MARS: Masked Automatic Ranks Selection in Tensor Decompositions

Maxim Kodryan
HSE University

Dmitry Kropotov
Lomonosov Moscow State University
HSE University

Dmitry Vetrov
HSE University
AIRI

Abstract

Tensor decomposition methods have proven effective in various applications, including compression and acceleration of neural networks. At the same time, the problem of determining optimal decomposition ranks, which present the crucial parameter controlling the compression-accuracy trade-off, is still acute. In this paper, we introduce MARS — a new efficient method for the automatic selection of ranks in general tensor decompositions. During training, the procedure learns binary masks over decomposition cores that "select" the optimal tensor structure. The learning is performed via relaxed maximum a posteriori (MAP) estimation in a specific Bayesian model and can be naturally embedded into the standard neural network training routine. Diverse experiments demonstrate that MARS achieves better results compared to previous works in various tasks.

1 INTRODUCTION

Tensor decomposition methods are leveraged in various areas of machine learning, such as multi-way data analysis (Cichocki et al., 2009), higher-order representation learning (Castellana and Bacciu, 2019), recommender systems (Frolov and Oseledets, 2017), and many others (Ji et al., 2019). Perhaps the most famous and perspective application of these techniques is deep neural networks (DNNs) tensorization (Cichocki et al., 2017). Decomposition methods cope with redundancy in DNNs parameterization via an efficient representation of neural network parameters as decomposed tensors. Recent works on applying tensor decomposition techniques in neural networks have demonstrated the success of this approach for compression, speed-up, and regularization of DNN models. For instance, Tucker (Tucker, 1966) and canonical polyadic (CP) (Carroll and Chang, 1970) tensor decompositions are widely used for compressing and accelerating convolutional neural networks (Lebedev et al., 2015; Kim et al., 2016; Kossaifi et al., 2019), and Tensor Train (TT) (Oseledets, 2011) decomposition has been successfully applied for tensorization of a variety of neural networks layers, like fully-connected (FC) (Novikov et al., 2015), convolutional (Garipov et al., 2016), recurrent (Yang et al., 2017; Yu et al., 2017), embedding (Hrinchuk et al., 2020), etc.

Probably the main crux of the tensorization approach is the need to carefully select decomposition hyperparameters, namely the ranks. Tensor decomposition ranks determine the shape of the core tensors in the decomposition (the cores) and hence are responsible for the trade-off between the quality of the model and the required computational and memory resources. Therefore, decomposition ranks represent extremely important hyperparameters. Yet, the problem of finding an efficient way to select optimal ranks in a general tensor decomposition automatically still remains unresolved. Typical hyperparameter selection techniques, like cross-validation, are poorly suited for the choice of multiple tensor ranks. Hence, the common practice is to set all ranks equal and validate a single hyperparameter. However, such a simplification is quite coarse and might significantly degrade model performance compared to a non-uniform ranks selection, which we empirically demonstrate in our experiments.

In this work, we present *Masked Automatic Ranks Selection* (MARS) — a new efficient method for dynamic selection of tensor decomposition ranks grounded in the Bayesian framework. The main idea is to learn binary masks that cover decomposition cores and "select" only the slices required for the best model performance, thus automatically adjusting the optimal ranks arrangement. We also propose a way to reformulate the emerged NP-hard discrete problem via scalable continuous optimization. MARS operates end-to-end with model training without introducing any noticeable additional computational overhead. In sum, our **major contributions** consist in 1) proposing a *general* scheme for automatic ranks selection and 2) developing an *efficient scalable* method for applying that scheme.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

In the experiments, we demonstrate that our method is applicable for various tensorized neural networks (TNNs) and even more general tensorized models (Appendix D). We evaluate MARS on a variety of tasks and architectures involving convolutional, fully-connected, and embedding tensorized layers and demonstrate its ability to improve previous results on tensorization in terms of compression, accuracy, and speed-up. Our code is available at `https://github.com/MaxBourdon/mars`.

1.1 Related work

Here we highlight related work on tensor rank determination with a focus on application to deep learning. We refer to Appendix A for more related literature on tensorization.

Kim et al. (2016) perform full DNN compression via approximating FC and convolutional layers with low-rank matrix factorization and Tucker-2 tensor decomposition, respectively, where ranks are estimated with a special Bayesian matrix rank selection technique (Nakajima et al., 2012). However, the involved training procedure consisting of decomposition of the pre-trained model and fine-tuning of the decomposed model turned out to be inefficient. The MUSCO algorithm (Gusak et al., 2019), which repeatedly performs decomposition and fine-tuning steps, partially resolved this disadvantage. Lately, Cheng et al. (2020) proposed a reinforcement learning-based rank selection scheme for TNNs, which, however, also introduces extra computational requirements by separating agent and model training. In contrast, MARS operates end-to-end with model training without splitting it into stages, which is naturally more preferable. Moreover, it is not confined to specific types of tensor decompositions, models, or tasks.

Existing methods for automatic rank selection that also take advantage of the Bayesian approach cover only certain types of tensor decompositions (Rai et al., 2014; Zhao et al., 2015; Xu et al., 2020, 2021; Fang et al., 2021) or are based on peculiarities of the task, e.g., tensor approximation (Mørup and Hansen, 2009) or linear regression (Guhaniyogi et al., 2017). These approaches mostly embody structured pruning of the decomposition cores. For instance, Hawkins and Zhang (2021) propose a specific shrinking coupling prior distribution over TT-cores and perform Bayesian inference to obtain Low-Rank Bayesian Tensorized Neural Networks (LR-BTNN). We, instead, propose a general-purpose ranks selection technique applicable for any tasks and decompositions.

Alternative procedures for obtaining low-rank tensor representation, e.g., those utilizing nuclear norm minimization (Phien et al., 2016; Imaizumi et al., 2017; Shi et al., 2021), also leverage properties of the particular objective and/or suggest excessively computationally complex algorithms involving a series of SVDs. This makes such approaches impracticable in the domain of deep learning. MARS does not impose any significant extra computations for obtaining a low-rank tensorized solution, since slice-wise mask multiplication is a much less computationally expensive operation than tensors contraction.

2 MARS

In this section, we introduce the necessary notions regarding tensors, decompositions, and general tensorized models and describe the details of the proposed method.

2.1 Tensors, decompositions, and tensorized models

Tensors By a *d-dimensional tensor*, we mean a multi-dimensional array $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of real numbers, e.g., vectors and matrices are 1- and 2-dimensional tensors, respectively. We denote $\mathcal{A}(i_1, \ldots, i_d)$ as element $(i_1, \ldots, i_d)$ of a tensor $\mathcal{A}$. We use notation $\dims(\mathcal{A}) = (n_1, \ldots, n_d)$ to denote the tuple of dimensions of a tensor $\mathcal{A}$.

Contraction of two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $\mathcal{B} \in \mathbb{R}^{m_1 \times \cdots \times m_{d'}}$ with $n_d = m_1$ results in a tensor $\mathcal{AB} \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times m_2 \cdots \times m_{d'}}$:

$$\mathcal{AB}(i_1, \ldots, i_{d-1}, j_2, \ldots, j_{d'}) = \\ = \sum_{i_d=1}^{n_d} \mathcal{A}(i_1, \ldots, i_d)\mathcal{B}(i_d, j_2, \ldots, j_{d'}). \quad (1)$$

The contraction operation can also be naturally generalized to multiple modes. In this case, summation in eq. (1) is performed over these modes, and dimensions of the resulting tensor will contain the dimensions of both tensors $\mathcal{A}$ and $\mathcal{B}$ excluding the contracted ones.

A special case of contraction (up to modes permutation) for a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and a matrix $B \in \mathbb{R}^{m_k \times n_k}$ is their *mode-k product* $\mathcal{A} \times_k B \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times m_k \times n_{k+1} \times \cdots \times n_d}$:

$$(\mathcal{A} \times_k B)(i_1, \ldots, i_{k-1}, j_k, i_{k+1}, \ldots, i_d) = \\ = \sum_{i_k=1}^{n_k} \mathcal{A}(i_1, \ldots, i_d)\mathcal{B}(j_k, i_k).$$

We also introduce *mode-k broadcast Hadamard product* of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and a vector $b \in \mathbb{R}^{n_k}$ which is a tensor $\mathcal{A} \odot_k b$ with the same dimensions as $\mathcal{A}$ and elements

$$(\mathcal{A} \odot_k b)(i_1, \ldots, i_d) = \mathcal{A}(i_1, \ldots, i_d)b(i_k).$$

Tensor decompositions In general, we assume that tensor decomposition of a d-dimensional tensor $\mathcal{A}$ consists of a set of simpler tensors $\boldsymbol{G} = \{\mathcal{G}_k\}$ called *cores* of the decomposition. The original tensor can be expressed (up to modes permutation) via these cores as a sequence of contractions.

For the Tensor Train decomposition $\boldsymbol{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_d\}$, $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, $r_0 = r_d = 1$ and

$$\mathcal{A} = \mathcal{G}_1 \mathcal{G}_2 \dots \mathcal{G}_d,$$

i.e., tensor $\mathcal{A}$ is directly obtained from the Tensor Train cores as a sequence of contractions.

For the Tucker decomposition $\boldsymbol{G} = \{\mathcal{G}, U_1, \dots, U_d\}$, $U_k \in \mathbb{R}^{n_k \times r_k}$, $\mathcal{G} \in \mathbb{R}^{r_1 \times \dots \times r_d}$ and

$$\mathcal{A} = \mathcal{G} \times_1 U_1 \dots \times_d U_d,$$

i.e., tensor $\mathcal{A}$ is expressed via mode-k products of the core tensor $\mathcal{G}$ and matrices U_k which is again a sequence of contractions up to modes permutation.

The set of numbers $\boldsymbol{r} = \{r_k\}$, the intermediate dimensions of the contracted cores modes, are called *ranks* of the decomposition. Clearly, they define the expressivity of the decomposition on the one hand and the number of the occupied parameters on the other.

Tensorized models Consider any model parameterized by a tensor $\mathcal{A}$ decomposed into cores $\boldsymbol{G}$.[1] In practice, it is often convenient (in terms of memory and computations) to utilize tensors in the decomposed format explicitly. In other words, given a particular decomposition, one could rewrite model operations more efficiently via the cores $\boldsymbol{G}$ directly, without the need of reconstructing the full tensor $\mathcal{A}$. Hence, a single large parameter tensor can be substituted with a set of smaller tensors to obtain a more compact model. We call such models, parameterized by the cores of the decomposed tensors, *tensorized models* and assume that their inference is performed directly via these cores.

A typical example of a tensorized model is a neural network with decomposed layers, or tensorized neural network. Representing layer parameters via a decomposed tensor may result in substantial memory and computational savings. For most NN layers, there is a variety of decomposed representations: factorized FC-layer, Tucker-2 convolutional layer, various TT-layers, etc. We provide more detail in Appendix B.

Ultimately, in a tensorized model, the shapes of the decomposition cores simultaneously influence model flexibility and complexity. The key hyperparameter that determines them are decomposition ranks, as discussed earlier. Further, we describe the details of the proposed method for ranks selection in arbitrary tensorized models.

2.2 The proposed method

Consider a predictive tensorized model, which defines a distribution over output y conditioned on input x, with cores $\boldsymbol{G}$: $p(y \mid x, \boldsymbol{G})$. We assume that the initial shapes of the cores (i.e., ranks $\boldsymbol{r}$) are fixed in advance. Our goal is to shrink them optimally: remove redundant ranks without significant accuracy drop to achieve maximum compression and speed-up.

MARS suggests obtaining such reduced structures via multiplying slices of the cores by binary masking vectors, mostly consisting of zeros. Zeroed slices are not be involved in tensors contractions and, therefore, the whole model workflow. Hence, such slices can be freely removed from the cores. In this way, non-zero masks elements would "select" only slices required for the effectual model performance, automatically determining the optimal cores shapes. Figure 1 illustrates the concept.

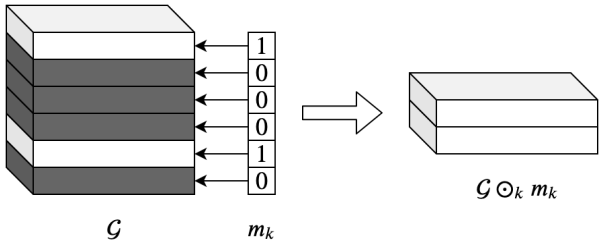

Figure 1: A schematic illustration of the MARS concept: slices of the core tensor $\mathcal{G}$ along mode k are multiplied by elements of the binary mask m_k; only "selected" non-zero slices will participate in model inference, therefore, the core shape can be reduced.

Formally, given a dataset $(X, Y) = \{(x_i, y_i)\}_{i=1}^N$, consider the following discriminative Bayesian model:[2]

$$p(Y, \boldsymbol{m}, \boldsymbol{G} \mid X) = \prod_{i=1}^{N} p(y_i \mid x_i, \boldsymbol{G} \odot \boldsymbol{m})\, p(\boldsymbol{m})\, p(\boldsymbol{G}), \tag{2}$$

where $\boldsymbol{m} = \{m_k \mid m_k \in \{0,1\}^{r_k}\}$ is a set of binary vectors, or *masks*, one-to-one corresponding to the decomposition ranks, $\boldsymbol{G} \odot \boldsymbol{m} = \{\mathcal{G}_k \odot \boldsymbol{m}\}$ is a set of *masked cores*:

$$\mathcal{G}_k \odot \boldsymbol{m} \coloneqq \mathcal{G}_k \odot_{k_1} m_{k_1} \dots \odot_{k_p} m_{k_p},$$

$r_{k_1}, \dots, r_{k_p} \in \dim(\mathcal{G}_k)$ are all ranks belonging to the dimensions of the core $\mathcal{G}_k$. The likelihood $p(y \mid x, \boldsymbol{G} \odot \boldsymbol{m})$ is defined by the tensorized model, so, for example, it can be (exponent of negative) cross-entropy loss for classification tasks. Note that in model (2), we completely remove the dependency between cores and masks unlike, e.g., the widely known spike-and-slab prior model (Ishwaran and Rao, 2005). This allows to ensure universality of our approach by accepting arbitrary couplings of the cores in general decompositions.

[1] For simplicity, we consider a single-tensor model, though the same applies to models with multiple tensors.

[2] The presented model can be straightforwardly extended to other non-discriminative settings even with no available data points, see Appendix D for a concrete example.

We assume the factorized Bernoulli prior over masks with the success probability π, which is a natural hyperparameter of our model regulating the intensity of compression:

$$p(\boldsymbol{m}) = p(\boldsymbol{m} \mid \pi) = \prod_k \prod_{s=1}^{r_k} \pi^{m_k(s)}(1-\pi)^{1-m_k(s)}. \quad (3)$$

This prior term is the key ingredient that allows achieving low-rank solutions as it sparsifies the selection masks. Note that instead of adjusting several or even dozens of ranks in the decomposition, one needs to validate only one hyperparameter in our model (along with careful initialization). Furthermore, in our experiments, we found that π does influence the final compression-accuracy trade-off but not crucially (see Appendix E). Taking $\pi \approx 10^{-2}$ is usually a good choice. Note that setting π to values more than 0.5 is of no use, as therefore, the prior term would foster dense masks, which contradicts with obtaining a low-rank solution corresponding to mostly zero-valued masks.

We also put the factorized zero-mean Gaussian with a large variance as the prior distribution over the cores $p(\boldsymbol{G})$. It serves as a slight L_2 regularization and we have empirically found that it helps to balance the coefficients in the cores, stabilize the training process and improve test accuracy. We did not conduct a thorough search for the optimal values of the prior variance and used the same fixed value of 10^2 in all our experiments.

In this work, we consider finding *maximum a posteriori* (MAP) estimates of parameters $\boldsymbol{G}$ and $\boldsymbol{m}$ in model (2):

$$\sum_{i=1}^{N} \log p(y_i \mid x_i, \boldsymbol{G} \odot \boldsymbol{m}) + \log p(\boldsymbol{m}) + \\ + \log p(\boldsymbol{G}) \longrightarrow \max_{\boldsymbol{m}, \boldsymbol{G}}. \quad (4)$$

Naturally, this problem implies discrete optimization over binary masks and, hence, is infeasible due to exhaustive search in the general case. To tackle this, we first substitute problem (4) with an equivalent:

$$\mathbb{E}_{\boldsymbol{m} \sim q(\boldsymbol{m})} \left[\sum_{i=1}^{N} \log p(y_i \mid x_i, \boldsymbol{G} \odot \boldsymbol{m}) + \log p(\boldsymbol{m}) \right] + \\ + \log p(\boldsymbol{G}) \longrightarrow \max_{q(\boldsymbol{m}), \boldsymbol{G}}, \quad (5)$$

where the family of distributions $q(\boldsymbol{m})$ includes all deterministic ones, i.e., taking only a single value. The solutions of problems (4) and (5) coincide according to Lemma 1.

Lemma 1 *For an arbitrary scalar function $F(x)$ with attainable maximum the following problems are equivalent:*

$$\max_x F(x) \equiv \max_{q(x)} \mathbb{E}_{x \sim q(x)} [F(x)]$$

if the family of distributions $q(x)$ includes degenerate ones.

Proof. The statement follows from the fact that for *any* distribution $q(x)$ we have

$$\mathbb{E}_{x \sim q(x)} F(x) \leq F(x^*), \quad (6)$$

where $x^* = \operatorname{argmax}_x F(x)$, and eq. (6) turns into equality when $q(x) = \delta(x - x^*)$. ■

Next, we constrain $q(\boldsymbol{m})$ to be a factorized Bernoulli distribution over each mask element $m_k(s)$ with parameters $\boldsymbol{\phi} = \{\phi_k(s)\}$. Note that this family meets the requirement to include all degenerate solutions in order to ensure equivalence of eq. (4) and eq. (5). Now the problem (5) translates into the following:

$$\mathbb{E}_{\boldsymbol{m} \sim q_{\boldsymbol{\phi}}(\boldsymbol{m})} \left[\sum_{i=1}^{N} \log p(y_i \mid x_i, \boldsymbol{G} \odot \boldsymbol{m}) \right] + \\ + \sum_k \sum_{s=1}^{r_k} [\phi_k(s) \log \pi + (1 - \phi_k(s)) \log(1-\pi)] + \\ + \log p(\boldsymbol{G}) \longrightarrow \max_{\boldsymbol{\phi}, \boldsymbol{G}}. \quad (7)$$

One can notice that adding the q entropy term into eq. (7) yields the evidence lower bound (ELBO) maximization, a well-known Bayesian technique for the variational posterior approximation, with a factorized Bernoulli variational distribution. However, we *do not* perform variational inference with MARS, but look for a single-point solution instead. We discuss this further at the end of the article.

We solve the maximization problem (7) with the stochastic gradient method. To calculate low-variance stochastic gradients w.r.t. parameters $\boldsymbol{\phi}$ in eq. (7), we use the *reparameterization trick* (Kingma and Welling, 2013). To this end, we soften the discrete samples from $q_{\boldsymbol{\phi}}(\boldsymbol{m})$ in the expectation term via the Binary Concrete relaxation (Maddison et al., 2017; Jang et al., 2017) with temperature, which defines "discreteness" of the relaxed samples, decaying to zero in the course of training. After training, we round the probabilities $\boldsymbol{\phi}$ to binary masks $\boldsymbol{m}_{MAP}$ and use a compact solution with reduced cores $\boldsymbol{G}_{MAP} \odot \boldsymbol{m}_{MAP}$ to predict for a new data sample x^*: $p(y^* \mid x^*, \boldsymbol{G}_{MAP} \odot \boldsymbol{m}_{MAP})$.

Algorithm 1 summarizes the training procedure. $RB(\phi, \tau)$ denotes the Relaxed Bernoulli distribution, which is Binary Concrete with temperature τ and location $\frac{\phi}{1-\phi}$. Sampling from $RB(\phi, \tau)$ is as simple as applying a differentiable operation

$$\sigma\left(\frac{\log(u) - \log(1-u) + \log(\phi) - \log(1-\phi)}{\tau}\right)$$

to $u \sim \mathcal{U}_{[0,1]}$, where $\sigma(x) = 1/(1+e^{-x})$ is a logistic sigmoid function. Note that Algorithm 1 is essentially an SGD on a regularized loss, therefore they have similar computational complexity; it can be considered extremely lightweight compared to other (e.g., SVD-based) rank selection schemes.

Algorithm 1 MARS relaxed MAP learning procedure

Input: data (X, Y), prior parameter π, temperature τ, batch size B
Output: MAP estimate of cores $\boldsymbol{G}_{MAP}$ and masks $\boldsymbol{m}_{MAP}$
Initialize $\boldsymbol{G}$ and $\boldsymbol{\phi}$
repeat
  Sample masks $\hat{\boldsymbol{m}} = \{\hat{m}_k(s) \sim RB(\phi_k(s), \tau)\}$
  Sample a mini-batch of objects $\{(x_{i_l}, y_{i_l})\}_{l=1}^{B}$
  $L \coloneqq \sum_{l=1}^{B} \log p\left(y_{i_l} \mid x_{i_l}, \boldsymbol{G} \odot \hat{\boldsymbol{m}}\right)$
  $g_{\boldsymbol{\phi}} \coloneqq \frac{\partial L}{\partial \boldsymbol{G} \odot \hat{\boldsymbol{m}}} \frac{\partial \boldsymbol{G} \odot \hat{\boldsymbol{m}}}{\partial \hat{\boldsymbol{m}}} \frac{\partial \hat{\boldsymbol{m}}}{\partial \boldsymbol{\phi}} + \log\left(\frac{\pi}{1-\pi}\right)$
  $g_{\boldsymbol{G}} \coloneqq \frac{\partial L}{\partial \boldsymbol{G} \odot \hat{\boldsymbol{m}}} \frac{\partial \boldsymbol{G} \odot \hat{\boldsymbol{m}}}{\partial \boldsymbol{G}} + \frac{\partial \log p(\boldsymbol{G})}{\partial \boldsymbol{G}}$
  Update $\boldsymbol{\phi}$ using stochastic gradient $g_{\boldsymbol{\phi}}$
  Update $\boldsymbol{G}$ using stochastic gradient $g_{\boldsymbol{G}}$
  Decay τ
until stop criterion is met
Define $\boldsymbol{G}_{MAP} \coloneqq \boldsymbol{G}$
Define $\boldsymbol{m}_{MAP} \coloneqq round(\boldsymbol{\phi})$

3 EXPERIMENTS

In this section, we present the results of the conducted experiments with MARS that show its ability to improve previous results on DNN tensorization. In Appendix D, we also provide additional experiments involving a different tensorized model to demonstrate that MARS is a general ranks selection scheme not confined to TNNs only.

We train tensorized models using MARS according to Algorithm 1. The learned hard binary masks are then applied to the trained cores to remove excess ranks and obtain a compact architecture that is used during test-time inference. Careful parameter initialization is required for optimal performance and training. We propose to initialize logits of $\boldsymbol{\phi}$ using the normal distribution centered at some value α, which is an important hyperparameter with a role similar to that of hyperparameter π. We refer to Appendix C for more details on implementation.

Our experiments are conducted in three ways to prove the efficiency and versatility of MARS. First, we show how MARS can restore the actual ranks when the ground truth (GT) is known (Section 3.1, Appendix D). Second, we compare with alternative approaches for Bayesian rank selection (Sections 3.2, 3.4 and 3.5) to show that MARS can perform just as well, being a more general method. Third, we apply MARS to various practical tensorized models to show that it can significantly improve previous results where ranks were mostly selected using cross-validation (Sections 3.3 and 3.4). In tables, we report mean $\pm$ std where applicable.

Table 1: True rank restoring with MARS in a toy linear classification task with a factorized parameter matrix. Results are averaged over 10 runs.

GT rank r^*	Reduced R	Accuracy	Baseline
8	8.4 ± 0.5	$\mathbf{91.8 \pm 0.6}\%$	87.3%
12	12.6 ± 0.7	$\mathbf{89.5 \pm 0.7}\%$	85%
16	18 ± 1.3	$\mathbf{85.4 \pm 0.7}\%$	82.8%

3.1 Toy experiment

Our first experiment serves as a mere proof of concept. We evaluate our method on a toy linear classification task with a factorized parameter matrix to verify how MARS can approximate the true rank.

Let N, D, C, r^*, R denote, respectively, the number of samples, input, output dimensions, the ground truth rank of the problem, and the initial rank of the parameters to be reduced. At first, we sample elements of the input matrix $X \in \mathbb{R}^{N \times D}$ i.i.d. from the standard normal distribution. We similarly sample the ground truth parameter matrices $U^* \in \mathbb{R}^{D \times r^*}$ and $V^* \in \mathbb{R}^{r^* \times C}$. After that, we obtain the output matrix $Y = \mathrm{OHE}\left(XU^*V^*\right)$ of size (N, C), where OHE denotes row-wise argmax one-hot encoding operation. Finally, we initialize the learnable parameter matrices $U \in \mathbb{R}^{D \times R}$ and $V \in \mathbb{R}^{R \times C}$ and train a linear classifier with a factorized parameter matrix UV using MARS to reduce the initial rank $R > r^*$ and restore the GT rank r^*.

We fixed $N = 10000$, $D = 128$, $C = 32$, $R = 32$ and varied r^*. Namely, we considered three cases: $r^* = 8$, $r^* = 12$, and $r^* = 16$. We took $\pi = 10^{-2}$ and $\alpha = -4$, $\alpha = -3.5$, and $\alpha = -3$ for each case, respectively. We evaluated models on a separate test set and trained a vanilla linear classifier as a baseline.[3]

We report the results averaged over 10 runs in Table 1. As can be seen, MARS can rather accurately restore the true rank r^* starting from a higher initial value R. Furthermore, the obtained models are not only more compact but also exhibit better test accuracy than the baseline.

As neural network training encounters a manifold of different local minima, the problem of revealing the "true rank" of a tensorized DNN, rather than a simple linear model, is ill-posed: it highly depends on initialization, optimization, and hyperparameters. Yet, that can be especially useful for *ensembling*, which will be discussed further.

3.2 MNIST 2FC-Net

In this experiment, we compare MARS against the LR-BTNN method of Hawkins and Zhang (2021) on the

[3] We have also tried a factorized baseline with $R = 32$ but obtained much worse results and decided not to include it.

MNIST (LeCun, 1998) dataset. In this task, both methods aim to automatically select ranks in a relatively small image classification neural network with two TT-decomposed fully-connected layers of sizes 784×625 and 625×10. As proposed by Hawkins and Zhang (2021), we take the following dimensions factorization of the TT-layers: $(n_1, n_2, n_3, n_4) = (7, 4, 7, 4)$, $(m_1, m_2, m_3, m_4) = (5, 5, 5, 5)$ and $(n_1, n_2) = (25, 25)$, $(m_1, m_2) = (5, 2)$ for the first and second layer, respectively. All the initial ranks are set to 20, which gives $18\times$ compression at the start.

Table 2: Compression/accuracy on MNIST with 2FC-Net. Results are averaged over 10 runs.

Model	Compression	Accuracy
Baseline	$1\times$	$\mathbf{98.2}\%$
Baseline-TT	$18\times$	97.7%
LR-BTNN	$137\times$	97.8%
MARS (soft)	$141 \pm 18.6\times$	$\mathbf{98.2 \pm 0.11}\%$
MARS (hard)	$\mathbf{205 \pm 30.9}\times$	$97.9 \pm 0.19\%$

We execute MARS in two modes for this task: soft compression mode ($\alpha = -1.5$, $\pi = 10^{-1}$) and hard compression mode ($\alpha = -1.75$, $\pi = 10^{-2}$). In each mode, we trained 10 networks from different random initializations and averaged the results. Table 2 shows that MARS surpasses the approach of Hawkins and Zhang (2021) in this task both in terms of compression and final accuracy, even though LR-BTNN is specifically tailored for the Tensor Train decomposition.

We would also like to note that an ensemble of small MAP networks, obtained in the soft compression mode, gives the accuracy of $\mathbf{98.9}\%$. We argue that compact TNN ensembling might be a promising research direction.

Figure 2 shows the bar plot of ϕ values of the three masks corresponding to the first TT-layer. We see that the relaxed MAP estimate is actually quite close to the deterministic binary masks. After rounding to strictly binary values and applying the resulted masks to the TT-cores, the ranks of the first layer shrink to $(r_0, r_1, r_2, r_3, r_4) = (1, 4, 3, 4, 1)$ which leads to more than $556\times$ layer compression.

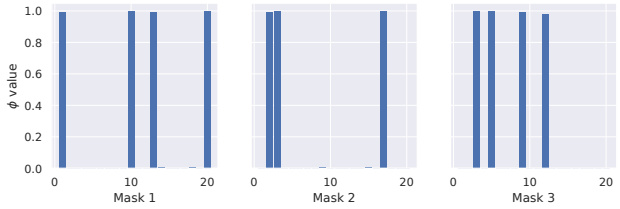

Figure 2: Learned binary masks probabilities ϕ corresponding to the first TT-layer in MNIST 2FC-Net. Note that the relaxed MARS MAP estimate is quite close to the deterministic solution.

3.3 Sentiment analysis with TT-embeddings

A recent work of Hrinchuk et al. (2020) leverage Tensor Train decomposition for compressing embedding layers in various NLP models. The authors propose to convert the matrix of embeddings into the TT-format alike TT-FC layers. They provide a heuristic to automatically determine optimal (in terms of the occupied memory) factorization of dimensions in TT-matrices given the number of factors d. However, in their experiments, the ranks in the TT-decomposition were still manually set equal to some predefined value.

We repeat their experiment on the sentiment analysis task and apply MARS on top of the tensorized model. The model consists of a TT-embedding layer with ranks equal to 16, followed by an LSTM, which performs sentiment classification. The authors evaluated on two datasets: IMDB (Maas et al., 2011) and Stanford Sentiment Treebank (SST) (Socher et al., 2013). On each dataset they tried three tensorized models: with $d = 3$, $d = 4$ and $d = 6$ factors in the TT-matrix of embeddings, respectively. On IMDB, the authors obtained both maximal accuracy and compression with the model using $d = 6$ factors. On SST, the best compression was achieved at $d = 6$, while the best accuracy was achieved at $d = 3$. Thus, we choose the best model on IMDB and the medium one ($d = 4$) on SST and train them with MARS. We set $\pi = 10^{-2}$ in both models and $\alpha = -0.25$, $\alpha = -1.0$ for the first and the second one, respectively.

Table 3 presents the obtained results. Automatic ranks selection with MARS allowed to significantly improve both quality and compression of the best IMDB TT-model. On SST, we managed to overtake the best compressing and best performing models with a medium model trained using our method. The final selected ranks are $(r_0, r_1, r_2, r_3, r_4, r_5, r_6) = (1, 8, 11, 15, 16, 16, 1)$ and $(r_0, r_1, r_2, r_3, r_4) = (1, 6, 14, 14, 1)$ for IMDB and SST MARS TT-models, respectively. We hypothesize that such an escalating rank distribution could be explained by the hierarchical indexing in TT-embeddings, where first TT-cores are responsible for indexing large blocks in the embedding matrix, and subsequent cores index inside those blocks. The compressed model might find only a few large blocks in the whole embedding matrix relevant for prediction, thus, the first cores could be made less expressive. On the whole, one can see that setting decomposition ranks equal, which is a common heuristics in tensorized networks, is quite inefficient as opposed to the nonuniform ranks selection.

3.4 MNIST LeNet-5

In Wang et al. (2018) Tensor Ring (TR) decomposition (Zhao et al., 2016), a generalization of the Tensor Train decomposition, was applied to compress convolutional net-

Table 3: Compression and accuracy on sentiment analysis with TT-embedding layers. TT-d denotes TT-embedding with d factors.

Dataset	Model	Compression	Accuracy
IMDB	Baseline	$1\times$	88.6%
	TT-6	$441\times$	89.7%
	MARS + TT-6	$\mathbf{559\times}$	$\mathbf{90.1\%}$
SST	Baseline	$1\times$	37.4%
	TT-3	$78\times$	41.5%
	TT-6	$307\times$	39.9%
	MARS + TT-4	$\mathbf{340\times}$	$\mathbf{42.4\%}$

works. Such neural networks with TR-decomposed convolutions and FC-layers are called Tensor Ring Nets (TRNs). The authors compared their approach against Kim et al. (2016), where Tucker-2 and low-rank matrix factorization, which represent a simpler decomposition family, are used for the same purpose. In one of the experiments, both methods were evaluated on compressing and accelerating LeNet-5 (LeCun et al., 1998), a relatively small convolutional neural network with 2 convolutional layers, followed by 2 fully-connected layers, on the MNIST dataset. TRN significantly surpassed the simpler Tucker approach.

In this experiment, we demonstrate that even using less expressive types of decompositions, one can achieve results comparable to TRN by training with MARS. Namely, we apply Tucker-2 decomposition to the second convolution and low-rank factorization to the first FC-layer, as the other layers occupy less than 1.3% of all model parameters. We automatically select the two Tucker-2 decomposition ranks r_1, r_2 and the matrix rank r_3 using our method, starting from $r_1 = r_2 = 20$, $r_3 = 100$ ($2.9\times$ compression at the start). We initialize the mean value of $\boldsymbol{\phi}$ logits α with zero and set $\pi = 10^{-2}$.

The averaged results over 5 runs are presented in Table 4. MARS enhanced compression of the Tucker model by a factor of 5 with about the same quality, making it comparable to TRN, which is based on a significantly more complex decomposition family. We would like to note that the Tucker model already has an inner mechanism of ranks selection, yet, it can only approximate the ranks required for the layers decomposition, after which the model is fine-tuned. MARS performs end-to-end ranks selection with model training, which leads to significantly better results.

Another important achievement of our model is the ability to actually accelerate networks.[4] Even though TR-decomposition allows achieving better compression, it, however, slows down the inference. The authors argue that such an effect is caused by the suboptimality of the existing hard- and software for tensor routines. Using simpler layer factorizations, we managed to speed up LeNet-5 by $1.2\times$.

Similarly to the previous experiment, we measured the quality of the ensemble of LeNet-5 networks compressed with MARS. Ensembling aids to improve model test accuracy up to $\mathbf{99.5\%}$. Note that the ensemble of 5 networks compressed by $10\times$ still requires twice less memory than the original model and, provided parallel computing, can even work faster.

We recognize the power of the Tensor Ring decomposition in compressing neural networks. Since in TRN all decomposition ranks are set equally, we believe that MARS could further improve its results and leave it for future work.

3.5 CIFAR-10 ResNet-110

The main experiment of Hawkins and Zhang (2021) consisted in applying their LR-BTNN method to ResNet-110 (He et al., 2016) on CIFAR-10 dataset (Krizhevsky et al., 2014). The authors used Tensor Train decomposition for compressing all convolutional layers except for the first ResNet block (first 36 layers) and the 1×1 convolutions.

However, they implemented a simplified scheme of decomposing convolutions, which we call *naive*. At first, the numbers of input and output channels N and M are factored into $N = \prod_{k=1}^{d} n_k$, $M = \prod_{k=1}^{d} m_k$. After that, the 4-dimensional convolutional kernel with kernel size k is reshaped into a $(2d+1)$-way tensor with dimensions $(n_1, \ldots, n_d, m_1, \ldots, m_d, k^2)$. The reshaped tensor is then decomposed into Tensor Train with $2d + 1$ cores. Such a scheme could be fruitful in terms of compression, yet it does not have a potential for efficient computing due to the need to construct the full convolutional tensor from the TT-cores on each forward pass. Unlike this method, Garipov et al. (2016) proposed to represent convolutions as $k^2N \times M$ matrices in TT-format based on the fact that most frameworks reduce the convolution operation to a matrix-by-matrix multiplication. We call the scheme of Garipov et al. (2016) *proper*. This approach, for instance, was leveraged to achieve more than $4\times$ better energy efficiency and $5\times$ acceleration compared to state-of-the-art solutions on a special TT-optimized hardware (Deng et al., 2019).

We repeat the ResNet experiment of Hawkins and Zhang (2021) with MARS using both naive and proper schemes for TT-decomposition of convolutions. The original paper does not provide much detail on the experiment setting, however, we could deduce that the authors used $d = 2$ and $d = 3$ factors for the second and third ResNet block, respectively, i.e., in the second block, they reshaped convolutional kernels from $(32, 32, 3, 3)$ to $(8, 4, 8, 4, 9)$ and in the third one from $(64, 64, 3, 3)$ to $(4, 4, 4, 4, 4, 4, 9)$. In order to obtain similar number of TT-cores for the proper scheme,

[4] We do not specify run times in other experiments, as this information is not provided in the related works; however, we have observed acceleration in other cases as well.

Table 4: Compression, accuracy, and speed-up on MNIST with LeNet-5. TRN-r denotes the TRN model with the same Tensor Ring rank r. Speed-up is evaluated as the ratio of test time per 10000 samples of the baseline and the given model, as proposed in Wang et al. (2018). Results are averaged over 5 runs.

Model	Compression	Accuracy	Speed-up
Baseline	$1\times$	99.2%	$1.0\times$
Tucker	$2\times$	99.1%	$0.58\times$
TRN-10	$\mathbf{39\times}$	98.6%	$0.48\times$
TRN-15	$18\times$	99.2%	$0.97\times$
TRN-20	$11\times$	$\mathbf{99.3\%}$	$0.73\times$
MARS + Tucker	$10 \pm 0.8\times$	$99.0 \pm 0.07\%$	$\mathbf{1.19 \pm 0.01\times}$

we choose the following respective shapes of convolutional TT-matrices: $(2, 2) \times (3, 2) \times (3, 2) \times (4, 2) \times (4, 2)$ and $(2, 2) \times (2, 2) \times (3, 2) \times (3, 2) \times (4, 2) \times (4, 2)$. At the start, all ranks equal 20, which gives $2.7\times$ and $2.3\times$ compression of the naive and proper models, respectively. We set $\pi = 10^{-2}$, $\alpha = 2.25$ and $\pi = 4 \cdot 10^{-3}$, $\alpha = 3.0$ in those models, respectively.

The results are given in Table 5. Using the naive scheme, MARS achieved the results comparable to LR-BTNN: it performed slightly worse in compression but better in accuracy. Proper TT-decomposition of convolutions and training with MARS allowed to reach the same quality as with the naively decomposed baseline TT-model but at a significantly higher compression ratio, which once again emphasizes the efficiency of the Garipov et al. (2016) scheme and nonuniform rank distribution in tensorized models.

Table 5: Compression and accuracy on CIFAR-10 with ResNet-110. We put the type of the used decomposition scheme in parentheses.

Model	Compression	Accuracy
Baseline	$1\times$	$\mathbf{92.6\%}$
Baseline (naive)	$2.7\times$	91.1%
LR-BTNN (naive)	$\mathbf{7.4\times}$	90.4%
MARS (naive)	$7.0\times$	90.7%
MARS (proper)	$5.5\times$	91.1%

4 CONCLUSION AND DISCUSSION

In this paper, we present MARS, the method for efficient automatic selection of ranks in tensorized models leveraging arbitrary tensor decompositions. The basic principle of MARS is learning binary masks along with overall model training that cover the cores of the decomposition and automatically select the optimal structure. We perform learning of masks and model parameters via relaxed MAP estimation in a special Bayesian probabilistic model. The experiments demonstrate that our technique can improve the accuracy and compression of tensorized models with manually selected ranks and surpasses or performs comparably with alternative rank selection methods specialized on concrete types of tensor decompositions.

It is widely known that the ensembling of deep neural networks leads to significant quality improvement (Lakshminarayanan et al., 2017; Lobacheva et al., 2020). In our experiments, we observed a similar trend with ensembles of compact MARS-trained networks. However, usual DNN ensembles require training and evaluating several neural networks, which might be inapplicable in resource-constrained environments. By contrast, the whole ensemble of tensorized networks often occupies less memory than a single standard model. This opens a very promising prospect for future research.

MARS obtains a single MAP estimate of masks. However, learning the variational distribution over binary masks or sampling from the posterior could allow efficient ensembling of compact tensorized models. We noted in Section 2 that our objective (7) resembles ELBO up to the entropy term. Unfortunately, our preliminary experiments in variational inference with factorized Bernoulli $q_{\phi}(\boldsymbol{m})$ led to distributions with overly low variance. In other words, sampling from $q_{\phi}(\boldsymbol{m})$ did not improve accuracy compared to the model spawned by its mode. This might mean that fully-factorized Bernoulli distribution cannot appropriately approximate the true posterior due to numerous correlations between mask variables. However, it is quite effective for finding the MAP estimate. We believe that more flexible variational families, e.g., those based on hierarchical models, may better approximate the posterior, and leave it for future study.

Other research directions may include improvements of the model and the learning method, for instance, trying REINFORCE-like algorithms (Williams, 1992) for optimization over discrete masks. Applying MARS to other types of tensor decompositions (like TR-decomposition) and tensorized models is also very intriguing.

In Appendix F, we provide more discussion on the limitations of this research, its perspectives, and possible societal impact.

Acknowledgements

We would like to thank Artem Grachev for the valuable discussions on the original idea of selecting ranks in tensor decompositions. We are also very grateful to the reviewers for their constructive and helpful comments on the submitted manuscript. The publication was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University №70-2021-00139.

References

Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319.

Castellana, D. and Bacciu, D. (2019). Bayesian tensor factorisation for bottom-up hidden tree markov models. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Cheng, Z., Li, B., Fan, Y., and Bao, Y. (2020). A novel rank selection scheme in tensor ring decomposition based on reinforcement learning for deep neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3292–3296. IEEE.

Cichocki, A., Phan, A., Oseledets, I., Zhao, Q., Sugiyama, M., Lee, N., and Mandic, D. (2017). Tensor networks for dimensionality reduction and large-scale optimizations: Part 2 applications and future perspectives. *Foundations and Trends in Machine Learning*, 9(6):431–673.

Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons.

De Lathauwer, L. (2008). Decompositions of a higher-order tensor in block terms—part II: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066.

Deng, C., Sun, F., Qian, X., Lin, J., Wang, Z., and Yuan, B. (2019). Tie: energy-efficient tensor train-based inference engine for deep neural network. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 264–278.

Fang, S., Kirby, R. M., and Zhe, S. (2021). Bayesian streaming sparse tucker decomposition. In *Uncertainty in Artificial Intelligence*, pages 558–567. PMLR.

Frolov, E. and Oseledets, I. (2017). Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3):e1201.

Garipov, T., Podoprikhin, D., Novikov, A., and Vetrov, D. P. (2016). Ultimate tensorization: compressing convolutional and FC layers alike. *CoRR*, abs/1611.03214.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

Guhaniyogi, R., Qamar, S., and Dunson, D. B. (2017). Bayesian tensor regression. *The Journal of Machine Learning Research*, 18(1):2733–2763.

Gusak, J., Kholiavchenko, M., Ponomarev, E., Markeeva, L., Blagoveschensky, P., Cichocki, A., and Oseledets, I. (2019). Automated multi-stage compression of neural networks. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 2501–2508. IEEE Computer Society.

Hashemizadeh, M., Liu, M., Miller, J., and Rabusseau, G. (2020). Adaptive tensor learning with tensor networks. *arXiv preprint arXiv:2008.05437*.

Hawkins, C. and Zhang, Z. (2021). Bayesian tensorized neural networks with automatic rank selection. *Neurocomputing*, 453:172–180.

Hayashi, K., Yamaguchi, T., Sugawara, Y., and Maeda, S.-i. (2019). Exploring unexplored tensor network decompositions for convolutional neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 5552–5562. Curran Associates, Inc.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Hrinchuk, O., Khrulkov, V., Mirvakhabova, L., Orlova, E., and Oseledets, I. (2020). Tensorized embedding layers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4847–4860.

Imaizumi, M., Maehara, T., and Hayashi, K. (2017). On Tensor Train rank minimization : Statistical efficiency and scalable algorithm. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3933–3942.

Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of statistics*, 33(2):730–773.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations*.

Ji, Y., Wang, Q., Li, X., and Liu, J. (2019). A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990.

Khrulkov, V., Hrinchuk, O., Mirvakhabova, L., Orlova, E., and Oseledets, I. (2019). Tensorized Embedding Layers For Efficient Model Compression. *arXiv preprint arXiv:1901.10787*.

Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. (2016). Compression of deep convolutional neural networks for fast and low power mobile applications. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kossaifi, J., Bulat, A., Tzimiropoulos, G., and Pantic, M. (2019). T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7822–7831.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I. V., and Lempitsky, V. S. (2015). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

LeCun, Y. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, C. and Sun, Z. (2020). Evolutionary topology search for tensor network decomposition. In *International Conference on Machine Learning*, pages 5947–5957. PMLR.

Li, C., Zeng, J., Tao, Z., and Zhao, Q. (2022). Permutation search of tensor network structures via local sampling. In *International Conference on Machine Learning*, pages 13106–13124. PMLR.

Li, C. and Zhao, Q. (2021). Is rank minimization of the essence to learn tensor network structure? *Second Workshop on Quantum Tensor Networks in Machine Learning (QTNML), 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*.

Lobacheva, E., Chirkova, N., Kodryan, M., and Vetrov, D. P. (2020). On power laws in deep ensembles. *Advances in Neural Information Processing Systems*, 33:2375–2385.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Ma, X., Zhang, P., Zhang, S., Duan, N., Hou, Y., Zhou, M., and Song, D. (2019). A tensorized transformer for language modeling. In *Advances in Neural Information Processing Systems*, pages 2229–2239.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Maddison, C., Mnih, A., and Teh, Y. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Mørup, M. and Hansen, L. K. (2009). Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363.

Nakajima, S., Tomioka, R., Sugiyama, M., and Babacan, S. D. (2012). Perfect dimensionality recovery by variational Bayesian PCA. In *Advances in Neural Information Processing Systems*, pages 971–979.

Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. (2015). Tensorizing neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 442–450.

Oseledets, I. V. (2011). Tensor-Train decomposition. *SIAM J. Scientific Computing*, 33(5):2295–2317.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Phien, H. N., Tuan, H. D., Bengua, J. A., and Do, M. N. (2016). Efficient tensor completion: Low-rank tensor train. *CoRR*, abs/1601.01083.

Rai, P., Wang, Y., Guo, S., Chen, G., Dunson, D., and Carin, L. (2014). Scalable bayesian low-rank decomposition of incomplete multiway tensors. In *International Conference on Machine Learning*, pages 1800–1808. PMLR.

Shi, Q., Cheung, Y.-M., and Lou, J. (2021). Robust tensor SVD and recovery with rank estimation. *IEEE Transactions on Cybernetics*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Wang, W., Sun, Y., Eriksson, B., Wang, W., and Aggarwal, V. (2018). Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Xu, L., Cheng, L., Wong, N., and Wu, Y.-C. (2020). Learning tensor train representation with automatic rank determination from incomplete noisy data. *arXiv preprint arXiv:2010.06564*.

Xu, L., Cheng, L., Wong, N., and Wu, Y.-C. (2021). Probabilistic tensor train decomposition with automatic rank determination from noisy data. In *2021 IEEE Statistical Signal Processing Workshop (SSP)*, pages 461–465. IEEE.

Yang, Y., Krompass, D., and Tresp, V. (2017). Tensor-Train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3891–3900.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. (2017). Long-term forecasting using Tensor-Train RNNs. *CoRR*, abs/1711.00073.

Zhao, Q., Zhang, L., and Cichocki, A. (2015). Bayesian sparse tucker models for dimension reduction and tensor completion. *arXiv preprint arXiv:1505.02343*.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. (2016). Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*.

A ADDITIONAL RELATED WORK

In this section, we provide additional related work on neural networks tensorization and optimal decomposition topology estimation, which goes beyond the scope of our study.

Tensor methods allow achieving significant compression, acceleration, and sometimes even quality improvement of neural networks. In Lebedev et al. (2015), 4-dimensional convolutional kernel tensors are decomposed with CP decomposition. The authors were able to accelerate a network by more than 8 times without significantly decreasing accuracy. In Novikov et al. (2015), TT-decomposition was leveraged to achieve up to $200000\times$ compression of fully-connected layers in a VGG-like network. Hrinchuk et al. (2020) used a similar approach to compress embedding layers in NLP models, which in some cases led to a noticeable quality increase due to the induced regularization. In Yang et al. (2017) the authors managed to achieve comparable performance with state-of-the-art models on very high-dimensional video classification tasks using orders of magnitude less complex TT-tensorized recurrent neural networks. Recently, Ma et al. (2019) applied Block-Term tensor decomposition (BTD) (De Lathauwer, 2008), a combination of CP and Tucker decompositions, to efficiently compress Multi-linear attention layers in Transformers and improved the single-model SoTA in language modeling. However, in all of these works, ranks selection was done manually for each decomposed layer.

A series of works has recently emerged based on greedy/evolutionary algorithms to learn the optimal tensor network topology (Hayashi et al., 2019; Li and Sun, 2020; Hashemizadeh et al., 2020; Li and Zhao, 2021; Li et al., 2022). These methods tackle a more general problem than optimal ranks selection in a concrete decomposition and, hence, impose overly complex multi-step algorithms to be directly applied for full DNN tensorization (mainly, these approaches consider simpler tasks like plain tensor decomposition or tensor completion).

B TENSORIZED LAYERS

In this section, we provide details concerning different tensorized neural network layers used in this work.

The simplest example of a decomposed layer is a fully-connected layer approximated via low-rank matrix factorization. In this case the matrix of weights $W \in \mathbb{R}^{M\times N}$ is represented via contraction (or matrix product) of two low-rank matrices $U_1 \in \mathbb{R}^{M\times r}$ and $U_2 \in \mathbb{R}^{r\times N}$:

$$W = U_1 U_2.$$

Mapping the input $x \in \mathbb{R}^N$ through these matrices in series leads to FLOPs reduction from $O\left(MN\right)$ to $O\left(r(M+N)\right)$, which could give a significant gain when r is smaller than M and N.

Similarly, Tucker-2 decomposition of a convolutional kernel (Kim et al., 2016) results in three consecutive smaller-sized convolutions. Namely, the convolutional kernel $\mathcal{K} \in \mathbb{R}^{C_{in}\times C_{out}\times k\times k}$, where C_{in}, C_{out} are the numbers of input and output channels and k is the kernel size, decomposes into two matrices $U_1 \in \mathbb{R}^{C_{in}\times r_1}$, $U_2 \in \mathbb{R}^{C_{out}\times r_2}$ and a smaller 4-dimensional tensor $\mathcal{G} \in \mathbb{R}^{r_1\times r_2\times k\times k}$ via the partial Tucker decomposition as:

$$\mathcal{K} = \mathcal{G} \times_1 U_1 \times_2 U_2.$$

Convolution operation with such a kernel can be rewritten as the following series of simpler convolutions: 1×1-convolution, reducing the number of channels from C_{in} to r_1, $k\times k$-convolution with r_1 input and r_2 output channels and again 1×1-convolution, restoring the number of output channels from r_2 to C_{out}. This trick helps to compress and speed up convolutions when the number of intermediate channels (i.e., ranks) is smaller than C_{in} and C_{out}.

In a fully-connected TT-layer (TT-FC) (Novikov et al., 2015), the matrix of weights $W \in \mathbb{R}^{M\times N}$, input and output vectors $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^M$ are reshaped into tensors $\mathcal{W} \in \mathbb{R}^{(m_1,n_1)\times\cdots\times(m_d,n_d)}$, $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ and $\mathcal{Y} \in \mathbb{R}^{m_1\times\cdots\times m_d}$, respectively, where $M = \prod_{k=1}^d m_k$, $N = \prod_{k=1}^d n_k$. Then $\mathcal{W}$ is converted into the TT-format with 4-dimensional cores $\boldsymbol{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_d\}$, $\mathcal{G}_k \in \mathbb{R}^{r_{k-1}\times m_k\times n_k\times r_k}$. The linear mapping $y = Wx$ translates into a series of contractions:[5]

$$\mathcal{Y} = \mathcal{G}_1 \ldots \mathcal{G}_d \mathcal{X},$$

which, calculated from right to left, yields the computational complexity $O\left(dr^2 n \max\{M,N\}\right)$, where r is the maximal TT-rank, $n = \max_{k=1\ldots d} n_k$. Similar technique, based on matrices represented in TT-format, or *TT-matrices*, underlies most other types of TT-layers.

[5] Strictly speaking, contractions over two modes n_k and r_k.

C IMPLEMENTATION DETAILS

Our implementation is based on `tt-pytorch`[6] library (Khrulkov et al., 2019), which provides the minimal required tools for working with TT-decomposition in neural networks using `PyTorch` (Paszke et al., 2019).

Initialization We use the Glorot-like (Glorot and Bengio, 2010) initialization for the TT-cores, implemented in the library and described in the corresponding paper, and the Kaiming Uniform initialization (He et al., 2015) for the Tucker-2 cores and matrices, which is default in `PyTorch`. We discovered that initialization and parameterization of masks probabilities matter: we use the logit reparameterization and initialize logits of $\boldsymbol{\phi}$ from the normal distribution with scale 10^{-2} and mean α, which is a hyperparameter (concrete values were chosen with a simple cross-validation procedure and are provided in each experiment). Variance of the normal prior over cores $p(\boldsymbol{G})$ is fixed and equals 10^2.

Choosing the initial rank is generally a non-trivial problem that can be attributed to common sparsifying/pruning techniques. We can state, however, that MARS does not significantly depend on this hyperparameter (as long as it is set somewhat reasonably) and is able to restore the ground truth rank from different initializations, as, e.g., demonstrated in Section 3.1 in the main text. We have empirically observed that running MARS from a higher initial rank with a slight tuning of π and/or α hyperparameters produces similar results in most cases.

Training In practice, to assist optimization, we do not multiply each of the decomposition cores, coupled via a shared mode, by the same corresponding relaxed binary mask, but instead perform only one multiplication. For instance, in Tucker-2 convolutional layer with masks $\boldsymbol{m} = \{m_1, m_2\}$ we apply the respective mask multiplication directly to the results of the first and second convolutions[7] instead of carrying out $U_1 \odot_2 m_1$, $U_2 \odot_2 m_2$, $\mathcal{G} \odot_1 m_1 \odot_2 m_2$. We use the cross-entropy loss as the negative model log-likelihood. We use Adam (Kingma and Ba, 2015) as the optimizer of choice. The temperature τ is exponentially decayed from 10^{-1} to 10^{-2} in the course of training. We discovered that *hard concrete* trick (Louizos et al., 2018), i.e., stretching the Binary Concrete distribution and then transforming its samples with a hard-sigmoid, allows achieving better results due to inclusion of $\{0, 1\}$ into the support. We also found that warming up with a plain tensorized model for several epochs can help optimization, therefore, we do not apply masks multiplication at the first epochs in most of our experiments.

In Tensor Train models we do not shrink the first and the last ranks, as they equal 1 by definition.

D ADDITIONAL EXPERIMENT: MARS WITH OTHER TENSORIZED MODELS

In this section, we demonstrate that MARS provides a unified way to select ranks in arbitrary models leveraging decomposed tensors that we name tensorized models, thus it should not be considered solely as a compression technique for neural networks. Namely, we consider a low-rank tensor approximation task as an illustrative example. We emphasize that our method can be naturally extended to other tasks by substituting the given likelihood with other cost functions depending on tensor parameters.

First, we construct a 4-dimensional tensor $\mathcal{A}$ with shape $\dim(\mathcal{A}) = (d, d, d, d), d = 8$ from a random Tucker decomposition with ranks $\boldsymbol{r} = \{r, r, r, r\}, r = 4$, i.e.,

$$\mathcal{A} = \mathcal{G}^{\mathcal{A}} \times_1 U_1^{\mathcal{A}} \cdots \times_4 U_4^{\mathcal{A}},$$

where cores $\mathcal{G}^{\mathcal{A}} \in \mathbb{R}^{r \times r \times r \times r}, U_k^{\mathcal{A}} \in \mathbb{R}^{d \times r}, k = 1 \dots 4$ are randomly initialized from a standard normal distribution. Then we consider a tensorized model parameterized by cores $\boldsymbol{G}^{\mathcal{B}} = \left\{\mathcal{G}^{\mathcal{B}}, U_1^{\mathcal{B}}, \dots, U_4^{\mathcal{B}}\right\}$, where $\mathcal{G}^{\mathcal{B}} \in \mathbb{R}^{R \times R \times R \times R}, U_k^{\mathcal{B}} \in \mathbb{R}^{d \times R}, k = 1 \dots 4, R = 8$, with the log-likelihood defined as a negative MSE between tensor

$$\mathcal{B}\left(\boldsymbol{G}^{\mathcal{B}}\right) = \mathcal{G}^{\mathcal{B}} \times_1 U_1^{\mathcal{B}} \cdots \times_4 U_4^{\mathcal{B}}$$

and tensor $\mathcal{A}$, i.e.,

$$\log p\left(\boldsymbol{G}^{\mathcal{B}}\right) = -\frac{1}{d^4} \left\| \mathcal{B}\left(\boldsymbol{G}^{\mathcal{B}}\right) - \mathcal{A} \right\|^2.$$

In other words, this model approximates the given tensor $\mathcal{A}$ with an 8-rank Tucker decomposition. Due to redundancy induced by the construction of $\mathcal{A}$ (GT rank is $r = 4$), this model naturally requires ranks selection.

[6] `https://github.com/KhrulkovV/tt-pytorch`

[7] We remind that Tucker-2 convolution decomposes into three consecutive smaller convolutions.

We applied MARS with $\pi = 10^{-2}$ and $\alpha = -0.5$ to the described model. We trained using standard gradient descent with LR 10^{-2} for 10^4 epochs to achieve full convergence. Mean results over 10 runs are the following: ranks $\boldsymbol{r} = \{4.7, 4.4, 4.6, 4.2\}$ and log-likelihood $\log p\left(\boldsymbol{G}^{\mathcal{B}}\right) = -0.027$, which is very close to the GT solution. For comparison, a typical MSE value between two random tensors initialized according to the scheme of $\mathcal{A}$ is ≈ 300.

Moreover, we trained a similar model with $R = r = 4$ without MARS (because there are no more extra ranks) and achieved a log-likelihood value of only ≈ -0.15. We believe that more degrees of freedom in the redundant model helps MARS find a more optimal solution than even starting from the GT ranks. It would be interesting to get a deeper theoretical insight explaining the differences in the experimental results of the direct learning methods with lower ranks and MARS.

In this synthetic experiment, we demonstrated that MARS is applicable for tensorized models other than tensorized neural networks and can be used to efficiently restore GT ranks in, e.g., low-rank tensor approximation task. Yet, we can expect that other methods, e.g., Tucker-ARD (Mørup and Hansen, 2009), tailored explicitly for automatic ranks determination in tensor approximation, might exhibit similarly on this simple problem. Further extension of MARS to other tasks along with a detailed comparison with relevant baselines is future work.

E ABLATION STUDY: INFLUENCE OF HYPERPARAMETER π

In this section, we report the results of an ablation experiment concerning the influence of the value of hyperparameter π in MARS on the final model performance and compression. Specifically, we repeat the 2FC-Net MNIST experiment from Section 3.2 using soft-mode initialization ($\alpha = -1.5$) with different values of π. We present the results in Figure 3.

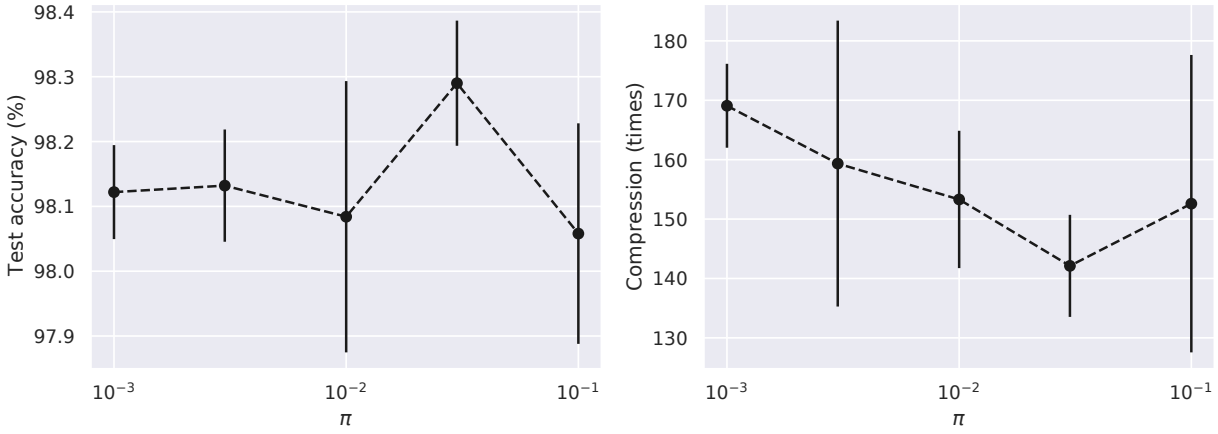

Figure 3: Test accuracy (left) and compression (right) for different values of hyperparameter π. Mean $\pm$ std over 5 runs is reported for each value of π.

As can be seen from the plots, the final test accuracies of the trained models are similar and weakly depend on the value of hyperparameter π. At the same time, there is a tendency for compression to gradually decrease with increasing π, which fully corresponds to its role in our model. Note also that too high values of π can lead to inconsistent results and harm performance, since the prior term in model (2) loses its ability to promote sparse mask solutions to select optimal ranks.

In the end, we conclude that taking $\pi \approx 10^{-2}$ is a reasonable choice to achieve satisfactory compression-accuracy trade-off in most situations.

F LIMITATIONS AND SOCIETAL IMPACT

In this section, we want to discuss the limitations and possible societal impact of our work.

We position our method, MARS, as a general and efficient way to select ranks in various tensorized models. While we demonstrate its possible applicability to other tasks in the previous section, in our main experiments, we mostly concentrate on tensorized neural networks, since they are widely known in the community and overall recognized as a promising direction. Further development of MARS and its evaluation on other tensor models and problems is a promising future research.

Another limitation of our work is the simplicity of the considered family of distributions over masks. We conjecture that more advanced variational families could put our inference method beyond a simple MAP estimate and/or further improve MARS performance. We also find this an important direction for future studies.

While modern DNN models continue increasing the total number of learnable parameters, they require more computational resources than ever before. This circumstance under no doubts imposes negative environmental influence. As discussed in Appendix A, tensor methods allow a significant reduction of occupied resources and energy consumption. We hope that our method will further disseminate these methods and serve to promote environmental protection. To the best of our knowledge, we cannot find any negative consequences from the misuse of the paper's contribution.