Distribution-aware tensor decomposition for compression of convolutional neural networks

Alper Kalle, Théo Rudkiewicz, Mohamed-Oumar Ouerfelli, Mohamed Tamaazousti

Distribution-Aware Tensor Decomposition for Compression of Convolutional Neural Networks

Alper Kalle[*†]
alper.kalle@cea.fr

Theo Rudkiewicz[*‡]
theo.rudkiewicz@inria.fr

Mohamed-Oumar Ouerfelli[†]
mohamed-oumar.ouerfelli@cea.fr

Mohamed Tamaazousti[†]
mohamed.tamaazousti@cea.fr

Abstract

Neural networks are widely used for image–related tasks but typically demand considerable computing power. Once a network has been trained, however, its memory- and compute-footprint can be reduced by compression. In this work, we focus on compression through tensorization and low-rank representations. Whereas classical approaches search for a low-rank approximation by minimizing an isotropic norm such as the Frobenius norm in weight-space, we use data-informed norms that measure the error in function space. Concretely, we minimize the change in the layer's output distribution, which can be expressed as $\|(W - \widetilde{W})\Sigma^{1/2}\|_F$ where $\Sigma^{1/2}$ is the square root of the covariance matrix of the layer's input and W, $\widetilde{W}$ are the original and compressed weights. We propose new alternating least square algorithms for the two most common tensor decompositions (Tucker-2 and CPD) that directly optimize the new norm. Unlike conventional compression pipelines, which almost always require post-compression fine-tuning, our data-informed approach often achieves competitive accuracy without any fine-tuning. We further show that the same covariance-based norm can be transferred from one dataset to another with only a minor accuracy drop, enabling compression even when the original training dataset is unavailable. Experiments on several CNN architectures (ResNet-18/50, and GoogLeNet) and datasets (ImageNet, FGVC-Aircraft, Cifar10, and Cifar100) confirm the advantages of the proposed method.

1 Introduction

For the past decade, neural networks have consistently outperformed other algorithms across a wide array of tasks particularly in computer vision, including character recognition, image classification, object detection, image segmentation, and depth estimation. Notably, even with the rise of other architectures, Convolutional Neural Networks (CNNs) continue to push the boundaries of performance in computer vision, with recent advancements demonstrating their sustained competitiveness for the state-of-the-art results [Liu et al., 2022, Woo et al., 2023]. A defining characteristic of these networks, in contrast to traditional algorithms (e.g., nearest neighbors, decision trees), is their substantial number of parameters. For instance, prominent architectures such as AlexNet [Krizhevsky et al., 2017], VGG-16 [Simonyan and Zisserman, 2015], ResNet [He et al., 2016] and GoogleNet [Szegedy et al., 2015] possess tens of millions of trainable parameters.

[*]These authors contributed equally
[†]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[‡]TAU team, LISN, Université Paris-Saclay, CNRS, Inria, 91405, Orsay, France

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Despite the performance benefits often associated with a large parameter count during training, opportunities exist for creating more compact representations of trained models. This is particularly critical for applications requiring efficient execution on resource-constrained embedded devices, such as on-device image classification for smartphones and systems for autonomous vehicles. Consequently, a significant body of research has emerged, focusing on techniques to reduce the parameter count in neural networks [Cheng et al., 2018], including methods like quantization, pruning, knowledge distillation, and low-rank factorization.

Classical approaches to low-rank tensor approximation for network compression typically involve finding a compressed representation that minimizes an isotropic norm, such as the Frobenius norm, of the difference between the original and compressed weights. This minimization is performed directly in the weight space of the network. While mathematically convenient, such an approach does not explicitly consider the impact of weight changes on the network's functional behavior or its output given the data it processes. Consequently, these methods often lead to a notable drop in accuracy, necessitating a subsequent, computationally intensive fine-tuning stage to recover performance. This fine-tuning step typically requires access to the original training dataset, which may not always be available.

In this paper, we formalize a paradigm shift from classical weight-space error minimization for tensorized layers. Traditional tensorization approaches replace a computational block, typically a single layer f_θ parameterized by θ, with a sequence of more compact layers—for instance, a composition $f_{A,B,C} := g_A \circ h_B \circ i_C$, where A, B, C are the new, smaller parameter tensors. Classical methods then seek to find A, B, C such that their effective combined parameters are close to the original θ i.e. $\|r(A, B, C) - \theta\|_F$ is small (where r is the operator that combines the low rank tensors). In contrast, the data-informed approach aims to ensure that the function implemented by the sequence of compressed layers, $f_{A,B,C}$, closely approximates the original layer's function, f_θ, for the actual data the network encounters. This means we measure the error directly in the function space $\|f_{A,B,C} - f_\theta\|_{\mathsf{L}^2}$, prioritizing the preservation of the layer's input-output behavior rather than just its parameter values. This make it a functional norm and distribution-aware norm. This leads to properly justify the "data norm" that was used by Denton et al. [2014] in the case of SVD decomposition.

To operationalize this concept, we develop novel Alternating Least Squares (ALS) algorithms specifically designed to optimize this new data-informed norm for two of the most widely used tensor decomposition formats: Tucker-2 and Canonical Polyadic Decomposition (CPD). A significant advantage of our approach is its ability to achieve competitive accuracy levels often without any post-compression fine-tuning, thereby streamlining the compression pipeline and reducing its dependency on extensive retraining.

Furthermore, we demonstrate a crucial property of our covariance-based norm: its transferability. We show that the input covariance statistics learned from one dataset can be effectively applied to compress models for different, related datasets with only a minor degradation in accuracy. This finding is particularly valuable as it opens up possibilities for compressing models even when the original training data is inaccessible due to privacy, proprietary restrictions, or other constraints.

Our contributions are:

1. The use of distribution-informed norm for the neural network tensor decomposition.

2. Development of new ALS algorithms for Tucker-2 and CPD tensor decompositions that directly optimize this distribution-informed norm.

3. In depth comparison of tensor decomposition with Tucker and CP decompositions with and without the knowledge of data distribution, and also comparing two different approach: tensor deflation and CP-ALS algorithms with respect to distribution-informed norm.

4. Empirical evidence showing that our approach often achieves competitive accuracy without fine-tuning, and that the learned covariance-based norm can be transferred across datasets.

We validate our proposed methods through comprehensive experiments on several prominent CNN architectures, including ResNet-18, ResNet-50, and GoogLeNet, across diverse benchmark datasets such as ImageNet, FGVC-Aircraft, Cifar10, and Cifar100. The results consistently underscore the advantages of our distribution-informed compression strategy.

2 Related work

Our work intersects three active threads in neural-network compression: tensor-based methods, functional norms, and data-free compression techniques.

Tensorization Tensor decomposition reduces both parameter count and computational cost by approximating weight tensors with low-rank formats. Early efforts include Denton et al. [2014]; subsequent milestones span CP decomposition for CNNs [Lebedev et al., 2015], Tensor-Train for fully connected layers [Novikov et al., 2015], Tucker decomposition [Kim et al., 2016], and Tensor-Ring networks [Zhao et al., 2019]. Most of these methods follow a two-step pipeline: (i) compute a low-rank approximation, then (ii) fine-tune to restore accuracy. Several studies instead alternate tensorization with fine-tuning—e.g., the tensor-power method for successive layer compression [Astrid and Lee, 2017] and the MUSCO pipeline, which repeatedly compresses and fine-tunes [Gusak et al., 2019]. Rank selection has also been explored: variational Bayesian matrix factorization (VBMF) offers a data-free criterion, applied to Tucker layers [Kim et al., 2016] and later to CP layers [Astrid et al., 2018], building on the analytic VBMF solution of Nakajima et al. [2010].

Functional norms Most of the above techniques minimise error in *weight space*, which is not directly linked to a network's functional behaviour. Functional norms instead measure distances between functions. They underpin natural-gradient optimisation [Ollivier, 2017, Schwencke and Furtlehner, 2024] and can guide network growth [Verbockhaven et al., 2024]. In compression, alternative norms are less common; one example is Lohit and Jones [2022], who employ an optimal-transport loss during distillation. Another important example is Denton et al. [2014] who propose the use of two non-isotropic norms: the Mahalanobis distance and the data distance. The latest was not presented as a functional norm but, as we will show later, their data norm can in fact be seen as the $L2$ functional norm. For this data distance, they proposed replacing the approximation criterion from the Frobenius norm with a data covariance distance metric, which accounts for the empirical covariance of the input layer. They recommended using this criterion alongside either the SVD method or a greedy CP algorithm, which iteratively computes rank-one tensors for approximate CP decomposition and conducted experiments for this distance using the SVD method. Furthermore, to the best of our knowledge, we are the first to conduct multiple experiments on several classical CNNs across different datasets to assess the benefits of this new norm. Finally, we also expand the analysis to evaluate the transferability of this norm between different datasets.

Data-free compression Several mainstream compression techniques now operate without access to the original training data. Data-free knowledge distillation was introduced by Lopes et al. [2017]. Zero-shot quantization approaches such as ZeroQ [Cai et al., 2020] and DFQ [Nagel et al., 2019] remove the need for calibration data. For pruning, Srinivas and Babu [2015] merge redundant neurons, Tang et al. [2021] prune using synthetically generated data, and SynFlow prunes networks even before training [Tanaka et al., 2020]. To the best of our knowledge, no existing work tackles the data-free high order tensor approximation problem. We address this by evaluating the use of a generic distinct dataset to the one used during model training.

Alternative Norms Defining an importance score, or norm, is central to structured network compression. Prevailing methods rely on simple weight magnitude or more complex, loss-based metrics derived from the Hessian Matrix [LeCun et al., 1989] or Fisher Information Matrix [Tu et al., 2016]. Our work explores an alternative based on activation statistics. Specifically, our distribution-aware norm uses the activation covariance matrix to model the complete second-order statistical structure of feature maps. This provides a richer measure of importance than common activation-based heuristics, which are often limited to first-order statistics like mean magnitude [Rhu et al., 2018]. By capturing inter-channel dependencies, our norm offers a more holistic criterion for guiding the compression.

3 Background on compression by tensor decomposition

The general principle of tensor decomposition is to replace a part of the neural network by a lighter one. For example if we have a neural network $f = q \circ l \circ p$ where q and p are two neural networks and l is one layer of the neural network, we can replace l by a new squence of layers $l' := l_1 \circ l_2 \circ \cdots \circ l_n$. The new neural network is then $f' := q \circ l' \circ p$. The goal of the tensor decomposition is to find the best

l' such that f and f' are as close as possible. In most of the case the layer l is either a fully connected layer or a convolutional layer. In the case of a fully connected layer, the tensor decomposition is done through the singular value decomposition (SVD) and lead to replace the layer by a couple of fully connected layers. In the case of a convolutional layer, the tensor decomposition can be done through the CP decomposition or the Tucker decomposition as described in the next section.

Convolution A convolution $\mathbf{Conv}_{\mathcal{K}}$ parameterized by a tensor $\mathcal{K}$ of size (T, S, H, W) is a function from the space of images with S channels like $\mathcal{X} \in \mathbb{R}^{S \times H_y \times W_x}$ where H_y, W_x represent the height and width of the image to the space of images with T channels like $\mathcal{Y} \in \mathbb{R}^{T \times H'_y \times W'_x}$. The convolution layer has $SHWT$ parameters and the cost of the convolution operation given above is $(SHWT)(H'_y W'_x)$ additions-multiplications.

By decomposing the kernel tensor we can replace the convolution by a series of smaller layers in terms parameters and computational cost. In the next sections, we present two possible decompositions that can be seen as generalizations of singular value decomposition (SVD) to the tensor case.

3.1 CP Decomposition for Convolutional Layer Compression

The CANDECOMP/PARAFAC (CP) decomposition [Hitchcock, 1927] represents a tensor as sum of rank 1 tensors, where the rank 1 tensors are them self decomposed as the outer product of vectors. For a rank R, CP decomposition of a kernel tensor $\mathcal{K} \in \mathbb{R}^{T \times S \times H \times W}$ is given by the following formula:

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

where $\boldsymbol{U}_r^{(n)} \in \mathbb{R}^n$ for $n \in [T, S, H, W]$ are the equivalent of the singular vectors and $\otimes$ denotes the outer product.

This decomposition leads as suggested by Lebedev et al. [2015] to replace the convolution by a series of four layers: a 1×1 convolution parameterized by $(\boldsymbol{U}_r^{(S)})_r$, a $H \times 1$ convolution parameterized by $(\boldsymbol{U}_r^{(H)})_r$, a $1 \times W$ convolution parameterized by $(\boldsymbol{U}_r^{(W)})_r$ and a 1×1 convolution parameterized by $(\boldsymbol{U}_r^{(T)})_r$. The number of parameters of the CP decomposition is $R \times (T + S + H + W)$ which for a small rank R is smaller than the number of parameters of the original convolutional layer; the reduction factor is the same for the computational cost.

To choose a rank for the decompostion without reliying on trial and errors, we use the VBMF (Variational Bayesian Matrix Factorization) algorithm as suggested by Astrid et al. [2018]. We note R_{VBMF} the maximum of ranks computed through application of VBMF to the each possible unfolding of the kernel tensor and $R_{\max}$ an upper bound of the rank with $R_{\max} := \frac{TSHW}{\max\{T,S,H,W\}}$. We use a rank R_α that is a linear combination of those ranks determined by a parameter α that we call VBMF ratio:

$$R_\alpha = R_{\text{VBMF}} + (1 - \alpha)(R_{\max} - R_{\text{VBMF}}). \tag{2}$$

3.2 Tucker Decomposition for Convolutional Layer Compression

The Tucker decomposition [Tucker, 1966] countrary to the CP decomposition use a different number of eigenvectors for each mode. In the case of convolutional layers, as the two mode corresponding to the kernel size (H and W) are small, we do not decompose them which leads to the Tucker2 decomposition. For a kernel tensor $\mathcal{K} \in \mathbb{R}^{T \times S \times H \times W}$, the Tucker2 decomposition with rank R_T and R_S is $\widetilde{\mathcal{K}} = \mathcal{G} \times_1 \boldsymbol{U}^{(T)} \times_2 \boldsymbol{U}^{(S)*}$.

As suggested by Kim et al. [2016], this leads to replace the convolution by a series of two layers: a 1×1 convolution parameterized by $\boldsymbol{U}^{(T)}$, a full convolution parameterized by $\mathcal{G}$ and a 1×1 convolution parameterized by $\boldsymbol{U}^{(S)}$.

To select the ranks R_T and R_S, we use the same method as for CP decomposition. We compute the ranks R_{T-VBMF} and R_{S-VBMF} by applying the VBMF algorithm to the unfolding matrices of sizes $T \times SHW$ and $S \times THW$. Then we compute the ranks R_T and R_S using the formula given

[*] $\times_i$ indicates a product along the ith axis of the tensor $\mathcal{G}$

in equation (2) on the VBMF ranks R_{T-VBMF} and R_{S-VBMF}, respectively (with the $R_{\max}$ being either T or S).

4 The Distribution-Aware Tensor Decomposition

4.1 Functional Metric for Network Compression

In general when replacing the layer l_θ from $f = q \circ l_\theta \circ p$ to get $f' = q \circ l'_{\theta'} \circ p$ we don't want to retrain the network so we aim to have $f \approx f'$. Mathematically for a data distribution $\mathcal{D}$, for a classification task we want to have $\mathbb{E}_{x\sim\mathcal{D}}(d(f(x), f'(x))) \approx 0$ where d is a way to measure the distance between two distributions like the Wasserstein distance or the KL divergence.

Most of the works use as proxy the Frobenius norm $\|\theta - \theta'\|_F$ where θ are the parameters of the layer l_θ and θ' are the combined parameters of the new layer $l'_{\theta'}$ (this combination is different depending on the method (see previous section)). Here we propose instead to minimize the functional norm $\|l_\theta \circ p - l'_{\theta'} \circ p\|_{\mathsf{L}^2}$ where p is the network before the layer l_θ where we take as measure the data distribution $\mathcal{D}$. Hence, we have:

$$\|l_\theta \circ p - l'_{\theta'} \circ p\|_{\mathsf{L}^2} = \sqrt{\mathbb{E}_{x\sim\mathcal{D}}\Big(\|l_\theta \circ p(x) - l'_{\theta'} \circ p(x)\|_F^2\Big)}. \tag{3}$$

To compute the $\|\cdot\|_{\mathsf{L}^2}$ in the case where l_θ is a convolution we use the following result:

Proposition 1. *Consider a distribution $\mathcal{D}$, a partial neural network p, and two convolution $\mathbf{Conv}_\mathcal{K}$ and $\mathbf{Conv}_{\widetilde{\mathcal{K}}}$ parametrized by the kernel tensor $\mathcal{K} \in \mathbb{R}^{T\times S\times H\times W}$ and $\widetilde{\mathcal{K}}$. Under reasonable assumptions [†], we can define $\Sigma := \mathbb{E}_{x\sim\mathcal{D}}\big(u(p(x))u(p(x))^\top\big)$ where u is the unfolding operator [‡] that transforms the image $p(x)$ into a matrix that can be used to compute the convolution as a matrix product. We can also define $\Sigma^{1/2}$ the square root of Σ such that $\Sigma^{1/2}(\Sigma^{1/2})^\top = \Sigma$. Then, we have:*

$$\big\|\mathbf{Conv}_\mathcal{K} \circ p - \mathbf{Conv}_{\widetilde{\mathcal{K}}} \circ p\big\|_{\mathsf{L}^2} = \left\|\Big(\mathcal{K} - \widetilde{\mathcal{K}}\Big)_{(1)} \Sigma^{1/2}\right\|_F \tag{4}$$

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

See appendix A for the proof.

Frobenius norm Here we can notice that in the case where $\Sigma = I_n$, for example if we consider that $\mathcal{D} = \mathcal{N}(0, I)$, we recover the Frobenius norm over the parameters of the layer. Hence the functional norm is a generalization of the Frobenius norm over the parameters of the layer, suited for the case of a non isotropic data distributions. In the following, we call the distribution-aware norm given in the Eq. 4 as Sigma norm.

4.2 Decomposition Computation

Although taking the input data distribution into account lead to better compression results, we cannot directly apply the highly efficient algorithms used for kernel approximation based on the Frobenius norm. In this paper, we introduce new efficient algorithms, equivalent to the widely used CP-ALS and Tucker-ALS methods for the Frobenius norm, that approximate the kernel using this new metric.

4.2.1 Alternating Least Square Algorithm for CP with Distribution-Aware Norm

The CP decomposition as presented in Eq. 1 involves 4 terms. The main idea of the ALS algorithm is to minimize successively each of the terms and then to iterate this process. The minimization of each term is convex and can be done with a close formula using the pseudo-inverse. We details below how the minimization can be done with Sigma norm.

[†] The reasonable assumptions are that the function p is in L^2 for the distribution $\mathcal{D}$. This is likely the case in our range of applications since p is a neural network that is in most cases continuous and $\mathcal{D}$ can be considered as restricted to a compact set.

[‡] `https://docs.pytorch.org/docs/stable/generated/torch.nn.Unfold.html`

Firstly, we note the properties of vectorization of matrices which we will use to adopt the Sigma norm for ALS algorithm. Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times k}$ be two matrices. Then, we can state the followings, for all $i \in [\![1, m]\!], j \in [\![1, n]\!]$,

$$\text{Vec}(A)[m(j-1)+i] = A[i,j], \text{ and} \tag{5}$$

$$\text{Vec}(AB) = (B^\top \otimes \text{Id}(m))\ \text{Vec}(A) = (\text{Id}(k) \otimes A)\ \text{Vec}(B). \tag{6}$$

Given that the first unfolding of CP decomposition is $\widetilde{\mathcal{K}}_{(1)} = \boldsymbol{U}^{(T)}\ (\boldsymbol{U}^S \odot \boldsymbol{U}^H \odot \boldsymbol{U}^W)^\top$ where $\odot$ denotes the Khatri–Rao product, we have:

$$\text{Vec}(\widetilde{\mathcal{K}}_{(1)}\Sigma^{1/2}) = \Big((\Sigma^{1/2})^\top \otimes \text{Id}(T)\Big)\ \text{Vec}(\widetilde{\mathcal{K}}_{(1)}) \tag{7}$$

$$= \Big((\Sigma^{1/2})^\top \otimes \text{Id}(T)\Big)\ \text{Vec}(\boldsymbol{U}^{(T)}\ (\boldsymbol{U}^S \odot \boldsymbol{U}^H \odot \boldsymbol{U}^W)^\top) \tag{8}$$

$$= \Big((\Sigma^{1/2})^\top \otimes \text{Id}(T)\Big)\ \Big((\boldsymbol{U}^{(S)} \odot \boldsymbol{U}^{(H)} \odot \boldsymbol{U}^{(W)}) \otimes \text{Id}(T)\Big)\ \text{Vec}(\boldsymbol{U}^{(T)}). \tag{9}$$

Thus, we can iterate over each factor matrices to minimize the error between the kernel tensor $\mathcal{K}$ and its approximation $\widetilde{\mathcal{K}}$ under Sigma norm. In other words, for the factor matrix $\boldsymbol{U}^{(T)}$ we have the following minimization problem:

$$\min_{U^{(T)}} \Big\| \text{Vec}(\mathcal{K}\Sigma^{1/2}) - \boldsymbol{P}^{(T)}\text{Vec}(\boldsymbol{U}^{(T)}) \Big\|_F, \tag{10}$$

where $\boldsymbol{P}^{(T)} = \big((\Sigma^{1/2})^\top \otimes \text{Id}(T)\big)\ \big((\boldsymbol{U}^{(S)} \odot \boldsymbol{U}^{(H)} \odot \boldsymbol{U}^{(W)}) \otimes \text{Id}(T)\big)$, and similarly for the other matrices corresponding to the components S, H, W, which are detailed in the appendix B.2.

Since the least square minimization problems have a closed form solution, we can deduce the factor matrix $U^{(T)}$ (and similarly the other factors) with the following formula:

$$\text{Vec}(U^{(T)}) \leftarrow (\boldsymbol{P}^{(T)})^\dagger \text{Vec}(\mathcal{K}\Sigma^{1/2}) \tag{11}$$

where $\boldsymbol{A}^\dagger$ denotes the Moore–Penrose inverse of the matrix $\boldsymbol{A}$. As a result, we present the full algorithm called CP-ALS-Sigma in algorithm 1 and we refer the appendix B.4 for the practical use of the algorithm 1.

Algorithm 1 CP-ALS-Sigma

function $[\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}, \boldsymbol{U}^{(W)}, \boldsymbol{U}^{(H)}, m] =$ CP-ALS-SIGMA
  Give initializations for matrices $\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}, \boldsymbol{U}^{(H)}, \boldsymbol{U}^{(W)}$
  for $n = 1, \ldots, m$ **do**
    $\text{Vec}(\boldsymbol{U}^{(T)}) \leftarrow (\boldsymbol{P}^{(T)})^\dagger\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(T)}$ with a least square solution
    $\text{Vec}(\boldsymbol{U}^{(S)}) \leftarrow (\boldsymbol{P}^{(S)})^\dagger\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(S)}$ with a least square solution
    $\text{Vec}(\boldsymbol{U}^{(H)}) \leftarrow (\boldsymbol{P}^{(H)})^\dagger\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(H)}$ with a least square solution
    $\text{Vec}(\boldsymbol{U}^{(W)}) \leftarrow (\boldsymbol{P}^{(W)})^\dagger\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(W)}$ with a least square solution
  return factor matrices $\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}, \boldsymbol{U}^{(W)}, \boldsymbol{U}^{(H)}$

4.2.2 Alternating Least Square Algorithm for Tucker2 with Distribution-Aware Norm

The principle of the ALS algorithm for Tucker2 is very similar to the CP case. We derive a quadratic minimization problem for each factor when both others are fixed. Those minimization problems can be solved in closed form using the matrix pseudo-inverse. In the end, this leads to the following algorithm 2. Full details can be found in appendix B.3 and the practical use of the algorithm 2 is provided in the appendix B.4.

5 Experimental Validation

In our experiments, we investigate the convolutional neural network models Resnet18, Resnet50, and GoogLeNet (pretrained on ImageNet 1K) to assess the performance of the proposed algorithms CP-ALS-Sigma (Algorithm 1) and Tucker2-ALS-Sigma (Algorithm 2). We do not compress the first

Algorithm 2 Tucker2-ALS-Sigma

function $[\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}, \mathcal{G}, m] =$ TUCKER2-ALS-SIGMA
  Give initializations for matrices $\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}$
  for $n = 1, \ldots, m$ **do**
    $\text{Vec}(\boldsymbol{U}^{(T)}) \leftarrow (\boldsymbol{P}^{(T)})^{\dagger}\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(T)}$ with a least square solution
    $\text{Vec}(\boldsymbol{U}^{(S)}) \leftarrow (\boldsymbol{P}^{(S)})^{\dagger}\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\boldsymbol{U}^{(S)}$ with a least square solution
    $\text{Vec}(\mathcal{G}) \leftarrow (\boldsymbol{P}^{(\mathcal{G})})^{\dagger}\ \text{Vec}(\mathcal{K}\Sigma^{1/2})$ ▷ update $\mathcal{G}$ with a least square solution
  return core tensor $\mathcal{G}$ and factor matrices $\boldsymbol{U}^{(T)}, \boldsymbol{U}^{(S)}$

convolutional layer for each model, as doing so degrades performance. Additionally, we used the Tensorly package [Kossaifi et al., 2019] to compute the standard CP and Tucker decompositions, referred to as CP-ALS and Tucker2-ALS, respectively. Some parts of the code were also adapted from the MUSCO library [Gusak et al., 2019] to perform the compression on neural networks.

5.1 Evaluation of the CP-ALS and Tensor Deflation with Distribution-Aware Norm

An alternative to the full ALS method is the tensor deflation method. In details, tensor deflation method suggests to update iteratively $W^{(k+1)} \leftarrow W^{(k)} - \alpha_k \otimes \beta_k \otimes \gamma_k \otimes \delta_k$ where $\alpha_k \otimes \beta_k \otimes \gamma_k \otimes \delta_k$ is the rank-one approximation of $W^{(k)}$ such that the low rank approximation is given by $\widetilde{\mathcal{W}} = \sum_{k=1}^{R} \alpha_k \otimes \beta_k \otimes \gamma_k \otimes \delta_k$. We compare this greedy approach with our ALS algorithm that optimize all the ranks at the same time, in the case where we optimize the distribution-aware norm. In Table 1, we show the reconstruction error for the distribution-aware norm ($\left\| \left(\mathcal{K} - \widetilde{\mathcal{K}} \right)_{(1)} \Sigma^{1/2} \right\|_F / \left\| \mathcal{K}_{(1)} \Sigma^{1/2} \right\|_F$ where $\mathcal{K}$ is the original tensor and $\widetilde{\mathcal{K}}$ is the decomposed one) when decomposing kernels of ResNet 18. We observe a clear superiority of the full ALS algorithm in reconstruction error (Table 1) and in accuracy after compression (Table 2).

Table 1: Relative reconstruction errors of Greedy Tensor Deflation(TD) and CP-ALS-Sigma algorithms applied to convolutional layers of ResNet18 where the ranks are estimated through VBMF with the ratio $\alpha = 0.8$.

Conv Layer	Rank	Recons. Err. (Greedy TD)	Recons. Err. (CP-ALS-Sigma)	Improvement (Greedy / ALS)
Layer1.0.conv2	134	0.195	**0.053**	3.7
Layer2.0.conv1	140	0.277	**0.118**	2.3
Layer3.0.conv1	291	0.217	**0.094**	2.3
Layer3.1.conv1	520	0.207	**0.070**	3.0
Layer4.1.conv1	1023	0.159	**0.063**	2.5
Layer4.1.conv2	1167	0.292	**0.051**	5.7

Table 2: Top-1 accuracies of the decomposed models obtained via Greedy Tensor Deflation (TD) and CP-ALS-Sigma algorithms with respect to distribution-aware norm applied to ResNet18 on ImageNet dataset with different compression rates.

VBMF Ratio	Compression Rate	Acc. (Greedy TD)	Acc.(CP-ALS-Sigma)
0.8	2.07	35.2	**67.9**
0.85	2.53	20.9	**66.5**
0.9	3.25	5.1	**63.1**

5.2 Model Compression and Fine-Tuning with Limited Data Access

In this section, we consider the scenario where only a limited amount of data is available, a common situation in many applications. Specifically, we consider the case where only 50,000 images from the ImageNet training set are available to estimate the matrix Σ and to fine-tune the model.

We compare the classification accuracy of the compressed model with Tucker2-ALS-Sigma, Tucker2-ALS algorithms for Frobenius norm and its fine-tuned version. For the network compressed using the standard ALS algorithm under the Frobenius norm, we fine-tune it with the Adam optimizer, selecting the optimal learning rate from the range 10^{-5} to 10^{-10}. We test the performance of our method at different compression rates by varying the parameter α in the rank formula (2). For this, we selected the following values of α: $[0.4, 0.45, 0.5, 0.55]$ for Resnet18, $[0.8, 0.9, 0.95, 1]$ for Resnet50, $[0.6, 0.7, 0.8]$ for GoogLeNet.

As shown in Figure 1, our results indicate that the Tucker2-ALS-Sigma algorithm consistently outperforms both the standard Tucker2-ALS algorithm and the fine-tuned compressed models obtained with Tucker2-ALS across all the neural networks evaluated. For additional results on the CP-ALS-Sigma algorithm, we refer to Appendix C.1.

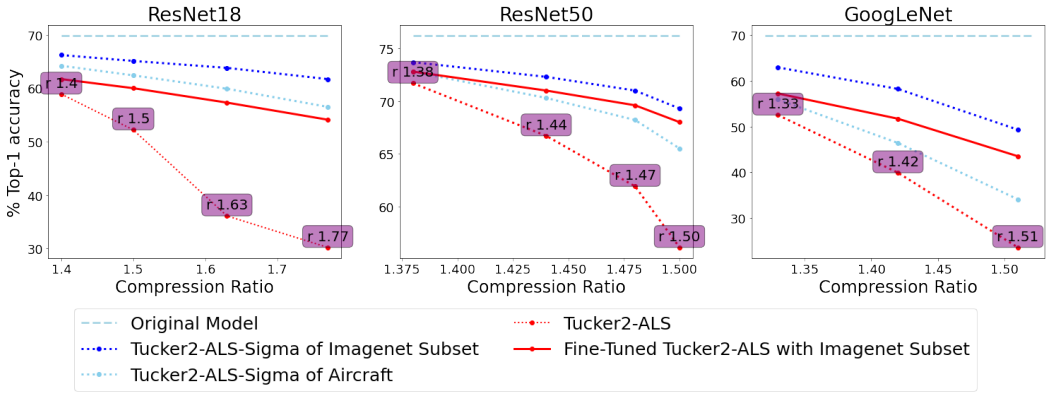

Figure 1: Accuracy comparison of decomposed models obtained with Tucker2-ALS-Sigma and Tucker2-ALS algorithms, also including fine-tuned decomposed model (with Tucker2-ALS algorithm) results where the fine-tuning done on the subset of ImageNet train dataset. (rX denotes the compression ratio, calculated by dividing the number of parameters of the original model by that of the compressed model.)

5.3 Impact of Dataset Changes on Proposed Algorithms

This section aims to assess how variations in the dataset affect the performance of the proposed CP-ALS-Sigma and Tucker2-ALS-Sigma algorithms. In particular, we demonstrate that the distribution-aware norm can be computed from a dataset different from the one used for pretraining, highlighting its transferability. To achieve this, we performed compression on ResNet18, ResNet50, and GoogLeNet models, all trained on the CIFAR-10 dataset, using our proposed ALS-Sigma algorithms (with the Σ matrix obtained from different datasets) and the standard ALS algorithm, applied to both CP and Tucker decompositions. In addition, we fine-tuned the compressed models that are obtained using the standard ALS algorithms to compare their performance with the Sigma-based methods (without fine-tuning). Fine-tuning was conducted on the CIFAR-10 training dataset, employing the Adam optimizer and testing various learning rates from 10^{-3} to 10^{-7}, selecting the best-performing learning rate. Results for the CP-ALS-Sigma algorithm can be found in Appendix C.2.

We compare Tucker2-ALS-Sigma with the standard Tucker2-ALS algorithm and the fine-tuned compressed model obtained from Tucker2-ALS. We chose α from $[0.5, 0.6, 0.7, 0.8, 0.9, 1]$ for Resnet18, $[0.9, 1, 1.1, 1.2, 1.3, 1.4]$ for Resnet50, $[0.6, 0.7, 0.8, 0.9, 0.95, 1]$ for GoogLeNet. Our results, shown in Figure 2, indicate that Tucker2-ALS-Sigma algorithm has better performance than the standard Tucker2-ALS algorithm across all models, even when the Σ matrix is computed using different datasets, such as a subset of the ImageNet training set or the CIFAR-100 training set. Additionally, while Tucker2-ALS experiences rapid performance degradation as the compression rate increases, our method remains much more consistent. Furthermore, the Tucker2-ALS-Sigma algorithm produces results that are close to those of the fine-tuned compressed model obtained using the Tucker2-ALS algorithm. Notably, the performance of the Tucker2-ALS-Sigma algorithm on the CIFAR-100 dataset is comparable to that on the CIFAR-10 dataset, suggesting that our approach is not limited to using the original training dataset for compression with a distribution-aware norm. In

fact, as illustrated in Figure 2, the results for the Tucker2-ALS-Sigma algorithm with the CIFAR-100 dataset even outperform those with CIFAR-10 for the ResNet50 model. However, when the Sigma matrix is derived from a subset of the ImageNet training set, the performance of the Tucker2-ALS-Sigma algorithm is somewhat less effective compared to when it is computed on CIFAR-10 or CIFAR-100, likely due to differences in image resolution. We refer to the Appendix G for additional experiments which investigate the effects of image resolution, dataset diversity, and the number of samples chosen for the Σ matrix on the performance of proposed algorithms.

Figure 1 demonstrates that the Tucker2-ALS-Sigma algorithm, when the Sigma matrix is derived from the FGVC-Aircraft training dataset, yields higher classification accuracy than the standard Tucker2-ALS method for all tested ImageNet models. Notably, in the case of ResNet18, the Tucker2-ALS-Sigma variant surpasses even the fine-tuned model obtained with the standard algorithm.

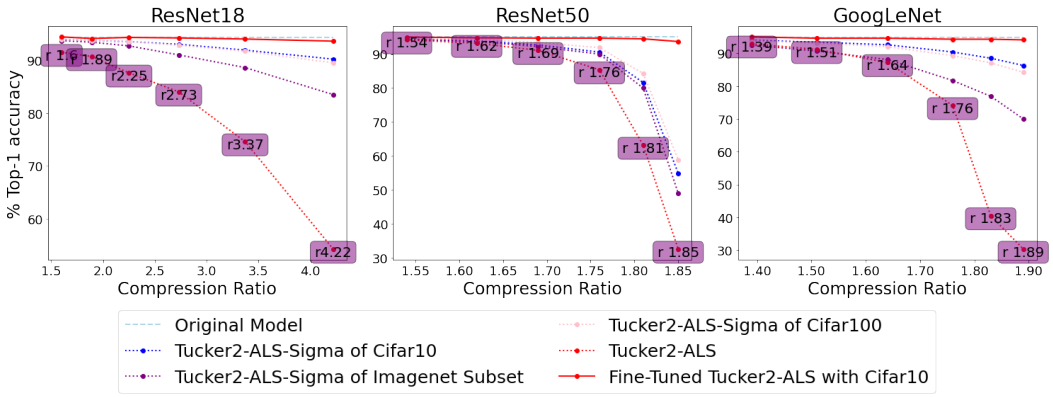

Figure 2: Accuracy comparison of Tucker2-ALS-Sigma and Tucker2-ALS algorithms including the fine-tuned model results after compression with Tucker2-ALS using Cifar10 dataset.

5.4 Evaluation Against Alternative Data-Informed Baselines

To rigorously evaluate our approach, we designed and implemented two strong, data-aware baseline methods inspired by metrics from pruning and activation analysis: Fisher information [Tu et al., 2016] and activation sparsity [Rhu et al., 2018]. Each baseline guides the Tucker2 decomposition by solving a specific Weighted Alternating Least Squares (WALS) problem, enabling a direct comparison against our proposed method.

Fisher-Weighted Low-Rank Approximation (FW-LRA) This baseline adapts Fisher information, a well-established criterion from pruning literature, to guide decomposition. The method solves the weighted objective $\min ||\mathbf{H}_{\text{fisher}} \circledast (\mathcal{K} - \widetilde{\mathcal{K}})||_F^2$, where the weighting tensor $\mathbf{H}_{\text{fisher}}$ is derived from the diagonal of the Fisher Information Matrix. The intuition is to prioritize the preservation of weights that are most sensitive to the network's loss function.

Activation-Guided Low-Rank Approximation (AG-LRA) Inspired by sparsity-based compression techniques, this baseline prioritizes filters based on their activation magnitude. It solves the objective $\min ||\mathbf{H}_{\text{act}} \circledast (\mathcal{K} - \widetilde{\mathcal{K}})||_F^2$, where the weighting tensor $\mathbf{H}_{\text{act}}$ is calculated from the mean absolute activation of each output channel. This heuristic assumes that filters with higher average activation are more critical for the network's predictions.

Comparison with Baselines Tables 3 and 4 summarize the accuracy of our Tucker2-ALS-Sigma method compared to the baseline Tucker2-ALS and the two data-aware variants on GoogLeNet and ResNet18, respectively. Across all compression rates, our method consistently outperforms both FW-LRA and AG-LRA, as well as the vanilla baseline Tucker2-ALS, demonstrating the effectiveness of our distribution-aware norm.

Table 3: Accuracy comparison of different data-aware baseline methods and Tucker2-ALS-Sigma with Tucker2-ALS at various compression rates on GoogLeNet.

Compr. Rate	Tucker2-ALS	AG-LRA	FW-LRA	Our Method
1.24	60.2	60.5	53.0	**66.3**
1.33	52.5	54.9	35.8	**64.2**
1.42	39.9	43.0	6.0	**60.1**
1.51	23.6	25.1	1.1	**52.2**

Table 4: Accuracy comparison of different data-aware baseline methods and Tucker2-ALS-Sigma with Tucker2-ALS at various compression rates on Resnet18.

Compr. Rate	Tucker2-ALS	AG-LRA	FW-LRA	Our Method
1.4	58.9	54.9	36.7	**66.8**
1.5	52.2	45.0	18.9	**66.1**
1.63	36.1	37.5	12.2	**64.9**
1.77	30.2	31.2	4.2	**63.3**

6 Limitations and Future Work

Complete functional norm A first limitation of our work is that when minimizing the reconstruction error in the layer l we only take into account the first part of the network in the functional norm. Indeed, we want to minimize $\|f - f'\|$ where $f = q \circ l_\theta \circ p$ and $f' = q \circ l'_{\theta'} \circ p$. To do so we minimize $\|l_\theta \circ p - l'_{\theta'} \circ p\|_{\mathsf{L}^2}$ as a proxy. This is an important improvement compared to the standard proxy $\|\theta - \theta'\|_F$ but this could be improved by taking the part q of the network into account in the optimization process.

Anisotropic VBMF To choose the rank of the decomposed tensors we use the Variational Bayesian Matrix Factorization (VBMF) algorithm. This algorithm is based on the assumption that the matrix we try to factorize are perturbed by a Gaussian isotropic noise. Similarly to replacing the Frobenius norm by the functional norm, we could use a more general assumption on the noise. Doing so would require to design a new algorithm to compute the VBMF rank but could lead to a better rank choice.

Performance guarantee In data-free compression, we show that our algorithm improves performance for all targeted compression ratios. However, we are not able to give a performance guarantee of the compressed network without testing it on a dataset. Hence to select the rank we rely on a priori heuristic like the VBMF rank. We believe that a possible future work would be to use the reconstruction error in the functional norm to select the rank. Indeed, preliminary experiments shown in the appendix K demonstrate that the reconstruction error in the functional norm is a way better indicator of the performance of the compressed network than the reconstruction error in the Frobenius norm.

7 Conclusion

In summary, we have shown that incorporating *distribution-aware* (functional) norms into tensor-based network compression leads to substantial performance gains. By deriving ALS procedures that directly optimize CP and Tucker decompositions under the Sigma norm, we achieve markedly lower reconstruction error and higher accuracy than with traditional Frobenius-based methods. In addition, CP-ALS-Sigma consistently surpasses greedy tensor deflation optimized with the same norm. The advantage of our approach becomes even more pronounced at higher compression rates, where standard methods degrade sharply. Remarkably, the distribution-aware decompositions recover almost all of the accuracy otherwise obtained by fine-tuning models compressed with conventional ALS, yet they require no additional training when the Σ matrix is estimated from the data distribution. Even when that distribution is learned from a smaller, dataset such as FGVC-Aircraft, the benefits persist. Finally, when the original training data are unavailable, the Σ matrix can be transferred from a related dataset, still delivering significant improvements—highlighting the practicality and robustness of our distribution-aware compression framework.

8 Acknowledgements

We thank Clément Laroudie and Charles Villard for helpful discussions that contributed to this work. This publication was made possible by the use of the FactoryIA supercomputer, financially supported by the Ile-de-France Regional Council. This work is also supported by the PEPR-IA : ANR-23-PEIA-0010 and DeepGreen : ANR-23-DEGR-0001.

References

Marcella Astrid and Seung-Ik Lee. Cp-decomposition with tensor power method for convolutional neural networks compression. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 115–118. IEEE, 2017.

Marcella Astrid, Seung-Ik Lee, and Beom-Su Seo. Rank selection of CP-decomposed convolutional layers with variational Bayesian matrix factorization. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 347–350, February 2018. doi: 10.23919/ICACT.2018.8323750. URL `https://ieeexplore.ieee.org/document/8323750/`.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. ZeroQ: A Novel Zero Shot Quantization Framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering*, 19(1):64–77, January 2018. ISSN 2095-9230. doi: 10.1631/FITEE.1700789.

Daria Cherniuk, Stanislav Abukhovich, Anh-Huy Phan, Ivan Oseledets, Andrzej Cichocki, and Julia Gusak. Quantization Aware Factorization for Deep Neural Network Compression. *Journal of Artificial Intelligence Research*, 81:973–988, December 2024. ISSN 1076-9757. doi: 10.1613/jair.1.16167.

Remi Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems 27*, pages 1269–1277. Curran Associates, Inc., 2014.

Shupeng Gui, Haotao Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. *Advances in Neural Information Processing Systems*, 32, 2019.

Julia Gusak, Maksym Kholiavchenko, Evgeny Ponomarev, Larisa Markeeva, Philip Blagoveschensky, Andrzej Cichocki, and Ivan Oseledets. Automated multi-stage compression of neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

Frank L. Hitchcock. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. ISSN 1467-9590. doi: 10.1002/sapm192761164.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications, February 2016.

Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research (JMLR)*, 20(26), 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *ICLR*, 2015.

Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

Suhas Lohit and Michael Jones. Model Compression Using Optimal Transport. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2764–2773, 2022.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-Free Knowledge Distillation for Deep Neural Networks, November 2017.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-Free Quantization Through Weight Equalization and Bias Correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.

Shinichi Nakajima, Masashi Sugiyama, and Ryota Tomioka. Global analytic solution for variational Bayesian matrix factorization. *Advances in Neural Information Processing Systems*, 23, 2010.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Yann Ollivier. True Asymptotic Natural Gradient Optimization, December 2017.

Minsoo Rhu, Mike O'Connor, Niladrish Chatterjee, Jeff Pool, Youngeun Kwon, and Stephen W. Keckler. Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 78–91, February 2018. doi: 10.1109/HPCA.2018.00017.

Nilo Schwencke and Cyril Furtlehner. ANaGRAM: A Natural Gradient Relative to Adapted Model for efficient PINNs learning. In *The Thirteenth International Conference on Learning Representations*, October 2024.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015.

Suraj Srinivas and R. Venkatesh Babu. Data-free Parameter Pruning for Deep Neural Networks. In *Procedings of the British Machine Vision Conference 2015*, pages 31.1–31.12, Swansea, 2015. British Machine Vision Association. ISBN 978-1-901725-53-7. doi: 10.5244/C.29.31. URL http://www.bmva.org/bmvc/2015/papers/paper031/index.html.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper With Convolutions. In *CVPR*, pages 1–9, 2015.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389. Curran Associates, Inc., 2020.

Jialiang Tang, Mingjin Liu, Ning Jiang, Huan Cai, Wenxin Yu, and Jinjia Zhou. Data-Free Network Pruning for Model Compression. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2021. doi: 10.1109/ISCAS51556.2021.9401109.

Ming Tu, Visar Berisha, Yu Cao, and Jae-Sun Seo. Reducing the Model Order of Deep Neural Networks Using Information Theory. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 93–98, July 2016. doi: 10.1109/ISVLSI.2016.117.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, September 1966. ISSN 1860-0980. doi: 10.1007/BF02289464.

Manon Verbockhaven, Théo Rudkiewicz, Guillaume Charpiat, and Sylvain Chevallier. Growing Tiny Networks: Spotting Expressivity Bottlenecks and Fixing Them Optimally. *Transactions on Machine Learning Research*, July 2024.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-Designing and Scaling ConvNets With Masked Autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16133–16142, 2023.

Qibin Zhao, Masashi Sugiyama, Longhao Yuan, and Andrzej Cichocki. Learning Efficient Tensor Representations with Ring-structured Networks. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8608–8612, May 2019. doi: 10.1109/ICASSP.2019.8682231.