Skoltech

Skolkovo Institute of Science and Technology

Geometrical Methods in Machine Learning and Tensor Analysis

Doctoral Thesis
by

Valentin Khrulkov

Doctoral Program in Computational and Data Science and Engineering

Supervisor
Full Professor, Ivan Oseledets

Moscow - 2020

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Candidate (Valentin Khrulkov)

Supervisor (Prof. Ivan Oseledets)

Abstract

In machine learning and optimization tasks, it is common for data to have an underlying geometrical structure, usually realized in the form of a low-dimensional underlying manifold or specific inductive biases, e.g., presumed hierarchical nature of data. Utilization of this structure often leads to performance improvement or allows one to draw new insights and design better algorithms. This thesis is built upon a series of papers devoted to theoretical and practical results in deep learning and numerical optimization achieved via the application of ideas from such fields as algebraic and differential geometry, tensor analysis, and hyperbolic geometry. Concretely, we utilize such tools as Riemannian optimization, desingularization of singular manifolds, topological data analysis, persistent homology, Gromov δ-hyperbolicity, and several others. We start with a new optimization algorithm on matrix manifolds, allowing one to deal with a challenging problem of singular points and curvature blow up. Then we move to tensor manifolds and discuss intriguing connections of the geometry of these manifolds with theoretical properties of recurrent neural networks (RNNs). We extend these results to the class of generalized tensor decompositions and RNNs with rectifier nonlinearity. For practical applications, we show how universal adversarial perturbations for neural networks can be designed employing matrix analysis. We propose a new way to estimate the quality of generative models by comparing the topological properties of the underlying data manifold and generated manifolds. We introduce hyperbolic geometry to the computer vision area and evaluate our ideas on the few-shot learning tasks.

Publications

Main contribution (theory and experiment designs) by the author under the supervision of I. Oseledets

1. **Generalized Tensor Models for Recurrent Neural Networks**
   V. Khrulkov, O. Hrinchuk, and I. Oseledets.
   International Conference on Learning Representations (ICLR) 2019
   h5 index = 150

2. **Geometry Score: A Method For Comparing Generative Adversarial Networks**
   V. Khrulkov and I. Oseledets.
   International Conference on Machine Learning (ICML) 2018
   CORE rating = A*

3. **Expressive power of recurrent neural networks**
   V. Khrulkov, A. Novikov, and I. Oseledets.
   International Conference on Learning Representations (ICLR) 2018
   h5 index = 150

4. **Desingularization of bounded-rank matrix sets**
   V. Khrulkov and I. Oseledets.
   SIAM Journal on Matrix Analysis and Applications (SIMAX) 2018; Q1

Co-authored papers with significant contribution by the author

1. **Hyperbolic Image Embeddings**
   V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky.
   Computer Vision and Pattern Recognition (CVPR) 2020 (equal contribution of the first two authors); CORE rating = A*
   Contributions:
   - Implementation of hyperbolic prototypical networks;
   - *Mini*ImageNet few-shot experiment design and experiments;

- Evaluated uncertainty estimation via distance to the origin for hyperbolic networks;
- Contributed to paper writing and visualizations.

Contents

Chapter 1

Introduction

1.1 Theoretical analysis of neural networks

1.1.1 Depth efficiency of neural networks

One of the biggest challenges in modern deep learning is achieving a better understanding of the theory underlying many empirically observed phenomena. A particularly important open problem is achieving a better understanding of *universality* and *expressivity* of neural networks. Classical works [Cybenko, 1989, Hornik et al., 1989] demonstrated that neural networks are *universal approximators*, i.e., informally, they can approximate any given function with arbitrary precision. These results, however, are not practical, as the constructed networks are shallow — they have only one hidden layer. On the other hand, there is mounting empirical evidence that for a given budget of resources (e.g., neurons), the deeper one goes, the better the eventual performance will be. It is widely believed that recent progress in many fields, e.g., Computer Vision (CV) and Natural Language Processing (NLP), in large part, can be attributed to the increase of the depth of networks [He et al., 2016, Dai et al., 2019, Vaswani et al., 2017, Simonyan and Zisserman, 2014]. Moreover, there is evidence that the indefinite increase of the depth of a network only improves the test accuracy [Nakkiran et al., 2019], which, at first sight, contradicts the standard bias-variance tradeoff paradigm in classical statistics. Older results such as [Hastad, 1986, Håstad and Goldmann, 1991, Delalleau and Bengio, 2011, Martens and Medabalimi, 2014] only apply to specific types of networks, and not common architectures such as Convolutional Neural Networks (CNNs)[LeCun et al., 1990]. A big step towards a better understanding of depth efficiency of CNNs was made in [Cohen et al., 2016] and further extended in [Cohen and Shashua, 2016]. They addressed the following question: is depth efficiency *typical* in the space of neural networks? This is formalized as follows: given a neural network, let us consider the equivalent *shallow network* (i.e., a neural network of width one, realizing the same function). We say that the original network is exponentially more expressive than the obtained shallow network if the latter has an exponentially large width with respect to the width of the former network. Can we understand how often this is the case? The authors of [Cohen et al., 2016] demonstrated that this property holds for CNNs with multiplicative nonlinearities, besides a set of measure zero. Their analysis is based on the formalism of tensor decompositions, which is one of the key concepts in this thesis.

1.1.2 Tensor Decompositions

In modern deep learning and numerical analysis it is common to work with data *tensors*, i.e., multi-way arrays $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \dots \times I_d}$. As the number of modes d increases, the number of parameters grows exponentially. When the number of possible configurations is huge, much larger than our number of examples, it becomes statistically difficult to say something meaningful. This is also known as the *curse of dimensionality*. Despite the curse of dimensionality, algorithms may be developed based on the assumptions that real data will often be confined to a region of the space having lower effective dimensionality, in the directions over which important variations in variables occur. One of the most appealing approaches for this is based on the apparatus of tensor networks (or tensor decompositions) [Vasilescu and Terzopoulos, 2002, 2003, Cichocki et al., 2016, 2017, Orús, 2014]. Informally, a tensor network allows one to efficiently represent a tensor by a network of smaller building blocks (often) arranged based on a pre-specified tree-like pattern. Some examples are CANDECOMP/PARAFAC (CP) decomposition [Harshman, 1970, Carroll and Chang, 1970], Tensor Train (TT) decomposition [Oseledets, 2011], Hierarchical Tucker (HT) decomposition [Grasedyck, 2010], Tensor Ring (TR) decomposition [Zhao et al., 2016]. Vasilescu and Kim [2019], Vasilescu et al. [2020] compute a part-based compositional hierarchical data tensor decomposition for arbitrary shapes and sizes that has an architecture that parallels the CNN architecture. Importantly, spaces of tensor networks often (in the case of general HT decomposition) form *algebraic sets*. These are sets that are specified by polynomial equations and can be well studied using the apparatus of Algebraic Geometry (AG) [Hartshorne, 2013, Shafarevich and Hirsch, 1994].

In Cohen et al. [2016] the authors found a link between the (binary) HT decomposition and CNNs, while CP decomposition corresponded to shallow nets. They showed that such a CNN can only be represented by an **exponentially** wide shallow network. This means that a tensor represented in the HT format with probability one has an exponentially high CP rank. This analysis was also extended to CNNs with Rectified Linear Unit (ReLU) nonlinearities in [Cohen and Shashua, 2016]. In this case, the expressivity result *not always* holds; there exists an open set of CNN's equivalent to rank one shallow network.

The first part of this thesis is devoted to understanding whether similar results hold for Recurrent Neural Networks (RNNs)[Rumelhart et al., 1986]. We explore connections between multiplicative and generalized RNNs and TT decomposition in Chapter 2 and Chapter 3, respectively.

1.2 Practical applications of geometrical ideas

1.2.1 Riemannian optimization

Suppose that we are given an optimization task, where the variable belongs to a manifold. In order to capitalize on this knowledge, the apparatus of Riemannian optimization was developed [Absil et al., 2009, Uschmajew and Vandereycken, 2020] and recently reintroduced in the area of deep learning [Fonarev et al., 2017, Bécigneul and Ganea, 2018]. Traditionally, Riemannian optimization is applied to problems dealing with matrix or tensor variables, such as ordinary or partial differential equations, tensor or matrix completion, tensor, or matrix approximation. In these cases, we assume that the data lies in a subspace that can be approximated by the low-rank matrix structure or the low-rank TT structure for a tensor. For instance, in the task of matrix completion, one attempts to reconstruct the missing data based on given entries, which is an ill-posed task without any assumptions on the data structure. Riemannian optimization methods have been very successful in dealing with these problems [Vandereycken, 2013, Kressner et al., 2014, Lubich et al., 2013]. Another benefit of the usage of matrix/tensor factorized variables is the great reduction in the required memory footprint and computational power [Rakhuba and Oseledets, 2016]. For instance, in the case of TT decomposition, the number of parameters falls down to logarithmic with respect to the number of parameters in the full tensor. In the common cases of low-rank matrix and tensor manifolds, however, exists a certain challenging problem related to the nature of these manifolds. Specifically, these manifolds contain singular points, where the tangent space is not defined, and the curvature term, appearing in second-order optimization methods, tends to infinity. Traditional methods [Vandereycken, 2013] ignore this issue by setting the curvature term to zero, which leads to subpar performance. In Chapter 4 we discuss the way to resolve this issue by utilizing the concept of desingularization from AG.

1.2.2 Generative modeling

In practice, however, the true underlying data manifold is not known. This is the case, for instance, for visual datasets such as ImageNet [Krizhevsky et al., 2012] or CIFAR10 [Krizhevsky and Hinton, 2009]. The task of generative modeling is concerned with the following problem. Given a dataset sampled from some unknown distribution, can we learn a model to generate more samples from the same distribution? Recent progress in this field is mostly based on the rapid development of Generative Adversarial Networks (GANs) [Goodfellow et al., 2014a]. Recent models, e.g., StyleGAN [Karras et al., 2019] or BigGAN [Brock et al., 2019], are able to produce samples of excellent quality. Such models are particularly useful when data is limited, and collecting new samples is costly. E.g., one interesting application of GANs was found in particle physics [Paganini et al., 2018, Chekalina et al., 2019]. One of the biggest challenges in GAN research is estimating quality of the model.

Compared to previous generative models (producing samples of lower visual quality) such as Variational Autoencoders (VAEs) [Kingma and Welling, 2013], GANs have no explicit optimization objective. In order to estimate their quality several metrics were proposed: Inception score [Salimans et al., 2016a], Frechét Inception Distance [Heusel et al., 2017], Kernel Inception Distance [Bińkowski et al., 2018], Multiscale Intrinsic Distance [Tsitsulin et al., 2020]. Typically, such metrics rely on a pre-trained network, e.g., Inception [Szegedy et al., 2015b]. In Chapter 5, we introduce an approach to estimate the quality of a generative model, using the apparatus of Topological Data Analysis (TDA).

1.2.3 Hyperbolic geometry

In certain cases, it is possible to make assumptions on what is the natural geometry of data at hand. In particular, this is the case when we deal with hierarchical data, such as various taxonomies. For instance, the existence of power-law distributions in datasets can often be traced back to hierarchical structures [Ravasz and Barabási, 2003]. The celebrated work [Krioukov et al., 2010] demonstrated that many properties of complex networks could be explained under the hypothesis that hyperbolic space underlies these networks. These networks (graphs) can be characterized by the following two properties:

1. Power-law degree distribution;

2. Strong clustering properties.

Formally, n-dimensional hyperbolic space denoted as $\mathbb{H}^n$ is defined as the homogeneous, simply connected n-dimensional Riemannian manifold of constant negative sectional curvature. The property of constant negative curvature makes it analogous to the ordinary Euclidean sphere (which has constant positive curvature); however, the geometrical properties of the hyperbolic space are very different. The authors demonstrated that these two properties emerge as a simple consequence of the negative curvature of the hyperbolic space.

Hyperbolic geometry was reintroduced to the machine learning community in Nickel and Kiela [2017]. The authors applied it to learning taxonomies (such as

WordNet) in the Poincaré ball model of hyperbolic space and demonstrated the superior quality of hyperbolic embeddings relative to Euclidean ones. This model was later extended to the Lorentz model of hyperbolic geometry [Nickel and Kiela, 2018b]. Recent developments [Ganea et al., 2018, Skopek et al., 2019] demonstrated that it is possible to design completely hyperbolic neural networks using the apparatus of gyrovector spaces and even build hyperbolic VAEs, where the latent distribution is supported in the hyperbolic space. Additional applications were found in Recommender Systems [Tran et al., 2018] and language modeling [Gulcehre et al., 2019]. The results above, however, are limited to models dealing with discrete data. In Chapter 6, we discuss our approach on learning hyperbolic embeddings for various *visual tasks*.

1.3 Summary of papers

1.3.1 Expressive Power of Recurrent Neural Networks

In this paper, we analyze the expressivity properties of RNNs, capitalizing on the machinery developed in Cohen et al. [2016]. We consider RNNs with multiplicative nonlinearity, specifically, the hidden state is updated as follows.

$$\mathbf{h}^{n+1} = \sum_{ij} \mathcal{G}_{ijk} \mathbf{h}_i^n \mathbf{x}_j^n, \tag{1.1}$$

where $\mathbf{h}^n$ is the hidden state at time step n, $\mathbf{x}^n$ is the current input, and $\mathcal{G}_{ijk}$ is a trainable weight tensor. We show their connection to TT decomposition and translate analysis of their expressivity to a statement about the manifold of tensors represented in the TT format. We utilize the fact that this manifold forms an algebraic variety and show that given a random D-way tensor represented in the TT format (with arbitrary ranks), with probability 1, this tensor will have a CP-rank exponential in D. CP-rank of a tensor is defined as the number of terms in the CP decomposition of a tensor. Note that in this case, we consider two different decompositions of the same tensor and find a connection between their complexities. On the language of deep learning, this means that RNNs with multiplicative nonlinearities with probability one is equivalent to exponentially wide *shallow network*, i.e., they are exponentially more expressive. We additionally compare the expressivity of CNNs and RNNs, corresponding to HT and TT decompositions, respectively. Our numerical experiments with these architectures confirm our theoretical findings and demonstrate the superiority of RNNs over the shallow networks in the sense of test accuracy.

1.3.2 Generalized Tensor Models for Recurrent Neural Networks

This work is the follow up of our previous paper, "Expressive power of recurrent neural networks". Networks considered in the latter utilized multiplication as nonlinearity, which, though used in practice [Wu et al., 2016], is not very popular. We extend our analysis to more practical ReLU nonlinearities. This makes the analysis much more intricate since we cannot simply rely on the known results in algebraic geometry (which only deals with polynomials). We utilize so-called *generalized tensor decompositions* which introduce an arbitrary commutative and associative operator $\xi(\cdot,\cdot)$. For the case of $\xi(x,y) = xy$, we get standard tensor decompositions. Following [Cohen and Shashua, 2016] we use the apparatus of *grid tensors* (grid of values). Rather than comparing two functions *exactly*, an RNN and the corresponding shallow network are compared on a large but finite *grid* of points. Our main results are twofold. Firstly, we show that ReLU RNNs are universal, i.e., that can represent any possible function (on a fixed grid of points). Secondly, we show that they are expressive, but only to some extent: even though there exist exponentially expressive RNNs, there also exists an open set of non-expressive RNNs, equivalent to thin shallow networks. Our numerical experiments demonstrate that the effect of inexpressiveness becomes negligible as we increase the depth/width of ReLU RNNs.

1.3.3 Desingularization of Bounded Rank Matrix Sets

The low-rank matrix manifold defined as

$$\mathcal{M}_{\leq r} = \left\{ A \in \mathbb{R}^{n \times m} : \operatorname{rank}(A) \leq r \right\},$$

often appears in practical tasks, such as matrix completion or recommender systems [Vandereycken, 2013]. An appealing tool for solving optimization problems on this manifold is the framework of Riemannian optimization [Vandereycken, 2013, Absil et al., 2009], which allows to efficiently capitalize on the available manifold data, such as the tangent space. In order to speed up the convergence, it is common to utilize second-order methods (e.g., Newton method on manifolds). However, when dealing with the low-rank matrix manifold, second-order Riemannian optimization methods suffer from the so-called curvature blow up. Specifically, the curvature term of $\mathcal{M}_{\leq r}$ at a point X is proportional to Σ^{-1}, where Σ denotes the truncated (at rank r) singular values of X. When X approaches a matrix of strictly smaller rank, this term tends to infinity. To alleviate this problem, we utilize the concept of desingularization, a well-known technique in algebraic geometry. Informally, we move the optimization problem from this singular set to a new, smooth manifold, which, however, is intimately related to an original manifold. Concretely, we use the following manifold:

$$\widehat{\mathcal{M}}_r = \left\{ (A, Y) \in \mathbb{R}^{n \times m} \times Gr(m - r, m) : AY = 0 \right\},$$

here Gr denotes the *Grassmann manifold.* It is easy to see that we can 'lift' optimization problems from $\mathcal{M}_{\leq r}$ to $\widehat{\mathcal{M}}_r$, which is, as we prove, a smooth manifold. Using these observations, we build a second-order method on $\widehat{\mathcal{M}}_r$ and show how to implement it efficiently. We conclude with numerical experiments which demonstrate the superiority of our method compared to more traditional ones, such as truncated Newton method [Absil et al., 2009] or Riemannian conjugate gradient [Smith, 1994].

1.3.4 Geometry Score: A Method For Comparing Generative Adversarial Networks

In this work, we attack the problem of evaluation of the quality of generative models, in particular, GANs. Our analysis is inspired by the Manifold Hypothesis [Goodfellow et al., 2016]. Informally, it states that any real-life data is supported on a small dimensional manifold. Thus, if we have some generative model, we expect the *generated manifold* to at least be close to the original manifold in 'shape'. However, how to quantify the difference in shapes of two manifolds, to which we do not have direct access? We use Topological Data Analysis (TDA) in order to achieve this. On a very high level, we construct an approximation of manifolds using *simplicial complexes* — primitive spaces built out of simplexes. Note, however, that the task of reconstruction of a manifold given simples from it is ill-posed: it could have been a discrete set of points or a single blob. To alleviate this, the reconstruction happens at *all possible scales* at once, tracking the evolution from a discrete set to a connected space. After simplicial complexes are built, we compute their topological properties, namely *persistent homology* [Ghrist, 2008]. Homology, widely used in algebraic topology, represents certain properties of a manifold shape, concretely, the number of *holes* in it. Persistent homology allows one to find an approximation of this characteristic for a sequence of simplicial complexes, as described above. We then compare real data and generated data by comparing their topological characteristic, and build a new metric termed *Geometry Score*. We show that it allows us to distinguish between spaces of various shapes and compare GANs (even applied to non-visual data, where such metrics as FID and Inception Distance are not applicable). We find that in cases when Inception Score fails, our metric still allows distinguishing between two generative models.

1.3.5 Hyperbolic Image Embeddings

Hyperbolic geometry, recently introduced to the Machine Learning community in [Nickel and Kiela, 2017], was shown to be very successful in tasks of graph/taxonomy embeddings and several NLP problems. There was, however, no extension to the visual domain. In this work, we argue that hyperbolic geometry may be beneficial for certain image-based tasks as well. We start by analyzing whether the visual datasets contain hyperbolic structure. Our primary tool for this is δ-Hyperbolicity introduced in [Gromov, 1987]. It allows us to estimate the 'degree' to which the given dataset is hyperbolic, and we find that this degree is quite high in such datasets as CIFAR10, CUB, and MiniImageNet. We additionally suggest a new data-based approach for estimation of the hyperparameter c, inversely related to the curvature of hyperbolic space, which is necessary when building hyperbolic models. We show how standard pipelines for *few-shot learning* and *re-identification* tasks can be modified to incorporate hyperbolic geometry and perform extensive numerical experiments. We find that even simple Euclidean models, when modified to hyperbolic geometry, can perform on the level of state-of-the-art models.

Chapter 2

Expressive Power of Recurrent Neural Networks

2.1 Introduction

Deep neural networks solve many practical problems both in computer vision via Convolutional Neural Networks (CNNs) [LeCun et al., 1995, Szegedy et al., 2015b, He et al., 2016] and in audio and text processing via Recurrent Neural Networks (RNNs) [Graves et al., 2013, Mikolov et al., 2011, Gers et al., 1999]. However, although many works focus on expanding the theoretical explanation of neural networks success [Martens and Medabalimi, 2014, Delalleau and Bengio, 2011, Cohen et al., 2016], the full theory is yet to be developed.

One line of work focuses on *expressive power*, i.e. proving that some architectures are more expressive than others. [Cohen et al., 2016] showed the connection between Hierarchical Tucker (HT) tensor decomposition and CNNs, and used this connection to prove that deep CNNs are exponentially more expressive than their shallow counterparts. However, no such result exists for Recurrent Neural Networks. The contributions of this paper are three-fold.

1. We show the connection between recurrent neural networks and Tensor Train decomposition (see Sec. 2.4);

2. We formulate and prove the expressive power theorem for the Tensor Train

decomposition (see Sec. 2.5), which – on the language of RNNs – can be interpreted as follows: to (exactly) emulate a recurrent neural network, a shallow (non-recurrent) architecture of exponentially larger width is required;

3. Combining the obtained and known results, we compare the expressive power of recurrent (TT), convolutional (HT), and shallow (CP) networks with each other (see table 2.2).

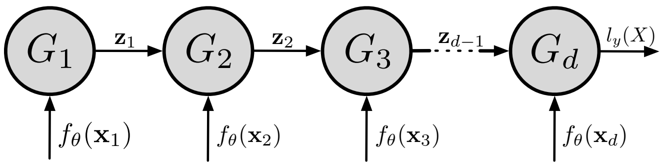


Figure 2-1: Recurrent-type neural architecture that corresponds to the Tensor Train decomposition. Gray circles are bilinear maps (for details see section 2.4).

2.2 Deep Learning and Tensor Networks

In this section, we review the known connections between tensor decompositions and deep learning and then show the new connection between Tensor Train decomposition and recurrent neural networks.

Suppose that we have a classification problem and a dataset of pairs

$$\{(X^{(b)}, y^{(b)})\}_{b=1}^N$$

. Let us assume that each object $X^{(b)}$ is represented as a sequence of vectors

$$X^{(b)} = (\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_d), \quad \mathbf{x}_k \in \mathbb{R}^n, \tag{2.1}$$

which is often the case. To find this kind of representation for images, several approaches are possible. The approach that we follow is to split an image into patches of small size, possibly overlapping, and arrange the vectorized patches in a certain order. An example of this procedure is presented on fig. 2-2.

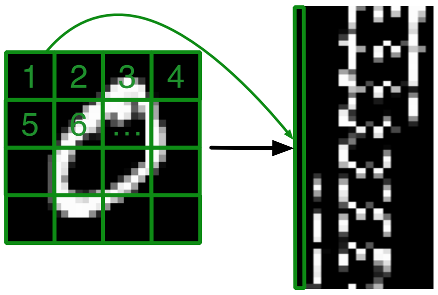


Figure 2-2: Representation of an image in the form of eq. (2.1). A window of size 7×7 moves across the image of size 28×28 extracting image patches, which are then vectorized and arranged into a matrix of size 49×16.

We use lower-dimensional *representations* of $\{\mathbf{x}_k\}_{k=1}^{d}$. For this we introduce a collection of parameter dependent feature maps $\{f_{\theta_\ell} : \mathbb{R}^n \to \mathbb{R}\}_{\ell=1}^{m}$, which are organized into a representation map

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m.$$

A typical choice for such a map is

$$f_\theta(\mathbf{x}) = \sigma(A\mathbf{x} + b),$$

that is an affine map followed by some nonlinear activation σ. In the image case if X was constructed using the procedure described above, the map f_θ resembles the traditional convolutional maps – each image patch is projected by an affine map with parameters shared across all the patches, which is followed by a pointwise activation function.

Score functions considered in [Cohen et al., 2016] can be written in the form

$$l_y(X) = \langle \mathcal{W}_y, \Phi(X) \rangle, \tag{2.2}$$

where $\Phi(X)$ is a *feature tensor*, defined as

$$\Phi(X)^{i_1 i_2 \ldots i_d} = f_{\theta_{i_1}}(\mathbf{x}_1) f_{\theta_{i_2}}(\mathbf{x}_2) \ldots f_{\theta_{i_d}}(\mathbf{x}_d), \tag{2.3}$$

and $\mathcal{W}_y \in \mathbb{R}^{m \times m \times \ldots m}$ is a trainable weight tensor. Inner product in eq. (2.2) is just a total sum of the entry-wise product of $\Phi(X)$ and $\mathcal{W}_y$. It is also shown that the hypothesis space of the form eq. (2.2) has the universal representation property for $m \to \infty$. Similar score functions were considered in [Novikov et al., 2016, Stoudenmire and Schwab, 2016].

Storing the full tensor $\mathcal{W}_y$ requires an exponential amount of memory, and to reduce the number of degrees of freedom one can use a *tensor decompositions*. Various decompositions lead to specific network architectures and in this context, expressive power of such a network is effectively measured by *ranks* of the decomposition, which determine the complexity and a total number of degrees of freedom. For the Hierarchical Tucker (HT) decomposition, [Cohen et al., 2016] proved the expressive power property, i.e. that for almost any tensor $\mathcal{W}_y$ its HT-rank is exponentially smaller than its CP-rank. We analyze Tensor Train-Networks (TT-Networks), which correspond to a recurrent-type architecture. We prove that these networks also have exponentially larger representation power than shallow networks (which correspond to the CP-decomposition).

2.3 Tensor formats reminder

In this section we briefly review all the necessary definitions. As a d-dimensional tensor $\mathcal{X}$ we simply understand a multidimensional array:

$$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}.$$

To work with tensors it is convenient to use their *matricizations*, which are defined as follows. Let us choose some subset of axes $s = \{i_1, i_2 \ldots i_{m_s}\}$ of $\mathcal{X}$, and denote its compliment by $t = \{j_1, j_2 \ldots j_{d-m_s}\}$, e.g. for a 4 dimensional tensor s could be $\{1, 3\}$ and t is $\{2, 4\}$. Then matricization of $\mathcal{X}$ specified by (s, t) is a matrix

$$\mathcal{X}^{(s,t)} \in \mathbb{R}^{n_{i_1} n_{i_2} \ldots n_{i_{m_s}} \times n_{j_1} n_{j_2} \ldots n_{j_{d-m_s}}},$$

obtained simply by transposing and reshaping the tensor $\mathcal{X}$ into matrix, which in practice e.g. in `Python`, is performed using `numpy.reshape` function. Let us now introduce tensor decompositions we will use later.

2.3.1 Canonical

Canonical decomposition, known as CANDECOMP/PARAFAC or CP-decomposition for short [Harshman, 1970, Carroll and Chang, 1970], is defined as follows

$$\mathcal{X}^{i_1 i_2 \dots i_d} = \sum_{\alpha=1}^{r} \mathbf{v}_{1,\alpha}^{i_1} \mathbf{v}_{2,\alpha}^{i_2} \dots \mathbf{v}_{d,\alpha}^{i_d}, \quad \mathbf{v}_{i,\alpha} \in \mathbb{R}^{n_i}. \tag{2.4}$$

The minimal r such that this decomposition exists is called the *canonical* or *CP-rank* of $\mathcal{X}$. We will use the following notation

$$\operatorname{rank}_{CP} \mathcal{X} = r.$$

When $\operatorname{rank}_{CP} \mathcal{X} = 1$ it can be written simply as

$$\mathcal{X}^{i_1 i_2 \dots i_d} = \mathbf{v}_1^{i_1} \mathbf{v}_2^{i_2} \dots \mathbf{v}_d^{i_d},$$

which means that modes of $\mathcal{X}$ are perfectly separated from each other. Note that storing all entries of a tensor $\mathcal{X}$ requires $O(n^d)$ memory, while its canonical decomposition takes only $O(dnr)$. However, the problems of determining the exact CP-rank of a tensor and finding its canonical decomposition are NP-hard, and the problem of approximating a tensor by a tensor of lower CP-rank is ill-posed.

2.3.2 Tensor Train

A tensor $\mathcal{X}$ is said to be represented in the Tensor Train (TT) format [Oseledets, 2011] if each element of $\mathcal{X}$ can be computed as follows

$$\mathcal{X}^{i_1 i_2 \dots i_d} = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \dots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1^{i_1 \alpha_1} G_2^{\alpha_1 i_2 \alpha_2} \dots G_d^{\alpha_{d-1} i_d}, \tag{2.5}$$

where the tensors $G_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ ($r_0 = r_d = 1$ by definition) are the so-called *TT-cores*. The element-wise minimal ranks $\mathbf{r} = (r_1, \dots r_{d-1})$ such that decomposition (2.5) exists are called TT-ranks

$$\operatorname{rank}_{TT} \mathcal{X} = \mathbf{r}.$$

Note that for fixed values of $i_1, i_2 \ldots, i_d$, the right-hand side of eq. (2.5) is just a product of matrices

$$G_1[1, i_1, :]G_2[:, i_2, :] \ldots G_d[:, i_d, 1].$$

Storing $\mathcal{X}$ in the TT-format requires $O(dnr^2)$ memory and thus also achieves significant compression of the data. Given some tensor $\mathcal{X}$, the algorithm for finding its TT-decomposition is constructive and is based on a sequence of Singular Value Decompositions (SVDs), which makes it more numerically stable than CP-format. We also note that when all the TT-ranks equal to each other

$$\operatorname{rank}_{TT} \mathcal{X} = (r, r, \ldots, r),$$

we will sometimes write for simplicity

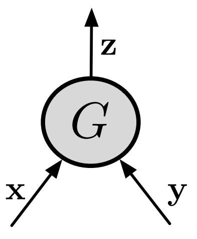
$$\operatorname{rank}_{TT} \mathcal{X} = r.$$

2.3.3 Hierarchical Tucker

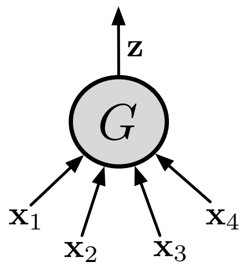
A further generalization of the TT-format leads to the so-called Hierarchical Tucker (HT) format. The definition of the HT-format is a bit technical and requires introducing the *dimension tree* [Grasedyck, 2010, Definition 3.1]. In the next section we will provide an informal introduction into the HT-format, and for more details, we refer the reader to [Grasedyck, 2010, Grasedyck and Hackbusch, 2011, Hackbusch, 2012].

2.4 Architectures based on Tensor Decompositions

To construct the tensorial networks we introduce *bilinear* and *multilinear* units, which perform a bilinear (multilinear) map of their inputs (see fig. 2-3 for an illus-



(a) Bilinear unit



(b) Multilinear unit

Figure 2-3: Nodes performing multilinear map of their inputs. d-linear unit is specified by a $d+1$ dimensional core G.

tration). Suppose that $\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m$ and $G \in \mathbb{R}^{n \times m \times k}$. Then a bilinear unit G performs a bilinear map $G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$, defined by the formula

$$\begin{aligned} &G(\mathbf{x}, \mathbf{y}) = \mathbf{z}, \\ &\mathbf{z}^k = \sum_{i,j} G^{ijk} \mathbf{x}^i \mathbf{y}^j. \end{aligned} \tag{2.6}$$

Similarly, for $\mathbf{x}_1 \in \mathbb{R}^{n_1}, \dots \mathbf{x}_d \in \mathbb{R}^{n_d}$, a multilinear unit $G \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d \times n_j}$ defines a multilinear map $G : \prod_{k=1}^d \mathbb{R}^{n_k} \to \mathbb{R}^{n_j}$ by the formula

$$\begin{aligned} &G(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d) = \mathbf{z} \\ &\mathbf{z}^j = \sum_{i_1, i_2, \dots, i_d} G^{i_1 i_2 \dots i_d j} \mathbf{x}_1^{i_1} \mathbf{x}_2^{i_2} \dots \mathbf{x}_d^{i_d}. \end{aligned} \tag{2.7}$$

In the rest of this section, we describe how to compute the score functions $l_y(X)$ (see eq. (2.1)) for each class label y, which then could be fed into the loss function (such as cross-entropy). The architecture we propose to implement the score functions is illustrated on fig. 2-1. For a vector $\mathbf{r} = (r_1, r_2, \dots r_{d-1})$ of positive integers (rank hyperparameter) we define bilinear units

$$G_k \in \mathbb{R}^{r_{k-1} \times m \times r_k},$$

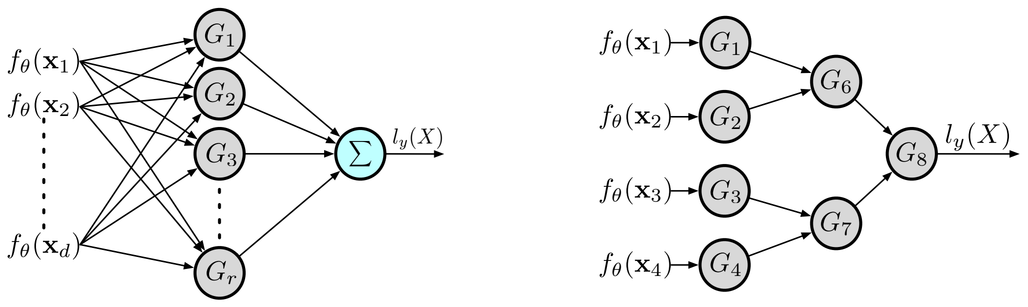
with $r_0 = r_d = 1$. Note that because $r_0 = 1$, the first unit G_1 is in fact just a linear map, and because $r_d = 1$ the output of the network is just a number. On a step $k \geq 2$ the representation $f_\theta(\mathbf{x}_k)$ and output of the unit G_{k-1} of size r_k are fed into the unit G_k. Thus we obtain a recurrent-type neural network with multiplicative connections and without non-linearities.

To draw a connection with the Tensor Train decomposition we make the following observation. For each of the class labels y let us construct the tensor $\mathcal{W}_y$ using the definition of TT-decomposition (eq. (2.5)) and taking $\{G_k\}_{k=1}^d$ used for constructing $l_y(X)$ as its TT-cores. Using the definition of the eq. (2.3) we find that the score functions computed by the network from fig. 2-1 are given by the formula

$$l_y(X) = \sum_{i_1,i_2,\dots i_d} \mathcal{W}_y^{i_1 i_2 \dots i_d} \Phi(X)^{i_1 i_2 \dots i_d}, \tag{2.8}$$

which is verified using eq. (2.5) and eq. (2.3). Thus, we can conclude that the network presented on fig. 2-1 realizes the TT-decomposition of the weight tensor. We also note that the size of the output of the bilinear unit G_k in the TT-Network is equal to r_k, which means that the TT-ranks correspond to the *width* of the network.

Let us now consider other tensor decompositions of the weight tensors $\mathcal{W}_y$, construct corresponding network architectures, and compare their properties with the original TT-Network.



(a) CP-Network (b) HT-Network

Figure 2-4: Examples of networks corresponding to various tensor decompositions.

A network corresponding to the CP-decomposition is visualized on fig. 2-4a. Each multilinear unit G_α is given by a summand in the formula eq. (2.4), namely

$$G_\alpha^{i_1 i_2 \dots i_d} = \mathbf{v}_{1,\alpha}^{i_1} \mathbf{v}_{2,\alpha}^{i_2} \dots \mathbf{v}_{d,\alpha}^{i_d}, \quad \alpha \in \{1, \dots r\}.$$

Note that the output of each G_α in this case is just a number, and in total there are $\text{rank}_{CP}\, \mathcal{W}_y$ multilinear units. Their outputs are then summed up by the Σ node. As before rank of the decomposition corresponds to the width of the network. However, in this case the network is *shallow*, meaning that there is only one hidden layer.

On the fig. 2-4b a network of other kind is presented. Tensor decomposition which underlies it is the Hierarchical Tucker decomposition, and hence we call it the HT-Network. It is constructed using a binary tree, where each node other than leaf corresponds to a bilinear unit, and leaves correspond to linear units. Inputs are fed into leaves, and this data is passed along the tree to the root, which outputs a number. Ranks, in this case, are just the sizes of the outputs of the intermediate units. We will denote them by $\operatorname{rank}_{HT} \mathcal{X}$. These are networks considered in [Cohen et al., 2016], where the expressive power of such networks was analyzed and was argued that they resemble traditional CNNs. In general Hierarchical Tucker decomposition may be constructed using an arbitrary tree, but not much theory is known in general case.

Our main theoretical results are related to a comparison of the expressive power of these kinds of networks. Namely, the question that we ask is as follows. Suppose that we are given a TT-Network. How complex would be a CP- or HT-Network realizing the same score function? A natural measure of complexity, in this case, would be the rank of the corresponding tensor decomposition. To make transitioning between tensor decompositions and deep learning vocabulary easier, we introduce the following table.

Table 2.1: Correspondence between languages of Tensor Analysis and Deep Learning.

Tensor Decompositions	**Deep Learning**
CP-decomposition	shallow network
TT-decomposition	RNN
HT-decomposition	CNN
rank of the decomposition	width of the network

2.5 Theoretical Analysis

In this section we prove the expressive power theorem for the Tensor Train decomposition, that is we prove that given a random d-dimensional tensor in the TT format with ranks $\mathbf{r}$ and modes n, with probability 1 this tensor will have exponentially large CP-rank. Note that the reverse result can not hold true since TT-ranks can not be larger than CP-ranks: $\operatorname{rank}_{TT} \mathcal{X} \leq \operatorname{rank}_{CP} \mathcal{X}$.

It is known that the problem of determining the exact CP-rank of a tensor is NP-hard.

To bound CP-rank of a tensor the following lemma is useful.

Lemma 1. *Let* $\mathcal{X}^{i_1 i_2 \dots i_d}$ *and* $\operatorname{rank}_{CP} \mathcal{X} = r$. *Then for any matricization* $\mathcal{X}^{(s,t)}$ *we have* $\operatorname{rank} \mathcal{X}^{(s,t)} \leq r$, *where the ordinary matrix rank is assumed.*

Proof. Proof is based on the following observation. Let

$$\mathcal{A}^{i_1 i_2 \dots i_d} = \mathbf{v}_1^{i_1} \mathbf{v}_2^{i_2} \dots \mathbf{v}_d^{i_d},$$

be a CP-rank 1 tensor. Note for any s, t

$$\operatorname{rank} \mathcal{A}^{(s,t)} = 1,$$

because $\mathcal{A}^{(s,t)}$ can be written as $\mathbf{u}\mathbf{w}^T$ for some $\mathbf{u}$ and $\mathbf{w}$. Then the statement of the lemma follows from the facts that matricization is a linear operation, and that for matrices

$$\operatorname{rank}(A + B) \leq \operatorname{rank} A + \operatorname{rank} B.$$

□

We use this lemma to provide a lower bound on the CP-rank in the theorem formulated below. For example, suppose that we found some matricization of a tensor $\mathcal{X}$ which has matrix rank r. Then, by using the lemma we can estimate that $\operatorname{rank}_{CP} \mathcal{X} \geq r$.

Let us denote $\mathbf{n} = (n_1, n_2 \dots n_d)$. Set of all tensors $\mathcal{X}$ with mode sizes $\mathbf{n}$ representable in TT-format with

$$\operatorname{rank}_{TT} \mathcal{X} \leq \mathbf{r},$$

for some vector of positive integers $\mathbf{r}$ (inequality is understood entry-wise) forms an *irreducible algebraic variety* ([Shafarevich and Hirsch, 1994]), which we denote by $\mathcal{M}_{\mathbf{r}}$. This means that $\mathcal{M}_{\mathbf{r}}$ is defined by a set of polynomial equations in $\mathbb{R}^{n_1 \times n_2 \ldots n_d}$, and that it can not be written as a union (not necessarily disjoint) of two proper non-empty algebraic subsets. An example where the latter property does not hold would be the union of axes $x = 0$ and $y = 0$ in $\mathbb{R}^2$, which is an algebraic set defined by the equation $xy = 0$. The main fact that we use about irreducible algebraic varieties is that any *proper* algebraic subset of them necessarily has measure 0 ([Ilyashenko and Yakovenko, 2008]).

For simplicity let us assume that number of modes d is even, that all mode sizes are equal to n, and we consider $\mathcal{M}_{\mathbf{r}}$ with $\mathbf{r} = (r, r \ldots r)$, so for any $\mathcal{X} \in \mathcal{M}_{\mathbf{r}}$ we have

$$\operatorname{rank}_{TT} \mathcal{X} \leq (r, r, \ldots, r),$$

entry-wise.

As the main result we prove the following theorem

Theorem 1. *Suppose that $d = 2k$ is even. Define the following set*

$$B = \{\mathcal{X} \in \mathcal{M}_{\boldsymbol{r}} : \operatorname{rank}_{CP} \mathcal{X} < q^{\frac{d}{2}}\},$$

where $q = \min\{n, r\}$.

Then

$$\mu(B) = 0,$$

where μ is the standard Lebesgue measure on $\mathcal{M}_{\boldsymbol{r}}$.

Proof. Our proof is based on applying lemma 1 to a particular matricization of $\mathcal{X}$. Namely, we would like to show that for $s = \{1, 3, \ldots d-1\}$, $t = \{2, 4, \ldots d\}$ the following set

$$B^{(s,t)} = \{\mathcal{X} \in \mathcal{M}_{\mathbf{r}} : \operatorname{rank} \mathcal{X}^{(s,t)} \leq q^{\frac{d}{2}} - 1\},$$

has measure 0. Indeed, by lemma 1 we have

$$B \subset B^{(s,t)},$$

so if $\mu(B^{(s,t)}) = 0$ then $\mu(B) = 0$ as well. Note that $B^{(s,t)}$ is an algebraic subset of $\mathcal{M}_{\mathbf{r}}$ given by the conditions that the determinants of all $q^{\frac{d}{2}} \times q^{\frac{d}{2}}$ submatrices of $\mathcal{X}^{(s,t)}$ are equal to 0. Thus to show that $\mu(B^{(s,t)}) = 0$ we need to find at least one $\mathcal{X}$ such that $\operatorname{rank} \mathcal{X}^{(s,t)} \geq q^{\frac{d}{2}}$. This follows from the fact that because $B^{(s,t)}$ is an algebraic subset of the irreducible algebraic variety $\mathcal{M}_{\mathbf{r}}$, it is either equal to $\mathcal{M}_{\mathbf{r}}$ or has measure 0, as was explained before.
One way to construct such tensor is as follows. Let us define the following tensors:

$$\begin{aligned}
&G_1^{i_1\alpha_1} = \delta_{i_1\alpha_1}, \quad G_1 \in \mathbb{R}^{1\times n\times r} \\
&G_k^{\alpha_{k-1}i_k\alpha_k} = \delta_{i_k\alpha_{k-1}}, \quad G_k \in \mathbb{R}^{r\times n\times 1}, k = 2,4,6,\dots,d-2 \\
&G_k^{\alpha_{k-1}i_k\alpha_k} = \delta_{i_k\alpha_k}, \quad G_k \in \mathbb{R}^{1\times n\times r}, k = 3,5,7,\dots,d-1 \\
&G_d^{\alpha_{d-1}i_d} = \delta_{i_d\alpha_{d-1}}, \quad G_d \in \mathbb{R}^{r\times n\times 1}
\end{aligned} \tag{2.9}$$

where $\delta_{i\alpha}$ is the Kronecker delta symbol:

$$\delta_{i\alpha} = \begin{cases} 1, & \text{if } i = \alpha, \\ 0, & \text{if } i \neq \alpha. \end{cases}$$

The TT-ranks of the tensor $\mathcal{X}$ defined by the TT-cores (2.9) are equal to

$$\operatorname{rank}_{TT} \mathcal{X} = (r, 1, r, \dots, r, 1, r).$$

Lets consider the following matricization of the tensor $\mathcal{X}$

$$\mathcal{X}^{(i_1,i_3,\dots,i_{d-1}),(i_2,i_4,\dots,i_d)}$$

The following identity holds true for any values of indices such that

$$i_k = 1,\dots,q, \;\; k = 1,\dots,d.$$

$$\begin{aligned}\mathcal{X}^{(i_1,i_3,\ldots,i_{d-1}),(i_2,i_4,\ldots,i_d)} = \sum_{\alpha_1,\ldots,\alpha_{d-1}} G_1^{i_1\alpha_1}\ldots G_d^{\alpha_{d-1}i_d} = \\ \sum_{\alpha_1,\ldots,\alpha_{d-1}} \delta_{i_1\alpha_1}\delta_{i_2\alpha_1}\delta_{i_3\alpha_3}\ldots\delta_{i_d,\alpha_{d-1}} = \delta_{i_1i_2}\delta_{i_3i_4}\ldots\delta_{i_{d-1}i_d}\end{aligned} \tag{2.10}$$

The last equality holds because $\sum_{\alpha_k=1}^{r}\delta_{i_k\alpha_k}\delta_{i_{k+1}\alpha_k} = \delta_{i_ki_{k+1}}$ for any $i_k = 1,\ldots,q$. We obtain that

$$\mathcal{X}^{(i_1,i_3,\ldots,i_{d-1}),(i_2,i_4,\ldots,i_d)} = \delta_{i_1i_2}\delta_{i_3i_4}\ldots\delta_{i_{d-1}i_d} = I^{(i_1,i_3,\ldots,i_{d-1}),(i_2,i_4,\ldots,i_d)}, \tag{2.11}$$

where I is the identity matrix of size $q^{d/2}\times q^{d/2}$ where $q=\min\{n,r\}$.

To summarize, we found an example of a tensor $\mathcal{X}$ such that $\operatorname{rank}_{TT}\mathcal{X}\le\mathbf{r}$ and the matricization $\mathcal{X}^{(i_1,i_3,\ldots,i_{d-1}),(i_2,i_4,\ldots,i_d)}$ has a submatrix being equal to the identity matrix of size $q^{d/2}\times q^{d/2}$, and hence $\operatorname{rank}\mathcal{X}^{(i_1,i_3,\ldots,i_{d-1}),(i_2,i_4,\ldots,i_d)}\ge q^{d/2}$.

This means that the canonical $\operatorname{rank}_{CP}\mathcal{X}\ge q^{d/2}$ which concludes the proof. $\square$

In other words, we have proved that for all TT-Networks besides negligible set, the equivalent CP-Network will have exponentially large width. To compare the expressive powers of the HT- and TT-Networks we use the following theorem [Grasedyck, 2010, Section 5.3.2].

Theorem 2. *For any tensor $\mathcal{X}$ the following estimates hold.*

- *If* $\operatorname{rank}_{TT}\mathcal{X}\le r$, *then* $\operatorname{rank}_{HT}\mathcal{X}\le r^2$.
- *If* $\operatorname{rank}_{HT}\mathcal{X}\le r$, *then* $\operatorname{rank}_{TT}\mathcal{X}\le r^{\log_2(d)/2}$.

It is also known that this bounds are *sharp* (see [Buczyńska et al., 2015]). Thus, we can summarize all the results in the following table 2.2.

Example that requires exponential width in a shallow network A particular example used to prove Theorem 1 is not important per se since the Theorem states that TT is exponentially more expressive than CP for almost any tensor (for a set of tensors of measure one). However, to illustrate how the Theorem translates into neural networks consider the following example.

Table 2.2: Comparison of the expressive power of various networks. Given a network of width r, specified in a column, rows correspond to the upper bound on the width of the equivalent network of other type (we assume that the number of feature maps m is greater than the width of the network r).

	TT-Network	**HT-Network**	**CP-Network**
TT-Network	r	$r^{\log_2(d)/2}$	r
HT-Network	r^2	r	r
CP-Network	$\geq r^{\frac{d}{2}}$	$\geq r^{\frac{d}{2}}$	r

Consider the task of getting d input vectors with n elements each and aiming to compute the following measure of similarity between $\mathbf{x}_1, \dots, \mathbf{x}_{d/2}$ and $\mathbf{x}_{d/2+1}, \dots, \mathbf{x}_d$:

$$l(X) = (\mathbf{x}_1^\intercal \mathbf{x}_{d/2+1}) \dots (\mathbf{x}_{d/2}^\intercal \mathbf{x}_d) \tag{2.12}$$

We argue that it can be done with a TT-Network of width n by using the TT-tensor $\mathcal{X}$ defined in the proof of Theorem 1 and feeding the input vectors in the following order: $\mathbf{x}_1, \mathbf{x}_{d/2+1}, \dots \mathbf{x}_{d/2}, \mathbf{x}_d$. The CP-network representing the same function will have $n^{d/2}$ terms (and hence $n^{d/2}$ width) and will correspond to expanding brackets in the expression (2.12).

The case of equal TT-cores In analogy to the traditional RNNs we can consider a special class of Tensor Trains with the property that all the intermediate TT-cores are equal to each other: $G_2 = G_3 = \dots = G_{d-1}$, which allows for processing sequences of varied length. We hypothesize that for this class exactly the same result as in Theorem 1 holds i.e. if we denote the variety of Tensor Trains with equal TT-cores by $\mathcal{M}_{\mathbf{r}}^{eq}$, we believe that the following hypothesis holds true:

Hypothesis 1. *Theorem 1 is also valid if $\mathcal{M}_{\mathbf{r}}$ is replaced by $\mathcal{M}_{\mathbf{r}}^{eq}$.*

To prove it we can follow the same route as in the proof of Theorem 1. While we leave finding an analytical example of a tensor with the desired property of rank maximality to a future work, we have verified numerically that randomly generated tensors $\mathcal{X}$ from $\mathcal{M}_{\mathbf{r}}^{eq}$ with $d = 6$, n ranging from 2 to 10 and r ranging from 2 to 20 (we have checked 1000 examples for each possible combination) indeed satisfy

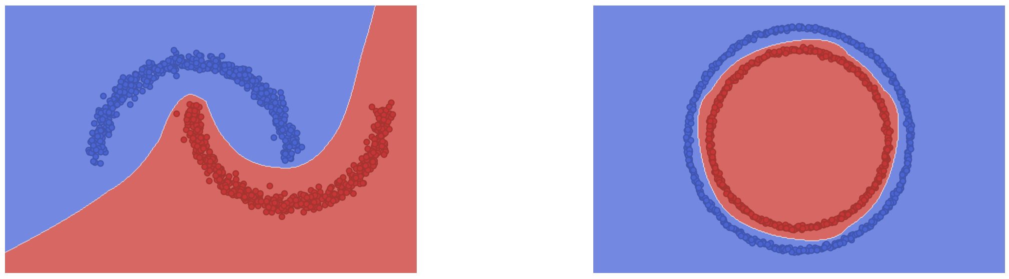


Figure 2-5: Decision boundaries of the TT-Network on toy 2-D datasets.

$\operatorname{rank}_{CP} \mathcal{X} \geq q^{\frac{d}{2}}$.

2.6 Experiments

In this section, we experimentally check if indeed – as suggested by theorem 1 – the CP-Networks require exponentially larger width compared to the TT-Networks to fit a dataset to the same level of accuracy. This is not clear from the theorem since for natural data, functions that fit this data may lay in the neglectable set where the ranks of the TT- and CP-networks are related via a polynomial function (in contrast to the exponential relationship for all function outside the neglectable set). Other possible reasons why the theory may be disconnected with practice are optimization issues (although a certain low-rank tensor exists, we may fail to find it with SGD) and the existence of the feature maps, which were not taken into account in the theory.

To train the TT- and CP-Networks, we implemented them in `TensorFlow` ([Abadi et al., 2015]) and used Adam optimizer with batch size 32 and learning rate sweeping across {4e-3, 2e-3, 1e-3, 5e-4} values. Since we are focused on assessing the expressivity of the format (in contrast to its sensitivity to hyperparameters), we always choose the best performing run according to the training loss.

For the first experiment, we generate two-dimensional datasets with `Sklearn` tools 'moons' and 'circles' [Pedregosa et al., 2011] and for each training example feed the two features as two patches into the TT-Network (see fig. 2-5). This example shows that the TT-Networks can implement nontrivial decision boundaries.

For the next experiments, we use computer vision datasets MNIST [LeCun et al.,

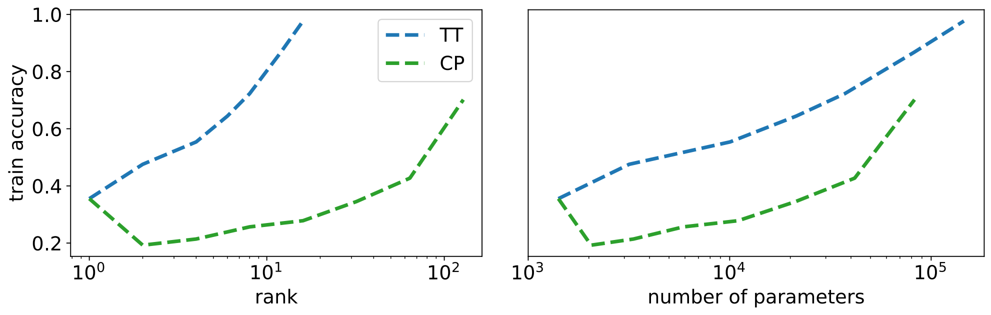


Figure 2-6: Train accuracy on CIFAR-10 for the TT- and CP-Networks wrt rank of the decomposition and total number of parameters (feature size 4 was used). Note that with rank increase the CP-Networks sometimes perform worse due to optimization issues.

1990] and CIFAR-10 [Krizhevsky and Hinton, 2009]. MNIST is a collection of 70000 handwritten digits, CIFAR-10 is a dataset of 60000 natural images which are to be classified into 10 classes such as bird or cat. We feed raw pixel data into the TT- and CP-Networks (which extract patches and apply a trainable feature map to them, see section 2.2). In our experiments we choose patch size to be 8×8, feature maps to be affine maps followed by the ReLU activation and we set number of such feature maps to 4. For MNIST, both TT- and CP-Networks show reasonable performance (1.0 train accuracy, 0.95 test accuracy without regularizers, and 0.98 test accuracy with dropout 0.8 applied to each patch) even with ranks less than 5, which may indicate that the dataset is too simple to draw any conclusion, but serves as a sanity check.

We report the training accuracy for CIFAR-10 on fig. 2-6. Note that we did not use regularizers of any sort for this experiment since we wanted to compare expressive power of networks (the best test accuracy we achieved this way on CIFAR-10 is 0.45 for the TT-Network and 0.2 for the CP-Network). On practice, the expressive power of the TT-Network is only polynomially better than that of the CP-network (fig. 2-6), probably because of the reasons discussed above.

2.7 Related work

A large body of work is devoted to analyzing the theoretical properties of neural networks [Cybenko, 1989, Hornik et al., 1989, Shwartz-Ziv and Tishby, 2017]. Recent studies focus on depth efficiency [Raghu et al., 2017, Montufar et al., 2014, Eldan and Shamir, 2016, Sutskever et al., 2013], in most cases providing worst-case guaranties such as bounds between deep and shallow networks width. Two works are especially relevant since they analyze depth efficiency from the viewpoint of tensor decompositions: expressive power of the Hierarchical Tucker decomposition [Cohen et al., 2016] and its generalization to handle activation functions such as ReLU [Cohen and Shashua, 2016]. However, all of the works above focus on feedforward networks, while we tackle recurrent architectures. The only other work that tackles expressivity of RNNs is the concurrent work that applies the TT-decomposition to explicitly modeling high-order interactions of the previous hidden states and analyses the expressive power of the resulting architecture [Yu et al., 2017]. This work, although very related to ours, analyses a different class of recurrent models.

Models similar to the TT-Network were proposed in the literature but were considered from the practical point of view in contrast to the theoretical analyses provided in this paper. [Novikov et al., 2016, Stoudenmire and Schwab, 2016] proposed a model that implements eq. (2.2), but with a predefined (not learnable) feature map Φ. [Wu et al., 2016] explored recurrent neural networks with multiplicative connections, which can be interpreted as the TT-Networks with bilinear maps that are shared $G_k = G$ and have low-rank structure imposed on them.

2.8 Conclusion

In this paper, we explored the connection between recurrent neural networks and Tensor Train decomposition and used it to prove the expressive power theorem, which states that a shallow network of exponentially large width is required to mimic a recurrent neural network. The downsides of this approach is that it provides worst-case analysis and do not take optimization issues into account. In the future work, we would like to address the optimization issues by exploiting the Riemannian geometry properties of the set of TT-tensors of fixed rank and extend the analysis to networks with non-linearity functions inside the recurrent connections (as was done for CNNs in [Cohen and Shashua, 2016]).

Chapter 3

Generalized Tensor Models For Recurrent Neural Networks

3.1 Introduction

Recurrent Neural Networks are firmly established to be one of the best deep learning techniques when the task at hand requires processing sequential data, such as text, audio, or video [Graves et al., 2013, Mikolov et al., 2011, Gers et al., 1999]. The ability of these neural networks to efficiently represent a rich class of functions with a relatively small number of parameters is often referred to as *depth efficiency*, and the theory behind this phenomenon is not yet fully understood. A recent line of work [Cohen and Shashua, 2016, Cohen et al., 2016, Khrulkov et al., 2018, Cohen et al., 2018] focuses on comparing various deep learning architectures in terms of their *expressive power*.

It was shown in [Cohen et al., 2016] that ConvNets with product pooling are *exponentially* more expressive than shallow networks, that is there exist functions realized by ConvNets which require an exponentially large number of parameters in order to be realized by shallow nets. A similar result also holds for RNNs with multiplicative recurrent cells [Khrulkov et al., 2018]. We aim to extend this analysis to RNNs with rectifier nonlinearities which are often used in practice. The main challenge of such analysis is that the tools used for analyzing multiplicative networks, namely, properties of standard *tensor decompositions* and ideas from algebraic geometry, can not be applied in this case, and thus some other approach is required. Our objective is to apply the machinery of *generalized tensor decompositions*, and show universality and existence of depth efficiency in such RNNs.

3.2 Related work

Tensor methods have a rich history of successful application in machine learning. [Vasilescu and Terzopoulos, 2002], in their framework of TensorFaces, proposed to treat facial image data as multidimensional arrays and analyze them with tensor decompositions, which led to significant boost in face recognition accuracy. [Bailey and Aeron, 2017] employed higher-order co-occurence data and tensor factorization techniques to improve on word embeddings models. Tensor methods also allow to produce more accurate and robust recommender systems by taking into account a multifaceted nature of real environments [Frolov and Oseledets, 2017].

In recent years a great deal of work was done in applications of tensor calculus to both theoretical and practical aspects of deep learning algorithms. [Lebedev et al., 2014] represented filters in a convolutional network with CP decomposition [Harshman, 1970, Carroll and Chang, 1970] which allowed for much faster inference at the cost of a negligible drop in performance. [Novikov et al., 2015] proposed to use Tensor Train (TT) decomposition [Oseledets, 2011] to compress fully–connected layers of large neural networks while preserving their expressive power. Later on, TT was exploited to reduce the number of parameters and improve the performance of recurrent networks in long–term forecasting [Yu et al., 2017] and video classification [Yang et al., 2017] problems.

In addition to the practical benefits, tensor decompositions were used to analyze theoretical aspects of deep neural nets. [Cohen et al., 2016] investigated a connection between various network architectures and tensor decompositions, which made possible to compare their expressive power. Specifically, it was shown that CP and Hierarchial Tucker [Grasedyck, 2010] decompositions correspond to shallow networks and convolutional networks respectively. Recently, this analysis was extended by [Khrulkov et al., 2018] who showed that TT decomposition can be represented as a recurrent network with multiplicative connections. This specific form of RNNs was also empirically proved to provide a substantial performance boost over standard RNN models [Wu et al., 2016].

First results on the connection between tensor decompositions and neural networks were obtained for rather simple architectures, however, later on, they were extended in order to analyze more practical deep neural nets. It was shown that theoretical results can be generalized to a large class of CNNs with ReLU nonlinearities [Cohen and Shashua, 2016] and dilated convolutions [Cohen et al., 2018], providing valuable insights on how they can be improved. However, there is a missing piece in the whole picture as theoretical properties of more complex nonlinear RNNs have yet to be analyzed. In this paper, we elaborate on this problem and present new tools for conducting a theoretical analysis of such RNNs, specifically when rectifier nonlinearities are used.

3.3 Architectures inspired by tensor decompositions

Let us now recall the known results about the connection of tensor decompositions and multiplicative architectures, and then show how they are generalized in order to include networks with ReLU nonlinearities.

3.3.1 Score functions and feature tensor

Suppose that we are given a dataset of objects with a sequential structure, i.e. every object in the dataset can be written as

$$X = \left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}\right), \quad \mathbf{x}^{(t)} \in \mathbb{R}^N. \tag{3.1}$$

We also introduce a parametric *feature map* $f_\theta : \mathbb{R}^N \to \mathbb{R}^M$ which essentially preprocesses the data before it is fed into the network. Assumption 3.1 holds for many types of data, e.g. in the case of natural images we can cut them into rectangular patches which are then arranged into vectors $\mathbf{x}^{(t)}$. A typical choice for the feature map f_θ in this particular case is an affine map followed by a nonlinear activation:

$f_\theta(\mathbf{x}) = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$. To draw the connection between tensor decompositions and feature tensors we consider the following *score functions* (logits[1]):

$$\ell(X) = \langle \boldsymbol{\mathcal{W}}, \boldsymbol{\Phi}(X) \rangle = (\operatorname{vec} \boldsymbol{\mathcal{W}})^\top \operatorname{vec} \boldsymbol{\Phi}(X), \tag{3.2}$$

where $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{M \times M \times \ldots \times M}$ is a trainable T–way weight tensor and $\boldsymbol{\Phi}(X) \in \mathbb{R}^{M \times M \times \ldots \times M}$ is a rank 1 *feature tensor*, defined as

$$\boldsymbol{\Phi}(X) = f_\theta(\mathbf{x}^{(1)}) \otimes f_\theta(\mathbf{x}^{(2)}) \ldots \otimes f_\theta(\mathbf{x}^{(T)}), \tag{3.3}$$

where we have used the operation of outer product $\otimes$, which is important in tensor calculus. For a tensor $\boldsymbol{\mathcal{A}}$ of order N and a tensor $\boldsymbol{\mathcal{B}}$ of order M their outer product $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$ is a tensor of order $N + M$ defined as:

$$\boldsymbol{\mathcal{C}}_{i_1 i_2 \ldots i_N j_1 j_2 \ldots j_M} = \boldsymbol{\mathcal{A}}_{i_1 i_2 \cdots i_N} \boldsymbol{\mathcal{B}}_{j_1 j_2 \cdots j_M}. \tag{3.4}$$

It is known that (3.2) possesses the *universal approximation property* (it can approximate any function with any prescribed precision given sufficiently large M) under mild assumptions on f_θ [Cohen et al., 2016, Poggio and Girosi, 1990].

3.3.2 Tensor Decompositions

Working the entire weight tensor $\boldsymbol{\mathcal{W}}$ in eq. (3.2) is impractical for large M and T, since it requires exponential in T number of parameters. Thus, we compactly represent it using *tensor decompositions*, which will further lead to different neural network architectures, referred to as *tensor networks* [Cichocki et al., 2017].

CP-decomposition The most basic decomposition is the so-called Canonical (CP) decomposition [Harshman, 1970, Carroll and Chang, 1970] which is defined

[1]By *logits* we mean immediate outputs of the last hidden layer before applying nonlinearity. This term is adopted from classification tasks where neural network usually outputs *logits* and following softmax nonlinearity transforms them into valid probabilities.

as follows

$$\boldsymbol{\mathcal{W}} = \sum_{r=1}^{R} \lambda_r \mathbf{v}_r^{(1)} \otimes \mathbf{v}_r^{(2)} \otimes \ldots \otimes \mathbf{v}_r^{(T)} \tag{3.5}$$

where $\mathbf{v}_r^{(t)} \in \mathbb{R}^M$ and minimal value of R such that decomposition (3.3.2) exists is called *canonical rank of a tensor (CP–rank)*. By substituting section 3.3.2 into eq. (3.2) we find that

$$\ell(X) = \sum_{r=1}^{R} \lambda_r \left[\langle f_\theta(\mathbf{x}^{(1)}), \mathbf{v}_r^{(1)}\rangle \otimes \ldots \otimes \langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}_r^{(T)}\rangle\right] = \sum_{r=1}^{R} \lambda_r \prod_{t=1}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{v}_r^{(t)}\rangle. \tag{3.6}$$

In the equation above, outer products $\otimes$ are taken between scalars and coincide with the ordinary products between two numbers. However, we would like to keep this notation as it will come in handy later, when we generalize tensor decompositions to include various nonlinearities.

TT-decomposition Another tensor decomposition is Tensor Train (TT) decomposition [Oseledets, 2011] which is defined as follows

$$\boldsymbol{\mathcal{W}} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{T-1}=1}^{R_{T-1}} \mathbf{g}_{r_0 r_1}^{(1)} \otimes \mathbf{g}_{r_1 r_2}^{(2)} \otimes \ldots \otimes \mathbf{g}_{r_{T-1} r_T}^{(T)}, \tag{3.7}$$

where $\mathbf{g}_{r_{t-1} r_t}^{(t)} \in \mathbb{R}^M$ and $r_0 = r_T = 1$ by definition. If we gather vectors $\mathbf{g}_{r_{t-1} r_t}^{(t)}$ for all corresponding indices $r_{t-1} \in \{1, \ldots, R_{t-1}\}$ and $r_t \in \{1, \ldots, R_t\}$ we will obtain three–dimensional tensors $\boldsymbol{\mathcal{G}}^{(t)} \in \mathbb{R}^{M \times R_{t-1} \times R_t}$ (for $t = 1$ and $t = T$ we will get matrices $\boldsymbol{\mathcal{G}}^{(1)} \in \mathbb{R}^{M \times 1 \times R_1}$ and $\boldsymbol{\mathcal{G}}^{(T)} \in \mathbb{R}^{M \times R_{T-1} \times 1}$). The set of all such tensors $\{\boldsymbol{\mathcal{G}}^{(t)}\}_{t=1}^{T}$ is called *TT–cores* and minimal values of $\{R_t\}_{t=1}^{T-1}$ such that decomposition (3.7) exists are called *TT–ranks*. In the case of TT decomposition, the score function has the following form:

$$\ell(X) = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{T-1}=1}^{R_{T-1}} \prod_{t=1}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}_{r_{t-1} r_t}^{(t)}\rangle. \tag{3.8}$$

3.3.3 Connection between TT and RNN

Now we want to show that the score function for Tensor Train decomposition exhibits particular recurrent structure similar to that of RNN. We define the following *hidden states*:

$$\begin{aligned} &\mathbf{h}^{(1)} \in \mathbb{R}^{R_1} : \mathbf{h}^{(1)}_{r_1} = \langle f_\theta(\mathbf{x}^{(1)}), \mathbf{g}^{(1)}_{r_0 r_1} \rangle, \\ &\mathbf{h}^{(t)} \in \mathbb{R}^{R_t} : \mathbf{h}^{(t)}_{r_t} = \sum_{r_{t-1}=1}^{R_{t-1}} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \mathbf{h}^{(t-1)}_{r_{t-1}} \quad t = 2, \dots, T. \end{aligned} \tag{3.9}$$

Such definition of hidden states allows for more compact form of the score function.

Lemma 2. *Under the notation introduced in eq.* (3.9)*, the score function can be written as*

$$\ell(X) = \mathbf{h}^{(T)} \in \mathbb{R}^1.$$

Proof of Lemma 2 as well as the proofs of our main results from Section 3.5 were moved to Section 3.8 due to limited space.

Note that with a help of TT–cores we can rewrite eq. (3.9) in a more convenient index form:

$$\mathbf{h}^{(t)}_k = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(t)}_{ijk} \, f_\theta(\mathbf{x}^{(t)})_i \, \mathbf{h}^{(t-1)}_j = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(t)}_{ijk} \left[f_\theta(\mathbf{x}^{(t)}) \otimes \mathbf{h}^{(t-1)} \right]_{ij}, \quad k = 1, \dots, R_t, \tag{3.10}$$

where the operation of tensor contraction is used. Combining all weights from $\boldsymbol{\mathcal{G}}^{(t)}$ and $f_\theta(\cdot)$ into a single variable $\Theta^{(t)}_{\boldsymbol{\mathcal{G}}}$ and denoting the composition of feature map, outer product, and contraction as $g : \mathbb{R}^{R_{t-1}} \times \mathbb{R}^N \times \mathbb{R}^{N \times R_{t-1} \times R_t} \to \mathbb{R}^{R_t}$ we arrive at the following vector form:

$$\mathbf{h}^{(t)} = g(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \Theta^{(t)}_{\boldsymbol{\mathcal{G}}}), \quad \mathbf{h}^{(t)} \in \mathbb{R}^{R_t}. \tag{3.11}$$

This equation can be considered as a generalization of hidden state equation for Recurrent Neural Networks as here all hidden states $\mathbf{h}^{(t)}$ may in general have different dimensionalities and weight tensors $\Theta^{(t)}_{\boldsymbol{\mathcal{G}}}$ depend on the time step. However, if we set $R = R_1 = \cdots = R_{T-1}$ and $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}^{(2)} = \cdots = \boldsymbol{\mathcal{G}}^{(T-1)}$ we will get simplified hidden state equation used in standard recurrent architectures:

$$\mathbf{h}^{(t)} = g(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \Theta_{\mathcal{G}}), \quad \mathbf{h}^{(t)} \in \mathbb{R}^R, \quad t = 2, \dots, T-1. \tag{3.12}$$

Note that this equation is applicable to all hidden states except for the first $\mathbf{h}^{(1)} = \mathcal{G}^{(1)} f_\theta(\mathbf{x}^{(1)})$ and for the last $\mathbf{h}^{(T)} = f_\theta^\top(\mathbf{x}^{(T)}) \mathcal{G}^{(T)} \mathbf{h}^{(T-1)}$, due to two–dimensional nature of the corresponding TT–cores. However, we can always pad the input sequence with two auxiliary vectors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(T+1)}$ to get full compliance with the standard RNN structure. Figure 3-1 depicts tensor network induced by TT decomposition with cores $\{\mathcal{G}^{(t)}\}_{t=1}^T$.

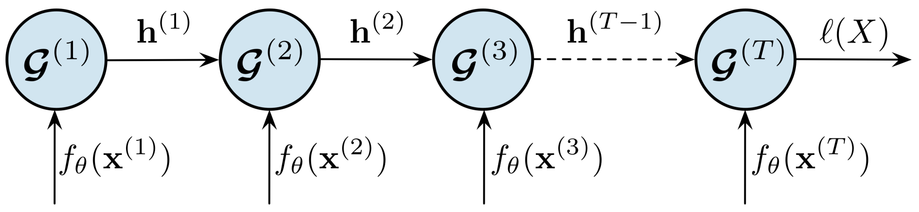


Figure 3-1: Neural network architecture which corresponds to recurrent TT–Network.

3.4 Generalized tensor networks

3.4.1 Generalized outer product

In the previous section we showed that tensor decompositions correspond to neural networks of specific structure, which are simplified versions of those used in practice as they contain multiplicative nonlinearities only. One possible way to introduce more practical nonlinearities is to replace outer product $\otimes$ in eq. (3.6) and eq. (3.10) with a generalized operator $\otimes_\xi$ in analogy to kernel methods when scalar product is replaced by nonlinear kernel function. Let $\xi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be an associative and commutative binary operator ($\forall x, y, z \in \mathbb{R} : \xi(\xi(x, y), z) = \xi(x, \xi(y, z))$ and $\forall x, y \in \mathbb{R} : \xi(x, y) = \xi(y, x)$). Note that this operator easily generalizes to the arbitrary number of operands due to associativity. For a tensor $\mathcal{A}$ of order N and a tensor $\mathcal{B}$ of order M we define their generalized outer product $\mathcal{C} = \mathcal{A} \otimes_\xi \mathcal{B}$ as an

$(N+M)$ order tensor with entries given by:

$$\boldsymbol{\mathcal{C}}_{i_1 \ldots i_N j_1 \ldots j_M} = \xi\left(\boldsymbol{\mathcal{A}}_{i_1 \ldots i_N}, \boldsymbol{\mathcal{B}}_{j_1 \ldots j_M}\right). \tag{3.13}$$

Now we can replace $\otimes$ in eqs. (3.6) and (3.10) with $\otimes_\xi$ and get networks with various nonlinearities. For example, if we take $\xi(x, y) = \max(x, y, 0)$ we will get an RNN with rectifier nonlinearities; if we take $\xi(x, y) = \ln(e^x + e^y)$ we will get an RNN with softplus nonlinearities; if we take $\xi(x, y) = xy$ we will get a simple RNN defined in the previous section. Concretely, we will analyze the following networks.

Generalized shallow network with $\boldsymbol{\xi}$–nonlinearity

- Score function:

$$\begin{aligned}\ell(X) &= \sum_{r=1}^{R} \lambda_r \left[\langle f_\theta(\mathbf{x}^{(1)}), \mathbf{v}_r^{(1)}\rangle \otimes_\xi \ldots \otimes_\xi \langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}_r^{(T)}\rangle\right] \\ &= \sum_{r=1}^{R} \lambda_r \xi\left(\langle f_\theta(\mathbf{x}^{(1)}), \mathbf{v}_r^{(1)}\rangle, \ldots, \langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}_r^{(T)}\rangle\right)\end{aligned} \tag{3.14}$$

- Parameters of the network:

$$\Theta = \Big(\{\lambda_r\}_{r=1}^{R} \in \mathbb{R}, \{\mathbf{v}_r^{(t)}\}_{r=1,t=1}^{R,T} \in \mathbb{R}^M\Big) \tag{3.15}$$

Generalized RNN with $\boldsymbol{\xi}$–nonlinearity

- Score function:

$$\begin{gathered}\mathbf{h}_k^{(t)} = \sum_{i,j} \boldsymbol{\mathcal{G}}_{ijk}^{(t)} \left[\mathbf{C}^{(t)} f_\theta(\mathbf{x}^{(t)}) \otimes_\xi \mathbf{h}^{(t-1)}\right]_{ij} = \sum_{i,j} \boldsymbol{\mathcal{G}}_{ijk}^{(t)}\, \xi\left([\mathbf{C}^{(t)} f_\theta(\mathbf{x}^{(t)})]_i, \mathbf{h}_j^{(t-1)}\right) \\ \ell(X) = \mathbf{h}^{(T)}\end{gathered} \tag{3.16}$$

- Parameters of the network:

$$\Theta = \Big(\{\mathbf{C}^{(t)}\}_{t=1}^{T} \in \mathbb{R}^{L\times M}, \{\boldsymbol{\mathcal{G}}^{(t)}\}_{t=1}^{T} \in \mathbb{R}^{L\times R_{t-1}\times R_t}\Big) \tag{3.17}$$

Note that in eq. (3.16) we have introduced the matrices $\mathbf{C}^{(t)}$ acting on the input states. The purpose of this modification is to obtain the plausible property of generalized shallow networks being able to be represented as generalized RNNs of width 1 (i.e., with all $R_i = 1$) for an arbitrary nonlinearity ξ. In the case of $\xi(x, y) = xy$, the matrices $\mathbf{C}^{(t)}$ were not necessary, since they can be simply absorbed by $\boldsymbol{\mathcal{G}}^{(t)}$ via tensor contraction (see Section 3.8 for further clarification on these points).

Initial hidden state Note that generalized RNNs require some choice of the initial hidden state $\mathbf{h}^{(0)}$. We find that it is convenient both for theoretical analysis and in practice to initialize $\mathbf{h}^{(0)}$ as *unit* of the operator ξ, i.e. such an element u

that $\xi(x, y, u) = \xi(x, y)\ \forall x, y \in \mathbb{R}$. Henceforth, we will assume that such an element exists (e.g., for $\xi(x, y) = \max(x, y, 0)$ we take $u = 0$, for $\xi(x, y) = xy$ we take $u = 1$), and set $\mathbf{h}^{(0)} = u$. For example, in eq. (3.9) it was implicitly assumed that $\mathbf{h}^{(0)} = 1$.

3.4.2 Grid tensors

Introduction of generalized outer product allows us to investigate RNNs with wide class of nonlinear activation functions, especially ReLU. While this change looks appealing from the practical viewpoint, it complicates following theoretical analysis, as the transition from obtained networks back to tensors is not straightforward.

In the discussion above, every tensor network had corresponding weight tensor $\boldsymbol{\mathcal{W}}$ and we could compare expressivity of associated score functions by comparing some properties of this tensors, such as ranks [Khrulkov et al., 2018, Cohen et al., 2016]. This method enabled comprehensive analysis of score functions, as it allows us to calculate and compare their values for all possible input sequences $X = \left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}\right)$. Unfortunately, we can not apply it in case of generalized tensor networks, as the replacement of standard outer product $\otimes$ with its generalized version $\otimes_\xi$ leads to the loss of conformity between tensor networks and weight tensors. Specifically, not for every generalized tensor network with corresponding score function $\ell(X)$ now exists a weight tensor $\boldsymbol{\mathcal{W}}$ such that $\ell(X) = \langle \boldsymbol{\mathcal{W}}, \boldsymbol{\Phi}(X) \rangle$. Also, such properties as *universality* no longer hold automatically and we have to prove them separately. Indeed as it was noticed in [Cohen and Shashua, 2016] shallow networks with $\xi(x, y) = \max(x, 0) + \max(y, 0)$ no longer have the universal approximation property. In order to conduct proper theoretical analysis, we adopt the apparatus of so-called *grid tensors*, first introduced in [Cohen and Shashua, 2016].

Given a set of fixed vectors $\mathbb{X} = \left\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}\right\}$ referred to as *templates*, the grid tensor of $\mathbb{X}$ is defined to be the tensor of order T and dimension M in each mode, with entries given by:

$$\boldsymbol{\Gamma}^{\ell}(\mathbb{X})_{i_1 i_2 \dots i_T} = \ell\left(X\right), \quad X = \left(\mathbf{x}^{(i_1)}, \mathbf{x}^{(i_2)}, \dots, \mathbf{x}^{(i_T)}\right), \tag{3.18}$$

where each index i_t can take values from $\{1, \dots, M\}$, i.e. we evaluate the score function on every possible input assembled from the template vectors $\{\mathbf{x}^{(i)}\}_{i=1}^M$. To put it simply, we previously considered the equality of score functions represented by tensor decomposition and tensor network on set of all possible input sequences $X = \left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}\right)$, $\mathbf{x}^{(t)} \in \mathbb{R}^N$, and now we restricted this set to exponentially large but finite grid of sequences consisting of template vectors only.

Define the matrix $\mathbf{F} \in \mathbb{R}^{M\times M}$ which holds the values taken by the representation function $f_\theta : \mathbb{R}^N \to \mathbb{R}^M$ on the selected templates $\mathbb{X}$:

$$\mathbf{F} \triangleq \left[f_\theta(\mathbf{x}^{(1)}) \quad f_\theta(\mathbf{x}^{(2)}) \quad \dots \quad f_\theta(\mathbf{x}^{(M)}) \right]^\top . \tag{3.19}$$

Using the matrix $\mathbf{F}$ we note that the grid tensor of generalized shallow network has the following form (see Section 3.8 for derivation):

$$\mathbf{\Gamma}^\ell(\mathbb{X}) = \sum_{r=1}^{R} \lambda_r \left(\mathbf{F}\mathbf{v}_r^{(1)}\right) \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(2)}\right) \otimes_\xi \dots \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(T)}\right) . \tag{3.20}$$

Construction of the grid tensor for generalized RNN is a bit more involved. We find that its grid tensor $\mathbf{\Gamma}^\ell(\mathbb{X})$ can be computed recursively, similar to the hidden state in the case of a single input sequence. The exact formulas turned out to be rather cumbersome and we moved them to Section 3.8.

3.5 Main results

With grid tensors at hand we are ready to compare the expressive power of generalized RNNs and generalized shallow networks. In the further analysis, we will assume that $\xi(x, y) = \max(x, y, 0)$, i.e., we analyze RNNs and shallow networks with *rectifier nonlinearity*. However, we need to make two additional assumptions. First of all, similarly to [Cohen and Shashua, 2016] we fix some templates $\mathbb{X}$ such that values of the score function outside of the grid generated by $\mathbb{X}$ are *irrelevant for classification* and call them *covering* templates. It was argued that for image data values of M of order 100 are sufficient (corresponding covering template vectors may represent Gabor filters). Secondly, we assume that the feature matrix $\mathbf{F}$ is invertible, which is a reasonable assumption and in the case of $f_\theta(\mathbf{x}) = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$ for any distinct template vectors $\mathbb{X}$ the parameters $\mathbf{A}$ and $\mathbf{b}$ can be chosen in such a way that the matrix $\mathbf{F}$ is invertible.

3.5.1 Universality

As was discussed in section 3.4.2 we can no longer use standard algebraic techniques to verify universality of tensor based networks. Thus, our first result states that generalized RNNs with $\xi(x, y) = \max(x, y, 0)$ are *universal* in a sense that any tensor of order T and size of each mode being m can be realized as a grid tensor of such RNN (and similarly of a generalized shallow network).

Theorem 3 (Universality)**.** *Let $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{M \times M \times \cdots \times M}$ be an arbitrary tensor of order T. Then there exist a* ***generalized shallow network*** *and a* ***generalized RNN*** *with rectifier nonlinearity $\xi(x, y) = \max(x, y, 0)$ such that grid tensor of each of the networks coincides with $\boldsymbol{\mathcal{H}}$.*

Part of Theorem 3 which corresponds to generalized shallow networks readily follows from [Cohen and Shashua, 2016, Claim 4]. In order to prove the statement for the RNNs the following two lemmas are used.

Lemma 3. *Given two generalized RNNs with grid tensors $\boldsymbol{\Gamma}^{\ell_A}(\mathbb{X})$, $\boldsymbol{\Gamma}^{\ell_B}(\mathbb{X})$, and arbitrary ξ-nonlinearity, there exists a generalized RNN with grid tensor $\boldsymbol{\Gamma}^{\ell_C}(\mathbb{X})$ satisfying*

$$\boldsymbol{\Gamma}^{\ell_C}(\mathbb{X}) = a\boldsymbol{\Gamma}^{\ell_A}(\mathbb{X}) + b\boldsymbol{\Gamma}^{\ell_B}(\mathbb{X}), \quad \forall a, b \in \mathbb{R}.$$

This lemma essentially states that the collection of grid tensors of generalized RNNs with any nonlinearity is closed under taking arbitrary linear combinations. Note that the same result clearly holds for generalized shallow networks because they are linear combinations of rank 1 shallow networks by definition.

Lemma 4. *Let $\boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}$ be an arbitrary one–hot tensor, defined as*

$$\boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}_{i_1 i_2 \ldots i_T} = \begin{cases} 1, & j_t = i_t \quad \forall t \in \{1, \ldots, T\}, \\ 0, & \textit{otherwise.} \end{cases}$$

Then there exists a generalized RNN with rectifier nonlinearities such that its grid tensor satisfies

$$\mathbf{\Gamma}^{\ell}(\mathbb{X}) = \boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}.$$

This lemma states that in the special case of rectifier nonlinearity $\xi(x, y) = \max(x, y, 0)$ any *basis* tensor can be realized by some generalized RNN.

Proof of Theorem 3. By Lemma 4 for each one–hot tensor $\boldsymbol{\mathcal{E}}^{(i_1 i_2 \ldots i_T)}$ there exists a generalized RNN with rectifier nonlinearities, such that its grid tensor coincides with this tensor. Thus, by Lemma 3 we can construct an RNN with

$$\mathbf{\Gamma}^{\ell}(\mathbb{X}) = \sum_{i_1, i_2, \ldots, i_T} \boldsymbol{\mathcal{H}}_{i_1 i_2 \ldots i_d} \boldsymbol{\mathcal{E}}^{(i_1 i_2 \ldots i_T)} = \boldsymbol{\mathcal{H}}.$$

For generalized shallow networks with rectifier nonlinearities see the proof of [Cohen and Shashua, 2016, Claim 4]. □

The same result regarding networks with product nonlinearities considered in [Khrulkov et al., 2018] directly follows from the well–known properties of tensor decompositions (see Section 3.8).

We see that at least with such nonlinearities as $\xi(x, y) = \max(x, y, 0)$ and $\xi(x, y) = xy$ all the networks under consideration are universal and can represent any possible grid tensor. Now let us head to a discussion of *expressivity* of these networks.

3.5.2 Expressivity

As was discussed in the introduction, expressivity refers to the ability of some class of networks to represent the same functions as some other class much more compactly. In our case the parameters defining *size* of networks are *ranks* of the decomposition, i.e. in the case of generalized RNNs ranks determine the size of the hidden state, and in the case of generalized shallow networks rank determines the width of a network. It was proven in [Cohen et al., 2016, Khrulkov et al., 2018] that ConvNets and RNNs with multiplicative nonlinearities are *exponentially* more expressive than the equivalent shallow networks: shallow networks of exponentially large width are required to realize the same score functions as computed by these deep architectures. Similarly to the case of ConvNets [Cohen and Shashua, 2016], we find that expressivity of generalized RNNs with rectifier nonlinearity holds only partially, as discussed in the following two theorems. For simplicity, we assume that T is even.

Theorem 4 (Expressivity I)**.** *For every value of R there exists a generalized RNN with ranks $\leq R$ and rectifier nonlinearity which is exponentially more efficient than shallow networks, i.e., the corresponding grid tensor may be realized only by a shallow network with rectifier nonlinearity of width at least $\frac{2}{MT}\min(M,R)^{T/2}$.*

This result states that at least for some subset of generalized RNNs expressivity holds: exponentially wide shallow networks are required to realize the same grid tensor. Proof of the theorem is rather straightforward: we explicitly construct an example of such RNN which satisfies the following description. Given an arbitrary input sequence $X = \left(\mathbf{x}^{(1)}, \dots \mathbf{x}^{(T)}\right)$ assembled from the templates, these networks (if $M = R$) produce 0 if X has the property that $\mathbf{x}^{(1)} = \mathbf{x}^{(2)}, \mathbf{x}^{(3)} = \mathbf{x}^{(4)}, \dots, \mathbf{x}^{(T-1)} = \mathbf{x}^{(T)}$, and 1 in every other case, i.e. they measure *pairwise similarity* of the input vectors. A precise proof is given in Section 3.8.
In the case of multiplicative RNNs [Khrulkov et al., 2018] *almost every* network possessed this property. This is not the case, however, for generalized RNNs with rectifier nonlinearities.

Theorem 5 (Expressivity II)**.** *For every value of R there exists an open set (which thus has positive measure) of generalized RNNs with rectifier nonlinearity $\xi(x,y) = \max(x,y,0)$, such that for each RNN in this open set the corresponding grid tensor can be realized by a rank 1 shallow network with rectifier nonlinearity.*

In other words, for every rank R we can find a set of generalized RNNs of positive measure such that the property of expressivity does not hold. In the numerical experiments in Section 6.5 and Section 3.8 we validate whether this can be observed in practice, and find that the probability of obtaining CP–ranks of polynomial size becomes negligible with large T and R. Proof of Theorem 5 is provided in Section 3.8.

Shared case Note that all the RNNs used in practice have *shared weights*, which allows them to process sequences of arbitrary length. So far in the analysis we have not made such assumptions about RNNs (i.e., $\boldsymbol{\mathcal{G}}^{(2)} = \cdots = \boldsymbol{\mathcal{G}}^{(T-1)}$). By imposing this constraint, we lose the property of universality; however, we believe that the statements of Theorems 4 and 5 still hold (without requiring that shallow networks also have shared weights). Note that the example constructed in the proof of Theorem 5 already has this property, and for Theorem 4 we provide numerical evidence in Section 3.8.

3.6 Experiments

In this section, we study if our theoretical findings are supported by experimental data. In particular, we investigate whether generalized tensor networks can be used in practical settings, especially in problems typically solved by RNNs (such as natural language processing problems). Secondly, according to Theorem 5 for some subset of RNNs the equivalent shallow network may have a low rank. To get a grasp of how strong this effect might be in practice we numerically compute an estimate for this rank in various settings.

Performance For the first experiment, we use two computer vision datasets MNIST [LeCun et al., 1990] and CIFAR–10 [Krizhevsky and Hinton, 2009], and natural language processing dataset for sentiment analysis IMDB [Maas et al., 2011]. For the first two datasets, we cut natural images into rectangular patches which are then arranged into vectors $\mathbf{x}^{(t)}$ (similar to [Khrulkov et al., 2018]) and for IMDB dataset the input data already has the desired sequential structure.

Figure 3-2 depicts test accuracy on IMDB dataset for generalized shallow networks and RNNs with rectifier nonlinearity. We see that generalized shallow network of much higher rank is required to get the level of performance close to that achievable by generalized RNN. Due to limited space, we have moved the results of the experiments on the visual datasets to Section 3.9.

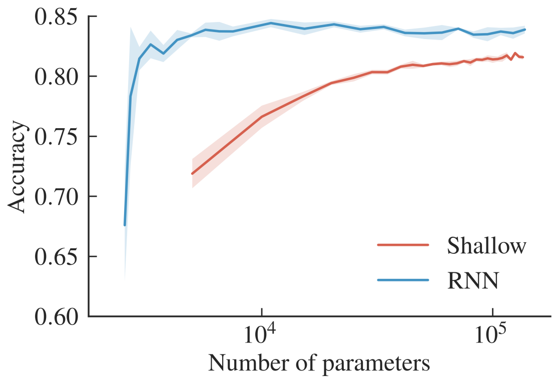


Figure 3-2: Test accuracy on IMDB dataset for generalized RNNs and generalized shallow networks with respect to the total number of parameters ($M = 50$, $T = 100$, $\xi(x, y) = \max(x, y, 0)$).

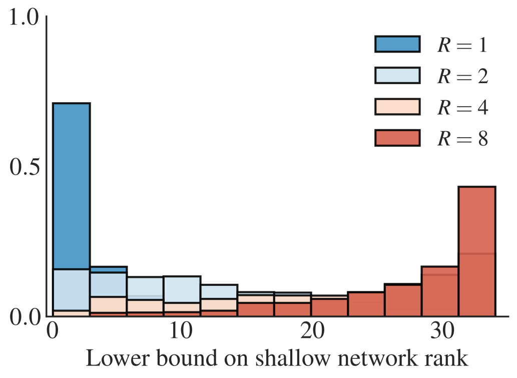


Figure 3-3: Distribution of lower bounds on the rank of generalized shallow networks equivalent to randomly generated generalized RNNs of ranks $1, 2, 4, 8$ ($M = 10$, $T = 6$).

Expressivity For the second experiment we generate a number of generalized RNNs with different values of TT-rank r and calculate a lower bound on the rank of shallow network necessary to realize the same grid tensor (to estimate the rank we use the same technique as in the proof of Theorem 4). Figure 3-3 shows that for different values of R and generalized RNNs of the corresponding rank there exist shallow networks of rank 1 realizing the same grid tensor, which agrees well with Theorem 5. This result looks discouraging, however, there is also a positive observation. While increasing rank of generalized RNNs, more and more corresponding shallow networks will necessarily have exponentially higher rank. In practice we usually deal with RNNs of $R = 10^2 - 10^3$ (dimension of hidden states), thus we may expect that effectively any function besides negligible set realized by generalized RNNs can be implemented only by exponentially wider shallow networks. The numerical results for the case of shared cores and other nonlinearities are given in Section 3.9.

3.7 Conclusion

In this paper, we sought a more complete picture of the connection between Recurrent Neural Networks and Tensor Train decomposition, one that involves various nonlinearities applied to hidden states. We showed how these nonlinearities could be incorporated into network architectures and provided complete theoretical analysis on the particular case of rectifier nonlinearity, elaborating on points of generality and expressive power. We believe our results will be useful to advance theoretical understanding of RNNs. In future work, we would like to extend the theoretical analysis to most competitive in practice architectures for processing sequential data such as LSTMs and attention mechanisms.

3.8 Proofs

Lemma 2. *Under the notation introduced in eq. (3.9), the score function can be written as*

$$\ell(X) = \mathbf{h}^{(T)} \in \mathbb{R}^1.$$

Proof.

$$\begin{aligned}
l(X) &= \sum_{r_1=1}^{R_1} \dots \sum_{r_{T-1}=1}^{R_{T-1}} \prod_{t=1}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \\
&= \sum_{r_1=1}^{R_1} \dots \sum_{r_{T-1}=1}^{R_{T-1}} \prod_{t=2}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \underbrace{\langle f_\theta(\mathbf{x}^{(1)}), \mathbf{g}^{(1)}_{r_0 r_1} \rangle}_{\mathbf{h}^{(1)}_{r_1}} \\
&= \sum_{r_{T-1}=1}^{R_{T-1}} \dots \sum_{r_1=1}^{R_1} \prod_{t=2}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \mathbf{h}^{(1)}_{r_1} \\
&= \sum_{r_{T-1}=1}^{R_{T-1}} \dots \sum_{r_2=1}^{R_2} \prod_{t=3}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \underbrace{\sum_{r_1=1}^{r_1} \langle f_\theta(\mathbf{x}^{(2)}), \mathbf{g}^{(2)}_{r_1 r_2} \rangle \mathbf{h}^{(1)}_{r_1}}_{\mathbf{h}^{(2)}_{r_2}} \\
&= \sum_{r_{T-1}=1}^{R_{T-1}} \dots \sum_{r_2=1}^{R_2} \prod_{t=3}^{T} \langle f_\theta(\mathbf{x}^{(t)}), \mathbf{g}^{(t)}_{r_{t-1} r_t} \rangle \mathbf{h}^{(2)}_{r_2} \\
&= \dots \\
&= \sum_{r_{T-1}=1}^{R_{T-1}} \langle f_\theta(\mathbf{x}^{(T)}), \mathbf{g}^{(T)}_{r_{T-1} r_T} \rangle \mathbf{h}^{(T-1)}_{r_{T-1}} = \mathbf{h}^{(T)}_{r_T} = \mathbf{h}^{(T)}.
\end{aligned}$$

□

Proposition 3.8.1. *If we replace the generalized outer product $\otimes_\xi$ in eq. (3.16) with the standard outer product $\otimes$, we can subsume matrices $\mathbf{C}^{(t)}$ into tensors $\boldsymbol{\mathcal{G}}^{(t)}$*

without loss of generality.

Proof. Let us rewrite hidden state equation eq. (3.16) after transition from $\otimes_\xi$ to $\otimes$:

$$\begin{aligned}
\mathbf{h}_k^{(t)} &= \sum_{i,j} \boldsymbol{\mathcal{G}}_{ijk}^{(t)} \left[\mathbf{C}^{(t)} f_\theta(\mathbf{x}^{(t)}) \otimes \mathbf{h}^{(t-1)}\right]_{ij} \\
&= \sum_{i,j} \boldsymbol{\mathcal{G}}_{ijk}^{(t)} \sum_l \mathbf{C}_{il}^{(t)} f_\theta(\mathbf{x}^{(t)})_l \mathbf{h}_j^{(t-1)} \qquad \left\{ \tilde{\boldsymbol{\mathcal{G}}}_{ljk}^{(t)} = \sum_i \boldsymbol{\mathcal{G}}_{ijk}^{(t)} \mathbf{C}_{il}^{(t)} \right\} \\
&= \sum_{l,j} \tilde{\boldsymbol{\mathcal{G}}}_{ljk}^{(t)} f_\theta(\mathbf{x}^{(t)})_l \mathbf{h}_j^{(t-1)} \\
&= \sum_{l,j} \tilde{\boldsymbol{\mathcal{G}}}_{ljk}^{(t)} \left[f_\theta(\mathbf{x}^{(t)}) \otimes \mathbf{h}^{(t-1)}\right]_{lj}.
\end{aligned}$$

We see that the obtained expression resembles those presented in eq. (3.10) with TT-cores $\boldsymbol{\mathcal{G}}^{(t)}$ replaced by $\tilde{\boldsymbol{\mathcal{G}}}^{(t)}$ and thus all the reasoning applied in the absence of matrices $\mathbf{C}^{(t)}$ holds valid. □

Proposition 3.8.2. *Grid tensor of generalized shallow network has the following form (eq. (3.20)):*

$$\boldsymbol{\Gamma}^\ell(\mathbb{X}) = \sum_{r=1}^R \lambda_r \left(\mathbf{F}\mathbf{v}_r^{(1)}\right) \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(2)}\right) \otimes_\xi \ldots \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(T)}\right).$$

Proof. Let $X = \left(\mathbf{x}^{(i_1)}, \mathbf{x}^{(i_2)}, \ldots, \mathbf{x}^{(i_T)}\right)$ denote an arbitrary sequence of *templates*. Corresponding element of the grid tensor defined in eq. (3.20) has the following form:

$$\begin{aligned}
\boldsymbol{\Gamma}^\ell(\mathbb{X})_{i_1 i_2 \ldots i_T} &= \sum_{r=1}^R \lambda_r \left[\left(\mathbf{F}\mathbf{v}_r^{(1)}\right) \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(2)}\right) \otimes_\xi \ldots \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(T)}\right)\right]_{i_1 i_2 \ldots i_T} \\
&= \sum_{r=1}^R \lambda_r \left(\mathbf{F}\mathbf{v}_r^{(1)}\right)_{i_1} \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(2)}\right)_{i_2} \otimes_\xi \ldots \otimes_\xi \left(\mathbf{F}\mathbf{v}_r^{(T)}\right)_{i_T} \\
&= \sum_{r=1}^R \lambda_r \xi \left(\langle f_\theta(\mathbf{x}^{(i_1)}), \mathbf{v}_r^{(1)} \rangle, \ldots, \langle f_\theta(\mathbf{x}^{(i_T)}), \mathbf{v}_r^{(T)} \rangle\right) = \ell(X).
\end{aligned}$$

□

Proposition 3.8.3. *Grid tensor of a generalized RNN has the following form:*

$$\begin{aligned}
&\boldsymbol{\Gamma}^{\ell,0}(\mathbb{X}) = \mathbf{h}^{(0)} \in \mathbb{R}^1,\\
&\boldsymbol{\Gamma}^{\ell,1}(\mathbb{X})_{km_1} = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(1)}_{ijk} \left(\mathbf{C}^{(1)}\mathbf{F}^\top \otimes_\xi \boldsymbol{\Gamma}^{\ell,0}\right)_{im_1j} \in \mathbb{R}^{R_1 \times M},\\
&\boldsymbol{\Gamma}^{\ell,2}(\mathbb{X})_{km_1m_2} = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(2)}_{ijk} \left(\mathbf{C}^{(2)}\mathbf{F}^\top \otimes_\xi \boldsymbol{\Gamma}^{\ell,1}\right)_{im_2jm_1} \in \mathbb{R}^{R_2 \times M \times M},\\
&\cdots\\
&\boldsymbol{\Gamma}^{\ell,T}(\mathbb{X})_{km_1m_2\dots m_T} = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(T)}_{ijk} \left(\mathbf{C}^{(T)}\mathbf{F}^\top \otimes_\xi \boldsymbol{\Gamma}^{\ell,T-1}\right)_{im_Tjm_1\dots m_{T-1}} \in \mathbb{R}^{1 \times M \times M \times \dots \times M},\\
&\boldsymbol{\Gamma}^{\ell}(\mathbb{X}) = \boldsymbol{\Gamma}^{\ell,T}(\mathbb{X})_{1,:,:,\dots,:}
\end{aligned} \tag{3.21}$$

Proof. Proof is similar to that of Proposition 3.8.2 and uses eq. (3.16) to compute the elements of the grid tensor. □

Lemma 3. *Given two generalized RNNs with grid tensors $\boldsymbol{\Gamma}^{\ell_A}(\mathbb{X})$, $\boldsymbol{\Gamma}^{\ell_B}(\mathbb{X})$, and arbitrary ξ-nonlinearity, there exists a generalized RNN with grid tensor $\boldsymbol{\Gamma}^{\ell_C}(\mathbb{X})$ satisfying*

$$\boldsymbol{\Gamma}^{\ell_C}(\mathbb{X}) = a\boldsymbol{\Gamma}^{\ell_A}(\mathbb{X}) + b\boldsymbol{\Gamma}^{\ell_B}(\mathbb{X}), \quad \forall a, b \in \mathbb{R}.$$

Proof. Let these RNNs be defined by the weight parameters

$$\Theta_A = \left(\{\mathbf{C}^{(t)}_A\}_{t=1}^T \in \mathbb{R}^{L_A \times M}, \{\boldsymbol{\mathcal{G}}^{(t)}_A\}_{t=1}^T \in \mathbb{R}^{L_A \times R_{t-1,A} \times R_{t,A}}\right),$$

and

$$\Theta_B = \left(\{\mathbf{C}^{(t)}_B\}_{t=1}^T \in \mathbb{R}^{L_B \times M}, \{\boldsymbol{\mathcal{G}}^{(t)}_B\}_{t=1}^T \in \mathbb{R}^{L_B \times R_{t-1,B} \times R_{t,B}}\right).$$

We claim that the desired grid tensor is given by the RNN with the following weight settings.

$$
\begin{aligned}
\mathbf{C}_C^{(t)} &\in \mathbb{R}^{(L_A+L_B)\times M} \\
\mathbf{C}_C^{(t)} &= \begin{bmatrix} \mathbf{C}_A^{(t)} \\ \mathbf{C}_B^{(t)} \end{bmatrix} \\
\boldsymbol{\mathcal{G}}_C^{(1)} &\in \mathbb{R}^{(L_A+L_B)\times 1\times (R_{t,A}+R_{t,B})} \\
[\boldsymbol{\mathcal{G}}_C^{(1)}]_{i,:,:} &= \begin{cases} \begin{bmatrix} [\boldsymbol{\mathcal{G}}_A^{(1)}]_{i,:,:} & 0 \end{bmatrix}, & i \in \{1,\dots,L_A\} \\ \\ \begin{bmatrix} 0 & [\boldsymbol{\mathcal{G}}_B^{(1)}]_{(i-L_A),:,:} \end{bmatrix}, & i \in \{L_A+1,\dots,L_A+L_B\} \end{cases} \\
\boldsymbol{\mathcal{G}}_C^{(t)} &\in \mathbb{R}^{(L_A+L_B)\times (R_{t-1,A}+R_{t-1,B})\times (R_{t,A}+R_{t,B})}, \quad 1 < t < T \\
[\boldsymbol{\mathcal{G}}_C^{(t)}]_{i,:,:} &= \begin{cases} \begin{bmatrix} [\boldsymbol{\mathcal{G}}_A^{(t)}]_{i,:,:} & 0 \\ 0 & 0 \end{bmatrix}, & i \in \{1,\dots,L_A\} \\ \\ \begin{bmatrix} 0 & 0 \\ 0 & [\boldsymbol{\mathcal{G}}_B^{(t)}]_{(i-L_A),:,:} \end{bmatrix}, & i \in \{L_A+1,\dots,L_A+L_B\} \end{cases} \\
\boldsymbol{\mathcal{G}}_C^{(T)} &\in \mathbb{R}^{(L_A+L_B)\times (R_{t-1,A}+R_{t-1,B})\times 1} \\
[\boldsymbol{\mathcal{G}}_C^{(T)}]_{i,:,:} &= \begin{cases} \begin{bmatrix} a[\boldsymbol{\mathcal{G}}_A^{(T)}]_{i,:,:} \\ 0 \end{bmatrix}, & i \in \{1,\dots,L_A\} \\ \\ \begin{bmatrix} 0 \\ b[\boldsymbol{\mathcal{G}}_B^{(T)}]_{(i-L_A),:,:} \end{bmatrix}, & i \in \{L_A+1,\dots,L_A+L_B\}. \end{cases}
\end{aligned}
$$

It is straightforward to verify that the network defined by these weights possesses the following property:

$$
\mathbf{h}_C^{(t)} = \begin{bmatrix} \mathbf{h}_A^{(t)} \\ \mathbf{h}_B^{(t)} \end{bmatrix}, \quad 0 < t < T,
$$

and

$$\mathbf{h}_C^{(T)} = a\mathbf{h}_A^{(T)} + b\mathbf{h}_B^{(T)},$$

concluding the proof. We also note that these formulas generalize the well–known formulas for addition of two tensors in the Tensor Train format [Oseledets, 2011]. □

Proposition 3.8.4. *For any associative and commutative binary operator ξ, an arbitrary generalized rank* 1 *shallow network with ξ–nonlinearity can be represented in a form of generalized RNN with unit ranks ($R_1 = \cdots = R_{T-1} = 1$) and ξ–nonlinearity.*

Proof. Let $\Theta = \left(\lambda, \{\mathbf{v}^{(t)}\}_{t=1}^{T}\right)$ be the parameters specifying the given generalized shallow network. Then the following weight settings provide the equivalent generalized RNN (with $\mathbf{h}^{(0)}$ being the unity of the operator ξ).

$$\begin{aligned}
&\mathbf{C}^{(t)} = \left(\mathbf{v}^{(t)}\right)^{\top} \in \mathbb{R}^{1\times M},\\
&\boldsymbol{\mathcal{G}}^{(t)} = 1, \quad t < T,\\
&\boldsymbol{\mathcal{G}}^{(T)} = \lambda.
\end{aligned}$$

Indeed, in the notation defined above, hidden states of generalized RNN have the following form:

$$\begin{aligned}
\mathbf{h}^{(t)} &= \boldsymbol{\mathcal{G}}^{(t)}\xi\left([\mathbf{C}^{(t)} f_\theta(\mathbf{x}^{(t)})], \mathbf{h}^{(t-1)}\right)\\
&= \xi\left(\langle f_\theta(\mathbf{x}^{(t)}), \mathbf{v}^{(t)}\rangle, \mathbf{h}^{(t-1)}\right), \quad t = 1, \ldots, T-1\\
\mathbf{h}^{(T)} &= \lambda\xi\left(\langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}^{(T)}\rangle, \mathbf{h}^{(T-1)}\right).
\end{aligned}$$

The score function of generalized RNN is given by eq. (3.16):

$$\begin{aligned}
\ell(X) = \mathbf{h}^{(T)} &= \lambda\xi\left(\langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}^{(T)}\rangle, \mathbf{h}^{(T-1)}\right)\\
&= \lambda\xi\left(\langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}^{(T)}\rangle, \langle f_\theta(\mathbf{x}^{(T-1)}), \mathbf{v}^{(T-1)}\rangle, \mathbf{h}^{(T-2)}\right)\\
&\ldots\\
&= \lambda\xi\left(\langle f_\theta(\mathbf{x}^{(T)}), \mathbf{v}^{(T)}\rangle, \ldots, \langle f_\theta(\mathbf{x}^{(1)}), \mathbf{v}^{(1)}\rangle\right),
\end{aligned}$$

which coincides with the score function of rank 1 shallow network defined by parameters Θ.

□

Lemma 4. *Let $\boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}$ be an arbitrary one–hot tensor, defined as*

$$\boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}_{i_1 i_2 \ldots i_T} = \begin{cases} 1, & j_t = i_t \quad \forall t \in \{1, \ldots, T\}, \\ 0, & \textit{otherwise.} \end{cases}$$

Then there exists a generalized RNN with rectifier nonlinearities such that its grid tensor satisfies

$$\boldsymbol{\Gamma}^{\ell}(\mathbb{X}) = \boldsymbol{\mathcal{E}}^{(j_1 j_2 \ldots j_T)}.$$

Proof. It is known that the statement of the lemma holds for generalized shallow networks with rectifier nonlinearities (see [Cohen and Shashua, 2016, Claim 4]). Based on Proposition 3.8.4 and Lemma 3 we can conclude that it also holds for generalized RNNs with rectifier nonlinearities. □

Proposition 3.8.5. *Statement of Theorem 3 holds with $\xi(x, y) = xy$.*

Proof. By assumption the matrix $\mathbf{F}$ is invertible. Consider the following tensor $\widehat{\boldsymbol{\mathcal{H}}}$:

$$\widehat{\boldsymbol{\mathcal{H}}}_{i_1 i_2 \ldots i_T} = \sum_{j_1, \ldots, j_T} \boldsymbol{\mathcal{H}}_{j_1, \ldots, j_T} \mathbf{F}^{-1}_{j_1 i_1} \ldots \mathbf{F}^{-1}_{j_T i_T},$$

and the score function in the form of eq. (3.2):

$$\ell(X) = \langle \widehat{\boldsymbol{\mathcal{H}}}, \boldsymbol{\Phi}(X) \rangle.$$

Note that by construction for any input assembled from the template vectors we obtain $\ell\left((\mathbf{x}^{(i_1)}, \ldots, \mathbf{x}^{(i_T)})\right) = \boldsymbol{\mathcal{H}}_{i_1 \ldots i_T}$. By taking the standard TT and CP decompositions of $\widehat{\boldsymbol{\mathcal{H}}}$ which always exist [Oseledets, 2011], and using Lemma 2 and eq. (3.6) we conclude that universality holds. □

Theorem 4 (Expressivity I)**.** *For every value of R there exists a generalized RNN with ranks $\leq R$ and rectifier nonlinearity which is exponentially more efficient than shallow networks, i.e., the corresponding grid tensor may be realized only by a shallow network with rectifier nonlinearity of width at least $\frac{2}{MT} \min(M, R)^{T/2}$.*

In order to prove the theorem we will use the standard technique of matricizations. Simply put, by matricizing a tensor we reshape it into a matrix by splitting the indices of a tensor into two collections, and converting each one of them into one long index. I.e., for a tensor $\boldsymbol{\mathcal{A}}$ of order T with mode sizes being m, we split the set $\{1, \dots, T\}$ into two non–overlapping ordered subsets s and t, and define the matricization $\mathbf{A}^{(s,t)} \in \mathbb{R}^{M^{|s|} \times M^{|t|}}$ by simply reshaping (and possibly transposing) the tensor $\boldsymbol{\mathcal{A}}$ according to s and t. We will consider the matricization obtained by taking $s_{odd} = (1, 3, \dots, T-1)$, $t_{even} = (2, 4, \dots, T)$, i.e., we split out even and odd modes. A typical application of matricization is the following: suppose that we can upper and lower bound the ordinary matrix rank of a certain matricization using the parameters specifying each of the architectures being analyzed. Then under the assumption that both architectures realize the same grid tensor (and thus ranks of the matricization coincide) we can compare the sizes of corresponding architectures. In the case of generalized shallow networks with rectifier nonlinearity we will use the following result [Cohen and Shashua, 2016, Claim 9].

Lemma 5. *Let $\boldsymbol{\Gamma}^{\ell}(\mathbb{X})$ be a grid tensor generated by a generalized shallow network of rank R and $\xi(x, y) = \max(x, y, 0)$. Then*

$$\operatorname{rank} \left[\boldsymbol{\Gamma}^{\ell}(\mathbb{X})\right]^{(s_{odd}, t_{even})} \leq R\frac{TM}{2},$$

where the ordinary matrix rank is assumed.

This result is a generalization of a well–known property of the standard CP-decomposition (i.e. if $\xi(x, y) = xy$), which states that for a rank R decomposition, the matrix rank of *every* matricization is bounded by R.

In order to prove Theorem 4 we will construct an example of a generalized RNN with exponentially large matrix rank of the matricization of grid tensor, from which and Lemma 5 the statement of the theorem will follow.

Lemma 6. *Without loss of generality assume that $\mathbf{x}_i = \mathbf{e}_i$ (which can be achieved since $\mathbf{F}$ is invertible). Let $\mathbf{1}^{(p,q)}$ denote the matrix of size $p \times q$ with each entry being 1, $\mathbf{I}^{(p,q)}$ denote the matrix of size $p \times q$ with $\mathbf{I}^{(p,q)}_{ij} = \delta_{ij}$ (δ being the Kronecker* *symbol), and* $\mathbf{b} = [1 - \min(M, R), \mathbf{0}_{R-1}^\top] \in \mathbb{R}^{1\times R}$. *Consider the following weight setting for a generalized RNN with* $\xi(x, y) = \max(x, y, 0)$.

$$\mathbf{C}^{(t)} = \begin{cases} \mathbf{1}^{M,M} - \mathbf{I}^{M,M}, \ t \text{ odd}, \\ \mathbf{1}^{M+1,M} - \mathbf{I}^{M+1,M}, \ t \text{ even}. \end{cases}$$

$$\boldsymbol{\mathcal{G}}^{(t)} = \begin{cases} \mathbf{I}^{M,R} \in \mathbb{R}^{M\times 1\times R}, \ t \text{ odd}, \\ \begin{bmatrix} \mathbf{I}^{M,R} \\ \mathbf{b} \end{bmatrix} \in \mathbb{R}^{(M+1)\times R\times 1}, \ t \text{ even}. \end{cases}$$

Then grid tensor $\boldsymbol{\Gamma}^\ell(\mathbb{X})$ *of this RNN satisfies*

$$\operatorname{rank} \left[\boldsymbol{\Gamma}^\ell(\mathbb{X})\right]^{(s_{odd}, t_{even})} \geq \min(M, R)^{T/2},$$

where the ordinary matrix rank is assumed.

Proof. Informal description of the network defined by weights in the statement in the lemma is the following. Given some input vector $\mathbf{e}_i$ it is first transformed into its bitwise negative $\overline{\mathbf{e}}_i$, and its first R components are saved into the hidden state. The next block then measures whether the first $\min(R, M)$ components of the current input coincide with the hidden state (after again taking bitwise negative). If this is the case, the hidden state is set 0 and the process continues. Otherwise, the hidden state is set to 1 which then flows to the output independently of the other inputs. In other words, for all the inputs of the form $X = (\mathbf{x}_{i_1}, \mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{T/2}}, \mathbf{x}_{i_{T/2}})$ with $i_1 \leq R, \ldots, i_{T/2} \leq R$ we obtain that $\ell(X) = 0$, and in every other case $\ell(X) = 1$. Thus, we obtain that $\left[\boldsymbol{\Gamma}^\ell(\mathbb{X})\right]^{(s_{odd}, t_{even})}$ is a matrix with all the entries equal to 1, except for $\min(M, R)^{T/2}$ entries on the diagonal, which are equal to 0. Rank of such a matrix is $R^{T/2} + 1$ if $R < M$ and $M^{T/2}$ otherwise, and the statement of the lemma follows. □

Based on these two lemmas we immediately obtain Theorem 4.

Proof of Theorem 4. Consider the example constructed in the proof of Lemma 6. By Lemma 5 the rank of the shallow network with rectifier nonlinearity which is able to represent the same grid tensor is at least $\frac{2}{TM}\min(M,R)^{T/2}$. □

Theorem 5 (Expressivity II). *For every value of R there exists an open set (which thus has positive measure) of generalized RNNs with rectifier nonlinearity $\xi(x,y) = \max(x,y,0)$, such that for each RNN in this open set the corresponding grid tensor can be realized by a rank 1 shallow network with rectifier nonlinearity.*

Proof. As before, let us denote by $\mathbf{I}^{(p,q)}$ a matrix of size $p \times q$ such that $\mathbf{I}^{(p,q)}_{ij} = \delta_{ij}$, and by $\mathbf{a}^{(p_1,p_2,\dots p_d)}$ we denote a tensor of size $p_1 \times \dots \times p_d$ with each entry being a (sometimes we will omit the dimensions when they can be inferred from the context). Consider the following weight settings for a generalized RNN.

$$\mathbf{C}^{(t)} = \left(\mathbf{F}^\top\right)^{-1},$$

$$\boldsymbol{\mathcal{G}}^{(t)} = \begin{cases} \mathbf{2}^{(M,1,R)}, & t = 1 \\ \mathbf{1}^{(M,R,R)}, & t = 2,\dots,T-1 \\ \mathbf{1}^{(M,R,1)}, & t = T \end{cases}$$

The RNN defined by these weights has the property that $\boldsymbol{\Gamma}^\ell(\mathbb{X})$ is a constant tensor with each entry being $2(MR)^{T-1}$, which can be trivially represented by a rank 1 generalized shallow network. We will show that this property holds under a small perturbation of $\mathbf{C}^{(t)}, \boldsymbol{\mathcal{G}}^{(t)}$ and $\mathbf{F}$. Let us denote each of these perturbation (and every tensor appearing size of which can be assumed indefinitely small) collectively by $\boldsymbol{\varepsilon}$. Applying eq. (3.21) we obtain (with $\xi(x,y) = \max(x,y,0)$).

$$\begin{aligned}
&\boldsymbol{\Gamma}^{\ell,0}(\mathbb{X}) = \mathbf{0} \in \mathbb{R}^1, \\
&\boldsymbol{\Gamma}^{\ell,1}(\mathbb{X})_{km_1} = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(1)}_{ijk} \left((\mathbf{I}^{(M,M)} + \boldsymbol{\varepsilon}) \otimes_\xi \mathbf{0}\right)_{im_1j} = \mathbf{1} \otimes (\mathbf{2} + \boldsymbol{\varepsilon}), \\
&\boldsymbol{\Gamma}^{\ell,2}(\mathbb{X})_{km_1m_2} = \sum_{i,j} \boldsymbol{\mathcal{G}}^{(2)}_{ijk} \left((\mathbf{I}^{(M,M)} + \boldsymbol{\varepsilon}) \otimes_\xi \boldsymbol{\Gamma}^{\ell,1}(\mathbb{X})\right)_{im_2jm_1} = \mathbf{1} \otimes (\mathbf{2MR} + \boldsymbol{\varepsilon}) \otimes \mathbf{1}, \\
&\dots \\
&\boldsymbol{\Gamma}^{\ell,T}(\mathbb{X})_{km_1m_2\dots m_T} = 1 \otimes (\mathbf{2(MR)^{T-1}} + \boldsymbol{\varepsilon}) \otimes \mathbf{1} \dots \otimes \mathbf{1}, \\
&\boldsymbol{\Gamma}^{\ell}(\mathbb{X}) = \boldsymbol{\Gamma}^{\ell,T}(\mathbb{X})_{1,:,:,\dots,:} = (\mathbf{2(MR)^{T-1}} + \boldsymbol{\varepsilon}) \otimes \mathbf{1} \dots \otimes \mathbf{1},
\end{aligned}$$

where we have used a simple property connecting $\otimes_\xi$ with $\xi(x, y) = \max(x, y, 0)$ and ordinary $\otimes$: if for tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ each entry of $\boldsymbol{\mathcal{A}}$ is greater than each entry of $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{A}} \otimes_\xi \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \otimes \mathbf{1}$. The obtained grid tensors can be represented using rank 1 generalized shallow networks with the following weight settings.

$$\begin{aligned} &\lambda = 1, \\ &\mathbf{v}_t = \begin{cases} \mathbf{F}_{\boldsymbol{\varepsilon}}^{-1}(\mathbf{2}(\mathbf{MR})^{\mathbf{T-1}} + \boldsymbol{\varepsilon}), \ t = 1, \\ \mathbf{0}, \ t > 1, \end{cases} \end{aligned}$$

where $\mathbf{F}_{\boldsymbol{\varepsilon}}$ is the feature matrix of the corresponding perturbed network. □

3.9 Additional experiments

In this section we provide the results additional computational experiments, aimed to provide more thorough and complete analysis of generalized RNNs.

Different $\boldsymbol{\xi}$-nonlinearities In this paper we presented theoretical analysis of rectifier nonlinearity which corresponds to $\xi(x, y) = \max(x, y, 0)$. However, there is a number of other associative binary operators ξ which can be incorporated in generalized tensor networks. Strictly speaking, every one of them has to be carefully explored theoretically in order to speak about their generality and expressive power, but for now we can compare them empirically.

$\xi(x, y)$	xy	$\max(x, y, 0)$	$\ln(e^x + e^y)$	$x + y$	$\sqrt{x^2 + y^2}$
MNIST	97.39	97.45	**97.68**	96.28	96.44
CIFAR-10	43.08	48.09	55.37	**57.18**	49.04
IMDB	83.33	**84.35**	82.25	81.28	79.76

Table 3.1: Performance of generalized RNN with various nonlinearities.

Table 3.1 shows the performance (accuracy on test data) of different nonlinearities on MNIST, CIFAR—10, and IMDB datasets for classification. Although these problems are not considered hard to solve, we see that the right choice of nonlinearity can lead to a significant boost in performance. For the experiments on the visual datasets we used $T = 16, m = 32, R = 64$ and for the experiments on the IMDB dataset we had $T = 100, m = 50, R = 50$. Parameters of all networks were optimized using Adam (learning rate $\alpha = 10^{-4}$) and batch size 250.

Expressivity in the case of shared cores We repeat the expressivity experiments from Section 6.5 in the case of equal TT–cores ($\boldsymbol{\mathcal{G}}^{(2)} = \cdots = \boldsymbol{\mathcal{G}}^{(T-1)}$). We observe that similar to the case of different cores, there always exist rank 1 generalized shallow networks which realize the same score function as generalized RNN of higher rank, however, this situation seems too unlikely for big values of R.

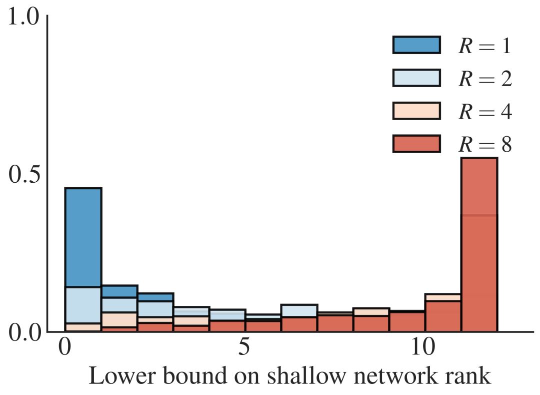


Figure 3-4: Distribution of lower bounds on the rank of generalized shallow networks equivalent to randomly generated generalized RNNs of ranks ($M = 6$, $T = 6$, $\xi(x, y) = \max(x, y, 0)$).

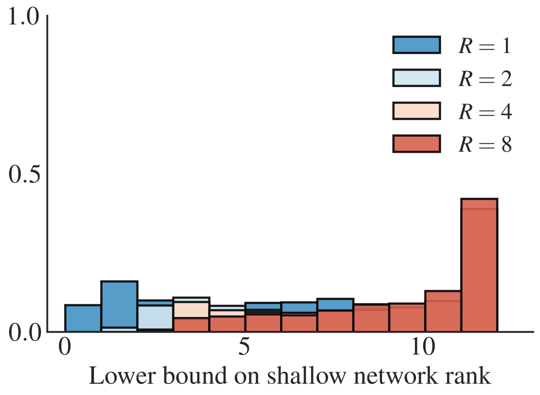


Figure 3-5: Distribution of lower bounds on the rank of generalized shallow networks equivalent to randomly generated generalized RNNs of ranks ($M = 6$, $T = 6$, $\xi(x, y) = \sqrt{x^2 + y^2}$).

Chapter 4

Desingularization of bounded-rank matrix sets

4.1 Introduction

Although low-rank matrices appear in many applications, the structure of the corresponding matrix variety (real algebraic) is not fully utilized in the computations, and the theoretical investigation is complicated because of the existence of singular points [Lakshmibai and Brown, 2015] on such a variety, which correspond to matrices of smaller rank. We tackle this problem by utilizing the modified Room-Kempf desingularization [Naldi, 2015] of determinantal varieties that is classical in algebraic geometry, but has never been applied in the context of optimization over matrix varieties. Briefly, it can be summarized as follows. Idea of the the Room-Kempf procedure is to consider a set of tuples of matrices (A, Y) satisfying equations $AY = 0$ and $BY = 0$ for some fixed matrix B. These equations imply that the rank of A is bounded and moreover a set of such tuples is a smooth manifold (for reasonable matrices B). However, conditions of the form $BY = 0$ can be numerically unstable, so we modify it by imposing the condition $Y^T Y = I$ instead. The precise definition of the manifold we work with is given in terms of Grassmannians and then we transition to the formulas given above. We also show that the dimension of this manifold is the same as of the original matrix variety. Our main contributions are:

- We propose and analyze a modified Room-Kempf desingularization technique for the variety of matrices of shape $n \times m$ with rank bounded by r (section 4.2.2).
- We prove smoothness and obtain bounds on the curvature of the desingularized variety in section 4.2.2 and section 4.2.3. The latter is performed by estimating singular values of the operator of the orthogonal projection onto the tangent space of the desingularized variety.
- We find an effective low-dimensional parametrization of the tangent space (section 4.2.4). Even though the desingularized variety is a subset of a space of much bigger dimension, this allows us to construct robust second order method with $O((n+m)r)$ complexity.
- We implement an effective realization of a reduced Hessian method for the optimization over the desingularized variety (section 4.3). We start with the Lagrange multipliers method for which we derive a formula for the Newton method for the corresponding optimization problem. The latter takes the saddle point form which we solve using the null space method found in [Benzi et al., 2005]. In section 4.3.6 we show how to reduce the total complexity of the algorithm to $O((n+m)r)$ per iteration.
- We also briefly discuss a few technical details in the implementation of the algorithm (section 4.4)
- We present results of numerical experiments and compare them with some other methods found in section 4.5.

The manifolds that we work with in this paper will always be C^∞ and in fact smooth algebraic varieties.

4.1.1 Idea of desingularization

Before we define desingularization of bounded rank matrix sets, we will introduce its basic idea. The low-rank matrix case will be described in next section. Let V be a variety (not necessarily smooth) and f be a function

$$f : V \to \mathbb{R},$$

which is smooth in an open neighborhood of V (which is assumed to be embedded in $\mathbb{R}^k$). To solve

$$f(x) \to \min, x \in V,$$

we often use methods involving the tangent bundle of V. However, due to the existence of the singular points where the tangent space is not well-defined, it is hard to prove correctness and convergence using those methods. To avoid this problem we construct a smooth variety $\widehat{V}$ and a surjective smooth map π

$$\pi : \widehat{V} \to V.$$

Let $\widehat{f}$ be a pullback of f via map π i.e.

$$\widehat{f} : \widehat{V} \to \mathbb{R},$$

$$\widehat{f} = f \circ \pi.$$

It is obvious that

$$\min_{x \in V} f(x) = \min_{y \in \widehat{V}} \widehat{f}(y),$$

so we reduced our non-smooth minimization problem to a smooth one. Typically $\widehat{V}$ is a variety in a space of bigger dimension and is constructed to be of the same dimension as the smooth part of V. To have some geometrical idea one can think about the following example (see fig. 4-1). Let V be a cubic curve given by the following equation

$$y^2 = x^2(x+1),$$

and parametrized as

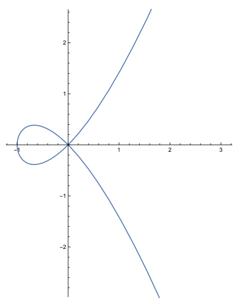
$$(x(t), y(t)) = (t^2 - 1, t(t^2 - 1)).$$

It is easy to see that $(0, 0)$ is a singular point of V. Then its desingularization is given by

$$\widehat{V} = (x(t), y(t), z(t)) = (t^2 - 1, t(t^2 - 1), t) \subset \mathbb{R}^3,$$

which is clearly smooth. Projection is then just

$$\pi : (x(t), y(t), z(t)) = (x(t), y(t)).$$


(a) Singular cubic

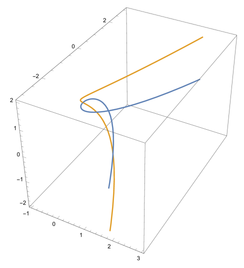

(b) Desingularized cubic

Figure 4-1: Desingularization of the cubic.

4.2 Desingularization of low-rank matrix varieties via kernel

4.2.1 2×2 matrices

Let V be a variety of 2×2 matrices with the rank ≤ 1. We have

$$V = \{(x_{11}, x_{21}, x_{12}, x_{22}) \in \mathbb{R}^4 : x_{11}x_{22} - x_{12}x_{21} = 0\}, \tag{4.1}$$

so it is indeed an algebraic variety. In order to analyze its smoothness and compute the tangent space we recall the following result.

Let $h_i \quad i \in \{1 \dots k\}$ be some smooth functions

$$h_i : \mathbb{R}^l \to \mathbb{R},$$

with $k \leq l$. Define the set M as

$$M = \{x \in \mathbb{R}^l : h_1(x) = 0, h_2(x) = 0 \ldots h_k(x) = 0\}.$$

Then for a point $p \in M$ we construct the matrix $N(p)$,

$$N(p) = \begin{bmatrix} \nabla h_1(p) \\ \nabla h_2(p) \\ \vdots \\ \nabla h_k(p) \end{bmatrix},$$

where $\nabla h_i(p)$ is understood as the row vector

$$\nabla h_i(p) = \left(\frac{\partial h_i}{\partial x_1}, \ldots, \frac{\partial h_i}{\partial x_l} \right).$$

A point p is called nonsingular if $N(p)$ has maximal row rank at p. In this case, by implicit function theorem, M is locally a manifold (see [Lee, 2013, Theorem 5.22]) and tangent space at p is defined as

$$T_pM = \{v \in \mathbb{R}^l : N(p)v = 0\}.$$

Applying this to V defined in eq. (4.1) we obtain

$$N(x_{11}, x_{21}, x_{12}, x_{22}) = \begin{bmatrix} x_{22} & -x_{12} & -x_{21} & x_{11} \end{bmatrix},$$

and then $(0, 0, 0, 0)$ is a singular point of V.

We desingularize it by considering $\widehat{V}$ which is defined as the set of pairs $(A, Y) \in \mathbb{R}^{2\times 2} \times \mathbb{R}^2$ with coordinates

$$A = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix},$$

and

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix},$$

satisfying

$$AY = 0,$$

and

$$Y^\top Y = 1.$$

Such choice of equations for Y is based on the Room-Kempf procedure described in [Naldi, 2015], which suggests the following equations:

$$AY = 0, \quad BY = 0,$$

with some fixed matrix B. Since the latter equation is numerically unstable, using an orthogonality condition instead allows us to maintain the manifold property while making computations more robust.

More explicitly we have

$$\widehat{V} = \{p : (x_{11}y_1 + x_{12}y_2 = 0, x_{21}y_1 + x_{22}y_2 = 0, y_1^2 + y_2^2 = 1)\},$$

$$p = (x_{11}, x_{21}, x_{12}, x_{22}, y_1, y_2) \in \mathbb{R}^6.$$

We find that the normal space at p is spanned by rows of the following matrix $N(p)$:

$$N(p) = \begin{bmatrix} y_1 & 0 & y_2 & 0 & x_{11} & x_{12} \\ 0 & y_1 & 0 & y_2 & x_{21} & x_{22} \\ 0 & 0 & 0 & 0 & 2y_1 & 2y_2 \end{bmatrix}. \tag{4.2}$$

Since $y_1^2 + y_2^2 = 1$ the matrix eq. (4.2) clearly has rank 3 at any point of $\widehat{V}$ which proves that $\widehat{V}$ is smooth. The projection π is just

$$\pi : (x_{11}, x_{21}, x_{12}, x_{22}, y_1, y_2) \to (x_{11}, x_{21}, x_{12}, x_{22}),$$

whose image is the entire V. However, we would also like to estimate how close the tangent spaces are at close points. Recall that by definition of the Grassmanian metric the distance between subspaces C and D is given by

$$d_{Gr}(C, D) := \|P_C - P_D\|_F,$$

where P_C and P_D are the orthogonal projectors on the corresponding planes. Since $P_{C^\perp} = I - P_C$ the distance between any two subspaces is equal to the distance between their orthogonal complements.
It is well known that the projection on the subspace spanned by the rows of a matrix M is given by $M^\dagger M$, where $M^\dagger$ is a pseudoinverse which for matrices of maximal row rank is defined as

$$M^\dagger = M^\top (M M^\top)^{-1}.$$

Hence, for two different points p and p' on the desingularized manifold we obtain

$$\|P_{N(p)} - P_{N(p')}\|_F = \|N(p)^\dagger N(p) - N(p')^\dagger N(p')\|_F.$$

We will use the following classical result to estimate $\|P_{N(p)} - P_{N(p')}\|_F$ (we use it in the form appearing in [Dutta and Li, 2017, Lemma 3.4] which is based on the [Davis and Kahan, 1970, The $\sin\theta$ Theorem]):

$$\|N(p)^\dagger N(p) - N(p')^\dagger N(p')\|_F \leq 2 \max\{\|N(p)^\dagger\|_2, \|N(p')^\dagger\|_2\} \|N(p) - N(p')\|_F. \quad (4.3)$$

In order to estimate the smoothness we need to estimate how $P_{N(p)}$ changes under small changes of p. It is sufficient to estimate the gradient of P. Thus, we have to uniformly bound $\|N^\dagger\|_2$ from above, which is equivalent to bounding the minimal singular value of N from below. Denote the latter by $\sigma_{\min}(N)$. By taking the defining equations of the desingularized manifold into account, we find that

$$N(p)N(p)^\top = \begin{bmatrix} 1 + x_{11}^2 + x_{12}^2 & x_{11}x_{21} + x_{12}x_{22} & 0 \\ x_{11}x_{21} + x_{12}x_{22} & 1 + x_{21}^2 + x_{22}^2 & 0 \\ 0 & 0 & 4 \end{bmatrix}. \quad (4.4)$$

Hence $\sigma_{\min}^2(N(a)) \geq 1$ and $\|N(a)^\dagger\|_2 \leq 1$. From the definition of $N(p)$ it follows that for $p = (A, Y)$ and $p' = (A', Y')$:

$$\|N(p) - N(p')\|_F \leq \sqrt{6}\|Y - Y'\|_F + \|A - A'\|_F.$$

and from eq. (4.3) we obtain

$$d_{Gr}(T_p\widehat{V}, T_{p'}\widehat{V}) \leq 2\sqrt{6}(\|A - A'\|_F + \|Y - Y'\|_F).$$

We will derive and prove similar estimates for the general case in the next section.

4.2.2 General construction and estimation of curvature

Remark. *We will often use vectorization of matrices which is a linear operator*

$$\mathrm{vec} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn\times 1},$$

which acts by stacking columns of the matrix into a single column vector. To further simplify notation, variables denoted by uppercase and lowercase variables are understood as a matrix and vectorization of the corresponding matrix, e.g. $p = \mathrm{vec}(P)$*. We will also define the transposition operator* $T_{m,n}$*:*

$$T_{m,n} : \mathrm{vec}(X) \to \mathrm{vec}(X^\top),$$

for $X \in \mathbb{R}^{m\times n}$*.*

Consider a variety $\mathcal{M}_{\leq r}$ of $n \times m$ of matrices of rank not higher than r,

$$\mathcal{M}_{\leq r} = \{A \in \mathbb{R}^{n\times m} : \mathrm{rank}(A) \leq r\}.$$

We recall the following classical result [Lakshmibai and Brown, 2015, Theorem 10.3.3].

Lemma 7. *$A \in \mathcal{M}_{\leq r}$ is a singular point if and only if A has rank smaller than r.*

By definition, the dimension of a variety X is equal to the dimension of the manifold $X \setminus X_{sing}$ where X_{sing} is the set of all singular points of X [Griffiths and

Harris, 2014]. In the case of $\mathcal{M}_{\leq r}$ we find that

$$\dim \mathcal{M}_{\leq r} = \dim \mathcal{M}_{=r},$$

where

$$\mathcal{M}_{=r} = \{A \in \mathbb{R}^{n\times m} : \operatorname{rank}(A) = r\},$$

is known to be of dimension $(n+m)r - r^2$ (e.g. [Vandereycken, 2013, Proposition 2.1]).

Now we return to the main topic of the paper.
Let $Gr(m-r, m)$ be the Grassmann manifold:

$$Gr(m-r, m) = \mathbb{R}_*^{m,m-r}/GL_{m-r},$$

where $\mathbb{R}_*^{m,m-r}$ is the noncompact Stiefel manifold

$$\mathbb{R}_*^{m,m-r} = \{Y \in \mathbb{R}^{m\times(m-r)} : Y \text{ full rank}\},$$

and GL_{m-r} is the group of invertible $m-r \times m-r$ matrices.
It is known [Lee, 2013] that

$$\dim Gr(m-r, m) = r(m-r).$$

We propose the following desingularization for $\widehat{\mathcal{M}}_r$:

$$\widehat{\mathcal{M}}_r = \{(A, Y) \in \mathbb{R}^{n\times m} \times Gr(m-r, m) : AY = 0\}, \tag{4.5}$$

and prove the following theorem.

Theorem 6. *$\widehat{\mathcal{M}}_r$ as defined by eq. (4.5) is a smooth manifold of dimension $(n+m)r - r^2$.*

Proof. Let U_α be a local chart of $Gr(m-r, m)$. To prove the theorem it suffices to show that $\widehat{\mathcal{M}}_r \cap (\mathbb{R}^{n\times m} \times U_\alpha)$ is a smooth manifold for all α. Without loss of generality let us assume that the coordinates of $Y \in Gr(m-r, m) \cap U_\alpha$ are given by

$$Y = \begin{bmatrix} I_{m-r} \\ Y_\alpha \end{bmatrix},$$

where

$$Y_\alpha = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \dots & \alpha_{1,m-r} \\ \alpha_{2,1} & \alpha_{2,2} & \dots & \alpha_{2,m-r} \\ \dots & \dots & \dots & \dots \\ \alpha_{r,1} & \alpha_{2,2} & \dots & \alpha_{r,m-r} \end{bmatrix}.$$

In this chart equation eq. (4.5) reads

$$A \begin{bmatrix} I_{m-r} \\ Y_\alpha \end{bmatrix} = 0. \tag{4.6}$$

Splitting A as

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix},$$

where

$$A_1 \in \mathbb{R}^{n \times (m-r)}, \quad A_2 \in \mathbb{R}^{n \times r},$$

and by using properties of the Kronecker product $\otimes$ we obtain that the Jacobian matrix of eq. (4.6) is equal to

$$\begin{bmatrix} I_{n(m-r)} & Y_\alpha^\top \otimes I_n & I_{m-r} \otimes A_2 \end{bmatrix},$$

which is clearly of full rank, since it contains identity matrix. To conclude the proof we note that

$$\dim \widehat{\mathcal{M}}_r = \underbrace{nm + (m-r)r}_{\text{number of variables}} - \underbrace{n(m-r)}_{\text{number of equations}} = (n+m)r - r^2,$$

as desired. □

The use of $\widehat{\mathcal{M}}_r$ is justified by the simple lemma

Lemma 8. *The following statements hold:*

- *If* $(A, Y) \in \widehat{\mathcal{M}}_r$ *then* $A \in \mathcal{M}_{\leq r}$,
- *If* $A \in \mathcal{M}_{\leq r}$ *then there exists* Y *such that* $(A, Y) \in \widehat{\mathcal{M}}_r$.

Proof. These statements obviously follow from the equation

$$AY = 0,$$

which implies that the dimension of the nullspace of A is at least $m - r$. □

We would like to construct Newton method on the manifold $\widehat{\mathcal{M}}_r$. In order to work with quotient manifolds such as $Gr(m-r, m)$ the conventional approach is to use the total space of the quotient. The tangent space is then dealt with using the concept of *horizontal space* (sometimes this is referred to as *gauge condition*) which is isomorphic to the tangent space of the quotient manifold. This approach is explained in great detail in [Absil et al., 2009]. Although we will not go into the details of these concepts, we will apply them to $\widehat{\mathcal{M}}_r$ in the next section.

4.2.3 Tangent space of $\widehat{\mathcal{M}}_r$

For our analysis, it is more convenient to employ the following representation of the Grassmanian:

$$Gr(m-r, m) = St(m-r, m)/O_{m-r}, \tag{4.7}$$

where $St(m-r, m)$ is the orthogonal Stiefel manifold

$$St(m-r, m) = \{Y \in \mathbb{R}_*^{m, m-r} : Y^\top Y = I_{m-r}\},$$

and O_{m-r} is the orthogonal group.

Let π be the quotient map eq. (4.7) and id $\times\, \pi$ is the obvious map

$$\mathbb{R}^{n\times m} \times St(m-r, m) \to \mathbb{R}^{n\times m} \times Gr(m-r, m).$$

It is easy to see that $\widehat{\mathcal{M}}_r^{\text{tot}} := (\text{id} \times \pi)^{-1}(\widehat{\mathcal{M}}_r)$ is the following manifold:

$$\widehat{\mathcal{M}}_r^{\text{tot}} = \{(A, Y) \in \mathbb{R}^{n\times m} \times \mathbb{R}_*^{m,m-r} : AY = 0, \quad Y^\top Y = I_{m-r}\}. \tag{4.8}$$

Let us now compute the horizontal distribution on $\widehat{\mathcal{M}}_r^{\text{tot}}$. As described in [Absil et al., 2009, Example 3.6.4] in the case of the projection

$$\pi : \mathbb{R}_*^{m,m-r} \to Gr(m-r, m),$$

$$Y \to \text{span}(Y),$$

the horizontal space at Y is defined as the following subspace of $T_Y\mathbb{R}_*^{m,m-r}$:

$$\{\delta Y \in T_Y\mathbb{R}_*^{m,m-r} : (\delta Y)^\top Y = 0\}. \tag{4.9}$$

It immediately follows that in the case eq. (4.8) the horizontal space at (A, Y) is equal to

$$\mathcal{H}(A, Y) = T_{(A,Y)}(\widehat{\mathcal{M}}_r^{\text{tot}}) \cap \mathcal{H}_{\text{Gr}}(A, Y),$$

where $\mathcal{H}_{\text{Gr}}(A, Y)$ is similarly to eq. (4.9) defined as:

$$\mathcal{H}_{\text{Gr}}(A, Y) = \{(\delta A, \delta Y) \in T_{(A,Y)}(\mathbb{R}^{n\times m} \times \mathbb{R}_*^{m,m-r}) : (\delta Y)^\top Y = 0\}. \tag{4.10}$$

Note that the dimension of $\mathcal{H}$ is equal to the dimension of $\widehat{\mathcal{M}}_r$ since it is, by construction, isomorphic to the $T\widehat{\mathcal{M}}_r$. We now proceed to one of the main results of the paper

Theorem 7. *The orthogonal projection on $\mathcal{H}(A, Y)$ is Lipschitz continuous with respect to (A, Y) and its Lipschitz constant is no greater than $2(\sqrt{n} + \sqrt{m-r})$ in the Frobenius norm.*

Proof. In order to prove the theorem, first we need to find the equations of $\mathcal{H}(A, Y)$. Recall the defining equations of $\widehat{\mathcal{M}}_r^{\text{tot}}$ eq. (4.8) and that for a given $p = (A, Y)$ the tangent space is the nullspace of the gradient of the constraints. By taking into account the gauge condition eq. (4.10) we find that

$$\mathcal{H}(A, Y) = \{v : N(p)v = 0\},$$

where the matrix $N(p)$ has the following block structure:

$$N(p) = \begin{bmatrix} Y^\top \otimes I_n & I_{m-r} \otimes A \\ 0 & I_{m-r} \otimes Y^\top \end{bmatrix}. \tag{4.11}$$

For simplicity of notation we will omit p in $N(p)$. The projection onto the horizontal space of a given vector z is given by the following formula

$$v = (I - N^\top (NN^\top)^{-1} N)z = P_N z, \tag{4.12}$$

where

$$P_N = (I - N^\dagger N),$$

is the orthogonal projector onto the row range of N. Using exactly the same idea as in previous section we estimate $\sigma_{\min}(N)$ from below. Consider the Gram matrix

$$\begin{aligned} Z = NN^\top = \begin{bmatrix} Y^\top \otimes I_n & I_{m-r} \otimes A \\ 0 & I_{m-r} \otimes Y^\top \end{bmatrix} \begin{bmatrix} Y \otimes I_n & 0 \\ I_{m-r} \otimes A^\top & I_{m-r} \otimes Y \end{bmatrix} = \\ = \begin{bmatrix} Y^\top Y \otimes I_n + I_{m-r} \otimes AA^\top & I_{m-r} \otimes AY \\ I_{m-r} \otimes Y^\top A^\top & I_{m-r} \otimes Y^\top Y \end{bmatrix}. \end{aligned}$$

Now we recall that for each point at the manifold $\widehat{\mathcal{M}}_r^{\text{tot}}$ eq. (4.8) holds, therefore

$$Z = \begin{bmatrix} I + I_{m-r} \otimes AA^\top & 0 \\ 0 & I \end{bmatrix}. \tag{4.13}$$

It is obvious that $\sigma_{\min}(Z) \geq 1$ since it has the form $I + DD^\top$. Finally, $\sigma^2_{\min}(N) = \sigma_{\min}(Z) \geq 1$, therefore

$$\sigma_{\min}(N) = \sigma_{\min}(N^\top) \geq 1, \quad \|(N^\top)^\dagger\|_2 \leq 1. \tag{4.14}$$

Putting eq. (4.14) into eq. (4.3) we get

$$\|P_N - P_{N'}\|_F \leq 2\|N - N'\|_F,$$

with $N = N(A, Y), N' = N(A', Y')$. Finally, we need to estimate how N changes under the change of A and Y. We have

$$N - N' = \begin{bmatrix} (Y - Y') \otimes I_n & I_{m-r} \otimes (A - (A')) \\ 0 & I_{m-r} \otimes (Y^\top - (Y')^\top) \end{bmatrix},$$

therefore

$$\|N - N'\|_F \leq (\sqrt{n} + \sqrt{m-r})\|Y - Y'\|_F + \sqrt{m-r}\|A - A'\|_F.$$

Thus

$$\begin{aligned} d_{Gr}(\mathcal{H}(A', Y'), \mathcal{H}(A, Y)) &= \|P_N - P_{N'}\|_F \\ &\leq 2\|N - N'\|_F \\ &\leq 2(\sqrt{n} + \sqrt{m-r})(\|Y - Y'\|_F + \|A - A'\|_F). \end{aligned} \tag{4.15}$$

□

For small r

$$(m+n)r - r^2 \ll nm,$$

so to fully utilize the properties of $\widehat{\mathcal{M}_r}$ in computations we first have to find an explicit basis in the horizontal space. This will be done in the next section.

4.2.4 Parametrization of the tangent space

To work with low rank matrices it is very convenient to represent them using the truncated singular value decomposition (SVD). Namely for $A \in \mathcal{M}_{\leq r}$ we have

$$A = USV^\top,$$

with U and V having r orthonormal columns and S being a diagonal matrix. Using this notation we find that the following result holds:

Theorem 8. *The orthogonal basis in the kernel of N from eq. (4.11) is given by columns of the following matrix Q*

$$Q = \begin{bmatrix} V \otimes I_n & -Y \otimes (US_1) \\ 0 & I_{m-r} \otimes (VS_2) \end{bmatrix},$$

where

S_1 and S_2 are diagonal matrices defined as

$$S_1 = S(S^2 + I_r)^{-\frac{1}{2}}, \quad S_2 = (S^2 + I_r)^{-\frac{1}{2}}$$

Proof. It suffices to verify that $Q^\top Q = I$ and $NQ = 0$ which is performed by direct multiplication. The number of columns in Q is $nr + (m - r)r$ which is exactly the dimension of the $\mathcal{H}(A, Y)$. □

Now we will use smoothness of $\widehat{\mathcal{M}}_r$ to develop an optimization algorithm over $\mathcal{M}_{\leq r}$. The idea of using kernel of a matrix in optimization problems has appeared before [Markovsky and Usevich, 2012, 2013]. Algorithm constructed there is a variable–projection—like method with $O(m^3)$ per iteration complexity, where m is number of columns in the matrix. We explain this approach in more detail in section 5.6.

4.3 Newton method

4.3.1 Basic Newton method

Consider the optimization problem

$$F(A) \to \min, \quad \text{s.t. } A \in \mathcal{M}_{\leq r},$$

where F is twice differentiable. Using the idea described in section 4.1.1 this problem is equivalent to

$$\widehat{F}(A, Y) \to \min, \quad \text{s.t. } (A, Y) \in \widehat{\mathcal{M}}_r,$$

and

$$\widehat{F}(A, Y) = F(A).$$

Following the approach described in e.g. [Absil et al., 2009, Section 4.9] we solve this problem by lifting it to the total space $\widehat{\mathcal{M}}_r^{\text{tot}}$ defined by eq. (4.8) with the additional condition that the search direction lies in the horizontal bundle $\mathcal{H}$, that is

$$\widetilde{F}(A, Y) \to \min, \quad \text{s.t. } (A, Y) \in \widehat{\mathcal{M}}_r^{\text{tot}},$$

$$\widetilde{F}(A, Y) = F(A),$$

$$(\delta A, \delta Y) \in \mathcal{H}(A, Y). \tag{4.16}$$

To solve it we will rewrite it using the Lagrange multipliers method, with the additional constraint eq. (4.16). Taking into account the defining equations of $\widehat{\mathcal{M}}_r^{\text{tot}}$ eq. (4.8) the Lagrangian for the constrained optimization problem reads

$$\mathcal{L}(A, Y, \Lambda, M) = F(A) + \langle AY, \Lambda \rangle + \frac{1}{2}\langle M, Y^\top Y - I \rangle,$$

where $\Lambda \in \mathbb{R}^{n \times m-r}$ and $M \in \mathbb{R}^{(m-r)\times(m-r)}$, $M^\top = M$ are the Lagrange multipliers. We now find the first-order optimality conditions.

4.3.2 First order optimality conditions

By differentiating $\mathcal{L}$ we find the following equations

$$\nabla F + \Lambda Y^\top = 0, \quad YM + A^\top \Lambda = 0, \quad AY = 0, \quad Y^\top Y = I.$$

Multiplying second equation by $Y^\top$ from the left and using equations $AY = 0$ and $Y^\top Y = I$ we find that $M = 0$. Thus, the first-order optimality conditions reduce to

$$\nabla F + \Lambda Y^\top = 0, \quad A^\top \Lambda = 0, \quad AY = 0, \quad Y^\top Y = I. \tag{4.17}$$

4.3.3 Newton method and the reduced Hessian system

Now we can write down the Newton method for the system eq. (4.17), which can be written in the saddle point form

$$\begin{bmatrix} \widehat{G} & N^\top \\ N & 0 \end{bmatrix} \begin{bmatrix} \delta z \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{4.18}$$

and

$$f = -\mathrm{vec}(\nabla F + \Lambda Y^\top).$$

where we assumed that the initial point satisfies the constraints ($AY = 0, Y^\top Y = I_{m-r}$), the vectors δz and $\delta\lambda$ are

$$\delta z = \begin{bmatrix} \mathrm{vec}(\delta A) \\ \mathrm{vec}(\delta Y) \end{bmatrix}, \quad \delta\lambda = \begin{bmatrix} \mathrm{vec}(\delta\Lambda) \\ \mathrm{vec}(\delta M) \end{bmatrix},$$

and the matrix $\widehat{G}$ in turn has a saddle-point structure:

$$\widehat{G} = \begin{bmatrix} H & C \\ C^\top & 0 \end{bmatrix},$$

where $H = \nabla^2 F$ is the ordinary Hessian, and C comes from differentiating the term $\Lambda Y^\top$ with respect to Y and will be derived later in the text. The constraints on the search direction δz are written as

$$N\delta z = 0,$$

and

$$N = \begin{bmatrix} Y^\top \otimes I_n & I_{m-r} \otimes A \\ 0 & I_{m-r} \otimes Y^\top \end{bmatrix},$$

which means that δz is in the $\mathcal{H}(A, Y)$ as desired. In what follows our approach is similar to the null space methods described in [Benzi et al., 2005, Section 6]. Using a parametrization via the matrix Q defined in theorem 8 we obtain that $\delta z = Q\delta w$.

The first block row of system eq. (4.18) reads

$$\widehat{G}Q\delta w + N^\top \delta\lambda = f.$$

Multiplying by $Q^\top$ we can eliminate $\delta\lambda$, which leads to the reduced Hessian equation

$$Q^\top \widehat{G}Q\delta w = Q^\top f. \tag{4.19}$$

Note that $Q^\top \widehat{G}Q$ is a small $(n+m)r-r^2 \times (n+m)r-r^2$ matrix as claimed. We now would like to simplify equation eq. (4.19). Using the transposition operator defined in section 4.2.2 we find that matrix C is written as

$$C = (I_m \otimes \Lambda)T_{m,m-r}.$$

An important property of the matrix C is that if $Q_{12} = -Y \otimes (US_1)$ is the (1, 2) block of the matrix Q, then

$$Q_{12}C = 0,$$

if

$$A^\top \Lambda = 0,$$

which is again verified by direct multiplication using the properties of the Kronecker product. The direct evaluation of the product

$$\widehat{G}^{loc} = Q^\top \widehat{G}Q,$$

(together with the property above) gives

$$\widehat{G}^{loc} = \begin{bmatrix} Q_{11}^\top HQ_{11} & Q_{11}^\top HQ_{12} + Q_{11}^\top CQ_{22} \\ Q_{12}^\top HQ_{11} + Q_{22}^\top C^\top Q_{11} & Q_{12}^\top HQ_{12} \end{bmatrix}, \tag{4.20}$$

and the system we need to solve has the form

$$\begin{bmatrix} Q_{11}^\top H Q_{11} & Q_{11}^\top H Q_{12} + Q_{11}^\top C Q_{22} \\ Q_{12}^\top H Q_{11} + Q_{22}^\top C^\top Q_{11} & Q_{12}^\top H Q_{12} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} Q_{11}^\top f \\ Q_{12}^\top f \end{bmatrix}, \tag{4.21}$$

with

$$\delta U \in \mathbb{R}^{n \times r}, \delta P \in \mathbb{R}^{r \times (m-r)}.$$

We also need to estimate Λ. Recall that to get $Q_{12}C = 0$ we have to require that $A^\top \Lambda = 0$ exactly, thus

$$\Lambda = Z\Phi,$$

where Z is the orthonormal basis for the left nullspace of A, and Φ is defined from the minimization of

$$\|\nabla F + Z\Phi Y^\top\| \to \min,$$

i.e.

$$\Phi = -Z^\top \nabla F Y,$$

and

$$\Lambda = -ZZ^\top \nabla F Y.$$

Note that f then is just a standard projection of ∇F on the tangent space:

$$f = -\mathrm{vec}(\nabla F - ZZ^\top \nabla F Y Y^\top) = -\mathrm{vec}(\nabla F - (I - UU^\top)\nabla F(I - VV^\top)),$$

which is always a vectorization of a matrix with a rank not larger than $2r$. Moreover,

$$g_1 = Q_{11}^\top f = (V^\top \otimes I) f = \tag{4.22}$$

$$-\mathrm{vec}((\nabla F - (I - UU^\top)\nabla F(I - VV^\top))V) = \mathrm{vec}(-\nabla F V),$$

and the second component

$$g_2 = Q_{12}^\top f = -(Y^\top \otimes (US_1)^\top) f = \tag{4.23}$$

$$\mathrm{vec}((US_1)^\top(\nabla F - (I - UU^\top)\nabla F(I - VV^\top))Y) = \mathrm{vec}(S_1^\top U^\top \nabla F Y).$$

The solution is recovered from δu, δp as

$$\delta a = (V \otimes I_n)\delta u - (Y \otimes (US_1))\delta p,$$

or in the matrix form,

$$\delta A = \delta U V^\top - US_1 \delta P Y^\top,$$

and the error in A (which we are interested in) is given by

$$\|\delta A\|_F^2 = \|\delta U\|_F^2 + \|S_1 \delta P\|_F^2.$$

We can further simplify the off-diagonal block. Consider

$$\widehat{C} = Q_{11}^\top C Q_{22} = (V^\top \otimes I)(I \otimes \Lambda)T(I \otimes V)(I \otimes S_2).$$

Then multiplication of this matrix by a vector takes the form:

$$\mathrm{mat}(\widehat{C}\mathrm{vec}(\Phi)) = \Lambda(VS_2\Phi)^\top V = \Lambda\Phi^\top S_2^\top V^\top V = \Lambda\Phi^\top S_2^\top,$$

thus

$$\widehat{C} = (S_2 \otimes \Lambda)T_{r,n-r}.$$

4.3.4 Retraction

Note that since we assumed that the initial points satisfy the constraints

$$AY = 0, \quad Y^\top Y = I_{m-r}, \tag{4.24}$$

after doing each step of the Newton algorithm we have to perform the retraction back to the manifold $\widehat{\mathcal{M}}_r^{tot}$. One such possible retraction is the following. Define a map

$$R : \widehat{\mathcal{M}}_r^{tot} \oplus \mathcal{H} \to \widehat{\mathcal{M}}_r^{tot},$$

$$R((A,Y),(\delta Y,\delta A)) = (R_1(A,\delta A), R_2(Y,\delta Y))$$

$$R_2(Y,\delta Y) = \mathbf{qf}(Y+\delta Y) = Y_1,$$

$$R_1(A,\delta A) = A(I - Y_1 Y_1^\top)$$

where $\mathbf{qf}(\xi)$ denotes the Q factor of the QR-decomposition of ξ, which is a standard second-order retraction on the Stiefel manifold [Absil et al., 2009, Example 4.1.3].

In the fast version of the Newton method which will be derived later, we will also use the standard SVD-based retraction which acts on the matrix $A+\delta A$ simply by truncating it's SVD to the rank r. It is also known that given the SVD of the matrix A then for certain small corrections δA, the SVD of $A+\delta A$ can be recomputed with low computational cost as described in [Vandereycken, 2013, §3]. It is also known to be a second order retraction [Absil and Malick, 2012]. We denote this operation by $R_{\text{SVD}}(A,\delta A)$.

4.3.5 Basic Algorithm

The basic Newton method on the manifold $\widehat{\mathcal{M}}_r$ is summarized in the following algorithm

Algorithm 1 Newton method

```
1: Initial conditions $A_0, Y_0$, functional $F(A)$ and tolerance $\varepsilon$
2: Result: minimum of $F$ on $\mathcal{M}_{\leq r}$
3: while $\|\delta U^i\|_F^2 + \|(S_1)^i \delta P^i\|_F^2 > \varepsilon$ do
4:     $U^i, S^i, V^i = \mathrm{svd}(A^i)$
5:     Solve $\widehat{G}^{loc} \begin{bmatrix} \delta u^i \\ \delta p^i \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$ where $\widehat{G}^{loc}, g_1, g_2$ are defined by formulas eqs. (4.20), (4.22) and (4.23)
6:     $\delta A^i = \delta U^i V^{i\top} - U^i (S_1)^i \delta P^i Y^{i\top}$  $\delta Y^i = V^i (S_2)^i \delta P^i$
7:     $A^{i+1}, Y^{i+1} = R((A^i, \delta A^i), (Y^i, \delta Y^i))$
8:     $i = i + 1$
9: end while
10: return $A^i$
```

Even though this algorithm demonstrates that our approach is rather inefficient in terms of memory and complexity – storing and doing multiplications by Y are of order $O(m^2)$ instead of desired $O((n+m)r)$. We resolve this issue in the next section. Analysis of the convergence and behavior of the algorithm near the points (A, Y) corresponding to matrices of strictly smaller rank is performed in section 4.5.2.

4.3.6 Semi-implicit parametrization of the tangent space

Let us introduce a new variable

$$\delta\Phi^\top = Y\delta P^\top,$$

$$\delta\Phi \in \mathbb{R}^{r\times m}.$$

This results in an implicit constraint on $\delta\Phi$

$$\delta\Phi V = 0.$$

In order to make an arbitrary Φ satisfy it, we first multiply it by the projection operator $I - VV^\top$,

$$\Phi' = \Phi(I - VV^\top),$$

or in the matrix form

$$\begin{bmatrix} \delta u \\ \delta\phi' \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I - VV^\top \otimes I \end{bmatrix} \begin{bmatrix} \delta u \\ \delta\phi \end{bmatrix}.$$

Notice also that

$$\delta P = \delta\Phi Y,$$

and again using the properties of the Kronecker product we obtain

$$\begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & Y^\top \otimes I \end{bmatrix} \begin{bmatrix} \delta u \\ \delta\phi \end{bmatrix}.$$

Denote

$$\Pi = \begin{bmatrix} I & 0 \\ 0 & I - VV^\top \otimes I \end{bmatrix},$$

$$W = \begin{bmatrix} I & 0 \\ 0 & Y^\top \otimes I \end{bmatrix}.$$

The equations for the Newton method in the new variables take the following form:

$$\Pi^\top W^\top \widehat{G}^{loc} W \Pi \begin{bmatrix} \delta u \\ \delta \phi \end{bmatrix} = \Pi^\top W^\top \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{4.25}$$

where $g_1, g_2, \widehat{G}^{loc}$ are as in eqs. (4.20), (4.22) and (4.23) and the linear system in eq. (4.25) is of size $(n+m)r$.

4.3.7 Iterative method

For a large n and m forming the full matrix eq. (4.25) is computationally expensive, so we switch to iterative methods. To implement the matvec operation we need to simplify

$$\begin{bmatrix} l_1 \\ l_2 \end{bmatrix} = \Pi^\top W^\top \widehat{Q}^{loc} W \Pi \begin{bmatrix} \delta u \\ \delta \phi \end{bmatrix},$$

first.

Direct computation shows that

$$\begin{bmatrix} l_1 \\ l_2 \end{bmatrix} = \Pi^\top W^\top \begin{bmatrix} (V^\top \otimes I) H (V \otimes I) \delta u - \\ (V^\top \otimes I) H (I \otimes U) \mathrm{vec}(S_1 \delta \Phi (I - VV^\top)) - \\ \mathrm{vec}((I - UU^\top) \nabla F (I - VV^\top) \delta \Phi^\top S_2) \\ \\ -(Y^\top \otimes I)(I \otimes S_1)(I \otimes U^\top) H (V \otimes I) \delta u + \\ (Y^\top \otimes I) \mathrm{vec}(S_2 (\delta U)^\top (-(I - UU^\top) \nabla F)) + \\ (Y^\top \otimes I)(I \otimes S_1)(I \otimes U^\top) H (I \otimes U) \mathrm{vec}(S_1 \delta \Phi (I - VV^\top)) \end{bmatrix},$$

and the right hand side has the following form:

$$\begin{bmatrix} g_1' \\ g_2' \end{bmatrix} = \Pi^\top W^\top \begin{bmatrix} -\mathrm{vec}\nabla F V \\ (Y^\top \otimes I)\mathrm{vec}(S_1 U^\top \nabla F) \end{bmatrix}.$$

Since both Π and W only act on the second block it is easy to derive the final formulas:

$$\begin{aligned} l_1 = {} & (V^\top \otimes I)H(V \otimes I)\delta u - (V^\top \otimes I)H(I \otimes U)\mathrm{vec}(S_1 \delta\Phi(I - VV^\top)) \\ & - \mathrm{vec}((I - UU^\top)\nabla F(I - VV^\top)\delta\Phi^\top S_2), \end{aligned} \tag{4.26}$$

$$\begin{aligned} l_2 = {} & -(I - VV^\top \otimes I)(I \otimes S_1)(I \otimes U^\top)H(V \otimes I)\delta u \\ & + \mathrm{vec}(S_2(\delta U)^\top(-(I - UU^\top)\nabla F(I - VV^\top)) \\ & + (I - VV^\top \otimes I)(I \otimes S_1)(I \otimes U^\top)H(I \otimes U)\mathrm{vec}(S_1\delta\Phi(I - VV^\top)), \end{aligned} \tag{4.27}$$

$$g_1' = -\mathrm{vec}\nabla F V, \tag{4.28}$$

$$g_2' = \mathrm{vec}(S_1 U^\top \nabla F(I - VV^\top)). \tag{4.29}$$

Note that in new variables we obtain

$$\delta A = \delta U V^\top - U S_1 \delta\Phi,$$

and

$$A + \delta A = U(SV^\top - S_1\delta\Phi) + \delta U V^\top.$$

Using this representation of $A + \delta A$ we can recompute its SVD without forming the full matrix as described in section 4.3.4. This allows us not to store the matrix A itself but only the U, S and V that we get from the SVD. We obtain the following algorithm

Algorithm 2 Fast Newton method

```
1: Initial conditions $U_0, S_0, V_0$, functional $F(A)$ and tolerance $\varepsilon$
2: Result: minimum of $F$ on $\mathcal{M}_{\leq r}$
3: while $\|\delta U^i\|^2 + \|(S_1)^i \delta\Phi^i\|_F^2 > \varepsilon$ do
4:     Solve linear system with matvec defined by formulas (4.26),(4.27) and right hand side defined by formulas eqs. (4.28) and (4.29) using GMRES, obtaining $\delta u^i, \delta\phi^i$.
5:     $\delta A^i = \delta U^i V^{i\top} - U^i (S_1)^i \delta\Phi^i$
6:     $U^{i+1}, S^{i+1}, V^{i+1} = R_{\mathrm{SVD}}(A^i, \delta A^i)$
7:     $i = i + 1$
8: end while
9: return $U^i, S^i, V^i$
```

4.4 Technical aspects of the implementation

4.4.1 Computation of the matvec and complexity

To efficiently compute the matvec for a given functional F one has to be able to evaluate the following expressions of the first order:

$$\nabla F V, (\nabla F)^\top U, \nabla F \delta X, \delta X \nabla F, \tag{4.30}$$

and of the second order:

$$(V^\top \otimes I) H (V \otimes I) \delta x, (V^\top \otimes I) H (I \otimes U) \delta x \tag{4.31}$$

$$(I \otimes U^\top) H (V \otimes I) \delta x, (I \otimes U^\top) H (I \otimes U) \delta x.$$

The computational complexity of algorithm 2 depends heavily on whether we can effectively evaluate eqs. (4.30) and (4.31), which, however, any similar algorithm requires. Let us now consider two examples.

4.4.2 Matrix completion

Given some matrix B and a set of indices Γ define

$$F(x) = \frac{1}{2} \sum_{(i,j)\in\Gamma} (x_{i,j} - B_{i,j})^2 \to \min, x \in \mathcal{M}_{\leq r}.$$

Then

$$\nabla F_{ij} = x_{ij} - B_{ij}, (i,j) \in \Gamma,$$

$$\nabla F_{ij} = 0, (i,j) \notin \Gamma.$$

Then H in this case is a diagonal matrix with ones and zeroes on the diagonal, the exact position of which are determined by Γ. Assuming that the cardinality of Γ is small, the matrix products from eq. (4.30) can be performed efficiently by doing sparse matrix multiplication. Note that multiplication by H in eq. (4.31) acts as a mask, turning the first factor into a sparse matrix, allowing for effective multiplication by the second factor.

4.4.3 Approximation of a sparse matrix

Consider the approximation functional

$$F(x) = \frac{1}{2}\|x - B\|_F^2 \to \min, x \in \mathcal{M}_{\leq r},$$

and B is a sparse matrix. Then

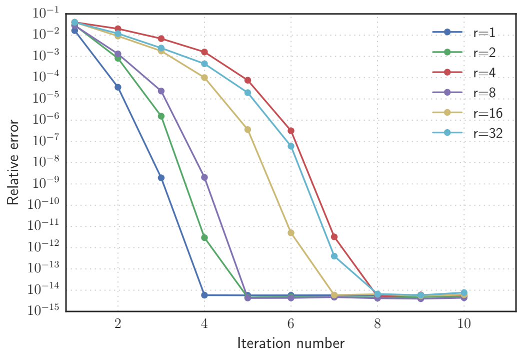
$$\nabla F = x - B,$$

and expressions eq. (4.31), can be heavily simplified by noticing that H in this case is the identity matrix and the sparseness of B is used to evaluate eq. (4.30).

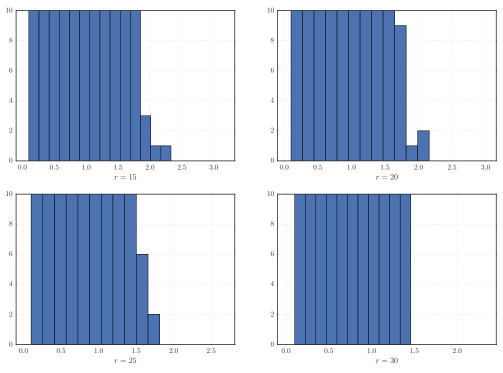
4.5 Numerical results

4.5.1 Convergence analysis

algorithm 2 was implemented in Python using `numpy` and `scipy` libraries. We tested it on the functional described in section 4.4.3 for B being the matrix constructed from the MovieLens 100K Dataset [Harper and Konstan, 2015], so $n = 1000, m = 1700$ and B has 100000 non-zero elements. Since the pure Newton method is only local, for a first test we choose small random perturbation (in the form $0.1\mathcal{N}(0,1)$) of the solution obtained via SVD as initial condition. We got the following results for various r (see fig. 4-2a). This shows the quadratic convergence of our method.



(a) Close initial guess.

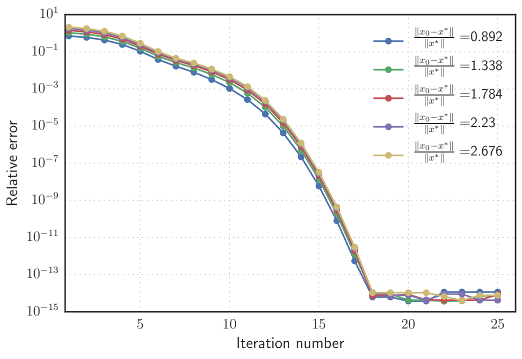


(b) Convergence histograms.

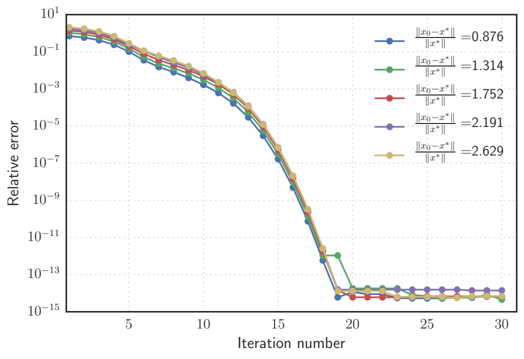
Figure 4-2: Sparse matrix approximation: test of local convergence.

Now we fix the rank and test whether the method converges to the exact answer for a perturbation of the form $\alpha\mathcal{N}(0,1)$ for various α and plot a number of convergent iterations vs $\alpha \in [0.1, 2.5]$ (see fig. 4-2b). We see that for a sufficiently distant initial condition the method does not converge to the desired answer. To fix this we introduce a simple version of the trust-region algorithms described in [Yuan, 2000] (to produce initial condition we perform a few steps of the power method). Results are summarized in fig. 4-3. We also test our algorithm for the matrix completion problem. As an initial data we choose first 15000 entries in the database described above. Using the same trust-region algorithm we obtained the following results (see fig. 4-4a).

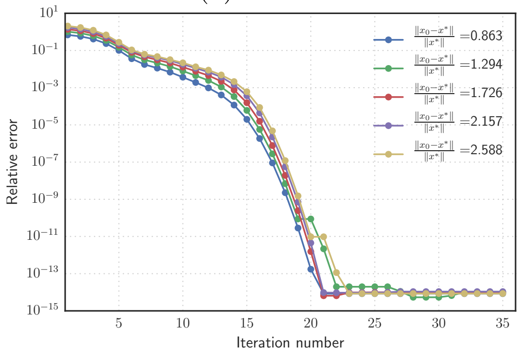
As a final test we show quadratic convergence even in a case when the exact solution is of rank smaller than r for which the method is constructed. To do



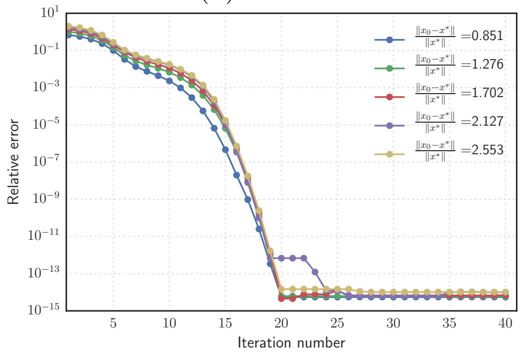
(a) $r = 20$



(b) $r = 25$



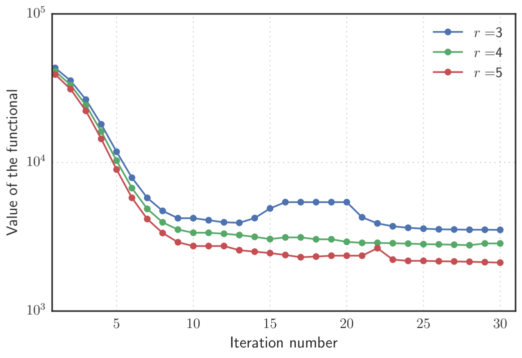
(c) $r = 30$



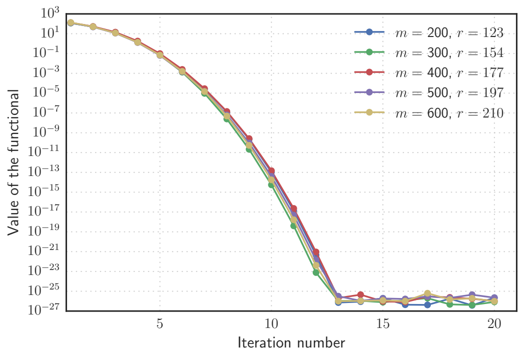
(d) $r = 35$

Figure 4-3: Sparse matrix approximation: trust-region method.

this we take first k elements of the dataset for various k, find the rank r_0 of the matrix constructed from these elements, and run the trust-region Newton method for $r = r_0 + 10$. The results are presented in fig. 4-4b.



(a) Trust region method.



(b) Quadratic convergence in the case of rank defficiency.

Figure 4-4: Matrix completion tests.

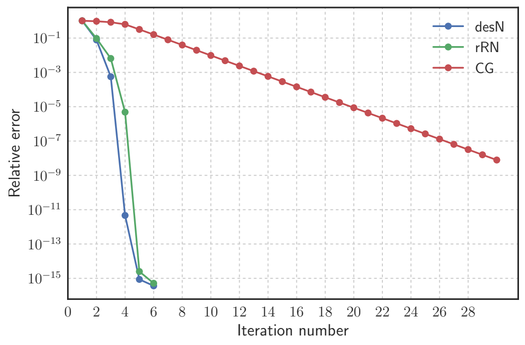
4.5.2 Behavior of the algorithm in cases of rank deficiency and underestimation

In the Newton equation of algorithm 1, one has to solve a linear system with

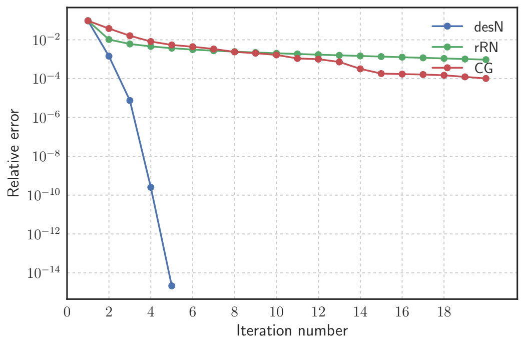
$$\widehat{G}^{loc} = \begin{bmatrix} Q_{11}^\top H Q_{11} & Q_{11}^\top H Q_{12} + Q_{11}^\top C Q_{22} \\ Q_{12}^\top H Q_{11} + Q_{22}^\top C Q_{11} & Q_{12}^\top H Q_{12} \end{bmatrix},$$

with H the Hessian of the objective function $F : \mathbb{R}^{n\times m} \to \mathbb{R}$, which we can assume to be positive definite. Suppose that a matrix of rank $< r$ is the global minimum of F. Then S_1 is singular and $\Lambda = 0$, which in turn imply that $Q_{12} = -Y \otimes (US_1)$ is singular and $C = 0$. Hence, the matrix $\widehat{G}^{loc}$ is singular. It is easy to understand the reason of this behavior. The function $\widehat{F}$ defined on $\widehat{\mathcal{M}}_r$ now has non-unique critical point, — the set of critical points is now in fact a submanifold of $\widehat{\mathcal{M}}_r$. Thus any vector tangent to this submanifold will be a solution of the Newton system. An analysis of the behavior of the Newton method for such functions is studied in e.g. [Decker and Kelley, 1980]. While we plan to analyze it and prove quadratic convergence in our future work, now we note that Krylov iterative methods handle singular systems if we choose initial condition to be the zero vector, and quadratic convergence has been observed in all the numerical experiments.

We will now compare our method (desN) with the reduced Riemannian Newton (rRN) (which is also known as constrained Gauss-Newton method [Kressner et al., 2016]) and CG methods on the fixed-rank matrix manifolds for the approximation problem. The former is obtained by neglecting the curvature term involving S^{-1} in the Hessian (see [Vandereycken, 2013, Proposition 2.3]) and for the latter we use the `Pymanopt` [Townsend et al., 2016] implementation. We choose $n = m = 30, r = 10$ and for the first test we compare the behavior of these algorithms in the case of the exact solution being of rank $r_0 < r$ with $r_0 = 5$. In the second test, we study the converse situation when the rank is underestimated — the exact solution has rank $r_0 > r$ with $r_0 = 15$. As before, for the reference solution we choose a truncated SVD of the approximated matrix. The results are summarized in the figs. 4-5a and 4-5b. Note that the case of rank underestimation was also studied in fig. 4-4b. We observe that the proposed algorithm maintains quadratic convergence in both cases. Even though the reduced Riemannian Newton method is quadratic in the case of rank deficiency, it becomes linear in the case of rank underestimation. This phenomenon is well-known and explained e.g. in [Kressner et al., 2016, Section 5.3] and is related to the fact that when exact minimum is on the variety this approximate model in fact becomes exact second order model. CG is linear in both cases.



(a) Rank defficiency case



(b) Rank underestimation.

Figure 4-5: Comparison of the convergence behaviour of various optimization algorithms.

4.5.3 Comparison with the regularized Newton method

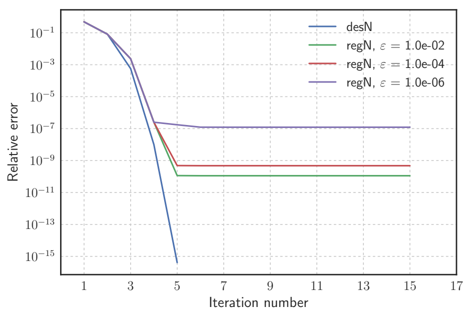
In this subsection we will compare behavior of our method and of the full Riemannian Newton method on the low-rank matrix variety. To avoid problems with zero or very small singular values we choose some small parameter ε, and in the summands involving S^{-1} in the formulas for the Hessian matrix [Vandereycken, 2013, Proposition 2.3] we use the regularized singular values

$$\sigma_i^{\varepsilon} = \max\{\sigma_i, \varepsilon\},$$

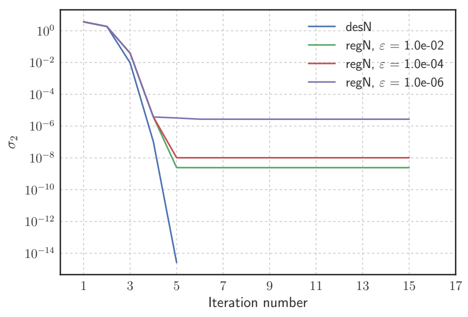
thus obtaining regularized Newton method (regN). As a test problem we choose a matrix completion problem where the exact answer is known (given sufficiently many elements in the matrix) and of a small rank. To construct such a matrix A we take the uniform grid of size $N = 40$ in the square $[-1, 1]^2$ and sample values of the function

$$f(x, y) = e^{-x^2-y^2},$$

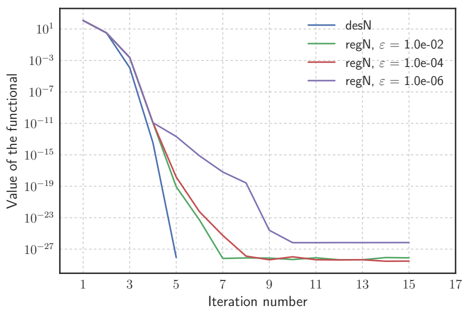
on this grid. It is easy to check that this matrix has rank exactly 1. We choose $r_0 = 5$ and compare relative error with respect to the exact solution A, value of the functional as defined in section 4.4.2 and value of the second singular value σ_2. Results are given in fig. 4-6. We see that even though in all the cases value of the



(a) Relative error w.r.t the exact answer



(b) σ_2 of the current iterate



(c) Value of the functional

Figure 4-6: Matrix completion tests in the case of strong rank deficiency.

functional goes to 0, regularized Newton method fails to recover that σ_2 of the exact answer is in fact 0 and it's behavior depends on the value of ε.

4.6 Related work

Partly similar approach using so-called *parametrization via kernel* is described in [Markovsky and Usevich, 2012, 2013]. However, optimization algorithm proposed there is not considered as an optimization problem on a manifold of tuples (A, Y) and is based on two separate optimization procedures (with respect to A and to Y, where the latter belongs to the orthogonal Stiefel manifold), thus separating the variables. As stated in [Markovsky and Usevich, 2013] in general it has $O(m^3)$ complexity per iteration. An overview of Riemannian optimization is presented in [Qi, 2011]. An example of the traditional approach to bounded-rank matrix sets using Stiefel manifolds is given in [Koch and Lubich, 2007] where explicit formulas for projection onto the tangent space are presented. An application of Riemannian optimization to low-rank matrix completion where $\mathcal{M}_{\leq r}$ is considered as a subvariety in the set of all matrices is given in [Vandereycken, 2013]. The case of F being non-smooth but only Lipschitz is studied in [Hosseini and Uschmajew, 2017]. Theoretical properties of matrix completion such as when exact recovering of the matrix is possible are studied in [Candès and Tao, 2010]. Standard references for introductory algebraic geometry are [Hartshorne, 2013] and [Shafarevich and Hirsch, 1994]. For more computational aspects see [Grayson and Stillman, 2002].

Chapter 5

Geometry Score: A Method For Comparing Generative Adversarial Networks

5.1 Introduction

Generative adversarial networks (GANs) [Goodfellow et al., 2014b] are a class of methods for training generative models, which have been recently shown to be very successful in producing image samples of excellent quality. They have been applied in numerous areas [Radford et al., 2015, Salimans et al., 2016b, Ho and Ermon, 2016]. Briefly, this framework can be described as follows. We attempt to mimic a given target distribution $p_{\text{data}}(\mathbf{x})$ by constructing two networks $G(\mathbf{z};\boldsymbol{\theta}^{(G)})$ and $D(\mathbf{x};\boldsymbol{\theta}^{(D)})$ called the generator and the discriminator. The generator learns to sample from the target distribution by transforming a random input vector $\mathbf{z}$ to a vector $\mathbf{x} = G(\mathbf{z};\boldsymbol{\theta}^{(G)})$, and the discriminator learns to distinguish the model distribution $p_{\text{model}}(\mathbf{x})$ from $p_{\text{data}}(\mathbf{x})$. The training procedure for GANs is typically based on applying gradient descent in turn to the discriminator and the generator in order to minimize a loss function. Finding a good loss function is a topic of ongoing research, and several options were proposed in [Mao et al., 2016, Arjovsky et al., 2017].

One of the main challenges [Lucic et al., 2017, Barratt and Sharma, 2018] in the GANs framework is estimating the quality of the generated samples. In traditional

GAN models, the discriminator loss cannot be used as a metric and does not necessarily decrease during training. In more involved architectures such as WGAN [Arjovsky et al., 2017] the discriminator (critic) loss is argued to be in correlation with the image quality, however, using this loss as a measure of quality is nontrivial. Training GANs is known to be difficult in general and presents such issues as *mode collapse* when $p_{\text{model}}(\mathbf{x})$ fails to capture a multimodal nature of $p_{\text{data}}(\mathbf{x})$ and in extreme cases all the generated samples might be identical. Several techniques to improve the training procedure were proposed in [Salimans et al., 2016b, Gulrajani et al., 2017].

In this work, we attack the problem of estimating the quality and diversity of the generated images by using the machinery of topology. The well-known Manifold Hypothesis [Goodfellow et al., 2016] states that in many cases such as the case of natural images the support of the distribution $p_{\text{data}}(\mathbf{x})$ is concentrated on a low dimensional manifold $\mathcal{M}_{\text{data}}$ in a Euclidean space. This manifold is assumed to have a very complex non-linear structure and is hard to define explicitly. It can be argued that interesting features and patterns of the images from $p_{\text{data}}(\mathbf{x})$ can be analyzed in terms of topological properties of $\mathcal{M}_{\text{data}}$, namely in terms of loops and higher dimensional *holes* in $\mathcal{M}_{\text{data}}$. Similarly, we can assume that $p_{\text{model}}(\mathbf{x})$ is supported on a manifold $\mathcal{M}_{\text{model}}$ (under mild conditions on the architecture of the generator this statement can be made precise [Shao et al., 2017]), and for sufficiently good GANs this manifold can be argued to be quite similar to $\mathcal{M}_{\text{data}}$ (see fig. 5-1). This intuitive claim will be later supported by numerical experiments. Based on this hypothesis we develop an approach which allows for comparing the topology of the underlying manifolds for two point clouds in a stochastic manner providing us with a visual way to detect mode collapse and a score which allows for comparing the quality of various trained models. Informally, since the task of computing the precise topological properties of the underlying manifolds based only on samples is ill-posed by nature, we estimate them using a certain probability distribution (see section 5.4).

We test our approach on several real–life datasets and popular GAN models (DCGAN, WGAN, WGAN-GP) and show that the obtained results agree well with the intuition and allow for comparison of various models (see section 5.5).

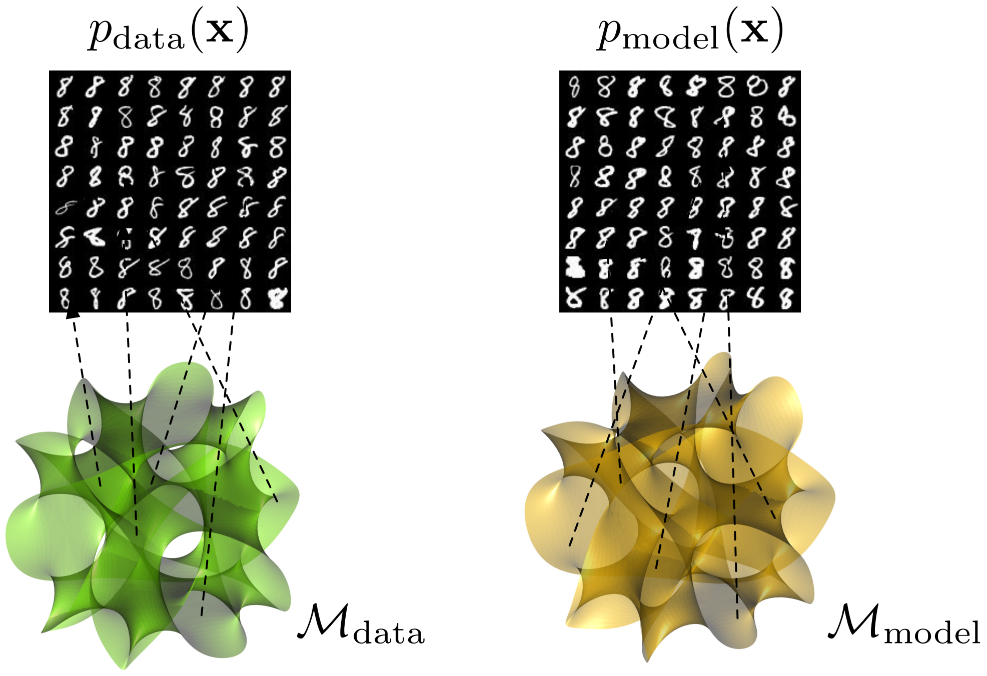

$p_{\text{data}}(\mathbf{x})$
$p_{\text{model}}(\mathbf{x})$
$\mathcal{M}_{\text{data}}$
$\mathcal{M}_{\text{model}}$

Figure 5-1: The Manifold Hypothesis suggests that in the case of natural images the data is supported on a low dimensional *data manifold* $\mathcal{M}_{\text{data}}$. Similarly, GANs sample images from an immersed manifold $\mathcal{M}_{\text{model}}$. By comparing topological properties of the manifolds $\mathcal{M}_{\text{data}}$ and $\mathcal{M}_{\text{model}}$ we can get insight in how strongly GAN captured intricacies in the data distribution $p_{\text{data}}(\mathbf{x})$, and quantitatively estimate the difference.

5.2 Main idea

Let us briefly discuss our approach before dwelling into technical details. As described in the introduction we would like to compare topological properties of $\mathcal{M}_{\text{data}}$ and $\mathcal{M}_{\text{model}}$ in some way. This task is complicated by the fact that we do not have access to the manifolds themselves but merely to samples from them. A natural approach in this case is to approximate these manifolds using some simpler spaces in such a way that topological properties of these spaces resemble those of $\mathcal{M}_{\text{data}}$ and $\mathcal{M}_{\text{model}}$. The main example of such spaces are **simplicial complexes** (fig. 5-2), which are build from intervals, triangles and other higher dimensional simplices. In order to reconstruct the underlying manifold using a simplicial complex several

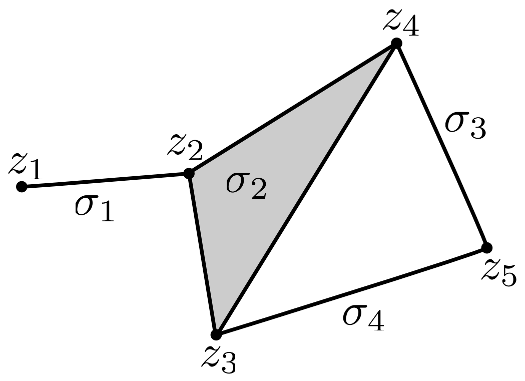


Figure 5-2: Simplicial complex. Topological space X is constructed from several edges $(\sigma_1, \sigma_3, \sigma_4)$ and a two dimensional face σ_2.

methods exist. In all such approaches proximity information of the data is used, such as pairwise distances between samples. Typically one chooses some threshold parameter ε and based on the value of this parameter one decides which simplices are added into the approximation (see fig. 5-3). However a single value ε is not

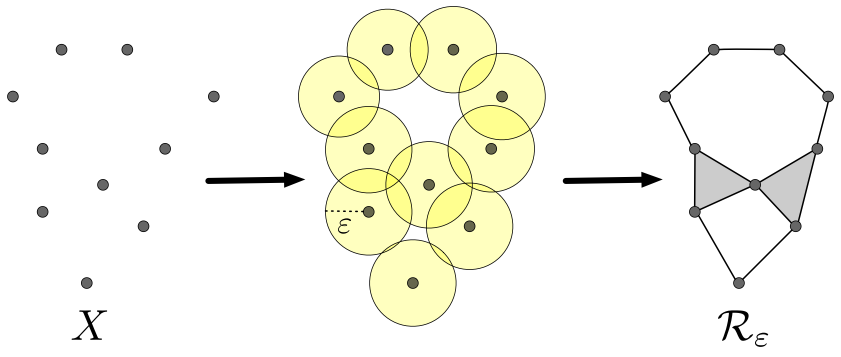


Figure 5-3: A simplicial complex constructed on a sample X. First, we fix the proximity parameter ε. Then we take balls of the radius ε centered at each point, and if for some subset of X of size $k+1$ all the pairwise intersections of the corresponding balls are non-empty, we add the k-dimensional simplex spanning this subset to the simplicial complex $\mathcal{R}_\varepsilon$.

enough — for very small values the reconstructed space will be just a disjoint union of points and for very large ε it will be a single connected blob, while the correct approximation is somewhere in between. This issue is resolved by considering a *family* (fig. 5-4, a) of simplicial complexes, parametrized by the ('persistence') parameter ε. It is also convenient to refer to the parameter ε as *time*, with the idea that we gradually throw more simplices into our simplicial complex as time goes by. For each value of ε we can compute topological properties of the corresponding

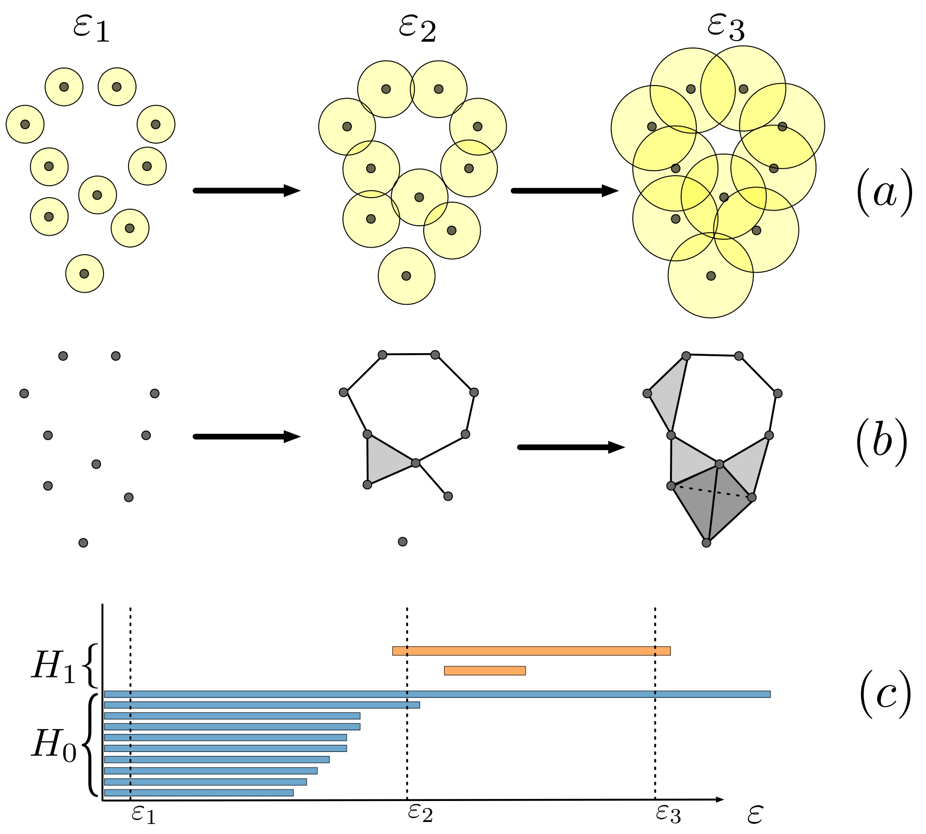


Figure 5-4: Using different values of the proximity parameter ε we obtain different simplicial complexes (a). For $\varepsilon = \varepsilon_1$ the balls do not intersect and there are just 10 isolated components (b, [left]). For $\varepsilon = \varepsilon_2$ several components have merged and one *loop* appeared (b, [middle]). The filled triangle corresponding to the triple pairwise intersection is topologically trivial and does not affect the topology (and similarly darker tetrahedron on the right). For $\varepsilon = \varepsilon_3$ all the components merged into one and the same hole still exists (b, [right]). In the interval $[\varepsilon_2, \varepsilon_3]$ one smaller hole as on fig. 5-3 appeared and quickly disappeared. This information can be conveniently summarized in the *persistence barcode* (c). The number of connected components (holes) in the simplicial complex for some value ε_0 is given by the number of intervals in H_0 (H_1) intersecting the vertical line $\varepsilon = \varepsilon_0$.

simplicial complex, namely *homology* which encodes the number of holes of various dimensions in a space. Controlling the value of ε allows us to decide holes of which size are meaningful and should not be discarded as a noise. For simplicial complex presented on fig. 5-3 there are two one-dimensional holes, and for slightly bigger value of ε the lower hole disappeared (fig. 5-4, b), while the top one remained intact, which suggests that the top hole is more important topological feature. Information about how homology is changing with respect to ε can be conveniently encoded in the so-called *persistence barcodes* [Ghrist, 2008, Zomorodian and Carlsson, 2005].

An example of such barcode is given on (fig. 5-4, c). In general, to find the rank of k-homology (delivering the number of k-dimensional holes) at some fixed value ε_0 one has to count intersections of the vertical line $\varepsilon = \varepsilon_0$ with the intervals at the desired block H_k.

These barcodes provide a way to compare topological properties of the underlying manifolds. In principle, we could obtain a metric of similarity of two datasets by comparing the barcodes of the simplicial complexes constructed based on each dataset (as described on fig. 5-3), but there are disadvantages of this approach, such as a huge number of simplices for large datasets. Moreover, in order to extract interesting topological properties from such large simplicial complexes various tricks are required [Ghrist, 2008]. To remedy these issues we can note that we are in fact interested in *topological* approximations rather than *geometrical*. The difference is that to obtain a correct estimate of the topological properties much smaller number of simplices is often sufficient, e.g., for any number of points sampled from a circle the correct answer could be obtained by taking just three points (thus obtaining a triangle which is topologically equivalent to the circle). Based on these ideas the so-called **witness complex** is introduced [De Silva and Carlsson, 2004], which provides a topological approximation with a small number of simplices. In order to achieve this a small subset of *landmark* points is chosen and a simplicial complex is constructed using these points as vertices (while also taking into account the proximity information about all the remaining points called *witnesses*).

To construct a numerical measure which could be compared across datasets we would like to estimate the correct values of homology. Comparing the computed barcodes is a challenging task since they are non-trivial mathematical objects (though some metrics exist they are hard to compute). We take the simpler route and to extract meaningful topological data from the barcode we propose computing **Relative Living Times** (RLT) of each number of holes that was observed. They are defined as the ratio of the total time when this number was present and of the value $\varepsilon_{\max}$ when points connect into a single blob. These relative living times could be interpreted as a *confidence* in our approximation — if say for 50% of all period of topological activity we have observed that there is at least 1 one-dimensional hole (as

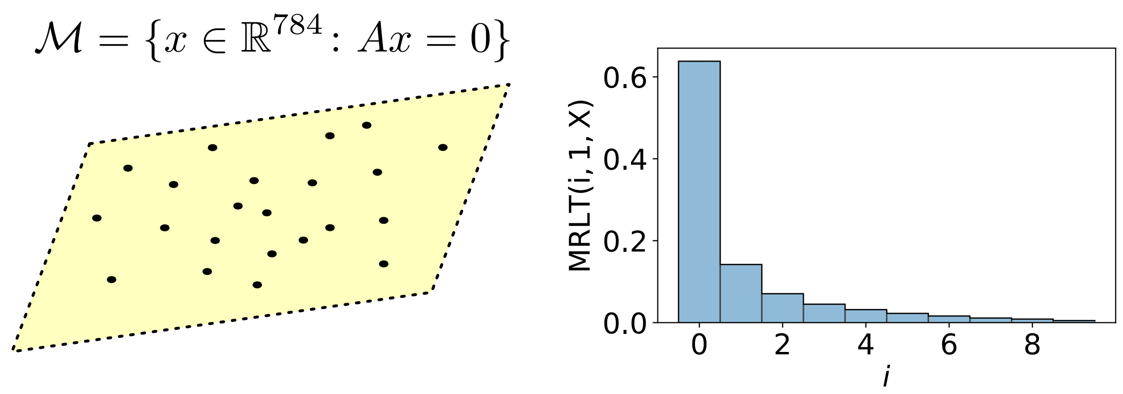


Figure 5-5: Estimation of the topology of a dataset sampled from the 32-dimensional hyperplane in 784-dimensional space. With high confidence, we can say that there are no 1-dimensional holes. For details see section 5.4.

on fig. 5-4), then it is probably an accurate estimation of topology of the underlying space.

Choosing the correct landmarks is a nontrivial task. We follow the discussion in [De Silva and Carlsson, 2004] which advises doing it randomly. To account for this randomness, we compute the RLT stochastically by repeating the experiment a large number of times. By averaging the obtained RLT we compute the **Mean Relative Living Times** (MRLT). By construction, they add up to 1 and employing Bayesian point of view we can interpret them as a probability distribution reflecting our confidence about the correct number of holes on *average*. An example of such distribution is given on fig. 5-5, where we run our method for a simple planar dataset (in a high dimensional space). To quantitatively evaluate the topological difference between two datasets we propose computing the L_2–error between these distributions. Note that in practice (when activation functions such as ReLU are used) the resulting space $\mathcal{M}_{\text{model}}$ may fail to be a manifold in precise mathematical sense, however, the analysis is still applicable since it deals with arbitrary topological spaces. Now let us introduce all the technical details.

5.3 Homology to the rescue

In this section we briefly discuss the important concepts of simplicial complexes and homology. For thorough introduction we refer the reader to the classical texts such as [Hatcher, 2002, May, 1999].

Simplicial complexes Simplicial complex is a classical concept widely used in topology. Formally it is defined as follows.

Definition 1. *A simplicial complex $\mathcal{S}$ (more precisely an* abstract *simplicial complex) is specified by the following data:*

- *The vertex set $Z = \{z_1, z_2, \dots, z_n\}$*
- *A collection of simplices Σ, where p-dimensional simplex σ_p is defined just as a $p+1$ element subset of Z:*

$$\sigma_p = \{z_{i_1}, z_{i_2}, \dots, z_{i_{p+1}}\}$$

- *We require that the collection Σ is closed under taking* faces, *that is for each p-dimensional simplex σ_p all the $(p-1)$-dimensional simplices obtained by deleting one of the vertices $z_{i_1}, \dots, z_{i_p}$ are also elements of Σ.*

An example of a simplicial complex $\mathcal{S}$ is presented on fig. 5-2. It contains 5 vertices $\{z_1, z_2 \dots, z_5\}$ and several edges and faces: two-dimensional face σ_2 and one-dimensional edges $\sigma_1, \sigma_3, \sigma_4$. Note that these are *maximal* simplices, since by the third property all the edges of σ_2 are also elements of $\mathcal{S}$. Important topological properties of $\mathcal{S}$ (such as connectedness, existence of one-dimensional loop) do not depend on in which Euclidean space $\mathcal{S}$ is embedded or on precise positions of vertices, but merely on the combinatorial data — the number of points and which vertices together span a simplex.

As was described in section 5.2 given a dataset X sampled from a manifold $\mathcal{M}$ we would like to compute a family of simplicial complexes topologically approximating $\mathcal{M}$ on various scales, namely witness complexes. This family is defined as follows. First we choose some subset $L \subset X$ of points called landmarks (whereas points in X are called witnesses) and some distance function $d(x, x')$, e.g., the ordinary Euclidean distance. There is not much theory about how to choose the best landmarks, but several strategies were proposed in [De Silva and Carlsson, 2004]. The first one is to choose landmarks sequentially by solving a certain minimax problem, and the second one is to just pick landmarks at random (by uniformly selecting a fixed number of points from X). We follow the second approach since the minimax strategy is known to have some flaws such as the tendency to pick up outliers. The selected landmarks will serve as the vertices of the simplicial complex and witnesses will help to decide on which simplices are inserted via a predicate "is witnessed":

$$\begin{aligned} &\sigma \subset L \text{ is witnessed by } w \in X \text{ if } \forall l \in \sigma, \forall l' \in L \setminus \sigma \\ &d(w,l)^2 \leq d(w,l')^2 + \alpha, \end{aligned} \tag{5.1}$$

with α being a relaxation parameter which provides us with a sequence of simplicial complexes. The maximal value of α for the analysis is typically chosen to be proportional to the maximal pairwise distance between points in L. Witness complexes even for small values of α are good topological approximations to $\mathcal{M}$. The main advantage of a witness complex is that it allows constructing a reliable approximation using a relatively small number of simplices and makes the problem tractable even for large datasets. Even though it is known that in some cases it may fail to recover the correct topology [Boissonnat et al., 2009], it still can be used to compare topological properties of datasets, and if any better method is devised, we can easily replace the witness complex by this new more reliable simplicial complex.

Homology The precise definition of the homology is technical, and we have to omit it due to the limited space. We refer the reader to [Chapter 2] [Hatcher, 2002] for a thorough discussion. The most important properties of homology can be summarized as follows. For any topological space X the so-called i^{th} homology groups H_i are introduced. The actual number of i-dimensional holes in X is given the *rank* of H_i, the concept which is quite similar to the dimension of a vector space. These ranks are called the *Betti numbers* β_i and serve as a coarse numerical measure of homology.

Homology is known to be one of the most easily computable topological invariants. In the case of X being a simplicial complex H_i can be computed by pretty much linear algebra, namely by analyzing kernels and images of certain linear maps. Dimensions of matrices appearing in this task are equal to the numbers d_k of simplices of specific dimension k in X, e.g. in the case of fig. 5-2 we have $d_0 = 5, d_1 = 6, d_2 = 1$ and matrices will be of sizes 6×1 and 5×6. Existent algorithms [Kaczynski et al., 2006] can handle extremely large simplicial complexes (with millions of simplices) and are available in numerous software packages. An important property of homology is that k^{th} homology depends only on simplices of dimension at most $k + 1$, which significantly speeds up computations.

Persistent homology In section 5.2 we discussed that to find a proxy of the correct topology of $\mathcal{M}$ it is insufficient to use single simplicial complex but rather a family of simplicial complexes is required. As we transition from one simplicial complex to another, some holes may appear, and some disappear. To distinguish between which are essential and which should be considered noise the concept of persistence was introduced [Edelsbrunner et al., 2000, Zomorodian and Carlsson, 2005]. The formal Structure Theorem [Zomorodian and Carlsson, 2005] states that for each generator of homology ("hole" in our notation) one could provide the time of its "birth" and "death". This data is pictorially represented as (fig. 5-4, [bottom]), with the horizontal axis representing the parameter and the vertical axis representing various homology generators. To perform the computation of these barcodes, an efficient algorithm was proposed in [Zomorodian and Carlsson, 2005]. As an input to this algorithm one has to supply a sequence of $(\sigma_i, \varepsilon_i)$, with σ_i being a simplex and ε_i being its time of appearance in a family. This algorithm is implemented in several software packages such as `Dionysus` and `GUDHI` [Maria et al., 2014], but the witness complex is supported only in the latter.

5.4 Algorithm

Let us now explain how we apply these concepts to construct a metric to compare the topological properties of two datasets. First let us define the key part of the algorithm – the relative living times (RLT) of homology. Suppose that for a dataset X and some choice of landmarks L we have obtained a persistence barcode with the persistence parameter α spanning the range $[0, \alpha_{\max}]$. Let us fix the dimension k in which we study the homology, and let $\mathcal{I}_k = \{[b_i, d_i]\}_{i=1}^n$ be the collection of persistence intervals in this dimension. Then in order to find the k^{th} Betti number for a fixed value α one has to count the number of persistence intervals containing α, and we obtain the integer valued function

$$\beta_k(\alpha) \triangleq |\{[b_i, d_i] \in \mathcal{I}_k \colon \alpha \in [b_i, d_i]\}|. \tag{5.2}$$

Then the RLT are defined as follows (for non-negative integers i):

$$\mathrm{RLT}(i, k, X, L) \triangleq \frac{\mu(\{\alpha \in [0, \alpha_{\max}] \colon \beta_k(\alpha) = i\})}{\alpha_{\max}}, \tag{5.3}$$

that it is for each possible value of $\beta_k(\alpha)$ we find how long it existed *relatively* to the whole period of topological activity. Note that in our analysis we use witness complexes which depend on the choice of landmarks, which is random. Thus it is reasonable to consider the distribution of $\mathrm{RLT}(i, k, X, L)$ on the set of landmarks (tuples of points), in other words, we repeatedly sample the landmarks and compute the RLT of the obtained persistence barcode. After sufficiently many experiments we can approximate the *Mean* Relative Living Times (MRLT):

$$\mathrm{MRLT}(i, k, X) \triangleq \mathbb{E}_L[\mathrm{RLT}(i, k, X, L)]. \tag{5.4}$$

We hypothesize that these quantities provide us with a good way to compare the topological properties of datasets, as they serve as measures of confidence in the estimation of the topology of the underlying manifolds. From eq. (5.3) it follows that

$$\sum_i \mathrm{MRLT}(i, k, X) = 1,$$

which suggest that for a fixed value of k we could interpret $\mathrm{MRLT}(i, k, X)$ as a *probability distribution* (over integers). This distribution defines our certainty about the number of k-dimensional holes in the underlying manifold of X on *average*. In this work we consider the case $k = 1$, i.e. we study the first homology of datasets.

We motivate this by drawing an analogy with the Taylor series: we can get a good understanding of behavior of a function by looking at the first term of the series (see also [Ghrist, 2008] for discussion). Based on the probabilistic understanding given two datasets X_1 and X_2 we define a measure of their topological similarity (*Geometry Score*) in the following way:

$$\text{GeomScore}(X_1, X_2) \triangleq \sum_{i=0}^{i_{\max}-1} \left(\text{MRLT}(i, 1, X_1) - \text{MRLT}(i, 1, X_2)\right)^2, \tag{5.5}$$

with $i_{\max}$ being an upper bound on $\beta_1(\alpha)$ for X_1 and X_2 (for typical datasets we found that $i_{\max} = 100$ suffices).

To construct the witness complex given the sets of landmarks L and witnesses X one has to provide the matrix of pairwise distances between L and X and the maximal value of persistence parameter α (see eq. (5.1)). In our experiments, we have chosen $\alpha_{\max}$ to be proportional to the maximal pairwise distance between points in L with some coefficient γ. Since we only compute $\beta_1(\alpha)$ the simplices of dimension at most 2 are needed. In principle to compare two datasets any value of γ suffices, however in our experiments we found that to get a reasonable distribution for datasets of size ~ 5000 the value $\frac{1}{128}$ yields good results (for large γ a lot of time is spend in the regime of a single connected blob which shifts the distributions towards 0). We summarize our approach in algorithm 3 and algorithm 4. We also suggest that to obtain accurate results datasets of the same size should be used for comparison

Complexity Let us briefly discuss the complexity of each step in the main loop of algorithm 3. Suppose that we have a dataset $X \in \mathbb{R}^{N \times D}$. Computing the matrix of pairwise distances between all points in the dataset and the landmarks points requires $O(NDL_0)$ operations. The complexity of the next piece involving computing the persistence barcode is hard to estimate, however we can note that it does not depend on the dimensionality D of the data. In practice this computation is done faster than computing the matrix in the previous step (for datasets of significant

Algorithm 3 The algorithm to compute RLT of a dataset. See section 5.4 for details. Suggested default values of the parameters for a dataset $X \in \mathbb{R}^{N\times D}$ are $L_0 = 64$, $\gamma = \frac{1}{128}/\frac{N}{5000}$, $i_{\max} = 100$, $n = 10000$.

```
Require: $X$: $2D$ array representing the dataset
Require: $L_0$: Number of landmarks to use
Require: $\gamma$: Coefficient determining $\alpha_{\max}$
Require: $i_{\max}$: Upper bound on $i$ in $\mathrm{RLT}(i, 1, X, L)$
Require: $n$: Number of experiments
Require: dist($A$, $B$): Function computing the matrix of pairwise (Euclidean) distances between samples from $A$ and $B$
Require: witness($d$, $\alpha$, $k$): Function computing the family of witness complexes using the matrix of pairwise distances $d$, maximal value of persistence parameter $\alpha$ and maximal dimension of simplices $k$
Require: persistence($w, k$): Function computing the persistence intervals of a family $w$ in dimension $k$
Returns: An array of size $n \times i_{\max}$ of the obtained RLT for each experiment
Initialize: rlt = zeros($n$, $i_{\max}$)
for $i = 0$ to $n - 1$ do
    $L^{(i)} \leftarrow$ random_choice($X$, size=$L_0$)
    $d^{(i)} \leftarrow$ dist($L^{(i)}$, $X$)
    $\alpha^{(i)}_{\max} \leftarrow \gamma \cdot$ max(dist($L^{(i)}$, $L^{(i)}$))
    $W^{(i)} \leftarrow$ witness($d^{(i)}$, $\alpha^{(i)}_{\max}$, 2)
    $\mathcal{I}^{(i)} \leftarrow$ persistence($W^{(i)}$, 1)
    for $j = 0$ to $i_{\max} - 1$ do
        Compute $\mathrm{RLT}(j, 1, X, L^{(i)})$ using eqs. (5.2) and (5.3)
        rlt[$i$, $j$] $\leftarrow \mathrm{RLT}(j, 1, X, L^{(i)})$
    end for
end for
```

Algorithm 4 *Geometry Score*, the proposed algorithm to compute topological similarity between datasets

```
Require: $X_1, X_2$: arrays representing the datasets
Returns: $s$: a number representing the topological similarity of $X_1$ and $X_2$
Initialize: $s = 0$
For $X_1$ and $X_2$ run algorithm 3 with the same collection of parameters, obtaining arrays $\mathrm{rlt}_1$ and $\mathrm{rlt}_2$
$\mathrm{mrlt}_1 \leftarrow$ mean($\mathrm{rlt}_1$, axis=0)
$\mathrm{mrlt}_2 \leftarrow$ mean($\mathrm{rlt}_2$, axis=0)
$s \leftarrow$ sum(($\mathrm{mrlt}_1 - \mathrm{mrlt}_2)^2$)
```

dimensionality). All the remaining pieces of the algorithm take negligible amount of time. This linear scaling of the complexity w.r.t dimensionality of the data allows us to apply our method even for high–dimensional datasets. On a typical laptop

(3.1 GHz Intel Core i5 processor) one iteration of the inner loop of algorithm 3 for one class of the MNIST dataset takes approximately 900 ms.

5.5 Experiments

Experimental setup We have implemented algorithms 3 and 4 in `Python` using `GUDHI`[1] for computing witness complexes and persistence barcodes. Our code is available on Github[2]. Default values of parameters in algorithm 3 were used for experiments unless otherwise specified. We test our method on several datasets and GAN models:

- **Synthetic data** — on synthetic datasets we demonstrate that our method allows for distinguishing the datasets based on their topological properties.
- **MNIST** — as the next experiment we test our approach on the MNIST dataset of handwritten digits. We compare two recently proposed models: WGAN [Arjovsky et al., 2017] and WGAN-GP [Gulrajani et al., 2017] in order to verify if the improved model WGAN-GP indeed produces better images.
- **CelebA** — to demonstrate that our method can be applied to datasets of large dimensionality we analyze the CelebA dataset [Liu et al., 2015] and check if we can detect mode collapse in a GAN using MRLT.
- **CaloGAN** — as the final experiment we apply our algorithm to a dataset of a non-visual origin and evaluate the specific generative model CaloGAN [Paganini et al., 2017].

Synthetic data For this experiment we have generated a collection of simple $2D$ datasets $\{X_j\}_{j=1}^5$ (see fig. 5-6) each containing 5000 points. As a test problem we would like to evaluate which of the datasets $\{X_j\}_{j=2}^5$ is the best approximation to the ground truth X_1. For each of $\{X_j\}_{j=1}^5$ we ran algorithm 3 using $L_0 = 32,\ n = 2000,\ i_{\max} = 3,\ \gamma = \frac{1}{8}$ and compute MRLT using eq. (5.4). The

[1]http://gudhi.gforge.inria.fr/
[2]https://github.com/KhrulkovV/geometry-score

resulting distributions are visualized on fig. 5-6, [bottom]. We observe that we can correctly identify the number of 1-dimensional holes in each space X_j using the MAP estimate

$$\beta_1^*(X_j) = \underset{i}{arg\,max}\, \mathrm{MRLT}(i, 1, X_j). \tag{5.6}$$

It is clear that X_4 is the most similar dataset to X_1, which is supported by the fact that their MRLT are almost identical. Note that on such simple datasets we were able to recover the correct homology with almost 100% confidence and this will not be the case for more complicated manifolds in the next experiment.

MNIST In this experiment we compare topological properties of the MNIST dataset and samples generated by the WGAN and WGAN-GP models trained on MNIST. It was claimed that the WGAN-GP model produces better images and we would like to verify if we can detect it using topology. For the GAN implementations we used the code[3] provided by the authors of [Gulrajani et al., 2017]. We have trained each model for 25 epochs and generated 60000 samples. To compare topology of each class individually we trained a CNN classifier on MNIST (with 99.5% test accuracy) and split generated datasest into classes (containing roughly 6000 images each). For every class and each of the 3 corresponding datasets (‘base’, ‘wgan’, ‘wgan–gp’) we run algorithm 3 and compute MRLT with $\gamma = \frac{1}{128}$. Similarly we evaluate MRLT for the entire datasets without splitting them into classes using $\gamma = \frac{1}{1000}$. The obtained MRLT are presented on fig. 5-7 and the corresponding *Geometry Scores* for each model are given in table 5.1. We observe that both models produce distributions which are very close to the ground truth, but for almost all classes WGAN-GP shows better scores. We can also note that for the entire datasets (fig. 5-7, [right]) the predicted values of homology does not seem to be much bigger than for each individual digit. One possible explanation is that some samples (like say of class ‘7’) fill the holes in the underlying manifolds of other classes (like class ‘1’ in this case) since they look quite similar.

[3] https://github.com/igul222/improved_wgan_training

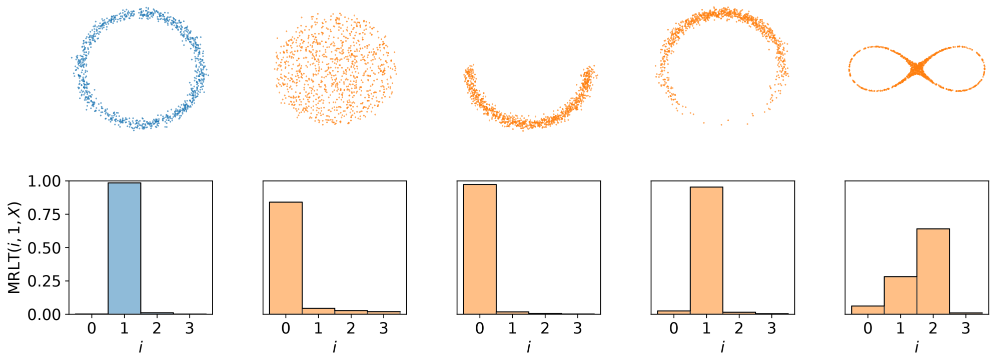


Figure 5-6: Mean Relative Living Times (MRLT) for various $2D$ datasets. The number of one-dimensional holes is correctly identified in all the cases. By comparing MRLT we find that the second dataset from the left is the most similar to the 'ground truth' (noisy circle on the left).

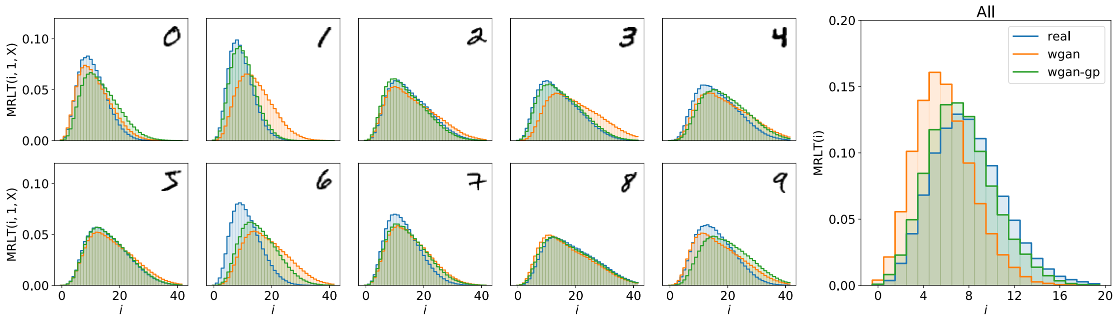


Figure 5-7: Comparison of MRLT of the MNIST dataset and of samples generated by WGAN and WGAN-GP trained on MNIST. MRLT match almost perfectly, however, WGAN-GP shows slightly better performance on most of the classes.

Table 5.1: *Geometry Scores* $\times 10^3$ of WGAN and WGAN-GP trained on the MNIST dataset (see also fig. 5-7). Each class contained roughly 6000 images, except for 'All' which corresponds to the total datasets of 60000 images.

Label	0	1	2	3	4	5	6	7	8	9	All
WGAN	**0.85**	21.4	0.60	7.04	**1.52**	0.47	22.8	2.20	0.76	**1.27**	26.1
WGAN-GP	5.19	**1.44**	**0.54**	**0.27**	2.16	**0.03**	**13.5**	**1.38**	**0.14**	5.00	**2.04**

CelebA We now analyze the popular CelebA dataset consisting of photos of various celebrities. In this experiment we would like to study if we can detect mode collapse using our method. To achieve this we train two GAN models — a good model with the generator having high capacity and a second model with the generator much weaker than the discriminator. In this experiment we utilize the DCGAN model and use the implementation provided[4] by the authors [Radford et al., 2015]. For the first model ('dcgan') we use the default settings, and for the second ('bad-dcgan') we set the latent dimension to 8 and reduce the size of the fully connected layer in the generator to 128 and number of filters in convolutional layers to 4. Images in the dataset are of size 108×108 and to obtain faces we perform the central crop which reduces the size to 64×64. We trained both models for 25 epochs and produced 10000 images for our analysis. Similarly, we randomly picked 10000 (cropped) images from the original dataset. We report the obtained results on fig. 5-8. MRLT obtained using the good model matches the ground truth almost perfectly and *Geometry Score* of the generated dataset is equal to 14×10^{-3}, confirming the good visual quality of the samples [Radford et al., 2015]. MRLT obtained using the weak model are maximized for $i = 0$, which suggests that the samples are either identical or present very little topological diversity (compare with fig. 5-5), which we confirmed visually. On fig. 5-8, [right] we report the behavior of the *Geometry Score* and *Inception Score* [Salimans et al., 2016b] w.r.t the iteration number. The *Inception Score* introduced uses the pretrained Inception network [Szegedy et al., 2015a] and is defined as

$$I(\{x_n\}_{n=1}^N) \triangleq \exp \mathbb{E}_{\mathbf{x}}(\mathrm{D}_{\mathrm{KL}}(p(y|\mathbf{x})||p(y))),$$

where $p(y|\mathbf{x})$ is approximated by the Inception network and $p(y)$ is computed as $p(y) = \frac{1}{N}\sum_i p(y|\mathbf{x}_i)$. Note that the *Geometry Score* of the better model rapidly decreases and of the mode collapsed model stagnates at high values. Such behavior could not be observed in the *Inception Score*.

CaloGAN In this experiment, we will apply our technique to the dataset appearing in the experimental particle physics. This dataset[5] represents a collection of a calorimeter (an experimental apparatus measuring the energy of particles) responses, and it was used to create a generative model [Paganini et al., 2017] in order to help physicists working at the LHC. Evaluating the obtained model[6] is a non-

[4]https://github.com/carpedm20/DCGAN-tensorflow
[5]https://data.mendeley.com/datasets/pvn3xc3wy5/1
[6]https://github.com/hep-lbdl/CaloGAN

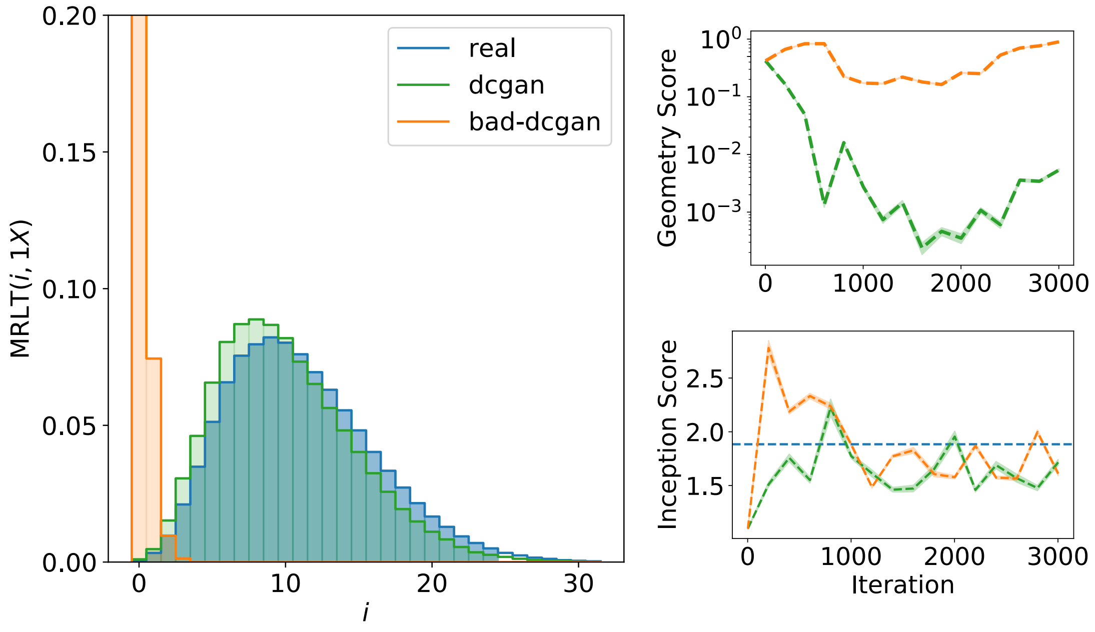


Figure 5-8: MRLT of the (cropped) CelebA dataset and samples generated using DCGAN and DCGAN with forced mode collapse. Plots on the right present the behavior of the *Geometry Score* and *Inception Score* for these two models during the training. Mode collapse in 'bad-dcgan' is easily observable using the *Geometry Score*.

trivial task and was performed by comparing physical properties of the obtained and the real data. Since our method is not limited to visual datasets we can apply it in order to confirm the quality of this model. For the analysis we used 'eplus' dataset which is split into 3 parts ('layer 0', 'layer 1', 'layer 2') containing matrices of sizes $3 \times 96, 12 \times 12, 12 \times 6$ correspondingly. We train the CaloGAN model with default settings for 50 epochs and generate 10000 samples (each sample combines data for all 3 layers). We then randomly pick 10000 samples from the original dataset and compare MRLT of the data and generated samples for each layer. Results are presented on fig. 5-9. It appears that topological properties of this dataset are rather trivial, however, they are correctly identified by CaloGAN. Slight dissimilarities between the distributions could be connected to the fact that the physical properties of the generated samples do not exactly match those of the real ones, as was analyzed by the authors of [Paganini et al., 2017].

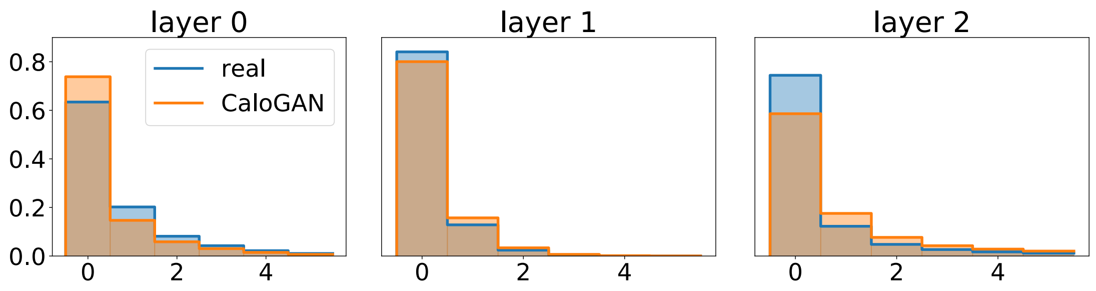


Figure 5-9: MRLT of the dataset used in experimental particle physics and of the samples generated using the corresponding CaloGAN model.

5.6 Related work and discussion

Several performance measures have been introduced to assess the performance of GANs used for natural images. *Inception Score* [Salimans et al., 2016b] uses the outputs of the pretrained Inception network, and a modification called *Fréchet Inception Distance (FID)* [Heusel et al., 2017] also takes into account second order information of the final layer of this model. Contrary to these methods, our approach does not use auxiliary networks and is not limited to visual data. We note, however, that since we only take topological properties into account (which do not change if we say shift the entire dataset by 1) assessing the *visual* quality of samples may be difficult based only on our algorithm, thus in the case of natural images we propose to use our method in conjunction with other metrics such as *FID*. We also hypothesize that in the case of the large dimensionality of data *Geometry Score* of the features extracted using some network will adequately assess the performance of a GAN.

5.7 Conclusion

We have introduced a new algorithm for evaluating a generative model. We show that the topology of the underlying manifold of generated samples may be different from the topology of the original data manifold, which provides insight into properties of GANs and can be used for hyperparameter tuning. We do not claim however that the obtained metric correlates with the visual quality as estimated by humans and leave the analysis to future work. We hope that our research will be useful to further theoretical understanding of GANs.

Chapter 6

Hyperbolic Image Embeddings

6.1 Introduction

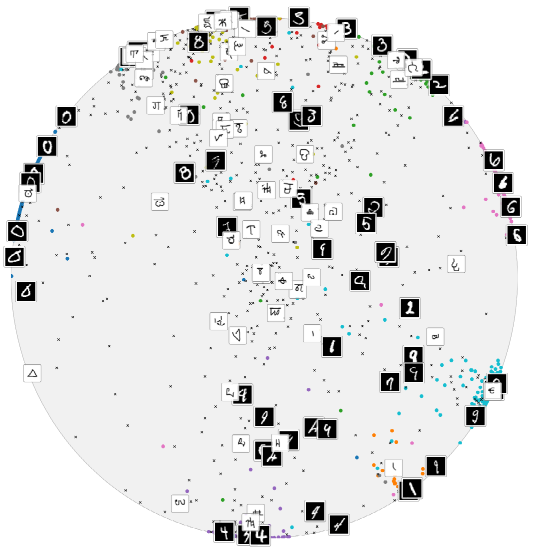


Figure 6-1: An example of two–dimensional Poincaré embeddings computed by a hyperbolic neural network trained on MNIST, and evaluated additionally on Omniglot. Ambiguous and unclear images from MNIST, as well as most of the images from Omniglot, are embedded near the center, while samples with clear class labels (or characters from Omniglot similar to one of the digits) lie near the boundary. *For inference, Omniglot was normalized to have the same background color as MNIST. Omniglot images are marked with black crosses, MNIST images with colored dots.

Learned high-dimensional embeddings are ubiquitous in modern computer vision. Learning aims to group together semantically-similar images and to separate semantically-different images. When the learning process is successful, simple classifiers can be used to assign an image to classes, and simple distance measures can be used to assess the similarity between images or image fragments. The operations at the end of deep networks imply a certain type of geometry of the embedding spaces. For example, image classification networks [Krizhevsky et al., 2012, LeCun et al., 1989] use linear operators (matrix multiplication) to map embeddings in the penultimate layer to class logits. The class boundaries in the embedding space are thus piecewise-linear, and pairs of classes are separated by Euclidean hyperplanes. The embeddings learned by the model in the penultimate layer, therefore, live in the Euclidean space. The same can be said about systems where Euclidean distances are used to perform image retrieval [Oh Song et al., 2016, Sohn, 2016, Wu et al., 2017], face recognition [Parkhi et al., 2015, Wen et al., 2016] or one-shot learning [Snell et al., 2017].

Alternatively, some few-shot learning [Vinyals et al., 2016], face recognition [Schroff et al., 2015], and person re-identification methods [Ustinova and Lempitsky, 2016, Yi et al., 2014] learn spherical embeddings, so that sphere projection operator is applied at the end of a network that computes the embeddings. Cosine similarity (closely associated with sphere geodesic distance) is then used by such architectures to match images.

Euclidean spaces with their zero curvature and spherical spaces with their positive curvature have certain profound implications on the nature of embeddings that existing computer vision systems can learn. In this work, we argue that hyperbolic spaces with negative curvature might often be more appropriate for learning embedding of images. Towards this end, we add the recently-proposed hyperbolic network layers [Ganea et al., 2018] to the end of several computer vision networks, and present a number of experiments corresponding to image classification, one-shot, and few-shot learning and person re-identification. We show that in many cases, the use of hyperbolic geometry improves the performance over Euclidean or spherical embeddings.

Our work is inspired by the recent body of works that demonstrate the advantage of learning hyperbolic embeddings for language entities such as taxonomy entries [Nickel and Kiela, 2017], common words [Tifrea et al., 2018], phrases [Dhingra et al., 2018] and for other NLP tasks, such as neural machine translation [Gul-

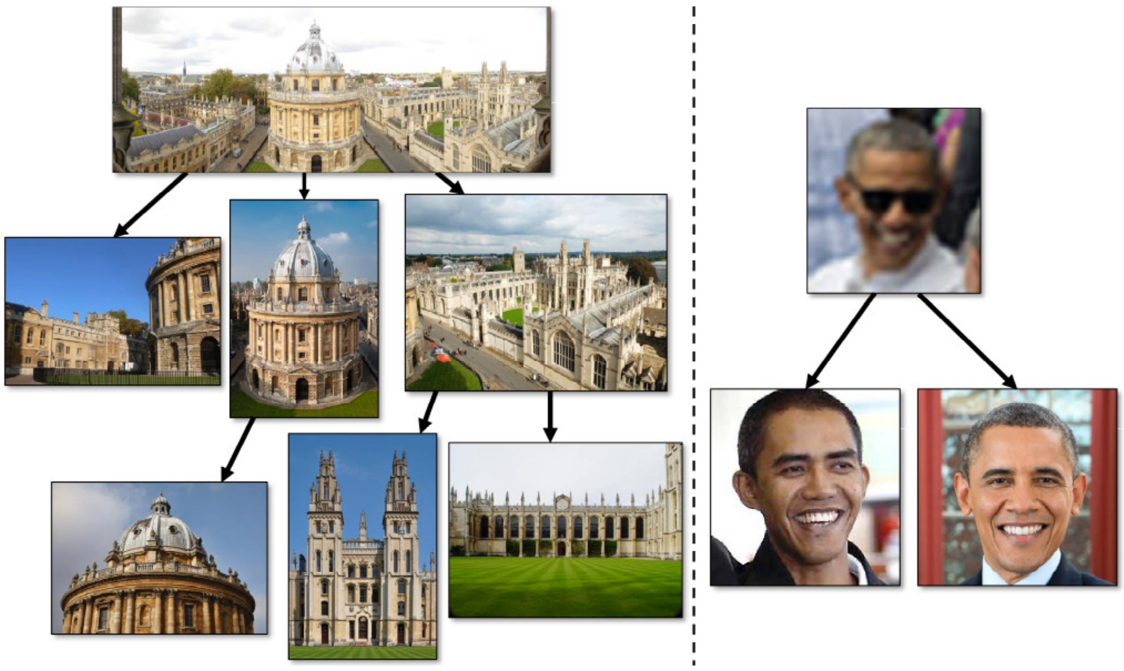


Figure 6-2: In many computer vision tasks, we want to learn image embeddings that obey the hierarchical constraints. E.g., in image retrieval (left), the hierarchy may arise from whole-fragment relation. In recognition tasks (right), the hierarchy can arise from image degradation, when degraded images are inherently ambiguous and may correspond to various identities/classes. Hyperbolic spaces are more suitable for embedding data with such hierarchical structure.

cehre et al., 2019]. Our results imply that hyperbolic spaces may be as valuable for improving the performance of computer vision systems.

Motivation for hyperbolic image embeddings. The use of hyperbolic spaces in natural language processing [Nickel and Kiela, 2017, Tifrea et al., 2018, Dhingra et al., 2018] is motivated by the ubiquity of hierarchies in NLP tasks. Hyperbolic spaces are naturally suited to embed hierarchies (e.g., tree graphs) with low distortion [Sarkar, 2011, Sala et al., 2018]. Here, we argue that hierarchical relations between images are common in computer vision tasks (Figure 6-2):

- In image retrieval, an overview photograph is related to many images that correspond to the close-ups of different distinct details. Likewise, for classification tasks in-the-wild, an image containing the representatives of multiple classes is related to images that contain representatives of the classes in isolation. Embedding a dataset that contains composite images into continuous space is, therefore, similar to embedding a hierarchy.

- In some tasks, more generic images may correspond to images that contain less information and are therefore more ambiguous. E.g., in face recognition, a blurry and/or low-resolution face image taken from afar can be related to many high-resolution images of faces that clearly belong to distinct people. Again natural embeddings for image datasets that have widely varying image quality/ambiguity calls for retaining such hierarchical structure.

- Many of the natural hierarchies investigated in natural language processing transcend to the visual domain. E.g., the visual concepts of different animal species may be amenable for hierarchical grouping (e.g. most felines share visual similarity while being visually distinct from pinnipeds).

Hierarchical relations between images call for the use of Hyperbolic spaces. Indeed, as the volume of hyperbolic spaces expands exponentially, it makes them continuous analogues of *trees*, in contrast to Euclidean spaces, where the expansion is polynomial. It therefore seems plausible that the exponentially expanding hyperbolic space will be able to capture the underlying hierarchy of visual data.

In order to build deep learning models which operate on the embeddings to hyperbolic spaces, we capitalize on recent developments [Ganea et al., 2018], which construct the analogues of familiar layers (such as a feed–forward layer, or a multinomial regression layer) in hyperbolic spaces. We show that many standard architectures used for tasks of image classification, and in particular in the few–shot learning setting can be easily modified to operate on hyperbolic embeddings, which in many cases also leads to their improvement.

The main contributions of our paper are twofold:

- First, we apply the machinery of hyperbolic neural networks to computer vision tasks. Our experiments with various few-shot learning and person re-identification models and datasets demonstrate that hyperbolic embeddings are beneficial for visual data.

- Second, we propose an approach to evaluate the hyperbolicity of a dataset based on the concept of Gromov δ-hyperbolicity. It further allows estimating the radius of Poincaré disk for an embedding of a specific dataset and thus can serve as a handy tool for practitioners.

6.2 Related work

Hyperbolic language embeddings. Hyperbolic embeddings in the natural language processing field have recently been very successful [Nickel and Kiela, 2017, 2018a]. They are motivated by the innate ability of hyperbolic spaces to embed hierarchies (e.g., tree graphs) with low distortion [Sala et al., 2018, Sarkar, 2011]. However, due to the discrete nature of data in NLP, such works typically employ Riemannian optimization algorithms in order to learn embeddings of individual words to hyperbolic space. This approach is difficult to extend to visual data, where image representations are typically computed using CNNs.

Another direction of research, more relevant to the present work, is based on imposing hyperbolic structure on *activations of neural networks* [Ganea et al., 2018, Gulcehre et al., 2019]. However, the proposed architectures were mostly evaluated on various NLP tasks, with correspondingly modified traditional models such as RNNs or Transformers. We find that certain computer vision problems that heavily use image embeddings can benefit from such hyperbolic architectures as well. Concretely, we analyze the following tasks.

Few–shot learning. The task of few–shot learning is concerned with the overall ability of the model to generalize to unseen data during training. Most of the existing state-of-the-art few–shot learning models are based on metric learning approaches, utilizing the distance between image representations computed by deep neural networks as a measure of similarity [Vinyals et al., 2016, Snell et al., 2017, Sung et al., 2018, Nichol and Schulman, 2018, Chen et al., 2019a, Chu et al., 2019, Li et al., 2019, Bauer et al., 2017, Rusu et al., 2019, Chen et al., 2019b]. In contrast, other models apply meta-learning to few-shot learning: e.g., MAML by [Finn et al., 2017], Meta-Learner LSTM by [Ravi and Larochelle, 2016], SNAIL by [Mishra et al., 2017]. While these methods employ either Euclidean or spherical geometries (like in [Vinyals et al., 2016]), there was no extension to hyperbolic spaces.

Person re-identification. The task of person re-identification is to match pedestrian images captured by possibly non-overlapping surveillance cameras. Papers [Ahmed et al., 2015, Guo and Cheung, 2018, Wang et al., 2018] adopt the pairwise models that accept pairs of images and output their similarity scores. The resulting similarity scores are used to classify the input pairs as being matching or non-matching. Another popular direction of work includes approaches that aim at learning a mapping of the pedestrian images to the Euclidean descriptor space. Several papers, e.g., [Suh et al., 2018, Yi et al., 2014] use verification loss functions based on the Euclidean distance or cosine similarity. A number of methods utilize a simple classification approach for training [Chang et al., 2018, Su et al., 2017, Kalayeh et al., 2018, Zhao et al., 2017], and Euclidean distance is used in test time.

6.3 Reminder on hyperbolic spaces and hyperbolicity estimation.

Formally, n-dimensional hyperbolic space denoted as $\mathbb{H}^n$ is defined as the homogeneous, simply connected n-dimensional Riemannian manifold of constant negative sectional curvature. The property of constant negative curvature makes it analogous to the ordinary Euclidean sphere (which has constant positive curvature); however, the geometrical properties of the hyperbolic space are very different. It is known that hyperbolic space cannot be isometrically embedded into Euclidean space [Krioukov et al., 2010, Linial et al., 1998], but there exist several well–studied *models* of hyperbolic geometry. In every model, a certain subset of Euclidean space is endowed with a *hyperbolic metric*; however, all these models are isomorphic to each other, and we may easily move from one to another base on where the formulas of interest are easier. We follow the majority of NLP works and use the *Poincaré ball* model.

The Poincaré ball model $(\mathbb{D}^n, g^{\mathbb{D}})$ is defined by the manifold $\mathbb{D}^n = \{\mathbf{x} \in \mathbb{R}^n \colon \|\mathbf{x}\| < 1\}$ endowed with the Riemannian metric $g^{\mathbb{D}}(\mathbf{x}) = \lambda_{\mathbf{x}}^2 g^E$, where $\lambda_{\mathbf{x}} = \frac{2}{1-\|\mathbf{x}\|^2}$ is the *conformal factor* and g^E is the Euclidean metric tensor $g^E = \mathbf{I}^n$. In this model the *geodesic distance* between two points is given by the following expression:

$$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}\left(1 + 2\frac{\|\mathbf{x}-\mathbf{y}\|^2}{(1-\|\mathbf{x}\|^2)(1-\|\mathbf{y}\|^2)}\right). \tag{6.1}$$

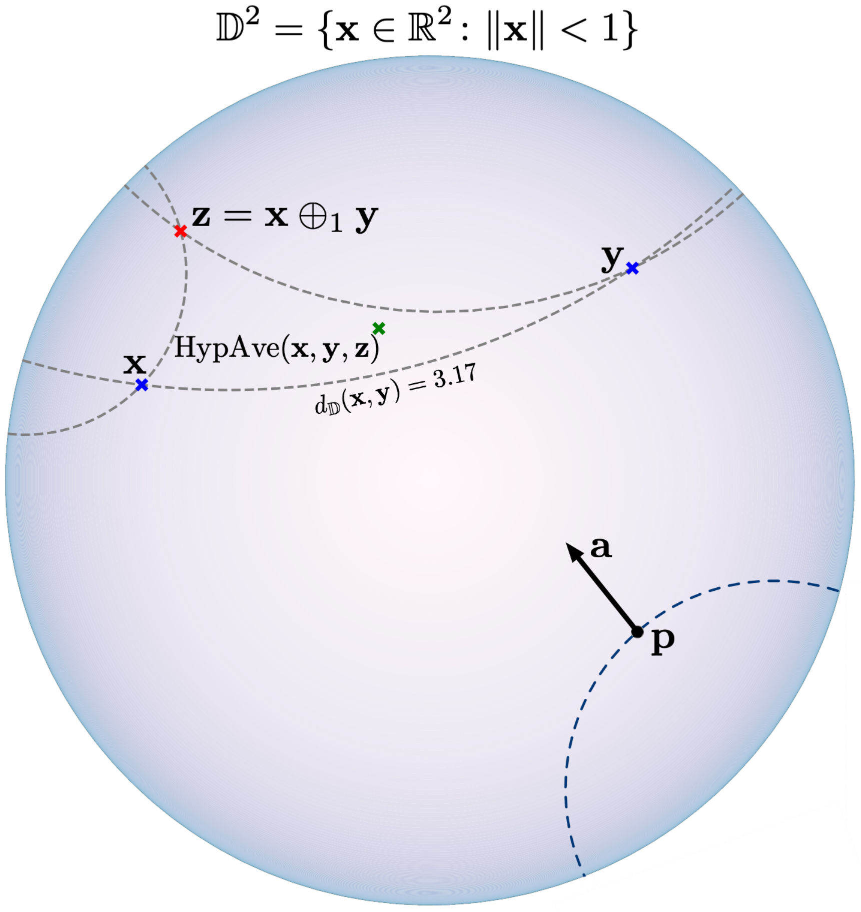

$\mathbb{D}^2 = \{\mathbf{x} \in \mathbb{R}^2 \colon \|\mathbf{x}\| < 1\}$
$\mathbf{z} = \mathbf{x} \oplus_1 \mathbf{y}$
$\mathbf{y}$
HypAve$(\mathbf{x}, \mathbf{y}, \mathbf{z})$
$\mathbf{x}$
$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = 3.17$
$\mathbf{a}$
$\mathbf{p}$

Figure 6-3: Visualization of the two–dimensional Poincaré ball. Point $\mathbf{z}$ represents the *Möbius sum* of points $\mathbf{x}$ and $\mathbf{y}$. HypAve stands for hyperbolic averaging. Gray lines represent *geodesics*, curves of shortest length connecting two points. In order to specify the *hyperbolic hyperplanes* (bottom), used for multiclass logistic regression, one has to provide an origin point $\mathbf{p}$ and a normal vector $\mathbf{a} \in T_{\mathbf{p}}\mathbb{D}^2 \setminus \{\mathbf{0}\}$. For more details on hyperbolic operations see Section 6.4.

In order to define the *hyperbolic average*, we will make use of the *Klein model* of hyperbolic space. Similarly to the Poincaré model, it is defined on the set $\mathbb{K}^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$, however, with a different metric, not relevant for further discussion. In Klein coordinates, the hyperbolic average (generalizing the usual Euclidean mean) takes the most simple form, and we present the necessary formulas in Section 6.4.

From the viewpoint of hyperbolic geometry, all points of Poincaré ball are equivalent. The models that we consider below are, however, hybrid in the sense that most layers use Euclidean operators, such as standard generalized convolutions, while only the final layers operate within the hyperbolic geometry framework. The hybrid nature of our setups makes the origin a special point, since, from the Euclidean viewpoint, the local volumes in Poincare ball expand exponentially from the origin to the boundary. This leads to the useful tendency of the learned embeddings to place more generic/ambiguous objects closer to the origin while moving more specific objects towards the boundary. The distance to the origin in our models, therefore, provides a natural estimate of uncertainty, that can be used in several ways, as we show below.

This choice is justified for the following reasons. First, many existing vision architectures are designed to output embeddings in the vicinity of zero (e.g., in the unit ball). Another appealing property of hyperbolic space (assuming the standard Poincare ball model) is the existence of a reference point – the center of the ball. We show that in image classification which construct embeddings in the Poincare model of hyperbolic spaces the distance to the center can serve as a measure of *confidence* of the model — the input images which are more familiar to the model get mapped closer to the boundary, and images which confuse the model (e.g., blurry or noisy images, instances of a previously unseen class) are mapped closer to the center. The geometrical properties of hyperbolic spaces are quite different from the properties of the Euclidean space. For instance, the sum of angles of a geodesic triangle is always less than π. These interesting geometrical properties make it possible to construct a "score" which for an arbitrary metric space provides a degree of similarity of this metric space to a hyperbolic space. This score is called δ-hyperbolicity, and we now discuss it in detail.

6.3.1 δ-Hyperbolicity

Let us start with an illustrative example. The simplest discrete metric space possessing hyperbolic properties is a *tree* (in the sense of graph theory) endowed with the natural shortest path distance. Note the following property: for any three ver-

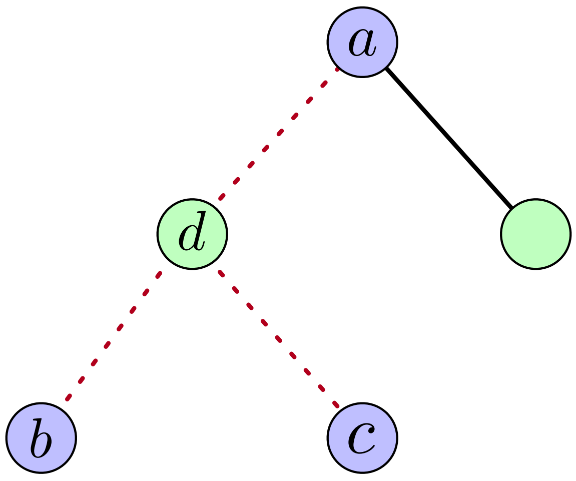


Figure 6-4: Visualization of a *geodesic triangle* in a tree. Such a tree endowed with a natural shortest path metric is a 0–Hyperbolic space.

tices a, b, c, the geodesic triangle (consisting of geodesics — paths of shortest length connecting each pair) spanned by these vertices (see Figure 6-4) is *slim*, which informally means that it has a center (vertex d) which is contained in every side of the triangle. By relaxing this condition to allow for some slack value δ and considering so-called δ-slim triangles, we arrive at the following general definition.

Table 6.1: Comparison of the theoretical degree of hyperbolicity with the relative delta δ_{rel} values estimated using Equations (6.2) and (6.4). The numbers are given for the two-dimensional Poincaré ball $\mathbb{D}^2$, the 2D sphere S_2, the upper hemisphere S_2^+, and a (random) tree graph.

	$\mathbb{D}^2$	S_2^+	S_2	Tree
Theory	0	1	1	0
δ_{rel}	0.18 ± 0.08	0.86 ± 0.11	0.97 ± 0.13	0.0

Table 6.2: The relative delta δ_{rel} values calculated for different datasets. For image datasets we measured the Euclidean distance between the features produced by various standard feature extractors pretrained on ImageNet. Values of δ_{rel} closer to 0 indicate a stronger hyperbolicity of a dataset. Results are averaged across 10 subsamples of size 1000. The standard deviation for all the experiments did not exceed 0.02.

Encoder	**Dataset**			
	CIFAR10	CIFAR100	CUB	*Mini*ImageNet
Inception v3 [Szegedy et al., 2015b]	0.25	0.23	0.23	0.21
ResNet34 [He et al., 2016]	0.26	0.25	0.25	0.21
VGG19 [Simonyan and Zisserman, 2014]	0.23	0.22	0.23	0.17

Let X be an arbitrary (metric) space endowed with the distance function d. Its δ-hyperbolicity value then may be computed as follows. We start with the so-called *Gromov product* for points $x, y, z \in X$:

$$(y, z)_x = \frac{1}{2}(d(x, y) + d(x, z) - d(y, z)). \tag{6.2}$$

Then, δ is defined as the minimal value such that the following four-point condition holds for all points $x, y, z, w \in X$:

$$(x, z)_w \geq \min((x, y)_w, (y, z)_w) - \delta. \tag{6.3}$$

The definition of hyperbolic space in terms of the Gromov product can be seen as saying that the metric relations between any four points are the same as they would be in a tree, up to the additive constant δ. δ-Hyperbolicity captures the basic common features of "negatively curved" spaces like the classical real-hyperbolic space $\mathbb{D}^n$ and of discrete spaces like trees.

For practical computations, it suffices to find the δ value for some fixed point $w = w_0$ as it is independent of w. An efficient way to compute δ is presented in [Fournier et al., 2015]. Having a set of points, we first compute the matrix A of pairwise Gromov products using Equation (6.2). After that, the δ value is simply the largest coefficient in the matrix $(A \otimes A) - A$, where $\otimes$ denotes the min-max matrix product

$$A \otimes B = \max_k \min\{A_{ik}, B_{kj}\}. \tag{6.4}$$

Results. In order to verify our hypothesis on hyperbolicity of visual datasets we compute the scale-invariant metric, defined as $\delta_{rel}(X) = \frac{2\delta(X)}{\mathrm{diam}(X)}$, where $\mathrm{diam}(X)$ denotes the set diameter (maximal pairwise distance). By construction, $\delta_{rel}(X) \in [0, 1]$ and specifies how close is a dataset to a hyperbolic space. Due to computational complexities of Equations (6.2) and (6.4) we employ the batched version of the algorithm, simply sampling N points from a dataset, and finding the corresponding δ_{rel}. Results are averaged across multiple runs, and we provide resulting mean and standard deviation. We experiment on a number of toy datasets (such as samples from the standard two–dimensional unit sphere), as well as on a number of popular computer vision datasets. As a natural distance between images, we used the standard Euclidean distance between feature vectors extracted by various CNNs pretrained on the ImageNet (ILSVRC) dataset [Deng et al., 2009]. Specifically, we consider VGG19 [Simonyan and Zisserman, 2014], ResNet34 [He et al., 2016] and Inception v3 [Szegedy et al., 2015b] networks for distance evaluation. While other metrics are possible, we hypothesize that the underlying hierarchical structure (useful for computer vision tasks) of image datasets can be well understood in terms of their deep feature similarity.

Our results are summarized in Table 6.2. We observe that the degree of hyperbolicity in image datasets is quite high, as the obtained δ_{rel} are significantly closer to 0 than to 1 (which would indicate complete non-hyperbolicity). This observation suggests that visual tasks can benefit from hyperbolic representations of images.

Relation between δ-hyperbolicity and Poincaré disk radius. It is known [Tifrea et al., 2018] that the standard Poincaré ball is δ-hyperbolic with $\delta_P = \log(1+\sqrt{2}) \approx 0.88$. Formally, the diameter of the Poincaré ball is infinite, which yields the δ_{rel} value of 0. However, from computational point of view we cannot approach the boundary infinitely close. Thus, we can compute the *effective* value of δ_{rel} for the Poincaré ball. For the clipping value of 10^{-5}, i.e., when we consider only the subset of points with the (Euclidean) norm not exceeding $1 - 10^{-5}$, the resulting diameter is equal to ~ 12.204. This provides the effective $\delta_{rel} \approx 0.144$. Using this constant we can estimate the radius of Poincaré disk suitable for an embedding of a specific dataset. Suppose that for some dataset X we have found that its δ_{rel} is equal to δ_X. Then we can estimate $c(X)$ as follows.

$$c(X) = \left(\frac{0.144}{\delta_X}\right)^2. \tag{6.5}$$

For the previously studied datasets, this formula provides an estimate of $c \sim 0.33$. In our experiments, we found that this value works quite well; however, we found that sometimes adjusting this value (e.g., to 0.05) provides better results, probably because the image representations computed by deep CNNs pretrained on ImageNet may not have been entirely accurate.

6.4 Hyperbolic operations

Hyperbolic spaces are not vector spaces in a traditional sense; one cannot use standard operations as summation, multiplication, etc. To remedy this problem, one can utilize the formalism of Möbius gyrovector spaces allowing to generalize many standard operations to hyperbolic spaces. Recently proposed hyperbolic neural networks adopt this formalism to define the hyperbolic versions of feed-forward networks, multinomial logistic regression, and recurrent neural networks [Ganea et al., 2018]. In Appendix 6.7, we discuss these networks and layers in detail, and in this section, we briefly summarize various operations available in the hyperbolic space. Similarly to the paper [Ganea et al., 2018], we use an additional hyperparameter c which modifies the curvature of Poincaré ball; it is then defined as $\mathbb{D}^n_c = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 < 1, c \geq 0\}$. The corresponding conformal factor now takes the form $\lambda^c_{\mathbf{x}} = \frac{2}{1-c\|\mathbf{x}\|^2}$. In practice, the choice of c allows one to balance between hyperbolic and Euclidean geometries, which is made precise by noting that with $c \to 0$, all the formulas discussed below take their usual Euclidean form. The following operations are the main building blocks of hyperbolic networks.

Möbius addition. For a pair $\mathbf{x}, \mathbf{y} \in \mathbb{D}^n_c$, the Möbius addition is defined as follows:

$$\mathbf{x} \oplus_c \mathbf{y} := \frac{(1 + 2c\langle \mathbf{x}, \mathbf{y}\rangle + c\|\mathbf{y}\|^2)\mathbf{x} + (1 - c\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2c\langle \mathbf{x}, \mathbf{y}\rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \tag{6.6}$$

Distance. The induced distance function is defined as

$$d_c(\mathbf{x}, \mathbf{y}) := \frac{2}{\sqrt{c}} \operatorname{arctanh}(\sqrt{c}\|-\mathbf{x} \oplus_c \mathbf{y}\|). \tag{6.7}$$

Note that with $c = 1$ one recovers the geodesic distance (6.1), while with $c \to 0$ we obtain the Euclidean distance $\lim_{c\to 0} d_c(\mathbf{x}, \mathbf{y}) = 2\|\mathbf{x} - \mathbf{y}\|$.

Exponential and logarithmic maps. To perform operations in the hyperbolic space, one first needs to define a bijective map from $\mathbb{R}^n$ to $\mathbb{D}^n_c$ in order to map Euclidean vectors to the hyperbolic space, and vice versa. The so-called exponential and (inverse to it) logarithmic map serves as such a bijection.

The *exponential* map $\exp^c_{\mathbf{x}}$ is a function from $T_{\mathbf{x}}\mathbb{D}^n_c \cong \mathbb{R}^n$ to $\mathbb{D}^n_c$, which is given by

$$\exp^c_{\mathbf{x}}(\mathbf{v}) := \mathbf{x} \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda^c_{\mathbf{x}}\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}\right). \tag{6.8}$$

The inverse *logarithmic* map is defined as

$$\log_{\mathbf{x}^c}(\mathbf{y}) := \frac{2}{\sqrt{c}\lambda^c_{\mathbf{x}}}\operatorname{arctanh}(\sqrt{c}\|-\mathbf{x}\oplus_c \mathbf{y}\|)\frac{-\mathbf{x}\oplus_c \mathbf{y}}{\|-\mathbf{x}\oplus_c \mathbf{y}\|}. \tag{6.9}$$

Hyperbolic averaging. One important operation common in image processing is averaging of feature vectors, used, e.g., in prototypical networks for few–shot learning [Snell et al., 2017]. In the Euclidean setting this operation takes the form $(\mathbf{x}_1, \dots, \mathbf{x}_N) \to \frac{1}{N}\sum_i \mathbf{x}_i$. Extension of this operation to hyperbolic spaces is called the *Einstein midpoint* and takes the most simple form in *Klein* coordinates:

$$\mathrm{HypAve}(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{i=1}^N \gamma_i \mathbf{x}_i / \sum_{i=1}^N \gamma_i, \tag{6.10}$$

where $\gamma_i = \frac{1}{\sqrt{1-c\|\mathbf{x}_i\|^2}}$ are the Lorentz factors. Recall from the discussion in Section 6.3 that the Klein model is supported on the same space as the Poincaré ball; however, the same point has different coordinate representations in these models. Let $\mathbf{x}_{\mathbb{D}}$ and $\mathbf{x}_{\mathbb{K}}$ denote the coordinates of the same point in the Poincaré and Klein models correspondingly. Then the following transition formulas hold.

$$\mathbf{x}_{\mathbb{D}} = \frac{\mathbf{x}_{\mathbb{K}}}{1+\sqrt{1-c\|\mathbf{x}_{\mathbb{K}}\|^2}}, \tag{6.11}$$

$$\mathbf{x}_{\mathbb{K}} = \frac{2\mathbf{x}_{\mathbb{D}}}{1+c\|\mathbf{x}_{\mathbb{D}}\|^2}. \tag{6.12}$$

Thus, given points in the Poincaré ball, we can first map them to the Klein model, compute the average using Equation (6.10), and then move it back to the Poincaré model.

Numerical stability. While implementing most of the formulas described above is straightforward, we employ some tricks to make the training more stable. In particular, to ensure numerical stability, we perform clipping by norm after applying the exponential map, which constrains the norm not to exceed $\frac{1}{\sqrt{c}}(1 - 10^{-3})$.

6.5 Experiments

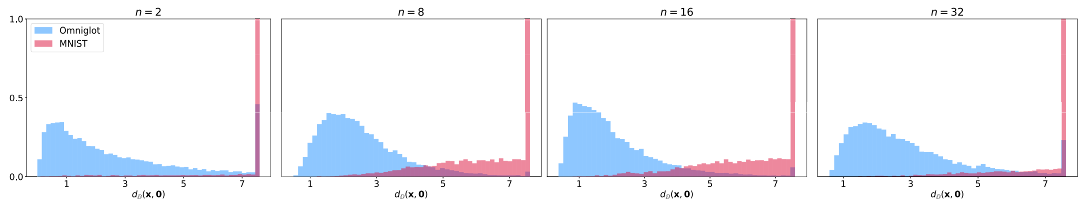


Figure 6-5: Distributions of the hyperbolic distance to the origin of the MNIST (red) and Omniglot (blue) datasets embedded into the Poincaré ball; parameter n denotes embedding dimension of the model trained for MNIST classification. Most Omniglot instances can be easily identified as out-of-domain based on their distance to the origin.

Experimental setup. We start with a toy experiment supporting our hypothesis that the distance to the center in Poincaré ball indicates a model uncertainty. To do so, we first train a classifier in hyperbolic space on the MNIST dataset [LeCun et al., 1998] and evaluate it on the Omniglot dataset [Lake et al., 2013]. We then investigate and compare the obtained distributions of distances to the origin of hyperbolic embeddings of the MNIST and Omniglot test sets.

In our further experiments, we concentrate on the few-shot classification and person re-identification tasks. The experiments on the Omniglot dataset serve as a starting point, and then we move towards more complex datasets. Afterwards, we consider two datasets, namely: *Mini*ImageNet [Ravi and Larochelle, 2016] and Caltech-UCSD Birds-200-2011 (CUB) [Wah et al., 2011b]. Finally, we provide the re-identification results for the two popular datasets: Market-1501 [Zheng et al., 2015] and DukeMTMD [Ristani et al., 2016, Zheng et al., 2017]. Further in this section, we provide a thorough description of each experiment. Our code is available at github[1].

[1]https://github.com/leymir/hyperbolic-image-embeddings

Table 6.3: Kolmogorov-Smirnov distances between the distributions of distance to the origin of the MNIST and Omniglot datasets embedded into the Poincaré ball with the hyperbolic classifier trained on MNIST, and between the distributions of $p_{\max}$ (maximum probablity predicted for a class) for the Euclidean classifier trained on MNIST and evaluated on the same sets.

	$n = 2$	$n = 8$	$n = 16$	$n = 32$
$d_{\mathbb{D}}(\mathbf{x}, \mathbf{0})$	**0.868**	0.832	**0.853**	**0.859**
$p_{\max}(\mathbf{x})$	0.834	**0.835**	0.840	0.846

6.5.1 Distance to the origin as the measure of uncertainty

In this subsection, we validate our hypothesis, which claims that if one trains a hyperbolic classifier, then the distance of the Poincaré ball embedding of an image to the origin can serve as a good measure of confidence of a model. We start by training a simple hyperbolic convolutional neural network on the MNIST dataset (we hypothesized that such a simple dataset contains a very basic hierarchy, roughly corresponding to visual ambiguity of images, as demonstrated by a trained network on Figure 6-1). The output of the last hidden layer was mapped to the Poincaré ball using the exponential map (6.8) and was followed by the hyperbolic multi-linear regression (MLR) layer [Ganea et al., 2018].

After training the model to $\sim 99\%$ test accuracy, we evaluate it on the Omniglot dataset (by resizing its images to 28×28 and normalizing them to have the same background color as MNIST). We then evaluated the hyperbolic distance to the origin of embeddings produced by the network on both datasets. The closest Euclidean analogue to this approach would be comparing distributions of $p_{\max}$, maximum class probability predicted by the network. For the same range of dimensions, we train ordinary Euclidean classifiers on MNIST and compare these distributions for the same sets. Our findings are summarized in Figure 6-5 and Table 6.3. We observe that distances to the origin represent a better indicator of the dataset dissimilarity in three out of four cases.

We have visualized the learned MNIST and Omniglot embeddings in Figure 6-1. We observe that more "unclear" images are located near the center, while the images that are easy to classify are located closer to the boundary.

6.5.2 Few–shot classification

We hypothesize that a certain class of problems — namely the few-shot classification task can benefit from hyperbolic embeddings, due to the ability of hyperbolic space to accurately reflect even very complex hierarchical relations between data points. In principle, any metric learning approach can be modified to incorporate the hyperbolic embeddings. We decided to focus on the classical approach called prototypical networks (ProtoNets) introduced in [Snell et al., 2017]. This approach was picked because it is simple in general and simple to convert to hyperbolic geometry. ProtoNets use the so-called *prototype representation* of a class, which is defined as a mean of the embedded support set of a class. Generalizing this concept to hyperbolic space, we substitute the Euclidean mean operation by HypAve, defined earlier in (6.10). We show that Hyperbolic ProtoNets can achieve results competitive with many recent state-of-the-art models. Our main experiments are conducted on *Mini*ImageNet and Caltech-UCSD Birds-200-2011 (CUB). Additional experiments on the Omniglot dataset, as well as the implementation details and hyperparameters, are provided in Section 6.8. For a visualization of learned embeddings see Section 6.9.

***Mini*ImageNet.** *Mini*ImageNet dataset is the subset of ImageNet dataset [Russakovsky et al., 2015] that contains 100 classes represented by 600 examples per class. We use the following split provided in the paper [Ravi and Larochelle, 2016]: the training dataset consists of 64 classes, the validation dataset is represented by 16 classes, and the remaining 20 classes serve as the test dataset. We test the models on tasks for 1-shot and 5-shot classifications; the number of query points in each batch always equals to 15. Similarly to [Snell et al., 2017], the model is trained in the 30-shot regime for the 1-shot task and the 20-shot regime for the 1-shot task. We test our approach with two different backbone CNN models: a commonly used four-block CNN [Snell et al., 2017, Chen et al., 2019a] (denoted '4 Conv' in the table) and ResNet18 [He et al., 2016]. To find the best values of hyperparameters, we used the grid search; see Section 6.8 for the complete list of values.

Table 6.4 illustrates the obtained results on the *Mini*ImageNet dataset (alongside

Table 6.4: Few-shot classification accuracy results on *Mini*ImageNet on 1-shot 5-way and 5-shot 5-way tasks. All accuracy results are reported with 95% confidence intervals.

Baselines	**Embedding Net**	**1-Shot 5-Way**	**5-Shot 5-Way**
MatchingNet [Vinyals et al., 2016]	4 Conv	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
MAML [Finn et al., 2017]	4 Conv	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
RelationNet [Sung et al., 2018]	4 Conv	$50.44 \pm 0.82\%$	$65.32 \pm 0.70\%$
REPTILE [Nichol and Schulman, 2018]	4 Conv	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
ProtoNet [Snell et al., 2017]	4 Conv	$49.42 \pm 0.78\%$	$68.20 \pm 0.66\%$
Baseline* [Chen et al., 2019a]	4 Conv	$41.08 \pm 0.70\%$	$54.50 \pm 0.66\%$
Spot&learn [Chu et al., 2019]	4 Conv	$51.03 \pm 0.78\%$	$67.96 \pm 0.71\%$
DN4 [Li et al., 2019]	4 Conv	$51.24 \pm 0.74\%$	$71.02 \pm 0.64\%$
Hyperbolic ProtoNet	4 Conv	$\mathbf{54.43 \pm 0.20\%}$	$\mathbf{72.67 \pm 0.15\%}$
SNAIL [Mishra et al., 2017]	ResNet12	$55.71 \pm 0.99\%$	$68.88 \pm 0.92\%$
ProtoNet^{+} [Snell et al., 2017]	ResNet12	$56.50 \pm 0.40\%$	$74.2 \pm 0.20\%$
CAML [Jiang et al., 2019]	ResNet12	$59.23 \pm 0.99\%$	$72.35 \pm 0.71\%$
TPN [Liu et al., 2019]	ResNet12	59.46%	75.65%
MTL [Sun et al., 2019]	ResNet12	$61.20 \pm 1.8\%$	$75.50 \pm 0.8\%$
DN4 [Li et al., 2019]	ResNet12	$54.37 \pm 0.36\%$	$74.44 \pm 0.29\%$
TADAM [Oreshkin et al., 2018]	ResNet12	58.50%	76.70%
Qiao-WRN [Qiao et al., 2018]	Wide-ResNet28	$59.60 \pm 0.41\%$	$73.74 \pm 0.19\%$
LEO [Rusu et al., 2019]	Wide-ResNet28	$\mathbf{61.76 \pm 0.08\%}$	$\mathbf{77.59 \pm 0.12\%}$
Dis. k-shot [Bauer et al., 2017]	ResNet34	$56.30 \pm 0.40\%$	$73.90 \pm 0.30\%$
Self-Jig(SVM) [Chen et al., 2019b]	ResNet50	$58.80 \pm 1.36\%$	$76.71 \pm 0.72\%$
Hyperbolic ProtoNet	ResNet18	$59.47 \pm 0.20\%$	$76.84 \pm 0.14\%$

other results in the literature). Interestingly, **Hyperbolic ProtoNet** significantly improves accuracy as compared to the standard ProtoNet, especially in the one-shot setting. We observe that the obtained accuracy values, in many cases, exceed the results obtained by more advanced methods, sometimes even in the case of architecture of larger capacity. This partly confirms our hypothesis that hyperbolic geometry indeed allows for more accurate embeddings in the few–shot setting.

Caltech-UCSD Birds. The CUB dataset consists of 11, 788 images of 200 bird species and was designed for fine-grained classification. We use the split introduced in [Triantafillou et al., 2017]: 100 classes out of 200 were used for training, 50 for validation and 50 for testing. Due to the relative simplicity of the dataset, we consider only the 4-Conv backbone and do not modify the training shot values as was done for the *Mini*ImageNet case. The full list of hyperparameters is provided in Section 6.8.

Our findings are summarized in Table 6.5. Interestingly, for this dataset, the hyperbolic version of ProtoNet significantly outperforms its Euclidean counterpart (by more than 10% in both settings), and outperforms many other algorithms.

Table 6.5: Few-shot classification accuracy results on CUB dataset [Wah et al., 2011a] on 1-shot 5-way task, 5-shot 5-way task. All accuracy results are reported with 95% confidence intervals. For each task, the best-performing method is highlighted.

Baselines	Embedding Net	**1-Shot 5-Way**	**5-Shot 5-Way**
MatchingNet [Vinyals et al., 2016]	4 Conv	61.16 ± 0.89	72.86 ± 0.70
MAML [Finn et al., 2017]	4 Conv	55.92 ± 0.95%	72.09 ± 0.76%
ProtoNet [Snell et al., 2017]	4 Conv	51.31 ± 0.91%	70.77 ± 0.69%
MACO [Hilliard et al., 2018]	4 Conv	60.76%	74.96%
RelationNet [Sung et al., 2018]	4 Conv	62.45 ± 0.98%	76.11 ± 0.69%
Baseline++ [Chen et al., 2019a]	4 Conv	60.53 ± 0.83%	79.34 ± 0.61%
DN4-DA [Li et al., 2019]	4 Conv	53.15 ± 0.84%	81.90 ± 0.60%
Hyperbolic ProtoNet	4 Conv	**64.02 ± 0.24%**	**82.53 ± 0.14%**

Table 6.6: Person re-identification results for Market-1501 and DukeMTMC-reID for the classification baseline (*Euclidean*) and its hyperbolic counterpart (*Hyperbolic*). (See 6.5.3 for the details). The results are shown for the three embedding dimensionalities and for two different learning rate schedules. For each dataset and each embedding dimensionality value, the best results are bold, they are all given by the hyperbolic version of classification (either by the schedule *sch#1* or *sch#2*). The second-best results are underlined.

	Market-1501				DukeMTMC-reID			
	Euclidean		Hyperbolic		Euclidean		Hyperbolic	
dim, lr schedule	r1	mAP	r1	mAP	r1	mAP	r1	mAP
32, sch#1	71.4	49.7	69.8	45.9	56.1	35.6	56.5	34.9
32, sch#2	68.0	43.4	**75.9**	**51.9**	57.2	35.7	**62.2**	**39.1**
64, sch#1	80.3	60.3	83.1	60.1	69.9	48.5	**70.8**	**48.6**
64, sch#2	80.5	57.8	**84.4**	**62.7**	68.3	45.5	70.7	48.6
128, sch#1	86.0	67.3	**87.8**	**68.4**	74.1	53.3	**76.5**	**55.4**
128, sch#2	86.5	68.5	86.4	66.2	71.5	51.5	74.0	52.2

6.5.3 Person re-identification

The DukeMTMC-reID dataset [Ristani et al., 2016, Zheng et al., 2017] contains 16,522 training images of 702 identities, 2,228 query images of 702 identities and 17,661 gallery images. The Market1501 dataset [Zheng et al., 2015] contains 12,936 training images of 751 identities, 3,368 queries of 750 identities and 15,913 gallery images respectively. We report Rank1 of the Cumulative matching Characteristic Curve and Mean Average Precision for both datasets. The results (Table 6.6) are reported after the 300 training epochs. The experiments were performed with the ResNet50 backbone, and two different learning rate schedulers (see Appendix 6.8 for more details). The hyperbolic version generally performs better than the Euclidean baseline, with the advantage being bigger for smaller dimensionality.

6.6 Discussion and conclusion

We have investigated the use of hyperbolic spaces for image embeddings. The models that we have considered use Euclidean operations in most layers, and use the exponential map to move from the Euclidean to hyperbolic spaces at the end of the network (akin to the normalization layers that are used to map from the Euclidean space to Euclidean spheres). The approach that we investigate here is thus compatible with existing backbone networks trained in Euclidean geometry.

At the same time, we have shown that across a number of tasks, in particular in the few-shot image classification, learning hyperbolic embeddings can result in a substantial boost in accuracy. We speculate that the negative curvature of the hyperbolic spaces allows for embeddings that are better conforming to the intrinsic geometry of at least some image manifolds with their hierarchical structure.

Future work may include several potential modifications of the approach. We have observed that the benefit of hyperbolic embeddings may be substantially bigger in some tasks and datasets than in others. A better understanding of when and why the use of hyperbolic geometry is warranted is therefore needed. Finally, we note that while all hyperbolic geometry models are equivalent in the continuous setting, fixed-precision arithmetic used in real computers breaks this equivalence. In practice, we observed that care should be taken about numeric precision effects. Using other models of hyperbolic geometry may result in a more favourable floating point performance.

6.7 Hyperbolic Neural Networks

Linear layer. Assume we have a standard (Euclidean) linear layer $\mathbf{x} \to \mathrm{M}\mathbf{x} + \mathbf{b}$. In order to generalize it, one needs to define the Möbius matrix by vector product:

$$\mathrm{M}^{\otimes_c}(\mathbf{x}) := \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathrm{M}\mathbf{x}\|}{\|\mathbf{x}\|} \operatorname{arctanh}(\sqrt{c}\|\mathbf{x}\|)\right) \frac{\mathrm{M}\mathbf{x}}{\|\mathrm{M}\mathbf{x}\|}, \tag{6.13}$$

if $\mathrm{M}\mathbf{x} \neq \mathbf{0}$, and $\mathrm{M}^{\otimes_c}(\mathbf{x}) := \mathbf{0}$ otherwise. Finally, for a bias vector $\mathbf{b} \in \mathbb{D}_c^n$ the operation underlying the hyperbolic linear layer is then given by $\mathrm{M}^{\otimes_c}(\mathbf{x}) \oplus_c \mathbf{b}$.

Concatenation of input vectors. In several architectures (e.g., in siamese networks), it is needed to concatenate two vectors; such operation is obvious in Euclidean space. However, straightforward concatenation of two vectors from hyperbolic space does not necessarily remain in hyperbolic space. Thus, we have to use a generalized version of the concatenation operation, which is then defined in the following manner. For $\mathbf{x} \in \mathbb{D}_c^{n_1}$, $\mathbf{y} \in \mathbb{D}_c^{n_2}$ we define the mapping $\mathrm{Concat} : \mathbb{D}_c^{n_1} \times \mathbb{D}_c^{n_2} \to \mathbb{D}_c^{n_3}$ as follows.

$$\mathrm{Concat}(\mathbf{x}, \mathbf{y}) = \mathrm{M}_1^{\otimes_c} \mathbf{x} \oplus_c \mathrm{M}_2^{\otimes_c} \mathbf{y}, \tag{6.14}$$

where M_1 and M_2 are trainable matrices of sizes $n_3 \times n_1$ and $n_3 \times n_2$ correspondingly. The motivation for this definition is simple: usually, the Euclidean concatenation layer is followed by a linear map, which when written explicitly takes the (Euclidean) form of Equation (6.14).

Multiclass logistic regression (MLR). In our experiments, to perform the multiclass classification, we take advantage of the generalization of multiclass logistic regression to hyperbolic spaces. The idea of this generalization is based on the observation that in Euclidean space logits can be represented as the distances to certain *hyperplanes*, where each hyperplane can be specified with a point of origin and a normal vector. The same construction can be used in the Poincaré ball after a suitable analogue for hyperplanes is introduced. Given $\mathbf{p} \in \mathbb{D}_c^n$ and $\mathbf{a} \in T_{\mathbf{p}}\mathbb{D}_c^n \setminus \{\mathbf{0}\}$, such an analogue would be the union of all geodesics passing through $\mathbf{p}$ and orthogonal to $\mathbf{a}$.

The resulting formula for hyperbolic MLR for K classes is written below; here $\mathbf{p}_k \in \mathbb{D}_c^n$ and $\mathbf{a}_k \in T_{\mathbf{p}_k}\mathbb{D}_c^n \setminus \{\mathbf{0}\}$ are learnable parameters.

$$p(y = k|\mathbf{x}) \propto \\ \exp\left(\frac{\lambda_{\mathbf{p}_k}^c \|\mathbf{a}_k\|}{\sqrt{c}} \operatorname{arcsinh}\left(\frac{2\sqrt{c}\langle -\mathbf{p}_k \oplus_c \mathbf{x}, \mathbf{a}_k\rangle}{(1 - c\|-\mathbf{p}_k \oplus_c \mathbf{x}\|^2)\|\mathbf{a}_k\|}\right)\right).$$

For a more thorough discussion of hyperbolic neural networks, we refer the reader to the paper [Ganea et al., 2018].

6.8 Experiment details

Omniglot. As a baseline model, we consider the prototype network (ProtoNet). Each convolutional block consists of 3×3 convolutional layer followed by batch normalization, ReLU nonlinearity and 2×2 max-pooling layer. The number of filters in the last convolutional layer corresponds to the value of the embedding dimension, for which we choose 64. The hyperbolic model differs from the baseline in the following aspects. First, the output of the last convolutional block is embedded into the Poincaré ball of dimension 64 using the exponential map. Results are presented in Table 6.7. We can see that in some scenarios, in particular for one-shot learning, hyperbolic embeddings are more beneficial, while in other cases, results are slightly worse. The relative simplicity of this dataset may explain why we have not observed a significant benefit of hyperbolic embeddings. We further test our approach on more advanced datasets.

Table 6.7: Few-shot classification accuracies on Omniglot. In order to obtain Hyperbolic ProtoNet, we augment the standard ProtoNet with a mapping to the Poincaré ball, use hyperbolic distance as the distance function, and as the averaging operator we use the HypAve operator defined by Equation (6.10).

	ProtoNet	Hyperbolic ProtoNet
1-shot 5-way	98.2	**99.0**
5-shot 5-way	99.4	99.4
1-shot 20-way	95.8	**95.9**
5-shot 20-way	**98.6**	98.15

***mini*ImageNet.** We performed the experiments with two different backbones, namely the previously discussed 4-Conv model and ResNet18. For the former, embedding dim was set to 1024 and for the latter to 512. For the one-shot setting both models were trained for 200 epochs with Adam optimizer, learning rate being $5 \cdot 10^{-3}$ and step learning rate decay with the factor of 0.5 and step size being 80 epochs. For the 4-Conv model we used $c = 0.01$ and for ResNet18 we used $c = 0.001$. For 4-Conv in the five-shot setting we used the same hyperparameters except for $c = 0.005$ and learning rate decay step being 60 epochs. For ResNet18 we additionally changed learning rate to 10^{-3} and step size to 40.

Caltech-UCSD Birds. For these experiments we used the same 4-Conv architecture with the embedding dimensionality being 512. For the one-shot task, we used learning rate 10^{-3}, $c = 0.05$, learning rate step being 50 epochs and decay rate of 0.8. For the five-shot task, we used learning rate 10^{-3}, $c = 0.01$, learning rate step of 40 and decay rate of 0.8.

Person re-identification. We use ResNet50 [He et al., 2016] architecture with one fully connected embedding layer following the global average pooling. Three embedding dimensionalities are used in our experiments: 32, 64 and 128. For the baseline experiments, we add the additional classification linear layer, followed by the cross-entropy loss. For the hyperbolic version of the experiments, we map the descriptors to the Poincaré ball and apply multiclass logistic regression as described in Section 6.4. We found that in both cases the results are very sensitive to the learning rate schedules. We tried four schedules for learning 32-dimensional descriptors for both baseline and hyperbolic versions. The two best performing schedules were applied for the 64 and 128-dimensional descriptors. In these experiments, we also found that smaller c values give better results. We therefore have set c to 10^{-5}. Based on the discussion in 6.4, our hyperbolic setting is quite close to Euclidean. The results are compiled in Table 6.6. We set starting learning rates to $3 \cdot 10^{-4}$ and $6 \cdot 10^{-4}$ for $sch\#1$ and $sch\#2$ correspondingly and multiply them by 0.1 after each of the epochs 200 and 270.

6.9 Visualizations

For the visual inspection of embeddings we computed projections of high dimensional embeddings obtained from the trained few–shot models with the (hyperbolic) UMAP algorithm [McInnes et al., 2018] (see Figure 6-6). We observe that different classes are neatly positioned near the boundary of the circle and are well separated.

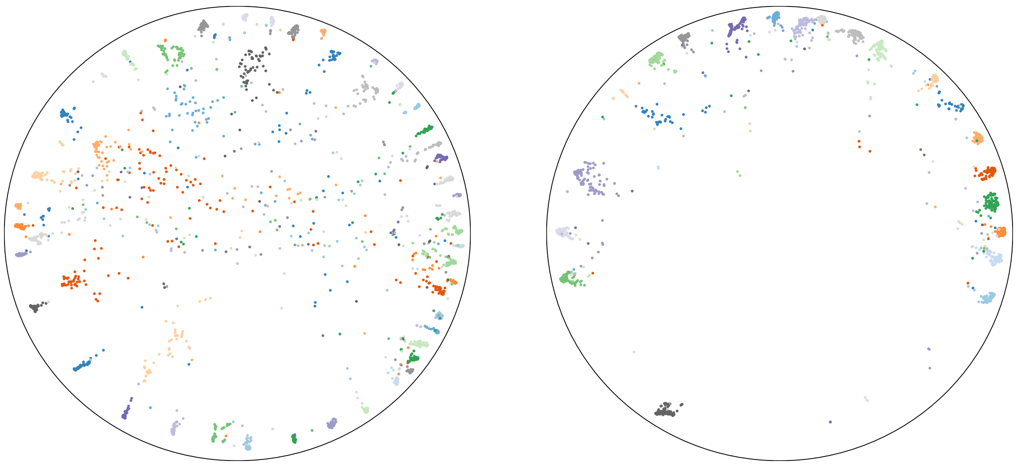


Figure 6-6: A visualization of the hyperbolic embeddings learned for the few–shot task. **Left**: 5-shot task on CUB. **Right**: 5-shot task on *Mini*ImageNet. The two-dimensional projection was computed with the UMAP algorithm [McInnes et al., 2018].

"If you optimize everything, you will always be unhappy."

Donald Knuth

Chapter 7

Conclusion

In this thesis we explored how geometrical ideas can be applied to further out understanding of deep learning and develop new practical algorithms. Ultimately, our goal is to produce learning algorithms that utilize the geometrical structure of underlying data manifolds in an automatic, unsupervised manner. We anticipate one of the main directions towards this goal to be a utilization of powerful generative models, which infer a solid model of the data manifold. Another area where similar ideas could be explored is reinforcement learning. In Srinivas et al. [2020], it was shown that even simple image-based augmentations allow one to achieve state-of-the-art sample efficiency on a large number of tasks where the inputs are represented as images. However, in many RL tasks, the state space has some non-trivial manifold structure. An additional complication is that *actions* have to be transformed as well as states — as an example, we can think about the state reflection in the Snake game. We believe that an automatic discovery of state-action symmetries is a crucial step towards truly intelligent RL agents.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

P-A Absil and Jérôme Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Ejaz Ahmed, Michael J. Jones, and Tim K. Marks. An improved deep learning architecture for person re-identification. In *Conf. Computer Vision and Pattern Recognition, CVPR*, pages 3908–3916, 2015.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Eric Bailey and Shuchin Aeron. Word embeddings via tensor factorization. *arXiv preprint arXiv:1704.02686*, 2017.

Shane Barratt and Rishi Sharma. A note on the Inception Score. *arXiv preprint arXiv:1801.01973*, 2018.

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartlomiej Swiatkowski, Bernhard Scholkopf, and Richard E Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017.

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018.

Michele Benzi, Gene H Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14:1–137, 2005.

Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Jean-Daniel Boissonnat, Leonidas J Guibas, and Steve Y Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete & Computational Geometry*, 42(1):37–70, 2009.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Weronika Buczyńska, Jarosław Buczyński, and Mateusz Michałek. The Hackbusch conjecture on tensor formats. *Journal de Mathématiques Pures et Appliquées*, 104 (4):749–761, 2015.

Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5): 2053–2080, 2010.

J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

Xiaobin Chang, Timothy M Hospedales, and Tao Xiang. Multi-level factorisation net for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2109–2118, 2018.

Viktoria Chekalina, Elena Orlova, Fedor Ratnikov, Dmitry Ulyanov, Andrey Ustyuzhanin, and Egor Zakharov. Generative models for fast calorimeter simulation: the lhcb case. In *EPJ Web of Conferences*, volume 214, page 02034. EDP Sciences, 2019.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019a. URL https://openreview.net/forum?id=HkxLXnAcFQ.

Zitian Chen, Yanwei Fu, Kaiyu Chen, and Yu-Gang Jiang. Image block augmentation for one-shot learning. In *AAAI*, 2019b.

Wen-Hsuan Chu, Yu-Jhe Li, Jing-Cheng Chang, and Yu-Chiang Frank Wang. Spot and learn: A maximum-entropy patch sampler for few-shot image classification. In *CVPR*, pages 6251–6260, 2019.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan Oseledets, Masashi Sugiyama, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

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

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

Nadav Cohen, Ronen Tamari, and Amnon Shashua. Boosting dilated convolutional networks with mixed tensor decompositions. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1JHhv6TW.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

Vin De Silva and Gunnar E Carlsson. Topological estimation using witness complexes. *SPBG*, 4:157–166, 2004.

DW Decker and CT Kelley. Newton's method at singular points. ii. *SIAM Journal on Numerical Analysis*, 17(3):465–471, 1980.

Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313*, 2018.

Aritra Dutta and Xin Li. On a problem of weighted low-rank approximation of matrices. *SIAM Journal on Matrix Analysis and Applications*, 38(2):530–553, 2017.

Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454–463. IEEE, 2000.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135. JMLR, 2017.

Alexander Fonarev, Oleksii Hrinchuk, Gleb Gusev, Pavel Serdyukov, and Ivan Oseledets. Riemannian optimization for skip-gram negative sampling. *arXiv preprint arXiv:1704.08059*, 2017.

Hervé Fournier, Anas Ismail, and Antoine Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115(6-8): 576–579, 2015.

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

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, pages 5350–5360, 2018.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. 1999.

Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014a.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014b.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

Lars Grasedyck and Wolfgang Hackbusch. An introduction to hierarchical (H-) rank and TT-rank of tensors with examples. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 11(3):291–304, 2011.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

Daniel R Grayson and Michael E Stillman. Macaulay2, a software system for research in algebraic geometry, 2002.

Phillip Griffiths and Joseph Harris. *Principles of algebraic geometry*. John Wiley & Sons, 2014.

Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxHsjRqFQ.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

Yiluan Guo and Ngai-Man Cheung. Efficient and deep person re-identification using multi-level similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2335–2344, 2018.

Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer Science & Business Media, 2012.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

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

Robin Hartshorne. *Algebraic geometry*, volume 52. Springer Science & Business Media, 2013.

Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

John Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986.

Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Seyedehsomayeh Hosseini and André Uschmajew. A riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. *SIAM Journal on Optimization*, 27(1):173–189, 2017.

Yu S Ilyashenko and Sergei Yakovenko. *Lectures on analytic differential equations*, volume 86. American Mathematical Soc., 2008.

Xiang Jiang, Mohammad Havaei, Farshid Varno, Gabriel Chartrand, Nicolas Chapados, and Stan Matwin. Learning to learn with conditional class dependencies. In *ICLR*, 2019. URL https://openreview.net/forum?id=BJfOXnActQ.

Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157. Springer Science & Business Media, 2006.

Mahdi M Kalayeh, Emrah Basaran, Muhittin Gökmen, Mustafa E Kamasak, and Mubarak Shah. Human semantic parsing for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1062–1071, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

Valentin Khrulkov, Alexander Novikov, and Ivan Oseledets. Expressive power of recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1WRibb0Z.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Preconditioned low-rank riemannian optimization for linear systems with tensor product structure. *SIAM Journal on Scientific Computing*, 38(4):A2018–A2044, 2016.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*, pages 2526–2534, 2013.

V Lakshmibai and Justin Brown. *The Grassmannian variety*, volume 42. Springer, 2015.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun et al. Generalization and network design strategies. In *Connectionism in perspective*, volume 19. Citeseer, 1989.

John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer, 2013.

Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, pages 7260–7268, 2019.

Nathan Linial, Avner Magen, and Michael E Saks. Low distortion Euclidean embeddings of trees. *Israel Journal of Mathematics*, 106(1):339–348, 1998.

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning To Propagate Labels: Transductive Propagation Network For Few-Shot Learning. In *ICLR*, 2019. URL https://openreview.net/forum?id=SyVuRiC5K7.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical approximation by hierarchical tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher 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.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The Gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

Ivan Markovsky and KONSTANTIN Usevich. *Low rank approximation*. Springer, 2012.

Ivan Markovsky and Konstantin Usevich. Structured low-rank approximation with missing data. *SIAM Journal on Matrix Analysis and Applications*, 34(2):814–830, 2013.

James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

J Peter May. *A concise course in algebraic topology*. University of Chicago press, 1999.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Simone Naldi. *Exact algorithms for determinantal varieties and semidefinite programming*. PhD thesis, Toulouse, INSA, 2015.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of Hyperbolic geometry. In *Proc. ICML*, pages 3776–3785, 2018a.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347, 2017.

Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3779–3788, 2018b.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450, 2015.

Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, pages 719–729, 2018.

Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

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

Michela Paganini, Luke de Oliveira, and Benjamin Nachman. CaloGAN: Simulating 3D high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks. *arXiv preprint arXiv:1705.02355*, 2017.

Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Calogan: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Physical Review D*, 97(1):014021, 2018.

O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

Chunhong Qi. Numerical optimization methods on riemannian manifolds. 2011.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, pages 7229–7238, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Maxim Rakhuba and Ivan Oseledets. Calculating vibrational spectra of molecules using tensor train decomposition. *The Journal of chemical physics*, 145(12):124101, 2016.

Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical review E*, 67(2):026112, 2003.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. URL `https://openreview.net/forum?id=BJgklhAcK7`.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International Conference on Machine Learning*, pages 4457–4466, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016a.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016b.

Rik Sarkar. Low distortion Delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

Igor Rostislavovich Shafarevich and Kurt Augustus Hirsch. *Basic algebraic geometry*, volume 2. Springer, 1994.

Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The Riemannian geometry of deep generative models. *arXiv preprint arXiv:1711.08014*, 2017.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

Steven T Smith. Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4799–4807. Curran Associates, Inc., 2016.

Chi Su, Jianing Li, Shiliang Zhang, Junliang Xing, Wen Gao, and Qi Tian. Pose-driven deep convolutional model for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3960–3969, 2017.

Yumin Suh, Jingdong Wang, Siyu Tang, Tao Mei, and Kyoung Mu Lee. Part-aligned bilinear representations for person re-identification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 402–419, 2018.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, pages 403–412, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015a. URL `http://arxiv.org/abs/1409.4842`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015b.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré GloVe: Hyperbolic word embeddings. *arXiv preprint arXiv:1810.06546*, 2018.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016.

Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems. *James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang, editors, WSDM*, 20:3–7, 2018.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2255–2265, 2017.

Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Alex Bronstein, Ivan Oseledets, and Emmanuel Müller. The shape of data: Intrinsic distance for data distributions. In *Iclr 2020: Proceedings of the International Conference on Learning Representations*, 2020.

André Uschmajew and Bart Vandereycken. Geometric methods on low-rank matrix and tensor manifolds. In *Handbook of Variational Methods for Nonlinear Geometric Data*, pages 261–313. Springer, 2020.

Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016.

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

M Alex O Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pages 447–460. Springer, 2002.

M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE, 2003.

M.Alex O. Vasilescu, Eric Kim, and Xiao S. Zeng. Causal explanations and block multilinear factor analysis. In *The 25th International Conference on Pattern Recognition (ICPR 2020)*, 2020.

M.A.O. Vasilescu and E. Kim. Compositional hierarchical tensor factorization: Representing hierarchical intrinsic and extrinsic causal factors. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19) and Workshops, Tensor Methods for Emerging Data Science Challenges*, New York, NY, USA, August 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638, 2016.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011a.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. 2011b.

Yicheng Wang, Zhenzhong Chen, Feng Wu, and Gang Wang. Person re-identification with cascaded pairwise convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1470–1478, 2018.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864, 2016.

Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. *arXiv preprint arXiv:1707.01786*, 2017.

Dong Yi, Zhen Lei, and Stan Z Li. Deep metric learning for practical person re-identification. *arXiv prepzrint arXiv:1407.4979*, 2014.

Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term forecasting using tensor-train RNNs. *arXiv preprint arXiv:1711.00073*, 2017.

Ya-xiang Yuan. A review of trust region algorithms for optimization. In *Iciam*, volume 99, pages 271–282, 2000.

Haiyu Zhao, Maoqing Tian, Shuyang Sun, Jing Shao, Junjie Yan, Shuai Yi, Xiaogang Wang, and Xiaoou Tang. Spindle net: Person re-identification with human body region guided feature decomposition and fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1077–1085, 2017.

Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015.

Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Glossary

AG Algebraic Geometry. 11, 12

CNNs Convolutional Neural Networks. 10, 11

CP CANDECOMP/PARAFAC. 10, 11

CV Computer Vision. 9

GANs Generative Adversarial Networks. 12, 17

HT Hierarchical Tucker. 10, 11, 15

NLP Natural Language Processing. 9

ReLU Rectified Linear Unit. 11, 15

RNNs Recurrent Neural Networks. 11, 14

TDA Topological Data Analysis. 13, 17

TR Tensor Ring. 11

TT Tensor Train. 10, 14, 15

VAEs Variational Autoencoders. 13, 14