sensors

MDPI

Article

A Method of Deep Learning Model Optimization for Image Classification on Edge Device

Hyungkeuk Lee [1], NamKyung Lee [1] and Sungjin Lee [2,*]

[1] Media Intelligence Research Section, Electronics and Telecommunications Research Institute, 218, Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea
[2] Electronic Engineering, Dong Seoul University, 76 Bokjeong-ro, Sujeong-gu, Seongnam-si 13117, Korea
* Correspondence: sungjinlee@du.ac.kr

Abstract: Due to the recent increasing utilization of deep learning models on edge devices, the industry demand for Deep Learning Model Optimization (DLMO) is also increasing. This paper derives a usage strategy of DLMO based on the performance evaluation through light convolution, quantization, pruning techniques and knowledge distillation, known to be excellent in reducing memory size and operation delay with a minimal accuracy drop. Through experiments regarding image classification, we derive possible and optimal strategies to apply deep learning into Internet of Things (IoT) or tiny embedded devices. In particular, strategies for DLMO technology most suitable for each on-device Artificial Intelligence (AI) service are proposed in terms of performance factors. In this paper, we suggest a possible solution of the most rational algorithm under very limited resource environments by utilizing mature deep learning methodologies.

Keywords: image classification; lightweight network; network compression; convolutional neural network; quantization; pruning; knowledge distillation

Citation: Lee, H.; Lee, N.; Lee, S. A Method of Deep Learning Model Optimization for Image Classification on Edge Device. *Sensors* **2022**, *22*, 7344. https://doi.org/10.3390/s22197344

Academic Editors: Cosimo Distante and Anastasios Doulamis

Received: 25 July 2022
Accepted: 21 September 2022
Published: 27 September 2022

1. Introduction

As athe demand for applying deep learning models on mobile and Internet-of-Things (IoT) devices increases, industrial needs for a Deep Learning Model Optimization (DLMO) and Neural Network Compression (NCC) suitable for on-device Artificial Intelligence (AI) are also increasing. In particular, an AI service on the edge devices, referred to as edge computing or Artificial Internet of Things (AIoT) [1], is being applied in various fields such as a smart cities, smart factories, smart agriculture, smart mobility, etc. Since the native neural networks are difficult to deploy on tiny devices and embedded systems with limited resources, researchers in this field have studied model optimization and network compression [2]. Many studies have tried to apply deep learning into several applications such as the detection of diabetic retinopathy [3], the management of security in Internet of Medical Things (IoMT) environments [4], and optimization techniques for IoT data [5,6]. Due to model optimization and network compression, the memory size and a computational delay of deep learning models are reduced compared to the native neural networks, while the performance of models is maintained as well [7].

In this paper, we address usage strategies of DLMO based on performance evaluation through several combinations of Lightweight Convolution, Quantization and Pruning techniques. First, in order to examine each performance among combinations, we are using VGGNet [8] and ResNet [9] as baseline networks. Then, for a comparison, MobileNet v1, v2 and v3 [10–12] are used as lightweight networks. In various IoT use cases, we evaluate the performances of several quantization techniques, which comprise Quantization Aware Training (QAT) and subtypes of Post Training Quantization (PTQ), i.e., Baseline Quantization (BLQ), Full Integer Quantization (FIQ) and Float 16 Quantization (F16) [13,14]. Lastly, for the pruning technique, the performance improvement will be analyzed by applying the training method to the basic Convolution Neural Network (CNN) and lightweight CNN

technologies [14,15]. As datasets for performance analysis, we will use the Canadian Institute For Advanced Research 10 (CIFAR10) and CIFAR100. Rather than present a new high-level algorithm, we try to guide a possible combination of the most rational algorithm under very limited resource environments by utilizing the dominant technologies with high maturity.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 illustrates the proposed methodology with respect to lightweight network techniques, while Section 4 shows the simulation results. Section 5 provides the conclusions of the work.

2. Related Work

The performance improvement in the deep learning-based image classification models has come from the excellence of the CNN's feature extraction [16]. Thus, how to design the network layer as an extractor's role has become key to improving the overall performance and computational efficiency. In the early days of deep learning, the convolutional extractor network was mainly focused on performance improvement, so the number of layers gradually increased and the structure was designed to be more complex. Nevertheless, the authors in [17,18] pointed out that a bottleneck in CNN's performance is due to the imbalanced memory distribution in CNN designs, i.e., the first several blocks have an order-of-magnitude larger memory usage than the rest of the network. However, even though VGGNet [8], with a simple 3×3 convolutional block-based structure, has dramatically reduced computational complexity, it showed a comparable performance to the Inception model [19] with a complex structure. Then, an interest in reducing computational efficiency of convolution extractor networks has begun to rise. Afterwards, ResNet, with a skip-connection structure [9], also contributed to both a reduction in computational complexity and an improvement in accuracy. In addition, enhanced versions of ResNet appeared, such as Wide Residual Network (WRN) [20] and ResNeXt [21]. In this trend, as subtypes of MobileNets (MobileNet v1 using Depthwise Separable Convolution [10], MobileNet v2 [11] using Bottleneck Residual Block [9] and Squeeze and Excitation Block [22], MobileNet v3 [12] using Network Architecture Search (NAS) [23]) appear, convolution lightweight technology has made substantial progress.

The authors in [24] systematically studied model scaling and identified that carefully balancing network depth, width and resolution can lead to better performance. They proposed a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. The authors in [25] improved the performance by combining ResNet scaling strategies. The strategy depends on the training regime and offers two new scaling strategies: (1) scale model depth in regimes where overfitting can occur (width scaling is preferable otherwise); (2) increase image resolution more slowly. The Google brain team has thoroughly researched a method to reduce significant amounts of computational resources, memory and power to train and run on mobile and IoT devices. Google provided one of the core Machine Learning (ML) kits, 'Learn2Compress [26]', to make machine learning accessible for all mobile developers. It is an automatic model compression service and enables custom on-device deep learning models in TensorFlow Lite that run efficiently on mobile devices, without developers having to worry about optimizing for memory and speed. TensorFlow Lite Micro (TFLM) is, in particular, an open-source ML inference framework for the tiny embedded systems [27]. Other AI industries have also provided network compression and light weight ML models such as the Pytorch module, NeuralMagic, Nvidia's TensorRT, OpenVINO, etc. The authors in [28] demonstrated the efficient neural network kernel to maximize the performance and minimize the memory consumption on Arm Cortex-M processor. On microcontrollers, which are small computing resources on a single VLSI integrated circuit (IC) chip for IoT or embedded devices, the authors in [17,18] suggested a joint framework of the efficient neural architecture and the lightweight inference engine for image classification. There have been also vigorous studies on reducing the amount of memory consumption or managing

memory resources efficiently in [29–31]. We have summarized related works mentioned above in Table 1.

Table 1. Summary of the related work.

Reference #	Proposed
Lin et al., 2020 [17]	A framework that jointly designs the efficient neural architecture and the lightweight inference engine, enabling ImageNet-scale inference on microcontrollers (MCUNet v1).
Lin et al., 2021 [18]	A generic patch-by-patch inference scheduling, which operates only on a small spatial region of the feature map and significantly cuts down the peak memory (MCUNet v2).
Tan et al., 2019 [24]	A new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple effective compound coefficient (EfficientNet).
Bello et al., 2021 [25]	Training and scaling strategies: (1) scale model depth; (2) increase image resolution depending on the training regime.
David et al., 2021 [27]	A model-architecture framework that enables hardware vendors to provide platform-specific optimizations and is open to a wide machine-learning ecosystem (TensorFlow Lite Micro).
Lai et al., 2018 [28]	An efficient kernels developed to maximize the performance and minimize the memory footprint of neural network applications on Arm Cortex-M processors targeted for intelligent IoT edge devices(CMSIS-NN)
Gural et al., 2019 [29]	Memory-optimal direct convolutions as a way to push classification accuracy as high as possible given strict hardware memory constraints at the expense of extra compute.
Sakr et al., 2021 [30]	An in-place computation strategy to reduce memory requirements of neural network inference.
Müksch et al., 2020 [31]	A comparison among several CNN variations (as like ProtoNN, Bonsai and FastGRNN) to apply 3-channel image classification using CIFAR10.

3. System Model

3.1. Quantization Technique

In [2], a model quantization was a widely used technique to compress and accelerate the inference stage of deep learning. Recent hardware accelerators for deep learning have begun to support mixed precision (1–8 bits) to further improve computation efficiency, which raises a great challenge to find the optimal bitwidth for each layer: it requires domain experts to explore the vast design space, trading off between accuracy, latency, energy, and model size, which is both time-consuming and sub-optimal.

A quantization technique compresses the network weights by reducing the number of bits, and then the network weight becomes smaller from 32 bits. Therefore, the quantization method limits the dynamic range and the expression accuracy of bits but also has the advantage of reducing the overall network weight size as much as the number of quantized bits.

To analyze the quantization transformation mathematically, let us define the former FP32-bit tensor as x_f and the quantized INT8 tensor as x_q. The basic transformation method of quantization then becomes:

$$x_q = Round\Big(s \cdot Clip\Big(x_f, -r, r\Big)\Big), \quad \text{where } s = \frac{127}{r}. \tag{1}$$

As shown in Figure 1, the distribution of the dynamic range of FP32 is in $[-3.4 \times 10^{-38}, 3.4 \times 10^{38}]$ based on IEEE 754 standard [32] and the dynamic range of INT8 can express 255 equally spaced numbers. In other words, in order to map numbers from FP32 to INT8, the $Clip$ function is used to discard some outlier numbers outside $-r$ and r in the dynamic range of FP32. In addition, s is used for spacing adjustment. Therefore, the conversion from FP32 to INT8 results in some latency due to the operations of Equation (1), such as $Clip$, $Round$ and $Scale$.

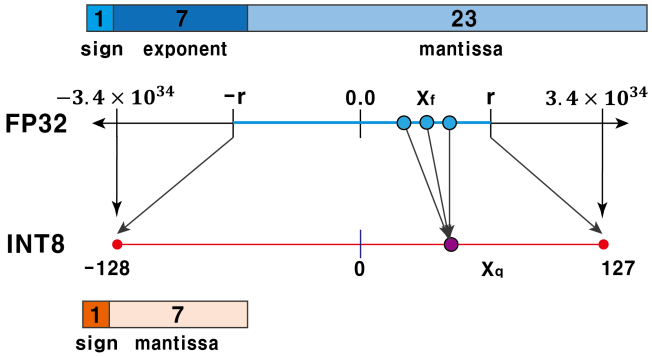

Figure 1. Quantization from FP32 to INT8.

On the other hand, the dynamic range of FP16 becomes $[-65504, 65504]$ with the half size of FP32 based on IEEE 754 standard [32]. As shown in Figure 2, the two floating point representations FP32 and FP16 shows the similarity in the format, so that their conversion needs just bitwise operation with little latency, i.e., the exponent and mantissa of FP16 can be obtained by removing 2 LSBs (Least Significant Bits) of the exponent in FP32 and 13 LSBs of mantissa in FP32.

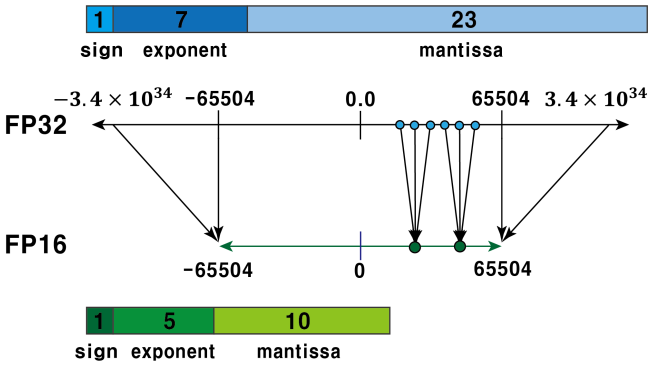

Figure 2. Quantization from FP32 to FP16.

The quantization technique can be divided into two types depending on whether the bit conversion is performed before or after training the weight values:

- *Post-Training Quantization(PTQ)*: A method of training weights with FP32 and then quantizing the results into smaller datatypes;
- *Quantization Aware Training (QAT)*: A method of training weights for maximizing their accuracy with the quantized datatype.

PTQ techniques can be also categorized into the following three methods according to how the weights are quantized: Baseline Quantization (BLQ), Full Integer Quantization (FIQ) and FP16 Quantization.

The BLQ is a method of quantizing FP32 weight values into the INT8 type. In particular, the inference is performed by reversing the quantized INT8 value to a FP32 decimal value with the reduced precision. Then, as shown in Figure 3, the inference is operated as:

$$\begin{aligned} &Input(\text{FP32}) \times Weight(\text{FP32 with reduced precision}) \\ &+ Bias(\text{FP32}) = Output(\text{FP32}). \end{aligned} \tag{2}$$

The FIQ is a method that quantizes all mathematical values of the network model using a sample data set, that is, it determines the quantization parameters such as the minimum and maximum values of the weight and activation function values and the bias values. Then, the inference is performed by quantizing input FP32 values to INT8 values with the predetermined quantization parameters. Here, to determine the quantization

parameters, a small data set of around 100–500 samples is needed for the additional training. Figure 4 represents the procedure.

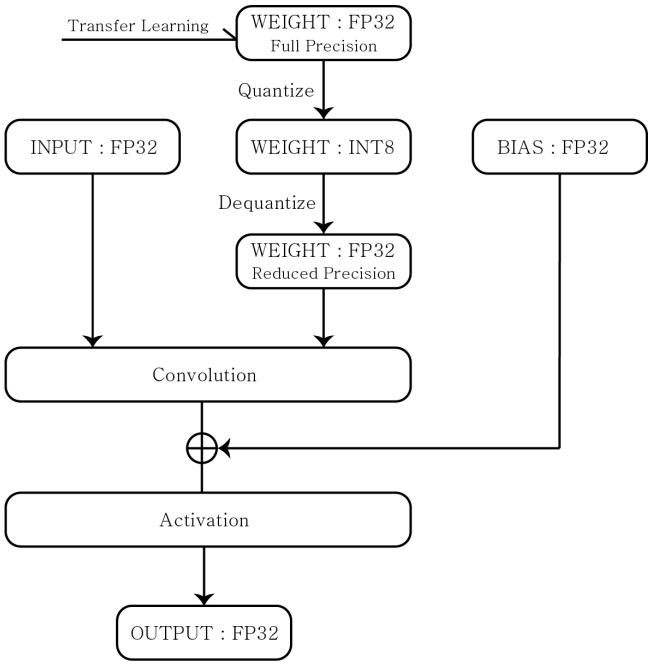

Figure 3. Baseline Quantization.

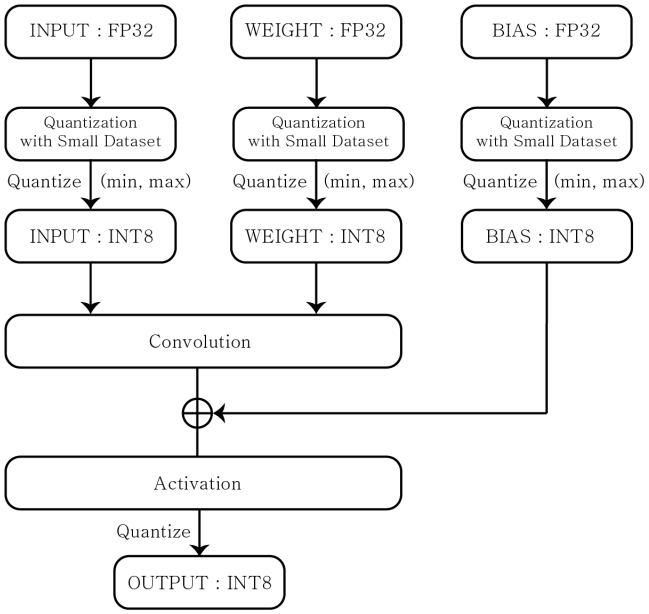

Figure 4. Full Integer Quantization.

The FIQ also includes an implementation of the float fallback method in case there is no implementation of conversion to integer values for each decimal value due to hardware limitations. The F16 is a method of expressing the FP32 weight values into to the nearest FP16 weight values with lower precision. It is possible to reduce the model size in half with minimal accuracy loss. The overall procedure is shown in Figure 5.

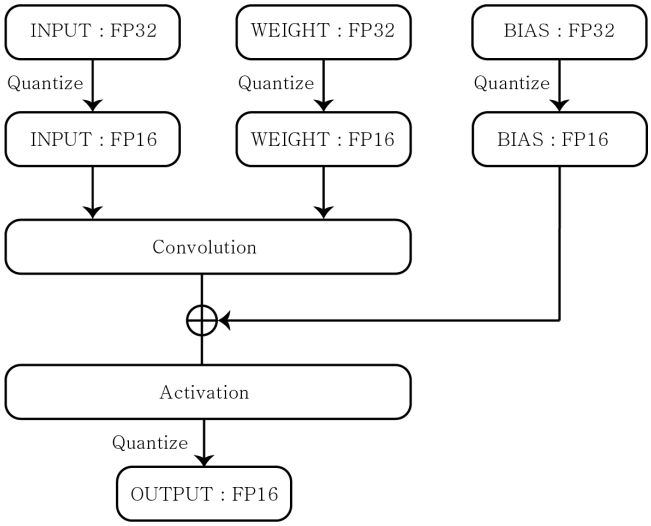

Figure 5. Float 16 Quantization.

3.2. Pruning

The pruning technique is a method to leave weights over the threshold value and set the rest of weights to zero within a deep learning operation. In general, a rule for choosing pruning weights is to sort some weights by their absolute values and to adjust the rest of weights to zero the smallest weights until some desired sparsity level is satisfied [15]. In this paper, weights set to zero in the aforementioned CNN models are gradually increased for 60–70 iterations to achieve maximal accuracy.

3.3. Knowledge Distillation

Knowledge Distillation (KD) is a deep learning method to transfer the knowledge from the cumbersome model with the large deep neural network to a small model with more suitable structure for deployment [33]. The cumbersome model and the small model are also known as teacher model and student model, respectively. A simple way to transfer the generalization ability of the teacher model to the student model is to use the class probabilities produced by the teacher model as "soft targets" for training the student model.

In KD, the soft target, i.e., each class probability q_i from each logit z_i in a "softmax" equation, can be more distributed through temperature T as:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{3}$$

Since the vanilla KD method was published in [33], various KD schemes have been developed. As shown in Tables 2–4, the KD methods can be generally classified in terms of Knowledge Type (KT), Distillation Type (DT) and Teacher–Student Architecture (TSA) [34]. First of all, the KT for KD can be classified into response, feature and relation according to the KT, as shown in Table 2. The aim of the response-based KT is to calculate the distillation loss from the logit outputs of the teacher and student models [35–38]. The feature-based knowledge type aims to calculate the distillation loss from the intermediate representations of the teacher and student models [39–44]. The relation-based knowledge type aims to utilize the relation from the feature maps [45–48]. Although it has a similarity with the previous feature-based KT in the perspective of using the intermediate feature map, it is distinguished from using the manipulated function of the feature maps such as the Gram matrix [45].

Table 2. KD Classification according to Knowledge Type.

Category	Meaning
Response	logit outputs of TSA
Feature	intermediate representations of TSA
Relation	relation between the feature maps

On the other hand, the KD can be classified into online, offline and self-distillation according to the DT, as shown in Table 3. The offline-based DT aims to transfer the knowledge from a pre-trained teacher model into a student model [33,42–44,49–51]. The online-based DT aims to update the teacher and student models simultaneously [41,47,52–55]. In the special case of online distillation, the self-distillation-based distillation type aims to utilize the same network model for the teacher and student models [56–61].

Table 3. KD Classification according to Distillation Type.

Category	Meaning
Offline	KD from a pre-trained teacher model
Online	Update the TSM simultaneously
Self-Distillation	Online method with same TSA

Last, as shown in Table 4, the KD can be classified into the same architecture as the teacher [62–64]; reduced architecture from teacher and light-weight architecture using light-weight convolution modules [51,65,66]; and Quantization and Pruning according to the TSA [10,67–72]. In particular, the light-weight architecture-based student model is based on the various light-weight convolution structure mentioned in Section 2. Likewise, the quantization and pruning based student model is based on the structure mentioned in Sections 3.1 and 3.2.

Table 4. KD Classification according to Teacher–Student Architecture.

Category	Meaning
Same as Teacher	Same architecture with Teacher
Reduced Teacher	Reduced architecture from Teacher
Light Network	Design with Light-Weight Conv, Quantization and Pruning

This paper evaluates the performance of the Vanilla KD with a Light-Network-based student architecture in terms of Response, Offline and Light Network types (MobileNetv1, v2 and v3).

4. Simulation Results

4.1. Performance of Quantization and Pruning

In order to examine the performance of the quantization and pruning technology groups, let us define the evaluation setups. First, the basic technology without quantization and pruning is denoted as NQ (No Quantization). Then, as mentioned in Section 3.1, the other quantization techniques for the evaluation are denoted by BLQ, FIQ, F16 Quantization and QAT.

In addition, let us define the pruning technique as PRN (Pruning). Then, a technique that applies both quantization BLQ and pruning is PRQ (Pruning Quantization). Finally, VGGNet and ResNet50, as the baseline networks, are tested on the CIFAR10 and CIFAR100 datasets. The recognition delay or latency (Lat) is the value computed for 10,000 images of the validation set in CIFAR10 and CIFAR100. All of these experiments have gone through the TensorFlow-Lite conversion and code optimization procedures [14].

As shown in the results of Tables 5–7, when the quantizations are applied, all quantization setups have similar or better accuracy performance (as shown in "Acc") than NQ, whereas the model size (Size) is decreased by 25%∼50%. Among them, it is observed that BLQ, FIQ and QAT have similar performance, i.e., accuracy around 80% and size of 25%. However, F16 has a size of 50% with a similar accuracy around 80%. In terms of "Lat", F16 shows the best performance. On the other hand, BLQ, FIQ and QAT have rather long latencies. This proves that the conversion from the FP32 inputs to INT8 values takes a significant amount of time.

Table 5. VGGNet.

DLMO	**CIFAR10**			**CIFAR100**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	71.9	56	4	43.5	425	18
BLQ	71.9	19	20	43.7	112	35
FIQ	72.5	18	44	43.5	110	53
F16	72.1	30	6	43.8	214	19
QAT	72.5	18	42	42.5	106	45
PRN	68.7	16	2	39.7	123	2
PRQ	68.6	7	19	39.7	37	19

Table 6. ResNet50.

DLMO	**CIFAR10**			**CIFAR100**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	80.7	94,052	177	36.5	94,790	129
BLQ	80.3	24,161	2348	36.5	24,346	2642
FIQ	80.4	24,269	2078	37.2	24,454	1997
F16	80.6	47,072	152	39.4	47,441	162
QAT	80.5	24,281	2069	36.9	24,431	2001
PRN	81.2	27,503	155	43.2	27,857	148
PRQ	81.2	8023	2399	43.1	8059	2379

Table 7. ResNet101.

DLMO	**CIFAR10**			**CIFAR100**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	79.6	169961	336	41.3	170,698	365
BLQ	79.5	43776	4576	41.5	43,960	4646
FIQ	80.8	43990	4555	37.4	44,174	4726
F16	80.2	85072	329	41.4	85,441	352
QAT	80.1	43799	4529	37.0	44,111	4731
PRN	79.8	49704	347	38.7	49,697	337
PRQ	79.8	14812	4603	38.6	14,670	4587

For the performance of "PRN", it is observed that although its accuracy and size are similar with those of the quantization techniques, it has significant advantages in "Lat". The reason for this is that pruning has no conversion procedure from the FP32 input to other data units such as FP16 or INT8. Moreover, the density of effective weights, i.e., non-zero values, is sparse compared to that of the quantization, so the computational complexity can be reduced. Those two reasons are the main factors in the model compression performance of PRN.

In addition, if the quantization is added to this pruning technique, i.e., PRQ, the model compression performance can be extremely enhanced, but latency increases similar with the other quantization schemes due to the conversion of data units.

In particular, it is remarkable that pruning shows better accuracy in ResNet compared to VGGNet. From this fact, it can be seen that the higher the network depth and number of channels, the greater the pruning gain.

Based on the above experiments and observations, the following remarks can be derived:

Remark 1. *If the quantization is applied, the model can be compressed while maintaining the similar accuracy.*

Remark 2. *The data conversion procedure of the quantization can cause recognition delays.*

Remark 3. *If pruning is applied, both the model size and the recognition delay can be reduced while maintaining a similar accuracy.*

Remark 4. *The pruning shows a better performance improvement when applied to neural networks with a large capacity.*

4.2. Performance Evaluation of Light-Weight CNNs

In this section, we examine the performance of the lightweight CNNs, i.e., MobileNet v1, v2 and v3. As shown in Tables 8–11, it is observed that MobileNet v2 shows higher accuracy at only 5∼10% the size of ResNet50, whereas MobileNet v3 Small and Large show not enough accuracy, even with an increased size (141∼391% of MobileNet v2, 13∼37% of ResNet50). MobileNet v1 also shows similar performance to MobileNet v2.

Table 8. MobileNet v1.

DLMO	**CIFAR10**			**CIFAR100**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	81.8	12,840	18	42.2	13,208	34.6
BLQ	81.8	3477	346	42.2	3560	379.7
FIQ	82.4	3517	332	43.1	3612	330
F16	80.9	6435	22	45.1	6620	30.4
PRQ	81.0	1319	350	46.1	1349	350
PRN	81.0	3934	18	45.9	4063	19

Table 9. MobileNet v2.

DLMO	**CIFAR10**			**CIFAR100**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	81.6	8924	11	45.7	9386	21.7
BLQ	81.5	2663	224	45.5	2782	213.6
FIQ	81.2	2730	173	45.9	2848	161
F16	82.0	4509	10.6	41.5	4740	21.6
PRQ	80.2	1082	213	47.2	1118	221
PRN	80.5	2876	8	47.1	2997	7

Table 10. MobileNet v3 small.

DLMO	CIFAR10			CIFAR100		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	68.8	12,161	10	34.8	12,624	24
BLQ	68.7	3269	73	34.9	3389	97
FIQ	69.7	3301	41	35.4	3419	43
F16	72.1	6128	10	35.2	6359	20
PRQ	69.6	1172	64	36.0	1187	69
PRN	69.7	3740	6	36.2	3862	7

Table 11. MobileNet v3 Large.

DLMO	CIFAR10			CIFAR100		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	77.9	34,939	26	37.1	35,401	37
BLQ	78.0	9122	195	37.2	9239	218
FIQ	77.2	9178	149	36.7	9295	150
F16	77.9	17,525	26	38.1	17,756	28
PRQ	75.7	3062	206	39.4	3122	211
PRN	75.6	10,510	19	39.3	10,653	29

However, as shown in Table 12, MobileNet v3 shows a better performance in accuracy than MobileNet v2 in the large dataset, such as ImageNet [12]. The differences in performance between MobileNet v3 with different numbers of parameters are due to differences in training methods, in which training with large parameters in the small dataset introduces an overfitting problem.

On the other hand, the performances of WRN [20] are shown in Table 13. Even though WRN does not utilize the light-weight convolution schemes such as DSP, Linear Bottleneck and SE, it shows the best accuracy with the minimal size. Based on the aforementioned observation, it is recommended that the neural network model is selected with considering the scale of datasets.

Table 12. MobileNet v2, v3 in ImageNet.

Network	Top-1	MAdds	Params
V3-Large1.0	75.2	219	5.4M
V2 1.0	72.0	300	3.4M
V3-Small 1.0	67.4	56	2.5M

Table 13. Wide Residual Networks.

DLMO	CIFAR10			CIFAR100		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
NQ	88.8	1899	39	60.5	192	50
BLQ	88.8	537	1820	59.4	544	1860
FIQ	88.7	538	1913	59.3	544	2009
F16	86.7	982	39	60.0	994	51
QAT	88.6	540	1916	58.9	549	2002
PRN	87.6	547	42	60.2	572	37
PRQ	87.6	195	1902	60.3	200	1860

For the performance evaluation of the KD technique, ResNet54 and MobileNet series are respectively used as teacher and student models in the CIFAR10 dataset. In addition, in order to investigate the combination with the KD and the other model optimizations (quantization and pruning), F16 and PR are used as the representative techniques.

As shown in Table 14, the KD scheme has the benefit of boosting the accuracy. In other words, although the KD itself could not minimize the size or the latency, it could increase the accuracy of lightened models for stable deployment. Moreover, if quantization and pruning are used together in the KD technique, the effect can be further enhanced. As shown in Tables 8–12, it is remarkable that M3L and M3s show better performance in a large-scale dataset such as ImageNet than in a small-scale dataset such as CIFAR10. This is because the M3L and M3s are designed to have the best performance in ImageNet based on NAS, i.e., it can have an inferior performance in other datasets. Therefore, when applying the KD training into those M3L and M3s models for the small-scale dataset, their performance improvement was also limited compared to the other neural networks.

Table 14. Performance Evaluation of KD in the CIFAR10 Dataset.

DLMO	**Original**			**KD**		
Eval Setup	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**	***Acc* (%)**	***Size* (KB)**	***Lat* (ms)**
M1	81.8	12,840	18	86.3	12,840	18
M2	81.6	8924	11	85.4	8924	11
M3L	77.9	34,939	26	60.5	34,939	26
M3s	68.8	12,161	10	60.5	12,161	10
M1-F16	80.9	6435	22	86.3	192	50
M2-F16	82.0	4509	10.6	85.4	192	50
M1-PR	81.0	3934	18	86.3	192	50
M2-PR	80.5	2876	8	85.4	192	50

4.3. Optimization Strategy

First, let us define the service types of AIoT and find out about the optimal DLMO strategy for each service.

The AIoT service performs existing AI services at the edge device level without going through the cloud server. According to the performance requirements, it can be classified into the following three categories:

- **Low-end AIoT Service:** Aims for low-end AIoT service such as vacant parking space detection. Their deep learning models are loaded on the small memory of the IoT device in each parking lot, but the service is not delay-sensitive. In addition, the scale of the required dataset is small, e.g., two classes with vacant and occupied classes:
- **Mid-end AIoT Service:** Aims for Mid-end AIoT Service, such as license plate recognition. The number of classes to be recognized is around 10∼20, and the difficulty of recognition is easy. In addition, the model is normally embedded on IoT devices, and real-time performance is required.
- **High-end AIoT Service:** Aims for High-end AIoT Service, such as autonomous driving. Both real-time performance and accuracy are required.

Then, based on the aforementioned remarks, the following strategies are proposed for each AIoT service:

- **Neural Network Selection:** The neural network model for DLMO needs to be selected considering the scale of the datasets.
- **Low-end AIoT Service:** It is recommended to utilize PRQ because it can minimize the model size with minimal accuracy drops.
- **Mid-end AIoT Service:** It is recommended to utilize PRN because it guarantees all performances of accuracy, latency and size reduction.

- **High-end AIoT Service:** It is better to use F16 or PRN because it guarantees all performances of accuracy, latency and size reduction.
- **KD:** For the low-to-high end AIoT services, KD can be used simultaneously with the aforementioned techniques for boosting the accuracy of the reduced model.

5. Conclusions

In this paper, we analyzed four methods, i.e., Light Convolution, Quantization, Pruning and Knowledge Distillation, for DLMO of edge devices and also derived application strategies according to AI services through experiments. First of all, we found that quantization was the most effective in compressing the model size, but it led to a lot of delay in data conversion. We also found that the pruning technique was excellent in all aspects of model compression, accuracy loss minimization and delay minimization. In particular, the larger the model is, the more effects of model is. Moreover, it is recommended to train the aforementioned deep learning models with knowledge distillation, because it could improve the accuracy without additional increases in latency and size. We found that it was better to select an optimized network after analyzing the data set in the field to be used, rather than selecting the lightweight network technique unconditionally according to insights from this paper. Finally, by classifying AIoT services according to three performance factors (these are accuracy, size, and delay), we derived the optimal combinations of DLMO techniques depending on situations. Moreover, transformer-based approaches [73] have recently become a dominant deep learning method instead of CNNs. However, what we have researched in this paper will be valid for a while because transformer network models are very complicated and require high hardware specifications and are thus not suitable for IoT and embedded devices.

Author Contributions: Conceptualization, H.L., S.L. and N.L.; methodology, H.L., S.L. and N.L.; software, H.L. and S.L.; validation, H.L. and S.L.; resources, H.L. and S.L.; data curation, H.L. and S.L.; writing—original draft preparation, H.L. and S.L.; writing—review and editing, H.L., S.L. and N.L.; visualization, H.L. and S.L.; supervision, H.L. and S.L.; project administration, H.L., S.L. and N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00852, Development of Intelligent Media Attributes Extraction and Sharing Technology) and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1F1A1062878).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; Zhou, T. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* **2019**, *107*, 1537–1562. [CrossRef]
2. Han, S.; Mao, H.; Dally, J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2015**, arXiv:1510.00149.
3. Gundluru, N.; Rajput, D.S.; Lakshmanna, K.; Kaluri, R.; Shorfuzzaman, M.; Uddin, M.; Rahman Khan, M.A. Enhancement of Detection of Diabetic Retinopathy Using Harris Hawks Optimization with Deep Learning Model. *Comput. Intell. Neurosci.* **2022**, *2022*, 8512469 . [CrossRef] [PubMed]
4. Palve, A.; Patel, H. Towards Securing Real Time Data in IoMT Environment. In Proceedings of the International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 24–26 November 2018.
5. Lakshmanna, K.; Kaluni, R.; Gundluru, N.; Alzamil, Z.; Rajput, D.S.; Khan, A.A.; Haq, M.A.; Alhussen, A. A Review on Deep Learning Techniques for IoT Data. *Electronics* **2022**, *11*, 1604. [CrossRef]
6. Rajput, D.S.; Reddy, T.S.K.; Raju, D.N. Investigation on Deep Learning Approach for Big Data: Applications and Challenges. *Deep. Learn. Neural Netw. Concepts Methodol. Tools Appl.* **2020**, *11*, 1604.

7. Deng, L.; Li, G.; Han, S.; Shi, L.; Xie, Y. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proc. IEEE* **2020**, *108*, 485–532. [CrossRef]
8. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
9. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Las Vegas, NV, USA, 27–30 June 2016.
10. Howard, A.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
11. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv* **2018**, arXiv:1801.04381
12. Howard, A.; Sandler, M.; Chu, G.; Chen, L.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019.
13. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.; Keutzer, K. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv* **2021**, arXiv:2103.13630.
14. TensorFlow for Mobile and Edge. Available online: https://www.tensorflow.org/lite (accessed on 1 March 2022).
15. Zhu, M.; Gupta, S. To Prune, or Not To Prune: Exploring the Efficacy of Pruning for Model Compression. *arXiv* **2017**, arXiv:1710.01878.
16. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Lake Tahoe, NV, USA, 3–8 December 2012.
17. Lin, J.; Chen, W.-M.; Lin, Y.; Cohn, J.; Gan, C.; Han, S. MCUNet: Tiny Deep Learning on IoT Devices. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11711–11722.
18. Lin, J.; Chen, W.-M.; Cai, H.; Gan, C.; Han, S. MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), Online, 6–14 December 2021.
19. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Boston, MA, USA, 7–12 June 2015.
20. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. *arXiv* **2016**, arXiv:1605.07146.
21. Xie, S.; Girshick, R.; Dollar, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Honolulu, HI, USA, 21–26 July 2017.
22. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
23. Zoph, B.; Le, Q. Neural Architecture Search with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.01578
24. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning(PMLR), Long Beach, CA, USA, 9–15 June 2019.
25. Bello, I.; Fedus, W.; Du, X.; Cubuk, E.; Srinivas, A.; Lin, T.; Shlens, J.; Zoph, B. Revisiting ResNets: Improved Training and Scaling Strategies. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22614–22627.
26. Custom On-Device ML Models with Learn2Compress. *Google AI Blog*. 2018. Available online: https://ai.googleblog.com/2018/05/custom-on-device-ml-models.html (accessed on 1 March 2022).
27. David, R.; Duke, J.; Jain, A.; Reddi, V.J.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Regev, S.; et al. Tensorflow Lite Micro: Embedded Machine Learning for TinyML systems. *arXiv* **2021**, arXiv:2010.08678.
28. Lai, L.; Suda, N.; Chandra, V. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. *arXiv* **2018**, arXiv:1801.06601.
29. Gural, A.; Murmann, B. Memory-Optimal Direct Convolutions for Maximizing Classification Accuracy in Embedded Applications. In Proceedings of the 36th International Conference on Machine Learning(PMLR), Long Beach, CA, USA, 9–15 June 2019.
30. Sakr, F.; Bellotti, F.; Berta, R.; De Gloria, A.; Doyle, J. Memory-Efficient CMSIS-NN with Replacement Strategy. In Proceedings of the IEEE International Conference on Future Internet of Things and Cloud(FiCloud), Rome, Italy, 23–25 August 2021.
31. Müksch, S.; Olausson, T.; Wilhelm, J.; Andreadis, P. Quantitative Analysis of Image Classification Techniques for Memory-Constrained Devices. *arXiv* **2020**, arXiv:2005.04968.
32. *IEEE STD 754-2019*; IEEE Standard for Floating-Point Arithmetic. IEEE: Piscataway, NJ, USA, 2019; pp. 1–84.
33. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
34. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge Distillation: A Survey. *arXiv* **2021**, arXiv:2006.05525.
35. Meng, Z.; Zhao, Y.; Gong, Y. Conditional Teacher-Student Learning. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), Brighton, UK, 12–17 May 2019.
36. Kim, S.W.; Kim, H.E. Transferring Knowledge to Smaller Network with Class-Distance Loss. In Proceedings of the International Conference on Learning Representations(ICLR) RobustML Workshop, Toulon, France, 24–26 April 2017.
37. Muller, R.; Kornblith, S.; Hinton, G.E. When Does Label Smoothing Help? In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Vancouver, BC, Canada, 8–14 December 2019.
38. Ding, Q.; Wum S.; Sun, H.; Gou, J.; Xia, S. Adaptive Regularization of Labels. *arXiv* **2019**, arXiv:1908.05474.
39. Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y. Fitnets: Hints for Thin Deep Nets. *arXiv* **2015**, arXiv:1412.6550.

40. Zagoruyko, S.; Komodakis, N. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In Proceedings of the International Conference on Learning Representations(ICLR), Toulon, France, 24–26 April 2017.
41. Kim, J.; Park, S.; Kwak, N. Paraphrasing Complex Network: Network Compression via Factor Transfer. In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Montréal, QC, Canada, 2–8 December 2018.
42. Passalis, N.; Tefas, A. Learning Deep Representations with Probabilistic Knowledge Transfer. In Proceedings of the European Conference on Computer Vision(ECCV), Munich, Germany, 8–14 September 2018.
43. Heo, B.; Lee, M.; Yun, S.; Choi, J.Y. Knowledge Distillation with Adversarial Samples Supporting Decision Boundary. In Proceedings of the Association for the Advancement of Artificial Intelligence(AAAI), Honolulu, HI, USA, 27 January–1 February 2019.
44. Heo, B.; Lee, M.; Yun, S.; Choi, J.Y. Knowledge Transfer via Distillation of Activation Boundaries Formed by Hidden Neurons. In Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019.
45. Yim, J.; Joo, D.; Bae, J.; Kim, J. A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Honolulu, HI, USA, 21–26 July 2017.
46. Lee, S.H.; Kim, D.H.; Song, B.C. Self-supervised Knowledge Distillation using Singular Value Decomposition. *arXiv* **2018**, arXiv:1807.06819.
47. Zhang, C.; Peng, Y. Better and Faster: Knowledge Transfer from Multiple Self-supervised Learning Tasks via Graph Distillation for Video Classification. In Proceedings of the International Joint Conferences on Artificial Intelligence(IJCAI), Stockholm, Sweden, 13–19 July 2018.
48. Passalis, N.; Tzelepi, M.; Tefas, A. Heterogeneous Knowledge Distillation using Information Flow Modeling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Seattle, WA, USA, 14–19 June 2020.
49. Huang, Z.; Wang, N. Like What You Like: Knowledge Distill via Neuron Selectivity Transfer. *arXiv* **2019**, arXiv:1707.01219.
50. Mirzadeh, S.I.; Farajtabar, M.; Li, A.; Ghasemzadeh, H. Improved Knowledge Distillation via Teacher Assistant. In Proceedings of the Association for the Advancement of Artificial Intelligence(AAAI), New York, NY, USA, 7–12 February 2020.
51. Li, T.; Li, J.; Liu, Z.; Zhang, C. Few Sample Knowledge Distillation for Efficient Network Compression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Seattle, WA, USA, 14–19 June 2020.
52. Chen, D.; Mei, J.P.; Wang, C.; Feng, Y.; Chen, C. Online Knowledge Distillation with Diverse Peers. In Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), New York, NY, USA, 7–12 February 2020.
53. Xie, J.; Lin, S.; Zhang, Y.; Luo, L. Training Convolutional Neural Networks with Cheap Convolutions and Online Distillation. *arXiv* **2019**, arXiv:1909.13063.
54. Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G.E.; Hinton, G.E. Large Scale Distributed Neural Network Training through Online Distillation. In Proceedings of the International Conference on Learning Representations(ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
55. Zhou, G.; Fan, Y.; Cui, R.; Bian, W.; Zhu, X.; Gai, K. Rocket Launching: A Universal and Efficient Framework for Training Well-performing Light Net. In Proceedings of the Association for the Advancement of Artificial Intelligence(AAAI), New Orleans, LA, USA, 2–7 February 2018.
56. Phuong, M.; Lampert, C.H. Distillation-based Training for Multi-exit Architectures. In Proceedings of the International Conference on Computer Vision(ICCV), Seoul, Korea, 27 October–2 November 2019.
57. Mobahi, H.; Farajtabar, M.; Bartlett, P.L. Self-distillation Amplifies Regularization in Hilbert Space. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 3351–3361.
58. Zhang, Z.; Sabuncu, M.R. Self-Distillation as Instance-Specific Label Smoothing. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 2184–2195.
59. Yuan, L.; Tay, F.E.; Li, G.; Wang, T.; Feng, J. Revisit Knowledge Distillation: A Teacher-free Framework. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Seattle, WA, USA, 14–19 June 2020.
60. Yun, S.; Park, J.; Lee, K.; Shin, J. Regularizing Class-wise Predictions via Self-knowledge Distillation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Seattle, WA, USA, 14–19 June 2020.
61. Hahn, S.; Choi, H. Self-knowledge Distillation in Natural Language Processing. In Proceedings of the International Conference on Recent Advances in Natural Language Processing(RANLP), Varna, Bulgaria, 2–4 September 2019.
62. Zhang, Y.; Xiang, T.; Hospedales, T.M.; Lu, H. Deep Mutual Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
63. Furlanello, T.; Lipton, Z.; Tschannen, M.; Itti, L.; Anandkumar, A. Born Again Neural Networks. In Proceedings of the International Conference on Machine Learning(ICML), Stockholm, Sweden, 10–15 July 2018.
64. Tarvainen, A.; Valpola, H. Mean Teachers Are Better Role Models: Weight-averaged Consistency Targets Improve Semi-supervised Deep Learning Results. In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
65. Wang, H.; Zhao, H.; Li, X.; Tan, X. Progressive Blockwise Knowledge Distillation for Neural Network Acceleration. In Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), Stockholm, Sweden, 13–19 July 2018.

66. Zhu, X.; Gong, S. Knowledge Distillation by On-the-fly Native Ensemble. In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Montreal, QC, Canada, 3–8 December 2018.
67. Polino, A.; Pascanu, R.; Alistarh, D. Model Compression via Distillation and Quantization. In Proceedings of the International Conference on Learning Representations(ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
68. Mishra, A.; Marr, D. Apprentice: Using Knowledge Distillation Techniques to Improve Low-precision Network Accuracy. In Proceedings of the International Conference on Learning Representations(ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
69. Wei, Y.; Pan, X.; Qin, H.; Ouyang, W.; Yan, J. Quantization Mimic: Towards Very Tiny CNN for Object Detection. In Proceedings of the European Conference on Computer Vision(ECCV), Munich, Germany, 8–14 September 2018.
70. Shin, S.; Boo, Y.; Sung, W. Empirical Analysis of Knowledge Distillation Technique for Optimization of Quantized Deep Neural Networks. *arXiv* **2019**, arXiv:1909.01688.
71. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
72. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Honolulu, HI, USA, 21–26 July 2017.
73. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All you Need. In Proceedings of the Conference on Neural Information Processing Systems(NeurIPS), Long Beach, CA, USA, 4–9 December 2017.