MARS: Masked Automatic Ranks Selection in Tensor Decompositions

Maxim Kodryan[1], **Dmitry Kropotov**[1,2], **Dmitry Vetrov**[1,3]
[1]Samsung-HSE Laboratory, National Research University Higher School of Economics
[2]Lomonosov Moscow State University [3]Samsung AI Center Moscow
Moscow, Russia
{mkodryan,dkropotov,dvetrov}@hse.ru

Abstract

Tensor decomposition methods have recently proven to be efficient for compressing and accelerating neural networks. However, the problem of optimal decomposition structure determination is still not well studied while being quite important. Specifically, decomposition ranks present the crucial parameter controlling the compression-accuracy trade-off. 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. The proposed method achieves better results compared to previous works in various tasks.

1 Introduction

Deep neural networks (DNNs) are able to achieve state-of-the-art results in a vast range of problems such as image classification [14] or machine translation [40]. The key to such efficiency is their over-parameterized structure, which facilitates in finding good local optima [7, 38]. Moreover, recent studies [1, 32] empirically show that increasing model complexity, after a certain threshold, improves quality. However, over-parameterization, while useful for training, also leads to redundancy [6] which might hinder deployment of DNNs in resource-constrained environments, like mobile devices.

Decomposition methods cope with redundancy via an efficient decomposed representation of neural network parameters. The 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 [39] and canonical polyadic (CP) [2] tensor decompositions are widely known for compressing and accelerating convolutional networks [23, 17, 20], and Tensor Train (TT) [34] decomposition has been successfully applied for compressing fully-connected (FC) [33], convolutional [8], recurrent [43, 44], embedding [16] layers.

Probably the main crux of the tensor approach is the need to carefully select decomposition hyperparameters, namely the ranks. Tensor decomposition ranks are responsible for the trade-off between the quality of the model and the required resources, thus they represent extremely important hyperparameters. Yet, the problem of optimal ranks selection in general tensor decompositions is still acute. Typical hyperparameter selection techniques, like cross-validation, are inapplicable for efficient 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 worsen performance.

First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020).

In this work, we propose *Masked Automatic Ranks Selection* (MARS) — a new efficient method for dynamic selection of tensor decomposition ranks grounded in Bayesian framework. The core idea is to learn binary masks that cover decomposition cores and "select" only the ranks required for the optimal model performance, hence the name. The method is applicable for any models leveraging tensor decompositions and operates end-to-end with model training without introducing any significant additional computational overhead. 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 in tensorization.

2 Related work

Tensor methods allow achieving significant compression, acceleration and sometimes even quality improvement of neural networks. In Lebedev et al. [23], 4-dimensional convolutional kernel tensors are decomposed with CP decomposition. The authors were able to accelerate a network by more than 8 times without a significant decrease in accuracy. In Novikov et al. [33], TT-decomposition was leveraged to achieve up to $200000\times$ compression of fully-connected layers in a VGG-like network. In Khrulkov et al. [16], a similar approach was used 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. [43] 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. [27] applied Block-Term tensor decomposition (BTD) [4], 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.

Kim et al. [17] 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 [31]. The involved training procedure consisting of decomposition of the pre-trained model and fine-tuning of the decomposed model, however, turned out to be inefficient. The MUSCO algorithm [11], which repeatedly performs decomposition and fine-tuning steps, partially resolved this disadvantage. Very recently, Cheng et al. [3] proposed a reinforcement learning-based rank selection scheme for tensorized neural networks 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 any stages, which is naturally more preferable. Moreover, it is not confined to specific types of tensor decompositions, models or tasks.

Existing methods for automatic determination of the decomposition ranks, which also take advantage of the Bayesian approach, cover only certain types of tensor decompositions or are based on peculiarities of the task, e.g., tensor approximation [30] or linear regression [10]. These approaches mostly embody structured pruning of the decomposition cores. For instance, Hawkins and Zhang [12] propose a special 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 involving arbitrary tensorized models. In addition, our method is based on a fundamentally different idea of learning sparse binary masks over the tensor decomposition cores.

Alternative procedures aimed at obtaining low-rank tensor representation, e.g., those utilizing nuclear norm regularization [36, 15], also leverage properties of the particular objective, like tensor completion, 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.

3 MARS

In this section, we introduce general tensorized models and describe the details of the proposed method. The notions regarding tensors and tensor decompositions are provided in Appendix A.

3.1 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 computational complexity) to handle tensors in decomposed format explicitly. In other words, considering the concrete decomposition applied, one could rewrite model operations more efficiently via the cores $\boldsymbol{G}$ directly, without the need to construct 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 decomposed tensors, *tensorized models*, and assume that they support operations directly via these cores.

A typical case of a tensorized model is a neural network with decomposed layers. Representing parameters of a layer via a decomposed tensor may result in substantial memory and computational savings. For most types of NN layers there exists 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, shapes of the decomposition cores simultaneously influence model flexibility and complexity. The key hyperparameter determining them are decomposition ranks, as discussed earlier. Further, we describe the details of the proposed method for ranks selection in arbitrary tensorized models.

3.2 The proposed method

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

MARS suggests obtaining such reduced structures via multiplying slices of cores by binary masking vectors, mostly consisting of zeros. Zeroed slices will not be involved in tensors contractions and, therefore, all the further 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 optimal shapes of the cores. Figure 1 illustrates the concept.

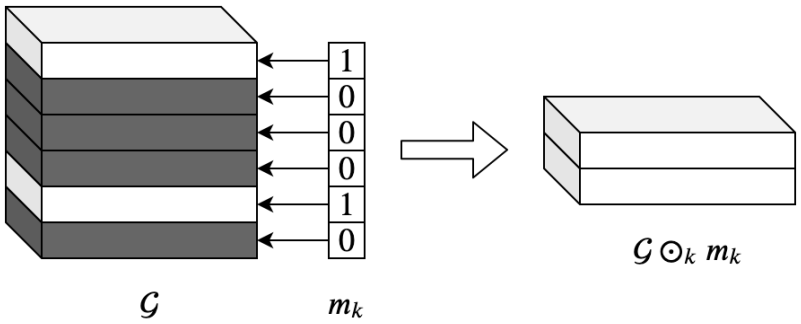

Figure 1: A schematic illustration of the MARS concept: slices of the core tensor $\mathcal{G}$ along mode k are multiplied by elements of the binary mask m_k; only "selected" non-zero slices will participate in further model operations, therefore, the core shape can be reduced.

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

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

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, $\boldsymbol{G} \circ \boldsymbol{m} = \{\mathcal{G}_k \circ \boldsymbol{m}\}$ is a set of *masked cores*:

$$\mathcal{G}_k \circ \boldsymbol{m} = \mathcal{G}_k \odot_{k_1} m_{k_1} \cdots \odot_{k_p} m_{k_p},$$

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

$r_{k_1}, \ldots, r_{k_p} \in \text{dims}\,(\mathcal{G}_k)$ are all ranks belonging to the dimensions of the core $\mathcal{G}_k$. The likelihood $p\,(y \mid x, \boldsymbol{G} \circ \boldsymbol{m})$ is defined by the tensorized model, e.g., it can be (exponent of negative) cross-entropy loss for classification tasks.

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:

$$p\,(\boldsymbol{m}) = p\,(\boldsymbol{m} \mid \pi) = \prod_k \prod_{s=1}^{r_k} \pi^{m_k(s)}(1-\pi)^{1-m_k(s)}. \tag{2}$$

We would like to emphasize that instead of adjusting several or even dozens of ranks in tensor decompositions, one needs to set only one hyperparameter in our model (along with careful parameter initialization). Furthermore, in our experiments, we found out that π does influence the final compression-accuracy trade-off but not crucially. 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 no ranks selection would be performed.

We also suggest to use the factorized zero-mean Gaussian with a fixed large variance as the prior distribution over the cores $p\,(\boldsymbol{G})$. It might serve as a slight L_2 regularization and is empirically shown to help balance coefficients in the cores, stabilize the training process and improve test accuracy. We did not exhaustively investigate the optimal values of the prior variance and used the same 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 (1):

$$\sum_{i=1}^{N} \log p\,(y_i \mid x_i, \boldsymbol{G} \circ \boldsymbol{m}) + \log p\,(\boldsymbol{m}) + \log p\,(\boldsymbol{G}) \longrightarrow \max_{\boldsymbol{m}, \boldsymbol{G}}. \tag{3}$$

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 the problem (3) with equivalent:

$$\mathbb{E}_{\boldsymbol{m} \sim q(\boldsymbol{m})} \left[\sum_{i=1}^{N} \log p\,(y_i \mid x_i, \boldsymbol{G} \circ \boldsymbol{m}) + \log p\,(\boldsymbol{m}) \right] + \log p\,(\boldsymbol{G}) \longrightarrow \max_{q(\boldsymbol{m}), \boldsymbol{G}}, \tag{4}$$

where the family of distributions $q(\boldsymbol{m})$ includes deterministic ones, i.e., taking only a single value. The solutions of problems (3) and (4) coincide. This follows from the fact that for any distribution $q(x)$ and any function $F(x)$:

$$\mathbb{E}_{x \sim q(x)} F(x) \leq F(x^*), \tag{5}$$

where $x^* = \text{argmax}_x\, F(x)$, and (5) turns into equality when $q(x) = \delta(x - x^*)$.

Next, we constrain $q(\boldsymbol{m})$ to be factorized Bernoulli over each mask element $m_k(s)$ with parameters $\boldsymbol{\phi} = \{\phi_k(s)\}$. The problem (4) translates into the following:

$$\mathbb{E}_{\boldsymbol{m} \sim q_{\boldsymbol{\phi}}(\boldsymbol{m})} \left[\sum_{i=1}^{N} \log p\,(y_i \mid x_i, \boldsymbol{G} \circ \boldsymbol{m}) \right] + \sum_k \sum_{s=1}^{r_k} \left[\phi_k(s) \log \pi + (1 - \phi_k(s)) \log(1-\pi) \right] + \\ + \log p\,(\boldsymbol{G}) \longrightarrow \max_{\boldsymbol{\phi}, \boldsymbol{G}}. \tag{6}$$

One can notice that adding the q entropy term into (6) yields the evidence lower bound (ELBO) maximization, a well-known Bayesian technique for variational posterior approximation, with factorized Bernoulli variational distribution. We discuss it in more detail at the end of the paper.

We perform maximization (6) with stochastic gradient ascent. In order to calculate low-variance stochastic gradients w.r.t. parameters $\boldsymbol{\phi}$ in (6) we use the *reparameterization trick* [19]. To this end, we relax discrete samples from $q_{\boldsymbol{\phi}}(\boldsymbol{m})$ in the expectation term by the means of the Binary Concrete relaxation [29] with temperature, which defines "discreteness" of the relaxed samples, decaying to zero in the course of training.

After training, we round probabilities $\boldsymbol{\phi}$ to binary masks $\boldsymbol{m}_{MAP}$ and can use a compact solution with reduced cores $\boldsymbol{G}_{MAP} \circ \boldsymbol{m}_{MAP}$ to predict for a new data sample x^*: $p\,(y^* \mid x^*, \boldsymbol{G}_{MAP} \circ \boldsymbol{m}_{MAP})$. Algorithm 1 summarizes the training procedure. $RB(\phi, \tau)$ denotes the Relaxed Bernoulli distribution which is essentially the Binary Concrete with temperature τ and location $\frac{\phi}{1-\phi}$.

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 a set of masks $\hat{\boldsymbol{m}} = \{\hat{m}_k(s) \sim RB(\phi_k(s), \tau)\}$
  Sample a mini-batch of objects $\{(x_{i_l}, y_{i_l})\}_{l=1}^{B}$
  $L \coloneqq \sum_{l=1}^{B} \log p\left(y_{i_l} \mid x_{i_l}, \boldsymbol{G} \circ \hat{\boldsymbol{m}}\right)$
  $g_{\boldsymbol{\phi}} \coloneqq \frac{\partial L}{\partial \boldsymbol{G} \circ \hat{\boldsymbol{m}}} \frac{\partial \boldsymbol{G} \circ \hat{\boldsymbol{m}}}{\partial \hat{\boldsymbol{m}}} \frac{\partial \hat{\boldsymbol{m}}}{\partial \boldsymbol{\phi}} + \log\left(\frac{\pi}{1-\pi}\right)$
  $g_{\boldsymbol{G}} \coloneqq \frac{\partial L}{\partial \boldsymbol{G} \circ \hat{\boldsymbol{m}}} \frac{\partial \boldsymbol{G} \circ \hat{\boldsymbol{m}}}{\partial \boldsymbol{G}} + \frac{\partial \log p(\boldsymbol{G})}{\partial \boldsymbol{G}}$
  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 $\boldsymbol{m}_{MAP} \coloneqq round(\boldsymbol{\phi})$

4 Experiments

In our experiments, we use tensorized neural networks with predefined decomposition ranks, and train them with 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 for further inference.

As was mentioned earlier, 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 π. We refer to Appendix C for more details on implementation.

Further in this section, we provide the results of the conducted experiments with our method that demonstrate the ability of MARS to improve previous results on tensorization. We also provide additional numerical experiments with MARS, including actual acceleration of a tensorized model and ResNet-110 tensorization, in Appendix D.

4.1 Toy experiment

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

We consider the following formulation. Let N, D, C, r^*, R denote, respectively, the number of samples, input, output dimensions, the true rank of the problem, and the initial parameter rank 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 $Y = \text{OHE}\left(XU^*V^*\right)$ 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 classificator with a factorized parameter matrix UV using MARS to reduce the initial rank $R > r^*$ and restore the ground truth rank of the problem.

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 validation set and trained a vanilla linear classificator as a baseline.

We report the results averaged over 10 runs in Table 1. As can be seen, MARS is able to 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 show better validation accuracy than the baseline.

Table 1: True rank restoring with MARS in a toy linear classification task with a factorized parameter matrix. Results are averaged over 10 runs. We report mean $\pm$ std.

GT rank r^*	Reduced rank 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%

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.

4.2 MNIST 2FC-Net

In this experiment, we compare against LR-BTNN [12] on the MNIST dataset [24]. In this task, both MARS and LR-BTNN aim to automatically select ranks in a relatively small classification neural network with two TT-decomposed fully-connected layers of sizes 784×625 and 625×10. As proposed by Hawkins and Zhang [12], we take the following dimensions factorizations 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.

We evaluated MARS in two modes on this task: soft compression ($\alpha = -1.5$, $\pi = 10^{-1}$) and hard compression ($\alpha = -1.75$, $\pi = 10^{-2}$). 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 [12] in this task both in terms of compression and final accuracy, even though LR-BTNN is specifically tailored for Tensor Train decomposition, whereas MARS proposes a unified way to select ranks in various tensorized models.

We would also like to note that ensemble of small MAP networks, obtained in soft compression mode, gives accuracy of $\mathbf{98.9}\%$. We argue that compact tensorized networks 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 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.

Table 2: Compression/accuracy on MNIST with 2FC-Net. Results are averaged over 10 runs. We report mean $\pm$ std.

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)	$\mathbf{205 \pm 30.9}\times$	$97.9 \pm 0.19\%$

4.3 Sentiment analysis with TT-embeddings

A recent work of Khrulkov et al. [16] 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 occupied memory) factorization of dimensions in TT-matrices given the number of factors d. However, in their experiments, the ranks of TT-decomposition were still manually set equal to some predefined value.

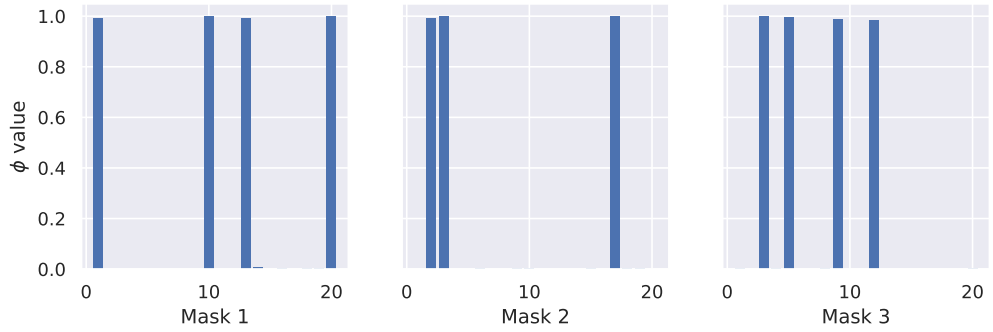

Figure 2: Learned binary masks probabilities ϕ corresponding to the first TT-layer in MNIST 2FC-Net. Note that the relaxed MARS MAP estimate is quite close to the deterministic solution.

We repeat their experiment on 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 [28] and Stanford Sentiment Treebank (SST) [37]. 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 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 contains 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 inefficient as opposed to nonuniform ranks selection.

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	$441\times$	88.8%
	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	$\mathbf{340\times}$	$\mathbf{42.4}\%$

5 Conclusion and future work

In this paper, we present MARS, the method for automatic selection of ranks in tensorized models leveraging arbitrary tensor decompositions. The basic principle of MARS is learning special binary masks along with overall model training, that cover the cores of 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 conducted experiments demonstrate that

our technique can improve accuracy and compression of tensorized models with manually selected ranks and surpass or perform comparably with alternative rank selection methods specialized on concrete types of tensor decompositions.

It is widely known that ensembling of deep neural networks leads to significant quality improvement [22]. 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 network. This opens a very promising perspective for future research.

In MARS we learn a single MAP estimate of masks. However, learning the distribution over binary masks could allow to build ensembles of compact tensorized models via sampling from it. We noted in section 3 that our objective (6) resembles ELBO up to the entropy term. Unfortunately, our experiments in variational inference with factorized Bernoulli $q_{\boldsymbol{\phi}}(\boldsymbol{m})$ led to distributions with overly low variance. In other words, sampling from $q_{\boldsymbol{\phi}}(\boldsymbol{m})$ did not improve accuracy compared to the model spawned by its mode. This might mean that fully factorized Bernoulli cannot appropriately approximate the true posterior due to numerous correlations between mask variables, although it is quite effective for finding the MAP estimate. We believe that more flexible variational families, e.g., those based on hierarchical models, may lead to better approximation of the posterior, and leave it for future study.

Other research directions include possible improvements of the model and learning method, e.g., trying REINFORCE-like algorithms [42] for optimization over discrete masks. We also consider applying MARS to other types of tensor decompositions and tensorized models, like TRNs, as discussed at the end of Appendix D.1.

Acknowledgments and Disclosure of Funding

The theoretical results on automatic ranks selection in general tensor decompositions were supported by the Russian Science Foundation grant № 19-71-30020. The experimental results on neural networks compression via automatic tensorization were supported by Samsung Research, Samsung Electronics.

References

[1] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2018). Reconciling modern machine learning and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*.

[2] Caroll, J. and Chang, J. (1970). Analysis of individual difference in multidimensional scaling via n-way generalization of ecart-young decomposition, psychometrika, vol. 35, n 3.

[3] 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.

[4] 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.

[5] 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.

[6] Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2148–2156.

[7] Du, S. S. and Lee, J. D. (2018). On the power of over-parametrization in neural networks with quadratic activation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1328–1337.

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

[9] 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.

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

[11] Gusak, J., Kholyavchenko, M., Ponomarev, E., Markeeva, L., Oseledets, I. V., and Cichocki, A. (2019). One time is not enough: iterative tensor decomposition for neural network compression. *CoRR*, abs/1903.09973.

[12] Hawkins, C. and Zhang, Z. (2019). Bayesian tensorized neural networks with automatic rank selection. *arXiv preprint arXiv:1905.10478*.

[13] 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.

[14] 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.

[15] 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.

[16] Khrulkov, V., Hrinchuk, O., Mirvakhabova, L., and Oseledets, I. V. (2019). Tensorized embedding layers for efficient model compression. *CoRR*, abs/1901.10787.

[17] 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*.

[18] 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*.

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

[20] 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.

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

[22] 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.

[23] 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*.

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

[25] 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.

[26] Louizos, C., Welling, M., and Kingma, D. P. (2017). Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

[27] 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.

[28] 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.

[29] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

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

[31] 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.

[32] Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2019). Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] Soltanolkotabi, M., Javanmard, A., and Lee, J. D. (2019). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Trans. Information Theory*, 65(2):742–769.

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

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. 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 6000–6010.

[41] 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.

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

[43] 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.

[44] Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. (2017). Long-term forecasting using tensor-train rnns. *CoRR*, abs/1711.00073.

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

A Tensors and tensor decompositions

In this section, we introduce the required notions regarding tensors and tensor decompositions.

A.1 Tensors

By a *d-dimensional tensor*, we mean a multidimensional 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 $\text{dims}\,(\mathcal{A}) = (n_1, \ldots, n_d)$ 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 $\mathcal{AB} \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times m_2 \cdots \times m_{d'}}$:

$$\mathcal{AB}(i_1, \ldots, i_{d-1}, j_2, \ldots, j_{d'}) = \sum_{i_d=1}^{n_d} \mathcal{A}(i_1, \ldots, i_d)\mathcal{B}(i_d, j_2, \ldots, j_{d'}). \tag{7}$$

Contractions can be generalized to multiple modes. In this case, summation in (7) is performed over these modes, and dimensions of the resulting tensor will contain 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* $\mathcal{A} \times_k B \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times m_k \times n_{k+1} \times \cdots \times n_d}$:

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

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

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

A.2 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 Tensor Train decomposition $\boldsymbol{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_d\}$, $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, $r_0 = r_d = 1$ and

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

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

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

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

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 cores modes over which contraction is applied, are called *ranks* of the decomposition. Clearly, they define the expressivity of the decomposition on the one hand and the number of required parameters on the other.

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 $O(MN)$ to $O(r(M+N))$ which could give a significant gain when r is smaller than M and N.

Similarly, Tucker-2 decomposition of a convolutional kernel [17] 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) [33] 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:[2]

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

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.

C Implementation details

Our implementation is based on `tt-pytorch`[3] library [16] which provides the minimal required tools for working with TT-decomposition in neural networks using `PyTorch` [35].

Initialization We use the Glorot-like [9] initialization for the TT-cores, implemented in the library and described in the relevant paper, and the Kaiming Uniform initialization [13] 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. Variance of the normal prior over cores $p(\boldsymbol{G})$ is fixed and equals 10^2.

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 to the results of the first and second convolutions[4] 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 cross-entropy loss as the negative model log-likelihood. We use Adam [18] as the optimizer of choice. The temperature τ is exponentially decayed from 10^{-1} to 10^{-2} in the course of training.

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

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

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

We discovered that *hard concrete* trick [26], 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 may improve 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 the definition.

D Additional experiments

In this section, we provide additional numerical experiments, involving tensorized models comparison in terms of speed-up and ResNet-110 compression via tensorization.

D.1 MNIST LeNet-5

In Wang et al. [41] Tensor Ring (TR) decomposition [45], a generalization of Tensor Train decomposition, was applied to compress convolutional networks. 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. [17], where Tucker-2 and low-rank matrix factorization (which are a simpler decomposition family) are used for the same purpose. In one of the experiments, both methods were evaluated on the task of compressing and accelerating LeNet-5 [25], a relatively small convolutional neural network with 2 convolutional layers, followed by 2 fully-connected layers, on MNIST dataset. TRN could significantly surpass the simpler Tucker approach.

In this experiment, we demonstrate that even using less expressive types of decompositions, one can achieve comparable results with 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 ranks r_1, r_2 of Tucker-2 decomposition and the matrix rank r using our method, starting from $r_1 = r_2 = 20$, $r = 100$ ($2.9\times$ compression at the start). We initialize the mean value α of $\boldsymbol{\phi}$ logits 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 at approximately the same quality which made it comparable to TRN, which leverages 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 to perform decomposition of layers, after which the model is fine-tuned. MARS performs ranks selection end-to-end with model training which results in significantly better results.

Table 4: Compression, accuracy and speed-up on MNIST with LeNet-5. TRN-r denotes 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. [41]. Results are averaged over 5 runs. We report mean $\pm$ std.

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	$10 \pm 0.8\times$	$99.0 \pm 0.07\%$	$\mathbf{1.19 \pm 0.01\times}$

Another important achievement of our model is the ability to actually accelerate network. Despite the fact that TR-decomposition allows to achieve better compression, it, however, slows down 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 work faster.

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

D.2 CIFAR-10 ResNet-110

The main experiment of Hawkins and Zhang [12] consisted in applying their LR-BTNN method to ResNet-110 [14] on CIFAR-10 dataset [21]. 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 of constructing the full convolutional tensor from the TT-cores on each forward pass. Unlike this method, Garipov et al. [8] 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. [8] *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 [5].

We repeat the ResNet experiment of Hawkins and Zhang [12] with MARS using both naive and proper schemes for TT-decomposition of convolutions. The 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, 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 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 naively decomposed baseline TT-model[5] but at a significantly higher compression ratio which once again emphasizes the efficiency of Garipov et al. [8] 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%

[5]We rely on the results reported by Hawkins and Zhang [12].