HOTCAKE: Higher Order TuCker Articulated KErnels for Deeper CNN Compression

Rui LIN

Nov 2020
Department of Electrical and Electronic Engineering

The University of Hong Kong

OUTLINE

Introduction

Problem: Deep neural networks are over-parameterized

Large and complicated deep neural networks (DNNs), especially convolutional neural networks (CNNs), can work well on the ever-growing datasets nowadays, but the **over-parameterization problem** unarguably impedes the deployment of sophisticated DNN/CNNs on **resource-limited** edge devices.

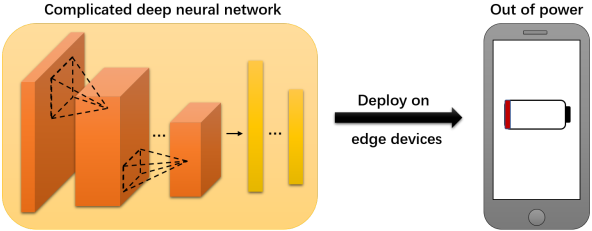
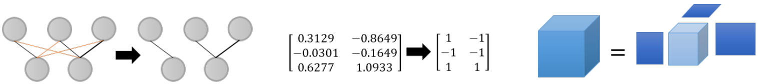


Figure 1: Over-parameterization hinders the deployment of modern DNNs on edge devices constrained by limited resources.

Introduction

Three mainstream DNN/CNN compression techniques:

1. ***Pruning*** It trims a dense network into a sparser one either by zeroing the small-weight connections or by removing entire filters and/or even layers.
2. ***Quantization*** It limits network weights and activations to be in low bit-widths for smaller storage and cheaper computation.
3. ***Low-rank Decomposition*** It decomposes the kernel tensors into low-rank factors with smaller sizes for compression.



(a) Pruning (b) Quantization (c) Low-rank decomposition

Figure 2: Three mainstream DNN compression techniques.

Tensor Basic

Tensor Basic: Tensor Network Diagram

Tensors are **multi-way** arrays that generalize vectors (viz. one-way tensors) and matrices (viz. two-way tensors) to their higher order counterparts. Figure 2 shows the so-called **tensor network diagram** for these data structures where an open edge stands for an index axis.

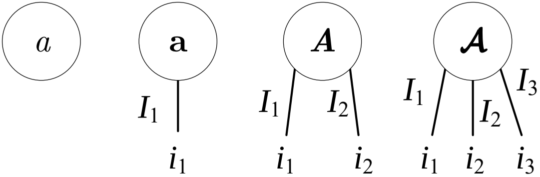


Figure 3: Graphical representation of a scalar a, vector $\boldsymbol{a}$, matrix $\boldsymbol{A}$, and third-order tensor $\boldsymbol{\mathcal{A}}$.

Tensor Basic: Tucker-2 Decomposition

Definition Tucker decomposition represents a d-way tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ as the full multilinear product of a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$ and a set of factor matrices $\mathbf{U}^{(k)} = [u_1^{(k)}, \cdots, u_{R_k}^{(k)}]$ for $k = 1, 2, \cdots, d$,

$$\mathcal{A} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_d=1}^{R_d} \mathcal{G}(r_1, \cdots, r_d)(u_{r_1}^{(1)} \circ \cdots \circ u_{r_d}^{(d)})$$
$$= \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_d \mathbf{U}^{(d)},$$

where $r_1, r_2, \cdots, r_d$ are auxiliary indices that are summed over, and $\circ$ denotes the outer product. The dimensions $(R_1, R_2, \cdots, R_d)$ are called the Tucker ranks.

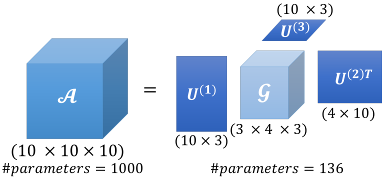


Figure 4: Tucker decomposition on a 3-way tensor can reduce the number of parameters significantly.

HOTCAKE

Regular Convolution

Figure 5 illustrates through tensor network diagram how convolution is done via a particular kernel (filter) producing the k_2th slice in the output tensor (a.k.a. feature map). The convolution operation is denoted by the symbol $\circledast$.

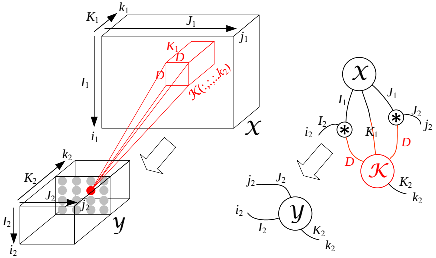


Figure 5: Convolution with the input tensor and the kernels.

Tucker-2 Decomposition and 3-stage Convolution

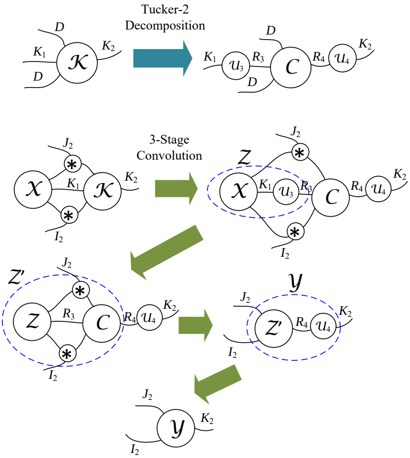


Figure 6: (Upper) Tucker-2 decomposition of kernel tensor and (Lower) the three successive, smaller size convolutions marked by blue dashed circles. Some obvious dimensions are omitted in the figure for brevity.

Motivation

Tucker-2 adopts a 4-way view of the convolutional kernel tensor. HOTCAKE is along the line of tensor decomposition and recognizes the unexploited rooms for deeper compression by ***going beyond 4-way***.

	Tucker-2	HOTCAKE
Data dimension	4-way	beyond 4-way
Rank selection	VBMF[1]	extended VBMF
Decomposition algorithm	HOSVD[2]	HOSVD with rSVD[3]

Table 1: Comparison between Tucker-2 and HOTCAKE.

[1] Variational Bayesian Matrix Factorization
[2] Higher Order Singular Value Decomposition
[3] Random Singular Value Decomposition

HOTCAKE: Input Channel Decomposition

Example 1 Suppose a convolution layer of kernel tensor $\mathcal{K} \in \mathbb{R}^{3\times3\times128\times256}$. In this case, the number of input channels is $K_1 = 128$, which can be decomposed into several branches of dimensions K_{1i}'s with $K_1 = \prod_i K_{1i}$, such as $K_{12} = 16$ and $K_{11} = 8$. These K_{1i}'s can be determined according to the estimated number of clusters of filters. Empirically, it is found that it works best when $K_{1j} \geq K_{1i}, \forall j \geq i$.

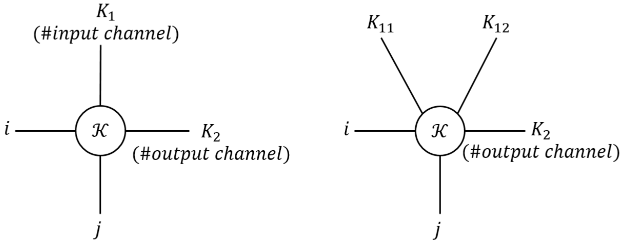


Figure 7: Input channel decomposition.

HOTCAKE: Tucker Rank Selection

Example 2 Given a kernel tensor $\mathcal{K} \in \mathbb{R}^{3\times3\times128\times256}$, suppose the input channel decomposition makes it a $\mathcal{K}_{new} \in \mathbb{R}^{3\times3\times8\times16\times256}$ by decomposing its #inputs axis into 2 branches. Assuming selected VBMF ranks of $\mathcal{K}_{new}$ being $(R_{31}, R_{32}, R_4) = (5, 7, 107)$ and a search diameter of 3, the rank search space in our algorithm is then $\{(R_{31}, R_{32}, R_4)|[4,5,6] \times [6,7,8] \times [106,107,108]\}$, containing 27 different combinations.

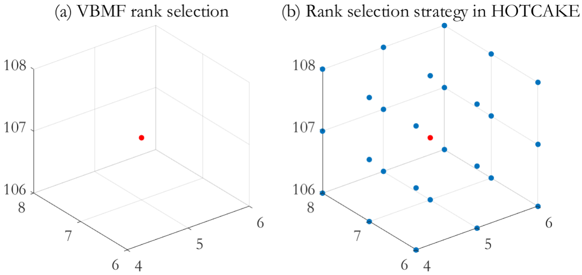


Figure 8: Tucker rank selection strategy in HOTCAKE.

HOTCAKE: Higher-Order Tucker Compression

Here, we employ truncated higher-order singular value decomposition (HOSVD) with rSVD in place of SVD to avoid the $\mathcal{O}(n^3)$ computational complexity.

Procedure 1 Modified truncated higher-order singular value decomposition (HOSVD)

Require: Tensor $\mathcal{K}_{new} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, ranks: $R_1, \ldots, R_d$.
Ensure: Core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$, factor matrices $\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}$, where $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for $k = 1, \ldots, d$.

for $n = 1, 2, \ldots, d$ **do**
  $[\boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R}^T] \leftarrow$ rSVD decomposition of $\boldsymbol{K}_{new(n)}$
  $\boldsymbol{U}^{(n)} \leftarrow R_n$ leading left columns of $\boldsymbol{L}$
end for
$\mathcal{G} \leftarrow [[\mathcal{K}_{new}; \boldsymbol{U}^{(1)T}, \ldots, \boldsymbol{U}^{(d)T}]]$

HOTCAKE: #params and Time Complexity

	Original	HOTCAKE
# Parameters	$\mathcal{O}(D^2K_1K_2)$	$\mathcal{O}(D^2R_{3i}^l + lR_{3i}K_{1i} + K_4R_4)$
Time Complexity	$\mathcal{O}(M^2D^2K_1K_2)$	$\mathcal{O}(M^2(D^2R_{3i}^l + lR_{3i}K_{1i} + K_4R_4))$

Table 2: Number of parameters and the time complexity of the CONV layer before and after Higher-order Tucker Compression.

In Table 2, R_{3i} and K_{1i} are the largest values in $R_{31}, R_{32}, \ldots, R_{3l}$ and $K_{11}, K_{12}, \ldots, K_{1l}$, respectively. The capital M is the output feature height or width value. It is worth noting that a huge computational complexity reduction can be achieved through HOTCAKE.

Experimental Results

Experimental Results: SimpNet

SimpNet is a lightweight CNN, Table 3 shows the overall result. We notice that Tucker-2 and HOTCAKE achieve **similar classification accuracy** after fine-tuning, while HOTCAKE produces a **more compact** model.

	Original	Tucker-2	HOTCAKE
Testing Accuracy	95.21%	90.84%	90.95%
Overall Parameters	$5.48M$	$2.24M$	$1.75M$
Compression Ratio	-	$2.45\times$	$3.13\times$

Table 3: An overview of SimpNet's performance and the number of parameters before and after compression.

Experimental Results: SimpNet

1. Figure 9 shows the classification accuracy of the compressed model obtained by employing HOTCAKE when increasing the number of compressed layers.
2. The sequence we compress the layer is determined by their compression ratios listed in Table 4.
3. Employing this strategy, we can achieve the highest classification accuracy when the overall model compression ratio is given.

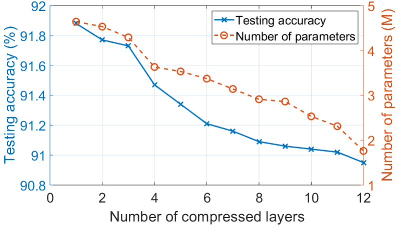


Figure 9: Classification accuracy and model parameters vs. the number of compressed CONV layers.

Experimental Results: SimpNet

Table 4 shows SimpNet's layer-wise analysis.

CONV layer#	Original	Tucker-2	HOTCAKE
2	$76K$	$30K(2.53\times)$	$24K$ $(3.17\times)$
3	$147K$	$61K$ $(2.41\times)$	$39K$ $(3.77\times)$
4	$147K$	$61K$ $(2.41\times)$	$43K$ $(3.42\times)$
5	$221K$	$88K$ $(2.72\times)$	$65K$ $(3.40\times)$
6	$332K$	$136K$ $(2.44\times)$	$103K$ $(3.22\times)$
7	$332K$	$137K$ $(2.42\times)$	$92K$ $(3.61\times)$
8	$332K$	$137K$ $(2.42\times)$	$104K$ $(3.19\times)$
9	$332K$	$135K$ $(2.46\times)$	$112K$ $(2.96\times)$
10	$498K$	$206K$ $(2.42\times)$	$162K$ $(3.07\times)$
11	$746K$	$314K$ $(2.37\times)$	$183K$ $(4.08\times)$
12	$920K$	$371K$ $(2.48\times)$	$257K$ $(3.58\times)$
13	$1.12M$	$569K$ $(1.97\times)$	$569K$ $(1.97\times)$

Table 4: SimpNet's layer-wise analysis. Numbers in brackets are compression ratios compared with the original CONV layers.

Experimental Results: MTCNN

MTCNN is designed for human face detection. It contains **three cascaded neural networks** called P-Net, R-Net and O-Net.The first two are too small such that we do not have much space to compress them. Therefore, **we compress only the O-Net**.

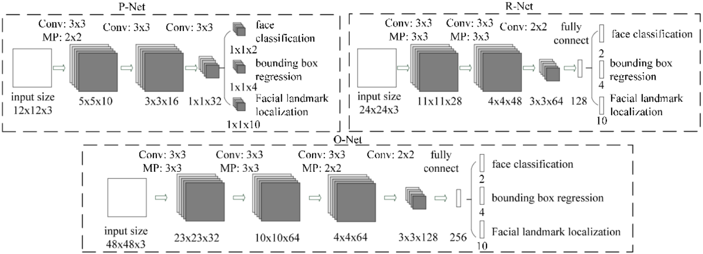


Figure 10: MTCNN is a lightweight network cascaded by P-Net, R-Net and O-Net†.

† Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

Table 5 shows the overall model compression results employing HOTCAKE. We achieved **at least** $3\times$ compression ratio on all the three CONV layers even though the original layer sizes are **already small enough.**

CONV layer#	Original	HOTCAKE
2	$18K$	$4K$ $(4.50\times)$
3	$37K$	$8K$ $(4.63\times)$
4	$33K$	$11K$ $(3.00\times)$

Table 5: O-Net's Layer-wise analysis. Numbers in brackets are compression ratios.

Experimental Results: MTCNN

Table 6 further illustrates the detailed performance of the compressed model. The performance of the MTCNN compressed by HOTCAKE is **almost the same** as the original one.

	Original	HOTCAKE
Face Classification Accuracy	95.36%	94.42%
Loss of Face Detection	0.648	0.686
Loss of Bounding Box	0.0137	0.0175
Loss of Face Landmarks	0.0107	0.0128
Total loss	0.546	0.569

Table 6: Performances of MTCNN before and after compression.

Experimental Results: AlexNet

AlexNet is **much larger** than the above two examples. Table 6 shows the layer-wise analysis of AlexNet. We observe that HOTCAKE can achieve **higher** compression ratio for **each** layer.

CONV layer#	Original	Tucker-2	HOTCAKE
2	307K	127K (2.42×)	56K (5.48×)
3	664K	197K (3.37×)	120K (5.53×)
4	885K	124K (7.14×)	51K (17.35×)
5	590K	71K (8.31×)	34K (17.35×)

Table 7: AlexNet's layer-wise analysis. Numbers in brackets are compression ratios compared with the original CONV layers.

Experimental Results: AlexNet

Table 8 further shows classification performance of the compressed models. Tucker-2 obtains a higher accuracy when its **compression ratio is half less than HOTCAKE**.

	Original	Tucker-2	HOTCAKE
Testing Accuracy	90.86%	90.29%	83.17%
Overall Parameters (CONV layers)	2.47M	520K	261K
Compression Ratio	——	$4.75\times$	$9.37\times$

Table 8: An overview of AlexNet's performance and number of parameters before and after compression.

Experimental Results: AlexNet

- To make the comparison fair, we further set ranks manually for Tucker-2 to reach the same compression ratio as HOTCAKE, and its classification accuracy drops from 90.29% to 81.39%, which is lower than that of HOTCAKE (83.17%).
- We assign ranks for both Tucker-2 and HOTCAKE, to reach higher compression ratios at around $12\times$, $14\times$ and $16\times$. The results indicates the superiority of HOTCAKE over Tucker-2 in high compression ratios.

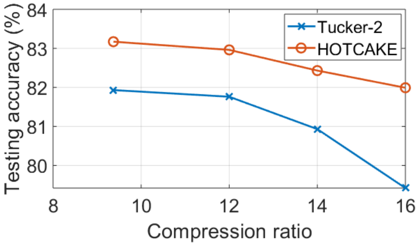


Figure 11: Accuracy vs. compression ratio on CIFAR-10.

Summary

Summary

1. HOTCAKE can compress not only ***bulky*** CNNs, but also ***compact and portable*** network models.
2. HOTCAKE reaches ***higher*** compression ratios with a ***graceful decrease*** of accuracy.
3. HOTCAKE can be selectively used for ***a better trade-off*** between accuracy and the number of parameters.
4. HOTCAKE is powerful yet flexible to be ***jointly*** employed with pruning and quantization.

Thanks!