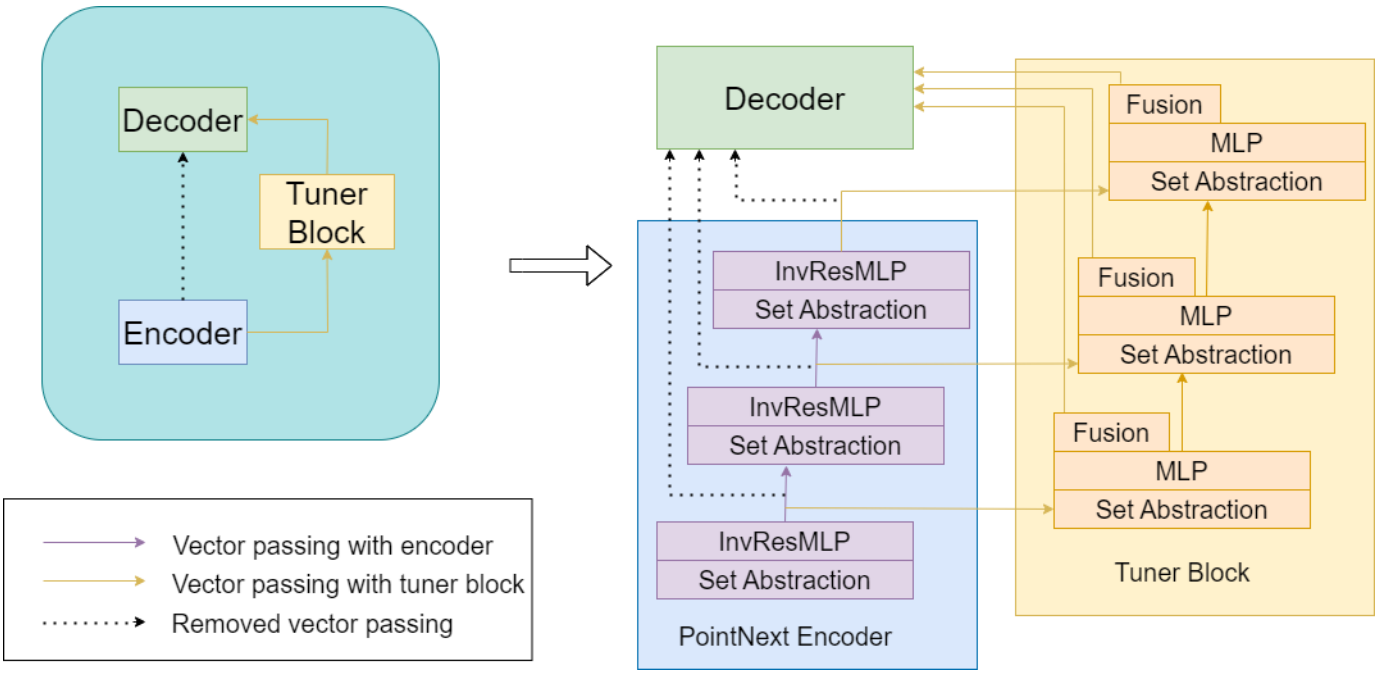

Fig. 2: Personalized FL with Tuner structure. On the left is the rough overview of each client. Tuner blocks replace the connection between the backbone encoders and decoders. The detailed view of the connections for a 3-layer structure is on the right side. The fusion block takes the vectors from the corresponding encoder layer and fuses them with the output of the current tuner block layer. The decoder remains the same except the initial dimension of each layer.

scenes. Every scene is made of a large number of points, and every point has its XYZ coordinates and RGB values recorded. These datasets are also manually labeled such that each point is assigned an object label. These labels are common objects we can find indoors, such as tables and chairs. S3DIS contains over 250 scenes captured from various rooms on university campuses. Therefore the majority of objects are tables, desks, and bookshelves. On the other hand, ScanNet has over 1500 scenes, mostly scanned from residential areas like living rooms and bathrooms. The labels for ScanNet are dining tables, toilets, sofas, and more items commonly found in one's homes.

To evaluate our model under the scenario of heterogeneous data distribution, we have designed multiple test cases. In the first test scenario, we want to run tests on the same dataset, where all the clients get local data from a single dataset. We separate the original 3D dataset into 6 and 12 partitions, respectively, to make each partition purposely contain different proportions of objects. Although some partitions may overlap with others on a small portion of samples, most scenes in a particular partition have the same specific object type, while the others barely have any(for example, a client's partition may contain desks in almost every sample, while other clients do not have that many desks). This is how we differ the data distributions among distinct clients, as they all have unbalanced labels.

Figure 3 shows the distributions of chairs in both S3DIS and ScanNet partitions under the settings of 12 and 6 partitions respectively. We can see that some clients are full of scenes with chairs, while a few clients have barely any scenes containing chairs. This unbalanced amount of data causes difficulties in FL tasks.

On top of unbalanced labels, figure 4 also suggests various sample sizes(number of points) in different clients from the S3DIS dataset. We collected the mean and standard deviation(STD) values for every client in the S3DIS setup. We can see that in both setups, the means have some fluctuations. If we look at STDs, we observe severe data distribution gaps. Generally, STD measures how spread the samples are from each other. Large STD suggests sample sizes are very different from each other, and vice versa. The charts show both high STD(i.e., client 4 in left figure 4) and low STD(client 3 in left figure4), further illustrating the high heterogeneity amount of the data partitions. We did not calculate means or STDs for the ScanNet dataset as we sampled it such that every sample has the same number of points. Due to the large dataset size and the great number of points in every ScanNet scene, sampling allows us to run experiments in a shorter time and observe FL on another aspect: different densities.

Figure 5 collected statistics of densities, the number of points within a cubic meter, on the sampled ScanNet dataset for 6 and 12 partition settings respectively. In a similar manner, we recorded the density of each scene for every partition and collected the mean and STD values. As illustrated in the charts, even though some of the statistical mean and STD values of density are similar, there are still a few distinctions showing

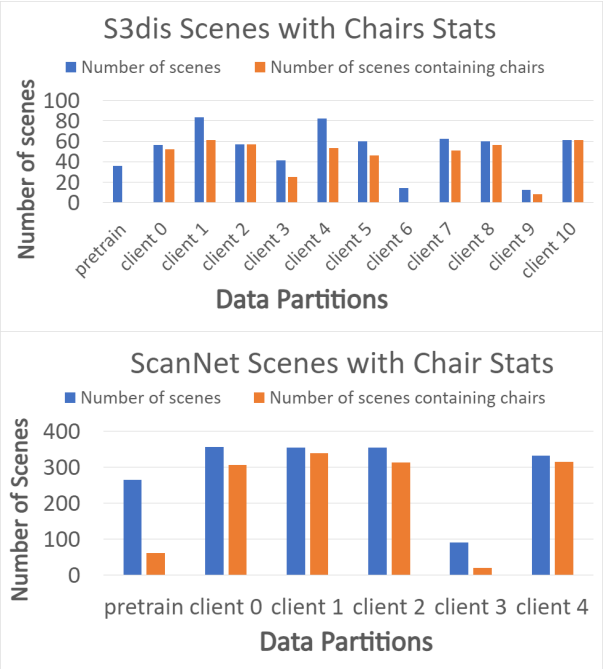

Fig. 3: Distribution of the scenes containing chairs in different setups. The blue bars indicate the total number of scenes in each client, and the orange bars show the number of scenes containing chairs. As a comparing example in S3dis chart, all scenes from client 10 contain chairs, but scenes from client 6 do not have any chairs.

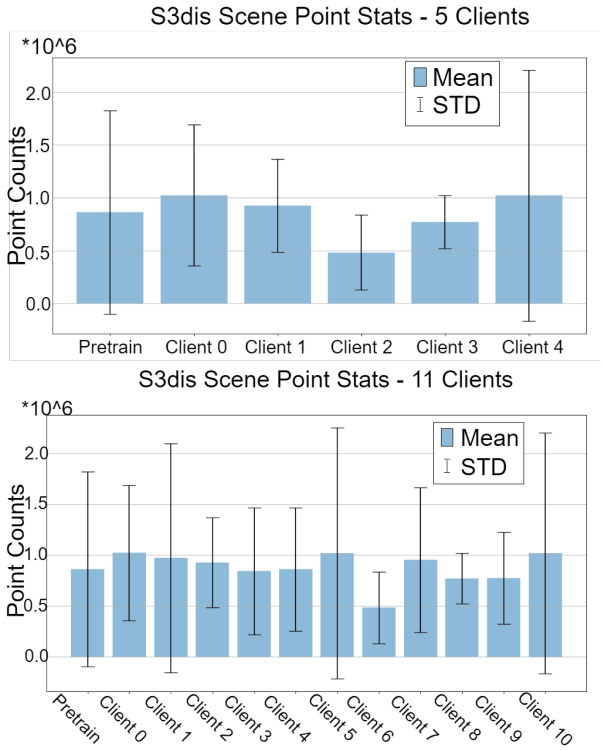

Fig. 4: The means and standard deviations of point counts for every client in different S3DIS setups. Blue bars show the average number of points in each client, while the line segments show plus/minus 1 standard deviation from the means. Long STD line segment with respect to its mean indicates extreme variations in the number of points from the scenes.

values with considerable differences. This causes an inevitable deviation in the density value for scenes in each partition.

From the above observations on data balance, point, and density statistics, we claim that the partitions on both datasets have non-IID properties and are well-suited for evaluating FL algorithms on various aspects.

B. Backbone Setup

As mentioned earlier, we use PointNext as a backbone for the experiments. It is not too complicated yet provides satisfying results on 3D datasets. We also replaced the backbone for other baseline FL methods so they are compatible with processing 3D data. We deployed PointNext-l as we want to have a balance between a reasonably well-performing backbone and a relatively small model size to save experiment time length.

C. FL Setup

At the beginning of every run, we use one of the partitions to train the backbone in the server as a warm-up for a fixed number of epochs, and other partitions are assigned to clients as their private data. When the server is done with the warm-up, FL starts. Due to small number of samples available from the 3D datasets(250+ samples for S3DIS and 1500+ for ScanNet), we have to limit the number of clients so every client has enough samples to learn. In a 5-client setup, we do both updates on all 5 and update on 3 random clients every round. For 11 clients, we only pick 5 clients randomly per round. The second scenario is that we mix both datasets. This task is much more complicated as the data distributions are vastly different(scene density, label varieties, etc.). We combine the earlier partitions to form a difficult test of 12(6 partitions from S3DIS and 6 partitions from ScanNet) clients, then pick 5 random clients per round. For the classification to work on two different sets of labels, we train on two classifiers, one for each dataset, and they are shared for clients using the data from the same dataset. We did not do a warm-up as we did not want the model to get biased to a specific dataset.

The PointNext backbone contains 7.13 million parameters. In comparison, the Tuner model only contains 1.75 million parameters.

We used 200 epochs to warm up the server models (except the mixed case), then the models were distributed to all clients. Next, we start the FL algorithm by letting the picked clients run 10 local epochs on their private data, pick their best models based on the evaluation results on their local validation datasets, and pass the model parameters back to the server for 100 rounds(150 for mixed data scenario due to not having a

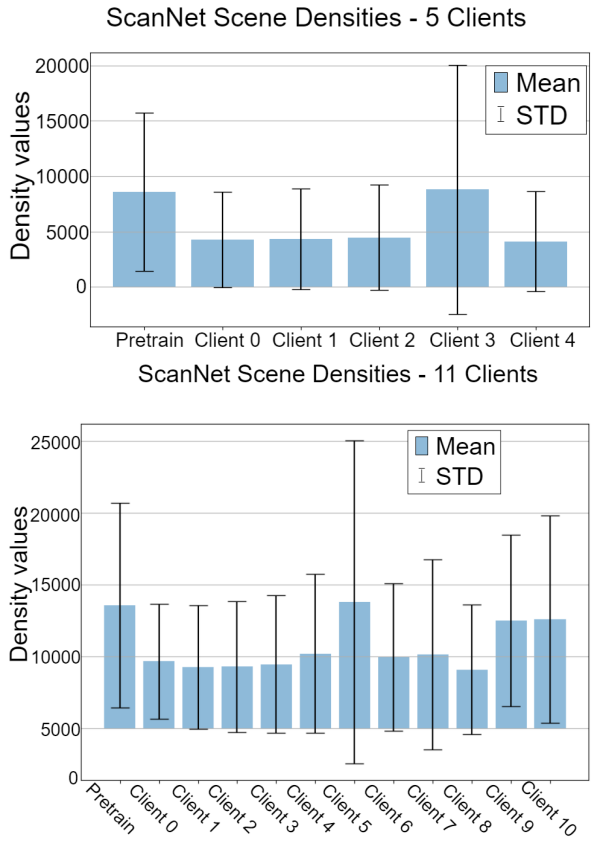

Fig. 5: The means and standard deviations of densities for every client in different setups. Blue bars show the average densities of points in each client, while the line segments show plus/minus 1 standard deviation from the means. Long STD line segment with respect to its mean indicates extreme variations in scene densities.

warm-up step).

Evaluation results are collected from each client on their test sets or merged test sets. The general metric to measure performance in 3D semantic segmentation is to find the IoUs(intersection over union) of ground truth labels and predicted truth labels. IoU is a better metric than accuracy when the labels are unbalanced. We find the mIoU(mean IoU, the weighted average of IoUs from all labels of current client) for every client and then find the average IoU to get the overall performance. In order to make sure the results are stable, we use the averaged results from the last ten rounds.

D. Test Results

Both table I and table II belong to scenario one. The number pairs on the first row indicate the number of randomly picked clients and the total number of clients in their setups. The values under setups without stars are evaluated conventionally: each client is evaluated on its own test set. On the other hand, the ones with stars are evaluated on a merged test dataset. The merged test sets are combinations of private test sets from all clients. We use the second test to measure how well the

TABLE I: Miou on S3DIS dataset

Method	3/5	5/5	5/11	3/5*	5/5*	5/11*
FedAvg	28.83	47.40	21.43	-	-	-
pFedMe	6.46	6.86	6.82	5.21	6.75	7.03
Ditto	23.83	23.46	23.01	29.48	28.39	29.63
FedRep	24.81	27.22	23.49	25.98	27.18	25.81
Tuner	**43.12**	**40.81**	**47.36**	**42.05**	**43.48**	**42.31**

TABLE II: Miou on ScanNet dataset

Method	3/5	5/5	5/11	3/5*	5/5*	5/11*
FedAvg	19.86	28.50	14.33	-	-	-
pFedMe	9.10	12.98	10.93	8.38	12.54	10.12
Ditto	30.97	32.79	28.66	31.56	**36.64**	**31.94**
FedRep	26.89	28.59	25.70	27.03	29.20	27.10
Tuner	**38.65**	**38.72**	**32.08**	**34.55**	34.28	30.11

TABLE III: MIoU on mixed datasets. Training is done with half of the clients using S3DIS data and the other half of clients with ScanNet data. Evaluation is calculated separately on S3DIS clients and ScanNet clients' test sets.

PFL method 5/12	S3DIS	ScanNet
FedAvg	18.82	9.38
pFedMe	8.91	8.63
Ditto	7.39	11.90
FedRep	6.88	3.02
Tuner	**37.79**	**31.48**

personalized models can generalize on all distributions despite having personalized structures. This is why numbers from FedAvg are not recorded as they do not have personalization, making standard tests equivalent to merged tests. From the results, we can conclude that Tuner can either outperform or come close to other methods. Looking at convention tests(the ones without stars), Tuner only loses to FedAvg on the S3DIS all-5-clients case. However, all clients rarely get to run in every single round. On the other side, merged tests also indicate that Tuner has the best performance on most test cases.

Table III has the evaluation results from scenario 2. All methods are tested with 6+6 clients using data partitions from S3DIS and ScanNet. Every client is evaluated on its test set, and we calculate the mean IoUs for clients using data from the same dataset(the 6 clients using S3DIS are evaluated together, and the other 6 are evaluated for ScanNet). The results from the table strongly suggest that the Tuner outperforms other methods by a large margin on both datasets. These two datasets have very different scene sizes, point densities, and object labels, which makes most federated learning methods unable to capture the information or find the balance between different distributions. The extra Tuner structure can capture the local distribution information while letting the server model learn mutual information.

Both Fig. 6 and Fig.7 show the growth trends of the algorithms. Every round, 5 random clients are picked, so sometimes more data is collected from S3DIS and sometimes collected mainly from ScanNet, and this causes the distribu-

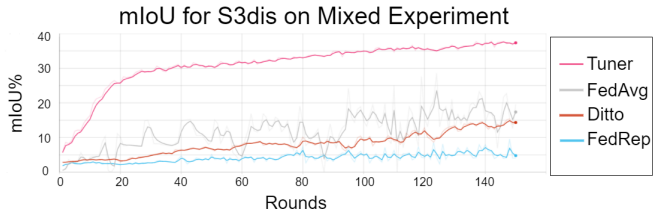

Fig. 6

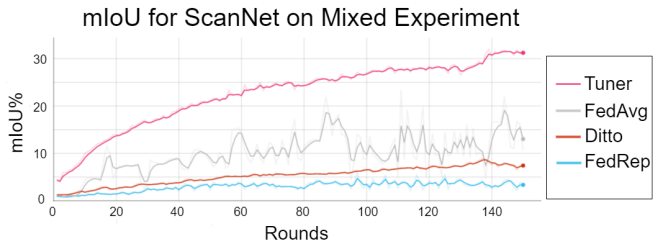

Fig. 7

tions learned by the server to drift back and forth between the two major distributions. On both datasets, we can see that Tuner is growing steadily(and seems to keep growing potentially), while the others do not do so well. FedAvg had been very unstable. We can see this issue by comparing the grey lines in two graphs, especially around rounds 20, 40, and 60, where if performance rises on one dataset, it will drop on the opposite. The results of FedRep and Ditto indicate that they cannot learn from this challenging test case. Their growths are almost flat, and they barely have any fluctuations.

V. Conclusion

In this paper, we proposed Tuner, a brand-new personalized federated learning framework, where a personalized Tuner is deployed on each client, incorporating a shared representation. We partitioned two representative 3D datasets to generate heterogeneous data. Moreover, through comprehensive experiments, we demonstrate that Tuner addresses the challenges of personalized federated learning on 3D indoor scene segmentation tasks and permits effective learning from distinct data distributions. Furthermore, Tuner outperforms other baseline methods in terms of overall performance and the model's generalization ability to heterogeneous datasets. Tuner has the potential to be used in diverse applications, and our work provides a promising direction for future research on personalized federated learning in 3D point-cloud related tasks.

References

[1] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. B. McMahan, "CpSGD: Communication-Efficient and Differentially-Private Distributed SGD," in Proc. of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Curran Associates Inc., Red Hook, NY, USA, pp. 7575–7586, 2018.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," ArXiv e-prints, arXiv:1702.01105 [cs.CV], Feb. 2017.

[3] D. Caldarola, B. Caputo, and M. Ciccone, "Improving Generalization in Federated Learning by Seeking Flat Minima," arXiv:2203.11834 [cs.LG], 2022.

[4] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting Shared Representations for Personalized Federated Learning," arXiv:2102.07078 [cs.LG], 2023.

[5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in Proc. of the Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[6] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized Federated Learning with Moreau Envelopes," arXiv:2006.08848 [cs.LG], 2022.

[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Privacy Aware Learning," J. ACM, vol. 61, no. 6, Article 38, pp. 1–57, Dec. 2014.

[8] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach," in Proc. of the 34th International Conference on Neural Information Processing Systems (NIPS'20), Curran Associates Inc., Red Hook, NY, USA, Article 300, 2020.

[9] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware Minimization for Efficiently Improving Generalization," in International Conference on Learning Representations, https://openreview.net/forum?id=6Tm1mposlrM, 2021.

[10] F. Haddadpour and M. Mahdavi, "On the Convergence of Local Descent Methods in Federated Learning," arXiv:1910.14425 [cs.LG], 2019.

[11] S. Hochreiter and J. Schmidhuber, "Flat minima," Neural Computation, vol. 9, no. 1, pp. 1–42, 1997.

[12] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging Weights Leads to Wider Optima and Better Generalization," arXiv:1803.05407 [cs.LG], 2019.

[13] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic Generalization Measures and Where to Find Them," arXiv:1912.02178 [cs.LG], 2019.

[14] J. Kwon, J. Kim, H. Park, and I. K. Choi, "ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks," in Proc. of the 38th International Conference on Machine Learning (PMLR, Vol. 139), pp. 5905–5914, 2021.

[15] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and Robust Federated Learning Through Personalization," arXiv:2012.04221 [cs.LG], 2021.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," arXiv:1812.06127 [cs.LG], 2020.

[17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution On X-Transformed Points," in Advances in Neural Information Processing Systems, Vol. 31, S. Bengio et al. (Eds.), Curran Associates, Inc., 2018.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proc. of the 20th International Conference on Artificial Intelligence and Statistics, Vol. 54, A. Singh and J. Zhu (Eds.), PMLR, pp. 1273–1282, 2017.

[19] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Recurrent Language Models," arXiv:1710.06963 [cs.LG], 2018.

[20] K. Pillutla, K. Malik, A.-R. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, "Federated Learning with Partial Model Personalization," in Proc. of the 39th International Conference on Machine Learning, Vol. 162, K. Chaudhuri et al. (Eds.), PMLR, pp. 17716–17758, 2022.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," arXiv:1612.00593 [cs.CV], 2017.

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," arXiv:1706.02413 [cs.CV], 2017.

[23] G. Qian, Y. Li, H. Peng, J. Mai, H. A. K. Hammoud, M. Elhoseiny, and B. Ghanem, "PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies," arXiv:2206.04670 [cs.CV], 2022.

[24] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized Federated Learning using Hypernetworks," arXiv:2103.04628 [cs.LG], 2021.

[25] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, J. Ji, G. Cao, D. Jiang, and M. Zhou, "K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters," arXiv:2002.01808 [cs.CL], 2020.

[26] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," arXiv:1801.07829 [cs.CV], 2019.
[27] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," arXiv:1811.07246 [cs.CV], 2020.
[28] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," 2018, https://doi.org/10.48550/ARXIV.1806.00582.
[29] W. Zhu, P. Kairouz, B. McMahan, H. Sun, and W. Li, "Federated Heavy Hitters Discovery with Differential Privacy," arXiv:1902.08534 [cs.CR], 2020.
[30] Li, Xiaoxiao, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou, "Fedbn: Federated learning on non-iid features via local batch normalization." arXiv preprint arXiv:2102.07623 (2021).