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

Introduction: Low-rank compression
When applying low-rank, 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

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!

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?
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]

Part I: Device Runtime Model
Let's define the runtime $\mathcal{R}(W)$ as the time to process a single image through a $K$-layer net with weights $W = \{W_1, \ldots, W_K\}$.
- runtime is function of layer's ranks
- runtime can be directly measured on device, but there's no different configurations

Model the runtime as the sum of inferences through individual layers:

\[
\mathcal{R}(W) = \mathcal{R}(r) = R_1(r_1) + R_2(r_2) + \cdots + R_K(r_K).
\]

This model allows to obtain runtime estimate $\mathcal{R}(W)$ much more efficiently:
- only need to measure $R$ different rank configurations for each layer
- total required measurements: $R \times K$ (vs $R^K$)

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:

\[
\frac{\min}{W_r} L(W) + \lambda \mathcal{R}(r)
\]
subject to $\text{rank}(W_k) = r_k$, $k = 1, \ldots, K$.
Here, the term $\lambda \mathcal{R}(r)$ controls the tradeoff between on-device inference speed and model loss.

We apply a penalty method and perform following alternating steps:
- The step over $W$, which we call a learning (L) step, has the form of:

\[
\frac{\min}{W} L(W) + \frac{\lambda}{k} \sum_{k=1}^K \|W_k - \Theta_k\|^2.
\]

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

\[
\frac{\min}{W} \left[ \sum_{k=1}^K \|W_k - \Theta_k\|^2 + \lambda \mathcal{R}(r) \right].
\]

Simplified pseudocode
Please refer to manual for a complete pseudocode.

References
[2] Miguel A. Carreira-Perpiñán, "Model compression as constrained optimization, with application to neural nets. Part II: Problem setup and optimization algorithm and perform following alternating steps:"

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 $R$.