TOWARDS THINNER CONVOLUTIONAL NEURAL NETWORKS THROUGH GRADUALLY GLOBAL PRUNING

Zhengtao Wang, Ce Zhu, Zhiqiang Xia, Qi Guo, Yipeng Liu
School of Electronic Engineering / Center for Robotics
University of Electronic Science and Technology of China (UESTC), Chengdu, China

ABSTRACT

Convolutional neural networks (CNNs) are always trapped by their huge amount of parameters when faced with resource-limited devices. To handle this problem, we propose a pruning scheme for neuron level pruning, in which the redundant neurons were selected globally in the network. Our scheme could automatically find a thinner network structure with a given performance.

INTRODUCTION

- Three types of current pruning methods
  - **Approximation**: Weight matrices and tensors in deep model could be approximated using tensor decomposition techniques.
  - **Quantization**: By searching or constructing a finite set for candidate parameters, one could map parameters from real number to several candidates
  - **Pruning**: Reduce redundant connections, neurons or entire layers of the model.
- Pruning methods in different granularities:
  - Layer-level: Shallower networks
  - Neuron-level: Thinner networks
  - Connection-level: Sparser networks
- Main problems
  - How to evaluate the importance of a neuron
  - How to conduct the pruning process
- We refer neuron as a node in fully-connected networks or a filter in convolutional networks.

METHOD

- Redundant neurons selection(neuron importance evaluation)
  - $l = layer\ index, i = neuron\ index, N = \#neurons\ in\ a\ layer$
  - Mean of activations:

$$S_{\bar{R}}(l,i) = \sum_{j=1}^{N} R_{ij}^{l}$$

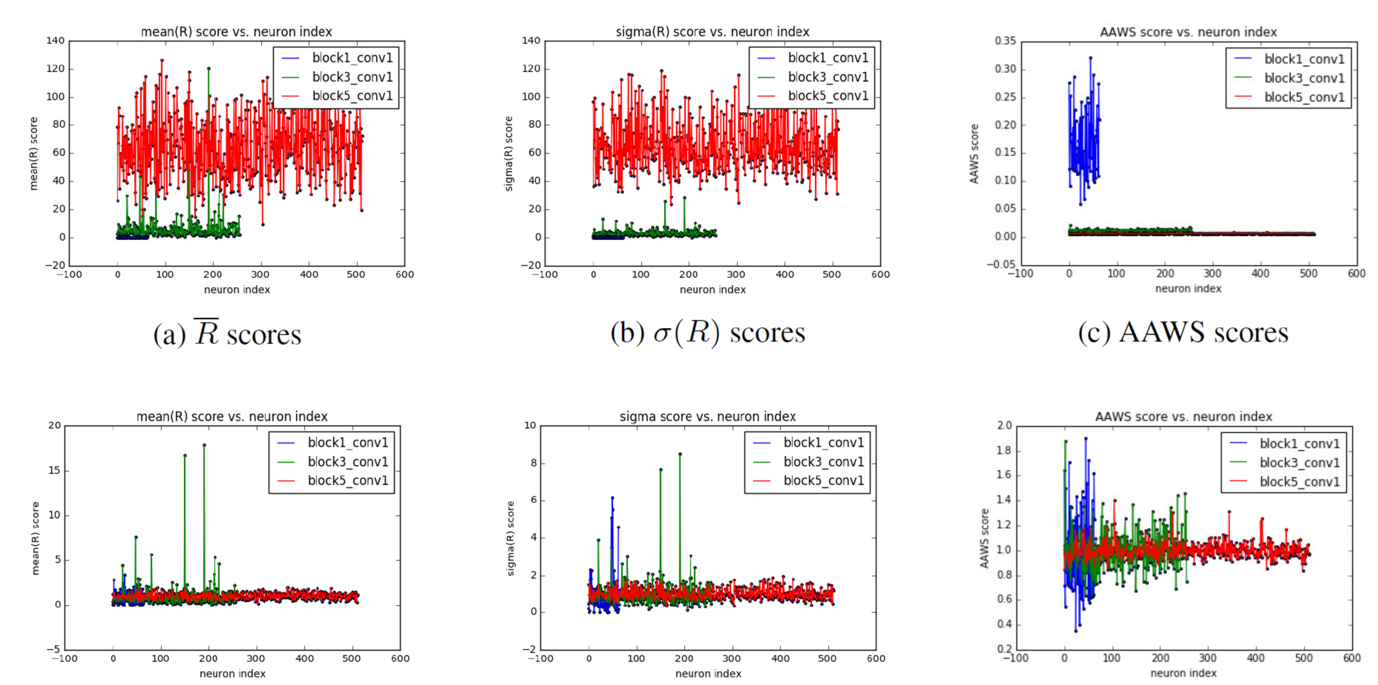
  - Standard derivation of activations:

$$S_{\sigma}(l,i) = \sqrt{\frac{\sum_{j=1}^{N}(R_{ij}^{l}-\bar{R})}{N}}$$

  - Average Absolute Weights Sum

$$S_{AAWS}(l,i) = \frac{1}{n_c \times n_m \times n_n} \sum_C \sum_M \sum_N |W_i^l|$$

  - $n_c \times n_m \times n_n$ is the number of elements in a filter, W_i^l is a parameter in the filter. We omit indexes for simplicity.
- The imbalance of scores in different layers



(a) $\bar{R}$ scores (b) $\sigma(R)$ scores (c) AAWS scores

(d) $\bar{R}$ scores after modification (e) $\sigma(R)$ scores after modification (f) AAWS scores after modification

Fig. 1: The score distribution of different metrics

- Adjust scores in different layers for feasible global pruning.

$$S_{modified}(l,i) = \frac{S(l,i)}{\frac{1}{N_l}\sum_{j=0}^{N_l} S(l,j)}$$

FRAMEWORK

Algorithm 1 Gradually global pruning scheme.

Input: A trained Model: M
Given performance target: P_t
Contribution score evaluator: $E(\cdot)$
Pruning ratio generator: r
Training set: X
Validation set: V
Output: A thinner model: M
1: Compute the performance P_m of M using V
2: **while** $P_m \geq P_t$ **do**
3: Compute the contribution scores of all neurons in M with evaluator $E(\cdot)$
4: Sort the scores
5: Select $N \times r$ neurons to be prune, where N is the number of neurons in current model
6: Drop the selected neurons in the network, get M_{drop}, update M by M_{drop}
7: Fine-tune M with training set X
8: Update P_m by the performance of M over V
9: **end while**
10: **return** M

EXPERIMENT RESULTS

- Pruning VGG-like network for CIFAR-10 classification(Prop. for "proportional pruning neurons in each layer, not proposed.")

name	org.	$\bar{R}$	$\sigma(R)$	AAWS	Prop.
conv1_1	64	35(54.7%)	3(4.7%)	33(51.6%)	45(70.31%)
conv1_2	64	52(81.3%)	14(21.9%)	34(53.1%)	45(70.31%)
conv2_1	128	85(66.4%)	70(54.9%)	83(64.8%)	89(69.5%)
conv2_2	128	72(56.3%)	70(54.9%)	128(100.0%)	89(69.5%)
conv3_1	256	93(36.3%)	168(65.6%)	254(99.2%)	179(69.9%)
conv3_2	256	173(67.6%)	194(75.8%)	256(100.0%)	179(69.9%)
conv3_3	256	169(66.0%)	218(85.2%)	256(100.0%)	179(69.9%)
conv4_1	512	257(50.2%)	314(61.3%)	486(94.9%)	357(69.7%)
conv4_2	512	405(79.1%)	395(77.1%)	500(97.7%)	357(69.7%)
conv4_3	512	490(95.7%)	382(74.6%)	448(87.5%)	357(69.7%)
conv5_1	512	468(91.4%)	452(88.3%)	321(62.7%)	357(69.7%)
conv5_2	512	436(85.2%)	434(84.8%)	276(53.9%)	357(69.7%)
conv5_3	512	398(77.7%)	397(77.5%)	229(44.7%)	357(69.7%)
fc1	512	177(34.6%)	199(38.9%)	6(1.2%)	357(69.7%)
total	4736	3310(69.9%)	3310(69.9%)	3310(69.9%)	3304(69.8%)
acc.	87.32%	84.35%	81.88%	86.89%	86.54%

- Pruning VGG network for Kaggle cat/dog classification(transfer learning)

name	org.	neurons	name	org.	neurons
conv1_1	64	28(43.8%)	conv4_3	512	512(100.0%)
conv1_2	64	28(43.8%)	conv5_1	512	512(100.0%)
conv2_1	128	59(46.1%)	conv5_1	512	512(100.0%)
conv2_2	128	74(57.8%)	conv5_2	512	512(100.0%)
conv3_1	256	169(66.1%)	conv5_3	512	506(98.8%)
conv3_2	256	192(75.0%)	fc1	4096	4096(100.0%)
conv3_3	256	216(84.4%)	fc2	4096	392(9.6%)
conv4_1	512	495(96.7%)	total	12416	8302(66.9%)
conv4_2	512	511(99.8%)	acc.	98.24%	97.22%