BEYOND FLOPS IN LOW-RANK COMPRESSION OF NEURAL NETWORKS: OPTIMIZING DEVICE-SPECIFIC INFERENCE RUNTIME

Yerlan Idelbayev and **Miguel Á. Carreira-Perpiñán**, CSE, UC Merced

ICIP 2021

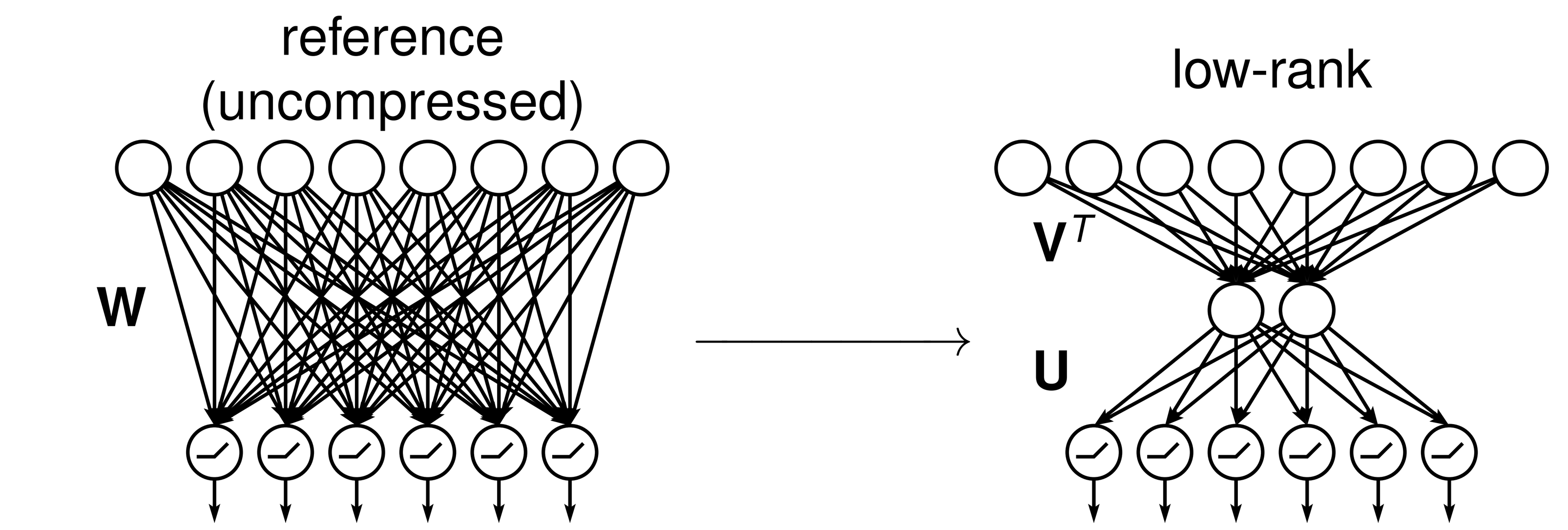
paper #2333

1 Abstract

We consider the problem of low-rank compression of the neural networks with the goal of optimizing the measured inference time. Given a neural network and a target device to run it, we want to find the matrix ranks and the weight values of the compressed model so that network runs as fast as possible on the device while having best task performance (e.g., classification accuracy). We first implement a simple yet accurate model of the on-device runtime (**part I**) and then we give a suitable formulation of the optimization problem (**part II**) involving the proposed runtime model. We validate our approach on various neural networks using an actual device.

https://github.com/UCMerced-ML/LC-model-compression

2 Introduction: Low-rank compression



When applying low-rank, we replace a matrix $\mathbf{W}_i$ with some rank-r matrix.

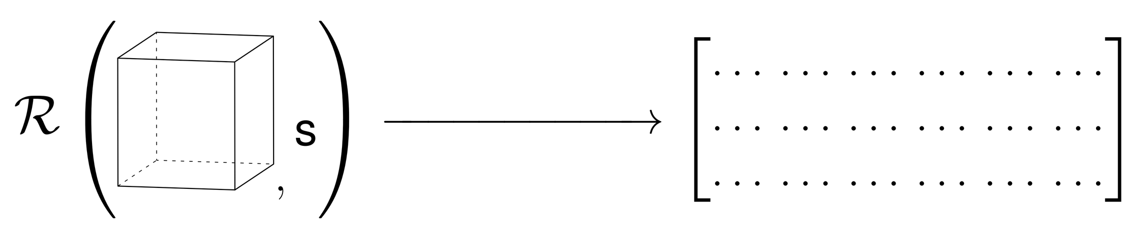
- Such matrix can be written as the product $\mathbf{UV}^T$, i.e., $\mathbf{W} = \mathbf{UV}^T$
- For small values of r this reduces FLOPs and storage
- Can achieve speed-up on any hardware (uses standard matrix-vector products)

3 Introduction: Non-matrix weights

Weights are not necessarily come as matrices, but often as tensors.

For example weights of convolutional layers are typically stored as NCHW or NHWC tensors.

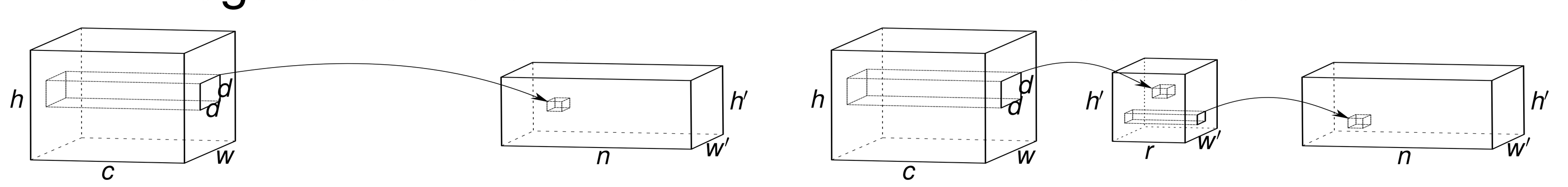
To apply low-rank to tensors, we reshape them into matrices!



Here $\mathcal{R}(\mathbf{W}, s)$ is a reshaping function: it takes a tensor and reshapes it into matrix according with shape s.

Some reshapes give a rise to efficient implementation as low-rank.

Regular convolution | Scheme 1



parameters: ncd^2 FLOPs: $ncd^2w'h'$ | parameters: $r(cd^2+n)$ FLOPs: $(cd^2+n)rw'h'$

4 Low-rank compression to target inference time?

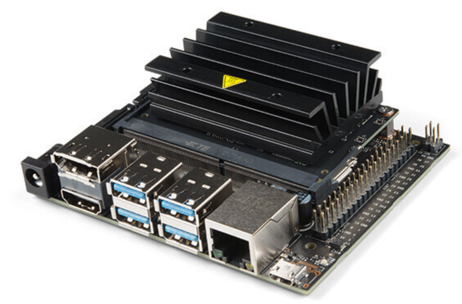
Historically, low-rank was used to reduce sizes and FLOPs of the models. But:

- fewer FLOPs not necessarily mean faster runtime!
- Can we select the ranks per each layer to minimize on-device runtime? (requires on-device measurements)

Hard problem There are combinatorial number of ranks and corresponding on-device measurements. We tackle it by

- building an accurate and fast to compute runtime model
- formulating a suitable optimization problem
- and giving an efficient optimization algorithm based on Learning-Compression framework [1, 2, 3, 4, 5]

Our target device :



Jetson Nano

CPU	4-core ARM Cortex-A57, 1.4 GHz
GPU	128 CUDA cores at 0.9 GHz
RAM	4 GB 64-bit LPDDR4, 1.6 GHz
OS	Ubuntu 18.04.5 LTS
Kernel	GNU/Linux 4.9.140-tegra
Storage	128 GB microSDXC memory card

5 Part I: Device Runtime Model

Let's define the runtime $\mathcal{R}(\mathbf{W})$ as the time to process a single image through a K-layer net with weights $\mathbf{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_K\}$.

- runtime is function of layer's ranks
- runtime can be directly measured on device, but there are R^K different configurations

We model the runtime as the sum of inferences through individual layers:

$$\mathcal{R}(\mathbf{W}) = \mathcal{R}(\mathbf{r}) = \mathcal{R}_1(r_1) + \mathcal{R}_2(r_2) + \dots + \mathcal{R}_K(r_K). \tag{1}$$

In reality $\mathcal{R}(\mathbf{W}) \leq$ RHS: when computational graph is executed optimally, some weights and inputs can be prefetched and layer-to-layer computations can be pipelined

This model allows to obtain runtime estimate $\mathcal{R}(\mathbf{W})$ much more efficiently:

- only need to measure R different rank configurations for each layer
- total required measurements: $R \times K$ (vs R^K)

6 Part II: Problem setup and optimization algorithm

Given a K-layer net trained on the loss $\mathcal{L}$ (e.g., cross-entropy), we formulate the following device-dependent rank selection problem:

$$\min_{\mathbf{W},\mathbf{r}} \quad \mathcal{L}(\mathbf{W}) + \lambda \mathcal{R}(\mathbf{r}) \qquad \text{s.t.} \quad \operatorname{rank}(\mathbf{W}_k) = r_k, \quad k = 1, \dots, K. \tag{2}$$

Here, the term $\lambda \mathcal{R}(\mathbf{r})$ controls the tradeoff between on-device inference speed and model loss.

We apply a penalty method and perform following alternating steps:

- The step over $\mathbf{W}$, which we call a learning (L) step, has the form of:

$$\min_{\mathbf{W}} \quad \mathcal{L}(\mathbf{W}) + \frac{\mu}{2} \sum_{k=1}^{K} \|\mathbf{W}_k - \mathbf{\Theta}_k\|^2.$$

- The step over $\{\mathbf{\Theta}, \mathbf{r}\}$, which we call a compression (C) step, has the form of:

$$\min_{\mathbf{\Theta},\mathbf{r}} \quad \frac{\mu}{2} \sum_{k=1}^{K} \|\mathbf{W}_k - \mathbf{\Theta}_k\|^2 + \lambda \mathcal{R}(\mathbf{r}) \qquad \text{s.t.} \quad \operatorname{rank}(\mathbf{\Theta}_k) = r_k, \quad k = 1, \dots, K$$

7 Simplified pseudocode

Please refer to main paper for a complete pseudocode.

for $\mu = \mu_1 < \mu_2 < \dots < \mu_T$

  $\mathbf{W} \leftarrow \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \frac{\mu}{2}\sum_{k=1}^{K} \|\mathbf{W}_k - \mathbf{\Theta}_k\|^2$ — L step

  for $k = 1, \dots, K$ — C step

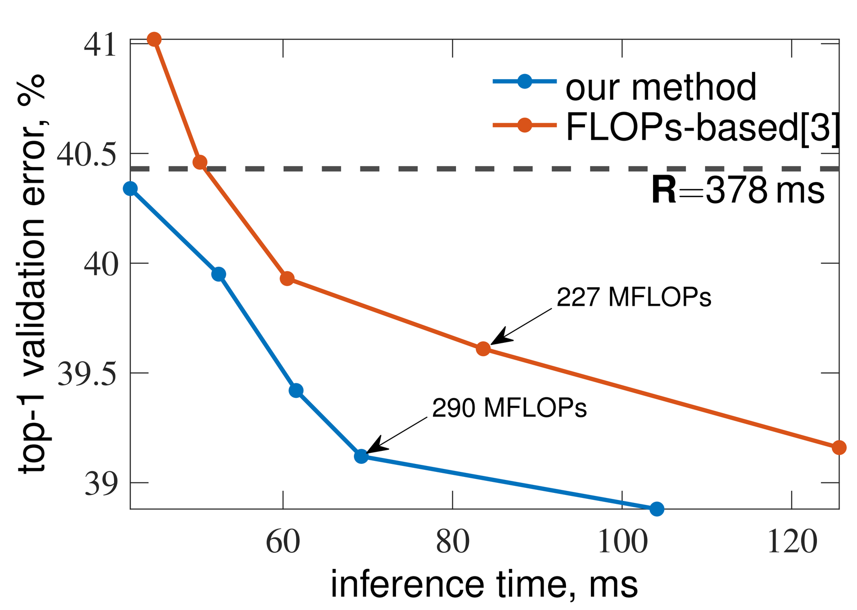
    $\mathbf{\Theta}_k, r_k \leftarrow \arg\min_{\mathbf{\Theta}_k, r_k} \frac{\mu}{2}\|\mathbf{\Theta}_k - \mathbf{W}_k\|^2 + \lambda \mathcal{R}_k(r_k)$

return $\mathbf{W}, \mathbf{\Theta}, \mathbf{r}$

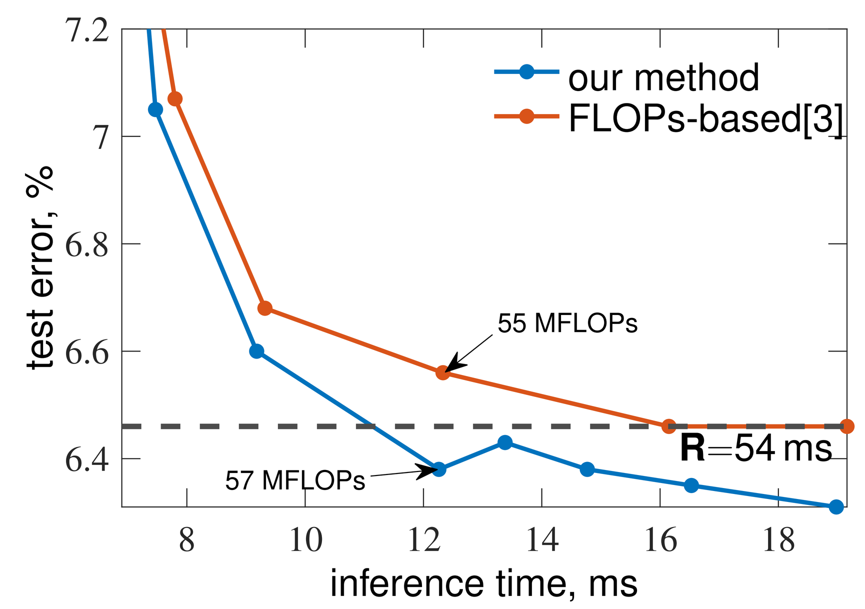
8 Experiments

Inference speed vs. error plot for our (blue) compressed AlexNet (top) and VGG16 models (bottom); for both networks, we additionally compare to the FLOPs based low-rank compression of [3] (given with red). The test errors and inference times of the reference models are indicated by horizontal dashed line labeled as **R**.

AlexNet on ImageNet



VGG16 on CIFAR10



	Model	MFLOPs	Infr. time	top-1 err	top-5 err
	reference (**R**)	1140	378.5 ms	40.43%	17.55%
	Caffe-AlexNet [6, 7]	727	328.7 ms	42.90%	19.80%
ours	$\lambda = 5.0 \times 10^{-3}$	421	104.1 ms	**38.88%**	**16.83%**
ours	$\lambda = 1.0 \times 10^{-2}$	290	69.2 ms	39.12%	17.03%
ours	$\lambda = 2.0 \times 10^{-2}$	186	**42.0 ms**	40.34%	17.64%
	low-rank AlexNet [3]	227	83.6 ms	39.61%	17.40%
	low-rank AlexNet [3]	**166**	50.2 ms	40.46%	17.71%
	ENC-AlexNet [8]	272	93.3 ms	43.40%	19.93%
	SqueezeNet 1.1 [9]	352	63.8 ms	42.90%	19.70%

9 References

[1] Miguel Á. Carreira-Perpiñán, "Model compression as constrained optimization, with application to neural nets. Part I: General framework," arXiv:1707.01209, July 5 2017.

[2] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev, ""Learning-compression" algorithms for neural net pruning," in *Proc. of the 2018 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'18)*, Salt Lake City, UT, June 18–22 2018, pp. 8532–8541.

[3] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "Low-rank compression of neural nets: Learning the rank of each layer," in *Proc. of the 2020 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'20)*, Seattle, WA, June 14–19 2020, pp. 8046–8056.

[4] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "A flexible, extensible software framework for model compression based on the LC algorithm," arXiv:2005.07786, May 15 2020.

[5] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "More general and effective model compression via an additive combination of compressions," in *Proc. of the 32nd European Conf. Machine Learning (ECML–21)*, Bilbao, Spain, Sept. 13–17 2021.

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," arXiv:1408.5093 [cs.CV], June 20 2014.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. 2012, vol. 25, pp. 1106–1114, MIT Press, Cambridge, MA.

[8] Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung, "Efficient neural network compression," in *Proc. of the 2019 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'19)*, Long Beach, CA, June 16–20 2019, pp. 12569–12577.

[9] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer, "SqueezeNet: AlexNet-level accuracy with 50*times* fewer parameters and <0.5MB model size," arXiv:1602.07360 [cs.CV], Nov. 4 2016.