Graph-Constrained Structure Search for Tensor Network Representation: Supplementary Materials

Anonymous Author(s)
Affiliation
Address
email

In the supplementary material (SM), we first show two additional results including the correspondence between TNs and graphs and the coding efficiency of the proposed method mentioned in the manuscript. After that, in section 2, we give the proofs for our theoretical results given in the manuscript and SM. Last, additional results and details about the experiments are introduced.

1 Additional discussion

Table 1 illustrates the correspondence between graphs and the TN models. As shown in Table 1, different TN models correspond to different graphs in general. For instance, the TT model corresponds to a path graph P_N, while the TR model corresponds to a cycle graph C_N. Furthermore, we observe that the number of vertices $|V|$ indicates the order of the TN, since all the cores are assumed to to external, *i.e.*, all the cores own "free-legs", which corresponds to the tensor modes. We can also observe that the maximum degree Δ_{G_0} is independent from $|V|$ in the models of TT, TR and PEPS. It implies that the cores in these TNs have a bounded tensor order, which is independent from the order of the TNs.

Below, we give the coding efficiency of our proposed method on coding the graph-constrained TN structures.

Corollary (Coding efficiency)**.** *With the assumption in Proposition 9 and a discrete uniform distribution on $\mathbb{H}_{G_0,R}$. Let $L_{\min}$ be the minimum lossless code length on $\mathbb{H}_{G_0,R}$, then the gap between $L_{\min}$ to the proposed method satisfies*

$$L_{ours} - L_{\min} \leq \mathcal{O}\left(|V| \log(|V|)\right).$$

The proof is given at the end of Section 2. As shown in the corollary, the coding redundancy by our method is upper-bounded by $\mathcal{O}\left(|V| \log(|V|)\right)$. It is because the random-key trick allows multiple codes in the key space to correspond to the same permutation. However, such the redundancy also helps to solve the irregularity issue mentioned in the manuscript.

2 Proofs

Proof of Lemma 5. According to the isomorphism relationship given in Definition 3 about the topology-constrained TN structures, we know that there exists a permutation matrix $\mathbf{P}$ such that the "unweighted form" of $\mathbf{H}$ satisfying $\mathbf{H}^u = \mathbf{P}\mathbf{H}_0\mathbf{P}^\top$, where $\mathbf{H}_0$ denotes the adjacency matrix of G_0. We then have its weighted form satisfying $\mathbf{H} = l_R\left(\mathbf{P}\mathbf{H}_0\mathbf{P}^\top\right)$, where $l_R(\,\cdot\,)$ represents the weighting function on each non-zero entries of the adjacency matrix. Since the permutation matrix corresponds to a bijective mapping of indices of a matrix, we can rewrite the above formula by

$$\mathbf{H} = \mathbf{P}\mathbf{H}_0^w\mathbf{P}^\top = \mathbf{P}\Psi^{-1}\left(G_0, f_R\right)\mathbf{P}^\top, \tag{1}$$

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Graphs corresponding to different tensor networks (TNs). In the table, $|V|$ $|E_0|$, $|Aut(G_0)|$ denotes the number of vertices, edges and automorphisms of G_0 respectively, and Δ_{G_0} denotes the maximum degree of G_0. The last row illustrates examples of graphical diagrams for each TN model. For brevity, we omit the "free-legs" from the tensor diagram.

TNs	TT [11, 13]	T-Tree [19]	TR [23, 7]	PEPS [15, 16]	CTN [24]
Graphs G_0	Path P_N	Tree T_N	Cycle C_N	Lattice $L_{m,n}$	Complete K_N
$\|V\|$	N	N	N	mn	N
$\|E_0\|$	$N-1$	$N-1$	N	$(m-1)(n-1)$	$N(N-1)/2$
$\|Aut(G_0)\|$	2	1 to $(N-1)!$	$2N$	$\leq mn$	$N!$
Δ_{G_0}	2	$[2, N-1]$	2	$2, 3, 4$	$N-1$
Examples					

where $\mathbf{H}_0^w$ denotes the adjacency matrix of G_0 weighted by f_R, which with Ψ is defined in Lemma 2 and can be constructed by l_R. Let $\mathbb{F}_R$ be the set containing all f_R under G_0, we then construct a bijective mapping $g : \mathbb{F} \to (\mathbb{Z}_R)^{|E_0|}$, in which we sequentially put the weight on each edge in E_0 into each entry of a vector in $(\mathbb{Z}_R)^{|E_0|}$ following a following a subtraction by one. It is apparent that the mapping g is bijective. Therefore we have

$$\mathbf{H} = \mathbf{P}\Psi^{-1}\left(G_0, f_R\right)\mathbf{P}^\top = \mathbf{P}\Psi^{-1}\left(G_0, g^{-1}(\mathbf{z})\right)\mathbf{P}^\top, \tag{2}$$

where $\mathbf{z} \in \mathbb{Z}_R^{|E_0|}$. Since both the Ψ and g are biejctive, their composition Ω: $\mathbf{z} \mapsto \Psi^{-1}\left(G_0, g^{-1}(\mathbf{z})\right)$ is also bijective. The result is therefore proved. □

Proof of Proposition 6. The idea to prove the first claim is based on the fact that the adjacency matrices of isomorphic graphs are the same up to permutation. Let $\mathbf{H}_0$ be the adjacency matrix of G_0. Since the graph or its complement is not complete, there is a pair of indices (i_k, j_k), $i_k \neq j_k$, $i, j \in [N]$, $k = 1, 2$ such that $\mathbf{H}_0(i_1, j_1) = 0$ and $\mathbf{H}_0(i_2, j_2) \neq 0$. Because of the definition of $\mathbb{H}_{G_0,R}$ we know that all the isomorphisms of G_0 are contained in $\mathbb{H}_{G_0,R}$. Thus there is a permutation mapping $\pi : [N] \to [N]$ and its corresponding $\mathbf{H}_1 \in \mathbb{H}_{G_0,R}$ such that $\mathbf{H}_1(i_1, j_1) = \mathbf{H_1}(\pi(i_2), \pi(j_2)) = \mathbf{H_0}(i_2, j_2)$. In this case, $nz(\mathbf{H}_0 + \mathbf{H}_1) \neq 0 > nz(\mathbf{H}_0 \neq 0)$, where $\mathbf{X} \neq 0$ represents the logic operation to check if the entries of $\mathbf{X}$ equal zero, and $nz(\,\cdot\,)$ denotes the function to have the number of non-zero entries of a matrix. It can be infered from the inequality that the number of edges of the graph G induced by $\mathbf{H}_0 + \mathbf{H}_1$ is larger than G_0. Then G is not isomorphic to G_0. Therefore $\mathbf{H}_0 + \mathbf{H}_1 \notin \mathbb{H}_{G_0,R}$. The proof for the first claim is complete.

The basic idea to prove the second claim is to have the joint probability of the perturabtion and the element from $\mathbb{H}_{G_0,R}$ such that their addition is not in $\mathbb{H}_{G_0,R}$. In particular, assuming we draw the elements from $\mathbb{H}_{G_0,R}$ at a uniform random distribution we have

$$\begin{aligned} &Pr\left(\{\mathbf{B} \in \mathbb{B}, \mathbf{H} \in \mathbb{H}_{G_0,R} | \mathbf{B} + \mathbf{H} \notin \mathbb{H}_{G_0,R}\}\right) \\ &\geq Pr\left(\{\mathbf{B} \in \mathbb{B}, \mathbf{H} \in \mathbb{H}_{G_0,R-1} | \mathbf{B} + \mathbf{H} \notin \mathbb{H}_{G_0,R}\}\right). \end{aligned} \tag{3}$$

The inequality is held since we shrink the size of the event. By some basic rules on probability we further have

$$\begin{aligned} &Pr\left(\{\mathbf{B}, \mathbf{H} \in \mathbb{H}_{G_0,R-1} | \mathbf{B} + \mathbf{H} \notin \mathbb{H}_{G_0,R}\}\right) \\ &= Pr\left(\mathbf{H} \in \mathbb{H}_{G_0,R-1}\right) Pr\left(\{\mathbf{B} | \mathbf{B} + \mathbf{H} \notin \mathbb{H}_{G_0,R}\} \,|\mathbf{H} \in \mathbb{H}_{G_0,R-1}\right) \\ &= \left(1 - \frac{1}{R}\right)^{|E_0|} \left(1 - \left(\frac{1}{2}\right)^{\frac{|V|^2 - |V| - 2|E_0|}{2}}\right). \end{aligned} \tag{4}$$

Suppose the graph G_0 is (k,l)-sparse, we have $|E_0| = k|V| - l$. By that we get

$$\begin{aligned} &Pr\left(\{\mathbf{B} \in \mathbb{B}, \mathbf{H} \in \mathbb{H}_{G_0,R} | \mathbf{B} + \mathbf{H} \notin \mathbb{H}_{G_0,R}\}\right) \\ &\geq \left(1 - \frac{1}{R}\right)^{k|V|-l} \left(1 - \left(\frac{1}{2}\right)^{\frac{|V|^2-(1+2k)|V|+2l}{2}}\right) \end{aligned} \tag{5}$$

The above inequality gives a lower-bound of the probability that the perturbation is not closed, and can be simplied as the result given in the manuscript for a sparse G_0 with a large R. The proof is therefore completed. □

Proof of Lemma 7. First, we know from the proof of Lemma 5 that for all $\mathbf{H} \in \mathbb{H}_{G_0}$ it can be decomposed as $\mathbf{H} = f_R\left(\mathbf{P}\mathbf{A}_0\mathbf{P}^\top\right)$, where $\mathbf{P}$ denotes the permutation matrix and f_R denotes the edge weighting function on adjacency matrices. Assume $\mathbf{H}_i = f_{R,i}\left(\mathbf{P}_i\mathbf{A}_0\mathbf{P}_i^\top\right)$, $i = 1, 2$, and $\mathbf{P}_1\mathbf{A}_0\mathbf{P}_1^\top \neq \mathbf{P}_2\mathbf{A}_0\mathbf{P}_2^\top$. It implies $\mathbf{P}_1$, $\mathbf{P}_2$ are not in the same automorphism. In this case we have $\mathbf{H}_1 \neq \mathbf{H}_2$ if and only if $f_{R,1} \neq f_{R,2}$. Hence $|\mathbb{H}_{G_0}| = |\mathbb{F}_R|\frac{|\mathbb{S}_{|V|}|}{|Aut(G_0)|} = R^{|E_0|}\frac{|V|!}{|Aut(G_0)|}$. The result is proved. □

Proof of Proposition 8. The results can be obtained by combining of Lemma 7 and the results given in Table 1. For the Tucker model, we can see its corresponded graph is a complete K-partite, *i.e.*, $K1, N$, where the left subset of vertices (only one vertex) corresponds to the internal core. We then know the size of graph automorphisms for the Tucker model equaling $N!$. □

Proof of Proposition 9. According to Lemma 7, we have

$$\begin{aligned} \log\left(\mathbb{H}_{G_0,R}\right) &\geq |E_0|\log(R) + \log\left(|V|!\right) - \log\left(|V|\right) - \log\left(\Delta_{G_0}!\right) - \left(N - \Delta_{G_0} - 1\right)\log\left(\Delta - 1\right) \\ &= |E_0|\log(R) + \log\left((|V|-1)!\right) - \log\left(\Delta_{G_0}!\right) - \left(N - \Delta_{G_0} - 1\right)\log\left(\Delta - 1\right) \\ &\geq \left(\frac{1}{2}\sum_{v\in V}\deg\left(v\right)\right)\log(R) + \log\left((|V|-1)!\right) - \log\left(\Delta_{G_0}!\right) - \left(N - \Delta_{G_0} - 1\right)\log\left(\Delta - 1\right) \\ &\geq \frac{1}{2}|V|\delta_{G_0}\log(R) + \log\left((|V|-1)!\right) - \log\left(\Delta_{G_0}!\right) - \left(N - \Delta_{G_0} - 1\right)\log\left(\Delta - 1\right) \\ &\geq \frac{1}{2}|V|\delta_{G_0}\log(R) + \frac{1}{2}\log(2\pi) + \left(|V| + \frac{1}{2}\right)\log(|V|) \\ &\quad - |V| - 1 - \left(\Delta_{G_0} + \frac{1}{2}\right)\log(\Delta) + \delta_{G_0} - \left(|V| - \Delta_{G_0} - 1\right)\log(\Delta_{G_0} - 1) \\ &\approx \mathcal{O}\left(|V|\log(R) + |V|\log(|V|)\right) \end{aligned} \tag{6}$$

where δ_{G_0} denotes the minimum degree of G_0. Note that the first inequality holds due to Theorem 2 in the work [6], the second inequality holds by the Handshaking lemma, and we obtain the fourth inequality by the Stirling's approximation. □

Proof of the coding efficiency corollary. We give the code length by the proposed method. First, we have the length corresponding to the rank aspect equalling $L_{rank} = |E_0|\log(R)$. Then the code length of the permutation aspect by the random key trick can be given by $L_p = C|E_0|$, where C denotes a constant *w.r.t.* the quantization accuracy on each random number. We therefore have the total code length as

$$L_{ours} = L_{rank} + L_p = |E_0|\log(R) + C|E_0| \leq |E_0|\log(R) + \frac{1}{2}|V|\Delta_{G_0,R}C, \tag{7}$$

where the inequality holds due to the handshaking lemma. On the other side, assume the discrete uniform distribution on $\mathbb{H}_{G_0,R}$, which obeys the principle of maximum entropy, we then know L_{min}, the entropy on $\mathbb{H}_{G_0,R}$, equals the logarithm of the cardinality of $\mathbb{H}_{G_0,R}$ known from Proposition 9. Therefore, the coding efficiency of our method is obtained as follows.

$$L_{ours} - L_{\min} \leq \mathcal{O}\left(|V|\log(|V|)\right). \tag{8}$$

Compared to Proposition 9, the term $|V|\log(R)$ is eliminated because L_{ours} also contains the same term as known from Eq. (8) and (6). □

3 Additional details of the experimental results

3.1 Structure search on synthetic tensor.

3.1.1 Structure search on data in TR format

Configuration of GA. In GA, throughout the synthetic data experiments, the maximum number of the generations is set to be 30. The population in each generation are set to be 150 under all settings. To balance the scale between the compression ratio and RSE, the trade-off parameter λ in the fitness score is set to be 200. During each generation in GA, 36% of the individuals with the worst fitness scores are eliminated and we adopted the reproduction trick in [] and set the reproduction number to be 2. Meanwhile, to calculate the selection probability of the recombination operation, we choose the hyper-parameter α = 20, β = 1. Moreover, we deploy a chance of 24% for each gene to mutate after the recombination is finished. We follow the differentiable programming approach [9] for computation of the RSE. Concretely, for each individual, we initialize the core tensors with Gaussian distribution of zero mean and 0.1 standard deviation, and apply the Adam optimizer [5] with a learning rate of 0.001 to carry out the gradient descent steps. we repeat the decomposition 4 times under different initialization for each individual so as to avoid the local minima during the TN decomposition, then select the smallest RSE for fitness evaluation.

Discussion on additional results. In this section, the GA-based algorithm that only learns the ranks by our coding method is also implemented termed as *TRGA-R*. The parameter setting of this algorithm is the same as our method and the experimental results are reported in Table 2. From the result we can see that TRGA-R fail dealing with the permutation on tensor-modes and these results indicate the fact that there is no TR decomposition which can perfectly learn the TR decomposition with permutation on tensor-modes and this demonstrates the importance of learning the permutation of the TR decomposition.

Moreover, we also attempt to search the optimal TR structure for an order-20 tensor in TR format. The aim is to evaluate the effectiveness of the methods in the high-order case. To generate the data, we first let the dimension of each tensor mode equal 2. Then, we randomly generate the TR-ranks at discrete uniform distribution on $\{1, 2, 3\}$ and the cores at Gaussian distribution $N(0, 0.3)$, and randomly permute the tensor modes after contracting the cores. In GA, the maximum number of the generations is set to be 50 and the TR-ranks bound R be equal to 3. The trade-off parameter λ is set to 100 and elimination rate is set to 10%. Furthermore, the initialization of core tensors is according to Gaussian distribution of $N(0, 0.3)$. Other parameters are same to the ones given in the above experiment. Table 3 illustrates experimental results obtained by different methods. As shown in Table 3, our method achieves the best $Eff.$, yet all the methods cannot learn the structure as good as the ground-truth even for ours. The reason is mainly about the extremely huge search space for the order-20 tensor.

3.1.2 Structure search on data in other TN format

Data Generation. For the synthetic data generation of T-tree (order-7) [19], PEPS (order-6) [15], hierarchical Tucker (H-Tucker, order-6) [3] and multi-scale entanglement renormalization ansatz (MERA, order-8) [2, 12], we first let the dimension of each tensor mode equal 3. Then, we randomly generate the TN-ranks at discrete uniform distribution on $\{1, 2, 3, 4\}$ according to the corresponding graphs demonstrated in Figure 1. In the Figure, the blue nodes with an outer indices indicate the external cores and the orange nodes indicate the internal cores. After that, the cores are generated at Gaussian distribution $N(0, 0.1)$, and randomly permute the tensor modes on the blue nodes after contracting the cores. Note that in MERA we impose additional cores (the blue ones) for evaluating the proposed in a larger search space.

Coding method on the H-Tucker and MERA model. Unlike T-tree and PEPS, which only contain external cores, the coding schemes for H-Tucker and MERA is different. Specifically, for H-Tucker and MERA, we fix the permutations of the internal cores, and therefore only use the random key to encode the permutation of the external cores.

Table 2: Experimental results of searching structures on synthetic data in TR format. In the table, *Eff.* denotes the parameter ratio between the structures by different methods and the ground-truths; *RSE* in round brackets indicates the relative square error (ignored if smaller than 10^{-4}.) and *Gen.* in angle brackets indicates the generation of the reported individual in TNGA, TRGA-R and our method.

Trial	**Order 4** – *Eff.*↑ *(RSE*↓*)* ⟨*Gen.*↓⟩						
	TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**TRGA-R**	**TNGA** [8]	**Ours**
A	1.00	1.00	0.21	1.00	1.00 ⟨006⟩	1.00 ⟨004⟩	1.00 ⟨**003**⟩
B	0.64	1.00	1.00	0.64	1.00 ⟨004⟩	1.00 ⟨002⟩	1.00 ⟨003⟩
C	1.17	1.17	0.23	1.00	1.17 ⟨006⟩	1.17 ⟨005⟩	1.17 ⟨**003**⟩
D	0.57	0.57	0.32	1.25 (0.10)	0.80 (0.01) ⟨007⟩	1.00 ⟨003⟩	1.00 ⟨**002**⟩
E	0.43	0.48	0.40	0.40	0.48 ⟨001⟩	1.00 ⟨007⟩	1.00 ⟨003⟩
Trial	**Order 6** – *Eff.*↑ *(RSE*↓*)* ⟨*Gen.*↓⟩						
	TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**TRGA-R**	**TNGA** [8]	**Ours**
A	0.21	0.44	0.14 (2e-3)	0.25 (2e-3)	0.46 (8e-3) ⟨026⟩	0.82 ⟨011⟩	**1.00** ⟨**010**⟩
B	0.14	0.15	0.14	0.44 (0.40)	0.29 (0.02) ⟨021⟩	0.90 (6e-3) ⟨015⟩	**1.00** ⟨**009**⟩
C	0.57	1.00	0.85	0.29	1.00 ⟨008⟩	1.00 ⟨022⟩	1.00 ⟨012⟩
D	0.21	0.39	0.10	0.13	0.55 (0.01) ⟨007⟩	1.03 ⟨018⟩	**1.16** ⟨010⟩
E	0.15	0.30	0.01 (0.02)	0.12	0.27 (1e-3) ⟨008⟩	1.00 ⟨016⟩	1.00 ⟨**007**⟩
Trial	**Order 8** – *Eff.*↑ *(RSE*↓*)* ⟨*Gen.*↓⟩						
	TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**TRGA-R**	**TNGA** [8]	**Ours**
A	0.10	0.16	0.03 (0.20)	0.03	0.16 (3e-3) ⟨027⟩	0.48 ⟨017⟩	**1.00** ⟨019⟩
B	0.09	0.43	0.06 (0.02)	0.06 (7e-4)	0.34 (2e-3) ⟨013⟩	0.29 (2e-3) ⟨020⟩	**1.02** ⟨015⟩
C	0.03	0.31	0.02 (0.01)	0.02	0.37 (3e-3) ⟨007⟩	0.49 ⟨015⟩	**1.11** ⟨025⟩
D	0.20	0.53	0.02 (0.07)	0.02 (0.02)	0.53 ⟨014⟩	0.32 ⟨027⟩	**1.06** ⟨**013**⟩
E	0.33	0.33	0.02 (0.02)	0.02 (3e-3)	0.33 ⟨006⟩	0.23 ⟨023⟩	**0.88** ⟨010⟩

Table 3: Experimental results of searching structures on an order-20 synthetic tensor in TR format. In the table, *Eff.* denotes the parameter ratio between the structures by different methods and the ground-truths; *RSE* in round brackets indicates the relative square error and *Gen.* in angle brackets indicates the generation of the reported individual in our method.

Order 20 – *Eff.*↑ *(RSE*↓*)* ⟨*Gen.*↓⟩				
TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**Ours**
0.16 (0.23)	0.25 (0.23)	0.19 (0.49)	0.46 (1.00)	0.64 (0.22) ⟨032⟩

3.2 Benchmarks on real-world data

3.2.1 Image compression

Data Preprocessing. In the experiment, we randomly select 14 natural images from the BSD500 [1]. We use the Matlab commands to "resize" and "rgb2gray" to turn these into grayscaled images of size 256×256, and then these grayscaled images is rescaled to $[0, 1]$, following tensorization of the size $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4$. The images used in this experiment are demonstrated in Figure 2.

Configuration of GA. For our method, we spawn a group of individuals with population 300 in each generation, and set the maximum number of generations, elimination rate to be 30 and 10%, respectively. In addition, the bound of TR-ranks is set to 14, and we set $\lambda = 5$ and the learning rate of the Adam optimizer to be 0.01. Moreover, we set the reproduction number to be 1, the chance of mutation to be 30%. We initialize the core tensors with Gaussian distribution of $N(0, 0.1)$. Meanwhile, to calculate the selection probability of the recombination operation, we choose the hyper-parameter α = 25, β = 1.

Additional results. The compression ratio (CR, in log form) and RSE (in round brackets) of 14 natural images by the prposed methods and TR-SVD, TR-LM, TR-ALSAR, Bayes-TR and TRGA-R

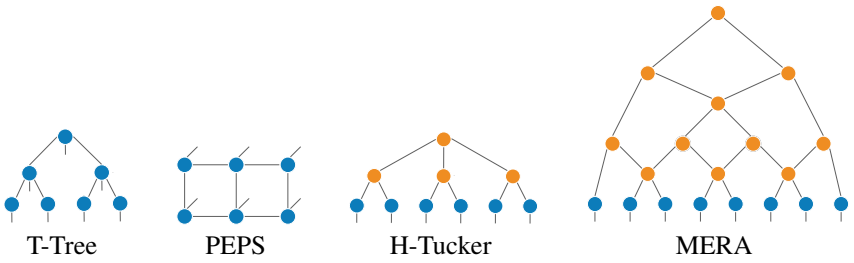

Figure 1: Illustration of the TN structures applied in the synthetic experiment.

Figure 2: Illustration of the employed images in image compression experiment.

are demonstrated in Table 4. In the table we also report the permutations learned by our method. For TRGA-R the parameter λ is manually adjust to meet the RSE obtained by our method and the other parameters is set as the same.

3.2.2 Image completion

Data Preprocessing. In the experiment, 8 images from USC-SIPI [18] are chosen. We use the Matlab command "resize" to turn those into images of size $256 \times 256 \times 3$, and then these images are further rescaled to $[0, 1]$, following VDT to get its tensorized form of the size $4^8 \times 3$. The images used in this experiment are demonstrated in Figure 3.

To generate image with missing data, we firstly use Matlab command "randperm" to generate random integer sequence with length equal to number of image elements. Based on the missing rate, we select a subset of this sequence to generate a $0 - 1$ mask tensor with size equal the image and using this mask we can generate the missing image.

Configuration of GA. For our method, we spawn a group of individuals with population 300 in each generation, and set the maximum number of generations, elimination rate to be 30 and 10% respectively. In addition, the bound of TR-ranks is set to 14, and we set λ which balance the scale between compression ratio and the observed values RSE to be $1.5, 0.0008, 0.0007$ for missing rate $0.5, 0.7, 0.9$. The learning rate of the Adam optimizer to be 0.001. Moreover, we set the reproduction number to be 1, the chance of mutation to be 24%. We initialize the core tensors with Gaussian distribution of $N(0, 0.1)$. Meanwhile, to calculate the selection probability of the recombination operation, we choose the hyper-parameter α = 25, β = 1.

Additional results. The RSE of predicting the missing values of 8 color images under different missing rate by the proposed method and TTSGD, TRLRF, TRALS, TRWOPT are demonstrated in Table 5. In these methods, we search the TR ranks from 2 to 14 and the TT ranks from 2 to 18 for each image to obtain the best results. Visual comparison of different methods in recovering 90% missing images are shown in Figure 4.

3.2.3 Reparameterization of tensorial Gaussian process

Datasets. In this task, we choose three univariate regression datasets from the UCI and LIBSVM archives. The Combined Cycle Power Plant (CCPP)[1] dataset consists of 9569 data points collected

[1] `https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant`

Table 4: Experimental results of images compression. In the table, *log CR* denotes the compression ratio in the log form; *RSE* in round brackets indicates the relative square error and *Permutation* in brace indicates the permutations learned by our method. We highlight the results if the both CR and RSE achieve the best.

Images	**Trivial** – *log CR*↑ *(RSE*↓*)* {*Permutation*}					
	TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**TRGA-R**	**Ours**
0	0.7616 (0.1549)	0.7452 (0.1539)	1.4563 (0.2587)	0.9031 (0.1556)	1.0771 (0.1572)	1.1045 (0.1549) {12348765}
1	0.9891 (0.1428)	0.8388 (0.1349)	1.2317 (0.1556)	1.1530 (0.1356)	1.3006 (0.1360)	**1.3428 (0.1338)** {15678432}
2	0.8497 (0.1539)	0.8201 (0.1549)	1.2549 (0.1803)	0.9591 (0.1661)	1.2660 (0.1568)	1.3162 (0.1559) {12348765}
3	0.9417 (0.1738)	0.9300 (0.1783)	1.3268 (0.1865)	1.0834 (0.1949)	1.3207 (0.1712)	**1.3675 (0.1706)** {15678432}
4	0.7571 (0.1806)	0.7549 (0.1792)	1.3988 (0.2553)	0.9591 (0.1871)	1.0513 (0.1806)	1.0658 (0.1780) {12567843}
5	1.2680 (0.0825)	1.2749 (0.0812)	1.1664 (0.0806)	1.6369 (0.0804)	1.7373 (0.0825)	**1.7673 (0.0800)** {13487562}
6	1.0942 (0.1000)	1.1722 (0.0995)	1.2953 (0.1179)	1.4028 (0.0985)	1.4421 (0.0975)	**1.4717 (0.0959)** {15678432}
7	1.1846 (0.1196)	1.1568 (0.1233)	1.3274 (0.1245)	1.4028 (0.1166)	1.5216 (0.1183)	1.5670 (0.1179) {13487652}
8	0.6712 (0.1673)	0.6712 (0.1673)	1.1895 (0.2223)	0.9031 (0.1694)	1.0998 (0.1676)	1.1154 (0.1676) {12348765}
9	0.7555 (0.1606)	0.8001 (0.1600)	0.3518 (0.2083)	0.9591 (0.1649)	1.1778 (0.1622)	**1.1928 (0.1597)** {12348765}
10	1.1005 (0.1043)	1.1151 (0.1026)	1.2943 (0.1024)	1.5051 (0.1054)	1.5680 (0.1039)	1.5789 (0.1025) {12348765}
11	0.9687 (0.1113)	0.9687 (0.1113)	1.5526 (0.1620)	1.2285 (0.1162)	1.3070 (0.1149)	1.3517 (0.1105) {12348765}
12	1.0896 (0.1337)	0.9694 (0.1258)	1.8113 (0.1552)	1.4028 (0.1291)	1.4480 (0.1257)	1.4877 (0.1245) {15678432}
13	1.0579 (0.1092)	1.0238 (0.1095)	1.6091 (0.1535)	1.2285 (0.1065)	1.3274 (0.1063)	1.3291 (0.1063) {18765432}
Images	**VDT** – *log CR*↑ *(RSE*↓*)* {*Permutation*}					
	TR-SVD [23]	**TR-LM** [10]	**TR-ALSAR** [23]	**Bayes-TR** [14]	**TRGA-R**	**Ours**
0	0.8871 (0.1679)	0.9191 (0.1738)	1.6716 (0.2902)	0.9591 (0.1751)	1.0975 (0.1682)	1.0906 (0.1676) {13456782}
1	1.1281 (0.1411)	1.0513 (0.1367)	1.4022 (0.1838)	1.1005 (0.1351)	1.2948 (0.1338)	1.2799 (0.1334) {18765432}
2	1.1281 (0.1664)	1.0621 (0.1622)	0.3208 (0.1581)	1.0597 (0.1662)	1.3207 (0.1597)	1.2974 (0.1581) {18765432}
3	1.0787 (0.1783)	1.0968 (0.1758)	1.2645 (0.1808)	1.4151 (0.1885)	1.3806 (0.1749)	1.3619 (0.1741) {12345768}
4	0.8559 (0.2015)	0.8382 (0.1931)	0.6981 (0.2490)	1.0430 (0.2149)	1.0658 (0.1884)	1.0752 (0.1892) {17865432}
5	1.7106 (0.0837)	1.6222 (0.0812)	0.0260 (0.0831)	1.6211 (0.0789)	1.7657 (0.0800)	1.7330 (0.0787) {18765432}
6	1.2349 (0.1000)	1.2487 (0.1015)	1.3656 (0.1225)	1.3223 (0.1020)	1.4213 (0.0995)	1.4248 (0.1000) {13287654}
7	1.2232 (0.1221)	1.0427 (0.1187)	1.0985 (0.1319)	1.4028 (0.1220)	1.4836 (0.1196)	1.4756 (0.1204) {18765432}
8	0.8852 (0.1780)	0.8673 (0.1766)	1.4055 (0.2324)	1.0430 (0.1936)	1.0812 (0.1744)	1.0860 (0.1764) {18675432}
9	0.9860 (0.1842)	0.8478 (0.1787)	1.1949 (0.2215)	0.9664 (0.1846)	1.1302 (0.1738)	1.1062 (0.1720) {12345678}
10	1.2675 (0.1036)	1.2450 (0.1020)	1.1537 (0.1086)	1.4028 (0.1043)	1.5555 (0.1049)	1.5137 (0.1020) {12876543}
11	1.1334 (0.1123)	1.1695 (0.1197)	1.4298 (0.1571)	1.2285 (0.1166)	1.2964 (0.1086)	1.3184 (0.1118) {18765432}
12	1.1309 (0.1226)	1.1313 (0.1234)	1.3845 (0.1459)	1.2285 (0.1236)	1.3668 (0.1200)	1.3916 (0.1225) {12345678}
13	1.0292 (0.1284)	1.0138 (0.1410)	1.2870 (0.1759)	1.1530 (0.1331)	1.2431 (0.1281)	1.2254 (0.1261) {13456872}

from a power plant over six years (2006-2011), where the response is the hourly electrical energy output (EP) and 4 features are hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V). The Protein[2] data contain 45730 instances with 9 attributes and a single response. The MG[3] data have 1385 data points with 6 features. For all the datasets, we standardize responses and features by removing the mean and scaling to unit variance, then randomly choose 80% of the data for training and the rest for testing, which is the same with settings in TTGP [4].

Task. In this experiment, we aim to demonstrate that our GA method is capable of searching more efficient structures of given TT representations in machine learning tasks, such as Gaussian process (GP). Specifically, tensorial Gaussian process (TTGP) [4] trains a GP by tensorizing and representing the variational mean vector of the inducing points with TT format. However, TTGP are restriced to TT format and the TT-ranks are treated as hyper-parameters and pre-defined. To learn more compacted structures, we firstly train a TTGP with given TT-ranks (we choose 10 here) and get the TT representation of the variational mean. Then we use the proposed GA method to search for alternative TN structures of the variational mean. Finally, we plug the learned variational mean into the original TTGP model for inference. We evaluate the results by mean squared error (MSE) on

[2] `https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure`

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#mg`

Figure 3: Illustration of the employed images in image completion experiment.

Table 5: images completion results under different missing rates

Images	50% **missing**				
	TTSGD [22]	**TRLRF** [21]	**TRALS** [17]	**TRWOPT** [20]	**Ours**
0	0.2963	0.2280	0.2332	0.2242	**0.2121**
1	0.1288	0.0751	0.0795	0.0839	**0.0704**
2	0.1864	0.1345	0.1452	0.1452	**0.1309**
3	0.1888	0.1537	0.1590	0.1583	**0.1527**
4	0.1178	**0.0770**	0.0845	0.0888	0.0772
5	0.1858	0.1363	0.1468	0.1396	**0.1294**
6	0.1343	**0.0942**	0.1039	0.1055	0.0961
7	0.0800	0.0565	0.0534	0.0596	**0.0503**
Images	70% **missing**				
	TTSGD [22]	**TRLRF** [21]	**TRALS** [17]	**TRWOPT** [20]	**Ours**
0	0.3029	0.2724	0.2521	0.2360	**0.2352**
1	0.1320	0.0814	0.0850	0.0851	**0.0762**
2	0.1853	0.1422	0.1484	0.1514	**0.1395**
3	0.1978	0.1646	0.1633	0.1682	**0.1603**
4	0.1176	0.0902	0.0899	0.0916	**0.0819**
5	0.1888	0.1407	0.1520	0.1574	**0.1345**
6	0.1351	0.1030	0.1060	0.1072	**0.0983**
7	0.0831	0.0691	0.0561	0.0649	**0.0556**
Images	90% **missing**				
	TTSGD [22]	**TRLRF** [21]	**TRALS** [17]	**TRWOPT** [20]	**Ours**
0	0.3227	0.4536	0.3679	0.3540	**0.3203**
1	0.1310	0.1392	0.1152	0.1286	**0.1139**
2	0.1960	0.2064	0.1884	0.2011	**0.1775**
3	0.2036	0.2113	**0.2004**	0.2028	0.2056
4	0.1301	0.1691	0.1330	0.1311	**0.1203**
5	0.1971	0.1960	0.1992	0.2078	**0.1835**
6	0.1471	0.1425	0.1389	0.1421	**0.1250**
7	0.0840	0.0917	0.0700	0.0741	**0.0697**

the test datasets. The results show that our method achieves almost the same MSE with the original TTGP by using fewer parameters, which reveals the potential of structure searching in machine learning tasks.

Configuration of GA. For our method, we spawn a group of individuals with population 150, 190, 300 in each generation for the TT variational mean of CCPP, MG and Protein regression task, respectively. Furthermore, we set the maximum number of generations, elimination rate to be 30 and 30% respectively. In addition, for these tasks, the bound of TR-ranks is set to 14, and we set $\lambda = 1 \times 10^7, 1 \times 10^7, 1 \times 10^3$, respectively. Moreover, we set the learning rate of the Adam optimizer to be 0.001 and set the reproduction number to be 1. The chance of mutation is set to be

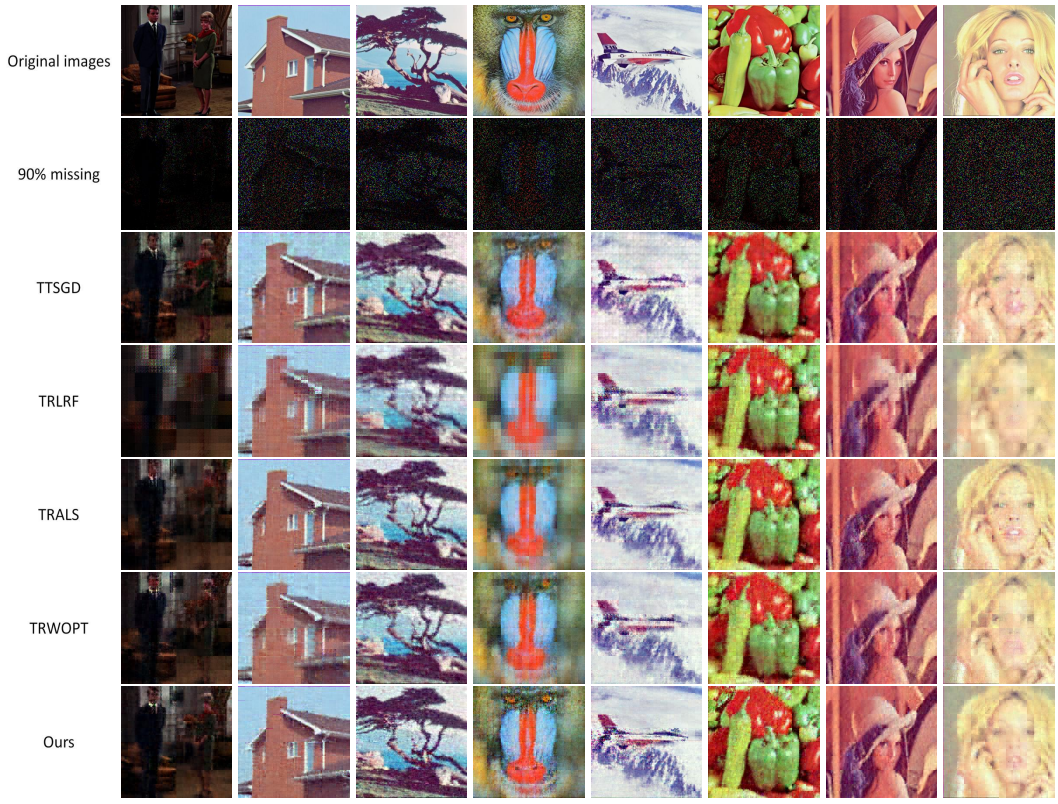

Figure 4: Visual completion results of eight color images. Original images, 90% missing images, images recovered by TTSGD, TRLRF, TRALS, TRWOPT and the proposed method are demonstrated from the top row to the bottom row correspondingly.

30%. We initialize the core tensors with Gaussian distribution of $N(0, 0.01)$, $N(0, 0.01)$, $N(0, 0.04)$, respectively. To calculate the selection probability of the recombination operation, we choose the hyper-parameter $\alpha = 20$, $\beta = 1$.

3.3 Implementation

In the experiments, we implement our GA on graphics processing unit (GPU, Nvidia® V100) clusters following a central processing unit (CPU, Intel® Xeon® E5-2690) node. Concretely, we exploit the CPU node for receiving the data, employing all genetic operators and assigning the individuals into different GPUs, which calculate the TN decomposition under given topology and output the fitness value. After the calculation for each generation, the CPU node will collect the fitness values and generates new individuals for the next generation.

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[2] Lukasz Cincio, Jacek Dziarmaga, and Marek M Rams. Multiscale entanglement renormalization ansatz in two dimensions: quantum ising model. *Physical Review Letters*, 100(24):240603, 2008.

[3] Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722, 2009.

[4] Masaaki Imaizumi, Takanori Maehara, and Kohei Hayashi. On tensor train rank minimization: Statistical efficiency and scalable algorithm. In *Advances in Neural Information Processing Systems*, pages 3930–3939, 2017.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] I Krasikov, A Lev, and BD Thatte. Upper bounds on the automorphism group of a graph. *Discrete Math.*, 256(math. CO/0609425):489–493, 2006.

[7] Maxim Kuznetsov, Daniil Polykovskiy, Dmitry P Vetrov, and Alex Zhebrak. A prior of a googol gaussians: a tensor ring induced prior for generative models. In *Advances in Neural Information Processing Systems*, pages 4104–4114, 2019.

[8] Chao Li and Zhun Sun. Evolutionary topology search for tensor network decomposition. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[9] Hai-Jun Liao, Jin-Guo Liu, Lei Wang, and Tao Xiang. Differentiable programming tensor networks. *arXiv preprint arXiv:1903.09650*, 2019.

[10] Oscar Mickelin and Sertac Karaman. On algorithms for and computing with the tensor ring decomposition. *Numerical Linear Algebra with Applications*, 27(3):e2289, 2020.

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

[12] Justin Reyes and Miles Stoudenmire. A multi-scale tensor network architecture for classification and regression. *arXiv preprint arXiv:2001.08286*, 2020.

[13] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.

[14] Zerui Tao and Qibin Zhao. Bayesian tensor ring decomposition for low rank tensor completion. In *International Workshop on Tensor Network Representations in Machine Learning, IJCAI*, 2020.

[15] Frank Verstraete and J Ignacio Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*, 2004.

[16] Maolin Wang, Zeyong Su, Xu Luo, Yu Pan, Shenggen Zheng, and Zenglin Xu. Concatenated tensor networks for deep multi-task learning. In *International Conference on Neural Information Processing*, pages 517–525. Springer, 2020.

[17] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Efficient low rank tensor ring completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5697–5705, 2017.

[18] Allan G Weber. The usc-sipi image database version 5. *USC-SIPI Report*, 315(1), 1997.

[19] Ke Ye and Lek-Heng Lim. Tensor network ranks. *arXiv preprint arXiv:1801.02662*, 2019.

[20] Longhao Yuan, Jianting Cao, Xuyang Zhao, Qiang Wu, and Qibin Zhao. Higher-dimension tensor completion via low-rank tensor ring decomposition. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1071–1076. IEEE, 2018.

[21] Longhao Yuan, Chao Li, Danilo Mandic, Jianting Cao, and Qibin Zhao. Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9151–9158, 2019.

[22] Longhao Yuan, Qibin Zhao, Lihua Gui, and Jianting Cao. High-order tensor completion via gradient-based optimization under tensor train format. *Signal Processing: Image Communication*, 73:53–61, 2019.

[23] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

[24] Yu-Bang Zheng, Ting-Zhu Huang, Xi-Le Zhao, Qibin Zhao, and Tai-Xiang Jiang. Fully-connected tensor network decomposition and its application to higher-order tensor completion. 2021.