More General and Effective Model Compression via an Additive Combination of Compressions

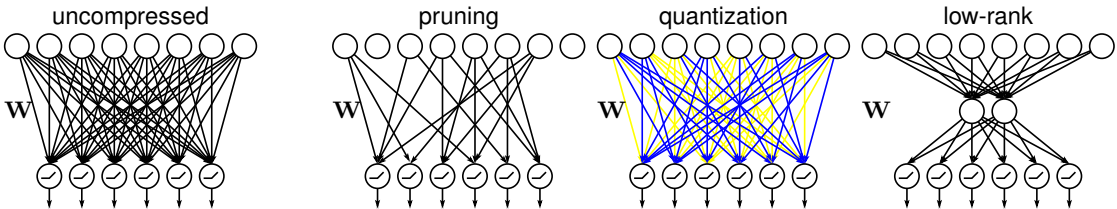
Yerlan Idelbayev and **Miguel Á. Carreira-Perpiñán**
Dept. CSE, University of California, Merced
http://eecs.ucmerced.edu

ECML PKDD 2021
VIRTUAL
13-17 September

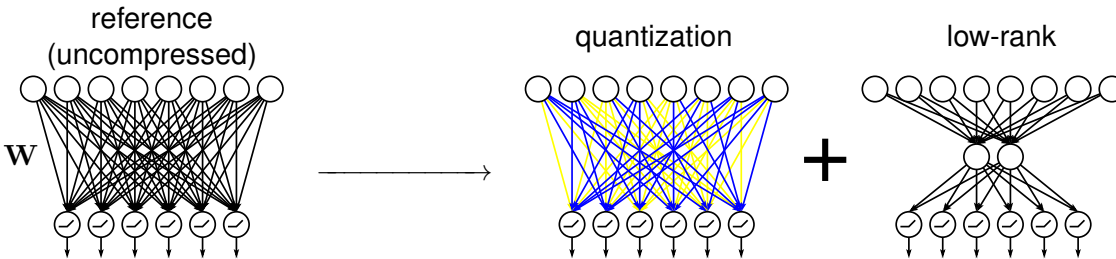
The code is available at:
https://github.com/UCMerced-ML/LC-model-compression

Introduction

Compression of neural networks become an important practical problem with plethora of works and approaches in recent years.



Introduction: Additive Combinations



We replace a matrix $\mathbf{W}$ with an additive combination of compressed matrices (e.g., quantization + low-rank)

- This allows to greatly improve the "power" of the scheme:
  - it is at least as powerful as each of the containing compressions
  - it allows techniques to help each other: i.e., correct where other scheme is failing
- How to make such compression happen for an arbitrary combination?
- We give a suitable formulation and optimization algorithm, followed by its empirical evaluation

Problem formulation

Given a model with weights $\mathbf{w}$ and loss L we formulate the problem of

$$\min_{\mathbf{w},\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \quad L(\mathbf{w}) \quad \text{s.t.} \quad \mathbf{w} = \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) + \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2). \tag{1}$$

- $\boldsymbol{\Delta}_i$ is a mapping from a low-dimensional (compressed) parameter $\boldsymbol{\theta}_i$ into original weight space
  - Low-rank: $\mathbf{W} = \mathbf{U}\mathbf{V}^T$ with $\mathbf{U}$ and $\mathbf{V}$ of rank r, so $\boldsymbol{\theta} = (\mathbf{U}, \mathbf{V})$.
  - Pruning: $\mathbf{w} = \boldsymbol{\theta}$ s.t. $\|\boldsymbol{\theta}\|_0 \leq \kappa$, so $\boldsymbol{\theta}$ is the indices of its nonzeros and their values.
  - Binarization: $w \in \{-1, +1\}$ or equivalently a scalar quantization with $\mathcal{C} = \{-1, +1\}$.
- Although our approach can handle any number of combinations, for brevity we discuss a combination of two compressions

Optimization algorithm

We apply a penalty method and obtain an equivalent formulation (with $\mu \to \infty$):

$$Q(\mathbf{w}, \boldsymbol{\theta}; \mu) = L(\mathbf{w}) + \frac{\mu}{2} \|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2 \tag{2}$$

Here, we use Quadratic Penalty for the ease of presentation, however, in practice we use augmented Lagrangian with the additional step over Lagrange multipliers

Let us now apply alternating optimization over variables $\mathbf{w}$ and $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$:

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

$$\min_{\mathbf{w}} \quad L(\mathbf{w}) + \frac{\mu}{2} \|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$$

  The problem wrt $\mathbf{w}$ is fully differentiable and has a form of loss + ℓ_2 penalty. We use SGD.

- The step over $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$, which we call a compression (C) step, has the form of:

$$\min_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2} \|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$$

  Actual compression step independent of NN weights and dataset.

Optimization algorithm: More on C step

Although our decomposition simplifies model compression into a problem of much smaller dimensions of

$$\min_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2} \|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$$

Its optimization wrt $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ can still be challenging. Therefore, we apply alternating optimization one more time:

$$\begin{aligned} \boldsymbol{\theta}_1 &= \arg\min_{\boldsymbol{\theta}} \|(\mathbf{w} - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)) - \boldsymbol{\Delta}_1(\boldsymbol{\theta})\|^2 \\ \boldsymbol{\theta}_2 &= \arg\min_{\boldsymbol{\theta}} \|(\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1)) - \boldsymbol{\Delta}_2(\boldsymbol{\theta})\|^2 \end{aligned} \tag{3}$$

These steps now can be computed much easier: for low-rank it requires SVD, for pruning it needs thresholding, etc.

Optimization algorithm (pseudocode)

input training data, neural net architecture with weights $\mathbf{w}$

$\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} L(\mathbf{w})$ — reference net

$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \leftarrow \arg\min_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2} \|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$ — init

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

  $\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} L(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$ — L step

  while alternation does not converge

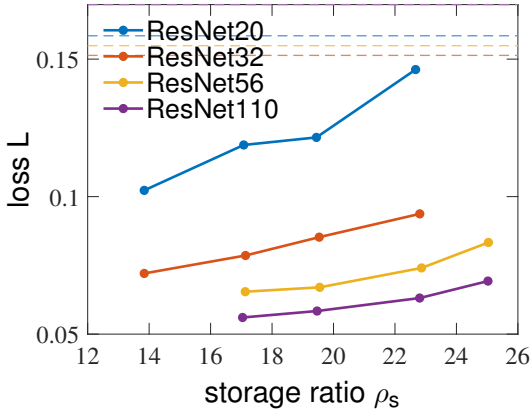
    $\boldsymbol{\theta}_1 \leftarrow \arg\min_{\boldsymbol{\theta}_1} \|(\mathbf{w} - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)) - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1)\|^2$ — C step

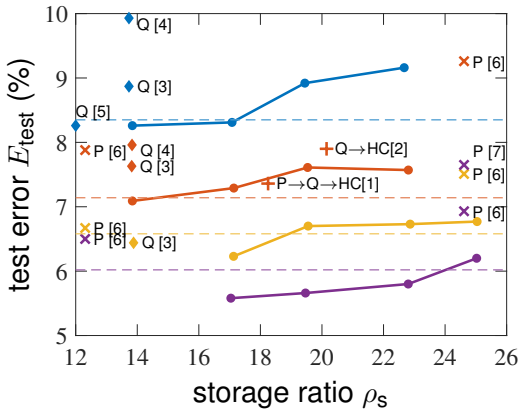
    $\boldsymbol{\theta}_2 \leftarrow \arg\min_{\boldsymbol{\theta}_2} \|(\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1)) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|^2$

  if $\|\mathbf{w} - \boldsymbol{\Delta}_1(\boldsymbol{\theta}_1) - \boldsymbol{\Delta}_2(\boldsymbol{\theta}_2)\|$ is small enough **then** exit the loop

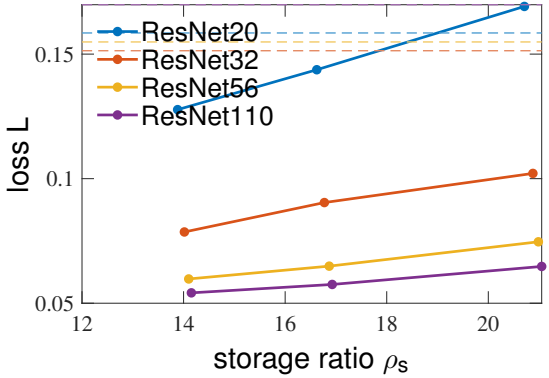
return $\mathbf{w}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2$

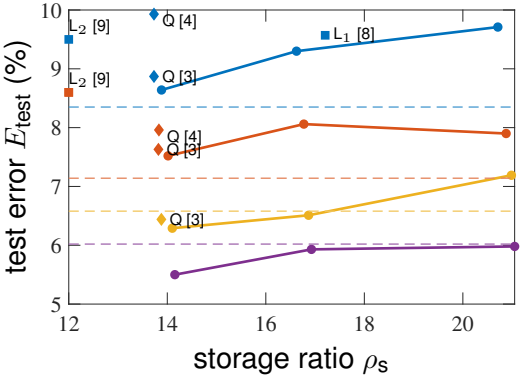
Experiments: Quantization + Pruning on ResNets (CIFAR10)





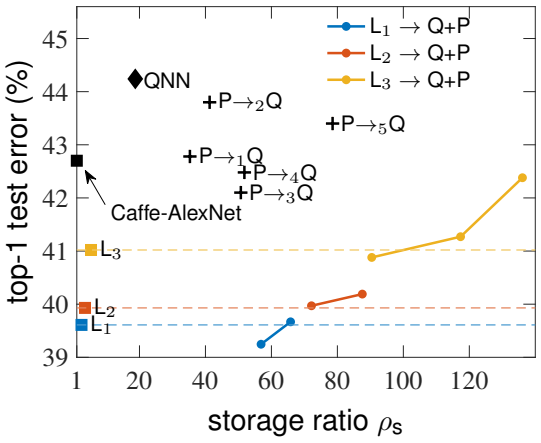
Experiments: Quantization + Low-rank on ResNets (CIFAR10)





Experiments: Q+P on already compressed AlexNets (ImageNet)

	Model	top-1	MBs	MFLOPs
	Caffe-AlexNet [10]	42.70	243.5	724
	R Low-rank AlexNet (L_1)	39.61	100.5	227
ours	$L_1 \rightarrow$ Q + P (0.25M)	39.67	3.7	227
ours	$L_1 \rightarrow$ Q + P (0.50M)	**39.25**	4.3	227
	R Low-rank AlexNet (L_2)	39.93	69.8	185
ours	$L_2 \rightarrow$ Q + P (0.25M)	40.19	2.8	185
ours	$L_2 \rightarrow$ Q + P (0.50M)	39.97	3.4	185
	R Low-rank AlexNet (L_3)	41.02	45.9	**152**
ours	$L_3 \rightarrow$ Q + P (0.125M)	42.38	**1.8**	**152**
ours	$L_3 \rightarrow$ Q + P (0.25M)	41.27	2.1	**152**
ours	$L_3 \rightarrow$ Q + P (0.50M)	40.88	2.7	**152**



Code is available online

Our code is written in `Python` using `PyTorch`, and we make it available as part our extensible model compression framework (under `BSD 3-clause` license):

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

Using the provided code, you will be able to:

- replicate all reported experiments
- compress your own models with our proposed scheme and many others.

But this library does much more than that. It is intended to support compression of an arbitrary model (not just neural nets) and an arbitrary compression technique. At the moment it offers the following:

- quantization (in various forms)
- pruning (in various forms)
- low-rank with automatic rank and/or scheme selection
- combinations of all the above

References

[1] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee, "Towards the limit of network quantization," in *Proc. of the 5th Int. Conf. Learning Representations (ICLR 2017)*, Toulon, France, Apr. 24–26 2017.

[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Advances in Neural Information Processing Systems (NIPS)*, I. Guyon, U. v. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, pp. 1141–1151, MIT Press, Cambridge, MA.

[3] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally, "Trained ternary quantization," in *Proc. of the 5th Int. Conf. Learning Representations (ICLR 2017)*, Toulon, France, Apr. 24–26 2017.

[4] Penghang Yin, Shuai Zhang, Jiancheng Lyu, Stanley Osher, Yingyong Qi, and Jack Xin, "BinaryRelax: A relaxation approach for training deep neural networks with quantized weights," *SIAM J. Imaging Sciences*, vol. 11, no. 4, pp. 2205–2223, 2018.

[5] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong, "Deep neural network compression with single and multiple level quantization," in *Proc. of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, New Orleans, LA, Feb. 2–7 2018, pp. 4335–4342.

[6] 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.

[7] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell, "Rethinking the value of network pruning," in *Proc. of the 7th Int. Conf. Learning Representations (ICLR 2019)*, New Orleans, LA, May 6–9 2019.

[8] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Coordinating filters for faster deep neural networks," in *Proc. 16th Int. Conf. Computer Vision (ICCV'17)*, Venice, Italy, Dec. 11–18 2017.

[9] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong, "TRP: Trained rank pruning for efficient deep neural networks," in *Proc. of the 29th Int. Joint Conf. Artificial Intelligence (IJCAI'20)*, Yokohama, Japan, Jan. 21–15 2020, pp. 977–983.

[10] 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.