Training a Tucker Model With Shared Factors: a Riemannian Optimization Approach

I. Peshekhonov
HSE University

A. Arzhantsev
HSE University

M. Rakhuba
HSE University

Abstract

Factorization of a matrix into a product of two rectangular factors, is a classic tool in various machine learning applications. Tensor factorizations generalize this concept to more than two dimensions. In applications, where some of the tensor dimensions have the same size or encode the same objects (e.g., knowledge graphs with entity-relation-entity 3D tensors), it can also be beneficial for the respective factors to be shared. In this paper, we consider a well-known Tucker tensor factorization and study its properties under the shared factor constraint. We call it a shared-factor Tucker factorization (SF-Tucker). Since sharing factors breaks polylinearity of classical tensor factorizations, common optimization schemes such as alternating least squares become inapplicable. Nevertheless, as we show in this paper, a set of fixed-rank SF-Tucker tensors preserves a Riemannian manifold structure. Therefore, we develop efficient algorithms for the main ingredients of Riemannian optimization on the SF-Tucker manifold and implement a Riemannian optimization method with momentum. We showcase the benefits of our approach on several machine learning tasks including knowledge graph completion and compression of neural networks.

1 INTRODUCTION

Tensor factorizations generalize low-rank matrix factorizations to multivariate arrays, called tensors. By analogy with the matrix case, they decompose a tensor into a set of lower-dimensional tensors with presumably much fewer parameters. This allows for efficiently compressing high-dimensional data, as well as extracting useful latent factors in a wide variety of applications. Among tensor factorizations, a classic Tucker decomposition (Tucker, 1963) plays a special role. Its ease of formulation, availability of reliable SVD-based procedures for constructing approximations and interpretability of its factors make it a popular choice in machine learning. For example, it has been used in recommender systems (Frolov and Oseledets, 2017), knowledge graphs (Balažević et al., 2019), and for compressing neural networks (Kossaifi et al., 2020).

In this paper, we generalize the Tucker decomposition by allowing any given number of its factors to be equal to each other, or, in other words, to be shared. We refer to such a decomposition as a shared-factor Tucker (SF-Tucker). Sharing factors appears to be useful in knowledge graphs that store facts in triple form of entity-relation-entity (Balažević et al., 2019) and, therefore, can be naturally represented as three-dimensional binary tensors with two modes of equal size. Sharing weights has also been utilized in the end-to-end training of neural networks with tensor-decomposed weight matrices (Obukhov et al., 2020). Besides applications for knowledge graphs and neural networks, we also utilize SF-Tucker decomposition for functions on a grid.

To approximate a tensor in the SF-Tucker format, we propose to use Riemannian optimization algorithms. Specifically, we prove that a set of SF-Tucker tensors with fixed factor sizes forms a smooth manifold and utilize a Riemannian optimization method with momentum. To run this method, we derive formulas for essential steps of optimization on smooth manifolds: projection to a tangent plane and retraction. A special care is taken to turn these formulas into efficient algorithms by using numerical linear algebra techniques and automatic differentiation. The code is available at https://github.com/johanDDC/tucker_riemopt.

Our contributions are:

- We propose a new factorization, called SF-Tucker, and study its properties (Section 2). We also

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

introduce and theoretically justify an algorithm for approximating tensors in the SF-Tucker format.

- We prove that a set of fixed-rank SF-Tucker tensors forms a smooth embedded submanifold and derive algorithms for essential tools of Riemannian optimization: projection to a tangent plane and retraction to the SF-Tucker manifold (Section 4,5).
- We numerically investigate the benefits of our approach on two examples: knowledge-base graphs and compression of neural networks (Section 6).

2 RELATED WORK

Tensor Factorizations And Optimization. Tucker (1963) proposed a Tucker decomposition, which by contrast to a CP decomposition (Hitchcock, 1927) can be computed using reliable SVD-based procedures, but is prone to the curse of dimensionality. A tensor-train (TT) (Oseledets, 2011) and the hieararchical Tucker (HT) decompositions (Hackbusch and Kühn, 2009) overcame this issue. Nevertheless, Tucker decomposition is often a method of choice for lower dimensional tensors. For example, in 3D it coincides with the HT decomposition and contains fewer parameters than TT.

A popular method for computing a Tucker decomposition is the higher-order SVD (HOSVD) De Lathauwer et al. (2000). It produces a quasioptimal approximation of a tensor by computing SVDs of tensor unfolding matrices. To approximate optimal solution, one may use iterative alternating-based approaches, such as higher order orthogonal iteration (HOOI) or alternating least squares methods (Kolda and Bader, 2009). We also mention the RESCAL model (Nickel et al., 2011) (relaxed version of DEDICOM (Harshman, 1978)) that imposes equal factors in a Tucker-2 decomposition using a heuristic approach. Nevertheless, these methods are only straightforwardly applicable for quadratic functionals to be minimized. By contrast, machine learning tasks are often formulated using more complicated loss functions and rely on automatic differentiation tools. Therefore, Riemannian optimization methods that combine gradient based optimization and take into account tensor structure are well-suited for these tasks. See, for example, (Fonarev et al., 2017; Novikov et al., 2016). We also mention works on matrix and tensor completion using Riemannian optimization (Vandereycken, 2013; Kressner et al., 2014; Kasai and Mishra, 2016).

Knowledge Graphs. One of the application domains that we consider is knowledge graphs. Tensor-based models represent the knowledge graph as a three-dimensional binary tensor and apply various tensor decompositions in order to learn embeddings for entities and relations. The model based on CP decomposition was firstly introduced by (Trouillon et al., 2017). The model **SimplE** (Kazemi and Poole, 2018) improved the quality by learning dependent embeddings for subjects and objects. **CP-N3** (Lacroix et al., 2018) improved it further by applying regularizer based on Nuclear p-norm. Another model based on CP decomposition, **MEKER** (Chekalina et al., 2022) utilizes classical CP-ALS algorithm for specifically constructed loss minimization. **TuckER** (Balažević et al., 2019) model utilizes Tucker tensor decomposition to represent a knowledge graph as a set of factor matrices with entities and relations embeddings and a core tensor capturing the interactions between them. Although, TuckER achieves state-of-the-art results on the link prediction task, it also uses batch normalizations, which make it is not fully equivalent to a Tucker decomposition (see Section 6.2).

There exists a number of approaches based on deep learning, including **ConvE** (Dettmers et al., 2018) which utilizes convolutions, **GAT** (Nathani et al., 2019) which is a graph neural network enhanced by an attention mechanism, and the recently proposed **Relphormer** (Bi et al., 2023) model, which is a transformer adapted for knowledge graphs. Although, these models achieve high quality in link prediction task, they also have deep learning-specific disadvantages. They need more time and memory resources, attention based models usually need more data and pre-training. In addition, Sun et al. (2020) has pointed out evaluation challenges with deep learning-based models.

Tensor Factorizations In Neural Networks. Many layers of neural networks can be represented as operations on tensors. Decomposing these tensors reduce the number of parameters and possibly compute times, while not losing much in accuracy. For example, tensor decompositions can be used to replace fully connected layers (Kossaifi et al., 2020; Kolbeinsson et al., 2019), convolutional layers (Hayashi et al., 2019; Lebedev et al., 2015; Wang et al., 2018), RNN layers (Yang et al., 2017). Even if a layer does not have an obvious tensor structure, it can be reshaped into a tensor. This approach is called tensorization and was used in (Novikov et al., 2015; Garipov et al., 2016) and in (Obukhov et al., 2020) with additional weight sharing.

3 SF-TUCKER DECOMPOSITION

3.1 Tucker Decomposition And Notation

A d-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ is said to represented in a Tucker format if it can be written as:

$$\mathcal{X}_{i_1,\ldots,i_d} = \sum_{j_1,\ldots,j_d=1}^{r_1,\ldots,r_d} \mathcal{G}_{j_1,\ldots,j_d} V^{(1)}_{i_1,j_1} \ldots V^{(d)}_{i_d,j_d}, \quad (1)$$

where $\mathcal{G} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ is called the Tucker core and $V^{(k)} \in \mathbb{R}^{r_k \times r_k}$ are called factors and the set $(r_1, \ldots, r_d)$ of minimal possible r_k is called the *Tucker rank*. Note that when $d = 2$ it reduces to a skeleton matrix factorization $\mathcal{X} = V^{(1)} \mathcal{G} V^{(2)\intercal}$. To simplify notation we will write (1) as

$$\mathcal{X} = \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d)} \right]\!\right],$$

where possible. Many properties of the Tucker decomposition are associated with the so-called *unfolding matrices* $\mathcal{X}_{(k)} \in \mathbb{R}^{n_k \times (n_1 \ldots n_d)/n_k}$ with the entries $\left(\mathcal{X}_{(k)}\right)_{i_k,j} = \mathcal{X}_{i_1 \ldots i_d}$, $j = 1 + \sum_{\substack{\alpha=1 \\ \alpha \neq k}}^{d} (i_\alpha - 1) n_\alpha$. For example, $\mathcal{X}_{(k)}$ can give us Tucker rank components: $r_k = \operatorname{rank}(\mathcal{X}_{(k)})$ (Kolda and Bader, 2009).

Notation. In what follows, we also use the following notation. We use Frobenius norm of $\mathcal{X}$: $\|\mathcal{X}\|_F^2 = \sum_{i_1,\ldots,i_d} \mathcal{X}^2_{i_1 \ldots i_d}$ and Frobenius scalar product: $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1,\ldots,i_d} \mathcal{X}_{i_1,\ldots,i_d} \mathcal{Y}_{i_1,\ldots,i_d}$. Kronecker product is denoted by $\otimes$. By

$$A = \left(A_1 \mid A_2 \mid \ldots \mid A_D \right) \in \mathbb{R}^{m \times (m_1 + \ldots + m_D)}$$

we denote a block-row matrix that consists of submatrices $A_i \in \mathbb{R}^{m \times m_i}$, $i = 1, \ldots, D$. A^+ stands for a Moore-Penrose pseudoinverse of A.

3.2 SF-Tucker Decomposition

In this section, we describe the proposed Tucker decomposition with shared factors. Due to factor sharing, we assume without loss of generality that $d = d_t + d_s$ are such that $n_1, \ldots, n_{d_t}$ have arbitrary sizes, while $n_{d_t+1} = \ldots = n_d = n$. The SF-Tucker decomposition then reads as

$$\mathcal{X} = \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U \right]\!\right], \quad (2)$$

for some core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times \ldots \times r_{d_t} \times r_s \times \ldots \times r_s}$, factor matrices $V^{(i)} \in \mathbb{R}^{n_i \times r_i}$ for $i = 1, \ldots, d_t$ and the *shared factor* matrix $U \in \mathbb{R}^{n \times r_s}$. Our decomposition differs from the vanilla Tucker decomposition in the last d_s factors: in SF-Tucker, we force them to be equal to each other. Note, however, that both $\mathcal{X}$ and $\mathcal{G}$ do not require to have any symmetries in the last d_s modes.

The set of minimal[1] r_i such that $\mathcal{X}$ has representation (2) is called *shared-factor Tucker rank* (SFT rank):

$$\operatorname{rank}^{d_s}_{\mathtt{sft}}(\mathcal{X}) = (r_1, \ldots, r_{d_t}, r_s).$$

[1]In Appx. A.2 we show that for each i minimal rank values r_i can be attained simultaneously.

Theorem 3.1. *The SFT rank of $\mathcal{X}$ is equal to $\boldsymbol{r} = (r_1, \ldots, r_{d_t}, r_s)$, where*

$$r_i = \operatorname{rank}(\mathcal{X}_{(i)}), \quad i = 1, \ldots, d_t,$$

and

$$r_s = \operatorname{rank}\left(\mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \ldots \mid \mathcal{X}_{(d)} \right).$$

Proof. See Appx. A.2. □

Note that r_i, $i = 1, \ldots, d_t$ coincide with the classical multilinear rank in Tucker decomposition, while r_s is a new object, which we refer to as a *shared rank*. The following proposition suggests that the representation (2) does not lead to additional restrictions compared with the Tucker model.

Proposition 3.1. *Let $\mathcal{X}$ admit a Tucker decomposition with the Tucker rank $(r_1, \ldots, r_d)$, then it also admits the SF-Tucker decomposition with the rank $(r_1, \ldots, r_{d_t}, r_s)$ satisfying $r_s \leqslant r_{d_t+1} + \cdots + r_d$.*

Proof. The proof immediately follows from Theorem 3.1 and the fact that $\operatorname{rank}\left(\mathcal{X}_{(d_t+1)} | \ldots | \mathcal{X}_{(d)}\right) \leqslant \operatorname{rank}(\mathcal{X}_{(d_t+1)}) + \cdots + \operatorname{rank}(\mathcal{X}_{(d)})$. □

Thanks to the rank bound from Prop. 3.1 the number of parameters in factors of the SF-Tucker does not exceed that of the classical Tucker decomposition. As we will see in Section 6 with numerical experiments, in practice SF-Tucker often wins in terms of the total number of parameters for large enough values of n.

Remark 3.1. *One can explicitly construct SF-Tucker from the Tucker decomposition by merging respective factor matrices and padding the core tensor with zeroes. For example, for $d = 2$:*

$$\mathcal{X} = V^{(1)} \mathcal{G} V^{(2)\intercal} = [V^{(1)} | V^{(2)}] \begin{bmatrix} 0 & \mathcal{G} \\ 0 & 0 \end{bmatrix} [V^{(1)} | V^{(2)}]^\intercal.$$

3.3 SF-HOSVD Approximation Algorithm

Theorem 3.1 suggests that the shared rank is connected with concatenation of the respective unfolding matrices. This motivates us to consider left singular values of this matrix to construct a low-rank approximation of $\mathcal{X}$ by analogy with the classical higher-order SVD (HOSVD) algorithm (De Lathauwer et al., 2000). We call this modification SF-HOSVD. The following theorem holds.

Theorem 3.2. *For $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_{d_t} \times n \times \cdots \times n}$, let $V^{(k)} \in \mathbb{R}^{n_k \times r_k}$ be the matrix of first r_k left singular vectors of $\mathcal{X}_{(k)}$, $k = 1, \ldots, d_t$ and $U \in \mathbb{R}^{n \times r_s}$ be the matrix of first r_s left singular vectors of the concatenation of unfolding matrices $\left(\mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \ldots \mid \mathcal{X}_{(d)} \right)$. Let* *also* $\mathcal{G} = [\![\mathcal{X}; V^{(1)^T}, V^{(2)^T}, \dots, V^{(d_t)^T}, U^T, \dots, U^T]\!]$. *Then*

$$P_{\mathbf{r}}^{\textit{sf-hosvd}}(\mathcal{X}) = [\![\mathcal{G}; V^{(1)}, \dots, V^{(d_t)}, U, \dots, U]\!]$$

is a quasioptimal approximation of $\mathcal{X}$*:*

$$\left\|\mathcal{X} - P_{\mathbf{r}}^{\textit{sf-hosvd}}(\mathcal{X})\right\|_F \leqslant \sqrt{d} \inf_{\operatorname{rank}_{sft}^{d_s}(\mathcal{Y}) \preccurlyeq \boldsymbol{r}} \|\mathcal{X} - \mathcal{Y}\|_F,$$

where $\preccurlyeq$ *is a componentwise comparison of two tuples.*

Proof. See Appx. A.3. □

The SF-HOSVD gives us a constructive way to approximate a given tensor in the SF-Tucker format. On the other hand, it provides quasioptimal approximation only in $\|\cdot\|_F$. Nevertheless, as we will see later, it can be used as a retraction operation in Riemannian optimization. To make it efficient as a retraction, we need to be able to implicitly apply it to SF-Tucker tensors without forming all of their elements. The Algorithm 1 summarizes this rank truncation operation with $\mathcal{O}(dNr^2 + d_s r^{d+1})$ complexity, where r is maximum input rank and $N = \max n_i$. We assume that factors of input tensors have orthonormal columns, which can always be done using QR decompositions.

Algorithm 1 Rank truncation using SF-HOSVD

```
Require: $\mathcal{X} = [\![\bar{\mathcal{G}}; \bar{V}^{(1)}, \dots, \bar{V}^{(d_t)}, \bar{U} \dots \bar{U}]\!]$.
Ensure: $P^{\texttt{sf-hosvd}}_{(r_1,\dots,r_{d_t},r_s)}(\mathcal{X})$
  for $k = 1, 2, \dots, d_t$ do
    $Y\Sigma W^{\intercal} = \texttt{thinSVD}(\bar{\mathcal{G}}_{(k)})$          $\triangleright \mathcal{O}(r^{d+1})$
    $V^{(k)} = \bar{V}^{(k)} Y[:, :r_k]$          $\triangleright \mathcal{O}(Nr^2)$
  $C = \left(\bar{\mathcal{G}}_{(d_t+1)} \mid \dots \mid \bar{\mathcal{G}}_{(d)}\right)$
  $Y\Sigma W^T = \texttt{thinSVD}(C)$          $\triangleright \mathcal{O}(d_s r^{d+1})$
  $U = \bar{U} Y[:, :r_s]$          $\triangleright \mathcal{O}(Nr^2)$
  $\mathcal{G} = \left[\!\!\left[\bar{\mathcal{G}}; V^{(1)^T}\bar{V}^{(1)}, \dots, U^T\bar{U}\right]\!\!\right]$     $\triangleright \mathcal{O}(dNr^2 + r^{d+1})$
  return $P^{\texttt{sf-hosvd}}_{(r_1,\dots,r_{d_t},r_s)}(\mathcal{X}) \equiv [\![\mathcal{G}; V^{(1)}, \dots, U]\!]$
```

4 FIXED RANK TUCKER MANIFOLD WITH SHARED FACTORS

Before we move to algorithms, we need to formulate key ingredients of Riemannian optimization. Let us introduce the set of $d = (d_t + d_s)$-dimensional tensors of a fixed SFT rank $\mathbf{r} = (r_1, \dots, r_{d_t}, r_s)$.:

$$\mathcal{M}_{\mathbf{r}}^s \overset{\text{def}}{=} \{\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_{d_t} \times n \times \dots \times n} \mid \operatorname{rank}_{\texttt{sft}}^{d_s} \mathcal{X} = \mathbf{r}\}.$$

Theorem 4.1. $\mathcal{M}_{\mathbf{r}}^s$ *forms a smooth embedded submanifold of* $\mathbb{R}^{n_1 \times \dots \times n_{d_t} \times n \times \dots \times n}$ *of dimension:*

$$\dim \mathcal{M}_{\mathbf{r}}^s = \sum_{i=1}^{d_t} r_i(n_i - r_i) + r_s(n_s - r_s) + r_s^{d_s} \prod_{i=1}^{d_t} r_i.$$

Proof. The proof can be found in Appx B.1. □

Now, as we know that $\mathcal{M}_{\mathbf{r}}^s$ is a smooth manifold, we may investigate the structure of its tangent spaces.

4.1 Tangent Spaces.

A vector ξ is called a tangent vector of a manifold $\mathcal{M}$ at a point $x \in \mathcal{M}$ if there exists a smooth curve $c\colon [0,1] \to \mathcal{M}$, such that $c(0) = x$ and $c'(0) = \xi$. The set of all tangent vectors at a point is called the tangent space at that point and is denoted as $T_x\mathcal{M}$. Tangent spaces play an important role in Riemannian optimization as they allow for locally replacing a manifold with a linear space. A tangent space $\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s$ of our manifold $\mathcal{M}_{\mathbf{r}}^s$ at $\mathcal{X} = [\![\mathcal{G}; V^{(1)}, \dots, V^{(d_t)}, U, \dots, U]\!]$ is given by:

$$\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s = \Big\{\xi = \left[\!\!\left[\dot{\mathcal{G}}; V^{(1)}, \dots, U\right]\!\!\right] + \\ + \left[\!\!\left[\mathcal{G}; \dot{V}^{(1)}, \dots, U\right]\!\!\right] + \dots + \left[\!\!\left[\mathcal{G}; V^{(1)}, \dots, \dot{U}\right]\!\!\right]\Big\}, \tag{3}$$

parametrized by arbitrary matrices $\dot{V}_i \in \mathbb{R}^{n_i \times r_i}$ and $\dot{U} \in \mathbb{R}^{n \times r_s}$. Similarly to the matrix case (Novikov et al., 2022), we impose on them gauge conditions: $\dot{V}_i^T V_i = 0$ and $\dot{U}^T U = 0$ to remove redundancies (see Appx B.2). One can straightforwardly verify, that every tangent vector $\xi \in \mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s$ can also be represented in the SF-Tucker form:

$$\xi = \left[\!\!\left[\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right); W^{(1)}, \dots, W^{(d_t)}, Y, \dots, Y\right]\!\!\right] \tag{4}$$

of rank at most $\mathbf{2r}$ with the factors $W^{(i)} = \left(V^{(i)} \mid \dot{V}^{(i)}\right) \in \mathbb{R}^{n_i \times 2r_i}$, $Y = \left(U \mid \dot{U}\right) \in \mathbb{R}^{n \times 2r_s}$ and the core $\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right) \in \mathbb{R}^{2r_1 \times \dots \times 2r_s}$ that is zero except for $\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right)(:r_1, \dots, :r_s) = \dot{\mathcal{G}}$ and for $i = 1, \dots, d$:

$$\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right)(:r_1, \dots, :r_{i-1}, r_i{:}2r_i, :r_{i+1}, \dots, :r_s) = \mathcal{G}.$$

4.2 Projection To The Tangent Tpace

In Riemannian optimization, an important ingredient is the projection to a tangent space $\pi_{\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}} : \mathbb{R}^{n_1 \times \dots \times n} \to \mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s$. In particular, the projection allows us to compute the key ingredient of Riemannian optimization – Riemannian gradient

$$\nabla_{\mathcal{M}_{\mathbf{r}}^s} f = \pi_{\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s} \nabla f.$$

The projection $\pi_{\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s}$ is associated with certain scalar products on $\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s \times \mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^s$ (Absil et al., 2008) called Riemannian metric. In particular, it will be used for computing Riemannian gradients of a certain function. In this paper, we choose a standard

Riemannian metric, which is a restriction of $\langle\cdot,\cdot\rangle_F$ to $\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}} \times \mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}$. The projection of a tensor $\mathcal{Y}$ to $\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}$ at $\mathcal{X} = [\![G; V^{(1)}, \dots, V^{(d_t)}, U, \dots, U]\!]$ has the form

$$\xi = \pi_{\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}}(\mathcal{Y})$$

from (4), where $\dot{V}^{(i)}, \dot{U}$ and $\dot{\mathcal{G}}$ are given by:

$$\dot{\mathcal{G}} = \left[\!\!\left[\mathcal{Y}; {V^{(1)}}^T, \dots, U^T\right]\!\!\right], \quad \dot{V}^{(i)} = P^{\perp}_{V^{(i)}} \mathcal{Y}_{(i)} \mathcal{U}_{\neq i} \mathcal{G}^{+}_{(i)},$$

$$\dot{U} = P^{\perp}_{U^{(i)}} \sum_{i=d_t+1}^{d} \mathcal{Y}_{(i)} \mathcal{U}_{\neq i} \mathcal{G}^T_{(i)} \left(\sum_{i=d_t+1}^{d} \mathcal{G}_{(i)} \mathcal{G}^T_{(i)} \right)^{-1},$$

where $\mathcal{U}_{\neq i} = U \otimes \dots \otimes V^{(i+1)} \otimes I \otimes V^{(i-1)} \otimes \dots \otimes V^{(1)}$ and $P^{\perp}_A = I - AA^T$.

The derivation of these formulas can be found in Appx B.2. Note that the formulas for $\dot{G}$ and $\dot{V}^{(i)}$ coincide with the Tucker case (Steinlechner, 2016, p. 48). However, the $\dot{U}$ part is specific to the SF-Tucker and is not equivalent to the projection on the tangent space of Tucker at a point with equal factors (compare formulas for $\dot{V}^{(i)}$ and $\dot{U}$).

4.3 Retraction.

Let $\mathcal{M}$ be some manifold and $T\mathcal{M} = \{(x, y) \mid x \in \mathcal{M}, y \in T_x\mathcal{M}\}$ its tangent bundle. A smooth mapping $R \colon \mathrm{T}\mathcal{M} \to \mathcal{M}$ such that $R(x, 0) = x$ and $\frac{d}{dt} R(x, t\xi)\big|_{t=0} = \xi$ is called a retraction (Steinlechner, 2016, p. 24). Retractions are an important element of Riemannian optimization algorithms, as they typically allow us to efficiently make a step along the chosen direction, avoiding computationally demanding exponential maps. The next theorem suggests that we can use the SF-HOSVD procedure, summarized in Section 3.3.

Theorem 4.2. *The mapping*

$$R \colon \mathrm{T}\mathcal{M}^s_{\mathbf{r}} \to \mathcal{M}^s_{\mathbf{r}}, \ (\mathcal{X}, \xi) \mapsto P^{\textit{sf-hosvd}}_{\mathbf{r}}(\mathcal{X} + \xi)$$

with $P^{\textit{sf-hosvd}}_{\mathbf{r}}$ *from Theorem 3.2 is a retraction.*

Proof. Since SF-HOSVD gives a quasioptimal approximation to a tensor, one may utilize the proof from (Steinlechner, 2016, p. 50-52). □

5 OPTIMIZATION ON THE MANIFOLD

In this section, we discuss a Riemannian gradient-based algorithm on SF-Tucker manifold to solve:

$$\min_{\mathcal{X} \in \mathcal{M}^s_{\mathbf{r}}} f(\mathcal{X}), \tag{5}$$

where the cost function $f \colon \mathcal{M}^s_{\mathbf{r}} \to \mathbb{R}$ is a restriction of some smooth function $\bar{f} \colon \mathbb{R}^{n_1 \times \dots \times n_d} \to \mathbb{R}$ defined on the whole Euclidean space $\mathbb{R}^{n_1 \times \dots \times n_d}$, with $f = \bar{f}\big|_{\mathcal{M}^s_{\mathbf{r}}}$. In particular, we will be interested in loss functions arising when training neural networks and log-loss from knowledge graphs (see Section 6.2).

5.1 Gradient Descent With Momentum

Our objective is to apply an algorithm that solves (5) by producing a sequence of points from the manifold, which converges to a local or a global minimum of f. We choose a method that utilizes gradient-based search direction enhanced by the previous search direction, called momentum. In other words, the new direction M_k is a linear combination of $\nabla_{\mathcal{M}^s_{\mathbf{r}}} f(\mathcal{X}_k)$ – the Riemannian gradient of f at the current iterate $\mathcal{X}_k$ and the momentum M_{k-1}. Considering the fact that $M_k \notin \mathrm{T}_{\mathcal{X}_k}\mathcal{M}_{\mathbf{r}}$, one typically applies the vector transport operation (Absil et al., 2008; Vandereycken, 2013) $\tau_k \colon \mathrm{T}_{\mathcal{X}_{k-1}}\mathcal{M}^s_{\mathbf{r}} \to \mathrm{T}_{\mathcal{X}_k}\mathcal{M}^s_{\mathbf{r}}$ to combine them correctly. Thus,

$$M_k = \nabla_{\mathcal{M}^s_{\mathbf{r}}} f(\mathcal{X}_k) + \beta \tau_k (M_{k-1}), \quad \beta > 0. \tag{6}$$

To obtain a new point from the manifold in the iterative process, we use retraction: $\mathcal{X}_{k+1} = R(\mathcal{X}_k, -\alpha M_k)$, where $\alpha > 0$ represents step size and is a hyperparameter of the algorithm. One can also use an adaptive strategy for choosing α based on line-search schemes, such as Armijo backtracking (Absil et al., 2008). Note that for the problems under consideration, it is common to split the dataset in batches and use stochastic versions of algorithms. We, therefore, also follow this approach and call the utilized method Riemannian SGD with momentum (R-SGD) (Ghadimi et al., 2015). In the remainder of the section we discuss efficient implementation of the proposed schemes for general f and summarize the method in 2.

Riemannian Gradient Computation. Since the Riemannian gradient $\nabla_{\mathcal{M}^s_{\mathbf{r}}} f = \pi_{\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}} \nabla f$ is an element of a tangent space $\mathrm{T}_{\mathcal{X}}\mathcal{M}^k_{\mathbf{r}}$, it can be represented via $\dot{G}, \dot{V}_i, \dot{U}$ from (4) with the rank at most $2\mathbf{r}$. Since it has a low-rank structure, one can expect to compute the Riemannian gradient without forming full-sized tensors. A naive low-rank approach of computing the Riemannian gradient is to first compute the Euclidean gradient $\nabla f(\mathcal{X})$ and then project the result to the tangent space using the projection formulas in Section 4.2. However, even if $\nabla f(\mathcal{X})$ admits a low-rank representation, typically is ranks are substantially larger than r (Novikov et al., 2022). At the same time the Riemannian gradient is guaranteed to have rank at most $2\mathbf{r}$. Therefore, we explicitly utilize the tangent vector representation (3) and use autodiff to avoid forming the Euclidean gradient and to obtain asymptotically optimal complexity.

Let us define the mapping $h = f \circ \mathcal{T}_{\mathcal{X}}$, where $\mathcal{T}_{\mathcal{X}}\left(S, B^{(1)}, \ldots, B^{(d_t)}, A\right) = \llbracket \mathcal{F}(\mathcal{G}, S); W^{(1)}, \ldots, Y \rrbracket$ with $W^{(k)} = \left(V^{(k)} \mid B^{(k)}\right)$ and $Y = \left(U \mid A\right)$. One may note that $h(G, 0, \ldots, 0) = f(\mathcal{X})$. Thus, similarly to (Novikov et al., 2022), we construct the Riemannian gradient of f at $\mathcal{X}$ by automatic differentiation of h with respect to its arguments. See Appx. C for a detailed derivation. Here we only state the final expression for the Riemannian gradient computation as a ξ from (4):

$$\dot{G} = \frac{\partial h}{\partial S}, \quad \dot{V}_i = P^{\perp}_{V^{(i)}} \frac{\partial h}{\partial B^{(i)}} \left(G_{(i)} G_{(i)}^T\right)^{-1},$$
$$\dot{U} = P^{\perp}_{U} \frac{\partial h}{\partial A} \left(\sum_{\gamma = d_t + 1}^{d} G_{(\gamma)} G_{(\gamma)}^T \right)^{-1}. \qquad (7)$$

The total complexity of the Riemannian gradient and momentum update computation is $\mathcal{O}\left(F + dNr^2 + dr^{d+1}\right)$ FLOPs, where F is time complexity of computing h.

Vector Transport. As a vector transport

$$\tau_k \colon \mathrm{T}_{\mathcal{X}_{k-1}} \mathcal{M}^s_{\mathbf{r}} \to \mathrm{T}_{\mathcal{X}_k} \mathcal{M}^s_{\mathbf{r}}$$

we use the projection (Absil et al., 2008) operator: $\tau(\xi) \equiv \pi_{\mathrm{T}_{\mathcal{X}_k} \mathcal{M}^s_{\mathbf{r}}} \xi$, $\xi \in \mathrm{T}_{\mathcal{X}_{k-1}} \mathcal{M}^s_{\mathbf{r}}$. The trick to compute the projection using the tools above is as follows. We define $g(\mathcal{X}) = \langle \mathcal{X}, M_k \rangle$. Then, $\nabla_{\mathcal{M}^s_{\mathbf{r}}} g(\mathcal{X}) = \pi_{\mathrm{T}_{\mathcal{X}_k} \mathcal{M}^s_{\mathbf{r}}} M_k$ and it can be computed using the aforementioned autodiff approach. The complexity of $g(\mathcal{X})$ is $\mathcal{O}(dNr^2 + dr^{d+1})$ (Steinlechner, 2016).

Retraction. The final step of the k-th iteration is the retraction operation $\mathcal{X}_{k+1} = R(\mathcal{X}_k, -\alpha M_k)$. Using Theorem 4.2, we have $\mathcal{X}_{k+1} = P^{\texttt{sf-hosvd}}_{\mathbf{r}}(\mathcal{X}_k - \alpha M_k)$. Note that $\mathcal{X}_k - \alpha M_k$ is also a vector on the tangent space and can be computed efficiently using a linear combination of the respective $\dot{\mathcal{G}}, \dot{V}^{(1)}, \ldots, \dot{U}$ from (3). SF-HOSVD of a vector in the SF-Tucker format can then be computed efficiently using Algorithm 1. The final algorithm is summarized in Algorithm 2.

Algorithm 2 Riemannian SGD with momentum (R-SGD)

```
Require: Initial guess $\mathcal{X}_1 \in \mathcal{M}^s_{\mathbf{r}}$, momentum weight $\beta$,
  learning rate $\alpha$.
Ensure: A sequence of iterates $\{\mathcal{X}_k\}$
  $M_0 = 0$
  for k = 1, 2, ... do
    Define $F(\mathcal{X}) = f(\mathcal{X}) + \langle \mathcal{X}, M_{k-1} \rangle$
    Compute $M_k = \nabla_{\mathcal{M}^s_{\mathbf{r}}} F(\mathcal{X}_k)$ using (4) and (7).
    $\mathcal{X}_{k+1} = P^{\texttt{sf-hosvd}}_{\mathbf{r}}(\mathcal{X}_k - \alpha M_k)$ using Algorithm 1.
```

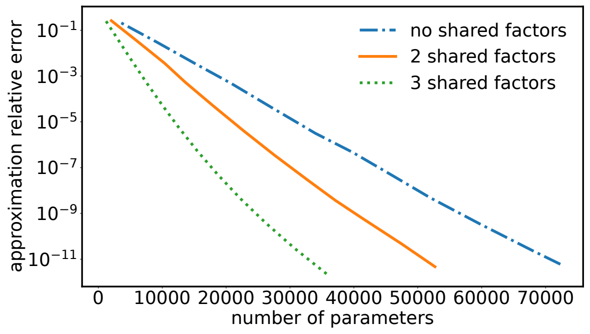

Figure 1: Compressing a $1024 \times 1024 \times 1024$ tensor of $f(x, y, z) = (x + 2y + 3z)^{-1}$ evaluated at nodes of a uniform tensor-product grid using SF-Tucker with the different number of shared factors.

6 NUMERICAL EXPERIMENTS

6.1 Approximating Grid Functions

In this demonstration, we show an example of how SF-Tucker decomposition can be used for data compression. We consider three-dimensional function $f(x, y, z) = (x + 2y + 3z)^{-1}$ on $(0, 1]^3$ and represent it on a $1024 \times 1024 \times 1024$ grid $\{x_i = i/n, i = 1, 2, \ldots, n\}^3$. We employ the approach described in Appx A.1 to construct the SF-Tucker representation. In Figure 1, we compare the total number of unique parameters in Tucker and SF-Tucker formats with 2 and 3 shared factors. Note, that the classic Tucker decomposition is equivalent to the case with no shared factors.

6.2 Knowledge Graphs Link Prediction

A knowledge graph can be represented as a three-dimensional binary tensor, where each element corresponds to a triple, with 1 indicating a true fact and 0 indicating the unknown. TuckER (Balažević et al., 2019) is a state-of-the-art method that uses the Tucker tensor decomposition to solve the link prediction problem. It represents the knowledge graph as $\llbracket \mathcal{W}; R, E, E \rrbracket$, where R and E are factors with embeddings of entities and relations, and the core tensor $\mathcal{W}$ shows the level of interactions between them. TuckER's representation can also be viewed as SF-Tucker decomposition since it does not differentiate between subject and object embeddings. For a given triple (r, e_s, e_o) TuckER estimates its score by the score function: $\varphi(e_s, r, e_o) = \llbracket \mathcal{W}; \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rrbracket$, where $\mathbf{w}_r, \mathbf{e}_s, \mathbf{e}_o$ are corresponding embeddings stored in factors E and R. After that logistic sigmoid function is applied to each $\varphi(r, e_s, e_o)$ to obtain the possibility of triple being true. We modified the TuckER model training to fit our framework, called R-TuckER, where R stands for Riemannian. To achieve this, we represent the knowledge graph as an SF-Tucker decomposition with one regular factor corresponding to relations, and two shared factors for subjects and objects. The rank $(r_{\text{rel}}, r_{\text{ent}})$ of the decomposition also determines the dimension of the corresponding embedding spaces. Although our approach inherited the link prediction-related ideas from TuckER, we do not employ techniques such as dropout (Srivastava et al. (2014)) and batch normalization (Ioffe and Szegedy (2015)), which are used in TuckER. We observe, that selecting a rank can prevent overfitting.

Implementation We use the link prediction data augmentation technique, formally described by Lacroix et al. (2018), of adding reciprocal relations for every triple in the dataset. We also use 1-to-N scoring procedure introduced by Dettmers et al. (2018), which is simultaneous score of a given pair (e_s, r) with all entities. The log-loss function is utilized to train the model, where a specific entity-relation pair's loss component defined as

$$l = -\frac{1}{N_e}\sum_{i=1}^{N_e}(y_i \log p_i + (1 - y_i)\log(1 - p_i)), \quad (8)$$

where N_e is the total number of entities in the knowledge graph, $p \in \mathbb{R}^{N_e}$ is the vector of predicted probabilities, $y \in \mathbb{R}^{N_e}$ is the binary label vector. To prevent the norm of the Riemannian gradient from becoming too low due to the vast number of entities present in modern knowledge graphs, we normalize it on each training step. For optimizing (8) we use our R-SGD with momentum described in Section 5.1. At evaluation time we combine a given pair (e_s, r) with all possible entities, generating the scores of the obtained triple, and rank them. We use the filtered setting from (Bordes et al., 2013) of removing all the known triples except the current test one from the candidate set. We use the same metrics as TuckER: mean reciprocal rank (MRR) and hits@k, $k \in \{1, 10\}$.

Experiments. Our method is evaluated using the standard benchmarks for knowledge graph link prediction, namely **FB15k-237** (Toutanova et al., 2015) and **WN18RR** (Dettmers et al., 2018). We determined the rank and learning rate hyperparameters through grid search on heldout data. For FB15k-237, we set the rank to $(200, 40)$, while for WN18RR, we choose rank be $(10, 200)$ as it contains more entities and only 11 relations. We choose learning rate (lr) within the range of $[100, 2000]$ due to the difficulty of using lower values without batch normalization. We also used one cycle scheduler: Initially, we begin with a lr of 100 and rapidly increase it to 500/850 for FB15k-237/WN18RR within the first 100 epochs. Subsequently, we gradually decrease the lr back to 100 over the following 400 epochs.

Table 1: Link prediction results on WN18RR and FB15k-237. The best scores are highlighted in bold font, top-2 scores are underlined.

Model	MRR	Hits@10	Hits@1
	WNRR18		
R-TuckER$_{\text{no BN}}$	**47.9** ±.2	**54.6** ±.2	**44.6** ±.1
TuckER$_{\text{no BN}}$	45.0 ±.3	46.1 ±.4	42.4 ±.2
TuckER	<u>47.0</u>	<u>52.6</u>	<u>44.3</u>
ConvE	43.0	52.0	40.0
DistMult	43.0	49.0	39.0
	FB15k-237		
R-TuckER$_{\text{no BN}}$	<u>32.9</u> ±.2	<u>50.5</u> ±.2	<u>24.2</u> ±.1
TuckER$_{\text{no BN}}$	32.6 ±.4	50.5 ±.4	23.7 ±.2
TuckER	**35.8**	**54.4**	**26.6**
R-GCN	24.8	41.7	15.1
ConvE	32.5	50.1	23.7
DistMult	24.1	41.9	15.5

The results of our experiments may be found in Table 1. We compare our results with TuckER, as we basically adapted this method to our framework. However, we discovered that the utilization of batch normalization, which the authors of TuckER employ to speed up the training process, has an impact on the model, causing it to deviate from Tucker decomposition-based model. Therefore, we compare our method with two types of TuckER models: one with batch normalization enabled (as provided by the authors), and the other with the second batch normalization and subsequent dropout disabled. Although, the second type basically implements the method described in the paper, we note, that it achieves worse performance than vanilla TuckER. We defer the question of the tensor decomposition type of the TuckER model for future research. We also conducted a comparison between our approach and several link prediction models we discussed in related works. See Appx. D for more experiments, hyperparameter values and comparison with various ranks.

Regularization. As our model does not utilize dropout, it tends to overfitting on large ranks. To prevent overfitting, we apply the regularization term, which is Frobenius norm of the entire knowledge graph tensor $\|[\![\mathcal{W}; E, R, E]\!]\|_F$. As we force factors R and E be orthogonal, the norm of the full tensor is equal to the norm of the core tensor $\|\mathcal{W}\|_F$. Furthermore, we incorporate a dynamic regularization coefficient. For FB15k-237, we initially use a lower value and gradually increase it until it reaches a threshold. Conversely, for WN18RR, we start with a larger value and exponentially decrease it. Once the threshold is reached, the training process moves on with the final regularization coefficient value.

Discussion. We mostly compare our method R-TuckER with TuckER model with second batch normalization disabled, as they both implements SF-Tucker decomposition of knowledge graph tensor. Our method outperforms both linearized and vanilla TuckER models on WN18RR dataset over all the metrics we observe. Our model also contains fewer parameters: rank $(10, 200)$ leads to $8.6M$ of parameters for our model against $9.8M$ of TuckER with rank $(30, 200)$.

On FB15k-237 dataset, our model shows comparable performance to the TuckER model without BN, but still falls short of the TuckER. We also note that our choice of model rank is $(200, 40)$ is lower than that of TuckER: $(200, 200)$, which leads to a significant decrease of trainable parameters: $967K$ for R-TuckER against $11.3M$ for TuckER. Furthermore, it should be noted that our regularization method enhanced performance, although, it is still unable to prevent overfitting on large ranks. Also, since the TuckER model utilizes BN to factor matrices, it introduces a bias tensor that makes the model different from the pure Tucker decomposition with shared factors. We make a conjecture that incorporating the bias into the model can boost the performance of our model. We postpone studying how to incorporate bias tensors into the Riemannian framework for future research.

6.3 Neural Networks

Tucker/SF-Tucker operators Here we apply our Riemannian framework for neural networks compression. We define a Tucker linear operator similar to TT-operator in (Novikov et al., 2015). Let $A \in \mathbb{R}^{M\times N}$ and M, N such that $M = m_1m_2m_3$, $N = n_1n_2n_3$. We apply the following chain of reshapes and transposes to represent A as a tensor $\mathcal{A} \in \mathbb{R}^{m_1n_1\times m_2n_2\times m_3n_3}$:

$$(M, N) \mapsto (m_1, m_2, m_3, n_1, n_2, n_3) \mapsto \\ \mapsto (m_1, n_1, m_2, n_2, m_3, n_3) \mapsto (m_1n_1, m_2n_2, m_3n_3).$$

Thus, $A_{ij} = \mathcal{A}_{\overline{i_1j_1}, \overline{i_2j_2}, \overline{i_3j_3}}$, where $i = i_3 + (i_2 - 1)m_1 + (i_1 - 1)m_1m_2$, and $\overline{i_1, j_1} = j_1 + (i_1 - 1)n_1$ and similarly for the others. Then we can decompose the tensor $\mathcal{A}$ using either Tucker or SF-Tucker formats: $\mathcal{A} = [\![\mathcal{W}; V^{(1)}, V^{(2)}, V^{(3)}]\!]$, $\mathcal{G} \in \mathbb{R}^{r_1\times r_2\times r_3}$, $V^{(k)} \in \mathbb{R}^{m_k\times n_k\times r_k}$, $k = 1, 2, 3$. Note, that factors $V^{(k)}$ are three-dimensional tensors, as they depend on both indices i_k, j_k. However, they can be stored in memory as two-dimensional matrices of size $m_kn_k \times r_k$. Hence the proposed approach from Section 4 is applicable.

Let us discuss how a linear layer is applied. If $x \in \mathbb{R}^{n_1n_2n_3}$, then it is firstly reshaped to tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times n_3} : x = \mathrm{vec}(\mathcal{X})$. The matrix-vector multiplication of $y \equiv \mathrm{vec}(\mathcal{Y}) = Ax$ is as follows:

$$\mathcal{Y}_{i_1i_2i_3} = \sum_{\substack{\alpha_1,\alpha_2,\alpha_3\\ j_1,j_2,j_3}} \mathcal{G}_{\alpha_1\alpha_2\alpha_3} V^{(1)}_{i_1j_1\alpha_1} V^{(2)}_{i_2j_2\alpha_2} V^{(3)}_{i_3j_3\alpha_3} \mathcal{X}_{j_1j_2j_3}.$$

BERT Compression We consider BERT[2] (Devlin et al., 2019) neural network pretrained on sentiment analysis task on IMDb Movie Reviews (Maas et al., 2011) dataset. The model has 12 BERT layers, $109M$ of trainable parameters and initial accuracy of 91.2%. Each BERT layer consists of an attention subblock and MLP subblock, and each attention subblock contains 4 linear layers of size 768×768. We compress these linear layers in the attention subblocks within eight inner middle BERT blocks. We remain the first and the last two BERT blocks unchanged, as their compression noticeably affects accuracy of the model.

Experimental Setups Following the procedure described above, we represent linear layers of sizes 768×768 as $16^2 \times 16^2 \times 3^2$ tensors. Then we utilize the Tucker model and the SF-Tucker model with sharing the first two factors. The ranks are hyperparameters and are chosen so as to Tucker and SF-Tucker decompositions contain approximately equal amount of parameters. The same rank is used for all layers we compress. Once the layers are decomposed using SF-HOSVD, we fine-tune them for 10 epochs in order to compensate for performance degradation caused by compression.

Algorithm 3 Riemannian Adam (R-Adam)

```
Require: Initial guess $\mathcal{X}_1 \in \mathcal{M}^s_{\mathbf{r}}$, momentum coefficients $\beta_1$ and $\beta_2$, $\varepsilon = 10^{-8}$, learning rate $\alpha$, loss function $f(\mathcal{X})$.
Ensure: A sequence of iterates $\{\mathcal{X}_k\}$
    $M_0 = 0, v_0 = 1$
    for k = 1, 2, ... do
        Compute $\mathcal{G}_k = \nabla_{\mathcal{M}^s_{\mathbf{r}}} f(\mathcal{X}_k)$ using (4) and (7).
        Compute $M_k = \beta_1\tau_k(M_{k-1}) + (1-\beta_1)\mathcal{G}_k$ similarly to (6).
        $v_k = \beta_2 v_{k-1} + (1-\beta_2)\|\mathcal{G}_k\|_F^2$
        $\hat{v}_k = \frac{v_k}{(1-\beta_2)}$
        $\mathcal{X}_{k+1} = P^{\texttt{sf-hosvd}}_{\mathbf{r}}(\mathcal{X}_k - \alpha\frac{v_k}{(1-\beta_1)\sqrt{\hat{v}_k+\varepsilon}})$ using Algorithm 1.
```

For optimization, we modify the Riemannian Adam method described in (Li et al., 2019) to our manifold. We optimize cross entropy loss using our proposed R-Adam 3 algorithm with $lr = 0.1$ and exponential decay rate of 0.7. The results of the experiments are

[2]Weights were taken from huggingface.

Table 2: Compressing fully-connected layers in attention blocks of BERT on IMDb. We provide the number of parameters per a compressed linear layer, the rank of the decomposition, and the test accuracy after fine-tuning using both non-Riemannian and Riemannian methods. The three-element rank refers to the Tucker decomposition, while the two-element rank – to the SF-Tucker decomposition with two shared factors. A separate line corresponds to the baseline.

Params	Rank	Acc. (%) (non-Riem.)	Acc. (%) (Riem.)
$1K$	$(4, 3)$	$87.54 \pm .07$	$\mathbf{89.17} \pm .11$
$1K$	$(2, 2, 3)$	$88.15 \pm .09$	$89.14 \pm .13$
$2.2K$	$(8, 3)$	$88.88 \pm .52$	$\mathbf{89.5} \pm .18$
$2.2K$	$(4, 4, 3)$	$88.67 \pm .64$	$89.23 \pm .06$
$5K$	$(16, 3)$	$89.31 \pm .36$	$\mathbf{89.6} \pm .07$
$5K$	$(8, 8, 3)$	$89.04 \pm .36$	$89.3 \pm .07$
$20K$	$(50, 3)$	$89.47 \pm .17$	$\mathbf{89.62} \pm .08$
$20K$	$(33, 33, 3)$	$89.4 \pm .19$	$89.35 \pm .08$
$50K$	$(94, 3)$	$\mathbf{89.66} \pm .1$	$89.65 \pm .1$
$50K$	$(70, 70, 3)$	$89.46 \pm .09$	$89.6 \pm .07$
$115K$	$(158, 3)$	$89.69 \pm .17$	$\mathbf{89.76} \pm .17$
$115K$	$(128, 128, 3)$	$89.71 \pm .19$	$89.48 \pm .19$
$590K$	–	–	91.2

presented in Table 2. For comparison, we also provide results of fine-tuning, obtained by a non-Riemannian approach: we use the classic Adam optimizer with lr of 0.001 and the same decay rate. We optimize over cores and factors of the obtained Tucker/SF-Tucker representations of layers.

Discussion Table 2 suggests that in this setting, the Riemannian approach systematically provides better quality than the classical optimization methods over factors of Tucker/SF-Tucker decompositions. We note, that even with a significantly smaller number of parameters ($2.2K$), SF-Tucker with Riemannian training performs almost on par with the case having $115K$ parameters. Conversely, the non-Riemannian approach with a similar number of parameters demonstrates noticeable accuracy reduction and with higher variance. Furthermore, the table suggests that SF-Tucker yields better quality than the Tucker decomposition with approximately the same number of parameters. This happens both in Riemannian and non-Riemannian settings.

By reducing the number of trainable parameters we also decrease both training and inference time. Note that the inference time is for the whole model. We summarize the time performance in Table 3. Note that direct optimization over factors of decomposition (without Riemannian approach) would be even faster during training.

Table 3: Comparing the time taken by one training epoch/inference stage. Baseline time is separated.

# params	$115K$	$50K$	$2.2K$	$590K$
Epoch time	645 s	580 s	516 s	678 s
Inference time	183 s	182 s	178 s	222 s

7 CONCLUSION

We proposed a tensor factorization that generalizes the classic Tucker decomposition by allowing some of the factors to be shared. We also develop an efficient numerical method based on Riemannian optimization techniques that naturally captures the shared-factor structure of the decomposition and is applicable to various problems with loss functions of general form. Numerical experiments suggest that using the SF-Tucker decomposition is superior to a non-constraint Tucker decomposition in several machine learning applications. Preliminary experiments also show benefits of utilizing SF-Tucker for approximating functions, which can be useful, e.g., when solving partial differential equations.

8 LIMITATIONS

As any tensor can be transformed into the SF-Tucker format from its Tucker representation (see Section 3.2), in practice our approach does not introduce any additional limitations. It rather inherits the limitations of the Tucker tensor decomposition.

Acknowledgments

The publication was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University №70-2021-00139. The calculations were performed in part through the computational resources of HPC facilities at HSE University (Kostenetskiy et al., 2021).

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*, volume 78. 12 2008. ISBN 978-0-691-13298-3. doi: 10.1515/9781400830244.

I. Balažević, C. Allen, and T. Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, 2019.

Z. Bi, S. Cheng, J. Chen, X. Liang, N. Zhang, Q. Chen, F. Xiong, W. Guo, and H. Chen. Relphormer: Relational graph transformer for knowledge graph representations. *arXiv preprint arXiv:2205.10852*, 2023.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

V. Chekalina, A. Razzhigaev, A. Sayapin, E. Frolov, and A. Panchenko. MEKER: Memory efficient knowledge embedding representation for link prediction and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 355–365, 2022.

L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4): 1253–1278, 2000.

T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2D knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

A. Fonarev, O. Hrinchuk, I. Oseledets, G. Gusev, and P. Serdyukov. Riemannian optimization for skip-gram negative sampling. In *ACL 2017-55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, pages 2028–2036, 2017.

E. Frolov and I. Oseledets. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3):e1201, 2017.

T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016.

E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. In *2015 European control conference (ECC)*, pages 310–315. IEEE, 2015.

W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722, 2009.

R. A. Harshman. Models for analysis of asymmetrical relationships among n objects or stimuli. In *Paper presented at the First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton*, 1978.

K. Hayashi, T. Yamaguchi, Y. Sugawara, and S.-i. Maeda. Exploring unexplored tensor network decompositions for convolutional neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189, 1927.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

H. Kasai and B. Mishra. Low-rank tensor completion: a riemannian manifold preconditioning approach. In *International conference on machine learning*, pages 1012–1021. PMLR, 2016.

S. M. Kazemi and D. Poole. SimplE embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

A. Kolbeinsson, J. Kossaifi, Y. Panagakis, A. Bulat, A. Anandkumar, I. Tzoulaki, and P. Matthews. Robust deep networks with randomized tensor regression layers. *arXiv*, 2019.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

J. Kossaifi, Z. C. Lipton, A. Kolbeinsson, A. Khanna, T. Furlanello, and A. Anandkumar. Tensor regression networks. *Journal of Machine Learning Research*, 21, 07 2020. URL https://www.jmlr.org/papers/volume21/18-503/18-503.pdf.

P. Kostenetskiy, R. Chulkevich, and V. Kozyrev. HPC resources of the higher school of economics. In *Jour-*

nal of Physics: Conference Series, volume 1740, page 012050, 2021.

D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54:447–468, 2014.

T. Lacroix, N. Usunier, and G. Obozinski. Canonical tensor decomposition for knowledge base completion. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2863–2872. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/lacroix18a.html.

V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings*, 2015.

J. Li, F. Li, and S. Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. In *International Conference on Learning Representations*, 2019.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

D. Nathani, J. Chauhan, C. Sharma, and M. Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1466. URL https://aclanthology.org/P19-1466.

M. Nickel, V. Tresp, H.-P. Kriegel, et al. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 3104482–3104584, 2011.

A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.

A. Novikov, M. Trofimov, and I. Oseledets. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.

A. Novikov, M. Rakhuba, and I. Oseledets. Automatic differentiation for riemannian optimization on low-rank matrix and tensor-train manifolds. *SIAM Journal on Scientific Computing*, 44(2):A843–A869, 2022.

A. Obukhov, M. Rakhuba, S. Georgoulis, M. Kanakis, D. Dai, and L. Van Gool. T-basis: a compact representation for neural networks. In *International Conference on Machine Learning*, pages 7392–7404. PMLR, 2020.

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

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

M. M. Steinlechner. *Riemannian optimization for solving high-dimensional problems with low-rank tensor structure*. PhD thesis, EPFL, 2016.

Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*, 2020.

K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1174. URL https://aclanthology.org/D15-1174.

T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research*, 18(130):1–38, 2017.

L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15(122-137):3, 1963.

B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):A1027–A1052, 2012.

W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

Y. Yang, D. Krompass, and V. Tresp. Tensor-train recurrent neural networks for video classification. In *International Conference on Machine Learning*, pages 3891–3900. PMLR, 2017.

Checklist

1. For all models and algorithms presented, check if you include:
    (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes.**
    (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes.**
    (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes.**

2. For any theoretical claim, check if you include:
    (a) Statements of the full set of assumptions of all theoretical results. **Yes.**
    (b) Complete proofs of all theoretical results. **Yes.**
    (c) Clear explanations of any assumptions. **Not Applicable.**

3. For all figures and tables that present empirical results, check if you include:
    (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes.**
    (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes.**
    (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes.**
    (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **No.**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
    (a) Citations of the creator If your work uses existing assets. **Yes.**
    (b) The license information of the assets, if applicable. **Yes.**
    (c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable.**
    (d) Information about consent from data providers/curators. **Not Applicable.**
    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable.**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
    (a) The full text of instructions given to participants and screenshots. **Not Applicable.**
    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable.**
    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable.**

A SF-Tucker decomposition

A.1 Existence of SF-Tucker decomposition

To provide some intuition we firstly describe the form of ST-Tucker decomposition in the matrix case. Therefore, we need to find $U \in \mathbb{R}^{n\times\tilde{r}}, G \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$ with some $\tilde{r}$, such that any matrix $A \in \mathbb{R}^{n\times n}$ of rank r is represented as

$$A = UGU^T.$$

Let $A = Y\Sigma W$, where $Y, W \in \mathbb{R}^{n\times r}, \Sigma \in \mathbb{R}^{r\times r}$ be a compact SVD of A. By introducing $\bar{U} = \begin{pmatrix} Y \mid W \end{pmatrix} \mathbb{R}^{n\times 2r}$, we have

$$A = \begin{pmatrix} Y \mid W \end{pmatrix} \begin{pmatrix} 0 & \Sigma \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Y \mid W \end{pmatrix}^T = \bar{U}\bar{G}\bar{U}^T, \quad \bar{G} \in \mathbb{R}^{2r\times 2r}.$$

We can always impose additional orthogonality constraint on $\bar{U}$ by performing the following reorthogonarization trick. Let $\bar{U} = QRP$ be the rank-revealing QR decomposition, where $Q \in \mathbb{R}^{n\times\tilde{r}}$ has orthonormal columns, $R \in \mathbb{R}^{\tilde{r}\times 2r}$, $\tilde{r} = \operatorname{rank}(\bar{U})$ – upper triangular and $P \in \mathbb{R}^{2r\times 2r}$ – a permutation matrix. Then, we obtain SF-Tucker matrix decomposition of A as

$$A = \bar{U}\bar{G}\bar{U}^T = QRP\bar{G}P^TR^TQ^T = UGU^T, \quad U = Q, \quad G = RP\bar{G}P^TR^T.$$

By construnction, $\tilde{r} = \operatorname{rank}\begin{pmatrix} Y \mid W \end{pmatrix} = \operatorname{rank}\begin{pmatrix} A \mid A^T \end{pmatrix} = \operatorname{rank}\begin{pmatrix} A_{(1)} \mid A_{(2)} \end{pmatrix}$.

Now we are in the position to describe an existence of SF-Tucker decomposition in the general case. The proof contains explicit formulas for the representation.

Proposition A.1. *Let $d \in \mathbb{N}$ and $d_s \leqslant d$ be such that $d = d_t + d_s$. Consider a d-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1\times\ldots\times n_d}$ where $n_{d_t+1} = \ldots = n_d \equiv n$. Then there exist matrices $V^{(k)} \in \mathbb{R}^{n_k\times r_k}$ for $k = 1,\ldots,d_t$, a matrix $U \in \mathbb{R}^{n_d\times r_s}$, and a tensor $\mathcal{G} \in \mathbb{R}^{r_1\times\ldots\times r_{d_t}\times r_s\times\ldots\times r_s}$, $r_s \leqslant n$ such that ${V^{(k)}}^T V^{(k)} = I_{r_k}$, $U^TU = I_{r_s}$, and $\mathcal{X}$ can be represented in the SF-Tucker format:*

$$\mathcal{X} = \left[\!\left[\mathcal{G}; V^{(1)},\ldots,V^{(d_t)},U,\ldots,U\right]\!\right],$$

with the SF-Tucker rank equal to $(r_1,\ldots,r_{d_t},r_s)$.

Proof. Consider the classic Tucker decomposition of $\mathcal{X}$:

$$\mathcal{X} = \left[\!\left[\bar{\mathcal{G}}; V^{(1)},\ldots,V^{(d)}\right]\!\right], \quad \bar{\mathcal{G}} \in \mathbb{R}^{r_1\times\ldots\times r_d}, \quad (r_1,\ldots,r_d) = \operatorname{rank}_{\mathrm{ML}}\mathcal{X}.$$

Let $\bar{U} = \begin{pmatrix} V^{(d_t+1)} \mid \ldots \mid V^{(d)} \end{pmatrix} \in \mathbb{R}^{n_d\times R}$, where $R = r_{d_t+1} + \ldots + r_d$. Let also $\tilde{\mathcal{G}} \in \mathbb{R}^{r_1\times\ldots\times r_{d_t}\times R\times\ldots\times R}$ be such that

$$\mathcal{X} = \left[\!\left[\tilde{\mathcal{G}}; V^{(1)},\ldots,V^{(d_t)},\bar{U},\ldots,\bar{U}\right]\!\right].$$

Now, we perform the reorthogonalization trick that we discussed for the matrix case: $\bar{U} = QRP$, $Q \in \mathbb{R}^{n_d\times r_s}$, $R \in \mathbb{R}^{r_s\times R}$ and $P \in \mathbb{R}^{R\times R}$. Now, let $\mathcal{G} = \left[\!\left[\tilde{\mathcal{G}}; I_{n_1},\ldots,I_{n_{d_t}},RP,\ldots,RP\right]\!\right] \in \mathbb{R}^{r_1\times\ldots\times r_{d_t}\times r_s\times\ldots\times r_s}$. Thus, we obtain the SF-Tucker decomposition of $\mathcal{X}$:

$$\mathcal{X} = \left[\!\left[\mathcal{G}; V^{(1)},\ldots,V^{(d_t)},U,\ldots,U\right]\!\right], \quad U = Q.$$

Finally, let $R_k = r_{d_t+1} + \ldots + r_{d_t+k}$. Then $\tilde{G}$ can be constructed as a tensor that is zero everywhere except for:

$$\tilde{G}(1{:}r_1, 1{:}r_2,\ldots,1{:}r_{d_t}, 1{:}R_1, 1+R_1{:}R_2,\ldots,1+R_{d_s-1}{:}R_{d_s}) = \bar{\mathcal{G}}.$$

□

As an illustration, we provide two examples for $\tilde{G}$ from the proof in the three-dimensional case.

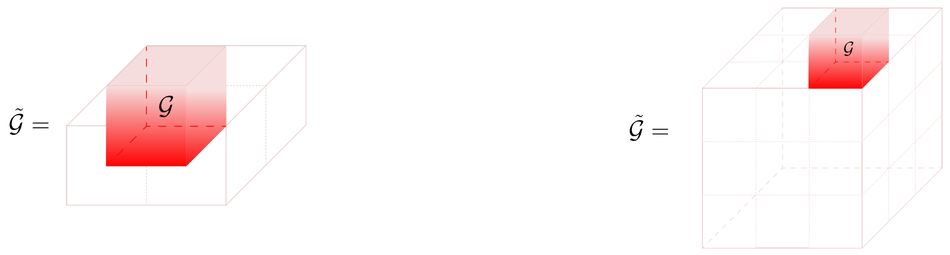

Figure 2: Block structure of the core tensor $\tilde{\mathcal{G}}$ in Example 1 (left) and Example 2 (right). Transparent blocks consist of zeros.

Example 1. *Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n \times n}$ We want to obtain SF-Tucker decomposition of the form $\mathcal{X} = [\![\bar{\mathcal{G}}; V, U, U]\!]$. Let $V^{(1)} \in \mathbb{R}^{n_1 \times r_1}, V^{(2)} \in \mathbb{R}^{n \times r_2}, V^{(3)} \in \mathbb{R}^{n \times r_3}$ and $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ be such that*

$$\mathcal{X} = \left[\!\!\left[\mathcal{G}; V^{(1)}, V^{(2)}, V^{(3)}\right]\!\!\right].$$

Following the proof above, we construct $\bar{U} = \left(V^{(2)} \mid V^{(3)}\right)$, and $\tilde{\mathcal{G}}$, which is zero everywhere except for $\tilde{\mathcal{G}}(1{:}r_1, 1{:}r_2, r_3{+}1{:}2r_3) = \mathcal{G}$. See Figure 2 (left). Further, we can apply the reorthogonalization trick and obtain the SF-Tucker form with orthogonal factors.

Example 2. *Let $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$. We want to obtain a decomposition of the following form: $\mathcal{X} = [\![\bar{\mathcal{G}}; U, U, U]\!]$. Again, let $V^{(1)} \in \mathbb{R}^{n \times r_1}, V^{(2)} \in \mathbb{R}^{n \times r_2}, V^{(3)} \in \mathbb{R}^{n \times r_3}$ and $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ be such that*

$$\mathcal{X} = \left[\!\!\left[\mathcal{G}; V^{(1)}, V^{(2)}, V^{(3)}\right]\!\!\right]$$

is a Tucker decomposition of $\mathcal{X}$. Combining its factors we obtain $\bar{U} = \left(V^{(1)} \mid V^{(2)} \mid V^{(3)}\right)$ and $\tilde{\mathcal{G}}$ that is zero except for

$$\tilde{\mathcal{G}}(1{:}r_1, 1 + r_1{:}r_1 + r_2, 1 + r_1 + r_2{+}1{:}r_1 + r_2 + r_3) = \mathcal{G}.$$

The structure of $\tilde{\mathcal{G}}$ is presented in Figure 2 (right).

A.2 SF-Tucker rank (Proof of Theorem 3.1)

To prove Theorem 3.1 and show that the rank is defined correctly, we need the fact we that if $\mathcal{X}$ admits the Tucker representation:

$$\mathcal{X} = \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d)}\right]\!\!\right],$$

then its k-th unfolding can be obtained as (Kolda and Bader, 2009):

$$\mathcal{X}_{(k)} = V^{(k)} \mathcal{G}_{(k)} \left(V^{(d)} \otimes \ldots \otimes V^{(k+1)} \otimes V^{(k-1)} \otimes \ldots \otimes V^{(1)}\right)^T. \tag{9}$$

Considering the fact that SF-Tucker can be viewed as a form of Tucker decomposition, the k-th unfolding of the tensor of SF-Tucker form is defined similarly. For non-shared modes, we have

$$\mathcal{X}_{(k)} = V^{(k)} \mathcal{G}_{(k)} \left(U \otimes U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(k+1)} \otimes V^{(k-1)} \otimes \ldots \otimes V^{(2)} \otimes V^{(1)}\right)^T$$

Since it is written in the form of skeleton matrix decomposition, we have $r_k \geqslant \operatorname{rank} \mathcal{X}_{(k)}$ for $1 \leqslant k \leqslant d_t$. For shared modes ($d_t < k \leqslant d$):

$$\mathcal{X}_{(k)} = U \mathcal{G}_{(k)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right)^T.$$

For simplicity, let us denote $M_k = \mathcal{G}_{(k)} \left(U \otimes \cdots \otimes U \otimes V^{(d_t)} \otimes \cdots \otimes V^{(1)}\right)^T$ and let us concatenate all these unfolding matrices into a single one:

$$\left(\, \mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \ldots \mid \mathcal{X}_{(d)} \,\right) = U \left(\, M_{d_t+1} \mid M_{d_t+2} \mid \ldots \mid M_d \,\right).$$

As a result, we get $r_s \geqslant \operatorname{rank}\left(\mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \dots \mid \mathcal{X}_{(d)} \right)$.

We have just obtained the lower bounds for our ranks. It remains to show that a decomposition with such ranks exists. Let the matrices $V^{(1)}, \dots, V^{(d_t)}$ be assembled from the columns forming the basis of the image of the respective unfolding matrices and the concatenation of unfoldings in the case of the shared factor U. We set:

$$\mathcal{G} = [\![\mathcal{X}; {V^{(1)}}^T, {V^{(2)}}^T, \dots, {V^{(d_t)}}^T, U^T, \dots, U^T]\!].$$

Then the resulting tensor is equal to

$$[\![\mathcal{G}; V^{(1)}, \dots, V^{(d_t)}, U, \dots, U]\!] = [\![\mathcal{X}; V^{(1)}{V^{(1)}}^T, \dots, V^{(d)}{V^{(d_t)}}^T, UU^T, \dots, UU^T]\!].$$

Note, that multiplication by $V^{(i)}{V^{(i)}}^T$ or UU^T for each mode corresponds to the projection onto the linear span of the columns of the respective matrices. However, since these columns are chosen to form a basis of the image of the unfolding, each projection will act identically. Hence, the resulting tensor will be equal to $\mathcal{X}$.

A.3 Quasi-optimality of SF-HOSVD (Proof of Theorem 3.2)

Let $\mathcal{T}_* = [\![\mathcal{G}_*; V_*^{(1)}, \dots, V_*^{(d_t)}, U_*, \dots, U_*]\!]$ be the optimal approximation with a rank not larger than a given rank. Let $\mathcal{T} = [\![\mathcal{G}; V^{(1)}, \dots, V^{(d_t)}, U, \dots, U]\!]$ be the decomposition obtained using SF-HOSVD.

It can be shown (De Lathauwer et al., 2000), that for fixed $V_*^{(1)}, \dots, V_*^{(d_t)}$ and U_* the core tensor $\mathcal{G}_*$ can be obtained as follows:

$$\mathcal{G}_* = [\![\mathcal{X}; {V_*^{(1)}}^T, {V_*^{(2)}}^T, \dots, {V_*^{(d_t)}}^T, U_*^T, \dots, U_*^T]\!],$$

and therefore

$$\mathcal{T}_* = [\![\mathcal{X}; V_*^{(1)}{V_*^{(1)}}^T, \dots, V_*^{(d_t)}{V_*^{(d_t)}}^T, U_*U_*^T, \dots, U_*U_*^T]\!].$$

We also introduce the tensor times matrix multiplication along the k-th mode:

$$(\mathcal{X} \times_k M)_{i_1 \dots i_{k-1} j_k i_{k+1} \dots i_d} = \sum_{i_k=1}^{n_k} \mathcal{X}_{i_1, \dots, i_d} M_{j_k i_k}.$$

and the respective orthogonal projection operators (Vannieuwenhoven et al., 2012):

$$\pi_i \mathcal{A} \equiv \mathcal{A} \times_i V^{(i)}{V^{(i)}}^T, \quad \pi_i^* \mathcal{A} \equiv \mathcal{A} \times_i V_*^{(i)}{V_*^{(i)}}^T.$$

If $i > d_t$, then $\pi_i \equiv \mathcal{A} \times_i UU^T$. And rewrite $\mathcal{T}, \mathcal{T}_*$ using the introduced notation:

$$\mathcal{T} = \pi_1 \pi_2 \dots \pi_d \mathcal{X}, \quad \mathcal{T}_* = \pi_1^* \pi_2^* \dots \pi_d^* \mathcal{X}.$$

Firstly, since π_i^* and π_j^* commute, we have

$$\|\mathcal{X} - \pi_1^*\pi_2^* \dots \pi_d^* \mathcal{X}\|_F^2 = \|\mathcal{X} - \pi_i^* \mathcal{X} + \pi_i^* \mathcal{X} - \pi_i^*(\pi_1^* \dots \pi_d^* \mathcal{X})\|_F^2 =$$
$$\|\mathcal{X} - \pi_i^* \mathcal{X}\|_F^2 + \|\pi_i^* \mathcal{X} - \pi_i^*(\pi_1^* \dots \pi_d^* \mathcal{X})\|_F^2 \geqslant \|\mathcal{X} - \pi_i^* \mathcal{X}\|_F^2$$

Therefore,

$$d\|\mathcal{X} - \mathcal{T}^*\|_F^2 = d\|\mathcal{X} - \pi_1^*\pi_2^* \dots \pi_d^* \mathcal{X}\|_F^2 \geqslant \|X - \pi_1^* \mathcal{X}\|_F^2 + \dots + \|\mathcal{X} - \pi_d^* \mathcal{X}\|_F^2 = \sum_{i=1}^{d} \|\mathcal{X} - \pi_i^* \mathcal{X}\|_F^2.$$

Now we rewrite the resulting expression using unfolding matrices and concatenate all expressions for the shared factor:

$$\sum_{i=1}^{d}\|\mathcal{X}-\pi_i^*\mathcal{X}\|_F^2=\sum_{i=1}^{d_t}\left\|\mathcal{X}_{(i)}-V_*^{(i)}V_*^{(i)^T}\mathcal{X}_{(i)}\right\|_F^2+\sum_{i=d_t+1}^{d}\|\mathcal{X}_{(i)}-U_*U_*^T\mathcal{X}_{(i)}\|_F^2=\sum_{i=1}^{d_t}\left\|\mathcal{X}_{(i)}-V_*^{(i)}V_*^{(i)^T}\mathcal{X}_{(i)}\right\|_F^2+$$

$$+\left\|\left(\mathcal{X}_{(d_t+1)}\mid\mathcal{X}_{(d_t+2)}\mid\ldots\mid\mathcal{X}_{(d)}\right)-U_*U_*^T\left(\mathcal{X}_{(d_t+1)}\mid\mathcal{X}_{(d_t+2)}\mid\ldots\mid\mathcal{X}_{(d)}\right)\right\|_F^2\geqslant\sum_{i=1}^{d_t}\left\|\mathcal{X}_{(i)}-V^{(i)}V^{(i)^T}\mathcal{X}_{(i)}\right\|_F^2+$$

$$+\left\|\left(\mathcal{X}_{(d_t+1)}\mid\mathcal{X}_{(d_t+2)}\mid\ldots\mid\mathcal{X}_{(d)}\right)-U_*U_*^T\left(\mathcal{X}_{(d_t+1)}\mid\mathcal{X}_{(d_t+2)}\mid\ldots\mid\mathcal{X}_{(d)}\right)\right\|_F^2\geqslant\sum_{i=1}^{d_t}\left\|\mathcal{X}_{(i)}-V^{(i)}V^{(i)^T}\mathcal{X}_{(i)}\right\|_F^2+$$

$$+\sum_{i=d_t+1}^{d_t}\left\|\mathcal{X}_{(i)}-UU^T\mathcal{X}_{(i)}\right\|_F^2.$$

Here we used that $\left\|\mathcal{X}_{(i)}-V_*^{(i)}V_*^{(i)^T}\mathcal{X}_{(i)}\right\|\geqslant\left\|\mathcal{X}_{(i)}-V^{(i)}V^{(i)^T}\mathcal{X}_{(i)}\right\|$, as each π_i is selected independently of the others $\pi_j, j\neq i$, in order to solve

$$\min_{\pi_i:\ \dim\operatorname{Im}\pi_i=r_i}\left\|\mathcal{X}_{(i)}-\pi_i\mathcal{X}_{(i)}\right\|_F^2.$$

Conversely, the choice of π_i^* is dependent on the choices of the other $\pi_j^*, j\neq i$. Similarly for the shared factor terms. Finally, using the fact that the use of a projector does not increase the norm of the matrix, we have:

$$\sum_{i=1}^{d_t}\left\|\mathcal{X}_{(i)}-V^{(i)}V^{(i)^T}\mathcal{X}_{(i)}\right\|_F^2+\sum_{i=d_t+1}^{d_t}\left\|\mathcal{X}_{(i)}-UU^T\mathcal{X}_{(i)}\right\|_F^2=\sum_{i=1}^{d}\|\mathcal{X}-\pi_i\mathcal{X}\|_F^2\geqslant\|\mathcal{X}-\pi_1\mathcal{X}\|_F^2+$$
$$+\|\pi_1\mathcal{X}-\pi_1\pi_2\mathcal{X}\|_F^2+\ldots+\|\pi_1\pi_2\ldots\pi_{d-1}\mathcal{X}-\pi_1\pi_2\ldots\pi_dX\|_F^2\geqslant$$
$$\geqslant\|\mathcal{X}-\pi_1\mathcal{X}+\pi_1\mathcal{X}-\pi_1\pi_2\mathcal{X}+\ldots+\pi_1\pi_2\ldots\pi_{d-1}X-\pi_1\pi_2\ldots\pi_dX\|_F^2\geqslant$$
$$\geqslant\|\mathcal{X}-\pi_1\pi_2\ldots\pi_d\mathcal{X}\|_F^2=\|\mathcal{X}-\mathcal{T}\|_F^2,$$

and as a result,

$$\|\mathcal{X}-\mathcal{T}\|_F^2\leqslant d\|\mathcal{X}-\mathcal{T}^*\|_F^2,$$

which completes the proof.

B Riemannian optimization on the manifold of fixed SF-rank tensors

B.1 SF-Tucker manifold (Proof of Theorem 4.1)

We follow the proof for the Tucker manifold (Steinlechner, 2016, Theorem 3.6) and modify it for the shared mode. From the proof, we know that the set

$$\mathcal{M}_{r_1,\ldots,r_{d_t}}^{1,\ldots,d_t}=\left\{\mathcal{X}\in\mathbb{R}^{n_1\times\cdots\times n_d}|\operatorname{rank}(X_{(i)})=r_i,\quad i\leqslant d_t\right\}$$

forms a smooth manifold. Now we need to show that

$$\mathcal{M}_{\mathbf{r}}^{s}=\left\{\mathcal{X}\in\mathcal{M}_{r_1,\ldots,r_{d_t}}^{1,\ldots,d_t}\middle|\operatorname{rank}\left(\left(\mathcal{X}_{(d_t+1)}\mid\mathcal{X}_{(d_t+2)}\mid\ldots\mid\mathcal{X}_{(d)}\right)\right)=r_s\right\}$$

forms a smooth embedded submanifold of $\mathcal{M}_{r_1,\ldots,r_{d_t}}^{1,\ldots,d_t}$. Any $\mathcal{X}\in\mathcal{M}_{r_1,\ldots,r_{d_t}}^{1,\ldots,d_t}$ can be represented as

$$\mathcal{X}=\mathcal{Y}\bigtimes_{i=1}^{d_t}V^{(i)},$$

where $V^{(i)}\in\mathbb{R}^{n_i\times r_i}$ have full column rank and $\mathcal{Y}\in\mathbb{R}^{r_1\times\cdots\times r_{d_t}\times n\times\cdots\times n}$. Similarly to the proof for a regular Tucker, $\mathcal{Y}$ and $V^{(i)}$ can be chosen to depend smoothly on $\mathcal{X}$, and $\mathcal{Y}$ has the same SFT rank as $\mathcal{X}$.

Let us say $\mathcal{Y}$ can be represented in SF-Tucker format with a shared factor U:

$$\mathcal{Y}=\left[\!\left[\mathcal{G};V^{(1)},\ldots,V^{(d_t)},U,\ldots,U\right]\!\right].$$

Since U is of full rank, by rearranging the fibers of $\mathcal{Y}$, we can write $U = \begin{pmatrix} U_1^T & U_2^T \end{pmatrix}^T, U_1 \in \mathbb{R}^{r_s \times r_s}, \det(U_1) \neq 0$. We know that the shared rank of $\mathcal{Y}$ is

$$r_s = \operatorname{rank}\left(\mathcal{Y}_{(d_t+1)} \mid \mathcal{Y}_{(d_t+2)} \mid \dots \mid \mathcal{Y}_{(d)}\right) = \operatorname{rank}\left(U\left(\mathcal{G}_{(d_t+1)} \mid \mathcal{G}_{(d_t+2)} \mid \dots \mid \mathcal{G}_{(d)}\right)(U \otimes \dots \otimes V^{(1)})^T\right).$$

Let us denote this matrix as $Y = \left(\mathcal{Y}_{(d_t+1)} \mid \mathcal{Y}_{(d_t+2)} \mid \dots \mid \mathcal{Y}_{(d)}\right)$. Now we replace U with U_1 in the right bracket and introduce the notation for a matrix of subcolumns of Y:

$$Y_1 = U\left(\mathcal{G}_{(d_t+1)} \mid \mathcal{G}_{(d_t+2)} \mid \dots \mid \mathcal{G}_{(d)}\right)(U_1 \otimes \dots \otimes V^{(1)})^T.$$

However, since both U and U_1 both have full column rank, the rank of $U_1 \otimes \dots \otimes V^{(1)}$ is also full and $r_s = \operatorname{rank}(Y) = \operatorname{rank}(Y_1)$. Hence any basis Y_1 columns space will also be a basis for Y. Let us assemble matrix $W \in \mathbb{R}^{n \times r_s}$ from basis columns of Y_1. We can choose W such that $W = \begin{pmatrix} W_1^T & W_2^T \end{pmatrix}^T, W_1 \in \mathbb{R}^{r_s \times r_s}, \det(W_1) \neq 0$. From Appendix A.2, we know that any matrix consisting of basis columns can be used as a shared factor. Let us denote this representation as $\mathcal{Y} = [\![\mathcal{G}'; V^{(1)}, \dots, V^{(d_t)}, W, \dots, W]\!]$. Let us also denote by $\mathcal{T}$ its subtensor:

$$\mathcal{T} = \mathcal{Y}_{:,\dots,:,:r_s,\dots,:r_s} = \left[\!\left[\mathcal{G}'; V^{(1)}, \dots, V^{(d_t)}, W_1, \dots, W_1\right]\!\right].$$

We note that columns of W_1 are fibers of $\mathcal{T}$. Finally, we have

$$\mathcal{Y} = \mathcal{T} \times_{d_t+1} WW_1^{-1} \times_{d_t+2} \dots \times_d WW_1^{-1}. \tag{10}$$

This is a formula for reconstructing the tensor $\mathcal{Y}$ by using $r_s^{d_s} + r_s(n - r_s)$ of its elements. In terms of unfoldings it can be written as follows:

$$\left(\mathcal{Y}_{(d_t+1)} \mid \mathcal{Y}_{(d_t+2)} \mid \dots \mid \mathcal{Y}_{(d)}\right) = WW_1^{-1}\left(\mathcal{T}_{(d_t+1)} \mid \mathcal{T}_{(d_t+2)} \mid \dots \mid \mathcal{T}_{(d)}\right)\left((WW_1^{-1}) \otimes \dots \otimes I_{n_1}\right)^T.$$

This formula is correct if and only if $\operatorname{rank}(\left(\mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \dots \mid \mathcal{X}_{(d)}\right)) = r_s$. Indeed, on the one hand, if the rank is r_s, we can choose $\mathcal{T}$ and W so that the formula is satisfied. On the other hand, as $\det(W_1) \neq 0$, the matrix WW_1^{-1} has full column rank and $\operatorname{rank}(\left(\mathcal{T}_{(d_t+1)} \mid \mathcal{T}_{(d_t+2)} \mid \dots \mid \mathcal{T}_{(d)}\right)) = r_s$. As a result, we have that $\operatorname{rank}(\left(\mathcal{X}_{(d_t+1)} \mid \mathcal{X}_{(d_t+2)} \mid \dots \mid \mathcal{X}_{(d)}\right)) = r_s$. Let us use this formula to apply the submersion theorem. Consider an open set

$$S = \left\{\mathcal{X} \in \mathcal{M}^{1,\dots,d_t}_{r_1,\dots,r_{d_t}} | \mathcal{X} = \left[\!\left[\mathcal{T}; V^{(1)}, \dots, V^{(d_t)}, WW_1^{-1}, \dots, WW_1^{-1}\right]\!\right],\ \det(W_1) \neq 0\right\},$$

where $\mathcal{Y}$ is smoothly obtained from $\mathcal{X}$, and $\mathcal{T}$ and W are assembled from elements of $\mathcal{Y}$ as described above. We can describe $S \cap \dim \mathcal{M}_{\mathbf{r}}$ by the level set of

$$\Phi\colon S \to \mathbb{R}^{r_1 r_2 \dots r_{d_t} n_s^{d_s} - r_1 r_2 \dots r_{d_t} r_s^{d_s} - r_s(n - r_s)}, \quad \Phi(\mathcal{X}) = \mathcal{P}(\mathcal{Y} - \mathcal{T} \times_{d_t+1} WW_1^{-1} \times_{d_t+2} \dots \times_d WW_1^{-1}),$$

where $\mathcal{P}$ is any bijection that represents as a column vector all elements of $\mathcal{Y}$ that were not used to construct $\mathcal{T}$ and are not contained in W. Now we need to show that $D\Phi$ is a surjection. Firstly, note that because $\mathcal{Y}, \mathcal{T}$ and W are smoothly obtained from $\mathcal{X}$, Φ is smooth. Now consider

$$\gamma : \mathbb{R} \to S, \quad \gamma(t) = (\mathcal{Y} + t\mathcal{E}) \times_1 U_1 \times_2 U_2 \times \dots \times_{d_t} U_{d_t},$$

where $\mathcal{E} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ has zeros on all positions that were used to construct $\mathcal{T}$ and W. We have,

$$D\Phi(\mathcal{X})[\gamma'(0)] = \frac{d}{dt}(\Phi \circ \gamma)(t)\Big|_{t=0} = \\ \frac{d}{dt}\left(P(\mathcal{Y} + t\mathcal{E} - \mathcal{T} \times_{d_t+1} WW_1^{-1} \times \dots \times_d WW_1^{-1})\right)\Big|_{t=0} = \frac{d}{dt}\left(tP(\mathcal{E})\right)\Big|_{t=0} = P(\mathcal{E}).$$

By choosing $\mathcal{E}$ we can get any desired value of $P(\mathcal{E})$ and hence $D\Phi$ is a surjection. By the submersion theorem (Steinlechner, 2016, Prop. 2.10) we have that $\mathcal{M}^s_{\mathbf{r}}$ is a smooth manifold of the dimension:

$$\dim \mathcal{M}_{\mathbf{r}} = \dim \ker \Phi = \dim S - \dim \mathbb{R}^{r_1 r_2 \dots r_{d_t} n_s^{d_s} - r_1 r_2 \dots r_{d_t} r_s^{d_s} - r_s(n - r_s)} = r_1 r_2 \dots r_{d_t} n_s^{d_s} + \\ \sum_{i=1}^{d_t}(n_i r_i - r_i^2) - r_1 r_2 \dots r_{d_t} n_s^{d_s} + r_1 r_2 \dots r_{d_t} r_s^{d_s} + r_s(n - r_s) = r_1 r_2 \dots r_{d_t} r_s^{d_s} + \sum_{i=1}^{d_t}(n_i r_i - r_i^2) + (n r_s - r_s^2).$$

B.2 Projection onto the tangent space

Firstly, let us introduce several notions of subspaces, which are related to a fixed $\mathcal{X} = [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U]\!]$. The first one is

$$L_0 = \left\{\xi \in \mathbb{R}^{n_1 \times \ldots \times n} \mid \xi = \left[\!\!\left[\dot{\mathcal{G}}; V^{(1)}, V^{(2)}, \ldots, V^{(d_t)}, U, \ldots, U\right]\!\!\right],\ \dot{\mathcal{G}} \in \mathbb{R}^{r_1 \times \ldots \times r_s}\right\}.$$

Next, for $1 \leqslant i \leqslant d_t$ we have, d_t subspaces of the form

$$L_i = \left\{\xi \in \mathbb{R}^{n_1 \times \ldots \times n} \mid \xi = [\![\mathcal{G}; V^{(1)}, V^{(2)}, \ldots, \dot{V}^{(i)}, \ldots, U]\!],\ \dot{V}^{(i)} \in \mathbb{R}^{n_i \times r_i},\ \dot{V}^{(i)^T} V^{(i)} = 0\right\}.$$

And the final one:

$$L_s = \Big\{\xi \in \mathbb{R}^{n_1 \times \ldots \times n_{d_t} \times n \times \ldots \times n} \mid \xi = [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t-1)}, V^{(d_t)}, \dot{U}, \ldots, U]\!] + \ldots$$
$$\ldots + [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t-1)}, V^{(d_t)}, U, \ldots, \dot{U}]\!],\ \dot{U} \in \mathbb{R}^{n \times r_s},\ \dot{U}^T U = 0\Big\}.$$

Proposition B.1. *The tangent space* $\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}$ *can be decomposed as*

$$\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}} = L_0 \oplus L_1 \oplus \ldots \oplus L_{d_t} \oplus L_s.$$

Moreover, for $i, j \in \{0, 1, \ldots, d_t, s\},\ i \neq j$ *it follows* $L_i \perp L_j$.

Proof. Following the technique from (Steinlechner, 2016), we firstly show, that L_0 and L_1 are orthogonal, which means, that for any pair $\xi \in L_0, , \eta \in L_1$ holds that $\langle \xi, \eta \rangle = 0$:

$$\langle \xi, \eta \rangle = \left\langle \left[\!\!\left[\dot{\mathcal{G}}; V^{(1)}, V^{(2)}, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; \dot{V^{(1)}}, V^{(2)}, \ldots, U\right]\!\!\right] \right\rangle = \left\langle \left[\!\!\left[\dot{\mathcal{G}}; I, V^{(2)}, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; V^{(1)^T}\dot{V^{(1)}}, V^{(2)}, \ldots, U\right]\!\!\right] \right\rangle =$$
$$= \left\langle \left[\!\!\left[\dot{\mathcal{G}}; I, V^{(2)}, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; 0, V^{(2)}, \ldots, U\right]\!\!\right] \right\rangle = 0.$$

Now, we show, that for any $i, j \in \{1, \ldots, d_t\},\ i \neq j$ holds that $L_i \perp L_j$. Again, let $\xi \in L_i, \eta \in L_j$, then

$$\langle \xi, \eta \rangle = \left\langle \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, \dot{V}^{(i)}, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, \dot{V}^{(j)}, \ldots, U\right]\!\!\right] \right\rangle =$$
$$= \left\langle \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(i-1)}, I, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, \underbrace{\dot{V}^{(i)^T} V^{(i)}}_{0}, \ldots, \dot{V}^{(j)}, \ldots, U\right]\!\!\right] \right\rangle = 0.$$

Finally, we show that subspaces L_1 and L_s are orthogonal. Let $\xi \in L_i, \eta \in L_j$, and $\eta = \eta_1 + \ldots + \eta_{d_s}$, where each $\eta_k = [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, \underbrace{\dot{U}}_{k\text{-th}}, \ldots, U]\!]$. We show that ξ and η_k are orthogonal for each $k = 1, \ldots, d_s$, which implies orthogonality of ξ and η:

$$\langle \xi, \eta_k \rangle = \left\langle \left[\!\!\left[\mathcal{G}; \dot{V}^{(1)}, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, \dot{U}, \ldots, U\right]\!\!\right] \right\rangle =$$
$$= \left\langle \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, I, \ldots, U\right]\!\!\right], \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, \dot{U}, \ldots, \underbrace{\dot{U}^T U}_{0}\right]\!\!\right] \right\rangle = 0.$$

□

Now we are ready to derive formulas for projection onto the tangent space from Section 4.

Proposition B.2. *For any* $\mathcal{Y} \in \mathbb{R}^{n_1 \times \ldots \times n_{d_t} \times n \times \ldots \times n}$*, the orthogonal projection onto the tangent space of* $\mathcal{M}^s_{\mathbf{r}}$ *at* $\mathcal{X} = [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U]\!] \in \mathcal{M}^s_{\mathbf{r}}$ *is given by*

$$P_{\mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}} \colon \mathbb{R}^{n_1 \times \ldots \times n_{d_t} \times n \times \ldots \times n} \to \mathrm{T}_{\mathcal{X}}\mathcal{M}^s_{\mathbf{r}}, \quad \mathcal{Y} \mapsto \left[\!\!\left[\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right); W^{(1)}, \ldots, W^{(d_t)}, Y, \ldots, Y\right]\!\!\right]$$
$$\left[\!\!\left[\mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right); W^{(1)}, \ldots, W^{(d_t)}, Y, \ldots, Y\right]\!\!\right] = \left[\!\!\left[\mathcal{G}; \dot{V}^{(1)}, V^{(2)}, \ldots, U\right]\!\!\right] + \ldots + \left[\!\!\left[\mathcal{G}; V^{(1)}, \ldots, U, \dot{U}\right]\!\!\right] + \left[\!\!\left[\dot{\mathcal{G}}; V^{(1)}, \ldots, U\right]\!\!\right],$$

where the components $\dot{\mathcal{G}}, \dot{V}^{(k)}$ and $\dot{U}$ are determined by

$$\dot{\mathcal{G}} = \left[\!\left[\mathcal{Y}; {V^{(1)}}^T, \ldots, U^T\right]\!\right],$$

$$\dot{V}^{(i)} = \left(I - V^{(i)}{V^{(i)}}^T\right) \left[\!\left[\mathcal{Y}; {V^{(1)}}^T, \ldots, \underbrace{I}_{i\text{-}th}, \ldots, V^{(d_t)}, U^T, \ldots, U^T\right]\!\right]_{(i)} \mathcal{G}_{(i)}^{+},$$

$$\dot{U} = \left(I - UU^T\right) \left(\sum_{i=d_t+1}^{d} \left[\!\left[\mathcal{Y}; {V^{(1)}}^T, \ldots, {V^{(d_t)}}^T, U^T, \ldots, \underbrace{I}_{i\text{-}th}, \ldots, U^T\right]\!\right]_{(i)} \mathcal{G}_{(i)}^{T}\right) \left(\sum_{i=d_t+1}^{d} \mathcal{G}_{(i)}\mathcal{G}_{(i)}^{T}\right)^{-1},$$

where $\mathcal{G}_{(i)}^{+} = \mathcal{G}_{(i)}^{T}\left(\mathcal{G}_{(i)}\mathcal{G}_{(i)}^{T}\right)^{-1}$ — Moore-Penrose pseudo-inverse of $\mathcal{G}_{(i)}$.

Proof. Since $\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^{s} = L_0 \oplus L_1 \oplus \ldots \oplus L_{d_t} \oplus L_s$, it follows, that there exists a set of unique $\eta_k, k \in \{0, 1, \ldots, d_t, s\}$, such that $P_{\mathrm{T}_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}^{s}}(\mathcal{Y}) = \eta_0 + \eta_1 + \ldots + \eta_{d_t} + \eta_s$. Now, let P_{L_k} — orthoprojector onto L_k. Then, η_k may be obtained as $P_{L_k}(\mathcal{Y})$. Therefore, we can consider each P_{L_k} separately and then combine them to construct the projection onto the entire tangent space.

The formulas for $P_{L_k}, k \in \{0, 1, \ldots, d_t\}$ coincide with the classic Tucker decomposition (Steinlechner, 2016). We therefore omit them and consider only derive the projection formula for L_s. As an orthoprojector, P_{L_s} has to satisfy

$$\langle P_{L_s}(\mathcal{Y}), \xi_k \rangle = \langle \mathcal{Y}, \xi_s \rangle \tag{11}$$

for any $\xi_s \in L_s$. Let $P_{L_s}(\mathcal{Y}) = \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, \dot{U}, U, \ldots, U\right]\!\right] + \ldots + \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U, \dot{U}\right]\!\right]$. We introduce the projector $P_U^{\perp} = I - UU^T$ onto the orthogonal complement of the image of U. Using $P_U^{\perp}$ we can represent $\xi_s \in L_s$ as $[\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, P_U^{\perp}W, U, \ldots, U]\!] + \ldots + [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U, P_U^{\perp}W]\!]$, as $P_U^{\perp}$ preserves gauge conditions of shared factor for any arbitrary matrix $W \in \mathbb{R}^{n_i \times r_i}$. Utilizing this representation of ξ_s we determine $\dot{U}$ such that (11) holds:

$$\langle P_{L_s}(\mathcal{Y}), \xi_k \rangle = \left\langle P_{L_s}(\mathcal{Y}), \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, P_U^{\perp}W, U, \ldots, U\right]\!\right] + \ldots + \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U, P_U^{\perp}W\right]\!\right]\right\rangle.$$

Using linearity of the scalar product by the second argument, we rewrite the expression as

$$\langle P_{L_s}(\mathcal{Y}), \xi_k \rangle = \sum_{i=1}^{d_s} \left\langle P_{L_s}(\mathcal{Y}), \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, P_U^{\perp}W, \ldots, U\right]\!\right]\right\rangle.$$

Now we utilize the fact, that if U and $P_U^{\perp}W$ are at different positions then the scalar products of corresponding terms is 0. For example:

$$\begin{aligned}&\left\langle \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, \dot{U}, U, \ldots, U\right]\!\right], \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, P_U^{\perp}W, \ldots, U\right]\!\right]\right\rangle \\ &= \left\langle \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, \dot{U}, P_U^{\perp}U, \ldots, U\right]\!\right], \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, W, \ldots, U\right]\!\right]\right\rangle = 0,\end{aligned}$$

as $\left(P_U^{\perp}\right)^T = P_U^{\perp}$ and $P_U^{\perp}U = 0$.

Therefore, the expression is further simplified as

$$\langle P_{L_s}(\mathcal{Y}), \xi_k \rangle = \sum_{i=1}^{d_s} \left\langle \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, \dot{U}, \ldots, U\right]\!\right], \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, P_U^{\perp}W, \ldots, U\right]\!\right]\right\rangle,$$

where in the k-th term $\dot{U}$ and $\pi^{\perp}W$ stand on the same position of the $d_t + k$-th mode. We can further drop all other factors (both non-shared and shared) due to orthogonality of their columns:

$$\langle P_{L_s}(\mathcal{Y}), \xi_k \rangle = \sum_{i=1}^{d_s} \left\langle \left[\!\left[\mathcal{G}; I, \ldots, \dot{U}, \ldots, I\right]\!\right], \left[\!\left[\mathcal{G}; I, \ldots, P_U^{\perp}W, \ldots, I\right]\!\right]\right\rangle.$$

Now we rewrite this expression in terms of unfolding matrices:

$$\sum_{i=1}^{d_s} \left\langle \left[\!\left[\mathcal{G}; I, \ldots, \dot{U}, \ldots, I \right]\!\right], \left[\!\left[\mathcal{G}; I, \ldots, P_U^{\perp} W, \ldots, I \right]\!\right] \right\rangle = \sum_{i=d_t+1}^{d} \left\langle \dot{U}\mathcal{G}_{(i)}, P_U^{\perp} W \mathcal{G}_{(i)} \right\rangle = \left\langle \sum_{i=d_t+1}^{d} \dot{U}\mathcal{G}_{(i)}\mathcal{G}_{(i)}^T, W \right\rangle,$$

as $P_U^{\perp}\dot{U} = \dot{U}$. On the other hand, we know that $\langle P_{L_s}(\mathcal{Y}), \xi_k\rangle = \langle \mathcal{Y}, \xi_s\rangle$. Considering the expression for $\langle \mathcal{Y}, \xi_s\rangle$, we obtain

$$\langle \mathcal{Y}, \xi_s\rangle = \sum_{i=1}^{d_s} \left\langle \mathcal{Y}, \left[\!\left[\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, P_U^{\perp} W, \ldots, U \right]\!\right] \right\rangle,$$

which we rephrase in terms of unflodings:

$$\begin{aligned}\langle \mathcal{Y}, \xi_s\rangle &= \sum_{i=d_t+1}^{d_s} \left\langle \mathcal{Y}_{(i)}, P_U^{\perp} W \mathcal{G}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right)^T \right\rangle = \\ &= \sum_{i=d_t+1}^{d_s} \left\langle P_U^{\perp} \mathcal{Y}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right) \mathcal{G}_{(i)}^T, W \right\rangle.\end{aligned}$$

Finally, combining both expressions we obtain the following sequence of equalities:

$$\begin{aligned}&\langle P_{L_s}(\mathcal{Y}), \xi_k\rangle = \langle \mathcal{Y}, \xi_s\rangle \\ &\left\langle \sum_{i=d_t+1}^{d} \dot{U}\mathcal{G}_{(i)}\mathcal{G}_{(i)}^T, W \right\rangle = \sum_{i=d_t+1}^{d_s} \left\langle P_U^{\perp} \mathcal{Y}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right) \mathcal{G}_{(i)}^T, W \right\rangle \\ &\sum_{i=d_t+1}^{d} \dot{U}\mathcal{G}_{(i)}\mathcal{G}_{(i)}^T = \sum_{i=d_t+1}^{d_s} P_U^{\perp} \mathcal{Y}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right) \mathcal{G}_{(i)}^T \\ &\dot{U}\left(\sum_{i=d_t+1}^{d} \mathcal{G}_{(i)}\mathcal{G}_{(i)}^T\right) = P_U^{\perp}\left(\sum_{i=d_t+1}^{d_s} \mathcal{Y}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right) \mathcal{G}_{(i)}^T\right) \\ &\dot{U} = \left(I - UU^T\right)\left(\sum_{i=d_t+1}^{d_s} \mathcal{Y}_{(i)} \left(U \otimes \ldots \otimes U \otimes V^{(d_t)} \otimes \ldots \otimes V^{(1)}\right) \mathcal{G}_{(i)}^T\right)\left(\sum_{i=d_t+1}^{d} \mathcal{G}_{(i)}\mathcal{G}_{(i)}^T\right)^{-1}.\end{aligned}$$

Finally, using square brackets we write:

$$\dot{U} = \left(I - UU^T\right)\left(\sum_{i=d_t+1}^{d} \left[\!\left[\mathcal{Y}; {V^{(1)}}^T, \ldots, {V^{(d_t)}}^T, U^T, \ldots, \underbrace{I}_{i\text{-th}}, \ldots, U^T \right]\!\right]_{(i)} \mathcal{G}_{(i)}^T\right)\left(\sum_{i=d_t+1}^{d} \mathcal{G}_{(i)}\mathcal{G}_{(i)}^T\right)^{-1}.$$

□

B.3 Computing retraction

Here we discuss the deriviation of the Algorithm 1. Suppose, we are given a tensor $\mathcal{X}$ of the SF-Tucker form $\left[\!\left[\bar{\mathcal{G}}; \bar{V}^{(1)}, \ldots, \bar{V}^{(d_t)}, \bar{U}, \ldots, \bar{U}\right]\!\right]$ after applying the reorthogonalization procedure. We are also given the desired SFT rank $\mathbf{r} = (r_1, \ldots, r_{d_t}, r_s)$. Additionally, we suppose that $\bar{V}^{(k)}$ has $\bar{r}_k$ columns, $k = 1, \ldots, d_t$; $\bar{U}$ has $\bar{r}_s$ columns, and $\bar{\mathcal{G}}$ has size $\bar{r}_1 \times \ldots \times \bar{r}_{d_t} \times \bar{r}_s \times \ldots \times \bar{r}_s$.

The algorithm of rank truncation can be summarized as follows: 1) Utilize the SF-HOSVD algorithm to obtain a valid SF-Tucker decomposition; 2) Perform rank truncation by retaining the first r_k columns from the respective factors, and also keeping the corresponding "corner" subtensor of the size $r_1 \times \ldots \times r_{d_t} \times r_s \times \ldots \times r_s$ from the core.

Next we discuss how to perform these steps efficiently. Firstly, we need to obtain a valid SF-Tucker decomposition of $\mathcal{X}$. Following the algorithm of SF-HOSVD quasioptimality (Theorem 3.2), we should obtain factors $V^{(k)}$ which are r_k left singular vectors of $\mathcal{X}_{(k)}, k = 1, \ldots, d_t$. Considering,

$$\mathcal{X}_{(k)} = \bar{V}^{(k)}\bar{\mathcal{G}}_{(k)}\left(\bar{U} \otimes \ldots \otimes \bar{U} \otimes \bar{V}^{(d_t)} \otimes \ldots \otimes \bar{V}^{(1)}\right)^T$$

we note that taking thin SVD of $\bar{\mathcal{G}}_{(k)}$ leads us to thin SVD of $\mathcal{X}_{(k)}$ since other matrices are already orthogonal. Therefore, let $Y\Sigma W^T, Y \in \mathbb{R}^{\bar{r}_k \times \bar{r}_k}, \Sigma \in \mathbb{R}^{\bar{r}_k \times \bar{r}_k}, W \in \mathbb{R}^{\bar{r}_k \times (\bar{r}_1 \cdot \ldots \cdot \bar{r}_d)/\bar{r}_k}$ be thin SVD of $\bar{\mathcal{G}}_{(k)}$. Then we construct $V^{(k)}$ as $\bar{V}^{(k)} Y[:, : r_k]$, where $Y[:, : r_k]$ stands for first r_k columns of Y.

We also need to obtain a new factor U which is the matrix of first r_s left singular vectors of the concatenation of unfolding matrices $\left(\mathcal{X}_{(d_t+1)} \mid \ldots \mid \mathcal{X}_{(d)}\right)$. Considering the expression for α-th unfolding of $\mathcal{X}$ ($\alpha = d_t + 1, \ldots, d$)

$$\mathcal{X}_{(\alpha)} = \bar{U} \bar{\mathcal{G}}_{(\alpha)} \underbrace{\left(\bar{U} \otimes \ldots \otimes \bar{U} \otimes \bar{V}^{(d_t)} \otimes \ldots \otimes \bar{V}^{(1)}\right)^T}_{\Omega},$$

we note, that only the middle matrix changes depending on α, while the other matrices remain fixed. Utilizing this fact, we acquire a concatenation matrix in the following form:

$$\left(\mathcal{X}_{(d_t+1)} \mid \ldots \mid \mathcal{X}_{(d)}\right) = \left(\bar{U} \mid \ldots \mid \bar{U}\right) \operatorname{Diag}\left(\mathcal{G}_{(d_t+1)}, \ldots, \mathcal{G}_{(d)}\right) \operatorname{Diag}\left(\Omega, \ldots, \Omega\right)^T,$$

which can be simplified as

$$\left(\mathcal{X}_{(d_t+1)} \mid \ldots \mid \mathcal{X}_{(d)}\right) = \bar{U} \left(\mathcal{G}_{(d_t+1)} \mid \ldots \mid \mathcal{G}_{(d)}\right) \operatorname{Diag}\left(\Omega, \ldots, \Omega\right)^T.$$

Here we denote block-diagonal matrix with blocks $A_1, \ldots, A_m$ on the diagonal as $\operatorname{Diag}(A_1, \ldots, A_m)$. Analogously to the previous case, the matrices $\bar{U}$ and $\operatorname{Diag}(\Omega, \ldots, \Omega)^T$ (which is $(I_{d_s} \otimes \Omega)^T$) are already orthogonal. Thus, we only need to take thin SVD of $\left(\mathcal{G}_{(d_t+1)} \mid \ldots \mid \mathcal{G}_{(d)}\right)$ and to obtain the factor U.

Finally, for the obtained factors $V^{(k)}, k = 1, \ldots, d_t$ and U we should derive new core tensor $\mathcal{G}$ by performing the following contraction:

$$\mathcal{G} = \left[\!\left[\mathcal{X}; {V^{(1)}}^T, \ldots, {V^{(d_t)}}^T, U^T, \ldots, U^T\right]\!\right] = \left[\!\left[\bar{\mathcal{G}}; {V^{(1)}}^T \bar{V}^{(1)}, \ldots, {V^{(d_t)}}^T \bar{V}^{(d_t)}, U^T \bar{U}, \ldots, U^T \bar{U}\right]\!\right].$$

C Riemannian autodiff

Let us introduce new notation for this section. In the cases where we perform tensor-matrix multiplication across all modes except one, we use the following notation:

$$\mathcal{X} \bigtimes_{\substack{k=1 \\ k \neq j}}^{d} V^{(k)}.$$

If $\mathcal{X}$ admits the SF-Tucker decomposition $\mathcal{X} = [\![\mathcal{G}; V^{(1)}, \ldots, V^{(d_t)}, U, \ldots, U]\!]$, then we also write it as

$$\mathcal{X} = \mathcal{G} \bigtimes_{i=1}^{d_t} V^{(i)} \bigtimes_{j=d_t+1}^{d} U.$$

In the following derivation, we will also require an elementwise representation of the SF-Tucker:

$$\mathcal{X}_{i_1, \ldots, i_d} = \sum_{j_1, \ldots, j_d = 1}^{r_1, \ldots, r_{d_t}, r_s, \ldots, r_s} \mathcal{G}_{j_1, \ldots, j_d} V^{(1)}_{i_1 j_1} \ldots V^{(d_t)}_{i_{d_t} j_{d_t}} U_{i_{d_t+1} j_{d_t+1}} \ldots U_{i_d j_d}. \tag{12}$$

We are given a mapping from Section 5:

$$\mathcal{T}_{\mathcal{X}} \colon \left(S, B^{(1)}, \ldots, B^{(d_t)}, A\right) \mapsto \mathcal{F}\left(\mathcal{G}, S\right) \bigtimes_{i=1}^{d_t} \left(V^{(i)} \mid B^{(i)}\right) \bigtimes_{j=d_t+1}^{d} \left(U \mid A\right),$$

and the corresponding mapping $h = f \circ \mathcal{T}_{\mathcal{X}}$. One may note that $h(\mathcal{G}, 0, \ldots, 0) = f(\mathcal{X})$. Thus, we will construct the Riemannian gradient of f at $\mathcal{X}$ by differentiating h with respect to its arguments. Let $T = \mathcal{T}_{\mathcal{X}}\left(S, B^{(1)}, \ldots, B^{(d_t)}, A\right)$. Then its partial derivative with respect to S is

$$\frac{\partial T_{i_1, \ldots i_d}}{\partial S_{q_1, \ldots, q_d}} = \frac{\partial}{\partial S_{q_1, \ldots, q_d}} \left(S \bigtimes_{\alpha=1}^{d_t} V^{(\alpha)} \bigtimes_{\beta=d_t+1}^{d} U\right)_{i_1, \ldots i_d}.$$

Applying (12) and taking partial derivatives we obtain

$$\frac{\partial T_{i_1,\dots i_d}}{\partial S_{q_1,\dots,q_d}} = V^{(1)}_{i_1 q_1},\dots,V^{(d_t)}_{i_{d_t} q_{d_t}} U_{i_{d_t+1} q_{d_t+1}} \dots U_{i_d q_d}.$$

Therefore, the partial derivative of h at $(\mathcal{G}, 0, \dots, 0)$ is

$$\left.\frac{\partial h}{\partial S_{q_1,\dots,q_d}}\right|_{\substack{S=\mathcal{G}\\ A=0\\ B_i=0}} = \sum_{i_1,\dots,i_d} \left.\frac{\partial f}{\partial T_{i_1,\dots i_d}}\right|_{T=\mathcal{X}} \frac{\partial T_{i_1,\dots i_d}}{\partial S_{q_1,\dots,q_d}} = \sum_{i_1,\dots,i_d} \frac{\partial f}{\partial \mathcal{X}_{i_1,\dots i_d}} V^{(1)}_{i_1 q_1},\dots,V^{(d_t)}_{i_{d_t} q_{d_t}} U_{i_{d_t+1} q_{d_t+1}} \dots U_{i_d q_d},$$

hence,

$$\frac{\partial h}{\partial S} = \nabla f(X) \bigtimes_{i=1}^{d_t} V^{(i)^T} \bigtimes_{j=d_t+1}^{d} U^T$$

To obtain the partial derivative of h with respect to $B^{(i)}$ we firstly again need to take the derivative of T:

$$\frac{\partial T_{i_1,\dots,i_d}}{\partial B^{(j)}_{pq}} = \frac{\partial}{\partial B^{(j)}_{pq}} \left(\sum_{\alpha=1}^{d_t} G \times_\alpha B^{(\alpha)} \bigtimes_{\substack{\beta=1\\ \beta\neq\alpha}}^{d_t} V^{(\beta)} \bigtimes_{d_t+1}^{d} U \right)_{i_1,\dots,i_d}.$$

Applying (12) we obtain

$$\frac{\partial}{\partial B^{(j)}_{pq}} \left(\sum_{\alpha=1}^{d_t} \sum_{\gamma_1,\dots,\gamma_d} \mathcal{G}_{\gamma_1,\dots,\gamma_d} B^{(\alpha)}_{i_\alpha \gamma_\alpha} \prod_{\substack{\beta=1\\ \beta\neq\alpha}}^{d_t} V^{(\beta)}_{i_\beta \gamma_\beta} \prod_{\xi=d_t+1}^{d} U_{i_\xi \gamma_\xi} \right) =$$

$$= \delta^p_{i_j} \sum_{\gamma_1,\dots,\gamma_{j-1},\gamma_{j+1},\dots,\gamma_d} \mathcal{G}_{\gamma_1,\dots,\gamma_{j-1},q,\gamma_{j+1},\dots,\gamma_d} \prod_{\substack{\beta=1\\ \beta\neq\alpha}}^{d_t} V^{(\beta)}_{i_\beta \gamma_\beta} \prod_{\xi=d_t+1}^{d} U_{i_\xi \gamma_\xi},$$

where δ^b_a is the Kronecker delta, is equal 1 if $a = b$ and 0 otherwise. Now, deriving the partial derivative of h at $(\mathcal{G}, 0, \dots, 0)$:

$$\left.\frac{\partial h}{\partial B^{(j)}_{pq}}\right|_{\substack{S=\mathcal{G}\\ A=0\\ B_i=0}} = \sum_{i_1,\dots,i_d} \left.\frac{\partial f}{\partial T_{i_1,\dots i_d}}\right|_{T=\mathcal{X}} \frac{\partial T_{i_1,\dots i_d}}{\partial S_{q_1,\dots,q_d}} =$$

$$= \sum_{\{i_1,\dots,i_d\}/\{i_j\}} \nabla f(\mathcal{X})_{i_1,\dots p \dots, i_d} \sum_{\{\gamma_1,\dots,\gamma_d\}/\{\gamma_j\}} \mathcal{G}_{\gamma_1,\dots,q,\dots,\gamma_d} \prod_{\substack{\beta=1\\ \beta\neq\alpha}}^{d_t} V^{(\beta)}_{i_\beta \gamma_\beta} \prod_{\xi=d_t+1}^{d} U_{i_\xi \gamma_\xi} =$$

$$= \sum_{\{\gamma_1,\dots,\gamma_d\}/\{\gamma_j\}} \mathcal{G}_{\gamma_1,\dots,q,\dots,\gamma_d} \left(\nabla f(\mathcal{X}) \bigtimes_{\substack{\beta=1\\ \beta\neq j}}^{d_t} V^{(\beta)^T} \bigtimes_{\xi=d_t+1}^{d} U^T \right)_{\gamma_1,\dots,p,\dots,\gamma_d},$$

hence,

$$\frac{\partial h}{\partial B^{(j)}} = \left[\nabla f(\mathcal{X}) \bigtimes_{\substack{\beta=1\\ \beta\neq j}}^{d_t} V^{(\beta)^T} \bigtimes_{\xi=d_t+1}^{d} U^T \right]_{(j)} \mathcal{G}^T_{(j)},$$

where $\sum_{\{i_1,\dots,i_d\}/\{i_j\}}$ means there is summation over all indices from i_1 to i_d except i_j. By a similar procedure, we obtain the partial derivative of T with respect to A:

$$\frac{\partial T_{i_1,\dots,i_d}}{\partial A_{pq}} = \sum_{\alpha=d_t+1}^{d} \delta^p_{i_\alpha} \sum_{\{\gamma_1,\dots,\gamma_d\}/\{\gamma_j\}} \mathcal{G}_{\gamma_1,\dots,q,\dots,\gamma_d} \prod_{\beta=1}^{d_t} V^{(\beta)}_{i_\beta \gamma_\beta} \prod_{\substack{\xi=1\\ \xi\neq\alpha}} U_{i_\xi \gamma_\xi}.$$

Table 4: Extended results of experiments on the link prediction problem on FB15k-237 and WN18RR datasets. The best scores for each dataset are highlighted in bold font. Top-2 scores are underlined.

Model	Rank	MRR	Hits@10	Hits@3	Hits@1
		WNRR18			
R-TuckER$_{\text{no BN}}$	$(10, 200)$	**47.9** $\pm .2$	**54.9** $\pm .2$	**49.2** $\pm .3$	**44.6** $\pm .1$
R-TuckER$_{\text{no BN}}$	$(10, 200, 200)$	44.0 $\pm .4$	47.6 $\pm .5$	45.1 $\pm .3$	41.7 $\pm .2$
TuckER$_{\text{no BN}}$	$(30, 200)$	45.0 $\pm .3$	46.1 $\pm .4$	46.7 $\pm .3$	42.4 $\pm .2$
TuckER	$(30, 200)$	$\underline{47.0}$	$\underline{52.6}$	$\underline{48.2}$	$\underline{44.3}$
ConvE	–	43.0	52.0	44.0	40.0
DistMult	–	43.0	49.0	44.0	39.0
		FB15k-237			
R-TuckER$_{\text{no BN}}$	$(200, 40)$	$\underline{32.9}$ $\pm .2$	$\underline{50.5}$ $\pm .2$	$\underline{35.9}$ $\pm .2$	$\underline{24.2}$ $\pm .1$
R-TuckER$_{\text{no BN}}$	$(200, 40, 40)$	31.2 $\pm .4$	47.6 $\pm .4$	33.9 $\pm .3$	23.1 $\pm .2$
R-TuckER$_{\text{no BN}}$	$(200, 200)$	27.8 $\pm .4$	44.3 $\pm .3$	30.5 $\pm .3$	19.3 $\pm .3$
TuckER$_{\text{no BN}}$	$(200, 200)$	32.6 $\pm .4$	50.5 $\pm .4$	35.3 $\pm .2$	23.7 $\pm .2$
TuckER	$(200, 200)$	**35.8**	**54.4**	39.4	**26.6**
R-GCN	–	24.8	41.7	26.4	15.1
ConvE	–	32.5	50.1	35.6	23.7
DistMult	–	24.1	41.9	26.3	15.5

Table 5: Best performing hyperparameter values for R-TuckER across both datasets.

Dataset	lr	dr	reg_init	reg_finish	reg_steps	beta	ls	num_epochs
WN18RR	2000	.9981	$1e-4$	$1e-9$	350	0.8	0.1	1450
FB15k-237	2000	.9981	$1e-4$	$1e-10$	100	0.8	0.1	500

The partial derivative of h at $(\mathcal{G}, 0, \ldots, 0)$:

$$\frac{\partial h}{\partial A} = \sum_{\alpha_{d_t+1}=1}^{d} \left(\nabla f(\mathcal{X}) \bigtimes_{\beta=1}^{d_t} {V^{(\beta)}}^T \bigtimes_{\substack{\xi=1 \\ \xi\neq\alpha}}^{d} U^T \right)_{(\alpha)} \mathcal{G}_{(\alpha)}^T.$$

Finally, we construct the Riemannian gradient:

$$\nabla_{\mathcal{M}_{\mathbf{r}}^k} f(\mathcal{X}) = \mathcal{F}\left(\mathcal{G}, \dot{\mathcal{G}}\right) \bigtimes_{i=1}^{d_t} \left(V^{(i)} \mid \dot{V}^{(i)}\right) \bigtimes_{j=d_t+1}^{d} \left(U^{(i)} \mid \dot{U}^{(i)}\right),$$

$$\dot{\mathcal{G}} = \frac{\partial h}{\partial S}, \quad \dot{V}_i = \left(I_{n_i} - V^{(i)}{V^{(i)}}^T\right) \frac{\partial h}{\partial B^{(i)}} \left(\mathcal{G}_{(i)} \mathcal{G}_{(i)}^T\right)^{-1}, \quad \dot{U} = \left(I_{n_d} - UU^T\right) \frac{\partial h}{\partial A} \left(\sum_{\gamma=d_t+1}^{d} \mathcal{G}_{(\gamma)} \mathcal{G}_{(\gamma)}^T\right)^{-1}.$$

D More experiments for link prediction

Here we provide more detailed results of experiments for the link prediction problem. In particular, we add hits@3 evaluation metric for both datasets and include experiments with more configurations for both datasets.

Table 4 includes the experiments from Table 1 in Section 6.2, along with an additional experiment for the WN18RR dataset and two additional experiments for the FB15k-237 dataset. In these experiments, we did not use shared embeddings for subjects and objects. This means that we essentially utilize standard Tucker decomposition instead of SF-Tucker. These models were trained using the same hyperparameters and optimization

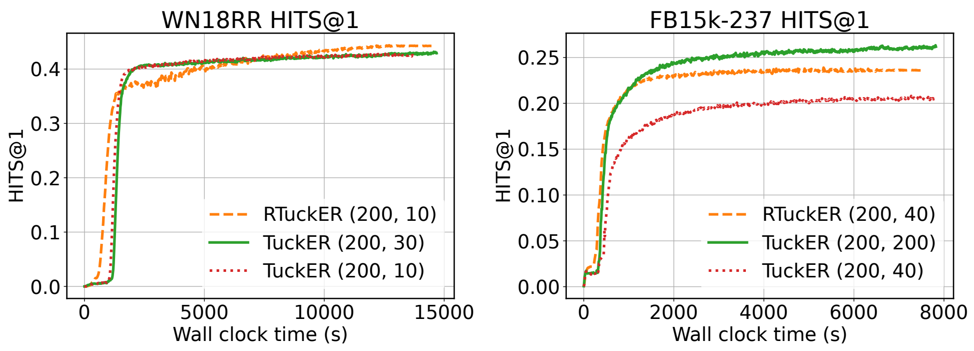

Figure 3: Comparing our R-TuckER model to the original TuckER model and TuckER model with a comparable number of parameters based on the Hits@1 metric, against the training time.

method as the models with shared factors. However, we observed that despite having a noticeably larger number of parameters, these models performed worse than the models with shared factors across all the metrics. We do not report the standard deviation for vanilla TuckER with or for Deep Learning-based model, as we take the best published results from the corresponding papers. In Figure 3, we also compare time performance of our R-TuckER model with the original TuckER in two settings: with the same and higher rank values. We provide Hits@1 metric values against wall clock time during training. It is clear, that our model achieves better metric value within comparable amount of run time on WN18RR dataset. In the case of FB15k-237 dataset, we outperform the TuckER model on the same rank value, but obtain worse metrics than TuckER with the larger rank value. As a potential direction for research it would be interesting to try different regularization strategies to avoid overfitting in regimes with large ranks on FB15k-237 dataset.

Table 5 shows best performing hyperparameters for R-TuckER across both datasets, where lr denotes learning rate, dr — exponential decay rate, reg_init and reg_finish are initial and final values of regularization coefficient, reg_steps is amount of regularization coefficient decay steps, beta is momentum beta parameter, ls is label smoothing and num_epochs is number of epochs that model was trained.