Weierstraß-Institut
für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Approximation and learning with compositional tensor trains

Martin Eigel[1], Charles Miranda[2], Anthony Nouy[2], David Sommer[1]

submitted: December 22, 2025

[1] Weierstrass Institute
Anton-Wilhelm-Amo-Str. 39
10117 Berlin
Germany
E-Mail: martin.eigel@wias-berlin.de

[2] Centrale Nantes, Nantes Université
Laboratoire de Mathématiques
Jean Leray UMR CNRS 6629
France
E-Mail: charles.miranda@ec-nantes.fr
anthony.nouy@ec-nantes.fr

No. 3253
Berlin 2025

2020 *Mathematics Subject Classification.* 41A63, 65D15, 68T07, 65K10, 15A69, 41A46, 49J15, 65F55, 90C26.

Key words and phrases. Compositional tensor networks, tensor train, approximation, natural gradient.

We acknowledge the partial funding by the ANR-DFG project COFNET (ANR-21-CE46-0015). This work was partially conducted within the France 2030 framework programme, Centre Henri Lebesgue ANR-11-LABX-0020-01.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Anton-Wilhelm-Amo-Straße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Approximation and learning with compositional tensor trains

Martin Eigel, Charles Miranda, Anthony Nouy, David Sommer

Abstract

We introduce compositional tensor trains (CTTs) for the approximation of multivariate functions, a class of models obtained by composing low-rank functions in the tensor-train format. This format can encode standard approximation tools, such as (sparse) polynomials, deep neural networks (DNNs) with fixed width, or tensor networks with arbitrary permutation of the inputs, or more general affine coordinate transformations, with similar complexities. This format can be viewed as a DNN with width exponential in the input dimension and structured weights matrices. Compared to DNNs, this format enables controlled compression at the layer level using efficient tensor algebra.

On the optimization side, we derive a layerwise algorithm inspired by natural gradient descent, allowing to exploit efficient low-rank tensor algebra. This relies on low-rank estimations of Gram matrices, and tensor structured random sketching. Viewing the format as a discrete dynamical system, we also derive an optimization algorithm inspired by numerical methods in optimal control. Numerical experiments on regression tasks demonstrate the expressivity of the new format and the relevance of the proposed optimization algorithms.

Overall, CTTs combine the expressivity of compositional models with the algorithmic efficiency of tensor algebra, offering a scalable alternative to standard deep neural networks.

1 Introduction

This work studies compositional tensor trains (CTTs), which are compositions of functions in tensor-train (TT) format, defining a new class of expressive and compressible models for high-dimensional approximation. We quantify their expressivity by showing that usual approximation tools can be efficiently encoded with CTTs, and propose optimization methods relying on optimal control algorithms (viewing CTTs as a time-discrete dynamical system) or natural gradient schemes, leveraging efficient tensor algebra techniques.

Low-rank tensor formats. Tensor networks represent a powerful tool in the realm of high-dimensional data analysis and functional approximation, having gained significant attention in recent years due to their ability to represent efficiently complex structures. Among such formats, TTs [44] were initially developed to address the curse of dimensionality in quantum physics, where they are also known as Matrix Product States, but have since been applied in a number of other fields, including machine learning and computational science.

Neural networks as compositional formats. The advent of neural networks (NNs) has transformed our ability to tackle complex problems and became the foundation of modern machine learning. These models are particularly adept at learning intricate patterns and relationships within data. However,

DOI 10.20347/WIAS.PREPRINT.3253 Berlin 2025

as the complexity of tasks increases, so does the necessity for bigger and more sophisticated architectures, which in turn give rise to challenging, time-consuming, and resource-intensive training processes. In fact, typically slow optimization is a central obstacle for achieving SOTA models in current AI topics such as generative modeling and LLMs.

Composition of low-rank formats. In this evolving landscape of computational techniques, compositional formats have emerged as a promising approach to approximate functions with high accuracy, alleviating the curse of dimensionality under certain conditions. NNs are, of course, the most famous example of such a compositional format. The literature on compositional TTs is still practically non-existent, but there are a few notable works. The first is the Deep Inverse Rosenblatt Transport (DIRT) [9], which constructs a transport map as a composition of TTs. The second is the tensor rank reduction via coordinate flows [12], which builds a nonlinear coordinate transformation so that the function in the new coordinate system has smaller tensor rank, leading to a ridge-like TT. A similar work can be found in [18].

Optimal control. A compositional format can be understood as a discretization of a continuous flow. An example of this is the connection between neural ODEs and residual neural networks. This continuous viewpoint offers a natural training procedure by considering the flow and the training loss as state equation and cost functional of an optimal control problem. In a classic optimize-then-discretize approach [43], we can then use classic optimal control techniques like Pontryagin's maximum principle (PMP) to approximate discrete-time optimal controls, which is equivalent to training compositional TTs in our sense.

Contributions of this paper. This paper investigates the composition of TTs, introduced in [50] as a new approximation format, delving into its fundamental principles and examining how they intersect with other approximation tools such as (sparse) polynomials, ridge approximation, deep neural networks, etc. We show how these approximation tools can be encoded into the CTT format, with similar complexities, thereby demonstrating that their approximation spaces are continuously embedded in those of CTTs. This new format comes with efficient and controlled SVD-based compression methods. In addition, this work introduces training procedures of CTTs for general machine learning tasks, based on algorithms inspired from optimal control or natural gradient schemes. These training procedures rely on efficient tensor algorithms that exploit low-rank structures of the layers.

Structure of the paper. The structure of this work is as follows. Section 2 covers the low-rank formats needed to define our compositional low-rank format, namely the canonical (CP) format and the tensor train (TT) format. Section 3 introduces the CTT format. Aside from rigorous definition and some theoretical properties of the format, we give examples of functions which are extremely difficult to approximate in the TT format due to high ranks, but can be very efficiently expressed in CTT format. Section 4 derives two learning algorithms for the CTT format. The first algorithm, coming from the optimal control literature, is based on Pontryagin's maximum principle (PMP) and the method of successive approximation, whose practical implementation benefits from efficient tensor algebra. Under certain assumptions on the bases used for defining the CTT format, we can show convergence of the scheme to a unique fixed point satisfying the PMP. The second algorithm is a modified (layer-wise) natural gradient scheme, allowing one to exploit efficient tensor methods and to preserve the low-rank format of the layers. Finally, Section 5 experimentally shows the power of CTTs and the learning methods on test problems.

2 Tensor Formats

We briefly review the TT factorization, its computational costs, and the TT-SVD construction. We also recall how linear maps are represented as TT-matrices and how ranks relate to matricization ranks.

Consider the task of approximating a function from $\mathbb{R}^d$ to $\mathbb{R}$. Given some finite-dimensional spaces $\mathcal{V}_{n_k}^{(k)}$ of univariate functions, $1 \leq k \leq d$, we define the finite-dimensional (functional) tensor space $\mathcal{V}_{\boldsymbol{n}} = \mathcal{V}_{n_1}^{(1)} \otimes \ldots \otimes \mathcal{V}_{n_d}^{(d)}$, which is the linear span of elementary tensor products $v^{(1)} \otimes \ldots \otimes v^{(d)} : (x_1, \ldots, x_d) \mapsto v^{(1)}(x_1) \ldots v^{(d)}(x_d)$, with $v^{(k)} \in \mathcal{V}_{n_k}^{(k)}$. Given bases $\{\phi_i^{(k)}\}_{i=1}^{n_k}$ of the spaces $\mathcal{V}^{(k)}$, a basis of $\mathcal{V}_{\boldsymbol{n}}$ is given by the collection of all tensor products $\{\phi_{i_1}^{(1)} \otimes \ldots \otimes \phi_{i_d}^{(d)} : 1 \leq i_1 \leq n_1, \ldots, 1 \leq i_d \leq n_d\}$.

For any function v in $\mathcal{V}_{\boldsymbol{n}}$, there is a unique algebraic tensor $\boldsymbol{A} \in \mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$, identified with a multidimensional array in $\mathbb{R}^{n_1 \times \ldots \times n_d}$, such that

$$v = \sum_{i_1, \ldots, i_d = 1}^{n_1, \ldots, n_d} \boldsymbol{A}(i_1, \ldots, i_d) \phi_{i_1}^{(1)} \otimes \ldots \otimes \phi_{i_d}^{(d)}, \tag{1}$$

or equivalently

$$v(x) = \langle \boldsymbol{A}, \Phi^{(1)}(x_1) \otimes \cdots \otimes \Phi^{(d)}(x_d) \rangle_F := \langle \boldsymbol{A}, \boldsymbol{\Phi}(x) \rangle_F, \tag{2}$$

where $\Phi^{(k)}(x_k) := (\phi_j^{(k)}(x_k))_{j=1}^{n_k}$ and $\langle ., . \rangle_F$ denotes the canonical (Frobenius) inner product. In the above notation, $\boldsymbol{A}$ is called the *coefficient tensor* of function v with respect to the tensor product basis.

The storage complexity of such a tensor is $\mathcal{O}(n^d)$, where $n = \max\{n_1, \ldots, n_d\}$. This exponential scaling, often called the *curse of dimensionality*, makes the treatment of high-dimensional problems intractable for $d \gg 1$. Low-rank *tensor formats* are specifically developed to tackle this *curse of dimensionality* and represent tensors in a storage-efficient way. We mention only two such formats, which are relevant for this work, namely the *canonical format* and the *tensor train* format.

Like any rank-1 matrix $A \in \mathbb{R}^{n_1 \times n_2}$ can be defined as the outer product $v \otimes w$ of two vectors $v \in \mathbb{R}^{n_1}, w \in \mathbb{R}^{n_2}$, a tensor $\boldsymbol{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ is said to have rank 1 if there exist $v^{(1)} \in \mathbb{R}^{n_1}, \ldots, v^{(d)} \in \mathbb{R}^{n_d}$ such that

$$\boldsymbol{A} = v^{(1)} \otimes \ldots \otimes v^{(d)}. \tag{3}$$

It is important to note that, while there is only one commonly used notion of matrix rank, there are several different concepts of ranks of tensors. These include the *canonical polyadic (CP) rank*, the *(hierarchical) Tucker rank* and the *tensor train (TT) rank*, to just name a few. These rank notions are each connected to different decompositions of the underlying tensor, i.e. the *canonical polyadic decomposition*, the *(hierarchical) Tucker decomposition* or the *tensor train* decomposition, much like the matrix rank is connected to a singular value decomposition. The reason we can describe a tensor of the form (3) as having general rank 1 is because the mentioned rank notions (and to the best of our knowledge other important rank concepts) agree on which tensors have rank 1 and which do not.

The canonical format. For any tensor $\boldsymbol{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$, there exists a $r_{\mathrm{cp}} \in \mathbb{N}$ and a representation

$$\boldsymbol{A} = \sum_{k=1}^{r_{\mathrm{cp}}} v_k^{(1)} \otimes \cdots \otimes v_k^{(d)} \tag{4}$$

with vectors $\{v_k^{(i)}\}_{k=1}^{r_{\mathrm{cp}}} \subset \mathbb{R}^{n_i}$, for all $i \in \{1, \ldots, d\}$. Such a representation of $\boldsymbol{A}$ as a sum of linearly independent rank-1 tensors is called a CP decomposition. The minimal number r_{cp} is the CP rank or

simply *canonical rank* of the tensor $\boldsymbol{A}$ and denoted by $\operatorname{rank}(\boldsymbol{A})$. The set of tensors with canonical rank smaller or equal to $r \in \mathbb{N}$ is not closed in general [23, Example 7.1], which makes the canonical format problematic for optimization, in particular it does not guarantee the existence of a best rank-r approximation. The tensor train format has emerged, among other formats, as a low rank format suitable for optimization. In particular, the set of tensor trains with bounded *tensor train rank* (defined in the following paragraph) is closed.

The tensor train format. Since the (functional) TT format is used primarily in this work, we introduce it in some detail. We start with the appropriate notion of rank. The *tensor train rank* of a tensor $\boldsymbol{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is the elementwise smallest tuple $\boldsymbol{r}_{\mathrm{TT}} = (r_1, \dots, r_{d-1}) \in \mathbb{N}^{d-1}$ such that there exist matrix valued mappings $U_j : \mathbb{N}_{\leq n_j} \to \mathbb{R}^{r_{j-1} \times r_j}$, $j = 1, \dots, d$, with $r_0 = r_d = 1$ by convention, such that

$$\boldsymbol{A}(i_1, \dots, i_d) = U_1(i_1) \dots U_d(i_d) \quad \text{for all} \quad (i_1, \dots, i_d) \in \mathbb{N}_{\leq n_1} \times \cdots \times \mathbb{N}_{\leq n_d}, \tag{5}$$

with $\mathbb{N}_{\leq n} := \{1, \dots, n\}$. The decomposition (5) is called *tensor train decomposition* and the functions U_j are called the *components* of the tensor train. The alternative name of *matrix product states* comes from the fact that each entry of $\boldsymbol{A}$ is represented by a product of matrices. Such a representation exists for any tensor, like a singular value decomposition exists for any matrix. In fact, the standard algorithm by which a TT decomposition of a tensor is obtained is called TT-SVD [44]. Note that the components of the tensor train can equivalently be defined as order-3 tensors $\boldsymbol{U}_j \in \mathbb{R}^{r_{j-1} \times n_j \times r_j}$, and therefore, any tensor $\boldsymbol{A}$ can be written in the form

$$\boldsymbol{A} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \boldsymbol{U}_1(k_0, \cdot, k_1) \otimes \cdots \otimes \boldsymbol{U}_d(k_{d-1}, \cdot, k_d).$$

In the following, we consider the components of a tensor train to be order-3 tensors. To ease notation, we collect the dimensions of the tensor into a tuple $\boldsymbol{n} = (n_1, \dots, n_d)$. The set of tensors of dimension $\boldsymbol{n}$ with *tensor train rank* $\boldsymbol{r}$ is denoted by

$$\mathcal{T}_{\boldsymbol{n},\boldsymbol{r}} := \{\boldsymbol{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d} : \operatorname{rank}_{TT}(\boldsymbol{A}) = \boldsymbol{r}\}.$$

For later use, we also define the set of tensors with ranks element-wise smaller or equal to $\boldsymbol{r}$ by $\mathcal{T}_{\boldsymbol{n},\leq \boldsymbol{r}} = \bigcup_{0 \leq \hat{\boldsymbol{r}} \leq \boldsymbol{r}} \mathcal{T}_{\boldsymbol{n},\hat{\boldsymbol{r}}}$, which is a closed set. Clearly, storing the tensor is equivalent to storing the d order-3 tensors $\boldsymbol{U}^{(j)}$. Hence, the storage complexity of a tensor in such a format is in $\mathcal{O}(dnr^2)$, where $n = \max\{n_1, \dots, n_d\}$, $r = \max\{r_1, \dots, r_{d-1}\}$. In the most general case, the rank bound r depends exponentially on the dimension d, reflecting the curse of dimensionality for general tensors. However, several notable functions, such as Gaussian potentials, exhibit low rank structures in the TT format [45, 20, 51], yielding (at most) a polynomial growth of r with d. In such cases, the TT format effectively mitigates the curse of dimensionality. The TT format is often represented with a tensor network diagram as illustrated in Figure 1.

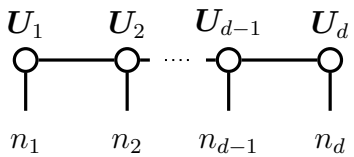


Figure 1: Tensor-train format with cores $\boldsymbol{U}_1, \dots, \boldsymbol{U}_d$

Interestingly, the space of tensors $\mathcal{T}_{\boldsymbol{n},\boldsymbol{r}}$ with fixed TT ranks is a Riemannian submanifold. The dimension of this manifold is $\dim \mathcal{T}_{\boldsymbol{n},\boldsymbol{r}} = \sum_{i=1}^{d} r_{i-1} n_i r_i - \sum_{i=1}^{d-1} r_i^2$ [24, Lemma 4.1]. Note that the first sum in this expression is just the added up storage complexity of all d components. The second term results from a *gauge invariance* inherent in the format that can be seen as follows. For any two subsequent components U_j, U_{j+1} we can transform $U_j(\cdot) \to U_j(\cdot)R^{-1}$ and $U_{j+1}(\cdot) \to RU_{j+1}(\cdot)$, where $R \in \mathbb{R}^{r_j \times r_j}$ is an invertible matrix, without changing the tensor. Hence, the components U_j are only unique up to such invertible transformations, which is taken into account by the second sum making up $\dim(\mathcal{T}_{\boldsymbol{n},\boldsymbol{r}})$.

Any tensor in CP format can easily be represented in TT format. Indeed, let $\boldsymbol{A} = \sum_{k=1}^{r_{\mathrm{cp}}} v_k^{(1)} \otimes \cdots \otimes v_k^{(d)}$, $v_k^{(i)} \in \mathbb{R}^{n_i}$. Then

$$\boldsymbol{A}(i_1, \ldots, i_d) = \begin{bmatrix} v_1^{(1)}(i_1) & \ldots & v_{r_{\mathrm{cp}}}^{(1)}(i_1) \end{bmatrix} \begin{bmatrix} v_1^{(2)}(i_2) & & 0 \\ & \ddots & \\ 0 & & v_{r_{\mathrm{cp}}}^{(2)}(i_2) \end{bmatrix} \cdots \begin{bmatrix} v_1^{(d)}(i_d) \\ \vdots \\ v_{r_{\mathrm{cp}}}^{(d)}(i_d) \end{bmatrix}$$

is a representation in TT format, which can easily be verified by multiplying out the matrix product. Note that the CP rank r_{cp} is a uniform upper bound for the TT ranks. Rounding to a prescribed rank in the TT format is done via the procedure described in [44, Algorithm 2], i.e. by successively truncating the singular value decompositions of component unfoldings to the desired rank (or accuracy). For later use, we let $\mathtt{round}_{\boldsymbol{r}}$ denote the rounding procedure of a tensor in $\mathbb{R}^{n_1 \times \ldots \times n_d}$ to fixed rank $\boldsymbol{r}$.

Functional tensor trains. The set of functions in $\mathcal{V}_{\boldsymbol{n}}$ with coefficient tensors in $\mathcal{T}_{\boldsymbol{n},\boldsymbol{r}}$ is denoted $\mathcal{T}_{\boldsymbol{r}}(\mathcal{V}_{\boldsymbol{n}})$ and called the *functional tensor train format*. A function $v \in \mathcal{T}_{\boldsymbol{r}}(\mathcal{V}_{\boldsymbol{n}})$ admits the representation

$$v(x_1, \ldots, x_d) = \hat{U}_1(x_1)\hat{U}_2(x_2) \cdots \hat{U}_d(x_d) \tag{6}$$

with matrix valued functions $\hat{U}_i(x_i) \in \mathbb{R}^{r_{i-1} \times r_i}$ for $i = 1, \ldots, d$, again with the convention $r_0 = r_d = 1$. The function $\hat{U}_i$ is identified with a third-order tensor $\boldsymbol{U}_i \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ through the relation $\hat{U}_i(k_{i-1}, x_i, k_i) = \sum_{j=1}^{n_i} \boldsymbol{U}_i(k_{i-1}, j, k_i)\phi_j^{(i)}(x_i)$.

The smallest $\boldsymbol{r}$ such that v is in $\mathcal{T}_{\boldsymbol{r}}(\mathcal{V}_{\boldsymbol{n}})$ is the tensor train rank of v, which coincides with the tensor train rank of its coefficient tensor, independently of the choice of bases of functions spaces $\mathcal{V}_{n_i}^{(i)}$. Results on approximation with TT format of functions of classical smoothness classes or with compositional structures can be found in [49, 21, 5, 6, 1, 2].

Vector valued tensor trains. In this work, we are frequently concerned with vector valued functions $v\colon \mathbb{R}^d \to \mathbb{R}^d$. For this case, we permit $r_0 = d$, leading to a first component $\boldsymbol{U}_1 \in \mathbb{R}^{d \times n_1 \times r_1}$ such that the contraction of all components yields a vector in $\mathbb{R}^d$. This first component tensor is identified with the component function $\hat{U}_1 \in \mathbb{R}^d \otimes \mathcal{V}_{n_1}^{(1)} \otimes \mathbb{R}^{r_1}$. A particular case of interest are functions of the form $v\colon \mathbb{R}^d \to \mathbb{R}^d, v(x) = (v_1(x), 0, \ldots, 0)^\top$ which map to zero except in the first output dimension. In this case, a functional TT representation is obtained by taking a (scalar valued) TT representation of v_1 and padding the first component of this representation with zeros for the other outputs. The TT representation of v inherits the ranks of v_1 in this case (except for r_0), since all other components remain the same.

Lemma 2.1. *Assume $v\colon \mathbb{R}^d \to \mathbb{R}^d$, $d \in \mathbb{N}$, has the form $v(x) = (v_1(x), 0, \ldots, 0)^\top$ for a $v_1\colon \mathbb{R}^d \to \mathbb{R}$ with TT rank $\boldsymbol{r} = (r_1, \ldots, r_{d-1})$ and $r_0 = r_d = 1$. Then, v can be represented by a TT with rank $\boldsymbol{r}$ and $r_0 = d$, $r_d = 1$.*

Proof. Assume the TT representation of v_1 is $v_1(x) = \hat{U}^{(1)}(x_1)\dots\hat{U}^{(d)}(x_d)$ with $\hat{U}^{(i)}(x_i) \in \mathbb{R}^{r_{i-1}\times r_i}$, $r_0 = r_d = 1$. Define $\tilde{U}^{(i)} = \hat{U}^{(i)}$ for $i = 2,\dots,d$ and

$$\tilde{U}^{(1)}(x_1) = \begin{bmatrix} \hat{U}^{(1)}(x_1) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d\times r_1}.$$

Then, $v(x) = \tilde{U}^{(1)}(x_1)\dots\tilde{U}^{(d)}(x_d)$ is a TT representation of v, which preserves all ranks, except r_0. □

This result enables us to represent vector-valued functions, where only the first output dimension is nonzero, as TTs. For later use, we state another immediate result regarding the rank of such functions with additional structure.

Lemma 2.2. *Assume $v\colon \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$, $d \in \mathbb{N}$, has the form $v(x) = (v_{2:d+1}(x_{2:d+1})v_1(x_1), 0_d)^\top$, where $x_{2:d+1} = (x_2,\dots,x_{d+1})$, 0_d is the zero vector in $\mathbb{R}^d$, $v_1\colon \mathbb{R}\to\mathbb{R}$, and $v_{2:d+1}\colon \mathbb{R}^d \to \mathbb{R}$ has TT rank $\boldsymbol{r}^* = (r_1^*,\dots,r_{d-1}^*)$. Then, v can be represented by a TT with rank*

$$\boldsymbol{r} = (r_1,\dots,r_d) = (1, r_1^*,\dots,r_{d-1}^*)$$

and $r_0 = d+1$, $r_{d+1} = 1$.

Proof. Let $v_{2:d+1}(x_{2:d+1}) = \hat{U}_1(x_2)\dots\hat{U}_{d+1}(x_{d+1})$ with $\hat{U}_i(x_i) \in \mathbb{R}^{r_{i-1}^*\times r_i^*}$ be a TT representation of $v_{2:d+1}$. Then, $v_1(x_1)\hat{U}_1(x_2)\dots\hat{U}_d(x_{d+1})$ is a TT representation of $v_1(x_1)v_{2:d+1}(x_{2:d+1})$. Since $v_1(x_1) \in \mathbb{R}^{1\times 1}$, the rank of this representation is $(1, r_1^*,\dots,r_d^*)$. The claim now follows with Lemma 2.1. □

3 Compositional Tensor Trains (CTT)

TTs provide efficient representations for many high-dimensional functions, but their performance depends critically on a favourable ordering of variables and the presence of intrinsic low-rank structure. When these conditions are violated, TT ranks can grow rapidly and the format becomes ineffective.

This section develops *Compositional Tensor Trains (CTT)* as a new representation model and establishes their efficiency and approximation power. We first motivate composition by exhibiting functions that are provably awkward for plain TT yet become efficiently encoded via CTT. We then formalize the new CTT format and show how standard function classes are encoded with controlled ranks (affine maps, univariate and multivariate polynomials, concatenation of functions), as well as classical deep neural networks. The representation allows for the derivation of a universal approximation property for CTT under mild assumptions on the feature basis $\boldsymbol{\Phi}$. Moreover, based on an error-propagation analysis, we give a compression algorithm of the layers with guarantees on the precision.

3.1 Where TTs fail

We justify the compositional design by constructing targets whose coefficient tensors have large intermediate TT ranks, while the same targets admit low-rank *compositions* of simple Euler layers, thus exposing the depth advantage exploited by CTT.

3.1.1 Markov processes

As a first example where TTs can fail, consider a discrete time Markov process $X = (X_1, \dots, X_d)$, as in [20, Section 3.2.2] whose density is given by

$$f(x) = f_{d|d-1}(x_d|x_{d-1}) \dots f_{2|1}(x_2|x_1) f_1(x_1),$$

where f_1 is the density of X_1 and $f_{i|i-1}(\cdot|x_{i-1})$ is the conditional density of X_i knowing $X_{i-1} = x_{i-1}$. Let $m_i := \operatorname{rank}(f_{i|i-1})$. Then, the TT ranks of f are

$$\boldsymbol{r} = (1, m_2, m_3, \dots, m_d, 1).$$

If the variables are reordered, for example $\tilde{x} = (x_{\sigma(1)}, \dots, x_{\sigma(d)}) =: P_\sigma(x)$ for some permutation σ, the TT ranks of the function $\tilde{f}(\tilde{x}) = f(P_\sigma^{-1}(\tilde{x}))$ can grow exponentially with the dimension d. For instance, consider the permutation

$$\sigma = \left(1, 3, 5, \dots, 2\left\lfloor \frac{d+1}{2} \right\rfloor - 1, 2, 4, 6, \dots 2\left\lfloor \frac{d}{2} \right\rfloor\right)$$

which places all odd-indexed variables first, followed by the even-indexed variables. The TT ranks then become

$$\tilde{r}_k = \begin{cases} \prod_{j=2}^{\min(2k,d)} m_j, & 1 \le k \le \lceil d/2 \rceil, \\ \prod_{j=2(k-\lceil d/2 \rceil)+2}^{d} m_j, & \lceil d/2 \rceil + 1 \le k \le d, \end{cases}$$

which grow exponentially with d. This growth occurs because the Markov property is local: each variable depends only directly on its immediate predecessor. When variables are reordered non-sequentially, dependencies become effectively non-local in the TT decomposition, requiring large ranks to capture the induced correlations.

We will see later that CTTs can encode such functions efficiently, with one layer encoding the permutation map $P_\sigma^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ (see Section 3.3.1), and one layer encoding f in TT format.

3.1.2 Gaussian densities

As a second example, consider the case of the Gaussian function

$$f_\Gamma(x) := \exp\left(-\frac{1}{2} x^\top \Gamma x\right), \tag{7}$$

where $\Gamma \in \mathbb{R}^{d \times d}$ is a symmetric positive definite precision matrix. In the trivial case where $\Gamma = \operatorname{diag}(\gamma_1, \dots, \gamma_d)$ is diagonal, f_Γ immediately factorises to $f_\Gamma(x) = \prod_{i=1}^d \exp\left(-\frac{1}{2}\gamma_i x_i^2\right)$, which is a rank 1 function. However, when the precision matrix Γ is non-diagonal, the situation changes drastically, depending on the matrix ranks of the sub-diagonal blocks. The approximation of f_Γ by TTs is studied in [47]. For example, the average maximal TT rank required to approximate a 15-dimensional Gaussian with sub-diagonal precision blocks of rank 4 up to a relative L^2 accuracy of 10^{-4} is experimentally shown to be close to 100. In the most general case, the ranks of the sub-diagonal blocks grow linearly in d, and the TT ranks to achieve an accuracy ε grow exponentially in d.

The flow perspective. As we show now, compositions of tensor trains is a better suited format to approximate a Gaussian. First, we define a *lift* from $\mathbb{R}^d$ to $\mathbb{R}^{d+1}$ by $\mathfrak{L}(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}$. Second, we define a flow $\Phi_t : \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ by

$$\begin{aligned} \partial_t \Phi_t(h) &= \left[(-\tfrac{1}{2}\Phi_t(h)_{2:d+1}^\top \Gamma \Phi_t(h)_{2:d+1})\Phi_t(h)_1 \quad 0_d\right], \\ \Phi_0(h) &= h \end{aligned} \tag{8}$$

for any $h \in \mathbb{R}^{d+1}$. Choosing $h = \mathfrak{L}(x)$, we get

$$\Phi_1(\mathfrak{L}(x)) = \left[f_\Gamma(x) \quad x\right].$$

Indeed, it can be easily shown that for $0 \le t \le 1$, we have $\Phi_t(\mathfrak{L}(x)) = \left[\exp\left(-\tfrac{1}{2}tx^\top \Gamma x\right) \quad x\right]$. We recover the Gaussian function by simply using the projection onto the first variable, i.e. $\mathfrak{R}\colon \mathbb{R}^{d+1} \to \mathbb{R}, x \mapsto x_1$, leading to

$$f_\Gamma(x) = \mathfrak{R} \circ \Phi_1 \circ \mathfrak{L}(x).$$

While a similar construction can be used to represent much more general functions by continuous flows, we want to focus on the Gaussian here. The reason for this is that the vector field on the right hand side of (8) has a provably low TT rank, as we show in the following.

Flows with provably low rank. First, note that by Lemma 2.2, the TT rank of the right-hand side of (8) is $(1, \boldsymbol{r})$, where $\boldsymbol{r}$ is the TT rank of the quadratic form $\frac{1}{2}y^\top \Gamma y$, which is bounded by $\frac{d}{2}$ [51, Lemma D.1]. By using polynomial basis functions up to degree 2, i.e. $\Phi = \{1, x, x^2\}$, we can get an exact TT representation of the right-hand side with a complexity in $\mathcal{O}(d^3)$.

Time discretization of the flow. The flow (8) can be generalized to the form

$$\partial_t \Phi_t(h) = f_t(\Phi_t(h)), \tag{9}$$

where $f_t\colon \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ for each $t \in [0,1]$, permitting explicit time dependence of f. Any time discretization of (9), e.g. by Euler or Runge-Kutta methods, leads to a compositional structure in terms of f_t. For a number of intervals $N \in \mathbb{N}$ and time points $t_0 = 0, t_n = \frac{n}{N}$, $n = 1, \dots, N$ such methods compute approximations $\widehat{\Phi}^N(h) \approx \Phi_1(h)$ by various forms of compositions of f_t. For two obvious examples, with $\tau = \frac{1}{N}$, Euler's method is defined by $\widehat{\Phi}^N = (I + \tau f_{t_{N-1}}) \circ (I + \tau f_{t_{N-2}}) \circ \dots \circ (I + \tau f_{t_1}) \circ (I + \tau f_0)$, whereas Heun's method is defined by $\widehat{\Phi}^N = (I + \frac{\tau}{2}(f_{t_{N-1}} + f_{t_N} \circ (I + \tau f_{t_{N-1}}))) \circ \dots$ If the right-hand side of (9) is a TT as is the case of the flow (8), such a discretization immediately defines an approximation of the function $\Phi_1(h)$ as a compositional tensor train (CTT). For many target functions it is not clear whether a suitable flow of the form (9) exists and if it does, if the right-hand side f_t has low-rank structure. For such cases, we need to define a general CTT architecture as well as a method to train it given the target in a way that it can adaptively uncover potential low rank structures.

3.2 Compositional architectures

In the following we introduce the CTT architecture proposed in this work. It is inspired by time discretization of low-rank flows such as for the Gaussian in (8). Furthermore, as we show in the next section, it comes with optimization procedures exploiting efficient tensor algebra.

Definition 3.1 (Compositional tensors)**.** Given linear operators $\mathfrak{L} : \mathbb{R}^d \to \mathbb{R}^p$ and $\mathfrak{R} : \mathbb{R}^p \to \mathbb{R}^{d_o}$, with $d, p, d_o \in \mathbb{N}$ and $p \geq d$, called respectively *lift* and *retraction*, we denote a function $v : \mathbb{R}^d \to \mathbb{R}^{d_o}$ a *compositional tensor (CT)* with $L \in \mathbb{N}$ layers and univariate basis $\Phi := \{\phi_j : \mathbb{R} \to \mathbb{R}\}_{j=1}^n$ with $n \in \mathbb{N}$ if there exist functional tensors $\psi_1, \ldots, \psi_L$ from $\mathbb{R}^p$ to $\mathbb{R}^p$, each with the same univariate basis Φ and a coefficient tensor $\boldsymbol{\psi}_k \in \mathbb{R}^{p \times n \times \ldots \times n}$, defined by

$$\psi_k : x \mapsto \Big(\sum_{i_1, \ldots, i_p} \boldsymbol{\psi}_k(j, i_1, \ldots, i_p) \phi_{i_1}(x_1) \ldots \phi_{i_p}(x_p) \Big)_{1 \leq j \leq p}, \tag{10}$$

such that

$$v(x) = \mathfrak{R} \circ (\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L}(x), \quad x \in \mathbb{R}^d. \tag{11}$$

Additionally, if the tensors $\psi_1, \ldots, \psi_L$ have TT-ranks $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_L$ respectively, we call v a *compositional tensor-train (CTT)* with ranks $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_L$.

The set of *compositional tensors* with basis Φ and with L layers is denoted $\mathcal{CT}^L(\Phi; \mathfrak{L}, \mathfrak{R})$. If the tensors have TT-ranks bounded by $\boldsymbol{r}$, then the corresponding set is denoted $\mathcal{CTT}^L_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$.

In the following, for simplicity we omit $\mathfrak{L}$ and $\mathfrak{R}$ in the notations and write

$$\mathcal{CT}(\Phi) := \bigcup_{L \geq 1} \mathcal{CT}^L(\Phi), \qquad \mathcal{CTT}_{\boldsymbol{r}}(\Phi) := \bigcup_{L \geq 1} \mathcal{CTT}^L_{\boldsymbol{r}}(\Phi).$$

CTTs as DNNs. Note that CTTs and DNNs are closely connected. Indeed, a function u in CTT format admits a recursive representation, letting $u_0(x) = \mathfrak{L}(x)$,

$$u_{k+1}(x) = u_k(x) + \psi_k(u_k(x)), \quad 0 \leq k \leq L - 1,$$

and $u(x) = \mathfrak{R}(u_L(x))$. Assuming $1, \mathrm{Id} \in \Phi$ and with the corresponding feature map $\boldsymbol{\Phi} : \mathbb{R}^p \to \mathbb{R}^{p \times n \times \cdots \times n}$, we can introduce a linear map $\boldsymbol{E} : \mathbb{R}^{pn^p} \to \mathbb{R}^p$ such that $\boldsymbol{E} \operatorname{vec}(\boldsymbol{\Phi}(y)) = y$. Then letting $v_k(x) = \operatorname{vec}(\boldsymbol{\Phi}(u_k(x))) \in \mathbb{R}^{pn^p}$ and $\psi_k(y) = \langle \boldsymbol{\psi}_k, \boldsymbol{\Phi}(y) \rangle_F$, we have that $u(x) = \mathfrak{R}(\boldsymbol{E} v_L(x))$ with the recursion

$$v_{k+1}(x) = \boldsymbol{\Phi}(\boldsymbol{A}_k v_k(x)), \quad 0 \leq k \leq L - 1,$$

where $\boldsymbol{A}_k = \boldsymbol{E} + \boldsymbol{B}_k$, with a linear map $\boldsymbol{B}_k : \mathbb{R}^{pn^p} \to \mathbb{R}^p$ such that $\langle \boldsymbol{\psi}_k, \Phi(y) \rangle_F = \boldsymbol{B}_k \operatorname{vec}(\boldsymbol{\Phi}(y))$. Therefore, a CTT with basis Φ can be seen as a DNN with a latent space of very high dimension pn^p, highly structured linear maps $\boldsymbol{A}_k : \mathbb{R}^{pn^p} \to \mathbb{R}^p$ and nonlinear activation function $y \mapsto \operatorname{vec}(\boldsymbol{\Phi}(y))$ associated with a tensorization of the chosen basis Φ.

3.3 Encoding of classical function classes

Compositional tensors possess the ability to exactly encode specific classes of functions, a property that arises not only from their inherent compositional architecture but also from the structured tensor representation employed at each layer of the model. This allows representation of complex mappings in terms of simpler functions. In particular, the expressive power of these tensors depends crucially on the choice of the basis function Φ. By selecting an appropriate basis (also called features and hence feature space, respectively) such as $\{1, x\}$ or $\{1, \mathrm{ReLU}\}$, the compositional tensor can represent exactly approximation tools such as polynomials or neural networks.

In this section, we provide explicit constructions that illustrate how compositional tensors can encode these function classes exactly. We begin by giving explicit encoding of functions such as linear maps, ridge function, and polynomials. Then, we show that with an appropriate choice of Φ, we can also exactly represent neural networks. This explicit encoding highlights both the theoretical expressivity and the practical utility of compositional tensors in approximating high-dimensional functions.

3.3.1 Linear maps, coordinate transformations

The subsequent result shows that linear affine maps can be efficiently encoded in the TT format.

Proposition 3.2 (Linear affine maps written in the tensor-train format)**.** *The linear map $x \in \mathbb{R}^d \mapsto \langle a, x\rangle$ with $a \in \mathbb{R}^d$ is an additive function, and therefore can be rewritten as a tensor train with ranks $(2, \dots, 2)$. For $\boldsymbol{A} \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, the linear affine map $x \in \mathbb{R}^d \mapsto \boldsymbol{A}x + b \in \mathbb{R}^m$ has a tensor train representation with ranks at most d.*

Proof. The first claim results from the fact that the k-th unfolding of $\langle a, x\rangle = \sum_{i=1}^{k} a_i x_i + \sum_{i=k+1}^{d} a_i x_i$ has rank $r_k = 2$, for all $1 \le k \le d-1$. Letting a_i be the i-th column of $\boldsymbol{A}$, a suitable representation is given by $\hat{U}_1(x_1) = \begin{pmatrix} x_1 a_1 + b & a_2 & \dots & a_d \end{pmatrix} \in \mathbb{R}^{m\times d}$, $\hat{U}_k(x_k) = \boldsymbol{I}_d + (x_k - 1)e_k e_k^\top = \mathrm{diag}(1, \dots, x_k, \dots, 1) \in \mathbb{R}^{d\times d}$ for $2 \le k \le d-1$ and $\hat{U}_d(x_d) = (1, \dots, 1, x_d)^\top \in \mathbb{R}^{d\times 1}$. □

Since the above result also holds for the linear map $h \mapsto -h + \boldsymbol{A}h$, we get the following corollary.

Corollary 3.3. *Any linear affine transformation $h \in \mathbb{R}^p \mapsto \boldsymbol{A}h + b \in \mathbb{R}^p$ in the lifted space, with $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ and $b \in \mathbb{R}^p$ can be represented by a single CTT layer with rank at most p.*

The above result shows that a CTT format with 2 layers can represent functions of the form $u(\boldsymbol{A}x)$, with u represented in TT format. Consequently, a CTT can encode linear coordinate transformations. This solves a main difficulty of the TT format (mentioned in Section 3.1.1), which is the choice of a good ordering of the variables, usually addressed using prior information on the function [12] or stochastic optimization procedures [19, 20, 36].

3.3.2 Polynomials

The following result shows that univariate polynomials can be represented exactly, provided we use a suitable lift.

Proposition 3.4. *Let $d \in \mathbb{N}$, and $(a_0, \dots, a_d) \subset \mathbb{R}$. We consider the polynomial*

$$g : x \mapsto a_0 + a_1 x + \cdots + a_d x^d.$$

Then, letting $\mathfrak{L} : x \mapsto (0, x)$, $\mathfrak{R} : (x, y) \mapsto x$ and

$$\begin{aligned} &f_1 : (x, y) \mapsto (a_d, 0), \\ &f_k : (x, y) \mapsto (a_{d-k+1} + x(y-1), 0), \quad 2 \le k \le d+1, \end{aligned}$$

it holds that

$$\mathfrak{R} \circ (\mathrm{Id} + f_{d+1}) \circ \cdots \circ (\mathrm{Id} + f_1) \circ \mathfrak{L} = g.$$

This shows that any univariate polynomial *can be represented by a* compositional tensor *with $d+1$ layers and with the basis $\Phi = \{1, \mathrm{Id}\}$.*

Proof. The proof is based on the Horner's method which rewrites the evaluation $g(x)$ as the recursion

$$b_d = a_d, \quad b_k = a_k + b_{k+1} x, \quad k = d-1, d-2, \dots, 1.$$

To show that indeed $\mathfrak{R} \circ (\mathrm{Id} + f_{d+1}) \circ \cdots \circ (\mathrm{Id} + f_1) \circ \mathfrak{L} = g$, we only have to show that for any $k \ge 1$ and $x \in \mathbb{R}$, we have

$$\langle (\mathrm{Id} + f_k) \circ \cdots \circ (\mathrm{Id} + f_1) \circ \mathfrak{L}(x), e_1 \rangle = b_{d-k+1}.$$

Then, since $\mathfrak{R}$ is only the projection onto the first variable, the result follows.

First, for $k = 1$ we have

$$(\mathrm{Id} + f_1) \circ \mathfrak{L}(x) = (0, x) + (a_d, 0) = (a_d, x) = (b_d, x).$$

Next, suppose that we have $(\mathrm{Id} + f_k) \circ \cdots \circ (\mathrm{Id} + f_1) \circ \mathfrak{L}(x) = (b_{d-k+1}, x)$ for $1 \le k \le p$. Then, since $f_{k+1} : (x, y) \mapsto (a_{d-k} + x(y-1), 0)$, we have

$$(\mathrm{Id} + f_{k+1}) \circ \cdots \circ (\mathrm{Id} + f_1) \circ \mathfrak{L}(x) = (a_{d-k} + b_{d-k+1} x, x) = (b_{d-k}, x),$$

which concludes the proof. □

Exploiting the fast exponentiation offered by the compositional format, we give an explicit encoding of a (sparse) multivariate polynomial. Let

$$P(x_1, \dots, x_d) = \sum_{\alpha \in \Lambda} C_\alpha x^\alpha, \quad x^\alpha = \prod_{j=1}^{d} x_j^{\alpha_j},$$

where $\Lambda \subset \mathbb{N}^d$ contains the set of multi-indices corresponding to non-zero coefficients C_α in the monomial basis. Define the maximal degree for each variable x_j by

$$N_j := \max\{\alpha_j : \alpha \in \Lambda\} \in \mathbb{N}, \quad j = 1 \dots d. \tag{12}$$

The case $\Lambda = \emptyset$ is trivial, and we do not consider it.

Proposition 3.5 (Encoding of multivariate polynomials)**.** *Let P be a multivariate polynomial in d variables, $\Lambda \subset \mathbb{N}^d$ the set of multi-indices associated with its non-zero coefficients. Suppose that $\Phi = \{1, \mathrm{Id}\}$. Let $q \geq 2$ be the number of* workspace slots *and $p = d + 2 + q$. Then, there exists a lift $\mathfrak{L} : \mathbb{R}^d \to \mathbb{R}^p$ and a retraction $\mathfrak{R} : \mathbb{R}^p \to \mathbb{R}$, respectively defined by*

$$\mathfrak{L}(x) = (0, x_1, \dots, x_d, 0, \dots, 0), \quad \mathfrak{R}(y) = y_1,$$

and L functions $\psi_1, \dots \psi_L$ in TT format such that

$$P = \mathfrak{R} \circ (\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L},$$

with

$$L = |\Lambda|(2 + d) + \sum_{\alpha \in \Lambda} \sum_{j=1}^{d} \lfloor \log_q(\alpha_j + 1) \rfloor = \mathcal{O}\left(d|\Lambda| \lfloor \log_q(N_{\max} + 1) \rfloor\right),$$

where $N_{\max} := \max_j N_j$. Moreover, the number of non-zero parameters in the encoding is $\mathcal{O}(|\Lambda| d[1 + \lfloor \log_q(N_{\max} + 1) \rfloor])$ and the number of parameters is $\mathcal{O}(L(d + q))$. The TT ranks of the functions ψ_ℓ are bounded by $\boldsymbol{r} = (p, 2, \dots, 2)$.

Proof. We decompose the state variable $y \in \mathbb{R}^p$, with $p = d + 2 + q$, as

$$y = (\underbrace{y_1}_{\text{accumulator}}, \underbrace{y_2, \dots, y_{d+1}}_{\text{variables}}, \underbrace{y_{d+2}}_{\text{register}}, \underbrace{y_{d+3}, \dots, y_{d+2+q}}_{\text{workspace}}).$$

The idea of the encoding is to iterate over the $\alpha \in \Lambda$ and construct iteratively the polynomial P by adding monomials successively to the accumulator y_1. For doing so, we loop over all the variables x_j, compute the powers $x_j^{\alpha_j}$ and multiplying to the register y_{d+2} in order to obtain the monomial $c_\alpha \prod_{j=1}^d x_j^{\alpha_j}$. In order to compute these powers, we use the "workspace slots" $y_{d+3}, \dots, y_{d+2+q}$ and *fast exponentiation*, which computes a^n, for some number a and power n, in a recursive manner and using only $\mathcal{O}(\log n)$ operations. Indeed, for $\alpha \in \Lambda$, the base-q representation of α_j, for $1 \leq j \leq d$, is given by

$$\alpha_j = \sum_{k=0}^{K_{\alpha,j}-1} \alpha_{j,k} q^k, \quad \alpha_{j,k} \in \{0, \dots, q-1\},$$

where $K_{\alpha,j} := \lfloor \log_q(\alpha_j + 1) \rfloor$. Next, when raising x_j to the power α_j we get

$$x_j^{\alpha_j} = x_j^{\sum_{k=0}^{K_{\alpha,j}-1} \alpha_{j,k} q^k} = \prod_{k=0}^{K_{\alpha,j}-1} \left(x_j^{q^k}\right)^{\alpha_{j,k}}.$$

We give below the encoding of P in the form of an algorithm, counting the number of non-zero parameters and TT ranks.

Algorithm 1: Computes the α-th monomial

Store $y_{d+2} \leftarrow c_\alpha$ ▷ *1 layer, 2 non-zero parameters, TT ranks* $\boldsymbol{r} = (p, 2, \dots, 2)$

for $j \leftarrow 1$ **to** d **do**

  Store $y_{d+3}, \dots, y_{d+2+q} \leftarrow x_j$ ▷ *1 layer, 2 non-zero parameters, TT ranks* $\boldsymbol{r} = (p, 2, \dots, 2)$

  for $k \leftarrow 0$ **to** $K_{\alpha,j} - 2$ **do**

    Store $y_{d+2} \leftarrow y_{d+2} \prod_{i=1}^{\alpha_{j,k}} y_{d+2+i}$ ▷ *computes* $\left(x_j^{q^k}\right)^{\alpha_{j,k}}$, $1 + \alpha_{j,k}$ *non-zero parameters, TT ranks* $\boldsymbol{r} = (p, 2, \dots, 2)$

    Store $y_{d+3}, \dots, y_{d+2+q} \leftarrow \prod_{i=1}^{q} y_{d+2+i}$ ▷ *computes* $x_j^{q^{k+1}}$, $1 + q$ *non-zero parameters, TT ranks* $\boldsymbol{r} = (p, 2, \dots, 2)$

  end

  Store $y_{d+2} \leftarrow y_{d+2} \prod_{i=1}^{\alpha_{j,k}} y_{d+2+i}$ ▷ *1 layer,* $1 + \alpha_{j,k}$ *non-zero parameters, TT ranks* $\boldsymbol{r} = (p, 2, \dots, 2)$

end

Add monomial to polynomial $y_1 \leftarrow y_1 + y_{d+2}$ ▷ *1 layer, 1 non-zero parameter, TT ranks* $\boldsymbol{r} = (p, 1, \dots, 1)$

Therefore, for constructing the polynomial P we need

$$\sum_{\alpha \in \Lambda} \left(1 + \sum_{j=1}^{d} (1 + K_{\alpha,j} - 1 + 1) + 1\right) \leq |\Lambda| \left(2 + d + d\lfloor \log_q(N_{\max} + 1) \rfloor\right)$$

layers and the TT ranks are bounded by $\boldsymbol{r} = (p, 2, \dots, 2)$. The reason for a 2 to appear in $\boldsymbol{r}$ is that in the definition of CTTs, a layer is of the form $\mathrm{Id} + \psi$, so we have to remove the previous value of the state y and replace it with the new one. □

Remark 3.6 (Special case $q = 1$)**.** *If we allow q to be equal to 1, then we can consider the base-2 representation of α_j since we already have x_j in the state y_{1+j}, and one copy in the workspace y_{d+3}. But in order to compute the power $x_j^{\alpha_j}$, we have to evaluate α_j multiplications. Consequently, the number of layers is*

$$L = |\Lambda|(2 + d) + \sum_{\alpha \in \Lambda} \sum_{j=1}^{d} \alpha_j = \mathcal{O}\left(d|\Lambda|N_{\max}\right),$$

which is linear in $N_{\max}$ instead of $\log_2(N_{\max})$ when using fast exponentiation.

Remark 3.7 (Encoding with a number of layers in $\mathcal{O}(|\Lambda|)$)**.** *If we allow the width to depend on the degrees N_j (12), then it is possible to encode P with complexity $\mathcal{O}(|\Lambda|)$. Indeed, by letting $p = 1 + \sum_{j=1}^{d} N_j$, we store N_j copies of x_j, so we can directly compute $x_j^{\alpha_j}$ for each $\alpha \in \Lambda$ and add the monomial $c_\alpha \prod_{j=1}^{d} x_j^{\alpha_j}$ to the accumulator. However, encoding polynomial approximation tools with adaptive polynomial degree would require CTTs with adaptive width.*

3.3.3 Neural networks

By the preceding results, the CTT architecture is able to represent coordinate transformations in early layers, which may highly reduce the required rank of later layers. An immediate consequence of the ability to represent linear transformations, is that the CTT format can also represent certain types of neural networks, provided the basis is chosen correctly.

Proposition 3.8 (Concatenation of two CTs)**.** *Let f and g be two compositional tensors with L_f (resp. L_g) layers, width p_f (resp. p_g), m_f (resp. m_g) non-zero parameters, lift $\mathfrak{L}_f$ (resp. $\mathfrak{L}_g$), retraction $\mathfrak{R}_f$ (resp. $\mathfrak{R}_g$), and common basis Φ such that $\{1\} \subset \Phi$. Then, there exists a compositional tensor h with $L := \max(L_f, L_g)$ layers, width $p := p_f + p_g$, $m := m_f + m_g$ non-zero parameters, lift $\mathfrak{L}(x) := (\mathfrak{L}_f(x), \mathfrak{L}_g(x))$, retraction $\mathfrak{R}(y) := (\mathfrak{R}_f(y^{(f)}), \mathfrak{R}_g(y^{(g)}))$, where $y = (y^{(f)}, y^{(g)})$, and such that $h(x) = (f(x), g(x))$. The TT ranks $\boldsymbol{r}^{\boldsymbol{\ell}}$ of the tensor $\boldsymbol{\psi}_\ell$ in layer ℓ are given by*

$$\begin{aligned} r_0^\ell &= r_0^{\ell,f} + r_0^{\ell,g}, \\ r_k^\ell &= r_k^{\ell,f} + 1, \quad 1 \le k \le p_f - 1, \\ r_{p_f}^\ell &= 1 + r_1^{\ell,g}, \\ r_{p_f+t+1}^\ell &= 1 + r_{t+1}^{\ell,g}, \quad 1 \le t \le p_g - 2, \\ r_{p_f+p_g}^\ell &= 1. \end{aligned} \tag{13}$$

Proof. The encoding of the concatenation $h(x) = (f(x), g(x))$ is done in an explicit block-wise way. The f-layers (resp. g-layers) act only the first (resp. second) block. Because these blocks are disjoint, a single layer can simultaneously perform one f-step and one g-step. If one CT has fewer layers, we pad it with identity layers. Thus, we require only $\{1\} \subset \Phi$ and $\max(L_f, L_g)$ layers. Without loss of generality, we assume that $\phi_1(x) = 1$. The state of f (resp. g) are denoted $y^{(f)}$ (resp. $y^{(g)}$), and we set $y := (y^{(f)}, y^{(g)})$.

We start by re-indexing the original layer tensors so they are defined for all $\ell = 1 \dots L$ by padding with zero-layers. For $1 \le \ell \le L_f$ (resp. $1 \le \ell \le L_g$) let $\psi_\ell^{(f)}$ (resp. $\psi_\ell^{(g)}$) be the given f-layer (resp. g-layer), and for $\ell > L_f$ (resp. $\ell > L_g$) define $\psi_\ell^{(f)} = 0$ (resp. $\psi_\ell^{(g)} = 0$).

Now we give the explicit encoding of the layers ψ_ℓ. If $1 \le j \le p_f$ (an f-output coordinate), set

$$\boldsymbol{\psi}_\ell(j, i_1, \dots, i_{p_f}, i_{p_{f+1}}, \dots, i_p) = \begin{cases} \boldsymbol{\psi}_\ell^{(f)}(j, i_1, \dots, i_{p_f}) & \text{if } i_{p_{f+1}} = \dots = i_p = 1, \\ 0 & \text{otherwise,} \end{cases}$$

and if $p_f < j \le p$ (an g-output coordinate), set

$$\boldsymbol{\psi}_\ell(j, i_1, \dots, i_{p_f}, i_{p_{f+1}}, \dots, i_p) = \begin{cases} \boldsymbol{\psi}_\ell^{(g)}(j - p_f, i_{p_f+1}, \dots, i_p) & \text{if } i_1 = \dots = i_{p_f} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{r}^{\boldsymbol{\ell},\boldsymbol{f}}$ (resp. $\boldsymbol{r}^{\boldsymbol{\ell},\boldsymbol{g}}$) be the TT ranks of $\boldsymbol{\psi}_\ell^{(f)}$ (resp. $\boldsymbol{\psi}_\ell^{(g)}$), with $r_0^{\ell,f} = p_f$ (resp. $r_0^{\ell,g} = p_g$) and $r_{p_f}^{\ell,f} = 1$ (resp. $r_{p_g}^{\ell,g} = 1$). Then the TT ranks $\boldsymbol{r}^{\boldsymbol{\ell}}$ of $\boldsymbol{\psi}_\ell$ are given by (13). In fact, $\boldsymbol{\psi}_\ell$ can be written as the sum of two tensors A_ℓ and B_ℓ. Let the TT cores of $\boldsymbol{\psi}_\ell^{(f)}$ (resp. $\boldsymbol{\psi}_\ell^{(g)}$) be $F_\ell^{(k)} \in \mathbb{R}^{r_{k-1}^{\ell,f} \times n_k \times r_k^{\ell,f}}$ (resp. $G_\ell^{(k)} \in \mathbb{R}^{r_{k-1}^{\ell,g} \times n_k \times r_k^{\ell,g}}$), with $1 \le k \le p_f$ (resp. p_g). Moreover, let $E_1 \in \mathbb{R}^{1 \times n \times 1}$ be such that $E_1(1, i, 1) = \delta_{i,1}$. Observe that

- A_ℓ is equal to $\boldsymbol{\psi}_\ell^{(f)}$ when $i_{p_f+1} = \cdots = i_p = 1$ and 0 otherwise, and can be written as

$$A_\ell(j, i_1, \ldots, i_{p_f}, i_{p_f+1}, \ldots, i_p) = F_\ell^{(1)}(j, i_1, :) F_\ell^{(2)}(:, i_2, :) \ldots F_\ell^{(p_f)}(:, i_{p_f}, 1) \times \underbrace{E_1(1, i_{p_f+1}, 1) \ldots E_1(1, i_p, 1)}_{p_g}.$$

- B_ℓ can be written with p_f rank-1 cores for the condition $i_1 = \cdots = i_{p_f} = 1$ and symmetrically

$$B_\ell(j, i_1, \ldots, i_{p_f}, i_{p_f+1}, \ldots, i_p) = \underbrace{E_1(1, i_1, 1) \ldots E_1(1, i_{p_f}, 1)}_{p_f} \times G_\ell^{(1)}(j, i_{p_f+1}, :) G_\ell^{(2)}(:, i_{p_f+2}, :) \ldots G_\ell^{(p_g)}(:, i_p, 1).$$

Finally, by letting $\mathfrak{L}(x) := (\mathfrak{L}_f(x), \mathfrak{L}_g(x))$ and $\mathfrak{R}(y) := (\mathfrak{R}_f(y^{(f)}), \mathfrak{R}_g(y^{(g)}))$, we have $h(x) := \mathfrak{R} \circ (\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L}(x)$ and this compositional tensor gives exactly the concatenation of f and g. □

Proposition 3.9 (Vectorization of activation function). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an activation function representable by a compositional tensor with L_σ layers, width p_σ, m_σ non-zero parameters, lift $\mathfrak{L}_\sigma$, retraction $\mathfrak{R}_\sigma(y) = y_1$, and with basis Φ such that $\{1\} \subseteq \Phi$. Then the vectorization $\boldsymbol{\sigma}(x_1, \ldots, x_d) = (\sigma(x_1), \ldots, \sigma(x_d))$ can be represented by a compositional tensor with L_σ layers, width $p := d \cdot p_\sigma$, $m := d \cdot m_\sigma$ non-zero parameters, lift $\mathfrak{L}(x) := (\mathfrak{L}_\sigma(x_1), \ldots, \mathfrak{L}_\sigma(x_d)) \in \mathbb{R}^p$, retraction $\mathfrak{R}(y) := (y_1, y_{1+p_\sigma}, \ldots, y_{1+(d-1)p_\sigma}) \in \mathbb{R}^d$. The TT ranks $\boldsymbol{r}^\ell$ of the tensor $\boldsymbol{\psi}_\ell$ in layer ℓ are given by*

$$\begin{aligned} r_0^\ell &= \sum_{k=1}^{d} r_0^{\ell,\sigma} = d \cdot p_\sigma, \\ r_K^\ell &= r_j^{\ell,\sigma} + (d-1), \quad K = (i-1) + j, \\ r_{dp_\sigma}^\ell &= 1, \end{aligned}$$

where $1 \le i \le d$ is in the index of the block, and $1 \le j \le p_\sigma - 1$ is the position within the block.

Proof. It is a direct consequence of Proposition 3.8. We apply σ independently to each coordinate y_{1+kp_σ}, $0 \le k \le d-1$. Each input coordinate x_i is lifted by $\mathfrak{L}_\sigma$, and then the L_σ layers of σ are applied to each block corresponding to the lifted input coordinate x_i. The retraction $\mathfrak{R}$ gets the output of the composition of tensors of each block.

Similarly as in Proposition 3.8, the tensor $\boldsymbol{\psi}_\ell$ in the ℓ-th layer can be written as a sum of d tensors $U_\ell^{(k)}$, $1 \le k \le d$, where $U_\ell^{(k)}$ acts on block k and is padded by rank-1 constant cores on all other blocks. This yields the desired result. □

Theorem 3.10 (Encoding of a DNN). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an activation function representable by a compositional tensor with L_σ layers, width p_σ, m_σ non-zero parameters, lift $\mathfrak{L}_\sigma(x) = (x, 0, \dots, 0)$, retraction $\mathfrak{R}_\sigma(y) = y_1$, and with basis Φ such that $\{1, x\} \subseteq \Phi$. Let $f = T_L \circ \sigma \circ T_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ T_1$ be a deep neural network of width $p_f \geq d, d_o$ with affine transformations $T_1 : \mathbb{R}^d \to \mathbb{R}^{p_f}$, $T_\ell : \mathbb{R}^{p_f} \to \mathbb{R}^{p_f}$ for $2 \leq \ell \leq L-1$, and $T_L : \mathbb{R}^{p_f} \to \mathbb{R}^{d_o}$. Then, f can be exactly encoded by a compositional tensor with $L + (L-1)L_\sigma$ layers, width $p := p_f \cdot p_\sigma$, lift $\mathfrak{L}$ and retraction $\mathfrak{R}$ given by*

$$\mathfrak{L}(x)_{i,k} = \begin{cases} x_i & \text{if } 1 \leq i \leq d_f \text{ and } k = 1, \\ 0 & \text{otherwise} \end{cases} \in \mathbb{R}^{p_f p_\sigma},$$
$$\mathfrak{R}(y) = (y_{1,1}, \dots, y_{d_o,1}) \in \mathbb{R}^{d_o}.$$

Moreover, the number of non-zero parameters is given by

$$\sum_{\ell=1}^{L-1} (\|A_\ell - I\|_0 + \|b_\ell\|_0) + (\|A_L - I\|_0 + \|b_L\|_0) + (L-1)p_f m_\sigma.$$

Suppose that the L_σ of σ have TT ranks bounded by $\boldsymbol{r_\sigma}$. Then the TT ranks of each affine layer are bounded by $(p, p_f, \dots, p_f)$, and those for computing σ are bounded by $\boldsymbol{r_\sigma}$.

If $\sigma \in \operatorname{span} \Phi$, then $p_\sigma = 1$, $m_\sigma = 1$, $L_\sigma = 1$ and $\mathfrak{L}_\sigma(x) = x$, which results in an encoding of f by a compositional tensor with $2L-1$ layers, with width p_f, and the TT ranks are bounded by $(p, p_f, \dots, p_f)$.

Proof. We represent each neuron j, $1 \leq j \leq p_f$, by a block of p_σ coordinates in the CT state. This allows us to apply the CT representation of σ to all neurons in parallel. Consequently, the total width required for the encoding of f in a CT is $p := p_\sigma p_f$. Within each block, the first coordinate stores the current scalar value of the neuron, while the remaining coordinates are used for computing σ through its CT representation. Define the lift $\mathfrak{L} : \mathbb{R}^d \to \mathbb{R}^{p_f p_\sigma}$ by $\mathfrak{L}(x)_{i,j} = x_i$ if $1 \leq i \leq d_f$ and $j = 1$ and $\mathfrak{L}(x)_{i,j} = 0$ otherwise. This choice of lift is motivated by the fact that $\mathfrak{L}_\sigma(x) = (x, 0, \dots, 0)$ and by Proposition 3.9. The state is $y = (y_k)_{k=1}^p$ and we reindex it as $y = (y_{1,1}, \dots, y_{1,p_\sigma}, y_{2,1}, \dots, y_{p_f,p_\sigma})$ where the first index denotes the block from 1 to p_f and the second index corresponds to the local coordinate, from 1 to p_σ.

Encoding of the affine layers. The encoding of the affine layers T_ℓ can be done similarly as in Proposition 3.2. If $p_\sigma = 1$, we may directly apply Proposition 3.2. If $p_\sigma > 1$, the affine layer T_ℓ must act only on the first coordinate $y_{i,1}$ of each block. The construction is given for $2 \leq \ell \leq L-1$; the cases $\ell = 1$ and $\ell = L$ are handled analogously. Since $p_\sigma > 1$, the matrix A_ℓ and the vector b_ℓ must be modified such that they only act on the variables $y_{i,1}$, $1 \leq i \leq p_f$. Define the matrix $\widetilde{A_\ell} \in \mathbb{R}^{p \times p_f}$ and vector $\widetilde{b_\ell} \in \mathbb{R}^p$ such that

$$(\widetilde{A_\ell})_{k,j} := [(A_\ell)_{s,j} - \delta_{s,j}]\delta_{k,(s-1)p_\sigma+1}, \quad (\widetilde{b_\ell})_k := (b_\ell)_s \delta_{k,(s-1)p_\sigma+1},$$

for $1 \leq s, j \leq p_f$ and $(s-1)p_\sigma + 1 \leq k \leq sp_\sigma$. Thus, only rows with indices k such that $k \equiv 1 \pmod{p_\sigma}$ are non-zero. Following Proposition 3.2 define

$$
\begin{aligned}
U_1^\ell(y_1) &= \begin{pmatrix} y_1(\widetilde{A_\ell})_{:,1} + \widetilde{b_\ell} & (\widetilde{A_\ell})_{:,2} & (\widetilde{A_\ell})_{:,p_f} \end{pmatrix} \in \mathbb{R}^{p \times p_f}, \\
U_k^\ell(y_k) &= I_{p_f} + \mathbf{1}_{k \equiv 1 \pmod{p_\sigma}} (y_k - 1) e_s e_s^\top \in \mathbb{R}^{p_f \times p_f}, \quad 2 \leq k = (s-1)p_\sigma + 1 \leq p-1, \\
U_p^\ell(y_p) &= (1, \dots, 1)^\top \in \mathbb{R}^{p_f \times 1},
\end{aligned}
$$

where $e_s \in \mathbb{R}^{p_f}$. Using these cores to define the tensor ψ_ℓ yields $\psi_\ell(y)_k = T_\ell(y_{1,1}, \dots, y_{p_f,1})_s - y_{s,1}$ for $k = (s-1)p_\sigma + 1$, $1 \leq s \leq p_f$, and $\psi_\ell(y)_k = 0$ otherwise. Therefore $(\mathrm{Id} + \psi_\ell)(y)_k = T_\ell(y_{1,1}, \dots, y_{p_f,1})_s$ for components $k = (s-1)p_\sigma + 1$, $1 \leq s \leq p_f$, and y_k otherwise. Thus, the TT ranks for encoding an affine layer is bounded by $(p, p_f, \dots, p_f)$.

Application of vectorized σ. We want to apply σ independently to each coordinate $y_{i,1}$. As $\mathfrak{L}_\sigma(x) = (x, 0, \dots, 0)$, we can directly apply the L_σ layers $\psi^{(\sigma,1)}, \dots, \psi^{(\sigma,L_\sigma)}$ of σ, as from the previous paragraph we have the output of the affine layer on the coordinates $(y_{1,1}, \dots, y_{p_f,1})$. The retraction $\mathfrak{R}_\sigma$ of σ can be absorbed by the layers. Hence, the number of non-zero parameters for encoding an affine layer is

$$\|A_\ell - I\|_0 + \|b_\ell\|_0$$

and the TT ranks are bounded by those of σ. In total, the construction uses $L + (L-1)L_\sigma$ layers, and the total number of non-zero parameters is

$$\sum_{\ell=1}^{L-1} (\|A_\ell - I\|_0 + \|b_\ell\|_0) + (\|A_L - I\|_0 + \|b_L\|_0) + (L-1)p_f m_\sigma.$$

□

Remark 3.11 (CTT can exactly represent a deep ReLU neural network)**.** *Consider feature functions $\phi_1(x) = 1$, $\phi_2(x) = x$, $\phi_3(x) = |x|$. The ReLU activation function $\sigma(x) = \max(x, 0) = \frac{x+|x|}{2} = \frac{1}{2}\phi_2(x) + \frac{1}{2}\phi_3(x)$. By Corollary 3.3, we can represent the affine transformation $\boldsymbol{A} \cdot x + b$, $\boldsymbol{A} \in \mathbb{R}^{p \times p}$, $b \in \mathbb{R}^p$, by a single CTT layer of rank at most p. Therefore, a deep ReLU neural network with L layers and constant width p can be written as a CTT with $2L$ layers.*

3.4 Universal approximation

To further motivate the composition of tensors and illustrate their potential, we show that they are universal approximators in $L^p_\mu(\mathcal{X}; \mathbb{R})$ and $C(\mathcal{X}; \mathbb{R})$ for a large class of bases Φ. The encoding of multivariate polynomials give the universality in these spaces. A rich body of results on the universality of deep neural networks has been obtained, cf. [30, 10, 33] to name but a few. It has also been shown in Theorem 3.10 that DNNs can be exactly represented by CTTs, leading to a connection between their approximation classes.

We give here a universality result for the approximation with compositional tensors and CTTs for basis Φ that can represent the constants and the identity function Id.

Corollary 3.12 (Universality in $C(\mathcal{X})$)**.** *Assume that $1, \mathrm{Id} \in \operatorname{span}\Phi$. Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact domain, and consider width $p := d+1$. Then the set of compositional tensors $\mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ and of compositional tensor trains $\mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$ with $\mathfrak{L}(x) := (0, x)$, $\mathfrak{R}(y) = y_1$ and $\boldsymbol{r} \geq (d+1, 2, \dots, 2)$ are dense in $C(\mathcal{X})$ equipped with the sup norm $\|\cdot\|_\infty$. The lower bound on $\boldsymbol{r}$ is tight.*

Proof. Proposition 3.5 shows that with this choice of basis Φ, width $p = d+1$, lift $\mathfrak{L}$ and $\mathfrak{R}$, we can represent any multivariate polynomial. As a direct consequence of the Stone-Weierstrass theorem, if we let the degree of the polynomial go to infinity (and so the number of layers go to infinity), we have that $\mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ is dense in $C(\mathcal{X})$ equipped with the sup norm.

Moreover, Proposition 3.5 shows that the layers can be exactly represented by TTs with ranks equal to $\boldsymbol{r} = (d+1, 2, \dots, 2)$, which gives the density of $\mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$ in $C(\mathcal{X})$ for the sup norm. □

Remark 3.13. *A direct consequence of Corollary 3.12 is that, since $C(\mathcal{X})$ is dense in $L^q_\mu(\mathcal{X})$, with $1 \leq q < \infty$ and μ a finite measure, we have the density of $\mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ and $\mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$ in $L^q_\mu(\mathcal{X})$.*

In an analogous manner to the approach of [46, 31], the universality in $L^q_\mu(\mathbb{R}^d)$, $1 \leq q < \infty$, for a finite measure μ is shown employing the density argument of the space of compactly supported continuous functions $C_c(\mathbb{R}^d)$ in $L^q_\mu(\mathbb{R}^d)$ for the norm $\|\cdot\|_{L^q_\mu}$.

Corollary 3.14 (Universality in $L^q_\mu(\mathbb{R}^d)$)**.** *Let μ be a regular Borel finite measure, $1 \leq q < \infty$, and $1, \mathrm{Id} \in \operatorname{span}\Phi$. Let the width be $p := d+1$. Then the set of compositional tensors $\mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ and of compositional tensor trains $\mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$, with $\mathfrak{L}(x) := (0, x)$, $\mathfrak{R}(y) = y_1$ and $\boldsymbol{r} = (d+1, 2, \dots, 2)$, are dense in $L^q_\mu(\mathbb{R}^d)$ for the norm $\|\cdot\|_{L^q_\mu}$.*

Proof. The proof is based on the fact that the space of continuous functions with compact support $C_c(\mathbb{R}^d)$ is dense in $L^q_\mu(\mathbb{R}^d)$, and that by Corollary 3.12 we can approximate any continuous function on a compact subset.

Let $f^* \in L^q_\mu(\mathbb{R}^d)$ and $\varepsilon > 0$. Because μ is a finite measure, we can choose a radius $R > 0$ such that

$$\|f^* - f_R\|_{L^q_\mu} \leq \frac{\varepsilon}{3},$$

where $f_R := f^* \cdot \mathbf{1}_{B(0,R)}$ and $B(0, R)$ is the ball of radius R. Since μ is a regular Borel finite measure, there exists a continuous $g \in C_c(\mathbb{R}^d)$ with support in some compact $K \subset B(0, R)$ [15, Chapter 7] with

$$\|f_R - g\|_{L^q_\mu} \leq \frac{\varepsilon}{3}.$$

The density of $\mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ (resp. $\mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$) in $C(K)$ for $\|\cdot\|_\infty$ given by Corollary 3.12 gives the existence of a function $h \in \mathcal{CT}(\Phi; \mathfrak{L}, \mathfrak{R})$ (resp. $h \in \mathcal{CTT}_{\boldsymbol{r}}(\Phi; \mathfrak{L}, \mathfrak{R})$) with

$$\|g - h\|_\infty \leq \frac{\varepsilon}{3\mu(K)^{1/q}}.$$

Then $\|g - h\|_{L^q_\mu} \leq \mu(K)^{1/q} \|g - h\|_\infty \leq \frac{\varepsilon}{3}$.

Combining the three estimates with the triangle inequality gives the final result

$$\|f^* - h\|_{L^q_\mu} \leq \|f^* - f_R\|_{L^q_\mu} + \|f_R - g\|_{L^q_\mu} + \|g - h\|_{L^q_\mu} \leq \varepsilon.$$

□

Remark 3.15 (Approximation spaces)**.** *From the previous results, we see a relation between the* approximation spaces *of DNNs and CTTs. The complexity of DNNs and CTTs is chosen to be the* number of parameters*. The width of both is* fixed to $p_f \geq d$*, which means only the number of layers is a free parameter. The corresponding approximation tools are denoted* $\Sigma^{DNN} = (\Sigma_n^{DNN})_{n\in\mathbb{N}}$ *and* $\Sigma^{CTT} = (\Sigma_n^{CTT})_{n\in\mathbb{N}}$.

Recall that the approximation space *associated with the approximation tool* Σ *is defined by* $A_q^\alpha(X,\Sigma) := \{f \in X : \|f\|_{A_q^\alpha} < \infty\}$*, with* $\|f\|_{A_q^\alpha} := \left[\sum_{n=1}^\infty (n^\alpha E(f,\Sigma_{n-1})_X)^q \frac{1}{n}\right]^{1/q}$ *for* $0 < q < \infty$ *and* $\|f\|_{A_\infty^\alpha} := \sup_{n\geq 1} n^\alpha E(f,\Sigma_{n-1})_X$*, where* $E(f,\Sigma_n)_X$ *is the best approximation of* f *by* Σ_n *(see [13]).*

By Theorem 3.10, the TT ranks are bounded by p_f*, which gives* $\mathcal{O}(Lp_f^3)$ *parameters so there exists* $c \geq 1$ *such that* $\Sigma_n^{DNN} \subseteq \Sigma_{cn}^{CTT}$*. This inclusion gives* $A_q^\alpha(X,\Sigma^{DNN}) \hookrightarrow A_q^\alpha(X,\Sigma^{CTT})$*, i.e.,* $A_q^\alpha(X,\Sigma^{DNN})$ *is* continuously embedded *in* $A_q^\alpha(X,\Sigma^{CTT})$.

The converse remains an open question.

A further analysis of approximation properties of CTT is let for a future work.

3.5 Compression of a CTT

It is well-known that given a tensor $u \in \mathcal{V}_{\boldsymbol{n}}$ we can find a tensor $\tilde{u} \in \mathcal{T}_{\boldsymbol{n},\boldsymbol{r}}$ in the tensor-train format such that $\|u - \tilde{u}\| \leq \varepsilon\|u\|$, where $\|\cdot\|$ is any inner product norm induced by inner product norms on univariate function spaces by using the TT-SVD [44].

One would extend this result to a compositional tensor-train, that is, we are looking for a method to compress a CTT $g = \mathfrak{R} \circ (\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L}$, with $\psi_k \in \mathcal{T}_{\boldsymbol{n},\boldsymbol{r}}$, by another CTT $\tilde{g} = \mathfrak{R} \circ (\mathrm{Id} + \widetilde{\psi_L}) \circ \cdots \circ (\mathrm{Id} + \widetilde{\psi_1}) \circ \mathfrak{L}$, where the $\widetilde{\psi_k}$ are in the tensor-train format with probably smaller ranks. Formally, given $g \in \mathcal{CTT}_{\leq \boldsymbol{r}}$, we are looking for $\tilde{g} \in \mathcal{CTT}_{\leq \tilde{\boldsymbol{r}}}$ such that

$$\|g - \tilde{g}\|_{L^2_\mu} \leq \varepsilon\|g\|_{L^2_\mu}. \tag{14}$$

We consider functions g from $\mathbb{R}^d$ to $\mathbb{R}^{d_o}$ with linear lift and retraction maps of the form $\mathfrak{L}(x) = (x,0) \in \mathbb{R}^p$ and $\mathfrak{R}(h) = (h_1,\ldots,h_{d_o}) \in \mathbb{R}^{d_o}$. Both maps $\mathfrak{L}$ and $\mathfrak{R}$ are 1-Lipschitz. We work with locally Lipschitz univariate basis Φ and a compactly supported measure μ, since in this case the basis Φ is Lipschitz continuous (with potentially a large Lipschitz constant). First, note that one has

$$\|g - \tilde{g}\|_{L^2_\mu} \leq \|\mathfrak{R}\|_{\ell^2\to\ell^2} \|(\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) - (\mathrm{Id} + \widetilde{\psi_L}) \circ \cdots \circ (\mathrm{Id} + \widetilde{\psi_1})\|_{L^2_{\mathfrak{L}\sharp\mu}}.$$

Since $\mathfrak{R}$ is a projector, we have $\|\mathfrak{R}\|_{\ell^2\to\ell^2} = 1$. Therefore, it suffices to find a $\tilde{g}$ such that

$$\|(\mathrm{Id}+\psi_L)\circ\cdots\circ(\mathrm{Id}+\psi_1) - (\mathrm{Id}+\widetilde{\psi_L})\circ\cdots\circ(\mathrm{Id}+\widetilde{\psi_1})\|_{L^2_{\mathfrak{L}\sharp\mu}} \leq \varepsilon\|(\mathrm{Id}+\psi_L)\circ\cdots\circ(\mathrm{Id}+\psi_1)\|_{L^2_{\mathfrak{L}\sharp\mu}}.$$

First, we show the following useful lemma.

Lemma 3.16. *Let $f_k, g_k : \mathbb{R}^p \to \mathbb{R}^p$ be Lipschitz functions, with $1 \leq k \leq L$, and $q \geq 1$. Let ρ be a probability measure on $\mathbb{R}^p$ and assume that $f_k \circ \cdots \circ f_1$ and $g_k \circ \cdots \circ g_1$ are in L^q_ρ for every $k = 1, \ldots, L$. Then we have*

$$\|f_L \circ \cdots \circ f_1 - g_L \circ \cdots \circ g_1\|_{L^q_\rho} \leq \varepsilon_L + \sum_{j=1}^{L-1} \varepsilon_j \prod_{k=j+1}^{L} \mathrm{Lip}(g_k), \tag{15}$$

where $\varepsilon_j := \|f_j - g_j\|_{L^q_{\rho_{j-1}}}$ and $\rho_j = [f_j \circ \cdots \circ f_1]_\sharp \rho$, with $\rho_0 = \rho$ by convention.

Proof. We show the result by induction on L. Assume that the result holds for some k. Then,

$$\begin{aligned}
\|f_{k+1} \circ \cdots \circ f_1 - g_{k+1} \circ \cdots \circ g_1\|_{L^q_\rho} &\leq \|f_{k+1} - g_{k+1}\|_{L^q_{\rho_k}} \\
&\quad + \|g_{k+1} \circ f_k \circ \cdots \circ f_1 - g_{k+1} \circ \cdots \circ g_1\|_{L^q_\rho} \\
&\leq \|f_{k+1} - g_{k+1}\|_{L^q_{\rho_k}} \\
&\quad + \mathrm{Lip}(g_{k+1}) \|f_k \circ \cdots \circ f_1 - g_k \circ \cdots \circ g_1\|_{L^q_\rho} \\
&\leq \varepsilon_{k+1} + \mathrm{Lip}(g_{k+1}) \left(\varepsilon_k + \sum_{j=1}^{k-1} \varepsilon_j \prod_{m=j+1}^{k} \mathrm{Lip}(g_m) \right) \\
&= \varepsilon_{k+1} + \sum_{j=1}^{k} \varepsilon_j \prod_{m=j+1}^{k+1} \mathrm{Lip}(g_m),
\end{aligned}$$

which concludes the proof. □

In the case of compositional tensors, the functions f_k and g_k are not globally Lipschitz, but only locally Lipschitz. We then have the corollary in the case where the input space $\Omega \subset \mathbb{R}^p$ is compact.

Corollary 3.17. *Let $f_k, g_k : \Omega_k \to \Omega_{k+1}$ be locally Lipschitz function, with $1 \leq k \leq L$, $q \geq 1$ and $\Omega_{k+1} := f_k(\Omega_k) \cup g_k(\Omega_k)$ where $\Omega_0 := \Omega$. Let ρ be a probability measure on Ω and assume that $f_k \circ \cdots \circ f_1$ and $g_k \circ \cdots \circ g_1$ are in L^q_ρ for every $k = 1, \ldots, L$. Then the functions f_k and g_k are Lipschitz on Ω_k and the inequality (15) holds.*

Based on Corollary 3.17, we design an algorithm that constructs a *compressed TT* $\tilde{g}$ such that (14) is satisfied.

Proposition 3.18. *Let $g = \mathfrak{R} \circ (\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L}$ be a CTT with $\psi_k \in \mathcal{T}_{\boldsymbol{n},\boldsymbol{r}}$ and $M := \|(\mathrm{Id} + \psi_L) \circ \cdots \circ (\mathrm{Id} + \psi_1)\|_{L^2_{\mathfrak{L}_\sharp \mu}}$. By choosing $\widetilde{\psi_k} \in \mathcal{T}_{\boldsymbol{n},\tilde{\boldsymbol{r}}}$ such that*

$$\delta_j := \|\boldsymbol{\psi_j} - \widetilde{\boldsymbol{\psi_j}}\|_F \leq \varepsilon \frac{M}{L \|\Phi\|_{L^2_{\rho_{j-1}}}} \prod_{k=j+1}^{L} \mathrm{Lip}(\widetilde{\psi_k})^{-1},$$

we get

$$\|g - \widetilde{g}\|_{L^2_\mu} \leq \varepsilon \|g\|_{L^2_\mu},$$

where $\widetilde{g} := \mathfrak{R} \circ (\mathrm{Id} + \widetilde{\psi_L}) \circ \cdots \circ (\mathrm{Id} + \widetilde{\psi_1}) \circ \mathfrak{L}$.

Proof. Let $\delta_j := \|\boldsymbol{\psi_j} - \widetilde{\boldsymbol{\psi_j}}\|_F$. By Corollary 3.17, we have,

$$\|(\mathrm{Id}+\psi_L)\circ\cdots\circ(\mathrm{Id}+\psi_1) - (\mathrm{Id}+\widetilde{\psi_L})\circ\cdots\circ(\mathrm{Id}+\widetilde{\psi_1})\|_{L^2_\rho} \leq \sum_{j=1}^{L} \varepsilon_j \prod_{k=j+1}^{L} \mathrm{Lip}(\widetilde{\psi_k})$$

with $\prod_{k=L+1}^{L} \mathrm{Lip}(\widetilde{\psi_k})^{-1} = 1$ by convention. Using the Cauchy-Schwarz inequality, we immediately get

$$\varepsilon_j \leq \delta_j \|\Phi\|_{L^2_{\rho_{j-1}}}.$$

By taking

$$\delta_j := \|\boldsymbol{\psi_j} - \widetilde{\boldsymbol{\psi_j}}\|_F \leq \varepsilon \frac{M}{L\|\Phi\|_{L^2_{\rho_{j-1}}}} \prod_{k=j+1}^{L} \mathrm{Lip}(\widetilde{\psi_k})^{-1},$$

we get the result. □

Note that $\|\Phi\|^{-1}_{L^2_{\rho_{j-1}}}$ and $\|(\mathrm{Id}+\psi_L)\circ\cdots\circ(\mathrm{Id}+\psi_1)\|_{L^2_{\mathfrak{L}_\sharp\mu}}$ can be estimated by Monte-Carlo sampling and that $\mathrm{Lip}(\widetilde{\psi_k})$ can be estimated using the gradient of $\widetilde{\psi_k}$, which we have access to. Starting from $j = L$, one can iteratively construct $\widetilde{\psi_j}$ using TT-rounding [44] for instance.

4 Optimization

In this section we aim to design a learning algorithm for the compositional tensor format. We propose to study two differents approaches: a first algorithm based on optimal control called the *method of successive approximation (MSA)*, and the second is (a modification of) *natural gradient descent*, which has recently gained popularity in training of DNNs and earlier in quantum physics.

4.1 An optimal control perspective

We focus on the regression problem

$$\min_{\Psi\in\mathcal{CT}^L} \|\eta - \Psi\|^2_{L^2_\mu(\mathcal{X})}, \tag{16}$$

where μ is a finite measure, $\mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ is the Cartesian product of closed intervals $\mathcal{X}_\nu$, and $\eta \in L^2_\mu(\mathcal{X})$ is a target function. We assume in the following that the CT Ψ is defined by functional tensors $\psi_1, \ldots, \psi_L$ and associated coefficient tensors $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_L$ together with an appropriate lift $\mathfrak{L}$ and retraction $\mathfrak{R}$ in the sense of Definition 3.1. Letting $\hat{X}_0 \sim \mu$, the above is equivalent to

$$\min_{\Psi\in\mathcal{CT}^L} \mathbb{E}_{\hat{X}_0\sim\mu} \|\eta(\hat{X}_0) - \Psi(\hat{X}_0)\|_2^2. \tag{17}$$

Since the expectation is in general not computable, it is in practice replaced by an unbiased empirical estimation via i.i.d. realizations of the random variable $\hat{X}_0$. Still, since the model class is highly nonlinear as a composition of nonlinear functions, it makes sense to introduce regularization. In the following we describe several related regularization approaches, their connection, and interpretations, as well as strengths and weaknesses.

Stochastic optimal control A first important observation is that a straightforward approach for regularization leads to a discrete-time stochastic optimal control problem. Consider for $X_0 = \mathfrak{L}(\hat{X}_0)$ the problem

$$\begin{cases} \min\limits_{\psi_1,\dots,\psi_L \in \mathcal{M}_{\boldsymbol{r}}} J(\psi_1,\dots,\psi_L) := \mathbb{E}_{\hat{X}_0\sim\mu}\left[\sum\limits_{k=1}^{L} \mathcal{L}_k(X_{k-1},\psi_k) + \|\eta(\hat{X}_0) - \mathfrak{R}(X_L)\|_2^2\right] \\ X_{k+1} = X_k + \psi_{k+1}(X_k), \quad k = 0,\dots,L-1, \end{cases} \tag{18}$$

for positive functions $\mathcal{L}_k \colon \mathbb{R}^p \times \mathcal{T}_{\boldsymbol{r}} \to [0,\infty)$. Since $\mathfrak{R}(X_L) = \Psi(X_0)$ by construction, the only difference between (17) and (18) lies in the introduction of the sum in the objective functional. From the point of view of the regression problem, this can naturally be understood as a regularization, in which each layer adds a penalty defined by $\mathcal{L}_k$ to the regression loss. (18) has the form of a stochastic optimal control (SOC) problem with deterministic dynamics but random initial condition X_0. In this setting, the regularization function $\mathcal{L}_k$ is often called the *running costs*, whereas the actual regression loss $\|\eta(\hat{X}_0) - \mathfrak{R}(X_L)\|_2^2$ is called the *terminal costs* of the system. From the point of view of SOC, the TTs $\psi_1,\dots,\psi_L$ are collectively called the control of the system. Hence, when we talk about an optimal control in the following, we mean a collection $\psi_1,\dots,\psi_L$ solving (18). First-order conditions for such a control in the general setting are given by the Pontryagin maximum principle.

Lemma 4.1 (Pontryagin maximum principle (PMP))**.** *Consider for some $p \in \mathbb{N}$ the optimal control problem*

$$\begin{cases} \min\limits_{u_1,\dots,u_L} \sum\limits_{k=1}^{L} \mathcal{L}_k(X_{k-1},u_k) + g(X_L) \\ X_k = f(X_{k-1},u_k) \end{cases}, \tag{19}$$

where $X_0 \sim \rho$, the control values u_k are in a linear space $\mathcal{U}$, $f \colon \mathbb{R}^p \times \mathcal{U} \to \mathbb{R}^p$ denotes the dynamics, $\mathcal{L}_k \colon \mathbb{R}^p \times \mathcal{U} \to [0,\infty)$ is the running costs and $g \colon \mathbb{R}^p \to [0,\infty)$ is the terminal costs. For any state-control pair $((X_k)_{k=0}^L, (u_k)_{k=1}^L)$ define the costates $(\lambda_k)_{k=0}^L$ via

$$\lambda_k = \partial_{X_k}\mathcal{L}_k(X_k,u_{k+1}) + \partial_{X_k} f(X_k,u_{k+1}) \cdot \lambda_{k+1}, \tag{20}$$

$$\lambda_L = \nabla g(X_L). \tag{21}$$

Then, an optimal control u_k^* and the corresponding state and costate values X_k^*, λ_k^* satisfy for each $k \in \{1,\dots,L\}$ the following first order necessary condition

$$\mathbb{E}[H_k(X_{k-1}^*,u_k^*,\lambda_k^*)] \le \mathbb{E}[H_k(X_{k-1},u_k,\lambda_k)] \tag{22}$$

for all state-control costate tuples $((X_k)_{k=0}^L, (u_k)_{k=1}^L, (\lambda_k)_{k=0}^L)$, where

$$H_k(X_{k-1},u_k,\lambda_k) := \mathcal{L}_k(X_{k-1},u_k) + \langle \lambda_k, f(X_{k-1},u_k)\rangle \tag{23}$$

is the Hamiltonian.

The proof of Lemma 4.1 immediately follows using Lagrange multipliers. Note that the costates λ_k are random variables as $X_0 \sim \rho$.

The Pontryagin maximum principle leads to the natural algorithm called method of successive approximation (MSA), which basically, given an initial control, tries to optimize iteratively the Hamiltonian by solving the state and costate equations, and then minimizes the Hamiltonian.

Algorithm 2: Method of successive approximation (MSA)

Require: $u_1^{(0)}, \dots, u_L^{(0)} \in \mathcal{U}$, $X_0 \sim \rho$

i $= 0$

while cond *not satisfied* **do**

  Let $X_k^{(\mathrm{i})} = f(X_{k-1}^{(\mathrm{i})}, u_k^{(\mathrm{i})}), \quad k = 1, \dots, L\,, \quad X_0^{(\mathrm{i})} = X_0$

  Let $\lambda_k^{(\mathrm{i})} = \partial_{X_k^{(\mathrm{i})}} \mathcal{L}(X_k^{(\mathrm{i})}, u_{k+1}^{(\mathrm{i})}) + \partial_{X_k^{(\mathrm{i})}} f(X_k^{(\mathrm{i})}, u_{k+1}^{(\mathrm{i})}) \cdot \lambda_{k+1}^{(\mathrm{i})}, \quad k = L-1, \dots, 0,$
  $\lambda_L^{(\mathrm{i})} = \nabla g(X_L^{(\mathrm{i})}),$

  Compute $u_k^{(\mathrm{i}+1)} \in \arg\min_{u \in \mathcal{U}} \mathbb{E}[H(X_{k-1}^{(\mathrm{i})}, u, \lambda_k^{(\mathrm{i})})], \quad k = 1, \dots, L$

  Update i $=$ i $+ 1$

end

4.1.1 On the choice of the regularization

The choice of $\mathcal{L}_k$ leads to different solutions with different interpretations.

Frobenius norm regularization. A naive choice is to define $\mathcal{L}_k$ as the squared Frobenius norm of $\boldsymbol{\psi}_k$, i.e. $\mathcal{L}_k(X_{k-1}, \psi_k) = \frac{R}{2}\|\boldsymbol{\psi}_k\|_F^2$, where $R > 0$ is some regularization parameter. This choice seems to make sense as an extension of a Tikhonov–like regularization to multiple layers. However, letting $\rho = \mathfrak{L}_{\#}\mu$ and choosing L^2_ρ-orthonormal basis functions, the space of coefficient tensors is isometric to L^2_ρ and in particular we have $\|\boldsymbol{\psi}_k\|_F^2 = \|\psi_k\|^2_{L^2_\rho}$. This is not desirable since in layer k, ψ_k no longer gets points from the distribution μ, but instead from the push-forward measure $\rho_{k-1} = [(\mathrm{Id} + \psi_{k-1}) \circ \cdots \circ (\mathrm{Id} + \psi_1)]_\sharp \rho \sim X_{k-1}$ (with $\rho_0 = \rho$ by convention). Therefore, the regularization is not natural in the sense that it does not take into account the dynamics, and forces ψ_k to be small in the L^2_ρ–sense.

Natural regularization. The above issue can be rectified by choosing $\mathcal{L}_k(X_{k-1}, \psi_k) = \frac{R}{2}\|\psi_k(X_{k-1})\|^2$, $R > 0$, and consequently, $\mathbb{E}[\|\psi_k(X_{k-1})\|^2] = \|\psi_k\|^2_{L^2_{\rho_{k-1}}}$. The regularization then forces ψ_k to be small in the push-forward sense, by taking into account the dynamics.

Connection of MSA to gradient descent. The application of the MSA algorithm to (18) with running costs chosen via a Frobenius norm or natural regularization has clear connections to gradient descent schemes. The connection between standard MSA and gradient descent in a neural network setting has already been discussed e.g. in [32]. There, it is shown that the MSA algorithm becomes standard gradient descent if one replaces the minimization of the Hamiltonian ("hard" update) with a relaxation ("soft" update).

When working on the full tensor space, which is linear, the connection is even stronger. Consider the running costs defined by $\mathcal{L}_k(X_{k-1}, \psi_k) = \frac{R}{2}\|\boldsymbol{\psi}_k\|_F^2$. The Hamiltonian in this case is given by

$$\begin{aligned} H_k(X_{k-1}, \psi_k, \lambda_k) &= \frac{R}{2}\|\boldsymbol{\psi}_k\|_F^2 + \langle \lambda_k, X_{k-1} + \psi_k(X_{k-1}) \rangle \\ &= \frac{R}{2}\|\boldsymbol{\psi}_k\|_F^2 + \langle \lambda_k, X_{k-1} + (\langle \boldsymbol{\psi}_k(j, \cdot), \Phi(X_{k-1}) \rangle_F)_{j=1}^p \rangle, \\ \partial_{\boldsymbol{\psi}} H_k(X_{k-1}, \psi_k, \lambda_k) &= R\boldsymbol{\psi}_k + \lambda_k \Phi(X_{k-1}), \end{aligned}$$

where $\lambda_k \Phi(X_{k-1})$ is shorthand for a random variable with realizations in $\mathbb{R}^{p\times n\times\ldots\times n}$ defined by

$$\lambda_k \Phi(X_{k-1})(j,\cdot) = \lambda_{k,j}\Phi(X_{k-1}).$$

Hence, the MSA yields an explicit update formula for the control in every step, which is

$$\begin{aligned}\boldsymbol{\psi}_k^{(\mathrm{i}+1)} &= -\frac{1}{R}\,\mathbb{E}[\lambda_k^{(\mathrm{i})}\Phi(X_{k-1}^{(\mathrm{i})})] \\ &= \boldsymbol{\psi}_k^{(\mathrm{i})} - \frac{1}{R}(R\boldsymbol{\psi}_k^{(\mathrm{i})} + \mathbb{E}[\lambda_k^{(\mathrm{i})}\Phi(X_{k-1}^{(\mathrm{i})})]) \\ &= \boldsymbol{\psi}_k^{(\mathrm{i})} - \frac{1}{R}\,\mathbb{E}[\partial_{\boldsymbol{\psi}} H_k(X_{k-1}^{(\mathrm{i})}, \psi_k^{(\mathrm{i})}, \lambda_k^{(\mathrm{i})})].\end{aligned}$$

Hence, in our setting, the hard update corresponds to a gradient descent update with step size $1/R$ in the parameter space.

In general, when working in the functional setting, suppose that we want to solve the minimization problem

$$\begin{cases} \min\limits_{\psi_1,\ldots,\psi_L} J(\psi_1,\ldots,\psi_L) := \mathbb{E}_{X_0\sim\rho}\left[\sum\limits_{k=1}^{L}\frac{R}{2}\|\psi_k(X_{k-1})\|_2^2 + g(X_L)\right] \\ X_{k+1} = X_k + \psi_{k+1}(X_k), \quad k = 0,\ldots,L-1, \\ X_0 \sim \rho \end{cases}$$

for functions $\psi_k \in L^2_{\rho_{k-1}}$. The MSA algorithm aims to minimize the quantity $\mathbb{E}[H_k(X_{k-1}^{(q)}, \psi_k, \lambda_k^{(q)})]$ with respect to ψ_k, where

$$H_k(X_{k-1}^{(q)}, \psi_k, \lambda_k^{(q)}) := \frac{R}{2}\|\psi_k(X_{k-1}^{(q)})\|_2^2 + (\lambda_k^{(q)})^\top(X_{k-1}^{(q)} + \psi_k(X_{k-1}^{(q)}))$$

and $(X_k^{(q)})_k$ and $(\lambda_k^{(q)})_k$ are the states and costates obtained using the controls $(\psi_k^{(q)})_k$, respectively. The Riesz representative of the Fréchet derivative of $\mathbb{E}[H_k(X_{k-1}^{(q)}, \psi_k, \lambda_k^{(q)}]$ with respect to ψ_k is denoted $\nabla_{\psi_k}\mathbb{E}[H_k(X_{k-1}^{(q)}, \psi_k, \lambda_k^{(q)}]$, and is given by

$$\nabla_{\psi_k}\mathbb{E}[H_k(X_{k-1}^{(q)}, \psi_k, \lambda_k^{(q)})] = R\psi_k + \lambda_k^{(q)},$$

where in this expression $\lambda_k^{(q)}$ has to be understood as a function. The optimality conditions give

$$\begin{aligned}\psi_k^{(q+1)} &= -\frac{1}{R}\lambda_k^{(q)} \\ &= \psi_k^{(q)} - \frac{1}{R}(R\psi_k^{(q)} + \lambda_k^{(q)}) \\ &= \psi_k^{(q)} - \frac{1}{R}\nabla_{\psi_k}\mathbb{E}[H_k(X_{k-1}^{(q)}, \psi_k^{(q)}, \lambda_k^{(q)})],\end{aligned}$$

which is a (functional) gradient descent with step size $\frac{1}{R}$.

Augmented Hamiltonian. It has been shown [29, 32] that classical MSA cannot be guaranteed to update the control in the optimal descent direction, and even diverges in general. To overcome this issue, we need to modify the Hamiltonian in such a way that the MSA converges. As proposed in [29, 32], we introduce the *augmented Hamiltonian*

$$\tilde{H}_\gamma(k,x,\lambda,\psi,\tilde{\psi}) := H_k(x,\lambda,\psi) + \frac{1}{2}\gamma\|\psi(x) - \tilde{\psi}(x)\|_2^2, \quad \gamma \geq 0. \tag{24}$$

The minimization step of the Hamiltonian in the classical MSA is then replaced by a minimization step of the augmented Hamiltonian. The difference between H and $\tilde{H}$ is that the augmented Hamiltonian now forces ψ staying in a neighbourhood of $\tilde{\psi}$ with respect to some metric. For example, if $x = X_{k-1}$ and $\tilde{\psi} = \psi_k$, then it forces the new dynamic induced by ψ to stay in a neighbourhood of the dynamic induced by ψ_k.

4.1.2 Application in the case of tensors

In our setting, ψ is a tensor and we derive the optimality condition to minimize the *augmented Hamiltonian* (24). Recall that any function $\psi \in \mathcal{V}_{\mathbf{n}}$ can be represented as

$$\psi : x \mapsto \langle \boldsymbol{\psi}, \Phi(x) \rangle_F = \left(\sum_{\alpha} \boldsymbol{\psi}(j, \alpha_1, \ldots, \alpha_p) \phi_{\alpha_1}(x_1) \ldots \phi_{\alpha_p}(x_p) \right)_{j=1}^{p},$$

so that ψ is a *vector-valued function* parametrized by a tensor $\boldsymbol{\psi} \in \mathbb{R}^{p \times n \times \cdots \times n}$. For notational convenience, we write $\tilde{H}_\gamma(k, x, \lambda, \boldsymbol{\psi}, \widetilde{\boldsymbol{\psi}}) = \tilde{H}_\gamma(k, x, \lambda, \psi, \tilde{\psi})$. Taking the gradient of (24) with respect to $\boldsymbol{\psi}$ leads to the expression

$$\begin{aligned} \nabla_{\boldsymbol{\psi}} \tilde{H}_\gamma(k, x, \lambda, \boldsymbol{\psi}, \widetilde{\boldsymbol{\psi}}) &= \nabla_{\boldsymbol{\psi}} \left(\frac{R}{2} \|\boldsymbol{\psi} \cdot \Phi(x)\|^2 + \lambda^\top (x + \boldsymbol{\psi} \cdot \Phi(x)) + \frac{1}{2} \gamma \|\boldsymbol{\psi} \cdot \Phi(x) - \widetilde{\boldsymbol{\psi}} \cdot \Phi(x)\|^2 \right) \\ &= RG(x)\boldsymbol{\psi} + \sum_{i=1}^{d} \lambda_i (e_i \otimes \Phi(x)) + \gamma G(x)(\boldsymbol{\psi} - \widetilde{\boldsymbol{\psi}}), \end{aligned}$$

where

$$G : \begin{cases} \mathbb{R}^d \longrightarrow \mathcal{L}(\mathbb{R}^p \otimes \mathbb{R}^n \otimes \cdots \otimes \mathbb{R}^n) \\ x \longmapsto I_d \otimes (\phi^{(1)}(x_1)\phi^{(1)}(x_1)^\top) \otimes \cdots \otimes (\phi^{(d)}(x_d)\phi^{(d)}(x_d)^\top) \end{cases}$$

gives a symmetric and positive-definite operator from tensor space $\mathbb{R}^{p \times n \times \ldots \times n}$ to itself at each point $x \in \mathbb{R}^d$.

In our case, we use B samples $(x_{j,k}^{(i)})_{j=1}^{b}$ for estimating the expectations leading to the minimization problem

$$\boldsymbol{\psi}_k^{(\mathrm{i}+1)} \in \arg\min_{\boldsymbol{\psi}} \frac{1}{B} \sum_{j=1}^{B} \tilde{H}_\gamma(k-1, x_{j,k-1}^{(\mathrm{i})}, \lambda_{j,k}^{(\mathrm{i})}, \boldsymbol{\psi}, \boldsymbol{\psi}_k^{(\mathrm{i})}).$$

Setting the gradient of the functional with respect to $\boldsymbol{\psi}$ to zero yields the linear tensor equation

$$\left(\frac{1}{B} \sum_{j=1}^{B} G(x_{j,k-1}^{(i)}) \right) ((R+\gamma)\boldsymbol{\psi}_k^{(i+1)} - \gamma \boldsymbol{\psi}_k^{(i)}) = -\frac{1}{B} \sum_{j=1}^{B} \sum_{n=1}^{d} \lambda_{j,k,n}^{(i)} (e_n \otimes \Phi(x_{j,k-1}^{(i)})).$$

Such an equation can be solved using ALS [25] or AMEn [14]. Note that the operator $\frac{1}{B} \sum_{j=1}^{B} G(x_{j,k-1}^{(i)})$ and the right-hand side have representations in canonical tensor format with canonical tensor rank Bd. In order to have the updates $\boldsymbol{\psi}_k^{(\mathrm{i}+1)}$ in the Tensor-Train format, one can perform a TT-rounding, or a truncation to some given ranks $\boldsymbol{r}$ [44].

4.2 Natural gradient descent

Gradient descent algorithm is a prominent optimization method, especially for neural networks, and have shown good convergence properties in many applications. It has been proven that its stochastic

version (this means, with a stochastic estimation of the true gradient) converges to critical points, even for non-convex objective functions [11]. Furthermore, it can escape and avoid saddle points [27, 28]. However, vanilla gradient descent tends to converge slowly, and often does not achieve a high accuracy. Following [39], we propose to employ the *natural gradient descent* [3] for this specific architecture. We first provide a formal derivation.

Let $\mathcal{M} \subset L^2_\mu := \mathcal{H}$ be a nonlinear model class. Consider the minimization problem

$$\inf_{u\in\mathcal{M}} \mathcal{L}(u), \qquad \mathcal{L} : L^2_\mu \to \mathbb{R}. \tag{25}$$

We aim to solve this problem using an iterative scheme of the form,

$$u_{k+1} = u_k + \alpha_k d_k, \qquad d_k = -\nabla\mathcal{L}(u_k). \tag{26}$$

Remark 4.2. *In general, for an arbitrary Hilbert space $\mathcal{H}$, d_k is defined by $d_k = -Q_{u_k}^{-1}(\nabla\mathcal{L}(u_k))$, where $\nabla\mathcal{L} \in \mathcal{H}$ is the gradient of $\mathcal{L}$ and Q_{u_k} is an invertible, symmetric and positive definite linear map. For example,*

- *by choosing $d_k = -\nabla\mathcal{L}(u_k)$ we recover the* Hilbert space gradient descent*,*
- *by choosing $d_k = -\nabla^2\mathcal{L}(u_k)^{-1}(D\mathcal{L}(u_k))$, with $\nabla^2\mathcal{L}(u_k) : \mathcal{H} \to \mathcal{H}$ the Hessian of $\mathcal{L}$ at u_k, we recover the* Hilbert space Newton method*.*

Henceforth, we consider a *parametric class of functions*, $\mathcal{M} = \{u_\theta = P(\theta) : \theta \in \Theta\}$ where $P : \Theta \to \mathcal{M} \subseteq \mathcal{H}$ is the *parametrization* and is assumed to be C^1, and $\Theta \subseteq \mathbb{R}^p$ is called the *parameter space*. Since in practice, performing a descent in the functional space is infeasible, we aim to design an iterative scheme in the parameter space that mimics the descent (26). Namely,

$$\theta_{k+1} = \theta_k + \alpha_k w_k, \tag{27}$$

such that $u_{\theta_{k+1}} \approx u_{k+1}$. Applying the Taylor theorem to $u_{\theta_{k+1}} = P(\theta_k + \alpha_k w_k)$ gives

$$u_{\theta_{k+1}} = u_{\theta_k} + \alpha_k DP(\theta_k)w_k + o(\alpha_k\|w_k\|).$$

Therefore, we would like to find w_k such that we have $DP(\theta_k)w_k \approx d_k$. The quantity $DP(\theta)w$ lives in the *tangent space* of $\mathcal{M}$ at u_θ, defined as

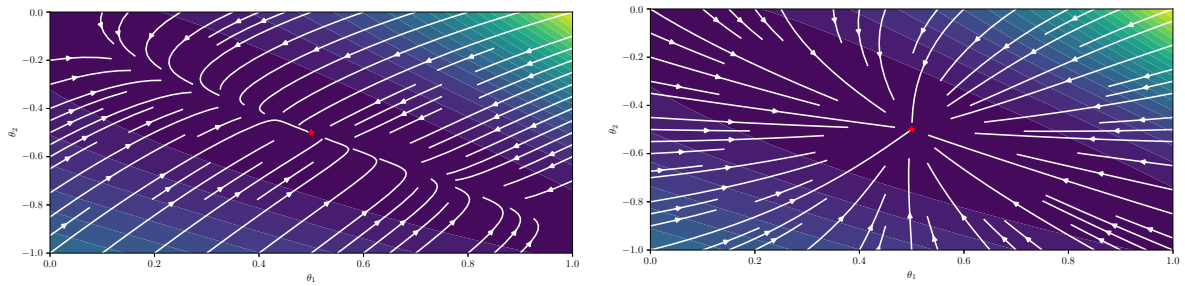
$$T_{u_\theta}\mathcal{M} := \operatorname{span}\{\partial_{\theta_1}u_\theta, \dots, \partial_{\theta_p}u_\theta\} = \operatorname{span}(DP(\theta)) \subset L^2_\mu.$$

This tangent space is equipped with the metric $\langle\cdot,\cdot\rangle_{L^2_\mu}$.

Remark 4.3. *In general, the tangent space $T_{u_\theta}\mathcal{M}$ can be equipped with a metric $\langle Q_{u_k}\cdot,\cdot\rangle_{\mathcal{H}}$ depending on u_k. The Hilbert space Newton method corresponds to $Q_{u_k} = \nabla^2\mathcal{L}(u_k)$.*

It follows that the w_k that "aligns the best with the functional descent" is given by,

$$w_k \in \arg\min_{w\in\Theta} \frac{1}{2}\|DP(\theta_k)w - d_k\|^2_{L^2_\mu}. \tag{28}$$



(a) Euclidean gradient $\nabla_\theta L(\theta)$ (b) Natural gradient $\widetilde{\nabla}_\theta L(\theta) := G(\theta)^\dagger \nabla_\theta L(\theta)$

Figure 2: Visualization of the gradient fields $\nabla_\theta L(\theta)$ and $\widetilde{\nabla}_\theta L(\theta)$ for the model $u_\theta(x, y) = \exp(\theta_1 x + \cos(y - \theta_2))$ and the loss function $\mathcal{L}(u) = \frac{1}{2}\|u - u^*\|^2_{L^2([0,1]^2)}$.

Remark 4.4 (Gauss-Newton iterate)**.** *The update direction* (28) *coincides with one Gauss-Newton step for the minimization of* $L(\theta) := \frac{1}{2}\|P(\theta) - u\|^2_{\mathcal{H}}$*, with* $\mathcal{H} = L^2_\mu$*. Indeed, a Gauss-Newton iteration is given by* $\theta_{k+1} = \theta_k + w_k$*, with* w_k *minimizing over* $w \in \Theta$ *the function*

$$w \mapsto \frac{1}{2}\|P(\theta_k) + DP(\theta_k)w - u\|^2_{\mathcal{H}} = \frac{1}{2}\|DP(\theta_k)w - d_k\|^2_{\mathcal{H}}$$

with $d_k = u - P(\theta_k) = -\nabla L(\theta_k) \in \mathcal{H}$*.*

Computing the normal equation of (28) gives the update direction

$$w_k = -G(\theta_k)^\dagger \nabla_\theta L(\theta_k), \tag{29}$$

where $L = \mathcal{L} \circ P : \Theta \to \mathbb{R}$ and $G(\theta_k) \in \mathbb{R}^{p\times p}$ is the *Gramian matrix* defined by,

$$G(\theta)_{ij} := \langle \partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta \rangle_{L^2_\mu}, \tag{30}$$

and by $G(\theta)^\dagger$ we denote a *pseudo-inverse* of $G(\theta)$, that means a matrix satisfying $GG^\dagger G = G$.

As illustrated in Figure 2 for a particular 2-dimensional model class $\mathcal{M}$, the flow obtained by the natural gradient naturally points towards the optimal point, and the streamlines are more straight.

An interesting property of the natural gradient is that it is reparametrization invariant [53]. That means, if we consider two parametrizations $P(\theta)$ and $P(f^{-1}(\xi))$, where f is a diffeomorphism, the update direction in $\mathcal{H}$ is the same. This implies that an update in the parameters for both parametrizations by an *infinitesimal amount*, this would give the same result on the model. However, it is important to point out that updating the parameters by a *finite amount* will, in general, give different locations on the model: the *natural gradient descent is dependent on the choice of parametrization*.

We would also point out that it is believed that natural gradient descent can avoid saddle points because it updates the parameters in the optimal direction in $\mathcal{M}$ [26]. However, unlike the Hessian, the Gramian has only *non-negative* eigenvalues, therefore it does not exploit explicitly negative curvature, which helps to escape saddle points. Consequently, it does not actively push away from saddle points. In [37], the authors argue that around a stationary solution, the Gramian matrix is rank-deficient, therefore the natural gradient descent may converge to instationary points and fail to escape. Despite this undesired behavior, natural gradient descent has shown very good performance, particularly for PINNs [38, 41]. However, when employed with neural networks, the computation of the natural gradient is very

resource-intensive, therefore its usage is very limited. Some methods have been developed to compute an approximation of this quantity, namely K-FAC [34, 4, 16] and techniques using randomized linear algebra [7].

Implementation for CTTs. In our case, if we want to exploit the tensor algebra, we cannot apply the natural gradient descent directly to the model class $\mathcal{CT}(\Phi)$ (and also $\mathcal{CTT}_{\leq \boldsymbol{r}}(\Phi)$), since it is not clear whether the sum of two compositional tensors yields a compositional tensor (even with probably a different number of layers). Instead, we propose to apply it layerwise, i.e. on each tensor ψ_ℓ on the ℓ-th layer, since in this case we preserve the tensor structure (and also the low-rank format if used). The function u_θ is an element of $\mathcal{CT}(\Phi)$, where $\theta = (\boldsymbol{\psi_1}, \dots, \boldsymbol{\psi_L})$ represents the tensors of the layers. By applying the chain rule, we immediately get

$$\frac{\partial u_\theta}{\partial \boldsymbol{\psi}_\ell}(x) = \underbrace{\frac{\partial u_\theta}{\partial u_\ell}(x)}_{\in \mathbb{R}^{d_o \times d}} \otimes \Phi^{\otimes d}(u_{\ell-1}(x)) \in \mathbb{R}^{d_o \times d} \otimes (\mathbb{R}^m)^{\otimes d}, \tag{31}$$

where $u_\ell(x) = (\mathrm{Id} + \psi_\ell) \circ \cdots \circ (\mathrm{Id} + \psi_1) \circ \mathfrak{L}(x)$. The Gramian of the ℓ-th layer is then given by

$$G_\ell(\theta) = \mathbb{E}_x \left[\left(\frac{\partial u_\theta}{\partial u_\ell}(x)^\top \frac{\partial u_\theta}{\partial u_\ell}(x) \right) \otimes \left((\Phi \Phi^\top)^{\otimes d}(u_{\ell-1}(x)) \right) \right] \in \mathbb{R}^{d \times d} \otimes (\mathbb{R}^{m \times m})^{\otimes d}. \tag{32}$$

For computing the natural gradient layerwise we thus have to solve the normal equation

$$G_\ell(\theta) \cdot d = \mathbb{E}_x \left[\left(\frac{\partial u_\theta}{\partial u_\ell}(x) \right)^\top \nabla \mathcal{L}(u_\theta)(x) \right]. \tag{33}$$

In our case, we can approximate the L^2_μ-inner product by a Monte-Carlo approximation using B samples and therefore consider an empirical version of (33). Indeed, a Monte-Carlo approximation of $G_\ell(\theta)$ is given by

$$\begin{aligned} \widehat{G_\ell(\theta)} = \frac{1}{B} \sum_{i=1}^{B} & \underbrace{\left(\frac{\partial u_\theta}{\partial u_\ell}(x^i)^\top \frac{\partial u_\theta}{\partial u_\ell}(x^i) \right)}_{C_{\ell,0}(x^i)} \otimes \underbrace{\Phi(u_{\ell-1}(x^i)_1)\Phi(u_{\ell-1}(x^i)_1)^\top}_{C_{\ell,1}(x^i)} \\ & \otimes \cdots \otimes \underbrace{\Phi(u_{\ell-1}(x^i)_d)\Phi(u_{\ell-1}(x^i)_d)^\top}_{C_{\ell,d}(x^i)} \end{aligned} \tag{34}$$

In general, if $u_\theta \in \mathcal{CT}(\Phi)$ and no low-rank format is assumed, the time complexity of solving (33) is in $\mathcal{O}(\min(Bd_o m^{2d}, B^2 d_o^2 m^d))$, where B is the number of samples used for estimating the inner product, d_o is the output dimension of the model, d is the dimension of the lifted space, and m is the number of univariate basis elements of Φ. Moreover, the storage complexity is in $\mathcal{O}(m^d)$ for each layer, which is infeasible in high dimensions. To circumvent this problem, one can assume low-rank format for the layers $\boldsymbol{\psi_\ell}$, e.g. the tensor-train format. For solving (33) with a low-rank tensor format, one can use the *alternating linear scheme (ALS)* [25]. However, the Gram matrix associated to the layer ℓ may have a bad condition number, so that ALS without preconditioning may show a slow convergence and yield a highly suboptimal low-rank approximation of the update direction. We should also note that in this case, the update (27) may leave the manifold, therefore it is required to use a *retraction* $R_U : T_U\mathcal{M} \to \mathcal{M}$. The update for each layer $\boldsymbol{\psi_\ell}$ is therefore given by

$$\boldsymbol{\psi}_\ell^{(k+1)} = R_{\boldsymbol{\psi}_\ell^{(k)}}(-\alpha^{(k)} \boldsymbol{w}_\ell^{(k)}), \tag{35}$$

where the iteration number k is written at superscript for ease of reading.

Algorithm 3: Layerwise natural gradient for CTTs

Require: Samples $(x^i)_{i=1}^B$, initial tensors $(\boldsymbol{\psi}_\ell)_{\ell=1}^L$

k $\leftarrow 0$
while *not converged* **do**
  for $\ell \leftarrow 1$ **to** L **do**
    Assemble the Gramian $\widehat{G_\ell(\theta)}$ (34)
    Compute the gradient $\nabla_{\boldsymbol{\psi}_\ell}\mathcal{L}(u_\theta)(x^i)$
    Solve the normal equation (33) and store $\boldsymbol{w}_\ell^{(k)}$
  end
  Compute the learning rate $\alpha^{(k)}$
  for $\ell \leftarrow 1$ **to** L **do**
    if *low-rank* **then**
      Update $\boldsymbol{\psi}_\ell^{(k+1)} \leftarrow R_{\boldsymbol{\psi}_\ell^{(k)}}(-\alpha^{(k)}\boldsymbol{w}_\ell^{(k)})$
    end
    else
      Update $\boldsymbol{\psi}_\ell^{(k+1)} \leftarrow \boldsymbol{\psi}_\ell^{(k)} - \alpha^{(k)}\boldsymbol{w}_\ell^{(k)}$
    end
  end
  k $\leftarrow$ k $+ 1$
end

The Gram matrices G_ℓ are typically highly ill-conditioned, which poses significant challenges for solving the corresponding normal equations (33). A well-established approach to mitigate this issue is to approximate G_ℓ with a low-rank surrogate. In particular, the randomized Nyström method [42, 54, 17, 35] achieves this by projecting G_ℓ onto a randomly generated, low-dimensional subspace. In the context of this work, the Gram matrix G_ℓ is a linear operator acting on tensor spaces $G_\ell : \mathbb{R}^{d\times n\times\cdots\times n} \to \mathbb{R}^{d\times n\times\cdots\times n}$. To construct a suitable low-rank approximation, we introduce a tensor-structured sketch $S \in \mathbb{R}^{s\times d\times n\cdots\times n}$ [48, 52], where $s \in \mathbb{N}$ is the sketching size. The projections $G_\ell S_j \in \mathbb{R}^{d\times n\cdots\times n}$ are computed for $1 \le j \le s$, and we seek a solution in the span of these random projections of G_ℓ. We choose the sketch S to be of the form

$$S_j = \frac{1}{\sqrt{s}} s_{j,0} \otimes \cdots \otimes s_{j,d}, \quad s_{j,p} \sim \mathcal{N}(0, I).$$

The contraction of the Monte-Carlo approximation of G_ℓ given in (34) and S_j can be efficiently computed

$$\widehat{G_\ell} S_j = \frac{1}{B}\sum_{i=1}^{B} C_{\ell,0}(x^i)s_{j,0} \otimes C_{\ell,1}(x^i)s_{j,1} \otimes \cdots \otimes C_{\ell,d}(x^i)s_{j,d}.$$

Moreover, it can be computed without forming explicitly the Jacobian by using the *Jacobian-vector product (JVP)*, see remark in the subsequent numerics section.

The key advantage of this Gaussian sketching approach is that, with high probability, the span of the sketch captures the dominant eigenspace of G_ℓ. Consequently, the low-rank approximation retains the principal components corresponding to the largest eigenvalues while attenuating the directions associated with small eigenvalues. The resulting system behaves as if a spectral filter is applied to G_ℓ, that is, the solution lies in the subspace spanned by the most significant eigenvectors of G_ℓ.

5 Numerical results

In this section, we would like to observe the performance of the *natural gradient descent (NGD)* and MSA algotihms presented in previous section, compared to other (higher-order) optimization methods such as Adam, L-BFGS and BFGS. All the experiments are written in the JAX python framework [8] and can be found on the GitHub repository `https://github.com/chmda/ctt`. For assembling the normal equation (33), the Jacobians and the right-hand side are computed using `jax.vjp` instead of simply calling `jax.jacfwd`, speeding up the computation.

5.1 Layers with full tensors (no low-rank formats)

We first study the behaviour of the algorithms where it is not assumed that the different layers ψ_ℓ are in a low-rank format, i.e., we consider full tensors. The basis functions are chosen to be $\Phi = \{1, \mathrm{Id}\}$. For the considered approximation and recovery problems, the *training dataset* $\mathcal{D}_{\text{train}}$ consists of $100,000$ samples drawn from the measure μ (which are specified for each problem), and the *validation dataset* $\mathcal{D}_{\text{val}}$ consists of $2,000$ independent samples from μ. The loss function considered is

$$\mathcal{L}(u) := \frac{1}{2}\|u - u^*\|^2_{L^2_\mu(\Omega)}, \quad \mathcal{L} : L^2_\mu(\Omega) \to \mathbb{R}^+,$$

where μ is the uniform measure on $\mathcal{X}$. This loss function is however replaced by its empirical version,

$$\widehat{\mathcal{L}(u)} := \frac{1}{2}\sum_{x\in\mathcal{D}} |u(x) - u^*(x)|^2,$$

where $\mathcal{D}$ is an ensemble of random points, e.g. a mini-batch or the whole training dataset. The performance of the algorithms is measured using the $L^2_\mu(\mathcal{X})$-relative error estimated by

$$\hat{\varepsilon}^2 := \frac{\sum_{x\in\mathcal{D}_{\text{val}}} |\widetilde{u}(x) - u^*(x)|^2}{\sum_{x\in\mathcal{D}_{\text{val}}} |u^*(x)|^2},$$

where $\widetilde{u}$ is an approximation of u^*. If specified, the line-search algorithm used for finding optimal α_k has to satisfy the *strong-Wolfe conditions* [40, Chapter 3]. Since the Gram matrix $G_\ell(\theta)$ can be rank-defficient and the computation of the natural gradient w_ℓ by solving the least-squares problem (28) can be tedious, we propose to add a *Tikhonov regularization* to (33), which translates into adding a regularization parameter $\lambda \cdot \mathrm{Id}$ to $G_\ell(\theta)$ so that now we have to solve the linear equation

$$(G_\ell(\theta) + \lambda \mathrm{Id}) \cdot d = \mathbb{E}_x\left[\left(\frac{\partial u_\theta}{\partial u_\ell}(x)\right)^\top \nabla\mathcal{L}(u_\theta)(x)\right].$$

This method can be seen as a *trust-region approach*. It shifts the spectrum of $G_\ell(\theta)$, making the solving more stable. The choice of λ remains an open question, but a rule of thumb is to increase λ when the algorithm tends to diverge and to decrease it otherwise. The tensor initialization is also an important key for the good behaviour of the optimization algorithms. In order to avoid the explosion of the variance with the layers and the number of features, we propose to choose the tensors as

$$\boldsymbol{\psi_\ell} \sim \mathcal{N}\left(0, \frac{c}{Ln^d}\right), \quad 1 \le \ell \le L,$$

where L is the number of layers, n is the number of features of Φ, and $c > 0$ is a positive constant chosen to be $c = 2$. This choice is motivated by the works on neural network weight initialization [22, 55].

5.1.1 A recovery problem

Let $d \geq 2$ and

$$u^*(x) = \prod_{i=1}^{m} (x_i y - x_{m+i})^2, \quad x \in \mathcal{X} = [0,1]^d, \tag{36}$$

$$y = \begin{cases} 1 & \text{if } d = 2m \\ x_d & \text{if } d = 2m+1 \end{cases}. \tag{37}$$

This function can be exactly represented by a compositional tensor with 2 layers. Indeed, for simplicity take $d = 2m$. For the case $d = 2m+1$, we just have to multiply the terms h_i, $1 \leq i \leq m$, by h_{2m+1}. Let

$$u_1(h) = (\underbrace{h_1 - h_m, h_2 - h_{m+1}, \ldots, h_m - h_{2m}}_{m}, \underbrace{h_1 - h_m, h_2 - h_{m+1}, \ldots, h_m - h_{2m}}_{m}), \tag{38}$$

$$u_2(h) = (h_1 h_2 \ldots h_{2m}, 0, \ldots, 0), \tag{39}$$

then we have exactly $u^* = u_2 \circ u_1$. Furthermore, u_1 can be represented by a TT with ranks bounded by 2 since it is just a difference of two variables, and u_2 can be represented by a TT with ranks equal to 1 since it is just the product of all variables.

For representing this function by a TT with polynomial basis, it must have maximal ranks, which grow exponentially with d. For example, if $d = 4$, then the TT ranks are $(1, 3, 9, 3)$ and the number of parameters is 180. This result can be verified experimentally: for the ranks ranging from 1 to 9, the relative L^2 error is $\{0.84, 0.68, 0.46, 0.16, 0.14, 0.11, 0.079, 0.013, 0\}$ (averaged over 25 experiments).

The lifting operator is chosen as $\mathfrak{L} = \mathrm{Id}$ and the retraction operator as $\mathfrak{R} = e_1$. The experiments are conducted for $d = 4$ and $d = 5$. The number of training samples is fixed to 2048, and the number of validation samples is fixed to 512. Regarding hyperparameters for each experiment,

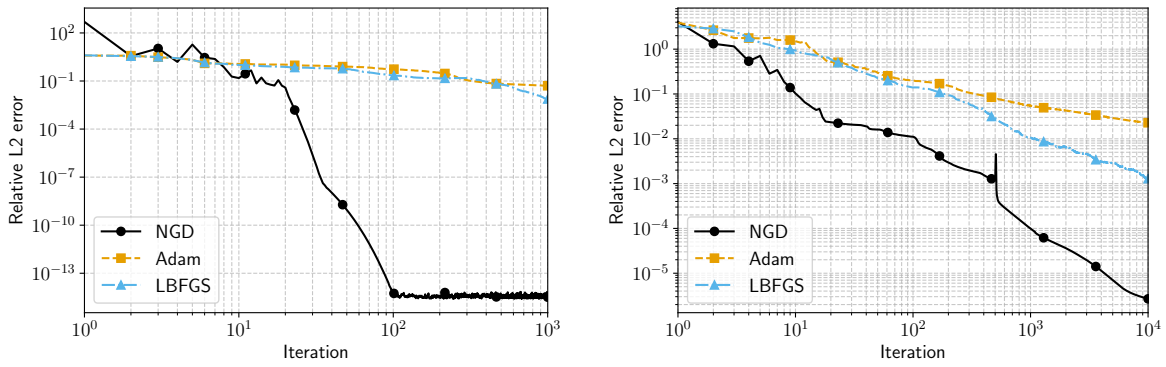
- for $d = 4$, the learning rate is fixed to $\alpha_k = 0.7$, and the regularization coefficient is $\lambda = 10^{-12}$,
- for $d = 5$, the learning rate is chosen constant $\alpha_k = 0.5$, and the regularization coefficient is $\lambda = 10^{-11}$.

The plots Figure 3 show that the NGD converges faster to an exact solution (or a good approximation) than the other optimisation algorithms. In Figure 3a we see that the NGD converges with a fast rate and converges quickly to the exact solution. In Figure 3b, the NGD does not converge to an exact solution of the problem. While it converges linearly, the rate is much better than of the other methods and a much lower error is reached after the same number of iterations.

The plots in Figure 4 show the evolution of the relative L^2 error versus the time spent for the optimisation algorithms. They show that the NGD is a bit faster than Adam for an iteration, and that, despite taking more time than L-BFGS, it converges faster.

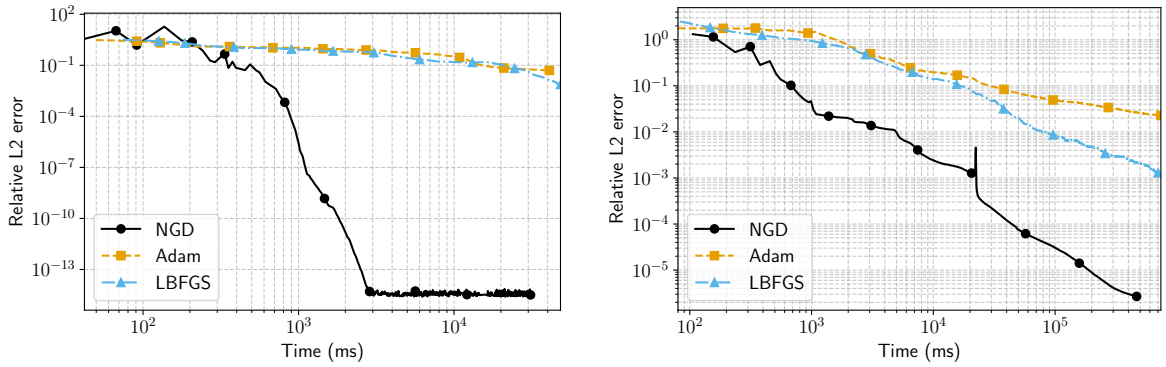
5.2 Layers in low-rank format

In this subsection, we study the properties and the behaviour of the Gram matrix $G(\theta)$ in order to check if the low-rank solvers like AMEn or (M)ALS can be used. A well-known major challenge with NGD is the occurring rank-defficiency. We also explore some regularization techniques to make these solvers exploitable.



(a) $d = 4$ (b) $d = 5$

Figure 3: Convergence plot for the optimizers Adam, NGD and L-BFGS for the recovery problem (37) in log-log scale, for dimensions $d = 4, 5$.



(a) $d = 4$ (b) $d = 5$

Figure 4: Relative L^2 error versus time for the optimizers Adam, NGD and L-BFGS for the recovery problem (37) in log-log scale, for dimensions $d = 4, 5$.

5.2.1 On the conditioning of $G(\theta)$

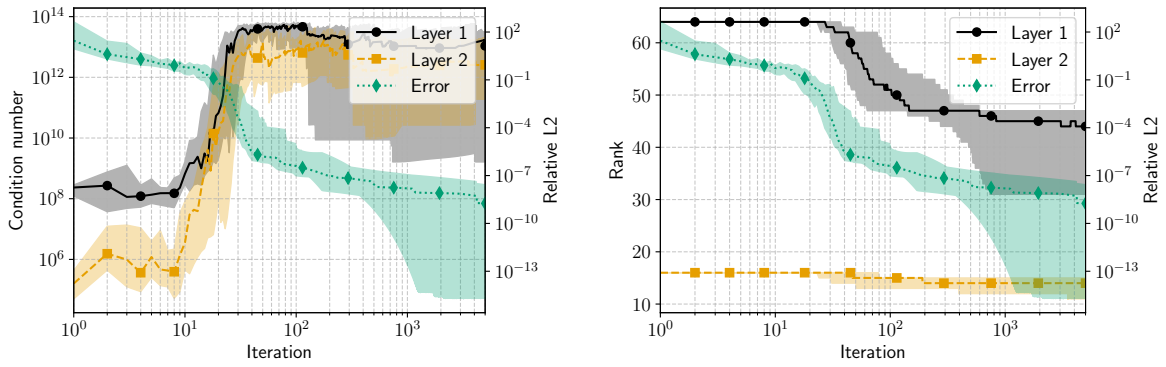
We propose to study the Gram matrix $G_\ell(\theta)$ of each layer $1 \leq \ell \leq L$. To do so, we track the *effective condition number*

$$\kappa_\ell(\theta) := \|G_\ell(\theta)\|_{2\to 2}\|G_\ell(\theta)^\dagger\|_{2\to 2}$$

during the optimisation of the problem (37) in dimension $d = 4$. We also track the rank of the Gram matrices G_ℓ. The experiment is repeated 25 times.

Figure 5a shows that the Gram matrices become highly ill-conditioned during optimization, with condition numbers κ_ℓ ranging from 10^6 up to 10^{14}. In fact, the condition number increases rapidly, reaching values around 10^{13} after approximately 20 iterations. Initially, directions associated with small eigenvalues play a useful role by guiding the optimizer toward a good configuration. However, as the solution approaches optimality, these directions contribute progressively less to the reduction of the loss.

Furthermore, as illustrated in Figure 5, the rank of the Gram matrices decreases over the iterations. This suggests that the effective feature space collapses onto a lower-dimensional subspace: the span of active feature directions shrinks, and the remaining features become increasingly correlated. Consequently, the Gram matrix becomes closer to being low-rank.



(a) Condition number κ_ℓ.

(b) Rank of G_ℓ.

Figure 5: Condition number and rank of G_ℓ, with $\ell = 1, 2$, for the recovery problem (37) for dimension $d = 4$. The plain line is the median, and the envelope corresponds to the interquartile.

5.2.2 Random sketching

Sketching size (s)	Learning rate (α)	Regularization parameter (λ)
20	0.85	10^{-11}
30	0.8	10^{-12}
40	0.7	10^{-12}

Table 1: Hyperparameters for each sketching size s.

As discussed in Section 4.2, one way to circumvent the ill-conditioned G_ℓ is to project it onto some random subspace. We apply this randomized method to the recovery problem defined in (37), considering different choice of sketching size s. In our experiments, the dimension is fixed to $d = 4$, resulting in $m = 4 \cdot 2^4 = 64$ parameters per layer. For each choice of s, the learning rate α and regularization parameter λ are adjusted accordingly and heuristically, as summarized in Table 1

Figure 6 shows that around 30 components of eigenspace of G_ℓ is enough to convergence to an optimal solution, which is less than half of the number of eigendirections. However, taking only 20 or 25 components is not enough. This plot also confirms that the more components we keep, the better the convergence is. One can remark that taking less than 64 eigendirections slows the convergence to an optimum.

Figure 6 illustrates the convergence behavior for various sketching sizes. We can observe that retaining a rank-30 approximation of the Gram matrix is sufficient to achieve convergence to an optimal solution, which is less than half of the total eigendirections. In contrast, retaining a rank 20 or 25 is inadequate for convergence. Moreover, the results confirm that increasing the rank improves convergence, although taking fewer than the full 64 eigendirections slows the rate at which the solution approaches the optimum.

We also want to point out that Figure 5b shows that the ranks of G_ℓ are not the same for each layer, and that in general the shallowest layer has a smaller rank. This suggests that the sketching size s should be adapted for each layer.

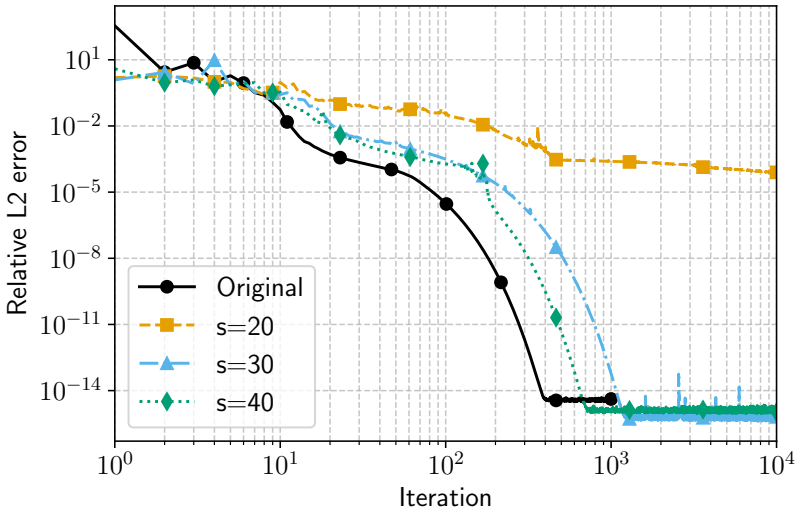


Figure 6: Random sketching for the problem (37) in dimension $d = 4$.

References

[1] Mazen Ali and Anthony Nouy. Approximation theory of tree tensor networks: Tensorized multivariate functions. *arXiv preprint arXiv:2101.11932*, 2021.

[2] Mazen Ali and Anthony Nouy. Approximation theory of tree tensor networks: Tensorized univariate functions. *Constructive Approximation*, 58(2):463–544, 2023.

[3] Shun-ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2):251–276, February 1998.

[4] Jimmy Ba, Roger Grosse, and James Martens. Distributed second-order optimization using Kronecker-factored approximations. In *International conference on learning representations*, 2017.

[5] Markus Bachmayr. Low-rank tensor methods for partial differential equations. *Acta Numerica*, 32:1–121, 2023.

[6] Markus Bachmayr, Anthony Nouy, and Reinhold Schneider. Approximation by tree tensor networks in high dimensions: Sobolev and compositional functions. *Pure and Applied Functional Analysis*, 8(2):405–428, 2023.

[7] Ivan Bioli, Carlo Marcati, and Giancarlo Sangalli. Accelerating natural gradient descent for PINNs with randomized numerical linear algebra, 2025.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[9] Tiangang Cui and Sergey Dolgov. Deep composition of Tensor-Trains using squared inverse Rosenblatt transports. *Foundations of Computational Mathematics*, 22(6):1863–1922, 2022.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

[11] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D. Lee. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, 20(1):119–154, January 2019.

[12] Alec Dektor and Daniele Venturi. Tensor rank reduction via coordinate flows. 491:112378.

[13] Ronald A. DeVore and George G. Lorentz. *Constructive Approximation*. Springer Berlin Heidelberg, 1993.

[14] Sergey V. Dolgov and Dmitry V. Savostyanov. Alternating Minimal Energy Methods for Linear Systems in Higher Dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, January 2014.

[15] Gerald B Folland. *Real analysis*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. John Wiley & Sons, Nashville, TN, 2 edition, March 1999.

[16] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker factored eigenbasis. *Advances in neural information processing systems*, 31, 2018.

[17] Alex Gittens and Michael Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 567–575, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[18] Erwan Grelier. *Learning with tree-based tensor formats : Application to uncertainty quantification in vibroacoustics*. PhD thesis, École centrale de Nantes, December 2019.

[19] Erwan Grelier, Anthony Nouy, and Mathilde Chevreuil. Learning with tree-based tensor formats, 2019.

[20] Erwan Grelier, Anthony Nouy, and Régis Lebrun. Learning high-dimensional probability distributions using tree tensor networks. *International Journal for Uncertainty Quantification*, 12(5), 2022.

[21] Michael Griebel, Helmut Harbrecht, and Reinhold Schneider. Low-rank approximation of continuous functions in Sobolev spaces with dominating mixed smoothness. *Mathematics of Computation*, 92(342):1729–1746, 2023.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[23] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[24] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed TT-rank. *Numerische Mathematik*, 120(4):701–731, September 2011.

[25] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, January 2012.

[26] Masato Inoue, Hyeyoung Park, and Masato Okada. On-line learning theory of soft committee machines with correlated hidden units–steepest gradient descent and natural gradient descent–. *Journal of the Physical Society of Japan*, 72(4):805–810, 2003.

[27] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pages 1724–1732. PMLR, 2017.

[28] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29, 2021.

[29] B. Kerimkulov, D. Šiška, and L. Szpruch. A modified MSA for stochastic control problems. *Applied Mathematics & Optimization*, 84(3):3417–3436, February 2021.

[30] Namjun Kim, Chanho Min, and Sejun Park. Minimum width for universal approximation using reLU networks on compact domain. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, January 1993.

[32] Qianxiao Li, Long Chen, Cheng Tai, and E Weinan. Maximum principle based algorithms for deep learning. *Journal of Machine Learning Research*, 18(165):1–29, 2018.

[33] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[34] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[35] Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, May 2020.

[36] Bertrand Michel and Anthony Nouy. Learning with tree tensor networks: Complexity estimates and model selection. *Bernoulli*, 28(2):910 – 936, 2022.

[37] Eiji Mizutani and Stuart Dreyfus. An analysis on negative curvature induced by singularity in multi-layer neural-network learning. *Advances in Neural Information Processing Systems*, 23, 2010.

[38] Johannes Müller and Marius Zeinhofer. Achieving high accuracy with PINNs via energy natural gradient descent. In *International Conference on Machine Learning*, pages 25471–25485. PMLR, 2023.

[39] Johannes Müller and Marius Zeinhofer. Position: Optimization in sciml should employ the function space geometry. In *International Conference on Machine Learning*, pages 36705–36722. PMLR, 2024.

[40] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[41] Levon Nurbekyan, Wanzhou Lei, and Yunan Yang. Efficient natural gradient descent methods for large-scale pde-based optimization problems. *SIAM Journal on Scientific Computing*, 45(4):A1621–A1655, 2023.

[42] E. J. Nyström. Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. *Acta Mathematica*, 54(0):185–204, 1930.

[43] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport, 2021.

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

[45] Ivan V Oseledets. Constructive representation of functions in low-rank tensor formats. *Constructive Approximation*, 37:1–18, 2013.

[46] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations*, 2021.

[47] Paul B Rohrbach, Sergey Dolgov, Lars Grasedyck, and Robert Scheichl. Rank bounds for approximating Gaussian densities in the tensor-train format. *SIAM/ASA Journal on Uncertainty Quantification*, 10(3):1191–1224, 2022.

[48] Mark Rudelson. Row products of random matrices. *Advances in Mathematics*, 231(6):3199–3231, 2012.

[49] R. Schneider and A. Uschmajew. Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. *Journal of Complexity*, 30(2):56 – 71, 2014. Dagstuhl 2012.

[50] Reinhold Schneider and Mathias Oster. *Some Thoughts on Compositional Tensor Networks*, page 419–447. Springer Nature Switzerland, 2024.

[51] David Sommer, Robert Gruhlke, Max Kirstein, Martin Eigel, and Claudia Schillings. Generative modelling with tensor train approximations of Hamilton–Jacobi–Bellman equations. *arXiv preprint arXiv:2402.15285*, 2024.

[52] Yiming Sun, Yang Guo, Joel A Tropp, and Madeleine Udell. Tensor random projection for low memory dimension reduction. *arXiv preprint arXiv:2105.00105*, 2021.

[53] Jesse van Oostrum, Johannes Müller, and Nihat Ay. Invariance properties of the natural gradient in overparametrised systems. *Information geometry*, 6(1):51–67, 2023.

[54] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[55] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.