

7-26-2024

## Efficient Deep Neural Network Compression for Environmental Sound Classification on Microcontroller Units


SHAN CHEN

Na MENG

HAOYUAN LI

WEIWEI FANG

Follow this and additional works at: <https://journals.tubitak.gov.tr/elektrik>

 Part of the [Computer Engineering Commons](#), [Computer Sciences Commons](#), and the [Electrical and Computer Engineering Commons](#)

### Recommended Citation

CHEN, SHAN; MENG, Na; LI, HAOYUAN; and FANG, WEIWEI (2024) "Efficient Deep Neural Network Compression for Environmental Sound Classification on Microcontroller Units," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 32: No. 4, Article 2. <https://doi.org/10.55730/1300-0632.4084>

Available at: <https://journals.tubitak.gov.tr/elektrik/vol32/iss4/2>



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

This Research Article is brought to you for free and open access by TÜBİTAK Academic Journals. It has been accepted for inclusion in Turkish Journal of Electrical Engineering and Computer Sciences by an authorized editor of TÜBİTAK Academic Journals. For more information, please contact [pinar.dundar@tubitak.gov.tr](mailto:pinar.dundar@tubitak.gov.tr).

## Efficient deep neural network compression for environmental sound classification on microcontroller units

Shan CHEN<sup>1</sup> , Na MENG<sup>1</sup> , Haoyuan LI<sup>1</sup> , Weiwei FANG<sup>1,2,\*</sup> 

<sup>1</sup>School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

<sup>2</sup>Hubei Engineering Research Center for Intelligent Detection and Identification of Complex Parts, Hubei, China

Received: 20.01.2024

Accepted/Published Online: 07.06.2024

Final Version: 26.07.2024

**Abstract:** Environmental sound classification (ESC) is one of the important research topics within the nonspeech audio classification field. While deep neural networks (DNNs) have achieved significant advances in ESC recently, their high computational and memory demands render them highly unsuitable for direct deployment on resource-constrained Internet of Things (IoT) devices based on microcontroller units (MCUs). To address this challenge, we propose a novel DNN compression framework specifically designed for such devices. On the one hand, we leverage pruning techniques to significantly compress the large number of model parameters in DNNs. To reduce the accuracy loss that follows pruning, we propose a knowledge distillation scheme based on feature information from multiple intermediate layers. On the other hand, we design a two-stage quantization-aware knowledge distillation scheme to mitigate the accuracy degradation of mandatory quantization required by MCU hardware. We evaluate our framework on benchmark ESC datasets (UrbanSound8K, ESC-50) using the STM32F746ZG device. The experimental results demonstrate that our framework can achieve compression rates up to 97% while maintaining competitive inference performance compared to the uncompressed baseline.

**Key words:** Environmental sound classification, deep neural networks, microcontroller units, knowledge distillation




### 1. Introduction

In everyday environments, people are surrounded by various sounds, such as air conditioners, car horns, and running water. These natural or artificial sounds can generally be categorized as environmental sounds [1]. In real life, applications related to environmental sound classification (ESC) are everywhere. For example, through data from outdoor sensor systems, sounds from different species can be identified to help with biodiversity assessment [1]. Smart home facilities can effectively collect and analyze the environmental sounds from home monitoring, providing real-time feedback on emergencies for the elderly [2]. Besides, there are also noise detection systems [3]. In nonspeech audio classification tasks, ESC is one of the hottest research topics [4]. With the rapid development of deep learning technology, deep neural networks (DNNs) [5] have greatly improved the accuracy of ESC tasks, and the application of DNNs in ESC tasks has attracted widespread attention [6, 7].

However, DNNs are computationally and memory-intensive [5], which poses a major challenge for IoT applications based on microcontroller units (MCUs) [8]. Unlike traditional computing devices, MCUs are small computers without an operating system or dynamic random-access memory (DRAM), and they have extremely limited resources in terms of computing, memory, and storage. An illustrative comparison of different computing

\*Correspondence: fangww@bjtu.edu.cn

devices is shown in Figure 1. For example, the STM32F746ZG device is an ARM Cortex-M7 MCU with a CPU clock frequency of 216 MHz, 1 MB of storage, and 320 KB of memory. The deployment of the classic DNN model ResNet-50 requires 102 MB of storage space. After INT8 quantization, it can be reduced to 26 MB, which can be applied to computing hardware such as GPUs and CPUs, but it is still impossible to deploy on MCUs. Tiny machine learning (TinyML), as a new computing paradigm [9], has emerged to run machine learning inference on ultralow-power and severe resource-constrained microcontroller units and IoT devices. To address the issue of resource limitations, aggressive compression of the currently overparameterized DNN models is necessary [8, 10].

	Cloud AI	Edge AI	TinyML
			
<b>Type</b>	NVIDIA V100	Jetson Nano	STM32F746ZG
<b>Mem</b>	16 GB	4 GB	<b>320 KB</b>
<b>Storage</b>	TB ~ PB	> 64 GB	<b>1 MB</b>
<b>Power</b>	250 W	5 W	0.3 W

**Figure 1.** Comparison of typical computing devices with different computational resources.

To address the challenge of deploying DNNs on MCUs for ESC tasks, compression based solutions have been proposed. Kumari et al. [11] proposed EdgeL<sup>3</sup>, a compact model obtained by pruning the L<sup>3</sup>-Net model to reduce redundant parameters. The theoretical compression ratio is very high, but this approach leads to sparse parameter matrices, which require dedicated hardware and software support [5]. Cerutti et al. [10] used the parameters of the VGGish model trained on the large sound dataset AudioSet, and designed a compact model structure called Micro-VGGish based on heuristic methods. Then, they proposed a knowledge distillation approach to improve the accuracy of small model. However, this work did not provide a general framework for obtaining lightweight models. Mohaimenuzzaman et al. [12] designed a DNN model ACDNet based on the characteristics of ESC data first, and then compressed it to obtain Micro-ACDNet by pruning and knowledge distillation. However, this work directly deploys the quantized model, which inevitably results in accuracy loss.

To tackle these problems, we propose a new DNN compression framework for running ESC tasks on MCUs in this paper. This framework mainly studies how to use knowledge distillation techniques to compensate for the inference accuracy loss of DNN models caused by aggressive pruning and mandatory quantization. In general, the main contributions of this work are summarized as follows:

- Considering the impact of aggressive pruning to the DNN model on inference accuracy, we propose a pruning-oriented knowledge distillation (PoKD) scheme for the pruned student network. We add auxiliary classifiers to the intermediate layers of both teacher and student networks, and combine attention mechanism to fuse output information from multiple intermediate layers of the teacher network, so as to achieve a more comprehensive knowledge transfer between teacher and student.
- Considering the impact of mandatory quantization to the DNN model on inference accuracy, we propose a two-state quantization-aware knowledge distillation (QaKD) scheme for the quantized network. This scheme takes into account the differences in scale between models, and effectively minimizes the quantization-induced accuracy loss through progressive training optimization.

- We conduct extensive experiments on the STM32F746ZG MCU using two benchmark datasets, including UrbanSound8K [13] and ESC-50 [14]. The results show that the proposed framework can attain remarkably high compression ratio and improved inference speed with modest accuracy loss, and outperform existing works at the same compression level.

The rest of this paper is organized as follows. Section 2 overviews the related work. Section 3 introduces the proposed framework, especially the two knowledge distillation schemes. Section 4 presents the experimental results as well as our analysis. Finally, Section 5 concludes the paper.

## 2. Related work

In recent years, numerous studies have been proposed to address how to compress and accelerate DNNs on resource-limited computing devices [15]. The relevant techniques include model pruning, knowledge distillation, quantization and low-rank approximation [5]. Considering the space limitation, here we only introduce typical studies from the first three techniques that are particularly relevant to our study, and interested readers can refer to review articles such as [5] for more information.

### 2.1. Model pruning

Model pruning is a technique for reducing the size of DNN model by eliminating unimportant and redundant network weights. It can be divided into two categories: unstructured pruning [15, 16] and structured pruning [17, 18]. Unstructured pruning can prune any single weight in DNNs, resulting in sparse weight matrices with high sparsity. It requires dedicated hardware and software support at runtime [17]. Structured pruning removes entire filters or channels, resulting in nonsparse compression. Thus, it overcomes the limitations of unstructured pruning, and has become a hot research topic in recent years. However, one of the key challenges in existing channel pruning solutions is how to determine the optimal layer sparsity. To address this problem, Duggal et al. [19] proposed cluster pruning (CUP), which prunes filters by first clustering them into groups using similarity metrics calculated from incoming and outgoing weight connections, then removing the redundant ones. Note that CUP only requires a single hyperparameter to determine the appropriate number of pruned filters from each layer according to their sensitivity to pruning. In this paper, we propose to use CUP as the first step of DNN compression to obtain a compact model that satisfies hardware resource constraints.

### 2.2. Knowledge distillation

Knowledge distillation was first proposed by Hinton et al. [20], using the output probabilities of a large and complex DNN (teacher network) to transfer knowledge to a small but less accurate DNN (student network). However, representing knowledge in such a highly abstract form neglects the abundant information present within intermediate layers. Recent studies [21–24] have attempted to leverage intermediate representations to capture the enriched knowledge learned by the teacher. Romero et al. [21] utilized not only the outputs but also intermediate-level hints from the teacher’s hidden layers to supervise student training. Zagoruyko et al. [22] leveraged channel attention to steer the student to mimic the spatial attention maps learned by the teacher. Wang et al. [23] proposed MHKD to match the intermediate features between the student and teacher by adding auxiliary classifiers at the intermediate layers. Yang et al. [24] performed hierarchical knowledge distillation to all auxiliary classifiers by taking advantage of self-supervised augmented knowledge. However, such manual, one-to-one layer associations between teacher and student may lead to semantic mismatch and performance

degradation [25–27]. To address this issue, ALP-KD [26] proposed to distill knowledge from all intermediate layers of the teacher rather than a specific layer to supervise every student layer. Inspired by these works, we propose the PoKD scheme to redeem the accuracy of pruned network.

### 2.3. Quantization

Quantization refers to approximating 32-bit finite-range floating point data with data types of fewer bits, so as to achieve the goals of reducing storage space and accelerating model inference [28]. Quantization methods can be divided into low-bit quantization (1 bit [29, 30], 2 bits [31], 4 bits [32, 33]) and normal quantization (8 bits [34]) according to the total number of quantization bits. Courbariaux et al. [30] proposed to train the DNN with binary weights in the forward propagation and replacing multiplication operations with simple add-subtract operations. Choi et al. [31] used 2 bits to store weight parameters by introducing a new zero state, achieving a good trade-off between model size and inference accuracy. However, such binary or ternary quantization suffers from nonnegligible accuracy loss, especially for models with a large number of parameters. In fact, existing research has shown that using 8-bit integers (INT8) to quantize weights can achieve relatively low accuracy loss [34]. Therefore, INT8 quantization has been mandatorily required for DNN deployment on resource-limited devices such as MCUs, and widely supported by deep learning frameworks such as TensorFlow Lite Micro [35] and NVIDIA TensorRT. The QaKD scheme proposed in this paper is to improve the inference accuracy of quantized model through knowledge distillation.

### 3. Proposed method

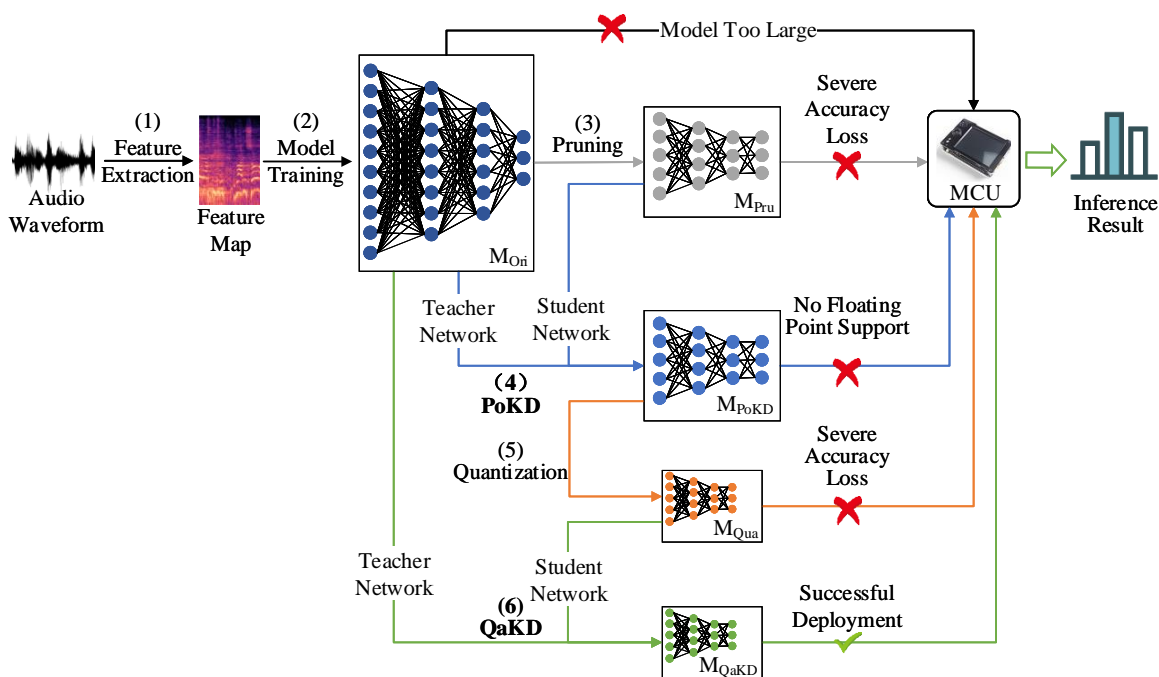


Figure 2. An overview of the proposed DNN compression framework.

### 3.1. Overview

The proposed DNN compression framework is shown in Figure 2. Specifically, the process of environmental sound classification on the MCU can be divided into six steps in general. In step 1, the input audio waveforms are transformed into log-scaled Mel spectrograms [10, 12]. In step 2, these spectrograms are used to train the original DNN model  $M_{Ori}$ . However,  $M_{Ori}$  cannot be directly deployed due to resource constraints of MCU. We decide to use pruning (step 3) and quantization (step 5) to solve this problem. In step 3, we apply the CUP algorithm proposed in [19] to prune redundant parameters from  $M_{Ori}$  to obtain a pruned model  $M_{Pru}$ . Considering the accuracy loss induced by aggressive pruning, we propose the PoKD scheme to transfer knowledge from  $M_{Ori}$  to  $M_{Pru}$  in step 4, resulting in the improved model  $M_{PoKD}$ . After quantizing  $M_{PoKD}$  (FP32) into  $M_{Qua}$  (INT8) in step 5, we propose the QaKD scheme to transfer knowledge from  $M_{Ori}$  to  $M_{Qua}$  in step 6, resulting in the final model  $M_{QaKD}$  that is deployed on the MCU. Note that PoKD is performed to provide a pretrained model  $M_{Qua}$  other than a randomly initialized one for further distillation in QaKD, which is beneficial for the final model's accuracy, convergence, and generalization [27]. In the following subsections, we will elaborate on the design of our knowledge distillation schemes, namely PoKD and QaKD, in detail.

### 3.2. PoKD

The design of our PoKD scheme is shown in Figure 3, in which  $M_{Ori}$  and  $M_{Pru}$  are the teacher network  $t$  and the student network  $s$ , respectively. Generally, we attach a total of  $N$  auxiliary classifiers into intermediate layers of  $M_{Ori}$  and  $M_{Pru}$ , respectively, at the same depths. In modern DNNs, the convolutional layers are typically stacked hierarchically in stages to extract increasingly abstract representations of the input data along the depth dimension. Therefore, we choose to attach an auxiliary classifier after each main stage. The auxiliary classifier is used to learn and distill hierarchical knowledge provided by multiscale intermediate feature maps. To this end, it is designed to contain a convolutional layer, followed by a batch normalization (BN) layer and a ReLU activation function, and finally a fully connected layer. With these added auxiliary classifiers, each student layer of  $M_{Pru}$  can learn informative knowledge contained in multiple teacher layers of  $M_{Ori}$ , rather than a fixed layer, for better supervision. Besides, the attention mechanism is integrated to make the student layer have better associate with the most semantic-related teacher layers. Note that these auxiliary classifiers are designed to help transfer knowledge and will be removed during inference.

Given a dataset  $\{\mathbf{x}, \mathbf{y}\}$  over a set of classes  $K$ ,  $\mathbf{x}$  is the input data and  $\mathbf{y}$  is the corresponding one-hot hard label vector denoting the ground-truth class. For each training sample  $\mathbf{x}$ , the neural network outputs the predictive class probability distribution as  $\mathbf{p}(\mathbf{x}, T)$ , in which the output prediction for class  $i$  is

$$p_i(\mathbf{x}, T) = \text{Softmax}(z_i, T) = \frac{\exp(z_i/T)}{\sum_{j=1}^K \exp(z_j/T)}, \quad (1)$$

where  $z_i$  and  $z_j$  are the logits of neural network, and  $T$  is the temperature to soften  $p_i(\mathbf{x}, T)$  [20]. Then, the crossentropy (CE) loss, which measures the difference between the predicted and ground-truth probabilities, is defined as

$$L_{CE}(\mathbf{p}, \mathbf{y}) = CE(\mathbf{p}(\mathbf{x}, 1), \mathbf{y}). \quad (2)$$

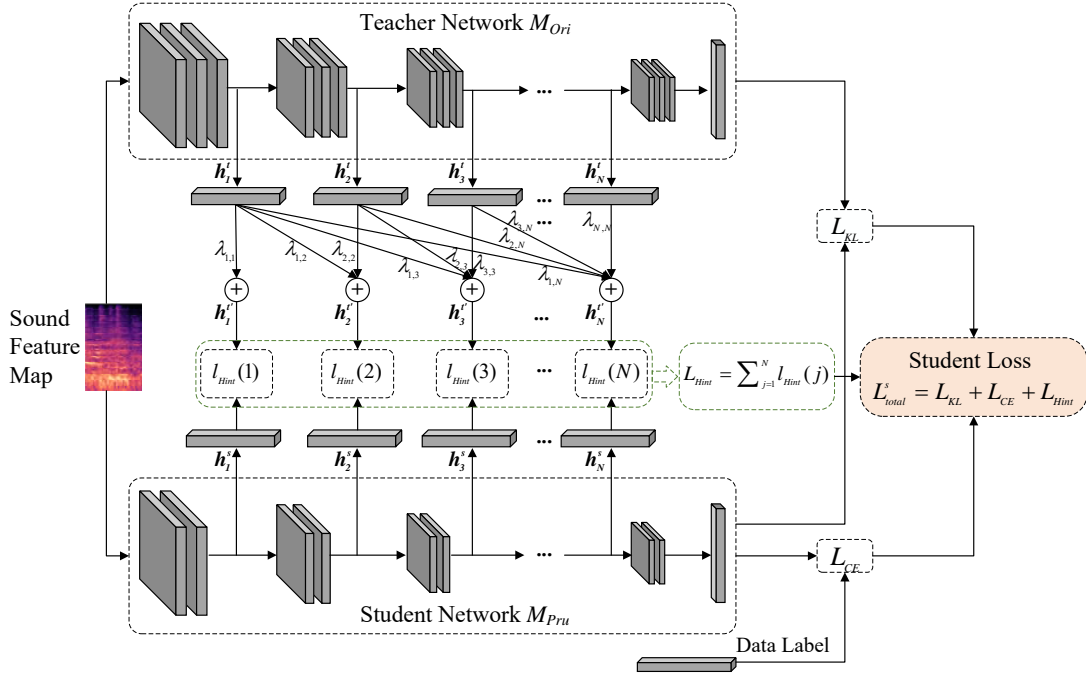


Figure 3. The design of PoKD scheme.

The Kullback–Leibler (KL) divergence loss, which measures the distance between the student and teacher probability distributions, is defined as

$$L_{KL}(\mathbf{p}, \mathbf{q}) = T^2 KL(\mathbf{p}(\mathbf{x}, T) || \mathbf{q}(\mathbf{x}, T)), \tag{3}$$

where  $T^2$  is used to retain the gradient contributions unchanged [20].

In PoKD, the overall loss for training the teacher network  $t$  includes the CE losses for the logits of both original network and all auxiliary classifiers. The first loss is to learn general classification capability by data fitting, while the second loss is to generate extra intermediate information for student supervision. Denote the logit for the  $j$ -th auxiliary classifier as  $\mathbf{h}_j^t(\mathbf{x}, \mathbf{T})$ , then the overall loss for training  $t$  is formulated as

$$L_{total}^t = L_{CE}(\mathbf{p}^t, \mathbf{y}) + \sum_{j=1}^N L_{CE}(\mathbf{h}_j^t, \mathbf{y}). \tag{4}$$

In PoKD, the overall loss for training the student network  $s$  includes not only the CE loss and the KL divergence loss well defined in vanilla knowledge distillation [20], but also a hint loss that forces each student layer to learn from multiple teacher layers [21]. Here, we introduce the attention mechanism to stimulate positive guidance and suppress negative impact induced by layer mismatch. Specifically, the  $j$ -th auxiliary classifier in  $s$  obtains hint knowledge from the first  $j$  auxiliary classifiers in  $t$ , and the attention value for the pair  $(i, j)$  is calculated as

$$\lambda_{i,j} = \frac{\exp(\mathbf{h}_i^t \cdot \mathbf{h}_j^s)}{\sum_{i=1}^j \exp(\mathbf{h}_i^t \cdot \mathbf{h}_j^s)}. \tag{5}$$

Based on this attention, the teacher  $t$  provides a fused output  $\mathbf{h}_j^{t'}$  to supervise the training of the  $j$ -th auxiliary classifier in  $s$  as

$$\mathbf{h}_j^{t'} = \sum_{i=1}^j \lambda_{i,j} \mathbf{h}_i^t. \quad (6)$$

We can obtain the hint loss for the  $j$ -th auxiliary classifier in  $s$  as

$$l_{Hint}(j) = L_{KL}(\mathbf{h}_j^s, \mathbf{h}_j^{t'}) + L_{CE}(\mathbf{h}_j^s, \mathbf{y}). \quad (7)$$

Finally, we can obtain the overall loss for training  $s$  as

$$L_{total}^s = L_{KL}(\mathbf{p}^s, \mathbf{p}^t) + L_{CE}(\mathbf{p}^s, \mathbf{y}) + L_{Hint} = L_{KL}(\mathbf{p}^s, \mathbf{p}^t) + L_{CE}(\mathbf{p}^s, \mathbf{y}) + \sum_{j=1}^N l_{Hint}(j). \quad (8)$$

### 3.3. QaKD

Due to severe resource constraints, 32-bit floating-point (FP32) parameters of DNN models often need to be quantized to 8-bit integer (INT8) parameters during deployment on MCUs [8]. Actually, INT8 quantization has become a well supported function by the deep learning framework for MCUs, such as CMSIS-NN and TensorFlow Lite Micro [8, 9]. Because  $M_{PoKD}$  retains very few parameters as compared to the original model  $M_{Ori}$ , directly quantizing it will also incur substantial accuracy loss, similar to pruning. One intuition is that knowledge distillation can be employed to compensate for the accuracy degradation of the quantized model  $M_{Qua}$  [10]. However,  $M_{Ori}$  may not be a good choice as the teacher, since it has been revealed in [25] that the size (capacity) gap between teacher and student networks is a key to distillation efficacy and student performance. Therefore, we propose a two-stage knowledge distillation scheme QaKD, which is shown in Figure 4.

Specifically, we introduce a new network  $M_{OriQ}$ , which is quantized into INT8 directly from  $M_{Ori}$ , as the teacher assistant in QaKD. Note that  $M_{OriQ}$  lies in between  $M_{Ori}$  and  $M_{Qua}$  in terms of size and capacity. We apply the proposed PoKD scheme to perform knowledge distillation between these two teacher-student pairs. Firstly,  $M_{OriQ}$  acts as the student to distill knowledge from its teacher  $M_{Ori}$ . Then,  $M_{OriQ}$  acts as the teacher to supervise its student  $M_{Qua}$ , which was directly quantized from  $M_{PoKD}$ . In fact, we only need to train  $M_{OriQ}$  and  $M_{Qua}$  once using the loss function  $L_{total}^s$  defined in PoKD, respectively. Note that when these two-stage knowledge distillation is completed, we can obtain the final model  $M_{QaKD}$ , and deploy it onto the MCU using the deep learning framework such as TensorFlow Lite Micro [35].

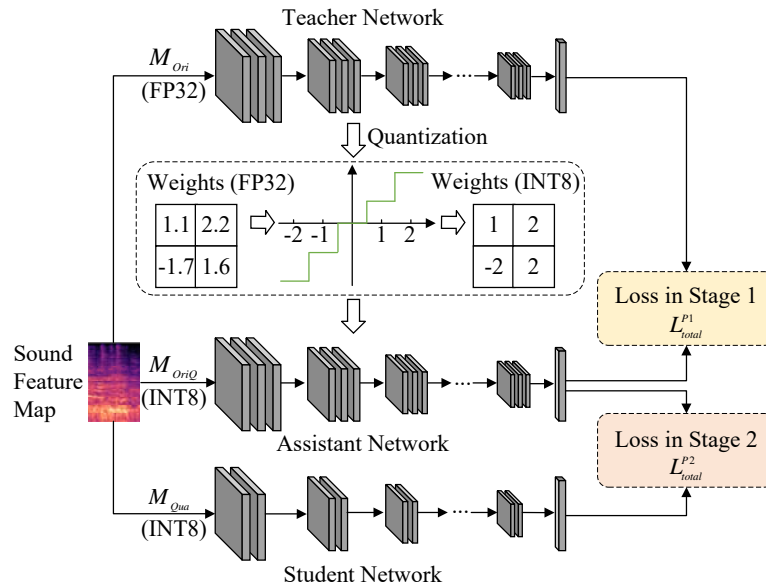
## 4. Performance evaluation

### 4.1. Experimental setup

We conduct our experiments to evaluate the proposed framework based on the following settings.

**Datasets:** The proposed framework is evaluated on two widely used ESC benchmark datasets, namely UrbanSound8K and ESC-50.





**Figure 4.** The design of QaKD scheme.

UrbanSound8K [13] is a collection of 8732 labeled sound clips of different sampling rates and up to 4 s in length, for use in research on automatic urban environmental sound classification. The dataset includes 10 classes, such as air conditioner, car horn, and dog bark. Compared to ESC-50, UrbanSound8K has a larger dataset size and the sound clips have different lengths.

ESC-50 [14] is a collection of 2000 environmental recordings, also widely used as a benchmark for environmental sound classification. The audio recordings in the dataset are 5 s long, sampled at 16 kHz and 44.1 kHz, and are evenly distributed across 50 balanced, nonoverlapping classes (40 audio samples per class). Compared to UrbanSound8K, ESC-50 covers more sound classes and all audio recordings have the same length.

**DNN Models:** Unlike image classification tasks, due to the relatively simpler features of sound data, we use shallower models from popular DNNs [4] to test the proposed framework, including VGG-11 and ResNet-18.

The VGG-11 model consists of 8 convolutional layers and 3 fully connected layers, with a batch normalization layer and a ReLU activation function after each convolutional layer. In the model training stage, auxiliary classifiers are added after the 1st, 2nd, 4th, and 6th convolutional layers of the network.

The ResNet-18 model consists of 17 convolutional layers and one fully connected layer. The first convolutional layer is used for preprocessing the input data. The remaining 16 convolutional layers are organized into four convolutional blocks, where each block consists of two residual blocks, in which each block has two convolutional layers with the same number of filters. The network has shortcut connections to skip blocks of convolutional layers, so as to avoid vanishing and exploding gradients. In the model training stage, auxiliary classifiers are added after the 1st, 2nd, and 3rd convolutional blocks of the network.

**Hardware:** The model was trained on a GPU-powered deep learning platform, using an NVIDIA GeForce RTX 2080 Ti GPU. The deep learning framework PyTorch was utilized for training DNN models. For model deployment, the target MCU device is STM32F746ZG. The STM32F746ZG is a high-performance MCU based on an ARM Cortex-M7 core running at up to 216 MHz with 1 MB of flash memory and 320 KB of SRAM. With ONNX as the intermediate representation, the trained neural network is converted to TensorFlow format,

and deployed on the MCU using TensorFlow Lite Micro. This tool is a port of TensorFlow Lite, designed to run machine learning models on DSPs, microcontrollers, and other devices with limited memory [35].

**Hyperparameter settings:** All model training was performed using the adaptive moment estimation (ADAM) optimizer. The batch size, learning rate, and weight decay factor were set to 64, 0.001, and 0.0005, respectively. The number of training iterations was set to 100, 50, 5, 50, corresponding to the phase of the original model training, the pruned model training, and the two stage training of QaKD, respectively.

**Baselines:** To further demonstrate its effectiveness and generalizability, the proposed framework is compared with two relevant solutions, namely Micro-VGGish [10] and Micro-ACDNet [12], which have both been reported to be deployed on the real MCU.

Micro-VGGish is a compressed model derived from the public feature extractor VGGish, trained on the YouTube-8M dataset. It achieves compression by reducing parameters layer-by-layer in a manual manner. To enhance its performance, particularly accuracy, a two-step knowledge distillation scheme (TKD) is proposed. This model has been evaluated on the UrbanSound8K dataset.

Micro-ACDNet is compressed from the ACDNet model, which is specially designed based on the characteristics of environmental sound. Similar to our framework, the model compression is based on structured pruning, vanilla knowledge distillation [20] and quantization. This model has been evaluated on the ESC-50 dataset.

## 4.2. Experimental results

We conducted multiple sets of experiments step-by-step to progressively validate the efficacy and efficiency of the proposed framework. We first evaluated the proposed PoKD and QaKD distillation schemes, then investigated the actual performance of the compressed models on our MCU, and finally made comparisons with Micro-VGGish and Micro-ACDNet respectively.

### 4.2.1. Results of knowledge distillation

**Pruning:** First, we conducted experiments on the CUP [19] pruning approach to observe the impact of compression ratio on model accuracy. As can be observed from Figure 5, in the ESC task, compression ratios below 85% can even slightly improve the model accuracy. Our conjecture is that removing channels of low importance has the potential to partially reduce overfitting of the model [18]. However, when the compression ratio increases to 97% or more, it can be found that the accuracy loss of both models significantly increases, ranging from 3% to 8%. Moreover, ResNet-18 is generally more robust against model pruning than VGG-11, due to the introduction of residual blocks and skip connections, as well as more layers to extract richer feature information. Based on the results in Figure 5 and the practical resource constraints of MCU, we decided to select the pruned model with a compression ratio of 97% as  $M_{Pru}$  for subsequent knowledge distillation experiments.

**PoKD:** Table 1 shows the inference accuracy of the original model  $M_{Ori}$ , the pruned model  $M_{Pru}$ , and the models after distillation using different methods with  $M_{Ori}$  as the teacher and  $M_{Pru}$  as the student. We can observe that aggressive pruning (i.e. with a compression ratio of 97%) indeed severely degrades the inference accuracy (approximately from 3% to 5.5%). Meanwhile, knowledge distillation can effectively compensate for such accuracy loss to some extent. Besides, we can observe that MHKD, ALP-KD, and PoKD have achieved much better performance than the other comparative methods. This indicates that leveraging auxiliary classifiers during distillation facilitates the generation of more discriminative feature representations. Moreover, PoKD significantly outperforms MHKD and ALP-KD, verifying that the integrated attention mechanism in PoKD is

more effective and efficient in the supervision, compared to the one-to-one layer association in MHKD [23] and the all-to-one layer association in ALP-KD [26].

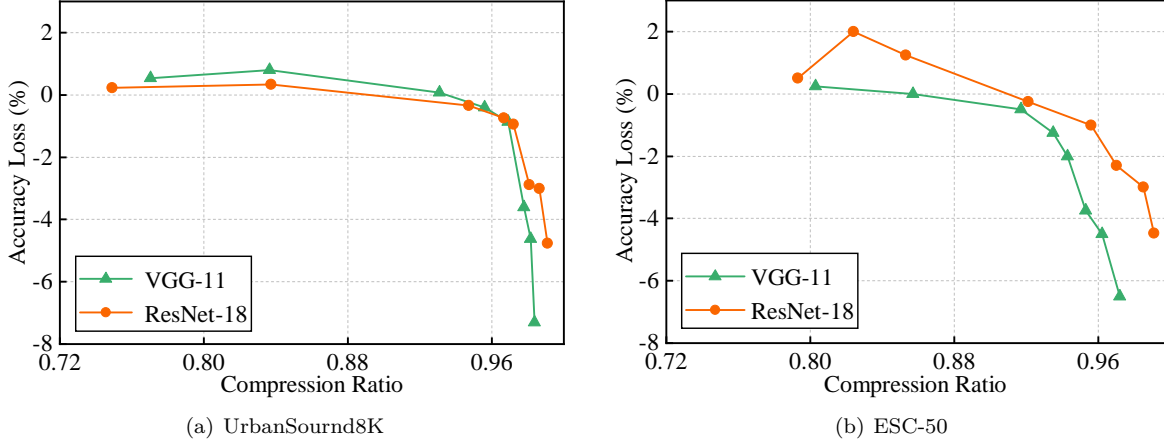


Figure 5. The impact of model pruning on inference accuracy.

Table 1. Comparison of PoKD with other knowledge distillation schemes.

Dataset	Method	Accuracy (%)		
		VGG-11	ResNet-18	
UrbanSound8K	Model variant	$M_{Ori}$	96.31	95.18
		$M_{Pru}$	92.70 (↓ 3.61)	92.17 (↓ 3.01)
		$M_{PoKD}$	96.58 (↑ 0.27)	95.12 (↓ 0.06)
	Distillation scheme	Vanilla KD [20]	93.82 (↓ 2.49)	93.90 (↓ 1.28)
		FitNet [21]	93.93 (↓ 2.38)	93.97 (↓ 1.21)
		MHKD [23]	95.18 (↓ 1.13)	94.00 (↓ 1.18)
		ALP-KD [26]	95.43 (↓ 0.88)	94.08 (↓ 1.1)
ESC-50	Model variant	$M_{Ori}$	81.75	75.25
		$M_{Pru}$	76.25 (↓ 5.5)	72.25 (↓ 3)
		$M_{PoKD}$	80.55 (↓ 1.2)	75.12 (↓ 0.13)
	Distillation scheme	Vanilla KD [20]	77.78 (↓ 3.97)	73.01 (↓ 2.24)
		FitNet [21]	77.94 (↓ 3.81)	73.85 (↓ 1.4)
		MHKD [23]	78.52 (↓ 3.23)	74.32 (↓ 0.93)
		ALP-KD [26]	78.93 (↓ 2.82)	74.57 (↓ 0.68)

**QaKD:** Table 2 shows the effectiveness of different distillation methods in improving the accuracy of the quantized model. The accuracy results of  $M_{Qua}$  indicate that INT8 quantization results in significant accuracy degradation for the small pruned model (approximately from 1.7% to 5.2%) in ESC tasks, as compared to  $M_{PoKD}$ . However, directly using  $M_{Ori}$  as a teacher to supervise the training of  $M_{Qua}$  has limited impact on accuracy improvement (approximately from 0.1% to 1.3%). The reason is that the two models have different sizes and capacities. It is obvious that using QoKD can significantly improve the final distillation gain in accuracy with the help of the teacher assistant.

**Table 2.** Comparison of QaKD with other knowledge distillation schemes.

Dataset	Method	Accuracy (%)	
		VGG-11	ResNet-18
UrbanSound8K	$M_{PoKD}$	96.58	95.12
	$M_{Qua}$	91.39 (↓ 5.19)	92.56 (↓ 2.56)
	DirectKD	92.60 (↓ 3.98)	93.00 (↓ 2.12)
	$M_{QaKD}$	94.95 (↓ 1.63)	94.18 (↓ 0.94)
ESC-50	$M_{PoKD}$	80.55	75.12
	$M_{Qua}$	78.86 (↓ 1.69)	70.10 (↓ 5.02)
	DirectKD	78.70 (↓ 1.85)	71.38 (↓ 3.74)
	$M_{QaKD}$	80.03 (↓ 0.52)	73.76 (↓ 1.36)

#### 4.2.2. Results of MCU deployment

We deployed the final lightweight model  $M_{QaKD}$  onto the STM32F746ZG MCU, using the TensorFlow Lite Micro tool. Table 3 shows the key performance results. With the framework proposed in this paper, we can achieve excellent performance with a parameter size of less than 256 KB and an accuracy loss of only 1% to 1.72% compared to the original model  $M_{Ori}$ , under a high compression ratio of 97% to 99%. Meanwhile, the average inference time of the models is approximately 200–300 ms, which reaches an acceptable level of inference latency in practical ESC applications [8, 12].

**Table 3.** Compressed model performance on the MCU.

Dataset	Model	Compression ratio (%)	Accuracy (%)	Parameter (K)	Latency (ms)
UrbanSound8K	VGG-11	97.78	94.95 (↓ 1.36)	208.60	300.52
	ResNet-18	98.63	94.18 (↓ 1.00)	186.12	256.78
ESC-50	VGG-11	97.20	80.03 (↓ 1.72)	206.12	296.64
	ResNet-18	98.91	73.76 (↓ 1.49)	150.78	206.32

#### 4.2.3. Comparisons with state-of-the-art solutions

**Comparison with Micro-VGGish:** For a fair comparison, VGGish is used as the target model for compression in this set of experiments. Due to the limitations in PyTorch’s quantization function, which does not support certain operations within the VGGish model’s architecture, we focused our performance comparison on the effects of compression and distillation.

Table 4 shows the comparison results on model pruning, in which CUP-31k and CUP-27k represent the compression of model parameters to 31 K and 27 K by the CUP pruning scheme, respectively. We can observe that both manual pruning in Micro-VGGish and CUP-based pruning lead to significant accuracy loss when compressing VGGish from 72.1 M to 30.6 K parameters. However, CUP can retain a relatively higher inference accuracy, even when we further compress VGGish to a size of 27 K parameters. These results suggest that model pruning needs to be carefully designed to avoid damaging the model’s feature extraction capability and reducing its inference accuracy. Our choice of the CUP algorithm provides a good foundation for the application of knowledge distillation for accuracy improvement. We further investigated the pruning details layer by layer, and the results are shown in Figure 6. In this figure, the number of filters is counted for the convolution layer (Conv) and the number of neurons is counted for the fully connected layer (FC). These results show that

**Table 4.** Performance comparison with Micro-VGGish on model pruning.

Method	Accuracy (%)	Parameter (K)	FLOPs (M)
VGGish	74.46	72100	3190
Micro-VGGish	63.83 (↓ 10.63)	30.60	5.87
CUP-31K	68.86 (↓ 5.6)	30.60	13.28
CUP-27K	67.89 (↓ 6.57)	27.14	10.88

manual pruning reduces the model parameters mainly by a certain proportion, while CUP prunes redundant parts according to specific needs. Moreover, as CUP preserves more filters in the convolutional layers, it requires more FLOPs than Micro-VGGish, as reported in Table 4.

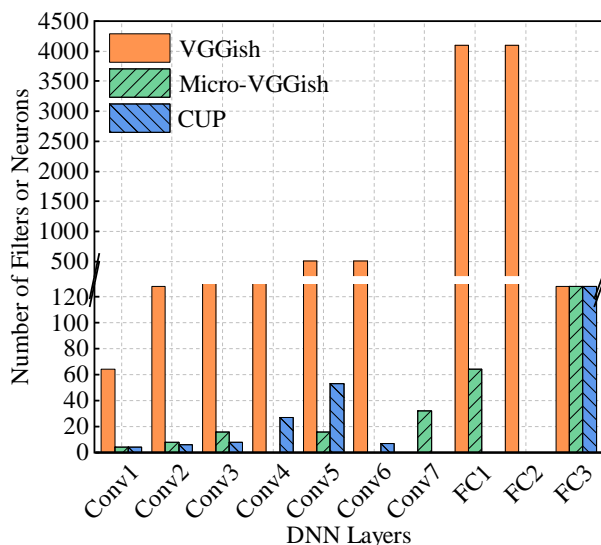
**Figure 6.** Comparison of network parameters of each layer.

Table 5 shows the accuracy results of different combinations of pruning and distillation approaches. We can make the following two observations. First, our PoKD scheme is more effective than the TKD scheme to improve the performance of Micro-VGGish, with an accuracy improvement of 1.48%. Second, with the same size of parameters, the pruned model obtained by CUP actually provides a better foundation than Micro-VGGish in learning from their common teacher VGGish with PoKD. These results fully demonstrate the effectiveness and efficiency of our selection of CUP pruning and our design of PoKD scheme.

**Table 5.** Performance comparison with Micro-VGGish on knowledge distillation.

Compression method	Accuracy (%)	Parameter (K)
VGGish	74.46	72100
Micro-VGGish + TKD	71.66	30.60
Micro-VGGish + PoKD	73.14	30.60
CUP-31K + PoKD	73.87	30.60

**Comparison with Micro-ACDNet:** For a fair comparison, ACDNet is used as the target model for compression in this set of experiments. We compared the performance of Micro-ACD and our proposed framework on MCUs, as shown in Table 6. It is obvious that our framework fully outperforms Micro-ACD in terms of accuracy, parameter size, computational cost, and inference latency. The reasons are 2-fold. On the one hand, in the design we chose CUP, a more superior pruning method, to remove redundant parameters as much as possible and avoid unnecessary computational cost. On the other hand, we adopted a two-stage QaKD knowledge distillation to minimize the impact of quantization on accuracy, while Micro-ACDNet is directly deployed on the MCU after quantization, which greatly affects its accuracy.

**Table 6.** Performance comparison with Micro-ACDNet.

Compression method	Accuracy (%)	Parameters (K)	FLOPs (M)	Latency (ms)
Micro-ACDNet	81.50	130	14.8	189.99
Our framework	83.46	120	12.6	178.04

## 5. Conclusion

In this paper, we propose a new compression framework to address the difficulty of deploying DNN models for ESC tasks on MCUs. On the one hand, we propose a pruning-oriented knowledge distillation (PoKD) scheme to address the accuracy loss caused by aggressive pruning of redundant network parameters. On the other hand, we propose a quantization-aware knowledge distillation (QaKD) scheme to address the accuracy loss caused by mandatory quantization for model deployment on MCUs. Through extensive experiments on classical DNN models and benchmark datasets, our framework has achieved a compression ratio of over 97%, with an accuracy loss of less than 2%. As compared with existing solutions, the inference accuracy is improved by 1% to 2% at the same compression level.

For future work, we will continue to investigate additional datasets, MCUs, and models related to ESC [36]. Additionally, we aim to explore other potential model compression techniques for deploying DNNs on MCUs. Our goal is also to implement the entire chain on an MCU-based IoT device, which includes sound signal acquisition, ESC task processing, and real-time result display [4].

## Acknowledgment/disclaimer/conflict of interest

This work has received funding from the National Science Foundation of China (NSFC) under grant 62172031, the Beijing Municipal Natural Science Foundation under grant L191019, and the Open Projects funded by Hubei Engineering Research Center for Intelligent Detection and Identification of Complex Parts, under grant IDICP-KF-2024-15.

The authors declare that they have no conflict of interest.

## Informed consent

The experiments were not conducted with humans.

## References

- [1] Nanni L, Maguolo G, Brahmam S, Paci M. An ensemble of convolutional neural networks for audio classification. *Applied Sciences* 2021; 11 (13): 5796. <https://doi.org/10.3390/app11135796>
- [2] Demir F, Turkoglu M, Aslan M, Sengur A. A new pyramidal concatenated CNN approach for environmental sound classification. *Applied Acoustics* 2020; 170: 107520. <https://doi.org/10.1016/j.apacoust.2020.107520>
- [3] Davis N, Suresh K. Environmental sound classification using deep convolutional neural networks and data augmentation. In: 2018 IEEE Recent Advances in Intelligent Computational Systems (RAICS); Thiruvananthapuram, India; 2018. pp. 41-45. <https://doi.org/10.1109/raics.2018.8635051>
- [4] Nordby JO. Environmental sound classification on microcontrollers using convolutional neural networks. Master, Norwegian University of Life Sciences, Ås, Norway, 2019.
- [5] Menghani G. Efficient deep learning: a survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys* 2023; 55 (12): 1-37. <https://doi.org/10.1145/3578938>
- [6] Sharma J, Granmo OC, Goodwin M. Environment sound classification using multiple feature channels and attention based deep convolutional neural network. In: Conference of the International Speech Communication Association; Shanghai, China; 2020. pp. 1186-1190. <https://doi.org/10.21437/interspeech.2020-1303>
- [7] Abdoli S, Cardinal P, Koerich AL. End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications* 2019; 136: 252-263. <https://doi.org/10.1016/j.eswa.2019.06.040>
- [8] Lin J, Chen WM, Lin Y, Cohn J, Gan C et al. Mccnet: tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems* 2020; 33: 11711-11722.
- [9] Doyu H, Morabito R, Höller J. Bringing machine learning to the deepest IoT edge with TinyML as-a-service. *IEEE IoT Newsletter* 2020; 11: 1-3.
- [10] Cerutti G, Prasad R, Brutti A, Farella E. Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms. *IEEE Journal of Selected Topics in Signal Processing* 2020; 14 (4): 654-664. <https://doi.org/10.1109/jstsp.2020.2969775>
- [11] Kumari S, Roy D, Cartwright M, Bello JP, Arora A. EdgeL<sup>3</sup>: compressing L<sup>3</sup>-Net for mote-scale urban noise monitoring. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW); Rio de Janeiro, Brazil; 2019. pp. 877-884. <https://doi.org/10.1109/ipdpsw.2019.00145>
- [12] Mohaimenuzzaman M, Bergmeir C, West I, Meyer B. Environmental sound classification on the edge: a pipeline for deep acoustic networks on extremely resource-constrained devices. *Pattern Recognition* 2023; 133: 109025. <https://doi.org/10.1016/j.patcog.2022.109025>
- [13] Salamon J, Jacoby C, Bello JP. A dataset and taxonomy for urban sound research. In: 22nd ACM international conference on Multimedia; Orlando, FL, USA; 2014. pp. 1041-1044. <https://doi.org/10.1145/2647868.2655045>
- [14] Piczak KJ. ESC: Dataset for environmental sound classification. In: 23rd ACM International Conference on Multimedia; Brisbane, Australia; 2015. pp. 1015-1018. <https://doi.org/10.1145/2733373.2806390>
- [15] Cheng J, Wang P, Li G, Hu Q, Lu H. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering* 2018; 19 (1): 64-77. <https://doi.org/10.1631/fitee.1700789>
- [16] Louizos C, Welling M, Kingma DP. Learning sparse neural networks through  $L_0$  regularization. arXiv preprint 2017; arXiv:1712.01312. <https://doi.org/10.48550/arXiv.1712.01312>
- [17] Liu N, Ma X, Xu Z, Wang Y, Tang J et al. AutoCompress: an automatic DNN structured pruning framework for ultra-high compression rates. In: 2020 AAAI Conference on Artificial Intelligence; New York, USA; 2020. pp. 4876-4883. <https://doi.org/10.1609/aaai.v34i04.5924>
- [18] Liu M, Fang W, Ma X, Xu W, Xiong N et al. Channel pruning guided by spatial and channel attention for DNNs in intelligent edge computing. *Applied Soft Computing* 2021; 110: 107636. <https://doi.org/10.1016/j.asoc.2021.107636>

- [19] Duggal R, Xiao C, Vuduc R, Chau DH, Sun J. Cup: cluster pruning for compressing deep neural networks. In: 2021 IEEE International Conference on Big Data (Big Data); Orlando, FL, USA; 2021; 5102-5106. <https://doi.org/10.1109/bigdata52589.2021.9671980>
- [20] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint 2015; arXiv:1503.02531. <https://doi.org/10.48550/arXiv.1503.02531>
- [21] Romero A, Ballas N, Kahou SE, Chassang A, Gatta C et al. Fitnets: hints for thin deep nets. arXiv preprint 2014; arXiv:1412.6550. <https://doi.org/10.48550/arXiv.1412.6550>
- [22] Zagoruyko S, Komodakis N. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. arXiv preprint 2016; arXiv:1612.03928. <https://doi.org/10.48550/arXiv.1612.03928>
- [23] Wang H, Lohit S, Jones M, Fu Y. Multi-head knowledge distillation for model compression. arXiv preprint 2020; arXiv:2012.02911. <https://doi.org/10.48550/arXiv.2012.02911>
- [24] Yang C, An Z, Cai L, Xu Y. Hierarchical self-supervised augmented knowledge distillation. In: 30th International Joint Conference on Artificial Intelligence (IJCAI); Montreal, Canada; 2021. pp. 1217-1223. <https://doi.org/10.24963/ijcai.2021/168>
- [25] Mirzadeh SI, Farajtabar M, Li A, Levine N, Matsukawa A et al. Improved knowledge distillation via teacher assistant. In: 2020 AAAI Conference on Artificial Intelligence; New York, USA; 2021. pp. 5191-5198. <https://doi.org/10.1609/aaai.v34i04.5963>
- [26] Passban P, Wu Y, Rezagholizadeh M, Liu Q. ALP-KD: attention-based layer projection for knowledge distillation. In: 2021 AAAI Conference on Artificial Intelligence; Palo Alto, CA, USA; 2021. pp. 13657-13665. <https://doi.org/10.1609/aaai.v35i15.17610>
- [27] Turc I, Chang MW, Lee K, Toutanova K. Well-read students learn better: on the importance of pre-training compact models. arXiv preprint 2019; arXiv:1908.08962. <https://doi.org/10.48550/arXiv.1908.08962>
- [28] Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW et al. A survey of quantization methods for efficient neural network inference. arXiv preprint 2021; arXiv:2103.13630. <https://doi.org/10.48550/arXiv.2103.13630>
- [29] Rastegari M, Ordonez V, Redmon J, Farhadi A. Xnor-net: imagenet classification using binary convolutional neural networks. In: Leibe B, Matas J, Sebe N, Welling M (editors). Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Cham, Switzerland: Springer, 2016, pp. 525-542. [https://doi.org/10.1007/978-3-319-46493-0\\_32](https://doi.org/10.1007/978-3-319-46493-0_32)
- [30] Courbariaux M, Bengio Y, David J P. BinaryConnect: Training deep neural networks with binary weights during propagations. *Advances in Neural Information Processing Systems* 2015; 28: 1-9.
- [31] Choi J, Venkataramani S, Srinivasan VV, Gopalakrishnan K, Wang Z et al. Accurate and efficient 2-bit quantized neural networks. In: 2019 Conference on Machine Learning and Systems; Stanford, CA, USA; 2019; 348-359.
- [32] Choukroun Y, Kravchik E, Yang F, Kisilev P. Low-bit quantization of neural networks for efficient inference. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW); Seoul, Korea (South); 2019. pp. 3009-3018. <https://doi.org/10.1109/iccvw.2019.00363>
- [33] Banner R, Nahshan Y, Soudry D. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems* 2019; 32: 1-9.
- [34] Liang T, Glossner J, Wang L, Shi S, Zhang X. Pruning and quantization for deep neural network acceleration: a survey. *Neurocomputing* 2021; 461: 370-403. <https://doi.org/10.1016/j.neucom.2021.07.045>
- [35] David R, Duke J, Jain A, Reddi Janapa V, Jeffries V et al. TensorFlow lite micro: embedded machine learning for TinyML systems. In: 2021 Conference on Machine Learning and Systems; Virtual Site; 2021. pp. 800-811.
- [36] Meedeniya D, Ariyaratne I, Bandara M, Jayasundara R, Perera C. A survey on deep learning based forest environment sound classification at the edge. *ACM Computing Surveys* 2023; 56 (3): 1-36. <https://doi.org/10.1145/3618104>