Data-Driven Tensor Decomposition: A New Approach to Compressing CNNs

Alper KALLE

Université Paris-Saclay
CEA/LVML

Supervisor: Mohamed Tamaazousti
Encadrant: Mohamed-Oumar Ouerfelli (CEA/LVML)
`In collaboration with Theo Rudkiewicz (INRIA/LISN)`

Outline

1. Introduction
2. Background on Tensor Decomposition
3. Data-Driven Tensor Decomposition
4. Tensor Rank Selection
5. Conclusion

https://epoch.ai/blog/machine-learning-model-sizes-and-the-parameter-gap

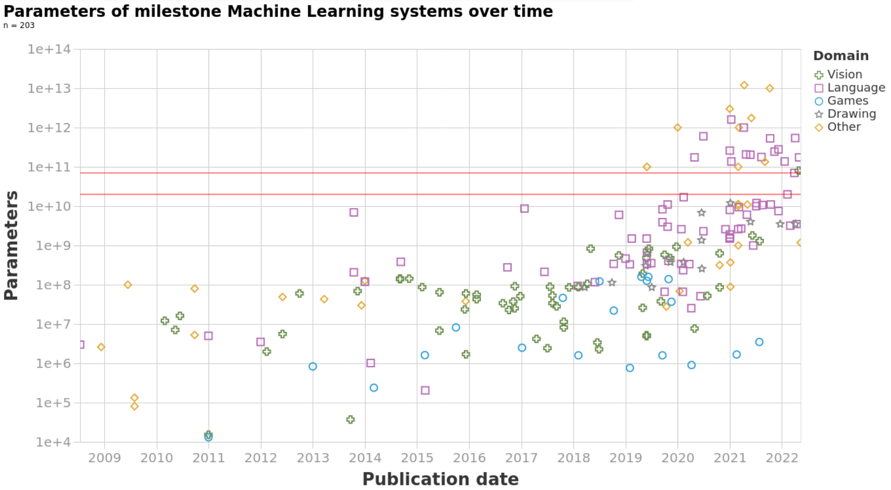


Figure 1: Evolution of the parameter counts of machine learning models year by year.

Why CNNs Remain Crucial

- **Powerful Feature Extractors**
- **Versatility**
- **Hybrid Relevance**

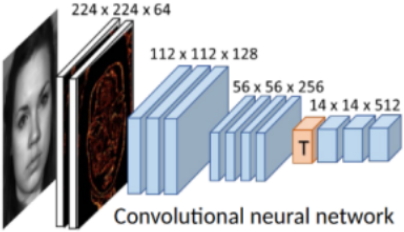


Figure 2: The layers of a CNN, where each feature map is a 3-way tensor.

Compression of CNNs

- **Neural Network Compression:** Essential for deploying deep learning models on resource-constrained embedded systems.
- **Improved Generalization:** Compressing a larger, more complex Deep Neural Network (DNN) often yields better performance than training a smaller model from scratch [1].
- **Combined Techniques:**
  - Pruning / Sparsification
  - Quantization
  - Knowledge Distillation
  - Low-Rank Factorization (Matrix and/or Tensor)

Tensor Decomposition as a Compression Method

Context: We are developing a new compression module for convolutional networks based on **Tensor Decomposition**.

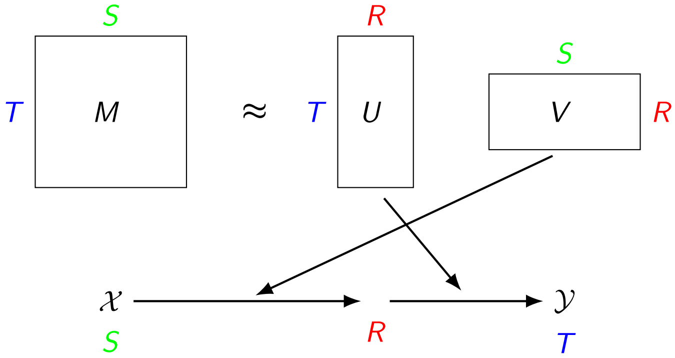
Motivation: A key advantage will be the ability to **combine** this method with other compression methods to achieve more efficient and powerful compression results.

The primary challenges:

- Reduce the network's memory footprint (size).
- Accelerate inference speed.
- Achieve this with minimal loss in accuracy.

SVD Decomposition of Linear Layers

- Approximate the weight matrix: $M \approx UV$
- Output: $\mathcal{Y} = U(V\mathcal{X})$



CP Decomposition for Convolutional Layer Compression

Original kernel: $\mathcal{K} \in \mathbb{R}^{T \times S \times H \times W}$
CP decomposition (rank-R) [2]:

$$\widetilde{\mathcal{K}} = \sum_{r=1}^{R} \boldsymbol{U}_r^{(T)} \otimes \boldsymbol{U}_r^{(S)} \otimes \boldsymbol{U}_r^{(H)} \otimes \boldsymbol{U}_r^{(W)} \approx \mathcal{K}$$

CP Decomposition for Convolutional Layer Compression

Original kernel: $\mathcal{K} \in \mathbb{R}^{T\times S\times H\times W}$
CP decomposition (rank-R) [2]:

$$\widetilde{\mathcal{K}} = \sum_{r=1}^{R} \boldsymbol{U}_r^{(T)} \otimes \boldsymbol{U}_r^{(S)} \otimes \boldsymbol{U}_r^{(H)} \otimes \boldsymbol{U}_r^{(W)} \approx \mathcal{K}$$

Parameter Reduction:

From $\boldsymbol{T} \times \boldsymbol{S} \times \boldsymbol{H} \times \boldsymbol{W}$ to $\boldsymbol{R} \times (\boldsymbol{T} + \boldsymbol{S} + \boldsymbol{H} + \boldsymbol{W})$

Data-Driven Tensor Decomposition

Question

Most tensor decomposition approaches reduces parameter counts, but ignores input data distribution. Can we improve compression by guiding tensor decomposition with the input data distribution?

Functional Norm for Tensor Decomposition

Standard Approach: Weight Matching

- **Method:** Minimize parameter distance using $\|\mathcal{K} - \widetilde{\mathcal{K}}\|_F$.
- **Interpretation:** $\left\|\mathcal{K} - \widetilde{\mathcal{K}}\right\|_F^2 = \mathbb{E}_{x \sim \mathcal{N}(0, Id)}\left(\|\mathcal{K}x - \widetilde{\mathcal{K}}x\|_F^2\right)$

Functional Norm for Tensor Decomposition

Standard Approach: Weight Matching

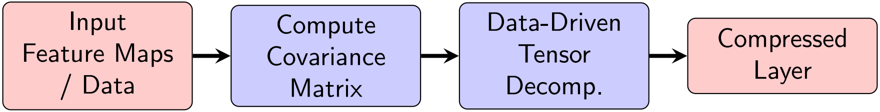
- **Method:** Minimize parameter distance using $\|\mathcal{K} - \widetilde{\mathcal{K}}\|_F$.
- **Interpretation:** $\left\|\mathcal{K} - \widetilde{\mathcal{K}}\right\|_F^2 = \mathbb{E}_{x\sim\mathcal{N}(0,Id)}\left(\|\mathcal{K}x - \widetilde{\mathcal{K}}x\|_F^2\right)$

Our Approach: Functional Preservation

- **Method:** Minimize the outputs difference directly using functional (L^2) norm including input data distribution.
- **Interpretation:** $\mathbb{E}_{x\sim\mathcal{N}(0,\Sigma^{1/2})}\left(\|\mathcal{K}x - \widetilde{\mathcal{K}}x\|_F^2\right)$ where Σ is the covariance matrix of input data. We call this Sigma norm.

ALS for CP and Tucker2 Decompositions with Sigma Norm

We propose a new ALS algorithm that incorporates the Sigma-norm for CP and Tucker-2 decompositions, called by CP-ALS-Sigma and Tucker2-ALS-Sigma.



Experimental Validation: Limited Data Access

- Target Models: Resnet18, Resnet50, GoogleNet
- Comparison of CP-ALS vs. CP-ALS-Sigma and Tucker2-ALS vs. Tucker2-ALS-Sigma.
- We consider the case where limited training data is available.
- Inference done on the ImageNet test dataset.

Resnet18: CPD vs CP Sigma - Limited Data Access

rX denotes the compression ratio, calculated by dividing the number of parameters of the original model by that of the compressed model.

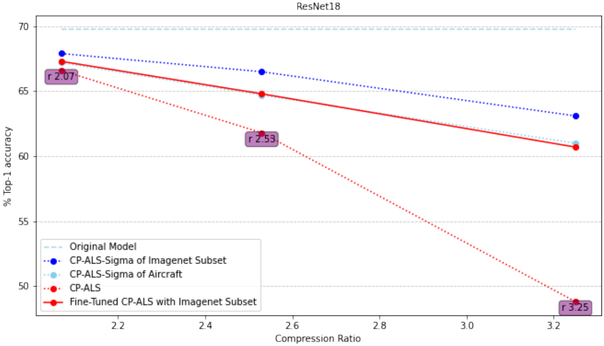


Figure 3: CP-ALS vs CP-ALS-Sigma on ResNet18.

Googlenet: CPD vs CP Sigma - Limited Data Access

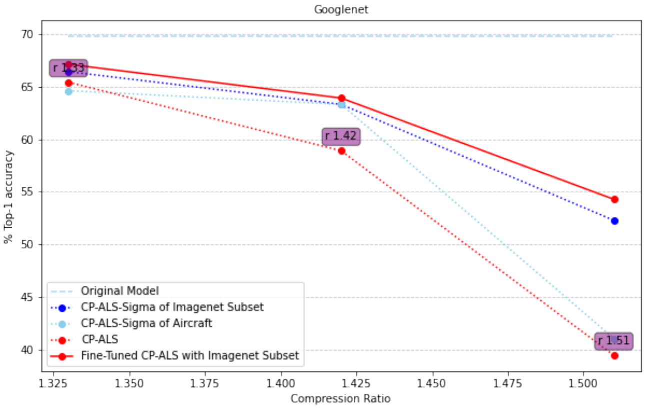


Figure 4: CP-ALS vs CP-ALS-Sigma on GoogLeNet.

Tensor Rank Selection

Challenge: Choosing tensor rank for compression with minimizing accuracy drop.

Rank = Compressibility of a layer

VBMF (Variational Bayesian Matrix Factorization [4])

- Estimate rank by thresholding singular values under a low-rank + noise model of the matricized tensor.

ALDS (Automatic Layer-wise Decomposition Selector [3])

- Estimates rank by controlling the relative approximation error obtained by spectral norm.

Conclusion and Future Work

Conclusion

Sigma algorithms:

- Outperform standard ALS and no post-compression fine-tuning needed.
- Remain effective even when using data different from the original training dataset.

Conclusion and Future Work

Conclusion

Sigma algorithms:

- Outperform standard ALS and no post-compression fine-tuning needed.
- Remain effective even when using data different from the original training dataset.

Future Work

- Optimize the rank selection
- Synergy of Tensor Decomposition and Quantization
- Extend the tensor decomposition module to large architectures like Transformers

References I

[1] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[2] Frank L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189. ISSN: 1467-9590. DOI: 10.1002/sapm192761164. (Visited on 02/14/2023).

[3] Adarsh Lakshmanan et al. "Compressing Neural Networks: Towards Determining the Optimal Layer-wise Decomposition". In: *Advances in Neural Information Processing Systems 34*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 23380–23391. URL: https://proceedings.neurips.cc/paper/2021/file/b21f9293-BibTeX.bib.

References II

[4] Shinichi Nakajima, Masashi Sugiyama, and Ryota Tomioka. "Global Analytic Solution for Variational Bayesian Matrix Factorization". In: *Advances in Neural Information Processing Systems* 23 (2010).

[5] Ledyard R. Tucker. "Some Mathematical Notes on Three-Mode Factor Analysis". In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311. ISSN: 1860-0980. DOI: 10.1007/BF02289464. (Visited on 02/14/2023).

Appendix: Tucker Decomposition for Convolutional Layer Compression

Tucker2 decomposition [5]:

$$\widetilde{\mathcal{K}} = \mathcal{G} \times_1 \boldsymbol{U}^{(T)} \times_2 \boldsymbol{U}^{(S)} \approx \mathcal{K}$$

- $\mathcal{G} \in \mathbb{R}^{R_T \times R_S \times H \times W}$: Core tensor
- $\boldsymbol{U}^{(T)} \in \mathbb{R}^{R_T \times T}$: Factor matrix along the first axis
- $\boldsymbol{U}^{(S)} \in \mathbb{R}^{R_S \times S}$: Factor matrix along the second axis
- $\times_i$ indicates a product along the i-th axis of the tensor $\mathcal{G}$

Appendix: Tucker Decomposition for Convolutional Layer Compression

Tucker2 decomposition [5]:

$$\widetilde{\mathcal{K}} = \mathcal{G} \times_1 \boldsymbol{U}^{(T)} \times_2 \boldsymbol{U}^{(S)} \approx \mathcal{K}$$

- $\mathcal{G} \in \mathbb{R}^{R_T \times R_S \times H \times W}$: Core tensor
- $\boldsymbol{U}^{(T)} \in \mathbb{R}^{R_T \times T}$: Factor matrix along the first axis
- $\boldsymbol{U}^{(S)} \in \mathbb{R}^{R_S \times S}$: Factor matrix along the second axis
- $\times_i$ indicates a product along the i-th axis of the tensor $\mathcal{G}$

Component-wise:

$$\widetilde{\mathcal{K}}[t, s, h, w] = \sum_{r_t=1}^{R_T} \sum_{r_s=1}^{R_S} \mathcal{G}[r_t, r_s, h, w] \boldsymbol{U}^{(T)}[r_t, t] \boldsymbol{U}^{(S)}[r_s, s]. \tag{6.1}$$

Appendix: Tucker Decomposition for Convolutional Layer Compression

Convolution operations based on Tucker-2 decomposed kernel:

$$X_1 = \sum_{s=1}^{S} \boldsymbol{U}^{(S)}[r_s, s] \cdot \mathcal{X}[s, y+h, x+w] \longrightarrow 1 \times 1 \text{ convolution}$$

$$X_2 = \sum_{r_s=1}^{R_S} \sum_{h=-h_d}^{h_d} \sum_{w=-w_d}^{w_d} \mathcal{G}[r_t, r_s, h, w] \cdot X_1 \longrightarrow H \times W \text{ convolution}$$

$$\mathcal{Y}[t, y, x] = \sum_{r_t=1}^{R_T} \boldsymbol{U}^{(T)}[r_t, t] \cdot X_2 \longrightarrow 1 \times 1 \text{ convolution}$$

Appendix: Functional Norm for Tensor Decomposition

Proposition

Let's define $\Sigma := \mathbb{E}_{x\sim\mathcal{D}}\left(u(p(x))u(p(x))^\top\right)$ *where* p *is the network before the convolutional layer with kernel tensor* $\mathcal{K}$ *and* u *is the unfolding operator that transforms the image* $p(x)$ *into a matrix. Then, we have:*

$$\left\|\mathrm{Conv}_{\mathcal{K}} \circ p - \mathrm{Conv}_{\widetilde{\mathcal{K}}} \circ p\right\|_{L^2} = \left\|\left(\mathcal{K} - \widetilde{\mathcal{K}}\right)_{(1)} \Sigma^{1/2}\right\|_F \tag{6.2}$$

where $(\cdot)_{(1)}$ *is the reshaping of the convolution kernel into* $(T, S \times H \times W)$.