OPTIMAL SELECTION OF MATRIX SHAPE AND DECOMPOSITION SCHEME FOR NEURAL NETWORK COMPRESSION

Yerlan Idelbayev *Miguel Á. Carreira-Perpiñán*

Department of Computer Science and Engineering, University of California, Merced

{yidelbayev, mcarreira-perpinan}@ucmerced.edu

http://eecs.ucmerced.edu

ABSTRACT

When applying the low-rank decomposition to neural networks, tensor-shaped weights need to be reshaped into a matrix first. While many matrix reshapes are possible, some of them induce a low-rank decomposition scheme that can be more efficiently implemented as a sequence of layers. This poses the following problem: how should one select both the matrix reshape and associated low-rank decomposition scheme in order to compress a neural network so that its implementation is as efficient as possible? We formulate this problem as a mixed-integer optimization over the weights, ranks, and decompositions schemes; and we provide an efficient alternating optimization algorithm involving two simple steps: a step over the weights of the neural network (solved by SGD), and a step over the ranks and decomposition schemes (solved by an SVD). Our algorithm automatically selects the most suitable ranks and decomposition schemes to efficiently reduce compression costs (e.g., FLOPs) of various networks.

***Index Terms*—** low-rank compression, rank selection, neural network compression, decomposition schemes

1. INTRODUCTION

With the enormous success of neural networks and their ever-increasing number of practical use cases, the problem of compressing deep nets has become an active and important area of research. Out of many compression forms, recently, the low-rank compression re-emerged as an efficient way to achieve both compressed in size and reduced in FLOPs neural networks [1, 2]. An ingredient that made low-rank more attractive for deep network compression is the research on an automatic way of setting the ranks of the decompositions [1–3], whereas previously, the ranks were fixed by hand [4, 5] or heuristically estimated [6–8]. To further improve the performance of the low-rank compression, we address one overlooked ingredient — the choice of the weight reshaping when applying the decomposition.

When applying the low-rank methods, we decompose the weight matrix as a product $\mathbf{U}\mathbf{V}^T$ of lower rank matrices. For fully connected layers, where weights are naturally in a matrix form, this parametrization is straightforward to apply. However, the weights of the convolutional layers come as tensors; therefore, to apply a low-rank, we should first reshape its weights into a matrix. Formally, a *matrix reshape*[1] is a reordering of the items in a tensor $\mathcal{A}$ into a matrix $\mathbf{A}$ so that matrix $\mathbf{A}$ contains the same set of items as $\mathcal{A}$. There are many possible matrix reshapes of a tensor, and each reshape gives a rise to a different *low-rank decomposition scheme*. A few of the decomposition schemes can be implemented as a sequence of convolutional layers, allowing to harness the compressive and speeding-up properties of low rank; we give a detailed overview of these schemes in sec. 2.

In previous low-rank compression works, the decomposition scheme (e.g., scheme 1) was fixed and applied throughout the network. This is suboptimal in practice, as each scheme has its own advantages and should be selected accordingly, per layer. To address this, we want to select the best decomposition scheme for every layer of a given neural network. One simple but not an efficient solution to this selection problem is to try all possible combinations of decompositions using an off-the-shelf low-rank compression algorithm. For a K layer neural network with M different decomposition schemes to try, the total number of combinations is M^K: this number of trials is unmanageable with the average depth of modern neural networks having dozens of layers (e.g., ResNet-152 has $K = 152$ layers).

The problem exacerbates when we include the rank selection problem: clearly, the performance of the compressed network is a function of the rank as well as the decomposition scheme. How can we select the ranks and the decompositions schemes for every layer of the neural network to fit into our constraints yet avoiding the associated combinatorial explosion? We approach this problem by formulating a model selection problem that captures both rank and shape selection as part of the objective. We then show that our formulation is amenable to the alternating optimization and give an efficient algorithm to learn ranks, shapes, and weights of the neural network.

2. EFFICIENT LOW-RANK DECOMPOSITION SCHEMES OF CONVOLUTIONAL LAYERS INDUCED BY SPECIFIC MATRIX RESHAPES

A convolutional layer with weight tensor $\mathcal{W}$ of size $n \times c \times d \times d$ has n filters of size $d \times d$ and operates on an input with c channels. The output of the layer is an image with n channels, where each channel is a result of convolving the input with an individual filter (see Fig. 1). To apply a low-rank compression, we should reshape (matricize) a weight tensor $\mathcal{W}$ into an $a \times b$ shaped matrix having the same number of parameters, i.e., $ab = ncd^2 = P$. The complete list of possible (a, b)-sized reshapes can be enumerated by performing the prime number decomposition on P. However, only a few of such reshapes allow an efficient implementation of a low-rank convolutional layer in terms of a sequence of convolutions.

Two efficient low-rank decomposition schemes were proposed and used in the literature. The scheme 1 [3, 5, 7–9] applies low-rank to an $n \times cd^2$ sized reshape of the weight tensor $\mathcal{W}$. The scheme 2 [1,

[1]This operation is commonly known as matricization or matrix folding.

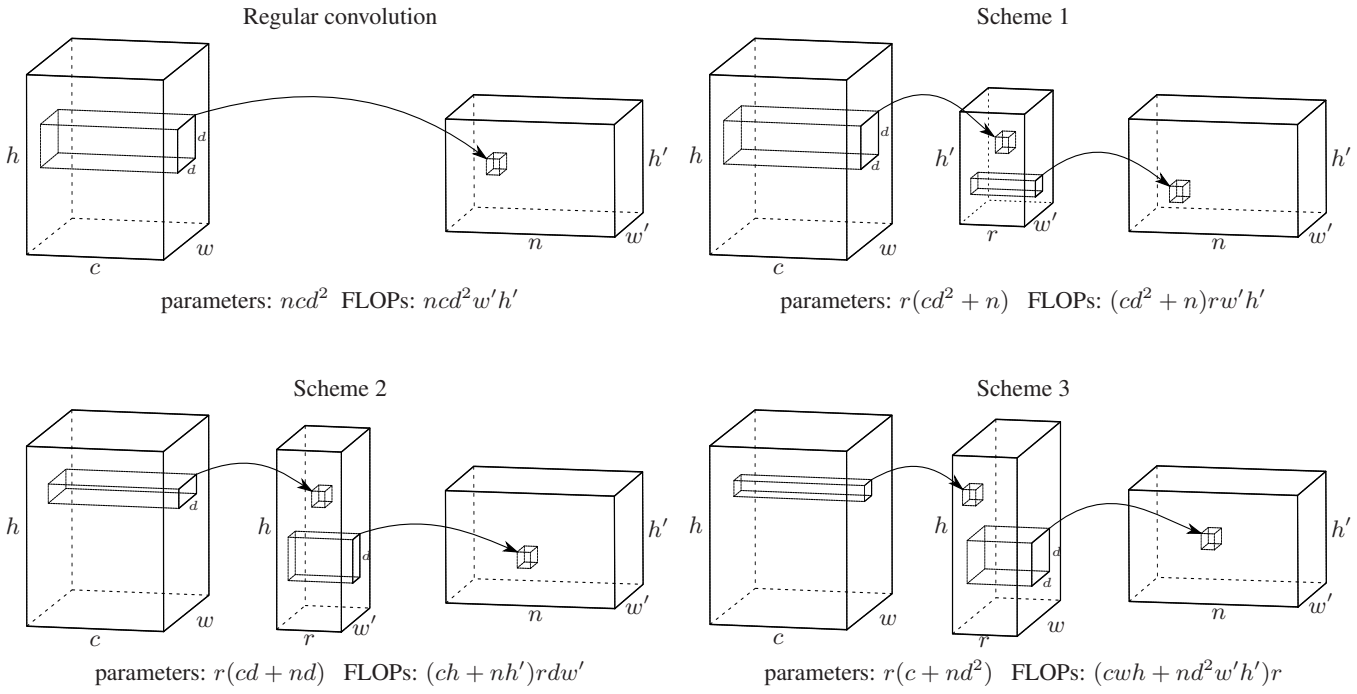

Fig. 1. Illustration of a regular convolution operation (top left) and its rank-r decompositions according to schemes 1, 2, and 3. The input to a layer has the shape of $c \times w \times h$ and is depicted as a cube with the appropriately marked dimensions. When the input is convolved with n filters of dimension $c \times d \times d$ (filters are not shown) it generates an output tensor of shape $n \times w' \times h'$. Each arrow represents a convolution of a portion of the input with a single filter, and points to the result of this convolution, a cube of size $1 \times 1 \times 1$. Low-rank decomposition schemes replace the convolutional layer with a sequence of two convolutions.

4, 6, 9] applies low-rank to an $nd \times cd$ reshape of $\mathcal{W}$. Both schemes can be efficiently implemented as a sequence of convolutional layers (Fig. 1). Another efficient scheme is possible if we apply the low-rank decomposition to an $nd^2 \times c$ sized reshape of the weight tensor, which we call scheme 3. To the best of our knowledge, scheme 3 has not been used in the literature yet.

3. PROBLEM FORMULATION AND AN OPTIMIZATION ALGORITHM

Assume we are given a K-layer neural network trained to minimize a task loss L (e.g., cross-entropy) over its weights $\mathbf{W} = \{\mathcal{W}_1, \dots, \mathcal{W}_K\}$ where the $\mathcal{W}_k$ is the weight tensor of the layer k. Let as denote the matrix reshape of the tensor $\mathcal{W}_k$ that induces the low-rank decomposition scheme s as $\mathcal{R}(\mathcal{W}_k, s)$, and the actual reshaped matrix as $\mathbf{\Theta}_k = \mathcal{R}(\mathcal{W}_k, s)$. We want to select the best scheme and rank for each layer to optimize the tradeoff between the model loss and a compression cost $\mathcal{C}(\mathbf{\Theta}, \mathbf{r})$. To achieve this goal, we impose the low-rank structure on the reshaped weights of each layer via explicit rank constraints on the corresponding $\mathbf{\Theta}_k$-terms, and form the following model selection problem over the ranks $\mathbf{r} = \{r_1, \dots, r_K\}$, decomposition schemes $\mathbf{s} = \{s_1, \dots, s_K\}$, and weights $\mathbf{W}$:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{\Theta},\mathbf{r},\mathbf{s}} \quad & \mathcal{L}(\mathbf{W}) + \lambda \mathcal{C}(\mathbf{\Theta}, \mathbf{r}) \\ \text{s.t.} \quad & \mathbf{\Theta}_k = \mathcal{R}(\mathcal{W}_k, s_k), \\ & \operatorname{rank}(\mathbf{\Theta}_k) = r_k, \quad \forall k = 1 \dots K \end{aligned} \tag{1}$$

We control the amount of the compression (and subsequent tradeoff) via a parameter $\lambda > 0$; and the compression cost function $\mathcal{C}(\mathbf{\Theta}, \mathbf{r})$ will encourage having smaller models. It is up to the user to determine the optimal operating point wrt λ: usually multiple values are considered to select among a family of compressed models (see Fig. 2). We define the $\mathcal{C}(\mathbf{\Theta}, \mathbf{r})$ to be layerwise separable function of:

$$\mathcal{C}(\mathbf{\Theta}, \mathbf{r}) = \mathcal{C}(\mathbf{\Theta}_1, r_1) + \dots + \mathcal{C}(\mathbf{\Theta}_K, r_K). \tag{2}$$

Such cost function can handle multiple targets of interest:

- It can target the storage and FLOPs of the compressed model, as both of these are the functions of the rank and can be written as $C(\mathbf{\Theta}_k, r_k) = \alpha \times r_k$ for some constant α (see Fig. 1).
- It can target the nuclear norm [10] of weight matrices instead: $\mathcal{C}(\mathbf{\Theta}_k, r_k) = \|\mathbf{\Theta}_k\|_*$. Such penalty has been well studied in the compressed sensing field and known to have low-rank inducing properties.

3.1. Optimization algorithm

The problem (1) is discrete over the ranks, schemes, and reshapes, but continuous over the weights, which makes it a challenging optimization problem. Fortunately, the formulation of (1) is in *learning-compression* form [11] which admits alternating optimization solution [2, 11–16]. To obtain the algorithm, we equivalently reformulate the problem (1) using the quadratic penalty [17]. (Here we use quadratic penalty for brevity of presentation. In practice we use augmented Lagrangian version which has an additional step over the Lagrange multipliers.) We apply the penalties only to the reshaping

Algorithm 1 Pseudocode to jointly learn weights, ranks, and low-rank decomposition schemes to compress a network

input K-layer neural net with weights $\mathbf{W} = \{\mathcal{W}_1, \dots, \mathcal{W}_K\}$, hyperparameter λ, cost function $\mathcal{C}$, set of reshaping schemes $\{\mathcal{S}_1, \dots, \mathcal{S}_m\}$

$\mathbf{W} = (\mathcal{W}_1, \dots, \mathcal{W}_K) \leftarrow \arg\min_{\mathbf{W}} L(\mathbf{W})$ reference net

$\mathbf{s} = (s_1, \dots, s_K) \leftarrow (\mathcal{S}_1, \dots, \mathcal{S}_1)$ decomposition schemes

$\mathbf{r} = (r_1, \dots, r_K) \leftarrow \mathbf{0}$ ranks

$\mathbf{\Theta} = (\mathbf{\Theta}_1, \dots, \mathbf{\Theta}_K) \leftarrow \mathbf{0}$ reshaped weights

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

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

for $k = 1, \dots, K$ C step

for $s'_k = \mathcal{S}_1, \dots, \mathcal{S}_m$

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

if $(\mathbf{\Theta}'_k, r'_k, s'_k)$ has a lower C-step objective **then**

$(\mathbf{\Theta}_k, r_k, s_k) \leftarrow (\mathbf{\Theta}'_k, r'_k, s'_k)$

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

constraints of $\mathbf{\Theta}_k = \mathcal{R}(\mathcal{W}_k, s_k)$ and optimize the following while driving $\mu \to \infty$:

$$\min_{\mathbf{W}, \mathbf{\Theta}, \mathbf{r}, \mathbf{s}} \quad \mathcal{L}(\mathbf{W}) + \lambda \mathcal{C}(\mathbf{\Theta}, \mathbf{r}) + \frac{\mu}{2} \sum_{k=1}^{K} \|\mathbf{\Theta}_k - \mathcal{R}(\mathcal{W}_k, s_k)\|_F^2 \quad \text{s.t.} \quad \operatorname{rank}(\mathbf{\Theta}_k) = r_k, \quad \forall k = 1 \dots K. \tag{3}$$

The reformulation (3) allows to efficiently optimize the problem by alternating over $\mathbf{W}$ and $\{\mathbf{\Theta}, \mathbf{r}, \mathbf{s}\}$. This results into learning (L) and compression (C) steps:

- **L step**: $\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \frac{\mu}{2} \sum_{k=1}^{K} \|\mathbf{\Theta}_k - \mathcal{R}(\mathcal{W}_k, s_k)\|_F^2$

  The step over $\mathbf{W}$ is fully differentiable, and has a simple ℓ_2-regularized form of the neural network training. We will use SGD to solve this step.

- **C step**: $\min_{\mathbf{\Theta}, \mathbf{r}, \mathbf{s}} \lambda \mathcal{C}(\mathbf{\Theta}, \mathbf{r}) + \frac{\mu}{2} \sum_{k=1}^{K} \|\mathbf{\Theta}_k - \mathcal{R}(\mathcal{W}_k, s_k)\|_F^2$

  The step over $\mathbf{\Theta}$, $\mathbf{r}$ and $\mathbf{s}$ is still a mixed-integer optimization problem, however, it admits an efficient solution depending on the form of compression cost $\mathcal{C}$.

The alternation of L and C steps guarantee a monotonic decrease of the objective function. More importantly, it confines the combinatorial search over the ranks and decomposition schemes to a subproblem that does not involve the network loss (which typically requires iteration over a large dataset).

3.2. Solution of the C step

The layerwise separable cost function (2) splits the C-step problem into subproblems over each layer k:

$$\min_{\mathbf{\Theta}_k, r_k, s_k} \quad \lambda \mathcal{C}(\mathbf{\Theta}_k, r_k) + \frac{\mu}{2} \|\mathbf{\Theta}_k - \mathcal{R}(\mathcal{W}_k, s_k)\|_F^2 \quad \text{s.t.} \quad \operatorname{rank}(\mathbf{\Theta}_k) = r_k. \tag{4}$$

For a fixed decomposition scheme s_k the solution of this optimization problem over $\mathbf{\Theta}_k, r_k$ is known in closed form for multiple costs $\mathcal{C}$. For the storage and FLOPs costs, the solution involves SVD and enumeration over the ranks [2]. For the nuclear-norm cost, the solution involves singular value shrinkage [18]. Therefore, to find the solution of (4) we iterate over possible schemes, and select the triplet $(\mathbf{\Theta}_k, r_k, s_k)$ attaining the minimum loss of (4). See Alg. 1 for the full pseudocode.

4. EXPERIMENTAL EVALUATION AND DISCUSSION

We demonstrate the power of jointly training weights, ranks, and decompositions schemes by compressing various models on different datasets. We compress the Caffe version of LeNet5 on MNIST dataset, batch normalized VGG16 on CIFAR10, and AlexNet on ImageNet. While our algorithm can handle different compression costs

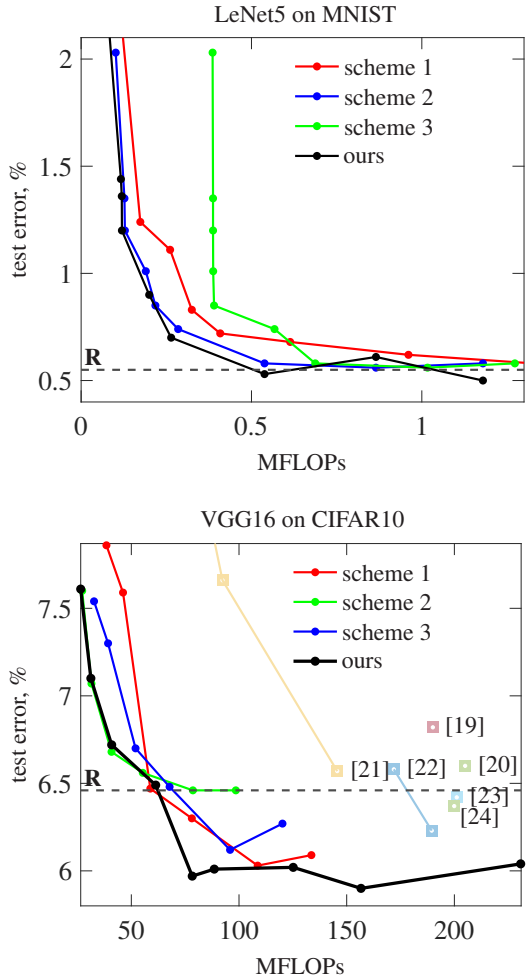

Fig. 2. Compression of LeNet5 and VGG16 networks trained on MNIST and CIFAR10 datasets using automatic rank selection with fixed decomposition schemes 1, 2, 3 and comparison to our approach where schemes and ranks are learned jointly. For each scheme (and our method) we run multiple compression and generate a family of model which we plot as a curve. We additionally plot some of the available compression results in the literature using square markers. Horizontal dashed lines marked with **R** indicate the test-error of reference (uncompressed) networks.

	test error	FLOPs	parameters	The selected scheme and rank over layers														
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\lambda = 2.0 \times 10^{-5}$	5.90%	156M	4.8M	$\mathcal{S}_1$ 16	$\mathcal{S}_2$ 32	$\mathcal{S}_2$ 71	$\mathcal{S}_2$ 97	$\mathcal{S}_3$ 116	$\mathcal{S}_2$ 238	$\mathcal{S}_2$ 263	$\mathcal{S}_3$ 254	$\mathcal{S}_2$ 292	$\mathcal{S}_2$ 172	$\mathcal{S}_2$ 122	$\mathcal{S}_2$ 99	$\mathcal{S}_2$ 105	31	11	9
$\lambda = 7.5 \times 10^{-5}$	5.97%	78M	2.5M	$\mathcal{S}_1$ 15	$\mathcal{S}_2$ 20	$\mathcal{S}_2$ 43	$\mathcal{S}_2$ 53	$\mathcal{S}_2$ 113	$\mathcal{S}_2$ 110	$\mathcal{S}_2$ 116	$\mathcal{S}_2$ 239	$\mathcal{S}_2$ 124	$\mathcal{S}_2$ 79	$\mathcal{S}_2$ 72	$\mathcal{S}_2$ 74	$\mathcal{S}_2$ 89	23	10	9

Table 1. The final selected ranks and reshaping schemes (across layers) for some of the compressed VGG16 models on CIFAR10 using our algorithm. The VGG16 network has 16 layers: layers 1–13 are convolutional, for which we selected both schemes and ranks, and layers 14–16 are fully connected, for which we select only the ranks. We denote schemes as: $\mathcal{S}_1$ (scheme 1), $\mathcal{S}_2$ (scheme 2), $\mathcal{S}_3$ (scheme 3). Notice how selected scheme and ranks change when we use a higher value of λ, e.g., the selected scheme for layer 5 changed from $\mathcal{S}_3$ to $\mathcal{S}_2$ The reference VGG16 has the test error of 6.45% with 317 MFLOPs and 15.2M parameters.

$\mathcal{C}$, we run our experiments with $\mathcal{C}$ targeting the resulting MFLOPs reduction of the models. Our algorithm is initialized from reasonably well pre-trained reference models and run with different values of λ to explore the entire error-FLOPS tradeoff space. We allow the algorithm to select over schemes 1, 2, and 3 of sec 2. Overall, the total runtime of compression does not take more than $3\times$ the time spend on training the reference model in the first place. We run L and C steps in a total of T times with the μ value of $\mu_{\text{init}} \times b^t$ at step t, and perform finetuning for 10–20 epochs afterwards. All L steps are optimized using SGD with a momentum of 0.9 and the initial learning rate is decayed by 0.99 after each epoch. The exact values are as follows:

- **LeNet5**: $T = 30$, $\mu_{\text{init}} = 0.001$, $b = 1.1$. Each L step runs for 30 epochs with a learning rate of 0.02.
- **VGG16**: $T = 60$, $\mu_{\text{init}} = 0.0002$, $b = 1.2$. Each L step runs for 15 epochs with a learning rate 0.0001.
- **AlexNet**: $T = 30$, $\mu_{\text{init}} = 0.001$ and $b = 1.1$. Each L step is run for 15 epochs with a learning rate 0.0005.

We plot our rank-and-scheme-optimized LeNet5 and VGG16 models on Fig 2. To give a perspective on whether the scheme selection improves the overall compression, we additionally run our algorithm with a fixed reshape (using schemes 1, 2, or 3) throughout the net, effectively disabling the scheme selection. As expected, low-rank networks trained with only a fixed scheme do not achieve competitive error-FLOPs tradeoff when compared to the scheme optimized counterparts. For instance, our scheme optimized low-rank LeNet5-s have no accuracy loss up to 0.6 MFLOPs, which corresponds to $\times 4.25$ speed-up; and our compressed VGG16 nets do not experience accuracy drop until reaching models with 61 MFLOPs ($\times 5.1$ speed-up). In fact, for VGG16 we see a substantial improvement in test error for moderately compressed models: our 78 MFLOPs network has a test error of 5.97%, which is a 0.54% improvement wrt reference model. We also plot recent results from the structured pruning literature (as square markers) that reduce the FLOPs count of VGG16 [19–24]. Our results *achieve significantly better error-FLOPs tradeoff* compared to low-rank compression using individual reshaping schemes and when compared to structured pruning results as well.

To illustrate the differences in selected ranks and schemes of our compressed models, we report some of the final architectures for VGG16 in Table 1. We notice non-trivial changes in both ranks and schemes of the final architectures: while a network with 156 MFLOPs has a mix of schemes 1, 2, and 3, the 78 MFLOPs network only uses schemes 1 and 2.

	MFLOPs	top-1	top-5
Caffe-AlexNet	724	42.70	19.80
Tai et al. [6], scheme 2	185	—	20.34
Wen et al. [7], scheme 1	269	—	20.14
Kim et al. [1], scheme 2	272	43.40	20.10
Idelbayev & Carreira-Perpiñán [2], scheme 1	240	42.83	19.93
Idelbayev & Carreira-Perpiñán [2], scheme 2	151	42.69	19.83
ours, with $\lambda = 1.5 \times 10^{-5}$	179	41.64	19.22
ours, with $\lambda = 2.0 \times 10^{-5}$	156	42.44	19.65

Table 2. Our low-rank AlexNet models (with rank and scheme selection) and comparison to other low-rank results in the literature. We report top-1 and top-5 validation accuracy on ImageNet dataset and the FLOPs count of the final model.

For the AlexNet experiments, we report the achieved FLOPs count and top-1/top-5 validation errors in Table 2. Our rank-and-scheme-optimized AlexNet models achieve better error-FLOPs tradeoff than most of the low-rank compression results existing in the literature and comparable to rank-optimized AlexNets of [2] which use scheme 2 throughout the network. Interestingly, our algorithm selects the scheme-2 decomposition for all convolutional layers of AlexNet as well, suggesting that scheme 2 might be a good default option for a high-compression regime.

5. CONCLUSION

Traditionally, low-rank has been applied with a specific choice of the matrix reshape, essential fixing the decomposition scheme. Instead, we formulated the problem of jointly optimizing weights, ranks, and decomposition schemes to optimally compress the neural network, and gave an optimization algorithm that manages to address the exponentially large search space over the ranks and schemes. We experimentally validated that optimizing over decomposition schemes significantly improves the error-compression tradeoff when compared to any single fixed scheme. At the same time, we found that some schemes tend to perform better than others, and can be used as a good default option for low-rank compression.

5.1. Acknowledgments

We thank NVIDIA Corporation for several GPU donations.

6. REFERENCES

[1] 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.

[2] 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.

[3] Chong Li and C. J. Richard Shi, "Constrained optimization based low-rank approximation of deep neural networks," in *Proc. 15th European Conf. Computer Vision (ECCV'18)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., Munich, Germany, Sept. 8–14 2018, pp. 746–761.

[4] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. of the 25th British Machine Vision Conference (BMVC 2014)*, Michel Valstar, Andrew French, and Tony Pridmore, Eds., Nottingham, UK, Sept. 1–5 2014.

[5] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems (NIPS)*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. 2014, vol. 27, pp. 1269–1277, MIT Press, Cambridge, MA.

[6] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E, "Convolutional neural networks with low-rank regularization," in *Proc. of the 4th Int. Conf. Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, May 2–4 2016.

[7] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Coordinating filters for faster deep neural networks," in *Proc. 17th Int. Conf. Computer Vision (ICCV'17)*, Venice, Italy, Dec. 11–18 2017.

[8] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016.

[9] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong, "Trained rank pruning for efficient deep neural networks," arXiv:1812.02402, Dec. 8 2018.

[10] Maryam Fazel, *Matrix Rank Minimization with Applications*, Ph.D. thesis, Stanford University, Mar. 2002.

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

[12] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev, "Model compression as constrained optimization, with application to neural nets. Part II: Quantization," arXiv:1707.04319, July 13 2017.

[13] 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.

[14] 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.

[15] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "More general and effective model compression via an additive combination of compressions," Submitted, 2020.

[16] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "Neural network compression via additive combination of reshaped, low-rank matrices," in *Proc. of the 2021 Data Compression Conference (DCC'21)*, Snowbird, UT, Mar. 23–26 2021.

[17] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.

[18] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[19] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian, "Variational convolutional neural network pruning," in *Proc. of the 2019 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'19)*, Long Beach, CA, June 16–20 2019, pp. 2780–2789.

[20] Hao Li, Asim Kadav, Igor Durdanovic, and Hans P. Graf, "Pruning filters for efficient ConvNets," in *Proc. of the 5th Int. Conf. Learning Representations (ICLR 2017)*, Toulon, France, Apr. 24–26 2017.

[21] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao, "HRank: Filter pruning using high-rank feature map," in *Proc. of the 2020 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'20)*, Seattle, WA, June 14–19 2020, pp. 1526–1535.

[22] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. of the 2019 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'19)*, Long Beach, CA, June 16–20 2019, pp. 2790–2799.

[23] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. of the 2019 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'19)*, Long Beach, CA, June 16–20 2019, pp. 4340–4349.

[24] Zehao Huang and Naiyan Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. 15th European Conf. Computer Vision (ECCV'18)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., Munich, Germany, Sept. 8–14 2018, pp. 304–320.