Cognitive Service in Mobile Edge Computing

Chuntao Ding, Ao Zhou, Xiao Ma, Shangguang Wang*
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, China
Email: {ctding, aozhou, maxiao18, sgwang}@bupt.edu.cn

***Abstract*—Cognitive services have revolutionized the way we live, work and interact with the world. In recent years, deep neural networks have become the mainstream approach in cognitive service, and mobile edge computing facilitates a variety of cognitive services for users by offloading computation tasks from resource-limited mobile devices to relatively wealthy edge servers. Combining the two to provide users with a higher quality of cognitive service is an issue worth researching. However, many related studies are not easy to provide fast responses because in these systems, edge servers are only used to pre-process data, and the cloud server is used to perform tasks. In this paper, we aim to study deploying deep neural network models on edge servers to provide fast services. However, a single edge server collects only a small amount of data, which results in low inference accuracy. To address this problem, we propose a cloud and edge collaboration framework. The key idea of the proposed framework is to use a cloud model to assist in training a edge model to improve the latter's inference accuracy and enable the latter to provide fast response and high-performance cognitive service. Experimental results demonstrate the effectiveness of our proposed framework.**

***Keywords*-Cognitive service; mobile edge computing; deep neural networks; collaboration;**

I. Introduction

Ubiquitous mobile devices can be used to provide a variety of cognitive services [1] [1], [2]. For example, a wearable camera with the ability to recognize objects and understand the surrounding environment is helpful for the visually impaired [3]. On the other hand, deep neural network models have become the dominant approach because of their good performance in speech recognition, visual object recognition, object detection, and many other domains [4]. Due to limited computation and storage resources, it is challenging for mobile devices to provide cognitive services with long duration, fast responses and high inference accuracy. Many existing cognitive services often rely on remote cloud servers with powerful computing capabilities. However, cloud-based solutions may incur long transmission delays since cloud servers are usually far away from users.

Mobile edge computing (MEC) enables mobile devices to provide cognitive services with real-time response by providing computing and storage resources at the edge of the network [5]–[7]. In recent years, a lot of existing MEC-based cognitive services have been proposed [8]–[12]. However, it is uneasy for them to provide fast responses because in these systems, edge servers are only used to pre-process data, and the cloud server is used to perform tasks.

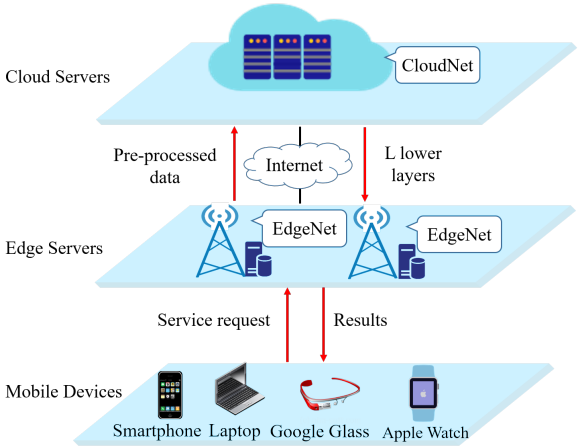


Figure 1. Architecture of the proposed framework.

In this paper, we propose a cloud and edge collaboration framework for cognitive services, as illustrated in Figure 1. In our framework, there are two neural network models, CloudNet and EdgeNet. CloudNet is a deep neural network model, and EdgeNet is a shallow neural network model. The shallow neural network means that the network has fewer layers. Inspired by recent advanced research [13], [14], using the lower layers of neural network model trained on other datasets to initialize new neural network models can improve their performance on new datasets, so our goal is to use CloudNet to assist in training EdgeNet and enables the latter to provides high quality of services. However, deep neural network models may leak information about the training data [15], [16] and cloud servers usually store large amounts of private data, such as personal images. Motivated by [13], we use the L (where L is a positive integer) lower layers of CloudNet to assist in training EdgeNet.

Furthermore, we consider that in real-world scenarios, users may continuously upload data. This motivated us to use the continuously uploaded data to further assist in training EdgeNet. However, the uploaded data is usually unlabeled, and training EdgeNet directly on edge servers with unlabeled data is a challenge. One naive solution

[1]https://azure.microsoft.com/en-us/services/cognitive-services/

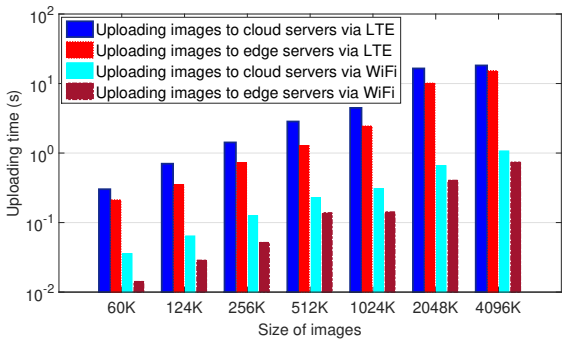


Figure 2. Uploading time in LTE and WiFi.

is to utilize manual and machine cooperation annotation techniques [17], [18] to get the labels of the data and then use them to assist in training EdgeNet on the edge server. However, the coverage of a single edge server is limited [19] and a small amount of data is collected, which is far from sufficient to significantly improve the inference accuracy of EdgeNet. To this end, we propose an adaptive algorithm. With the proposed adaptive algorithm, EdgeNet's inference accuracy can be further improved. Note that, CloudNet can be many other popular deep neural network models such as VGG [20] and ResNet [21]. Analogously, we can also design different EdgeNets on edge servers. In this sense, our framework is general and can be adapted to most existing neural network models. In summary, this paper makes the following three major contributions:

- We propose a cloud and edge collaboration framework that provides users with faster response, longer duration and high accuracy cognitive services.
- We propose an adaptive algorithm for EdgeNet, which can further improve its accuracy by using the continually uploaded data and CloudNet.
- We conduct extensive experiments on different datasets to evaluate the proposed framework. Results show that EdgeNet can provide a faster response and the proposed framework can improve EdgeNet's inference accuracy.

The following paper is organized as follows. Section II describes our motivation. Section III shows the detailed design of the proposed framework. Section IV provides an experimental evaluation. Section V reviews related work. Finally, we conclude and outline future work in Section VI.

II. Motivation

In this section, we implement a prototype system to quantify the fast response provided by MEC architecture. The experimental environment consists of three components: mobile device, edge server, and cloud server, and we build the system based on [22].

Figure 2 shows that in Long-Term Evolution (LTE), performing the recognition task on the edge server can reduce the average uploading time by 55.42%, and in WiFi, performing the recognition task on the edge server can reduce the average uploading time by 60.73%. This makes sense intuitively: users are closer to the edge server.

Figure 2 also shows that the very deep neural network model incurs long inference time. This is because deep neural network models require abstract images multiple times, and their inference procedures are complex. The more convolutional layers are, the longer the inference time is.

Therefore, in order to achieve a fast response of cognitive service, it is necessary to deploy shallow neural network models on the edge server. However, a single edge server collects only a small amount of data, which is not sufficient for training the neural network model to obtain good performance. This motivates us to study the use of limited training data to improve the inference accuracy of shallow neural network models.

III. Cloud and Edge Collaboration Framework

A. Overview

Figure 3 illustrates our framework, which consists of three layers of components, namely mobile devices, edge servers, and cloud servers.

In the proposed framework, these components work together. Mobile devices include smartphone, Laptop, Google Glass, Apple Watch, etc. They are responsible for lightweight tasks such as uploading data to and receiving results from edge servers. Edge servers are deployed at the edge of the network and are usually sensors, routers, and switches. In general, they are responsible for training EdgeNet and uploading pre-processed data to cloud servers. Cloud servers are regarded as having rich computation and storage resources and are responsible for training CloudNet and sending labels and partial layers to edge servers.

From Figure 3, the image recognition process is as follows. We first train CloudNet by using a large number of labeled images. Then, the cloud server sends the L lower layers of CloudNet to each edge server. After receiving the L lower layers, each edge server adds H layers to form the EdgeNet. Follow that, a small number of labeled images are used to train EdgeNet on the edge server. When training EdgeNet, L lower layers are frozen and H high layers are fine-tuned. By doing so, the EdgeNet can improve its inference accuracy. Furthermore, in real-world scenarios, users may continuously upload data to edge servers. To further improve the inference accuracy of the EdgeNet, we propose an adaptive algorithm by using the continuously uploaded data and CloudNet.

In the proposed framework, mobile devices only communicate with edge servers. When requesting a cognitive service, the device uploads data to the nearest edge server. After receiving the uploaded data, the edge server runs EdgeNet to obtain results and sends back the results to users. Note that, the interaction between edge servers and cloud servers is transparent to users.

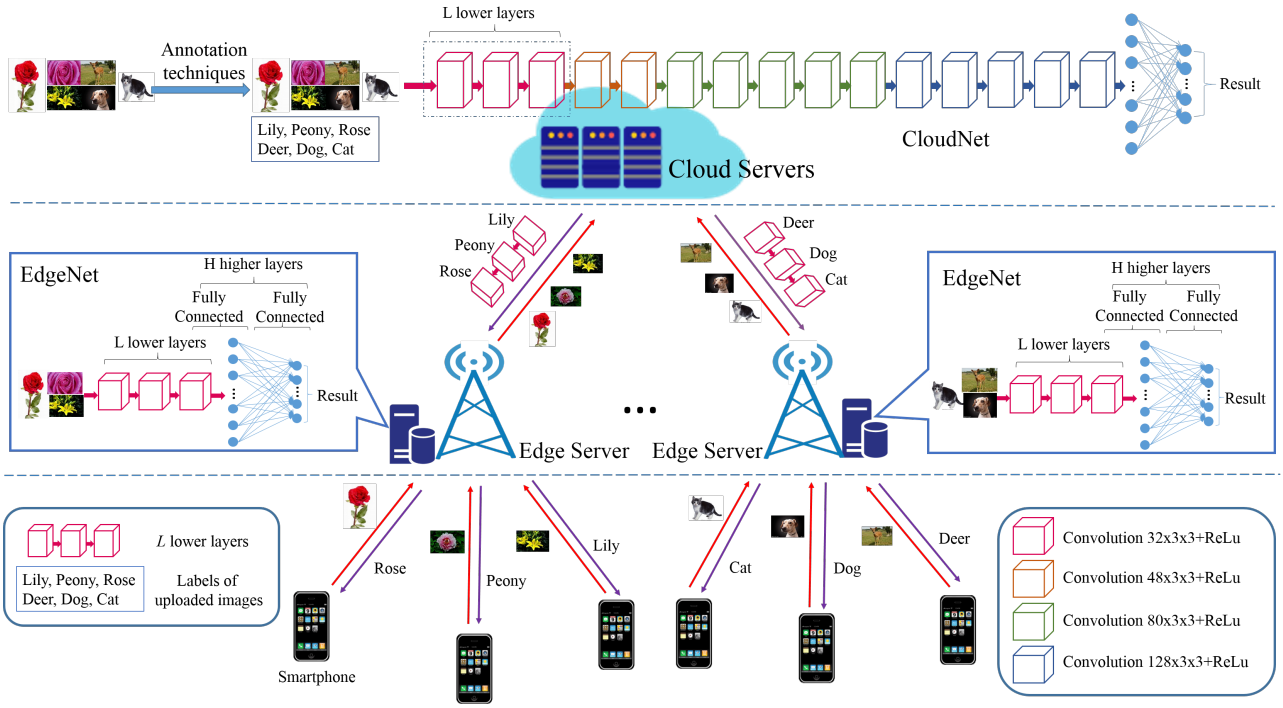


Figure 3. The detailed process of the proposed framework. When receiving the uploaded images (*e.g.,* Lily, Peony, Cat), the edge server pre-processes it and performs EdgeNet to obtain results and sends back the results to the user. In addition, the edge server saves the pre-processed images. When the core network is idle, the edge server uploads the preprocessed images to the cloud server. Then, the cloud server uses all the labeled images (including Lily, Peony, Cat, etc) to re-train CloudNet. Follow that, the cloud server sends the L lower layers of re-trained CloudNet and labels to each edge server. In the edge server, the L lower layers of the EdgeNet is replaced by the L lower layers of re-trained CloudNet.

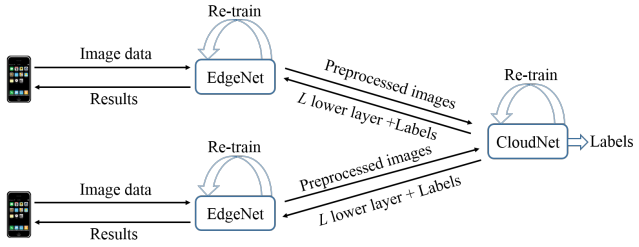


Figure 4. Adaptive phase.

B. An Adaptive Algorithm for EdgeNet

In the proposed framework, we use CloudNet to assist EdgeNet to improve the latter's inference accuracy. In addition, in real-world scenarios, users may continuously request cognitive services. In deep neural network models, available training data is extremely important [23]. According to [24], when the distribution of training data is the same as the distribution of test data, the probability of the test error distancing from the upper bound is given by

$$P(E_{test} \leq E_{train} + \sqrt{\frac{V(log(\frac{2N}{V})+1)-log(\frac{\eta}{4})}{N}}) = 1-\eta \quad , \quad s.t., V \ll N \tag{1}$$

where E_{test} is the test error and E_{train} is the training error. N is the size of the training set and V is the VC dimension of the classification model. η is a constant and $\eta \in [0,1]$. In addition, where

$$\sqrt{\frac{V(log(\frac{2N}{V})+1)-log(\frac{\eta}{4})}{N}} \quad , \tag{2}$$

is called model complexity penalty. From Eq. 2, the larger N is, the smaller the model complexity penalty is. Hence, to make the deep neural network model has good generalization, a large N is needed to depress the model complexity penalty.

However, users may continuously upload unlabeled images. It is challenging to use the unlabeled images to assist in training EdgeNet. A naive solution is first to get the labels of the uploaded data by utilizing manual and machine cooperation annotation techniques. Then, using the labeled images to assist in training EdgeNet on the edge server. However, a single edge server only collects little data. How to use the uploaded unlabeled data to improve the inference accuracy of EdgeNet is challenging.

To this end, we come up with using the CloudNet and uploaded data to assist in training EdgeNet together. This process is continual and illustrated in Figure 4. Users first upload data to the edge server. After receiving the uploaded data, the edge server first pre-processes it, such as object detection and object segmentation. Then, it saves the pre-processed data and sends it to the cloud server. After receiving the uploaded pre-processed data, the cloud server first utilizes annotation techniques to get the labels. Note

that, we assume that the annotation technique always marks the correct labels for new images, and we leave the error label situation for our future work. Then, the cloud server sends the labels to edge servers that stores corresponding data. We also use the labeled data to re-train CloudNet to further improve its generalization performance. After that, the cloud server sends the L lower layers of CloudNet to each edge server again. In each edge server, the L lower layers of EdgeNet is replaced by the recently shared L lower layers. Finally, we use the labeled data on the edge server to train EdgeNet by freezing L lower layers and fine-tuning H higher layers.

Formally, we assume that a small amount of labeled data is stored on the edge server and is represented as $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in X$ and $y_i \in Y$, N is the number of data, $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^c$, d is the dimension of x_i, c is the number of classes. We deploy EdgeNet to find a function $f(x;\alpha) \in \mathcal{F}$ that describes the relationship between X and Y, which follows the joint distribution $P(X,Y)$. $\mathcal{F}$ is a class of function from $\mathbb{R}^d$ to $\mathbb{R}^c$. The goal of function $f(x;\alpha)$ is to guarantee the smallest probability of incorrect classifications. To this end, we first define a loss function ℓ that penalizes the differences between predictions $f(x;\alpha)$ and actual targets y. Then, we minimize the average of the loss function ℓ over the data distribution P, also known as the expected risk:

$$\begin{aligned} R(f(x;\alpha)) &= \int \ell(f(x;\alpha), y) dP(x,y) \\ &= \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i;\alpha), y_i) \end{aligned} \quad , \tag{3}$$

where the loss function ℓ can be the cross entropy loss, *e.g.,* $\ell(y, \widehat{y}) = -\sum_{i=1}^{c} y_i log \widehat{y_i}$, where y_i is the true label of x_i, $\widehat{y_i} = f(x_i;\alpha)$. Our goal is to find a function $f(x,\alpha)$ that guarantees the smallest probability of incorrect classifications of EdgeNet with limited visual data.

Let $\mathbf{W}_c$ denote parameters of CloudNet, $\mathbf{W}_{lc}$ denote parameters of its pre-L lower layers, $\mathbf{W}_e$ denote parameters of EdgeNet, and $\mathbf{W}_{le}$ denote parameters of the pre-L lower layers of EdgeNet. We first train CloudNet with a large amount of labeled data $\{(x_i, y_i)\}_{i=1}^{M}$, where M is the number of data. The loss function is defined as follows:

$$R(f(x;\mathbf{W}_c)) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{H}(f(x_i;\mathbf{W}_c), y_i) \ . \tag{4}$$

Then, we send $\mathbf{W}_{lc}$ to each edge server. When receiving $\mathbf{W}_{lc}$, the edge server adds H layers based on the L lower layers to form the EdgeNet. In EdgeNet, we make $\mathbf{W}_{le} = \mathbf{W}_{lc}$, and the parameters of the H higher layers are randomly initialized. Follow that, we train EdgeNet with a small amount of labeled data $\{(x_i, y_i)\}_{i=1}^{N}$ by freezing L lower layers and fine-tuning H higher layers. We use Eq. 3 as the loss function and rewrite it as follows.

$$R(f(x;\mathbf{W}_e)) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{H}(f(x_i;\mathbf{W}_e), y_i) \ . \tag{5}$$

Algorithm 1: An adaptive algorithm for EdgeNet.

Input: $\mathbf{W}_c$, $\mathbf{W}_{lc}$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$, objects $\{x_i\}_{i=1}^{K}$, CloudNet
Output: EdgeNet

$\mathbf{W}_{le} \leftarrow \mathbf{W}_{lc}$;
Initialize $\mathbf{W}_{e-le}$ to small random values, where $\mathbf{W}_{e-le} = \mathbf{W}_e \setminus \mathbf{W}_{le}$;
$\mathbf{W}_{e-le} \leftarrow \arg\min_{\mathbf{W}_{e-le}} R(f(x;\mathbf{W}_{e-le}))$;
while $Iter$ **do**
  The edge server uploads data $\{x_i\}_{i=1}^{K}$ to the cloud server; After receiving $\{x_i\}_{i=1}^{K}$, the cloud server annotates their labels $\{y_i\}_{i=1}^{K}$ and sends $\{y_i\}_{i=1}^{K}$ to correspond edge servers;
  $\mathbf{W}_c \leftarrow \arg\min_{\mathbf{W}_c} R(f(x;\mathbf{W}_c))$ with $\{(x_i, y_i)\}_{i=1}^{M+K}$;
  $\mathbf{W}_{le} \leftarrow \mathbf{W}_{lc}$;
  $\mathbf{W}_{e-le} \leftarrow \arg\min_{\mathbf{W}_{e-le}} R(f(x;\mathbf{W}_{e-le}))$ with $\{(x_i, y_i)\}_{i=1}^{N+K}$;
  $\mathbf{W}_e \leftarrow \mathbf{W}_{le} \cup \mathbf{W}_{e-le}$;
  Obtain EdgeNet with parameter $\mathbf{W}_e$;
  if *EdgeNet's inference accuracy is increasing* **then**
    $Iter$=True
  else
    $Iter$=False
  end
end
return EdgeNet;

After receiving the uploaded data, the edge server first pre-processes and stores it. When the core network is idle, the edge server uploads it to the cloud server. After receiving the pre-processed data, we first annotate it and send its labels to corresponding edge servers.

For ease of explanation, we assume that one cloud server corresponds to one edge server. Assume that when saving K pre-processed images, the edge server uploads them to the cloud server. When receiving the K pre-processed images, we first annotate them and send their labels to the edge server. Then, we use the dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{M+K}$ to re-train CloudNet. Follow that, the cloud server sends the $\mathbf{W}_{lc}$ to the edge server and makes $\mathbf{W}_{le} = \mathbf{W}_{lc}$. Finally, we use the dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N+K}$ to re-train EdgeNet. When re-retraining EdgeNet, we freeze L lower layers and fine-tune H higher layers. The re-training process of CloudNet and EdgeNet continues until the inference accuracy of EdgeNet is no longer increased. Therefore, The upload data and CloudNet can assist in training EdgeNet to improve its inference accuracy. Note that, in each iteration, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{K}$ is a new dataset, and the dataset in the edge server and cloud server adds K samples. The detailed adaptive process is given in Algorithm 1.

It's worth noting that there are two ways to train EdgeNet.

Table I
CLOUDNET MODELS WITH DIFFERENT NUMBER OF TRAINING IMAGES

Model	Number of Training images		
	FASHION-MNIST	CIFAR-10	CIFAR-100
CloudNet1	12,000 (89.32%)	10,000 (77.83%)	10,000 (30.64%)
CloudNet2	36,000 (92.90%)	30,000 (85.24%)	30,000(49.84%)
CloudNet3	60,000 (93.32%)	50,000 (88.70%)	50,000 (60.43%)

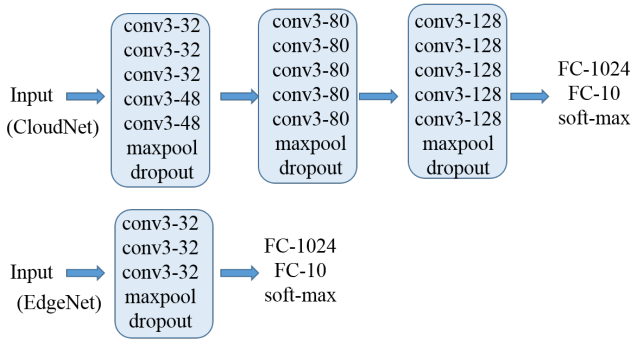


Figure 5. The detailed structure of CloudNet and EdgeNet. The conv3-32 indicates that the filter size is 3×3 and the number of channels is 32.

The first way is to freeze the L lower layers and fine-tune the H higher layers of EdgeNet by using the labeled images stored on edge servers. The second way is to fine-tune all parameters, *i.e.,* a complete EdgeNet. In this paper, we adopt the first way because the second way consumes more computing and storage resources. Moreover, two ways have similar inference accuracy. Later experimental results show that the difference between the inference accuracy of two ways is relatively minor.

IV. EXPERIMENTS

In this section, we present our experimental results on three datasets, FASHION-MNIST, CIFAR-10 and CIFAR-100. In addition, we also evaluate the training time of EdgeNet.

A. Datasets

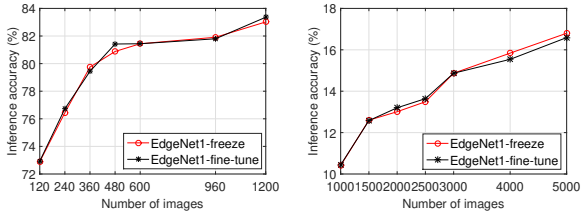
Three public datasets are briefly summarized as follows.

FASHION-MNIST [25]. It contains 70K fashion products from 10 classes, with 7K images per class. The training set has 60K images and the test set has 10K images.

CIFAR-10 and CIFAR-100 [26]. Both contain 50K training images and 10K testing images. The CIFAR-10 dataset has 6000 images of each of 10 classes and the CIFAR-100 dataset has 600 images of each of 100 classes.

B. Experimental Setup

We first train three CloudNets: CloudNet1, CloudNet2, and CloudNet3 have the same architecture, but with different training images. Their inference accuracy is shown in Table I. Then, we denote the shallow neural network model trained from scratch as EdgeNet0; assisted by CloudNet1 is denoted as EdgeNet1; assisted by CloudNet2 is denoted as EdgeNet2; and assisted by CloudNet3 is denoted as EdgeNet3.



(a) FASHION-MNIST (b) CIFAR-100 dataset

Figure 6. Inference accuracy of EdgeNet1-freeze and EdgeNet1-fine-tune. Note that, EdgeNet1-freeze refers to freezing the L lower layers and fine-tune the H higher layers of EdgeNet1; EdgeNet1-fine-tune refers to fine-tuning all layers of EdgeNet1.

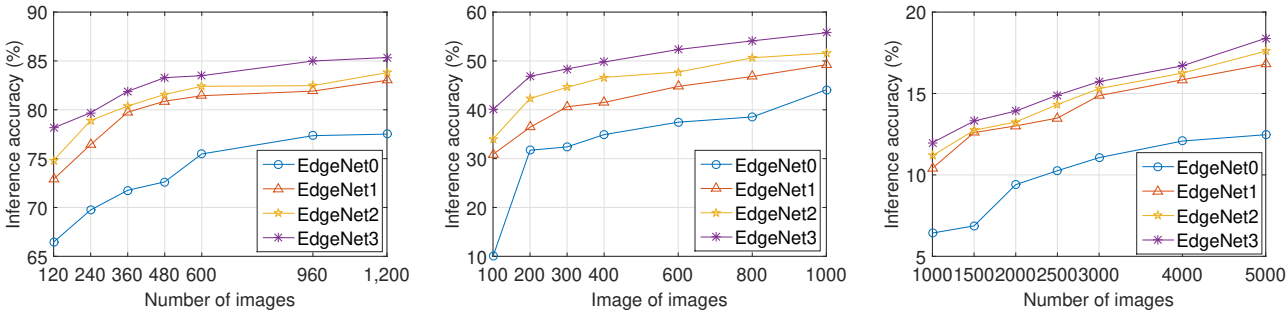
Figure 5 indicates the detailed structure of CloudNet and EdgeNet. They consist of four types of conv layers, some of which are followed by max-pooling layers, and one fully-connected layer with a final 10-way softmax (100-way softmax when handling CIFAR-100 dataset). We use one type of kernel with the size of 3×3 [20]. The fully-connected layers have 1024 neurons. The same type of conv layers are connected to one another without any intervening pooling or normalization layers. Max-pooling is performed over a 2×2 pixel window, with stride 1. The dropout ratio is set to 0.25. The EdgeNet consists of 3 first-type conv layers of CloudNet. The EdgeNet architecture ends with a max-pooling layer and a fully-connected layer with a final 10-way softmax. The fully-connected layer has 1024 neurons. The max-pooling, dropout ratio and learning rate are same as CloudNet. In CloudNet and EdgeNet, all conv layers are equipped with the ReLU.

The training procedures of CloudNets and EdgeNets follow [20], [21]. We use mini-batch gradient descent to train CloudNets and EdgeNets models by optimising the multinomial logistic regression objective. We set the batch size to 32 and the momentum to 0.9. In addition, we set the learning rate initially to 0.01, and then reduce the learning rate by a factor of 10 when the validation setting accuracy stops improving.

C. Experimental Results

1) Effects of freezing and fine-tuning on EdgeNet: Figure 6 (a) shows that on the FASHION-MNIST dataset, the maximum difference in inference accuracy between EdgeNet1-freeze and EdgeNet1-fine-tune is 0.54%. Figure 6 (b) shows that on the CIFAR-100 dataset, the maximum difference in inference accuracy between EdgeNet1-freeze and EdgeNet1-fine-tune is 0.21%. The reason is that these labeled images have been used to train CloudNet, and the 3 lower layer parameters of EdgeNet1 have inherited the knowledge that CloudNet learned with these labeled images. Therefore, they have similar inference accuracy.

2) Inference Accuracy of EdgeNet: Figure 7 shows that on three datasets, EdgeNet2 is more accurate than EdgeNet1 in different training images. In addition, EdgeNet3 is more



(a) FASHION-MNIST dataset (b) CIFAR-10 dataset (c) CIFAR-100 dataset

Figure 7. Inference accuracy of EdgeNet0 vs EdgeNet with our framework.

accurate than EdgeNet2. To be exact, Figure 7 (b) shows that compared with EdgeNet1, when the number of trainable images is 100, the inference accuracy of EdgeNet2 increased by 10.24%, the inference accuracy of EdgeNet3 increased by 17.72%. Similar results can also be found in Figures 7 (a) and (c). The EdgeNet3 is more accurate than EdgeNet2. This is because CloudNet3 sends more available knowledge to EdgeNe3. Other similar reason is that with the accumulation of images on cloud servers, features extracted from the three lower layers are more general. Hence, EdgeNet can achieve higher inference accuracy.

Figure 7 (b) shows that compared with EdgeNet0, on CIFAR-10 dataset, the inference accuracy of EdgeNet1 can be increased by 208.7%. Figure 7 (c) shows that compared with EdgeNet0, on CIFAR-100 dataset, the inference accuracy of EdgeNet1 can be increased by 56.04%. This is because EdgeNet0 has difficulty learning the distribution of training images when there are a small number of training images, resulting in low inference accuracy. However, CloudNet1 is trained with a large number of training images to learn the distribution of image data. This indicates that CloudNet1 can well learn the distribution of training images and can help EdgeNet1 learn the true distribution by sharing knowledge (3 lower layers) to EdgeNet1. Hence, CloudNet's assistance is beneficial for EdgeNet, especially when there is a small number of trainable images.

Figure 7 (b) shows that on the CIFAR-10 dataset, when the number of training images increased from 100 to 1000, EdgeNet0's inference accuracy increased by 340.09%, and EdgeNet3's inference accuracy increased by 39.22%. The reason is that a large amount of training image data helps to reveal the true distribution of image data. The experimental results also show that it is important to collect enough data to train neural network models in real-world services.

In addition, we also evaluate the inference accuracy of EdgeNets as the number of training data increased. Table II shows that when the number of training images is 60,000, the inference accuracy of EdgeNet3 is 93.30%, which is similar to the inference accuracy of CloudNet3 (*i.e.,* 93.32%). The

Table II
INFERENCE ACCURACY ON FASION-MNIST DATASET

Number of training images	EdgeNet1	EdgeNet2	EdgeNet3
6000	89.40	90.36	90.77
12000	90.55	91.22	91.30
18000	91.09	91.97	92.14
24000	91.45	91.91	91.98
30000	92.19	92.27	92.82
36000	92.27	92.42	92.83
42000	92.67	92.68	92.83
48000	92.51	92.61	93.11
54000	92.82	92.95	93.12
60000	93.09	93.15	93.30

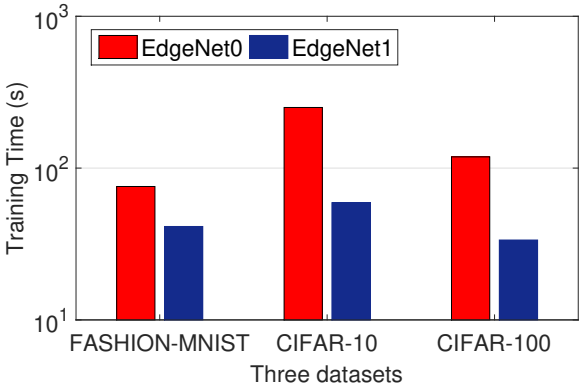


Figure 8. Training time in different datasets. Note that, there are 120 training images on FASHION-MNIST, 100 training images on the CIFAR-10 dataset, and 1000 training images on CIFAR-1000 dataset.

results show that with the continuous assistance of CloudNet, EdgeNet can learn how to extract effective features from CloudNet. The results also indicate that using EdgeNet can provide a faster response without losing too much accuracy.

3) Training Time of EdgeNet: Figure 8 shows that on the CIFAR-10 dataset, compared with training EdgeNet from scratch, the training time of training EdgeNet with the help of CloudNet can be reduced by 76.40%. This is because

with the guidance of the CloudNet, EdgeNet can easy to learn the distribution of training data.

Above results show that CloudNet can improve the inference accuracy of EdgeNet by sharing partial layers to assist in training EdgeNet. Moreover, the inference accuracy of EdgeNet can be further improved with the assistance of uploaded data and CloudNet. Therefore, our framework enables mobile devices to provide services with long duration, fast response and high inference accuracy.

V. Related Work

Existing work on cognitive services can be divided into three categories: cloud-based cognitive services, device-based cognitive services and edge-based cognitive services.

Cloud-based cognitive services is to run deep neural network models on remote cloud servers [27]–[29]. For example, Gabriel combines the first-person image capture and sensing capabilities of glass with remote processing to perform real-time scene interpretation [28]. MCDNN executes deep neural network models across mobile devices and cloud servers [29]. Although these approaches can achieve high inference accuracy, it causes long transmission delays because the cloud server is far away from users. If hundreds of millions of users upload data to cloud servers at the same time, the core network will be overloaded, resulting in longer transmission delays and even network congestion.

Device-based cognitive service is to run compressed neural network network models on mobile devices [3], [30], [31]. A popular approach is to use compression techniques [32] to compress deep neural network models to reduce its resource demands at the expense of inference accuracy. For example, MobileNets uses depth-wise separable convolutions to build light weight deep neural networks [30]. NestDNN uses compression techniques to enable resource-aware multi-tenant on-device deep learning [3]. However, deploying compressed deep neural network models on mobile devices will reduce the time for mobile devices to provide services. In addition, the deployed deep neural network models are static and not easily adaptable.

Edge-based cognitive service refers to the provision of services based on the MEC architecture. For example, *Drolia et al.* [8] modeled edge servers as caches for compute-intensive recognition applications. *Hu et al.* [9] proposed a face identification and resolution scheme based on fog computing. *Li et al.* [10] used the first few layers of the DNNs as a feature extractor to reduce the amount of network traffic that edge servers upload to cloud servers. *Liu et al.* [11] proposed a food recognition system based edge computing, which first pre-processes the captured image data on edge servers. Then, edge servers upload the pre-processed image data to cloud servers. However, it is not easy for them to provide fast responses, because in these systems, edge servers are only used to pre-process data, and the cloud server is used to perform tasks.

VI. Conclusion

In this paper, we propose a cloud and edge collaboration framework that provides cognitive services with high QoS. In addition, we propose an algorithm to make EdgeNet adaptive. EdgeNet's inference accuracy can be improved with ClouеNet's shared layers and can be further improved by continuing to use the uploaded data and CloudNet to assist with training. Note that, our framework is general and can be adapted to most high-performance deep neural network models. Experimental results demonstrate the effectiveness of our framework and adaptive algorithm.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (2018YFE0205503), the Funds for Creative Research Groups of China (61921003), and the National Natural Science Foundation of China (61922017).

References

[1] Q. Zhu, A. Zhou, Q. Sun, S. Wang, and F. Yang, "Fmsr: A fariness-aware mobile service recommendation method," in *Proceedings of the IEEE International Conference on Web Services*, 2018, pp. 171–178.

[2] C. Zhang, B. Liu, J. Y. L. Li, D. Zhang, X. Rui, and R. Bie, "Hybrid measurement of air quality as a mobile service: An image based approach," in *Proceedings of the IEEE International Conference on Web Services*, 2017, pp. 853–856.

[3] B. Fang, X. Zeng, and M. Zhang, "Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, Oct. 2018, pp. 115–127.

[4] L. Lin, K. Wang, W. Zuo, M. Wang, J. Luo, and L. Zhang, "A deep structured model with radius-margin bound for 3d human activity recognition," *International Journal of Computer Vision*, vol. 118, no. 2, pp. 256–273, 2016.

[5] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proceedings of the IEEE Conference on Computer Communications*, Apr. 2018, pp. 468–476.

[6] H. Wu, S. Deng, W. Li, J. Yin, X. Li, Z. Feng, and A. Y. Zomaya, "Mobile-aware service selection in mobile edge computing systems," in *Proceedings of the IEEE International Conference on Web Services*, 2017, pp. 201–208.

[7] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[8] U. Drolia, katherine Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Pull-in time dynamics as a measure of absolute pressure," in *Proceedings of the IEEE International Conference on Distributed Computing Systems*, Jun. 2017, pp. 276–286.

[9] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1910–1920, 2017.

[10] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[11] C. Liu, Y. Cao, G. Chen, V. Vokkarane, Y. Ma, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2018.

[12] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. B. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proceedings of the IEEE Symposium on Computers and Communications*, Jul. 2012, pp. 59–66.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the International Conference in Neural Information Processing Systems*, Dec. 2014, pp. 3320–3328.

[14] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning*, Jul. 2015, pp. 97–105.

[15] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2017, pp. 587–601.

[16] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani, and D. Vitali, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[17] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, "Extreme clicking for efficient object annotation," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2017, pp. 4940–4949.

[18] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-rnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 4485–4493.

[19] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proceedings of the IEEE Conference on Computer Communications*, Jun. 2018, pp. 207–215.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representation*, May 2015, pp. 1–14.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[22] S. Wang, C. Ding, N. Zhang, X. Liu, A. Zhou, J. Cao, and X. S. Shen, "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2019.

[23] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2017, pp. 843–852.

[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Heidelberg: Springer-Verlag Berlin, 1995.

[25] "The Fashion MNIST Dataset," https://www.kaggle.com/zalando-research/fashionmnist, [Online; accessed 18-June-2019].

[26] "The CIFAR10 and CIFAR100 datasets," https://www.cs.toronto.edu/~kriz/cifar.html, [Online; accessed 18-June-2019].

[27] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, applications, and services*, Jun. 2010, pp. 49–62.

[28] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th annual international conference on Mobile Systems*, Jun. 2014, pp. 68–81.

[29] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems*, Jun. 2016, pp. 123–136.

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *arxiv:1704.04861*, Apr. 2017, pp. 1–9.

[31] L. N. Huynh, Y. Lee, and R. K. Balan, "Deepmon: Mobile gpu-based deep learning framework for continuous vision applications," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, Jun. 2017, pp. 82–95.

[32] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *CoRR abs/1503.02531*, Mar. 2015, pp. 1–9.