Time-constrained Ensemble Sensing with Heterogeneous IoT Devices in Intelligent Transportation Systems

Xingyu Feng, Chengwen Luo*, Bo Wei, Jin Zhang, Jianqiang Li, Huihui Wang, Weitao Xu, Mun Choon Chan, Victor C.M. Leung

***Abstract*—Recently we have witnessed the rise of Artificial Intelligence of Things (AIoT) and the shift of sensing paradigm from cloud-centric to the edge-centric, which effectively improves the sensing capability of intelligence transportation systems. To improve the real-time sensing performance, in this work we propose an ensemble sensing based scheme to solve the time-constraint synchronized inference problem and achieve robust inference with heterogeneous IoT devices in intelligence transportation systems. We design and implement *Ensen*, which incorporates various novel techniques such as customized DNN model design, KD-based model training, and dynamic deep ensemble management, etc., to achieve improved accuracy and maximize the computational resource usage of the whole sensing group. Extensive evaluations on different types of common IoT devices have shown that *Ensen* achieves a robust performance and can be easily extended to different types of convolutional neural networks.**

***Index Terms*—Edge Intelligence, Ensemble Sensing, Time-constrained, Heterogeneous IoT Device**

I. Introduction

As the rapid development of artificial intelligence (AI) and sensing technologies for Internet of Things (IoT), the ubiquitously deployed Road Side Units (RSUs) and On board Units (OBUs) have brought new opportunities for smarter and more efficient Intelligent Transportation Systems (ITS) [1], [2]. With the increasing communication capability [3]–[6] and equipped with a sophisticated set of embedded sensors and high-performance chips, RSUs and OBUs continuously collect different types of sensor data from the traffic roads and provide intelligent transportation services in real time [7]. For example, [8] uses image data collected by multiple surveillance cameras for vehicle identification. [9] uses radar sensors for traffic monitoring and generate traffic accident alert, etc. These intelligent transportation applications are critical for traffic control and can improve the safety and security of the overall transportation systems [10], [11].

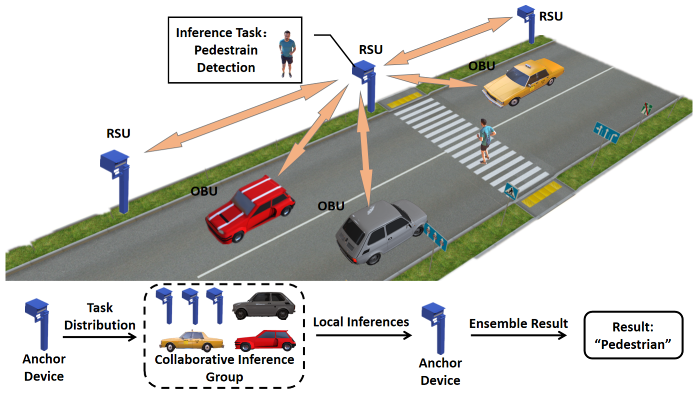

Fig. 1: Motivating scenario: collaborative inference in intelligent transportation systems

Intelligent transportation services often rely on advanced neural network models trained using large amounts of data from users. However, full deployment of intelligent transportation services is particularly challenging due to two critical issues: (1) High computational costs of deep neural network (DNN) models, especially for computer vision-based tasks, which may render it inapplicable for real-time inferences in intelligent transportation systems, in which large amount of devices mainly rely on low energy-consumption embedded devices [12]. Moreover, a wide variety of RSUs and OBUs have different hardware capabilities, and a unified DNN model may not be able to adapt to heterogeneous low-performance devices. (2) The training of DNN models usually requires the collection of training data from participating users. For example, for obstacle recognition tasks in intelligent transportation systems, training data often exists in the form of *of isolated islands* [13], where real obstacle images are collected jointly by cameras spread throughout the traffic network, rather than a centralized organization. Therefore, local images from all camera nodes in the system need to be transmitted to a central server, which greatly increases the difficulty of data acquisition and causes severe latency and bandwidth consumption. On the other hand, obstacle images usually contains landmark information, and data sharing can lead to the exposure of important private information such as travel trajectories and resident addresses of the users. To address these challenges, one solution is to suggest that each participant trains a private model using local data, and the design of the local DNN model structure is customized by its own hardware capability, so that each devices can easily form collaborative inference groups

X. Feng, C. Luo, J. Zhang, J. Li, and Victor C. M. Leung are with College of Computer Science and Software Engineering, Shenzhen University, China. Email: fengxingyu2017@email.szu.edu.cn, {chengwen, jin.zhang, lijq}@szu.edu.cn, vleung@ieee.org.

B. Wei is with Lancaster University, United Kingdom. Email: bo.wei@lancaster.ac.uk.

H. Wang is with St. Bonaventure University, USA. Email: hwang@sbu.edu.

W. Xu is with City University of Hong Kong, Hong Kong, China. Email: weitaoxu@cityu.edu.hk.

M. C. Chan is with National University of Singapore, Singapore. Email: chanmc@comp.nus.edu.sg

* Corresponding author.

in intelligent transportation systems and provide synchronized inference services.

Motivated by idea of ensemble sensing the recently proposed concept of *deep ensembles* [14], [15], in this paper we propose to achieve collaborative inference among neighboring intelligent transportation devices through ensemble sensing to solve the above problem. As illustrated in Figure 1, when a RSU device $\mathcal{D}_a$ needs to perform one inference task, we define $\mathcal{D}_a$ as the anchor device. Then, $\mathcal{D}_a$ sends the sensing task to neighboring devices to form a collaborative inference group for better inference accuracy. In the inference group, a private obstacle recognition model trained by local data is available on each participating device. Each device feeds sensing data into the local DNN model, and aggregates their local inference results back to the anchor device $\mathcal{D}_a$. Finally, $\mathcal{D}_a$ gets the final prediction by aggregating all local predictions. It is worth noting that the local data on different devices are *Non - Independent Identically Distributed (Non-iid)* [16], resulting in their private models to have different classification performances for different categories of input. For an unknown inference sample, we do not know which device can perform better. Therefore, aggregating all local predictions generates more robust inference results. Besides better accuracy and robustness in the inference, such paradigm also has multiple other benefits as well. First, by aggregating each device's local predictions, idle computation resources in the sensing area can be efficiently utilized. Second, the deep ensembles allow each device to dynamically join or leave the group. Members provide only their own local predictions to the inference group and are free to join and go, which is important for intelligent transportation systems where participating users can be mobile.

Challenges. To realize this vision, however, several challenges need to be tackled: (1) *DNN models customization for heterogeneous devices in the inference group.* In practice, devices within the vicinity in intelligent transportation systems are often heterogeneous. For example, various vehicles shown in Figure 1 might have different hardware capabilities. This makes it necessary to customize DNN models for each cooperating device based on its own hardware settings. In particular, for the prevalent real-time tasks in intelligent transportation systems, the total inference time of the group needs to be bounded. Time constraints need to be taken into account when designing customized DNN models for heterogeneous devices. (2) *Performance improvement for different local DNN models.* As mentioned above, each device typically has only a small local training dataset and the model structure needs to be customized for real-time applications, which leads to potential performance degradation of local DNN models. Therefore, improving the performance of each local DNN model is essential to improve the performance of the whole inference group. (3) *Dynamic inference group management.* In intelligent transportation systems each user can be mobile and therefore the group members in a deep ensemble needs to be dynamically managed. In applications where malicious devices can present and degrade the predication accuracy of the group by voting wrong predictions, it becomes especially important to dynamically filter out such devices to maintain the robust predictions of the whole group.

To address the above challenges, in this paper we propose *Ensen*, an ensemble sensing system for intelligent transportation systems, which generates customized DNN models for heterogeneous devices based on time constraints and hardware settings and achieves robust collaborative inference results. *Ensen* treats the inference time of the whole inference group as a constraint since task completing time is critical in transportation systems. To achieves this, *Ensen* employs a dynamic network architecture searching algorithm to trim the base model and search for the optimal model structure for heterogeneous devices under the time constraint. In the model training process, *Ensen* combines the knowledge distillation (KD) method to maximize the local inference accuracy. Finally, in the ensemble inference phase, the utility of each device is continuously evaluated during the voting process to dynamically filter out malicious/low accuracy nodes to ensure the robust inference performance of the group. The contributions of this paper are summarized as follows:

- To the best of our knowledge, this paper is the first to propose customized DNN model based ensemble sensing scheme for heterogeneous devices in intelligent transportation systems under time constraints.
- The paper proposes efficient network search algorithm for heterogeneous devices with hardware and execution time constraints. A KD-based training approach is applied to efficiently improve the local accuracy and a dynamic group management scheme is used to improve the global inference performance.
- We design and implement the *Ensen* system, extensive evaluations are conducted to evaluate the performance of *Ensen* in various settings, which shows that *Ensen* runs efficiently for collaborative inference among heterogeneous devices.

The rest of this paper is organized as follows. In Section II we discuss the design overview of the *Ensen*. In Section III we provide the details of the customized network architecture search algorithm for heterogeneous IoT devices, and in Section IV the KD-based training process. Section V discusses the dynamic inference group management, and Section VI provides the detailed evaluations. Section VII discusses the relative work, and finally we conclude in Section IX.

II. System Overview

The *Ensen* overview is shown in Figure 2. A set of n intelligent transportation devices $\boldsymbol{\mathcal{D}} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ within the vicinity are formed as an ensemble group (deep ensemble) dynamically to complete a sensing task collaboratively. Each device $\mathcal{D}_i$ in the deep ensemble has unique hardware capability and local data. The *Ensen* deploys a deep neural network model $\mathcal{M}_i$ on each device. When a device acquires the inference task x_i, it is defined as the anchor device. This device sends the inference task to $\mathcal{D}_i$, which runs $\mathcal{M}_i(x_i) \rightarrow y_i$ locally. The anchor device collects all local predictions and runs the ensemble function to generate the final inference result. An ensemble function is $\mathcal{G}(y_1, y_2, ..., y_n) \rightarrow y$, where y is the final prediction result output by the aggregation function

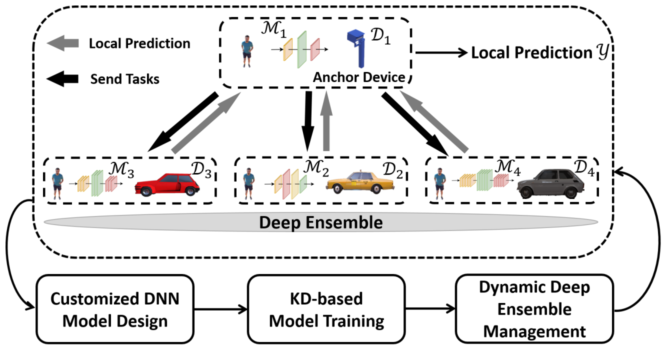

Fig. 2: System overview of *Ensen*

and treated as the agreed prediction by all members in the deep ensemble.

To achieve the above goal, the *Ensen* system is designed to have three modules as shown in Figure 2: the customized DNN model design module, KD-based model training module, and the dynamic ensemble group management module. To allow heterogeneous devices to cooperate efficiently, the customized DNN model based on the computational capability of each device $\mathcal{D}_i$ is firstly generated in the model design module. The KD-based module provides efficient training of each device's model by integrating both the global knowledge and local data distribution. Finally, the dynamical ensemble group management module integrates all local predictions and generates ensemble inference result. In the following sections we provide the detailed design rationale of each module.

III. Customized DNN Network Design for Heterogeneous IoT Devices in Transportation Systems

A. *Time-constrained Synchronized Inference*

In this work, we consider collaborative inference among n co-locating IoT devices $\boldsymbol{\mathcal{D}}$. In collaborative inference, one inference task needs to be participated by all members, and due to the hardware capability differences of all devices, faster devices always need to wait for the slower ones to complete the overall inference process, and idle waiting inevitably wastes computational resources. In addition, there are strict time constraints in transportation systems, such as signal light detection in assisted driving, where the inference time needs to be strictly constrained within $\mathcal{T}$. We take into account the efficiency of the collaborative inference group, and use the inference time for the same task as the constraint for model design on each device. We define the time-constrained synchronized inference problem as follows:

Problem 1 (*Time-constrained Synchronized Inference*)**:** *Given a set of n devices $\boldsymbol{\mathcal{D}}$, can we design customized DNN models $\mathcal{M}_i$ for each device such that for the given input x_i of device $\mathcal{D}_i$, the model inference time difference of all devices is minimized and satisfies the given time constraint $\mathcal{T}$?*

Since each device completes local inference around the same time in time-constrained synchronized inference setting, to maximize the computational resource usage of all devices, one optimal strategy is to design customized DNN models

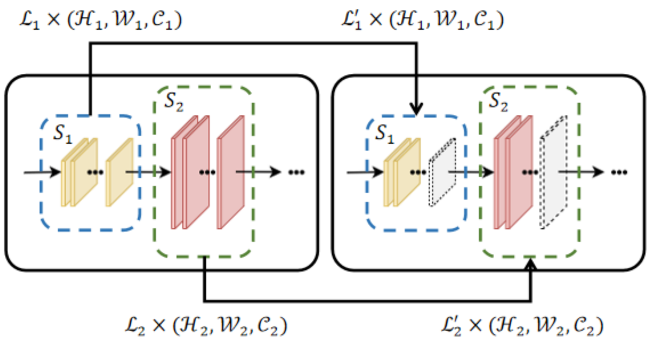
$\mathcal{L}_1 \times (\mathcal{H}_1, \mathcal{W}_1, \mathcal{C}_1)$
$\mathcal{L}'_1 \times (\mathcal{H}_1, \mathcal{W}_1, \mathcal{C}_1)$
S_1
S_2
S_1
S_2
$\mathcal{L}_2 \times (\mathcal{H}_2, \mathcal{W}_2, \mathcal{C}_2)$
$\mathcal{L}'_2 \times (\mathcal{H}_2, \mathcal{W}_2, \mathcal{C}_2)$

Fig. 3: Illustration of depth scaling of CNN models

such that all devices' model inference time closely approaches to $\mathcal{T}$. This problem therefore can be re-written as the following optimization problem:

$$\begin{aligned} \min_{\mathcal{M}_i} \quad & \sum_{i=1}^{n} (\mathcal{T} - T(\mathcal{M}_i(x_i))) \\ \text{s.t.} \quad & \forall_{i\in[1,n]}\, T(\mathcal{M}_i(x_i)) \leq \mathcal{T} \end{aligned} \tag{1}$$

where $T(\mathcal{M}_i(x_i))$ is the inference time of model $\mathcal{M}_i$ on device i for input x_i.

B. *Customized Model Design*

To solve the above problem, in this section we propose the deep neural network architecture search algorithm for heterogeneous IoT devices. In conventional cloud-centric sensing paradigm, a global DNN model is usually trained and deployed on the cloud to provide the cloud-based inference services. Though the global model is not specifically trained to reflect each IoT device's local data distribution, it represents the global knowledge and provides guidance to the design of the edge models. Inspired by the recent development of knowledge distillation (KD) [17], in *Ensen* we integrate both the global knowledge represented by the cloud model and local data distribution to design the customized models for IoT devices by adopting the KD-based approach. The structure of the student model will be considered as a compressed version of the teacher model in the cloud.

As shown in Figure 3, convolutional neural network (CNN) typically consists of multiple stages $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_1, \mathcal{S}_2 ..., \mathcal{S}_k\}$, and different stages serve different functionalities for feature extraction. Each stage $\mathcal{S}_i$ consists of a set of neural network layers $\boldsymbol{\mathcal{L}}_i = \{\mathcal{L}_i^{(1)}, \mathcal{L}_i^{(2)} ..., \mathcal{L}_i^{(m)}\}$, where each of m layers is connected one by one and with shape $(\mathcal{H}_i, \mathcal{W}_i, \mathcal{C}_i)$, and here $\mathcal{H}_i$, $\mathcal{W}_i$ are spatial dimensions and $\mathcal{C}_i$ is the channel dimension.

The structure of the teacher model in the cloud is defined as the basic model. To facilitate the fast design of the student model structure, we only consider the deep compression of the basic model. In this approach, the device only needs to count the inference time at different depths by opening and closing the convolutional layers until the inference time limit is satisfied. This allows the device to obtain the student model by searching the basic model only, without considering the width variation within the stage, i.e. $\mathcal{H}_i$, $\mathcal{W}_i$. That is, with depth scaling, the length of each stage i, i.e., the number of neural network layers in $\boldsymbol{\mathcal{L}}_i$ is adjusted dynamically to meet the inference time requirements, while the shape of each layer in $\boldsymbol{\mathcal{L}}_i$ is kept unchanged to maintain the fast generation of the student model.

To solve the optimization problem set in Equation (1), our task therefore is to use teacher model as the reference and use depth scaling to search for network model $\mathcal{M}_i$ for device i such that the inference time of $\mathcal{M}_i$ closely approaches to $\mathcal{T}$. The workflow of customized model design in *Ensen* is illustrated in Algorithm 1. The teacher model $\mathcal{M}_T$ is used as the reference model and each device's model is compressed from $\mathcal{M}_T$. In each round the model inference time $T(\mathcal{M}_i(x_i))$ is tested, and model compression process continues if the model inference time exceeds the time constraint $\mathcal{T}$. Each pruning process is only for one convolutional layer within a particular stage, and to ensure that each pruning selection is the current best. We refer to the heuristic that layers with smaller average weight contributes less to the final inference performance [18], in each depth-scaling round the layer with minimum average weight is pruned and model is updated with smaller size using *pruneMinAWLayer()* method. The $AW_{j,k}$ is calculated by finding the average weight of the k-th layer in the j-th stage. We count the average weights after each cut and retrain the model after completing a cut to ensure that each decision is the current best. Since the pruned model incurs less floating point operations per second (FLOPS), the inference time reduces accordingly. And this process continues until the model inference time becomes smaller than $\mathcal{T}$. At this point, the IoT device acquires a student model whose inference time meets the time constraint and maintains the compatibility for the KD process. However, one extreme case is that when the model has been compressed to the smallest form while still cannot meet the inference time limit, and in this case the device will not be used in the group to ensure the efficiency of the whole group.

Algorithm 1: Depth Scaling based Model Design

```
Input: The teacher model $\mathcal{M}_T$, time constraint $\mathcal{T}$
Output: Customized student model $\mathcal{M}_i$ for device $i$
Data: Testing input $x_i$ of device $i$
1 $\mathcal{M}_i \leftarrow \mathcal{M}_T$
2 while $T(\mathcal{M}_i(x_i)) > \mathcal{T}$ do
3     for j=1 : number of stages do
4         for k=1 : number of layers do
5             $AW_{j,k} = \frac{\sum_{a=1}^{\mathcal{H}_j}\sum_{b=1}^{\mathcal{W}_j}\sum_{c=1}^{\mathcal{C}_j}|w_{a,b,c}|}{(\mathcal{H}_j \times \mathcal{W}_j \times \mathcal{C}_j)}$
6         end
7     end
8     $\mathcal{M}_i \leftarrow pruneMinAWLayer(\mathcal{M}_i)$
9 end
```

IV. Knowledge Distillation based DNN Model Training

In this section, we detail the training process of the custom DNN model generated in the previous step. Since the models might be compressed due to the hardware constraint of each device, the performance of the local DNN model might degrade. In *Ensen*, we use knowledge distillation as the model training approach to effectively improve the classification accuracy for each IoT device, and thus improve the final classification accuracy for the whole inference group.

The overall training process of KD in *Ensen* is depicted in Figure 4. Three different distillation losses are integrated, i.e., the *attention distillation losses*, *logits distillation losses*, and *cross entropy distillation losses*, to ensure the convergence of the student model and improve its final classification performance.

A. Imitating Intermediate Behaviors : Attention Distillation

In convolutional neural networks (CNN), different stages in the network plays different roles in the feature extraction. The model on the device $\mathcal{M}_i$ is customized by cropping the convolutional layers within the stages, so using the teacher model to repair each stage is necessary to improve the performance. we first propose to use the attention distillation loss to allow student models to learn the in-network intermediate behavior from the teacher model. As illustrated in Figure 4, between each stage $\mathcal{S}_i$ and $\mathcal{S}_j$, we introduce an attention map as a representation of the behavior of the previous stage. Assume that the output 3D tensor from the stage $\mathcal{S}_i$ is $\mathcal{R} \in \mathbb{R}^{\mathcal{C}_i \times \mathcal{I} \times \mathcal{Q}}$, where $\mathcal{C}_i$ is the number of feature planes (channel dimensions), and $\mathcal{I}$ and $\mathcal{Q}$ are the spatial dimensions of the feature plane. We use a mapping function $\mathcal{F} : \mathbb{R}^{\mathcal{C}_i \times \mathcal{I} \times \mathcal{Q}} \rightarrow \mathbb{R}^{\mathcal{I} \times \mathcal{Q}}$ to map the 3D tensor to a 2D attention map. We refer to the attention map construction approach in [19] to construct the attention map for stage i as:

$$\mathcal{A}_i = \mathcal{F}(\mathcal{R}) = \sum_{i=1}^{\mathcal{C}_i} |\mathcal{R}_i|^2 \tag{2}$$

where $|\mathcal{R}_i|^2$ refers to the power two dot product of 2D tensor $\mathcal{R}_i$, and $\mathcal{F}(\mathcal{R})$ therefore superimposes the result of dot products of all channels and flattens the 3D tensor $\mathcal{R}$ to a 2D spatial attention map $\mathcal{A}_i$. The generated attention maps provide efficient summary of the behaviors of all stages in the network. For the cropped stages in the student model, this learning approach helps to repair the performance of the crippled stages. The attention distillation computes the distance between the attention maps of all stages of the student models and the teacher model. To form an overall representation of the attention maps of both the student models and teacher model, we concatenate the attention map of all stages in the student model as $\mathcal{A}_S = [\mathcal{A}_1^S \, \mathcal{A}_2^S \, \cdots \, \mathcal{A}_n^S]$, here $\mathcal{A}_i^S$ represents the attention map of the i-th stage of student model. And similarly, the teacher's concatenated attention map is expressed as $\mathcal{A}_T = [\mathcal{A}_1^T, \mathcal{A}_2^T, \cdots, \mathcal{A}_n^T]$. We perform L2 normalization on $\mathcal{A}_S$ and $\mathcal{A}_T$ and calculate the attention distillation loss function as:

$$\mathcal{LOSS}_{at} = \sum_{i=1}^{n} \| \frac{\mathcal{A}_i^S}{\|\mathcal{A}_i^S\|_2} - \frac{\mathcal{A}_i^T}{\|\mathcal{A}_i^T\|_2} \|_p \tag{3}$$

where n is the number of stages, and p is the norm type, which is set to $p = 2$ in this work. $\mathcal{LOSS}_{at}$ therefore is used as one loss function in the training process to train the intermediate behavior of stages in the student model. The smaller the $\mathcal{LOSS}_{at}$, the closer the expression of stages between the student model and the teacher model.

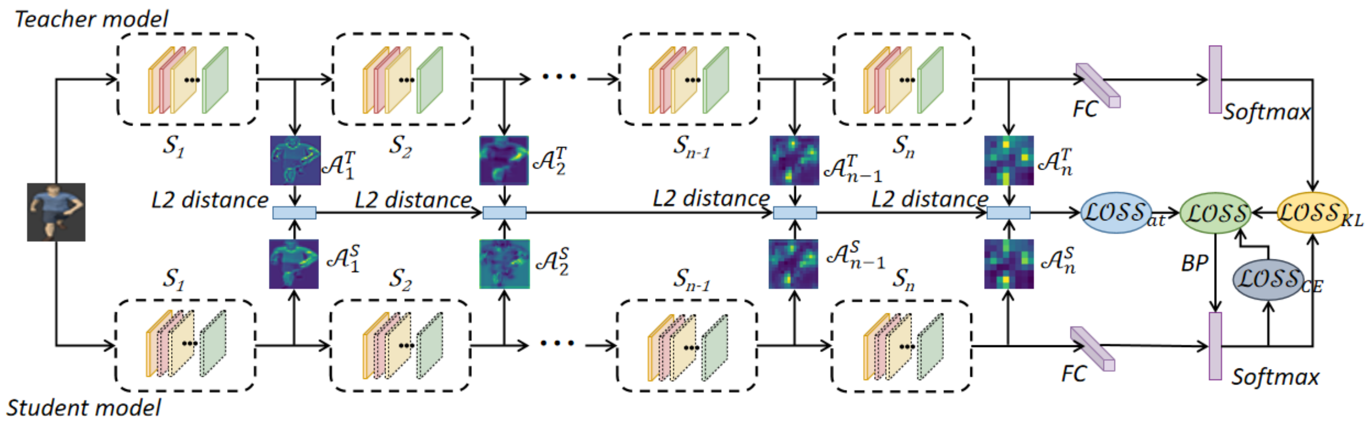

Fig. 4: KD-based model training in *Ensen*

B. Imitating Overall Behaviors : Logits Distillation

Logits knowledge distillation aims to promote the global knowledge transfer from the teacher model to the student models. Neural networks usually use softmax layers to generate class prediction probabilities. In traditional training approaches (known as hard target [20]), the probability of negative classes are usually treated as uniformly distributed. Different from the traditional training process, Logits knowledge distillation takes this as a soft target [20]. That is, Logits knowledge distillation incorporates the negative class probability distributions. Therefore, Logits knowledge distillation allows more information to be learned in the student model in the training process than the traditional hard target method, and improves the performance of the student model.

Inspired by [20], in *Ensen* we use a specific temperature T' to soften the softmax output and make the output probability smoother to further amplify the information carried by the negative classes:

$$q_i = \frac{e^{\frac{z_i}{T'}}}{\sum_{i=1}^{c} e^{\frac{z_i}{T'}}} \tag{4}$$

where q_i is the probability of the i-th class in the softened softmax output, z_i is its Logits value, c is the total number of classes, and T' is the temperature. The higher the T', the smoother the output probability distribution of the softmax is, and as a result the information carried by the negative classes will be relatively amplified.

In Logits knowledge distillation, we introduce another loss function between the softened softmax output of the teacher model and student models to allow student models to learn the overall behavior of the teacher model. To measure the similarity of two probability distributions, we use KL (Kullback-Leibler Divergence) [21] to calculate distance, and therefore the Logits knowledge distillation loss function is defined as:

$$\mathcal{LOSS}_{KL} = \sum_{i=1}^{c} (q_i^T)^{T'} \log(\frac{(q_i^T)^{T'}}{q_i^S}) \tag{5}$$

where q_i^T is the probability value of the i-th class in the softened softmax output of the teacher model, and q_i^S is the respective output of the student model.

C. Final Knowledge Distillation

The attention distillation and Logits distillation allow the student models to imitate both the intermediate behavior and the overall behavior of the teacher model. To allow student models to also be able to learn on its own, in *Ensen* we incorporate the the cross entropy loss ($\mathcal{LOSS}_{CE}$) produced by the hard predictions and true label of the student models [22] as the third loss function. Finally, the three loss functions are combined to constitute the final loss function in the model training process as:

$$\mathcal{LOSS} = \alpha \cdot \mathcal{LOSS}_{CE} + \beta \cdot \mathcal{LOSS}_{KL} + \gamma \cdot \mathcal{LOSS}_{at} \tag{6}$$

where α, β, and γ are the weights of three loss functions respectively, and referring to [23] in *Ensen* these weights are empirically set to $\alpha = 0.7, \beta = 0.15, \gamma = 0.15$. The effectiveness of the knowledge distillation process is validated in the evaluation section later.

V. Dynamic Voting Process

With the customized model design and KD-based training, each IoT device acquires a tailored built DNN model which is able to perform inferences locally on the device and meet the time constraints. A deep ensemble is then formed with neighboring devices to enable collaborative inferences. When a device collects an inference sample, it will send the sample to other devices around it and become an anchor node who aggregates all the local predictions and generates the final predictions. To enable efficient collaboration, two main issues need to be addressed: (1) *Local prediction aggregation.* The prediction results generated by all participating devices need to be efficiently aggregated to generate final inference results with high-accuracy. (2) *Dynamic deep ensemble management.* Group members need to be dynamically profiled and members with noisy predictions need to be filtered out to maintain robustness of the whole group performance.

A. Efficient Local Prediction Result Aggregation

In each round of prediction, each devices i runs the local model $\mathcal{M}_i$ with local input x_i captured by device's sensors onboard and generates a local vote, i.e., the prediction y_i. The anchor node hence collects the local prediction vector $\mathbf{y} = [y_1, y_2, ..., y_n]$ with n devices in the deep ensemble. The aggregation function $\mathcal{G}$ takes the local prediction vector and maps to the final ensemble prediction result. One simplest form of the aggregation function is to output the label with maximum occurrence in $\mathbf{y}$, that is:

$$\mathcal{G}(\mathbf{y}) = \arg\max_{y} |\{y_i | y_i \in \mathbf{y}, y_i = y\}| \tag{7}$$

where $|\{y_i | y_i \in \mathbf{y}, y_i = y\}|$ returns the cardinality, or number of occurrence of the vote y in $\mathbf{y}$. However, such simple aggregation suffers from several limitations. First, such aggregation generates group prediction label that only represents the final classification result and does not include the probability suggestions for other classes. Second, the aggregation treats all devices equally, however in practice some devices might be more robust and generate better quality of predictions than others. Therefore, to address the first limitation, it is a better option to refer to the logits output of all devices. Since the softmax layer of the model includes the predicted probabilities for all classes, the anchor device instead collects the softmax layer information of all participating, and averages the predicted probabilities of all classes using the following aggregation function:

$$\mathcal{G}'(\mathbf{y}) = \arg\max_{y_j} \frac{\sum_{i=1}^{N} q_{i,j}}{N}; \tag{8}$$

where N is total number of label classes, $q_{i,j}$ represents the softmax value of the j-th class reported by device i. In this way, the anchor device combines the average prediction probability of all devices for each class to get the final softmax layer, and output the final predict label with the maximum combined softmax value.

However, the above two vote aggregating approaches still do not address the second limitation, that is, the classification capabilities of heterogeneous devices are usually different, but they are treated equally in the aggregation process. The difference of classification capability is mainly due to two reasons. One is that the local data distribution in the training of each device's model is different, and the second is that the computational power of different devices are usually different, which is reflected in the different customized DNN models designed for each device. As a result, it is essential to assign different voting weights to different local predictions based on the capabilities of the device, so that members with more capabilities can play a greater role in the voting and improve the final performance of the system. To estimate such 'capability', usually the average classification accuracy of the model is used as a direct metric. However, in *Ensen* the distribution of training data of different IoT devices can be unbalanced, resulting in an unbalanced performance of different devices to classify different classes of data. Therefore, as a fine-grained estimation the performance of each device on each class of input, we pre-estimate the confidence with a testing input dataset. And the confidence is measured using the F1-score on the testing input:

$$c_{i,j} = \frac{2 * Precision_{i,j} * Recall_{i,j}}{Precision_{i,j} + Recall_{i,j}} \tag{9}$$

where $c_{i,j}$ is the confidence of the device i on label class j, and $Precision_{i,j}$ and $Recall_{i,j}$ are precision and recall of class j on device i for the testing input dataset. Note that this process needs to be measured only once at the offline stage and does not incur additional computation burden in the online inference stage. Finally, the aggregation based on the probabilities of each label class in the softmax layer is determined as:

$$\mathcal{G}''(\mathbf{y}) = \arg\max_{y_j} \frac{\sum_{i=1}^{N} c_{i,j} \cdot q_{i,j}}{N} \tag{10}$$

where the final aggregation function $\mathcal{G}''$ incorporates both the difference in the devices and generates probabilities for all label classes.

B. Dynamic Deep Ensemble Management

The votes of different devices in the deep ensemble provide different opinions of different devices to the same sensing target, and in the voting stage weighted approach is taken to reflect the expected contribution of each device on the inference result as discussed in the previous section. However, this proactive voting approach may pose security concerns. In some extreme cases, the participating devices can be malicious and deliberately votes random predictions to degrade the overall performance of the ensemble group. Especially for real-time tasks in traffic systems, malicious intrusions may lead to serious consequences. As a result, in the voting process the contribution of each device should be profiled over time and used as a factor for dynamic ensemble group management.

To profile the real-time performance of all devices, in *Ensen* the anchor node maintains a sliding window with size $K \times n$, where K is the length of the window and n is the number of devices in the ensemble group. In t-th inference round, the participating device i uploads the $Softmax_i$ and the corresponding local prediction y_i. The window value for device i at time t is set to $W(i,t) = 1$ if $y_i = \mathcal{G}''(\mathbf{y})$ as calculated in Equation (10), otherwise $W(i,t) = 0$. The window value is set in this way so that if $W(i,t) = 1$ the device's local inference matches with the final aggregated result and the device is a positive contributor at this round. We use $\mathcal{P}_{i,t+1}$ as the participating probability of device i at $t+1$ round, and it is defined as:

$$\mathcal{P}_{i,t+1} = \begin{cases} \epsilon & \sum\limits_{j=t-K}^{t} W(i,j) = 0 \\ \frac{2(1-\epsilon)}{K} W(i,j) + \epsilon & 0 < \sum\limits_{j=t-K}^{t} W(i,j) \leq \frac{K}{2} \\ 1, & \frac{K}{2} < \sum\limits_{j=t-K}^{t} W(i,j) \leq K \end{cases} \tag{11}$$

At $t+1$ round, the device participates the ensemble inference process with the probability $\mathcal{P}_{i,t+1}$. When $\sum_{j=t-K}^{t} W(i,j) = 0$, the $\mathcal{P}_{i,t+1}$ is set to a minimum value ϵ (which is set to 0.1 in *Ensen*) to avoid being permanently filtered out from the system, otherwise the probability increases as the contribution of the device in the current window improves. In this way, the participating probability of each device is dynamically changed based on its recent performance, and the robustness of the group inference further improves. The evaluation of the dynamic deep ensemble management scheme is detailed in Section VI.

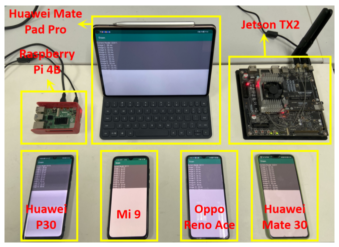

Fig. 5: Heterogeneous devices used in the evaluation

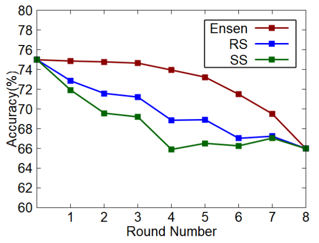

Fig. 6: Efficiency of the proposed customized model design approach

VI. Evaluations

In this section, we will provide the detailed evaluations on the respective modules and the overall performance of the *Ensen* system.

Experimental Settings. To evaluate the performance of the *Ensen* system, we implemented the system on 7 different types of common IoT devices, including 1 raspberry Pi, 1 Jetson TX2 development board, 4 different models of smartphones, and one tablet as shown in Figure 5. The detailed specification of 7 devices are listed in the Table I. These devices are with different hardware capabilities and are used as heterogeneous devices to evaluate the performance of *Ensen*. Although the *Ensen* is not limited to any specific type of deep neural networks, in *Ensen* we use the popular VGG-16 [24] as the reference teacher model to guide the design of the students models for IoT devices. VGG-16 consists of 5 stages, and each stage can have up to 3 convolutional layers. In the customized model design, the Algorithm 1 is used to generate the customized student models based on the original VGG-16 model. We use pytorch [25] to implement the DNN models and use Open Neural Network Exchange (ONNX) [26] to export the models to different embedded platforms.

Dataset. We use CINIC-10 [27] as the data set for evaluation. CINIC-10 is a popular data set for image recognition and is an augmented extension of the CIFAR-10. It includes all images from the CIFAR-10 (60,000 images, 32x32 RGB pixels) and a selection of ImageNet database images (210,000 images downsampled to 32x32 resolution). In total 270,000 images are included in the CINIC-10 dataset. 90,000 uniformly distributed images are randomly selected from the CINIC-10 as the training dataset for the teacher model. 150,000 other images are randomly distributed to the 7 devices. On each device, the total number of images is different, the number of images in each class is also different, forming a non-i.i.d local data distribution. Among the rest images, 10,000 are used to measure the accuracy of the trained models, and the last 20,000 are used as the testing dataset to evaluate the performances of the whole system.

A. *Efficiency of Customized Model Design*

The efficiency of the model design affects the overall ensemble sensing performance. In the evaluation, we use the VGG-16 as the teacher model and applies the depth-scaling based model design algorithm as described in Algorithm 1 for model search for each heterogeneous device. To compare the

TABLE I: Specifications of the devices used in the evaluation

Device ID	Device Name	Abbreviation	Specifications
1	Raspberry Pi 4b	R1	Broadcom BCM2711
2	Huawei P30	P1	Kirin 980
3	Mi 9	P2	Snapdragon 855
4	OPPO Reno Ace	P3	Snapdragon 855P
5	Huawei Mate30	P4	Kirin 990
6	Huawei Mate Pad Pro	T1	Kirin 990
7	Jetson TX2	J1	NVIDIA Denver 2

efficiency of the algorithm, we compare two different model search algorithms:

- *Random Scaling (RS)* [28]. In this approach, at each round a random convolutional layer is dropped from the current model to reduce the inference time and to meet the time constraint.
- *Sequential Scaling (SS)* [18]. At each round, the convolutional layer from each stage is dropped one-by-one sequentially from the last stage to the first stage.

Figure 6 shows the performance of the *Ensen* in the customized model design. In the experiment, we set the time constraint to be 0 to monitor all three model search processes. As shown in the figure, at round 0, all three processes have the same accuracy since they all start with the same teacher model. With a time constraint 0, all three model search algorithms take 8 rounds to stop to converge to the minimum network architecture, i.e., leaving only one layer in each stage of the original VGG-16 model. This is expected since no model can reach time constraint 0 and hence all model searching process will stop at the simplest network model. However, it can be observed that at each intermediate round, the *Ensen* model design algorithm all has the highest accuracy, which indicates that at each round the generated model of *Ensen* has better performance than *RP* and *SP*. This result indicates that the model design algorithm is efficient in making customized models for IoT devices.

Figure 7 shows the model inference time changes as the model evolves in the pruning process. For each inference task, we require each device to perform model inference for an input of one image. At round 0, all 7 devices have the original version of the VGG-16 model. As a result, it takes the longest time to complete an inference task for all devices. For Raspberry Pi, the inference time of 150ms is longest among all devices due to the limited computation power available on Pi, and the Jetson TX2 board completes task fastest at around 40ms. It can be seen that the slowest device completes a task requiring almost 4 times the time of the fastest device, which means that if both of them use the same DNN model in the ensemble sensing device, in around 75% of the time the fastest device will be waiting for the slowest device to complete and the computational resources are significantly wasted. As shown in Figure 7, as the pruning round goes on, the inference time of all devices decreases accordingly, and at round 8 all devices' model reduce to the simplest form with one layer per stage. If we set the time constraint to be 35ms, the 7 devices (R1, P1, P2, P3, P4, T1, J1) will stop at round 8, 7, 6, 6, 5, 4, 2, respectively following Algorithm 1, resulting in the respective inference time of 35ms, 34.4ms, 34.8ms, 34.1ms,

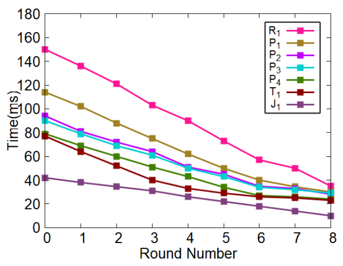

Fig. 7: Inference time changes with different customized models

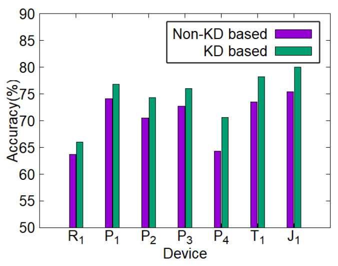

Fig. 8: Efficiency of KD-based training approach in *Ensen*

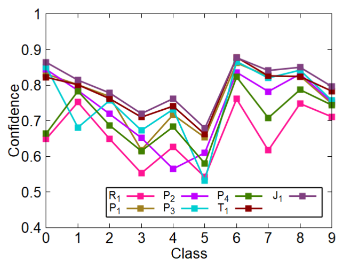

Fig. 9: Confidence distribution of each predicting class fro all devices

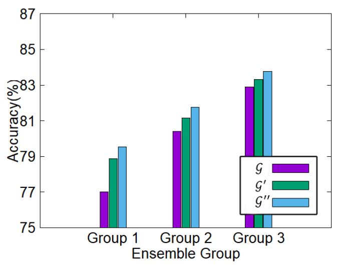

Fig. 10: Efficiency with different local prediction aggregation functions

34ms, 33.2ms, and 34.5ms. With the customized model, each device completes the inference task around the same time, resulting in little wastage on the computational resources and efficient cooperation in the ensemble sensing.

B. *Efficiency of KD-based Model Training*

As discussed in Section IV, in *Ensen* we integrate different loss functions in the knowledge distillation process, which allows the students models in the IoT devices to be able to efficiently learn from the global model and also adapt to their local data distribution. Figure 8 shows the comparison between the conventional training approach where only local data is used in the training, and the KD-based approach where the proposed loss integration in Section IV is adopted. The figure shows the accuracy comparison of all models for the 7 devices. It can be seen that the KD-based approach efficiently improves the accuracy of all devices, with an average increase of 4.02% in the classification accuracy. This results shows that the KD-based approach effectively improves the classification performance in the edge-centric sensing paradigm where both global knowledge and local data is available, and can be efficiently used in *Ensen* to improve the overall performance.

C. *Efficiency of Local Prediction Aggregation*

To evaluate the impact of local prediction aggregation, we study the final aggregation performance using three aggregation functions $\mathcal{G}$, $\mathcal{G}'$, and $\mathcal{G}''$ proposed in Section V. The aggregation functions are proposed with the assumption that different devices have different confidences in each predicting class. To validate the assumption, we first study the confidence distribution of each device on each predicting class. Figure 9 shows the confidence of each class for all 7 devices. It can be seen from the figure that, for devices with more sophisticate customized student models due to higher computational capabilities such as the Jetson TX2, the confidence for all classes are generally higher than those with less sophisticate customized model such as Raspberry Pi. However, due to the imbalanced local data distribution, some device with lower overall accuracy can perform better on some specific predicting classes. For example, R1 has lower overall accuracy than P3, but for class 1, R1 has higher confidence than that of P3. This results validate our assumption and show confidence based local prediction aggregation is important to provide fine-grained assessment to the weight of each local prediction. And in the real application scenarios such as transportation systems, the devices can be mobile, and they might join or leave the group frequently. In order to simulate the changing dynamics of the group members, We divide the 7 devices to the following three ensemble groups to compare the performance of different aggregation functions:

– *Group 1. R1, P1, P2*
– *Group 2. R1, P1, P2, P3, P4*
– *Group 3. R1, P1, P2, P3, P4, T1, J1*

Figure 10 shows the impact of three different aggregation functions in the ensemble inference process. As shown in the figure, as the group size becomes bigger, the accuracy of final aggregated predictions improves from 79.53% to 83.76% accordingly, and this shows the positive impact of local prediction aggregation in the ensemble sensing. It also shows that *Ensen* allows devices to leave or join groups at any time, but typically, the accuracy of ensemble sensing increases as the number of members involved in sensing increases. Within each group, the performance of $\mathcal{G}''$ which integrates the confidence of each class of each device all outperforms the other two and achieves the highest accuracy. For Group 3 with $\mathcal{G}''$, the final accuracy reaches 83.76%, resulting in an increase of 9.19% in the average accuracy comparing to the case without ensemble sensing. The result shows that the ensemble sensing proposed in this work efficiently improve the overall accuracy of classifications.

D. *Efficiency of Dynamic Deep Ensemble Management*

To evaluate the dynamic deep ensemble management, we add two malicious devices into the group and test whether these devices can be detected and filtered out from the ensemble group. Both of these two devices have only 10% classification accuracy. The window size K is set to 10 in the evaluation. As shown in Figure 11, each of 9 devices can have either state 0 (indicating the local prediction is not aggregated by the anchor node) or state 1 (indicating the local prediction is used in aggregation). In the first 10 rounds, all devices' local predictions are aggregated. However, after round 10, the probability of Device 8 (D8) and Device 9 (D9) being aggregated significantly reduces, indicating that the ensemble group has detected their abnormal behavior and lower the impact of these devices. After round 50, we reset Device 9 with normal DNN models, and we can observe that after that the participating probability of Device 9 increases accordingly and become a normal participant in the group. The result indicates that the deep ensemble management scheme

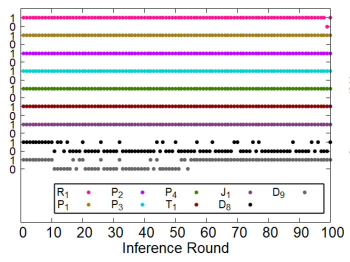
Fig. 11: Dynamic deep ensemble evolvement

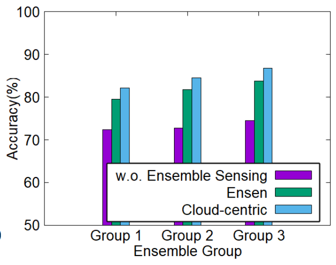
Fig. 12: Final classification performance for VGG-16

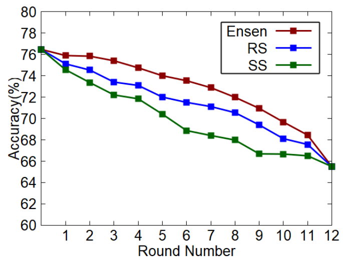
Fig. 13: Efficiency of model design for ResNet

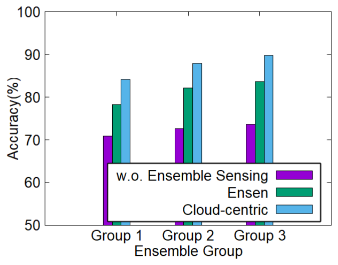
Fig. 14: Final classification performance for ResNet

proposed in *Ensen* works well and dynamically adapts to the performance of each individual member in the group to achieve a robust performance of the whole group.

E. Comparison with Cloud-centric and Non-ensemble Method

Although the edge-centric approach provides many benefits including privacy protection, low network latency and better real-time performances, etc., it is still important to understand the final performance of the proposed ensemble sensing approach comparing with the conventional cloud-centric approach, where all local data is uploaded to the cloud to train the global teacher model, and all the inference is done remotely on the cloud without time constraint. Figure 12 shows the comparison between *Ensen* and the conventional cloud-centric scheme for the above three ensemble group settings. As can be seen from the figure, although the accuracy of *Ensen* slightly decreases compared with the cloud-centric scheme, both approaches achieve comparable performances with an average of 2.8% differences in the accuracy, while *Ensen* also enjoys numerous other benefits due to the adoption of the edge-centric scheme. In comparison to the average accuracy without ensemble sensing, where all inferences are done locally on the IoT devices without aggregation, *Ensen* achieves an accuracy gain of 9.19%.

F. Generalization to other CNN Models

The *Ensen* can be extended to different types of convolutional neural networks. Figure 13 and Figure 14 show the efficiency of *Ensen* for the ResNet [29], another type of popular CNN model. It can be observed that similar results are achieved on ResNet, while the total round number increases due to more complex architecture of ResNet. Overall, using ResNet the final aggregated accuracy is comparable to the cloud-centric approach, and reaches 78.25%, 82.15%, and 83.7% for Group 1, Group 2, and Group 3 respectively, resulting an accuracy gain of 10.11% compared with average accuracy without aggregating local predictions.

VII. Related work

A. DNN Model Compression

The computational intensive DNN model execution has been a well-known problem that hinders the deployment of DNN models on hardware constrained IoT devices in edge computing [30]–[32]. To address this problem, various optimization techniques have been proposed to fit DNN models into low-cost embedded devices. Model compression [33] has attracted a stream of studies which aim to reduce the size and computation of common DNN models. In the literature, various techniques have been used for model compression, such as network pruning [34], quantization [35], Low-rank factorization [36], etc. Different from model compression, another approach which gains attention in the literature recently aims to build new smaller networks from existing models based on "teacher-student network" [20]. This paper follows this line of research, however, a different approach is taken to design customized models based on the computational power of the device and the time constraint. We propose and integrate new loss functions into the training process of the tailor-design model, and this inherently helps the student models to reduce the accuracy loss due to compression.

B. Ensemble Learning

Different from federated learning [37], ensemble learning improves the predictive performance of a single model by training multiple models and combining their predictions [38]. Ensemble learning has been widely used, such as Boosting [39], Stacking [40], Bagging [39], etc. In recent years, researchers have tried to use ensemble methods in deep learning scenarios, such as speech recognition [41], automatic optic cup and disc segmentation [42], remote sensing scenes [43], and so on. [14] proposed a framework for implementing DNNs on edge devices by allowing multiple users to form a deep ensemble. Our work is inspired by the concept of ensemble learning, however, to solve the challenges of time-constrained synchronized inference, multiple novel techniques have been proposed in *Ensen* to achieve robust ensemble sensing performance.

In summary, although this work shares some common concepts with model compression and ensemble learning, we mainly focus on different perspective, i.e., the imbalanced distribution of local data distribution and propose an ensemble sensing scheme for heterogeneous IoT devices. The proposed *Ensen* system is unique in terms that a customized model design approach is used to solve the time-constrained synchronized inference problem, and various novel techniques in modle training and sensing group management have been integrated to achieve a robust performance of ensemble sensing using heterogeneous IoT devices under time constraint.

VIII. Limitations and Future Work

The *Ensen* proposes an ensemble sensing approach towards collaborative sensing. Customized DNN models are designed for heterogeneous devices for time-constrained collaboration. Knowledge distillation is used to train the models to improve the performance, and finally ensemble groups of neighboring devices are dynamically form to perform the sensing tasks together. Due to the model deployment process and the collaborative approach, additional communication costs are inevitably incurred. For example, the anchor nodes need to collect the local predictions and send the final inference results to other devices.

Despite the above limitations, *Ensen* still has multiple advantages and efficiently improves the overall group performance, which makes it an effective approach and can be applied to several practical application scenarios, such as intelligent transportation systems. Some future extensions are possible to further improve the system performance, e.g., model compression optimization. When designing models for devices, other efficient methods can be incorporated. For example, pruning the width of the convolutional layers or using other techniques such as quantization to further optimize the model design.

IX. Conclusions

In this work, we propose *Ensen*, a collaborative inference system using heterogeneous IoT devices in intelligence transportation systems. *Ensen* incorporates various novel techniques such as customized DNN model design, KD-based model training, dynamic deep ensemble management, etc., to achieve efficient ensemble sensing under time constraints. Extensive evaluations show that *Ensen* works for different types on common IoT devices and achieve improved performances compared with conventional approaches.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (61972263, 62073225), Natural Science Foundation of Guangdong Province (2019A1515011608), and the Stable Support Plan for Higher Education Institutions in Shenzhen (20200810113310001).

References

[1] H. Song, R. Srinivasan, T. Sookoor, and S. Jeschke, *Smart cities: foundations, principles, and applications*. John Wiley & Sons, 2017.

[2] H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, *Cyber-physical systems: foundations, principles and applications*. Morgan Kaufmann, 2016.

[3] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE access*, vol. 4, pp. 766–773, 2016.

[4] C. Chen, L. Liu, T. Qiu, K. Yang, F. Gong, and H. Song, "Asgr: An artificial spider-web-based geographic routing in heterogeneous vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1604–1620, 2018.

[5] D. Jiang, L. Huo, Z. Lv, H. Song, and W. Qin, "A joint multi-criteria utility-based network selection approach for vehicle-to-infrastructure networking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3305–3319, 2018.

[6] J. Yang, Y. Han, Y. Wang, B. Jiang, Z. Lv, and H. Song, "Optimization of real-time traffic network assignment based on iot data using dbn and clustering model in smart city," *Future Generation Computer Systems*, vol. 108, pp. 976–986, 2020.

[7] Y. Sun, X. Yu, R. Bie, and H. Song, "Discovering time-dependent shortest path on traffic graph for drivers towards green driving," *Journal of Network and Computer Applications*, vol. 83, pp. 204–212, 2017.

[8] Z. Xiong, M. Li, Y. Ma, and X. Wu, "Vehicle re-identification with image processing and car-following model using multiple surveillance cameras from urban arterials," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[9] W. Y. Choi, S.-H. Lee, and C. C. Chung, "On-road object collision point estimation by radar sensor data fusion," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[10] C. Chen, C. Wang, T. Qiu, Z. Xu, and H. Song, "A robust active safety enhancement strategy with learning mechanism in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5160–5176, 2019.

[11] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 716–723, 2017.

[12] K. Qian, T. Koike, T. Nakamura, B. Schuller, and Y. Yamamoto, "Learning multimodal representations for drowsiness detection," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[14] N. Shlezinger, E. Farhan, H. Morgenstern, and Y. C. Eldar, "Collaborative inference via ensembles on the edge," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8478–8482.

[15] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "Sedmdroid: An enhanced stacking ensemble framework for android malware detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 984 – 994, 2021.

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

[17] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3967–3976.

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[19] N. Komodakis and S. Zagoruyko, "Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer," in *International Conference on Learning Representations (ICLR)*, 2017.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[21] T. Van Erven and P. Harremos, "Rényi divergence and kullback-leibler divergence," *IEEE Transactions on Information Theory (TIT)*, vol. 60, no. 7, pp. 3797–3820, 2014.

[22] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[23] D. Walawalkar, Z. Shen, and M. Savvides, "Online ensemble model compression using knowledge distillation," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 18–35.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 8026–8037, 2019.

[26] "Open neural network exchange (onnx)," https://onnx.ai/.

[27] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "Cinic-10 is not imagenet or cifar-10," *arXiv preprint arXiv:1810.03505*, 2018.

[28] S. Anwar and W. Sung, "Coarse pruning of convolutional neural networks with random masks," 2016.

[29] 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 (CVPR)*, 2016, pp. 770–778.

[30] X. Zhou, X. Yang, J. Ma, I. Kevin, and K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in iot," *IEEE Internet of Things Journal*, 2021.

[31] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, "Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart iot," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 588–12 596, 2021.

[32] S. Chen, Y. Tao, D. Yu, F. Li, and B. Gong, "Distributed learning dynamics of multi-armed bandits for edge intelligence," *Journal of Systems Architecture*, vol. 114, p. 101919, 2021.

[33] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 784–800.

[34] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.

[35] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4852–4861.

[36] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6655–6659.

[37] X. Zhou, W. Liang, J. She, Z. Yan, I. Kevin, and K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5308–5317, 2021.

[38] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[39] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[40] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[41] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Fifteenth Annual Conference of the International Speech Communication Association (ISCA)*, 2014.

[42] J. Zilly, J. M. Buhmann, and D. Mahapatra, "Glaucoma detection using entropy sampling and ensemble learning for automatic optic cup and disc segmentation," *Computerized Medical Imaging and Graphics (CMIG)*, vol. 55, pp. 28–41, 2017.

[43] S. Akodad, S. Vilfroy, L. Bombrun, C. C. Cavalcante, C. Germain, and Y. Berthoumieu, "An ensemble learning approach for the classification of remote sensing scenes based on covariance pooling of cnn features," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

Xingyu Feng is currently pursuing the Ph.D. degree in the School of Computer and software, Shenzhen University. He received the M.E. degree from Shenzhen University in 2020 and the B.S. degree from Jiangxi University of Finance and Economics in 2017. His primary research interests mainly include Internet of Things, mobile edge computing and deep learning.

Chengwen Luo received the PhD degree from the School of Computing, National University of Singapore (NUS), Singapore. He is currently an associate professor in the College of Computer Science and Software Engineering, Shenzhen University (SZU), China. His research interests include mobile and pervasive computing and security aspects of Internet of Things.

Bo Wei has been an assistant professor in the School of Computing and Communications at Lancaster University. He was a Postdoctoral research assistant in University of Oxford. He obtained his PhD degree in Computer Science and Engineering in 2015 from the University of New South Wales, Australia. His research interests are Mobile Computing, Internet of Things, and Wireless Sensor Networks.

Jin Zhang currently is an associate researcher College of Computer Science and Software Engineering, Shenzhen University. Jin worked in CSE, UNSW and CSIRO as a research engineer and PhD scholarship holder from 2013-2017. During this period, Jin obtained his PhD degree as well. Jin combined AI, machine learning technology with IoT devices i.e. commercial WiFi laptop or router for smart IoT sensing.

Jianqiang Li received his B.S and Ph.D. Degree from South China University of Technology in 2003 and 2008. He is currently a professor in the College of Computer Science and Software Engineering, Shenzhen University. His major research interests include embedded systems and Internet of Things.

Huihui Wang received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in 2013. She is an Associate Professor at St. Bonaventure University, St. Bonaventure, NY, USA and a Program Director at National Science Foundation, Alexandria, VA, USA. She has authored or coauthored over 50 papers. Her current research interests are cyber physical systems, Internet of Things, and engineering education. Dr. Wang was the Vice Chair of IEEE JAX Section, a member and the Program Chair of ECE Division of ASEE, as well as a member of Florida Engineering Society (FES), and the Treasurer of the Florida Engineers in Education (FEE) practice section of FES. She is also a member of ASME. She has served as a Technical Program Committee Chair/member as well as a Reviewer for international conferences and journals. She is an active panelist of NSF, NASA, and fellowships.

Weitao Xu is an Assistant Professor at the Department of Computer Science at City University of Hong Kong. He obtained his PhD degree from the University of Queensland in 2017. His research generally focuses on IoT such as smart sensing, IoT security, IoT+AI, and wireless networks.

Mun Choon Chan graduated with a BS in Computer and Electrical Engineering from Purdue University and Ph.D. from Columbia University. He was a Member of Technical Staff in the Networking Research Laboratory, Bell Labs, Lucent Technologies before joining NUS. He is currently Professor in the Department of Computer Science, School of Computing, National University of Singapore.

Victor C. M. Leung (Life Fellow, IEEE) is currently a Distinguished Professor of computer science and software engineering at Shenzhen University, China. He is also an Emeritus Professor of electrical and computer engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems, University of British Columbia (UBC), Canada. His research interests include wireless networks and mobile systems. He has published widely in these areas. He is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, IEEE Network, and several other journals. He is a fellow of the Royal Society of Canada (Academy of Science), the Canadian Academy of Engineering, and the Engineering Institute of Canada.