Deep neural network compression using tensor methods

Paris, 09 September 2022

Presentation by: Tan Minh **CAO**

Supervisor: Dr. Yassine **ZNIYED**
Dr. Thanh Phuong **NGUYEN**

Outline

The problem stament

Problem stament

Deep neural network face the following problems:

- The number parameters is large
- They consume a lot of memory
- Limited calculation speed of embedded systems
- They consume a lot of power

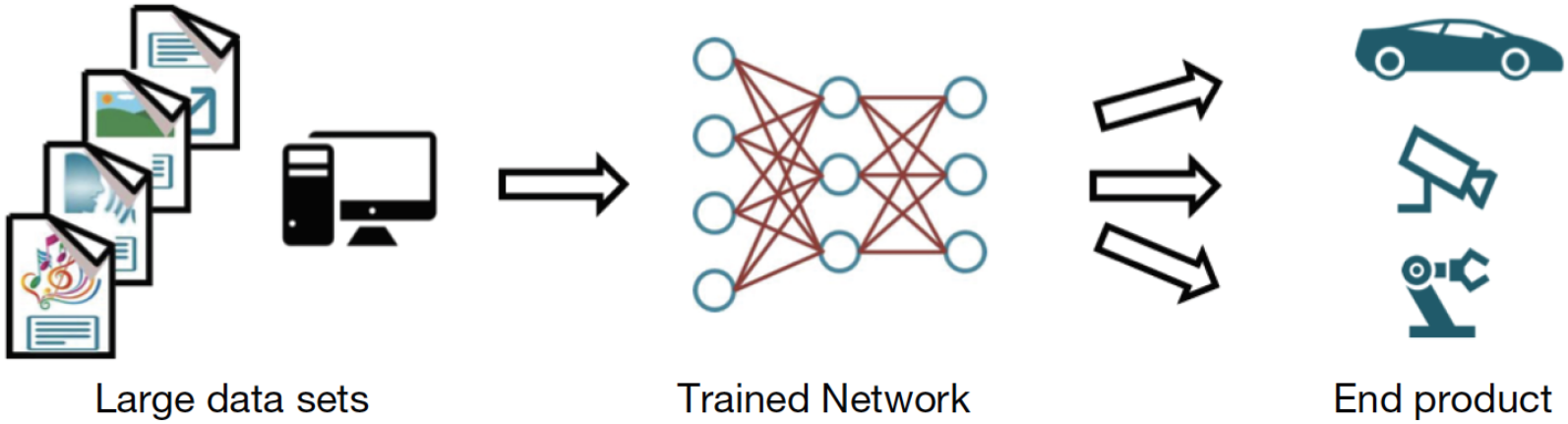


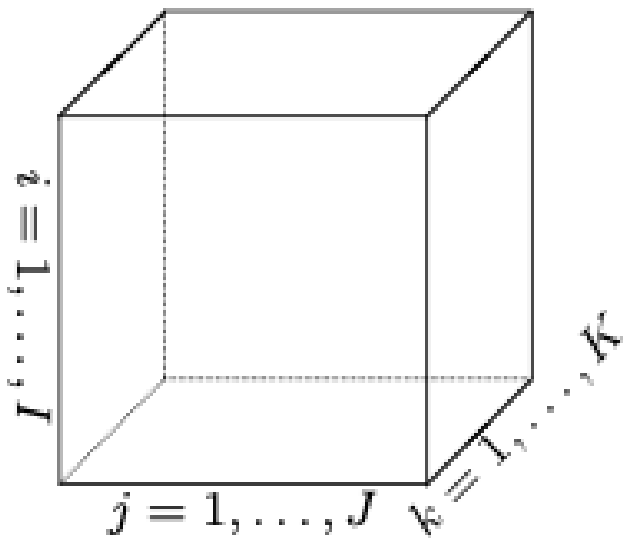
Figure: Machine learning for embedded systems

Tensor decompositions

What is a tensor?

Tensor is a geometric object, which can be defined as a multidimensional array

Example: A vector $a \in \mathbb{R}$ is a 1-order tensor, a matrix $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ is a 2-order tensor, a cube $\boldsymbol{X} \in \mathbb{R}^{I \times J \times K}$ is a 3-order tensor



Some usual operations in tensor calculation:

- n-mode product, Kronecker product, Khatri-Rao product, Hadamard product

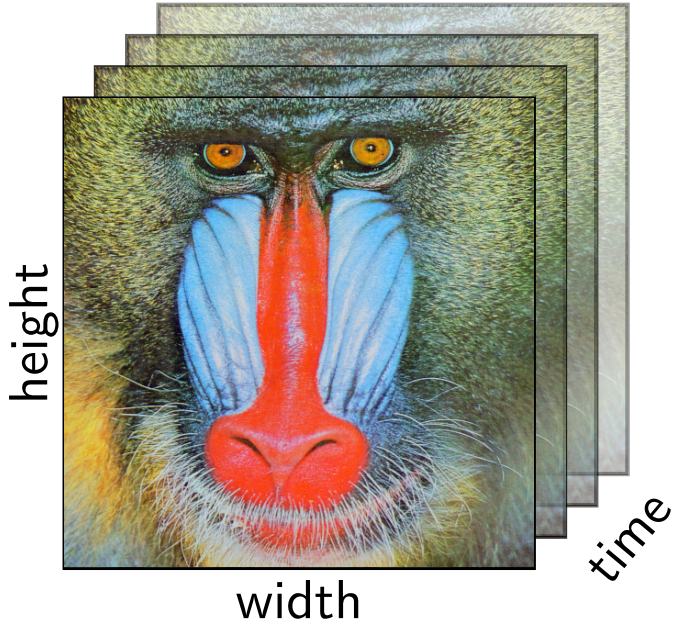
Why tensor decomposition?

Many objects in machine learning can be treated as tensors:

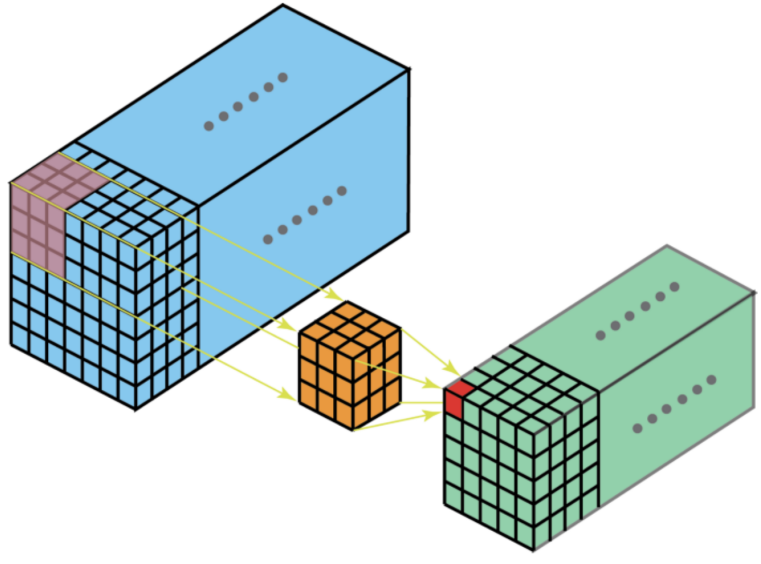
- Data cubes (RBG images, videos, different shape/orientations)
- Weight matrices can be treated as tensor, both in convolution layer and fully-connected layer

Example

The image RGB is a 3D tensor

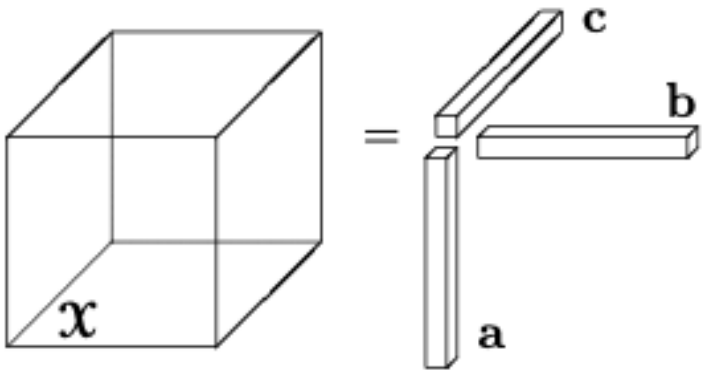


The kernel of convolution layer is a 3D tensor



Rank-one tensor: the tensor $\mathcal{X}$ is a rank-one tensor if it can be written as the outer product of N vectors:

$$\mathcal{X} = \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(2)} \otimes ... \otimes \boldsymbol{a}^{(N)} \tag{1}$$



Example: A 3-order tensor $\mathcal{X}$ can be written by $\mathcal{X} = \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}$.
The element $x_{ijk} = a_i \cdot b_j \cdot c_k$

Tensor rank R: The minimal possible R is called the rank (canonical) of tensor $\mathcal{X}$:

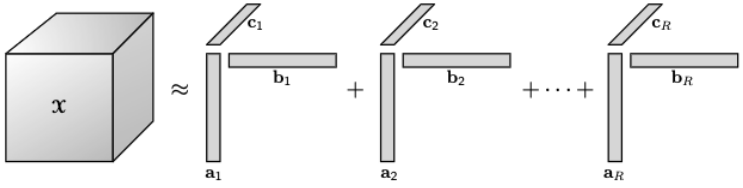
$$\mathcal{X}(i_1, i_2, ..., i_d) = \sum_{r=1}^{R} U_1(i_1, r) \cdot U_2(i_2, r) \dots U_d(i_d, r) \tag{2}$$

Graphical illustration of the CP and Tucker decompositions

CP-decomposition:

$$\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

$$x_{ijk} = a_{ir} \cdot b_{jr} \cdot c_{kr}$$



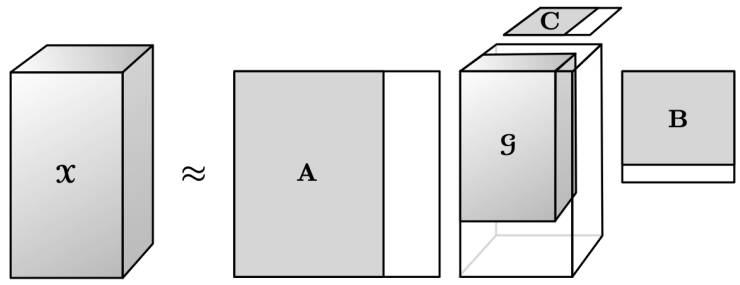
$$\mathcal{X} \approx \sum_{r=1}^{R} \boldsymbol{A_1}(i_1, r) \otimes \ldots \otimes \boldsymbol{A_d}(i_d, r)$$

$$\mathcal{X} \approx \tilde{\mathcal{X}} = [\![\lambda; \mathbf{A}_1, \mathbf{A}_2, \ldots \mathbf{A}_d]\!]$$

Tucker decomposition:

$$\mathcal{X} = [\![G, \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

$$x_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \cdot a_{ip} \cdot b_{jq} \cdot c_{kr}$$



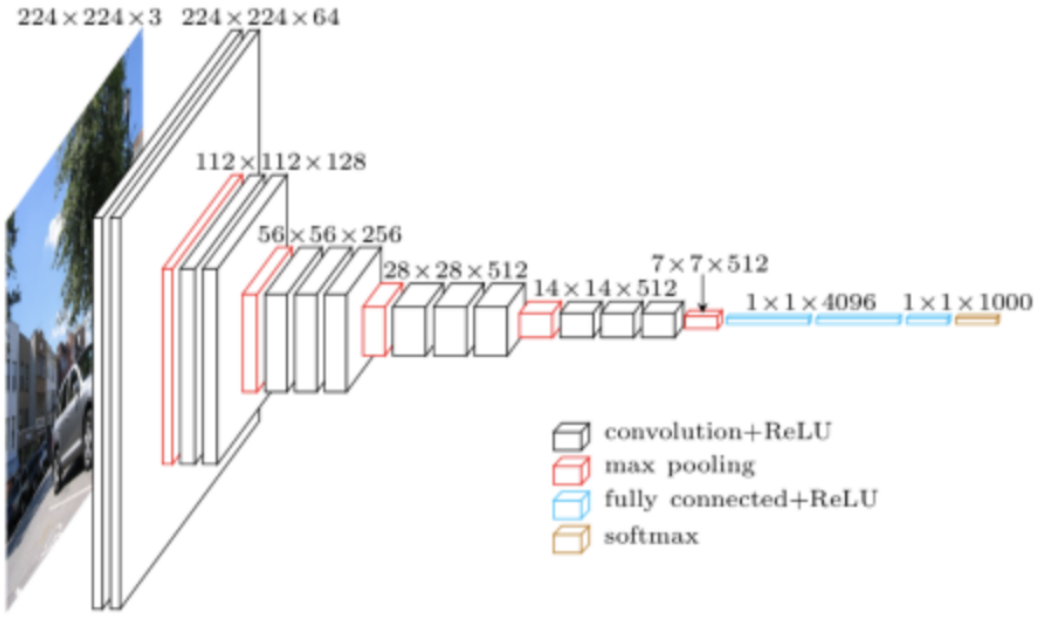
$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \ldots \times_N \boldsymbol{A}^{(N)}$$

$$\mathcal{X} \approx \tilde{\mathcal{X}} = [\![\mathcal{G}; \mathbf{A}_1, \mathbf{A}_2, \ldots \mathbf{A}_N]\!]$$

DNN compression using tensor decompositions

Transfer learning VGG16 with dataset Cifar10 (I)

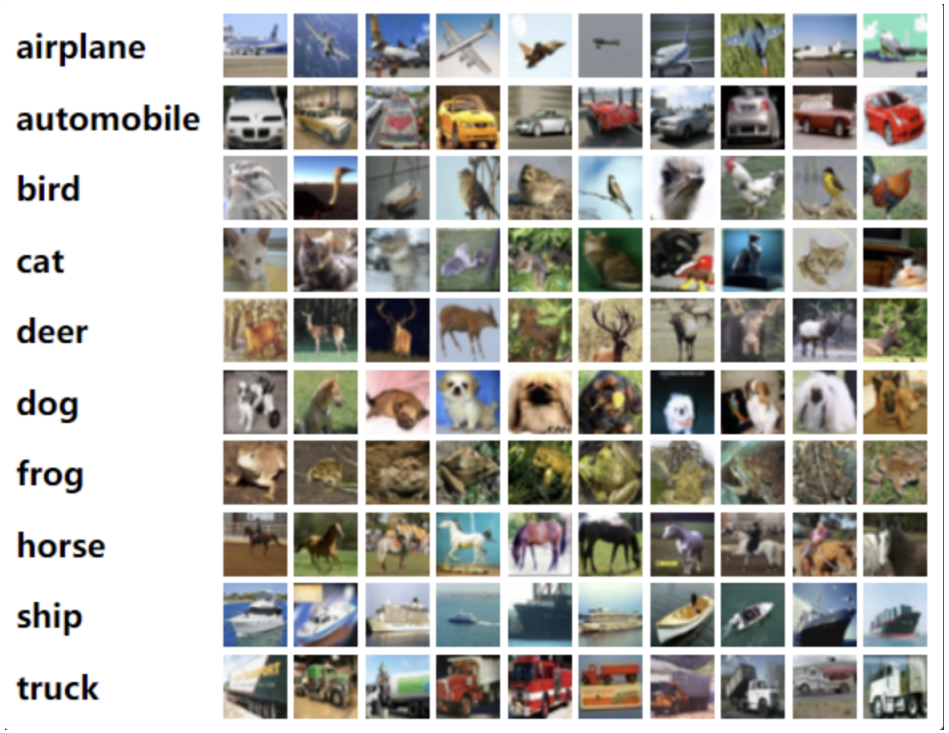
Model VGG16



- Input: color image of size 224×224
- 13 Convolution layers with kernel filers of size 3×3 and 3 Fully connected layers
- Output: 10 classes (the number classes of dataset Cifar 10)

Transfer learning VGG16 with dataset Cifar10 (II)

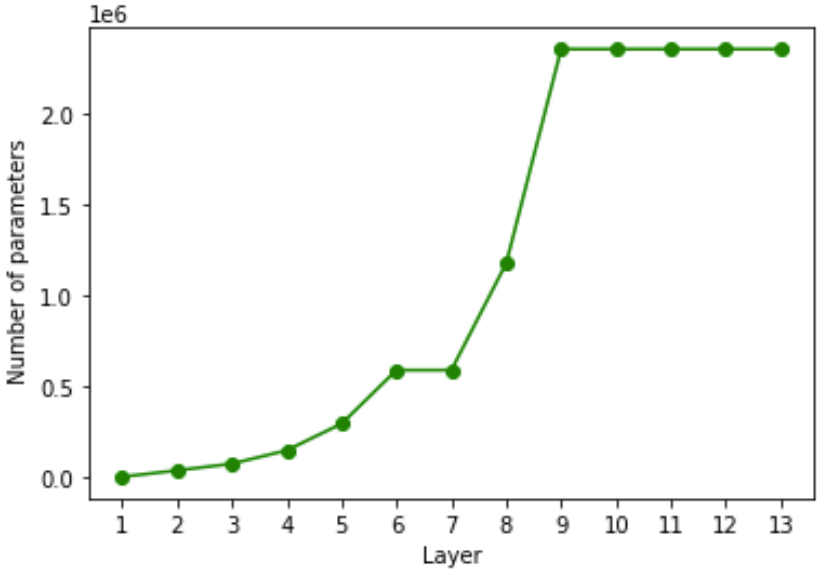
Dataset Cifar10

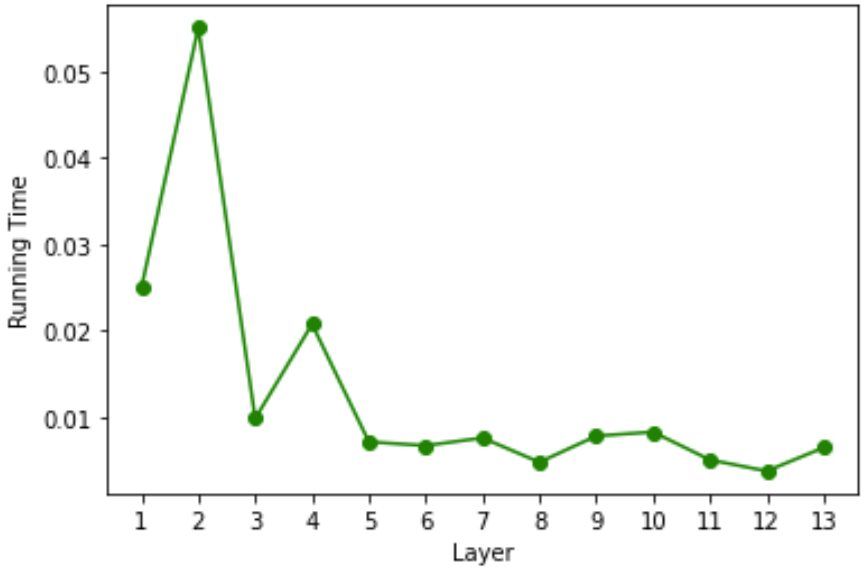


- 60000 color images of size 32×32
- 50000 images for training, 10000 images for testing
- 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

Convolution layer of VGG16

- Model VGG16 has about 138 million parameters.
- The accuracy of VGG16 pre-trained with dataset Cifar10 is 94.08%





- The number of parameters of layers 9 to 13 is very large.
- Layer 2 has the longest running time

Use tensor methods to compress layers 2 and 9.

DNN compression using CPD: mathematical formulation

Classical convolution operation:

$$\boldsymbol{\mathcal{V}}(x,y,t) = \sum_{i=x-\delta}^{i=x+\delta} \sum_{j=x-\delta}^{i=x+\delta} \sum_{s=1}^{S} \boldsymbol{\mathcal{K}}(i-x+\delta, j-y+\delta, s, t) \cdot \boldsymbol{\mathcal{U}}(i,j,s) \quad (3)$$

CPD of 4D kernel tensor $\boldsymbol{\mathcal{K}}$ of size $d \times d \times S \times T$:

$$\boldsymbol{\mathcal{K}}(j,j,s,t) = \sum_{r=1}^{R} \boldsymbol{K}^{x}(i-x+\delta, r) \times \boldsymbol{K}^{y}(j-x+\delta, r) \times \boldsymbol{K}^{s}(s,r) \times \boldsymbol{K}^{t}(t,r) \quad (4)$$

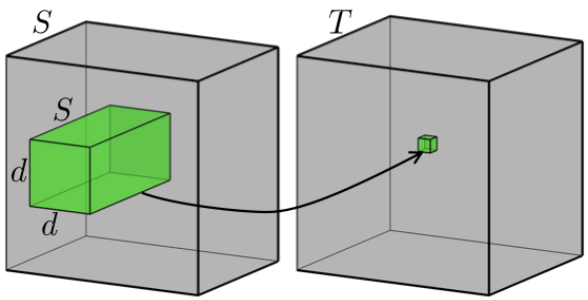
Convolution using CPD (replace (3) in (4))

$$\boldsymbol{\mathcal{V}}(x,y,t) = \sum_{r=1}^{R} \boldsymbol{K}^{t} \left(\sum_{i=x-\delta}^{i=x+\delta} \boldsymbol{K}^{x} \left(\sum_{j=y-\delta}^{j=y+\delta} \boldsymbol{K}^{y} \left(\sum_{s=1}^{S} \boldsymbol{K}^{s} \boldsymbol{U}(i,j,s) \right) \right) \right) \quad (5)$$

DNN compression using CPD: graphical illustration

Classical and CPD convolutions

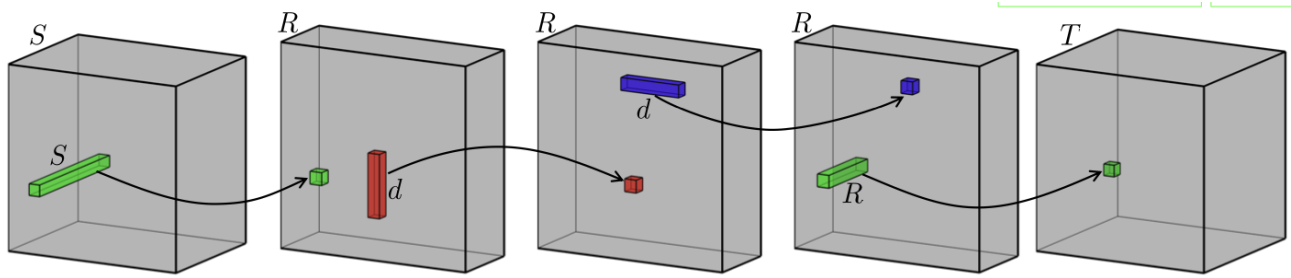
$$\boldsymbol{U}^s(i,j,r) = \sum_{s=1}^{S} \boldsymbol{K}^s(s,r)\boldsymbol{U}(i,j,s)$$



$$\boldsymbol{U}^{sy}(i,y,r) = \sum_{j=y-\delta}^{y+\delta} \boldsymbol{K}^y(j-y+\delta,r)\boldsymbol{U}^s(i,j,r)$$

$$\boldsymbol{U}^{syx}(x,y,r) = \sum_{i=x-\delta}^{x+\delta} \boldsymbol{K}^x(j-y+\delta,r)\boldsymbol{U}^{sy}(i,y,r)$$

$$\boldsymbol{V}(x,y,t) = \sum_{r=1}^{R} \boldsymbol{K}^t(t,r)\boldsymbol{U}^{syx}(x,y,r)$$



DNN compression using Tucker decomposition: mathematical formulation

Classical convolution operation:

$$\mathcal{V}(x,y,t) = \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{s=1}^{S}\mathcal{K}(i,j,s,t)\mathcal{U}(i,j,s) \tag{6}$$

Tucker decomposition of 4D $\mathcal{K}$ of size $d \times d \times S \times T$:

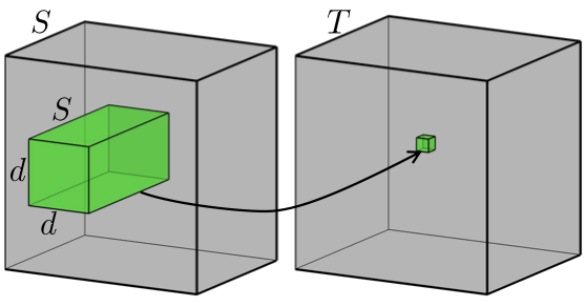
$$\mathcal{K}(i,j,s,t) = \sum_{r_3=1}^{R_3}\sum_{r_4=1}^{R_4}\mathcal{G}_{i,j,r3,r4}\boldsymbol{U}^3_{s,r3}\boldsymbol{U}^4_{t,r4} \tag{7}$$

Convolution using Tucker (replace (7) in (6))

$$\mathcal{V}(x,y,t) = \sum_{r_4=1}^{R_4}\boldsymbol{U}^3\left(\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{r_3=1}^{R_3}\mathcal{G}\left(\sum_{s=1}^{S}\boldsymbol{U}^4\cdot\mathcal{U}(i,j,s)\right)\right) \tag{8}$$

DNN compression using Tucker decomposition: graphical illustration

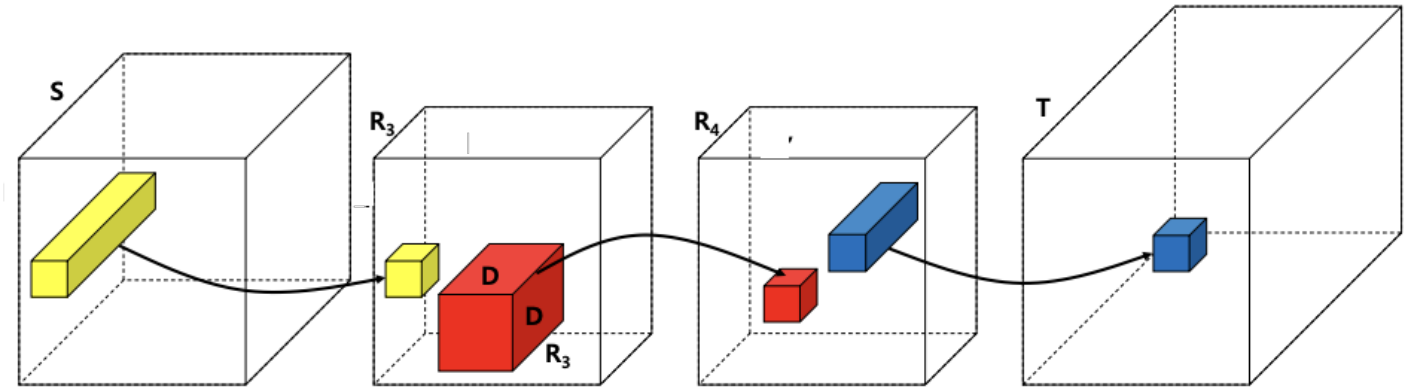
Classical and Tucker decomposition convolutions



$$U^{r_3}(i,j,r3) = \sum_{s=1}^{S} \boldsymbol{U}^3(s,r_3)\mathcal{U}(i,j,s)$$

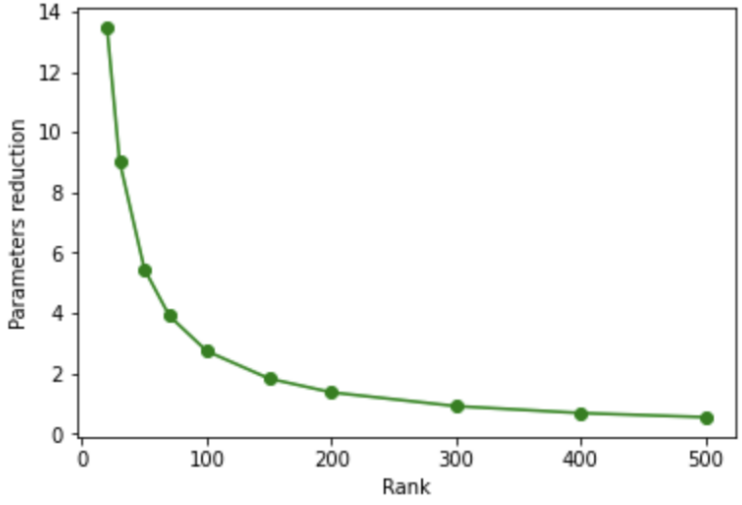
$$U^{r_3,r_4}(x,y,r_4) = \sum_{i=1}^{d}\sum_{i=1}^{d}\sum_{r_3=1}^{R_3} \mathcal{G}(i,j,r_3,r_4)U^{r_3}(i,j,r3)$$

$$V(x,y,t) = \sum_{r_4=1}^{R_4} \boldsymbol{U}^4(t,r_4)U^{r_3,r_4}(x,y,r_4)$$

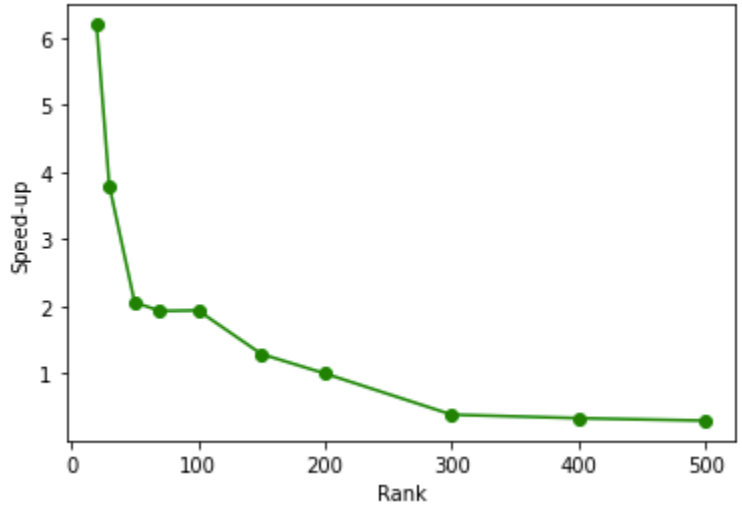


Simulation results

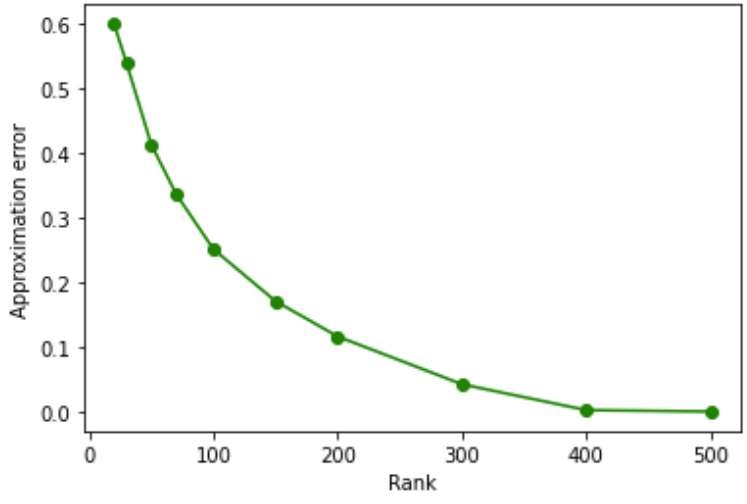
CP-decomposition for layer 2 (I)



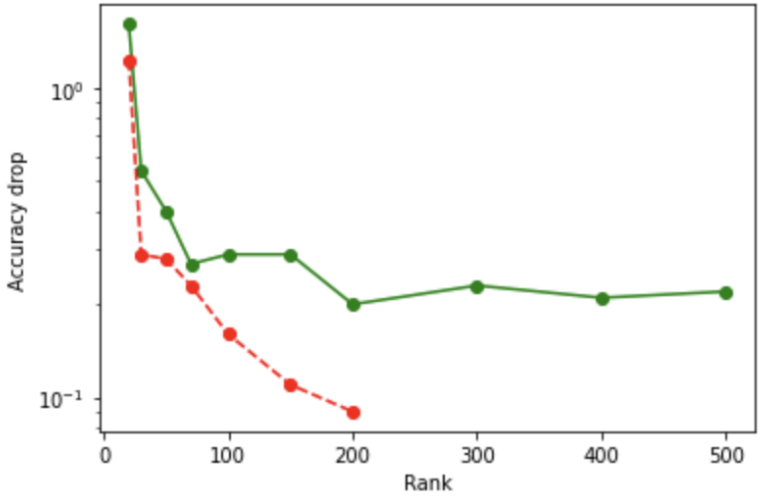
Parameter reduction



Speed-up



Approximation error



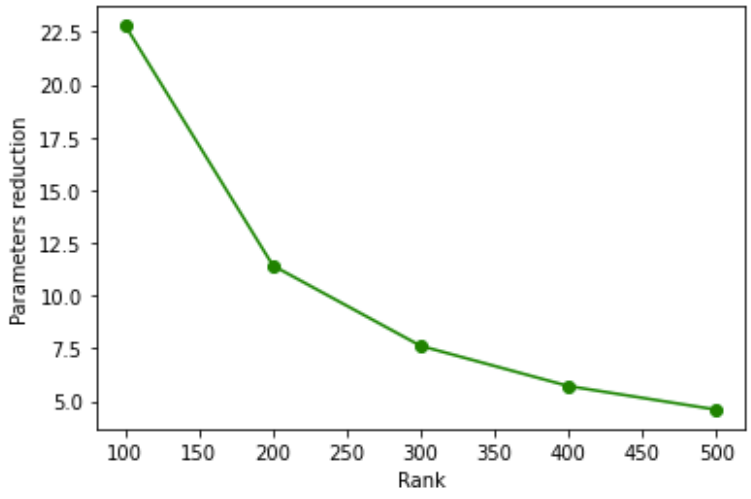
Accuracy drop

CP-decomposition layer 2 (II)

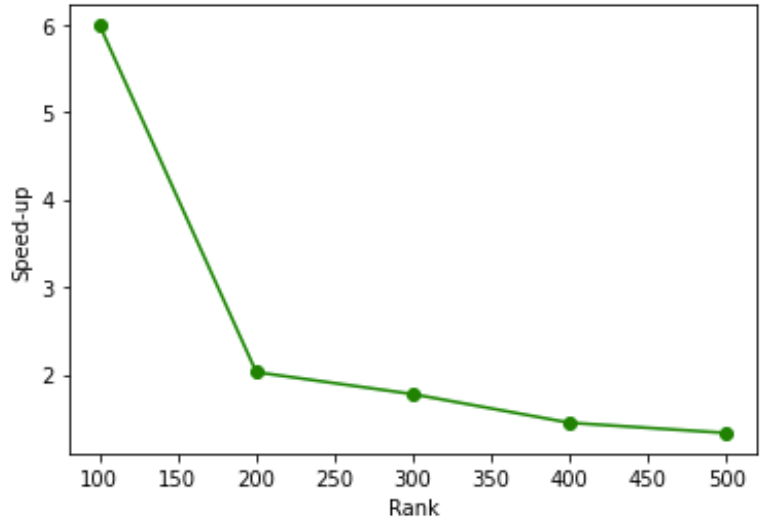
Setting the value of rank $R \in (20, 30, 50, 70, 100, 150, 200, 300, 400, 500)$

- The number of parameters of the convolution layer has been greatly reduced ($R = 20$ - 13.45 times, and $R = 200$ - 1.37 times)
- The time execution has been accelerated ($R = 20$ - 6.19 times, and $R = 200$ - 1.29 times)
- The approximation error decreased as the value of R increased
- Although the approximation error is high, the accuracy of the compression model is still high.(accuracy $= 92.47\%$ with $R = 20$, and accuracy $= 93.79\%$ with $R = 200$)
- After fine-tuning, the accuracy of the model is improved

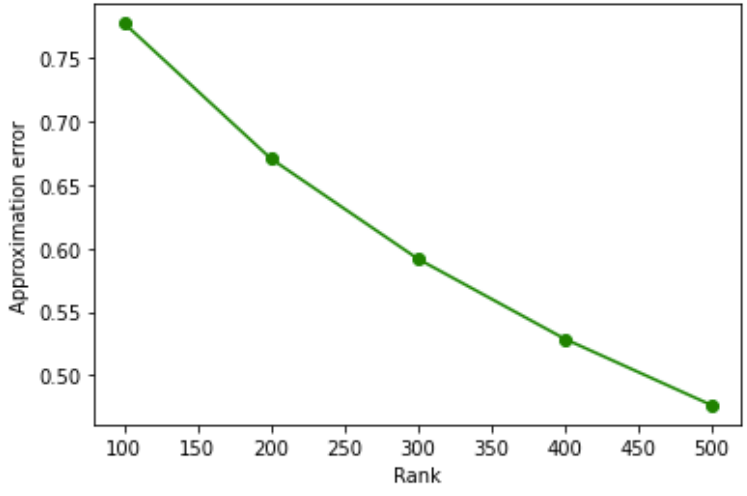
CP-decomposition for layer 9 (I)



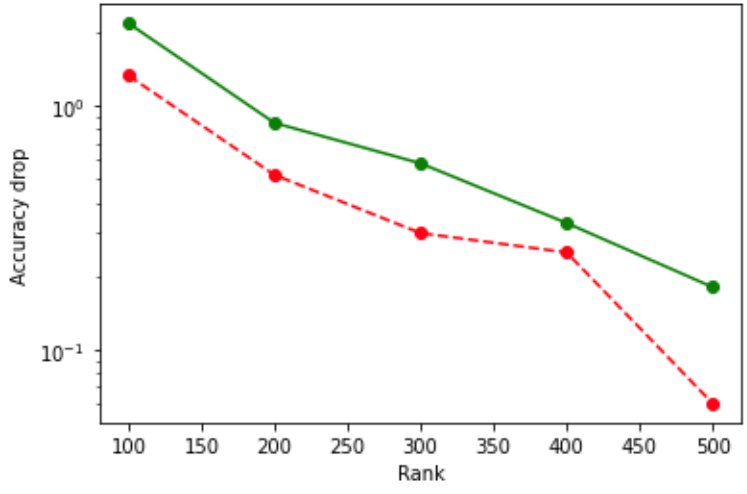
Parameter reduction



Speed-up



Approximation error



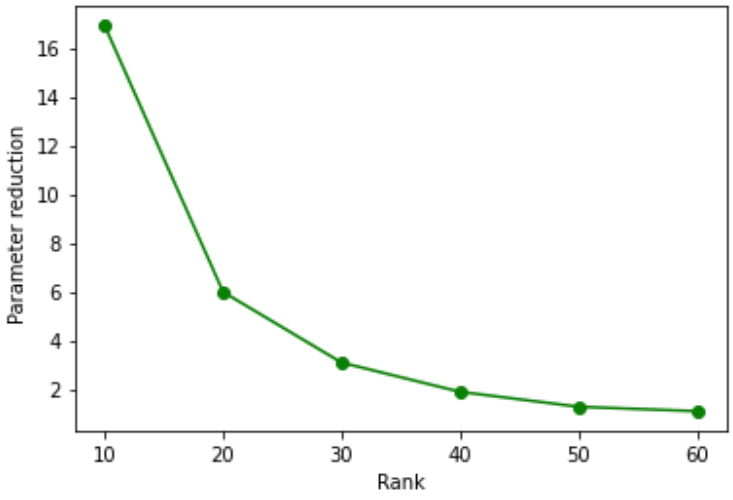
Accuracy drop

CP-decomposition for layer 9 (II)

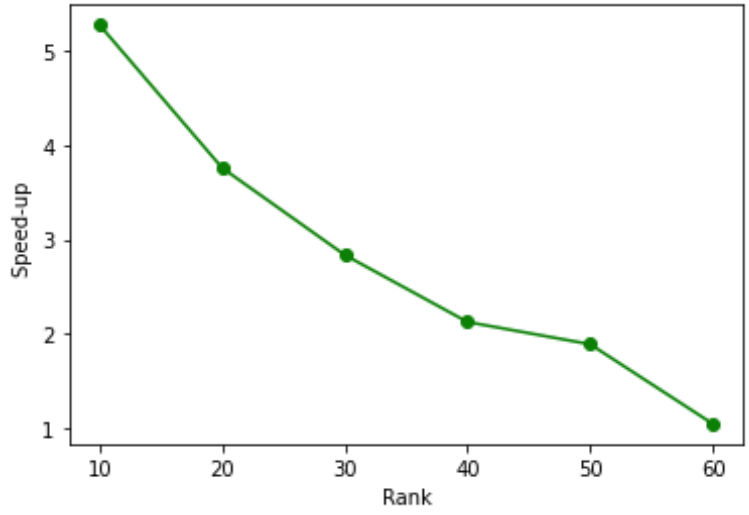
Setting the value of rank $R \in (100, 200, 300, 400, 500)$

- The number of parameters of the convolution layer has been greatly reduced ($R = 100$ - 22.79 times, and $R = 500$ - 4.6 times)
- The time execution has been accelerated ($R = 100$ - 5.98 times, and $R = 500$ - 1.33 times)
- The approximation error decreased as the value of R increased
- Although the approximation error is high, the accuracy of the compression model is still high.(accuracy $= 91.9\%$ with $R = 100$, and accuracy $= 93.9\%$ with $R = 500$)
- After fine-tuning, the accuracy of the model is improved

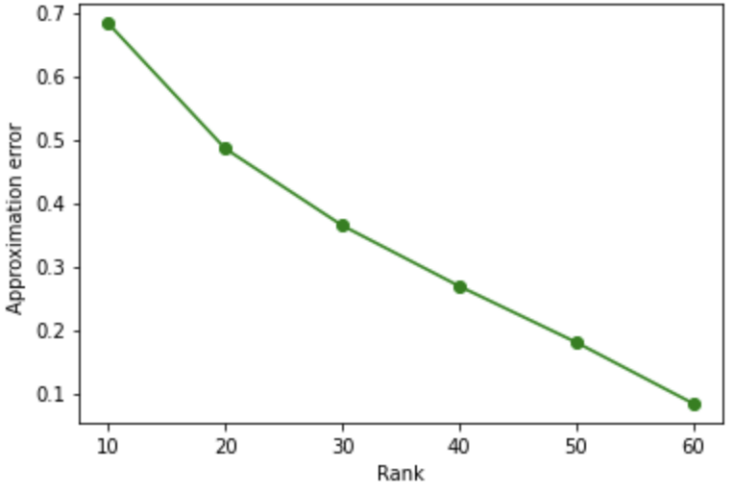
Tucker decomposition for layer 2 (I)



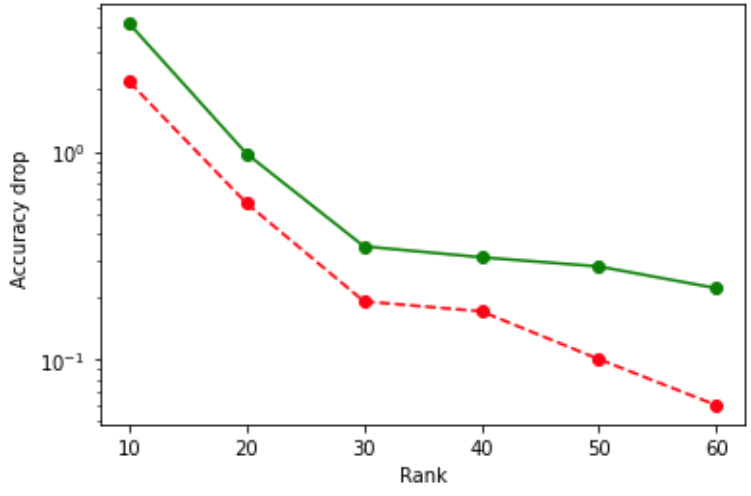
Parameter reduction



Speed-up



Approximation error



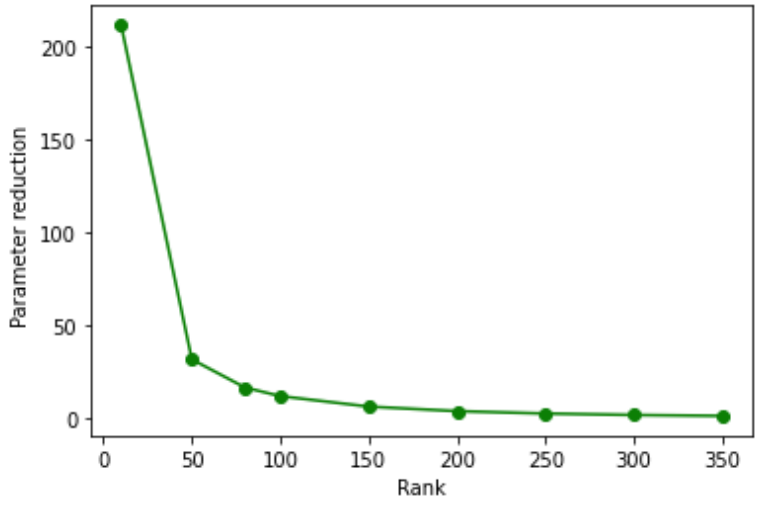
Accuracy drop

Tucker decomposition for layer 2 (II)

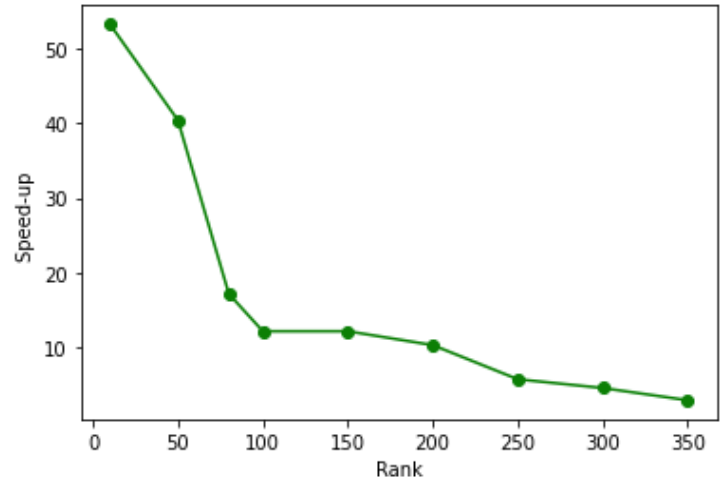
Setting the value of rank $R \in (10, 20, 30, 40, 50, 60)$

- The number of parameters of the convolution layer has been greatly reduced ($R = 10$ - 16.9 times, and $R = 60$ - 1.1 times)
- The time execution has been accelerated ($R = 10$ - 5.27 times, and $R = 60$ - 1.05 times)
- The approximation error decreased as the value of R increased
- Although the approximation error is high, the accuracy of the compression model is still high.(accuracy $= 89.92\%$ with $R = 10$, and accuracy $= 93.9\%$ with $R = 93.86$)
- After fine-tuning, the accuracy of the model is improved

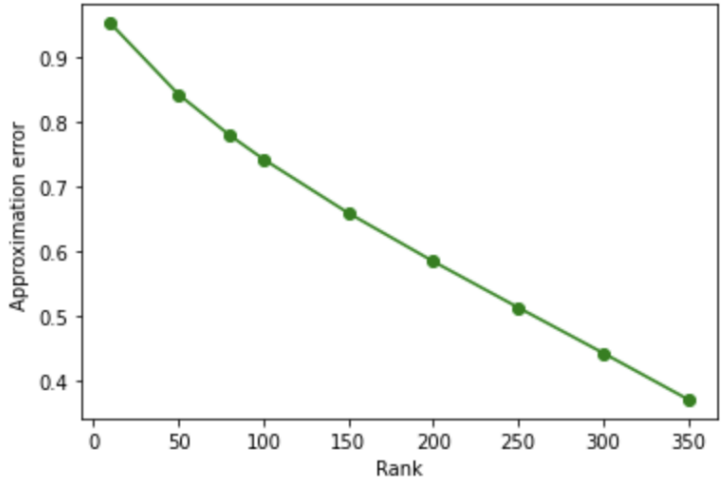
Tucker decomposition for layer 9 (I)



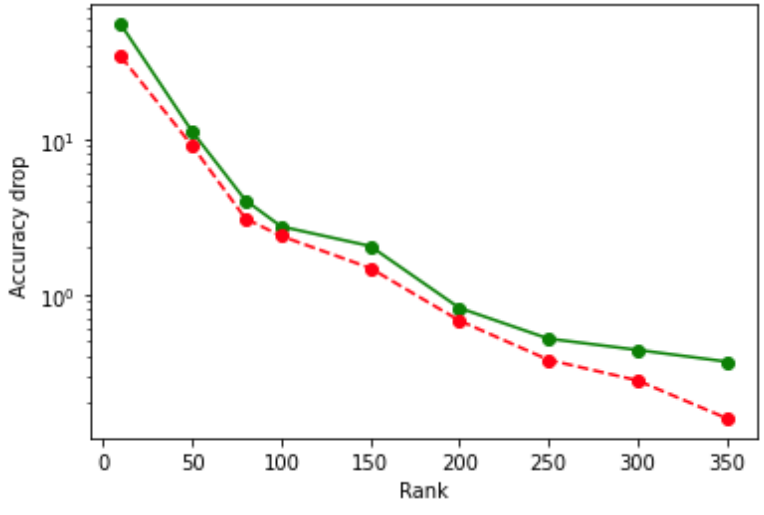
Parameter reduction



Speed-up



Approximation error



Accuracy drop

Tucker decomposition for layer 9 (II)

Setting the value of rank $R \in (10, 50, 80, 100, 150, 200, 250, 300, 350)$

- The number of parameters of the convolution layer has been greatly reduced ($R = 10$ - 211.8 times, and $R = 350$ - 1.6 times)
- The time execution has been accelerated ($R = 10$ - 53.3 times, and $R = 350$ - 2.85 times)
- The approximation error decreased as the value of R increased
- the value of rank R is small, the accuracy of the model is low (accuracy = 39.35% with R = 10). The accuracy of the model increases very rapidly as the value of R increases accuracy = 93.9% with R = 350)
- After fine-tuning, the accuracy of the model is improved

Conclusions

Conclusions

This work

- Transfer learning: VGG16 with dataset Cifar10.
- Decomposition of kernel tensors.
- CP and Tucker decompositions to compress the DNN convolution layers.
- Fine-turning after compression.

Future work

- Decomposition of all the layer using CP and Tucker decompositions.
- Comparison with other DNN compression method (Tensor train, Pruning, Quantization).

Thank you for your attention !

References

1. Tamara G.Kolda and Brett W.Bader, "Tensor Decompositions and Applications", in SIAM Review Volume 51 Issue 3 August 2009 pp 455–500, https://doi.org/10.1137/07070111X
2. Stephan Rabanser, Oleksandr Shchur, Stephan Günnemann, "Introduction to Tensor Decompositions and their Applications in Machine Learning", in Machine Learning (stat.ML); Machine Learning (cs.LG), https://arxiv.org/abs/1711.10781
3. Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, Victor Lempitsky, "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition", in International Conference on Learning Representations, ICLR
4. Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, Dongjun Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications", https://arxiv.org/abs/1511.06530

n-mode product

The multiplication of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times I_n \times I_{n+1} \times \ldots \times I_N}$ with a matrix $\boldsymbol{A} \in \mathbb{R}^{J \times I_n}$.

$$\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A}_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} \cdot a_{j i_n} \tag{9}$$

Kronecker product

The Kronecker product of matrices $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{R}^{K \times L}$:

$$\boldsymbol{A} \boxtimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} \ldots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} \ldots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} \ldots & a_{IJ}\boldsymbol{B} \end{bmatrix} \tag{10}$$

$$= [a_1 \otimes b_1 \, a_1 \otimes b_2 \ldots a_J \otimes b_{L-1} \; a_J \otimes b_L]$$

Appendix (II)

Khatri-Rao product

Given 2 matrices $\boldsymbol{A} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J \times K}$, their Khatri-Rao product, denoted by $\odot$, performs the following operation:

$$\boldsymbol{A} \odot \boldsymbol{B} = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 \ldots & a_K \otimes b_K \end{bmatrix} \tag{11}$$

The Hadamard product

The The Hadamard product of 2 matrices both of size $I \times J$ denoted by $\boxdot$:

$$\boldsymbol{A} \boxdot \boldsymbol{B} = \begin{bmatrix} a_{11} b_{11} & a_{12} b_{12} & \ldots & a_{1J} b_{1J} \\ a_{21} b_{21} & a_{22} b_{22} & \ldots & a_{2J} b_{2J} \\ \vdots & \vdots & & \vdots \\ a_{I1} b_{I1} & a_{I2} b_{I2} & \ldots & a_{IJ} b_{IJ} \end{bmatrix} \tag{12}$$

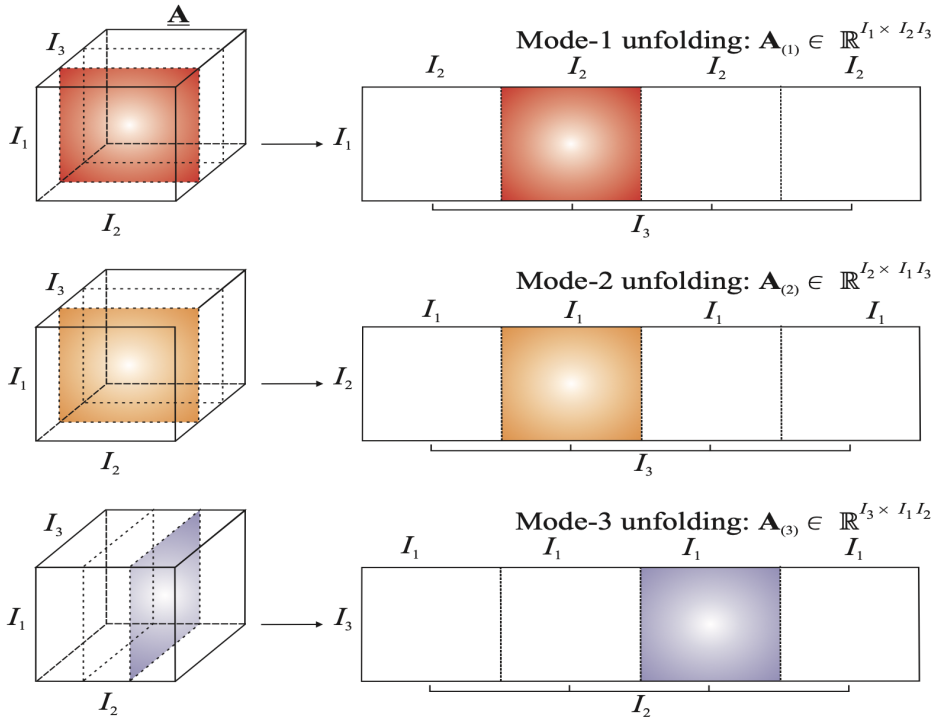
Unfolding matrices: definition

The mode-n matricization of tensor $\mathcal{X}$:

$$\boldsymbol{X}_{(n)} = [(i_1 \dots i_n; i_{n+1} \dots i_N)] \tag{13}$$

Where $\boldsymbol{X}(i_1 \dots i_n; i_{n+1} \dots i_N) = \boldsymbol{X}(i_1, \dots, i_d)$

Example: 3 modes of unfolding of tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$



Apporoximation error The apporoximation error is :

$$A^r = \frac{\| \boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}} \|}{\| \boldsymbol{\mathcal{X}} \|} \tag{14}$$