Supplementary Materials for "Learning Low-Rank Tensor Cores with Probabilistic ℓ_0-Regularized Rank Selection for Model Compression"

Submission 2486

This document is the supplementary material for the submitted paper "Learning Low-Rank Tensor Cores with Probabilistic ℓ_0-Regularized Rank Selection for Model Compression". This document provides extra experimental results and details.

1 Experiment Details

Implementation We implement all experiments based on PyTorch. Tensor operations are done using *torch.tensordot()*, and the library TENSORLY [Kossaifi *et al.*, 2019]. Experiments on neural machine translations are executed on a single NVIDIA A40, and the rest are executed on a single GPU of either NVIDIA RTX 2080Ti or NVIDIA RTX TITAN.

Methods for Comparison We note that due to the limited details and the unavailability of source code, we were unable to reproduce some of the comparing experiments. We acknowledge the contributions of [Cheng *et al.*, 2020; Gao *et al.*, 2020; Li *et al.*, 2021; Kodryan *et al.*, 2023], whose results we cite in this paper. This also results in certain methods not appearing in some models.

1.1 Tiny Example: 2-Layer MLP

This section corresponds to the experiment of Sec. 5.1 in the paper. Comparisons of the learned ranks of *Ours (TT)* and MARS are shown in Fig. 1.

1.2 LeNet-5 on MNIST

LeNet-5 Structure The structure of the used uncompressed LeNet-5 is introduced in Sec. 5.2 of the paper. The Tensor-Ring (TR) decomposed structure is summarized in Table 1. The first line and the second line of OURS (TR) in Table 2 of the paper use a regularization $\lambda = 1$ and $\lambda = 0.2$, respectively.

1.3 ResNets on CIFAR-10

ResNet Structure The TR decomposed structures of used ResNet-32 and ResNet-110 are summarized in Table 2. As introduced in the paper, the trainable gates are fixed after a pre-defined compression ratio is achieved. We set the target compression ratios as $5\times$ and $5.6\times$ for ResNet-32 and ResNet-110, respectively. We use a regularization $\lambda = 0.1$. For other details, we use an SGD with a momentum of 0.9 and a weight decay of 0.0001.

Table 1: The input dimension and out dimension of LeNet-5 model and corresponding tensorizations in Tensor-Ring formats. The TR structure is used in [Wang *et al.*, 2018; Li *et al.*, 2021] and our experiments.

Layer	in dim. $\times$ out dim.	Tensorization
CONV1	1×20	$(1) \times (4,5)$
CONV2	20×50	$(4,5) \times (5,10)$
FC1	1250×320	$(5,5,5,10) \times (5,8,8)$
FC2	320×10	$(5,8,8) \times (10)$

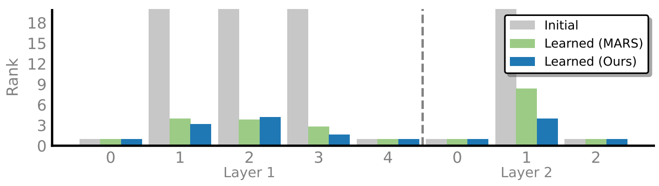

Figure 1: The average learned ranks of ours and MARS for the 2-layer MLP on MNIST. The horizontal axis is the index of the rank of the cores.

Table 2: The tensorization and details of ResNet model in TR formats. The numbers of blocks in ResNet-32 and ResNet-110 are shown. The TR structure is used in [Wang *et al.*, 2018; Li *et al.*, 2021; Cheng *et al.*, 2020] and our experiments.

Layer	Tensorization	ResNet-32	110
CONV	$(3) \times (4,2,2)$		
CONV BLOCK	$(4,2,2) \times (4,2,2)$	$\times 5$	$\times 18$
DOWNSAMPLE	$(4,2,2) \times (4,4,2)$		
CONV BLOCK	$(4,4,2) \times (4,4,2)$	$\times 4$	$\times 17$
DOWNSAMPLE	$(4,4,2) \times (4,4,4)$		
CONV BLOCK	$(4,4,4) \times (4,4,4)$	$\times 4$	$\times 17$
FC	$(4,4,4) \times (10)$		

Table 3: Training setting of IWSLT'14 De-En.

Hyperparameter	Value
label smoothing	0.1
dropout	0.1
sentence max tokens	128
batch size	100 sentence pairs
vocab size	32768
embedding dim	$(8, 8, 8)$
encoder/decoder	6/6
FFN dim	1024
attention heads	4
optimizer	Adam
betas	(0.9, 0.98)
lr	2e-4
lr scheduler	cosine annealing, no restarts
eps	1e-8
weight decay	1e-4
total epochs	30
warmup lr	linear
warmup iterations	100
token generation	greedy search

1.4 Embedding Layers

Text Sentiment Classification The details of optimization are introduced in this paragraph. We use an Adam optimizer with a learning rate being 0.0005. The model is trained for 30 epochs. The regularization is $\lambda = 0.005$.

Neural Machine Translation In the IWSLT'14 De-En task, we use $\lambda = 0.01$. A detailed hyperparameter setting is listed in Table 3.

2 Ablation Study

2.1 Direct Optimization of Discrete Bernoulli

We hold the proposition that samplings from discrete distribution suffer from high variance for the ℓ_0 optimization problem (12) in the paper and propose to use approximate Gaussian-based Bernoulli continuous relaxations. In this section, we derive the formulation for optimizing discrete Bernoulli random variables with gradient methods using REINFORCE-like [Williams, 1992] algorithms and empirically compare it with the formulation proposed in the paper by testing them against LeNet-5 on the MNIST dataset.

We start with the proposed gated tensor contraction and optimization problem (12). The diagonal gate matrix $\boldsymbol{Z}_i$ is re-defined to have diagonal entries *i.i.d.* and satisfies $\boldsymbol{Z}_i(j,j) \sim \text{Bern}(\pi_{i,j})$. Here, we put these Bernoulli gates into the optimization phase by introducing trainable parameters μ_{ij} and clamp them into $[0,1]$ by defining $\pi_{i,j} := \max(0, \min(1, \mu_{i,j}))$. For simplicity of notations, we consider one layer with n tensor cores of the neural network and will write the expected risk of the optimization problem (12) in the paper as

$$\min_{\boldsymbol{\theta}'} \quad \mathbb{E}_{\boldsymbol{Z}}\mathbb{E}_{X,Y}\left[\mathcal{L}(f_{\boldsymbol{\theta}'}(X),Y)+\lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0\right]. \quad (1)$$

Table 4: LeNet-5 on MNIST. Results show the mean and standard deviation of five runs.

Method	Accuracy (%)	Compression
LeNet-5	**99.37**±0.06	1×
REINFORCE-ℓ_0	96.27±1.02	19.00±2.33×
Ours (TR)	**99.37**±0.03	**20.53**±0.81×

The gradient w.r.t. $\pi_i = [\pi_{i1}, \pi_{i2}, \ldots]^\intercal$ is

$$\begin{aligned}
&\nabla_{\pi_i}\mathbb{E}_{\boldsymbol{Z}}\mathbb{E}_{X,Y}\left[\mathcal{L}(f_{\boldsymbol{\theta}'}(X),Y)+\lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0\right]\\
&=\mathbb{E}_{X,Y}\Bigg[\sum_{\boldsymbol{Z}_i\in\{0,1\}^{R_i}}\nabla_{\pi_i}\mathbb{P}(\boldsymbol{Z})\cdot\left(\mathcal{L}+\lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0\right)\\
&\qquad+\mathbb{P}(\boldsymbol{Z})\cdot\nabla_{\pi_i}\mathcal{L}]\\
&=\mathbb{E}_{X,Y}\Bigg[\sum_{\boldsymbol{Z}_i\in\{0,1\}^{R_i}}\mathbb{P}(\boldsymbol{Z})\cdot\Bigg\{\nabla_{\pi_i}\log\mathbb{P}(\boldsymbol{Z})\left(\mathcal{L}+\lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0\right)\\
&\qquad+\nabla_{\pi_i}\mathcal{L}\Bigg\}\Bigg]\\
&=\mathbb{E}_{X,Y}\mathbb{E}_{\boldsymbol{Z}}\left[\nabla_{\pi_i}\log\mathbb{P}(\boldsymbol{Z})\left(\mathcal{L}+\lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0\right)+\nabla_{\pi_i}\mathcal{L}\right],
\end{aligned} \quad (2)$$

where $\mathbb{P}(\boldsymbol{Z}) = \prod_{i=1}^{n}\mathbb{P}(\boldsymbol{Z}_i)$. The gradient w.r.t. μ_{ij} can be easily derived and calculated with automatic differentiation systems.

Then, we can do Monte Carlo samplings and optimize the following surrogate objective using gradient methods:

$$\frac{1}{M}\sum_{m=1}^{M}\left[\log\mathbb{P}(\boldsymbol{Z}^{[m]})\cdot c+\underbrace{\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}(f_{\{\boldsymbol{\mathcal{G}}_i,\boldsymbol{Z}_i^{[m]}\}}(\boldsymbol{x}_k),\boldsymbol{y}_k)}_{\mathcal{J}}\right], \quad (3)$$

where $c = \mathcal{J} + \lambda\sum_{i=1}^{n}\|\boldsymbol{Z}_i\|_0$ is a constant that is not involved in gradient computations, M is the number of samplings with $^{[m]}$ denoting the m-th MC sample, and $\{\boldsymbol{x}_k, \boldsymbol{y}_k\}_{k=1}^{K}$ are the training data.

2.2 Evaluation on LeNet-5

We test the method derived in Sec. 2.1 (which we will call REINFORCE-ℓ_0) and the proposed approximate Bernoulli gates in the paper for jointly learning weights and tensor ranks of LeNet-5 on the MNIST dataset. Empirically, we observe that REINROCE-ℓ_0 is not sensitive to the regularization coefficient λ but is sensitive to the initial values of the Bernoulli parameters. To control the compression ratio at $\sim 20\times$, we set the initial Bernoulli parameters π_{ij} as 0.55. During evaluation phases, the gates are rounded to zero or one to determine the learned ranks. We set $\lambda = 0.2$ for both methods and learn from initial ranks of 25. We set $M = 5$ for REINFORCE-ℓ_0 and $M = 1$ for our proposed method. Other settings are

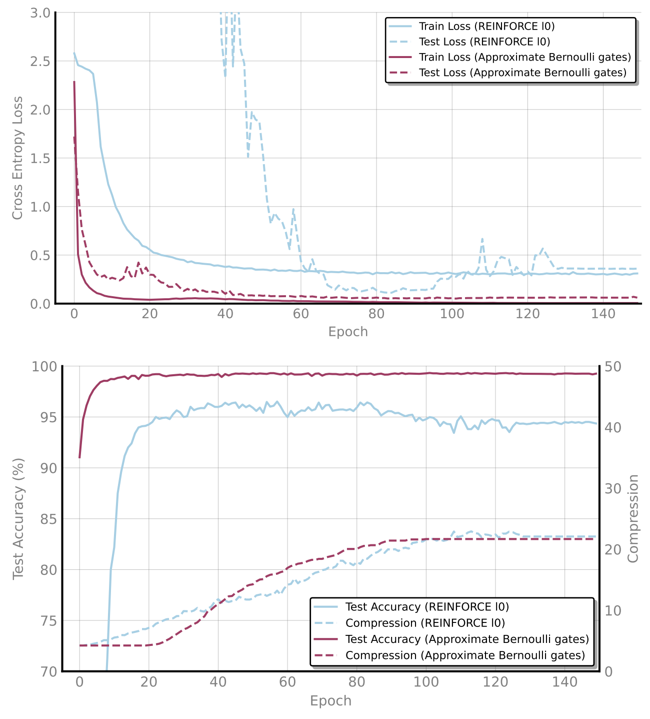

Figure 2: Train Loss / Test loss / Test Accuracy / Compression Ratio v.s. Epoch.

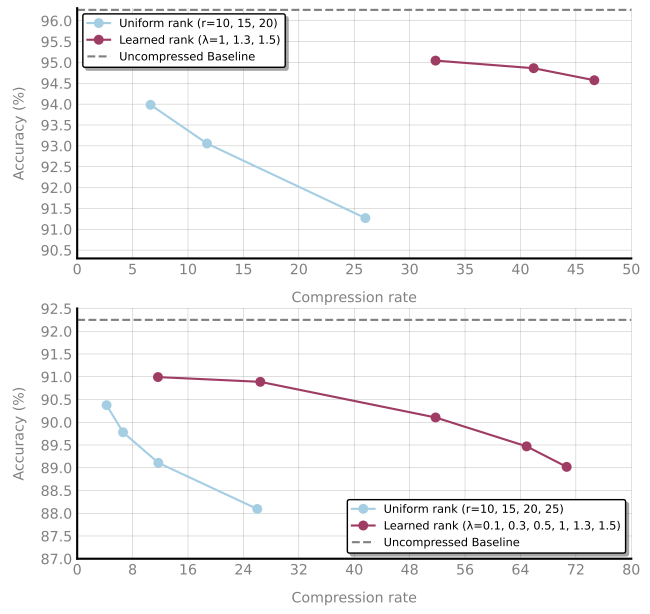

Figure 3: LeNet-5 on Kuzushiji-MNIST (the first figure) and Fashion-MNIST (the second figure). Each data point is averaged from five runs.

consistent with those in the paper. The results are shown in Table 4 and Fig. 2. Due to the large variance in the gradient, REINFORCE-ℓ_0 is unstable during training despite that it uses more Monte Carlo samplings. The final prediction accuracy of REINFORCE-ℓ_0 is worse than our proposed method. The results show the superiority of the approximate Bernoulli gates in stability and computational efficiency for optimizing the ℓ_0 rank selection formulation.

3 Additional Experiments

3.1 MNIST Drop-In Replacements

We tested LeNet-5 on [1]Kuzushiji-MNIST and [2]Fashion-MNIST datasets. They are two MNIST-like datasets, each consisting of 60k/10k 28×28 grayscale images. Kuzushiji-MNIST chooses one character to represent each of the 10 rows of Hiragana. Fashion-MNIST consists of fashion items, with each associated with a label from 10 classes. We use the same TR settings as for the MNIST dataset in the paper with different choices of λ (shown in the legend of Fig. 3). The results are shown in Fig. 3. Compared with the uniform rank TR baselines, the proposed ℓ_0-regularized models achieve a better accuracy-compression trade-off on both datasets.

3.2 Hyperparameter Sensitivity Analysis

The proposed method introduces two hyperparameters, which are the variance σ of the injected noise for the approximate Bernoulli gates and the coefficient for the regularization term λ. Specifically, σ and λ together control the strength of sparsity regularization. We investigate the sensitivity of the hyperparameters on the LeNet-5 models. Results are shown in Fig. 4. The variance σ controls the stochasticity the random variables bring, leading to performance loss due to the perturbation during the training phase. If σ is set too small, the regularization fails and the models are learned as full-rank. The best choice of σ for LeNet-5 on MNIST is around 0.5. Larger λ usually results in lower-rank models with worse accuracy.

4 Latency Analysis: Training and Inference Time

Training and inference time is one of the important metrics in the deep compression literature. In [Wang *et al.*, 2018], authors argue that the high compression that Tensor-Ring brings about in compressing neural networks is traded off with the increase of training and inference time. This is true in our case, while we observe that the inference time increased compared with the uncompressed baselines is comparatively acceptable and that the time slightly decreases compared with the Tensor-Ring baselines. The training time inevitably increases for our methods because the introduced parameters slightly complicate the training and we learn a compact model from an initially larger model.

We report the training and inference time of the uncompressed model, the Tensor-Ring model, and our model on ResNet-32. Results are shown in Table 5.

[1]https://github.com/rois-codh/kmnist

[2]https://github.com/zalandoresearch/fashion-mnist

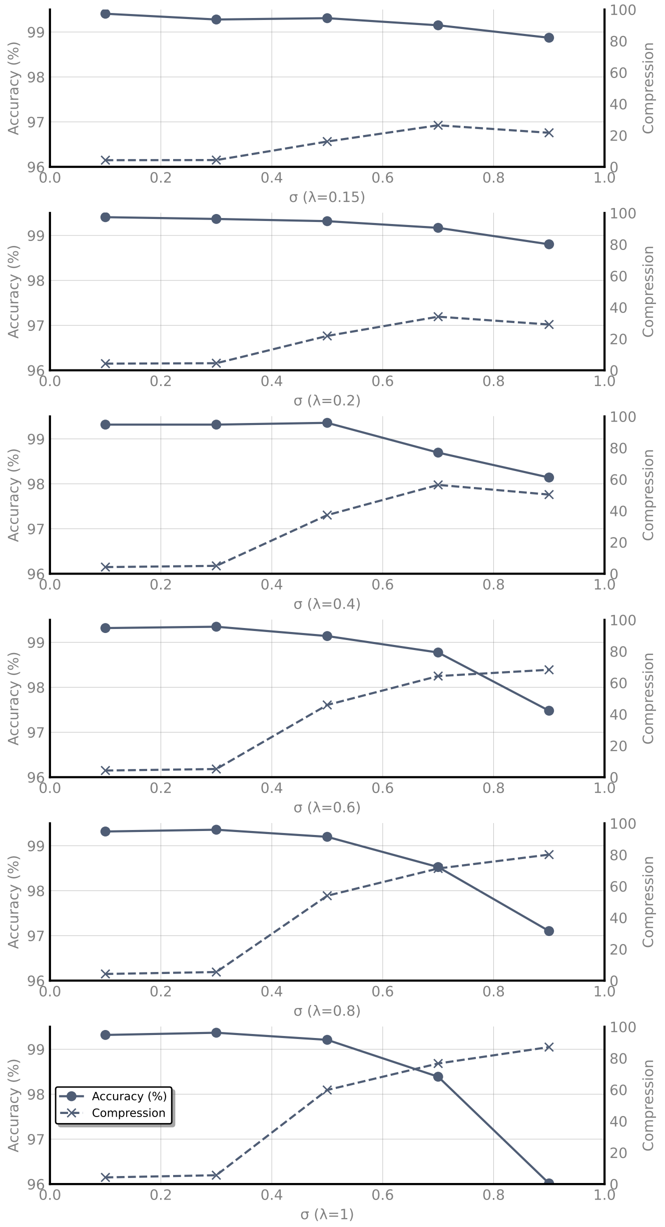

Figure 4: $\lambda \in \{0.15, 0.2, 0.4, 0.6, 0.8, 1\}$ is fixed in each subplot.

Method	Compression	Inference(s)	Train(s/batch)
Uncompressed	$1\times$	$\mathbf{3.31} \pm 0.01$	0.113
TRN($r = 15$)	$2.28\times$	4.13 ± 0.01	0.108
TRN($r = 10$)	$4.95\times$	4.12 ± 0.01	**0.106**
Ours	$\mathbf{5}\times$	4.09 ± 0.01	0.160

Table 5: Training time and inference time of ResNet-32 on CIFAR-10. INFERENCE is defined by the inference time of 10k test samples of CIFAR-10 with means and standard deviations from ten runs. TRAIN refers to the training time of a training batch, which is the average of the training times spent on batches within an epoch of 50k training samples.

References

[Cheng *et al.*, 2020] Zhiyu Cheng, Baopu Li, Yanwen Fan, and Yingze Bao. 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, 2020.

[Gao *et al.*, 2020] Ze-Feng Gao, Song Cheng, Rong-Qiang He, Zhi-Yuan Xie, Hui-Hai Zhao, Zhong-Yi Lu, and Tao Xiang. Compressing deep neural networks by matrix product operators. *Physical Review Research*, 2(2):023300, 2020.

[Kodryan *et al.*, 2023] Maxim Kodryan, Dmitry Kropotov, and Dmitry Vetrov. Mars: Masked automatic ranks selection in tensor decompositions. In *International Conference on Artificial Intelligence and Statistics*, pages 3718–3732. PMLR, 2023.

[Kossaifi *et al.*, 2019] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research*, 20(26):1–6, 2019.

[Li *et al.*, 2021] Nannan Li, Yu Pan, Yaran Chen, Zixiang Ding, Dongbin Zhao, and Zenglin Xu. Heuristic rank selection with progressively searching tensor ring network. *Complex & Intelligent Systems*, pages 1–15, 2021.

[Wang *et al.*, 2018] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.