Summary for AMC: AutoML for Model Compression and Acceleration on Mobile Devices

Siddharth Nayak

1 Introduction

Deep learning has been used in many applications with success. The only hindrance in making them ubiquitous is the requirement of costly and huge hardware components. To make them work on smaller devices like mobiles, the deep networks need to be pruned by not compromising on the accuracy of the network. The paper on 'AMC: AutoML for Model Compression and Acceleration on Mobile Devices' uses a reinforcement learning agent to prune a deep network which beats all the previous methods of manual pruning in terms of speed and accuracy.

2 Model

The RL agent compresses the model in such a way that the accuracy of the deep network remains almost same but the speed improves a lot. The state for the agent is a carefully featured vector as follows:

$$s_t = [t, n, c, h, w, stride, k, \text{FLOPs[t]}, reduced, rest, a_{t-1}]$$

where, t is the layer index, the dimension of the kernel is $n \times c \times k \times k$, the input is $c \times h \times w$. FLOPs[t] is the FLOPs of layer L_t. *Reduced* is the total number of reduced FLOPs in previous layers. *Rest* is the number of remaining FLOPs in the following layers. All these features are scaled within $[0, 1]$ before being passed to the agent. The authors propose a continuous action space $a \in (0, 1]$ which enables more fine-grained and accurate compression. The authors use a Deep Deterministic Policy Gradient(DDPG) for continuous control. The rewards are given according to two different modes:

Resource Constrained Compression: $R_{err} = -Error$
This type of reward offers no incentive for model size reduction. So to achieve the target compression ratio they limit the action space.

Accuracy Guaranteed Compression: $R_{\text{FLOPs}} = -Error.log(\text{FLOPs})$
$$R_{param} = -Error.log(Param)$$
They empirically find that the $Error$ is inversely proportional to $log(FLOPs)$ or $log(Param)$. This reward functio does provide a small incentive to reduce the FLOPs or model size.
The action given by the agent is the sparsity ratio. According to this ratio the channels are pruned in that particular layer.

3 Experiments

The model is evaluated CIFAR-10 and ImageNet. The model performs better than the other handcrafted pruned models and heuristic based methods. The graph of compression ratio vs layer shows us that the pruning ratio does not remain constant across all the layers. They also perform fine-tuning on the model after training which again improves its performance a little bit.

4 Possible Improvements

One of the possible methods that could be tried out is that the agent will try to get block-sparsity in the weights instead of channel wise-pruning. Block-sparsity does improve the speed a lot as shown by Gray et al. in 'GPU Kernels for Block-Sparse Weights'. The reward for the agent can be similar to this paper except for the addition of another term which rewards block sparsity by evaluating the ratio of contiguous zeros in that layer to the total number of elements in that layer. Another thing which could be tried out is that the agent also tries to minimise the power consumption. So the reward can be modelled as:

$$R = \alpha.f(\left|t_{baseline} - t_{model}\right|) + \beta.g(\left|a_{baseline} - a_{model}\right|) + \gamma.h(\left|P_{baseline} - P_{model}\right|) + \delta.F(BSF)$$

Where α, β, γ and δ are hyper parameters.
BSF stands for block sparsity factor.

$$BSF = \frac{\text{number of block sparse elements}}{\text{total number of elements}}$$

f, g, h, F are functions which can be linear or logarithm or quadratic.

5 Conclusion

Overall the paper was quite interesting. The use of an RL agent to prune the weights in a much better way than what humans could achieve was quite impressive. Overall the paper was well written and I could understand the paper well.

6 References

Y.He, J.Lin, Z.Liu, H.Wang, L.Li, S.Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. ECCV-18.
S.Gray, A.Radford, D.Kingma. GPU Kernels for Block-Sparse Weights.