Learning Tucker Compression for Deep CNN

Pengyi HAO*, Xiaojuan LI*, and Fuli WU†

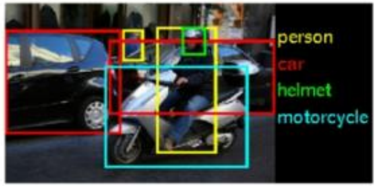
Zhejiang University of Technology, Hangzhou, China

haopy@zjut.edu.cn,2112012075@zjut.edu.cn,fuliwu@zjut.edu.cn

Why Model Compression?

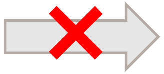
Object tracking

Image classification



Object detection

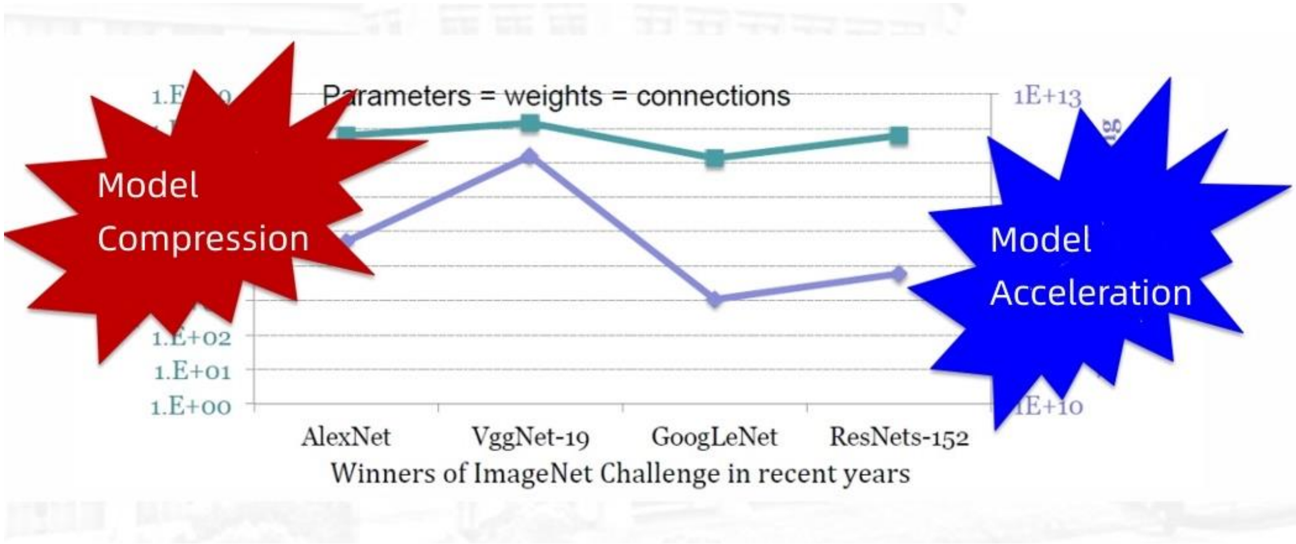
•••



Resource-limited Devices

Limited memory space, limited computing power,etc.

Deep CNN’s Challenge



- Large amount of parameters
- high computational cost

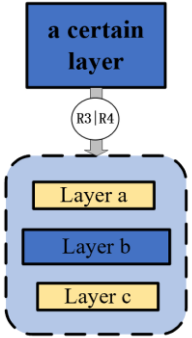
2022 DCC

What's tensor decomposition?

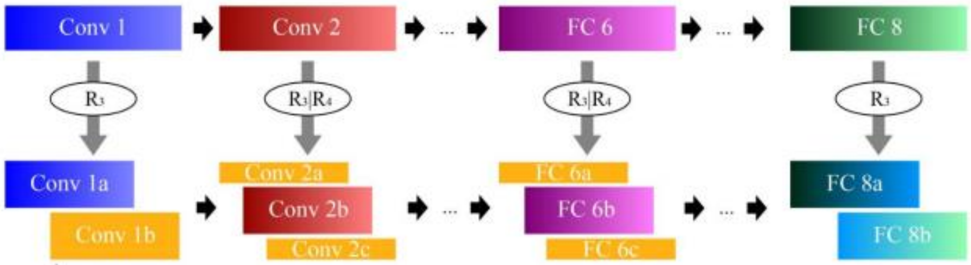
Tensor decomposition approximately decomposes the high-order tensors of CNN's layers into several low-dimensional tensors.

★Tensor Decompsition can be divided into **two categories**

Single layer decomposition



Global decomposition



Existing Problems

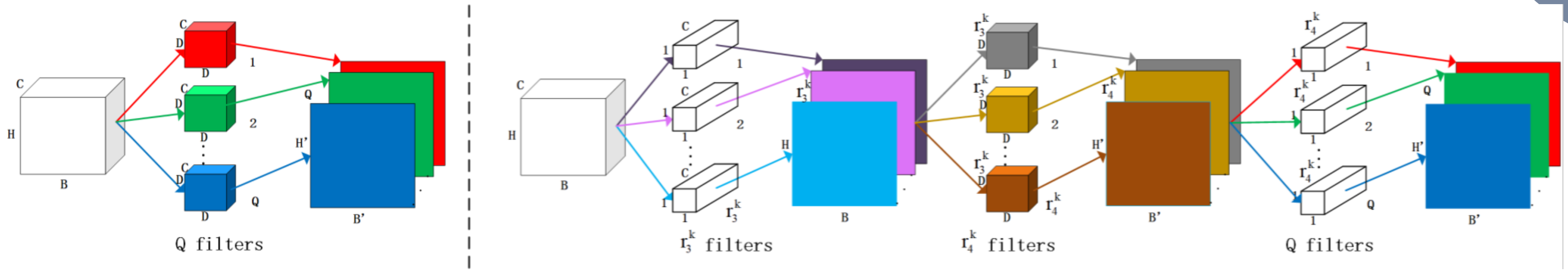
I. Decompose CNN layer by layer, **ignoring the correlation between layers**.

II. **Training and compressing a CNN is separated**

Rank Selection

2022 DCC

Tucker Decomposition



Original convolution

Tucker-Decomposed Group

$$\mathcal{Y}_{h',w',q} = \sum_{d=1}^{D}\sum_{d=1}^{D}\sum_{c=1}^{C} W_{d,d,c,q}\,\mathcal{X}_{h_i,w_j,c}$$

Tucker-2

$$W_{ddcq} \approx \sum_{r_3^k=1}^{R_3^k}\sum_{r_4^k=1}^{R_4^k} \mathcal{G}_{ddr_3^k r_4^k} A^{(3)}_{c,\boxed{r_3^k}} A^{(4)}_{q\boxed{r_4^k}}$$

$$Z_{h,b,r_3^k} = \sum_{c=1}^{C} A^{(3)}_{(c,r_3^k)} \mathcal{X}_{h,b,c}$$

$$Z'_{h',b',r_4^k} = \sum_{d=1}^{D}\sum_{d=1}^{D}\sum_{r_3^k=1}^{R_3^k} \mathcal{G}_{d,d,r_3^k,r_4^k} Z_{h_d,b_d,r_3^k}$$

$$\mathcal{Y}_{h',b',q} = \sum_{r_4^k=1}^{R_4^k} A^{(4)}_{(q,r_4^k)} Z'_{h',b',r_4^k}$$

Preserve spatial features and reduce channel dimension redundancy

Problem formulation

Jointly optimizing of CNN's loss function and Tucker's cost function (training and compressing is carried out at the same time)

$$\min_{W} L(W) + \lambda C(W) \quad \text{s.t.} \quad \operatorname{rank}(W_k) = r_n^k \leq R_n^k, k = 1, 2, \ldots, \mathrm{K}$$

- $L(W)$: loss function (such as cross entropy for classification)
- $C(W)$: the linear cost of tucker compression (depend on tucker rank)
- $\lambda \geq 0$: determines the distribution of tucker rank

$\lambda \geq 0$ ⇨ tucker rank ⇨ $C(W)$

Penalty form for Tucker Cost

2022 DCC

$$C(W) = C(W_1) + C(W_2) + \ldots + C(W_K)$$

Tucker-2

$$C\left(W_k\right) \ldots + C\left(r_4^k\right) = \alpha_3^k r_3^k + \alpha_4^k r_4^k$$

$$C(W) = \mathrm{C}\left(r_3, r_4\right) = C\left(\left[r_3^1, r_4^1\right]\right) + \cdots + C\left(\left[r_3^K, r_4^K\right]\right)$$

$$r_3 = \left[r_3^1, \ldots, r_3^K\right],\ r_4 = \left[r_4^1, \ldots, r_4^K\right]$$

Optimization (How to select tucker rank)

Introducing tucker approximate tensor: $\Theta = (\Theta_1, \ldots, \Theta_K)$:

$$\min_{W,\Theta,r_3,r_4} L(W) + \lambda C(r_3, r_4) \quad s.t. \quad \operatorname{rank}(\Theta_k) = r_3^k, r_4^k \leq R_3^k, R_4^k, \quad k = 1, \ldots, K$$

quadratic penalty method + augmented Lagrangian method

$$Q(W, \Theta, r_3, r_4; \mu) = \boxed{L(W)} + \lambda C(r_3, r_4) + \boxed{\tfrac{\mu}{2} \sum_{k=1}^{K} \left\| W_k - \Theta_k - \tfrac{1}{\mu} \varphi_k \right\|^2}$$
$$+ \tfrac{\mu}{2} \sum_{k=1}^{K} \left\| X_k^{(3)} - M_k^{(3)} - \tfrac{1}{\mu} \varphi_k \right\|^2 + \tfrac{\mu}{2} \sum_{k=1}^{K} \left\| X_k^{(4)} - M_k^{(4)} - \tfrac{1}{\mu} \varphi_k \right\|^2$$
$$\text{s.t.} \quad \operatorname{rank}(\boldsymbol{\Theta}_k) = r_3^k, r_4^k \leq R_3^k, R_4^k, k = 1, \ldots, K$$

optimize {W, Θ, r3, r4} jointly

$$\min_{W} L(W) + \frac{\mu}{2}\sum_{k=1}^{K}\left\| W_k - \Theta_k - \frac{1}{\mu}\varphi_k \right\|^2$$

$$\|W - \Theta\|^2$$

$$\lambda C\left(r_3^k\right) + \frac{\mu}{2}\sum_{k=1}^{K}\left\| X_k^{(3)} - M_k^{(3)} - \frac{1}{\mu}\varphi_k^{(3)} \right\|^2 \quad \text{s.t.} \quad \operatorname{rank}\left(X_k^{(3)}\right) = r_3^k \leq R_3^k$$

$$\lambda C\left(r_4^k\right) + \frac{\mu}{2}\sum_{k=1}^{K}\left\| X_k^{(4)} - M_k^{(4)} - \frac{1}{\mu}\varphi_k^{(4)} \right\|^2 \quad \text{s.t.} \quad \operatorname{rank}\left(X_k^{(4)}\right) = r_4^k \leq R_4^k$$

SVD

$$X_k^{(3)} = U_k S_k V_k^T \longrightarrow \min_{r_3} \lambda C\left(r_3^k\right) + \frac{\mu}{2}\sum_{i=r_3^k+1}^{R_3^k} s_{ki}^2 \quad \text{s.t.} \quad \boxed{r_3^k} \in \{0, 1, \ldots, R_3^k\}$$

2022 DCC

Algorithm 1 Learning Tucker Compression Algorithm.

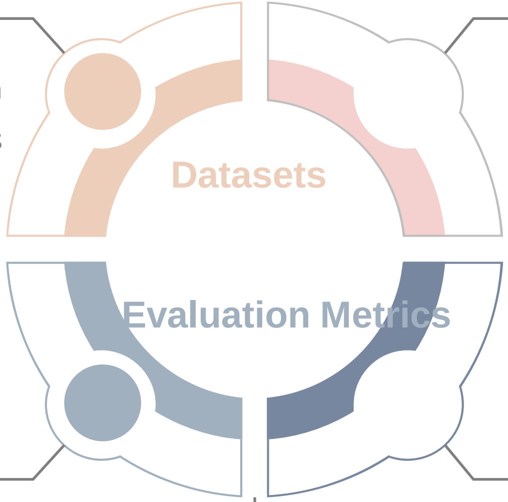
Input: K-layer CNN with weight W; layer-wise cost function C; hyperparameter λ

Initialization: r_3, r_4, $\mathbf{\Theta}$, $\mathcal{G}$, $A^{(3)}$, $A^{(4)}$, φ
for $\mu = \mu_0 < \mu_1 < \cdots < \infty$ **do**
    if $\|W - \Theta\|$ is not small enough: **then**
        get $\mathbf{W}$ from $Eq.(7)$
        for k = $1, 2 \cdots$ K **do**
            get $X_k^{(3)}$, $X_k^{(4)}$ based on Unfolding-Matricization of W_k
            get r_3^k, r_4^k from $Eq.(8), Eq.(9)$
            $A_k^{(3)}$, $A_k^{(4)}$ $= r_3^k, r_4^k$ leading left singular vector of $X_k^{(3)}, X_k^{(4)}$
            $\mathcal{G}_k = \mathbf{W}_k \times {}_3 A_k^{(3)} \times {}_4 A_k^{(4)}$
            get $\mathbf{\Theta}_k$ from $\mathcal{G}_k, A_k^{(3)}, A_k^{(4)}$
        end for
        $\varphi = \varphi - \mu(\mathbf{W} - \mathbf{\Theta})$
    end if
end for
Using global optimal r_3, r_4 to decompose W based on $Eq.(11)$

Output: the decomposed K-layer CNN

Datasets & Evaluation Metrics

2022 DCC



Chest X-Ray

Pneumonia classification published on Kaggle. There are 5,856 X-Ray images labelled as pneumonia or normal.

CIFAR10

natural images consists of 60,000 color images with size of 32 × 32.

FLOPs , SR

Speedup Ratio of FLOPs (SR) :

$$SR = \frac{\sum_{k=1}^{K} (D^2CQHB)_k}{\sum_{k=1}^{K} (CR_3HB + D^2R_3R_4H'B' + QR_4H'B')_k}$$

Parameter , CR

compression ratio of Parameters(CR):

$$CR = \frac{\sum_{k=1}^{K} (\mathrm{D}^2CQ)_k}{\sum_{k=1}^{K} (CR3 + D^2R3R4 + QR4)_k}$$

Top-1 accuracy

The Selection of CNNs for Datasets

Comparison between ResNet-34 and VGG-16 compressed by LTC

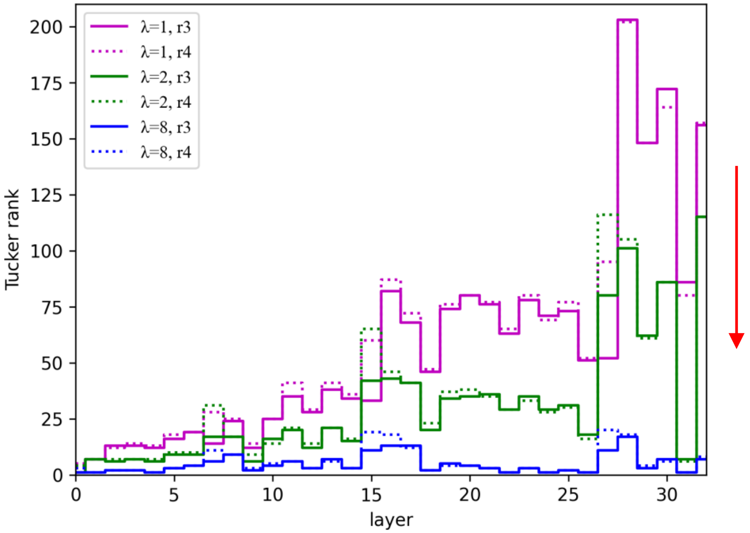
Dataset	Model	FLOPs(G)	SR	Params(M)	Acc (%)
Chest X-Ray	ResNet-34	3.68	1.00	21.80	96.79
	LTC	0.07	51.10	0.80	95.51
	VGG-16	15.65	1.00	134.28	97.00
	LTC	0.36	10.22	119.60	93.75
CIFAR10	ResNet-34	0.29	1.00	21.80	91.90
	LTC	0.01	22.38	1.21	89.13
	VGG-16	0.35	1.00	33.65	93.60
	LTC	0.06	5.61	19.62	91.89

√

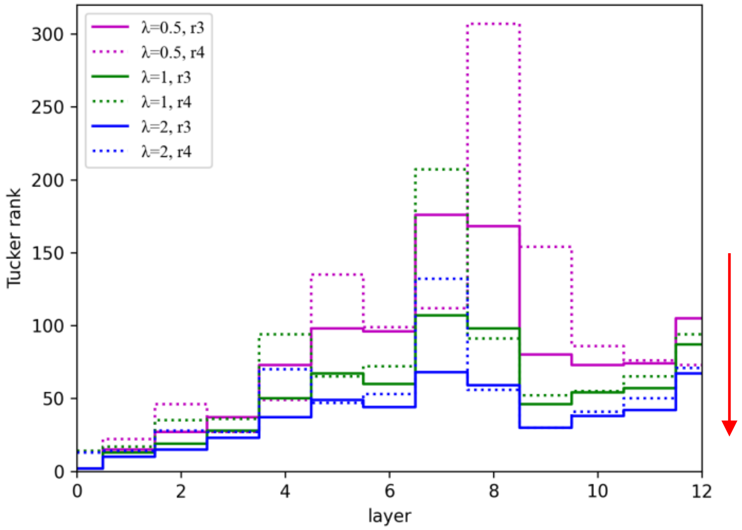
√

The hyperparameter λ

Tucker rank distribution of compressed networks with different λ :



(a) ResNet-34 on Chest X-Ray

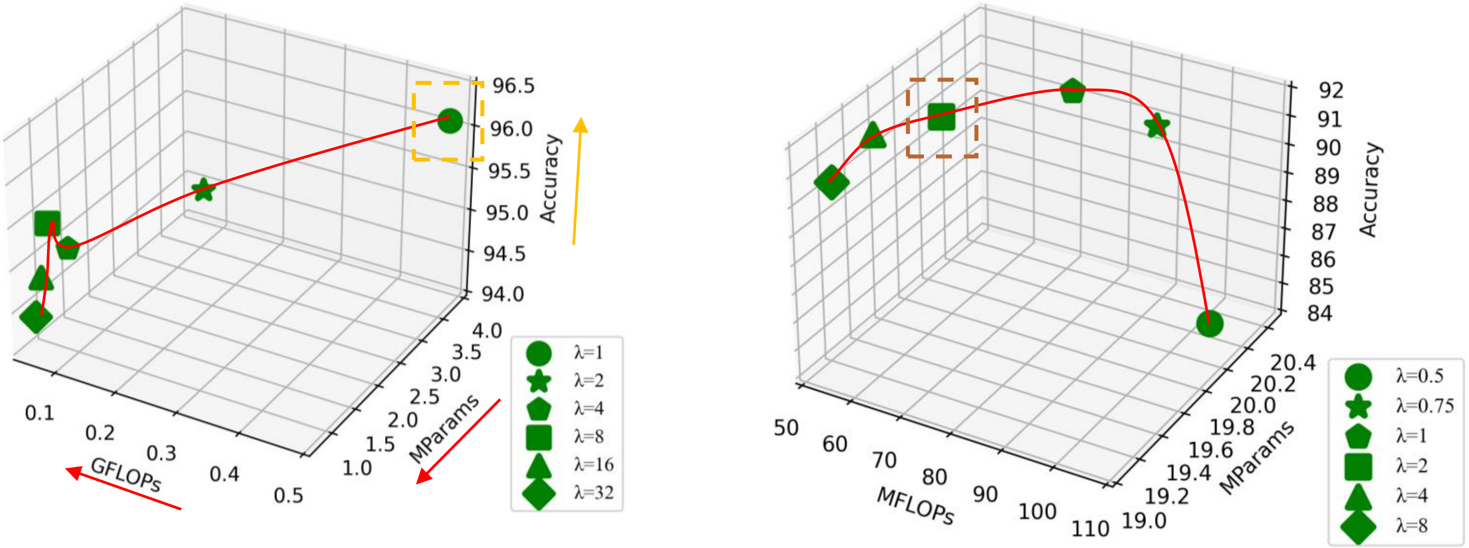


(b) VGG-16 on CIFAR10

The hyperparameter λ

2022 DCC

The Visualization of Accuracy, Parameters and FLOPs with different λ:



(a) ResNet-34 on Chest X-Ray

(b) VGG-16 on CIFAR10

Layer analysis

Table 2: Structural analysis of ResNet-34 on Chest X-Ray

Model	FLOPs(G)	Params(M)	Acc (%)
ResNet-34	3.68	21.80	96.79
Conv1	3.57	21.79	94.79
Layer1	3.05	21.60	96.63
Layer2	2.90	20.80	96.63
Layer3	2.53	15.94	96.96
Layer4	3.14	10.83	96.31
Layer1 and Layer4	2.51	10.61	96.79
Layer1 to Layer4	0.28	1.39	96.79
Conv1 to Layer4	0.28	1.39	96.00

Layer analysis

Table 3: Structural analysis of VGG-16 on CIFAR10

Model	FLOPs(M)	Params(M)	Acc(%)	Model	FLOPs(M)	Params(M)	Acc(%)
VGG-16	352.41	33.65	93.60	Conv8	335.40	32.58	81.13
Conv1	351.74	33.65	84.88	Conv9	315.21	31.32	83.68
Conv2	333.76	33.63	84.03	Conv10	315.18	31.32	80.97
Conv3	351.87	33.65	86.99	Conv11	343.18	31.34	78.25
Conv4	351.25	33.64	78.76	Conv12	343.19	31.34	88.36
Conv5	335.60	33.38	88.14	Conv13	343.27	31.36	76.99
Conv6	316.85	33.09	80.46	Conv2-13	63.38	19.62	92.45
Conv7	316.82	33.09	85.15	ConvAll	62.74	19.62	91.89

Comparison with other methods

Table 4: Comparison of different tensor decomposion methods and lightweight networks

	CIFAR10			Chest X-Ray		
	SR	CR	Acc (%)	SR	CR	Acc (%)
MobileNetV2[17]	/	/	66.54	/	/	86.22
ShuffleNetV2[18]	/	/	72.00	/	/	91.67
ADATucker(Resnet-20)[14]	/	12.00	90.97	/	/	/
HOTCAKE(SimpleNet)[15]	/	3.13	90.95	/	/	/
Tucker_VBMF(VGG-16)[4]	3.01	1.61	90.50	3.50	1.10	96.00
LRC(VGG-16)[16]	3.57	1.61	90.92	23.35	1.12	93.59
LTC(VGG-16)	5.61	1.71	91.89	41.18	1.12	94.39
Tucker_VBMF(ResNet-34)[4]	18.98	5.01	85.30	4.97	4.67	96.30
LRC(ResNet-34)[16]	11.28	15.35	88.30	11.15	15.35	95.35
LTC(ResNet-34)	22.38	17.99	89.13	13.14	15.58	96.00

Conclusion

- proposed Learning Tucker Compression (LTC) to speed up CNNs.

- LTC takes the tucker-2 decomposition as a joint optimization of CNN’s weights and tucker’s ranks.

- Experiments show that LTC can effectively reduce the amount of parameters and accelerate CNNs with satisfied classification accuracy.

SUMMARY REPORT

Thank You For Your Watching!