IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 February 2023, accepted 16 March 2023, date of publication 24 March 2023, date of current version 30 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3261266

APPLIED RESEARCH

FedNets: Federated Learning on Edge Devices Using Ensembles of Pruned Deep Neural Networks

BESHER ALHALABI[1], **SHADI BASURRA**[1], **AND MOHAMED MEDHAT GABER**[1,2]

[1]Department of Computing & Data Science, Birmingham City University, B4 7XG Birmingham, U.K.
[2]Faculty of Computer Science and Engineering, Galala University, Suez, Egypt

Corresponding author: Mohamed Medhat Gaber (mohamed.gaber@bcu.ac.uk)

ABSTRACT Federated Learning (FL) is an innovative area of machine learning that enables different clients to collaboratively generate a shared model while preserving their data privacy. In a typical FL setting, a central model is updated by aggregating the clients' parameters of the respective artificial neural network. The aggregated parameters are then sent back to the clients. However, two main challenges are associated with the central aggregation approach. Firstly, most state-of-the-art strategies are not optimised to operate in the presence of certain types of non-iid (not independent and identically distributed) applications and datasets. Secondly, federated learning is vulnerable to various privacy and security concerns related to model inversion attacks, which can be used to access sensitive information from the training data. To address these issues, we propose a novel federated learning strategy FedNets based on ensemble learning. Instead of sharing the parameters of the clients over the network to update a single global model, our approach allows clients to have ensembles of diverse-lightweight models and collaborate by sharing ensemble members. FedNets utilises graph embedding theory to reduce the complexity of running Deep Neural Networks (DNNs) on resource-limited devices. Each Deep Neural Network (DNN) is treated as a graph, from which respective graph embeddings are generated and clustered to determine which part of the DNN should be shared with other clients. Our approach outperforms state-of-the-art FL algorithms such as Federated Averaging (Fed-Avg) and Adaptive Federated Optimisation (Fed-Yogi) in terms of accuracy; on the Federated CIFAR100 dataset (non-iid), FedNets demonstrates a remarkable 63% and 92% improvement in accuracy, respectively. Furthermore, FedNets does not compromise the client's privacy, as it is safeguarded by the design of the method.

INDEX TERMS Federated learning, ensemble learning, convolutional neural networks, graph embedding, affinity propagation, non-IID datasets, privacy.

I. INTRODUCTION

With the proliferation of the Internet of Things (IoT), and the launch of 5G networks, the IoT has emerged as one of the major technological advances in our lives. We can see their advances in different domains, including wearable smart health devices [1], intelligent energy networks, smart transportation [2] and smart building [3], [4]. These tiny connected devices generate massive amounts of data on the network edge, giving great opportunities to generate valuable insights and complete sophisticated machine learning (ML) tasks. The traditional approach to analysing IoT data is to transfer user data (clients) to a central cloud server. Then the server completes the analysis and generates the required insights [5]. However, moving sensitive information to a remote server can pose a significant risk to data privacy and lead to breaches of data protection laws such as GDPR (General Data Protection Regulation) [6]. Federated learning is a machine learning paradigm that enables the collaborative training of a model on data that is distributed across multiple devices or data centres without the need to transfer raw data to a central server [7]. In the standard federated learning setting, each device

The associate editor coordinating the review of this manuscript and approving it for publication was Jiankang Zhang.

contributes an independent and identically distributed (IID) sample of data to the model. However, most current FL strategies are not optimised to handle statistical heterogeneity, such as non-iid (non-independent and identically distributed) data generated by different clients [8]. Since different clients are likely to exhibit different behaviour (distinct usage patterns), the local training samples may follow a different distribution. As a result, the local models are likely to become vastly different; hence they could reduce the accuracy and lead to slow covariance [9]. Furthermore, different studies raised some privacy concerns related to sharing weights and biases by models. Sharing the model updates during the training process make it vulnerable to penetration, potentially causing leaks of sensitive information [10]. A number of authors have considered tackling non-iid challenges using clustering [11], [12]. Clustering is a technique that can be used in non-IID federated learning to mitigate the impact of non-IID data distribution [13]. Clustering can group devices with similar data distributions together and allow local models to be trained on similar data; the resulting local models can then be combined to obtain a more accurate global model [14], [15], however performing clustering on resource-limited environments may cause performance issues.

This paper proposes a novel holistic approach to a federated learning strategy based on ensemble learning that improves accuracy and respects privacy at the edge end. This work is the first of its kind because it looks at federated learning from a different perspective. Unlike traditional federated learning strategies, our approach allows clients to collaborate by sharing complete models, not only weights. The proposed framework is an iterative process that begins by initialising a pool of pruned deep learning models (a global pool). These models are then randomly deployed to different clients to be trained on local datasets, taking into account any discrepancies in label distribution between the local and global datasets. Subsequently, the predictions of the models are combined using ensemble learning to achieve a better generalisation performance on the local testing sets. The next step in the process is to personalise the ensemble-based federated learning by clustering the models that exhibit similar behaviour. However, clustering DNNs on resource-constrained devices can be computationally expensive. To address this issue, we employ graph embeddings theory to reduce the complexity of the DNNs. Each ANN can be represented as a graph, with nodes representing layers and vertices representing connections between layers. We generate embeddings of all models and then cluster them. Finally, we select representatives of the resulting embedding clusters and ask the clients to share the corresponding models to be part of the global pool, thus initiating a new iteration.

Our key contributions could be summarised as follows:

- introducing a new federated learning strategy under non-iid settings; the proposed approach employs deep-ensemble learning to maximise the generalisation at the edge-end and provide better performance on different distributions of the clients' local data;
- proposing a novel ensemble pruning technique to reduce communication overheads over the network. It aims to minimise the storage footprint for ensembles by applying affinity propagation clustering. The clustering is applied to the embeddings of the models, considering that a graph could represent each artificial neural network, and
- establishing a new privacy approach to preserve clients' sensitive data in the applications that require running an ensemble of models.

The paper is organised as follows: Section II introduces related work in federated learning in non-iid settings. Section III describes the proposed approach in detail. The experimental study and the results are discussed in Section IV. Finally, the conclusion and future works are presented in Section IV-D.

II. RELATED WORK

Different works have been proposed to explore the high variance of federated learning algorithms in the presence of non-IID datasets (data heterogeneity) [16], [17], [18]. In the following section, we provide a deeper understanding of the effect of data heterogeneity on federated learning strategies. Subsequently, we list some popular techniques to address statistical heterogeneity in federated learning.

A. THE EFFECT OF NON-INDEPENDENTLY AND IDENTICALLY DISTRIBUTED (NON-IID DATA) IN FEDERATED LEARNING

FedAvg is one of the first central aggregation strategies that orchestrates the distributed federated learning process [19]. FedAvg employs SGD (Stochastic Gradient Decent) to optimise the averaged weights from the clients. However, SGD requires IID sampling of the training data sets to generate an unbiased estimate of the full gradient [20]. In real-world applications, it's unrealistic to assume that the data on edge devices is following IID distribution. Actually, dealing with non-IID datasets is a key challenge in federated learning [21]. Due to the statistical heterogeneity between the clients' datasets, the distribution of each local dataset is different from the global distribution (drift in local updates). As a result, each client's objective is inconsistent with the global optima. Furthermore, large local updates (a large number of epochs) lead to significant differences between the averaged model and global optima, leading to low accuracy in non-IID settings [22]. 1 explains the issue of FedAvg under non-iid settings. As shown in the figure above, if a client performs different local updates, then the updated global model $w^{(t+1,0)}$ stays close to the local minimum w_1* rather than straying toward the true global minimum x^*.

B. APPROACHES TO DEAL WITH NON-IID DATA IN FEDERATED LEARNING

FedAvg suffers when the heterogeneity of the data is different between clients [23] (non-IID datasets). This happens because the distribution of each local dataset is very different

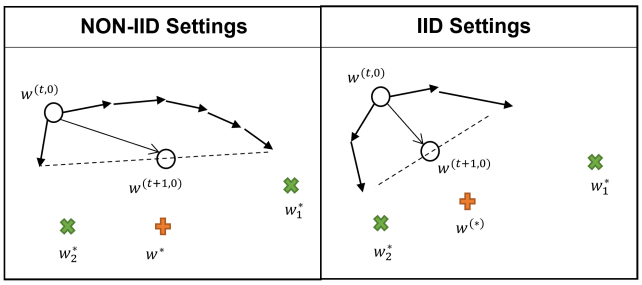

FIGURE 1. Model updates in the parameter space. Orange and green refer to the minima of global and local objectives, respectively.

from the global distribution. Different works have been proposed to optimise FL on non-IID datasets; we divide them into different categories and summarise the most recent ones:

1) DATA-BASED APPROACHES

Data-based approaches could be divided into two main categories, namely data sharing and data augmentation. **Data sharing** is a simple yet effective approach to tackle statistical heterogeneity in non-iid settings. Reference [9] proposes to create a subset of the data to be globally shared between the clients, hence generalising the learning task. The experiments on the CIFAR10 dataset show that the accuracy could be increased up to 30% while globally sharing only 5% of the dataset. Similarly, [24] developed a mechanism to select a global subset of the client's data to be used in a federated learning task which is popular in some domains like the health sector. There are notable deficiencies in the data-sharing approaches. Initially, obtaining a uniformly distributed global dataset is challenging due to the server's lack of knowledge regarding data distributions among connected clients. Additionally, distributing segments of the global dataset to each client for model training goes against the fundamental motivation of privacy-preserving learning, which is a requirement for this process.

Data augmentation methods are a set of techniques to increase the number of data samples by applying different transformations. Its mainly used to mitigate the issues of using imbalanced datasets in ML applications [25]. Reference [26] uses data augmentation to develop a self-balancing federated learning framework that outperforms FedAvg on imbalanced EMNIST and imbalanced CINIC-10 datasets. To address the issue of Non-IID data, XorMixFL framework has been introduced by [27] which applies a data augmentation technique. The basic concept in XorMixFL is that each client shares its encoded seed samples (encoded through the XOR operator) with the server for decoding. A new balanced dataset is constructed by combining the decoded samples with the base data samples on the server. Subsequently, a global model is trained on this reconstructed data, which is then downloaded to each client until the training is complete. In [28], on the other hand, the authors suggest a mean augmented method, which involves exchanging locally averaged batch data with the server. The mean data received is then combined and transferred back to each client, reducing the degree of local data imbalance. In general, the use of data augmentation techniques can greatly enhance the learning performance of models trained on Non-IID data by replenishing the imbalanced local data with augmentations. Nevertheless, as mentioned above, many of these techniques require data sharing, which could potentially increase the risk of data privacy breaches.

2) ALGORITHM-BASED APPROACHES

Several algorithm-based approaches were proposed in the literature as follows.

a: LOCAL FINE TURNING

The authors of FedProx [21], apply some modifications to FedAvg to allow partial information aggregation. This provides convergence guarantees when learning over data from non-identical distributions. Reference [29] is another optimisation attempt to work on non-idd data. The approach uses local batch normalisation to alleviate the feature shift before averaging the clients' models. FedNova [30] is another recent framework that relies on FedAvg; it normalises and scales the local updates of the clients according to their number of local steps before updating the global model.

b: PERSONALISATION LAYERS

edge clients are given the option to have a set of personalised layers that will not be shared with the server. A popular approach that falls under this category is FedMA [31] which was originally designed to offer extra support for deep learning models, and it works by sharing the global model in a layer-wise manner. Furthermore, LG-FEDAVG [32] is a new federated learning framework that outperforms FedAvg in federated learning settings. In LG-FEDAVG, the shallow layers of the deep learning models are considered personalised layers, and the base layers of the networks are shared with the server. In contrast to LG-FEDAVG, FedPer [33] allows the shallow layers to be shared with the aggregation server. FedPer and LG-GEDAVG have resulted in good accuracy results, and they also reduce communication costs since shallow layers are lightweight when shared over the network with the aggregation server. In general, Personalisation Layers are a promising approach to enhance accuracy in non-iid settings. However, one major drawback is that the clients are not able to release the personalisation layers.

c: MULTI-TASK LEARNING (MTL) METHODS

are inductive transfer approaches that aim to improve generalisation performance by learning multiple tasks simultaneously. Reference [34] has developed MOCHA as a new framework that considers the issues of high communication cost, stragglers, and fault tolerance in distributed multi-task settings. MOCHA employs primal-dual optimisation to generate separate but related models for each client. However, primal-dual optimisation is unsuitable for non-convex problems and is limited to shallow networks.

d: TRANSFER LEARNING (KNOWLEDGE DISTILLATION)

Transfer learning allows knowledge exchange between different domains to achieve higher learning rates. Following this approach, [35] has developed FedHealth as the first federated transfer learning framework for wearable health devices. FedHealth adapts the inputs from different domains by replacing fully connected layers with an alignment layer. Similarly, [36] has developed another federated transfer learning framework for smart manufacturing with cross-domain applications (Fed-LTD). However, one of the main disadvantages of knowledge distillation is the negative transfer that can cause clients to perform worse. Negative transfer occurs when data from the source domain and the task contribute to reduced learning performance in the target domain [37]. **Client clustering:** In the literature, two primary types of secure data similarity evaluation methods have been introduced to address this issue. One method involves evaluating the similarity of the loss value, while the other main goal is to evaluate the similarity of model weights. The first similarity evaluation approach, as reported in [38], [39], [40], involves comparing the loss values of various cluster models. The fundamental concept behind this technique is simple: instead of creating a single global model, the server produces multiple global models and distributes all cluster models to connected clients for local empirical loss computation. Each client then updates the received cluster model with the lowest loss value and transmits it back to the server for cluster model aggregation. The second approach involves assessing the similarity of local data and clustering based on the local model weights. In [41], [42] FedAvg is employed first to train and warm up the global model, and then the model is downloaded locally to each client for local training. Then the models are sent back to the server to be clustered based on the weights. Client clustering is both necessary and justifiable to tackle non-iid challenges, and this is because merging local models trained on vastly different data can lead to negative knowledge transfer; hence, the overall performance of the shared model will decline. Furthermore, creating multiple global models instead of a single one improves the scalability and flexibility of FL systems, enabling system developers to select or combine different cluster models to suit specific tasks. Nevertheless, this method requires additional computation and communication resources for model training and testing. **Ensemble learning:** In Fed-ensemble [43], the authors leverage ensemble learning to bring greater generalisation power to Federated Learning (FL). Fed-ensemble utilises random permutation to update a group of models and then produces the prediction through model averaging. By doing so, Fed-ensemble is able to achieve improved performance and accuracy compared to traditional FL methods.

e: GRAPH REPRESENTATION LEARNING

most recently, graph representation has become a prominent topic in the ML community due to its wide applications. Different works have been proposed to use graph learning in FL. GraphFL [44] is specifically designed to address the challenge of non-iid using a semi-supervised node classification approach based on graphs. First, GraphFL follows the training scheme of MAML (Model Agnostic Meta-Learning) to learn a global model on a server. Then, it leverages the traditional FL methods (e.g. FedAvg) to further improve generalisation on training sets. FedCG [45] is another framework to address the statistical heterogeneity in FL by means of GCN (Graph Convolutional Networks). FedCG consists of three major steps: a-identify the clusters that share the same data distributions; b-assign network components to the formed clusters; c-interaction with the GCN. References [46] and [47] preserve privacy in FL using similarity-based graph neural networks. Unlike traditional FL approaches to tackle the statistical heterogeneity in non-iid settings, our approach employs graph theory and ensemble learning aiming to reduce communication overhead over the network while preserving privacy. In the proposed approach, we use deep ensemble learning to address statistical heterogeneity in real-world applications of FL. Our solution allows resource-constrained devices to run deep ensembles smoothly. Second, we adopt graph theory and, specifically, graph embeddings to develop a novel clustering mechanism.

III. METHOD

In this section, we describe the proposed approach by providing a summary of the entire process and then explain each step in detail.

The proposed framework is an iterative process; it starts by initialising a pool of pruned deep learning models (a global pool θ_0). Then, the members of the proposed pool are randomly deployed to different clients to be trained on local datasets (assuming the label distribution of the local datasets does not match the global one). After that, the predictions of the models are combined using ensemble learning to obtain a better generalisation performance on the local testing sets. The next step in the process aims to personalise the ensemble-based federated learning by clustering the models that exhibit similar behaviour. However, clustering DNNs on devices that are resource-constrained is expensive. Thus, we employ graph embeddings theory to reduce the complexity of the DNNs. Since each ANN (Artificial Neural Network) could be represented as a graph (nodes are layers, vertices are connections of layers), we generate embeddings of all models, and then we cluster them. Finally, we choose representatives of the resulting embedding clusters and then ask the clients to share the corresponding models to be part of the global pool θ_0 and a new iteration begins.

The following is a detailed description of the steps of *FedNets* and the overall system topology.

A. SYSTEM TOPOLOGY

As many federated learning strategies, *FetNets* follows the star network communication topology where a central cloud server is connected to a network of resource-limited devices. The server orchestrates the learning process by aggregating deep learning ensembles from all connected clients

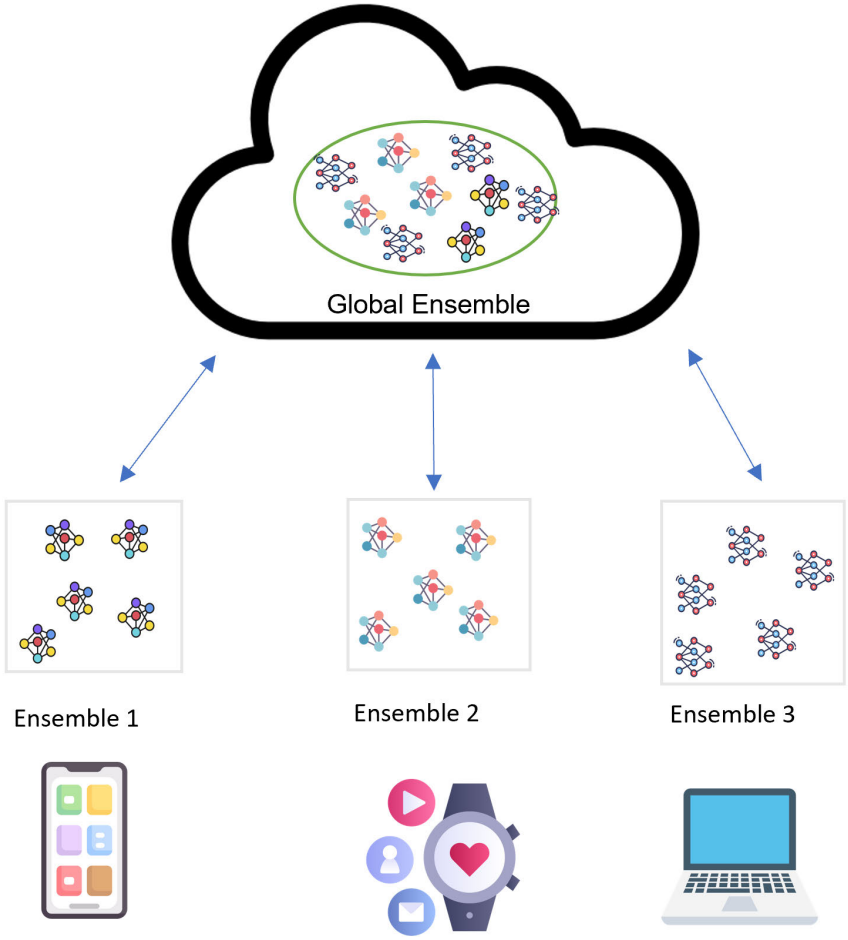

FIGURE 2. ***FedNets*** **follows a star communication topology where a server connects with all the remote clients. The server orchestrates the communications in each learning round; it starts by deploying the global ensemble to the clients. Then, each client shares a few members of their ensembles with the server.**

(each communication round). Figure 2 summarises the main communication topology.

B. ENSEMBLE GENERATION AND PRUNING

In non-iid federated learning settings, the statistical heterogeneity of clients can vary significantly [8]. This led to a different distribution of the local data on each client [48]; hence, traditional federated learning settings experience a drop in accuracy [49]. To overcome those challenges, our approach utilises deep ensemble learning to give more generalisation power at the edge end and boost performance. Typically, any pruning process has different hyperparameters, and it could be a challenging task to set the optimal values of these parameters. Thus, different optimisation techniques could be used [50], [51], [52], [53]. However, due to the high complexity of DNNs, we aim to prune the models before running them at the edge end. To complete the pruning and the deployment of models, we follow a similar approach to the one described here [54]. In this work, the authors proposed a multi-phase pruning framework that enables edge devices to run deep ensemble learning without draining the resources of devices. However, our approach has two main differences: (1) we use Constant Sparsity pruning instead of weight pruning; (2) integer quantisation is ignored as we do not run the experiments on real devices. At the end of this step, we have a group of N clients; each client has a group of M different models.

C. MODEL TRAINING ON LOCAL DATASETS (NON-IIDs) UPDATE LOCAL MODELS WEIGHTS

As all federated learning strategies, *FedNets* requires the local models (ensemble members) to be trained on the datasets of each client. In the training process, the models will simply learn the good values for all the weights and the bias from the labelled examples. However, *FedNets* employs deep learning ensembles. Thus, the training accuracy will depend on all

members. To calculate the accuracy of an ensemble: Let $En = m_1, \cdots, m_N$ be an ensemble of the deployed models m_i. The prediction of the ensemble for a training/testing example (x is the data feature, y is the data label) using max-voting is the class that receives the maximum support $\eta_{final}(W)$ from all members of the ensemble. Based on that, the output of the ensemble could be defined as:

$$\eta_{final}(W) = argmax_{j\in\{1,\cdots,C\}} \sum_{i=1}^{N} y_{i,j}$$

D. GRAPH CONVERSION

Graph clustering based on embedding is a popular technique that aims to convert graph structure and node attributes into the low dimensional feature and split similar nodes into non-overlapping groups [55], [56]. The purpose of this step is to cluster models with similar properties using the graph topology and node features. To convert an artificial neural network to a graph, let us have a k layer neural network; each layer has a set of weights $W = w_1, w_2, \cdots, w_n$, the output of layer k is $z = W \times x + b$.

A graph $G = (V, E)$ is defined as a collection of vertices $V = \{v_1, \ldots, v_n\}$ where $v_i \in En$, edges $E = \{e_{ij}\}_{i,j=1}^{n}$ where e_i is a connection between l_i and l_{i+1} assuming that k is a layer in m_i, and $X_i = \{(w_1, b_1), \cdots, (w_n, b_n)\}$ is the corresponding node vector that holds the average weights $\sum_{i=1}^{k} avg(W_k)$ and biases $Avg(\sum_{i=1}^{k}(b))$ in each layer k.

Figure 3 shows how *FedNets* converts an artificial neural network into a graph.

E. GRAPH EMBEDDING GENERATION

The representation of a node v_i according to [57] can be calculated as:

$$\mu_i^k = MLP_{W_k}^k \left((1 + \epsilon^k) \cdot \mu_i^{k-1} + \sum_{j\in N(i)} \mu_j^{k-1} \right) \quad (1)$$

where μ_i is the representation of node v_i, $N(i)$ is the neighbourhood of node V_i, ϵ could be learnt by a hyperparameter or GD (gradient descent) and $MLP_{W_k}^k$ refers to multilayer perceptron for the k^{th} GIN layer and weights W_k. After generating the node embeddings, we calculate the overall graph embeddings following the same approach described in [57]:

$$\boldsymbol{h}_{\mathcal{G}} = \mathrm{MLP}_{\boldsymbol{W}} \left(\|_{k=1}^{K} \mathrm{ATT}_{\boldsymbol{\Theta}(k)} \left(\boldsymbol{U}_{\mathcal{G}} \right) \right) \quad (2)$$

The embedding of the input graph of graph g is $U_{\mathcal{G}} \in \mathbb{R}^{N\times D}$ where the $n-th$ row, $u_n \in \mathbb{R}^D$. The output of this step is an array of the corresponding graph embeddings for each model Em_i.

F. CLUSTERING OF EMBEDDINGS

At this stage, all models $m_i \in \theta_0$ are transformed into a two-dimensional array Em_i. We apply affinity clustering on Em_i, and then we choose representatives from each cluster. The selection criteria will be based on the following: Let c_i be one of the generated clusters, cl is the length of the cluster (number of members of the group) and acc_i be the precision of the corresponding model m_i on a validation set VS (10% of the training set). The chosen model should satisfy the following:

$$\begin{cases} m_i \in C_i \text{ where : } cl > K \quad where \quad K = 1, \cdots, 20 \\ acc_i > 55\% \text{ on } \quad \mathbf{VS} \end{cases} \quad (3)$$

The pseudocode of *FedNets* is shown in Algorithm 1.

Algorithm 1 *FedNets* Pseudocode

Server
Initialise global pool θ_0
for Each communication Round R, r $\leftarrow$ 1 to T **do**
  Select N Client
  for Each client $i = 1, 2, .., N$ **do**
    Download θ_r
    $Client_i$ update and receive em_i
  end for
  $GE \leftarrow AffinityPropagation(EMs)$
  $M \leftarrow RepresentativeSelection(GE)$
  Update Global Pool θ_0 with M
end for

Client Update
Replace local ensemble $\theta_i \leftarrow \theta_{i+1}$
for Local Epoch $e \leftarrow 1$ to E **do**

$$\eta_{final}(W) = argmax_{j\in\{1,\cdots,C\}} \sum_{i=1}^{N} y_{i,j}$$

end for
for Each model M in θ_i **do**
  $em \leftarrow generatedEmbeddings(M)$ return em
end for

G. PRIVACY PRESERVING

Unlike the typical federated learning approaches, the proposed method provides a privacy-preserving-by-design approach. The sharing of a subset of the local ensemble per client makes it harder to reveal the behaviour of the users of individual clients. Formally, if the number of models making up the ensemble in a client is n, only m models are shared, where $m < n$. Thus, privacy increases when the value $\tau = \frac{m}{n}$ decreases. Note that τ can be a hyperparameter when applying the proposed method. An important factor in increasing privacy is the diversity of the ensemble. The more diverse the ensemble is the more private the proposed method is. In other words, with the same value for τ, the diverse ensemble is inherently more private.

For example, if $n = 10$ and $m = 3$, then $\tau = 0.3$. This is a setting that will result in high privacy. However, if $\tau = 0.9$, then privacy may be compromised because most of the models that make up the ensemble are shared centrally, making it possible to reproduce the data fed to the model locally. It is

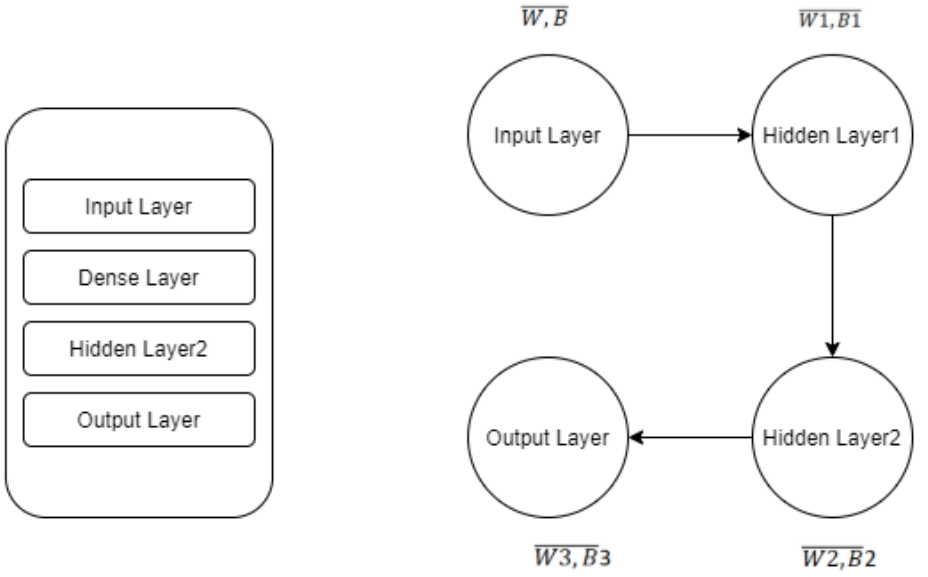

FIGURE 3. ***FedNets*** **approach to convert an ANN to a graph. Each layer will be converted to a node, and the mean value of the weights/biases will be added as a feature to the node.**

worth pointing out that in a typical federated learning setting, the parameters of the model are shared centrally (i.e. $\tau = 1$), making the models shared by the clients vulnerable.

IV. EXPERIMENTAL STUDY

In this section, we first explain in detail our simulation environment and setup, then evaluate the precision of *Fed-Nets* using the ResNetV2 and CIFAR100 federated dataset. Next, we compare our results with *Fed-Avg*. Finally, we provide important measures related to ensemble performance, such as inference time, required training time, and the number of models per client. Those measures are essential to give an idea about the feasibility of running deep ensembles on resource-limited devices (IoT). The code used in the experiment is publicly available on GitHub.[1]

A. DATA SET AND MODELS

1) FEDERATED CIFAR100 FOR SIMULATION

This dataset is specially designed to simulate non-independent and identically distributed data samples. It is derived from the original CIFAR100 dataset, and it has 50,000 training samples and 10,000 testing samples. Unlike the original dataset, the training and testing samples are partitioned across 500 and 100 clients (respectively, and no overlapping across the clients). The training clients' IDs range from 0 to 499, while the testing clients' IDs go from 0 to 99. The data partitioning part is done using PAM (Pachinko Allocation Method) [58], which is an improved version of LDA (Latent Dirichlet Allocation). This approach uses a two-stage LDA process, where each client has an associated multinomial distribution over the coarse labels of CIFAR-100, and a coarse-to-fine label multinomial distribution for that coarse label over the labels under that coarse label.

2) ResNetV2

This model belongs to the Deep Residual Networks (RNNs) family that achieved breakthroughs in the deep learning community. ResNetV2 is the new version of ResNet; the main improvements are related to the arrangement of the layers in residual blocks. The Model accepts an input of shape 299×299 pixels; the output is the probability distribution for the predicted classes [59].

B. SIMULATION SETUP

We run our simulation on a powerful workstation with multiple GPU and CPU nodes. The following are the hardware specifications:

- CPU Nodes:5 nodes - 72 cores per node, PowerEdge R740 Server, Intel Xeon Gold 6240 2.6G.
- GPU Nodes:2 nodes - 72 cores per node, PowerEdge R740 Server, Intel Xeon Gold 6240 2.6G, NVIDIA(R) Tesla(TM) T4 16GB Passive, Single Slot, Full Height GPU (2 cards per node) - 320 Turing Tensor cores and 2560 Cuda cores per card.

All nodes have 'CentOS-8.2.2004-x86_64' operating system installed. There are also different hyperparameters that control the design of FedNets, starting from initialising the

[1] https://github.com/besherh/FedNets

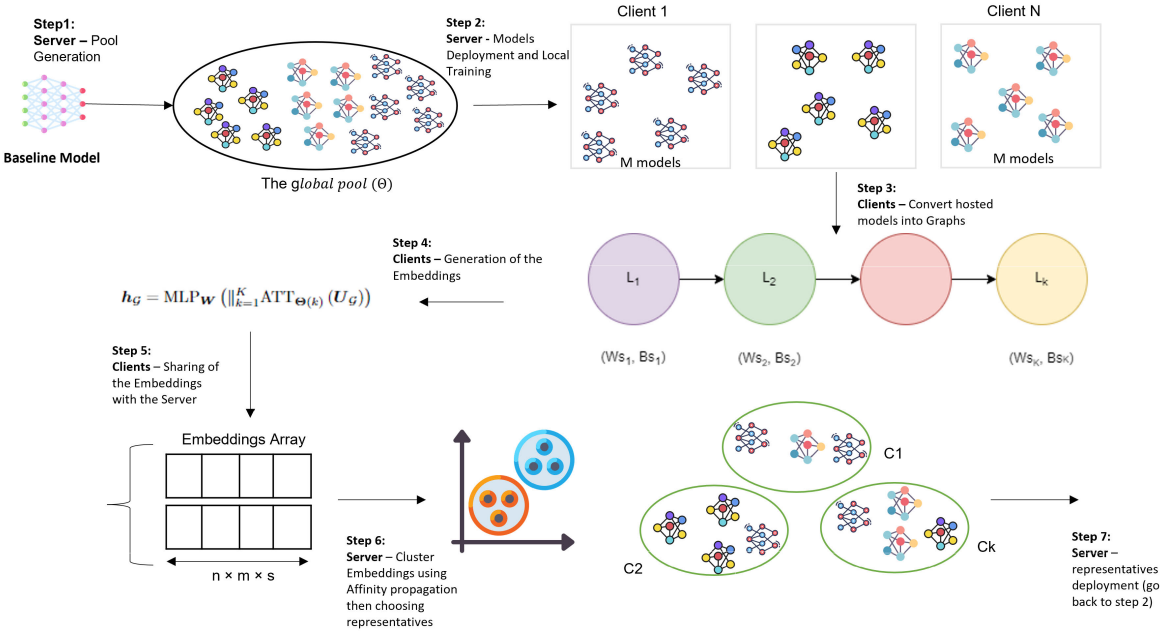

FIGURE 4. FedNets Framework involves the following steps:-Step1: server deploys deep learning ensembles to the clients and clients train them locally; step2: Each client converts the local models into graphs assuming: nodes are the layers, edges are links between layers and the attributes of the nodes are mean of weights and biases of whole layers; step3: clients generate the corresponding graph embeddings and share them with the server; step4: the server to cluster the embeddings using affinity propagation and choose a representative from each cluster, then models are deployed to clients. .

global pool and ending with the deployment of the models to the clients. Table 1 summarises all hyperparameters that are used in the different processes of FedNets.

As shown in Table 1, there are a lot of different hyperparameters that control the process flow of FedNets. Starting from the generation of the global pool and ending with representative selection. The values of the hyperparameters are selected based on a ''trial and error'' approach, and we only reported the values related to the presented results.

C. RESULTS AND ANALYSIS

We start this section by introducing details about the baseline model and the pool of models we use to deploy deep learning ensembles to clients. We move next to investigate the effectiveness of *FedNets* on non-iid settings. We use the federated CIFAR100 dataset as a benchmarking dataset and ResNetV2 as a baseline model. Additionally, we provide accuracy comparison with state-of-the-art federated learning algorithms including Fed-Avg and Fed-Yogi. The simulation is applied to a different number of clients (two clients, five clients and ten clients) for four federated learning rounds, where each client has an ensemble of ten pruned models. Finally, we provide time measures related to the performance of the deep learning ensembles on the proposed virtual clients.

The baseline of ResNetV2 (3,575 KB) is trained on the original CIFAR100 dataset, and the accuracy of the model on the testing set is 68%. We apply constant-sparsity pruning on ResNetV2 to generate a pool of 500 models. During the pruning process, we use different values of the pruning hyperparameters to ensure diversity between the pool's members. Diversity leads to better generalisation and provides better accuracy results at the edge end as shown in [60]. The maximum accuracy in the pruned pool against a validation set (20% of the testing set) is around 66%, and the minimum is 0.05%. The pruning leads to around 37% reduction of the original baseline model size(the average size of the pruned models is 1,295 KB).

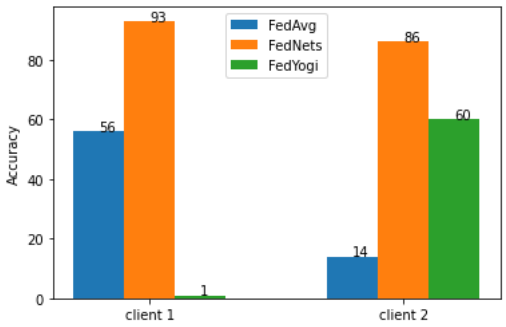

FIGURE 5. Accuracy comparison on two clients.

1) COMPARISON WITH STATE OF THE ART

Here we compare the accuracy results of *FedNets* with two of the state of the federated learning strategies (FedAvg, FedYogi) on the federated CIFAR100 dataset. In the next section, we provide the simulation results for different numbers of clients (two, five, and ten clients).

TABLE 1. FedNets hyperparameters.

Phase	Hyperparameter name	Values
Training and Pruning	Epochs	[2,3,4,5,6]
	Batch Size	[16,32,64]
	Loss	[Categorical Cross Entropy, Mean Squared Error, Mean Absolute Error]
	Optimiser	Adam
	Target Sparsity	[0.2,0.55]
	Frequency	[50,75,100]
Embeddings	Graph Classification Model-Layer Size	[32,64]
	Graph Classification Model- Activations	ReLU
	Graph Classification Model- Dropout	0
	Graph IDX - Size	(100, 2)
	Pair Model - optimiser	Adam((1e-2)
	Pair Model - Loss	MSE
Clustering	Validation Ratio	0.1
	Accuracy Threshold	0.3
	Cluster Length	10
	Sample Size	60

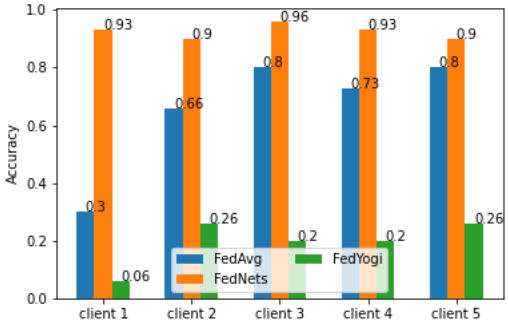

FIGURE 6. Accuracy comparison on five clients.

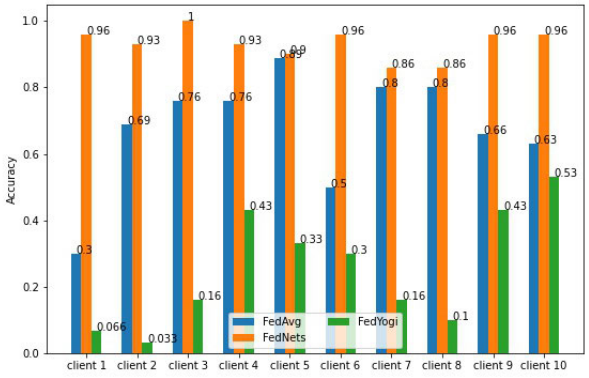

FIGURE 7. Accuracy comparison on ten clients.

As shown in 5 and when the number of clients is equal to two, *FedNets* accuracy is 93% and 86% in client1 and client2, respectively. On the other hand, the accuracy of FedAvg is 56% on client1 and 14% on client2. Similarly, the accuracy of FedYogi on client1 and client2 is 1% and 60%.

The results of the accuracy of *FedNets* on five different virtual clients are presented in figure6. From the chart, it can be seen that *FedNet*'s accuracy is better than FedAvg and FedYogi on all clients. The minimum accuracy for *FedNets* is 90% while the best accuracy reached by FedNets is 80%.

Figure7 compares the accuracy results between *FedNets*, FedAvg and FedYogi. Looking at figure7, it's apparent that *FedNets* is still achieving superior performance and can beat the state-of-the-art methods on all clients. We can also see that the maximum accuracy of *FedNets* is 100% on the client3, the maximum accuracy of FedAvg is 80% on the client7 and client 8, and the best accuracy of FedYogi is 53% on the client10.

As shown in Figures 5,6 and 7, there is a significant performance difference between *FedNets* and FedAvg/FedYogi. Our proposed approach provides better accuracy results on a different number of clients; the clients can share their knowledge by exchanging the members of the ensembles, and it seems to be better than the traditional approach of sharing the weights of the models.

2) ENSEMBLES PERFORMANCE

Before embarking on presenting and analysing the accuracy results on the federated data set, it is worth noting that as aforementioned in Section III, FedNets utilises the pruning framework reported in [54] to run deep learning ensembles on edge devices. In this work, the proposed framework is proven to produce lightweight models and provide a quick inference time. All details related to pruning effects, prediction time, temperature, and energy consumption could be found in [54].

Now, we move to present the simulation results of the required time to complete one federated learning round in*FedNets*. Each round consists of three main steps: training,

inferencing, and deployment. We run this simulation on CPU nodes for four federated learning rounds. The simulation results are presented in Table 2.

TABLE 2. Time required by the major steps of *FedNets* in seconds.

Measure	Round1	Round2	Round3	Round4
	2 Clients			
training locally	78.03	61	54.37	43.39
ensemble inference	25.36	19.95	19.22	14.27
deployment	64.87	62.37	62.03	60.16
	5 Clients			
training locally	236.69	228.82	256.42	222.21
ensemble inference	74.79	77.37	75.84	72.18
deployment	82.62	93.04	80.74	81.35
	10 Clients			
training locally	681.45	664.27	663.02	663.76
ensemble inference	186.63	175.94	178.53	117.06
deployment	120	121.18	116.41	116.38

On closer inspection of Table 2, it shows that training the deep learning ensembles on the local datasets is consuming most of the time. Training time is noticeably increased when the number of clients is ten (almost 12 minutes to complete). However, the inference time of the ensembles is acceptable in most of the applications.

It can also be seen from the data in Table 2 that, in general, *FedNets* requires a relatively long period of time to complete a federated learning round (especially when the number of clients is rather considerable, like ten). However, we should accept the fact that *FedNets* is an ensemble-based approach that aims to maximise the generalisation and accuracy at the edge end, so it requires more time to complete a federated learning round. Additionally, we are running the simulation on CPU nodes only. In a real-life application, resource-limited devices could be attached to cutting-edge AI accelerators that bring the power of TPUs to the edge. We believe that this led to a significant improvement in the time complexity of *FedNets* as shown in [54]. Turning now to examine the size of the composed ensembles on each client after each federated learning round. This could be directly related to the effectiveness of our approach in preserving the resources of IoT devices. Table 3 displays the changes in the number of models per client (ensemble size) after each federated learning round.

As shown in Table 3, *FedNets* reduces the size of the ensemble by 50% after round 4 (assuming that each client starts with ten models, as explained earlier). The simulation on both five and ten clients shows that *FedNets* is still able to reduce the number of models per ensemble which lead to reducing the required memory/storage required by the approach.

3) PRESERVING PRIVACY

As previously stated, *FedNets* prompts federated learning by allowing clients to share the members of the local ensembles; at the same time, *FedNets* respects the privacy of the clients.

TABLE 3. Changes to the number of models per client.

Number of models	Round1	Round2	Round3	Round4
	2 Clients			
Client1	7	8	7	6
Client2	6	7	7	5
	5 Clients			
Client1	8	10	7	9
Client2	5	5	9	9
Client3	9	9	9	8
Client4	7	9	7	9
Client5	7	9	7	5
	10 Clients			
Client1	10	10	8	9
Client2	9	10	9	7
Client3	10	9	10	8
Client4	9	10	8	8
Client5	10	10	9	10
Client6	10	9	10	8
Client7	9	10	10	9
Client8	9	10	10	9
Client9	10	9	9	10
Client10	10	10	8	10

Table 4 shows summary statistics about the number of shared models per client. It is worth mentioning that the value of τ is not controlled during the simulation. However, as explained earlier, τ could be a hyperparameter to the proposed approach which will be used to trade off accuracy with privacy by controlling the maximum number of models to be shared with the other clients. As seen in the table, when the number of models is equal to ten, *FedNets* tends to share a large number of models (average 9.6, $\tau = 0.96$). This could be minimised by defining $\tau \leq 0.5$ sharing less than half of the models per client, forcing greater privacy. However, this may come at the cost of compromising the accuracy of the federation.

4) DISCUSSION OF THE RESULTS

The results of FedNets as shown in Figures 5,6,7 indicate that the proposed approach is effective in providing high accuracy in non-iid settings where the distribution of class labels is vastly different among clients. This will work well for applications that cannot compromise on accuracy as the quality of the output directly impacts the reliability of the AI systems. For example, FedNets could be integrated into AI-Based Medical Diagnosis systems to offer further data privacy assurance to comply with Health Institutions' Data Protection Policies. It is possible that the results in Table 4 could be improved by adding the privacy factor τ into the list of hyperparameters introduced in Table 1. This could be useful when edge clients deal with very sensitive personal information, for example, smart wearable devices. The results obtained from Table 2 shows that FedNets could

run on resource-limited environments effectively. However, the required time to complete one federated round could take a reasonably long time. The observed increase in time could be attributed to: a) using a baseline model that has not been fully optimised to run on resource-limited devices; and b) utilising deep ensemble learning to provide better generalisation in non-iid settings. Using edge-friendly models like the MobilesNet family [61] could lead to shorter training and inference time. However, the fact that FedNets requires a relatively long period to complete a federated round will still be valid. This limitation renders FedNets a less desirable choice when there are no GPUs attached to the edge devices (operating solely on CPU) and when real-time applications necessitate instantaneous inference, such as autonomous vehicles.

TABLE 4. Statistics related to the numbers of shared models per round.

Metric	Round1	Round2	Round3	Round4
	5 Clients			
Minimum	5	5	7	6
Standard Deviation	1.48	2.07	1.14	1.30
Variance	1.48	2.07	1.14	1.30
Average	7.2	8.6	8.4	8.2
	10 Clients			
Minimum	7	6	8	7
Standard Deviation	0.51	0.48	0.87	1.03
Variance	0.26	0.23	0.76	1.06
Average	9.6	9.7	9.1	8.8

D. CONCLUSION AND FUTURE WORKS

In this work, we introduced *FedNets*, the first ensemble-based federated learning strategy that provides better generalisation in non-iid settings. The key differences between this approach and other federated learning strategies are: (1) clients are running deep learning ensembles rather than having one model per client; and (2) instead of sharing the models' weights to update a single global model, which is prone to a privacy breach, our approach allows clients to share models (members of their deep learning ensembles) to compose a shared pool of outperforming models, then the pool is shared with all participating clients. The experimental results on the federated CIFAR100 dataset demonstrate that our approach outperforms the Federated Learning Averaging strategy (FedAv), and Adaptive Federated Optimisation (FedYogi). The results also show that the cost of running deep learning ensembles (inference time, ensemble size) on edge devices is feasible in a resource-limited environment.

In future work, we plan to test our approach on other non-iid benchmarking datasets and to discover the effectiveness of using modern CNN architectures like MobileNet and EfficientNet when applied in deep learning ensemble settings. Moreover, we plan to utilise optimisation techniques to fine-tune the hyperparameters of FedNets in order to achieve superior results. By leveraging these techniques, we can maximise the performance of our models and unlock their full potential.

REFERENCES

[1] J. Ghosh, G. Samanta, and C. Chakraborty, "Smart health care for societies: An insight into the implantable and wearable devices for remote health monitoring," in *Green Technological Innovation for Sustainable Smart Societies*, C. Chakraborty, Ed. Cham, Switzerland: Springer, 2021, pp. 89–113.

[2] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 94–101, Feb. 2018.

[3] A. K.-F. Lui, Y.-H. Chan, and M.-F. Leung, "Modelling of destinations for data-driven pedestrian trajectory prediction in public buildings," in *Proc. IEEE Int. Conf. Big Data*, Orlando, FL, USA, Dec. 2021, pp. 1709–1717.

[4] A. K.-F. Lui, Y.-H. Chan, and M.-F. Leung, "Modelling of pedestrian movements near an amenity in walkways of public buildings," in *Proc. 8th Int. Conf. Control, Autom. Robot. (ICCAR)*, Xiamen, China, Apr. 2022, pp. 394–400.

[5] A. Ghosh and K. Grolinger, "Edge-cloud computing for IoT data analytics: Embedding intelligence in the edge with deep learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2191–2200, Jul. 2020.

[6] P. Voigt and A. V. D. Bussche, *The EU General Data Protection Regulation (GDPR)*. Cham, Switzerland: Springer, 2017.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[8] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[10] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, *arXiv:1710.06963*.

[11] W. Huang, T. Tiropanis, and G. Konstantinidis, "Federated learning-based IoT intrusion detection on non-IID data," in *Internet of Things* (Lecture Notes in Computer Science), A. González-Vidal, A. M. Abdelgawad, E. Sabir, S. Ziegler, L. Ladid, Eds. Cham, Switzerland: Springer, 2022, pp. 326–337.

[12] Y. Li, S. Wang, C.-Y. Chi, and T. Q. S. Quek, "Differentially private federated clustering over non-IID data," 2023, *arXiv:2301.00955*.

[13] P. Tian, W. Liao, W. Yu, and E. Blasch, "WSCC: A weight-similarity-based client clustering approach for non-IID federated learning," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20243–20256, Oct. 2022.

[14] J. Shu, T. Yang, X. Liao, F. Chen, Y. Xiao, K. Yang, and X. Jia, "Clustered federated multitask learning on non-IID data with enhanced privacy," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3453–3467, Feb. 2023.

[15] M. Morafah, S. Vahidian, W. Wang, and B. Lin, "FLIS: Clustered federated learning via inference similarity for non-IID data distribution," 2022, *arXiv:2208.09754*.

[16] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-IID federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10174–10183.

[17] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 1–11.

[18] L. Yang, J. Huang, W. Lin, and J. Cao, "Personalized federated learning on non-IID data via group-based meta-learning," *ACM Trans. Knowl. Discovery Data*, vol. 17, no. 4, pp. 1–20, Art. no. 49, doi: 10.1145/3558005.

[19] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.

[20] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT*, Y. Lechevallier and G. Saporta, Eds. Heidelberg, Germany: Physica-Verlag HD, 2010, pp. 177–186.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[22] Y. Li, W. Zhou, H. Wang, H. Mi, and T. M. Hospedales, "FedH2L: Federated learning with model and statistical heterogeneity," 2021, *arXiv:2101.11296*.

[23] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," 2019, *arXiv:1910.06378*.

[24] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Milan, Italy, Jan. 2021, pp. 5020–5027.

[25] T. Dao, A. Gu, A. J. Ratner, V. Smith, C. D. Sa, and C. Re, "A Kernel Theory of Modern Data Augmentation," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1528–1537.

[26] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Abu Dhabi, United Arab Emirates, Nov. 2019, pp. 246–254.

[27] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, and S.-L. Kim, "XOR mixup: Privacy-preserving data augmentation for one-shot federated learning," 2020, *arXiv:2006.05148*.

[28] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "FEDMIX: Approximation of mixup under mean augmented federated learning," 2021, *arXiv:2107.00233*.

[29] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," 2021, *arXiv:2102.07623*.

[30] J. Wang, Q. Liu, H. Liang, J. Gauri, and H. V. Poor, "A novel framework for the analysis and design of heterogeneous federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 5234–5249, 2021.

[31] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020, *arXiv:2002.06440*.

[32] P. Pu Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv:2001.01523*.

[33] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.

[34] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," 2017, *arXiv:1705.10467*.

[35] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A federated transfer learning framework for wearable healthcare," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 83–93, Jul. 2020.

[36] Y. Wang, Y. Tong, Z. Zhou, Z. Ren, Y. Xu, G. Wu, and W. Lv, "Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Washington, DC, USA, Aug. 2022, pp. 4079–4089.

[37] S. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, pp. 1345–1359, 2010.

[38] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints," 2019, *arXiv:1910.01991*.

[39] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," 2020, *arXiv:2006.04088*.

[40] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.

[41] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

[42] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.

[43] N. Shi, F. Lai, R. Al Kontar, and M. Chowdhury, "Fed-ensemble: Improving generalization through model ensembling in federated learning," 2021, *arXiv:2107.10663*.

[44] B. Wang, A. Li, H. Li, and Y. Chen, "GraphFL: A federated learning framework for semi-supervised node classification on graphs," 2020, *arXiv:2012.04187*.

[45] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodola, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Nashville, TN, USA, Jun. 2021, pp. 2743–2752.

[46] L. Zheng, J. Zhou, C. Chen, B. Wu, L. Wang, and B. Zhang, "ASFGNN: Automated separated-federated graph neural network," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 3, pp. 1692–1704, May 2021.

[47] G. Mei, Z. Guo, S. Liu, and L. Pan, "SGNN: A graph neural network based federated learning approach by hiding structure," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 2560–2568.

[48] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, Nov. 2021.

[49] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[50] S. Ahmed, K. H. Sheikh, S. Mirjalili, and R. Sarkar, "Binary simulated normal distribution optimizer for feature selection: Theory and application in COVID-19 datasets," *Exp. Syst. Appl.*, vol. 200, Aug. 2022, Art. no. 116834.

[51] Y. Yuan, X. Mu, X. Shao, J. Ren, Y. Zhao, and Z. Wang, "Optimization of an auto drum fashioned brake using the elite opposition-based learning and chaotic k-best gravitational search strategy based grey wolf optimizer algorithm," *Appl. Soft Comput.*, vol. 123, Jul. 2022, Art. no. 108947.

[52] W. Zhao, L. Wang, and S. Mirjalili, "Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications," *Comput. Methods Appl. Mech. Eng.*, vol. 388, Jan. 2022, Art. no. 114194.

[53] Y. Yuan, J. Ren, S. Wang, Z. Wang, X. Mu, and W. Zhao, "Alpine skiing optimization: A new bio-inspired optimization algorithm," *Adv. Eng. Softw.*, vol. 170, Aug. 2022, Art. no. 103158.

[54] B. Alhalabi, M. M. Gaber, and S. Basura, "MicroNets: A multi-phase pruning pipeline to deep ensemble learning in IoT devices," *Comput. Electr. Eng.*, vol. 96, Dec. 2021, Art. no. 107581.

[55] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," 2015, *arXiv:1511.06335*.

[56] L. Guo and Q. Dai, "Graph clustering via variational graph embedding," *Pattern Recognit.*, vol. 122, Feb. 2022, Art. no. 108334.

[57] Y. Bai, H. Ding, Y. Qiao, A. Marinovic, K. Gu, T. Chen, Y. Sun, and W. Wang, "Unsupervised inductive graph-level representation learning via graph-graph proximity," 2019, *arXiv:1904.01098*.

[58] W. Li and A. McCallum, "Pachinko allocation: DAG-structured mixture models of topic correlations," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*. Pittsburgh, PA, USA: ACM Press, 2006, pp. 577–584.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016, *arXiv:1603.05027*.

[60] B. Alhalabi, M. M. Gaber, and S. Basurra, "EnSyth: A pruning approach to synthesis of deep learning ensembles," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Bari, Italy, Oct. 2019, pp. 3466–3473.

[61] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*.

BESHER ALHALABI received the B.Sc. degree in software engineering and the M.Sc. degree in business intelligence. He is currently pursuing the Ph.D. degree with Birmingham City University. He is currently a Lecturer with Birmingham City University, where he teaches courses on data analytics and artificial intelligence. He is a key member of the university's esteemed data analytics and artificial intelligence group, where he researches cutting-edge techniques in machine learning and artificial intelligence. Throughout his academic career, he has demonstrated a deep commitment to his studies. He has also made significant contributions to the field of machine learning and artificial intelligence through his research. His work has focused primarily on deep ensemble learning on edge devices, a cutting-edge area of research that has significant implications for the development of next-generation AI systems. He has published several articles on this topic, which have been well-received by his peers in the academic community. In addition to his academic work, he has extensive experience in designing and developing business intelligence applications. He has worked with a wide range of organizations across multiple industries, helping them to leverage the power of data to make informed decisions and gain a competitive edge in their respective markets.

SHADI BASURRA received the B.Sc. degree (Hons.) in computer science from Exeter University, U.K., the M.Sc. degree in distributed systems and networks from Kent University, Canterbury, U.K, and the Ph.D. degree from the University of Bath in collaboration with Bristol University.

After completing his Ph.D., he worked with Sony Corporation developing Goal Decision Systems and then joined BCU as an academic. He is currently an Associate Professor in computer science with Birmingham City University (BCU), U.K., the Head of the master's program in computer science, and heading the Data Analytics and AI Research Group (DAAI), BCU. His recent research is related to edge computing, in particular, the use of lightweight machine- and deep-learning techniques that allow learning and inference to occur on devices with limited resources, such as IoT devices to aid rapid decision-making while maintaining the highest possible accuracy in classification and regression analysis.

MOHAMED MEDHAT GABER received the Ph.D. degree from Monash University, Australia. He is currently a Professor in data analytics with the School of Computing and Digital Technology, Birmingham City University. He is also seconded as the Dean of the Faculty of Computer Science and Engineering, Galala University. He then held appointments with the University of Sydney, CSIRO, and Monash University, all in Australia. Before joining Birmingham City University, he worked with Robert Gordon University, as a Reader in computer science and the University of Portsmouth, as a Senior Lecturer in computer science, both in the United Kingdom. He has published over 200 journal articles and conference papers, coauthored three monograph-style books, and edited/co-edited seven books on artificial intelligence. His published work has appeared in the proceedings of prestigious conferences including ICDM and ISWC and reputed journals including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and *Data Mining and Knowledge Discovery*, to mention a few. His work has attracted well over 8000 citations, with an H-index of 43. He has supervised/cosupervised to successful completion of 19 Ph.D. students. He has acted as an Examiner for 39 Ph.D. candidates. In 2007, he was awarded the CSIRO Teamwork Award. He has had numerous chairing roles serving the artificial intelligence community including the General Co-Chair of the 3rd IEEE International Conference on Data Science and Computational Intelligence (DSCI 2019) and the Programme Committee Co-Chair of the IEEE Mobile Data Management (MDM 2016). He is recognized as a fellow of the British Higher Education Academy (HEA).