

A Unifying Tensor View for Lightweight CNNs

Jason Chun Lok Li*, Rui Lin*, Jiajun Zhou, Edmund Yin Mun Lam, and Ngai Wong

Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

Abstract—Despite the decomposition of convolutional kernels for lightweight CNNs being well studied, existing works that rely on tensor network diagrams or hyperdimensional abstraction lack geometry intuition. This work devises a new perspective by linking a 3D-reshaped kernel tensor to its various slice-wise and rank-1 decompositions, permitting a straightforward connection between various tensor approximations and efficient CNN modules. Specifically, it is discovered that a pointwise-depthwise-pointwise (PDP) configuration constitutes a viable construct for lightweight CNNs. Moreover, a novel link to the latest ShiftNet is established, inspiring a first-ever shift layer pruning strategy which allows nearly 50% compression with < 1% drop in accuracy for ShiftResNet.

Index Terms—Convolutional Neural Network, Tensor Decomposition, Shift Layer, Pruning

I. INTRODUCTION

Lightweight deep neural networks (DNNs) are essential for edge artificial intelligence (AI), where DNNs operate on resource-limited hardware. This necessitates careful consideration of design constraints such as computation, storage, throughput and power consumption. Consequently, numerous efficient convolutional neural network (CNN) architectures have been developed, including prominent examples like MobileNet [1], EfficientNet [2], and ShiftNet [3]. Depthwise separable (DS) convolution [1], [2], [4], which replaces regular convolutional (CONV) kernels, has emerged as an effective solution, as it significantly reduces storage and computation requirements with little or no loss in output accuracy. This paper systematically and analytically demonstrates that these benefits arise from the underlying decomposition and approximation schemes of the reshaped CNN kernel tensor.

Matrix or tensor decomposition is a popular and powerful approach for compressing CNNs [5], [6], [7]. Ref. [5] is among the first to propose utilizing the canonical polyadic decomposition (CPD) for compressing a regular CONV layer whose nature is a 4D kernel tensor. Specifically, the 4D kernel tensor is broken down into two depthwise (DW) layers and two pointwise (PW) layers using CPD. Ref. [7] also employs CPD but treats the CONV layer as a reshaped 3D tensor, leading to a further reduction in the number of parameters. However, it only tests this method with the older-generation AlexNet with few CONV layers, using an ad hoc and non-scalable progressive layer-wise decomposition and fine-tuning scheme. Tucker decomposition is used in [6], which decomposes a CONV layer into two PW layers and a regular CONV layer with smaller input and output channels. Notably, all these

works are related to efficient CNNs [1], [2] containing DS or bottleneck layers. Ref. [8] enumerates various CNN decompositions using tensor network diagrams. However, the high-dimensional and abstract notations obscure the geometry and visual intuition associated with different DS variants. Although previous works have employed tensor network diagrams to represent higher-dimensional tensors and their decompositions into low-rank factors [5], [6], [9], the geometric view is lost, and the underlying arithmetic intuition becomes abstract at 4D and beyond. To address this issue, we demonstrate that it is possible to retain a 3D view and provide highly intuitive interpretations for various tensorized convolutions.

On another front of compacting CNNs, shift operations [3] offer an ideal solution for edge devices due to their zero parameters and FLOPs. We accommodate shift layers in the CPD framework as *one-hot* kernels, leading to a novel channel pruning scheme for shift layers. In fact, while tensors and various efficient CNN implementations are not new, it is *the first time* they are brought under a highly visualizable, unifying tensor umbrella to provide an intuitive illustration of their origin of success. The arithmetic under various decompositions further inspires effective ways to compress the model. The main contributions of this work include (i) A reshaped kernel tensor view that unifies different approximation schemes and naturally spins off various efficient CNN architectures; (ii) The first to link CPD to hardware-efficient shift layers, leading to a *first-of-its-kind* channel pruning scheme for shift layers.

II. KERNEL TENSOR AND ITS APPROXIMATION

To begin with, we instantiate the original 4-way kernel tensor of size $[|c_i|, |c_o|, |k|, |k|]$ and reshape it into a 3-way format of size $[|c_i|, |c_o|, |k^2|]$ (cf. the center of Fig. 1), where c_i, c_o denote the indices of input and output channels, and k represents the width/height of the kernel window, while $|\circ|$ denotes the dimension/cardinality. Next, we enumerate various decompositions from different viewpoints on the 3-way kernel, which in turn generates different efficient CNN architectures. To our knowledge, such unifying visual interpretation in Fig. 1 is presented only for the first time in the literature.

A. Frontal Slices

We assume $[|c_i|, |c_o|, |k^2|] = [3, 6, 9]$ for illustrative purpose. In Fig. 1.I(a), each slice or matrix can be decomposed into a sum of multiple rank-1 terms with decreasing dominance, e.g., by singular value decomposition (SVD). Fig. 1.I(b) depicts a rank-1 approximation to each frontal slice. Here we draw the spatial axis vectors of length $|k^2|$ with rectangular

*JL, and RL contributed equally to this work.

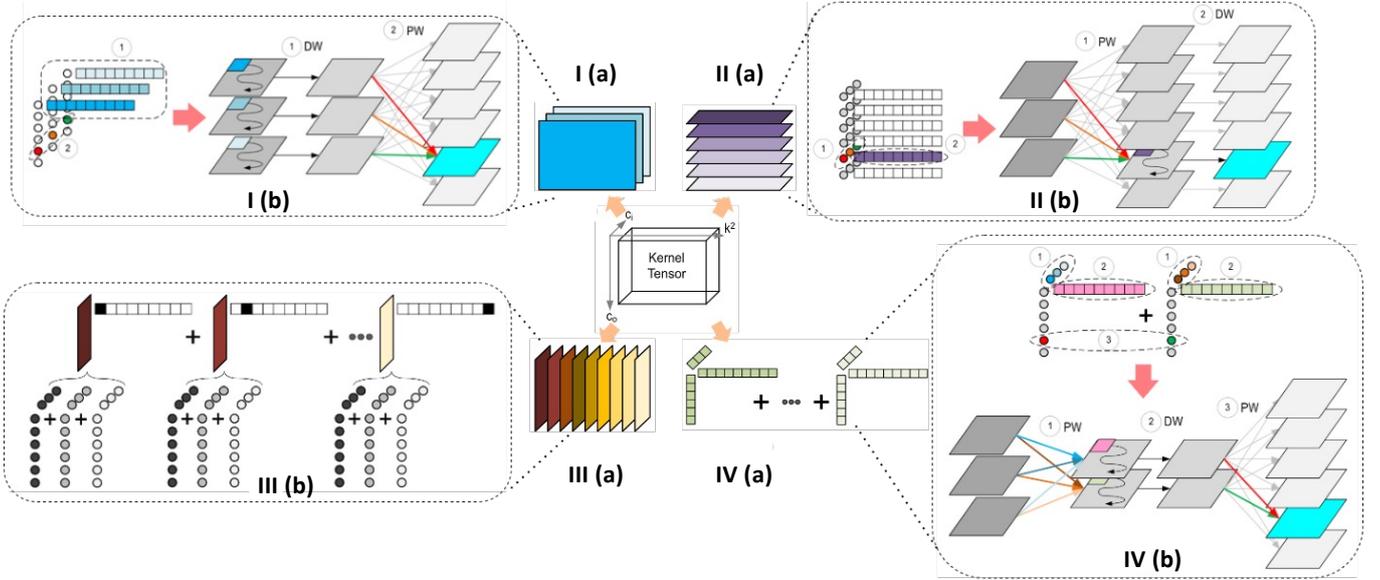


Fig. 1: The reshaped kernel tensor and its different views: **I(a)** frontal slices; **II(a)** horizontal slices; **III(a)** lateral slices; **IV(a)** CPD. We slightly abuse notations and use c_o , c_i and k^2 for both axis and index labels. Assume $[|c_i|, |c_o|, |k^2|] = [3, 6, 9]$. **I(b)** Approximating frontal slices with rank-1 terms translates into the standard DS convolution (viz. DW + PW, abbr. DP). **II(b)** Approximating horizontal slices with rank-1 terms translates into an “inverted” depthwise separable convolution (viz. PW + DW, abbr. PD). **III(b)** Lateral slices positioned with one-hot kernel filters translate into a shift layer (viz. PW + Shift + PW) convolution. **IV(b)** Approximating the kernel tensor with multiple rank-1 CPD terms (CPD rank $r_{cp} = 2$ here) translates into a linear bottleneck (viz. PW + DW + PW, abbr. PDP).

bars to differentiate them from other axes (using balls) to highlight their window sliding nature along the input channels they act on. In Fig. 1.I(b), we fix a particular output channel, e.g., $|c_o| := 6$, for illustration. It can be seen that there is one kernel filter along each c_i index, corresponding to a DW convolution along each input channel. Once the DW convolution is performed, the 3 convolved output slices are produced which are then contracted by the 3 colored balls along the c_i axis, namely, a PW operation to obtain the $|c_o| := 6$ channel. Such sequence of operations cannot be scrambled as the DW kernels are tied with each c_i index, and need to be applied on each input channel first. The number of CNN weight parameters involved is $(|c_o| + |k^2|)|c_i|$ which equals 45 in this example. We abbreviate this CNN replacement by DP (viz. DW+PW) which coincides with the standard DS scheme.

B. Horizontal Slices

Similarly, we study Fig. 1.II(a) with its corresponding rank-1 slices shown in Fig. 1.II(b). As before, we set $|c_o| := 6$, then it can be seen that there are 6 kernels along each c_o index, each corresponding to a DW convolution operating on the PW (1×1) -contracted slices along the c_i axis by the colored balls. Again, such sequence of operations cannot be swapped, since the 6 DW kernels are tied with each c_o index, and need to be applied for producing each output channel. The number of weight parameters in this setting is $(|c_i| + |k^2|)|c_o|$ which equals 72 in our example. We call this PW+DW operation the

PD scheme in this work, which can be seen as an “inverted” DS convolution.

C. Canonical Polyadic Decomposition

We detour to the CPD view in Fig. 1.IV(a) before returning to the lateral slices, which makes a more natural order of illustration. Referring to Fig. 1.IV(b) and borrowing from previous sections, we can segregate the contraction/convolution into three stages, namely: 1) PW along c_i to generate a number of slices equal to the CPD rank r_{cp} (i.e., number of CPD terms), 2) DW along these slices, and 3) PW along the desired c_o . This view corresponds to the celebrated bottleneck layer [1], whose residual variant is depicted in the upper part of Fig. 2. Whether it is a standard or inverted bottleneck is determined by r_{cp} , which decides the center DW block being wider or thinner than its “entrance” and “exit” PW layers. We name this PW+DW+PW combination *PDP*. In this example, the number of weight parameters is $(|c_i| + |c_o| + |k^2|)r_{cp}$ which equals 36. Apparently, this number depends heavily on r_{cp} which we can tune for different tradeoffs between complexity and representation capacity.

D. Lateral Slices

Finally, we turn to Fig. 1.III(a) for the lateral slices. It is clear we can view the original tensor as a slice-wise aggregation by positioning them at the appropriate k^2 -axis index via a one-hot kernel vector (cf. Fig. 1.III(b)). Likewise, each slice can be decomposed into multiple rank-1 terms,

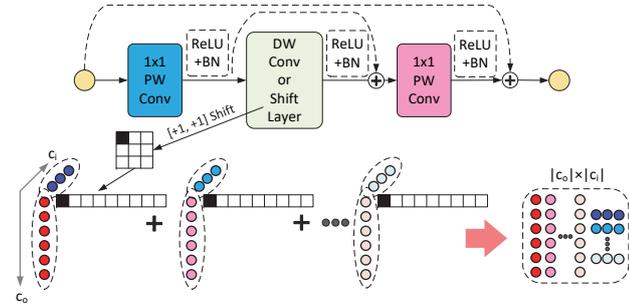


Fig. 2: (Upper) A generic PW+DW/Shift+PW block where the dashed-line shortcuts and nonlinearity blocks are optional. (Lower) Assuming a shift layer, one can collect c_o - c_i mode rank-1 terms of the same shift and sum them for the derivation of principal components.

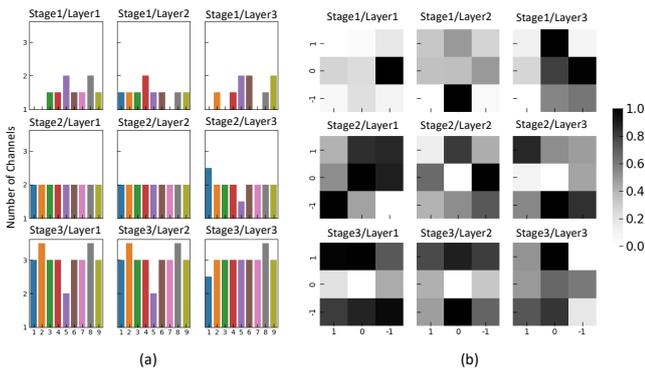


Fig. 3: Shift distribution of ShiftResNet-20 trained on CIFAR-10 after uneven shift pruning with $\phi = 1/16$. There are three stages in ShiftResNet-20, each having three shift layers. (a) The number of channels in each shift group; (b) The relative importance of each shift in various layers based on their singular values (which are summed at each position and normalized within each layer).

all sharing the same one-hot kernel vector. For instance, the leftmost term in Fig. 1.III(b) can spin off into three CPD terms, resembling those in Fig. 1.IV(b). The one-hot nature of the kernel vector effectively performs simple shifting, as advocated in [3]. As such, we draw equivalence of lateral slices to their ShiftNet counterparts. If we use r_{cp} to denote the total number of 3-way rank-1 terms selected, say, according to their importance characterized by their singular values, then the number of weights equals $(|c_i| + |c_o|)r_{cp}$.

III. SHIFT LAYER PRUNING

Based on the novel view of Section II, if a pretrained ShiftNet is available, then one way to do pruning of its shifted channels is to collect rank-1 terms corresponding to the same shift, and then add up the c_o - c_i mode rank-1 terms into a matrix (cf. lower part of Fig. 2). Then SVD is performed on this $|c_o| \times |c_i|$ matrix to decide the number of dominant terms for retention and the number of terms to drop. To the best of our knowledge, this is the first-ever shift layer pruning scheme.

Specifically, under the same shift, the c_o - c_i rank-1 terms are summed up into a $|c_o| \times |c_i|$ matrix for principal component analysis, e.g., via SVD. Then, the most dominant terms are retained as the new PW filters on the c_i and c_o modes. As an illustrative example, suppose for two channels with the same shift, the c_i -mode 1×1 filters are $v, \hat{v} \in \mathbb{R}^{|c_i|}$ whose corresponding c_o -mode 1×1 filters are $u, \hat{u} \in \mathbb{R}^{|c_o|}$, respectively. Then the summed matrix is $\begin{bmatrix} u & \hat{u} \end{bmatrix} \begin{bmatrix} v^T \\ \hat{v}^T \end{bmatrix}$.

Suppose the flattened input channel matrix is X and there exists nonlinearity (say, ReLU) after the first PW layer. Then, if $\hat{v} \approx v$, whether in terms of cosine or Euclidean distance, the mapped channel matrices $\text{ReLU}(v^T X) \approx \text{ReLU}(\hat{v}^T X)$, especially when the feature dimension is high. The same argument also holds when $\hat{u} \approx u$ which, together with $\hat{v} \approx v$, implies the above matrix would be close to rank-1 and its principal c_i and c_o filters can be obtained via SVD and further fine-tuned through backpropagation. Subsequently, when the entrance and exit PW layers have “close” PW filters, then they can be consolidated into fewer principal filters. Note that our pruning approach is different from [10], where shift operations are sparsified through a loss regularizer, while holding the size of the intermediate feature map unchanged. In contrast, we treat the one-hot shift kernel vector as an on/off switch carrying information of its associated c_i - c_o rank-1 term. Once pruned, the entire 3-way rank-1 term is dropped, so the feature map size and representation capacity both reduce.

Here, we define a new hyperparameter dubbed pruning ratio $\phi = \epsilon_{new}/\epsilon_{old}$, where the notion of expansion ratio ϵ is borrowed from [3] which specifies the ratio of bottleneck’s channel size to the output channel size of the exit PW. Obviously, ϕ controls the degree of compression, allowing for a tradeoff between accuracy and model size. There are two strategies to select the principal filters. The first involves comparing singular values within the same shift and preserving the same amount of dominant terms in each group. Concretely, suppose we have a 3×3 DW shift layer, then there are 9 shift groups each corresponding to one direction of shift (including zero shift). This method guarantees an equal number of channels in each shift group. Another approach is to compare singular values across all shift groups and retain the most dominant terms. Fig. 3 shows the shift distribution under “uneven” shift pruning, where missing bars in Fig. 3(a) illustrate that the whole shift group is pruned.

IV. EXPERIMENTS

A. VGG-16

Here approximations with different tensor views discussed in Section II are realized by replacing *every* CONV layer of a VGG-16. A baseline VGG-16 is first trained on CIFAR-10 for 200 epochs. The pretrained CONV kernels are then decomposed into various configurations followed by 300 epochs of fine-tuning. Table I shows the results of using DP and PD schemes to decompose the kernel tensors. The PD configuration has a slightly better performance as compared to DP

TABLE I: VGG-16 with SVD vs randomly initialized DP/PD kernels on CIFAR-10.

Method	CR (%)	#Params(M)	Acc (%)
			Random/SVD
DP	88.53	1.69	92.81 /92.70
PD	88.49	1.69	93.72 /93.54

TABLE II: VGG-16 with CPD vs randomly initialized PDP kernels on CIFAR-10.

Rank r_{cp}	CR (%)	#Params(M)	Acc (%)
			Random/CPD
4	99.65	0.05	51.31/ 62.48
8	99.44	0.08	61.63/ 72.27
16	99.00	0.15	65.06/ 86.09
32	98.12	0.28	75.19/ 89.73

with the same degree of compression. Here the compression is measured by the model-wise compression rate (CR) which will be used throughout. Surprisingly, random initialization leads to a higher test accuracy in both cases as compared to SVD initialization. We believe that the reason is approximation using a single rank-1 term for each tensor slice is insufficient. Table II, on the other hand, highlights the significance of CPD initialization when every CONV layer in the VGG-16 is turned into a PDP configuration comprising r_{cp} DW filters. VGG-16 with CPD-initialized kernels outperforms its random-initialized counterpart by a significant margin, ranging from 11% to 14%. By gradually raising the CPD rank (r_{cp}) from 4 to 32, classification accuracy rises from 62.48% to 89.73% with only a slight decrease in compression ratio (99.65% vs 98.12%). Indeed, such PDP slimming of a dense network has a distinct advantage over DP and PD, which achieves an additional $\approx 6\times$ parametric reduction than the latter two. Utilizing CPD initialization for PDP training, the final accuracy is mostly restored.

B. Shift Layer Pruning

To validate the idea of shift layer pruning as described in Section III, experiments are conducted on CIFAR datasets.

TABLE III: Shift Layer Pruning. Baseline accuracies are 93.37%@CIFAR-10, 71.47%@CIFAR-100.

ϕ	CIFAR-10		CIFAR-100	
	CR (%)	Acc (Even/Ueven) (%)	CR (%)	Acc (Even/Ueven) (%)
$\frac{1}{2}$	49.18	92.81/ 92.88	48.17	70.48/ 70.58
$\frac{1}{4}$	73.78	91.41/ 91.50	72.25	67.16/ 67.62
$\frac{1}{8}$	86.07	89.09/ 89.11	84.30	62.21/ 63.13
$\frac{1}{16}$	92.22	85.57/ 85.84	90.32	57.25 /56.83

ShiftResNets [3] with an expansion ratio $\epsilon = 9$ are employed. A shift module in the ShiftResNet consists of a 3×3 DW shift layer surrounded by two PW layers, interleaved with BN and nonlinearity. By varying ϵ_{new} , we obtain a set of pruning ratios $\phi \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$. ShiftResNet-20 is used as the backbone. Baseline models are first obtained by training on target datasets for 200 epochs, followed by another 200 epochs of fine-tuning. As seen from Table III, shift layer pruning achieves nearly 50% compression with $< 1\%$ accuracy drop for both CIFAR-10 (92.88% vs 93.37%) and CIFAR-100 (70.58% vs 71.47%).

V. CONCLUSION

In this paper, a novel 3D tensor exposition of various CNN kernel approximations is presented (cf. Fig. 1), revealing their close bonds to separable pointwise (PW) and depthwise (DW) convolutions. This perspective further enables the development of a first-ever channel pruning scheme for the zero-flop and hardware-efficient shift layers. We believe such a unifying tensor view can establish itself as a fundamental framework for CNN approximation theory, and serve as an essential basis for discovering new and efficient CNN architectures.

VI. ACKNOWLEDGMENT

This work was supported in part by the General Research Fund (GRF) project 17209721, and in part by the Theme-based Research Scheme (TRS) project T45-701/22-R of the Research Grants Council (RGC), and partially by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

REFERENCES

- [1] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," pp. 4510–4520, 2018.
- [2] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.
- [3] B. Wu, A. Wan, X. Yue, P. H. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. E. Gonzalez, and K. Keutzer, "Shift: A zero flop, zero parameter alternative to spatial convolutions," pp. 9127–9135, 2018.
- [4] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [5] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *International Conference on Learning Representations*, 2015.
- [6] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *International Conference on Learning Representations*, 2016.
- [7] M. Astrid, S.-I. Lee, and B.-S. Seo, "Rank selection of CP-decomposed convolutional layers with variational Bayesian matrix factorization," *International Conference on Advanced Communication Technology (ICACT)*, pp. 347–350, 2018.
- [8] K. Hayashi, T. Yamaguchi, Y. Sugawara, and S. ichi Maeda, "Einconv: Exploring unexplored tensor network decompositions for convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 5552–5562, Curran Associates, Inc., 2019.
- [9] J. Su, J. Li, B. Bhattacharjee, and F. Huang, "Tensorial neural networks: Generalization of neural networks and application to model compression," *arXiv preprint arXiv:1805.10352*, 2018.
- [10] W. Chen, D. Xie, Y. Zhang, and S. Pu, "All you need is a few shifts: Designing efficient convolutional neural networks for image classification," pp. 7234–7243, 2019.