Overview and Future Directions of Deep Learning Model Compression

Edgar Liberis and Nicholas D. Lane

University of Oxford

Deep learning has enabled numerous breakthroughs in many open problems, such as object detection, speech recognition and synthesis, and sensor data processing, due to its ability to accurately model high-level representations of data. Modern neural network architectures can consist of over a hundred layers of non-linear operations, parametrized by over 60 million trainable weights [He et al., 2016], making them difficult to apply when compute or memory resources are limited. To work in such constrained environments, neural networks have to be made more compact while retaining acceptable prediction performance.

Progress towards efficient deep learning has been made from both model design and engineering fronts. In this work, I briefly overview model compression research directions, present recent updates to this field, discuss practical applicability concerns and give future directions.

Current neural network compression and acceleration methods can be approximately divided into the following directions:

- **Cheaper building blocks.** Expensive neural network layers can be replaced with cheaper alternatives, which are easy to integrate with existing architectures but sacrifice predictive performance for execution speed.
- **Knowledge transfer.** Overparametrized networks are typically more amenable to gradient descent-based training than networks with few parameters. The training of such compact networks can be improved by instructing them to mimic large 'teacher' networks.
- **Dynamic routing and conditional execution.** A neural network does not have to perform the same computation for all of its inputs—instead it may choose to route to or skip certain layers based on the perceived difficulty of the input.
- **Weight compression.** Neural networks can be trained and used at reduced floating-point value precision. When taken to extreme, binary weights allow using cheap bit-wise operations instead of multiplication and addition. The optimization target of the neural network can also be used induce sparsity on its parameters, which allows the use of memory-efficient sparse linear algebra primitives.
- **Faster software implementations.** Most of primitive neural network operations can be expressed using matrix multiplication (GEMM), so an implementation that makes best use of processor capabilities and memory bandwidth would drastically improve inference latency. Computational graph operation fusion and intelligent scheduling can also improve memory throughput.
- **Specialized hardware accelerators.** Deep learning workload on mobile devices can be accelerated by introduced special co-processors designed for fast linear algebra.

Many of the above methods introduce a trade-off between the model's predictive performance (accuracy) and computational efficiency — the sweet spot is often determined by the requirements of the downstream application. Many of model compression techniques are conceptually independent, so several can be applied at once to achieve greater savings. However, practitioners should be wary that constraints and optimization goals imposed by some techniques can hinder the effectiveness of others.

References

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.