Automatic Selection of Tensor Decomposition for Compressing Convolutional Neural Networks

A Case Study on VGG-type Networks

Chia-Chun Liang, Che-Rung Lee

National Tsing Hua University

May 15, 2021

Outline

1. Introduction
   - Tensor Decompositions for CNN
2. Mixed Tensor Decomposition Algorithm
   - Case Studies on VGG11
   - MTD Algorithm
3. Experiments
4. Concluding Remarks

Introduction

- Convolution Neural Networks (CNN) have revolutionized the field of computer vision and became ubiquitous through a range of applications.
- Among all operations, the most time-consuming task is 2D convolution, taking 50% to 90% of the computation time for various CNN based models.
- Numerous methods for reducing the time of convolution operations have been proposed.
  - Fast convolution, octave convolution, split convolution, ...
  - Pruning methods, channel pruning, structure preserving pruning, ...
  - Low-rank approximations, SVD, tensor decomposition...
- Comparing to other methods, tensor decomposition based methods usually can explore more latent relation among structures, and give a better compression ratio.

Tensor and Tensor Decomposition

- Mathematically, a tensor is a multilinear map between vector spaces.
- A tensor may be represented as a potentially multidimensional array.
  - The *scalar components* of the tensor (or simply its components) are denoted by the indices giving their position in the array.
  - The number of the dimensions of a given tensor is called as *order*.
- We can create subarrays by fixing some of the given tensor's indices.
  - *Fibers* are created when fixing all but one index.
  - *Slices* are created by fixing all but two indices.
- A tensor decomposition is any scheme for expressing a tensor as a sequence of elementary operations acting on other, often simpler tensors.
- Two commonly used tensor decomposition methods are.
  - Canonical Polyadic Decomposition (CPD)
  - Tucker Decomposition (TKD)

Tensors in Convolutional Neural Networks

- A convolution layer in CNNs maps a 3-order input tensor X of size $C_{in} \times W_0 \times H_0$ into a 3-order output tensor Y of size $C_{out} \times W_1 \times H_1$ using a 4-order kernel tensor T of size $C_{out} \times C_{in} \times W_f \times H_f$ to represent the weights:
  - C_{in} is the number of input channels;
  - C_{out} is the number of output channels;
  - W_0 and H_0 are two spatial dimensions for the input data;
  - W_1 and H_1 are for the output data, and
  - W_f and H_f are for the kernel shape in a convolution layer.
- With these notations, a convolution operation can be expressed as

$$\mathsf{Y}(x, y, k) = \sum_{i=1}^{W_f} \sum_{j=1}^{H_f} \sum_{h=1}^{C_{in}} \mathsf{T}(k, j, i, h)\mathsf{X}(i + x - x_c, j + y - y_c, h), \quad (1)$$

  - $x_c = (W_f - 1)/2$ and $y_c = (H_f - 1)/2$.
  - For simplicity, we can assume W_f and H_f are both odd numbers.

Canonical Polyadic Decomposition

- CPD expresses a tensor as a sum of outer products of rank-one tensors. For an order d tensor A of size $n_1 \times ... \times n_d$, a CPD has the following form,

$$\mathrm{A} = \sum_{r=1}^{R} A_r, \tag{2}$$

where the minimal possible R is called the *canonical rank*.

- Each A_r is an outer product of rank-one tensors (ie. vectors) of dimension d,

$$\mathrm{A}_r = A_r^1 \otimes A_r^2 \otimes \cdots A_r^d, \tag{3}$$

where $\otimes$ denotes the Kronecker product operation, and each A_r^i is a rank-one tensor for $i = 1 \cdots d$.

- Element-wise, we can calculate the result by

$$\mathrm{A}(i_1, ..., i_d) = \sum_{r=1}^{R} A_1(i_1, r) A_2(i_2, r) ... A_d(i_d, r). \tag{4}$$

Tucker Decomposition

- The Tucker decomposition uses n-mode product of a tensor, which is a product of a tensor and a matrix.
  - The k-mode product of an d-order tensor A of size $n_1 \times \ldots, n_k, \ldots \times n_d$ and a matrix U of size $j \times n_k$, is another d-order tensor, denoted by A $\times_k U$, of size $n_1 \times \ldots, j, \ldots \times n_d$.
  - Each k-mode fiber in A $\times_k U$ is generated from the vector-matrix product $v^T U$, where v is a k-mode fiber in A.
- Tucker decomposition factorizes a tensor into a core tensor multiplied by a matrix along each mode,

$$\mathrm{A} = G \times_1 A^{(1)} \times_2 A^{(2)} \ldots \times_d A^{(d)}, \tag{5}$$

where $A^{(i)} \in \mathbb{R}^{N_i \times M_i}$ are the factor matrices.

  - The upper script in $A^{(i)}$ denotes the matrix in the ith-mode.
  - The tensor $G \in \mathbb{R}^{M_1 \times \cdots \times M_d}$ is called the core tensor, which has the same order as A.

Mixed Tensor Decomposition (MTD)

- Given a CNN model $\mathcal{M} = (\{\mathrm{T}_i, \{S_j\}\})$, where $\{\mathrm{T}_i\}$ is a series of tensors formed from convolution layers, and $\{S_j\}$ is the set of other weight and parameters.
  - Let $|\mathcal{M}|$ be the size of model, and $f(\mathcal{M}, X, L)$ be the model accuracy of $\mathcal{M}$ for input data X and their corresponding labels L.
- The goal is to minimize the size of a compressed model $\mathcal{M}' = (\{\mathrm{T}'_i, \{S_j\}\})$, where T'_i is obtained from T_i by some transformations $D_i = \{\mathrm{TKD}, \mathrm{CPD}, \mathrm{Identity}\}$.

$$\min_{D_i} |\mathcal{M}'| \tag{6}$$
$$\text{s.t.} f(\mathcal{M}, X, L) - f(\mathcal{M}', X, L) \leq \epsilon,$$

where ϵ is a small number for the tolerance of accuracy drop, which means the model accuracy of $\mathcal{M}'$ should be maintained.

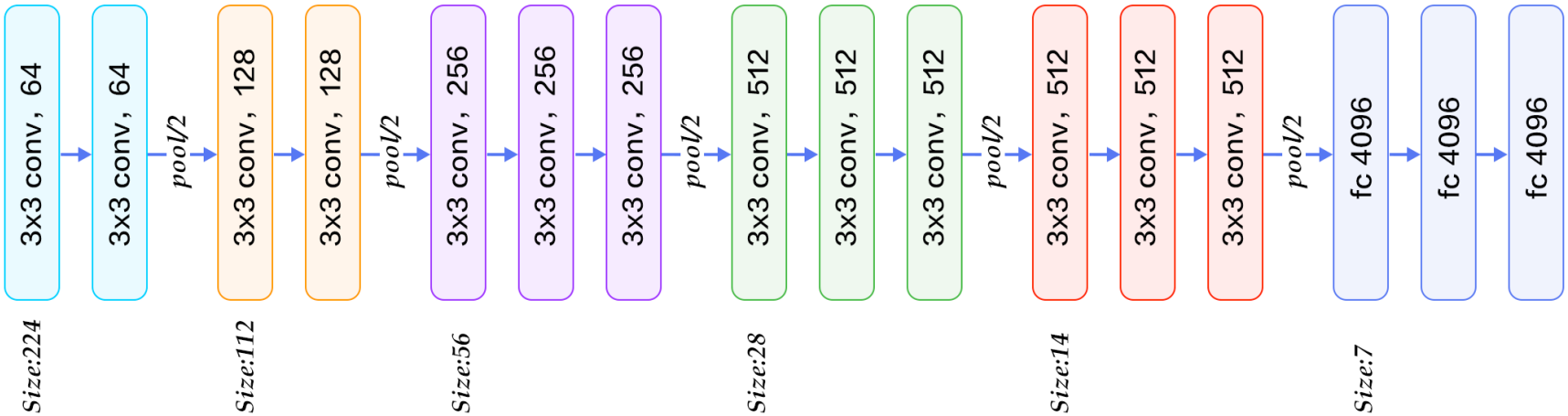
The Computational Challenges of MTD

- Each layer can be decomposed by one of three possible methods. If there are N layers, the possible combination is 3^N.
- For Tucker decomposition, the rank can be determined by the Variational Bayesian Matrix Factorization (VBMF) method. But for CPD, there is no good algorithm to determine a proper rank for decomposed network.

- The auto-tuning method can be used to solve this problem.
  - For each layer, decide which decomposition method should be used, and decide the proper rank automatically.
  - We first performed a case study on the VGG typed network, and based on the case study we proposed an automatic strategy.

VGG Network

- VGG network is invented by the Visual Geometry Group in Oxford University. This architecture is the runner up of ILSVR2014 in the classification task while the winner is GoogLeNet.
- VGG uses a lot of 3x3 convolution layers in the network, and partitions them into blocks using pooling.
- A typical VGG-16 network architecture. [1]



[1] Figure is from http://deanhan.com/2018/07/26/vgg16/

Case Study on VGG11

- We benchmarked VGG11, which has eight convolution layers in five blocks, indexed by block ID and layer ID.
  - The used dataset is CIFAR-10.
  - All networks are implemented with *PyTorch*.
  - The tensor decomposition is performed using the library *Tensorly*.
  - The rank of TKD is decided by VBMF; the rank of CPD is decided by keeping its compression ratio (CR) similar to TKD's.
  - The baseline is the uncompressed VGG11, whose accuracy is 88.9%.
- We performed the following studies.
  - Decompose each single layer of VGG11 using CPD or TKD.
  - Decompose all layers of VGG11 using CPD or TKD in the sequential order (forward) and the reverse order (backward).
  - Evaluate the accuracy and compression ratio (CR).

Decompose Single Layer

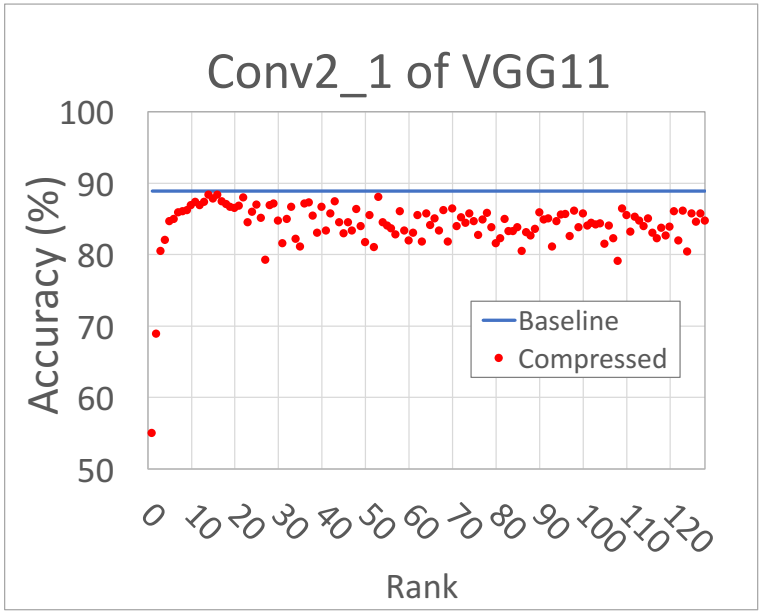
The compression ratio and model accuracy for VGG11.

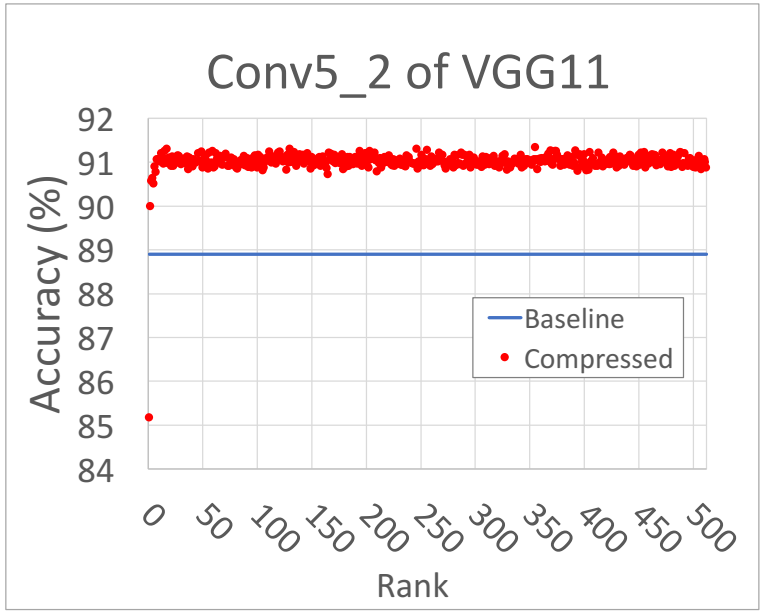
Conv. layer	Ipt chnl	Opt chnl	Tucker			CPD		
			R	CR	Acc	R	CR	Acc
1_1	3	64	(2, 12)	1.75	90.4%	9	1.59	91.1%
2_1	64	128	(32, 43)	3.70	90.9%	44	3.67	88.8%
3_1	128	256	(54, 59)	5.82	90.8%	65	5.86	89.4%
3_2	256	256	(61, 50)	10.56	90.4%	63	10.52	88.8%
4_1	256	512	(90, 103)	7.41	91.0%	112	7.31	88.5%
4_2	512	512	(123, 126)	8.84	91.1%	143	8.77	90.8%
5_1	512	512	(75, 75)	18.52	91.4%	84	18.38	91.3%
5_2	512	512	(61, 65)	23.55	91.3%	69	23.78	91.5%

$$\text{CR} = \frac{\text{No. parameters in the original layer}}{\text{No. parameters in the compressed layer}}, \text{ R is the rank.}$$

Rank of CPD

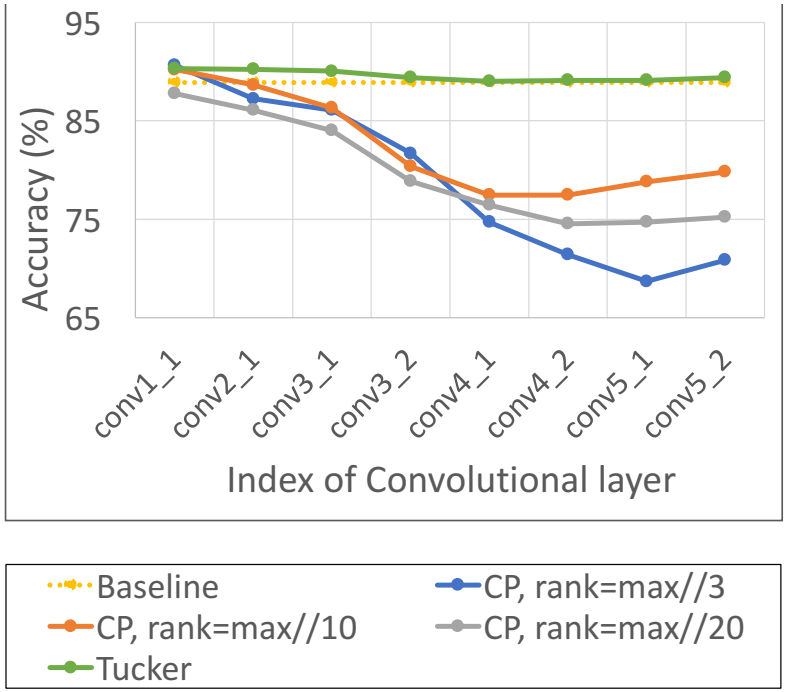
- We also studied the influence of CPD's rank to the model accuracy.
- The results show that in the earlier layer, conv2_1, no matter what rank is, the accuracy cannot be fully restored.
- On the other hand, for later layer, conv5_2, even a small rank can fully restore the model accuracy.
- The results suggest the later layers have more redundancy, and using CPD with lower ranks does not hurt the model accuracy.



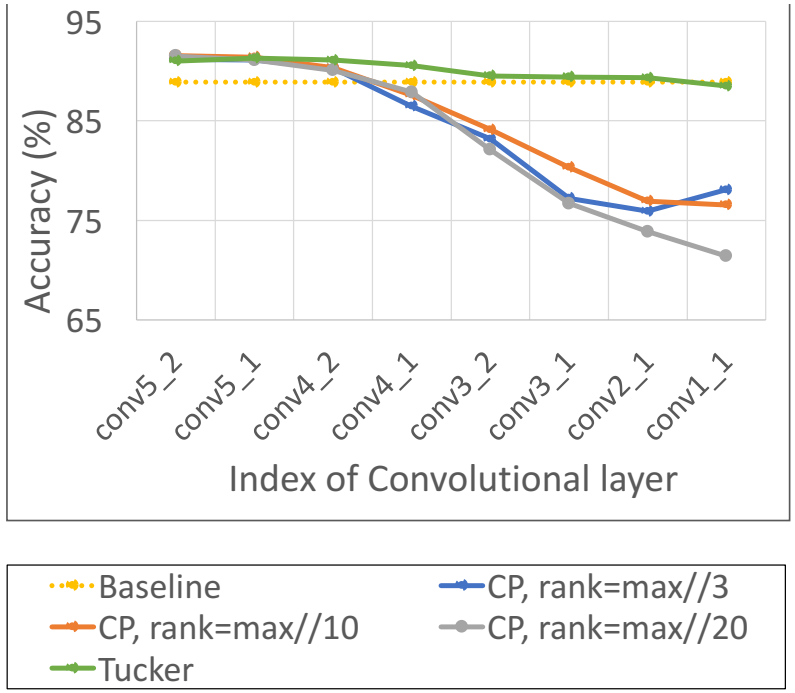


Decompose Multiple Layers

	TKD		CPD					
Rank	VBMF		max/3		max/10		max/20	
order	seq.	rev.	seq.	rev.	seq.	rev.	seq.	rev.
Acc	89%	88%	78%	78%	80%	77%	75%	71.4%
CR	11.84 ×	12.04 ×	6.01 ×		31.57 ×		73.68 ×	



(a) Sequential order



(b) Reverse order

Lessons Learned from the Case Study

1. For a layer, if both TKD and CPD can keep similar model accuracy, using CPD for compression, because it can give better compression ratio.
2. If CPD cannot maintain the model accuracy, using TKD; if TKD also fails, using Identity (no compression).
3. Compressing the model in the reverse order. Although the results show that sequential order gives better model accuracy, the plots in the previous page indicate the accuracy of CPD remains high in the first few layers in the reverse order. On the other hand, in the sequential order, the model accuracy of CPD drops quickly from the beginning. Thus, if CPD is tried first, one should use the reverse order.

Mixed Tensor Decomposition Algorithm

Let M be the model, and f be the original model accuracy. The high level description of the algorithm

1. Decompose the network layer by layer in the reverse order.
2. For each layer L_i, perform TKD first. Assume the decomposed layer is L_i^T and the model accuracy after TKD is ${f_i}^T$.
3. If $f - {f_i}^T > \epsilon$, this means even TKD cannot keep the desired accuracy, so we terminate the compression. This means the rest of the layers are un-compressed.
4. For the same layer L_i, perform CPD, and get the decomposed layer L_i^C and model accuracy ${f_i}^C$.
5. If $f - {f_i}^C > \epsilon$ or $|L_i^C| > |L_i^T|$, this means CPD cannot achieve the desired accuracy or the model size compressed by CPD is larger than that of TKD. In that case, we use TKD for layer L_i.
6. Continue step 2-step 5 until all the layers are complete.

Rank Selection and Fine Tuning

- The rank of TKD is determined by VBMF. [2]
- For CPD, numerical rank selection is an NP-hard problem. In addition, high numerical rank does not imply high model accuracy. We used the result of TKD and an empirical method: Iterative Two-Pass Decomposition. [3] to decide the rank of CPD.
- The fine-tuning process follows the suggestions given by Astrid and Lee [4], which gradually transforms a dense network into the decomposed one layer by layer.
- We found that the network is retrained after the decomposition of each layer, rather than the decomposition of the entire network, the accuracy drops would be negligible.

---

[2] Nakajima, S., Sugiyama, M., Babacan, S.D., Tomioka, R.: Global ana-lytic solution of fully-observed variational bayesian matrix factorization.J. Mach. Learn. Res.14(2013)

[3] Lin, W.S., Wu, H.N., Huang, C.T.: Accelerating Convolutional Neural Networks using Iterative Two-Pass Decomposition.

[4] Astrid, M., Lee, S.I., Seo, B.S.: Rank selection of CP-decomposed con-volutional layers with variational bayesian matrix factorization. In: 2018 IEEE ICACT 34

Experiments

- Our experiments run on a system with an Intel i7-4790 CPU and a NVIDIA GeForce GTX1080 Ti GPU.
- All networks are implemented with *PyTorch*
- The tensor decomposition is performed using the library *Tensorly*.
- We used the widely-used *VGG-11* and *VGG-16*, which have 8 and 13 convolution layers respectively.
- The used dataset is *CIFAR-10*, for which VGG11 can achieve 88.9% accuracy and VGG16 can reach 90.63% accuracy.
- We compared MTD with other four tensor decomposition methods: TKD, CPD, iterative two pass method, and MUSCO.[5]

[5] Gusak, J., Kholyavchenko, M., Ponomarev, E., Markeeva, L.,Blagoveschensky, P., Cichocki, A., Oseledets, I.V.: Automated Multi-stage Compression of Neural Networks. In: 2019 IEEE/CVF International Conference on Computer Vision Workshops

Compared Results for VGG11

VGG11	CR	CRFS	Accuracy	ITR (C)	ITR (G)
Baseline	-	-	88.90%	-	-
Tucker	11.82 ×	1.43 ×	89.41%(+0.07%)	2.24 ×	1.41 ×
CPD	11.76 ×	1.43 ×	84.64%(-4.26%)	1.75 ×	1.50 ×
Two pass	1.84 ×	1.38 ×	89.21% (+0.31%)	0.97 ×	1.34 ×
MUSCO	23.55 ×	1.46 ×	88.33%(-0.57%)	2.28 ×	1.54 ×
MTD	32.59 ×	1.47 ×	88.06%(-0.84%)	1.99×	1.56 ×

$$\text{CR} = \frac{\text{No. parameters in all convolution layers}}{\text{No. parameters in all compressed convolution layers}}$$

$$\text{CRFS} = \frac{\text{File size for the original model}}{\text{File size for the compressed model}}$$

$$\text{ITR} = \frac{\text{Inference time of the original model}}{\text{Inference time of the compressed model}}$$

(C) for CPU and (G) for GPU

Compared Results for VGG16

VGG16	CR	CRFS	Accuracy	ITR (C)	ITR (G)
Baseline	-	-	90.63%	-	-
Tucker	13.53 ×	1.68 ×	91.02%(+0.39%)	2.21 ×	1.53 ×
CPD	13.48 ×	1.68 ×	87.02%(-3.61%)	1.38 ×	1.50 ×
Two pass	2.02 ×	1.59 ×	89.57%(-1.06%)	1.69 ×	1.36 ×
MUSCO	14.05 ×	1.69 ×	87.07%(-3.56%)	2.05 ×	1.46 ×
MTD	37.25 ×	1.76×	89.80%(-0.83%)	2.19 ×	1.64 ×

$$\text{CR} = \frac{\text{No. parameters in all convolution layers}}{\text{No. parameters in all compressed convolution layers}}$$

$$\text{CRFS} = \frac{\text{File size for the original model}}{\text{File size for the compressed model}}$$

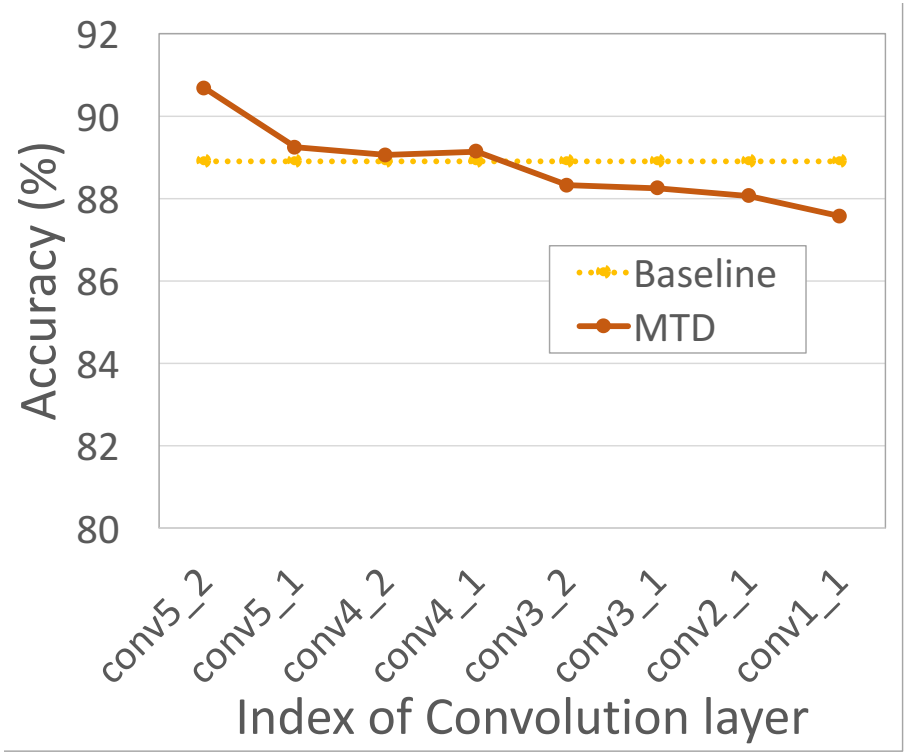
$$\text{ITR} = \frac{\text{Inference time of the original model}}{\text{Inference time of the compressed model}}$$

(C) for CPU and (G) for GPU

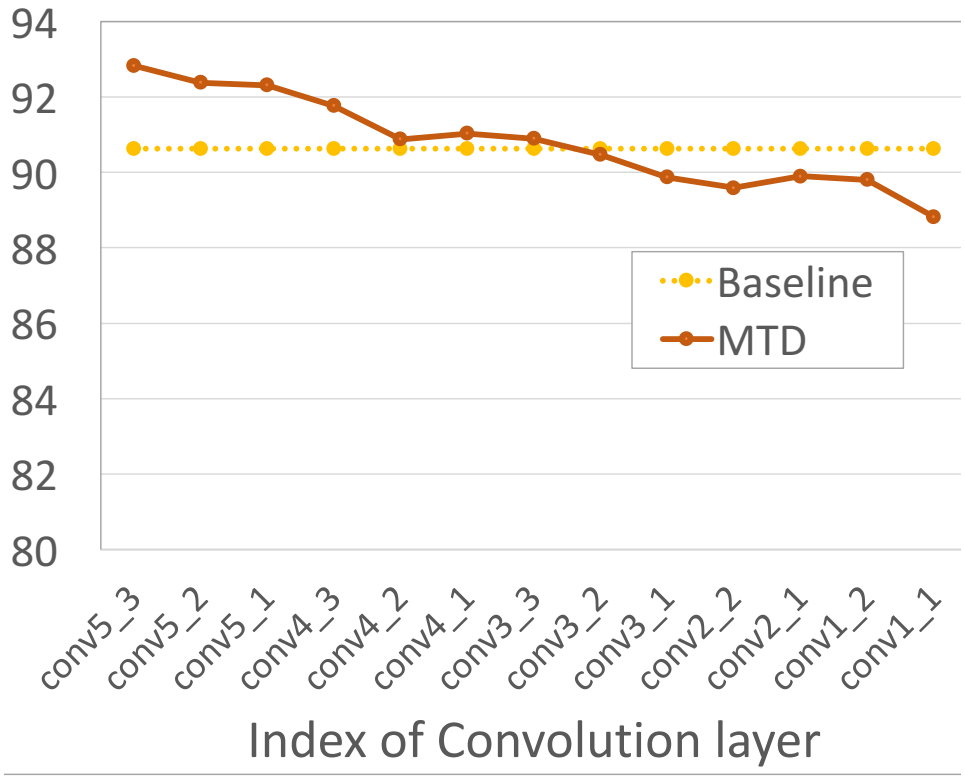
Selected Methods for VGG11 and VGG16

VGG11				VGG16			
layer.	Ipt	Opt	method	Conv.	Ipt	Opt	method
conv1_1	3	64	ID	conv1_1	3	64	ID
				conv1_2	64	64	TKD
conv2_1	64	128	TKD	conv2_1	64	128	TKD
				conv2_2	128	128	TKD
conv3_1	128	256	TKD	conv3_1	128	256	TKD
conv3_2	256	256	TKD	conv3_2	256	256	TKD
				conv3_3	256	256	TKD
conv4_1	256	512	TKD	conv4_1	256	512	TKD
conv4_2	512	512	CPD	conv4_2	512	512	CPD
				conv4_3	512	512	CPD
conv5_1	512	512	CPD	conv5_1	512	512	CPD
conv5_2	512	512	CPD	conv5_2	512	512	CPD
				conv5_3	512	512	CPD

Accuracy of VGG11 and VGG16



(a) VGG11



(b) VGG16

Concluding Remarks

- We used the auto-tuning technique to developed a mixed tensor decomposition algorithm (MTD) which selects the most suitable decomposition method for each layer.
  - We studied the properties of TKD and CPD applying to the compression of convolutional neural networks.
  - Based on the observation, we can design an automatic strategy to select the decomposition method for each layer.

- Experimental results show that MTD can achieve a great compression ratio without scarifying the accuracy.
  - For VGG11, the compression ratio is $32\times$, and for VGG16, it is $37\times$.
  - The accuracy drops for both model are less than 1%.
  - Comparing to existing methods, the compression ratio of MTD is more than two times better than that of other methods.

Future Work

- The proposed method is orthogonal to many other methods, such as MUSCO, so that they can be applied together.
- The current work is only verified on VGG-based networks. We believe our method or framework can be extended for more complex network models, such as ResNet or Inception, or even for the models of other tasks other than the convolution operations.
- There are other types of tensor decompositions, such as tensor train factorization or block decomposition methods. How to integrate them into the model compression is still a challenge.
- There are many fundamental problems for tensor decompositions, such as rank selection and model retraining, which require more investigation to solve.