ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition

ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition

Xiaomin Ouyang
The Chinese University of Hong Kong
Hong Kong SAR, China
xmouyang@link.cuhk.edu.hk

Zhiyuan Xie
The Chinese University of Hong Kong
Hong Kong SAR, China
xavier_ie@link.cuhk.edu.hk

Jiayu Zhou
Michigan State University
East Lansing, MI, USA
jiayuz@egr.msu.edu

Jianwei Huang
The Chinese University of Hong Kong, Shenzhen
Shenzhen Institute of Artificial Intelligence and Robotics for Society
Shenzhen, China
jianweihuang@cuhk.edu.cn

Guoliang Xing*
The Chinese University of Hong Kong
Hong Kong SAR, China
glxing@cuhk.edu.hk

ABSTRACT

Federated Learning (FL) has recently received significant interests thanks to its capability of protecting data privacy. However, existing FL paradigms yield unsatisfactory performance for a wide class of human activity recognition (HAR) applications since they are oblivious to the intrinsic relationship between data of different users. We propose ClusterFL, a similarity-aware federated learning system that can provide high model accuracy and low communication overhead for HAR applications. ClusterFL features a novel clustered multi-task federated learning framework that maximizes the training accuracy of multiple learned models while automatically capturing the intrinsic clustering relationship among the data of different nodes. Based on the learned cluster relationship, ClusterFL can efficiently drop out the nodes that converge slower or have little correlation with other nodes in each cluster, significantly speeding up the convergence while maintaining the accuracy performance. We evaluate the

KEYWORDS

Human activity recognition, Federated learning, Clustering, Multi-task learning, Communication optimization

ACM Reference Format:
Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition. In *The 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21), June 24-July 2, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458864.3467681

1 INTRODUCTION

Recently deep learning has been increasingly adopted in human activity recognition [11, 51, 61]. Most of the previous work is focused on the *centralized learning* approach that needs to be trained centrally

Related Work

- Human Activity Recognition (HAR)
  - Machine Learning has been increasingly adopted in human activity recognition. Various algorithms based on handcrafted features and deep neural networks have been developed to classify different human activities.
- Federated Learning
  - It is a distributed machine learning approach that enables training on a large corpus of decentralized data residing on devices.
  - Several existing FL approaches aim to learn a single model for all users by averaging the model weights.
  - However, the single learned model usually has limited generality, making it poorly suited for heterogeneous user data in HAR applications
- HAR with Federated Learning
  - Sozinov et al. utilize FedAvg for HAR
  - Chen et al. propose a federated transfer learning framework for Parkinson's disease auxiliary diagnosis
  - However, these transfer learning methods are based on the FedAvg model without exploiting the relationships among nodes.

ClusterFL

- A similarity-aware federated learning system.
  - High system accuracy
  - Low communication overhead
- Motivation
  - Despite the heterogeneity, data distributions of different users' activities may share significant spatial-temporal similarity which can be captured by cluster structures.
    - Subjects' biological features
    - Physical environment
    - Sensor biases
- *Clustered multi-task federated learning*
  - Cluster indicator matrix indicating the similarity of users
- *Cluster-wise straggler dropout*
- *Correlation-based node selection*

Impact of Clusterability

Centralized-cluster:

- K-means clustering → Training a model for each cluster using data of all subjects who belong to the cluster.

Compare with:

- Local learning
  - Overfitting
- Centralized single model learning
  - Privacy concern
- Federated average
- Federated transfer learning

Method	Local	FedAvg	FTL	Centralized-single	Centralized-cluster
Mean Accuracy (%)	53.67	33.61	54.61	72.22	73.17
STD (%)	9.74	12.812	10.19	6.06	3.40

Table 2: Accuracy comparison of different paradigms.

System Architecture

Cluster-wise straggler dropout

- The server will drop stragglers who converge slower than other nodes within each cluster

Correlation-based node selection

- The server will drop nodes that are less related to others in the same cluster

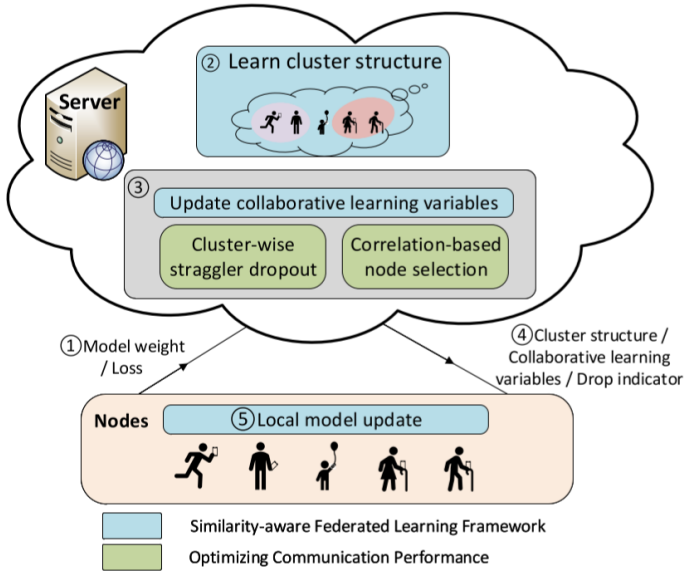


Figure 2: System Architecture of ClusterFL

Cluster Indicator Matrix

Problem Formulation

$$\min_{\mathbf{W},\mathbf{F}} \sum_{i=1}^{M} \frac{1}{N_i} \sum_{r=1}^{N_i} l(\mathbf{w}_i^T \mathbf{x}_i^r, y_i^r) + \alpha tr(\mathbf{W}\mathbf{W}^T) - \beta tr(\mathbf{F}^T \mathbf{W}\mathbf{W}^T \mathbf{F}) \quad (1)$$

- M is the number of total involved nodes, N_i is number of training data samples in node i. α and β are the hyperparameters and $\alpha \geq \beta > 0$.
- $(\boldsymbol{x}_i^r, y_i^r) \in \mathbb{R}^D \times \mathbb{R}$ is the r-th training pair of i-th node; $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M]^T \in \mathbb{R}^{M \times D}$ is the weight matrix to be estimated where each local model is an activity classifier; l is the loss function of local model.
- $\mathbf{F} \in \mathbb{R}^{M \times D}$ is an orthogonal cluster indicator matrix, with $F_{i,j} = \frac{1}{\sqrt{N_i}}$ if node i belongs to j-th cluster and $F_{i,j} = 0$ otherwise. Here K is the number of clusters and $\mathrm{N_j}$ denotes the number of nodes in j-th cluster.

Cluster Indicator Matrix

Problem Formulation

$$\min_{\mathbf{W},\mathbf{F}} \sum_{i=1}^{M} \frac{1}{N_i} \sum_{r=1}^{N_i} l(\mathbf{w}_i^T \mathbf{x}_i^r, y_i^r) + \alpha tr(\mathbf{W}\mathbf{W}^T) - \beta tr(\mathbf{F}^T \mathbf{W}\mathbf{W}^T \mathbf{F})$$

$$(\alpha - \beta) \sum_{i=1}^{M} ||\mathbf{w}_i||_2^2 + \beta \sum_{j=1}^{K} \sum_{v \in \mathcal{S}_j} ||\mathbf{w}_v - \overline{\mathbf{w}_j}||_2^2$$

- L_2-norm regularization of model weights
  - Prevent overfitting
- K-means clustering
  - Minimize the overall intra-cluster distances of model weights

Alternating Optimization

Alternating Direction Method of Multipliers (ADMM)

Algorithm 1: Alternating Optimization of ClusterFL

Initialization : set $\mathbf{W} = \mathbf{0}^{M \times D}$, $\mathbf{F} = \mathbf{0}^{M \times M}$

for $h = 0$ *to* H **do**
  1. Optimization of $\mathbf{W}$ with $\mathbf{F}$ fixed.
  for $t = 0$ *to* T **do**
    node update: parallelly update $\mathbf{w}_i$(i = 1,...,M)
      $\mathbf{w}_i^{t+1} \leftarrow$ **Local SGD**$(\mathbf{w}_i^t, (\mathbf{x}_i^r, y_i^r), \mathrm{F}, \Omega, \mathrm{U})$
    server update: update Ω and U
    for $j = 1$ *to* K **do**
      $\Omega_j^{t+1} \leftarrow \arg\min_{\Omega_j} \mathrm{L}_\rho(\Omega_\mathrm{j}, \mathrm{U}_j^t, \mathrm{F}_j^t, \mathbf{W}^{t+1})$
      $\mathrm{U}_j^{t+1} \leftarrow \mathrm{U}_j^t + \rho(\mathrm{F}_j \mathbf{W}^{t+1} - \Omega_j^{t+1})$
    end
  end
  2. Optimization of $\mathbf{F}$ with $\mathbf{W}$ fixed.
    server update: Update $\mathbf{F}$ based on Section 5.3
end

Optimize Model Weights

Let $\mathbf{C}\boldsymbol{\Omega} = \boldsymbol{F}^T\boldsymbol{W} \in \mathbb{R}^{K\times D}$

$$\min_{\mathbf{W},\Omega} f(\mathbf{W}) + g(\Omega)$$

$$s.t.\ \mathbf{F}^T\mathbf{W} - \Omega = 0$$

where: $$f(\mathbf{W}) = \sum_{i=1}^{M} \frac{1}{N_i} \sum_{r=1}^{N_i} l_t(\mathbf{w}_i^T \mathbf{x}_i^r, y_i^r) + \alpha tr(\mathbf{W}\mathbf{W}^T)$$

$$g(\Omega) = -\beta tr(\Omega\Omega^T)$$

Node Update

2 1. Optimization of $\mathbf{W}$ with $\mathbf{F}$ fixed.
3 **for** $t = 0$ *to* T **do**
4 **node update:** parallelly update $\mathbf{w}_i$(i = 1,...,M)
5 $\mathbf{w}_i^{t+1} \leftarrow$ **Local SGD**$(\mathbf{w}_i^t, (\mathbf{x}_i^r, y_i^r), \mathrm{F}, \Omega, \mathrm{U})$
6 **server update:** update Ω and U
7 **for** $j = 1$ *to* K **do**
8 $\Omega_j^{t+1} \leftarrow \arg\min_{\Omega_j} \mathrm{L}_\rho(\Omega_\mathrm{j}, \mathrm{U}_j^t, \mathrm{F}_j^t, \mathbf{W}^{t+1})$
9 $\mathrm{U}_j^{t+1} \leftarrow \mathrm{U}_j^t + \rho(\mathrm{F}_j \mathbf{W}^{t+1} - \Omega_j^{t+1})$
10 **end**
11 **end**

Node Update (line 4-5 in Algorithm 1): Each node will parallelly optimize (e.g., using gradient descent methods) its model weight $\mathbf{w}_i$ based on its local data $(\mathbf{x}_i, \mathbf{y}_i)$, the cluster structure $\mathbf{F}$ and the collaborative learning variables Ω, U from the server.

$$\mathbf{w}_i^{t+1} = \arg\min_{\mathbf{w}_i} \frac{1}{N_i} \sum_{r=1}^{N_i} l_t(\mathbf{w}_i^T \mathbf{x}_i^r, y_i^r) + (\alpha + \frac{\rho}{2} \sum_{j=1}^{K} F_{ij}^2) ||\mathbf{w}_i||_2^2$$

$$+ \sum_{j=1}^{K} F_{ij} (\mathrm{U}_j^t - \rho \Omega_j^t)^T \cdot \mathbf{w}_i$$

Server Update

1. Optimization of $\mathbf{W}$ with $\mathbf{F}$ fixed.
2. **for** $t = 0$ *to* T **do**
3. **node update:** parallelly update $\mathbf{w}_i$(i = 1,...,M)
4. $\mathbf{w}_i^{t+1} \leftarrow$ **Local SGD**$(\mathbf{w}_i^t, (\mathbf{x}_i^r, y_i^r), \mathrm{F}, \Omega, \mathrm{U})$
5. **server update:** update Ω and U
6. **for** $j = 1$ *to* K **do**
7. $\Omega_j^{t+1} \leftarrow \arg\min_{\Omega_j} \mathrm{L}_\rho(\Omega_\mathrm{j}, \mathrm{U}_j^t, \mathrm{F}_j^t, \mathbf{W}^{t+1})$
8. $\mathrm{U}_j^{t+1} \leftarrow \mathrm{U}_j^t + \rho(\mathrm{F}_j\mathbf{W}^{t+1} - \Omega_j^{t+1})$
9. **end**
10. **end**

Server Update (line 6-10 in Algorithm 1): The server will further utilize the newly-updated model weights from nodes $\mathbf{W}$ and the cluster structure $\mathbf{F}$ (optimized in Section 5.3) to update the collaborative learning variables Ω, U. For j = 1,...,K:

$$\Omega_j^{t+1} = \arg\min_{\Omega_j} -\beta||\Omega_j||_2^2 - \Omega_j \cdot (\mathrm{U}_j^t)^T + \frac{\rho}{2}||\Omega_j - \mathrm{F}_j^T\mathbf{W}^{t+1}||_2^2$$

$$\mathrm{U}_j^{t+1} = \mathrm{U}_j^t + \rho(\mathrm{F}_j^T\mathbf{W}^{t+1} - \Omega_j^{t+1})$$

Learn Cluster Structure

With W is fixed, problem (1) w.r.t F can be seen as a K-means clustering problem on the nodes' model weights W.

- Quantify the similarity of model weights
- Optimize cluster structure dynamically without knowing the number of cluster K.

Similarity of Model Weights.

Kullback–Leibler divergence(KLD)

$$D_{KL}(\mathbf{w}_i, \mathbf{w}_j) = \frac{1}{N_O} \sum_{r=1}^{N_O} \delta(\mathbf{w}_i, \mathbf{x}_o^r) log \frac{\delta(\mathbf{w}_i, \mathbf{x}_o^r)}{\delta(\mathbf{w}_j, \mathbf{x}_o^r)}$$

$$\delta(\mathbf{w}_i, \mathbf{x}_o^r) = softmax(\frac{\Phi(\mathbf{w}_i, \mathbf{x}_o^r)}{\tau})$$

where $\Phi(\mathbf{w}_i, \mathbf{x}_o^r)$ denotes the pre-softmax output of model $\mathbf{w}_i$ on the input data $\mathbf{x}_o^r$. Smaller KL divergence denotes closer relationship of models for node i and j. Then we calculate the KL divergence between any two models and therefore obtain a KL divergence matrix $\mathbf{D} \in \mathbb{R}^{M \times M}$, with $\mathbf{D}_{i,j} = D_{KL}(\mathbf{w}_i, \mathbf{w}_j)$.

Gibbs' inequality:

$$-\sum_{i \in I} p_i \ln q_i \geq -\sum_{i \in I} p_i \ln p_i .$$

Optimize Cluster Relationship

Principal Components Analysis (PCA) → K-means clustering

$\mathbf{F} \in \mathbb{R}^{M \times D}$ is an orthogonal cluster indicator matrix, with $F_{i,j} = \frac{1}{\sqrt{N_i}}$ if node i belongs to j-th cluster and $F_{i,j} = 0$ otherwise. Here K is the number of clusters and $\mathrm{N_j}$ denotes the number of nodes in j-th cluster.

According to [14], principal components are the continuous solutions to the discrete cluster membership indicators for K-means clustering. Suppose there are total M clusters among nodes, then $\mathbf{P} = \mathbf{Q}_{M-1}\mathbf{Q}_{M-1}^T \in \mathbb{R}^{M \times M}$ will be the continuous solution of $\mathbf{F}$, where $\mathbf{Q}_{M-1} = (\mathbf{v}_1, ..., \mathbf{v}_{M-1})$ collects the $M - 1$ principal components of $\mathbf{D}$ using principal components analysis (PCA).

If the data of nodes has a cluster structure, $\mathbf{P}$ is expected to yield a similar diagonal block structure after permutation clusters together. However, as $\mathbf{P}$ is an approximation of the discrete valued indicators, there may be outliers in $\mathbf{P}$. We recover $\mathbf{F}$ more accurately as follows: 1) we set $P_{ij} = 0$ if $P_{ij} < 0$ as $\mathbf{P}$ could contain negative elements; 2) we scale each element of $\mathbf{P}$ as $F_{ij} = \frac{P_{ij}}{\sqrt{\sum_{i=1}^{M} P_{ij}}}$ to obtain the final normalized cluster indicator matrix $\mathbf{F}$. Note that here we obtain a continuous approximation of F for the optimization steps in line 12-13 of Algorithm 1 without requiring to know the number of cluster K in the optimiztion process.

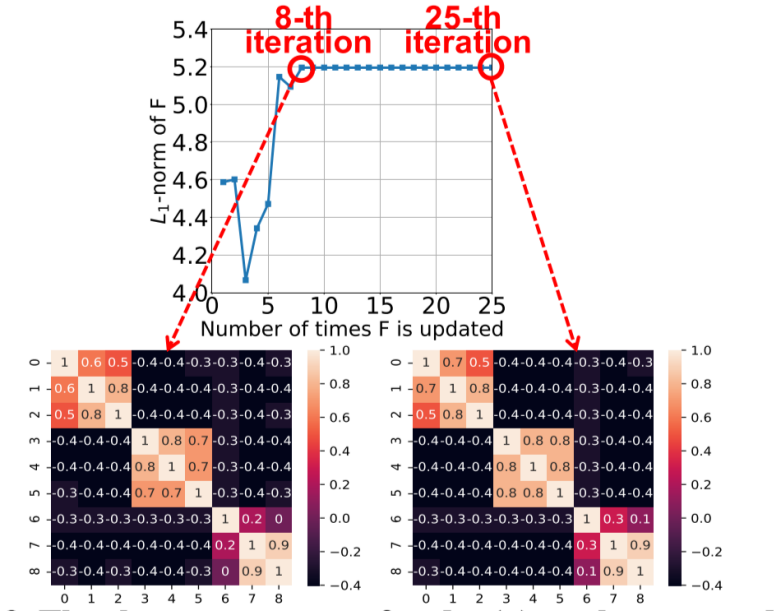
[14] Chris Ding and Xiaofeng He. 2004. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*. 29.

Optimizing Communication Performance

In traditional FL systems:

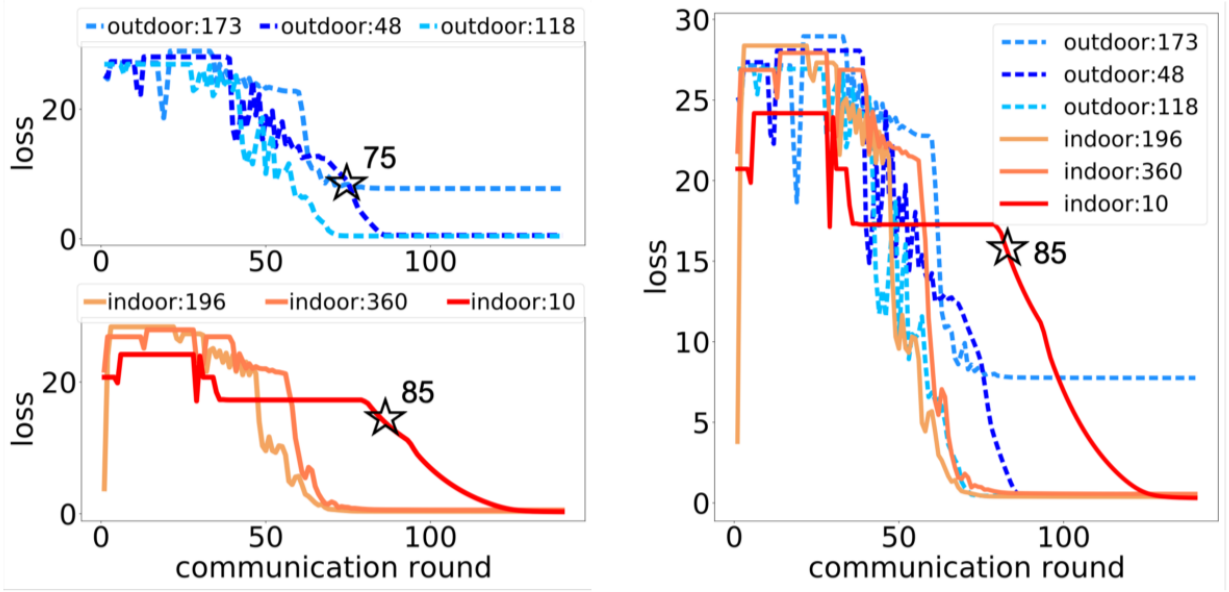
- Some focuses on the model compression and quantization techniques
- Other approaches chooses stragglers (the nodes who converge slower) from all nodes and drop them simultaneously, which does not consider the differences among the stragglers.

Leverage the inherent cluster relationship learned by ClusterFL to dynamically drop nodes during the FL process.



Cluster-wise Straggler Dropout

This example suggests that it's more flexible and efficient to identify stragglers within clusters.



(a) Within clusters (b) Among all nodes

Figure 4: Identifying stragglers within clusters (a) and among all nodes (b). Our cluster-wise straggler dropout can dynamically identify and drop stragglers within each cluster. "indoor:196" means the node has 196 data samples from the environment "indoor".

Measure the Convergence

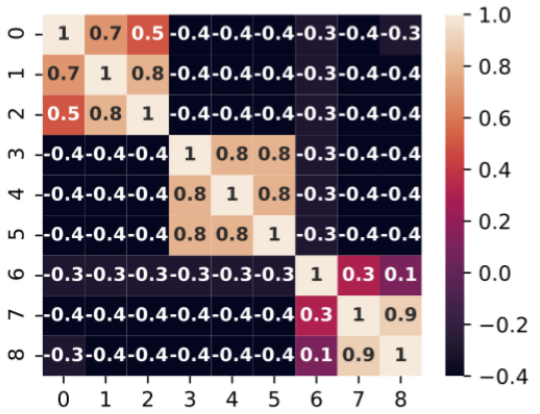
For cluster j ($j = 1, \ldots, K$) which contains N_j nodes, we define the averaged convergence rate γ_q for node q ($q = 1, \ldots, N_j$) in the cluster during the latest T_c iterations as follows:

$$\gamma_q = \frac{1}{T_c} \sum_{t=1}^{T_c} \varepsilon_q^t \quad for\ q = 1, 2, \ldots, N_j$$

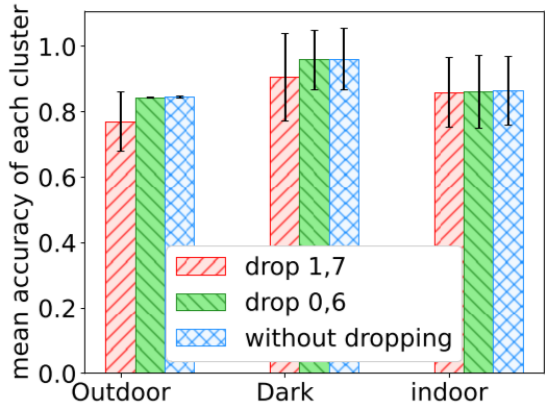
$$\varepsilon_q^t = \frac{|loss_q(t) - loss_q(t-1)|}{\sum_{q=1}^{N_j} |loss_q(t) - loss_q(t-1)|}$$

Here ε_q^t is the normalized loss update for node q compared to other nodes in cluster j at t-th iteration; γ_q is the mean value of ε_q^t in the latest T_c iterations; T_c is the threshold of iteration for determining stragglers, where a larger T_c means obtaining a more smooth loss update to filter out interference while also putting off the iteration to recognize stragglers.

Correlation-based Node Selection



(a) Correlation matrix of nodes



(b) Mean accuracy of each cluster

Figure 5: Effectiveness of correlation-based dropout. Left: The three block denotes the cluster "outdoor", "dark", "indoor", respectively. Compared to node 1 and node 7, node 0 and node 6 are less related to other nodes in the clusters. Right: The accuracy performance after dropping the less related nodes suffers minor degradation.

Specifically, the server will compute an importance vector σ for all nodes using the correlation matrix of **F** to measure their degree of correlation with other nodes in each cluster. Suppose node q is in the cluster j which contains N_j nodes, we define the importance metric σ_q for node q as follows:

$$\sigma_q = \frac{1}{N_j} \sum_{p=1}^{N_j} \mathbf{R}_{pq} \quad \text{for node } q \text{ in cluster } j$$

Correlation-based Node Selection

As shown in Figure 5, a node with a larger σ_q (e.g., node 1 with $\sigma_1 = 0.833$ while node 0 with $\sigma_0 = 0.733$) means it has a higher level of correlation with other nodes in the same cluster. The server will then order σ_q of all clusters from large to small and drop the last M_d nodes at iteration T_{thresh}. In this way, the total number of nodes communicating with the server will be reduced to $M - M_d$. The dropped-out nodes will then locally train their models based on the last update and do not communicate with the server. Finally, the total communication time will be reduced since the number of nodes interacting with the server is smaller, while relatively more important nodes are kept in the learning process to improve the overall performance.

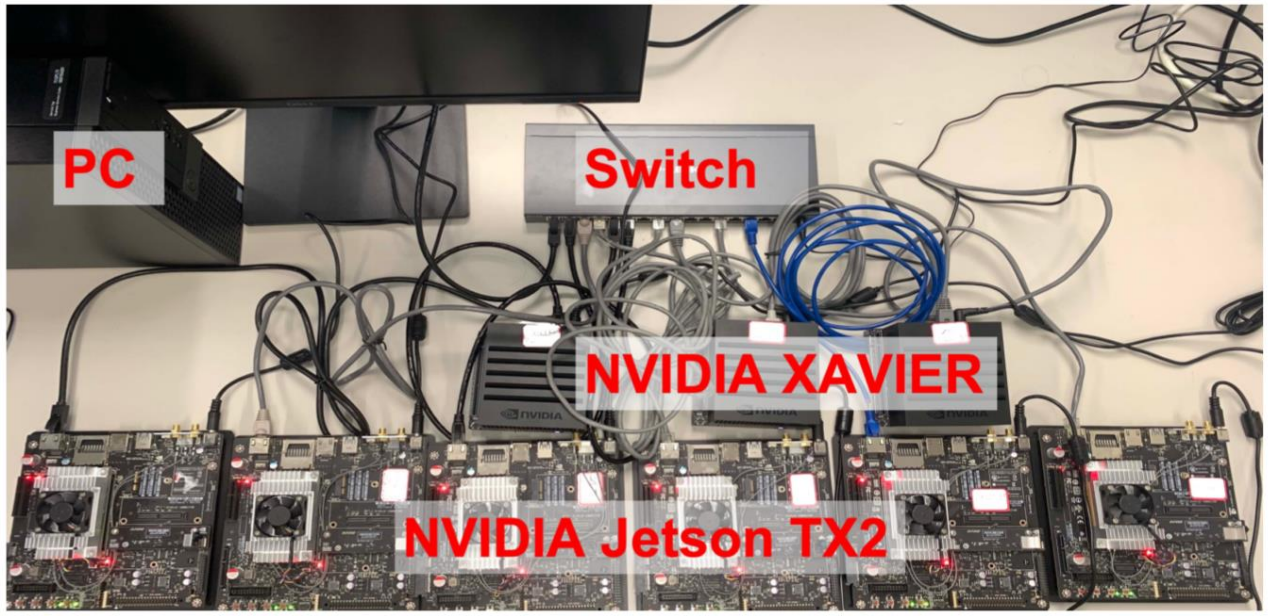
Data Collection

Application	Task	Data Dimension	Number of Subjects	Number of Data Records	Sensor	Environment
Human Movement Detection using UWB	with/without Human Movement	55	8	663	Decawave DWM1000 UWB	node 0,1 from parking lot node 2,3,4 from corridor node 5,6,7 from room
Walking Activity Recognition using IMU	walking on corridors/ upstairs/downstairs	900	7	1369	LPMS-B2 IMU	node 0,1,2,3 from building 1 node 4,5,6 from building 2
Gesture Recognition using Depth Camera	good/ok/ victory/stop/fist	1296	9	7422	PicoZense DCAM710	node 0,1,2 from “outdoor" node 3,4,5 from “dark" node 6,7,8 from “indoor"
HARBox: ADL Recognition using Smartphones	walking/hopping/ phone calls/waving/typing	900	121	32935	77 different smartphone models	121 subjects (17-55 years old) Sampling rate: 43.5-57.5Hz

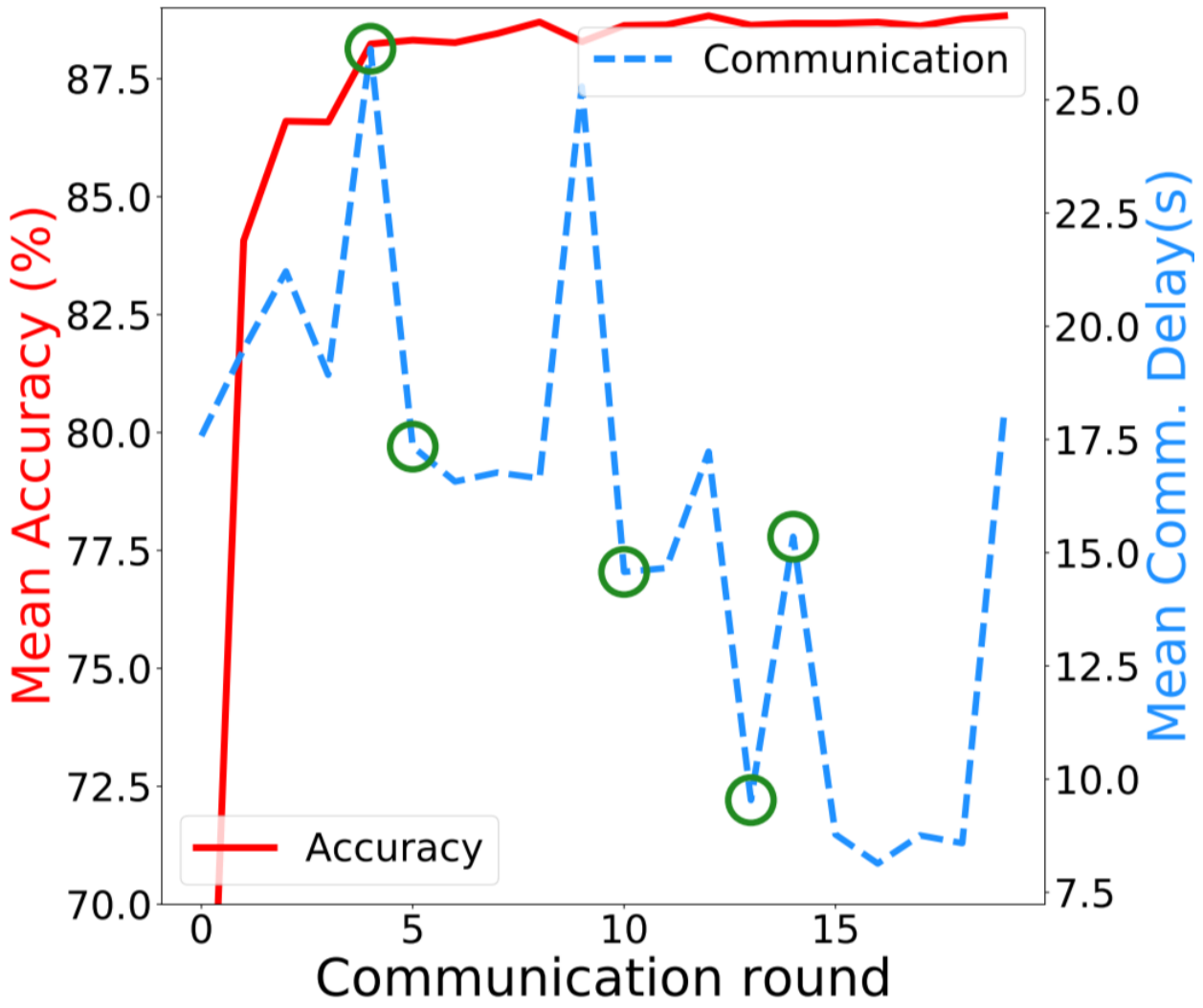
Table 3: Four new HAR datasets (UWB, IMU, Depth, HARBox) collected in real-world experiments

Implementation and Evaluation

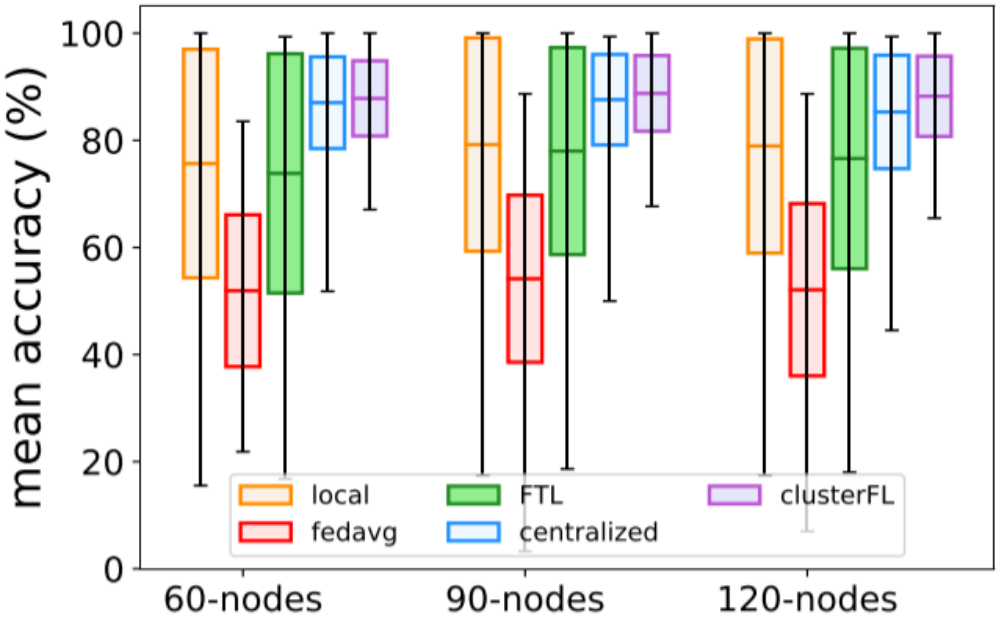
- Server
  - PC
    - Inter Core i7-9700 CPU 3.0GHz x 8
    - Ubuntu 18.04.5
- Node
  - NVIDIA Jetson TX2 x 7
    - 6 core ARM CPU
    - Ubuntu 18.04.5
  - NVIDIA Jetson AGX Xavier x 3
    - 8 core ARM CPU
    - Ubuntu 18.04.5
- TP-link TL-SG2016k
- Python3



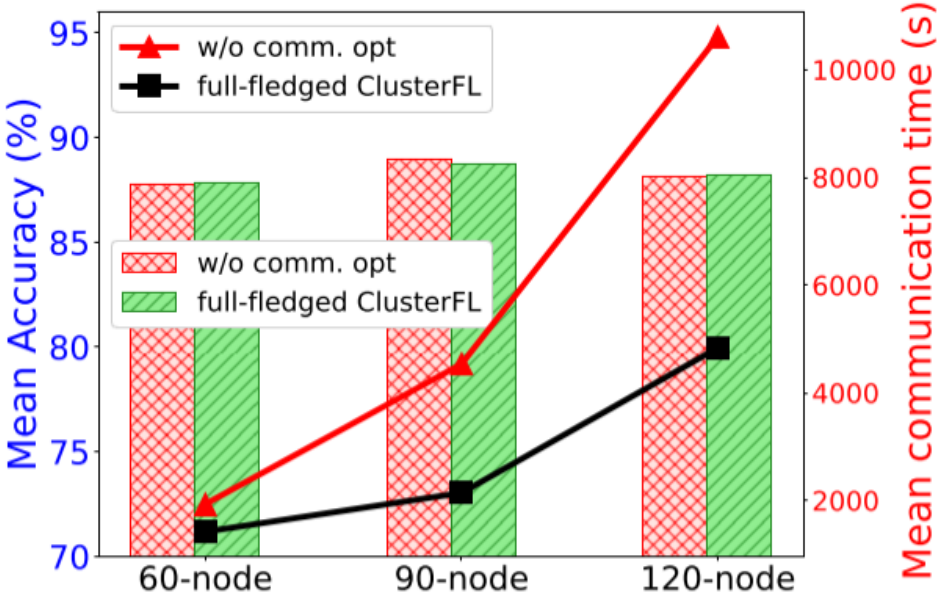
Impact of Dynamic Network Conditions



Scalability

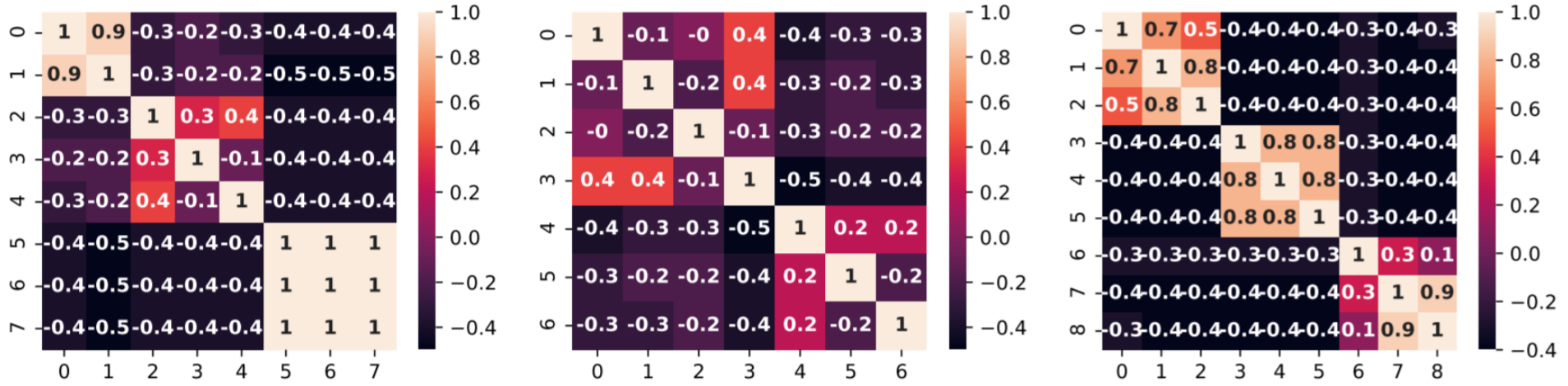


(a) Mean accuracy



(b) Mean commu. cost per node

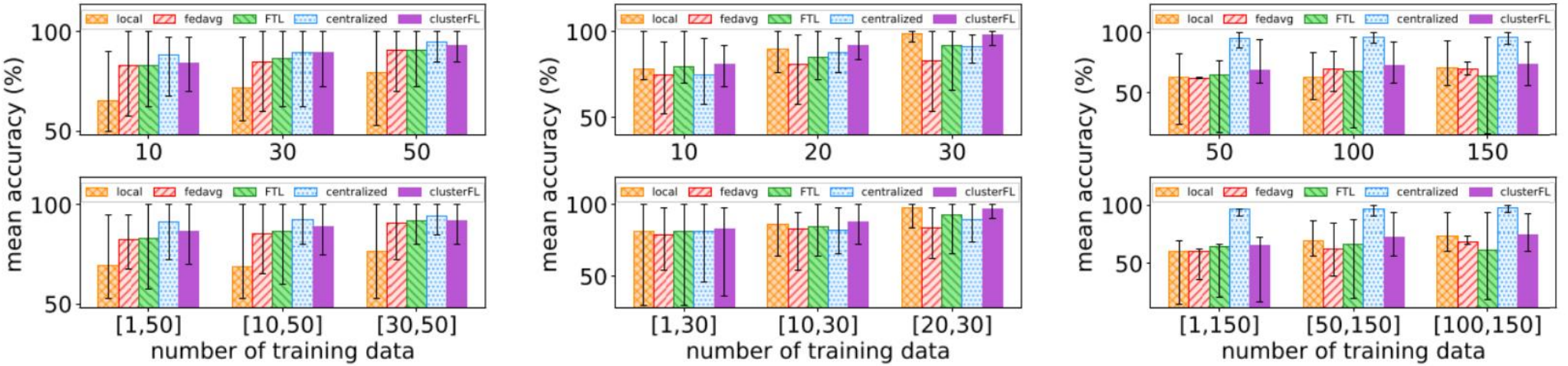
Performance on Different Datasets



(a) UWB structure **(b) IMU structure** **(c) Depth structure**

Figure 14: Learned relationship among nodes for UWB, IMU, Depth dataset, respectively.

Comparison of Different Methods



(a) UWB Accuracy Performance (b) IMU Accuracy Performance (c) Depth Accuracy Performance

Figure 12: Comparison of Accuracy Performance for UWB, IMU, Depth dataset, respectively. ClusterFL outperforms local/FedAvg/FTL in various settings and even performs better than centralized learning for IMU dataset.

Comparison of Different Methods

	Local	FedAvg	FTL	Centralized	ClusterFL
UWB	72.19%	86.25%	86.98%	**90.83 %**	89.06%
IMU	88.86%	79.52%	85.42%	84.48%	**90.47%**
Depth	65.81%	67.52%	65.69%	**96.29 %**	71.82%

Table 4: Mean accuracy in balanced data settings

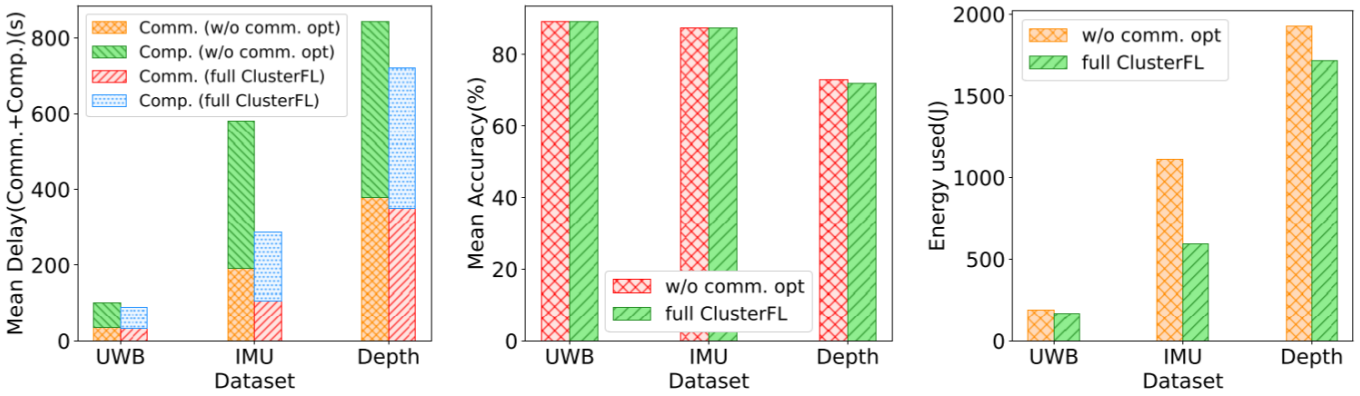
	Local	FedAvg	FTL	Centralized	ClusterFL
UWB	71.67%	86.25%	87.30%	**92.71%**	**92.71%**
IMU	88.29%	82.00%	86.20%	84.19%	**89.05%**
Depth	67.91 %	63.75%	63.86%	**96.82%**	70.68%

Table 5: Mean accuracy in unbalanced data settings

System Overhead

Dataset	number of nodes	number of data records in nodes	configuration of nodes	dropped nodes
UWB	8	[22, 25, 10, 13, 13, 17, 19, 29]	node 0,1,2 on Xavier node 3,4,5,6,7 on Tx2	node 3 and 4
IMU	7	[10, 13, 13, 49, 19, 29, 31]	node 0,1,2 on Xavier node 3,4,5,6 on Tx2	node 3 and 6
Depth	9	[94, 97, 114, 117, 117, 59, 133, 71, 86]	node 0,1,2 on Xavier node 3,4,5,6,7,8 on Tx2	node 5,7,8

Table 6: Setup of nodes for three datasets



(a) Time delay **(b) Accuracy** **(c) Energy**

Figure 15: Performance with or without communication optimization of ClusterFL for UWB, IMU, Depth dataset. w/o comm. opt means without communication optimization.

Performance with Different Model Depths

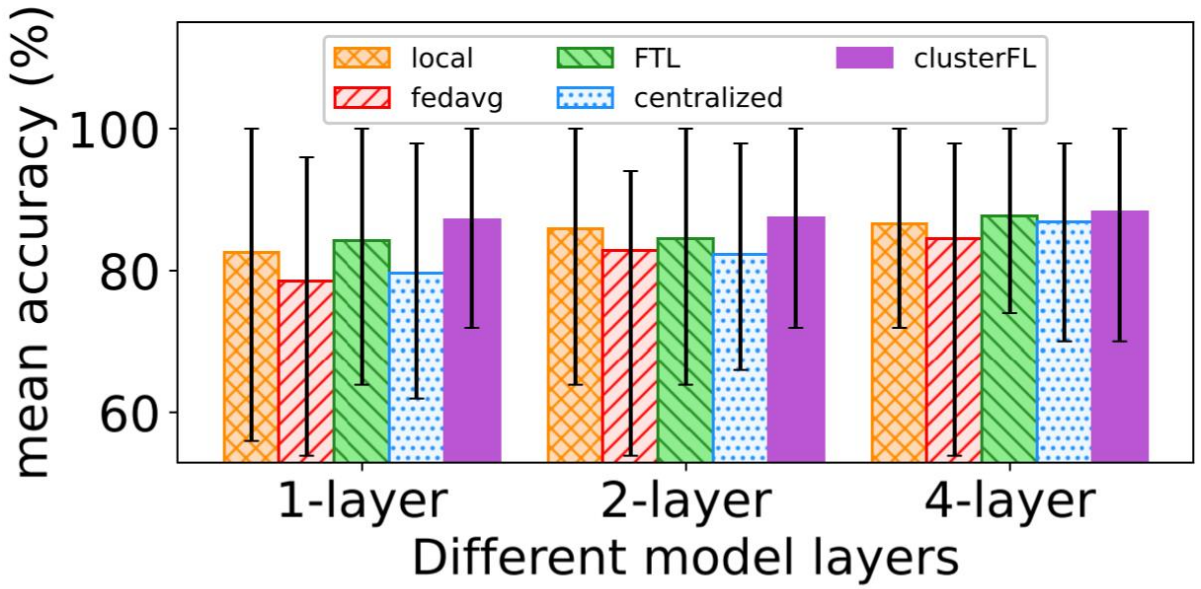


Figure 16: The accuracy of IMU dataset for models with different depths.