Pruning Convolutional Filters with First Order Taylor Series Ranking

Mengqing Wang[1]

Research school of Computer Science, The Australian National University
u6416782@anu.edu.au

Abstract. In this paper a ranking method is proposed to prune inactive or insignificant filters that have little contribution to the final prediction of a convolutional neural network (CNN). The contribution of a convolutional filter is approximated by the sum of products over the weights and graidents, which mathematically is equivalent to the first order Taylor expansion of loss w.r.t. the filter's output feature map. An iterative pruning algorithm is derived from this ranking algorithm: at each stage a filter with the lowest rank is pruned and the network is fine-tuned to recover from this damage (i.e. loss of functionality). The algorithm is tested on a CNN trained for optical digits recognition and yields significantly better performance than two baseline methods (i.e. rank by weights and rank by gradients) when same number of filters are pruned.

Keywords: pruning · rank · Taylor series · network size reduction · CNN

1 Introduction

Pruning is the process of removing/merging undesired operations carried out during the inference of neural networks. The obvious merits of pruning include improved efficiency, reduced model size, and increased generality [1]. The necessity of pruning arises from the fact that there does no exist a common prior for hyper-parameters and ofttimes an over-sized model is trained to obtain an expected level of accuracy; pruning is performed in this case as a dynamic adjustment of hyper-parameters (e.g. number of layers/neurons/connections). However, pruning algorithms are not only excersised to refine network structures - another prominent setting is transfer learning, where they seem to excel at purging pre-trained knowledge that is no longer relevant to the new task.

A standard purning algorithm generally involves a ranking process where parameters (or groups of parameters) recieve a score based on, e.g., significance, activeness, badness or redundancy. The ranking criteria are essential for pruning as they basically dictate which parameters are the least desired ones and should be subsequently removed.

Convolutional neural network (CNN) make heavy reuse of its parameters (filters) to achieve shift invariance. This poses a certain difficulty for pruning as the entire output feature map needs to be tracked for ranking the importance

of a single weight parameter, leading to a high demand for memory [4]. In this paper, a memory-efficient alternative to [4] is considerd: instead of keeping the feature maps, the weights and their gradients are used for ranking filters, which requires far less memory consumption (since size of filter is trivial compared to size of feature map). It can be further proved that the element-wise product of weights and gradients is able to gather exactly the first-degree Taylor polynomial of corresponding feature map w.r.t loss through back-propagation.

Once ranking is introduced, we shall simplify the pruning algorithm to iteratively removing a *convolutional filter* in a CNN with the lowest rank, which would work orthogonally to parameter-reduction technologies from other categories (e.g. global average pooling). Unlike the second order derivatives ranking methods introduced in [1, 2], only the gradient is needed for ranking filters. As such, Hessian matrix is not required and computational cost is greatly reduced to a margin of [1, 2]. The algorithm is tested on a toy example CNN and is shown to outperform two baseline ranking methods by a large margin.

2 Methodology

In this section details on the ranking method are discussed. To facilitate a thorough understanding of proposed algorithm, we shall proceed to examine two intuitive baseline ranking methods: (1) rank by weights and (2) rank by gradients.

But let us first formalise the problem. Borrowing from [1, 2], we shall use the term *optimal damage* to refer to an optimal pruning step, which is mathematically defined as the removal of a subset of parameters that would lead to the least impact the to loss of prediction. Generally, this can be formulated as a combinatorial optimization process:

$$\arg_{\beta} \min R(\beta) = E\big(f(X; \theta - \beta), Y\big) \quad \text{s.t. } P(\beta). \tag{1}$$

Here E denotes the loss or error function and $R(\beta)$ is the rank of pruned parameter set β. $f(X; \theta - \beta)$ denotes the prediction of neural network with pruned parameter set $\theta - \beta$ on data X (where θ is the parameters before pruning and $\beta \subset \theta$ is the pruned parametres) and Y is the expected output (labels). We put a predicate $P()$ to constraint β, so that β must be a non-trivial subset of θ (a common practise is let $P(\beta)$ be true iff. $|\beta| \geq k$ to guarantee at least k parameters are removed.

Ideally the combinatorial problem can be solved using what is called an 'oracle ranking' - a generally NP-hard process by brute-force. In this paper, we let $P(\beta)$ true true when β is the set of parameters inside a convolutional kernel. In natural language, we want to find the filter of a CNN, such that removing it will lead to the optimal damage, i.e. a least increase in loss function.

2.1 Baseline ranking methods

Ranking with weights is a straightforward approximation of optimal damage. The intuition is that empirically, filters with greater absollute weights usually

have more significance than filters with close-to-zero weights, and removing them will likely cause a greater increase in loss. Following this observation, a ranking by weights can be calculated using the norm:

$$R(\beta) = \|\beta\| \tag{2}$$

Ranking with gradients assumes that filters recieve approximately same degree of training, as such less important filters becomes easily saturated after a few backward passes thus have smaller gradients. Therefore ranking can be calculated using norm of gradients:

$$R(\beta) = \|\frac{\partial \text{loss}}{\partial \beta}\| \tag{3}$$

2.2 Ranking using first order Taylor expansian

We first give the formulation for ranking. Denote weights of a filter by β (here only filtering weights are considered but not the bias, we shall see the reason later), its rank is calculated as:

$$R(\beta) = |\sum \beta \cdot \frac{\partial \text{loss}}{\partial \beta}| \tag{4}$$

Intuitively, the rank combines both baseline methods for ranking by using the element-wise product of weights and gradients on weights. This means that both weights and gradients need to have a large absolute value for the filter to recieve a high rank. To quantify the difference in loss after removing a filter with weights β, let us define the loss function w.r.t the convolution output of pruned filter, by which we denote h_β: by pruning the filter, h_β is effectively zeroed-out, thus the loss before and after pruning can be written as a function w.r.t h_β. We denote the loss before pruning by $E(h_\beta)$, the loss after pruning becomes $E(0)$. Hence the change in loss is:

$$|E(h_\beta) - E(0)| = |\sum_{n=0}^{\infty} \frac{f^{(n)}(h_\beta)}{n!}(-h_\beta)^n| \tag{5}$$

$$\approx |\sum \nabla E \cdot (-h_\beta) + h_\beta^T H(h_\beta) h_\beta| \tag{6}$$

$$\approx |\sum \nabla E \cdot (h_\beta)| \tag{7}$$

Here we use the Taylor expansion on the righthand side of (5) for approximation and slightly abuse the symbols (since β is a vector and thus $E(\cdot)$ a multivariable function). In the following lines the approximation using second order and first order Taylor series is given. Equation (7) can be rewritten using the backward propagation chain rule:

$$|\sum \nabla E \cdot h_\beta| = |\sum_{h_\beta} \frac{\partial \text{loss}}{\partial h_\beta} \cdot h_\beta| = |\sum_{h_\beta} \sum_{\beta} \frac{\partial \text{loss}}{\partial \beta} \cdot (\frac{\partial \beta}{\partial h_\beta} \cdot \frac{h_\beta}{\beta}) \cdot \beta| = |\sum_{\beta} \beta \cdot \frac{\partial \text{loss}}{\partial \beta}| \tag{8}$$

It is therefore proved that the ranking method of (4) is exactly the first degree Taylor term of loss w.r.t the filter output, and approximates the actual damage (change in loss). After each pruning, a fine-tuning process follows to allow the network to recover from damage. The complete algorithm can be written in pseudocode:

Algorithm 1 Pruning by Taylor ranking

```
procedure PRUNE(model, k, data)
    while i=1:k do
        weights = model.weights
        grads = model(data).loss.gradient
        ranks = abs(multiply(weights, grads))
        filter = filter with the smallest rank
        model.remove(filter)
        model.train(data)
    end while
end procedure
```

3 Experiments

The pruning algorithm is tested on a pretrained CNN for handwritten digits recognition task. For comparison three different ranking methods are included: rank by weights, by gradients and the proposed method. 15 out of 30 filters are iteratively pruned, at each stage the lowest ranking filter is selected and removed, and the network is fine-tuned for 10 epochs on the entire dataset.

3.1 Dataset

The MNIST [3] dataset is selected for this task. The dataset consists of 70k optical handwritten digits of size 28-by-28. Each image contains an aligned optical hand-written digit from 0 to 9 with known label. 60k images are selected as training set and 10k images for the test set with no overlapping image.

3.2 Model and training routine

A CNN with two convolutional layers and two dense layers is trained for this task. The first and second convolutional layers have 10 and 20 filters respectively and the kernel size is set to 5-by-5. After the second convolutional layer, the feature maps are flatten to a vector of 320 dementions and passed to dense layers (50 and 10 neurons respectively). The model is trained using a negative log likelihood loss and a stochastic gradient descend optimizer at momentum of 0.5 for 40 epochs until the testing loss converges. After removing a filter, the fine-tuning takes an additional 10 epochs per removal.

3.3 Pruning

Three methods are compared using the same pruning process described before. If a filter in first convolutional layer is pruned, a feature map in second layer is removed along with the filter and its bias; if a filter in second layer is pruned, all connections to its output feature map in the next dense layer also need to be subsequently removed. To facilitate inter-layer filter ranking, the filters are normalized by the L2 norm of ranks of all filters in the same layer.

The training loss and accuracy w.r.t number of prunings are given in Figure 1 and Figure 2. It is shown that the proposed method maintains the highest prediction accuracy when same number of filters are pruned. The proposed method is able to prune 11 filters before accuracy drops below 0.95, by contrast two baseline methods can only prune half of that number. After the 12-th pruning, the accuracy drops significantly, at which point the pruning should stop to prevent damage caused by removing filters of high importance.

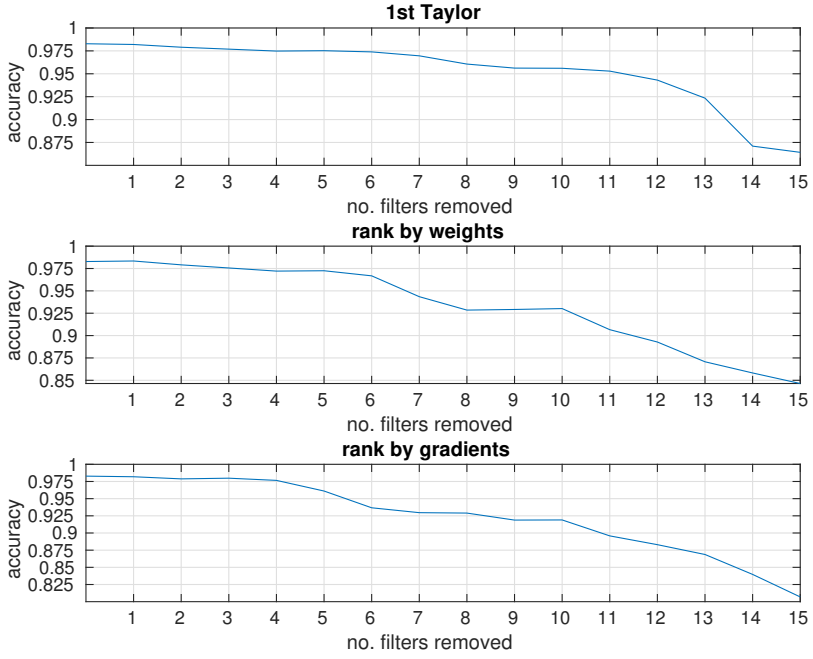

Fig. 1. Accuracy versus number of prunings using three ranking methods.

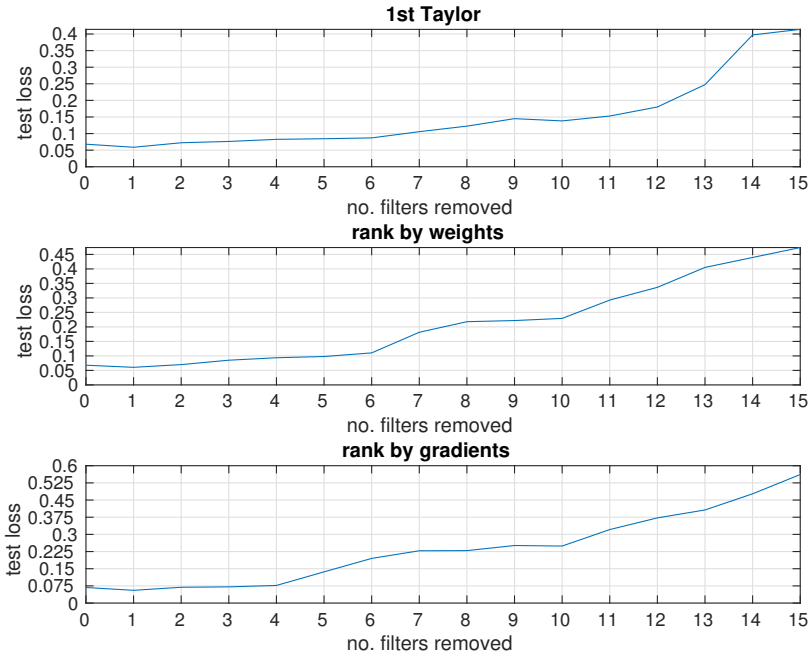

Fig. 2. Test loss versus number of prunings using three ranking methods.

4 Conclusion

In this paper we study the performance of a filter ranking and pruning algorithm on a CNN with digit recognition as applicational background. The algorithm iteratively removes a single filter with the lowest rank, with the rank appromating the change in loss caused by zeroing-out the output feature map of a filter. After each removal, the entire neural network is fine-tuned to recover from the damage caused by pruning. This algorithm can significantly reduce the number of filters when maintaining a satisfactory level of prediction accuracy. In our experiments, we found that more than 30% of filters can be removed before the accuracy drops below 95%, and the ranking consitently outperforms two baseline methods.

References

1. LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems* (pp. 598-605).
2. Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems* (pp. 164-171).

3. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
4. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv*:1611.06440.