Optimal Selection of Matrix Shape and Decomposition Scheme for Neural Network Compression

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: Low-rank for neural nets

A $K$-layer neural net is a computational graph that computes output $f$ for an input $x$:

$$f(x; w) = \sigma(W_K \ldots \sigma(W_2\sigma(W_1x)))$$

The weights $W = \{W_1, \ldots, W_K\}$ are trained on a dataset of input-output pairs $(x, y)$ to make the network output $f(x; w)$ closer to the true output $y$:

**regression:**  $\min_w L(w) = \sum_{x,y} \|y - f(x; w)\|^2$

**classification:**  $\min_w L(w) = \sum_{x,y} \text{CrossEntropy}(y, f(x; w))$
We replace a matrix $W$ with some rank-$r$ matrix

- Such matrix can be written as the product $UV^T$, i.e., $W = UV^T$
  - For small values of $r$ this reduces FLOPs and storage
  - Can achieve speed-up on any hardware (uses standard matrix-vector products)
- If ranks are known, training is not hard: simply decompose and then use SGD
What happens with non-matrix weights?

Weights are not necessarily come as matrices. For example weights of convolutional layers are typically stored as NCHW or NHWC tensors. To apply low-rank, we reshape the tensors into matrices!

\[ \mathcal{R} \rightarrow \begin{bmatrix} \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} \]

*This is known as matricization in tensor algebra.*
More on reshapes: Efficient implementation

Some of the reshapes give a rise to efficient low-rank schemes.

- **Regular convolution**
  
  \[ \begin{align*}
  \text{parameters: } & ncd^2 \\
  \text{FLOPs: } & ncd^2 w'h'
  \end{align*} \]

- **Low-rank using scheme 1 reshape**
  
  \[ \begin{align*}
  \text{parameters: } & r(cd^2 + n) \\
  \text{FLOPs: } & (cd^2 + n)rw'h'
  \end{align*} \]

- **Low-rank using scheme 2 reshape**
  
  \[ \begin{align*}
  \text{parameters: } & r(cd + nd) \\
  \text{FLOPs: } & (ch + nh')rdw'
  \end{align*} \]

- **Low-rank using scheme 3 reshape**
  
  \[ \begin{align*}
  \text{parameters: } & r(c + nd^2) \\
  \text{FLOPs: } & (cwh + nd^2 w'h')r
  \end{align*} \]
Which reshapes are the best? How to select them optimally?

- Historically a single fixed scheme was used throughout the NN for the compression
  - This is suboptimal!
- Can we select the best scheme per each layer?
- The problem involves selecting ranks as well.

Hard problem. There are combinatorial number of configurations of ranks and schemes. We tackle it by

- formulating a suitable optimization problem
- and giving an efficient optimization algorithm based on Learning-Compression framework [1, 2, 3, 4, 5]
Problem formulation

Given a $K$-layer net with weights $\mathbf{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_K\}$ trained on the loss $\mathcal{L}$ (e.g., cross-entropy), we formulate the following rank and scheme selection problem:

\[
\begin{align*}
\min_{\mathbf{W}, \Theta, r, s} & \quad \mathcal{L}(\mathbf{W}) + \lambda \mathcal{C}(\Theta, r) \\
\text{s.t.} & \quad \Theta_k = \mathcal{R}(\mathbf{W}_k, s_k), \\
& \quad \text{rank} (\Theta_k) = r_k, \quad \forall k = 1, \ldots, K
\end{align*}
\]  

(1)

Here, the term $\lambda \mathcal{C}(\Theta, r)$ controls the amount of compression and can target a specific cost of interest like FLOPs or storage.
The cost $C$ is a function of the layers’ ranks:

$$C(\Theta, r) = C(\Theta_1, r_1) + \cdots + C(\Theta_K, r_K).$$

- Can target storage and FLOPs of the model. Typically, cost per layer has form $C(\Theta_k, r_k) = \alpha \times r_k$ for some constant $\alpha$.
- Can target the nuclear norm [6] of weight matrices instead: $C(\Theta_k, r_k) = \|\Theta_k\|_*$. 
Problem formulation

Given a $K$-layer net with weights $\mathbf{W} = \{W_1, \ldots, W_K\}$ trained on the loss $\mathcal{L}$ (e.g., cross-entropy), we formulate the following rank and scheme selection problem:

$$\min_{\mathbf{W}, \Theta, r, s} \mathcal{L}(\mathbf{W}) + \lambda \mathcal{C}(\Theta, r)$$

s.t. $\Theta_k = \mathcal{R}(W_k, s_k)$,

$$\text{rank}(\Theta_k) = r_k, \quad \forall k = 1, \ldots, K$$

This is a mixed-integer optimization problem over ranks and schemes. To solve it efficiently, we need an algorithm that can natively handle both integer and real-valued weights. To perform the optimization we use a learning-compression algorithm.
Let us apply a penalty method and obtain an equivalent formulation (with $\mu \to \infty$):

$$
\begin{align*}
\min_{W, \Theta, r, s} & \quad \mathcal{L}(W) + \lambda C(\Theta, r) + \frac{\mu}{2} \sum_{k=1}^{K} \| \Theta_k - R(W_k, s_k) \|_F^2 \\
\text{s.t.} & \quad \text{rank } (\Theta_k) = r_k, \quad \forall k = 1, \ldots, K.
\end{align*}
$$

(2)

Under standard assumptions, the stationary point of (2) at $\mu \to \infty$ is the stationary point (solution) of the original problem (1).
Let us now apply alternating optimization over variables $\mathbf{W}$ and $\{\Theta, r, s\}$:

- The step over $\mathbf{W}$, which we call a **learning (L) step**, has the form of:
  \[
  \min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \frac{\mu}{2} \sum_{k=1}^{K} \| \Theta_k - \mathcal{R}(\mathcal{W}_k, s_k) \|_F^2.
  \]

  Regular NN training independent of compression, typically solved by SGD

- The step over $\{\Theta, r, s\}$, which we call a **compression (C) step**, has the form of:
  \[
  \min_{\Theta, r, s} \lambda \mathcal{C}(\Theta, r) + \frac{\mu}{2} \sum_{k=1}^{K} \| \Theta_k - \mathcal{R}(\mathcal{W}_k, s_k) \|_F^2
  \]

  Actual compression step independent of NN weights and dataset.
Optimization algorithm: Solution of the C step

Due to the layerwise separability of cost function, the C-step problem separates over the layers into $K$ smaller problems:

$$\begin{align*}
\min_{\Theta_k, r_k, s_k} & \quad \lambda C(\Theta_k, r_k) + \frac{\mu}{2} \|\Theta_k - \mathcal{R}(\mathcal{W}_k, s_k)\|_F^2 \\
\text{s.t.} & \quad \text{rank}(\Theta_k) = r_k.
\end{align*}$$

(3)

Solution:

- For a fixed scheme $s_k$ the solution is known in closed form for multiple costs $C$
  - For storage and FLOPs, the solution involves SVD and enumeration [4]
  - For nuclear-norm cost, the solution involves singular value shrinkage [7]

- Therefore, to find global solution, we iterate over possible schemes and re-use steps for fixed scheme.
Optimization algorithm: Pseudocode

**input** $K$-layer neural net with weights $\mathbf{W} = \{\mathcal{W}_1, \ldots, \mathcal{W}_K\}$,
hyperparameter $\lambda$, cost function $\mathcal{C}$, set of reshaping schemes $\{S_1, \ldots, S_m\}$

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

reference net

$r = (r_1, \ldots, r_K) \leftarrow 0$
ranks

$s = (s_1, \ldots, s_K) \leftarrow (S_1, \ldots, S_1)$
decomposition schemes

$\Theta = (\Theta_1, \ldots, \Theta_K) \leftarrow 0$
reshaped weights

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

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

$L$ step

for $k = 1, \ldots, K$

for $s_k' = S_1, \ldots, S_m$

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

$C$ step

if $(\Theta_k', r_k', s_k')$ has a lower $C$-step objective then

$(\Theta_k, r_k, s_k) \leftarrow (\Theta_k', r_k', s_k')$

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

LeNet5 on MNIST

VGG16 on CIFAR10
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


