Scaling Recurrent Models via Orthogonal Approximations in Tensor Trains

Ronak Mehta[1], Rudrasis Chakraborty[2], Yunyang Xiong[1], and Vikas Singh[1]

ronakrm@cs.wisc.edu rudra@berkeley.edu yxiong43@wisc.edu vsingh@biostat.wisc.edu
[1]University of Wisconsin Madison [2]University of California Berkeley

Abstract

Modern deep networks have proven to be very effective for analyzing real world images. However, their application in medical imaging is still in its early stages, primarily due to the large size of three-dimensional images, requiring enormous convolutional or fully connected layers – if we treat an image (and not image patches) as a sample. These issues only compound when the focus moves towards longitudinal analysis of 3D image volumes through recurrent structures, and when a point estimate of model parameters is insufficient in scientific applications where a reliability measure is necessary. Using insights from differential geometry, we adapt the tensor train decomposition to construct networks with significantly fewer parameters, allowing us to train powerful recurrent networks on whole brain image volume sequences. We describe the "orthogonal" tensor train, and demonstrate its ability to express a standard network layer both theoretically and empirically. We show its ability to effectively reconstruct whole brain volumes with faster convergence and stronger confidence intervals compared to the standard tensor train decomposition. We provide code and show experiments on the ADNI dataset using image sequences to regress on a cognition related outcome.

1. Introduction

Recurrent Neural networks (RNNs) and its variants are the de facto tool of choice for modeling sequential data in machine learning and vision. But until only recently, these models have been limited in their ability to model high-dimensional data. Part of the reason is that recurrent structures often lead to large model sizes dependent on sequence length, and thus also require an equivalent number of increased computation. While RNNs have been successfully applied to video data in some cases, the strategy requires problem specific innovations because of the large mapping necessary from inputs to hidden representations. It is fair to say that the growth in the number of model parameters in various types of recurrent models remains a bottleneck for high dimensional datasets. Convolutional neural networks (CNNs), on the other hand, handle high dimensional data far better and can reduce the dimension of an input significantly by deriving rich feature maps. Most computer vision tasks involve some form of a CNN within the architecture, but incorporating CNNs within recurrent structures seamlessly to mitigate the RNN specific model size issues described above is not always straightforward. Notice that a direct replacement of input and output layers with CNNs leads to a shrinkage of the sequence length considerably [24], and pretraining CNN layers may lead to poor local minima when we train without using an end-to-end pipeline [7]. Some recent works suggest the use of dilated convolutional networks for sequence modeling [28] to partly mitigate these issues, but this line of work is still developing [31]. For model-size reduction, both for RNN style networks and otherwise, PCA or random projections [2,27] style "compression" ideas have also been used with varying degrees of success.

An interesting perspective on the effective degrees of freedom afforded by a given network, a surrogate for the actual "size" of the architecture, is provided by tensor methods. Tensor decomposition based methods have recently been shown to enable low dimensional representations of very high dimensional data [13], and while these ideas were known to be effective in the "shallow" regime much earlier, new results also demonstrate their applicability for deep neural networks. In particular, in the last year, we see a number of tensor based methods being successfully adapted for deep neural network design and compression [4, 25, 29, 30]. Specifically, [26] shows that these compression methods can be very effective in reducing the parameter cost of weight layers in RNNs, enabling simple video analysis tasks that previously would have been computationally prohibitive.

There are a number of key reasons why the size of the model, especially in the context of formulations for sequential data, is central to this paper. Our goal is to design rich sequential or recurrent models to analyze a longitudinal sequence of high dimensional 3D brain images. This task raises two issues. **First**, unless the model size is parsimonious, we find that merely instantiating the model with data involving 3D images over multiple time points, even on multiple high end GPU instances, is challenging. **Second**, the eventual goal of medical image analysis is either scientific discovery or generating actionable knowledge for the patient. Both goals require evaluating a model's confidence via classical or contemporary statistical techniques: for instance, how confident is the model of its prediction? Most, if not all, available tools for assessing model uncertainty of deep neural network models have a strong dependence on the number of parameters in the model. Therefore, even if the first issue above could be mitigated by clever implementation ideas, purely as a practical matter, the design of rich and expressive models with a small number of parameters yields immense benefits for calculating model uncertainty.

This paper and its contributions. We tackle the problem of modeling sequential 3D brain imaging data using recurrent/sequential models. Our development starts from well known results on tensor decomposition. In particular, we make use of the tensor train representation, which has been shown to be effective in several applications in vision and machine learning. We derive a reformulation of the decomposition using orthogonality constraints and show that while this makes the estimation slightly more challenging, it **reduces the number of parameters by as much as half**. We present a novel parameter estimation scheme based on Stiefel manifold optimization and demonstrate how the end to end construction yields benefits for convergence and uncertainty estimation. Finally, from the empirical side, we discuss how we enable analysis of and prediction using sequential 3D brain imaging datasets, which to our knowledge is the first such result using deep recurrent architectures.

2. Preliminaries

2.1. Tensor Decompositions and Tensor Trains.

Let $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a d-dimensional array, or tensor, with each mode having length n_i. To store a full rank tensor, n^d storage would be required. A number of tensor factorizations have been developed to reduce this storage cost. The CANDECOMP/PARAFAC (CP) [3, 10] decomposition reduces the storage to $O(dnr)$, but finding the exact CP-rank r is NP-hard. Hierarchical tensor methods have also proven to be effective in tensor compression [4, 5].

A more recent decomposition, the *Tensor Train* decomposition (TT) [22], defines an element of the tensor as

$$\mathcal{X}(x_1, \dots, x_d) = A_1(x_1) \cdots A_d(x_d) \tag{1}$$

where $x_i \in \{1, \dots, n_i\}$, and $A_i(x_i) \in \mathbb{R}^{r_{i-1} \times r_i}$ for each $i \in \{1, \dots, d\}$ are called the *cores* of the tensor train, with $r_0 = r_d = 1$. Equivalently, the full tensor is written as:

$$\mathcal{X} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} A_1(k_0, :, k_1) \otimes \cdots \otimes A_d(k_{d-1}, :, k_d) \tag{2}$$

where $A_i \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$. This format requires $O(dnr^2)$ storage, but has two major advantages over the CP format. First, finding the TT-rank (the smallest set of r_i's that satisfy the decomposition with equality) of any arbitrary tensor is tractable, and as such all tensors can be efficiently rewritten in the TT format. Second, projecting arbitrary tensors onto the TT format of a fixed rank requires only a set of QR and singular value decompositions [22]. This projection, *TT-rounding*, additionally allows for a given TT tensor of some rank to be projected onto the space of TTs with lower rank, and requires $O(dr^3)$ computational complexity. Separately, specific tensor train constructions have recently been identified as forms of general recurrent networks [15].

We denote a tensor operator $\mathcal{G}$ as a grouping of tensor modes into an "input" and "output" list, such that $\mathcal{G} \in \mathbb{R}^{(n_1^{in}, \times \dots \times, n_d^{in}) \times (n_1^{out}, \times \dots \times, n_d^{out})}$. This operator $\mathcal{G}$ can be seen as the TT representation of a matrix $W \in \mathbb{R}^{(n_1^i \cdots n_d^i) \times (n_1^o \cdots n_d^o)}$. In [20], authors use this formulation to directly compress the weight layers in neural networks. Cores in the operator are indexed by both an input and output index, i.e., $A_i(x_i, y_i) \in \mathbb{R}^{r_{i-1} \times r_i}$, where $x_i \in [1, \dots, n_i^{in}], y_i \in [1, \dots, n_i^{out}]$.

Common operations on tensor trains require *matricizing* the cores of the TT format. Here, we define the left matricization of core $A_i(x_i)$ as $A_i^L \in \mathbb{R}^{r_{i-1} n_i \times r_i}$ and the right matricization similarly.

2.2. Differential Geometry of Tensor Trains

Tensor trains with fixed TT-ranks form a Riemannian submanifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ [12, 18]:

$$\mathcal{M}_r := \{\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \text{ with TT-ranks } r_0, \dots, r_d\} \tag{3}$$

A Riemannian manifold is a differentiable manifold with a smoothly varying inner product. The tangent space is a vector space defined at a specific point on the manifold, consisting of all possible tangent vectors of all possible curves along the manifold passing through that point. The tangent bundle, the disjoint union of all tangent spaces for all points on the manifold, is canonically equipped with a *projection map*: $\Pi : T\mathcal{M} \to \mathcal{M}$. The Exponential Map defines a local map from the tangent space at a specific point on the manifold $\text{Exp}(x, \cdot) : T_x\mathcal{M} \to \mathcal{M}$. With these definitions, optimizing a function with respect to a Riemannian manifold-valued variable amounts to computing a free derivative in the ambient space, projecting the gradient to the tangent space of the current iterate, and using the (retraction) exponential map to compute the next iterate on the manifold. The authors in [21] use this procedure to more effectively learn a model of all exponentially many interactions in a linear model.

Orthogonal matrices of fixed size and rank also form a manifold, the (compact) Stiefel Manifold: $\mathrm{St}(p, n) = \{Y \in \mathbb{R}^{n\times p} | Y^T Y = I_p,\ p \le n\}$. An arbitrary $X \in \mathbb{R}^{n\times p}$ matrix can be projected onto the Stiefel manifold $\mathrm{St}(p, n)$ using $X \mapsto UV^T$ where $X = U\Sigma V^T$ is the (thin) singular value decomposition of X.

3. Orthogonal Tensor Trains

As described above, a number of TT operations with respect to approximation and projection require computing the QR decomposition of matricized cores. In the applications for which tensor trains were originally developed, these operations were necessary [17,22]. For modern neural network applications, where the tensor operator may be our target of learning, it may be sufficient to treat each matrix product as its own variable, and through the standard TT decomposition learn the cores along the **product of Stiefels**.

A naïve approach may orthogonalize the reshaped cores, and progressively push the upper triangular part of the core decomposition into the next core, resulting in the following exact formulation with appropriate reshaping:

$$\begin{aligned}\mathcal{X} &= A_1^L A_2^L \cdots A_d^L \\ &= Q_1^L R_1 A_2^L \cdots A_d^L = Q_1^L \left(R_1 A_2^L\right) \cdots A_d^L \\ &= Q_1^L Q_2^L R_2 \cdots A_d^L \\ &= Q_1^L \cdots Q_d^L R_d \end{aligned} \tag{4}$$

where $[Q_1^L, R_1] = qr(A_1^L)$, $[Q_i^L, R_i] = qr(R_{i-1}A_i^L),\ i \in \{2,\dots,d\}$, and $Q_i^L \in \mathbb{R}^{r_{i-1}n_i\times r_i}$, $R_i \in \mathbb{R}^{r_i\times r_i}$. Each Q_i^L is on a Stiefel given by $\mathrm{St}(r_i, r_{i-1}n_i)$. Here, the number of components in the product space of Stiefels is d, with the 'residual' $R_d \in \mathbb{R}$. This decomposition is exact and only requires a reshaping of the tensor cores. If all $r_i = r, n_i = n$, then the total number of parameters needed is $dnr^2 - (d-1)\frac{r^2+r}{2}$, compared to the full format with dnr^2 total parameters. It is important to note that in this formulation, the cores themselves are **not** orthogonal. Reshaping is required to bring the matricized form back to TT-cores of size $r_{i-1} \times r_i$, and in practice it is not easy to perform simple TT-tensor multiplication in this form. Additionally, we now need to optimize over Stiefel manifolds of a larger size, namely $O(nr^2)$.

3.1. A Nicer Tensor Train Approximation

Ideally, we would prefer a construction which keeps the standard TT-core format and involves optimization over "smaller" Stiefel manifolds. Consider the following representation, in which each TT-core itself is orthogonal.

Definition 1. *(Orthogonal Tensor Train) The Orthogonal Tensor Train is defined as*

$$\mathcal{X}(x_1,\dots,x_d) = Q_1(x_1)\cdots Q_d(x_d), \tag{5}$$

where each $Q_i(x_i)$ lies on the Stiefel $\mathrm{St}(m_i, M_i)$, where $m_i = \min(r_{i-1}, r_i),\ M_i = \max(r_{i-1}, r_i)$.

While in this formulation the total number of components in the product space of Stiefels is nd, the dimension of each manifold is **significantly smaller**, dependent *only* on the core rank as opposed to the mode size. The total number of parameters, if $n_i = n, r_i = r$, is

$$n\sum_{i=1}^{d}\left[r^2 - \frac{r^2+r}{2}\right] = dnr^2 - dn\frac{r^2+r}{2}. \tag{6}$$

When compared to the full TT representation, the Orthogonal Tensor Decomposition (OTT) requires $(r+1)/2r \approx 1/2$ as many parameters. If $r_i = r_{i+1}$, then $\mathrm{St}(m_i, M_i) = SO(m_i)$, where SO is the special orthogonal group.

This construction can be seen as an approximation to the full tensor train format, in which the upper triangular part of each core is set to identity:

$$\begin{aligned}\mathcal{X}(x_1,\dots,x_d) &= A_1(x_1)\cdots A_d(x_d) \\ &= Q_1(x_1)R_1(x_1)\cdots Q_d(x_d)R_d(X_d) \\ &\approx Q_1(x_1)\cdots Q_d(x_d) \end{aligned} \tag{7}$$

Is this useful? It is not obvious that this construction is useful at all. How much is lost through this approximation? What is gained by using this construction? In what follows, we demonstrate that we can approximate any tensor with bounded norm using an OTT, and that with a full rank assumption and a trainable constant, our formulation admits a solution with ϵ error.

3.2. Theoretical Analysis

We start by reshaping any tensor $\mathcal{X}$ to a matrix X^M by grouping the modes into two groups, $X^M \in \mathbb{R}^{n\times m}$. We may fix this arbitrary matrix as $X^M = A \in \mathbb{R}^{n\times m}$.

Proposition 1. *Given a 2D tensor $A \in \mathbb{R}^{m\times m}$, $A_{ij} \in [-1, 1]$, there exists sets of unit vectors, $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^m$, $\{\mathbf{y}_j\}_{j=1}^m \subset \mathbb{R}^m$ such that, $\forall\epsilon > 0$, $\|A - \widetilde{A}\| < \epsilon$, where, $\forall i, j,\ \widetilde{A}_{ij} = \mathbf{x}_i^t \mathbf{y}_j$.*

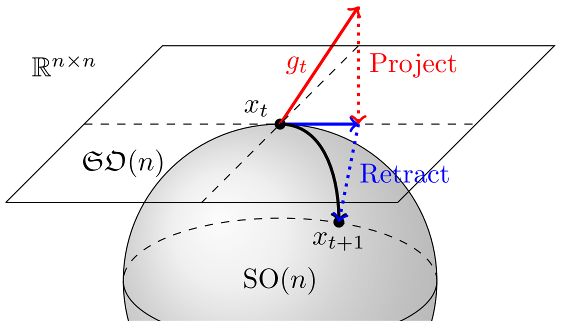
Proof. Let $A = USV^T$ be the SVD of A. Let $\epsilon > 0$, we will perturb S along the diagonal to generate $\widetilde{S}$ such that, $\|S - \widetilde{S}\| < \epsilon$. Let $X = [\mathbf{x}_i]$ and $Y = [\mathbf{y}_i]$. We will first give an algorithm to generate $\widetilde{X}$ and $\widetilde{Y}$ with each of its column being orthonormal such that, $\widetilde{X}^T\widetilde{Y} = S$. Then, $X = \widetilde{X}U^T$ and $Y = \widetilde{Y}V^T$.

We begin with an algorithm for $m = 3$. Choose $\{\widetilde{\mathbf{x}}_i\}$ to be unit vectors and assign $\widetilde{\mathbf{y}}_3 = \widetilde{\mathbf{x}}_1 \times \widetilde{\mathbf{x}}_2$, $\widetilde{\mathbf{y}}_2 = \widetilde{\mathbf{x}}_3 \times \widetilde{\mathbf{x}}_1$.

Stochastic OTT Optimization

```
for t=1,...,T do
    $g_t := \frac{df}{d\mathcal{W}} f(X^{mini-batch})$
    for Core $Q_t^i \in \mathcal{W}_t$ and Core Gradient $g_t^i \in g_t$ do
        $G_t^i = P_{T_{\mathcal{W}_t} M}(g_t^i)$                ▷ Projection Step
        $Q_{t+1}^i \leftarrow \mathrm{Exp}(Q_t^i, G_t^i)$            ▷ Retraction Step
    end for
end for
```

(a)



(b)

Figure 1: Algorithm (a) and visualization (b) of the gradient descent update using the projection and retraction on the Stiefel manifold. The update is applied to each core individually, allowing for smaller manifold operations that would otherwise scale poorly with dimension.

Then, make $\widetilde{\mathbf{y}}_2$ and $\widetilde{\mathbf{y}}_3$ to be of unit length. Now, rotate $\widetilde{\mathbf{x}}_2$ in the plane spanned by $\{\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2\}$ such that, $\widetilde{\mathbf{x}}_2^t\widetilde{\mathbf{y}}_2 = \widetilde{S}_{22}$. Similarly, rotate $\widetilde{\mathbf{x}}_3$ in the plane spanned by $\{\widetilde{\mathbf{x}}_3, \widetilde{\mathbf{x}}_1\}$ such that, $\widetilde{\mathbf{x}}_3^t\widetilde{\mathbf{y}}_3 = \widetilde{S}_{33}$. Now, assign, $\widetilde{\mathbf{y}}_1 = \widetilde{\mathbf{x}}_2 \times \widetilde{\mathbf{x}}_3$ and make it unit length. Now, fixing $\widetilde{\mathbf{x}}_2$ and $\widetilde{\mathbf{x}}_3$, the above steps are a continuous mapping, F from $\mathbf{S}^2$ to $[-1,1]$, i.e., by changing different $\widetilde{\mathbf{x}}_1 \in \mathbf{S}^2$, we will get different values for $\widetilde{\mathbf{x}}_1^t\widetilde{\mathbf{y}}_1$. Also, notice that, if, for a particular choice of $\{\widetilde{\mathbf{x}}_i\}$, $\widetilde{\mathbf{x}}_1^t\widetilde{\mathbf{y}}_1 > 0$, then, for the choice of $\{\widetilde{-\mathbf{x}}_i\}$, the above construction returns $-\widetilde{\mathbf{y}}_2$ and $-\widetilde{\mathbf{y}}_3$ and F returns, $-\widetilde{\mathbf{x}}_1^t\widetilde{\mathbf{y}}_1 < 0$. Thus if $a \in F(\mathbf{S}^2)$, $-a \in F(\mathbf{S}^2)$. Furthermore, $1 \in F(\mathbf{S}^2)$ and hence, $-1 \in F(\mathbf{S}^2)$. As $\mathbf{S}^2$ is connected and F is continuous, $F(\mathbf{S}^2)$ is connected, and so, $\exists\, \{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^m$ and $\{\mathbf{y}_j\}_{j=1}^m \subset \mathbb{R}^m$, s.t., $(\forall \{i,j\})$, $\widetilde{\mathbf{x}}_i^t\widetilde{\mathbf{y}}_j = \widetilde{S}_{ij}$. Since $\|S - \widetilde{S}\| < \epsilon$ and the choice of $\epsilon > 0$ is arbitrary, we can see that $\|A - \widetilde{A}\| < \epsilon$.

Using the generalization of cross product by exterior algebra, the above procedure can be naturally extended to arbitrary $m > 3$. □

A direct corollary of the above result allows approximating an arbitrary 2D matrix,

Corollary 1. *Given a 2D tensor $A \in \mathbb{R}^{m\times m}$, there exists sets of unit vectors, $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^m$, $\{\mathbf{y}_j\}_{j=1}^m \subset \mathbb{R}^m$ and fixed constant c such that, $\forall \epsilon > 0$, $\|A - \widetilde{A}\| < \epsilon$, where, $\forall i,j$, $\widetilde{A}_{ij} = c\mathbf{x}_i^t\mathbf{y}_j$.*

Proof. Given any arbitrary matrix A, define $A' = A/|A|_\infty$. Then $A'_{ij} \in [-1,1]$, and by Proposition 1 we can construct unit vectors $\mathbf{x}_i, \mathbf{y}_j$ such that $\forall \epsilon > 0$, $\|A'_{ij} - \mathbf{x}_i^T\mathbf{y}_j\| < \epsilon$. Then immediately $\forall A_{ij}$, we have $A_{ij} = cA'_{ij}$ where $c = |A|_\infty$. □

We also have the following directly from Proposition 1.

Corollary 2. *Given a 2D tensor $A \in \mathbb{R}^{m\times m}$, with $\|A\|_F \leq 1$, there exists a set of orthonormal matrices $\{B_i\} \subset SO(m)$ and a set of unit vectors $\{\mathbf{y}_j\}_{i=1}^m \subset \mathbb{R}^m$ such that $\forall \epsilon > 0$, $\|A - \widetilde{A}\| < \epsilon$, where, $\forall i,j$, $\widetilde{A}_{ij} = \mathbf{1}^t B_i^t \mathbf{y}_j$.*

Example 3.1. Applying the above result to OTT, equivalence is relatively straightforward to show. Consider the problem of approximating a 4 dimensional tensor $\mathcal{X}$ with $n_{1,2,3,4} = n = r$. Let $Q_1(x_1) \in \mathbb{R}^{1\times n}, Q_2(x_2), Q_3(x_3) \in \mathbb{R}^{n\times n}$, and $Q_4(x_4) \in \mathbb{R}^{n\times 1}$. By Corollary 2 we can write two vectors indexed by x_1, x_2 and x_3, x_4 as $X^A(x_1,x_2) = Q_1(x_1)^\top Q_2(x_2)$ and $X^B(x_3,x_4) = Q_3(x_3)Q_4(x_4)$ respectively. The multiplication of these vectors X^A, X^B again yields a single element indexed by x_1, x_2, x_3, x_4, which can take any value between $[-1,1]$ by Proposition 1. Then clearly the cores Q form an equivalent definition of $\mathcal{X}$.

We can then apply Corollary 2 and find that the product of indexed orthonormal matrices and orthonormal vectors with full rank can approximate any matrix with bounded norm. Applying this to our OTT format, it immediately follows that with the addition of at most dn constants in $\mathbb{R}$ we can approximate any arbitrary tensor. While this addition would put the format well over the number of parameters in the standard format, this provides sufficient evidence that, in typical learning settings in which our model is already overparameterized, we can still capture the full expressive power of the model class in which an OTT format is inserted.

Remark. It also important to note that the above calculation of dimensionality is the *intrinsic* dimension. The number of actual allocated variables is indeed dn^3 for an exact formulation. It remains open to theoretically analyze the degradation of the approximation as $r < n$.

3.3. Efficient Stiefel Optimization

Here, we describe how to compute an OTT approximation of a tensor $\mathcal{W}$, which can be posed as the following minimization problem.

$$\min_{\{Q_i(x_i)\}_{i=1}^d} E = \sum_{\{x_i\}} \|\mathcal{W}(x_1,\ldots,x_d) - Q_1(x_1)\cdots Q_d(x_d)\|$$
$$\text{s.t.} \quad Q_i(x_i)^\top Q_i(x_i) = I_p \quad \forall i, x_i \tag{8}$$

Notice that this optimization is difficult because of the orthogonality constraint [6, 8]. An efficient way to solve this is by doing the optimization on the product of (compact) Stiefel manifolds: let it be denoted by $\mathcal{P}_S$. We will use the product ℓ_2 metric on this product space. Given $x_1, \cdots x_d$, we perform an optimization on the product of Stiefel manifolds to solve for $\{Q_i(x_i)\}$ for $i \in [1 \dots d]$. We use a Riemannian gradient descent technique on this product of Stiefel manifolds $\mathcal{P}_S$. Given $\{Q_i^t(x_i)\}$ as the solution of the t^{th} step, the $(t+1)^{th}$ solution, $\{Q_i^{t+1}(x_i)\}$, can be computed using

$$\left\{Q_i^{t+1}(x_i)\right\} = \text{Exp}\left(\left\{Q_i^t(x_i)\right\}, \frac{\partial E}{\partial\left\{Q_j^t(x_j)\right\}}\right), \quad (9)$$

where Exp is the Riemannian Exponential map on $\mathcal{P}_S$. On $\mathcal{P}_S$, computation of Riemannian Exponential map is not tractable and needs an optimization, hence we use a Riemannian retraction map as proposed in [14].

Figure 1 summarizes this procedure. For each orthogonal core, the gradient is computed with respect to the Euclidean ambient space and projected to the tangent space at the current iterate. The update is constructed by moving back to the Stiefel with the Riemannian exponential map.

3.4. Square Stiefels/SO(n)

In practice, when learning an OTT operator, we will primarily be setting the rank to be fixed for all cores. The Stiefel manifold, $\text{St}(n, n)$ with $n = p$ is equal to the special orthogonal group $\text{SO}(n)$. The Riemannian Exponential map on $\text{SO}(n)$ is the matrix exponential, computationally intensive to both compute and backpropagate through. Hence, we use the Cayley map from $\mathfrak{SO}(n)$ to $\text{SO}(n)$, given by $A \mapsto (I - A)(I + A)^{-1}$, where $\mathfrak{SO}(n)$ (the space of $n \times n$ skew-symmetric matrices) is the tangent space of $\text{SO}(n)$ at identity. Although the Cayley map requires a matrix inverse, it is much easier to handle using standard tools in modern toolboxes (e.g., TensorFlow, PyTorch).

Observe that the work in [11] used the Cayley map for RNNs, but does not make use of the sparse representation of a skew-symmetric matrix $A \in \mathfrak{SO}(n)$. In contrast, in our formulation we use the Cayley map as a mapping from $\mathbb{R}^{\frac{n(n-1)}{2}}$ to $\text{SO}(n)$. This enables a strict reduction in the number of trainable/learnable variables in a network, and provides a direct path through which gradients can be computed and backpropagated. Algorithm 1 describes the procedure for constructing an OTT-core. The Euclidean variable vector w is mapped directly to the upper triangular part, defined as $\text{triu}(\cdot)$, of a new matrix R, and by subtracting its transpose, we arrive at a skew symmetric matrix A. The Cayley map, as described above, maps to our Orthogonal OTT Core.

Remark. Note that the Cayley map is not a bijective mapping between $\mathfrak{SO}(n)$ and $\text{SO}(n)$ as the range is not the entire $\text{SO}(n)$. This is because the Cayley map cannot generate matrices with negative eigenvalue(s). Empirically, we do not find this to be an issue when learning the OTT representation directly.

Algorithm 1 Constructing an OTT Variable

```
function OTT-VARIABLE($d, n_{in}, n_{out}, r$)
    $OTT \leftarrow \emptyset$
    for $i \in 1, \ldots, d$ do
        for $j, k \in 1, \ldots, n_{in}[i], 1, \ldots, n_{out}[i]$ do
            $OTT$.append(OTT-Core($r$))
        end for
    end for
    return $OTT$
end function
function OTT-CORE($r$)
    $w \leftarrow \mathbb{R}^{r(r-1)/2}$
    $R \leftarrow \text{triu}(w)$
    $A \leftarrow R - R^\top$
    $Q \leftarrow (I - A)(I + A)^{-1}$
    return $Q$
end function
```

With this efficient approximation in hand, we are able to directly apply our OTT formulation to architectures for a variety of applications.

4. Evaluating performance on Simulations, Moving MNIST and Video data

First, we evaluate how well our OTT formulation performs relative to existing methods, on synthetic datasets as well as other popular datasets used for sequential deep models. A Nvidia Titan Xp GPU was used.

(A) OTT vs Riemannian SGD on synthetic data. To empirically verify the claims in Section 3 and to evaluate the value of our OTT construction over the existing Riemannian SGD framework, we simulate a simple least squares problem with the goal of learning a tensorized weight matrix,

$$\min_{W_{TT}} \sum_{i=1}^{n} ||y_i - W_{TT} x_i||^{\cdot 2}$$

Here we use the naïve but exact OTT construction, using the optimization scheme in Section 3.3. A weight matrix W is initialized to a random matrix with size 784×625, and samples are drawn from $y = Wx$. The matrix is reshaped as a tensor with modes $[4, 7, 4, 7] \times [5, 5, 5, 5]$.

Results. Figure 2 shows the convergence rates of both methods with fixed learning rates for various TT-ranks. **(a) Quality and speed.** For this toy problem, not only is the OTT construction able to find a good solution, it is able to find it significantly faster than Riemannian SGD. **(b) Update steps.** Additionally, we note that the time per iteration is significantly shorter for the OTT construction. OTT allows for each manifold update step to be performed on a low

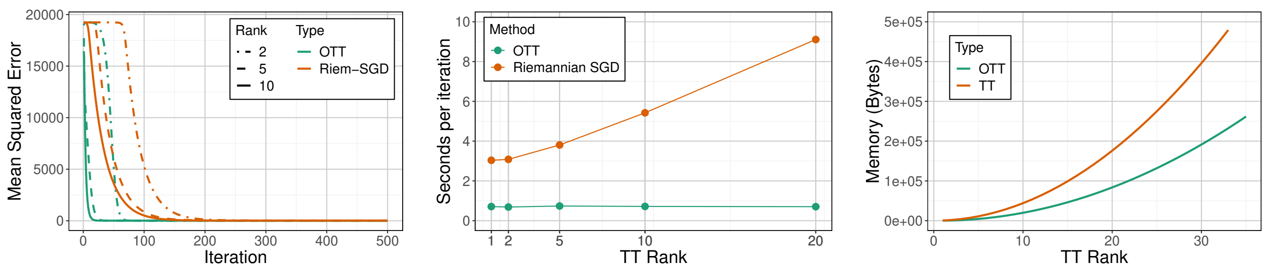


Figure 2: (left) Mean squared error for different TT-ranks, using both the Riemannian formulation (3) and the approximate Stiefel formulation (4). (center) Effect of TT-rank on per iteration runtime of both methods. OTT is significantly faster (10x) than the Riemannian formulation. (right) Memory Dependence of both TT and OTT constructions as a function of rank. The OTT formulation allows for models roughly double the size of TT.

dimensional Stiefel, and so retraction and projection is *fast*. The Riemannian method requires left orthogonalization and QR decompositions of larger matrices, leading to a slower, TT-rank dependent runtime, shown in Figure 2. **(c) Memory footprint.** Finally, we see in Figure 2 that the memory consumption of OTT is quite modest compared to TT (which already offers significant memory savings over alternative existing schemes). This may be a beneficial feature when running a large sequential model on less expensive GPUs. Given these results, we use a basic SGD update for TT in subsequent experiments.

(B) Moving MNIST. The moving MNIST dataset [24] consists of handwritten digits moving within a specified larger image. We first demonstrate that for simple sequences, reconstruction under a complete tensor train framework is possible, and representing fully connected layers with an OTT layer reduces the number of parameters **without** image degradation. Here, we use a vanilla RNN, with a state size of 4096 and TT-Rank 64.

Results. Figure 3 shows the ground truth and reconstruction results for images with size 256×256, where each sequence is of length 8, and the direction and orientation of the digit is random. **(a) Reconstruction accuracy and model size.** The entire recurrent network is compressed with OTT layers for input-to-hidden, hidden-to-hidden, and hidden-to-output maps. With a large state size of 4096, we are able to nicely capture and rebuild the entire sequence with a significantly smaller model size. **(b) Scaling to larger images.** This effective compression also allows us to scale up – to significantly larger images of size 1024×1024, *with no loss in reconstruction quality*, without the need for more sophisticated convolutional architectures.

(C) Hollywood2. We find that these results extend nicely to LSTMs/GRUs and for classification tasks as well. The Hollywood2 dataset [19] consists of video clips from 69 movies labeled with 12 different actions from "answering the phone" to "driving a car" (Figure 4). Following the preprocessing steps of [26], we feed resized clips of size $234 \times 100 \times 3 \times T$ to our model, where the length of a sequence (number of frames) T ranges from 29 to 1496. We tensorize the input as $10 \times 18 \times 13 \times 30$ for all input sequences (padded to 1496) and the hidden states as $4 \times 4 \times 4 \times 4$, with TT and OTT ranks set as 4.

Results. Tensor trains here allow us to completely operate on the *entire video sequence.* **(a) Parameter size.** The number of parameters in our model is a few thousands (1864 for OTT, 3104 for TT) compared to millions needed for a standard fully connected model. **(b) Accuracy comparison.** Using Mean Average Precision (MAP) as a measure of accuracy for this multi-label problem, we find that using an OTT-LSTM or OTT-GRU in place of a TT-LSTM or TT-GRU leads to *no significant difference in MAP.*

5. Identifying Differential Progression in AD

Motivation. The Alzheimer's Disease Neuroimaging Initiative (ADNI, `adni.loni.usc.edu`) provides a comprehensive dataset targeted towards understanding AD. The goals of the initiative include measuring the development of the disease as a function of different imaging modalities, other biological markers, and clinical and neuropsychological assessments. Deep learning methods traditionally applied to this corpus require imaging data to be heavily preprocessed into summary measures, such as *regions of interest*. In other cases, based on the needs of the application (e.g., segmentation), the approach may operate with 3D image patches instead of the entire image. The size of the images, especially when considered longitudinally, can be impractical for modern deep learning frameworks unless some novel implementation tricks are utilized.

Data. Our dataset consists of 522 subjects with Magnetic Resonance Imaging (MRI) scans collected over three years. For each individual, an MRI was collected annually, along with a battery of neuropsychological evaluations.

Pre-processing. Full head MRIs were processed using SPM12 [1]. Each image was segmented/registered using the MNI152 template. Gray matter probabilities were computed, and these gray matter density (GMD) images were used as input to our models. The processed image size was $121 \times 145 \times 121$ (voxel size $1.5mm^3$), with **3 images per subject**.

Model. At this scale, we use convolutional input and de-

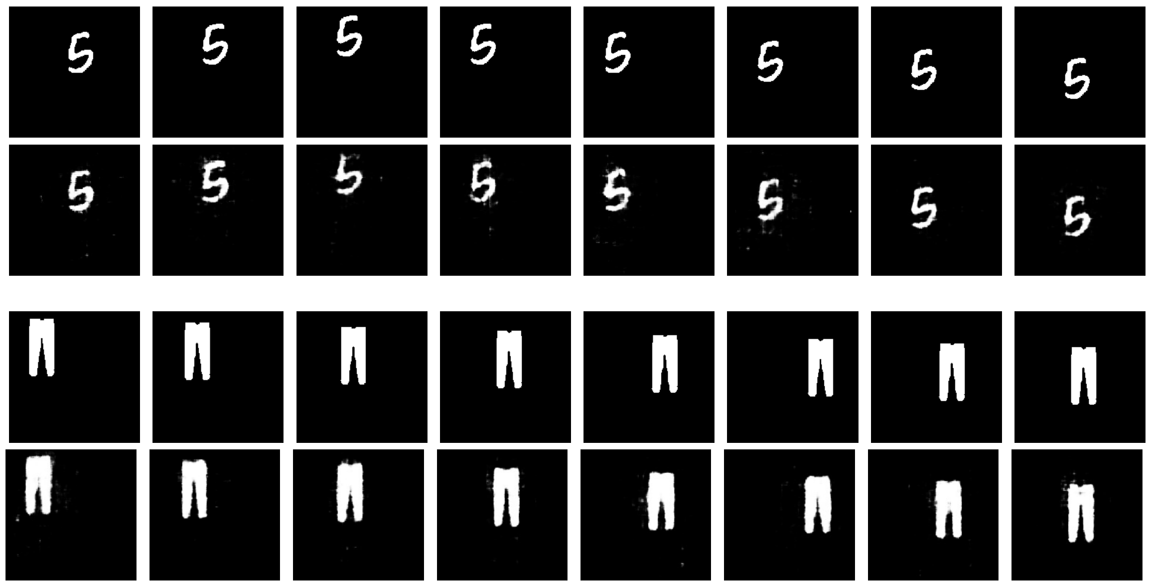


Figure 3: Sample ground truth (top) and reconstruction (bottom) of Moving MNIST digit and fashion sequences of size 256×256. We see good consistency between each upper/lower rows for both datasets.

convolutional output layers to incorporate local information with respect to reconstruction and prediction. The architecture consists of a straightforward 3-state RNN with input-hidden, hidden-hidden, and hidden-output layers replaced with TT and OTT layers. Input volumes are passed through a 3D convolutional input network, with max-pooling layers and ReLUs. Hidden states are passed through an output convolutional network consisting of max-unpooling layers using indices saved from the input CNN. Strides were fixed at 1 with a kernel size of $3 \times 3 \times 3$, with successive convolutions decreasing (increasing) the number of channels by 2. Max pooling was applied uniformly to all 3 input channels with a stride of 2. Adam optimization [16] was used for all ADNI experiments, with learning rate $1e^{-3}$, and decay rate 0.9 and 0.999 for the first and second moments. TT and OTT layers were fixed with a rank of 64. Batch sizes were fixed at 4.

5.1. Modeling gray matter progression in AD

Our first goal is to predict the next MR image given the previously seen ones. Importantly, standard RNN constructions cannot easily handle inputs of this size. On a single NVIDIA Titan Xp, the images must be **downsampled** by over $20\times$ to allow for a batch size of 4 in a standard LSTM model with a hidden state size of 2048 (4.3 billion parameters for a full sized input map).

Results. Fig 5 show the results for a held-out subject in the study using OTT-RNN on a single representative 2D slice, with their predicted third timepoint image. While higher levels of compression (lower OTT ranks) lead to "blocky" reconstructions, our model is still able to identify boundaries of edges between low and high probabililty voxels.

5.2. Cognition from gray matter sequence

Based on the results from the above experiment, one may ask if, in fact, a good model of progression is being learned, or if only the "average" of all participants is being predicted by the model.

(A) Predicting Cognition from 3D image sequences. To answer this question, we can directly try to predict summary cognition measures which are used in practice. Diagnoses themselves can often be based on partial information available to medical experts at that time. Indeed, a small number of individuals in the ADNI cohort have been diagnosed with AD or mild cognitive impairment and have *regressed* to a cognitively healthy diagnosis at their next visit. In these situations, categorical diagnoses can be seen as a noisy summary measure of decline. We predict a real-valued measure collected at each timepoint. The Rey Auditory Verbal Learning Test (RAVLT) evaluates a large variety of cognitive functions, including short and long term memory,

Figure 4: Sample sequences from the Hollywood2 dataset. Labels are (Top) Answer Phone, (Middle) Drive Car, and (Bottom) Get Out Car.

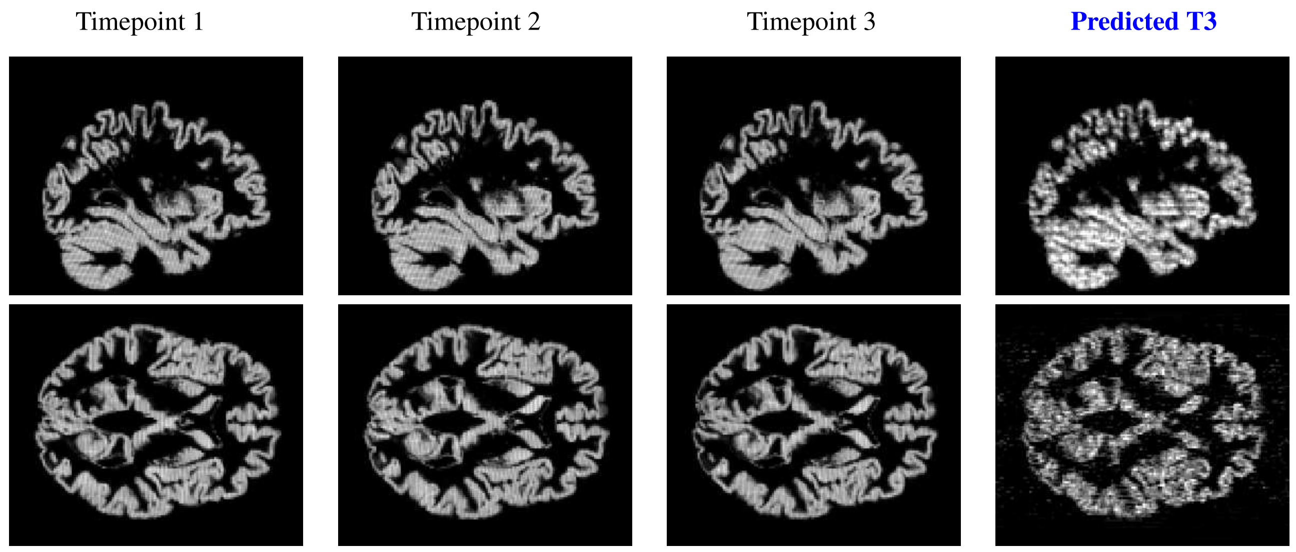


Figure 5: Ground truth progression and prediction of gray matter probabilities in an individual from our validation set. From the left, the first three images are the ground truth images at visits 1, 2, and 3, followed by our prediction at visit 3.

cognitive function, and learning ability [23], and has been identified as a strong indicator for developing AD pathology. We train both TT and OTT models with dropout for 200 epochs.

Results. Figure 6 (left) shows the results of this analysis. Here, the advantage of the OTT construction is clear, we are able to converge significantly faster compared to the TT construction, with half as many parameters.

(B) Quantifying Model Uncertainty. Broad application of deep learning models in neuroimaging remains limited, namely due to sensitivity of black box models to mild perturbations in input data or model parameters, leading to unreliable predictions. MC Dropout [9], approximates model (epistemic) uncertainty by using dropout *at prediction time*. Simulating an ensemble of networks with different structures can yield direct estimates of uncertainty. Obtaining good measures of this uncertainty requires sampling all parameters a significant number of times: large networks may require many samples before a reasonable uncertainty estimate. Using tensor train constructions allows us to feasibly compute an estimate of uncertainty over all outputs, and with OTT we can further reduce this required sampling rate.

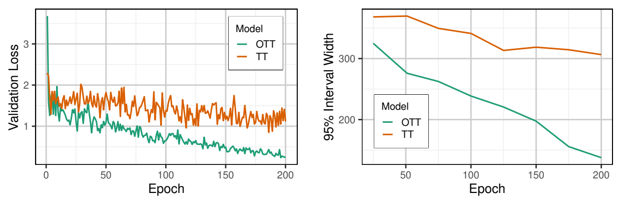


Figure 6: Validation losses for reconstruction (left) and confidence interval widths (right) for uncertainty estimation of RAVLT prediction (lower is better).

Results. Figure 6 (right) shows 95% interval widths computed over 100 MC Dropout instantiations, averaged over individuals in the validation set. The advantage of our compressed orthogonal construction is clear, resulting in smaller confidence intervals compared to the standard TT decomposition.

6. Conclusion

Taking advantage of the structure inherent in tensor train decompositions, we propose and analyze the Orthogonal Tensor Train Decomposition, yielding direct benefits in both parameter efficiency and computation time. This is an important step in instantiating recurrent or sequential models for a set of longitudinal 3D brain images, either in the context of generating new images in the sequence or for classification. Using a mapping from Euclidean space, we construct a neural network variable that can efficiently be learned through existing deep learning optimization frameworks. Our results yield promising developments in applying deep learning methods for analyzing sequential 3D medical imaging data, and we show that our method can perform favorably in reconstruction and prediction tasks with such image volumes. While a focus here was brain imaging, we anticipate numerous applications in other medical imaging settings. Code is available at https://github.com/ronakrm/OTT.

Acknowledgements

This work was supported by grants NSF CAREER award RI 1252725, UW CPCP (U54AI117924), R01AG059312, R01EB022883, RF1AG062336, and a NIH predoctoral fellowship to RM via T32 LM012413. We thank Jen Birstler for their help with plots.

References

[1] John Ashburner, Gareth Barnes, C Chen, Jean Daunizeau, Guillaume Flandin, Karl Friston, Stefan Kiebel, James Kilner, Vladimir Litvak, Rosalyn Moran, et al. Spm12 manual. *Wellcome Trust Centre for Neuroimaging, London, UK*, 2014.

[2] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *SIGKDD ICKDDM*, 2001.

[3] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3), 1970.

[4] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[5] Nadav Cohen and Amnon Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*, pages 955–963, 2016.

[6] Maxwell D Collins, Ji Liu, Jia Xu, Lopamudra Mukherjee, and Vikas Singh. Spectral clustering with a convex regularizer on millions of images. In *European Conference on Computer Vision*, pages 282–298. Springer, 2014.

[7] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015.

[8] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016.

[10] Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[11] Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. *arXiv preprint arXiv:1707.09520*, 2017.

[12] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120(4), 2012.

[13] Seong Jae Hwang, Sathya N. Ravi, Zirui Tao, Hyunwoo Kim, Maxwell D. Collins, and Vikas Singh. Tensorize, factorize and regularize: Robust visual relationship learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] T. Kaneko, S. Fiori, and T. Tanaka. Empirical arithmetic averaging over the compact stiefel manifold. *IEEE Transactions on Signal Processing*, 61(4):883–894, 2013.

[15] Valentin Khrulkov, Oleksii Hrinchuk, and Ivan Oseledets. Generalized tensor models for recurrent neural networks. In *International Conference on Learning Representations*, 2019.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Stefan Klus, Patrick Gelß, Sebastian Peitz, and Christof Schütte. Tensor-based dynamic mode decomposition. *Nonlinearity*, 31(7):3359, 2018.

[18] Christian Lubich, Ivan V Oseledets, and Bart Vandereycken. Time integration of tensor trains. *SIAM Journal on Numerical Analysis*, 53(2):917–941, 2015.

[19] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[20] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *NIPS*, pages 442–450, 2015.

[21] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. Exponential machines. *ICLR Workshop Track*, 2017.

[22] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[23] Michael Schmidt et al. *Rey auditory verbal learning test: A handbook*. Western Psychological Services Los Angeles, CA, 1996.

[24] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[25] Yunyang Xiong, Hyunwoo J Kim, and Varsha Hedau. Antnets: Mobile convolutional neural networks for resource efficient image classification. *arXiv preprint arXiv:1904.03775*, 2019.

[26] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In *ICML*, 2017.

[27] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. In *NIPS*, pages 1569–1576, 2005.

[28] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[29] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, pages 7370–7379, 2017.

[30] Qingchen Zhang, Laurence T Yang, Xingang Liu, Zhikui Chen, and Peng Li. A tucker deep computation model for mobile multimedia feature learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 13(3s):39, 2017.

[31] Xingjian Zhen, Rudrasis Chakraborty, Nicholas Vogt, Barbara B. Bendlin, and Vikas Singh. Dilated convolutional neural networks for sequential manifold-valued data. In *International Conference on Computer Vision*, 2019.