Inplace knowledge distillation with teacher assistant for improved training of flexible deep neural networks

Alexey Ozerov, Ngoc Q K Duong

Inplace knowledge distillation with teacher assistant for improved training of flexible deep neural networks

Alexey Ozerov
InterDigital R&D France
Cesson-Sévigné, France
alexey.ozerov@interdigital.com

Ngoc Q. K. Duong
InterDigital R&D France
Cesson-Sévigné, France
quang-khanh-ngoc.duong@interdigital.com

***Abstract*—Deep neural networks (DNNs) have achieved great success in various machine learning tasks. However, most existing powerful DNN models are computationally expensive and memory demanding, hindering their deployment in devices with low memory and computational resources or in applications with strict latency requirements. Thus, several resource-adaptable or flexible approaches were recently proposed that train at the same time a big model and several resource-specific sub-models. Inplace knowledge distillation (IPKD) became a popular method to train those models and consists in distilling the knowledge from a larger model (*teacher*) to all other sub-models (*students*). In this work a novel generic training method called *IPKD with teacher assistant (IPKD-TA)* is introduced, where sub-models themselves become *teacher assistants* teaching smaller sub-models.**
We evaluated the proposed IPKD-TA training method using two state-of-the-art flexible models (MSDNet and Slimmable MobileNet-V1) with two popular image classification benchmarks (CIFAR-10 and CIFAR-100). Our results demonstrate that the IPKD-TA is on par with the existing state of the art while improving it in most cases.

***Index Terms*—Deep Neural Networks, Flexible Models, Inplace Knowledge Distillation with Teacher Assistant**

I. Introduction

Deep neural networks (DNNs) have achieved state-of-the-art results in many machine learning applications in the areas such as computer vision [1], [2], speech recognition [3] and natural language processing [4]. Most of actual architectures are trained for specific tasks and have fixed complexity and performance at the inference. Although it is established that introducing more parameters often improves the accuracy of a model, bigger models are computationally and memory-wise too expensive to be deployed on the consumer electronics (CE) or internet of things (IoT) devices which have limited capacities (computational and memory resources).

One way to solve this problem consists in using model compression techniques targeting at the same time to reduce the model size and to accelerate it at the inference [5]. Most popular model compression methods are based on parameter quantization [6] and pruning [7], low-rank factorization [8], and knowledge distillation (KD) [9]. Quantization consists in simply quantizing the weights, while pruning consists in removing non-important weights [7] or entire convolutional channels [10] (structural pruning) according to some criterion. Low-rank factorization-based techniques use matrix/tensor structural decompositions to approximate matrices/tensors of weights. In this work particular attention will be paid to KD-based methods that consist in compressing a big so-called *teacher* model by distilling its knowledge to a smaller (compressed) so-called *student* model. The KD itself is achieved during the training of student model by replacing or completing the ground truth labels with their teacher model probabilistic predictions. Model compression schemes, while very efficient, allow producing just one compressed model at a time that has fixed memory, run-time and performance characteristics. However, in many cases the available resources of the device on which the model will be executed may not be known in advance, and yet on the same device those resources might vary in time due to other processes. As such, models allowing for an on-demand trade-off between resources and performance, without any re-training or fine-tuning, become of great interest.

Several resource-adaptable models/frameworks, here referred to as *flexible models*, that allow for such on-demand resources/performance trade-off have recently emerged [11]–[17]. At a very high level of abstraction, a flexible model is constituted of several models (one per resources/performance operating point) that are embedded one into another like Matryoshka dolls with strong parameters sharing: one largest model and several sub-models. At the inference a suitable sub-model corresponding to the available resources may be instantly extracted and deployed, and, thanks to the strong parameters sharing, the full model may be efficiently transmitted and stored, as compared to a dummy solution of training several independent models (one per resources/performance operating point). Without loss of generality we here consider flexible DNNs applied to classification tasks, and, in particular, to image classification. Among most recent and efficient flexible models, we may mention the following ones. Multi-scale dense network (MSDNet) [11] is a particular architecture with early-exits (classifications), where the flexibility is achieved by

The present work has been supported by European Union's Horizon 2020 research and innovation program under grant number 951911 - AI4Media.

stopping computation at any desired classifier. Slimmable [12] and universally slimmable [13] networks represent a general framework allowing for an instantaneous slimming [10] (or structural pruning) of a single network into different sub-networks that are all trained jointly. One for all framework [14] is based on the same principles as slimmable networks, while introducing more variability in sub-model's design, including elastic resolution, kernel size, depth and width. Ruiz and Verbee introduced convolutional neural mixture models [15] and hierarchical neural ensembles [16], where the flexibility is achieved by selecting respective sub-mixtures or sub-ensembles. Finally, switchable precision neural networks [17] allow for a flexibility thanks to a possibility of on-demand switching between different levels of network weight quantization.

The most straightforward way to learn flexible models is to train all sub-models jointly from the annotated data [11], [12]. A more efficient training scheme called *inplace knowledge distillation (IPKD)* was introduced in [13] (then re-used in [14]–[17]), where the training is again joint, though only the biggest model is fully trained from the data, while all other sub-models are distilled from the biggest one. The new term *inplace* refers to the fact that there is a strong parameters sharing between the sub-models. However, in an alternative work on KD [18], that is not related to flexible models, it was noted that when the gap (in size) between the teacher and student models is large, the KD might be less efficient. To overcome this drawback the authors of [18] introduce an in-between model, a so-called *teacher assistant*, that first learns from the teacher and then teaches the student.

In this work we build on the idea of KD with teacher assistant [18] to improve the IPKD training of flexible models. Indeed, in IPKD all sub-models are distilled from the biggest one, and thus for smaller sub-models the teacher-student gaps are large. To overcome this issue, we introduce *IPKD with one teacher assistant (IPKD-TA-1)* and *IPKD with multiple teacher assistants (IPKD-TA-M)*, where each sub-model is distilled from the larger sub-model next to it (one teacher assistant) or from all the larger sub-models (multiple teacher assistants), respectively. Our proposal is general and applicable for training any flexible model, where IPKD is applicable. We investigate the effectiveness of the proposed approach in case of MSDNet [11] and Slimmable [12], [13] MobileNet-V1 [19] models on two standard image classification benchmarks: CIFAR-10 and CIFAR-100 [20]. Note that, to our best knowledge, even the IPKD approach was not yet explored for MSDNet model, possibly because MSDNet [11] was published before IPKD was introduced in [13]. Our contributions may be summarized as: (i) introducing novel IPKD-TA-1 and IPKD-TA-M approaches for improved training of general flexible models, (ii) experimental investigation of IPKD approach for MSDNet, and (iii) experimental investigation of proposed IPKD-TA-1 and IPKD-TA-M approaches for MSDNet and Slimmable MobileNet-V1.

The rest of this paper is organized as follows. Related work and necessary background are described in Section II. IPKD-TA-1 and IPKD-TA-M approaches for improved flexible models training are introduced in Section III. Section IV is devoted to experiments and conclusions are drawn in Section V.

II. Related work and background

A. Knowledge distillation

Knowledge distillation (KD) [9] is a general technique allowing to compress a big pre-trained model or model ensemble called *teacher* into a smaller *student* model. It is shown in [9] that using KD allows often for better performance than simply training the student model from the same data via supervised learning.

Let us consider a classification problem with C classes. Let C-dimensional vectors a_s and a_t be the logits (the inputs to the final softmax) of the teacher and student networks, respectively. In classical supervised learning the student model is usually learned by optimizing the cross-entropy loss: [1]

$$\mathcal{L}_{CE}(a_s, y_r) = \mathcal{H}\left(\mathrm{softmax}(a_s), y_r\right), \tag{1}$$

where y_r is a C-dimensional hot vector encoding of the ground truth label, and $\mathcal{H}(x, z) = -\sum_{c=1}^{C} x_c \log(z_c)$ is the cross-entropy.

In KD framework [9] an additional term distilling the knowledge from the teacher is considered

$$\mathcal{L}_{KD}(a_s, a_t) = \tau^2 \mathcal{D}_{KL}\left(\mathrm{softmax}(a_s/\tau), \mathrm{softmax}(a_t/\tau)\right), \tag{2}$$

where $\mathcal{D}_{KL}(x, z) = \sum_{c=1}^{C} x_c \log\left(x_c/z_c\right)$ is the Kullback-Leibler (KL) divergence, [2] and hyperparameter τ referred to temperature is introduced to put additional control on softening of signal arising from the output of the teacher model.

The final loss for training student model combines both the supervised learning loss and the KD loss as:

$$\mathcal{L}_{KD}^{student} = (1-\lambda)\mathcal{L}_{CE}(a_s, y_r) + \lambda \mathcal{L}_{KD}(a_s, a_t), \tag{3}$$

where $\lambda \in [0, 1]$ is a constant hyperparameter weighing the contribution of each term.

B. Knowledge distillation with teacher assistant

It was remarked in [18] that when the gap in model size between the teacher and the student is big enough, distilling the knowledge directly from the teacher might be sub-optimal. To overcome this issue the authors of [18] introduced an intermediate *teacher assistant* model that is first learned by the teacher and then teaches the student. This approach has been shown [18] to lead to a better student model performance.

The KD with teacher assistant concept may be easily understood intuitively. Indeed, an university professor may not be an ideal teacher for primary school children. However, the following usual way is efficient: a university professor teaches a school teacher who then teaches primary school children.

[1] Throughout the paper and without loss of generality, all the losses are expressed for just one data sample, and they should be averaged over the corresponding batch for a final implementation.

[2] According to [9] cross-entropy may be used as well in KD term (2) instead of the KL divergence.

C. Flexible DNNs

We here describe briefly the two flexible models we consider in this work: MSDNet [11] and Slimmable networks [12], [13]. Though, there are many other flexible models [14]–[17] to which our proposed approach is applicable.

1) Models: **MSDNet** [11] is a densely-connected convolutional neural network (CNN) proposed for image classification. It has a two-dimensional (scale and depth) architecture and is divided along the depth dimension into several blocks. An early-exit classifier is implemented at the end of each block, and the flexibility is achieved by a possibility to stop the computation at any desired classifier.

Slimmable networks [12], [13] is a general framework applicable to CNNs that are suitable for slimming (a particular structural pruning) [10], and we here investigate it in case of MobileNet-V1 architecture [19]. Sub-networks of a slimmable network are slimmed versions of the full network obtained by removing entire convolutional channels so as to have different widths, *e.g.,* via scaling the full network width by 0.25, 0.5, 0.75, and 1.0. All sub-networks share their parameters, except for the batch normalization [21] statistics.

2) Conventional training: Let flexible DNN include n sub-models enumerated in the order of their sizes (the largest network is indexed by n). Let vector $a_s[i]$ or $a_t[i]$ be the logits of the i-th model ($i = 1, \dots, n$). [3] The most conventional way to train flexible DNN consists in a supervised joint learning of all sub-networks as:

$$\mathcal{L}^{flex} = \sum_{i=1}^{n} \mathcal{L}_{CE}(a_s[i], y_r), \tag{4}$$

with $\mathcal{L}_{CE}(\cdot, \cdot)$ specified in (1).

D. Inplace knowledge distillation for Flexible DNNs

It was proposed in [13] to still train the sub-models jointly, but not all of them in a completely supervised manner: the largest n-th model is trained in a supervised way, while all other sub-models are distilled from the largest model. The resulting inplace knowledge distillation (IPKD) optimization loss writes

$$\mathcal{L}^{flex}_{IPKD} = \mathcal{L}_{CE}(a_s[n], y_r) + (1-\lambda) \sum_{i=1}^{n-1} \mathcal{L}_{CE}(a_s[i], y_r) + \\ + \lambda \sum_{i=1}^{n-1} \mathcal{L}_{KD}(a_s[i], a_t[n]), \tag{5}$$

with $\mathcal{L}_{KD}(\cdot, \cdot)$ specified in (2). A new adverb *inplace* in IPKD refers to the fact that the KD is now performed jointly with a strong parameters sharing between the sub-models. It was shown in [13] for Slimmable networks that the IPKD training outperforms the conventional training (4). The IPKD strategy was then adopted in [14]–[17].

[3] Though the vectors $a_s[i]$ and $a_t[i]$ represent the same quantity, we distinguish between their notations to indicate within corresponding criteria whether a vector is in the role of a student ($a_s[i]$: the corresponding model parameters are optimized) or of a teacher ($a_t[i]$: the vector is just used as an input).

III. Proposed approaches

We build our proposed training approaches on the idea of KD with teacher assistant (Sec. II-B), though introducing it within IPKD training of flexible DNNs. Indeed, all previous approaches [12]–[17] are using IPKD without teacher assistant, *i.e.,* by always distilling the knowledge from the largest model.

A. IPKD with one teacher assistant

We first introduce the IPKD with one teacher assistant (IPKD-TA-1), where each sub-model is taught by another sub-model that is just next to it from the top. This leads to the following re-formulation of IPKD loss (5):

$$\mathcal{L}^{flex}_{IPKD-TA-1} = \mathcal{L}_{CE}(a_s[n], y_r) + (1-\lambda) \sum_{i=1}^{n-1} \mathcal{L}_{CE}(a_s[i], y_r) + \\ + \lambda \sum_{i=1}^{n-1} \mathcal{L}_{KD}(a_s[i], a_t[i+1]). \tag{6}$$

B. IPKD with multiple teacher assistants

Another strategy we introduce and investigate is the IPKD with multiple teacher assistants (IPKD-TA-M), where each sub-model is taught by all the larger sub-models [4]. This is achieved by writing the corresponding loss as follows:

$$\mathcal{L}^{flex}_{IPKD-TA-M} = \mathcal{L}_{CE}(a_s[n], y_r) + (1-\lambda) \sum_{i=1}^{n-1} \mathcal{L}_{CE}(a_s[i], y_r) + \\ + \lambda \sum_{i=1}^{n-1} \frac{1}{n-i} \sum_{j=i+1}^{n} \mathcal{L}_{KD}(a_s[i], a_t[j]), \tag{7}$$

where the weights $\frac{1}{n-i}$ (such that $\sum_{j=i+1}^{n} \frac{1}{n-i} = 1$) are introduced in order to re-balance the impacts of the teacher assistants, since the number of teacher assistants varies from one sub-model to another.

IV. Experiments

A. Datasets

We evaluate the purposed approaches on two following popular image classification benchmarks. CIFAR-10 [20] consists of 60k color images 32×32 split into 10 classes. CIFAR-100 [20] consists of the same images as CIFAR-10, though they are split onto 100 classes.

B. Experimental setup

We investigate the proposed approach on two different flexible models: MSDNet [11] and Slimmable [12], [13] MobileNet-V1 [19]. Our approach and all the baselines are implemented based on the corresponding implementations and model architectures available at https://github.com/kalviny/MSDNet-PyTorch and https://github.com/JiahuiYu/slimmable_networks, respectively. Since the original Slimmable model implementation was only for ImageNet dataset (and not CIFAR-10 or CIFAR-100), some parameters,

[4] In general each sub-model can be taught by a subset of larger sub-models.

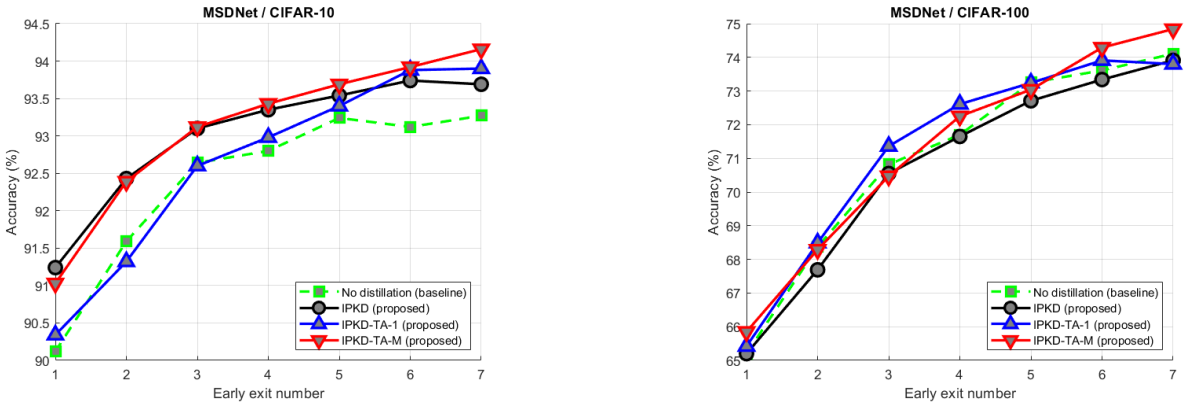

Fig. 1: Results in terms of classification accuracy for MSDNet on CIFAR-10 (left) and CIFAR-100 (right).

Dataset	CIFAR-10				CIFAR-100			
Method	No dist. (baseline)	IPKD (proposed)	IPKD-TA-1 (proposed)	IPKD-TA-M (proposed)	No dist. (baseline)	IPKD (proposed)	IPKD-TA-1 (proposed)	IPKD-TA-M (proposed)
Exit 1	90.12	**91.24**	90.34	91.03	65.20	65.20	65.42	**65.85**
Exit 2	91.59	**92.43**	91.32	92.39	68.37	67.69	**68.49**	68.29
Exit 3	92.64	93.10	92.60	**93.12**	70.82	70.55	**71.36**	70.47
Exit 4	92.80	93.35	92.98	**93.43**	71.70	71.65	**72.61**	72.25
Exit 5	93.24	93.54	93.40	**93.69**	**73.25**	72.71	73.24	73.04
Exit 6	93.12	93.74	93.88	**93.92**	73.61	73.34	73.91	**74.29**
Exit 7	93.27	93.69	93.90	**94.16**	74.11	73.91	73.80	**74.84**
Avg	92.39	93.01	92.63	**93.10**	71.00	70.72	71.26	**71.29**

TABLE I: Detailed results in terms of classification accuracy (%) for MSDNet (best performance for each exit is in bold).

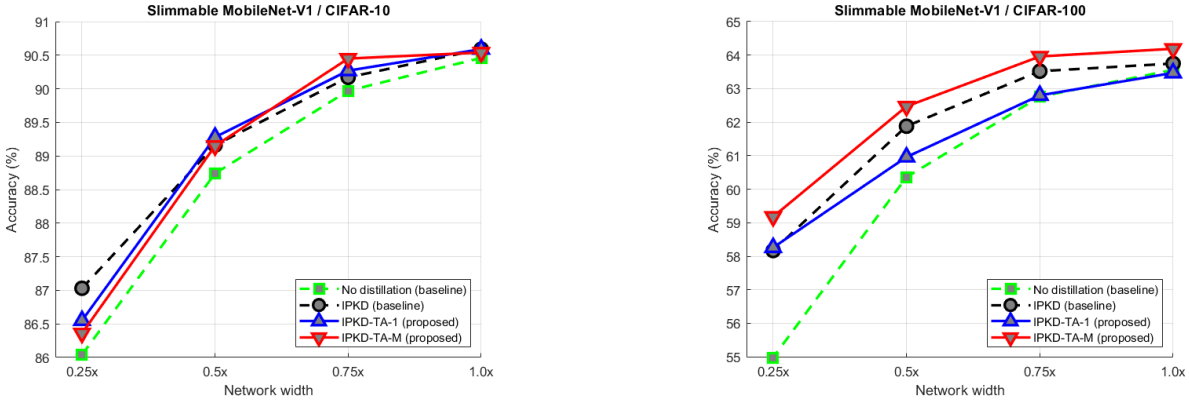

Fig. 2: Results in terms of classification accuracy for Slimmable MobileNet-V1 on CIFAR-10 (left) and CIFAR-100 (right).

Dataset	CIFAR-10				CIFAR-100			
Method	No dist. (baseline)	IPKD (baseline)	IPKD-TA-1 (proposed)	IPKD-TA-M (proposed)	No dist. (baseline)	IPKD (baseline)	IPKD-TA-1 (proposed)	IPKD-TA-M (proposed)
Switch 1 ($\times$ 0.25)	90.46	**90.59**	**90.59**	90.54	63.56	63.75	63.47	**64.19**
Switch 2 ($\times$ 0.5)	89.97	90.17	90.27	**90.45**	62.76	63.52	62.80	**63.96**
Switch 3 ($\times$ 0.75)	88.74	89.16	**89.28**	89.15	60.36	61.88	60.96	**62.47**
Switch 4 ($\times$ 1.0)	86.04	**87.03**	86.55	86.36	54.98	58.17	58.26	**59.18**
Avg	88.80	**89.24**	89.17	89.13	60.42	61.83	61.37	**62.45**

TABLE II: Detailed results in terms of classification accuracy (%) for Slimmable MobileNet-V1 (best performance for each switch is in bold).

incuding stride, were adapted as in the following library: https://github.com/weiaicunzai/pytorch-cifar100.

All models are trained for 300 and 200 epochs for MSDNet and Slimmable MobileNet-V1, respectively, with an initial learning rate of 0.1 using the stochastic gradient descent (SGD) optimizer. We have chosen temperature hyperparameter

$\tau = 5$ and $\tau = 1$ in (2) for MSDNet and Slimmable MobileNet-V1, respectively. Note also that for Slimmable MobileNet-V1 we used cross-entropy loss instead of the KL divergence in (2), since the same choice was done in the baseline implementation [12], [13]. Penalty factor λ in (5), (6) and (7) was set to $\lambda = 0.8$.

The performance of each flexible DNN sub-model is measured in terms of classification accuracy [11], [12]. For MSDNet we measure the performance for each exit, and consider training without distillation as a baseline, since even IPKD was not yet applied for MSDNet. For Slimmable MobileNet-V1 we measure the performance for each of four switches ($\times$ [0.25, 0.5, 0,75, 1.0]), and consider the IPKD training with no distillation as a baseline. Following [12], [13], we report for each experiment the results of the epoch leading to the highest classification accuracy averaged over all sub-models (*i.e.,* exits or switches) on validation set.

C. Results

Results of our experiments with MSDNet are reported in Figure 1 and Table I. We may see that, as compared to the baseline supervised training without distillation, the investigated IPKD training improves the results for CIFAR-10 dataset. The proposed IPKD-TA-1 offers the best performance for some early exits in CIFAR-100 dataset. Overall, the proposed IPKD-TA-M training approach offers better performance for both CIFAR-10 and CIFAR-100 datasets than the IPKD and the baseline. It is also interesting to see that the improvement is consistent at most intermediate classifiers.

Figure 2 and Table II summarize our experiment results with Slimmable MobileNet-V1. The improvements of the IPKD-TA-M, as compared to the IPKD training baseline, are also consistent and convincing for the CIFAR-100 dataset. We can see that the proposed IPKD-TA-M offers the best performance for all sub-models for CIFAR-100 dataset while resulting in similar performance with the IPKD and IPKD-TA-1 on CIFAR-10 dataset. This confirms our hypothesis that reducing big gaps between the teacher model and small student sub-models by exploiting teacher assistants is helpful.

V. Conclusion

In this work we have considered a family of flexible DNNs that are able instantly adapting to the available (*e.g.,* computational and memory) resources for an efficient deployment. We have mainly focused on improving training of those models. Starting from recently proposed IPKD training, we have noted that this approach might be less efficient when the gap (in size) between the largest teacher model and a student sub-model is big. As such, to overcome this drawback and inspired by recently proposed knowledge distillation with teacher assistant, we have introduced new so-called *IPKD with teacher assistant (IPKD-TA)* flexible model training strategies.

Our proposed training strategies are general and applicable to many existing flexible DNN approaches. We have investigated them and compared to the state-of-the-art for two different flexible architectures (MSDNet and Slimmable MobileNet-V1) on two popular image classification benchmarks (CIFAR-10 and CIFAR-100). We have observed that in most cases one of the proposed IPKD-TA approach outperforms the state-of-the-art training methods.

References

[1] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[2] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in English and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018.

[6] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao, "Improving the speed of neural networks on CPUs," in *in Deep Learning and Unsupervised Feature Learning Workshop, NIPS*. Citeseer, 2011.

[7] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[8] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[11] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations*, 2018.

[12] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang, "Slimmable neural networks," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[13] Jiahui Yu and Thomas S Huang, "Universally slimmable networks and improved training techniques," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1803–1811.

[14] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2019.

[15] Adria Ruiz and Jakob Verbeek, "Adaptative inference cost with convolutional neural mixture models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1872–1881.

[16] Adrià Ruiz and Jakob Verbeek, "Distilled hierarchical neural ensembles with adaptive inference cost," *arXiv preprint arXiv:2003.01474*, 2020.

[17] Luis Guerra, Bohan Zhuang, Ian Reid, and Tom Drummond, "Switchable precision neural networks," *arXiv preprint arXiv:2002.02815*, 2020.

[18] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh, "Improved knowledge distillation via teacher assistant," *arXiv preprint arXiv:1902.03393*, 2019.

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," 2009.

[21] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.