Industrial Cyber-Physical Systems-Based Cloud IoT Edge for Federated Heterogeneous Distillation

Chengjia Wang, *Member, IEEE*, Guang Yang, *Member, IEEE*, Giorgos Papanastasiou, Heye Zhang, *Member, IEEE*, Joel J. P. C. Rodrigues, *Fellow, IEEE*, and Victor Hugo C. de Albuquerque, *Senior Member, IEEE*

***Abstract*—Deep convoloutional networks have been widely deployed in modern cyber–physical systems performing different visual classification tasks. As the fog and edge devices have different computing capacity and perform different subtasks, models trained for one device may not be deployable on another. Knowledge distillation technique can effectively compress well trained convolutional neural networks into light-weight models suitable to different devices. However, due to privacy issue and transmission cost, manually annotated data for training the deep learning models are usually gradually collected and archived in different sites. Simply training a model on powerful cloud servers and compressing them for particular edge devices failed to use the distributed data stored at different sites. This offline training approach is also inefficient to deal with new data collected from the edge devices. To overcome these obstacles, in this article, we propose the heterogeneous brain storming (HBS) method for object recognition tasks in real-world Internet of Things (IoT) scenarios. Our method enables flexible bidirectional federated learning of heterogeneous models trained on distributed datasets with a new "brain storming" mechanism and optimizable temperature parameters. In our comparison experiments, this HBS method outperformed multiple state-of-the-art single-model compression methods, as well as the newest multinetwork knowledge distillation methods with both homogeneous and heterogeneous classifiers. The ablation experiment results proved that the trainable temperature parameter into the conventional knowledge distillation loss can effectively ease the learning process of student networks in different methods. To the best of authors' knowledge, this is the first IoT-oriented method that allows asynchronous bidirectional heterogeneous knowledge distillation in deep networks.**

***Index Terms*—Deep learning, heterogeneous classifiers, Internet of Things (IoT), knowledge distillation (KD), online learning.**

Manuscript received April 7, 2020; revised June 12, 2020; accepted June 30, 2020. Date of publication July 7, 2020; date of current version May 3, 2021. This work was supported by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/EEA/50008/2020, in part by the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant 309335/2017-5, Grant 304315/2017-6, and Grant 430274/2018-1, in part by the Key Program for International Cooperation Projects of Guangdong Province under Grant 2018A050506031, in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B010110001, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2019B151502031, and in part by the Fundamental Research Funds for the Central Universities under Grant 19LGZD36. Paper no. TII-20-1728. *(Corresponding author: Heye Zhang.)*

Chengjia Wang is with the BHF Centre for Cardiovascular Science, The University of Edinburgh, EH16 4TJ Edinburgh, U.K. (e-mail: chengjia.wang@ed.ac.uk).

Guang Yang is with the Cardiovascular Research Centre, Royal Brompton Hospital and National Heart and Lung Institute, Imperial College London, SW7 2AZ London, U.K. (e-mail: g.yang@imperial.ac.uk).

Giorgos Papanastasiou is with the Edinburgh Imaging Facility QMRI, The University of Edinburgh, EH16 4TJ Edinburgh, U.K. (e-mail: g.papanas@ed.ac.uk).

Heye Zhang is with the School of Biomedical Engineering, Sun Yat-Sen University, Guangzhou 510275, China (e-mail: heye.zhang@gmail.com).

Joel J. P. C. Rodrigues is with the Federal University of Piauí (UFPI), Teresina 64049-550, Brazil, and also with the Instituto de Telecomunicações, 1049-001 Lisboa, Portugal (e-mail: joeljr@ieee.org).

Victor Hugo C. de Albuquerque is with the University of Fortaleza, Fortaleza 60811-905, Brazil (e-mail: victor.albuquerque@unifor.br).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.3007407

I. Introduction

Deep learning has achieved tremendous successes in a wide range of visual applications [1]. Effective training of deep neural networks (DNN) often requires powerful computing hardware, availability of massive training data, and optimal hyper-parametric setups. These prerequisites limited the applicability of experimentally verified DNNs when being deployed into industrial Internet of Things (IoT). This article aims to overcome these difficulties when solving image classification tasks. Fig. 1 shows a conceptual demonstration of an example IoT system with these practical limitations.

First, most deep learning models with the reported state-of-the-arts performance are often trained on devices with sufficient computing power, such as a GPU-based cloud server. As shown in Fig. 1, adopting these pretrained models on fog/edge devices with diverse capacities requires to compress a large network into various light-weight models optimal for different devices, for example, deploying an face recognition model initialized on a GPU server to security cameras at different sites. Knowledge distillation [2] (KD) has been one of the most popular paradigms for robust and efficient model compression. The core idea of the original KD is to teach a student model using a soft categorical

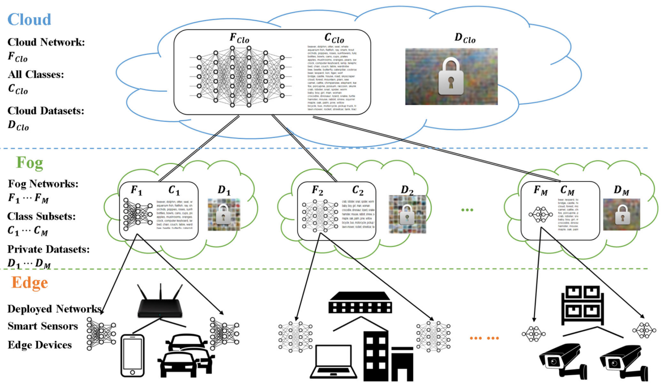

Fig. 1. General framework of the proposed online KD solutions in fog-based IoT: A large neural network that perform the complete classification task is trained in the cloud server. A fog server can train a light-weight model based on its computing power to perform a subtask. Each fog device stores its private datasets for the corresponding subtask (classifying a subset of the all categories).

labels produced by a "temperature" hyperparameter T. Softened labels revealed more information in the training dataset while reduce the difficulty of learning from hard one-hot labels for a smaller student model. However, the value of T has to be manually picked for different distillation tasks and methods. This is often impractical in an IoT that connect numerous fog and edge devices, and a new training strategy that can dynamically adjust the temperature to a specific scenario is required. In this article, we reformulated the original KD softmax function for a temperature that is dynamically adjustable during training.

Second, the customized DNNs running on different edge devices may performing different classification tasks with overlapped or unoverlapped categorizations of data (see Fig. 1). For example, in a smart home system, an outdoor surveillance device may be trained to recognize vehicles while ignoring the foods which is picked by an indoor camera, yet both may recognize human-beings. Here, we assume that the edge devices are grouped and managed by fog servers locally available at different sites, and each fog device is capable of training a light-weight fog network its own private data. In more extreme cases, different fog networks can be targeting at completely different sets of categories. This requires to distill the knowledge of a large teacher network trained on the cloud server to multiple heterogeneous fog models. In this work, we define this process as "cloud-to-fog" distillation [see Fig. 2(a)] and propose a new single-teacher-multistudent learning procedure to generate a set of optimal heterogeneous classifiers. The distributed fog models are then further trained for specific classification tasks on the edge devices. The proposed single teacher distillation method is evaluated in both heterogeneous and homogeneous distillation experiments, and outperformed most state-of-the-art baselines.

Another limitation of a real-world IoT application is, as shown in Fig. 1, the training data collected at one site are often unavailable to others due to privacy and legal issues, for example, data collected by smart home systems from different sites. To enable supervision with all training data distributed and online update of all networks, we introduce a "brain storming" mechanism that simultaneously ensemble the fog models and perform a "fog-to-cloud" distillation to update weights of the large cloud network as shown in Fig. 2(b). To further merge the gap between the theoretically designed deep learning models to practical IoT applications, we design the complete life cycle of the online KD in edge cloud industrial IoT. Experiment results show that the proposed multiteacher distillation methods achieved better results than state-of-the-art multinetwork ensemble algorithms. It also show better robustness with an online learning setup.

To sum up, there are three main limitations of applying deep learning models to a real-world cyber-physical system: diverse computing power, heterogeneous tasks, and decentralized storage and acquisition of training data. We model the practical image classification in cyber-physical IoT as an online learning problem of multiple heterogeneous classifiers, and solve it through a bidirectional online KD between the cloud and fog models. We name the proposed online KD method as heterogeneous brain storming (HBS). Contributions are summarized as follows.

1) We propose the HBS online KD method, which include a "cloud-to-fog" and a "fog-to-cloud" processes. HBS enables a complete online learning life cycle for IoT applications.
2) We convert the traditional temperature hyperparameter to a trainable parameter. Comparison experiments have verified that this parameter can dynamically adjust the softness of label during training. We also perform the experiment, for the first time, to find optimal temperature for different networks, tasks and methods.
3) We design a brain storming mechanism in the fog-to-cloud process to simultaneously unify and distill the knowledge from independent heterogeneous classifiers back to the cloud server, and we introduce a new combination of feature and logit distillation losses that achieved an outstanding performance in both heterogeneous and homogeneous distillation tasks.

II. Related Work

A. Knowledge Distillation

The idea of compressing ensemble models was first introduced by [3]. Then, Ba and Caruana [4] showed that information within one network can be transferred to a shallower one. The core idea of distilling the knowledge of a teacher network using soft labels was proposed by [2]. The soft labels are generated with logits distribution of the teacher smoothed by a temperature hyperparameter (5). This provides extra supervision for the student network to learn the generalizability of the teacher network [5], thus, achieve a better performance than training directly from the hard labels. Distillation process then can be implemented as minimization of the Kullback–Leibler (KL)-divergence or mutual information between the teacher and student logits distributions. Extensions of the original KD idea have been proposed to improve the teaching strategy. For example, self-distillation methods [6]–[8] and deep mutual learning (DML) train a student network that has a similar architecture with the teacher to ease the feature alignment in the latent space. Another type of methods learn an alternative representation from

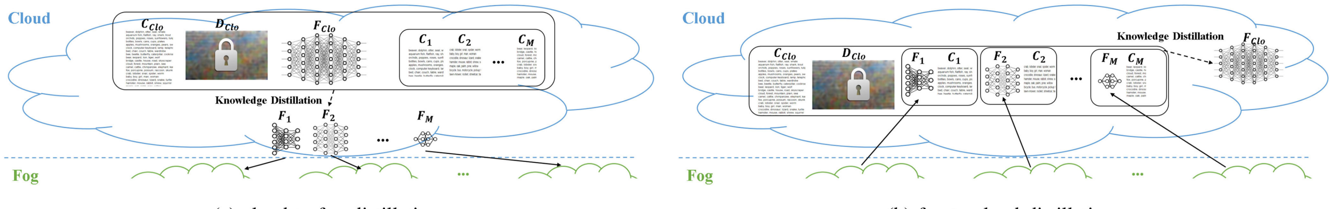

Fig. 2. The two distillation process: (a) The heterogeneous fog networks are generated by the cloud-tofogdistillation process, with the data stored on server side. After the networks are further trained with the private data stored at each sites, they are delivered back to server for a fog-to-cloud distillation as in (b).

the teacher network. These models distill the feature representations from the intermediate layers [9]–[13], or learn model the cross layer [14] or inter sample correlations [15], [16]. Most state-of-the-art methods are either unable to perform distillation between networks with very different architectures [6]–[8], [14], or suffers a significant performance drop as proved in [16]. The contrastive representation distillation (CRD) method [16] overcame this obstacle, but it has not been tested in cross-task distillation. Furthermore, all methods work with the logit-based KD losses [2] fix the temperature T to an empirically selected value. As a result, when training multiple students with diverse architectures, this T is impossible to be optimal for all the students.

B. Multiteacher KD

The key ability of the fog-to-cloud process is unifying the distributed heterogeneous knowledge from multiple teachers. A variety of multinetwork distillation methods have been proposed in the past two years. However, most of them requires the networks share the same architecture or performing the same classification task [8], [17], [18]. Or the teachers are modeled as different branches in a large network, which is difficult to be applied to IoT [5], [19], [20]. The BAM model [8], can perform "multi $\rightarrow$ single" distillation tasks, but places limitation on network structures. Similar limits can be found in recent classifier amalgamation works.[1] A few recent works [21]–[23] has been proposed to unifying heterogeneous teacher classifiers. Without a predefined *dustbin* class, Luo *et al.* [23] require overlapped classes of objects recognized by teacher models, otherwise the model failed to find an optimal feature alignments. Both [22] and [23] learns to extract common feature representation using additional knowledge amalgamation networks. This caused extra use of memory as the number of teachers increasing. Our HBS method removed these limits.

C. Data-Free KD

To deal with inaccessible private training data, data-free KD methods have been proposed. Early model compression methods directly merge similar neurons in the same layers [24]. The general strategy used in modern data-free distillation algorithms is to generate surrogate data based on the teacher network for the training of the student network. Lopez *et al.* [25] generate the surrogate data using statistical metadata of the activation in the teacher network. Nayak *et al.* [26] propose to generate training data for student using data impressions by modeling the softmax space using Dirichlet distribution. Recently proposed methods directly generate synthesized training data for the student network using adversarial learning [27]–[30] or deepdream data propagation [31], [32]. Although these algorithms can train the student networks without any available training data of the teacher, none has reported performance comparable to supervised methods. We consider a more common situation in real-world cyber–physical systems where the training data are stored on multiple devices.

[1][Online]. Available: https://github.com/zju-vipa/KamalEngine

III. Distillation in Fog-Based IoT

We define the training of multiple networks in visual IoT as an decentralized learning problem. As shown in Fig. 1, let $\{f_i\}_{i=1}^M$ be a set of deep convolutional neural networks (DCNN) to be deployed to different edge devices managed by M sites. The network f_i can be trained or finetuned with its own private dataset $\mathcal{D}_i$ to predict the label $\mathbf{y} \in \mathcal{C}_i$ of a input image $\mathbf{x}$, where the set $\mathcal{C}_i$ is the object categories targeted by the network f_i. Specifically, $\mathbf{q} = f_i(\mathbf{x})$ is the predicted probability vector output by network f_i. On a cloud server with sufficient computing power, a large network f_{clo} is trained with the cloud-side dataset $\mathcal{D}_{\text{clo}}$ to predict a label $\mathbf{y} \in \mathcal{C}_{\text{clo}}$, where $\mathcal{C}_{\text{clo}} = \bigcup_{i=1}^M \mathcal{C}_i = \{c_1, c_2, \ldots, c_K\}$ represents all categories of objects classified by all edge devices, in total of K classes. Note that all the networks $\{f_i\}$ can have different architecture and might be trained for heterogeneous subtasks, i.e., $\mathcal{C}_i \neq \mathcal{C}_j$ or even $|\mathcal{C}_i| \neq |\mathcal{C}_j|$ for $i \neq j$. For a input x not in the target categories $\mathcal{C}_i$, i.e., $x \notin \mathcal{C}_i$, we let the network f_i classify it into a *dustbin class* belongs to $\mathcal{C}_{-i}$. Let K_i the size of set $\mathcal{C}_i$ ($K_i = |\mathcal{C}_i|$), each fog network f_i performs a $K_i + 1$-way classification subtask.[2] The purpose of distillation in fog-based IoT is to realize online update of $\{f_i\}$ and f_{clo} using $\{D_i\}$ and D_{clo} without transferring these data between devices.

A. Classification and Conventional Distillation Losses

A classifier f_i, parameterized by θ_i is typically trained by minimizing the cross-entropy loss between the predicted probability vector $\mathbf{q}_i$ and the ground-truth one-hot label $\mathbf{y}$ encoded in one-hot representation, $\arg\min - \sum y \log q_{i,c_k}$. With the

[2]We call the K-way classification performed by the cloud network f_{clo} the "full classification task," where $K = |\mathcal{C}_{\text{clo}}|$.

dustbin classes $\mathcal{C}_{-i}$, the classification loss can be defined as

$$\mathcal{L}_{\text{cla}_i} = -\sum_{y_j \in \mathcal{C}_i} y_j \log q_{i,c_k} - \sum_{y_j \notin \mathcal{C}_i} y_j \log q_i(y \in \mathcal{C}_{-i}) \quad (1)$$

where q_{i,c_k} is the short hand of the probability $q_i(y = c_k, c_k \in \mathcal{C}_i)$ for brevity. For convenient computation of the losses between two heterogeneous classifiers, we estimate a homogeneous probability vector $\bar{\mathbf{q}}_i$, so that

$$\bar{q}_i(y = c_k) = \begin{cases} q_i(y = c_k, c_k \in \mathcal{C}_i) & c_k \in \mathcal{C}_i \\ q_i(y \in \mathcal{C}_{-1})/|\mathcal{C}_{-i}| & c_k \in \mathcal{C}_{-i} \end{cases} \quad (2)$$

where $\mathcal{L}_{\text{cla}_i}$ can be rewritten as

$$\mathcal{L}_{\text{cla}_i} = -\sum_{y_j \in \mathcal{C}_{\text{clo}}} y_j \log \bar{q}_i(y = c_k). \quad (3)$$

Similarly, the traditional KD is achieved by minimizing the cross-entropy loss between the probability distributions predicted by a well trained teacher network and a student network. For example, when distilling the knowledge of the trained f_{clo} to a f_i, the distillation loss is

$$\begin{aligned}\mathcal{L}_{\text{dis}_i} &= -\sum_{c_k \in \mathcal{C}_i} q_{\text{clo},c_k} \log q_{i,c_k} - \left(\sum_{c_k \in \mathcal{C}_{-i}} q_{\text{clo},c_k}\right) \log q_{i,C_{-i}} \\ &= -\sum_{c_k \in \mathcal{C}_{\text{clo}}} q_{\text{clo},c_k} \log \bar{q}_{i,c_k} \end{aligned} \quad (4)$$

where $q_{\text{clo}}(c_k \in \mathcal{C}_{-i})$ is estimated by the sum of the dustbin probabilities. As discussed in [21], grouping classes in $\mathcal{C}_{-i}$ into a single dustbin classes imposes design constraint on f_i. We argue that the presence of dustbin facilitates efficient implementation of the distillation loss function. It is also critical for gaining optimal alignments of the distributions $\mathbf{q}_i$ and $\mathbf{q}_{\text{clo}}$ given by f_i and f_{clo}, especially when $\mathcal{C}_i \bigcap \mathcal{C}_k = \emptyset$ when $i \neq k$. For a network f, q is estimated as

$$q(y = c_k) = \frac{\exp(z_k/T)}{\sum_l \exp(z_l/T)} \quad (5)$$

where $z_k = f(\mathbf{x}; \boldsymbol{\theta})$ is logits of class c_k output by f. The temperature hyperparameter T controls the softness of the predicted labels. Larger T results in flatter probability distributions.

B. Training Heterogeneous Models in IoT

Our approach to tackle the online training of the heterogeneous networks with distributed private data involves four steps: i) pretraining the cloud network f_{clo} using dataset $\mathcal{D}_{\text{clo}}$, and ii) distilling the knowledge of f_{clo} to the set of fog networks $\{f_i\}_{i=1}^{M}$. After iii) deploying and finetuning $\{f_i\}$ on the fog devices, iv) collect them back to the cloud and distill the updated knowledge back to f_{clo}. This process is repeated in the visual IoT system especially when new training data added to any of the cloud and fog devices. Step i) and iii) can be done following conventional network training and finetuning procedure and will not be discussed in this article. Our method focus on the critical step (ii), which use a large well trained network to teach a group of heterogeneous classifiers, and step (iv) which distills the knowledge from these classifiers back to a single network. Details of these two steps are presented in Sections IV and V. We name these two steps as *Cloud-to-fog distillation* and *fog-to-cloud distillation* that can be separately evaluated by independent experiments. Details of these two steps are shown in Fig. 3.

IV. Cloud-to-Fog Distillation

A. Trainable Temperature T

The temperature T in (5) plays an important role in KD. Previous methods manually pick it as an hyperparameter. It has been shown that models with different architecture have different optimal Ts, for example, smaller student networks should learn from flatter teacher distributions given by larger T [2]. As the cloud-to-fog distillation aims to produce multiple heterogeneous models simultaneously, it is difficult to select optimal T for each network. To enable automatic optimization of temperature, we modify (5)

$$q = \frac{\exp(\exp(\beta) z_k)}{\sum_l \exp(\exp(\beta z)_l)} \quad (6)$$

where $\bar{q}$ is estimated using (2), β is a trainable parameter so that $T = \exp(-\beta)$, $\beta \in [-\infty, +\infty]$, and $T > 0$. In the backpropagation process, gradients of the cross-entropy losses $\mathcal{L}_{\text{cla}}$ and $\mathcal{L}_{\text{dis}}$ shown in (1) and (4) can be easily computed.

B. Empirical Risk

For each fog network, minimizing the cross-entropy loss $\mathcal{L}_{\text{dis}_i}$ leads to larger temperature. To balance this effect, we propose to add another softmax cross-entropy loss to compute the empirical risk. With a batch of input images $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_B\}$, we calculate logits similarity between the outputs of a student f_i and a teacher f_j as

$$S(f_i, f_j) = \text{softmax}^\top \left(\frac{f_i(\mathbf{x}_l)}{T_i}\right) \text{softmax}\left(\frac{f_j(\mathbf{x}_k)}{T_i}\right) \quad (7)$$

where $f(\mathbf{x})$ represents logits output by a normalization layer of the network. The empirical risk loss is then computed as

$$\begin{aligned}\mathcal{L}_{\text{ER}_i} = &\sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}} \\ y_l = y_k, l \neq k}} -\log \frac{S(f_i, f_i)}{S(f_i, f_i) + \sum_{y_l \neq y_k} S(f_i, f_i)} \\ &+ \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}} \\ y_l = y_k}} -\log \frac{S(f_i, f_{\text{clo}})}{S(f_i, f_{\text{clo}}) + \sum_{y_l \neq y_k} S(f_i, f_{\text{clo}})}.\end{aligned} \quad (8)$$

The first term in (8) measures the internal empirical risk of the student network f_i, and the second term measures the cross-model risk between f_i and f_{clo}. The empirical risk loss is closely related to the contrastive loss used in [16] and the validation empirical loss in [18]. However, the contrastive loss [16] requires to reindex the training dataset and [18] just measure the internal empirical risk using the logits. We use the batch data distribution and the softmax output of the networks without prior knowledge of the dataset size.

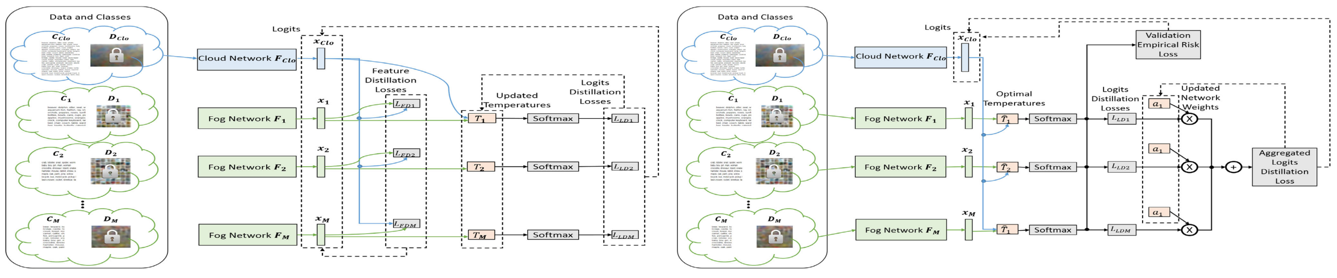
(a) (b)

Fig. 3. Training of the HBS method. (a) Cloud-to-fog distillation. (b) Fog-to-cloud distillation. (best viewed in color)

Follow our setup, the empirical risk loss can be efficiently computed. Let $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_B\}$ the one-hot ground truths labels of the input batch. We define a positive-pair indication matrix $\mathbf{G} = \mathbf{Y}^\top\mathbf{Y}$, where for a positive pair of input $(\mathbf{x}_l, \mathbf{x}_k)$, $G_{lk} = 1$ otherwise $G_{lk} = 0$. In the meantime, for two networks f_i and f_j, we define the model covariance matrix $\mathbf{F}_{ij} = \text{softmax}^\top(f_i(\mathbf{X})/T_i)\text{softmax}(f_j(\mathbf{X})/T_i)$. Then, the sum of negative pairs $\sum_{y_l \neq y_k} S(f_i, f_j)$ can be computed as

$$SN_{i,j} = \|(\mathbf{1} - \mathbf{G}) \odot \mathbf{F}_{ij}\|_1 \tag{9}$$

where $\|\cdot\|_1$ represents the first-order norm. Similarity, $\sum_{y_l \neq y_k} S(f_i, f_i)$ can be computed as

$$SN_{i,i} = \|(\mathbf{1} - \mathbf{G}) \odot \mathbf{F}_{ii}\|_1. \tag{10}$$

As $SN_{i,i}$ and $SN_{i,j}$ are computed for each negative pairs, (8) can be rewritten as

$$\mathcal{L}_{\text{ER}_i} = \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}} \\ y_l = y_k, l\neq k}} -\log \frac{S(f_i, f_i)}{S(f_i, f_i) + SN_{i,i}} + \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}} \\ y_l = y_k}} -\log \frac{S(f_i, f_{\text{clo}})}{S(f_i, f_{\text{clo}}) + SN_{i,\text{clo}}}. \tag{11}$$

The temperature T_i for the fog network f_i is trained by minimizing the logit distillation loss

$$\mathcal{L}_{\text{LD}_i} = \mathcal{L}_{\text{cla}_i} + \lambda_1 \mathcal{L}_{\text{dis}_i} + \lambda_2 \mathcal{L}_{\text{ER}_i}. \tag{12}$$

In our experiments, we empirically set $\lambda_1 = \lambda_2 = 1$. Note that as different fog networks may group different subsets of $\mathcal{C}$ into the dustbin, $\mathbf{F}$ are for fog networks are calculated based on the estimated probability $\bar{\mathbf{q}}$ given in (2).

C. Feature Distillation Loss

Heterogeneous distillation with distributed data involves training models with different lengths of logit features. However, computing the covariance matrix $\mathcal{F}_{ii}$ provide a nonparametric way to encode relational knowledge structure into feature spaces with the same dimensions. We use a batch sample similarity loss similar to [18] by building up a similarity matrix $\mathbf{A} \in \mathbb{R}^{B\times B}$ for each f_i

$$\mathbf{A}_i = \text{softmax}^\top(f_i(\mathbf{X}))\text{softmax}(f_i(\mathbf{X})). \tag{13}$$

Unlike the method used in [18] where the logits $\mathbf{z}_i = f_i(\mathbf{x})$ is normalized by a restricted rectified linear unit (RELU) function. Here, we use a normal softmax to rescale the logits to get rid of the affection of the changing temperature T_i. Note that computation of $\mathbf{A}_i$ is different from $\mathbf{F}_i i$ as no temperature T and no adjustment of the feature matrix $f(\mathbf{X})$ are involved. Because the covariance matrix $\mathbf{A}_i$ is a symmetric positive definite (SPD) matrix lying on a Riemannian SPD manifold, the feature distillation loss is then computed in the log-Euclidean space [18]

$$\mathcal{L}_{\text{FD}_i} = \|\log(\mathbf{A}_{\text{clo}}) - \log(\mathbf{A}_i)\|_F^2. \tag{14}$$

Details about derivation of the log-Euclidean space similarity loss can be found in [33]. The parameters $\boldsymbol{\theta}_i$ of the student network f_i is optimized by minimizing the cloud-to-fog loss

$$\mathcal{L}_{\text{CF}} = \mathcal{L}_{\text{LD}_i} + \lambda_f \mathcal{L}_{\text{FD}_i} \tag{15}$$

where λ_f is empirically set to 0.1.

D. Training Procedure

In the training phase, optimization of the temperature parameter T_i and the network weights $\boldsymbol{\theta}_i$ are separated learning rate. Specifically, in each iteration, $\boldsymbol{\theta}_i$ is optimized subject to $\arg\min_{\boldsymbol{\theta}_i} \mathcal{L}_{\text{CF}}(f_i, f_{\text{clo}}; \boldsymbol{\theta}_i)$ for one epoch, then T_i is updated based on $\arg\min_{T_i} \mathcal{L}_{\text{LD}_i}(f_i, f_{\text{clo}}; T_i)$. During this process, each fog networks can be either updated simultaneously with sufficient computing power on the cloud server. However, they can be trained asynchronously through a group-by-group approach. This can be seen as sampling a subset from $\{f_i\}_{i=1}^M$ in each training iteration. In this work, we focus on simultaneous training of fog networks, and leave the asynchronous training for the future work. A brief description of this cloud-to-fog training process is presented by Algorithm 1.

V. Brain Storming: Fog-to-Cloud Distillation

As shown in figure, for a fog network with specific architecture and subtask, the pretained temperature T_i will converge at the end of the cloud-to-fog training. In the fog-to-cloud process, we fix this optimal $\hat{T}_i$ for each f_i. Inverse to the cloud-to-fog distillation, the fog-to-cloud learning is a *multiteacher-single-student* process. As described earlier, the networks $\{f_i\}_{i=1}^M$ distributed and finetuned at corresponding fog devices, are then used for updating the large cloud network f_{clo}. Although the finetuning process running on each fog sites can be relatively

Algorithm 1: Minibatch Cloud-to-Fog Training.

```
Input: Well trained cloud network $f_{clo}$ parameterized by
    $\theta_{clo}$; Batch size $B$; Maximum number of epochs
    $E_{max}$.
Output: Fog networks $\{f_i\}_{i=1}^M$ trained on cloud server.
Data: Cloud Dataset $\mathcal{D}_{clo}$.
1  Initialize: For networks $\{f_i\}_{i=1}^M$ with parameters $\{\theta_i\}_{i=1}^M$
     and temperature parameters $\{T_i\}_{i=1}^M$.
   /* $\{\theta_i\}_{i=1}^M$ and $\{T_i\}_{i=1}^M$ randomly initialized or
      pretrained on fog devices */
2  while epoch number $\leqslant E$ AND not converged do
3      for iteration number $\leqslant |\mathcal{D}_{clo}|/E$ do
4          Randomly sample a minibatch from $\mathcal{D}_{clo}$;
5          Compute $\mathcal{L}_{CF}$ using equation (15);
6          foreach fog network $f_i$ do Compute $\nabla_{\theta_i} f_i$;
7          Update $\{\theta_i\}_{i=1}^M$ using $\{\nabla_{\theta_i} f_i\}_{i=1}^M$;
8      for iteration number $\leqslant |\mathcal{D}_{clo}|/E$ do
9          Randomly sample a minibatch from $\mathcal{D}_{clo}$;
10         Compute $\mathcal{L}_{CF}$ using equation (15);
11         foreach fog network $f_i$ do Compute $\nabla_{T_i} f_i$;
12         Update $\{T_i\}_{i=1}^M$ using $\{\nabla_{T_i} f_i\}_{i=1}^M$;
13     epoch number + 1.
```

short, the fog networks suffer higher risk of overfitting. To reduce this risk and effectively update the networks, we design a brain storming mechanism using aggregated empirical risk and mixture of logits distillation loss.

A. Mixture Similarity Distances

With finetuned $\{f_i\}$ acting as teachers, a natural approach to train f_{clo} is learning from a mixture of $\mathbf{q}_i$ in the form of weighted sum. However, this may lead to a oversmoothed belief distribution due to the bias between the fog networks. To address this issue, we adopt the idea of [18] to compute a weighted sum of the feature distance losses

$$\mathcal{L}_{\text{FD}_{\text{clo}}} = \sum_{i=1}^{M} a_i \mathcal{L}_{\text{FD}_i}(f_i, f_{\text{clo}}; \boldsymbol{\theta}_{\text{clo}}) \tag{16}$$

where a_i is the weight of different feature similarity losses and is dynamically updated during training. The weight is constrained so that $\sum_{i=1}^M a_i = 1$ and $a_i \geq 0$. This can be simply satisfied by $a_i = |\tilde{a}_i|/\|\tilde{\mathbf{a}}\|$ where $\tilde{\mathbf{a}}$ is the vector of all unconstrained a_is.

B. Aggregated Empirical Risk

In this multiteacher brain storming process, the aggregated empirical risk loss is computed as

$$\mathcal{L}_{\text{ER}_{\text{clo}}} = \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}}\\ y_l=y_k, l\neq k}} -\log\frac{S(f_{\text{clo}}, f_{\text{clo}})}{S(f_{\text{clo}}, f_{\text{clo}}) + SN_{i,i}}$$
$$+ \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}}\\ y_l=y_k}} \sum_{i} -\log\frac{S(f_i, f_{\text{clo}})}{S(f_i, f_{\text{clo}}) + SN_{i,\text{clo}}}$$
$$+ \sum_{\substack{(\mathbf{x}_l,\mathbf{x}_k)\sim\mathcal{P}_{\text{batch}}\\ y_l=y_k}} \sum_{\substack{i,j\\ i\neq j}} -\log\frac{S(f_i, f_j)}{S(f_i, f_j) + SN_{i,i}}. \tag{17}$$

Optimization of the weight vector $\mathbf{a}$ can be then expressed as

$$\underset{\mathbf{a}}{\text{minimize}} \quad \mathcal{L}_{\text{ER}_{\text{clo}}}$$
$$\text{subject to} \quad \sum_{i=1}^{M} a_i = 1,\ a_i \geq 0.$$

As shown in (17), in the brain storming process, mixture of the feature losses are not only dependent on the mutual information between each fog networks. Iterative computing $L_{\text{ER}_{\text{clo}}}$ as in the cloud-to-fog process is expensive and slow. We compute this using an efficient matrices manipulation based way similar to (7) and (13). It can be show that with sufficient GPU memory $L_{\text{ER}_{\text{clo}}}$ can be efficiently computed.

C. Brain Storm Training

To sum up, the loss function used for updating the cloud network in the fog-to-cloud training process is a combination of the classification loss and the feature similarity loss

$$\mathcal{L}_{\text{FC}} = \mathcal{L}_{\text{cla}_{\text{clo}}} + \lambda_{\text{FD}}\mathcal{L}_{\text{FD}_{\text{clo}}} + \lambda_{\text{ER}}\mathcal{L}_{\text{ER}_{\text{clo}}} \tag{18}$$

where $\lambda_{\text{FD}} = \lambda_{\text{ER}} = 0.1$.

To learn the weight vector $\mathbf{a}$ for the fog networks, that can minimizing the aggregated empirical risk $\mathcal{L}_{\text{ER}_{\text{clo}}}$, we use a training procedure similar to [18]. The difference is that we optimize the student network parameter by minimizing $\mathcal{L}_{\text{FC}_{\text{clo}}}(f_{\text{clo}}, \{f_i\}; \boldsymbol{\theta}_{\text{clo}})$ while [18] only uses $\mathcal{L}_{\text{FD}}$.

At each iteration of gradient descent, we simulate one step of the cloud network logit features $\mathbf{z}_{\text{clo}} = f_{\text{clo}}(\mathbf{x})$ parameterized by $\{a_i\}_{i=1}^M$ using gradients of $\mathcal{L}_{\text{FC}_{\text{clo}}}(\mathbf{z}_{\text{clo}}, \{f_i\}_{i=1}^M; \{a_i\}_{i=1}^M)$

$$\mathbf{z}'_{\text{clo}} = \mathbf{z}_{\text{clo}} - \alpha\frac{\partial\mathcal{L}_{\text{FC}_{\text{clo}}}\left(\mathbf{z}_{\text{clo}}, \{f_i\}_{i=1}^M; \{a_i\}_{i=1}^M\right)}{\partial\mathbf{z}_{\text{clo}}} \tag{19}$$

where $\mathbf{z}'_{\text{clo}}$ is the simulated updated logits and α is the step size. Then, the gradient of updating $\{a_i\}_{i=1}^M$ is computed as

$$a'_i = a_i - \gamma\frac{\partial L_{\text{ER}_{\text{clo}}}(\mathbf{z}_{\text{clo}}, \{f_i\}_{i=1}^M)}{\partial a_i} \tag{20}$$

where a'_i is the updated weight for f_i and γ is the learning rate. Note that the updated logit feature $\mathbf{z}_{\text{clo}}$ is related to $\{a_i\}_{i=1}^M$ as the gradient computed shown in (19) contains a_i. Minimizing $L_{\text{ER}_{\text{clo}}}$ enables dynamic adaption of the weights assigned to the fog networks. At the beginning of training, a_i is initialized as $1/M$. Procedure of updating the cloud network f_{clo} is similar to the training of the cloud-to-fog distillation process with temperature T_i. In each epoch, $\{a_i\}_{i=1}^M$ is first updated as shown by (19) and (20), then $\boldsymbol{\theta}_{\text{clo}}$ is updated by minimizing $\mathbf{L}_{\text{FC}}(f_{\text{clo}}, \{f_i\}_{i=1}^M)$.

After the fog-to-cloud distillation, the cloud-to-fog process is performed again with smaller learning rate. Different from the process shown in Section IV, at this time we use $\mathcal{L}_{\text{ER}_{\text{clo}}}$ as empirical risk rather than $\mathcal{L}_{\text{ER}_i}$ shown in (12). Specifically, $\mathcal{L}_{\text{CF}} = \sum\mathcal{L}_{\text{cla}_i} + \lambda_1\sum\mathcal{L}_{\text{dis}_i} + \lambda_2\mathcal{L}_{\text{ER}_{\text{clo}}} + \lambda_f\mathcal{L}_{\text{FD}_i}$.

VI. Experiments and Results

A. Experiment Design and Implementation

We performed three comparison experiments and an ablation study to evaluate the cloud-to-fog and fog-to-cloud distillation separately, then the whole HBS framework as a continuous online training method. The first experiment compares the HBS model to multiple state-of-the-art (SOTA) methods in single-to-single distillation task with heterogeneous classifiers (Comparison 1). Experiment Comparison 2 looks at the core task of the fog-to-cloud distillation: unifying multiple heterogeneous teacher models to train one high-capacity student model. The bidirectional distillation processes are jointly evaluated in Comparison 3 with distributed data. We investigated the temperature parameter T across different architecture, tasks and methods in an ablation experiment.

We use CIFAR-100 datasets for all comparison experiments. We use a variety of popular backbone models with different depths, widths and parameter sizes, including: ResNet, VGG, wide ResNet (WRN), ShuffleNetV1, ShuffleNetV2. For WRN, WRN-d-w represents depth and width factors. To simulate a IoT environment, we train and test the cloud and fog networks on a nVidia Tesla P40 GPU with 24 G memory as the simulated cloud server and a P100 GPU with 16 GPU as the fog device. We implemented the HBS framework in Python using Pytorch1.3. All methods evaluated in our experiments use stochastic gradient descent as optimization. We initialize the learning rate as 0.05, and decay it by 0.1 every 30 epochs after the first 150 epochs until the 240th epoch. In all comparison experiments, we compute top-1 accuracies of different distillation objectives. For clear presentation of multiple fog networks, we use three heterogeneous fog networks for the experiments Comparison 2 and Comparison 3. When setting the batch size to 64, the peak of GPU memory used in the training phase is less than 3.5 GB. This means much more fog networks can be trained simultaneously for datasets, which are similar to CIFAR-100.

B. Comparison 1: HBS Cloud-to-Fog Versus Modern Supervised KD Methods

The first experiment assess the ability of our HBS cloud-to-fog method in single-teacher KD training both homogeneous and heterogeneous student models. We use Resnet110, Resnet32X4, Resnet50, WRN-40-2, and VGG13 as teacher networks trained for full classification task on CIFAR-100. For each teacher model, we testing its performance on KD to a smaller lightweight student with similar and different architectures. For example, compression Resnet32X4 to Resnet8X4 and to ShuffleNetv2. This comparison provides a insight into the effectiveness of our new distillation losses and the flexibility of our method when applied to different tasks. We selected eight SOTA methods as baselines for the conventional full-category KD task, and compare eight of them for the partial-category classification task. The involved methods include: KD [2], FitNet [9], attention transfer (AT) [10], similarity-preserving (SP) KD [34], correlation congruence (CC) [15], variational information distillation (VID) [35], relational KD (RKD) [11], and contrastive representational distillation (CRD) [16].

Table I shows the Top-1 accuracy of the fog networks obtained as student using all the baselines mentioned above, classifying a randomly sampled subset of CIFAR100 classes (average on 20,50,70 classes). The student networks are generated by learning from teachers that performing full-range classification. Most baselines obtained high accuracy compared to a student network that is trained from scratch. When performing cross-architecture distillation, several failed to beat the indepedently trained student. It is clear that, for example, compressing a Resnet50 to a ShuffleNetV2 is more difficult than obtaining a Resnet32. Our method outperformed almost all the compared baselines, except in the "WRN-4-2 to ShuffleNetV1" task where CRD achieved better results. Surprisingly, the conventional logit distillation method (KD) performed better than most of recently proposed baselines. It is evident that the HBS model, with help of the new distillation losses, are more suitable for partial KD in real-world IoT scenarios.

C. Comparison 2: HBS Fog-to-Cloud Versus State-of-the-Art Multiteacher Distillation Methods

To test the ability of our HBS model dealing with homogeneous classifiers, we selected four state-of-the-art multiteacher distillation methods, including: DML [17], CL-ILR [20], ONE [5], and OKDDip [19]. We use a single pretained network as the "baseline" providing a lower bound of classification accuracy. For a fair comparison, implementation of these methods were forked and modified from OKDDip code.[3] Note that for CL-ILR and ONE, the three teachers are unified as different branches on a higher capacity student model. In this case, we follow the setup in [19] where the fog networks share the first few blocks and branched by the last block. The backbone architectures used in this experiment include: WRN-20-8, Resnet32, VGG16, and Resnet110. Table II shows the Top-1 accuracy of the cloud network obtained as the student during the fog-to-cloud distillation process. All the compared group distillation methods gained close performance in most cases. The branch based methods got higher accuracy with Resnet110. With shallower models, network based methods perform better. The HBS model performed slightly better than the state-of-the-art methods when dealing with homogeneous classifiers with VGG16 and Resnet architectures, except for WRN-20-8, where our method obtained almost the same performance with OKDDip. Based on our discussion in Section III, dealing with homogeneous classifiers means $\mathcal{C}_{-i} = \emptyset$ for networks, and $q_i(y \in \mathcal{C}_{-i}) = 0$. As a result, our cloud feature distance loss $\mathcal{L}_{\mathrm{FD}_{\mathrm{clo}}}$ is similar with OKDDip loss. However, OKDDip uses a simplified self-attention module for each involved network to compute the weighted mean of all output distributions. This requires extra trainable parameters for better learning capacity. Rather than directly combine the output distributions, our HBS model assigns weights to feature distance losses $\mathcal{L}_{\mathrm{FD}_i}$, and adds the aggregated empirical risk loss without extra parameters. As a result, even though HBS is not designed specifically for homogeneous multinetwork distillation, it can archived comparable performance with less parameters.

[3][Online]. Available: https://github.com/DefangChen/OKDDip-AAAI2020

TABLE I
TEST ACCURACY(%) OF STUDENT (FOG) NETWORKS ON CLASSIFYING PART OF CIFAR100 CATEGORIES (50 CLASSES)

Teacher	WRN-40-2		Resnet50		Resnet110		Resnet32X4		VGG13	
Student	WRN-40-1	ShuffleNetV1	Resnet32	MobileNetV2	Resnet32	VGG8	Resnet8x4	ShuffleNetV2	VGG8	MobileNetV2
Teacher	75.61	75.61	79.34	79.34	74.31	74.31	79.42	79.42	72.34	72.34
Student	75.56	72.9	66.4	73.44	71.09	73.56	72.11	73.91	72.52	67.16
KD	73.91	74.13	68.35	75.81	73.67	73.91	76.01	76.55	74.86	69.73
FitNet	75.86	75.12	65.69	73.99	71.02	73.96	75.56	74.44	74.01	67.14
AT	76.06	75.22	61.18	74.04	73.01	75.01	73.12	74.63	73.68	61.45
SP	75.31	76.22	69.98	75.54	73.01	74.83	75.84	76.69	75.1	69.96
CC	76.61	74.85	68.43	72.45	71.06	72.18	72.29	74.92	72.91	67.71
VID	76.81	75.94	70.05	73.31	72.19	74.91	75.85	75.68	73.43	67.97
RKD	76.35	76.21	66.42	74.02	71.92	73.03	74.91	75.77	73.82	66.71
CRD	77.68	77.99	71.62	76.81	73.66	75.87	77.83	78.29	76.06	72.04
HBS	**78.21**	**78.02**	**72.4**	**76.95**	**73.98**	**76.01**	**77.87**	**78.91**	**76.59**	**72.32**

Eight State-of-the-art distillation methods are compared to our HBS cloud-to-fog methods. average over five runs. All bold entities show statistical significance compared to the second best results (pvalue $< 0{:}05$).

TABLE II
TEST ACCURACY(%) OF CLOUD (STUDENT) NETWORKS ON CLASSIFYING ALL TEST IMAGES IN CIFAR100

Student	Baseline	DML	CL-ILR	ONE	OKDDip	HBS
WRN-20-8	76.50	78.79	78.56	77.81	**79.37**	79.34
VGG16	72.81	73.67	73.38	73.37	74.12	**74.77**
Resnet32	71.24	73.53	72.56	73.50	74.60	**74.81**
Rsnet110	74.88	76.50	77.44	77.34	77.90	**78.09**

Results of four multiteacher distillation methods are compared to our HBS fog-to-cloud distillation. The baseline is a single network trained solely with hard labels. mean and standard deviation of the results are obtained over five runs. All bold entities show statistical significance compared to the second best results (pvalue $< 0{:}05$)

TABLE III
TEST ACCURACY(%) OF CLOUD (STUDENT) NETWORKS ON CLASSIFYING ALL TEST IMAGES IN CIFAR100

Task	Task1: Overlapping Classes			Task2: With Unoverlapped Classes		
Student	WRN	R32X4	R50	WRN	R32X4	R50
CFL	73.18	**76.84**	76.78	73.07	74.65	74.96
UHC	73.97	76.81	76.23	70.05	72.40	72.83
HBS	**74.61**	76.83	**77.91**	**74.74**	**76.62**	**77.09**

Results of two multiteacher distillation methods are compared to our HBS fog-to-cloud distillation. Three backbone networks were tested: WRN-40-2 (WRN), Resnet32X4 (R32X4), and Resnet50 (R50). All bold entities show statistical significance compared to the second best results (pvalue $< 0{:}05$)

To perform joint distillation for heterogeneous classifiers, we prepared two different tasks with three pretrained teachers: Resnet8X4, Vgg8, and WRN-40-1. We chose two architectures for the single student: Resnet32X4 and Resnet50. In the first task, each teacher classify 75 classes of CIFAR100 data so that each pair of teachers share some overlapped classes. In the second task, the Vgg8 and WRN-40-1 teachers each classify 40 classes, and shared 10 overlapped classes. The Resnet32 teacher classify 30 classes of images, without any overlapping with the targets of the other two teachers. The two baselines we used that can deal with heterogeneous teachers with totally different architectures are: common feature learning (CFL) [23] model and the unifying heterogeneous classifiers (UHC) [21] model. Both models can also used for homogeneous classifiers, but in our experiments they did not generate impressive results in the homogeneous experiment shown earlier. We reimplemented the best performed UHC entropy-based loss by removing the dustbin class from the classifiers, then added the same label balancing and regularization operations. The CFL implementation was forked from KAmalEngine.[4]

As shown in Table III, when unifying the three teachers with overlapping predicted classes, the three compared methods have close performance. HBS shows slightly better results than other two methods for Resnet32X4 and Resnet50, and obtained 0.3% less accuracy compared to CFL for the WRN-40-2 student. However, with a class-independent teacher, the performance of UHC drops over 4% in some cases. Accuracy of CFL were also slightly reduced. Our HBS model shows the best stability, and even got better accuracy for WRN-40-2 compared to its performance in the "overlapping classes" task. Without a "dustbin" class, it is difficult to find an optimal alignment between the heterogeneous distributions obtained from multiple networks that are trained for completely different classes. This is one reason why HBS model shows an advanced performance in the 'unoverlapped classes" task.

[4][Online]. Availabl https://github.com/zju-vipa/KamalEngine

D. Comparison 3: Online Training

Follow the abovementioned experiment, we use 70% of the CIFAR100 training data as the cloud dataset $\mathcal{D}_{\text{clo}}$ and split the rest data following the setup of Comparison 2: split them into three fog datasets ($\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$) for the three heterogeneous teachers. For convenient computation, the data belongs to overlapped classes are shared by the teachers. Following the steps listed in Section III-B. A Resnet32X4 cloud network and the three fog networks were then trained only with $\mathcal{D}_{\text{clo}}$. Then, the fog networks were separately finetuned with $\mathcal{D}_1, \mathcal{D}_2$, and $\mathcal{D}_3$ using an early stop setup. Till now, the whole CIFAR100 training set used up. We then perform the fog-to-cloud distillation as shown in Comparison 2 using the finetuned fog networks. The

TABLE IV
TOP-1 ACCURACY(%) OF CLOUD (STUDENT) NETWORKS ON CLASSIFYING ALL TEST IMAGES IN CIFAR100

Task	Overlapping Classes	With Unoverlapped Classes
Cloud Net (70% train)	73.30	
Cloud Net (100% train)	75.61	
CFL	73.29	73.07
UHC	73.75	70.05
HBS	**74.66**	**74.58**

All bold entities show statistical significance compared to the second best results (pvalue $< 0{:}05$)

TABLE V
TOP-5 ACCURACY(%) OF SINGLE-TEACHER DISTILLATION PERFORMED BY HBS AND OTHER FIVE BASELINES ON FULL CIFAR-100 DATASET

	Fixed T	Trainable T
KD	92.52	**92.75**
FitNet	93.23	**93.38**
VID	92.71	**92.99**
Attention	**93.11**	92.98
CRD	93.33	**93.76**
HBS (ours)	93.40	**94.23**

All bold entities show statistical significance compared to the second best results (pvalue $<$ 0:05)

cloud network trained by 70% training data provide a lower bound that can verify the applicability to the distributed data storage in IoT, as its performance should not drop after the fog-to-cloud distillation.

Table IV shows the top-1 test accuracy obtained from our online training procedure. As there are less training data available to the cloud network, its performance dropped and suffer from more severe overfitting compared to being trained with the full dataset. Note that the fog networks here are generated through distillation with a less accurate cloud network compared to Comparison 2, the finetuned fog networks also performed worse. When all the fog networks have overlapped classes, CFL and UHC methods only lead to tiny improvements on the final cloud network. In other cases, they did not show advantages of group distillation. The BHS model shows 2% higher accuracy while a fully trained cloud net is only 1.2% better. When dealing with unoverlapped fog networks, only our HBS model improved the initially trained cloud network.

E. Ablation: Trainable T

As the temperature hyperparameter T control the smoothness of the original KD loss $\mathcal{L}_{\text{dis}}$, it can be applied to all the compared methods in the Comparison 1 experiment. We first test the effect of the trainable T against using a fixed T. We perform this test on our HBS and other five single-teacher distillation methods: CRD, Attention, FitNet, VID, and the original KD algorithm.[5] The training curves obtained with and without the trainable T are presented in Fig. 4. It can be seen that using

[5]For attention transfer and VID methods, we added the $\mathcal{L}_{\text{dis}}$ to as a weighted term in the total loss, follow the setup of [16]. Other methods we use the setups from the original papers.

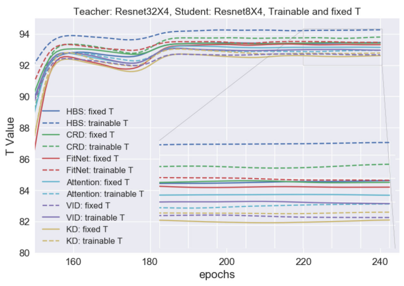

Fig. 4. Evolution of top-5 test accuracy during training. Our HBS and other five single-teacher baselines were compared with trainable and fixed temperature hyperparameters. Experiments are performed on full CIFAR-100 dataset.

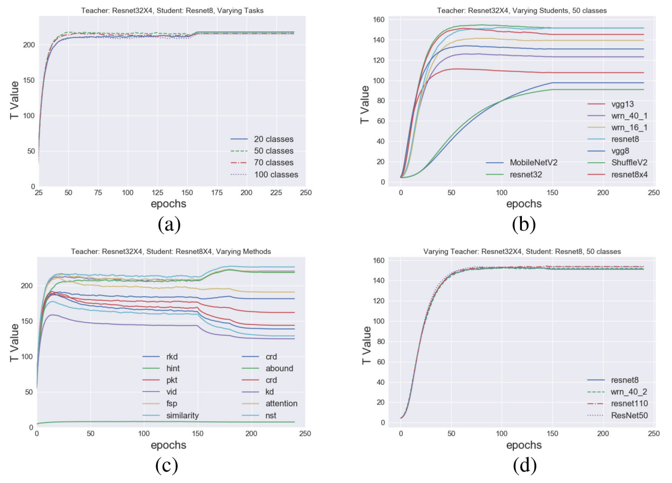

Fig. 5. Evolution of the trainable temperature T with different (a) tasks, (b) student architectures, (c) methods, and (d) teachers.

the trainable temperature hyperparameter increased accuracy of HBS by about 1%, and slightly enhanced the performance of FitNet, VID, and CRD. The Attention transfer method has a better performance with a fixed T as it focuses on transfering attention features in both shallow and deep layers.

As discussed earlier in this article, all previous works uses an empirically fixed temperature hyperparameter without considering the network architectures, task complexities and specific KD algorithms. Based on abovementioned ablation experiment, we perform a further step to find the optimal T values for different distillation methods, teacher and student architectures, and different classification tasks. The evolution of T in the training process are shown in Fig. 5. It clearly shows that the optimal T value is only affected by the network structure and specific combination of losses in different methods. Teacher architecture and specific task have negligible effects. Generally, network with less parameters requires larger T. This is consistent with [2], and also verified the effects of using the trainable temperature.

VII. CONCLUSION

We proposed an novel KD method for object recognition in real-world IoT scencarios. Our method enabled flexible bidirectional online training of heterogeneous models distributed datasets with a new "brain storming" mechanism and optimizable temperature parameters. In our comparison experiments, this HBS method were compared to multiple state-of-the-art single-model compression methods, as well as the newest heterogeneous and homogeneous multiteacher KD methods. Our methods outperformed the state-of-the-art methods in both conventional and heterogeneous tasks. Further analysis of the ablation experiment results shows that introducing the trainable temperature parameters into the conventional KD loss can effectively ease the learning process of student networks in different methods.

REFERENCES

[1] C. Wang, S. Dong, X. Zhao, G. Papanastasiou, H. Zhang, and G. Yang, "Saliencygan: Deep learning semi-supervised salient object detection in the fog of IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2667–2676, Apr. 2020.

[2] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS 2014 Deep Learn. Workshop*, 2015, *arxiv:1503.02531*.

[3] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 535–541.

[4] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[5] X. Zhu *et al.*, "Knowledge distillation by on-the-fly native ensemble," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7517–7527.

[6] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3713–3722.

[7] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *Int. Conf. Mach. Learn.*, 2018, pp. 1607–1616.

[8] K. Clark, M.-T. Luong, U. Khandelwal, C. D. Manning, and Q. V. Le, "Bam! born-again multi-task networks for natural language understanding," in *Proc. 57th Annual. Meeting. Assoc. Comput. Linguistics.*, 2019, pp. 5931–5937.

[9] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.

[10] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *Int. Conf. Learn. Represent.*, 2017, *arXiv:1612.03928*.

[11] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3967–3976.

[12] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 3779–3787.

[13] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2760–2769.

[14] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4133–4141.

[15] B. Peng *et al.*, "Correlation congruence for knowledge distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 5007–5016.

[16] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in *Int. Conf. Learn. Represent.*, 2020.

[17] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[18] A. Wu, W.-S. Zheng, X. Guo, and J.-H. Lai, "Distilled person re-identification: Towards a more scalable system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1187–1196.

[19] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *AAAI*, 2020, pp. 3430–3437.

[20] G. Song and W. Chai, "Collaborative learning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1832–1841.

[21] J. Vongkulbhisal, P. Vinayavekhin, and M. Visentini-Scarzanella, "Unifying heterogeneous classifiers with distillation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3175–3184.

[22] C. Shen, M. Xue, X. Wang, J. Song, L. Sun, and M. Song, "Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3504–3513.

[23] S. Luo, X. Wang, G. Fang, Y. Hu, D. Tao, and M. Song, "Knowledge amalgamation from heterogeneous networks by common feature learning," in *Proc. 28th Int. Joint. Conf. Artificial. Intell.*, 2019, pp. 3087–3093.

[24] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *British. Mach. Vis. Conf.*, 2015.

[25] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," *Workshop Learn. Limited Data*, 2017.

[26] G. K. Nayak, K. R. Mopuri, V. Shaj, R. V. Babu, and A. Chakraborty, "Zero-shot knowledge distillation in deep networks," in *Int. Conf. Mach. Learn.*, pp. 4743–4751, 2019.

[27] H. Chen *et al.*, "Data-free learning of student networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3514–3522.

[28] J. Yoo, M. Cho, T. Kim, and U. Kang, "Knowledge extraction with no observable data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2701–2710.

[29] J. Ye, Y. Ji, X. Wang, X. Gao, and M. Song, "Data-free knowledge amalgamation via group-stack dual-GAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 12516–12525.

[30] P. Micaelli and A. J. Storkey, "Zero-shot knowledge transfer via adversarial belief matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9547–9557.

[31] K. Bhardwaj, N. Suda, and R. Marculescu, "Dream distillation: A data-independent model compression framework," in *ICML Joint. Workshop. On-Device Mach. Learning. & Compact. Deep. Neural. Network. Represent.*, 2019, *arXiv:1905.07072*.

[32] H. Yin *et al.*, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2020, 2019, pp. 8715–8724.

[33] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Fast and simple calculus on tensors in the log-euclidean framework," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2005, pp. 115–122.

[34] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1365–1374.

[35] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9163–9171.

Chengjia Wang (Member, IEEE) received the M.Sc. degree in vision and robotics from the European Erasmus Mundus Master Programme Heriot-watt University, Edinburgh, U.K.; the University of Girona, Girona, Spain; and the University of Burgundy, Dijon, France, in 2011, and the Ph.D. degree in machine learning-based medical image analysis from Clinical Research Imaging Centre, The University of Edinburgh, Edinburgh, U.K., in 2016.

He joined the BHF Centre for Cardiovascular Science, The University of Edinburgh, as a Scientist in Machine Learning for Machine Learning, in 2016. His current research interests include computer vision, AI in medicine, and machine learning-based medical image analysis.

Guang Yang (Member, IEEE) received the M.Sc. degree in vision imaging and virtual environments and the Ph.D. degree in medical image analysis from University College London, London, U.K., in 2006 and 2012, respectively.

He is currently an Honorary Lecturer with the Neuroscience Research Centre, Cardiovascular and Cell Sciences Institute, St. George's, University of London, London, U.K. He is also an Image Processing Physicist and Honorary Senior Research Fellow with the Cardiovascular Research Centre, Royal Brompton Hospital and also affiliate with National Heart and Lung Institute, Imperial College London. His research interests include: pattern recognition, machine learning, and medical image processing and analysis. His current research projects are funded by the British Heart Foundation.

Giorgos Papanastasiou received the Ph.D. degree in medical physics and mathematical modeling with the Clinical Research Imaging Centre and the British Heart Foundation-Centre for Cardiovascular Science, The University of Edinburgh, Edinburgh, U.K.

He was a Postdoctoral Research Fellow in Edinburgh Imaging with the Queen's Medical Research Institute, The University of Edinburgh. He is currently an Assistant Professor in Artificial Intelligence and Engineering in Medicine.

Heye Zhang (Member, IEEE) received the B.S. and M.E. degrees from Tsinghua University, Beijing, China, in 2001 and 2003, respectively, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2007, all in physics.

Before Joining Sun Yat-sen University, Guangzhou, China, in 2018, he was a Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. He is currently leading a Health Informatics Computing Lab, School of Biomedical Engineering, Sun Yat-sen University. His research interests include cardiac electrophysiology and cardiac image analysis.

Joel J. P. C. Rodrigues (Fellow, IEEE) received the B.Sc. degree (licentiate) in informatics engineering from the University of Coimbra, Coimbra, Portugal, and the M.Sc. degree in informatics engineering and the Ph.D. degree in informatics engineering from the University of Beira Interior (UBI), Covilhã, Portugal, and the Habilitation degree in computer science and engineering from the University of Haute Alsace, Mulhouse, France.

He received the Academic Title of Aggregated Professor in informatics engineering from UBI. He is currently a Professor with the Federal University of Piauí (UFPI), Teresina, Brazil, and Senior Researcher with the Instituto de Telecomunicações, Aveiro, Portugal. His main research interests include IoT and sensor networks, e-health technologies vehicular communications, and mobile and ubiquitous computing.

Victor Hugo C. de Albuquerque (Senior Member, IEEE) received the bachelor's degree in mechatronics engineering from the Federal Center of Technological Education of Cearáthe, the M.Sc. degree in teleinformatics engineering from the Federal University of Ceará, Fortaleza, Brazil, the Ph.D. degree in mechanical engineering from the Federal University of Paraíba, Belém, Brazil.

He is currently a Full Professor and Senior Researcher with the University of Fortaleza, Fortaleza, Brazil, and Data Science Director with the Superintendency for Research and Public Safety Strategy of Ceará State, Brazil. He leads the Graduate Program in Applied Informatics and Electronics and Health Research Group (CNPq). He mainly researches IoT, machine/deep learning, pattern recognition, and robotics.