Journal of Machine Learning Research 21 (2020) 1-21 Submitted 7/18; Published 7/20

Tensor Regression Networks

Jean Kossaifi JEAN.KOSSAIFI@IMPERIAL.AC.UK
NVIDIA & Imperial College London

Zachary C. Lipton ZLIPTON@CMU.EDU
Carnegie Mellon University

Arinbjörn Kolbeinsson AK711@IMPERIAL.AC.UK
Imperial College London

Aran Khanna ARANKHAN@AMAZON.COM
Amazon AI

Tommaso Furlanello FURLANEL@USC.EDU
University of Southern California

Anima Anandkumar ANIMA@CALTECH.EDU
NVIDIA & California Institute of Technology

Editor: Aapo Hyvarinen

Abstract

Convolutional neural networks typically consist of many convolutional layers followed by one or more fully connected layers. While convolutional layers map between high-order activation tensors, the fully connected layers operate on flattened activation vectors. Despite empirical success, this approach has notable drawbacks. Flattening followed by fully connected layers discards multilinear structure in the activations and requires many parameters. We address these problems by incorporating tensor algebraic operations that preserve multilinear structure at every layer. First, we introduce *Tensor Contraction Layers* (TCLs) that reduce the dimensionality of their input while preserving their multilinear structure using tensor contraction. Next, we introduce *Tensor Regression Layers* (TRLs), which express outputs through a low-rank multilinear mapping from a high-order activation tensor to an output tensor of arbitrary order. We learn the contraction and regression factors end-to-end, and produce accurate nets with fewer parameters. Additionally, our layers regularize networks by imposing low-rank constraints on the activations (TCL) and regression weights (TRL). Experiments on ImageNet show that, applied to VGG and ResNet architectures, TCLs and TRLs reduce the number of parameters compared to fully connected layers by more than 65% while maintaining or increasing accuracy. In addition to the space savings, our approach's ability to leverage topological structure can be crucial for structured data such as MRI. In particular, we demonstrate significant performance improvements over comparable architectures on three tasks associated with the UK Biobank dataset.

Keywords: Machine Learning, Tensor Methods, Tensor Regression Networks, Low-Rank Regression, Tensor Regression Layers, Deep Learning, Tensor Contraction

1. Introduction

Many natural datasets exhibit multi-modal structure. We represent audio spectrograms as 2nd-order tensors (matrices) with modes corresponding to frequency and time. We represent images as 3rd-order tensors with modes corresponding to width, height and the color channels. Videos are expressed as 4th-order tensors, and the signal processed by an array of video sensors can be described as a 5th-order tensor. Multilinear structure arises naturally in many medical applications: MRI images are 3rd-order tensors and functional MRI images are 4th-order tensors. Generally, a broad array of multimodal data can be naturally encoded as tensors. Tensor methods extend linear algebra to higher order tensors and are promising tools for manipulating and analyzing such data.

The mathematical properties of tensors have long been the subject of theoretical study. Previously, in machine learning, data points were typically assumed to be vectors and datasets to be matrices. Hence, spectral methods, such as matrix decompositions, have been popular in machine learning. Recently, tensor methods, which generalize these techniques to higher-order tensors, have gained prominence. One class of broadly useful techniques within tensor methods are tensor decompositions, which have been studied for a variety of applications in signal processing and machine learning (Sidiropoulos et al., 2017; Cichocki et al., 2015), data mining and fusion (Papalexakis et al., 2016), blind source separation (Cichocki et al., 2009), computer vision (Vasilescu and Terzopoulos, 2002) and learning latent variable models (Anandkumar et al., 2014).

Deep Neural Networks (DNNs) frequently manipulate high-order tensors: in a standard deep Convolutional Neural Network (CNN) for image recognition, the inputs and the activations of convolutional layers are 3rd-order tensors. And yet, to wit, most architectures output predictions by first flattening the activation tensors and then connecting to the output neurons via one or more fully connected layers. This approach presents several issues: (i) we lose multimodal information during the flattening process; and (ii) the fully connected layers require a large number of parameters.

In this paper, we propose Tensor Contraction Layers (TCLs) and Tensor Regression Layers (TRLs) as end-to-end trainable components of neural networks. In doing so, we exploit multilinear structure without giving up the power and flexibility offered by modern deep learning methods. By replacing fully connected layers with tensor contractions, we aggregate long-range spatial information while preserving multi-modal structure. Moreover, by enforcing low rank, we reduce the number of parameters needed significantly with minimal impact on accuracy.

Our proposed TRL expresses the regression weights through the factors of a low-rank tensor decomposition. The TRL obviates the need for flattening, instead leveraging the structure when generating output. By combining tensor regression with tensor contraction, we further increase efficiency. Augmenting the VGG and ResNet architectures, we demonstrate improved performance on the ImageNet dataset despite significantly reducing the number of parameters (almost by 65%). The ability to preserve the topological structure in the data is particularly crucial for prediction from MRI data. In this context, we conduct experiments for 3 different tasks (gender classification, body mass index prediction and age regression) on the UK biobank dataset, the largest available MRI dataset. There, we demonstrate superior performance with our approach and show large performance improvements for all 3 tasks. This is the first paper to present an end-to-end trainable architecture that retains the multi-modal tensor structure throughout the network.

Related work: Several recent papers apply tensor decomposition to deep learning. One notable line of application is to re-parametrize existing layers using tensor decomposition either to speed these up or reduce the number of parameters. Lebedev et al. (2015) propose using CP decomposition to speed up convolutional layers. Similarly, (Tai et al., 2015) propose to use tensor decomposition to remove redundancy in convolutional layers and express these as the composition of two convolutional layers with less parameters.

Kim et al. (2016) take a pre-trained network and apply tensor (Tucker) decomposition on the convolutional kernel tensors and then fine-tune the resulting network. Yang and Hospedales (2017) propose weight sharing in multitask learning and Chen et al. (2018) propose sharing residual units. Novikov et al. (2015) use the Tensor-Train (TT) format to impose low-rank tensor structure on weights of the fully connected layers in order to compress them. However, they still retain the fully connected layers for the output. In addition, the reshaping to arbitrary higher orders and dimensions does not guarantee that the multilinear structure is preserved. By contrast, we present an end-to-end tensorized network architecture that focuses on leveraging that structure. Many of these contributions are orthogonal to ours and can be applied together.

Despite the success of DNNs, many open questions remain as to why they work so well and whether they really need so many parameters. Tensor methods have emerged as promising tools of analysis to address these questions and to better understand the success of deep neural networks. Cohen et al. (2016), for example, use tensor methods as tools of analysis to study the expressive power of CNNs, while the follow up work (Sharir and Shashua, 2018) focuses on the expressive power of overlapping architectures of deep learning. Haeffele and Vidal (2015) derive sufficient conditions for global optimality and optimization of non-convex factorization problems, including tensor factorization and deep neural network training. Grefenstette and Sadrzadeh (2011) explore contraction as a composition operator for NLP. Other papers investigate tensor methods as tools for devising neural network learning algorithms with theoretical guarantees of convergence (Sedghi and Anandkumar, 2016; Janzamin et al., 2015a,b).

Several prior papers address the power of tensor regression to preserve natural multi-modal structure and learn compact predictive models (Guo et al., 2012; Rabusseau and Kadri, 2016; Zhou et al., 2013; Yu and Liu, 2016). However, these works typically rely on analytical solutions and require manipulating large tensors containing the data. They are usually used for small datasets or require to downsample the data or extract compact features prior to fitting the model, and do not scale to large datasets such as ImageNet.

To our knowledge, no prior work combines tensor contraction or tensor regression with deep learning in an end-to-end trainable fashion.

2. Mathematical background

Notation Throughout the paper, we define tensors as multidimensional arrays, with indexing starting at 0. First order tensors are vectors, denoted $\mathbf{v}$. Second order tensors are matrices, denoted $\mathbf{M}$ and $\mathbf{I}$ is the identity matrix. By $\mathcal{X}$, we denote tensors of order 3 or greater. For a third order tensor $\mathcal{X}$, we denote its element (i, j, k) as $\mathcal{X}_{i_1,i_2,i_3}$. A colon is used to denote all elements of a mode, e.g., the mode-1 fibers of $\mathcal{X}$ are denoted as $\mathcal{X}_{:,i_2,i_3}$.

The transpose of $\mathbf{M}$ is denoted $\mathbf{M}^\top$. Finally, for any $i, j \in \mathbb{N}, i < j, [i \,..\, j]$ denotes the set of integers $\{i, i+1, \cdots, j-1, j\}$.

Tensor unfolding Given a tensor, $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$, its mode-$n$ unfolding is a matrix $\mathbf{X}_{[n]} \in \mathbb{R}^{I_n \times I_M}$, with $I_M = \prod_{\substack{k=0,\\k\neq n}}^{N} I_k$ and is defined by the mapping from element $(i_0, i_1, \cdots, i_N)$ to (i_n, j), with $j = \sum_{\substack{k=0,\\k\neq n}}^{N} i_k \times \prod_{\substack{l=k+1,\\l\neq n}}^{N} I_l$. We use the definition introduced in Kossaifi et al. (2019), which corresponds to an underlying row-wise ordering of the elements. This differs from the definition used by Kolda and Bader (2009), which correponds to an underlying column-wise ordering of the elements. Throughout the paper, we assume the elements are arranged in a row-wise manner, which is reflected in the definitions of unfolding, vectorization, and the resulting formulas. This row-ordering elements matches the actual ordering on GPUs and allows for more efficient implementation.

Tensor vectorization Given a tensor, $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$, we can flatten it into a vector $\text{vec}(\mathcal{X})$ of size $(I_0 \times \cdots \times I_N)$ defined by the mapping from element $(i_0, i_1, \cdots, i_N)$ of $\mathcal{X}$ to element j of $\text{vec}(\mathcal{X})$, with $j = \sum_{k=0}^{N} i_k \times \prod_{m=k+1}^{N} I_m$. As for unfolding, we assume a row-ordering of the elements, following Kossaifi et al. (2019).

n-mode product For a tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$ and a matrix $\mathbf{M} \in \mathbb{R}^{R \times I_n}$, the n-mode product of a tensor is a tensor of size $(I_0 \times \cdots \times I_{n-1} \times R \times I_{n+1} \times \cdot \times I_N)$ and can be expressed using unfolding of $\mathcal{X}$ and the classical dot product as

$$\mathcal{X} \times_n \mathbf{M} = \mathbf{M}\mathcal{X}_{[n]} \in \mathbb{R}^{I_0 \times \cdots \times I_{n-1} \times R \times I_{n+1} \times \cdots \times I_N}. \tag{1}$$

Generalized inner-product For two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$ of same size, their inner product is defined as $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_0=0}^{I_0-1} \sum_{i_1=0}^{I_1-1} \cdots \sum_{i_n=0}^{I_N-1} \mathcal{X}_{i_0, i_1, \cdots, i_n} \mathcal{Y}_{i_0, i_1, \cdots, i_n}$ For two tensors $\mathcal{X} \in \mathbb{R}^{I_x \times I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times I_y}$ sharing N modes of same size, we similarly define a "generalized inner product" along the N last (respectively first) modes of $\mathcal{X}$ (respectively $\mathcal{Y}$) as

$$\langle \mathcal{X}, \mathcal{Y} \rangle_N = \sum_{i_1=0}^{I_1-1} \sum_{i_2=0}^{I_1-1} \cdots \sum_{i_n=0}^{I_N-1} \mathcal{X}_{:, i_1, i_2, \cdots, i_n} \mathcal{Y}_{i_1, i_2, \cdots, i_n, :}, \tag{2}$$

with $\langle \mathcal{X}, \mathcal{Y} \rangle_N \in \mathbb{R}^{I_x \times I_y}$.

Tucker decomposition Given a tensor $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$, we can decompose it into a low rank core $\mathcal{G} \in \mathbb{R}^{R_0 \times R_1 \times \cdots \times R_N}$ by projecting along each of its modes with projection factors $\left(\mathbf{U}^{(0)}, \cdots, \mathbf{U}^{(N)}\right)$, with $\mathbf{U}^{(k)} \in \mathbb{R}^{R_k \times I_k}, k \in [0 \,..\, N]$. In other words, we can write

$$\begin{aligned} \mathcal{X} &= \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(2)} \times \cdots \times_N \mathbf{U}^{(N)} \\ &= [\![\mathcal{G}; \mathbf{U}^{(0)}, \cdots, \mathbf{U}^{(N)}]\!]. \end{aligned} \tag{3}$$

Typically, the factors and core of the decomposition are obtained by solving a least squares problem. In particular, closed form solutions can be obtained for the factor by considering the $n-$mode unfolding of $\mathcal{X}$ that can be expressed as

$$\mathbf{X}_{[n]} = \mathbf{U}^{(n)} \mathbf{G}_{[n]} \left(\mathbf{U}^{(-k)}\right)^\top, \tag{4}$$

where $\mathbf{U}^{(-k)}$ is defined as follows:

$$\mathbf{U}^{(-k)} = \mathbf{U}^{(0)} \otimes \cdots \mathbf{U}^{(n-1)} \otimes \mathbf{U}^{(n+1)} \otimes \cdots \otimes \mathbf{U}^{(N)}.$$

Notice the natural ordering of the factors, from 0 to N, that follows from our definition of unfolding.

Similarly, we can optimize the core in a straightforward manner by isolating it using the equivalent rewriting of the above equality:

$$\text{vec}(\mathbf{X}) = \left(\mathbf{U}^{(0)} \otimes \cdots \otimes \mathbf{U}^{(N)}\right) \text{vec}(\mathbf{G}). \tag{5}$$

We refer the interested reader to the thorough review of the literature on tensor decompositions by Kolda and Bader (2009).

3. Tensor Contraction Layer

One natural way to incorporate tensor operations into a neural network is to apply tensor contraction to an activation tensor in order to obtain a low-dimensional representation (Kossaifi et al., 2017). In this section, we explain how to incorporate tensor contractions into neural networks as a differentiable layer.

We call this technique the Tensor Contraction layer (TCL). Compared to performing a similar rank reduction with a fully connected layer, TCLs require fewer parameters and less computation, while preserving the multilinear structure of the activation tensor.

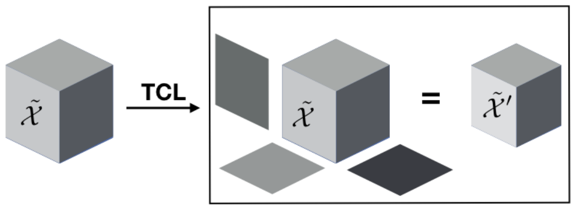


Figure 1: A representation of the Tensor Contraction Layer (TCL) on a tensor of order 3. The input tensor $\mathcal{X}$ is contracted into a low rank core $\mathcal{X}'$. In practice, for efficiency, tensor contraction is applied to a *mini-batch* of such activation tensors, not just a single sample as represented here.

3.1. Tensor contraction layers

Given an activation tensor $\mathcal{X}$ of size $(S_0, I_0, I_1, \cdots, I_N)$, the TCL will produce a compact core tensor $\mathcal{G}$ of smaller size $(S_0, R_0, R_1, \cdots, R_N)$ defined as

$$\mathcal{X}' = \mathcal{X} \times_1 \mathbf{V}^{(0)} \times_2 \mathbf{V}^{(1)} \times \cdots \times_{N+1} \mathbf{V}^{(N)}, \tag{6}$$

with $\mathbf{V}^{(k)} \in \mathbb{R}^{R_k \times I_k}, k \in [0 \,..\, N]$. Note that the projections start at the second mode because the first mode S_0 corresponds to the batch.

This layer allows us to contract the input activation tensor without discarding its multilinear structure. By contrast, a flattening layer followed by a fully connected layer would discard that information. One way to see this is to recognise that a fully connected layer is simply a tensor contraction over the second mode of a batch of flattened activations, as exposed in Subsection 3.3 below.

The projection factors $\left(\mathbf{V}^{(k)}\right)_{k\in[1,\cdots N]}$ are learned end-to-end with the rest of the network by gradient backpropagation. In the rest of this paper, we denote *size*–$(R_0, \cdots, R_N)$ *TCL*, or *TCL*–$(R_0, \cdots, R_N)$ a TCL that produces a compact core of dimension $(R_0, \cdots, R_N)$.

3.2. Gradient back-propagation

In the case of the TCL, we simply need to take the gradients with respect to the factors $\mathbf{V}^{(k)}$ for each $k \in 0, \cdots, N$ of the tensor contraction. Specifically, we compute

$$\frac{\partial \mathcal{X}'}{\partial \mathbf{V}^{(k)}} = \frac{\partial \mathcal{X} \times_1 \mathbf{V}^{(0)} \times_2 \mathbf{V}^{(1)} \times \cdots \times_{N+1} \mathbf{V}^{(N)}}{\partial \mathbf{V}^{(k)}}.$$

By rewriting the previous equality in terms of unfolded tensors, we get an equivalent rewriting where we have isolated the considered factor:

$$\frac{\partial {\mathcal{X}'}_{[k]}}{\partial \mathbf{V}^{(k)}} = \frac{\partial \mathbf{V}^{(k)} \mathbf{X}_{[k]} \left(\mathbf{I} \otimes \mathbf{V}^{(-k)}\right)^\top}{\partial \mathbf{V}^{(k)}},$$

with

$$\mathbf{V}^{(-k)} = \mathbf{V}^{(0)} \otimes \cdots \mathbf{V}^{(k-1)} \otimes \mathbf{V}^{(k+1)} \otimes \cdots \otimes \mathbf{V}^{(N)}.$$

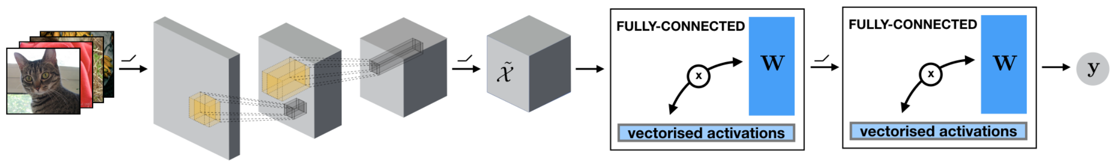


Figure 2: In standard CNNs, the input $\mathcal{X}$ is flattened and then passed to a fully connected layer, where it is multiplied by a weight matrix $\mathbf{W}$.

3.3. Model analysis

Link with fully connected layers Let's considering an activation tensor $\mathcal{X}$ of size $(S_0, I_0, I_1, \cdots, I_N)$. A size–$(R_0, R_1, \cdots, R_N)$ TCL parameterized by weight factors $\mathbf{V}^{(0)}, \cdots, \mathbf{V}^{(N)}$ is equivalent to a fully connected layer parametrized by the weight matrix $\mathbf{W} = \left(\mathbf{V}^{(0)} \otimes \cdots \otimes \mathbf{V}^{(N)}\right)^\top$ and would compute

$$\mathcal{X}_{[0]}\mathbf{W} = \mathcal{X}_{[0]} \left(\mathbf{V}^{(0)} \otimes \cdots \otimes \mathbf{V}^{(N)}\right)^\top,$$

where $\mathcal{X}_{[0]}$ corresponds to the unfolding of $\mathcal{X}$ along the first mode, e.g., a vectorization of each of the samples in the batch.

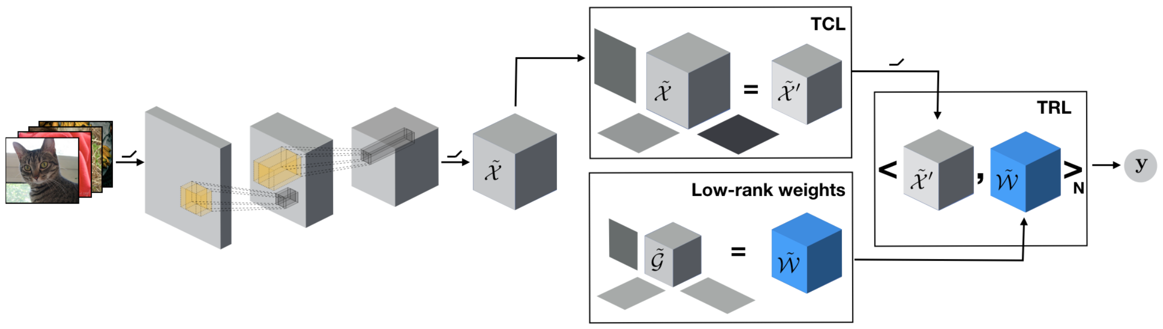


Figure 3: We propose to first reduce the dimensionality of the activation tensor by applying tensor contraction before performing tensor regression. We then replace flattening operators and fully connected layers by a TRL. The output is a product between the activation tensor and a low-rank weight tensor $\mathcal{W}$. For clarity, we illustrate the case of a binary classification, where y is a scalar. For multiclass, y becomes a vector and the regression weights would become a 4^{th} order tensor.

Number of parameters Considering an activation tensor $\mathcal{X}$ of size $(S_0, I_0, I_1, \cdots, I_N)$, a size–$(R_0, R_1, \cdots, R_N)$ TCL parameterized by weight factors $\mathbf{V}^{(0)}, \cdots, \mathbf{V}^{(N)}$ and taking $\mathcal{X}$ as input will have a total of $\sum_{k=0}^{N} I_k \times R_k$ parameters.

This is to contrast with an equivalent fully connected layer (as presented above), parametrized a weight matrix $\mathbf{W} = \left(\mathbf{V}^{(0)} \otimes \cdots \otimes \mathbf{V}^{(N)}\right)^{\top}$, which would have a total of $\prod k = 0^N I_k \times R_k$ parameters.

Notice how the product in number of parameters of the fully connected layer becomes a sum when using a TCL. In other words, in addition to preserving the topological structure in the activation tensor, the TCL has significantly less parameters than a corresponsing fully connected layer.

4. Tensor Regression Layer

In this section, we introduce the Tensor Regression Layer, a new differentiable neural network layer. In order to generate outputs, CNNs typically either flatten the activations or apply a spatial pooling operation. In either case, they discard all multimodal structure and subsequently apply a fully-connected output layer. Instead, we propose leveraging that multilinear structure in the activation tensor and formulate the output as lying in a low-rank subspace that jointly models the input and the output. We do this by means of a low-rank tensor regression, where we enforce a low multilinear rank of the regression weight tensor.

4.1. Tensor regression as a layer

Let us denote by $\mathcal{X} \in \mathbb{R}^{S \times I_0 \times I_1 \times \cdots \times I_N}$ the input activation tensor corresponding to a batch of S samples $(\mathcal{X}_1, \cdots, \mathcal{X}_S)$ and $\mathbf{Y} \in \mathbb{R}^{S \times O}$ the O corresponding labels for each sample. We are interested in the problem of estimating the regression weight tensor $\mathcal{W} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N \times O}$ under some fixed low rank $(R_0, \cdots, R_N, R_{N+1})$ and a bias $\mathbf{b} \in \mathbb{R}^O$, such that $\mathbf{Y} = \langle \mathcal{X}, \mathcal{W} \rangle_N + \mathbf{b}$, i.e.,

$$\begin{aligned} \mathbf{Y} &= \langle \mathcal{X}, \mathcal{W} \rangle_N + \mathbf{b} \\ \text{subject to } \mathcal{W} &= [\![\mathcal{G}; \mathbf{U}^{(0)}, \cdots, \mathbf{U}^{(N)}, \mathbf{U}^{(N+1)}]\!], \end{aligned} \tag{7}$$

with $\langle \mathcal{X}, \mathcal{W} \rangle_N = \mathcal{X}_{[0]} \times \mathcal{W}_{[N+1]}$ the contraction of $\mathcal{X}$ by $\mathcal{W}$ along their N last (respectively first) modes, $\mathcal{G} \in \mathbb{R}^{R_0 \times \cdots \times R_N \times R_{N+1}}$, $\mathbf{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for each k in $[0 \,..\, N]$ and $\mathbf{U}^{(N+1)} \in \mathbb{R}^{O \times R_{N+1}}$.

Previously, this setting has been studied as a standalone problem. In that setting, the input data is directly mapped to the output, and the problem solved analytically (Guo et al., 2012; Rabusseau and Kadri, 2016; Zhou et al., 2013; Yu and Liu, 2016). However, this either limits the model to raw data (e.g., pixel intensities) or requires pre-processing the data to extract (hand-crafted) features to feed the model. For instance, fiducial points that encode the geometry (e.g. facial landmarks) are extracted (Guo et al., 2012). In addition, analytical solutions are prohibitive in terms of computation and memory usage for large datasets.

In this work, we incorporate tensor regressions as trainable layers in neural networks, which allow to learn jointly the features (e.g. via convolutional layers) and the tensor regression. We do so by replacing the traditional flattening + fully connected layers with a tensor regression applied directly to the high-order input and enforcing low rank constraints on the weights of the regression. We call our layer the *Tensor Regression Layer* (*TRL*). Intuitively, the advantage of the TRL comes from leveraging the multi-modal structure in the data and expressing the solution as lying on a low rank manifold encompassing both the data and the associated outputs.

4.2. Gradient backpropagation

The gradients of the regression weights and the core with respect to each factor can be obtained by writing:

$$\frac{\partial \mathcal{W}}{\partial \mathbf{U}^{(k)}} = \frac{\partial \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_{N+1} \mathbf{U}^{(N+1)}}{\partial \mathbf{U}^{(k)}}$$

Using the unfolded expression of the regression weights, we obtain the equivalent formulation:

$$\frac{\partial \mathcal{W}_{[k]}}{\partial \mathbf{U}^{(k)}} = \frac{\partial \mathbf{U}^{(k)} \mathbf{G}_{[k]} \mathbf{R}^\top}{\partial \mathbf{U}^{(k)}}, \tag{8}$$

with

$$\mathbf{R} = \mathbf{U}^{(0)} \otimes \cdots \mathbf{U}^{(k-1)} \otimes \mathbf{U}^{(k+1)} \otimes \cdots \otimes \mathbf{U}^{(N+1)}.$$

Similarly, we can obtain the gradient with respect to the core by considering the vectorized expressions:

$$\frac{\partial \text{vec}(\mathcal{W})}{\partial \text{vec}(\mathcal{G})} = \frac{\partial \big(\mathbf{U}^{(0)} \otimes \cdots \otimes \mathbf{U}^{(N+1)} \big)\, vec(\mathbf{G})}{\partial \text{vec}(\mathcal{G})}.$$

4.3. Model analysis

We consider as input an activation tensor $\mathcal{X} \in \mathbb{R}^{S \times I_0 \times I_1 \times \cdots \times I_N}$, and a rank-$(R_0, R_1, \cdots, R_N, R_{N+1})$ tensor regression layer, where, typically, $R_k \leq I_k$. Let's assume the output is n-dimensional.

A fully connected layer taking $\mathcal{X}$ (after a flattening layer) as input will have n_{FC} parameters, with

$$n_{\text{FC}} = n \times \prod_{k=0}^{N} I_k$$

By comparison, a rank-$(R_0, R_1, \cdots, R_N, R_{N+1})$ TRL taking $\mathcal{X}$ as input has a number of parameters n_{TRL}, with:

$$n_{\text{TRL}} = \prod_{k=0}^{N+1} R_k + \sum_{k=0}^{N} R_k \times I_k + R_{N+1} \times n.$$

5. Efficient implementation of tensor regression layers

Based on the previous layers, we propose an equivalent, more efficient practical implementation of the tensor regression layer. Equation 7 can be written:

$$\mathbf{Y} = \langle \mathcal{X}, \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_{N+1} \mathbf{U}^{(N+1)} \rangle_N + \mathbf{b}$$

We can rewrite this equivalently as

$$\mathbf{Y} = \langle \mathcal{X} \times_0 (\mathbf{U}^{(0)})^\top \times_1 (\mathbf{U}^{(1)})^\top \times \cdots \times_N (\mathbf{U}^{(N)})^\top, \mathcal{G} \times_{N+1} \mathbf{U}^{(N+1)} \rangle_N + \mathbf{b}.$$

This way, most of the computation is done in the *low-rank* subspace rather than directly on the dimensions of $\mathcal{X}$.

In practice, we use this formulation for implementation, and directly learn the pseudo-inverse of each factor$\mathbf{V}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for each k in $[0 \mathrel{..} N]$:

$$\mathbf{Y} = \langle \mathcal{X} \times_0 \mathbf{V}^{(0)} \times_1 \mathbf{V}^{(1)} \times \cdots \times_N \mathbf{V}^{(N)}, \mathcal{G} \times_{N+1} \mathbf{U}^{(N+1)} \rangle_N + \mathbf{b}$$

Note that this is equivalent to first applying a TCL on $\mathcal{X}$, and then applying a TRL on the result to produce $\mathbf{Y}$, with the first factors set to the identity (i.e., the low-rank constraint is applied only to the modes corresponding to the output).

In addition to this efficient formulation, it is also possible to achieve computational speedup via hardware acceleration by leveraging recent work, e.g., Shi et al. (2016), on extending BLAS primitives. This would allow us to reduce the computational overhead for transpositions, which are necessary when computing tensor contractions.

6. Experiments

We empirically demonstrate the effectiveness of preserving the tensor structure through tensor contraction and tensor regression by integrating it into state-of-the-art architectures and demonstrating similar performance on the popular ImageNet dataset. We show that a TRL gives equal or greater performance as compared to flattening followed by fully connected layers, while allowing for large space savings. In particular, we show that our proposed layers allow us to best leverage multi-linear structure in data. This is particularly important for MRI data. On the largest database available, the U.K. biobank, we show that this approach outperforms its traditional counterparts by large margins on three separate prediction tasks.

We empirically verify the effectiveness of the TCL on VGG-19 (Simonyan and Zisserman, 2015). We also conduct thorough experiments and ablation studies with the proposed layers on VGG-19, AlexNet, ResNet-50 and ResNet-101 (He et al., 2016) in various scenarios.

6.1. Implementation details

We implemented all models using the MXNet library (Chen et al., 2015) as well as the PyTorch library Paszke et al. (2017). For all tensor methods, we used the TensorLy library Kossaifi et al. (2019)s. The models were trained with data parallelism across multiple GPUs on Amazon Web Services, with 4 NVIDIA k80 GPUs. For training on ImageNet, we adopt the same data augmentation procedure as in the original Residual Networks (ResNets) paper (He et al., 2016).

When training the layers from scratch, we found it useful to add a batch normalization layer (Ioffe and Szegedy, 2015) before and after the TCL/TRL to avoid vanishing or exploding gradients, and to make the layers more robust to changes in the initialization of the factors. In addition, we constrain the weights of the tensor regression by applying ℓ_2 normalization (Salimans and Kingma, 2016) to the factors of the Tucker decomposition.

When experimenting with the tensor regression layer, instead of retraining the whole network each time it is possible to start from a pre-trained ResNet. We experimented with two settings: (i) We replaced the last average pooling, flattening and fully connected layers by either a TRL or a combination of TCL + TRL and trained these from scratch while keeping the rest of the network fixed; and (ii) We investigate replacing the pooling and fully connected layers with a TRL that jointly learns the spatial pooling as part of the tensor regression. In that setting, we also explore initializing the TRL by performing a Tucker decomposition on the weights of the fully connected layer.

6.2. Large scale image classification

First, we report results in the typical setting for large scale image classification on the the widely-used ImageNet-1K dataset, by learning the spatial pooling as part of the tensor regression.

The ILSVRC dataset (Deng et al., 2009) (ImageNet) is composed of 1.2 million images for training and $50,000$ for validation, all labeled for 1,000 classes. Following (Huang et al., 2017; He et al., 2016; Huang et al., 2016; He et al., 2016), we report results on the validation set in terms of Top-1 accuracy and Top-5 accuracy across all $1,000$ classes. Specifically, we evaluate the classification error on single 224×224 single center crop from the raw input images.

In this setting, we remove the average pooling layer and feed the tensor of size (batch size, number of channels, height, width) to the TRL, while imposing a rank of 1 on the spatial dimensions of the core tensor of the regression. Effectively, this setting simultaneously learns weights for the multilinear spatial pooling as well as the regression.

Our experimental results show that our method enables an effective trade-off between performance and space savings (Table 1). In particular, small space savings (e.g. about 25%) translate in marginal increases in performance. It is possible to obtain more than 65% space savings without impacting accuracy, and to reach larger performance space savings with litte impact on performance (e.g. almost 80% space savings with less than 1% decrease in Top-1 and Top-5 accuracy). We can express the space savings of a model M with n_M total parameters in its fully connected layers with respect to a reference model R with n_R total parameters in its fully connected layers as $1 - \frac{n_M}{n_\mathrm{R}}$ (bias excluded).

Table 1: Results obtained with a ResNet-101 architecture on ImageNet, learning spatial pooling as part of the TRL. We report the Top-1 and Top-5 accuracies, as well as the space savings obtained by replacing the fully connected layer with a tensor regression layer. Our approach enables large space savings with minimal impact on accuracy. In particular, for smaller space savings (about 25%), our approach enables marginal improvements in performance, space savings of up to about 65% do not impact performance; larger space savings translate into small decrease in performance.

	Performance (%)		
TRL rank	**Top-1**	**Top-5**	**Space savings**
Baseline	**77.1**	**93.4**	0
(300, 1, 1, 700)	**77.2**	**93.5**	25.6
(200, 1, 1, 200)	**77.1**	**93.2**	68.2
(120, 1, 1, 300)	**76.7**	**93.1**	71.2
(150, 1, 1, 150)	76	92.9	76.6
(100, 1, 1, 100)	74.6	91.7	84.6
(50 , 1, 1, 50)	73.6	91	**92.4**

To study the TRL in isolation, we consider the weights of a pre-trained model and replace the flattening and fully connected layer with a TRL while keeping the rest of the network fixed. In practice, to initialize the weights of the TRL in this setting, it is possible to consider the weights of the fully connected layer as a tensor of size (batch size, number of channels, 1, 1, number of classes) and apply a partial Tucker decomposition to it by keeping the first dimension (batch-size) untouched. The core and factors of the decomposition then give us the initialization of the TRL. The projection vectors over the spatial dimension are then initialized to $\frac{1}{\text{height}}$ and $\frac{1}{\text{width}}$, respectively. The Tucker decomposition was performed using TensorLy (Kossaifi et al., 2019). In this setting, we show that we can drastically decrease the number of parameters with little impact on performance.

6.3. Large scale phenotypical trait prediction from MRI data

A major challenge in neuroscience and healthcare is analysing structure-rich data, such as 3D brain scans (Mills and Tamnes, 2014; Johnson et al., 2012). The human brain is a highly complex and structured organ. It is composed of interconnected heterogeneous components, each of which is structurally distinct (Sporns et al., 2005). The organizational properties are critical to the brains, and the individuals, overall health. Studies of brain topology have revealed that structural changes are associated with cognitive function (Andreasen et al., 1993) and diseases such as diabetes (Raji et al., 2010). Retaining this topological information is thus critical in order to maximize modelling accuracy. However, flattening layers followed by fully connected layers discards that information and are therefore sub-optimal for the task. By contrast, our proposed tensor regression layers naturally leverages and preserves the topological information, allowing for better performance. We empirically demonstrate this on the UK biobank MRI dataset, the largest imaging study of its kind (Sudlow et al., 2015), the results can be seen in Table 2.

Table 2: **Performance of for UK Biobank MRI** using a regular 3D-ResNet with a fully connected layer for prediction (*baseline FC*) and with a 3D-ResNet where the fully connected layer was replaced with our proposed TRL. The ResNet models with TRL significantly outperforms the baseline version with a fully connected (FC) layer, as it preserves and leverages topological structure.

Method	**Age**	**Gender**	**BMI**
	MAE (years)	Classif.Err. (%)	MAE (kg/m^2)
baseline FC	2.96	0.79	2.37
(256, 3, 4, 3, 1)–TRL	**2.70**	**0.53**	**2.26**
(128, 3, 4, 3, 1)–TRL	2.72	0.60	2.33
(64, 2, 4, 2, 1)–TRL	**2.69**	0.69	2.35
(32, 2, 4, 2, 1)–TRL	2.78	0.73	2.38

We split the data into a training set containing $11,500$ scans, a validation set of $3,800$ scans and $3,800$ scans for a held-out test set. Each scan is a T1-weighted three-dimensional structural MRI with dimensions $182 \times 218 \times 182$. We select three tasks: classifying gender, predicting body mass index (BMI) and predicting age from raw brain MRI scans. Predicting age is an important task as the difference between true and predicted age is a biomarker that has a number of clinical applications (Cole et al., 2017; Kolenic et al., 2018; Franke et al., 2018). Similarly, learning to predict BMI from MRI is a valuable objective, as influence of obesity on brain structure has already been established Alosco et al. (2014), but its mechanism is not fully understood. For the third task of gender classification, differences between genders (more precisely biological sex) in brain functional mappings have already been described (Ingalhalikar et al., 2014). The reasons and implications (if any) of these differences are not known, but understanding the biological processes underpinning them could provide insights into how the brain works. For gender prediction the classes are fairly balanced, the male:female ratio in the UK Biobank is $0.46 : 0.54$.

We compare a standard ResNet with 3D convolutions (3D-ResNet) to a 3D-ResNet where the final fully connected layer was replaced with a TRL. The baseline ResNet contains a global-average pooling layer that compresses all three spatial dimensions, giving an output size of $512 \times n_{outputs}$. The models model were implemented using PyTorch (Paszke et al., 2017) and TensorLy (Kossaifi et al., 2019) and trained was done on a Tesla P100 GPU. Both networks were trained from random initialization. The same number of iterations was used for all comparisons. After validating the number of iterations, we found the baseline and the TRL model required same number of iterations in order to reach convergence and the same number of iterations was subsequently used for all models and all experiments.

The results (table. 2) demonstrate that our tensor-based architecture is able to learn a mapping between structural MRI and phenotypes. It performs significantly better on all tasks compared to a 3D-ResNet with traditional average pooling operation and a fully connected layer. In particular, the 3D-ResNet achieved a mean absolute error (MAE) of 2.96 years on the age-regression task compared with 2.70 years for our TRL. Similarly, the TRL out-performed the baseline on the gender classifying task, achieving an error of only 0.53%

(accuracy of 99.47%) compared to an error of 0.79% for the baseline. For BMI regression, the TRL also performed with a notable improvement over the baseline. A secondary baseline, 3D-ResNet with no average pooling, failed to train at all due to unstable gradients.

These three important tasks demonstrate that the TRL is leveraging additional information in the structural MRI data beyond that of the baseline model. Our method improves on previously reported brain age accuracy in UK Biobank (Ning et al., 2018), and published studies (Gaser et al., 2013; Kolenic et al., 2018; Cole et al., 2018) on other brain MRI datasets. The method also improves on BMI prediction beyond those previously published for UK Biobank data (Vakli et al., 2020). Improved performance gives further support for use of machine learning as a support tool in neuroradiological research and clinical decision making.

One question is whether this improvements are due to the ability of the TRL to preserve and leverage topological structure or whether it is simply a result of the regularizing effect of the TRL. To answer this question, we experimented with a TRL, where the entire $6 \times 7 \times 6$ full-rank activation tensor was used for the age prediction task. Using the full-rank tensor the network achieved a MAE of 2.71 years, which is similar to the results obtained with lower-rank set ups, and significantly better than the baseline. This empirically confirms the importance of leveraging topological structure.

6.4. Ablation studies

Synthetic setting To illustrate the effectiveness of the low-rank tensor regression, in terms of learning and data efficiency, we first investigate it in isolation.

We apply it to synthetic data $y = \text{vec}(\mathcal{X}) \times \mathbf{W}$ where each sample $\mathcal{X} \in \mathbb{R}^{(64)}$ follows a Gaussian distribution $\mathcal{N}(0, 3)$. $\mathbf{W}$ is a fixed matrix and the labels are generated as $y = \text{vec}(\mathcal{X}) \times \mathbf{W}$. We then train the data on $\mathcal{X} + \mathcal{E}$, where $\mathcal{E}$ is added Gaussian noise sampled from $\mathcal{N}(0, 3)$. We compare i) a TRL with squared loss and ii) a fully connected layer with a squared loss.

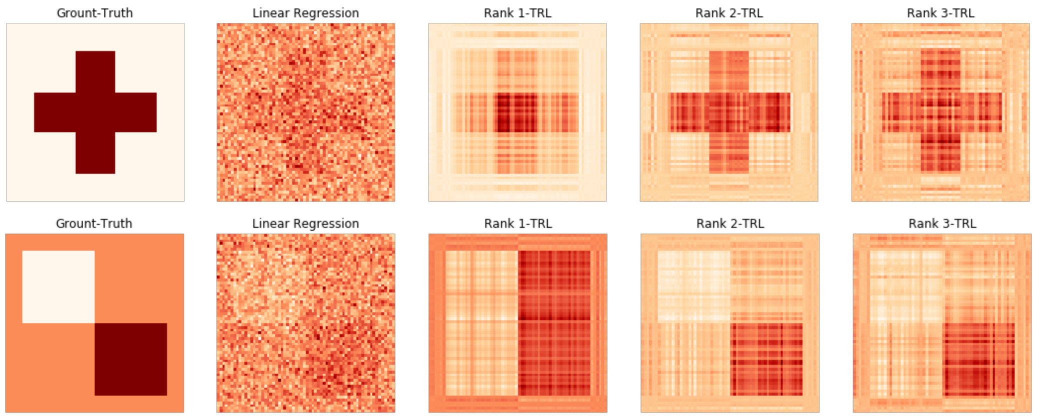


Figure 4: Empirical comparison (4) of the TRL against linear regression with a fully connected layer. We plot the weight matrix of a TRL and a fully connected layer. Due to its low-rank weights, the TRL better captures the structure in the weights and is more robust to noise.

In Figure 4, we show the trained weight of both a linear regression based on a fully connected layer and a TRL with various ranks, both obtained in the same setting. The TRL is able to better recover the ground-truth weight due to the low-rank structure imposed on these, which allows to leverage the multi-linear structure and acts as an implicit regularizer. This is in line with findings from existing works focusing on analytical solution to tensor regression problems Rabusseau and Kadri (2016). Additional regularizers such as Lasso (l1) Hoefling (2010); Tibshirani and Taylor (2011) can provide further advantages, especially in the face of noisy inputs. Going forward, we plan to investigate such additional regularization terms.

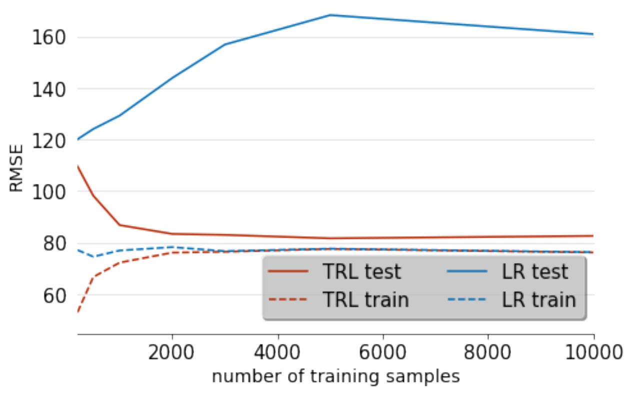


Figure 5: Evolution of the RMSE as a function of the training set size for both the TRL and fully connected regression. Due to the low-rank structure of its weights tensor, the TRL requires less training data and is less prone to overfitting than fully connected layers.

We also compare the data efficiency of a tensor regression layer compared to a linear regression, in the same setting, by varying the number of training samples. As can be observed in Figure 5, the TRL is easier to train on small datasets and less prone to overfitting, due to the low rank structure of its regression weights, as opposed to typical (fully connected) linear regression.

Impact of the tensor contraction layer We first investigate the effectiveness of the TCL using a VGG-19 network architecture (Simonyan and Zisserman, 2015). This network is especially well-suited for our methods because of its $138,357,544$ parameters, $119,545,856$ of which (more than 80% of the total number of parameters) are contained in the fully-connected layers. By adding a TCL to contract the activation tensor prior to the fully connected layers, we can achieve large space saving.

Table 3 presents the accuracy obtained by the different combinations of TCL in terms of top-1 and top-5 accuracy as well as space saving. By adding a TCL that preserves the size of its input we are able to obtain slightly higher performance with little impact on the space saving (0.21% of space loss) while by decreasing the size of the TCL we got more than 65% space saving with almost no performance deterioration.

Overcomplete TRL We first tested the TRL with a ResNet-50 and a ResNet-101 architectures on ImageNet, removing the average pooling layer to preserve the spatial information in the tensor. The full activation tensor is directly passed on to a TRL which produces

Table 3: Results obtained on ImageNet by adding a TCL to a VGG-19 architecture. We reduce the number of hidden units proportionally to the reduction in size of the activation tensor following the tensor contraction. Doing so allows more than 65% space savings over all three fully connected layers (i.e. 99.8% space saving over the fully connected layer replaced by the TCL) with no corresponding decrease in performance (comparing to the standard VGG network as a baseline).

Method		**Accuracy**		**Space Savings**
TCL–size	**Hidden Units**	**Top-1 (%)**	**Top-5 (%)**	**(%)**
Baseline	4096	68.7	88	0
(512, 7, 7)	4096	**69.4**	**88.3**	-0.21
(384, 5, 5)	3072	68.3	87.8	**65.87**

Table 4: Results obtained with ResNet-50 on ImageNet. The first row corresponds to the standard ResNet. Rows 2 and 3 present the results obtained by replacing the last average pooling, flattening and fully connected layers with a TRL. In the last row, we have also added a TCL.

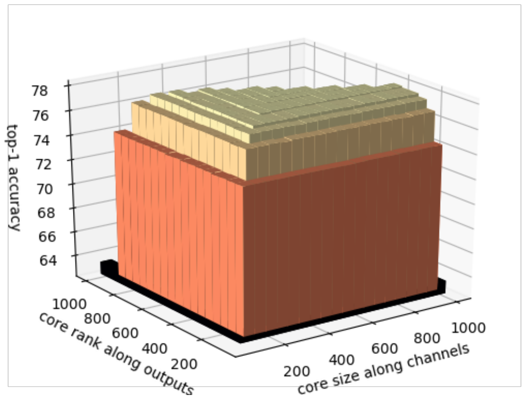
Method			**Accuracy**	
Architecture	**TCL–size**	**TRL rank**	**Top-1 (%)**	**Top-5 (%)**
ResNet-50	NA (baseline)		74.58	92.06
ResNet-50	no TCL	(1000, 2048, 7, 7)	73.6	91.3
ResNet-50	no TCL	(500, 1024, 3, 3)	72.16	90.44
ResNet-50	(1024, 3, 3)	(1000, 1024, 3, 3)	73.43	91.3
ResNet-101	NA (baseline)		77.1	93.4
ResNet-101	no TCL	(1000, 2048, 7, 7)	76.45	92.9
ResNet-101	no TCL	(500, 1024, 3, 3)	76.7	92.9
ResNet-101	(1024, 3, 3)	(1000, 1024, 3, 3)	76.56	93

the outputs on which we apply softmax to get the final predictions. This results in more parameters as the spatial dimensions are preserved. To reduce the computational burden but preserve the multi-dimensional information, we alternatively insert a TCL before the TRL.

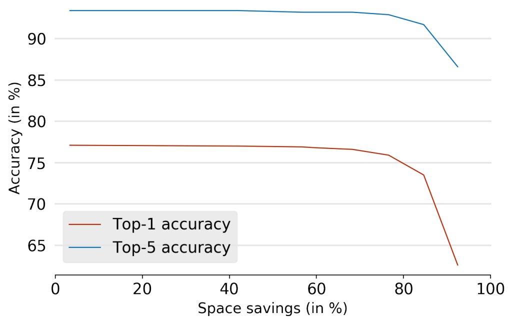
In Table 4, we present results obtained in this setting on ImageNet for various configurations of the network architecture. In each case, we report the size of the TCL (i.e. the dimension of the contracted tensor) and the rank of the TRL (i.e. the dimension of the core of the regression weights).

Choice of the rank of the TRL While the rank of the TRL is an additional parameter to validate, it turns out to be easy to tune in practice.

In Figure 6, we show the effect on Top-1 and Top-5 accuracy of decreasing the size of the core tensor of the TRL. We also show the corresponding space savings. The results suggest that choosing the rank is easy because there is a large range of values of the rank for which



(a) Accuracy as a function of the core size



(b) Accuracy as a function of space savings

Figure 6: Study of the impact of the rank of the tensor regression layer on performance. On the left side, 6a shows the Top-1 accuracy (in %) as we vary the size of the core along the number of outputs and number of channels (the TRL does spatial pooling along the spatial dimensions, i.e., the core has rank 1 along these dimensions). On the right side, 6b shows the evolution of the Top-1 and Top-5 accuracy (in %) as a function of the space savings by reducing the rank of the TRL (also in %). As can be observed, there is a large region for which the reduction of the rank of the tensor regression layer does not negatively impact the performance while enabling large space savings.

the performance does not decrease. In particular, we can obtain up to 80% space savings with negligible impact on performance.

7. Conclusions

Deep neural networks already operate on multilinear activation tensors, the structure of which is typically discarded by flattening operations and fully-connected layers. This paper proposed preserving and leveraging the tensor structure of the activations by introducing two new, end-to-end trainable, layers that enable substantial space savings while preserving and leveraging the multi-dimensional topological structure. The TCL that we propose reduces the dimension of the input without discarding its multi-linear structure, while TRLs directly map their input tensors to the output with low-rank regression weights. These techniques are easy to plug in to existing architectures and are trainable end-to-end.

Our experiments demonstrate that by imposing a low-rank constraint on the weights of the regression, we can learn a low-rank manifold on which both the data and the labels lie. Furthermore these new layers act as an additional type of regularization on the activations (TCL) and the regression weight tensors (TRL). The result is a compact network that achieves similar accuracies with far fewer parameters. The structure in the regression weight tensor allows for more interpretable models while requiring less data to train. Going forward, we plan to apply the TCL and TRL to more network architectures and leverage recent work to avoid computational overhead from transpositions when computing tensor contractions.

Acknowledgements

This research has been conducted using the UK Biobank Resource under Application Number 18545. The authors would like to thank the editor and anonymous reviewers for the constructive feedback which helped improve this manuscript.

References

Michael L Alosco, Kelly M Stanek, Rachel Galioto, Mayuresh S Korgaonkar, Stuart M Grieve, Adam M Brickman, Mary Beth Spitznagel, and John Gunstad. Body mass index and brain structure in healthy children and adolescents. *International Journal of Neuroscience*, 124(1):49–55, 2014.

Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

Nancy C Andreasen, Michael Flaum, Victor Swayze, Daniel S O'Leary, Randall Alliger, Gregg Cohen, James Ehrhardt, William T Yuh, et al. Intelligence and brain structure in normal individuals. *American Journal of Psychiatry*, 150:130–130, 1993.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

Yunpeng Chen, Xiaojie Jin, Bingyi Kang, Jiashi Feng, and Shuicheng Yan. Sharing residual units through collective tensor factorization to improve deep neural networks. In *IJCAI*, pages 635–641. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. PHAN. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, March 2015.

Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-Ichi Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation.* John Wiley & Sons, Ltd, 2009.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 698–728, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

James H Cole, Rudra PK Poudel, Dimosthenis Tsagkrasoulis, Matthan WA Caan, Claire Steves, Tim D Spector, and Giovanni Montana. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163: 115–124, 2017.

James H Cole, Stuart J Ritchie, Mark E Bastin, MC Valdés Hernández, S Muñoz Maniega, Natalie Royle, Janie Corley, Alison Pattie, Sarah E Harris, Qian Zhang, et al. Brain age predicts mortality. *Molecular psychiatry*, 23(5):1385, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Katja Franke, Christian Gaser, Tessa J Roseboom, Matthias Schwab, and Susanne R de Rooij. Premature brain aging in humans exposed to maternal nutrient restriction during early gestation. *NeuroImage*, 173:460–471, 2018.

Christian Gaser, Katja Franke, Stefan Klöppel, Nikolaos Koutsouleris, Heinrich Sauer, Alzheimer's Disease Neuroimaging Initiative, et al. Brainage in mild cognitive impaired patients: predicting the conversion to alzheimers disease. *PloS one*, 8(6), 2013.

Edward Grefenstette and Mehrnoosh Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, page 13941404, USA, 2011. Association for Computational Linguistics. ISBN 9781937284114.

W. Guo, I. Kotsia, and I. Patras. Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2):816–827, Feb 2012.

Benjamin D. Haeffele and René Vidal. Global optimality in tensor factorization, deep learning, and beyond. *CoRR*, abs/1506.07540, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645. Springer International Publishing, 2016. ISBN 978-3-319-46493-0.

Holger Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010. doi: 10.1198/jcgs.2010.09208.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661. Springer International Publishing, 2016. ISBN 978-3-319-46493-0.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4700–4708, 2017.

Madhura Ingalhalikar, Alex Smith, Drew Parker, Theodore D Satterthwaite, Mark A Elliott, Kosha Ruparel, Hakon Hakonarson, Raquel E Gur, Ruben C Gur, and Ragini Verma. Sex differences in the structural connectome of the human brain. *Proceedings of the National Academy of Sciences*, 111(2):823–828, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning (ICML)*, page 448456, 2015.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Generalization bounds for neural networks through tensor factorization. *CoRR*, abs/1506.08473, 2015a.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *CoRR*, 2015b.

Keith A Johnson, Nick C Fox, Reisa A Sperling, and William E Klunk. Brain imaging in alzheimer disease. *Cold Spring Harbor perspectives in medicine*, 2(4):a006213, 2012.

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. In *ICLR*, 2016.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM REVIEW*, 51(3):455–500, 2009.

Marian Kolenic, Katja Franke, Jaroslav Hlinka, Martin Matejka, Jana Capkova, Zdenka Pausova, Rudolf Uher, Martin Alda, Filip Spaniel, and Tomas Hajek. Obesity, dyslipidemia and brain age in first-episode psychosis. *Journal of psychiatric research*, 99:151–158, 2018.

Jean Kossaifi, Aran Khanna, Zachary Lipton, Tommaso Furlanello, and Anima Anandkumar. Tensor contraction layers for parsimonious deep nets. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research*, 20(26):1–6, 2019.

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

Kathryn L Mills and Christian K Tamnes. Methods and considerations for longitudinal structural brain imaging analysis across development. *Developmental cognitive neuroscience*, 9: 172–190, 2014.

Kaida Ning, Lu Zhao, Will Matloff, Fengzhu Sun, and Arthur Toga. Association of brain age with smoking, alcohol consumption, and genetic variants. *bioRxiv*, page 469924, 2018.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 442–450, 2015.

Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Trans. Intell. Syst. Technol.*, 8(2):16:1–16:44, October 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017.

Guillaume Rabusseau and Hachem Kadri. Low-rank regression with tensor responses. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1867–1875. Curran Associates, Inc., 2016.

Cyrus A Raji, April J Ho, Neelroop N Parikshak, James T Becker, Oscar L Lopez, Lewis H Kuller, Xue Hua, Alex D Leow, Arthur W Toga, and Paul M Thompson. Brain structure and obesity. *Human brain mapping*, 31(3):353–364, 2010.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 901–909. Curran Associates, Inc., 2016.

Hanie Sedghi and Anima Anandkumar. Training input-output recurrent neural networks through spectral methods. *CoRR*, abs/1603.00954, 2016.

Or Sharir and Amnon Shashua. On the expressive power of overlapping architectures of deep learning. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/forum?id=HkNGsseC-`.

Y. Shi, U. N. Niranjan, A. Anandkumar, and C. Cecka. Tensor contractions with extended blas kernels on cpu and gpu. In *2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*, pages 193–202, Dec 2016.

N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4), 2005.

Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779, 2015.

Cheng Tai, Tong Xiao, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. *CoRR*, abs/1511.06067, 2015.

Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *Ann. Statist.*, 39(3):1335–1371, 06 2011. doi: 10.1214/11-AOS878. URL `https://doi.org/10.1214/11-AOS878`.

Pál Vakli, Regina J Deák-Meszlényi, Tibor Auer, and Zoltán Vidnyánszky. Predicting body mass index from structural mri brain images using a deep convolutional neural network. *Frontiers in Neuroinformatics*, 14:10, 2020.

M. A. O. Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, pages 447–460, London, UK, UK, 2002. Springer-Verlag.

Yongxin Yang and Timothy M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *ICLR*, 2017.

Rose Yu and Yan Liu. Learning from multiway data: Simple and efficient tensor regression. In *International Conference on Machine Learning (ICML)*, volume 48, pages 373–381, New York, New York, USA, 20–22 Jun 2016. PMLR.

Hua Zhou, Lexin Li, and Hongtu Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013.