2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-7281-8470-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICMLA51294.2020.00120

Federated Learning of Neural Network Models with Heterogeneous Structures

Kundjanasith Thonglek, Keichi Takahashi, Kohei Ichikawa, Hajimu Iida
Nara Institute of Science and Technology
Nara, Japan
Email: {thonglek.kundjanasith.ti7, keichi, ichikawa}@is.naist.jp, iida@itc.naist.jp

Chawanat Nakasan
Kanazawa University
Ishikawa, Japan
Email: chawanat@staff.kanazawa-u.ac.jp

***Abstract*—Federated learning trains a model on a centralized server using datasets distributed over a large number of edge devices. Applying federated learning ensures data privacy because it does not transfer local data from edge devices to the server. Existing federated learning algorithms assume that all deployed models share the same structure. However, it is often infeasible to distribute the same model to every edge device because of hardware limitations such as computing performance and storage space. This paper proposes a novel federated learning algorithm to aggregate information from multiple heterogeneous models. The proposed method uses weighted average ensemble to combine the outputs from each model. The weight for the ensemble is optimized using black box optimization methods. We evaluated the proposed method using diverse models and datasets and found that it can achieve comparable performance to conventional training using centralized datasets. Furthermore, we compared six different optimization methods to tune the weights for the weighted average ensemble and found that tree parzen estimator achieves the highest accuracy among the alternatives.**

***Index Terms*—Ensemble learning, Federated learning, Decentralized dataset, Deep Neural Network**

I. Introduction

Federated learning trains a single global model on a centralized server from training data distributed over a large number of edge devices, while it also trains personalized local models on each edge device. The first framework for federated learning was proposed by Google in 2016 [1]. Federated learning has been introduced because of data privacy concerns and data license agreements. For example, the ubiquitous use of cameras in a home environment raises privacy concerns, which is an impediment to install smart home systems [2]. For this reason, edge devices need to have the capability to train neural networks locally [3]. These powerful devices are therefore recently used by many people and produce massive amounts of data including their private information. Thus it is important to use their data while keeping their privacy.

Ensuring data privacy is one of the advantages when applying federated learning because it does not require transferring the local data from edge devices to a centralized server. Furthermore, it reduces the amount of data transfer between the edge devices and the centralized server. The global model is trained by parameters extracted from the local models instead of the local training data itself [4]. There are several types of parameters used in training the global model such as model weights and gradients used to update model weights. In addition to the global model, a personalized local model is built on each edge device by retraining the global model with the local data [5]. The personalized local model is able to integrate both the generic characteristics of datasets on all edge devices and the specific characteristics of the local dataset on each edge device. Users prefer to use the personalized models because the model is tuned for their private information.

However, existing federated learning algorithms such as FedSGD [6] and FedAVG [7] have a critical limitation that assumes the models distributed on the edge devices share the same homogeneous structure. In practical situations, not all edge devices can support the same model due to limitations in available computing resources, storage capacity, physical space, power consumption, network bandwidth and so on. In addition, there are advanced edge devices equipped with accelerators such as GPUs and Google's Edge TPUs[1]. They provide impressive computing performance while occupying less physical space and consuming less power. Each device therefore has different available hardware and limitations. To aggregate information from various edge devices and perform federated learning, we need a method that can handle multiple neural networks with different structures. For this purpose, we look into a method to ensemble heterogeneous models.

In this paper, we focus on the image classification task. We propose a method based on weighted average to ensemble federated neural networks with heterogeneous model structures. It is reasonable to weight each model differently since the local training dataset may have different amount of data for each output class. Hence we should not use the same weight to average all models. Black box optimization is applied to determine the optimal weight values.

The rest of this paper is organized as follows. Section II describes related work on federated learning of neural network models. Section III presents our proposed method to ensemble neural network models with heterogeneous structures over decentralized dataset to improve the accuracy of the models. Section IV shows experimental results when applying the proposed method to various neural network models with different parameter optimization methods. Section V discusses the proposed method in terms of model accuracy and runtime. Section VI concludes this paper and discusses future work.

[1] https://cloud.google.com/edge-tpu

978-1-7281-8470-8/20/$31.00 ©2020 IEEE
DOI 10.1109/ICMLA51294.2020.00120

Algorithm 1: Federated Averaging (Server)

for $t \leftarrow 1$ **to** *number of communication rounds* **do**
  for $k \leftarrow 1$ **to** *number of selected clients* **do**
    Receive w_{t+1}^k from client k
  end
  $w_{t+1} \leftarrow \sum_{t=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  Send w_{t+1} to selected clients
end

Algorithm 2: Federated Averaging (Client)

Receive w_t from server
$w_{t+1}^k \leftarrow w_t$
for $e \leftarrow 1$ **to** *number of local epochs* **do**
  for $b \leftarrow 1$ **to** *number of batches* **do**
    $w_{t+1}^k \leftarrow w_{t+1}^k - \eta g_k$
  end
end
Send w_{t+1}^k to server

II. Background

This section gives a brief overview of existing federated learning methods and combining multiple heterogeneous neural networks.

A. Federated Stochastic Gradient Descent (FedSGD)

Federated Stochastic Gradient Descent (FedSGD) is a federated learning algorithm based on SGD [8]. The global model and local models in FedSGD are trained in the following manner [9]. In each communication round, the centralized server broadcasts the current global model w_t to all clients (t is the current communication round). Each client k then computes the gradient g_k using its local training data and sends the computed gradient to the server. The server averages the gradients received from all clients and generates the new global model w_{t+1} according to Equation 1. Here, η denotes the learning rate, n denotes the the total number of samples, K denotes the total number of clients and n_k denotes the number of training samples on client k.

$$w_{t+1} = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k. \tag{1}$$

One of the drawbacks of FedSGD is high communication cost. Since FedSGD needs to frequently communicate between the server and the client [10], it suffers from slow convergence. The most popular approach to tackle this problem is to increase the number of local training epochs and decrease the size of local batches on the client side [11].

B. Federated Averaging (FedAVG)

Federated Averaging (FedAVG) is the current state-of-the-art federated learning algorithm [12]. FedAVG was designed to alleviate the high communication cost and and improve the convergence speed of FedSGD [13]. It increases the number of local training epochs performed on the edge device.

Algorithm 1 and 2 show the pseudocodes of FedAVG for the server and the client, respectively. On the server side, a subset of participating clients is randomly selected in each round. Next, the server receives the local model w_{t+1}^k from each selected client k. The server then computes the new global model w_{t+1} by averaging the local models received from the clients and broadcasts the new model w_{t+1} to the clients. On the client side, the client receives the latest global model w_t from the server. The client updates the local model w_{t+1}^k using its local training data and sends the updated model w_{t+1}^k to the server. FedSGD can be thought of as a special case of FedAVG where the local batch size is ∞ and the number of local training epochs is one.

Both FedSGD and FedAVG assume that all edge devices run the same model. Therefore, these existing methods cannot aggregate information from heterogeneous models. This paper aims at overcoming this limitation and proposes a federated learning algorithm that is able to combine models with heterogeneous structures.

C. Combining Heterogeneous Neural Networks

Ensemble learning combines the predictions from multiple models to produce a more accurate prediction than only using one of the models [14]. Ensemble learning has been used in previous works to combine multiple neural networks. Lee *et al.* applied ensemble learning to recognize human actions [15]. They combined multiple LSTM models with different hyperparameters using average ensemble to model both short-term and long-term dependencies. However, they did not consider multiple models with heterogeneous structures.

Deng *et al.* proposed a machine learning model for speech recognition that uses stacking ensemble to combine two models with different structures (an RNN and a CNN) [16]. They assumed traditional centralized learning while this paper focuses on federated learning.

III. Methodology

This section describes the proposed federated learning algorithm for heterogeneous neural network models.

A. Overview

The basic idea behind the proposed method is to ensemble the heterogeneous models. Naively applying FedSGD or FedAVG is not possible since there does not exist any obvious mapping between parameters in different models. Therefore, we employ *weighted average ensemble* as shown in Equation 2. Here, α_{ij} represents the weight for the i-th model and j-th class, x is the input image and y is the final output. We employ weighted average because simple average or majority voting combines all models equally and results in suboptimal performance if the local training datasets distributed across the edge devices are biased.

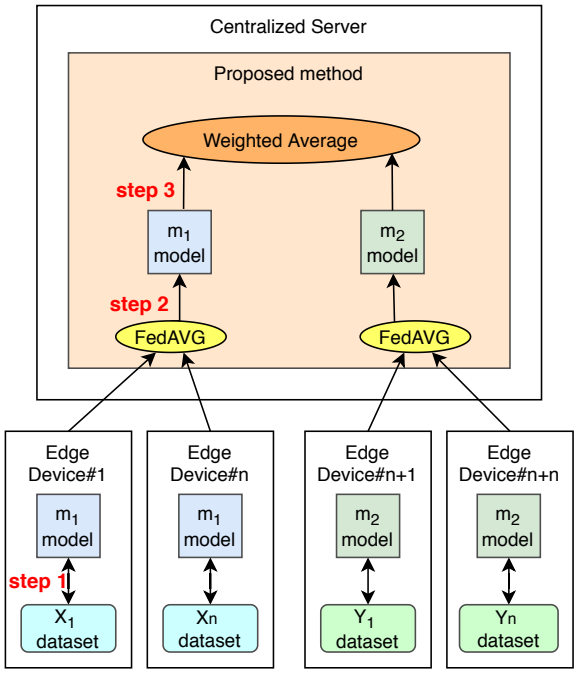

Fig. 1: Overview of the proposed method

$$y = \sum_{i=1}^{N} \sum_{j=1}^{C} \alpha_{ij} \cdot m_i(x) \tag{2}$$

Figure 1 illustrates the overview of the proposed method. We first deploy different models on each edge device considering its hardware constraints such as processing capability and storage capacity. The edge devices train their models using their local datasets (step 1 in Fig. 1) and send their updated weights to the centralized server. The server aggregates the updated weights for each global model using FedAVG (step 2). Finally, weighted average ensemble is used to combine the outputs from each model (step 3).

B. Tuning of weights

The weight α for the weighted average ensemble is tuned by following the procedure as shown in Algorithm 3. We use black box optimization algorithms to tune α. The specific optimization algorithms we consider are shown in the next subsection.

First, α is initialized to a uniform value. In each trial, the black box optimizer suggests a new candidate for α based on the historical data of parameters and achieved accuracy (line 2 in Algorithm 3). The combined output is then computed by multiplying the weight a with the output vectors from each model m_i (line 4). The accuracy of the ensemble model is calculated (line 5) and the obtained accuracy is fed back to the optimizer (line 7–9). This procedure is repeated for a fixed number of trials.

Algorithm 3: Optimization of weights

Input: i-th model m_i, images from the tuning dataset x, labels from the tuning dataset y
Parameters: Number of output classes C, Number of models N
Output: $C \times N$ matrix α where α_{ij} represents the optimized weight for the i-th model and j-th class

1 $\alpha_{ij} \leftarrow \frac{1}{N}$
2 **for** $e \leftarrow 1$ **to** *number of trials* **do**
3 $\alpha \leftarrow$sampling$(optimizer)$
4 **for** $d \leftarrow 1$ **to** *number of images* **do**
5 $output \leftarrow \sum_{i=1}^{N} \sum_{j=1}^{C} \alpha_{ij} \cdot m_i(x_d)$
6 $accuracy \leftarrow$validate$(y_d, output)$
7 **end**
8 set_parameters$(optimizer, \alpha)$
9 set_target$(optimizer, accuracy)$
10 optimize$(optimizer)$
11 **end**
12 **return** α

TABLE I: Hardware specification

Hardware	Specification
CPU	Intel Xeon E5-2650 v2 (2.20 GHz, 12 cores)
Main Memory	256 GB
GPU	NVIDIA Tesla P100
GPU Memory	16 GB

C. Optimization algorithms

The optimizer is responsible for tuning α such that the accuracy of the ensembled model is maximized. In this paper, we consider and compare six well-known black box optimization algorithms: (1) Grid Search (GS) [17], (2) Random Search (RS) [18], (3) Particle Swarm Optimization (PSO) [19], (4) Bayesian Search (BS) [20], (5) Tree Parzen Estimator (TPE) [21], and (6) Sequential Model-based Algorithm Configuration (SMAC) [22].

IV. Evaluation

This section evaluates the proposed method from four aspects to investigate the characteristics of the ensemble model created with the proposed method. We first evaluate the accuracy achieved with each optimization method along with its runtime. We then investigate if the proposed method can be applied to different combinations of heterogeneous models and datasets.

A. Experimental Setup

Table I presents the hardware used for the evaluation. Using this server, we simulate the centralized server and all edge devices for the experiments. To evaluate the proposed method, we prepared three experimental setups using two, three and four different models, respectively. Table II details of each setup. We evaluated our method with four image classification

TABLE II: Experimental setup

	Setup A	Setup B	Setup C
Total # of devices	1,200	1,200	1,200
# of models	2	3	4
# of devices per model	600	400	300
# of images per model	24,000	16,000	12,000
MobileNet	✓	✓	✓
DenseNet169	✓	✓	✓
ResNet50		✓	✓
VGG16			✓

TABLE III: Dataset specification

Name	# of images	# of output classes
R-Cellular	73,000	1,108
CIFAR-10	70,000	10
CIFAR-100	70,000	100
ImageNet	100,000	1,000

datasets as shown in Table III: R-Cellular, CIFAR-10, CIFAR-100 and ImageNet. Since the smallest dataset (CIFAR-10 and CIFAR-100) contains 70,000 images, we divided each dataset into 48,000 images for training, 10,000 images for tuning and 10,000 images for validation. The training dataset is distributed over the edge devices while the tuning and validation datasets are deployed on the centralized server.

We simulated 1,200 edge devices in all setups. In a communication round, 1,000 devices are randomly selected to participate in the federated learning. Each edge device performs 10 local epochs in a communication round and sends its update local model to the server. The server updates the global model and broadcasts the new model to the edge devices. We perform 10 communication rounds in all experiments. Thus, the local batch size is 4 (4 images $\times$ 10 communication rounds). Once the FedAVG step is complete, the optimization step is performed for 50 trials.

We selected four image classification models (MobileNet, DenseNet169, ResNet50 and VGG16) to evaluate the proposed method. Table IV summarizes the required storage size (MB) to deploy the models and the required number of floating point operations (MFLOPs) in each epoch. Since VGG16 occupies large storage space (553.43 MB) and requires a lot of computation (276.68 MFLOPs), not all edge devices can run VGG16 using their limited resources. On the other hand, if we deploy MobileNet to all edge devices, it would waste the resources of devices that could handle larger models capable of achieving higher accuracy. Through the evaluation, we will confirm that our method efficiently leverages heterogeneous environments and achieves a good accuracy close to that of running VGG16 on all edge devices.

B. An Example of the Optimized Weights

Figure 2 visualizes the weights to average the heterogeneous models optimized using the proposed method in the experimental setup C with the CIFAR-10 dataset. For the same output class, models with darker cells are weighted heavier than

TABLE IV: Model specification

Name	Size [MB]	MFLOPs
MobileNet	17.02	8.52
DenseNet169	57.23	28.77
ResNet50	102.55	51.27
VGG16	553.43	276.68

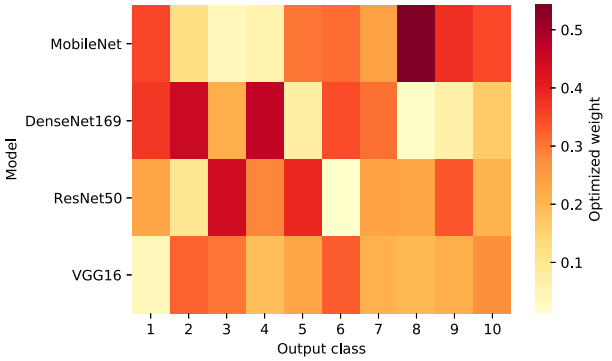

Fig. 2: Optimized weights (Setup C, CIFAR-10 dataset and TPE optimization)

the models with lighter cells. Figure 2 clearly reveals that the optimized weight for each output class and each heterogeneous model is different. Taking the 8th output class as an example, MobileNet has the highest weight while DenseNet169 has the lowest weight. This suggests that MobileNet is able to identify the 8th class more accurately than DenseNet169. In this manner, models that achieve higher accuracy on a particular class will have higher weights assigned to them after the weight optimization.

C. Accuracy of Optimization Methods

Tuning the wights to ensemble the heterogeneous models is the key in our proposed method. We evaluated the accuracy with respect to the optimization methods for tuning the weights. We used the experimental setup A, which ensembles MobileNet and DenseNet169, and the R-Cellular dataset. Figure 3 shows the achieved accuracy using each of the optimization methods. Figure 4 shows the trends of improvement during the trials. Before applying the proposed method, the accuracy of the combined model was 63.19%. This is effectively averaging the outputs from each model since the weights are uniformly initialized before the optimization. After performing 50 optimization trials, TPE achieved the highest accuracy, 75.65%, while GS produced the lowest accuracy, 70.05%. However, the differences are not that significant. Similar results were also observed with the other experimental setups B and C.

D. Runtime of Optimization Methods

We compared the runtime required for tuning the weights with different optimization methods. Here we used setup A and the R-Cellular dataset. Figure 5 shows the comparison of runtime. Evidently, optimization methods that achieve higher

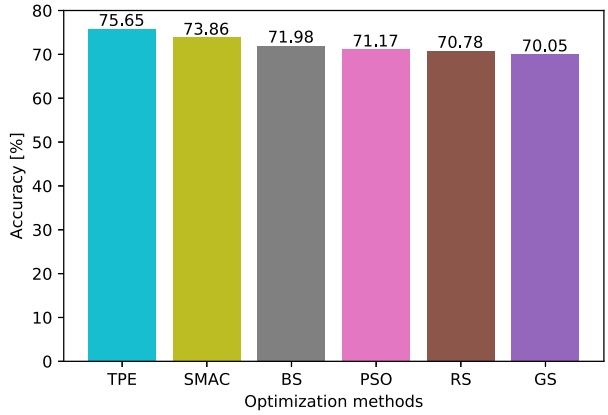

Fig. 3: Comparison of accuracy with different optimization methods (Setup A and R-Cellular dataset)

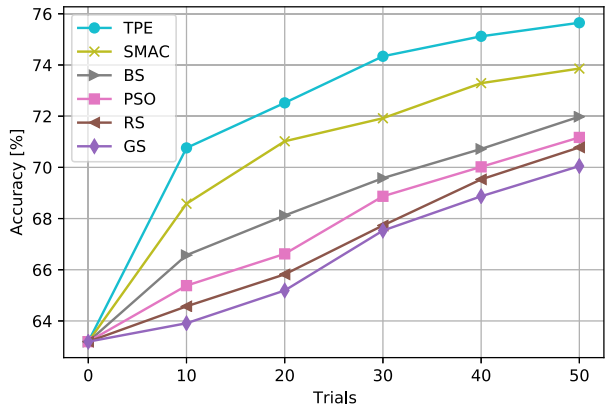

Fig. 4: Improvement of accuracy during the trials (Setup A and R-Cellular dataset)

accuracy require longer runtime. TPE was the slowest taking more than 3 hours to complete the optimization whereas GS was the fastest only taking 17 minutes. We also measured the runtime using experimental setups B and C and observed the same trend. Moreover, we also found out that the runtime increases with the number of models and the number of output classes of each model.

E. Results for Different Combinations of Models

The purpose of this evaluation is to confirm that the proposed method is still effective if the number of heterogeneous model increases. We measured the accuracy of the proposed method with varying number of heterogeneous models (R-Cellular dataset and TPE optimization). In addition, we measured the accuracy of each model when trained with a centralized and a distributed dataset using FedAVG as baselines.

Table V shows the comparison of accuracy using the different models over centralized and decentralized R-cellular dataset. Table VI shows the comparison of accuracy using different combinations of models (R-Cellular dataset and TPE

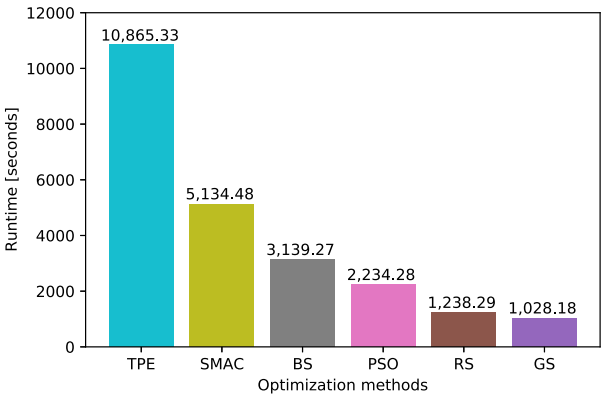

Fig. 5: Comparison of runtime with different optimization methods (Setup A and R-Cellular dataset)

TABLE V: Comparison of accuracy using different models (R-cellular dataset and TPE optimization)

Model	Centralized	FedAVG
MobileNet	75.43	75.29
DenseNet169	75.94	75.71
ResNet50	76.29	76.14
VGG16	78.57	77.92

optimization). Table VI indicates that the accuracy of the proposed method in setup A, which combines MobiletNet and DenseNet169, is 75.65%. In FedAVG, MobiletNet and DenseNet169 achieve 75.29% and 75.71%, respectively. This suggests that the proposed method is able to effectively combine its constituent heterogeneous models without significant overhead in terms of accuracy. The proposed method can combine the heterogeneous models in setup A, B, and C to 75.65%, 76.02%, and 77.43%, respectively. Furthermore, MobiletNet and DenseNet169 trained with a centralized dataset each reach 75.43% and 75.94% accuracy, which is very close to FedAVG and the proposed method. Thus we conclude that the overhead of federated learning is minimal.

The same trend is observed in setups B and C as well. In summary, the proposed method can effectively combine different combinations of models without incurring significant loss of accuracy.

F. Results for Different Datasets

Lastly, we confirm that the proposed method can be applied to datasets with different number of output classes. Table VII shows the accuracy of the proposed method for the four datasets. In addition, the accuracy of VGG16 models trained using centralized and decentralized datasets is shown as baselines. The accuracy of the ensemble of the four models (77.43%) is very close to a VGG16 model training using FedAVG (77.92%). The difference of accuracy between VGG16 and the proposed method is consistently less than 1% for all datasets. The accuracy of the proposed mehthod is even

TABLE VI: Comparison of accuracy using different combination of models (R-cellular dataset and TPE optimization)

Combination of Models	Proposed
MobileNet + DenseNet169	75.65
MobileNet + DenseNet169 + ResNet50	76.02
MobileNet + DenseNet169 + ResNet50 + VGG16	77.43

TABLE VII: Comparison of accuracy using different datasets (Setup C and TPE optimization)

Dataset	Centralized	FedAVG	Proposed
R-Cellular	78.57	77.92	77.43
CIFAR-10	89.95	89.67	89.36
CIFAR-100	88.56	88.37	88.29
ImageNet	86.86	86.42	85.88

comparable to a VGG16 model trained using a centralized dataset.

V. Discussion

The proposed method was evaluated by varying the number of output classes in the datasets and the number of models with heterogeneous structures. We found that the runtime of the proposed methods increases with the number of classes in the dataset and the number of heterogeneous models. This is because the proposed method needs to iterate over every output class of every heterogeneous model.

We found that TPE achieves the highest accuracy among the six optimization methods we compared. On the other hand, TPE takes the longest runtime. This is likely because TPE defines the parameter space as a tree structure, thus it requires longer time to set up the parameter space and to traverse the tree structure to select the parameter to optimize. However, TPE is the best parameter optimization method in terms of accuracy because the tree traversal is able to to select the optimized parameter and lookup the optimized parameter in the next trial by focusing on the same sibling.

VI. Conclusion

In this paper, we proposed a novel federated learning algorithm for neural network models with heterogeneous structures. The proposed method utilizes weighted average ensemble to combine the outputs from different models. Black box optimization is used to tune the weights for the output classes of each model. We compared six black box optimization algorithms (grid search, random search, particle swarm optimization, bayesian search, tree Parzen estimator and sequential model-based algorithm configuration) and found that TPE was able to achieve the highest accuracy. However, TPE took the longest runtime among the six methods.

As a future work, we will enhance the proposed method to regression and localization tasks. In addition, other parameter optimization techniques and federated learning algorithms will be investigated.

References

[1] Y. Qiang, L. Yang, C. Tianjian, and T. Yongxin, "Federated machine learning: Concept and applications," *ACM Transaction Intelligence System Technology*, vol. 10, no. 2, pp. 12–19, Jan. 2019.

[2] E. Fernandes, G. Yang, M. Do, and W. Sheng, "Detection of privacy-sensitive situations for social robots in smart homes," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 727–732.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralize data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Apr. 2017, pp. 1273–1282.

[4] Q. Qi, Q. Huo, J. Wang, H. Sun, Y. Cao, and J. Liao, "Personalized sketch-based image retrieval by convolutional neural network and deep transfer learning," *IEEE Access*, vol. 7, pp. 16 537–16 549, Jan. 2019.

[5] C. Nadiger, A. Kumar, and S. Abdelhak, "Federated reinforcement learning for fast personalization," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Jun. 2019, pp. 123–127.

[6] V. Felbab, P. Kiss, and T. Horvath, "Optimization in federated learning," in *Proceedings of the International Conference on Information Technologies - Application and Theory (ITAT)*, Sep. 2019, pp. 58–65.

[7] A. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, Jan. 2019.

[8] J. Rie and Z. Tong, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, vol. 1, Dec. 2013, pp. 315–323.

[9] G. K. Dziugaite and D. M. Roy, "Entropy-SGD optimizes the prior of a pac-bayes bound: Generalization properties of entropy-SGD and data-dependent priors," in *Proceedings of the International Conference on Machine Learning (ICML)*, Feb. 2018, pp. 1376–1385.

[10] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Apr. 2016, pp. 1–10.

[11] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *CoRR*, vol. abs/1905.10497, May 2019.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] B. McMahan, E. Moore, D. Ramage, and B. Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, Feb. 2016.

[14] F. Huang, G. Xie, and R. Xiao, "Research on ensemble learning," in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, vol. 3, Jan. 2009, pp. 249–252.

[15] I. Lee, D. Kim, S. Kang, and S. Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1012–1020.

[16] L. Deng and J. Platt, "Ensemble deep learning for speech recognition," in *Proceedings of INTERSPEECH*, September 2014.

[17] F. Xue, D. Wei, Z. Wang, T. Li, Y. Hu, and H. Huang, "Grid searching method in spherical coordinate for PD location in a substation," in *Proceeding of the International Conference on Condition Monitoring and Diagnosis (CMD)*, Sep. 2018, pp. 1–5.

[18] Y. Shang and J. Chu, "A method based on random search algorithm for unequal circle packing problem," in *Proceedings of the International Conference on Information Science and Cloud Computing Companion (ISCC-C)*, Dec. 2013, pp. 43–47.

[19] M. Kirschenbaum and D. W. Palmer, "Perceptualization of particle swarm optimization," in *Proceedings of the Swarm/Human Blended Intelligence Workshop (SHBI)*, Sep. 2015, pp. 1–5.

[20] A. Garcia, E. Campos, and C. Li, "Distributed on-line Bayesian search," in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Dec. 2005, pp. 1–5.

[21] M. Zhao and J. Li, "Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators," in *Proceedings of the International Conference on Advanced Computational Intelligence (ICACI)*, Mar. 2018, pp. 613–618.

[22] H. Frank, H. Holger, and L. B. Kevin, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*, Jan. 2011, p. 507–523.