Tensor Methods for Neural Network Compression

Lokesh Veeramacheneni

Fraunhofer SCAI

Moritz Wolter

Prof. Jochen Garcke

Table of contents

Why tensor decomposition?

- Training a CNN on low end hardware
  - Memory constraint
  - Computational Speed
  - Power intensive
- FPGA based ConvNet [1]
  - Inapplicability on high end architectures
  - Change in architecture requires complete redesigning
- Early attempts made using tensor decomposition [2]
  - CP Decomposition - Nonlinear Least Squares (NLS)
  - Performed on second layer of AlexNet and CharNet
  - Rank 140 approximation

What is a Tensor?

- Tensor is a d-dimensional array

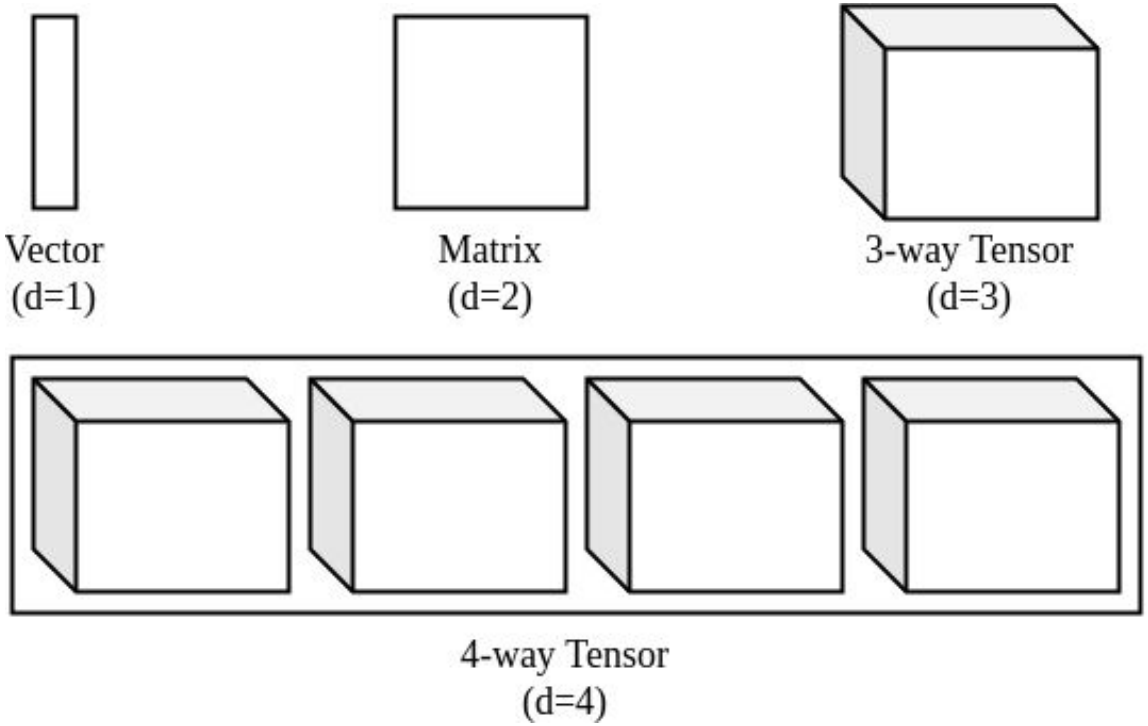
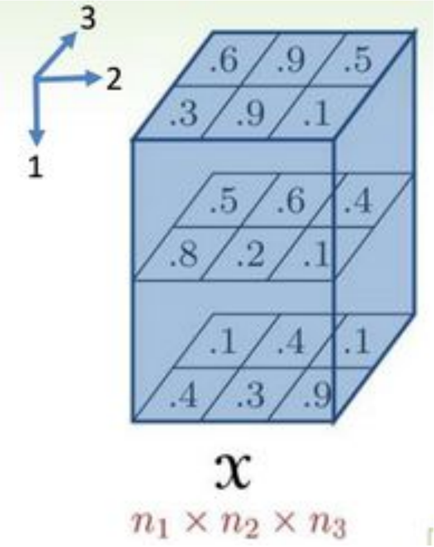


Figure 1: Representation of tensor upto four dimensions [3]

Unfolding a tensor (3D)



$$(i_1, i_2, i_3) \rightarrow (i_i, i_1'), \quad i_1' = (i_3 - 1)n_2 + i_2$$

$$\mathbf{X}_{(1)} = \begin{bmatrix} 0.3 & 0.9 & 0.1 & 0.6 & 0.9 & 0.5 \\ 0.8 & 0.2 & 0.1 & 0.5 & 0.6 & 0.4 \\ 0.4 & 0.3 & 0.9 & 0.1 & 0.4 & 0.1 \end{bmatrix} \quad n_1 \times n_2 n_3$$

$$(i_1, i_2, i_3) \rightarrow (i_2, i_2'), \quad i_2' = (i_3 - 1)n_1 + i_1$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 0.3 & 0.8 & 0.4 & 0.6 & 0.5 & 0.1 \\ 0.9 & 0.2 & 0.3 & 0.9 & 0.6 & 0.4 \\ 0.1 & 0.1 & 0.9 & 0.5 & 0.4 & 0.1 \end{bmatrix} \quad n_2 \times n_1 n_3$$

$$(i_1, i_2, i_3) \rightarrow (i_3, i_3'), \quad i_3' = (i_2 - 1)n_1 + i_1$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 0.3 & 0.8 & 0.4 & 0.9 & 0.2 & 0.3 & 0.1 & 0.1 & 0.9 \\ 0.6 & 0.5 & 0.1 & 0.9 & 0.6 & 0.4 & 0.5 & 0.4 & 0.1 \end{bmatrix}$$

$$n_3 \times n_1 n_2$$

Figure 2: Illustration of unfolding a 3d tensor [4]

Kronecker Product

- Generalization of outer product of vectors to matrices
- Denoted by ⓧ

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} & 2\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} \\ 3\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} & 4\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}$$

Figure 3: Illustration of kronecker product over matrices [5]

Khatri-Rao Product

- Column wise kronecker product
- Denoted by $\odot$

$$\mathbf{C} = \left[\begin{array}{c|c|c} \mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 \end{array}\right] = \left[\begin{array}{c|c|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right], \quad \mathbf{D} = \left[\begin{array}{c|c|c} \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 \end{array}\right] = \left[\begin{array}{c|c|c} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{array}\right]$$

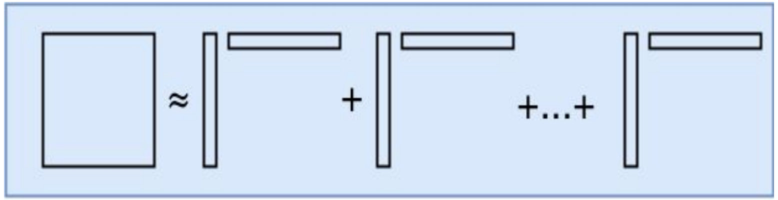
$$\left[\begin{array}{c|c|c} \mathbf{C}_1 \otimes \mathbf{D}_1 & \mathbf{C}_2 \otimes \mathbf{D}_2 & \mathbf{C}_3 \otimes \mathbf{D}_3 \end{array}\right] = \left[\begin{array}{c|c|c} 1 & 8 & 21 \\ 2 & 10 & 24 \\ 3 & 12 & 27 \\ 4 & 20 & 42 \\ 8 & 25 & 48 \\ 12 & 30 & 54 \\ 7 & 32 & 63 \\ 14 & 40 & 72 \\ 21 & 48 & 81 \end{array}\right]$$

Figure 4: Illustration of khatri-rao product over matrices [5]

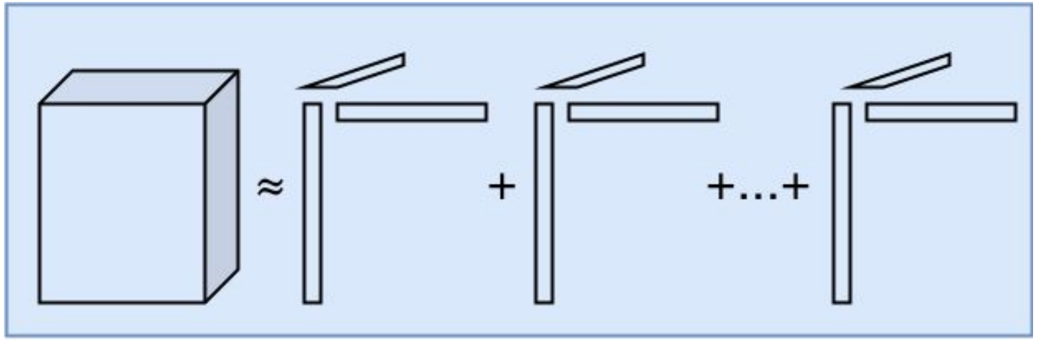
Candecomp/PARAFAC (CP) decomposition

- CP-decomposition can be viewed as matrix SVD generalized to tensors
- Unlike SVD, no orthogonality constraints are required
- It is defined as sum of d-dimensional outer products

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r,$$



Singular Value Decomposition



CP Decomposition

CP Decomposition

- Computed using Alternating Least Squares(ALS) method
- In 3-way decomposition, A, B and C are optimized sequentially [1]
- In ALS, we minimize the cost function $||X-M||^2$

$$\min_{\hat{\mathbf{A}}} \|\mathbf{X}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^{\mathsf{T}}\|_F,$$

$$\hat{\mathbf{A}} = \mathbf{X}_{(1)} \left[(\mathbf{C} \odot \mathbf{B})^{\mathsf{T}}\right]^{\dagger}.$$

Equation representing the optimization of variables in ALS method [1]

CP-ALS algorithm

procedure CP-ALS($\mathcal{X}$, R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Algorithm 1: ALS algorithm to compute decomposition factors [3]

CP Decomposition

- Input tensor shape (2 x 3 x 3)
- Approximation by various ranks

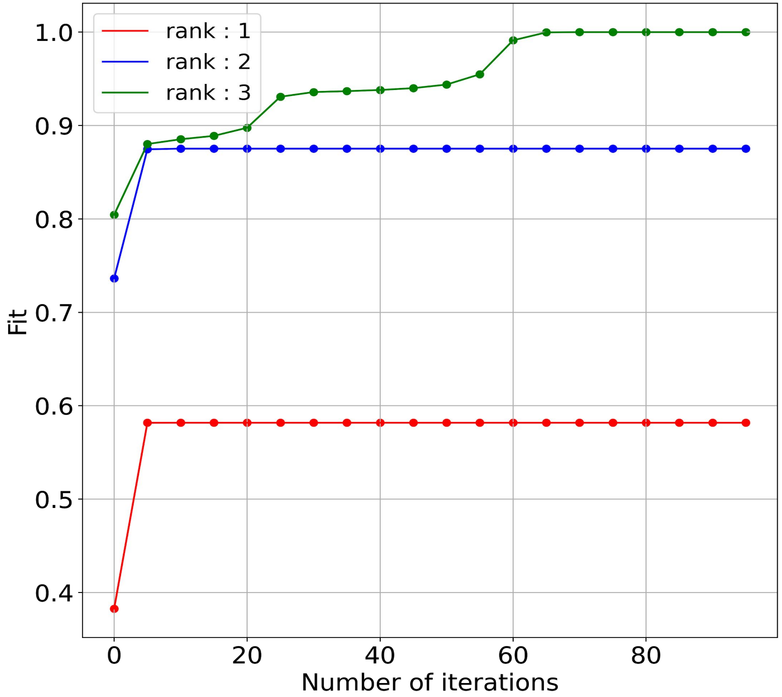


Figure 2: Convergence of ALS method with different ranks

CP Decomposition

- Unit tested with tensorly implementation
- Parameters
  - Rank : 2
  - Maximum iterations : 100

Reconstruction error	0.315	0.591

```
[[[-1.4192, -0.5040,  0.4957],
  [-0.9996,  0.8199,  0.6812],
  [ 1.0946, -2.0008,  0.7139]],

 [[ 0.6136,  1.2167, -0.5914],
  [-0.0321,  1.2354, -0.4008],
  [ 0.2624,  1.1269,  0.4031]]]
```

Input Tensor

```
[[[-1.2103, -0.3307,  0.5578],
  [-1.3484,  0.5286,  0.3366],
  [ 0.8815, -2.1799,  0.3622]],

 [[ 0.3615,  1.4637, -0.5998],
  [ 0.2641,  0.9407, -0.3974],
  [ 0.1108,  0.7960, -0.2941]]]
```

Reconstruction from ALS method

```
[[[-1.2102, -0.3308,  0.5573],
  [-1.3483,  0.5285,  0.3363],
  [ 0.8820, -2.1798,  0.3621]],

 [[ 0.3618,  1.4638, -0.5998],
  [ 0.2639,  0.9406, -0.3971],
  [ 0.1120,  0.7961, -0.2945]]]
```

Reconstruction from tensorly

AlexNet

- Convolutional Neural Network (CNN)
- First five convolutional layers
- Last three fully connected layers act as classifier

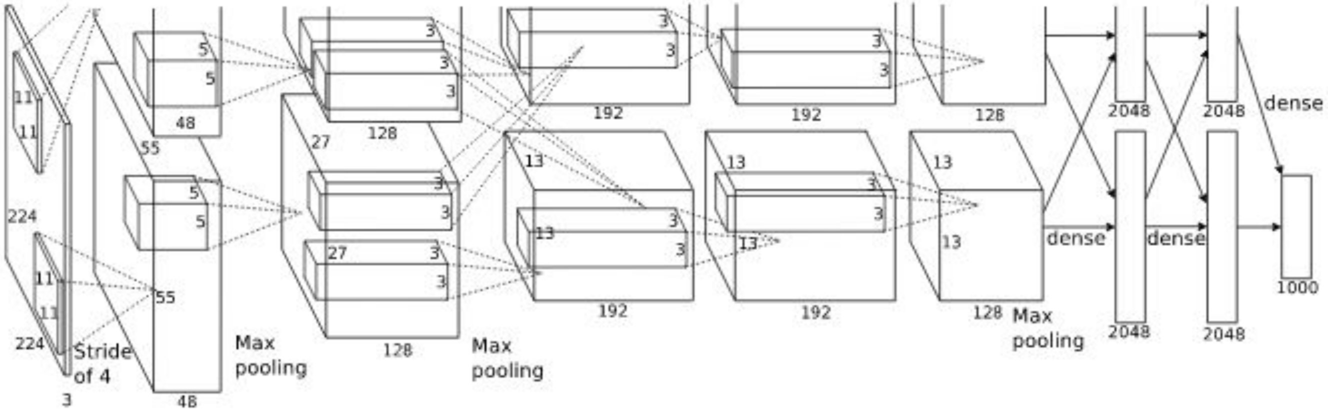


Figure 3: AlexNet architecture [6]

AlexNet

- Convolutional Neural Network (CNN)
- First five convolutional layers
- Last three fully connected layers act as classifier

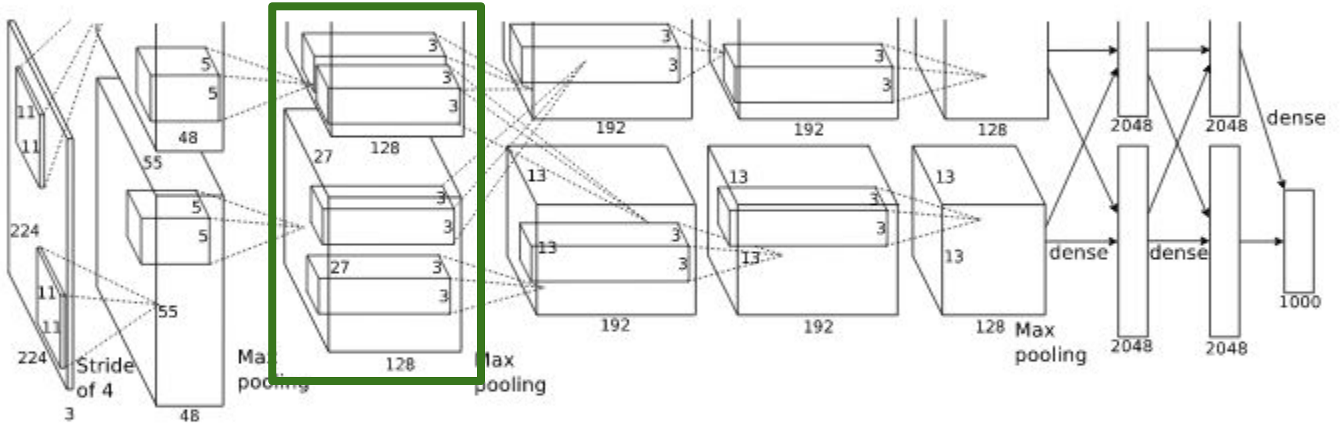


Figure 3: AlexNet architecture [6]

AlexNet Results

- Classification task on CIFAR 10 dataset
- Learning rate : 0.002
- Original rank : 256
- Approximated rank : 45
- Maximum iterations : 10

	Without CP Decomposition	With CP Decomposition
Classification accuracy	84%	65%

Future Work

- Perform hyper parameter tuning over AlexNet
- Investigate vanishing gradient problem on increase of learning rate

References

[1] Farabet, Clément, Yann LeCun, Koray Kavukcuoglu, Eugenio Culurciello, Berin Martini, Polina Akselrod, and Selcuk Talay. "Large-scale FPGA-based convolutional networks." *Scaling up Machine Learning: Parallel and Distributed Approaches* (2011): 399-419.

[2] Lebedev, Vadim, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. "Speeding-up convolutional neural networks using fine-tuned cp-decomposition." *arXiv preprint arXiv:1412.6553* (2014).

[3] Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." *Society for Industrial and Applied Mathematics (SIAM) review* 51, no. 3 (2009): 455-500.

[4] The Canonical Polyadic Tensor Decomposition and Variants for Mining Multi-Dimensional Data, https://drive.google.com/file/d/1l7GmqErogAnEeNJycvOw7IhA70A9Mcn6/view, Accessed 25-October-2020.

[5] Kronecker Product, https://en.wikipedia.org/wiki/Kronecker_product, Accessed 25-October-2020.

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105. 2012.