INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation

Jangho Kim[1,2,*], *Simyung Chang*[1], *Nojun Kwak*[2]
[1]Qualcomm AI Research[†], Qualcomm Korea YH, South Korea
[2]Seoul National University, South Korea
kjh91@snu.ac.kr, simychan@qti.qualcomm.com, nojunk@snu.ac.kr

Abstract

As edge devices become prevalent, deploying Deep Neural Networks (DNN) on edge devices has become a critical issue. However, DNN requires a high computational resource which is rarely available for edge devices. To handle this, we propose a novel model compression method for the devices with limited computational resources, called *PQK* consisting of pruning, quantization, and knowledge distillation (KD) processes. Unlike traditional pruning and KD, PQK makes use of unimportant weights pruned in the pruning process to make a teacher network for training a better student network without pre-training the teacher model. PQK has two phases. Phase 1 exploits iterative pruning and quantization-aware training to make a lightweight and power-efficient model. In phase 2, we make a teacher network by adding unimportant weights unused in phase 1 to a pruned network. By using this teacher network, we train the pruned network as a student network. In doing so, we do not need a pre-trained teacher network for the KD framework because the teacher and the student networks coexist within the same network (See Fig. 1). We apply our method to the recognition model and verify the effectiveness of PQK on keyword spotting (KWS) and image recognition.

Index Terms: keyword spotting, model pruning, model quantization, knowledge distillation.

1. Introduction

Nowadays, Deep Neural Networks (DNNs) have shown astonishing capabilities in various domains such as computer vision and signal processing. Although DNN shows remarkably high performance, it requires high computational cost and memory. Also, DNN models are spreading from personal computers or servers into edge devices. Deploying DNN on edge devices such as smartphones and IoT devices is still a challenge due to its computational resource constraint and restricted memory.

In recent years, model compression has been actively studied to deal with the above issues. In general, model compression can be categorized into three: pruning, quantization, and knowledge distillation. (1) *Pruning* method prunes the unimportant weights or channels based on different criteria [1, 2, 3, 4]. Pruning method can reduce model memory and the number of flops by eliminating unimportant weights or channels. (2) *Quantization* method quantizes floating point values into discrete values to approximate them by a set of integers and scaling factors [5]. Quantization allows for more power-efficient operations and convolution computations at the expense of lower bitwidth representation. Recently, hardware accelerators such as NVIDIA's Tensor Core and CIM (Compute-in-memory) devices have launched for 4-bit processing to improve the power efficiency [6, 7]. (3) *Knowledge distillation* is a learning framework using teacher and student networks. Teacher network transfers its knowledge to student network to enhance the performance of student network. Feature maps [8, 9, 10] and logits of a network [11, 12] are widely used as knowledge. Model compression has actively been studied mainly on computer vision tasks. However, with the increase of various voice assistants such as Siri, Hey Google, and Alexa on IoT devices, model compression has also become an important research topic in speech processing [13, 14, 15, 16, 17].

In this work, we aim to design the PQK to leverage pruning, quantization, and knowledge distillation by considering each method's characteristics. In contrast with traditional pruning and knowledge distillation, we use unimportant weights considered in the pruning process to make a teacher network, so PQK does not need a pre-trained teacher model. We propose PQK to compress the keyword spotting (KWS) recognition model. PQK can also be used for the image recognition model because the design of PQK is focused on the training framework regardless of datatype and model, which has high applicability. PQK consists of two phases. In the first phase, we train the model from scratch using both iterative pruning and quantization-aware training (QAT). We prune the model and quantize the pruned model with a learnable step-size for QAT. This phase focus on finding a pruned model from scratch together with quantizing the model. In phase 2, we make a teacher network called full net shown in Fig. 1 by combining the pruned net and the unused weights considered unimportant in phase 1. Then, we train the pruned network as a student network. This phase improves the performance of the pruned net (student) by knowledge distillation with the full net (teacher). The details of PQK are explained in Sec. 2 and Fig. 1.

2. Proposed Method

2.1. Preliminaries

Pruning Iterative pruning [1] is widely used in machine learning because it generally outperforms the one-shot pruning method [2]. One-shot pruning just prunes the model once with specific sparsity after model training. Then, it finetunes the model to improve the performance of the pruned model. On the other hand, iterative pruning gradually prunes the model while training and the final model contains unpruned important weights. In this work, we use iterative pruning and adopt the gradually increasing pruning ratio scheme based on the current epoch (c) [1]:

$$p_c = p_t + (p_i - p_t)(1 - \frac{c - c_0}{n})^3. \tag{1}$$

[*]Author completed the research in part during an internship at Qualcomm Technologies, Inc.
[†] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

http://dx.doi.org/10.21437/Interspeech.2021-248

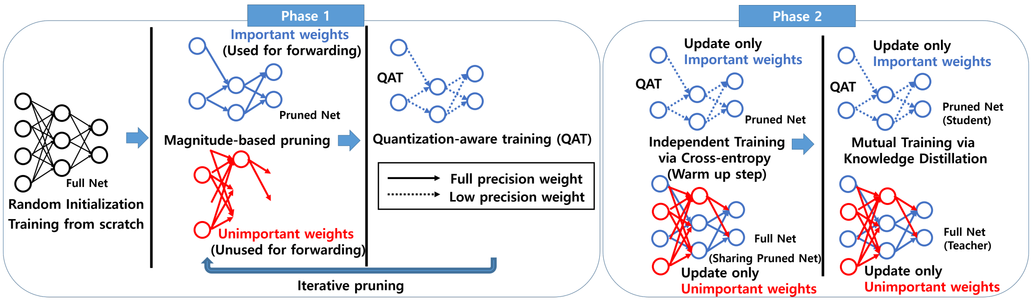

Figure 1: *The overall process of PQK. Phase 1 trains the model from scratch with iterative pruning and quantization-aware training (QAT). The blue nodes and arrows corresponds to important weights used for the QAT, and the red ones are unimportant weights based on pruning method. The solid line and dotted line represent the full precision and k-bit quantized weights, respectively. At phase 2, we make a teacher network with a full network (Blue graph+Red graph). After some warm-up steps, we train the pruned net (student) and the full net (teacher) using KD framework. At teacher training, The blue graph from the student is fixed and shared in the full net, and the the red graph is only updated. However, the blue graph is shared at the forwarding of the full net.*

We increase the pruning ratio from an initial ratio ($p_i = 0$) to a target pruning ratio p_t over the training epoch (n). p_c represents the current pruning ratio per epoch $c \in \{c_0, ..., c_0 + n\}$, where c_0 means an initial epoch ($c_0 = 0$).

Quantization After pruning step, we train the model with quantization-aware training (QAT) using important weights. We choose the uniform symmetric quantization method and the per-layer quantization scheme considering hardware friendliness [5, 18]. Consider the range of model weight [min_w,max_w]. The weight w is quantized to an integer value $\hat{w}$ with the range of [$-2^{k-1}+1$,$2^{k-1}-1$] according to k-bit. Quantization and dequantization for the weight are defined with the learnable stepsize S_w. The overall quantization process is as follows:

$$\hat{w} = Clip(\left\lfloor \frac{w}{S_w} \right\rceil, -2^{k-1}+1, 2^{k-1}-1), \tag{2}$$

where $\lfloor \cdot \rceil$ is the round operation and

$$Clip(w,a,b) = \begin{cases} b & \text{if} \quad w > b \\ a & \text{if} \quad w < a \\ w & \text{otherwise.} \end{cases}$$

Dequantization step just brings the quantized value back to the original range by multiplying the step-size:

$$\bar{w} = \hat{w} \times S_w. \tag{3}$$

These quantization and dequantization processes are non-differentiable, so we utilize a straight-through estimator (STE) [19] for backpropagation. STE approximates the gradient $\frac{d\bar{w}}{dw}$ by 1. Therefore, we can approximate gradients $\mathcal{L}$, $\frac{d\mathcal{L}}{dw}$, with $\frac{d\mathcal{L}}{d\bar{w}}$.

$$\frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{d\bar{w}}\frac{d\bar{w}}{dw} \approx \frac{d\mathcal{L}}{d\bar{w}}. \tag{4}$$

2.2. PQK

Notations Consider a Convolutional Neural Network (CNN) with L layers[1] as an example. Then, we can represent the weights of the CNN model as $\{w_l : 1 \le l \le L\}$. To represent the pruned model with a binary matrix, we use $\{\mathcal{M}_l : 1 \le l \le L\}$. Each $\mathcal{M}_l$, is a binary matrix indicating whether they are pruned or not. Set $\mathcal{I}_l$ is all indices of w_l at the l-th layer. $\mathcal{I}_{\mathcal{M}_l}$ and $\mathcal{I}_{\sim\mathcal{M}_l}$ indicate indices of the important weights (Blue graph in Fig. 1) and unimportant weights (Red graph in Fig. 1) at the l-th layer, respectively ($\mathcal{I}_l = \mathcal{I}_{\mathcal{M}_l} \cup \mathcal{I}_{\sim\mathcal{M}_l}$).

[1]In our notation, a layer contains the corresponding weights.

Algorithm 1 PQK

Input: Untrained model $\mathbf{W}$;
Number of epochs for each phase P_1, P_2;
Number of iterations for mask update p_u and number of epochs for warm up stage s;
pruning mask $\mathcal{M}$, Step-size S_w
Output: Trained model (Full Net) $\mathbf{W}$ and pruned and quantized model (Pruned Net) $\bar{\mathbf{W}} \odot \mathcal{M}$

Phase 1: Pruning and Quantization
for Epoch = 1 ,..., P_1 **do**
  compute sparsity p_c (1)
  for Iter = 1 ,..., N **do**
    if Iter % p_u==0 **then**
      compute mask $\mathcal{M}$ with p_c and magnitude pruning // Update mask every p_u iteration
    end if
    Update S_w, $\mathbf{W}$ (7) by minimizing $\mathcal{L}^S_{ce}$ (6)
  end for
end for
Phase 2: Knowledge Distillation
init $\alpha = 1, \beta = 0$. S_w and $\mathcal{M}$ are fixed // Warm up stage only uses cross-entropy
for Epoch = 1 ,..., P_2 **do**
  if $s <$ Epoch **then**
    set α, β // KD training after warm up stage
  end if
  for Iter = 1 ,..., N **do**
    Update $\mathbf{W}$ (11,12) by minimizing $\mathcal{L}^S_{KD}$ (9) and $\mathcal{L}^T_{KD}$ (10)
  end for
end for

Assuming that we handle a recognition task with m classes, the logit vector of a model is defined as $\mathbf{z}^t$, where $t \in \{S, T\}$ is the type of the network, i.e. either the student or the teacher. The network can have different paths depending on the target bit-width k. We can consider the pruned network as a student network whose path is determined by the masks $\{\mathcal{M}_l\}_{l=1}^L$ and the full network as a teacher network which utilizes all weights (important + unimportant weights). At phase 2, we make a teacher network using both unimportant weights and important weights. Then, we make a soft probability distribution with temperature $\mathcal{T}$ as:

$$\sigma_a(\mathbf{z}^t; \mathcal{T}) = \frac{e^{z^t_a/\mathcal{T}}}{\sum_b^m e^{z^t_b/\mathcal{T}}}, \quad t \in \{S, T\}. \tag{5}$$

Here, $\mathbf{z}^S$ is the logit forwarded by blue graph and $\mathbf{z}^T$ is the logit from blue+red graph, shown in Fig. 1. Based on this notation, we can define the cross-entropy loss as below:

$$\mathcal{L}_{ce}^{t} = -\sum_{a=1}^{m} y_a \log(\sigma_a(\mathbf{z}^t; 1)), \;\; t \in \{S, T\} \tag{6}$$

where y is a ground truth and the subscript a denotes the a-th element of the corresponding vector.

Phase 1 Generally, phase 1 has the same number of epochs compared to conventional training and trains model from scratch. At phase 1, PQK combines iterative pruning and the quantization-aware training. First, PQK prunes the model at some epochs based on the pruning ratio in Eq. (1) by magnitude-based unstructured pruning [2]. It calculates the pruning mask $\mathcal{M}$ which acts as gate functions. Note that, PQK update the pruning mask every p_u-th iteration similar to [1]. Then, QAT is performed with important weights using trainable step-size S_w. By using STE and the chain rule, the update rule at the l-th layer becomes

$$w_l^{(i,j)} \leftarrow w_l^{(i,j)} - \eta \frac{\partial \mathcal{L}_{ce}^{S}}{\partial \bar{w}_l^{(i,j)} \mathcal{M}_l^{(i,j)}}, \;\; \forall (i,j) \in \mathcal{I}_l \tag{7}$$

where, (i, j) is a index of weight matrix. Note that, PQK also updates S_w with $\mathcal{L}_{ce}^{S}$. As depicted in Fig 1, $\mathcal{L}_{ce}^{S}$ is calculated by forwarding only important weights.

Phase 2 At phase 2, PQK trains the full network and pruned network with additional epochs. Commonly, to leverage knowledge distillation, a pre-trained teacher model is needed. Unlike traditional KD, PQK makes a teacher model with unimportant weights which means unused weights at phase 1 (See Fig 1). Note that there is no pre-trained teacher network because the teacher and student network are in the same network (full net).

We can compute the Kullback–Leibler divergence (KL) between student and teacher network.

$$KL(\mathbf{z}^T || \mathbf{z}^S; \mathcal{T}) = \sum_{a=1}^{m} \sigma_a(\mathbf{z}^T; \mathcal{T}) \log(\frac{\sigma_a(\mathbf{z}^T; \mathcal{T})}{\sigma_a(\mathbf{z}^S; \mathcal{T})}) \tag{8}$$

Then, we update each network with cross entropy and KL loss as below:

$$\mathcal{L}_{KD}^{S} = \alpha \mathcal{L}_{ce}^{S} + \beta(\mathcal{T}^2 * KL(\mathbf{z}^T || \mathbf{z}^S; \mathcal{T})) \tag{9}$$

$$\mathcal{L}_{KD}^{T} = \alpha \mathcal{L}_{ce}^{T} + \beta(\mathcal{T}^2 * KL(\mathbf{z}^S || \mathbf{z}^T; \mathcal{T})) \tag{10}$$

$\mathcal{L}_{KD}^{S}$ and $\mathcal{L}_{KD}^{T}$ are the KD loss of pruned net (student) and full net (teacher), respectively. α and β are hyper-parameters for balancing between *KL* and cross-entropy losses. $\mathcal{T}^2$ is multiplied to the KL loss because the gradient with respect to the logit decrease as much as $1/\mathcal{T}^2$. The update rules at the l-th layer becomes

$$w_l^{(i,j)} \leftarrow w_l^{(i,j)} - \eta \frac{\partial \mathcal{L}_{KD}^{S}}{\partial \bar{w}_l^{(i,j)} \mathcal{M}_l^{(i,j)}}, \;\; \forall (i,j) \in \mathcal{I}_{\mathcal{M}_l} \tag{11}$$

$$w_l^{(i,j)} \leftarrow w_l^{(i,j)} - \eta \frac{\partial \mathcal{L}_{KD}^{T}}{\partial w_l^{(i,j)}}, \;\; \forall (i,j) \in \mathcal{I}_{\sim \mathcal{M}_l} \tag{12}$$

Note that, with respect to the pruned network, based on Eq. (11) keeping the same bitwidth of phase 1, phase 2 updates only important weights unlike phase 1 (Eq. (7)) updating all weights. Analogous to the pruned network, in terms of the full network, phase 2 updates only unimportant weights (Eq. (12)). At the forwarding path of the full network, pruned network is shared and the full network does not use QAT. Also, we fix S_w at phase 2 for a stable training.

At first few epochs, we set the hyper-parameters as $\alpha = 1, \beta = 0$ meaning that both pruned and full nets are trained by cross-entropy only because initial unimportant weights are not trained well at phase 1. Thus, it needs a warm up stage. The overall process of PQK is depicted in Fig 1 and Algorithm 1.

3. Experiments

We verify the proposed PQK on a keyword spotting task and an image recognition task. We set the target pruning ratio p_t (Eq. (1)) as 0.9 that means we only use 10% parameters of the baseline model. For ResNet-8 [20], we also conduct various target pruning ratio ($p_t \in \{0.9, 0.7, 0.5\}$). We quantize the model by 8-bit and 4-bit ($k \in \{8, 4\}$, Eq. (2)) compared to the 32-bit baseline model. Although we have one network in PQK, at phase 2, we have twice forwarding for pruned net and full net, so they need different batch statistics in phase 2. Therefore, we use different batchnorm parameters for each net in phase 2.

3.1. Experimental Setup

We update the pruning mask per 32 iterations (p_u) at phase 1. After the warm up stage in phase 2, we set $\mathcal{T} = 2, \alpha = 0, 5, \beta = 0.5$. We did not conduct a grid search for finding hyper-parameters but choose them based on recommendations from related works [21, 12, 1]. For the learning rate of learnable step-size S_w, we multiply 10^{-4} to the initial learning rate of model parameters because of its sensitivity.

Keyword Spotting We use Google's Speech Commands Dataset v1 [22], choosing ResNet-8 and ResNet-8-narrow [20] as baselines using the official code in pytorch[2]. At phase 1, we follow overall training details from [20]. At phase 2, we run 9 epochs for additional training. We start learning rate of 0.1 and decay it at 1000 and 2000 iteration by multiplying 0.1. We set the warm up iteration (s) as 1500.

Image Recognition We use CIFAR100 Dataset [23] and choose ResNet-32 [24] for the baseline. We follow the training details same as [24]. At phase 2, we use 60 additional epochs. We train the model with an initial learning rate of 0.2 and decay it at 20,40 epochs with a factor of 0.1. We start KD after 30 epochs ($s = 30$).

3.2. Experimental Results

In this section, we show the results of PQK with various methods, bitwidths, and pruning ratios. We will refer to the baseline network which is trained with cross-entropy as vanilla in all experiments. There are two forwarding paths in the output of phase 2. The first one, pruned net (student), uses $(1-p_t) \times 100\%$ and quantized parameters. The other one, full net (teacher), uses the whole parameters same as the vanilla. At every table, we refer to our method at the end of each phase and network type $\in \{P, F\}$, where P and F represent pruned net and full net, respectively. Full net contains and shares the pruned net so the bitwidth of full net is 32-bit containing the pruned net trained with k bitwidth QAT. For example, at the 4-th row in Table 1, phase2-F and 32 ($P = 8$) means full net of phase 2 with 32-bitwidth, sharing pruned net trained with 8-bitwidth QAT.

Keyword Spotting As shown in Table ($p_t = 0.9$), the performance of phase1-P decreases compared to vanilla. This is because, in this phase, we pruned unimportant 90% weights of full net and quantize important 10% weights from 32-bit to 8-bit or 4-bit using iterative pruning and QAT. A compact model,

[2] https://github.com/castorini/honk

Table 1: *Test accuracy with various setting on speech and image dataset.*

			CIFAR100	Google's Speech Commands Dataset			
				$p_t = 0.9$		$p_t = 0.7$	$p_t = 0.5$
			ResNet-32	ResNet-8-narrow	ResNet-8	ResNet-8	ResNet-8
Method	Bitwidth	Pruning ratio	Accuracy (%)	Accuracy (%)	Accuracy (%)	Accuracy (%)	Accuracy (%)
Vanilla	32	0	69.7	91.4	94.3	94.3	94.3
Phase1-P	8	p_t	67.4	81.7	92.6	94.3	94.3
Phase2-P	8	p_t	69.8	86.4	94.0	94.6	94.7
Phase2-F	32 ($P = 8$)	0	71.1	90.1	94.4	94.8	94.9
Phase1-P	4	p_t	66.2	74.1	91.7	94.3	94.1
Phase2-P	4	p_t	67.7	83.1	93.6	94.1	94.6
Phase2-F	32 ($P = 4$)	0	69.8	85.1	93.4	93.2	93.7

ResNet-8-narrow using fewer channels than ResNet-8, is more sensitive to the model compression. It degrades 9.7% and 17.3% at 8-bit and 4-bit in phase1-P. Such severe performance degradation of the compact model with quantization is also reported in other researches [25, 12]. At phase 2, by training unimportant 90% weights of the full net, it becomes a teacher to improve the performance of the pruned net. Surprisingly, there is a large performance gap between phase 1 and 2 in the pruned model of ResNet-8-narrow compared to that of ResNet-8. Table 1 shows that PQK is more effective at the compact model in terms of recovering the decreased performance at phase 1, where performance enhancements are 4.7% and 9% at 8- and 4-bit in ResNet-8-narrow. Concerning bitwidth and accuracy, 8-bit consistently outperforms 4-bit because of its high representative power from more bitwidth. In ResNet-8, Regardless of bitwidth, as pruning ratio decreases, the performance of pruned net increases. These numbers show the usage of model parameters is important to the performance. In ResNet-8 with 4-bit, although pruned net performs well, accuracies of phase2-F are lower than those of phase2-P. ResNet-8 with 8-bit shows the opposite trend, meaning that combining 32-bit and 4-bit model is more unstable than combining 32-bit and 8-bit model.

Image Recognition In this experiment, we can show the applicability of PQK. Image recognition task has a similar tendency with KWS task. Interestingly, the performance of phase2-F containing 8-bit pruned net outperforms vanilla by 1.4%. At phase 2, the teacher network is also trained with KD using the student network (Eq. (10)). In doing so, the full net can outperform the vanilla.

Ablation study To show the effectiveness of phase 2 in PQK, we conduct an ablation study in Table 2. At phase 2, we have additional epochs for the boosting performance of the pruned net. We make various baselines using the same training budget as phase 2. Finetune in Table 2 represents the performance of finetuning 4-bit ResNet-8-narrow model from phase1-P (Table 1) with additional training using various learning rates. We use the same experiment setting with phase2-P using 9 epochs and decay learning rate at 1000 and 2000 iteration. The only difference is the existence of KD with full net. In finetuning methods, the high learning rate is more efficient than the lower one, where 0.1 shows the best performance along with various learning rates. However, phase 2 using KD framework utilizing unused weights in phase 1 outperforms the best finetuning method by 3.9%. We also plot the validation accuracy of finetuning and phase-2 of PQK per every epoch in Fig. 2. In this figure, orange and blue line represent the validation accuracy of phase2-P and finetune (lr=0.1) in Table 2. During the warm up step, two methods show very similar trends because they are trained with only cross-entropy. After warm up step, the performance gap between them increases because mutual KD training helps to enhance the performance of both pruned and full net.

Table 2: *Ablation study on PQK: comparing PQK (phase2-P) with finetuning using same training budget on various learning rate, where all methods start from phase1-P.*

Google's Speech Commands Dataset ResNet-8-narrow			
Method	Bitwidth	Accuracy (%)	Pruning ratio
Phase1-P	4	74.1	0.9
Finetune (lr=0.1)	4	79.2	0.9
Finetune (lr=0.01)	4	78.0	0.9
Finetune (lr=0.001)	4	73.8	0.9
Phase2-P	4	83.1	0.9

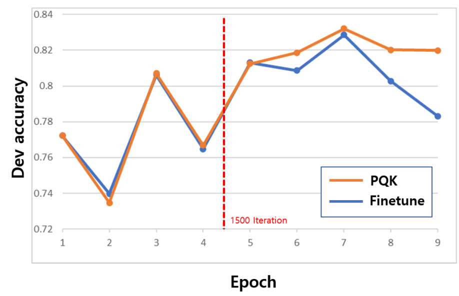

Figure 2: *Dev accuracy of ResNet-8-narrow on google's speech command dataset at every epoch :Orange line represents PQK (Phase2-P) and blue line shows the finetune (lr=0.1). Red dot line means the end of warm up iteration.*

4. Conclusions

We propose a novel model compression framework to cope with the limited computational resource. This is a new way of model compression by leveraging pruning, quantization, and knowledge distillation. In phase 1, we are combining pruning and quantization to make a lightweight and power-efficient model. Then, in phase 2, we boost the performance of a efficient model by KD. We verify the efficiency of PQK on KWS and image recognition tasks.

5. Acknowledgement

Nojun Kwak was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (2021R1A2C3006659). The research that is the subject of this paper, and the paper itself, was solely funded by Qualcomm Technologies, Inc.

6. References

[1] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," *International Conference on Learning Representations*, 2020.

[2] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *International Conference on Learning Representations*, 2015.

[3] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] J. Kim, S. Chang, S. Yun, and N. Kwak, "Prototype-based personalized pruning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3925–3929.

[5] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[6] Y. Pan, P. Ouyang, Y. Zhao, W. Kang, S. Yin, Y. Zhang, W. Zhao, and S. Wei, "A multilevel cell stt-mram-based computing in-memory accelerator for binary convolutional neural network," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.

[7] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

[8] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1921–1930.

[10] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4619–4625.

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

[12] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, "Qkd: Quantization-aware knowledge distillation," *arXiv preprint arXiv:1911.12491*, 2019.

[13] Y. Bai, J. Yi, J. Tao, Z. Tian, Z. Wen, and S. Zhang, "Listen Attentively, and Spell Once: Whole Sentence Generation via a Non-Autoregressive Architecture for Low-Latency Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 3381–3385.

[14] C. Jose, Y. Mishchenko, T. Sénéchal, A. Shah, A. Escott, and S. N. P. Vitaladevuni, "Accurate Detection of Wake Word Start and End Using a CNN," in *Proc. Interspeech 2020*, 2020, pp. 3346–3350.

[15] S. Adya, V. Garg, S. Sigtia, P. Simha, and C. Dhir, "Hybrid Transformer/CTC Networks for Hardware Efficient Voice Triggering," in *Proc. Interspeech 2020*, 2020, pp. 3351–3355.

[16] H. D. Nguyen, A. Alexandridis, and A. Mouchtaris, "Quantization Aware Training with Absolute-Cosine Regularization for Automatic Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 3366–3370.

[17] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.

[18] J. Kim, K. Yoo, and N. Kwak, "Position-based scaled gradient for model quantization and pruning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 20 415–20 426.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[20] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[21] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting." in *Interspeech*, 2016, pp. 1878–1882.

[22] P. Warden, "Launching the speech commands dataset," in *Google Research Blog*, 2017.

[23] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.