HFP: Hardware-Aware Filter Pruning for Deep Convolutional Neural Networks Acceleration

Fang Yu, Chuanqi Han, Pengcheng Wang, Ruoran Huang, Xi Huang, and Li Cui*
{yufang,hanchuanqi18b,wangpengcheng18s,huangruoran,huangxi,lcui}@ict.ac.cn.

Institute of Computing Technology
University of Chinese Academy of Sciences

Background

The promising performance of convolutional neural networks (CNNs) is companied by significant computational cost, making them infeasible to be directly deployed on the hardwares with the limited computational resources. Filter pruning is recognized as an effective method to compress and accelerate the CNNs.

However, most of pruning methods cannot prune a network while respecting a actual budget on the target hardware, such as latency, power or energy. As a consequence, these methods can only prune a network while respecting a hardware budget through trial and error, typically by pruning multiple times for a network with various compression hyper-parameters. Hence, these pruning methods are less efficient in practice. In this work, we propose a hardware-aware filter pruning method which can directly control the latency of pruned networks on the hardware platform.

Problem Definition

For classification task, let $P(Y|X;\Theta)$ be the class probability distribution of network output w.r.t. input sample $\{X, Y\}$, where X is the input data, Y is the corresponding label, and Θ is the set of all filters in the network. We use k to denote the index set of selected filters in Θ. Filter pruning is to choose a subset of filters $\theta_k^- \subset \Theta$ and remove corresponding parameters θ_k^- from the network. We note the remaining filters as θ_k^+, thus we have $\theta_k^+ \cup \theta_k^- = \Theta$. To minimize the accuracy drop while meeting the budget of latency on hardware, we need to carefully choose the index set k^* by solving the following constrained optimization problem:

$$k^* = \arg\min_k \mathcal{L}_{CE}(Y, P(Y|X, \theta_k^+))$$
$$\text{s.t.} \quad \text{LAT}(\theta_{k^*}^+) < \text{Bud},$$

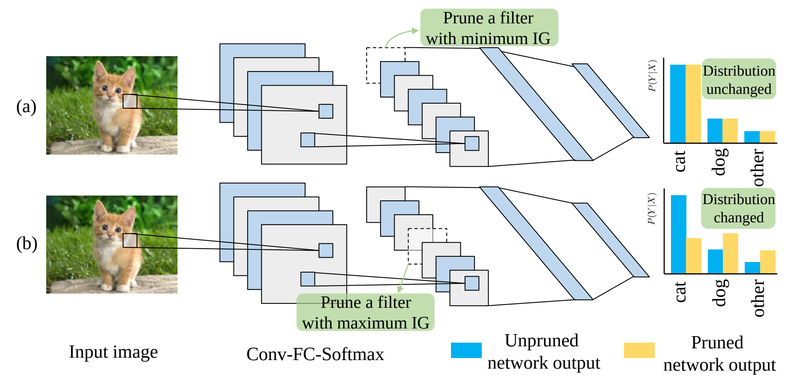
where $\mathcal{L}_{CE}$ is cross-entropy loss, $\text{LAT}(\cdot)$ evaluates the actual latency of pruned network consumed on the hardware, and Bud is the budget about latency. The constrained objective can be replaced by FLOPs in theory, memory or energy consumption on hardware, etc, or a combination of these metrics.

Lookup table

As the number of potential pruned networks in the pruning process is numerous, measuring the latency of each network of intermediate pruning process is extremely time-consuming. Therefore, we employ a lookup table to estimate the latency of pruned network. The lookup table $Lat_i(c_{in}, c_{out})$ provides the latency about filter configuration layer by layer, where c_{in}/c_{out} are the number of input/output channels at the i-th layer. We individually measure the latency of all layers with all configurations of input/output on the hardware of interest, and store them into the lookup table. By simply summing up the latency of each layer, we can efficiently estimate the latency $\text{LAT}(\theta_k^+)$ of a pruned network $\theta_k^+ \subset \Theta$. Formally, the latency of a pruned network can be denoted as:

$$\text{LAT}(\theta_k^+) = \sum_{i=1}^{n} Lat_i(c_{in}, c_{out}).$$

Greedily pruning via IG



Information gain (IG) can quantify the information change of a random variable from a prior state to a new state that takes some condition as given. We use information gain to evaluate the information change of network output distribution after removing filters. As shown in the figure, if pruning a filter with minimum IG, it will not change too much about network output distribution, so this filter can be safely deleted. Formally, IG of filter is described as:

$$\begin{aligned}\text{IG}[P(Y|X,\Theta), \theta_k^-] =& \text{H}[P(Y|X,\Theta)] - \\ & \text{H}[P(Y|X,\Theta)|\theta_k^- = 0] \\ =& \text{H}[P(Y|X,\Theta)] - \\ & \text{H}[P(Y|X,\theta_k^+)]\end{aligned}$$

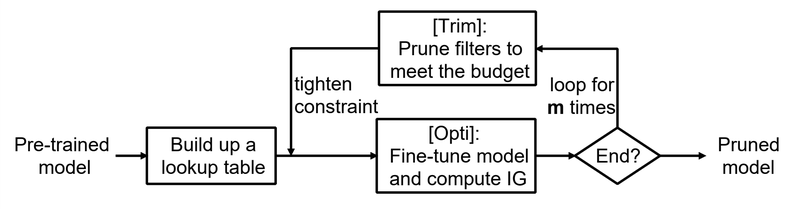
However, finding out the group filters with the minimum IG using above equation is non-trivial as it also requires numerous attempts. We use second-order Taylor series expansion near $\theta_k^- = 0$ to expand the entropy, and get the IG:

$$\text{IG}[p(y|x,\Theta), \theta_k^-] \approx \mathbf{g}^{\text{T}}\theta_k^+ + \frac{1}{2}\theta_k^{+\text{T}}\mathbf{H}\,\theta_k^+,$$

where $g_i = \frac{\partial \text{H}}{\partial \theta_i^+}$ are elements of the gradient $\mathbf{g}$, $H_{i,j} = \frac{\partial^2 \text{H}}{\partial \theta_i^+ \partial \theta_j^+}$ are elements of the Hessian matrix $\mathbf{H}$ and $R_2(\theta_k^- = 0)$ is the second-order remainder term which can be neglected.

Opti-Trim pruning framework

We propose an iterative pruning framework called *Opti-Trim* to solve the resource constrained pruning problem, which consists of *Opti* phase and *Trim* phase. The Opti phase is designed to fine-tune the slashed network and compute the IG of filters for next pruning. During the Opti phase, apart from cross-entropy loss, HFP attaches ℓ_1 group regularization on the filters to fine-tune network weights. The Trim phase is designed to prune filters, achieve the budget on hardware and tighten the resource constraint. This figure illustrates the all process of Opti-Trim pruning framework.



Acknowledgements

The paper is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61672498 and the National Key Research and Development Program of China under Grant No. 2016YFC0302300.

Pseudo Code

Algorithm 1: Algorithm Description of HFP

Input: Pre-trained network: Θ; Desired budget: Bud; Iteration number: m; Training set: $\{X, Y\}$
Output: Pruned network: $\theta_{k^*}^+$

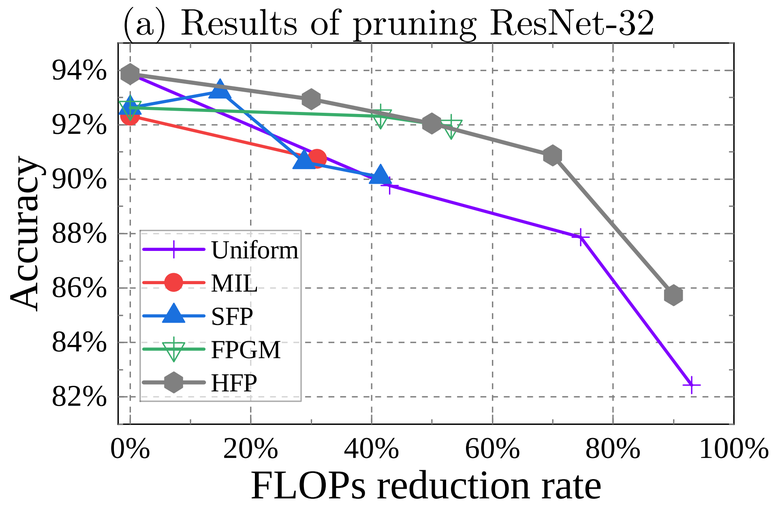
```
/* Initialization                                   */
Build up a lookup table on the target hardware;
Obtain the base latency B;
Obtain Δ = (B − Bud)/m;
/* Opti-Trim pruning framework                      */
for i ∈ [0, m] do
    /* Opti phase                                   */
    foreach {x,y}∈{X,Y} do
        Fine-tune the remaining filters in the network via Eq. (9);
        Calculate the IG of filter via Eq. (6) or Eq. (7);
    end
    /* Trim phase                                   */
    repeat
        Prune a filter with the minimum IG across all layers;
        Obtain the current latency LAT(θ_k^+) of pruned network via Eq. (8);
    until LAT(θ_k^+) < B − i * Δ;
end
```

Experimental Results

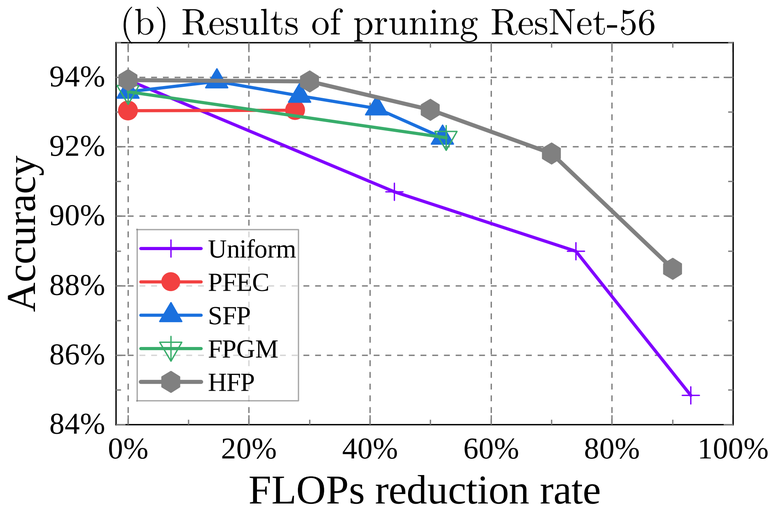
Pruning VGG-16 on Cifar-10

Uniform Baselines			HFP	
Ratio	Accuracy	Latency	Accuracy	Latency
1×	93.73%	1.68ms	-	-
0.75×	92.80%	1.45ms	**93.93%**	1.25ms
0.5×	91.89%	0.78ms	**93.36%**	0.81ms
0.25×	89.06%	0.42ms	**91.04%**	0.45ms

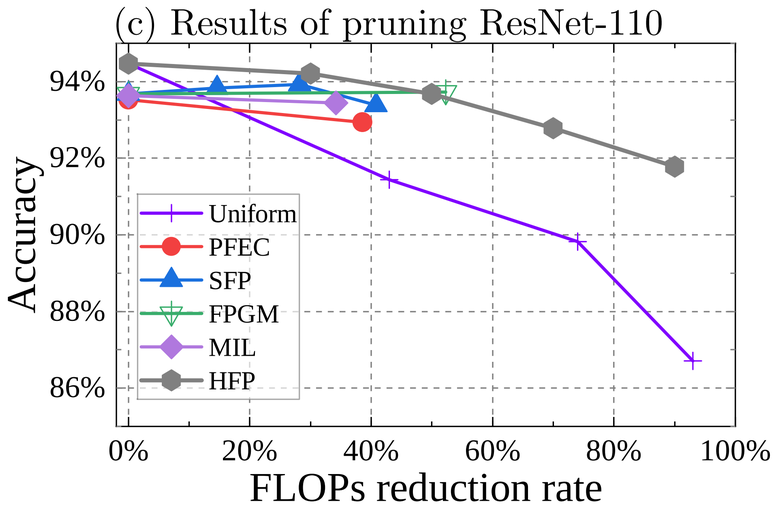
(a) Results of pruning ResNet-32



(b) Results of pruning ResNet-56



(c) Results of pruning ResNet-110



Compared with state-of-the-art methods, the proposed HFP outperforms these methods on pruning ResNet series networks.