Spectral Tensor Train Parameterization of Deep Learning Layers Supplementary Materials

Anton Obukhov ETH Zurich
Maxim Rakhuba HSE University
Alexander Liniger ETH Zurich
Zhiwu Huang ETH Zurich
Stamatios Georgoulis ETH Zurich
Dengxin Dai ETH Zurich
Luc Van Gool ETH Zurich, KU Leuven

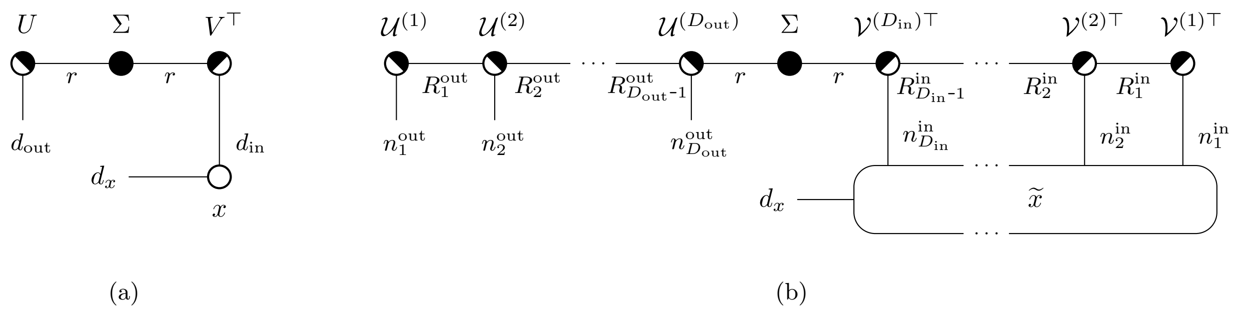

Figure S1: Tensor diagrams of a product of (a) SVDP and (b) STTP of a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and an input $x \in \mathbb{R}^{d_{\text{in}} \times d_x}$, where d_x is a batch or any other second dimension. See legend in Fig. 2 (of the main paper). $\widetilde{x}$ is obtained by tensorizing x along the dimension d_{in} using the same dimension factorization as the corresponding dimension of the matrix W. The factorized dimensions of $\widetilde{x}$ are connected with the recipient dimensions of TT-cores $\mathcal{V}^{(k)}$. The contraction order of all connected edges defines FLOPs and memory requirements for computing $y = Wx$. After the contraction, dimensions of the output $\widetilde{y}$ can be flattened to recover y.

S1 Computational Shortcuts of Low-Rank Affine Mappings

Both SVDP and STTP support "decompression" of the weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, which can be used for computing the mapping $\omega(x)$ directly as $y = Wx$ for some input $x \in \mathbb{R}^{d_{\text{in}} \times d_x}$. The last dimension of the input x may correspond to the batch dimension, so its value may be large during neural network training or equal to 1 during inference. The low-rank structure of the proposed parameterizations allows for taking certain computational shortcuts for computing either W or the mapping output y, as measured in floating-point operations (*FLOPs*).

SVDP The number of FLOPs required to decompress W given U, Σ, and V is $r \min(d_{\text{out}}, d_{\text{in}}) + 2rd_{\text{out}}d_{\text{in}}$. Computing $y = Wx$ then takes another $2d_{\text{out}}d_{\text{in}}d_x$ FLOPs. When $r \ll \min(d_{\text{in}}, d_{\text{out}})$, and $d_x = 1$, computing $y = U(\Sigma(V^\top x))$ following the order indicated by parentheses is preferred. Indeed, such computation brings the number of FLOPs down to $r(2d_{\text{in}} + 2d_{\text{out}} + 1)$. Overall, the *optimal contraction order* of a tensor diagram shown in Fig. S1a (which corresponds to arranging parentheses in the expression $U\Sigma V^\top x$) is defined by the sizes of all operands involved in the expression and can be precomputed upon the layer initialization.

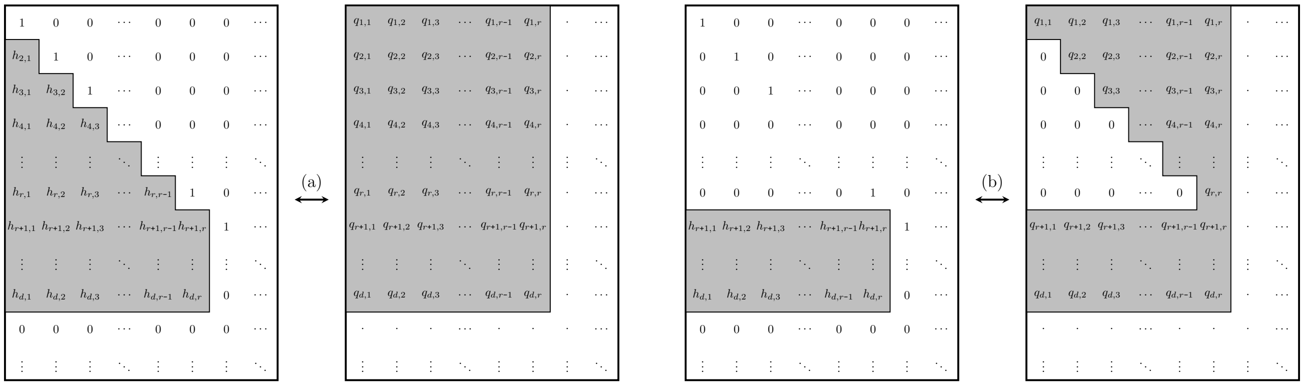

Figure S2: Visualization of the Padded Householder parameterization of orthonormal frames: (a) Full parameterization, (b) Reduced parameterization. Shaded areas with $h_{i,j}$ values represent parameters of reflectors; $q_{i,j}$ values represent orthonormal frame elements, affected by the parameterization. The padded parameterization allows for more efficient batching of orthonormal frames of different sizes for better parallelism at the expense of memory.

STTP After computing the TT-cores of matrices U and V from the underlying parameterizations, there are more than two ways to compute the mapping $\omega(x)$. As before, one can contract the tensor diagram of the matrix W first and then perform the regular computation of $y = Wx$. A slightly more efficient way is to contract matrices U and V and then re-use the approach to the low-rank mapping of SVDP.

Finally, the most efficient approach consists of the following steps: (1) factorization of the first dimension of x into D_{in} factors: $d_{\text{in}} = n_1 \dots n_{D_{\text{in}}}$, (2) tensorization of x into a tensor $\widetilde{x}$ according to the dimension factorization, (3) connecting factorized dimensions of $\widetilde{x}$ with the respective dimensions of the TT-cores $\mathcal{V}$, and finally, (4) contracting the resulting tensor diagram in Fig. S1b according to the optimal contraction order. While finding the optimal contraction order of a generic tensor diagram is an NP-hard problem, efficient algorithms exist for certain classes of graphs (Smith and Gray, 2018). This approach gives us the lowest possible FLOPs count of computing ω directly in the low-rank space, as both simpler approaches belong to the search space of the contraction order. As in the case of SVDP, the optimal contraction order depends on the topology of the tensor diagram and node sizes. Since the layer dimensions are known in advance, the mapping complexity is not increased at runtime.

S2 Batch Householder Transformation

Computations involving the proposed parameterizations are dominated by orthogonal transformations (6). In this section, we discuss some aspects that make our approach feasible as the size and the number of neural network layers grow. Despite Householder transformation being more amenable to SIMD implementation than Givens rotations and matrix exponential maps (Shepard et al., 2015), prior works avoid using them altogether due to the lack of framework support[1], complex implementations, and hardness to scale beyond a handful of layers. We overcome these limitations by utilizing a joint parameterization of orthonormal frames of the same size (Obukhov, 2021). In the context of SVDP, it allows us to generate multiple orthonormal frames of the same size $d \times r$ (potentially belonging to different layers) in a sequence of a total of r batched Householder reflections.

In the context of STTP, all matricized TT-cores are represented as orthonormal frames of a limited set of sizes and can be computed independently of each other. Specifically, the sizes are of the form $R_a n_b \times R_c$, where n_b belongs to the set of all possible factors of weight matrices' sizes in the whole network, and R_a, R_c belong to the set of all possible TT-rank values induced by matrix dimensions and rank r. In practice, we can reduce the set of different sizes of orthonormal frames used in the model by following high-level design recommendations discussed in Sec. S3. These observations lead to the improvement of TT-cores computation parallelism by having fewer different orthonormal frame sizes and a higher number of frames in each batch of a fixed size.

Padded Householder Although batching orthonormal frames of the same size improves parallelism, batches of different sizes are still processed in sequence. Here we show how to trade memory for parallelism and perform

[1]Even though an orthogonal transformation implementing (6) can be found in modern automatic differentiation packages as LAPACK bindings (`?ORGQR`), these functions rarely support batching or differentiation with respect to inputs.

parameterizations of multiple orthonormal frames of different sizes in a single batch. Concretely, given a set of sizes $d_1 \times r_1, \dots, d_k \times r_k$, we parameterize the respective orthonormal frames using the proposed Padded Householder parameterization in a batch of k matrices of size $\max(d_1, \dots, d_k) \times \max(r_1, \dots, r_k)$, as shown in Fig. S2a. Indeed, by zeroing $h_{p,i,j} : \forall i > d_p, \forall j > r_p$ in the p-th matrix of parameters, propagating 1 on the diagonal for $j > r_p$, and applying the Householder transformation, the resulting leading sub-matrix is in $\mathrm{St}(d_p, r_p), p = 1 \dots k$.

Householder parameterization padding can also be used together with the Reduced parameterization introduced in Sec. 3.1 (Fig. S2b) to describe elements in $\mathrm{St}_{\mathrm{U}}(d, r)$. Indeed, both Reduced and Padded variations employ the same transformation and differ only in the placement of constants $\{0, 1\}$. Thus it is possible to perform parameterization of all orthonormal frames needed by the model in a single batch of rank-r orthonormal frames.

S3 Neural Architecture Design for Efficient Batching of Orthonormal Frames

Sec. S2 points out the possibility of parallel computation of orthonormal frames, potentially belonging to different layers (and cores in STTP). The ability to compute most of the orthonormal frames of weight matrices in parallel is the defining factor of the compute throughput during training. There are a few neural architecture design traits, which have a direct impact on the effectiveness of such batching with or without padding.

Recall that SVDP of a 2D convolution with the weight matrix $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} K^2}$ requires the computation of orthonormal frames $U \in \mathbb{R}^{C_{\text{out}} \times R}$ and $V \in \mathbb{R}^{C_{\text{in}} K^2 \times R}$, where $R = \min(r, C_{\text{out}}, C_{\text{in}} K^2)$. To ensure that orthonormal frames from different layers can be batched, one should aim to reduce the amount of variation in the dimensions of matrices U and V belonging to different layers. For example, this is achieved with most residual architectures such as He et al. (2016); Zagoruyko and Komodakis (2016), which contain repetitions of residual blocks. Each unique orthonormal frame size forms a separate batch, which in turn requires a separate function call during training. Such cases include, for example, components of the preamble layer attaching to RGB inputs or layers with unique d_{out} or d_{in} less than r, causing rank demotion to satisfy the constraint $r \leq \min(d_{\text{out}}, d_{\text{in}})$.

STTP declares less strict constraints on the overall network architecture and weight matrix sizes than SVDP. Recall the TT parameterization of an orthonormal frame $U \in \mathbb{R}^{d \times r}$ requires computing dimension factorization $d = n_1 \dots n_D$ for some $n_1, \dots, n_D \in \mathbb{N} \setminus \{1\}$, for example, prime factors of d with repetition. Then U can have a low-rank parameterization through a number of TT-cores (8) with matricized dimensions $R_{k-1} n_k \times R_k$ for $k = 1, \dots, D$. The ranks R_k are defined as $\min(r, R_k^{\max})$ (3). To give a concrete example, consider a convolutional layer with the weight matrix $W \in \mathbb{R}^{16 \times 8 \cdot 3 \cdot 3}$, and $r = 4$. Then matrices $U \in \mathbb{R}^{16 \times 4}$ and $V^\top \in \mathbb{R}^{8 \cdot 3 \cdot 3 \times 4}$ will be tensorized into tensors $\widetilde{U} \in \mathbb{R}^{2 \times 2 \times 2 \times 2 \times 4}$ and $\widetilde{V} \in \mathbb{R}^{2 \times 2 \times 2 \times 3 \times 3 \times 4}$ using the prime factors of the first dimensions of matrices U and $V^\top$. The TT-rank and dimensions of $\widetilde{W}$ induced by such dimensions factorization (3.3) will be $R = (1, 2, 4, 4, 4, 4, 4, 4, 2, 1)$, $n = (2, 2, 2, 2, 3, 3, 2, 2, 2)$, and the complete set of matricized core sizes will contain: $\{(2 \times 2)^2, (4 \times 4)^2, (8 \times 4)^3, (12 \times 4)^2\}$ (upper index indicates the size of the batch). Thus, the relation of r to the size of the layer and the ability to factorize dimensions of the weight matrices play crucial roles in reducing the variation of orthonormal frame sizes involved in the parameterization. With this in mind, here are a few neural architecture design rules for maximum computation throughput and efficiency with STTP:

- r should be substantially smaller than the maximum dimension of a weight matrix in the whole network (e.g., $r = 64$ with 512 features in the largest layer);
- the set of convolutional filter sizes (e.g., $\{1, 3\}$) in the entire network should be small;
- usage of large (e.g., greater than 3) prime factors should be avoided in r, channel, and filter sizes;
- best throughput can be achieved with r, channel, and filter sizes being powers of a small factor (e.g., 2 or 3).

S4 Training Considerations

Optimizer weight decay, L2 regularization The role of regularization with SVDP and STTP is fundamentally different from the regularization of the regular affine layers. Whereas the latter results in simpler models due to the reduction of the Frobenius norm of weight matrices, the former will reduce individual reflectors' magnitudes, which promotes a truncated diagonal structure in weight matrices. This may or may not be the desired effect, depending on higher-level design decisions, such as the presence of skip connections; this topic is well beyond the scope of the current work. Frobenius regularization of the parameterized weight matrices can still be implemented by simply imposing an L2 penalty or performing weight decay of the learned singular values.

Initialization Most weight matrix initialization schemes in deep learning are motivated by norm preservation of the layer mapping (He et al., 2015). While SVDP and STTP achieve the same goal through the embedded spectral properties, a good initialization still plays an important role in the convergence speed. We experimented with three different ways of initializing orthonormal frames in both SVDP and STTP: (1) truncated identity matrix $I_{d\times r}$; (2) orthogonal initialization with QR decomposition of a random normal matrix (Saxe et al., 2014): $\mathrm{QR}(N_{d\times r})$, where elements of $N_{d\times r}$ are i.i.d. sampled from $\mathcal{N}(0,1)$; (3) orthogonal initialization with a noisy identity matrix $\mathrm{QR}(I_{d\times r}+\alpha N_{d\times r})$. All initialization schemes resulted in a good model performance at the end of the training; however, the noisy identity scheme with $\alpha = 1e{-}4$ exhibited faster convergence in the considered experiments with SNGAN. We conjecture that the best value of α depends on the dimensions d and r.

S5 Proof of Proposition 1

The fact that $\mathcal{M}(\mathcal{U}^{(k)}) \in \mathrm{St}(R_{k-1}n_k, R_k)$ implies $I_{R_k} = \mathcal{M}(\mathcal{U}^{(k)})^\top \mathcal{M}(\mathcal{U}^{(k)})$, or in the index notation,

$$\delta_{\mu\nu} = \sum_{\beta_{k-1}=1}^{R_{k-1}} \sum_{i_k=1}^{n_k} \mathcal{M}(\mathcal{U}^{(k)})_{\overline{\beta_{k-1}i_k},\mu}\, \mathcal{M}(\mathcal{U}^{(k)})_{\overline{\beta_{k-1}i_k},\nu} = \sum_{\beta_{k-1}=1}^{R_{k-1}} \sum_{i_k=1}^{n_k} \mathcal{U}^{(k)}_{\beta_{k-1},i_k,\mu}\mathcal{U}^{(k)}_{\beta_{k-1},i_k,\nu}, \tag{S1}$$

where $\delta_{\mu\nu}$ is the Kronecker delta. To show that $\mathcal{T}(\mathcal{U}^{(1)},\dots,\mathcal{U}^{(D)}) \in \mathrm{St}(n_1\cdots n_D, r)$, let us write the orthogonality condition in index notation (for the ease of notation, we omit ranges in which indices vary):

$$\begin{aligned}
&\sum_{i_1,\dots,i_D} \mathcal{T}(\mathcal{U}^{(1)},\dots,\mathcal{U}^{(D)})_{\overline{i_1\dots,i_D},\mu}\, \mathcal{T}(\mathcal{U}^{(1)},\dots,\mathcal{U}^{(D)})_{\overline{i_1\dots,i_D},\nu} \\
&\quad= \sum_{i_1,\dots,i_D} \left(\sum_{\beta_0,\dots,\beta_{D-1}} \mathcal{U}^{(1)}_{\beta_0,i_1,\beta_1}\mathcal{U}^{(2)}_{\beta_1,i_2,\beta_2}\cdots\mathcal{U}^{(D)}_{\beta_{D\text{-}1},i_D,\mu} \right) \left(\sum_{\beta_0,\dots,\beta_{D-1}} \mathcal{U}^{(1)}_{\beta_0,i_1,\beta_1}\mathcal{U}^{(2)}_{\beta_1,i_2,\beta_2}\cdots\mathcal{U}^{(D)}_{\beta_{D\text{-}1},i_D,\nu} \right) \\
&\qquad= \sum_{i_1,\dots,i_D} \sum_{\beta_0,\dots,\beta_{D-1}} \sum_{\tilde\beta_0,\dots,\tilde\beta_{D-1}} \mathcal{U}^{(1)}_{\beta_0,i_1,\beta_1}\mathcal{U}^{(1)}_{\tilde\beta_0,i_1,\tilde\beta_1}\cdots\mathcal{U}^{(D)}_{\beta_{D\text{-}1},i_D,\mu}\mathcal{U}^{(D)}_{\tilde\beta_{D\text{-}1},i_D,\nu},
\end{aligned} \tag{S2}$$

and note that β_0 and $\tilde\beta_0$ vary from 1 to 1, so with (S1) for $k=1$, we get

$$\sum_{i_1} \mathcal{U}^{(1)}_{1,i_1,\beta_1}\mathcal{U}^{(1)}_{1,i_1,\tilde\beta_1} = \delta_{\beta_1\tilde\beta_1}.$$

The latter expression implies that in the last line of (S2), after summing over i_1, only the terms with $\tilde\beta_1 = \beta_1$ remain. We can now apply (S1) for $k=2$:

$$\sum_{\beta_1,i_2} \mathcal{U}^{(2)}_{\beta_1,i_2,\beta_2}\mathcal{U}^{(2)}_{\beta_1,i_2,\tilde\beta_2} = \delta_{\beta_2\tilde\beta_2}.$$

Proceeding recursively, we obtain that (S2) equals $\delta_{\mu\nu}$, which completes the proof.

S6 Proof of Proposition 2

Let us first show that $\mathcal{T}\left(\mathcal{U}^{(1)},\mathcal{U}^{(2)},\dots,\mathcal{U}^{(D)}\right) = \mathcal{T}\left(\widetilde{\mathcal{U}}^{(1)},\widetilde{\mathcal{U}}^{(2)},\dots,\widetilde{\mathcal{U}}^{(D)}\right).$

Since $\widetilde{\mathcal{U}}^{(k)}_{:,i_k,:} = Q_{k-1}^\top \mathcal{U}^{(k)}_{:,i_k,:} Q_k$, $Q_k^\top Q_k = I$, and Q_0, Q_D are identity matrices of appropriate sizes, we have:

$$\begin{aligned}
\mathcal{T}\left(\widetilde{\mathcal{U}}^{(1)},\widetilde{\mathcal{U}}^{(2)},\dots,\widetilde{\mathcal{U}}^{(D)}\right)_{\overline{i_1\dots i_D},:} &= \left(Q_0^\top \mathcal{U}^{(1)}_{:,i_1,:}Q_1\right)\left(Q_1^\top \mathcal{U}^{(2)}_{:,i_2,:}Q_2\right)\cdots\left(Q_{D-1}^\top \mathcal{U}^{(D)}_{:,i_D,:}Q_D\right) \\
&= \mathcal{U}^{(1)}_{:,i_1,:}\mathcal{U}^{(2)}_{:,i_2,:}\cdots\mathcal{U}^{(D)}_{:,i_D,:} = \mathcal{T}\left(\mathcal{U}^{(1)},\mathcal{U}^{(2)},\dots,\mathcal{U}^{(D)}\right)_{\overline{i_1\dots i_D},:}.
\end{aligned}$$

Next, let us finally show that $\mathcal{M}(\widetilde{\mathcal{U}}^{(k)}) \in \mathrm{St}(R_{k-1}n_k, R_k)$. Indeed, $\mathcal{M}(\widetilde{\mathcal{U}}^{(k)}) = (Q_{k\text{-}1}^\top \otimes I_{n_k})\mathcal{M}(\mathcal{U}^{(k)})Q_k$ since

$$\begin{aligned}
\left(\mathcal{M}(\widetilde{\mathcal{U}}^{(k)})\right)_{\overline{\alpha_{k-1}i_k},\alpha_k} &= \sum_{\alpha_{k\text{-}1},\alpha_k=1}^{R_{k\text{-}1},R_k} (Q_{k\text{-}1})_{\alpha_{k\text{-}1},\beta_{k\text{-}1}} \mathcal{U}^{(k)}_{\alpha_{k\text{-}1},i_k,\alpha_k} (Q_k)_{\alpha_k,\beta_k} \\
&= \sum_{\alpha_{k\text{-}1},\alpha_k=1}^{R_{k\text{-}1},R_k} \sum_{j_k=1}^{n_k} (Q_{k\text{-}1})_{\alpha_{k\text{-}1},\beta_{k\text{-}1}} \delta_{i_k j_k} \mathcal{U}^{(k)}_{\alpha_{k\text{-}1},j_k,\alpha_k} (Q_k)_{\alpha_k,\beta_k} \\
&= \sum_{\alpha_{k\text{-}1},\alpha_k=1}^{R_{k\text{-}1},R_k} \sum_{j_k=1}^{n_k} (Q_{k\text{-}1})_{\alpha_{k\text{-}1},\beta_{k\text{-}1}} \delta_{i_k j_k} \left(\mathcal{M}(\mathcal{U}^{(k)})\right)_{\overline{\alpha_{k-1}j_k},\alpha_k} (Q_k)_{\alpha_k,\beta_k} \\
&= \left((Q_{k\text{-}1}^\top \otimes I_{n_k})\mathcal{M}(\mathcal{U}^{(k)})Q_k\right)_{\overline{\alpha_{k-1}i_k},\alpha_k}.
\end{aligned}$$

Hence,

$$\begin{aligned}
&\mathcal{M}(\widetilde{\mathcal{U}}^{(k)})^\top \mathcal{M}(\widetilde{\mathcal{U}}^{(k)}) = \left((Q_{k\text{-}1}^\top \otimes I_{n_k})\mathcal{M}(\mathcal{U}^{(k)})Q_k\right)^\top \left((Q_{k\text{-}1}^\top \otimes I_{n_k})\mathcal{M}(\mathcal{U}^{(k)})Q_k\right) \\
&= Q_k^\top \mathcal{M}(\mathcal{U}^{(k)})^\top (Q_{k\text{-}1}\,(Q_{k\text{-}1}^\top) \otimes I_{n_k})\,\mathcal{M}(\mathcal{U}^{(k)})Q_k = Q_k^\top \mathcal{M}(\mathcal{U}^{(k)})^\top \mathcal{M}(\mathcal{U}^{(k)})Q_k = Q_k^\top Q_k = I_{R_k},
\end{aligned}$$

which completes the proof.

S7 STTP Degrees of Freedom

We consider a tensor diagram from Fig. 2b made compatible with the TT decomposition introduced in Sec. 2 (consisting only of TT-cores) by contracting the matrix Σ into either left or right adjacent TT-core and recovering size-1 legs on the outer-most TT-cores. Such tensor diagram will have the following TT-rank and dimensions (3.3):

$$\begin{aligned}
R &= (1, R_1^{\text{out}}, \dots, R_{D_{\text{out}}-1}^{\text{out}}, r, R_{D_{\text{in}}-1}^{\text{in}}, \dots, R_1^{\text{in}}, 1), \\
n &= (n_1^{\text{out}}, \dots, n_{D_{\text{out}}}^{\text{out}}, n_{D_{\text{in}}}^{\text{in}}, \dots, n_1^{\text{in}}).
\end{aligned} \tag{S3}$$

Recall that R and n are indexed in the ranges $[0, D_{\text{out}} + D_{\text{in}}]$ and $[1, D_{\text{out}} + D_{\text{in}}]$ respectively (Sec. 2) and that STTP is obtained by parameterizing its Σ and TT-cores as follows (left to right in Fig. 2b, Sec. 3.3):

- $\mathcal{M}(\mathcal{U}^{(k)}) \in \mathrm{St}_{\mathsf{U}}(R_{k-1}^{\text{out}} n_k^{\text{out}} \times R_k^{\text{out}}), 1 \le k < D_{\text{out}}$, or equivalently $\mathrm{St}_{\mathsf{U}}(R_{k-1}n_k \times R_k), 1 \le k < D_{\text{out}}$ (TT-cores of U excluding the last one) in the notation (S3), parameterized by $R_{k-1}n_kR_k - R_k^2$ parameters,
- $\mathcal{M}(\mathcal{U}^{(k)}) \in \mathrm{St}(R_{k-1}^{\text{out}} n_k^{\text{out}} \times R_k^{\text{out}}), k = D_{\text{out}}$, or equivalently $\mathrm{St}(R_{k-1}n_k \times R_k), k = D_{\text{out}}$ (the last TT-core of U) in the notation (S3), parameterized by $R_{k-1}n_kR_k - R_k(R_k+1)\ /\ 2$ parameters,
- $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_r)$ is a matrix of singular values, parameterized by r parameters,
- $\mathcal{M}(\mathcal{V}^{(k)}) \in \mathrm{St}(R_{k-1}^{\text{in}} n_k^{\text{in}} \times R_k^{\text{in}}), k = D_{\text{in}}$, or equivalently $\mathrm{St}(R_k n_k \times R_{k-1}), k = D_{\text{out}} + 1$ (the last TT-core of V) in the notation (S3), parameterized by $R_kn_kR_{k-1} - R_{k-1}(R_{k-1}+1)\ /\ 2$ parameters,
- $\mathcal{M}(\mathcal{V}^{(k)}) \in \mathrm{St}_{\mathsf{U}}(R_{k-1}^{\text{in}} n_k^{\text{in}} \times R_k^{\text{in}}), D_{\text{in}} > k \ge 1$, or equivalently $\mathrm{St}_{\mathsf{U}}(R_k n_k \times R_{k-1}), D_{\text{out}} + 1 < k \le D_{\text{out}} + D_{\text{in}}$ (TT-cores of V excluding the last one) in the notation (S3), parameterized by $R_kn_kR_{k-1} - R_{k-1}^2$ parameters.

Summing up the degrees of freedom of STTP components listed above, and keeping in mind that $R_{D_{\text{out}}} \equiv r$,

$$\begin{aligned}
\text{DOF}(W) = &\left[\sum_{k=1}^{D_{\text{out}}-1} R_{k-1} n_k R_k - R_k^2\right] + \left[R_{k-1} n_k R_k - \frac{R_k(R_k+1)}{2} \;\Big|\; k = D_{\text{out}}\right] + r + \\
&+ \left[R_k n_k R_{k-1} - \frac{R_{k-1}(R_{k-1}+1)}{2} \;\Big|\; k = D_{\text{out}} + 1\right] + \left[\sum_{k=D_{\text{out}}+2}^{D_{\text{out}}+D_{\text{in}}} R_k n_k R_{k-1} - R_{k-1}^2\right] = \\
&= \left[r - \frac{r(r+1)}{2} - \frac{r(r+1)}{2}\right] + \sum_{k=1}^{D_{\text{out}}+D_{\text{in}}} R_{k-1} n_k R_k - \sum_{\substack{k \in [1, D_{\text{out}}-1] \cup \\ [D_{\text{out}}+1, D_{\text{out}}+D_{\text{in}}-1]}} R_k^2 \\
&= \sum_{k=1}^{D_{\text{out}}+D_{\text{in}}} R_{k-1} n_k R_k - \sum_{k=1}^{D_{\text{out}}+D_{\text{in}}-1} R_k^2,
\end{aligned}$$

we arrive at the same dimensionality of the fixed TT-rank tensor manifold as Holtz et al. (2012).

References

Smith, D., and Gray, J. (2018) opt_einsum - A Python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software 3*, 753.

Shepard, R., Brozell, S. R., and Gidofalvi, G. (2015) The representation and parametrization of orthogonal matrices. *The Journal of Physical Chemistry A 119*, 7924–7939.

Obukhov, A. Efficient Householder transformation in PyTorch. 2021; https://github.com/toshas/torch-householder.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. 2016; pp 770–778.

Zagoruyko, S., and Komodakis, N. Wide Residual Networks. Proceedings of the British Machine Vision Conference (BMVC). 2016; pp 87.1–87.12.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). USA, 2015; p 1026–1034.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. 2014.

Holtz, S., Rohwedder, T., and Schneider, R. (2012) On manifolds of tensors of fixed TT-rank. *Numerische Mathematik 120*, 701–731.