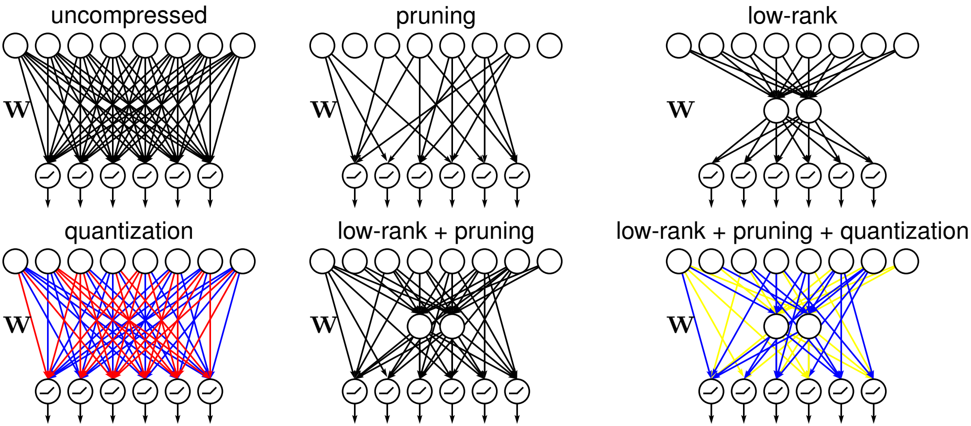
An Empirical Comparison of Quantization, Pruning and Low-rank Neural Network Compression using the LC Toolkit

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

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

Introduction: A need for NN compressions benchmark

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



Introduction: A need for NN compressions benchmark (cont.)

Which of these compressions is the best for a given model?

A simple solution: try different (possibly, off-the-shelf) compression schemes and algorithms and choose the best.

Unfortunately, it is often impossible due to:

- Fairness and objectivity of comparison: Compression methods have (very) different algorithmic base
- Availability of the code: Limited availability of readily usable code
- The choice of hyperparameters: Often, important hyperparamters are undisclosed or hard to select for a new task

Introduction: A need for NN compressions benchmark (cont.)

We want to compare pruning, low-rank, and quantization on multiple networks. To make the systematic comparison possible, we need a practical tool that is free of the aforementioned challenges.

We use recently proposed learning-compression (LC) algorithm, to lay a groundwork of such comparison work. The framework:

- seamlessly integrates number of compressions under the same roof
- is based on solid optimization principles and algorithms
- allows greater code reuse as model training is separated from compression
- combined, it allows an apples-to-apples comparison between techniques

Outline

This talk is structured in the following way:

- Overview of the LC algorithm
- Overview of the LC software
- Details of experimental setup & comparison
- Conclusion

Model compression as constrained optimization (MCCO)

General formulation:

uncompressed weights

low-dim. params

$$\min_{\mathbf{w},\boldsymbol{\Theta}} L(\mathbf{w}) + \lambda C(\boldsymbol{\Theta}) \quad \text{s.t.} \quad \mathbf{w} = \boldsymbol{\Delta}(\boldsymbol{\Theta})$$

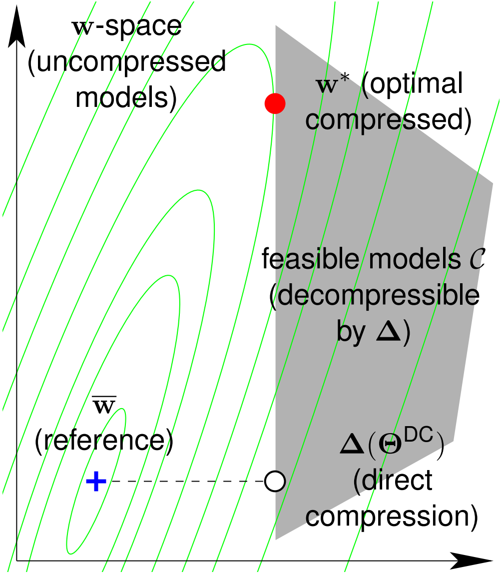
task loss

compression cost

decompression mapping $\boldsymbol{\Delta}\colon \boldsymbol{\Theta} \to \mathbf{w} \in \mathbb{R}^P$

We treat compression and decompression as mathematical mappings in parameter space.

The details of the compression technique are abstracted in $\boldsymbol{\Delta}(\boldsymbol{\Theta})$.



Optimization of MCCO, the LC algorithm

Reformulate using penalty method and optimize the following with $\mu \to \infty$:

$$\min_{\mathbf{w},\boldsymbol{\Theta}} \quad L(\mathbf{w}) + \lambda\, C(\boldsymbol{\Theta}) + \frac{\mu}{2}\left\|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\right\|^2$$

Alternation between $\mathbf{w}$ and $\boldsymbol{\Theta}$ gives the learning-compression (LC) algorithm:

- Learning (L) step:

$$\min_{\mathbf{w}} L(\mathbf{w}) + \frac{\mu}{2}\left\|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\right\|^2$$

  - This is a regular training of the model, but with a quadratic regularization term.
  - L step is independent of compression mechanism.
  - We will use SGD and standard NN software
- Compression (C) step:

$$\min_{\boldsymbol{\Theta}} \lambda\, C(\boldsymbol{\Theta}) + \frac{\mu}{2}\left\|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\right\|^2$$

  - For $\lambda = 0$, the C step becomes an optimal projection problem
  - The C step is independent of the dataset
  - Many well studied cases with fast solutions

LC algorithm: Pseudocode

input neural net with weights $\mathbf{w}$,
hyperparameter λ, cost function C

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

$\boldsymbol{\Theta} \leftarrow \mathbf{0}$ — compressed weights

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

$\quad \mathbf{w} \leftarrow \arg\min_{\mathbf{w}} L(\mathbf{w}) + \frac{\mu}{2} \|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\|_F^2$ — L step

$\quad \boldsymbol{\Theta} \leftarrow \arg\min_{\boldsymbol{\Theta}} \lambda\, C(\boldsymbol{\Theta}) + \frac{\mu}{2} \|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\|$ — C step

$\quad$ **if** $\|\mathbf{w} - \boldsymbol{\Delta}(\boldsymbol{\Theta})\| \approx 0$

$\quad\quad$ set $\mathbf{w} \leftarrow \boldsymbol{\Delta}(\boldsymbol{\Theta})$ and exit — early stopping

return $\mathbf{w}, \boldsymbol{\Theta}$

LC software: Library of implemented compressions

Type	Forms
Quantization	Adaptive Quantization into $\{c_1, c_2, \dots c_K\}$
	Binarization into $\{-1, 1\}$ and $\{-c, c\}$
	Ternarization into $\{-c, 0, c\}$
Pruning	ℓ_0-constraint (s.t., $\|\mathbf{w}\|_0 \leq \kappa$)
	ℓ_1-constraint (s.t., $\|\mathbf{w}\|_0 \leq \kappa$)
	ℓ_0-penalty ($\alpha\|\mathbf{w}\|_0$)
	ℓ_1-penalty ($\alpha\|\mathbf{w}\|_1$)
Low-rank	Low-rank compression to a given rank
	Low-rank with *automatic* rank selection for FLOPs reduction
	Low-rank with *automatic* rank selection for storage compression
Additive Combinations	Quantization + Pruning
	Quantization + Low-rank
	Pruning + Low-rank
	Quantization + Pruning + Low-rank

LC software: Easy exploration of compressions

Having an L-step implementation (you only need one), definition of compression is very simple:

quantize each layer with separate codebooks

```
compression_tasks = {
  Param(l1.weight): (AsVector, AdaptiveQuantization(k=2)),
  Param(l2.weight): (AsVector, AdaptiveQuantization(k=2)),
  Param(l3.weight): (AsVector, AdaptiveQuantization(k=2))
}
```

---

prune all but 5%

```
compression_tasks = {
  Param([l1.weight, l2.weight, l3.weights]):
    (AsVector, ConstraintL0Pruning(kappa=13310)) # 13310 = 5%
}
```

---

prune first layer, low-rank to second, quantize third

```
compression_tasks = {
  Param(l1.weight): (AsVector, ConstraintL0Pruning(kappa=5000)),
  Param(l2.weight): (AsIs,     LowRank(target_rank=10))
  Param(l3.weight): (AsVector, AdaptiveQuantization(k=2))
}
```

LC software: Source code and other features

Our code is written in `Python` using `PyTorch`, and open source 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 many available compression schemes

Our library is:

- modular and easily extensible
- only requires the L-step implementation: the regular learning of the model (using SGD)
- based on solid optimization principles
- single algorithm—many compressions
- time proven (development since 2017), with many publications [1–9]

Experiments: Setup

For our comparison we use the following forms of compressions:

- **Quantization**: adaptive quantization with a separate codebook of size K per each layer
- **Pruning**: constrained ℓ_0 based pruning, where we specify number of remaining nonzeros κ
- **Low-rank compression**: automatic rank-selection with cost function C targeting the number of parameters

Exact details of these compressions and the solution of corresponding C-step problems are in the main paper.

Experiments: Setup (cont.)

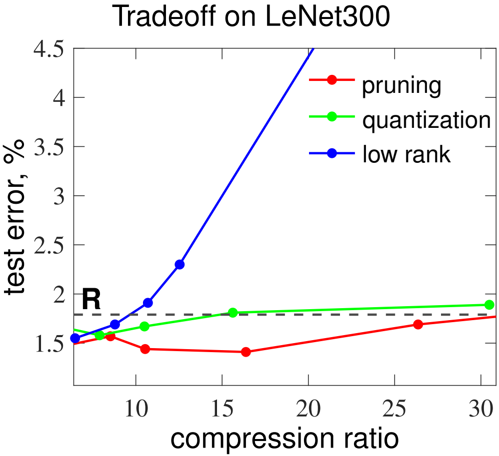
- For a given network, the L step of every compression experiment has same hyperparameters (learning rates, number of batches, etc) which allows us to make apples-to-apples comparisons across compressions.
- We vary the compression parameters (codebook size K, number of non-zeros κ, amount of penalty λ) to generate entire compression-error tradeoff curves.
- We report the compression ratio computed as:

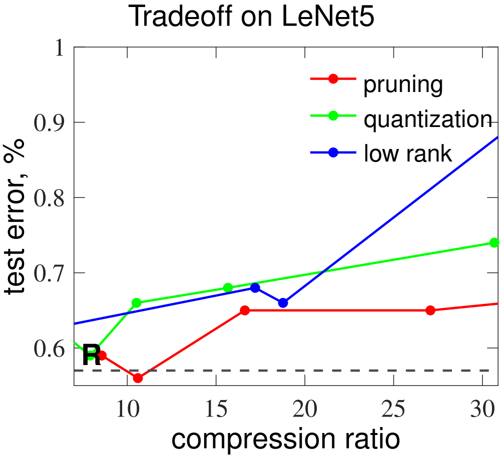
$$\text{compression ratio} = \frac{\text{uncompressed size}}{\text{compressed size}}$$

  where compressed size is computed as the actual amount of storage required to save the model to the disk.

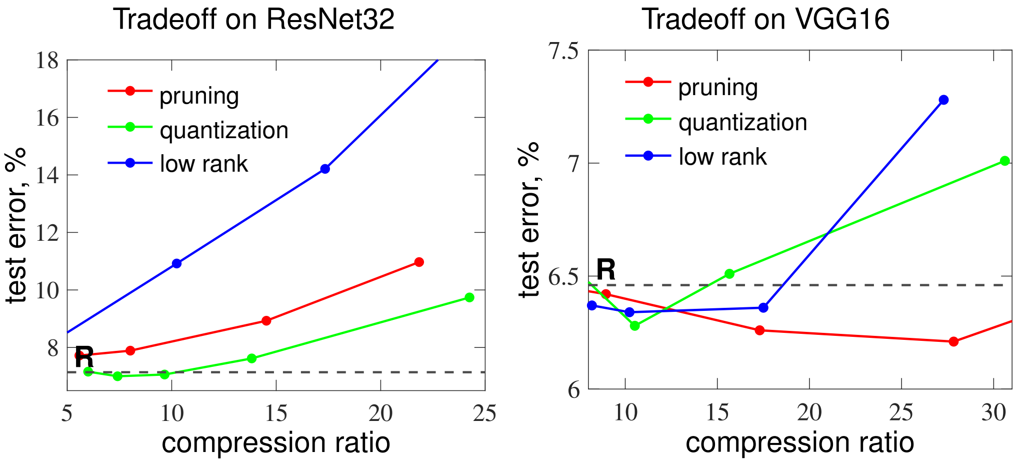
Exact details are in the paper.

Experiments: MNIST





Experiments: CIFAR10



Conclusion

- We argued and experimentally validated the necessity of fair and objective empirical comparison of different compression mechanisms
- The tool that allowed us to perform this comparison is the LC algorithm and its implementation — the LC Toolkit
- We have empirically observed that there is no single compression that universally outperforms other schemes
- However, for some networks pruning is a safe starting choice

References

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

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

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

[4] Y. Idelbayev and M. 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)*, pages 8046–8056, Seattle, WA, June 14–19 2020.

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

[6] Y. Idelbayev and M. Á. Carreira-Perpiñán. More general and effective model compression via an additive combination of compressions. Submitted, 2020.

[7] Y. Idelbayev and M. Á. Carreira-Perpiñán. Neural network compression via additive combination of reshaped, low-rank matrices. In *Proc. Data Compression Conference (DCC 2021)*, pages 243–252, Mar. 23–26 2021.

[8] Y. Idelbayev and M. Á. Carreira-Perpiñán. Optimal selection of matrix shape and decomposition scheme for neural network compression. In *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, Toronto, Canada, June 6–11 2021.

[9] Y. Idelbayev and M. Á. Carreira-Perpiñán. Beyond FLOPs in low-rank compression of neural networks: Optimizing device-specific inference runtime. In *IEEE Int. Conf. Image Processing (ICIP 2021)*, Anchorage, AK, Sept. 13–22 2021.