Adaptive Learning of Tensor Network Structures (Supplementary Material)

A Proof of Proposition 1

Proposition. *Let $\mathcal{G}^{(k)} \in \mathbb{R}^{R_{1,k} \times \cdots \times R_{k-1,k} \times d_k \times R_{k,k+1} \times \cdots \times R_{k,p}}$ for $k \in [p]$ be the core tensors of a tensor network and let $1 \leq i < j \leq p$. Let $\tilde{R}_{i'j'} = R_{i',j'} + 1$ if $(i', j') = (i, j)$ and $R_{i',j'}$ otherwise, and define the core tensors $\tilde{\mathcal{G}}^{(k)} \in \mathbb{R}^{\tilde{R}_{1,k} \times \cdots \times \tilde{R}_{k-1,k} \times d_k \times \tilde{R}_{k,k+1} \times \cdots \times \tilde{R}_{k,p}}$ for $k \in [p]$ by*

$$(\tilde{\mathcal{G}}^{(i)})_{(j)} = \begin{bmatrix} (\mathcal{G}^{(i)})_{(j)} \\ -\mathbf{0}- \end{bmatrix}, (\tilde{\mathcal{G}}^{(j)})_{(i)} = \begin{bmatrix} (\mathcal{G}^{(j)})_{(i)} \\ -\mathbf{0}- \end{bmatrix} \text{ and } \tilde{\mathcal{G}}^{(k)} = \mathcal{G}^{(k)} \text{ for } k \in [p] \setminus \{i, j\}$$

where $\mathbf{0}$ denotes a row vector of zeros of the appropriate size in each block matrix.

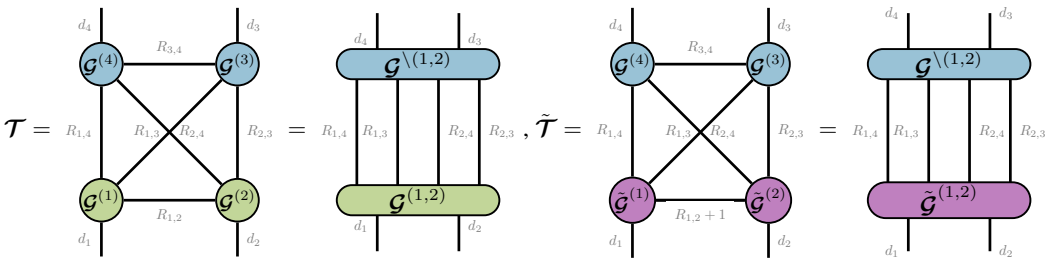
Then, the core tensors $\tilde{\mathcal{G}}^{(k)}$ correspond to the same tensor network as the core tensors $\mathcal{G}^{(k)}$, i.e., TN$(\tilde{\mathcal{G}}^{(1)}, \cdots, \tilde{\mathcal{G}}^{(p)})$ = TN$(\mathcal{G}^{(1)}, \cdots, \mathcal{G}^{(p)})$.

Proof. Let $\mathcal{T} = \text{TN}(\mathcal{G}^{(1)}, \cdots, \mathcal{G}^{(p)})$ and $\tilde{\mathcal{T}} = \text{TN}(\tilde{\mathcal{G}}^{(1)}, \cdots, \tilde{\mathcal{G}}^{(p)})$. We first split the TN $\mathcal{T}$ and $\tilde{\mathcal{T}}$ in two parts by isolating the ith and jth nodes from the other nodes of the TN:

- let $\mathcal{G}^{\setminus(i,j)} \in \mathbb{R}^{\prod_{k \neq i,j} d_k \times \prod_{k \neq j} R_{i,k} \times \prod_{k \neq i} R_{j,k}}$ be the tensor obtained by contracting all the core tensors of $\mathcal{T}$ except for the ith and jth cores,
- let $\mathcal{G}^{(i,j)} \in \mathbb{R}^{d_i \times d_j \times \prod_{k \neq j} R_{i,k} \times \prod_{k \neq i} R_{j,k}}$ be the tensor obtained by contracting $\mathcal{G}^{(i)}$ and $\mathcal{G}^{(j)}$ along their shared index (i.e., the jth mode of the ith core is contracted with the jth mode of the ith core),
- let $\tilde{\mathcal{G}}^{(i,j)} \in \mathbb{R}^{d_i \times d_j \times \prod_{k \neq j} R_{i,k} \times \prod_{k \neq i} R_{j,k}}$ be the tensor obtained by contracting $\tilde{\mathcal{G}}^{(i)}$ and $\tilde{\mathcal{G}}^{(j)}$ along their shared index.

One can check that the contraction between the last two modes of $\mathcal{G}^{\setminus(i,j)}$ and the last two modes of $\mathcal{G}^{(i,j)}$ is a reshaping of $\mathcal{T}$. Similarly, since $\tilde{\mathcal{G}}^{(k)} = \mathcal{G}^{(k)}$ for any k distinct from i and j, the contraction over the last two modes of $\mathcal{G}^{\setminus(i,j)}$ and $\tilde{\mathcal{G}}^{(i,j)}$ gives rise to the same reshaping of $\tilde{\mathcal{T}}$. Therefore to prove $\mathcal{T} = \tilde{\mathcal{T}}$, it suffices to show that $\mathcal{G}^{(i,j)} = \tilde{\mathcal{G}}^{(i,j)}$.

This argument is illustrated in the tensor network diagrams below for the particular case $p = 4$, $i = 1$, $j = 2$.

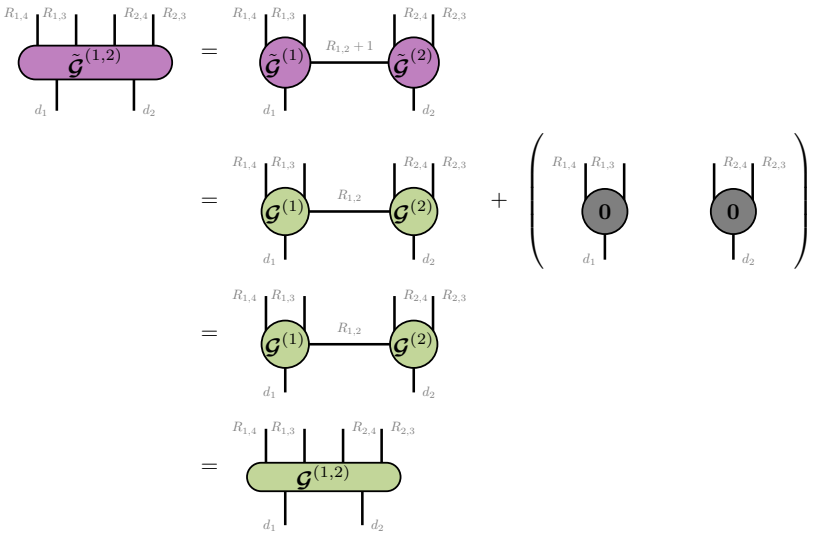


Let $(\mathcal{G}^{(i,j)})_{[1,3]}$ (resp. $(\tilde{\mathcal{G}}^{(i,j)})_{[1,3]}$) be the matricization of $\mathcal{G}^{(i,j)}$ (resp. $\tilde{\mathcal{G}}^{(i,j)}$) with mode 1 and 3 as the rows and modes 2 and 4 as the columns. We have

$$(\tilde{\mathcal{G}}^{(i,j)})_{[1,3]} = \tilde{\mathcal{G}}^{(i)\top}_{\langle j \rangle} \tilde{\mathcal{G}}^{(j)}_{\langle i \rangle} = \mathcal{G}^{(i)\top}_{\langle j \rangle} \mathcal{G}^{(j)}_{\langle i \rangle} + \mathbf{0}\mathbf{0}^\top = (\mathcal{G}^{(i,j)})_{[1,3]},$$

where the notation $\boldsymbol{\mathcal{A}}^{(m)}_{\langle n \rangle}$ denotes the matrix obtained by transposing the mth mode of $\boldsymbol{\mathcal{A}}^{(m)}$ to the first mode and matricizing the resulting tensor along the nth mode if $m < n$ and along the $(n+1)$th mode if $m > n$[†]. It then follows that $\boldsymbol{\mathcal{G}}^{(i,j)} = \tilde{\boldsymbol{\mathcal{G}}}^{(i,j)}$, hence $\boldsymbol{\mathcal{T}} = \widetilde{\boldsymbol{\mathcal{T}}}$.

Continuing with the particular case $p = 4$, $i = 1$, $j = 2$, the second part of the proof can be illustrated by the following tensor network diagrams.



□

B Algorithm Details

B.1 Subroutines of Greedy-TN

The pseudo-code of the subroutines of `Greedy-TN` (Algorithm 1) are given in Algorithms 2, 3 and 4.

Algorithm 2 `add-slice`$(\boldsymbol{\mathcal{G}}^{(i)}, j)$

Input: Core tensor to add new slice to $\boldsymbol{\mathcal{G}}^{(i)}$, mode to add new slice j.

1: **if** $j > i$ **then**

2: $\hat{\boldsymbol{\mathcal{G}}}^{(i)} \leftarrow \texttt{reshape}\Big(\begin{bmatrix} (\boldsymbol{\mathcal{G}}^{(i)})_{(j)} \\ -\mathbf{0}- \end{bmatrix}, (R_{1,i} \times \cdots \times R_{i-1,i} \times d_i \times R_{i,i+1} \times \cdots \times R_{i,j-1} \times (R_{i,j} + 1) \times R_{i,j+1} \times \cdots \times R_{i,p}) \Big)$

3: **else if** $j < i$ **then**

4: $\hat{\boldsymbol{\mathcal{G}}}^{(i)} \leftarrow \texttt{reshape}\Big(\begin{bmatrix} (\boldsymbol{\mathcal{G}}^{(i)})_{(j)} \\ -\mathbf{0}- \end{bmatrix}, (R_{1,i} \times \cdots \times R_{j-1,i} \times (R_{j,i}+1) \times R_{j+1,i} \times \cdots \times R_{i-1,i} \times d_i \times R_{i,i+1} \times \cdots \times R_{i,p}) \Big)$

Output: $\hat{\boldsymbol{\mathcal{G}}}^{(i)}$

[†]For example, if $\boldsymbol{\mathcal{A}}^{(2)} \in \mathbb{R}^{n_1 \times d \times n_3 \times n_4}$, $\boldsymbol{\mathcal{A}}^{(2)}_{\langle 3 \rangle} \in \mathbb{R}^{n_3 \times d n_1 n_4}$ is obtained by transposing $\boldsymbol{\mathcal{A}}^{(2)}$ in a tensor of size $d \times n_1 \times n_3 \times n_4$ and matricizing the resulting tensor along the 3rd mode. Similarly, $\boldsymbol{\mathcal{A}}^{(2)}_{\langle 1 \rangle} \in \mathbb{R}^{n_1 \times d n_3 n_4}$ is obtained by transposing $\boldsymbol{\mathcal{A}}^{(2)}$ in a tensor of size $d \times n_1 \times n_3 \times n_4$ and matricizing the resulting tensor along the 2nd mode. Note that $\boldsymbol{\mathcal{A}}^{(m)}_{\langle n \rangle}$ is always a column-wise permutation of the classical matricization $\boldsymbol{\mathcal{A}}^{(m)}_{(n)}$.

Algorithm 3 `find-best-edge`$(\mathcal{L}, (\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}))$

Input: Loss function $\mathcal{L}$, core tensors $\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}$.

best-loss $\leftarrow \infty$
for $i \leftarrow 1$ to p **do**
  for $j \leftarrow i+1$ to p **do**
    $\hat{\boldsymbol{\mathcal{G}}}^{(i)} \leftarrow$ `add-slice`$(\boldsymbol{\mathcal{G}}^{(i)}, j)$
    $\hat{\boldsymbol{\mathcal{G}}}^{(j)} \leftarrow$ `add-slice`$(\boldsymbol{\mathcal{G}}^{(j)}, i)$
    $(\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(i-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(i)}, \boldsymbol{\mathcal{G}}^{(i+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(j-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(j)}, \boldsymbol{\mathcal{G}}^{(j+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}) \leftarrow$ `optimize` $\mathcal{L}(\text{TN}(\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(i-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(i)}, \boldsymbol{\mathcal{G}}^{(i+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(j-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(j)}, \boldsymbol{\mathcal{G}}^{(j+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}))$ w.r.t. new slices in $\hat{\boldsymbol{\mathcal{G}}}^{(i)}$ and $\hat{\boldsymbol{\mathcal{G}}}^{(j)}$
    loss $= \mathcal{L}(\text{TN}(\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(i-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(i)}, \boldsymbol{\mathcal{G}}^{(i+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(j-1)}, \hat{\boldsymbol{\mathcal{G}}}^{(j)}, \boldsymbol{\mathcal{G}}^{(j+1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}))$
    if loss $<$ best-loss **then**
      best-edge $= (i, j)$
      best-loss $=$ loss

Output: best-edge

Algorithm 4 `split-nodes`$((\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}), \varepsilon)$

Input: Core tensors $\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)}$, splitting node threshold ε.

for $i \leftarrow 1$ to p **do**
  for every bi-partition (M, N) of $[p]$ **do**
    $\mathbf{U}, \mathbf{D}, \mathbf{V}^\top = \varepsilon$-`truncated-SVD`$(\text{reshape}(\boldsymbol{\mathcal{G}}^{(i)}, d_i \prod_{j \in M} R_{i,j} \times \prod_{j \in N} R_{i,j}))$
    $\hat{R} \leftarrow$ rank of the ε-`truncated-SVD`
    if splitting node $\boldsymbol{\mathcal{G}}^{(i)}$ reduces the number of parameters **then**
      $\forall j \in [p]$, let $\tilde{R}_{i,j} = R_{i,j}$ if $j \in M$ and 1 otherwise.
      $\boldsymbol{\mathcal{G}}^{(i)} \leftarrow \text{reshape}(\mathbf{U}, \tilde{R}_{1,i} \times \cdots \times \tilde{R}_{i-1,i} \times d_i \times \tilde{R}_{i,i+1} \times \cdots \times \tilde{R}_{i,p} \times \hat{R})$
      $\forall j \in [p]$, let $\tilde{R}_{i,j} = R_{i,j}$ if $j \in N$ and 1 otherwise.
      $\boldsymbol{\mathcal{G}}^{(p+1)} \leftarrow \text{reshape}(\mathbf{D}\mathbf{V}^\top, \tilde{R}_{1,i} \times \cdots \times \tilde{R}_{i-1,i} \times \hat{R} \times \tilde{R}_{i,i+1} \times \cdots \times \tilde{R}_{i,p} \times 1))$
      for $j \in [p] \setminus \{i\}$ **do**
        $\boldsymbol{\mathcal{G}}^{(j)} \leftarrow \text{reshape}(\boldsymbol{\mathcal{G}}^{(j)}, R_{1,j} \times \cdots \times R_{j-1,j} \times d_j \times R_{j,j+1} \times \cdots \times R_{j,p} \times 1))$
      $p \leftarrow p + 1$

Output: $(\boldsymbol{\mathcal{G}}^{(1)}, \cdots, \boldsymbol{\mathcal{G}}^{(p)})$

B.2 Computational Complexity

The overall time complexity of `Greey-TN` is dominated by the whose complexity is in $\mathcal{O}(p^2 T + pd^2 R^{2p})$ where T is the time complexity of optimizing the loss function w.r.t. one of the core tensors. The first term corresponds to the `find-best-edge` subroutine and the second one corresponds to the `split-nodes` sub-routine. For example, when optimizing a squared error loss with SGD, T is in $\mathcal{O}(R^{p-1} d^p)$ where $R = \max_{i,j} R_{i,j}$ is the maximum rank in the tensor network and $d = \max_i d_i$ is the maximum dangling dimension. Thus, in this case, when $R \leq d$ the overall complexity is dominated by the `find-best-edge` subroutine.

C Additional Experimental Results

C.1 Tensor Decomposition Experiment

In Figure 7, we show the most frequent tensor network structure recovered by the greedy algorithm for each of the 4 targets used in the tensor decomposition experiment (see Figure 4). We see that `Greedy-TN` and `Greedy-int` always recover the same structure except for the Tucker target, where `Greedy-TN` finds the best TN structure without internal nodes to approximate the target. We also observe that the greedy algorithm recovers the correct TN structure for all targets most of the time, except for the TR target.

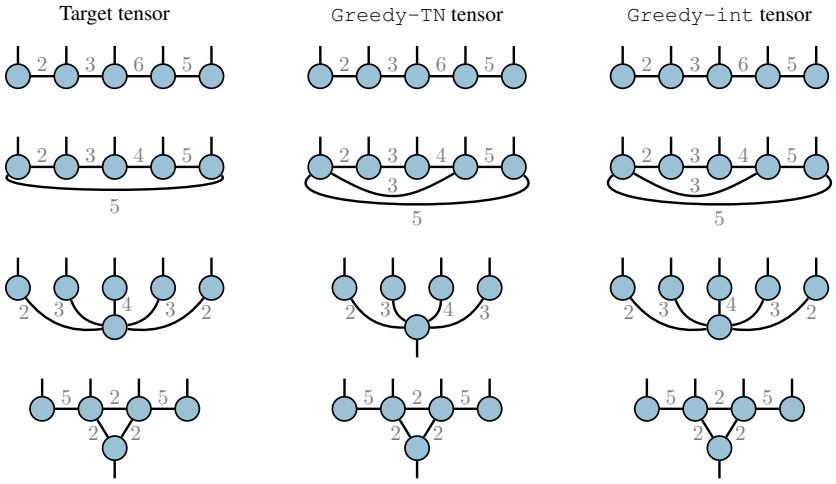


Figure 7: Most common tensor network structure returned by `Greedy-TN` and `Greedy-int` over the 100 runs of the tensor decomposition experiment.

C.2 Image Completion

The images recovered at each iteration of `Greedy-TN` on the Einstein image completion task, along with the relative test error and number of parameters for each step, are shown in Figures 8 and 9.

C.3 Ablation Study

Here, we study if transferring the weights at each step would lead to better results. We randomly generate 50 target tensors of size $7 \times 7 \times 7 \times 7 \times 7$ with a TT structure of rank $6, 3, 6, 5$. We run `Greedy-TN` with and without weight transfer until convergence.

The results are shown in Figure 10, where we see that using the weight transfer mechanism results in a lower loss with the same number of parameters, compared to using a random initialization at each greedy step. This shows that transferring the knowledge from the previous greedy iterations leads to a better initialization for the continuous optimization.

Figure 8: Solutions found by `Greedy-TN` for the Einstein image completion experiments, labeled by number of parameters and relative test error w.r.t. the full image. **[continued on next page]**

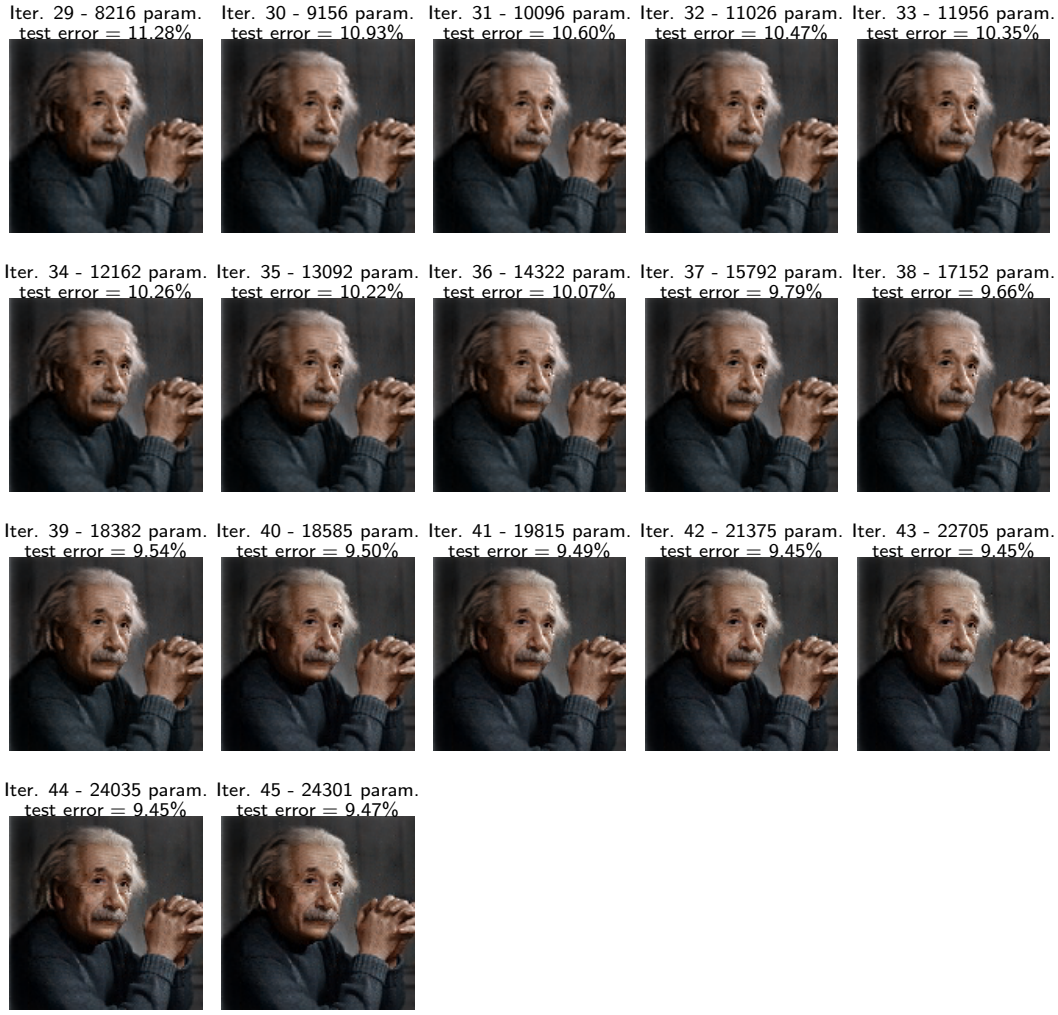


Figure 9: Solutions found by `Greedy-TN` for the Einstein image completion experiments, labeled by number of parameters and relative test error w.r.t. the full image. **[continued from previous page]**

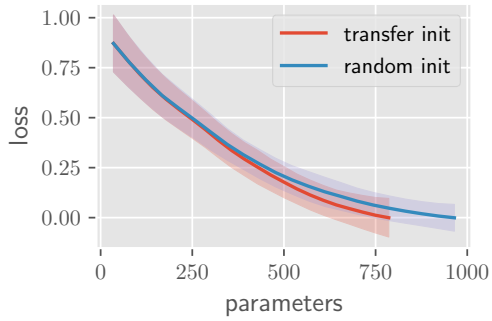


Figure 10: Comparison of `Greedy-TN` with and without weight transfer on a TT structure decomposition task. Curves represent the reconstruction error averaged over 50 runs, and shaded areas correspond to standard deviations.