

---

# Stochastically Rank-Regularized Tensor Regression Networks

---

Arinbjörn Kolbeinsson<sup>1,2\*</sup> Jean Kossaifi<sup>1,2\*</sup> Yannis Panagakis<sup>1,2</sup> Anima Anandkumar<sup>3</sup> Ioanna Tzoulaki<sup>2</sup>  
Paul Matthews<sup>2</sup>

## Abstract

Over-parametrization of deep neural networks has recently been shown to be key to their successful training. However, it also renders them prone to overfitting and makes them expensive to store and train. Tensor regression networks significantly reduce the number of effective parameters in deep neural networks while retaining accuracy and the ease of training. They replace the flattening and fully-connected layers with a tensor regression layer, where the regression weights are expressed through the factors of a low-rank tensor decomposition. In this paper, to further improve tensor regression networks, we propose a novel stochastic rank-regularization. It consists of a novel randomized tensor sketching method to approximate the weights of tensor regression layers. We theoretically and empirically establish the link between our proposed stochastic rank-regularization and the dropout on low-rank tensor regression. Extensive experimental results with both synthetic data and real world datasets (i.e., CIFAR-100 and the UK Biobank brain MRI dataset) support that the proposed approach i) improves performance in both classification and regression tasks, ii) decreases overfitting, iii) leads to more stable training and iv) improves robustness to adversarial attacks and random noise.

## 1. Introduction

Deep neural networks have been evolved to a general-purpose machine learning method with remarkable performance on practical applications (LeCun et al., 2015). Such models are usually over-parameterized, involving an enormous number (possibly millions) of parameters. This is much larger than the typical number of available training

samples, making deep networks prone to overfitting (Caruana et al., 2001). Coupled with overfitting, the large number of unknown parameters makes deep learning models extremely hard and computationally expensive to train, requiring huge amount of memory and computation power. Such resources are often available only in massive computer clusters, preventing deep networks to be deployed in resource limited machines such as mobile and embedded devices.

To prevent deep neural networks from overfitting and improve their generalization ability, several explicit and implicit regularization methods have been proposed. More specifically, explicit regularization strategies, such as weight decay involve  $\ell_2$ -norm regularization of the parameters (Nowlan & Hinton, 1992; Krogh & Hertz, 1992). Replacing the  $\ell_2$  with  $\ell_1$ -norm has been also investigated (Scardapane et al., 2017; Zhang et al., 2016). Besides the aforementioned general-purpose regularization functions, neural networks specific methods such as early stopping of back-propagation (Caruana et al., 2001), batch normalization (Ioffe & Szegedy, 2015), dropout (Srivastava et al., 2014) and its variants –e.g., DropConnect (Wan et al., 2013)– are algorithmic approaches to reducing overfitting in over-parametrized networks and have been widely adopted in practice.

Reducing the storage and computational costs of deep networks has become critical for meeting the requirements of environments with limited memory or computational resources. To this end, a surge of network compression and approximation algorithms have recently been proposed in the context of deep learning. By leveraging the redundancy in network parameters, methods such as Tai et al. (2015); Cheng et al. (2015); Yu et al. (2017); Kossaifi et al. (2018) employ low-rank approximations of deep networks weight matrices (or tensors) for parameter reduction. Network compression methods in the frequency domain (Chen et al., 2016) have also been investigated. An alternative approach for reducing the number of effective parameters in deep nets relies on sketching, whereby, given a matrix or tensor of input data or parameters, one first compresses it to a much smaller matrix (or tensor) by multiplying it by a (usually) random matrix with certain properties (Kasiviswanathan et al., 2017; Daniely et al., 2016).

---

\*Equal contribution <sup>1</sup>Samsung AI, Cambridge, United Kingdom <sup>2</sup>Imperial College London, London, United Kingdom <sup>3</sup>California Institute of Technology, Pasadena, United-States. Correspondence to: Arinbjörn Kolbeinsson <arinbjorn@gmail.com>, Jean Kossaifi <jean.kossaifi@gmail.com>.

A particularly appealing approach to network compression, especially for visual data<sup>1</sup> (and other types of multidimensional and multi-aspect data) is tensor regression networks (Kossaifi et al., 2018). Deep neural networks typically leverage the spatial structure of input data via series of convolutions, point-wise non-linearities, pooling, etc. However, this structure is usually wasted by the addition, at the end of the networks’ architectures, of a flattening layer followed by one or several fully-connected layers. A recent line of study focuses on alleviating this using tensor methods. Kossaifi et al. (2017) proposed tensor contraction as a layer, to reduce the size of activation tensors, and demonstrated large space savings by replacing fully-connected layers with this layer. However, a flattening layer and fully-connected layers were still ultimately needed for producing the outputs. Recently, tensor regression networks (Kossaifi et al., 2018) propose to replace flattening and fully-connected layers entirely with a tensor regression layer (TRL). This preserves the structure by expressing an output tensor as the result of a tensor contraction between the input tensor and some low-rank regression weight tensors. In addition, these allow for large space savings without sacrificing accuracy. Cao et al. (2017) explore the same model with various low-rank structures on the regression weight tensor.

In this paper, we combine ideas from networks regularization, low-rank approximation of networks, and randomized sketching in a principled way and introduce a novel stochastic regularization term to the tensor regression networks. It consists of a novel randomized low-rank tensor regression, which leads to the stochastic reduction of the rank, either by a fixed percentage during training or according to a series of Bernoulli random variables. This is akin to dropout, which, by randomly dropping units during training, prevents over-fitting. However, rather than dropping random elements from the *activation* tensor, this is done on the regression weight tensor. We explore two schemes: (i) selecting random elements to keep, following a Bernoulli distribution and (ii) keeping a random subset of the *fibers* of the tensor, with replacement. We theoretically and empirically establish the link between CP TRL with the proposed regularizer and the dropout on the deterministic low-rank tensor regression.

To demonstrate the practical advantages of this method, we conducted experiments in image classification and phenotypic trait prediction from MRI. To this end, the CIFAR-100 and the UK Biobank brain MRI datasets were employed. Experimental results demonstrate that the proposed method i) improves performance in both classification and regression tasks, ii) decreases over-fitting, iii) leads to more stable training and iv) largely improves robustness to adversarial

attacks and random noise.

One notable application of deep neural networks is in medical imaging, particularly magnetic resonance imaging (MRI). MRI analysis performed using deep learning includes age prediction for brain-age estimation (Cole et al., 2017a). Brain-age has been associated with a range of diseases and mortality (Cole et al., 2017b), and could be an early predictor for Alzheimer’s disease (Franke et al., 2012). A more accurate and more robust brain age estimation can consequently lead to more accurate disease diagnoses. We demonstrate a large performance improvement (more than 20%) on this task using a 3D-ResNet with our proposed stochastically rank-regularized TRL, compared to a regular 3D-ResNet.

## 2. Closely related work

**Network regularization and dropout.** Several methods that improve generalization by mitigating overfitting have been developed in the context of deep learning. The interested reader is referred to the work of Kukačka et al. (2017) and the references therein for a comprehensive survey of over 50 different regularization techniques for deep networks.

The most closely related regularization method to our approach is Dropout (Srivastava et al., 2014), which is probably the most widely adopted technique for training neural networks while preventing overfitting. Concretely, during dropout training each unit (i.e., neuron) is equipped with a binary Bernoulli random variable and only the networks weights whose corresponding Bernoulli variables are sampled with value 1 are updated at each back-propagation step. At each iteration, those Bernoulli variables are re-sampled again and the weights are updated accordingly. The proposed regularization method can be interpreted as dropout on low-rank tensor regression, a fact which is proved in Section 4.3.

**Sketching and deep networks approximation.** Daniely et al. (2016) apply sketching to the input data in order to sparsify them and reduce their dimensionality. Subsequently they show any sparse polynomial function can be computed, on all sparse binary vectors, by a single layer neural network that takes a compact sketch of the vector as input. In contrast, Kasiviswanathan et al. (2017), approximate neural networks and apply a random sketching on weight matrices/tensors instead of input data and demonstrate that given a fixed layer input, the output of this layer using sketching matrices is an unbiased estimator of the original output of this layer and has bounded variance. As opposed to the aforementioned sketching methods for deep networks approximation, the proposed method applies sketching in the low-rank factorization of weights.

<sup>1</sup>Most modern data is inherently multi-dimensional -color images are naturally represented by 3<sup>rd</sup> order tensors, videos by 4<sup>th</sup> order tensors, etc.)

**Randomized tensor decompositions.** Tensor decompositions exhibit high computational cost and low convergence rate when applied to massive multi-dimensional data. To accelerate computation, randomized tensor decompositions have been employed to scale tensor decompositions. A randomized least squares algorithm for CP decomposition is proposed by Battaglino et al. (2018), which is significantly faster than traditional CP decomposition. In (Erichson et al., 2017), CP is applied on a small tensor generated by tensor random projection of the high-dimensional tensor. The CP decomposition of the large-scale tensor is obtained by back projection of the CP decomposition of the small tensor. Wang et al. (2015) introduce a fast yet provable randomized CP decomposition that performs randomized tensor contraction using FFT. Methods in (Sidiropoulos et al., 2014; Vervliet et al., 2014) are highly computationally efficient algorithms for computing large-scale CP decompositions by applying randomization (random projections) into a set of small tensors, derived by subdividing a tensor into a set of blocks. Fast randomized algorithms that employ sketching for approximating Tucker decomposition have been also investigated (Tsourakakis, 2010; Zhou et al., 2014). More recently, a randomized tensor ring decomposition that employs tensor random projections has been developed in Yuan et al. (2019). The most similar method to ours is that of Battaglino et al. (2018), where elements of the tensor are sampled randomly, and each factor of the decomposition updated in an iterative manner. By contrast, our method allows for end-to-end training, and applies randomization on the *fibers* of the tensor, effectively randomizing the rank of the weight tensor.

### 3. Tensor Regression Networks

In this section, we introduce the notations and notions necessary to introduce our stochastic rank regularization.

**Notation:** We denote  $\mathbf{v}$  vectors (1<sup>st</sup> order tensors) and  $\mathbf{M}$  matrices (2<sup>nd</sup> order tensors).  $\mathbf{Id}$  is the identity matrix. We denote  $\mathcal{X}$  tensors of order  $N \geq 3$ , and denote its element  $(i, j, k)$  as  $\mathcal{X}_{i_0, i_1, \dots, i_{N-1}}$  or  $\mathcal{X}(i_0, i_1, \dots, i_{N-1})$ . A colon is used to denote all elements of a mode e.g. the mode-0 fibers of  $\mathcal{X}$  are denoted as  $\mathcal{X}_{:, i_1, \dots, i_{N-1}}$ . The transpose of  $\mathbf{M}$  is denoted  $\mathbf{M}^\top$ . Finally, for any  $i, j \in \mathbb{N}, i < j, [i \dots j]$  denotes the set of integers  $\{i, i+1, \dots, j-1, j\}$ , and  $i \operatorname{div} j$  the integer division of  $i$  by  $j$ .

**Tensor unfolding:** Given a tensor,  $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ , its mode- $n$  unfolding is a matrix  $\mathbf{X}_{[n]} \in \mathbb{R}^{I_n \times I_M}$ , with  $M = \prod_{k=0, k \neq n}^{N-1} I_k$  and is defined by the mapping from ele-

ment  $(i_0, i_1, \dots, i_N)$  to  $(i_n, j)$ , with

$$j = \sum_{\substack{k=0, \\ k \neq n}}^{N-1} i_k \times \prod_{\substack{m=k+1, \\ m \neq n}}^{N-1} I_m.$$

**Tensor vectorization:** Given a tensor,  $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ , we can flatten it into a vector  $\operatorname{vec}(\mathcal{X})$  of size  $(I_0 \times \dots \times I_{N-1})$  defined by the mapping from element  $(i_0, i_1, \dots, i_{N-1})$  of  $\mathcal{X}$  to element  $j$  of  $\operatorname{vec}(\mathcal{X})$ , with

$$j = \sum_{k=0}^{N-1} i_k \times \prod_{m=k+1}^{N-1} I_m.$$

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

$$(\mathcal{X} \times_n \mathbf{M})_{[n]} = \mathbf{M} \mathcal{X}_{[n]} \in \mathbb{R}^{I_0 \times \dots \times I_{n-1} \times R \times I_{n+1} \times \dots \times I_{N-1}}$$

**Generalized inner product:** For two tensors  $\mathcal{X} \in \mathbb{R}^{K_0 \times \dots \times K_x \times I_0 \times \dots \times I_{N-1}}$  and  $\mathcal{Y} \in \mathbb{R}^{I_0 \times \dots \times I_{N-1} \times L_0 \times \dots \times L_y}$ , we denote by  $\langle \mathcal{X}, \mathcal{Y} \rangle_N \in \mathbb{R}^{K_0 \times \dots \times K_x \times I_{N-1} \times L_0 \times \dots \times L_y}$  the contraction of  $\mathcal{X}$  by  $\mathcal{Y}$  along their  $N-1$  last (respectively first) modes.

$$\langle \mathcal{X}, \mathcal{Y} \rangle_N = \sum_{i_0=0}^{I_0} \sum_{i_1=0}^{I_1} \dots \sum_{i_{N-1}=0}^{I_{N-1}} \mathcal{X}_{\dots, i_0, i_1, \dots, i_{N-1}} \mathcal{Y}_{i_0, i_1, \dots, i_{N-1}, \dots}$$

**Kruskal tensor:** Given a tensor  $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ , the Canonical-Polyadic decomposition (CP), also called PARAFAC, decomposes it into a sum of  $R$  rank-1 tensors. The number of terms in the sum,  $R$ , is known as the rank of the decomposition. Formally, we find the vectors  $\mathbf{u}_k^{(0)}, \mathbf{u}_k^{(1)}, \dots, \mathbf{u}_k^{(2)}$ , for  $k = [0 \dots R-1]$  such that:

$$\mathcal{X} = \sum_{k=0}^{R-1} \underbrace{\mathbf{u}_k^{(0)} \circ \mathbf{u}_k^{(1)} \circ \dots \circ \mathbf{u}_k^{(N-1)}}_{\text{rank-1 components}} \quad (1)$$

These vectors can be collected in matrices, called factors or the decomposition. Specifically, we define, for each factor  $k \in [1 \dots N-1]$ ,  $\mathbf{U}^{(k)} = [\mathbf{u}_0^{(k)}, \mathbf{u}_1^{(k)}, \dots, \mathbf{u}_{R-1}^{(k)}]$ . The magnitude of the factors can optionally be absorbed in a vector of weights  $\boldsymbol{\lambda} \in \mathbb{R}^R$ , such that

$$\mathcal{X} = \sum_{k=0}^{R-1} \lambda_k \mathbf{u}_k^{(0)} \circ \mathbf{u}_k^{(1)} \circ \dots \circ \mathbf{u}_k^{(N-1)}$$

The decomposition can be denoted more compactly as  $\mathcal{X} = \llbracket \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket$ , or  $\mathcal{X} = \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket$  if a weights vector is used.

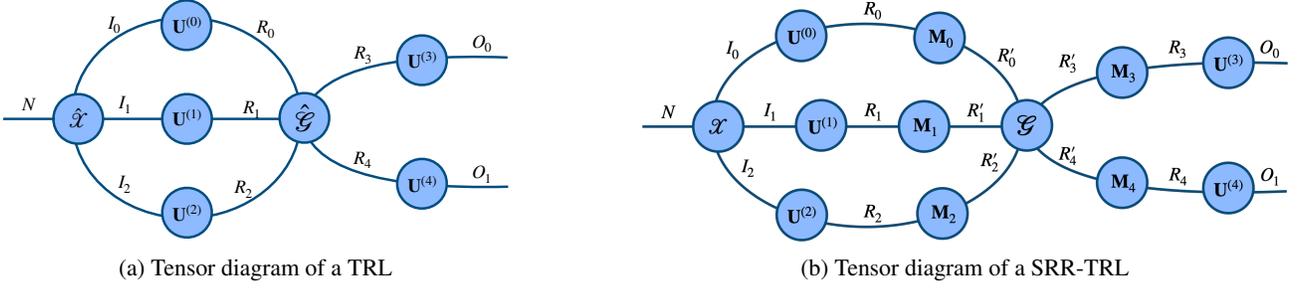


Figure 1. Tensor diagrams of the TRL (left) and our proposed SRR-TRL (right), with low-rank constraints imposed on the regression weights tensor using a Tucker decomposition. Note that the CP case is readily given by this formulation by additionally having the core tensor  $\mathcal{G}$  be super-diagonal, and setting  $\mathbf{M} = \mathbf{M}^{(0)} = \dots = \mathbf{M}^{(N)} = \text{diag}(\boldsymbol{\lambda})$ .

**Tucker tensor:** Given a tensor  $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$ , we can decompose it into a low rank core  $\mathcal{G} \in \mathbb{R}^{R_0 \times R_1 \times \dots \times R_{N-1}}$  by projecting along each of its modes with projection factors  $(\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)})$ , with  $\mathbf{U}^{(k)} \in \mathbb{R}^{R_k \times I_k}$ ,  $k \in (0, \dots, N-1)$ .

This allows us to write the tensor in a decomposed form as:

$$\begin{aligned} \mathcal{X} &= \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(2)} \times \dots \times_{N-1} \mathbf{U}^{(N-1)} \\ &= \llbracket \mathcal{G}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N-1)} \rrbracket \end{aligned} \quad (2)$$

Note that the Kruskal form of a tensor can be seen as a Tucker tensor with a super-diagonal core.

**Tensor diagrams:** In order to represent easily tensor operations, we adopt the tensor diagrams, where tensors are represented by vertices (circles) and edges represent their modes. The degree of a vertex then represents its order. Connecting two edges symbolizes a tensor contraction between the two represented modes. Figure 1 presents a tensor diagram of the tensor regression layer and its stochastic rank-regularized counterpart.

**Tensor regression layers (TRL):** Let us denote by  $\mathcal{X} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_{N-1}}$  the input activation tensor for a sample and  $\mathbf{y} \in \mathbb{R}^{I_N}$  the label vector. We are interested in the problem of estimating the regression weight tensor  $\mathcal{W} \in \mathbb{R}^{I_0 \times I_1 \times \dots \times I_N}$  under some fixed low rank  $(R_0, \dots, R_N)$ :

$$\begin{aligned} \mathbf{y} &= \langle \mathcal{X}, \mathcal{W} \rangle_N + \mathbf{b} \\ \text{with } \mathcal{W} &= \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \dots \times_N \mathbf{U}^{(N)} \end{aligned} \quad (3)$$

with  $\mathcal{G} \in \mathbb{R}^{R_0 \times \dots \times R_N}$ ,  $\mathbf{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$  for each  $k$  in  $[0 \dots N]$  and  $\mathbf{U}^{(N)} \in \mathbb{R}^{O \times R_N}$ .

## 4. Stochastic rank regularization

In this section, we introduce the stochastic rank regularization (SRR). Specifically, we propose a new stochastic rank-regularization, applied to low-rank tensors in decomposed forms. This formulation is general and can be applied

to any type of decomposition. We introduce it here, without loss of generality, to the case of Tucker and CP decompositions.

For any  $k \in [0 \dots N]$ , let  $\mathbf{M}^{(k)} \in \mathbb{R}^{R_0 \times R_0}$  be a sketch matrix (e.g. a random projection or column selection matrix) and,  $\tilde{\mathbf{U}}^{(k)} = \mathbf{U}^{(k)}(\mathbf{M}^{(k)})^\top$  be a sketch of factor matrix  $\mathbf{U}^{(k)}$ , and  $\tilde{\mathcal{G}} = \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_N \mathbf{M}^{(N)}$  a sketch of the core tensor  $\mathcal{G}$ .

Given an activation tensor  $\mathcal{X} \in \mathbb{R}^{I_0 \times \dots \times I_{N-1}}$  and a target label vector  $\mathbf{y} \in \mathbb{R}^{I_N}$ , a stochastically rank regularized tensor regression layer is written from equation 3 as follows:

$$\mathbf{y} = \langle \mathcal{X}, \tilde{\mathcal{W}} \rangle_{N-1} \quad (4)$$

with  $\tilde{\mathcal{W}}$  being a stochastic approximation of Tucker decomposition, namely:

$$\tilde{\mathcal{W}} = \tilde{\mathcal{G}} \times_0 \tilde{\mathbf{U}}^{(0)} \times \dots \times_N \tilde{\mathbf{U}}^{(N)} \quad (5)$$

Even though several sketching methods have been proposed, we focus here on SRR with two different types of binary sketching matrices, namely binary matrix sketching with replacement and binary diagonal matrix sketching with Bernoulli entries.

### 4.1. SRR with replacement:

In this setting, we introduce the SRR with binary sketching matrix (with replacement). We first choose  $\theta \in [0, 1]$ .

Mathematically, we introduce the uniform sampling matrices  $\mathbf{M}^{(0)} \in \mathbb{R}^{R_0 \times R_0}, \dots, \mathbf{M}^{(N)} \in \mathbb{R}^{R_N \times R_N}$ .  $\mathbf{M}_j$  is a uniform sampling matrix, selecting  $K_j$  elements, where  $K_j = R_j \text{div } \theta$ . In other words, for any  $i \in [0 \dots N]$ ,  $\mathbf{M}^{(i)}$  verifies:

$$\mathbf{M}^{(i)}(j, :) = \begin{cases} \mathbf{0} & \text{if } j > K \\ \text{Id}_m(r, :), m \in [0 \dots R_i] & \text{otherwise} \end{cases} \quad (6)$$

Note that in practice this product is never explicitly computed, we simply select the correct elements from  $\mathcal{G}$  and its corresponding factors.

#### 4.2. Tucker-SRR with Bernoulli entries

In this setting, we introduce the SRR with diagonal binary sketching matrix with Bernoulli entries.

For any  $n \in [0 \dots N]$ , let  $\boldsymbol{\lambda}^{(n)} \in \mathbb{R}_n^R$  be a random vector, the entries of which are i.i.d. Bernoulli( $\theta$ ), then a diagonal Bernoulli sketching matrix is defined as  $\mathbf{M}^{(n)} = \text{diag}(\boldsymbol{\lambda}^{(n)})$ .

When the low-rank structure on the weight tensor  $\tilde{\mathcal{W}}$  of the TRL is imposed using a Tucker decomposition, the randomized Tucker approximation is expressed as:

$$\begin{aligned} \tilde{\mathcal{W}} &= \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_{N+1} \mathbf{M}^{(N)} \\ &\times_0 \left( \mathbf{U}^{(0)} (\mathbf{M}^{(0)})^\top \right) \times \dots \times_{N+1} \left( \mathbf{U}^{(N)} (\mathbf{M}^{(N)})^\top \right) \\ &= \llbracket \tilde{\mathcal{G}}; \tilde{\mathbf{U}}^{(0)}, \dots, \tilde{\mathbf{U}}^{(N)} \rrbracket \end{aligned} \quad (7)$$

The main advantage of considering the above-mentioned sampling matrices is that the products  $\tilde{\mathbf{U}}^{(k)} = \mathbf{U}^{(k)} (\mathbf{M}^{(k)})^\top$  or  $\tilde{\mathcal{G}} = \mathcal{G} \times_0 \mathbf{M}^{(0)} \times \dots \times_N \mathbf{M}^{(N)}$  are never explicitly computed, we simply select the elements from  $\mathcal{G}$  and the corresponding factors.

Interestingly, in analogy to dropout, where each hidden unit is dropped independently with probability  $1 - \theta$ , in the proposed randomized tensor decomposition, the columns of the factor matrices and the corresponding fibers of the core tensor are dropped independently and consequently the rank of the tensor decomposition is stochastically dropped. Hence the name *stochastic rank-regularized* TRL of our method.

#### 4.3. CP-SRR with Bernoulli entries

An interesting special case of 5 is when the weight tensor  $\tilde{\mathcal{W}}$  of the TRL is expressed using a CP decomposition. In that case, we set  $\mathbf{M} = \mathbf{M}^{(0)} = \dots = \mathbf{M}^{(N)} = \text{diag}(\boldsymbol{\lambda})$ , with, for any  $k \in [0 \dots R]$ ,  $\lambda_k \sim \text{Bernoulli}(\theta)$ .

Then a randomized CP approximation is expressed as:

$$\tilde{\mathcal{W}} = \sum_{k=0}^{R-1} \tilde{\mathbf{U}}_k^{(0)} \circ \dots \circ \tilde{\mathbf{U}}_k^{(N)} \quad (8)$$

The above randomized CP decomposition on the weights is equivalent to the following formulation:

$$\begin{aligned} \tilde{\mathcal{W}} &= \sum_{k=0}^{R-1} \lambda_k \mathbf{U}_k^{(0)} \circ \dots \circ \mathbf{U}_k^{(N)} \\ &= \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket \end{aligned} \quad (9)$$

This is easy to see by looking at the individual elements of the sketched factors. Let  $k \in [0 \dots N]$  and  $i_k \in [0 \dots I_k], r \in [0 \dots R-1]$ . Then  $\tilde{\mathbf{U}}_{i_k, r}^{(k)} = \sum_{j=0}^{R-1} \mathbf{U}_{i_k, j}^{(k)} \mathbf{M}_{j, r}$ . Since  $\mathbf{M} = \text{diag}(\boldsymbol{\lambda})$ , i.e.  $\forall i, j \in [0 \dots R-1], \mathbf{M}_{ij} = 0$  if  $i \neq j$ , and  $\lambda_i$  otherwise, we get  $\tilde{\mathbf{U}}_{i_k, r}^{(k)} = \lambda_r \mathbf{U}_{i_k, r}^{(k)}$ . It follows that  $\tilde{\mathcal{W}}_{i_0, i_1, \dots, i_N} = \sum_{r=0}^{R-1} \lambda_k \mathbf{U}_k^{(0)} \circ \dots \circ \lambda_k \mathbf{U}_k^{(N)}$ . Since  $\lambda_r \in \{0, 1\}$ , we have  $\tilde{\mathcal{W}}_{i_0, i_1, \dots, i_N} = \sum_{r=0}^{R-1} \lambda_k \left( \mathbf{U}_k^{(0)} \circ \dots \circ \mathbf{U}_k^{(N)} \right)$ .

Based on the previous stochastic regularization, for an activation tensor  $\mathcal{X}$  and a corresponding label vector  $\mathbf{y}$ , the optimization problem for our tensor regression layer with stochastic regularization is given by:

$$\min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \langle \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \quad (10)$$

In addition, the above stochastic optimization problem can be rewritten as a deterministic regularized problem:

$$\begin{aligned} \mathbb{E}_\lambda \left[ \min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \langle \llbracket \boldsymbol{\lambda}; \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \right] \\ = \min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \langle \llbracket \mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)} \rrbracket, \mathcal{X} \rangle_{N-1}\|_F^2 \\ + \left( \frac{1-\theta}{\theta} \right) \sum_{k=0}^{R-1} \left( \prod_{i=0}^N \|\mathbf{U}_{:,k}^{(i)}\|_2 \right)^2 \quad (11) \end{aligned}$$

This is easy to see by considering the equivalent rewriting of the above optimization problem, using the mode- $N$  unfolding of the weight tensor. Equation 10 then becomes:

$$\min_{\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(N)}} \|\mathbf{y} - \frac{1}{\theta} \mathbf{U}^{(N)} \text{diag}(\boldsymbol{\lambda}) (\mathbf{U}^{(-N)})^\top \text{vec}(\mathcal{X})\|_F^2$$

with  $\mathbf{U}^{(-N)} = (\mathbf{U}^{(0)} \circ \dots \circ \mathbf{U}^{(N)})$ . The result can then be obtained following Mianjy et al. (2018, Lemma A.1).

## 5. Experimental evaluation

In this section, we introduce the experimental setting, databases used, and implementation details. We experimented on several datasets, across various tasks, namely image classification and MRI-based regression. All methods were implemented using PyTorch (Paszke et al., 2017) and TensorLy (Kossaifi et al., 2016).

### 5.1. Numerical experiments

In this section, we empirically demonstrate the equivalence between our stochastic rank regularization and the deterministic regularization based formulation of the dropout.

To do so, we first created a random regression weight tensor  $\mathcal{W}$  to be a third order tensor of size  $(25 \times 25 \times 25)$ , formed as

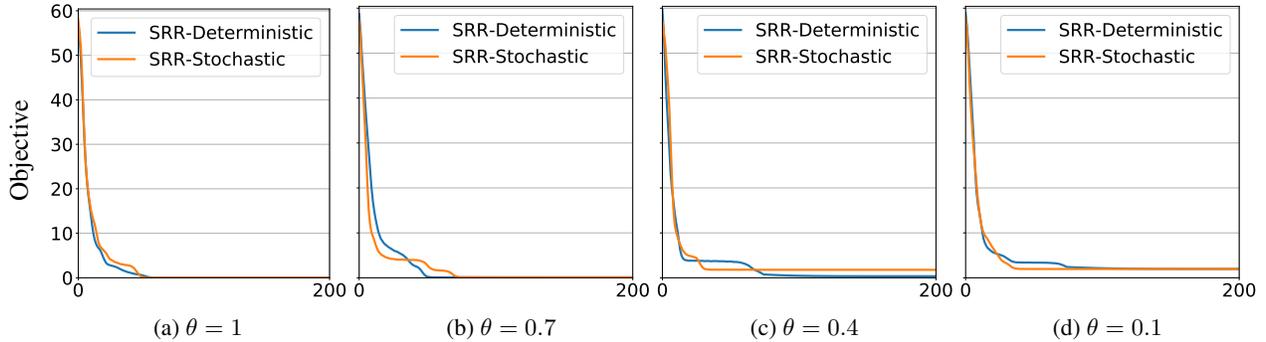


Figure 2. **Experiment on synthetic data:** loss of the TRL as a function of the number of epochs for the stochastic case (orange) and the deterministic version based on the regularized objective function (blue). As expected, both formulations are empirically the same.

a low-rank Kruskal tensor with 15 components, the factors of which were sampled from an i.i.d. Gaussian distribution. We then generated a tensor of 10000 random samples,  $\mathcal{X}$  of size  $(10000 \times 25 \times 25 \times 25)$ , the elements of which were sampled from a Normal distribution. Finally, we constructed the corresponding response array  $\mathbf{y}$  of size 10000 as:  $\forall i \in [1 \dots 1500], \mathbf{y}_i = \langle \mathcal{X}_i, \mathcal{W} \rangle$ . Using the same regression weight tensor and same procedure, we also generated 1000 testing samples and labels.

We use this data to train a rank-15 CP SRR-TRL, with both our Bernoulli stochastic formulation (equation 10) and its deterministic counter-part (equation 11). We train for 500 epochs, with a batch-size of 200, and an initial learning rate of  $10e - 4$ , which we decrease by a factor of 10 every 200 epochs. Figure 2 shows the loss function as a function of the epoch number. As expected, both formulations are identical.

## 5.2. Image classification results on CIFAR-100

In the image classification setting, we empirically compare our approach to both standard baseline and traditional tensor regression, and assess the robustness of each method in the face of adversarial noise.

**CIFAR-100** consists of 60,000  $32 \times 32$  RGB images in 100 classes (Krizhevsky & Hinton, 2009). We pre-processed the data by centering and scaling each image and then augmented the training images with random cropping and random horizontal flipping.

We compare the stochastic regularization tensor regression layer to full-rank tensor regression, average pooling and a fully-connected layer in an 18-layer residual network (ResNet) (He et al., 2016). For all networks, we used a batch size of 128 and trained for 400 epochs, and minimized the cross-entropy loss using stochastic gradient descent (SGD). The initial learning rate was set to 0.01 and lowered by a factor of 10 at epochs 150, 250 and 350. We used a weight decay ( $L_2$  penalty) of  $10^{-4}$  and a momentum of 0.9.

**Results:** Table 1 presents results obtained on the CIFAR-100 dataset, on which our method matches or outperforms other methods, including the same architectures without SRR. Our regularization method makes the network more robust by reducing over-fitting, thus allowing for superior performance on the testing set.

Table 1. Classification accuracy for CIFAR-100

Architecture	Accuracy
ResNet without pooling	73.31 %
ResNet	75.88 %
ResNet with TRL	76.02 %
ResNet with Tucker SRR	76.05 %
ResNet with CP SRR	<b>76.19 %</b>

A natural question is whether the model is sensitive to the choice of rank and  $\theta$  (or drop rate when sampling with repetition). To assess this, we show the performance as a function of both rank and  $\theta$  in figure 3. As can be observed, there is a large surface for which performance remains the same while decreasing both parameters (note the logarithmic scale for the rank). This means that, in practice, choosing good values for these is not a problem.

**Robustness to adversarial attacks:** We test for robustness to adversarial examples produced using the Fast Gradient Sign Method (Kurakin et al., 2016) in Foolbox (Rauber et al., 2017). In this method, the sign of the optimization gradient multiplied by the perturbation magnitude is added to the image in a single iteration. The perturbations we used are of magnitudes  $\lambda \times 10^{-3}$ ,  $\lambda \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ .

In addition to improving performance by reducing over-fitting, our proposed stochastic regularization makes the model more robust to perturbations in the input, for both random noise and adversarial attacks.

We tested the robustness of our models to adversarial attacks, when trained in the same configuration. In figure 4,

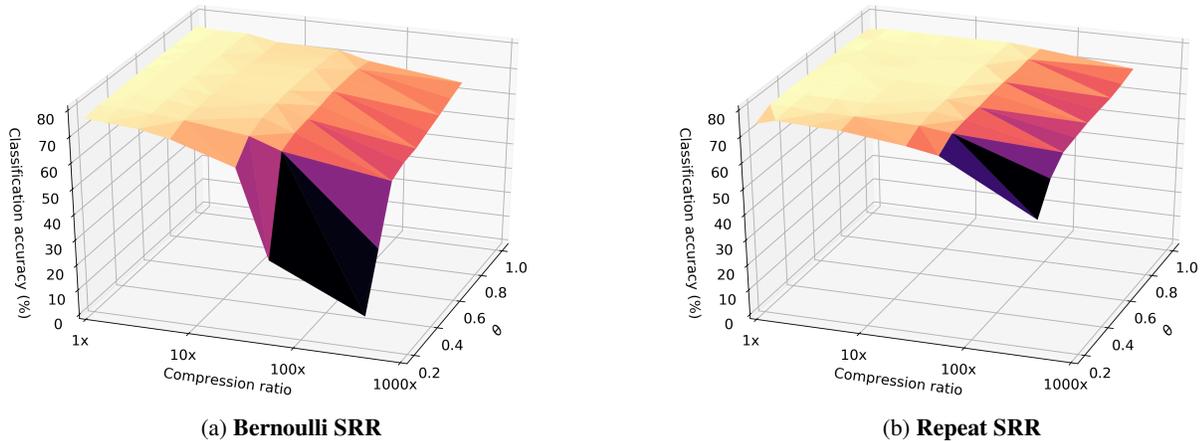


Figure 3. CIFAR-100 test accuracy as a function of the compression ratio (logarithmic scale) and the Bernoulli probability  $\theta$  (left) or the drop rate (right). There is a large region for which dropping both the rank and  $\theta$  does not hurt performance.

we report the classification accuracy on the test set, as a function of the added adversarial noise. The models were trained *without* any adversarial training, on the training set, and adversarial noise was added to the test samples using the Fast Gradient Sign method. Our model is much more robust to adversarial attacks. Finally, we perform a thorough comparison of the various regularization strategies, the results of which can be seen in figure 5.

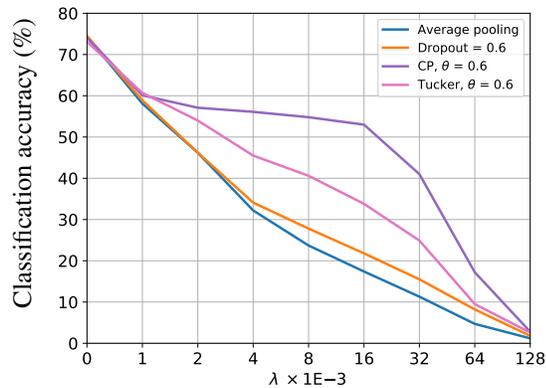


Figure 4. **Robustness to adversarial attacks** using Fast Gradient Sign attacks of various models, trained on CIFAR-100. Our stochastically rank-regularized architecture is much more robust to adversarial attacks, even though adversarial training was not used.

### 5.3. Phenotypic trait prediction from MRI data

In the regression setting, we investigate the performance of our SRR-TRL in a challenging, real-life application, on a very large-scale dataset. This case is particularly interesting since the MRI volumes are large 3D tensors, all modes of which carry important information. The spatial information is traditionally discarded during the flattening process,

which we avoid by using a tensor regression layer.

**The UK Biobank brain MRI dataset** is the world’s largest MRI imaging database of its kind (Sudlow et al., 2015). The aim of the UK Biobank Imaging Study is to capture MRI scans of vital organs for 100,000 primarily healthy individuals by 2022. Associations between these images and lifestyle factors and health outcomes, both of which are already available in the UK Biobank, will enable researchers to improve diagnoses and treatments for numerous diseases. The data we use here consists of T1-weighted  $182 \times 218 \times 182$  MR images of the brain for 7,500 individuals captured on a 3 T Siemens Skyra system. 5,700 are used for training and rest are used to test and validate. The target label is the age for each individual at the time of MRI capture. We use skull-stripped images that have been aligned to the MNI152 template (Jenkinson et al., 2002) for head-size normalization. We then center and scale each image to zero mean and unit variance for intensity normalization.

Table 2. Classification accuracy for UK Biobank MRI. The ResNet with TRL and our stochastic rank-regularization performs better, while the baseline ResNet without average pooling did not train at all. The version *with* average pooling did train but converged to a much worse performance.

Architecture	MAE
3D-ResNet without pooling	N/A
3D-ResNet	3.23 years
3D-ResNet with TRL	2.99 years
3D-ResNet with Tucker SRR	2.96 years
3D-ResNet with CP SRR	<b>2.58 years</b>

**Results:** For MRI-based experiments we implement an 18-layer ResNet with three-dimensional convolutions. We

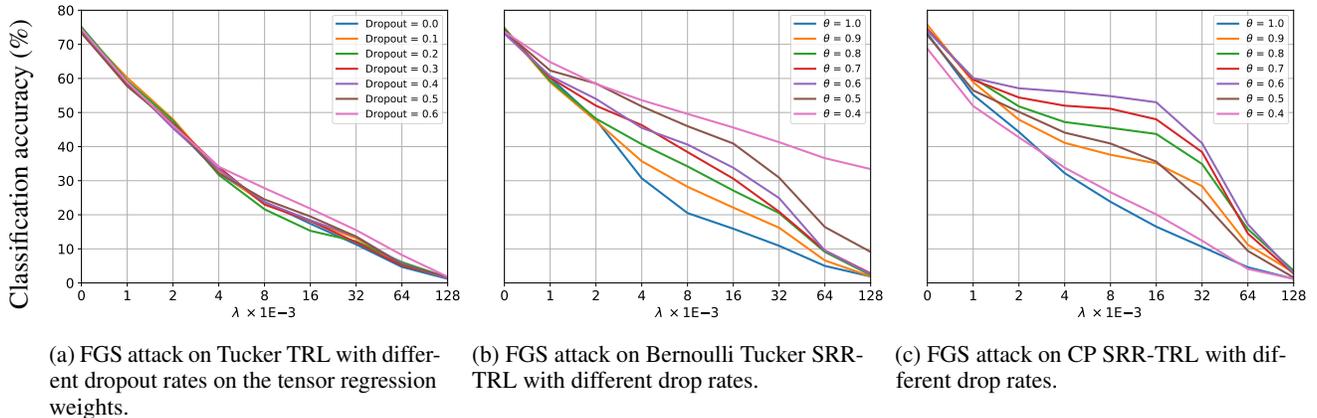


Figure 5. **Robustness to adversarial attacks**, measured by adding adversarial noise to the test images, using the Fast Gradient Sign, on CIFAR-100 and Bernoulli drop. We compare a Tucker tensor regression layer with dropout applied to the regression weight tensor 5a to our stochastic rank-regularized TRL, both in the Tucker (Subfig. 5b) and CP (Subfig. 5c) case.

minimize the mean squared error using Adam (Kingma & Ba, 2014), starting with an initial learning rate of  $10^{-4}$ , reduced by a factor of 10 at epochs 25, 50, and 75. We train for 100 epochs with a mini-batch size of 8 and a weight decay ( $L_2$  penalty) of  $5 \times 10^{-4}$ . As previously observed, our Stochastic Rank Regularized tensor regression network outperforms the ResNet baseline by a large margin, Table 2. To put this into context, the current state-of-art for convolutional neural networks on age prediction from brain MRI on most datasets is an MAE of around 3.6 years (Cole et al., 2017a; Herent et al., 2018).

**Robustness to noise:** We tested the robustness of our model to white Gaussian noise added to the MRI data. Noise in MRI data typically follows a Rician distribution but can be approximated by a Gaussian for signal-to-noise ratios (SNR) greater than 2 (Gudbjartsson & Patz, 1995). As both the signal (MRI voxel intensities) and noise are zero-mean, we define  $SNR = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{noise}}^2}$ , where  $\sigma$  is the variance. We incrementally increase the added noise in the test set and compare the error rate of the models.

The ResNet with SRR is significantly more robust to added white Gaussian noise compared to the same architectures without SRR (figure 6). At signal-to-noise ratios below 10, the accuracy of a standard ResNet with average pooling is worse than a model that predicts the mean of training set (MAE = 7.9 years). Brain morphology is an important attribute that has been associated with various biological traits including cognitive function and overall health (Pfefferbaum et al., 1994; Swan et al., 1998). By keeping the structure of the brain represented in MRI in every layer of the architecture, the model has more information to learn a more accurate representation of the entire input. Additionally, the stochastic dropping of ranks forces the representation to be robust to confounds. This a particularly important prop-

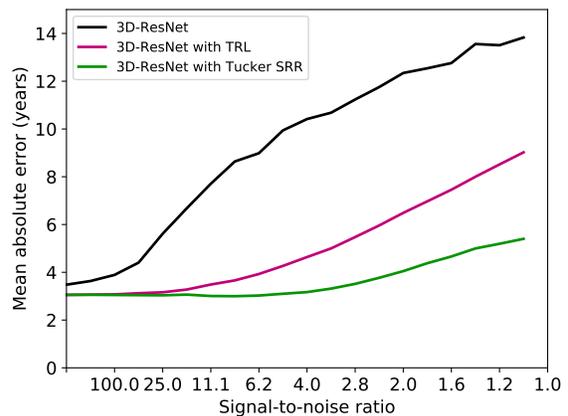


Figure 6. Age prediction error on the MRI test set as a function of increased added noise.

erty for MRI analysis since intensities and noise artifacts can vary significantly between MRI scanners (Wang et al., 1998). SRR enables both more accurate and more robust trait predictions from MRI that can consequently lead to more accurate disease diagnoses.

## 6. Conclusion

We introduced the stochastic rank-regularized tensor regression networks. By adding rank-randomization during training, this renders the network more robust and lead to better performance. This also translates to more stable training, and networks less prone to over-fitting. The low-rank, robust representation also makes the network more resilient to noise, both adversarial and random. Our results demonstrate superior performance and convergence on a variety of challenging tasks, including MRI data and images.

## Acknowledgements

This research has been conducted using the UK Biobank Resource under Application Number 18545.

## References

- Battaglino, C., Ballard, G., and Kolda, T. G. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.
- Cao, X., Rabusseau, G., and Pineau, J. Tensor regression networks with various low-rank tensor approximations. *CoRR*, abs/1712.09520, 2017.
- Caruana, R., Lawrence, S., and Giles, C. L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pp. 402–408, 2001.
- Chen, W., Wilson, J., Tyree, S., Weinberger, K. Q., and Chen, Y. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1475–1484. ACM, 2016.
- Cheng, Y., Yu, F. X., Feris, R. S., Kumar, S., Choudhary, A., and Chang, S.-F. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2857–2865, 2015.
- Cole, J. H., Poudel, R. P., Tsagkrasoulis, D., Caan, M. W., Steves, C., Spector, T. D., and Montana, G. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163:115–124, 2017a.
- Cole, J. H., Ritchie, S. J., Bastin, M. E., Hernández, M. V., Maniega, S. M., Royle, N., Corley, J., Pattie, A., Harris, S. E., Zhang, Q., et al. Brain age predicts mortality. *Molecular psychiatry*, 2017b.
- Daniely, A., Lazic, N., Singer, Y., and Talwar, K. Sketching and neural networks. *arXiv preprint arXiv:1604.05753*, 2016.
- Erichson, N. B., Manohar, K., Brunton, S. L., and Kutz, J. N. Randomized cp tensor decomposition. *arXiv preprint arXiv:1703.09074*, 2017.
- Franke, K., Luders, E., May, A., Wilke, M., and Gaser, C. Brain maturation: predicting individual brainage in children and adolescents using structural mri. *Neuroimage*, 63(3):1305–1312, 2012.
- Gudbjartsson, H. and Patz, S. The rician distribution of noisy mri data. *Magnetic resonance in medicine*, 34(6):910–914, 1995.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Herent, P., Jegou, S., Wainrib, G., and Clozel, T. Brain age prediction of healthy subjects on anatomic mri with deep learning: going beyond with an “explainable ai” mindset. *bioRxiv*, pp. 413302, 2018.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- Jenkinson, M., Bannister, P., Brady, M., and Smith, S. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *Neuroimage*, 17(2):825–841, 2002.
- Kasiviswanathan, S. P., Narodytska, N., and Jin, H. Deep neural network approximation using tensor sketching. *arXiv preprint arXiv:1710.07850*, 2017.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kossaifi, J., Panagakis, Y., and Pantic, M. Tensorly: Tensor learning in python. *arXiv preprint arXiv:1610.09555*, 2016.
- Kossaifi, J., Khanna, A., Lipton, Z., Furlanello, T., and Anandkumar, A. Tensor contraction layers for parsimonious deep nets. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1940–1946. IEEE, 2017.
- Kossaifi, J., Lipton, Z. C., Khanna, A., Furlanello, T., and Anandkumar, A. Tensor regression networks. *CoRR*, abs/1707.08308, 2018.
- Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pp. 950–957, 1992.
- Kukačka, J., Golkov, V., and Cremers, D. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.
- Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.
- Mianjy, P., Arora, R., and Vidal, R. On the implicit bias of dropout. In Dy, J. and Krause, A. (eds.), *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3540–3548, Stockholm, Sweden, 10–15 Jul 2018. PMLR.
- Nowlan, S. J. and Hinton, G. E. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.
- Pfefferbaum, A., Mathalon, D. H., Sullivan, E. V., Rawles, J. M., Zipursky, R. B., and Lim, K. O. A quantitative magnetic resonance imaging study of changes in brain morphology from infancy to late adulthood. *Archives of neurology*, 51(9):874–887, 1994.
- Rauber, J., Brendel, W., and Bethge, M. Foolbox: a python toolbox to benchmark the robustness of machine learning models (2017). [URL http://arxiv.org/abs/1707.04131](http://arxiv.org/abs/1707.04131), 2017.

- Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.
- Sidiropoulos, N. D., Papalexakis, E. E., and Faloutsos, C. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine*, 31(5):57–70, 2014.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., 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.
- Swan, G. E., DeCarli, C., Miller, B., Reed, T., Wolf, P., Jack, L., and Carmelli, D. Association of midlife blood pressure to late-life cognitive decline and brain morphology. *Neurology*, 51(4):986–993, 1998.
- Tai, C., Xiao, T., Zhang, Y., Wang, X., et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.
- Tsourakakis, C. E. Mach: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 689–700. SIAM, 2010.
- Vervliet, N., Debals, O., Sorber, L., and De Lathauwer, L. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Processing Magazine*, 31(5):71–79, 2014.
- Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pp. 1058–1066, 2013.
- Wang, L., Lai, H.-M., Barker, G. J., Miller, D. H., and Tofts, P. S. Correction for variations in mri scanner sensitivity in brain studies with histogram matching. *Magnetic resonance in medicine*, 39(2):322–327, 1998.
- Wang, Y., Tung, H.-Y., Smola, A. J., and Anandkumar, A. Fast and guaranteed tensor decomposition via sketching. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems (NIPS)*, pp. 991–999. 2015.
- Yu, X., Liu, T., Wang, X., and Tao, D. On compressing deep models by low rank and sparse decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 67–76. IEEE, 2017.
- Yuan, L., Li, C., Cao, J., and Zhao, Q. Randomized tensor ring decomposition and its application to large-scale data reconstruction. *arXiv preprint arXiv:1901.01652*, 2019.
- Zhang, Y., Lee, J. D., and Jordan, M. I.  $\ell_1$ -regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*, pp. 993–1001, 2016.
- Zhou, G., Cichocki, A., and Xie, S. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014.