Craft Distillation: Layer-wise Convolutional Neural Network Distillation

Cody Blakeney, Xiaomin Li, Yan Yan, Ziliang Zong*
Computer Science Department, Texas State University, San Marcos, TX, USA
{cjb92, x_l30, tom_yan, ziliang}@txstate.edu

***Abstract*—Convolutional neural networks (CNNs) have achieved tremendous success in solving many challenging computer vision tasks. However, CNNs are extremely demanding for computation capability, memory space, and power capacity. This limits their usage to the cloud and prevents them from being deployed on edge devices with constrained resources and power. To tackle this problem, we propose craft distillation, a novel model compression approach that leverages both depthwise separable convolutions and knowledge distillation to significantly reduce the size of a highly complex model. Craft distillation has three advantages over existing model compression techniques. First, it does not require prior experiences on how to design a good "student model" for effective knowledge distillation. Second, it does not require specialized hardware support (e.g. ASIC or FPGA). Third, it is compatible with existing model compression techniques and can be used with pruning and quantization together to further reduce weight storage and arithmetic operations. Our experimental results show that with proper layer block replacement design and replacement strategy, craft distillation reduces the computational cost of VGG16 by 74.6% compared to the original dense models with negligible influence on accuracy.**

***Index Terms*—convolutional neural network, model distillation, depthwise separable convolution**

I. Introduction

Nowadays, Convolutional Neural Networks (CNNs) have undoubtedly become the most promising method in solving many challenging tasks in computer vision such as image recognition [1], object detection [2], image segmentation [3], and multimodal data analysis [4]. However, deep neural network models are computationally expensive and memory intensive, as well as energy-hungry. Therefore, most of the heavy lifting in today's deep learning paradigm is limited to the cloud, which has powerful GPUs/TPUs and endless power supply. As the edge devices become more ubiquitously used and the issues (e.g. privacy, data transfer, and CO2 emission) of in-cloud AI become increasingly evident, the transition from in-cloud AI to on-device AI will be an emerging trend, as shown in Figure 1.

Nevertheless, deploying deep neural networks on edge devices is a daunting task because on-device AI must reduce power usage and keep the model size small without sacrificing accuracy. Currently, model acceleration and model compression [5] techniques have shown great potential in tackling this challenge. Model acceleration is primarily a hardware-based approach by adding ASIC or FPGA accelerators to devices for efficient and low power deep learning [6]–[9]. Model compression focuses on reducing the size of deep neural networks via software techniques, which include quantization [10], network pruning [11], depthwise separable filters [12], and knowledge distillation [13]. Quantization reduces the memory footprint and computation demand of a model by clustering weights and rounding them off. Pruning reduces redundant parameters of a CNN model by setting unimportant weights to zero or remove part of filters which can reduce the memory footprint and run-time computation needs to some degree. Both quantization and pruning techniques require fine-tuning to regain accuracy. In addition, compressed networks via unstructured pruning need special hardware support to exploit weight sparsity [14]–[16]. Recently, Howard et al. proved that depthwise separable convolutions can substantially reduce computation relative to standard convolutions and used it to create efficient convolutional neural networks in Google's MobileNets [12]. Hinton et al. [13] found that knowledge learned from a large complex "teacher model" (or an ensemble of models) can be distilled to a smaller "student model". The original knowledge distillation approach defines the architecture of the "student model" in advance and trains the entire "student model" during the knowledge distillation process.

In this paper, we propose a different methodology, craft distillation, that takes advantage of both depthwise separable convolutions and a special approach to knowledge distillation. Instead of generating an independent "student model" in advance and transferring knowledge to it, craft distillation transforms the "teacher model" into a smaller model by replacing its standard and costly convolution layers with cheaper depthwise separable convolution layers. Craft distillation has three advantages over existing model compression techniques. First, it does not require prior experiences on how to design a good "student model" for effective knowledge distillation. Second, it does not require specialized hardware support (e.g. ASIC or FPGA). The derived small models can run directly on CPUs or GPUs. Third, it is compatible with existing model compression techniques and can be used with pruning and quantization together to further reduce weight storage and arithmetic operations. In this work, we explore using layer-wise distillation to reduce both FLOPs and memory footprint of convolutional neural networks. We also provide a detailed process of selecting an effective architecture for replacing

*Corresponding author: Ziliang Zong. Email: ziliang@txstate.edu

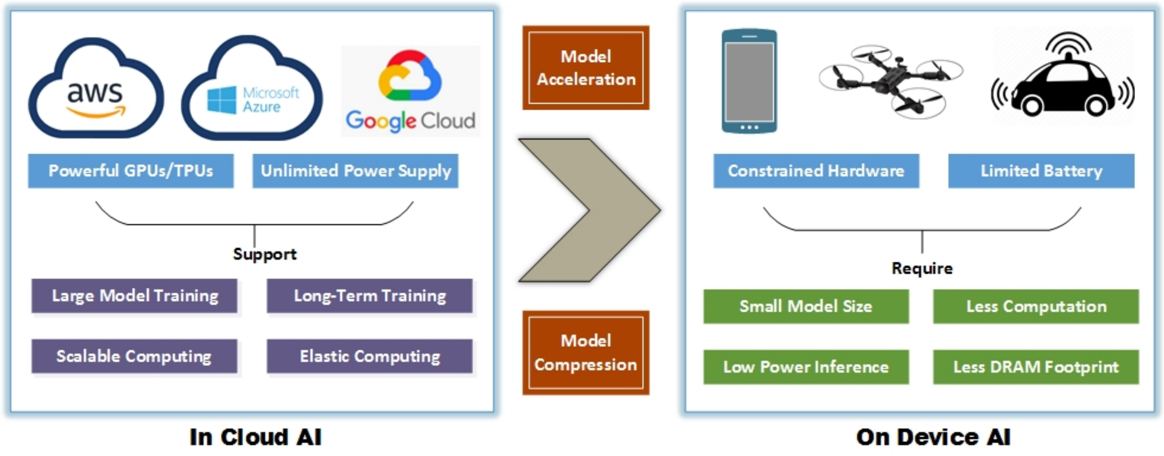


Fig. 1: Difference between in-cloud AI and on-device AI. Deep CNNs running in cloud (e.g. Amazon AWS, Microsoft Azure, and Google Cloud) require powerful GPUs/TPUs and unlimited power supply. Edge devices only have limited battery and constrained resources in computation and memory space. It is essential to accelerate large CNN models or compress them to smaller ones before they can be deployed on edge devices.

standard convolution layers and explain how to derive a small model from the original complex model. Our experimental results show that with proper layer block replacement design and replacement strategy, craft distillation can reduce the computational cost of VGG16 by 74.6% compared to the original dense models with negligible influence on accuracy. This proves that craft distillation is a new viable model compression strategy.

To summarize, the contributions of this paper can be highlighted as follows:

- We propose craft distillation, a novel model compression strategy that transforms complex CNN models to simple models via multiple layer-wise distillations.
- We design a detailed craft distillation process and provide answers to several key design questions.
- We compare the accuracy and compression rate of our method with filter level structured pruning methods and verify the effectiveness of the proposed craft distillation method.

II. Related Work

A. *Neural Network Efficiency*

The power of deep neural networks to solve challenging problems comes at the cost of excessive amount of computation, memory, energy, and CO2 emission. According to [17], the computation cost of deep learning algorithms, from AlexNet in 2012 to recent AlphaGo Zero in 2019, has increased by 300,000x in 6 years. This is not feasible for on-device AI. It is essential for deep neural networks to become smaller and more efficient before they can be deployed on edge devices. Research topics to reduce model size, training/inference time, memory footprint and energy cost have become increasingly popular.

Parameter Pruning. Pruning induces sparsity in weights and activations of neural networks to reduce parameter size and computation time. Optimal Brain Damage [18] and Optimal Brain Surgeon [19] are the earliest works that proposed the idea of network pruning in 1990s. Han et al. [11] pruned networks weights with a small magnitude threshold which brought network pruning back into public attention. Afterwards, pruning has diveraged into two branches, unstructured pruning and structured pruning. The former one increases the sparsity of weights in neural networks and generates fine-grained sparse weight tensors. The drawback is that the required sparse data structures can only accelarate models that are deployed on specialized hardware. For structured pruning, there are vector level, kernel level, filter level and channel level pruning strategies [20]. Distinguished from unstructured pruning, filter level structured pruning directly changes network architectures and it can be implemented on any general purpose computational hardware.

Low-rank Factorization. Low-rank factorization uses matrix/tensor decomposition to estimate the informative parameters which can be applied on both convolutional layers and fully connected layers. Inspired by the idea of dictionary learning, Rigamonti et al. [21] proposed a learning separable 1D filter scheme. Jaderberg et al. [22] achieved double speedup for a single convolutional layer with only 1% drop in classification accuracy in text recognition tasks, when using different tensor convolutional decomposition schemes. Tai et al. [23] introduced a new method that computes the low-rank tensor decomposition by training low-rank constrained CNNs from scratch.

B. *Knowledge Distillation*

Knowledge distillation transfers/distills the knowledge learned from a cumbersome model (a single complex model or

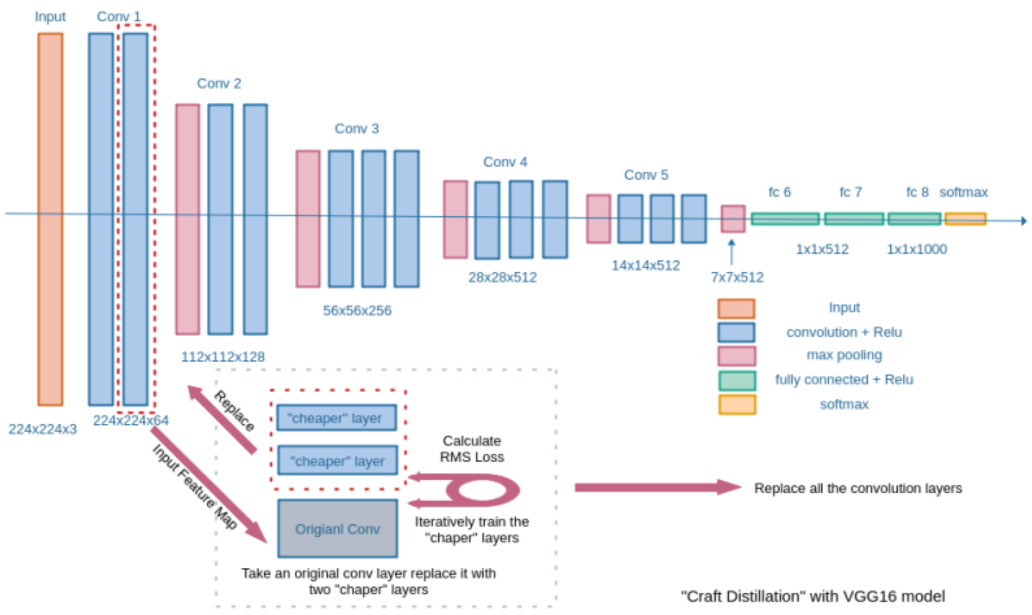


Fig. 2: Overview of the craft distillation process. A convolutional layer of the dense model is selected as the "Teacher" to train cheaper depthwise separable convolution layers. The trained cheaper layers are inserted back to replace the original convolutional layer. Such a layer-wise distillation process is repeated in a top-down manner to replace other convolutional layers in the dense model.

an ensemble of separate models) to a small model which has less amount of computation, smaller memory footprint, and equivalent accuracy with its large counterpart. This method became popular after Hinton et al.'s work [13]. The knowledge is transferred from the larger model to the smaller model by minimizing a loss function where the target is the output of a softmax function (class probabilities) on the large model's logits. Knowledge distillation can also be combined with other model compression techniques. For example, [24]–[26] combined knowledge distillation with quantization and [27], [28] combined it with network pruning. Our craft distillation method is distinct from the original knowledge distillation method because it transforms the complex model on a layer by layer basis using local activations instead of the final layer.

Progressive Blockwise Distillation. Progressive Blockwise Distillation [29] is a method that also leverages local loss combined with classification loss to progressively shrink the size of a student model. However, unlike their methodology which is only a match to the architecture of VGG, our method operates on a per layer basis and can be used to compress any traditional convolution layer, which is applicable to a much larger variety of network types. Additionally, our method only considers the local loss, which can significantly reduce the compute resources required to compress the model.

C. Depthwise Seperable Convolution

Depthwise separable convolutions [12] separate the standard convolution calculation into two layers: the depthwise convolution and the point-wise convolution. Depthwise convolution performs operations using one kernel for each input channel of the input feature map. The output of the depthwise convolutional layer is then used as input for the pointwise convolutional layer. Pointwise convolution operates by applying a 1×1 kernel over all channels of its input feature map. The number of output features are obtained through the number of 1×1 kernels applied over the input feature map. Depthwise separable convolutions have significant computational savings over standard convolutions depending on the size of the kernel. Consider computing convolutions on an input feature map of size $128 \times 128 \times 3$ with a 3×3 kernel to obtain an output with 256 features. Standard convolution would have the computational cost of $109,734,912$ multiplications. However, depthwise separable convolution has the computational cost of $428,652$ multiplications in the depthwise convolutional layer and $12,192,768$ multiplications in the pointwise convolutional layer, totaling to $12,621,420$ multiplications. This has a computational cost of approximately $\frac{1}{9}$ of the standard convolution.

III. Craft Distillation Process

In traditional knowledge distillation [13], a large "teacher model" is used to train a smaller "student model". Figure 2 illustrates the proposed craft distillation process, which is different because it selects a single layer of the original model at a time and teaches a block consisting of one or more depthwise separable layers (or other layer types that can reduce complexity). After selecting a convolution layer to replace, the input feature maps for the layer are used as the training data and act as the teacher. The output feature maps become the labels. The training is treated as a simple regression problem using the Mean Squared Error (MSE) loss. Let us consider the case of replacing layer l in our model. Let f_l be the function that maps an input image to the activations at layer l. Let g be the replacement block for layer l. Our loss function can be represented as $L = \frac{1}{N}\sum_{i=1}^{N} ||f_l(x_i) - g \circ f_{l-1}(x_i)||^2$. When the replacement block has reached convergence, it is inserted into the original model in place of the selected convolutional layer. This process is referred to as layer-wise distillation. Craft distillation performs layer-wise distillation repeatedly to replace several or all convolutional layers in the "teacher model". The large "teacher model" is transformed to a small model when craft distillation is completed.

In order to effectively carry out the craft distillation process described above, several key research questions need to be answered.

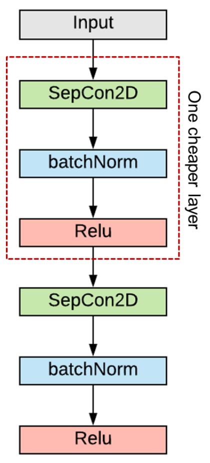
1) How to select the replacement blocks?
2) In what order should layers be replaced?
3) How does craft distillation perform compared to other structure altering compression techniques?

In the following subsections, we layout the experiments that are designed to answer each of these questions. Specifically, subsections A, B, and C answer question 1, 2, and 3 respectively. Each experiment is carefully designed to determine the best strategy when conducting craft distillation as well as evaluate the effectiveness of craft distillation compared to other model compression techniques.

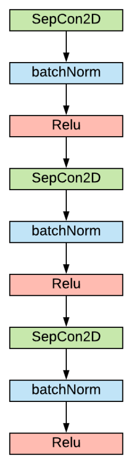
A. Architecture Search

Finding an efficient replacement block for the costly convolution layers is essential to the effectiveness of craft distillation. We first attempt to train a single depthwise separable layer as a replacement for a convolution layer but end up with very poor results. The next natural thought will be stacking several depthwise separable layers, which still uses less parameters and flops than a traditional convolution layer, thanks to the properties of depthwise separable layers. We conduct a series of architecture searches to evaluate the best way to stack depthwise separable layers. The ideal architecture for replacement blocks should keep the added number of parameters as low as possible and be able to minimize the MSE loss.

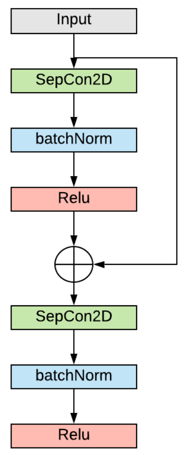
Figure 3 depicts the four candidate architectures explored in our experiments, which include the two depthwise separable layers (a), three depthwise separable layers (b), two depthwise separable layers with skip connections similar to residual blocks (c), and three depthwise separable layers with skip connections (d). We design experiments to evaluate the effectiveness of four candidates as follows. First, we train a VGG16 model on CIFAR-10 with the baseline accuracy of 86.45%. Each candidate architecture aims to replace the second convolution layer in the VGG model. We train each block for 20 epochs on the teacher model's intermediate activations. After that, the layers are inserted into the original model to replace layer 2 and the accuracy of the model is recorded. The weights of the 2nd layer are then fine-tuned using traditional cross entropy loss on the labeled CIFAR-10



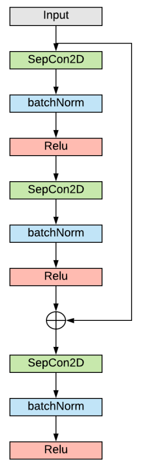
(a) Two Layer Replacement



(b) Three Layer Replacement



(c) Two Layers w/ Skip



(d) Three Layers w/ Skip

Fig. 3: Candidate Replacement Layer Architectures. For the purpose of this paper we refer to the combination of depth wise separable, batch normalization, and relu activation as one layer.

image data. During fine-tuning all other model weights other than the replacement block are frozen.

TABLE I: RMS Loss, Top-1 Accuracy, and Fine Tuning Top-1 Accuracy results of different replacement architectures.

Arch	Loss	Top-1	FT Top-1
Two Layer	3.028E-05	86.40%	87.32%
Three Layer	7.410E-05	85.28%	85.37%
Two Layer w/Skip	3.536E-05	86.28%	86.98%
Three Layer w/Skip	3.877E-05	86.16%	87.03%

Table 1 summarizes the results of the four candidate architectures. While the layers with skip connections do reasonably well, they do not outperform the simple two layer variation. We hypothesize that this might be because these blocks are not deep enough to have significant problems with disappearing gradient. Since the two layers architecture yields the best performance, we use it as the replacement block for convolution layers in the rest of experiments.

B. Order of Layer-wise Distillation

A deep CNN contains many convolution layers. A strategy of in what order to replace them must be decided in the craft distillation process. Obvious choices are either top down or bottom up replacement. To determine the best strategy, we take the pretrained VGG16 model on CIFAR-10 with an original accuracy of 86.45% as the baseline model. Each targeted layer for replacement is trained for 50 epochs on the intermediate layer activation. The layer is then inserted into the model and the loss on the test set is recorded with the replacement blocks. The model is then fine-tuned for 5 epochs with all layers but the replacement block frozen. After the fine-tuning process the weights that resulted in the lowest loss, either from fine-tuning or layer distillation, are kept. After the individual layer training, the resulting model becomes the "teacher model" and the next layer to be replaced was chosen.

TABLE II: Results of Different Replacement Strategies

Model	Loss	Top-1
Original	0.50657	86.45%
Bottom Up	0.76339	81.67%
Top Down	0.61581	86.59%

As can be seen from Table II, the top down strategy results in the highest accuracy. In fact, it is even higher than the original model, which could be the result of the base model not being trained optimally. It is also worth noting that even though the top down strategy yields a higher accuracy than the original model, it also has a higher loss. This may indicate that it is less robust to adversarial attacks and is less certain of its predictions. It could also be that the top down strategy alleviates the overfitting issue of the dataset.

A possible reason for the different accuracies is the fine-tuning. The bottom up strategy gives less or no opportunity to fix mistakes in the layer representations. When training and replacing happens in a top-down manner, if the previous layer reaches some sub-optimal local minimal that shifts the layer representations, the next layer still has a chance to repair it and restore accuracy through fine tuning.

C. Comparison to Structured Pruning

To the best of our knowledge, there is no existing work that compares directly to our proposed craft distillation method. Quantization [10] achieves similar model size reductions in memory, although it retains the same number of parameters and can increase throughput of the calculations by using FP16 or Int8 data types. But it doesn't alter the structure of the original model or the total number of arithmetic operations. Unstructured pruning [11] does modify the model's structure, but requires specialized hardware to run effectively. Structured pruning is probably the fairest comparison to craft distillation because it is performed on a layer by layer basis of a pretrained model and it does not require specialized hardware to work.

TABLE III: Comparison to L1-norm based Filter Pruning [30]. Results are calculated for the CIFAR-10 dataset.

Model	Top-1 (%)	MACCs	Reduced %	Parameters	Reduced %
VGG-16	93.3	3.13 x 10^8		1.5 x 10^7	
ResNet-50	**95.3**			2.38 x 10^7	
VGG-16 [30]	**94.0**	2.06 x 10^8	34	5.4 x 10^6	64
VGG-16 (ours)	92.7	7.93 x 10^7	**74.6**	3.57 x 10^6	**74.4**
ResNet-50 (ours)	94.5			1.49 x 10^7	**37.0**

TABLE IV: Individual layer comparison to L1-norm based Filter Pruning [30].

layer type	w_i x h_i	MACCs	MACC% [30]	MACC% (ours)
Conv_1	32 x 32	1.8E+06	50%	0%
Conv_2	32 x 32	3.8E+07	50%	74.7%
Conv_3	16 x 16	1.9E+07	0%	64.3%
Conv_4	16 x 16	3.8E+07	0%	76.2%
Conv_5	8 x 8	1.9E+07	0%	65.5%
Conv_6	8 x 8	3.8E+07	0%	77.0%
Conv_7	8 x 8	3.8E+07	0%	77.0%
Conv_8	4 x 4	1.9E+07	50%	66.1%
Conv_9	4 x 4	3.8E+07	75%	77.4%
Conv_10	4 x 4	3.8E+07	75%	77.4%
Conv_11	2 x 2	9.4E+06	75%	77.4%
Conv_12	2 x 2	9.4E+06	75%	77.4%
Conv_13	2 x 2	9.4E+06	75%	77.4%
Linear	1	2.6E+05	50%	0%
Linear	1	5.1E+03	0%	0%
Total		3.1E+08	34%	74.6%

The two right most columns indicate the reduction in percentage of MACCs for both methods. Craft distillation significantly reduces MACCs (by 74.6%) even though it does not apply to the fully connected layers. .

Table III and IV illustrate the comparison results of craft distillation to the L1-norm based filter pruning [30] for compressing VGG-16 on the CIFAR-10 dataset and our results on ResNet-50. While the structured pruning method achieves a higher accuracy (94%) than our method (92.7%), we are able to reduce the total number of flops by 74.6%. Table IV shows the detailed comparison at each individual layer. It appears that structured pruning does not prune each layer by the same percentage. In fact, it leaves several layers unpruned entirely. In craft distillation, however, there is no option of what percentage of weights will be removed from a layer. A layer is either completely replaced or it is not replaced at all. We argue that craft distillation is more efficient than structured pruning to compress CNN models with negligible accuracy degradation. The essence of craft distillation is distinct from structured pruning as well.

IV. Conclusions and Future Work

In this paper, we propose craft distillation, a novel method for compressing convolutional neural networks. Craft distillation can transform a complex deep CNN model to a simple model via multiple layer-wise distillations. In each layer-wise distillation, a standard convolution layer is replaced by more efficient depthwise separable layers. We illustrate several key design decisions when conducting craft distillation and show that with the top-down iterative replacement strategy, craft distillation can reduce 74.6% of the computation cost of the

VGG-16 model without compromising accuracy. Compared to the traditional knowledge distillation method [13], craft distillation does not require prior experiences in designing a good "student model" for effective knowledge distillation. In addition, it does not need special hardware support and performs better than structured pruning. Since convolutional layers are the most expensive layers in a CNN, we primarily focus on craft distilling convolutional layers of VGG-16 in this work. The methodology of craft distillation may also be applicable to other layers (e.g. fully connected layers) provided that a more efficient structure can be identified and trained with good accuracy to replace the original layers. It is also worth noting that the effectiveness of draft distillation method may degrade when applying to compressed models like ResNet, MobileNet, and EfficentNet. This is because our method takes advantage of replacing components with depthwise separable layers. If a compressed model already leverages it previously, our method is not able to double dipping the benefit.

In this study, we explore iteratively replacing layers in the top down and bottom up fashion. We plan to investigate and evaluate the third option, which is training them all independently first then making a composite in the future work. Meanwhile, although we confirm that draft distillation method works well on VGG in this work and we believe it can be applied more broadly, future work needs to be done to evaluate its effectiveness on other network models or layers. Since our method uses only local loss and there is no requirement of having labeled data, we plan to further investigate if the draft distillation method can work with unlabeled data and unsupervised learning.

V. Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. The work reported in this paper is supported by the U.S. National Science Foundation under Grant No. CNS-1908658. The opinions expressed in this paper are from the authors and do not reflect the views of the sponsors.

References

[1] 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.

[2] P. Viola, M. Jones *et al.*, "Robust real-time object detection," *International journal of computer vision*, vol. 4, no. 34-47, p. 4, 2001.

[3] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[4] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[5] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[6] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

[7] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[8] L. Song, Y. Wang, Y. Han, X. Zhao, B. Liu, and X. Li, "C-brain: A deep learning accelerator that tames the diversity of cnns through adaptive data-level parallelization," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[9] D. R. J. Jo, S. Cha and I.-C. Park, "Dsip: A scalable inference accelerator for convolutional neural networks,," *IEEE J. Solid-State Circuits,*, vol. 53, no. 2, 2018.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[11] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *NIPS*, 2015.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

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

[14] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 27–40.

[15] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 75–84.

[16] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Press, 2016, p. 20.

[17] D. H. Dario Amodei, "Ai and compute," in *Blog post*, 2018.

[18] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[19] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, 1993, pp. 164–171.

[20] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," *arXiv preprint arXiv:1705.08922*, 2017.

[21] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2754–2761.

[22] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[23] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," *arXiv preprint arXiv:1511.06067*, 2015.

[24] H. Tann, S. Hashemi, R. I. Bahar, and S. Reda, "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[25] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.

[26] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[27] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and fisher pruning," *arXiv preprint arXiv:1801.05787*, 2018.

[28] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2n learning: Network to network compression via policy gradient reinforcement learning," *arXiv preprint arXiv:1709.06030*, 2017.

[29] H. Wang, H. Zhao, X. Li, and X. Tan, "Progressive blockwise knowledge distillation for neural network acceleration." in *IJCAI*, 2018, pp. 2769–2775.

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