Federated Learning Based Mobile Edge Computing for Augmented Reality Applications

Dawei Chen*, Linda Jiang Xie†, BaekGyu Kim‡,
Li Wang§, Choong Seon Hong¶, Li-Chun Wang‖ and Zhu Han*¶
*Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
†Department of Electrical and Computer Engineering, The University of North Carolina at Charlotte, Charlotte, NC, USA
‡InfoTech Labs, Toyota Motor North America R&D, Mountain View, CA, USA
§School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China
¶Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea
‖Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan

***Abstract*—The past decade has witnessed the prosperous growth of augmented reality (AR) devices, as they provide immersive and interactive experience for customers. AR applications have the properties of high data rate and latency sensitivity. Currently, the available bandwidth is relatively limited to transmit and process enormous generated data. Meanwhile, it is challenging for AR to accurately detect and classify the object in order to perfectly combine the corresponding virtual contents with the real world. In this work, we focus on how to solve the computation efficiency, low-latency object detection and classification problems of AR applications. Firstly, we introduce and analyze the practical mathematical model of AR, and connect the AR operating principles with the object detection and classification problem. To address this problem and reduce the executing latency simultaneously, we propose a framework collaborating mobile edge computing paradigm with federated learning, both of which are decentralized configurations. To evaluate our method, numerical results are calculated based on the open source data CIFAR-10. Compared to centralized learning, our proposed framework requires significantly fewer training iterations.**

I. Introduction

The past decade has witnessed the prosperous growth in augmented reality (AR) devices. AR's ability to combine virtual contents with real world has attracted attention from different fields, which can provide immersive and interactive experience for gear wearers [1]. For example, at Mobile World Congress (MWC) 2019, Microsoft introduced the HoloLens 2 to the world, demonstrating its tremendous potentials in a variety of applications, such as food nutrition analysis, human organs 3-D visualizations in surgeries, real-time and adaptive forestage projection for concerts, and vehicle inspection and malfunction analysis, etc, as illustrated as an example in Fig. 1. In addition, AR brings great success for the gaming field such as the Pokémon Go, which harvests $1.8 billions revenue at the second anniversary. However, all these applications are sensitive to latency because delay can result in virtual objects to move and bring dizziness to users [2]. Currently, bandwidth is relatively limited. It is impossible to transmit and process the enormous data generated by multiple AR users. Therefore, latency is usually an inevitable challenge.

To overcome long latency, a new computing paradigm has emerged, called mobile edge computing (MEC). Basically, MEC is inspired by shortening the transmission delay through performing some computation and storage processes at the edge. According to Cisco's anticipation, mobile data of 77 exabytes per month will be generated by 2022, and annual traffic will reach approximately one zettabyte [3]. In addition, according to the Moore's Law, the computation abilities of these edge devices will increase exponentially in the near future [4]. Therefore, the available computational resources and storage capacity are plentiful. More importantly, these resources are at the edge side and are closer to the users than the centralized datacenter. Therefore, for high data rate and latency-sensitive applications, it is an effective way to take advantage of the MEC paradigm to reduce communication delay significantly.

Basically, what AR does is to implant 3-D virtual objects in a real-world context. Then, placing the object in a given scene becomes a vital problem that needs to be addressed. Generally, AR can be classified into two categories: one is the marker-based AR, and the other one is the markerless AR [5]. For the marker-based AR, it is necessary to produce a marker in advance, like a template card with a certain size and shape or a Quick Response (QR) Code, and manually place the marker at an ideal position so as to generate a desirable reference plane. Next, with the help of camera, it is practical to recognize the location and estimate the pose of marker [6]. In this way, we can obtain the marker coordinates, whose origin is the centroid of the marker. Combining the obtained marker coordinates with prior-known camera coordinates, we can easily calculate the observed screen coordinates through 3-D geometric transformation. For the markerless AR, the principles are the same as that of marker-based AR, except that a special physical template is not required. Instead, we utilize some feature points to define the boundaries of the projected model based on the computer vision technologies. In

This work is partially supported by US MURI AFOSR MURI 18RT0073, NSF EARS-1839818, CNS1717454, CNS-1731424, CNS-1702850, and CNS-1646607.

978-1-7281-4905-9/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

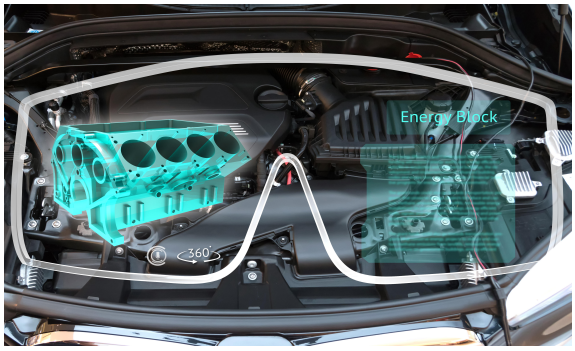

Fig. 1. The AR application for vehicle inspection.

addition, the demands for multi-object virtualization increase dramatically. Thus, the accuracy of detection and classification is of great importance, which directly affects user experience. Recent achievements in the machine learning field provide an effective and powerful tool to solve such feature extraction problems.

For traditional machine learning algorithms, such as support vector machine (SVM), random forest (RF) or deep neural network (DNN), the whole dataset is typically divided into the training set and the test set, or one more validation set sometimes. The training set is used to build a divinable model and then test the model among the test set so as to obtain the prediction results. These procedures are homogeneously performed on one single machine, in other words, these methods are centralized-performed. However, when it comes to the situation with multiple mobile users using latency-critical AR applications, this centralized machine learning architecture is not applicable any more. Because, generally speaking, for a model trained based on one single user can be unsuitable for another user. Whereas, if all the users decide to obtain their own customized model, the total computational consumption and power consumption can be extremely large. Therefore, how to obtain an accurate global model and reduce the computational and power costs for each user is of great challenges.

Actually, many of the existing literatures related to MEC are focused on the computation efficiency problem. In [7], a matching game based method is utilized to solve a joint radio and computational resource allocation problem in a fog computing system. In [8], a fine-grained collaborative offloading strategy is proposed to address a content-popularity based caching problem in a MEC network. In [9], the block successive upper bound minimization based method is proposed to tackle with the joint computing, caching, communication, and control optimization problem. However, these works are based on the game theory or the optimization. In this paper, in order to address the computation efficiency and low-latency feature extraction problem for AR applications, we propose a machine learning framework to address the object detection problem for the multi-user AR application in the MEC scenario, which is not considered in existing literatures.

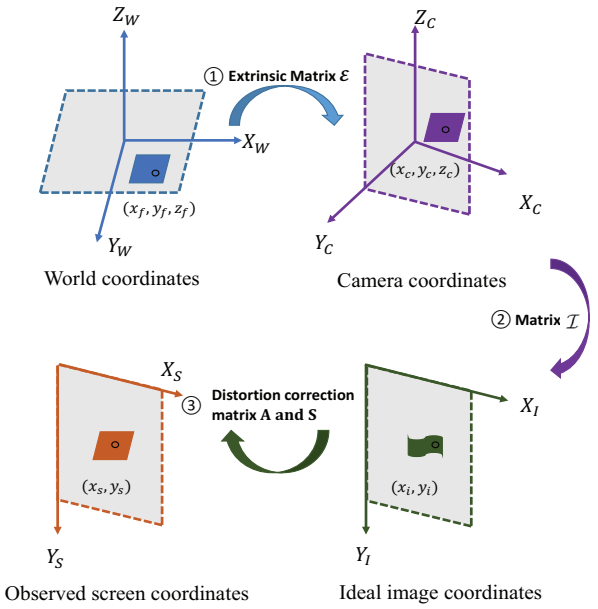

Fig. 2. The 3-D geometry transformation process in AR glasses.

To summarize, our main contributions are as follows:

- For the purpose of addressing the computation efficiency and low-latency object classification problem for AR applications, we propose a framework that integrates MEC and FL so as to obtain the global optimal machine learning model. To the best of our knowledge, there is no existing work utilizing this approach in AR application scenario.
- We explore and analyze the practical mathematical model of AR in details. Next, we connect the AR operating principles with the object detection and classification problem.
- For the experiments, we perform the proposed framework on the open dataset and compare it with centralized learning. Consequently, we can find that our proposed framework performs better, reflected in acceptable accuracy and much fewer training rounds.

The structure of the rest paper is as follows. Section II introduces the operation principles of AR in detail and the corresponding mathematical models. Also, with regard to the AR applications, the descriptions of the specific scenario and the corresponding formulation is given in this section. Section III states the proposed framework and the details of the FL in the MEC scenario. Section IV shows the performance of our proposed method compared with the baseline method. Finally, a conclusion is drawn in Section V.

II. System Model and Problem Formulation

In this section, we detail the operation principles of AR, the description of scenario, and the corresponding problem formulation. Overall, the 3-D geometry transformation process is shown in Fig. 2. Generally, the process can be divided into three steps, 3-D transformation, 2-D transformation,

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

and correction. We will introduce them one by one in the following subsections.

A. 3-D Transformation

In this part, we introduce the process of 3-D transformation. This is a mapping from world coordinates to camera coordinates. Firstly, we define the transformation between camera and a set of feature points as:

$$\mathbf{x_c} = \mathcal{E}\mathbf{x_f}, \tag{1}$$

where $\mathbf{x_f}$ denotes the positions of feature points in feature coordinates, $\mathbf{x_c}$ denotes the transformed position in camera coordinates, and $\mathcal{E}$ is the transformation matrix between feature coordinates and camera coordinates, which is also named as the pose matrix or extrinsic camera matrix [10]. Specifically, the transformation matrix $\mathcal{E}$ can be expressed in a combination of a translation vector $\mathbf{T}$ and a rotate matrix $\mathbf{R}$, i.e.,

$$\mathbf{x_c} = [\mathbf{R}|\mathbf{T}]\,\mathbf{x_f}. \tag{2}$$

Generally, the dimensions of $\mathbf{T}$ and $\mathbf{R}$ are 3×1 and 3×3, respectively. Thus, we can rewrite (2) in a homogeneous coordinates as the following form,

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix}. \tag{3}$$

Although there are nine elements in rotation matrix $\mathbf{R}$, these parameters can be calculated via three angles in practice: α, β, and γ, which are the rotation in camera coordinates along z-axis, x-axis, and y-axis, respectively, as shown in Fig. 3. Thus, intuitively, $\mathbf{R}$ can be decided via three components, as the following equation,

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \mathbf{R}_z\mathbf{R}_x\mathbf{R}_y \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & \cos\beta \end{bmatrix} \\ &\quad \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{4}$$

Obviously, this extrinsic camera matrix have six freedoms totally. The next step is to project the contents from camera coordinates to an ideal image plane.

B. 2-D Transformation

After getting 3-D positions in camera coordinates, in this part, we will introduce the process of mapping a 3-D domain into an image plain. At first, we describe the relationship

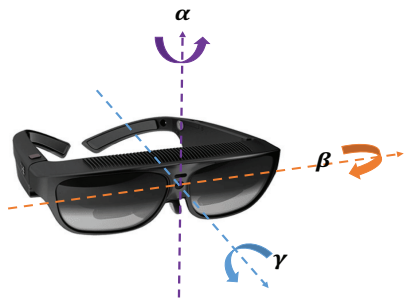

Fig. 3. Three freedoms for rotation matrix $\mathbf{R}$.

of observations from camera coordinates and ideal image coordinates as $\mathcal{I}$, i.e.,

$$\mathbf{x_I} = \mathcal{I}\mathbf{x_c}. \tag{5}$$

Generally, what matrix $\mathcal{I}$ does is 2-D translation. A 3-D point can be mapped to a 2-D image point through a perspective transformation matrix $\mathbf{F}$. Define $\mathbf{x}_i$ as the position in the ideal image coordinates. Then we have

$$\mathbf{x}_i = \mathbf{F}\mathbf{x}_c. \tag{6}$$

In homogeneous coordinates, we can rewrite (6) in a matrix form,

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}, \tag{7}$$

where $\lambda = z_c$, $x_s = x_c\frac{f}{z_c}$, $y_s = y_c\frac{f}{z_c}$, and f is the focal length. In the desirable case, the centroid of the image is located on the optical axis. But in practice, the centroid drifts away from the optical center due to the resembling technology or manufacturing process. Thus, this offset should also be considered. Define the principle point offset as (p_x, p_y). The corresponding offset calibration matrix $\mathbf{P}$ is

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & p_x & 0 \\ 0 & 1 & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

With the perspective transformation matrix, we can obtain the position in an ideal image coordinates,

$$\mathcal{I} = \mathbf{FP} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

C. Calibration

So far, we have accomplished the second step. However, most images are not square in size, such as $4:3$ or $16:9$, which results in non-square pixels. In this situation, it is

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

necessary to scale the projection from image coordinates into pixels coordinates, which is also called aspect ratio correction, expressed as matrix $\mathbf{A}$, i.e.,

$$\mathbf{A} = \begin{bmatrix} \frac{1}{m_x} & 0 & 0 \\ 0 & \frac{1}{m_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

where m_x is the pixel width and m_y is the pixel height. In addition, in the pixel coordinates, each pixel row should be orthogonal to the pixel column. However, in practice, there will be an angle θ between the pixel row and pixel column, which is called as skew, illustrated in Fig. 4. Skew causes significant distortion in image formation, which is another inevitable factor to consider. Skew is often a shear operation and can be described in matrix $\mathbf{S}$ as,

$$\mathbf{S} = \begin{bmatrix} 1 & \tan\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{11}$$

Define $\mathbf{x}_s$ as the position in the ideal image coordinates. Then, from (6) to (11), we obtain a total of four transformation or correction matrices, namely $\mathbf{F}$, $\mathbf{P}$, $\mathbf{A}$, and $\mathbf{S}$, respectively. These transformations can be accumulated together by multiplication, and the multiplication result is also called the intrinsic camera matrix in the field of computer vision. Thus, the mapping between camera coordinates and observed screen coordinates can be summarized as follows:

$$\begin{aligned} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} =& \frac{1}{\lambda} \begin{bmatrix} \frac{1}{m_x} & 0 & 0 \\ 0 & \frac{1}{m_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ & \begin{bmatrix} 1 & \tan\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \\ =& \frac{1}{\lambda} \begin{bmatrix} \frac{f}{m_x} & \tan\theta\frac{f}{m_x} & p_x & 0 \\ 0 & \frac{f}{m_y} & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}. \end{aligned} \tag{12}$$

Finally, combining the extrinsic camera matrix with the intrinsic camera matrix, we have a mapping between the feature points position in the real world coordinates and the positions in the AR generated images or videos coordinates. Thus, it follows that

$$\begin{aligned} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} =& \frac{1}{\lambda} \begin{bmatrix} \frac{f}{m_x} & \tan\theta\frac{f}{m_x} & p_x & 0 \\ 0 & \frac{f}{m_y} & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ & \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix}. \end{aligned} \tag{13}$$

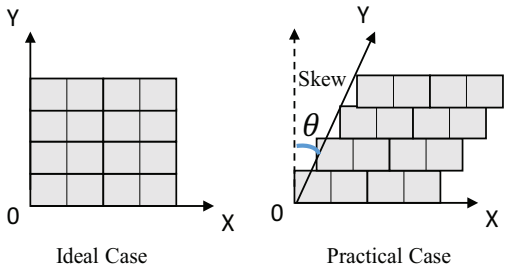

Fig. 4. The example for skew distortion.

As discussed in the previous section, object extraction and classification is important to combine the real world with virtual objects for certain AR applications, depending on where the 3-D model and which model is implanted. We assume a scenario that contains M AR wearers using the same application and one centralized cloud located far from the edge. Suppose that the area of the detected feature or object is denoted as pair $(\mathbf{x_d}, \mathbf{y_d})$, and the real or desirable area is $(\mathbf{x_f}, \mathbf{y_f})$. Define the object detection and classification machine learning model as

$$y = F(\omega; \mathcal{Q}) = F(\omega; (\mathbf{x_d}, \mathbf{y_d})), \tag{14}$$

where $F(\cdot)$ is the object detection model, ω is the general model parameters set, $\mathcal{Q}$ is the input image, and y is the prediction class. Also, we define $\hat{y}$ as the ground truth. Thereby, the problem is transformed to find the optimal value ω^* such that,

$$\|F(\omega^*; (\mathbf{x_d}, \mathbf{y_d})) - \hat{y}\| = 0. \tag{15}$$

III. Federated Learning in Mobile Edge Computing

AR applications generate a huge amount of data and are latency-sensitive. In the centralized paradigm, all data generated by different individual users will be accumulated together by uploading them separately to the data center, which leads to tremendous communication consumption. Therefore, we propose a framework that combines FL with MEC, both of which are distributed structures. FL was firstly proposed by Google [11], a distributed machine learning framework which is computation efficient and can cooperate with any machine learning model. In the AR application case, we can treat edge devices as federation group, mainly performing data generation and local model updates. Although the centralized cloud can be regarded as the central agent, it is regarded as an aggregator to compute the global model based on the uploaded local models. We define $\hat{\omega}(j)$ as the global parameters after the j-th global aggregation. And $\omega_i^j(k)$ is denoted as the local model parameter for the k-th local update steps after the j-th global aggregation for the i-th device. $f_i(\cdot)$ is the local model for the i-th device. The total number of global iterations is defined as T, so we have $j = 0, 1, \cdots, T$. And the total number of local updates iterations between

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

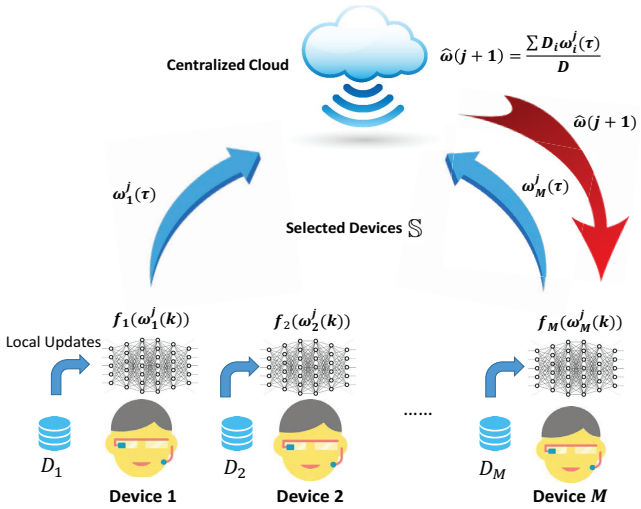

Fig. 5. The illustration for federated learning in mobile edge computing paradigm.

two adjacent global aggregations is defined as τ, so we have $k = 0, \cdots, \tau$. The corresponding FL process can be divided into the following steps:

1) The centralized cloud selects a part of existing edge devices $\mathbb{S} \subseteq \{1, \cdots, M\}$.
2) The centralized cloud sends a naive object detection and classification model with parameters $\hat{\omega}(0)$ to the selected devices $\mathbb{S}$. The naive model can even be an initial model without any training.
3) The edge device will train the received naive model locally for a certain iterations τ, based on the local-generated data, which is followed by

$$\omega_i^0(k) = \omega_i^0(k-1) - \eta \nabla \omega_i^0(k-1), \qquad (16)$$

where η is the learning rate.
4) After τ rounds, every selected device uploads the trained model with parameters $\omega_i^0(\tau)$ to the centralized cloud.
5) The centralized cloud will aggregate all the local models $\omega_i^0(\tau)$, $i \in \mathbb{S}$, generally based on the weighted average method [12], i.e.,

$$\hat{\omega}(1) = \frac{\sum_{i \in \mathbb{S}}^{M} D_i \omega_i^0(\tau)}{D}, \qquad (17)$$

where $D \triangleq \sum_{i \in \mathbb{S}} D_i$, D_i is the dataset size of device i. Here we assume $D_m \cap D_n = \emptyset$ for any $m \neq n$.
6) Iteratively, after the T-th global aggregation, we obtain the final global model parameters $\hat{\omega}(T)$. We can obtain the object classification that needs to be replaced with corresponding 3-D model via $F(\hat{\omega}(T))$, i.e., $y = F(\hat{\omega}(T))$.

This process is shown in Fig. 5. We summarize the above steps as Algorithm 1. Compared to the conventional machine learning method, the advantages of FL can be summarized as follows:

Algorithm 1 Federated Learning Algorithm

```
Input: the total number of global aggregation T; the total number of local updates between two global aggregation τ; the set of selected devices 𝕊; the learning rate η.
Initialize the global model parameters ω̂(0).
for j=0:T do
  for k=1:τ do
    if k ≠ τ then
      For each selected device i ∈ 𝕊, performing local updates based on D_i simultaneously using (16).
    else
      The devices upload ω_i^j(τ) to the centralized cloud. The datacenter performs aggregation by (17) and obtain ω̂(j+1).
      Set ω̂(j+1) = ω_i^{j+1}(0).
    end if
  end for
end for
Output: the optimum global model parameters ω̂(T).
```

TABLE I
THE COMPARISON BETWEEN FEDERATED LEARNING AND CENTRALIZED LEARNING

Models	Accuracy	Rounds
Federated Learning (i.i.d.)	88.67%	400 (for each device)
Federated Learning (non-i.i.d.)	86.00%	400 (for each device)
Centralized Learning	90.10%	1000

- Generally, the data generated by different users are non-i.i.d. data due to the various behavior characteristics. However, the task aims at obtaining a model that is suitable for each individual user. FL has been proved to be an effective way to tackle with non-i.i.d. data [13], which is perfectly suitable for multi-user scenario.
- The traditional centralized method brings huge communication workloads due to data transmission. However, this can be easily relieved by FL because what are transmitted between edge devices and datacenter are the machine learning model or even the model parameters, whose data size is greatly smaller than the original dataset [14].
- In addition, because the original data will not be uploaded, FL is an effective way to reduce the probabilities of eavesdropping, which means the user's privacy can be ensured [15].

IV. Numerical Results

In this section, we conduct numerical results using CNN to perform the classification on ten thousand labeled images from CIFAR-10 dataset [16] in Tensorflow framework, since AR is a video or image based application. Among them, 83.33% are training samples. The structure of CNN is one $3 \times 6 \times 5$ convolutional layer, one 2×2 maxpool layer, one $6 \times 16 \times 5$ convolutional layer, and followed by a fully connected network. For the proposed method, we assume the

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

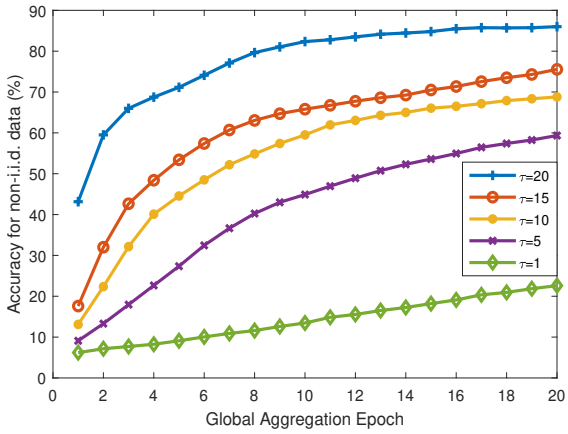

Fig. 6. Accuracy for different local updates iterations with non-i.i.d. data.

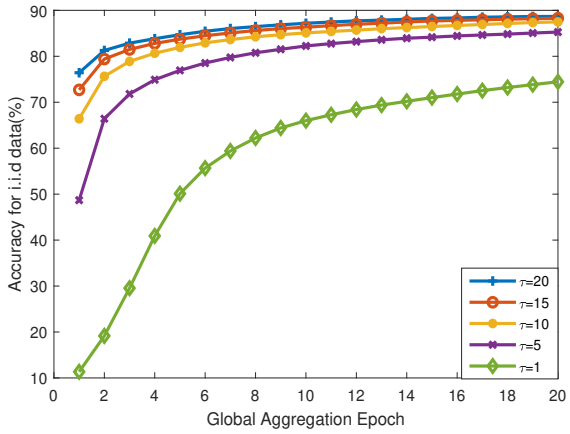

Fig. 8. Accuracy for different local updates iterations with i.i.d. data.

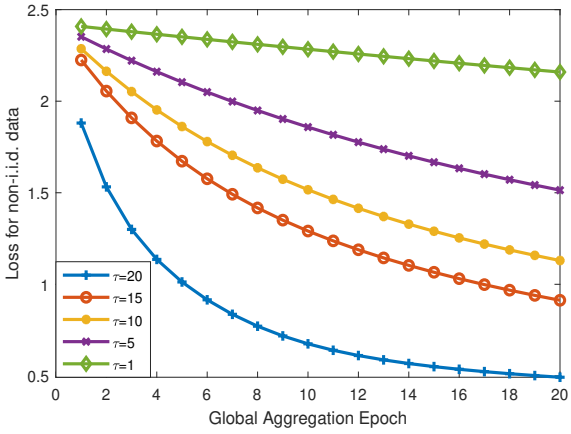

Fig. 7. Loss for different local updates iterations with non-i.i.d. data.

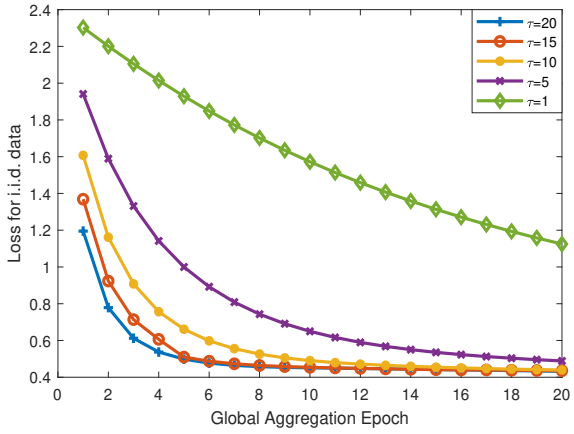

Fig. 9. Loss for different local updates iterations with i.i.d. data.

number of users is five and they have the same data size, i.e., $D_1 = \cdots = D_5$. We set the learning rate to 0.002. The local updates iteration τ is 20, the global aggregation iteration T is 20, the local batch size is 2000, and the activation function is Relu. To evaluate the performance, we also introduce the centralized learning as a baseline, which shares the same network structure. The results are shown in Table I. For i.i.d. situation setting, each user is randomly assigned a uniform distribution over 10 classes. For non-i.i.d. setting, the data is sorted by class and each user is randomly assigned equally from two classes.

We can see that from the perspective of accuracy, the centralized learning gets better performance. Since the model is trained on the whole dataset, the bias from some individual devices cannot make a difference. However, to achieve the 88.67% accuracy with i.i.d. data and 86.00% accuracy with non-i.i.d. data, FL takes 600 rounds fewer than the centralized learning, which is significantly faster. Reducing six hundred iteration rounds can alleviate latency dramatically because machine learning algorithms are usually high-complexity and each iteration consumes a lot of computational resources. In particular, for the centralized learning, each iteration is performed throughout the whole dataset. The executing time of procedure like back-propagation is proportional to the data size. As a consequence, it requires more computation time for each iteration in centralized learning than FL. Therefore, our proposed framework can bring remarkable time efficiency, with the condition that the accuracy is only reduced by 3.9% with non-i.i.d. data and 1.43% with i.i.d. data. Besides, regarding to accuracy, the difference between i.i.d. and non-i.i.d. settings is 2.47%, which also demonstrates the effectiveness of FL for dealing with non-i.i.d. data.

In addition, we explore the influence of different local update iteration on accuracy and loss. The concrete illustrations are shown in Figs. 6, 7, 8 and 9, respectively. Figs. 6 and 7 are conducted under non-i.i.d. settings. And Figs. 8 and 9 are conducted under i.i.d. settings. Obviously, if the local update performs more rounds between the two global aggregation, the accuracy can be better and the loss is reduced more quickly. This is because, with more local iterations, the uploaded local models are more mature or well trained. In other words, the parameters are more suitable for the corresponding local data distributions. The benefits are continuously obtained by the global model via weighted

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.

average. Therefore, in the case of sufficient local computation resources, the more local rounds are calculated, the more accurate the model we can obtain in a shorter time. Besides, comparing the performance of non-i.i.d. setting with i.i.d. setting, we can see that accuracy is higher and the loss decreases faster in i.i.d. environment. This is because with same data distribution, the local models should have some features in common. Therefore, the performance of aggregated global model can be similar as centralized learning. However, in non-i.i.d. case, data distribution varies from user to user, which means one local model cannot be suitable for the others, resulting in the lower convergence.

V. Conclusion

In this paper, to tackle with the classification problem for high data rate and latency-sensitive AR applications, we introduce and analyze the mathematical model of AR in detail and connect the AR operating principles with the object detection and classification problem. Then, we proposed a framework that combines FL with MEC to address the corresponding challenges. To evaluate our method, we conduct the experiments among open source dataset CIFAR-10 in i.i.d. settings and non-i.i.d. settings, and also compare with centralized learning. Our proposed framework can bring fewer iteration rounds for model training, meanwhile only leads to 3.9% accuracy less with non-i.i.d. data and 1.43% accuracy less with i.i.d. data. In addition, we also explore the influence of local updates iterations on the accuracy and loss or convergence speed. As a consequence, we find that when more local updates are performed, we can get more accurate model and the loss decreases faster. Besides, training by i.i.d. data leads to better accuracy and faster convergence, comparing with non-i.i.d. case.

References

[1] R. Pascoal, B. Alturas, A. de Almeida, and R. Sofia, "A survey of augmented reality: Making technology acceptable in outdoor environments," in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, Caceres, Spain, June 2018.

[2] M. Jia and W. Liang, "Delay-sensitive multiplayer augmented reality game planning in mobile edge computing," in *21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Montreal, Canada, November 2018.

[3] C. V. N. Index, "Global mobile data traffic forecast update, 2016–2021 white paper," *Cisco: San Jose, CA*, 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourth quarter 2017.

[5] M. Billinghurst, A. Clark, G. Lee *et al.*, "A survey of augmented reality," *Foundations and Trends® in Human–Computer Interaction*, vol. 8, no. 2-3, pp. 73–272, March 2015.

[6] T. Kilgus, E. Heim, S. Haase, S. Prüfer, M. Müller, A. Seitel, M. Fangerau, T. Wiebe, J. Iszatt, H.-P. Schlemmer *et al.*, "Mobile markerless augmented reality and its application in forensic medicine," *International journal of computer assisted radiology and surgery*, vol. 10, no. 5, pp. 573–586, May 2015.

[7] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, August 2018.

[8] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11 098–11 112, November 2018.

[9] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Transactions on Mobile Computing*, to appear 2019.

[10] S. Siltanen, "Theory and applications of marker based augmented reality," Ph.D. dissertation, VTT Technical Research Centre of Finland, January 2012.

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

[12] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, to appear 2019.

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

[14] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, April 2017.

[15] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," in *the 31st Conference on Neural Information Processing Systems*, Long Beach, CA, December 2017.

[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

Authorized licensed use limited to: University of North Carolina at Charlotte. Downloaded on December 16,2020 at 07:03:56 UTC from IEEE Xplore. Restrictions apply.