Network Slimming Notes

Kelvin Lee

April 20, 2021

The following notes are taken from *Learning Efficient Convolutional Networks through Network Slimming* by Liu et al.[1].

Contents

1 Introduction **2**

2 Network Slimming **2**

2.1 Approach 2
2.2 Advantages of Channel-level Sparsity 3
2.3 Challenges 3
2.4 Solution: Scaling Factors and Sparsity-induced Penalty 3
2.5 Leveraging the Scaling Factors in BN Layers 4
2.6 Channel Pruning and Fine-tuning 4

1 Introduction

Deployment of CNNs in real world application are constrained by

1. model size,
2. run-time memory,
3. Number of computing operations.

Works that have been proposed to compress large CNNs or directly learn more efficient CNN models for fast inference:

1. low-rank approximation,
2. network quantization,
3. binarization,
4. weight pruning,
5. dynamic inference,
6. etc.

However, these methods only address one or two challenges mentioned above.

Another direction to reduce the resource consumption of large CNNs is to sparsify the network, which can be done on different level of structures.

2 Network Slimming

Network slimming is a simple yet effective network training scheme that addresses all the aforementioned challenges.

2.1 Approach

- Imposes $L1$ regularization on the scaling factors in batch normalization (BN) layers (for simpler implementation without introduction of any change to existing CNN architectures).
- Pushing the values of BN scaling factors towards zero with $L1$ regularization enables to identify insignificant channels (or neurons) since each scaling factor corresponds to a specific convolutional channel (or a neuron in a fully-connected layer).
- This facilitates the channel-level pruning at the followed step.
- Pruning unimportant channels may sometimes temporarily degrade the performance, but this effect can be compensated by the followed fine-tuning of the pruned network.

2.2 Advantages of Channel-level Sparsity

Prior works suggest that sparsity can be realized at different levels, e.g., weight-level, kernel-level, channel-level or layer-level.

Weight-level sparsity gives the highest flexibility and generally leads to higher compression rate, but it usually requires special software or hardware accelerators. Layer-level is less flexible as some whole layers need to be pruned. It is only effective when the network is sufficiently deep (more than 50 layers).

- Channel-level sparsity provides a nice tradeoff between flexibility and ease of implementation.
- Can be applied to any typical CNNs or fully-connected networks (treat each neuron as a channel).
- The resulting network is a "thinned" version of the unpruned network, which can be efficiently inferenced on conventional CNN platforms.

2.3 Challenges

Achieving channel-level sparsity requires pruning all the incoming and outgoing connections associated with a channel.

2.4 Solution: Scaling Factors and Sparsity-induced Penalty

- Introduce a scaling factor γ for each channel, which is multiplied to the output of that channel.
- Then jointly train the network weights and these scaling factors with sparsity regularization imposed.
- Finally prune chanells with smal factors, and fine-tune the pruned network.
- Training objective:

$$L = \sum_{(x,y)} \ell(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma)$$

  - (x, y) denote the training input and target;
  - W are the trainable weights;
  - The first sum-term is the normal training loss of a CNN.
  - $g(\cdot)$ is a sparsity-induced penalty on the scaling factors;
  - λ balances the two terms.

- The scaling factors act as the agents for channel selection. Since they are jointly optimized with the weights, the network automatically identifies insignificant channels, which can be safely removed without greatly affecting the generalization performance.

2.5 Leveraging the Scaling Factors in BN Layers

- Let z_{in}, z_{out} be the input and output of a BN layer and $\mathcal{B}$ be the current mini-batch. BN layer performs the following transformation:

$$\hat{z} = \frac{z_{in} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}; \qquad z_{out} = \gamma \hat{z} + \beta$$

- Conventionally BN layer is inserted after a convolutional layer, with channel-wise scaling/shifting parameter. Thus directly leverage the γ parameters in BN layers as the scaling factors for network slimming.
- It introduces no extra cost.
- If add scaling layers to CNN without BN layer, the scaling factors are meaningless for evaluating the importance of a channel since convolutional and scaling layers are linear transformations.

2.6 Channel Pruning and Fine-tuning

- After training under channel-level sparsity-induced regularization, prune channels with near-zero scaling factors by removing all incoming and outgoing connections and corresponding weights.
- Prune channels with global threshold across all layers (a certain percentile of all the scaling factor values),

References

[1] Zhuang Liu et al. *Learning Efficient Convolutional Networks through Network Slimming*. 2017. arXiv: 1708.06519 [cs.CV].