Learning the Number of Neurons in Deep Networks

Jose M. Alvarez[1] Mathieu Salzmanno[2]

[1]Data61 @ CSIRO,Canberra, ACT 2601, Australia

[2]CVLab, EPFL,CH-1015 Lausanne, Switzerland

NIPS,2016
Presenter: Arshdeep Sekhon

Motivation

1. Designing a deep NN architecture
2. Configure:
   - number of layers
   - number of units
3. mostly hand-designed
4. have redundant parameters

Automatic Model Selection: Constructive Approaches

1. Incrementally add layers/parameters
2. But Shallow networks are less expressive
3. bad initialization when incrementally adding layers

Automatic Model Selection: Destructive Approaches

1. Very Deep networks more expressive
2. Start from very deep networks, eliminate redundant parameters
3. check influence of every parameter
4. for example, check network Hessian wrt every parameter in an over complete network
5. not scalable to large networks

Automatic Model Selection

1. Automatically get number of neurons for each layer
2. cancel effects of individual neurons
3. jointly as we learn
4. no preprocessing

Model Selection

1. Start with an overcomplete network
2. Find neurons for each layer
3. A general deep network:
   L layers in network architecture
   N_l neurons in each layer
4. weights $\Theta = [\theta_l, b_l]$ for layer l $\theta_l = [\theta_l^n]$ $1 \leq l \leq L$ *and* $1 \leq n \leq N_l$
5. The optimization problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{1}$$

Model Selection: Regularizer

1. The optimization problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{2}$$

2. Goal: Cancel entire neurons

Model Selection: Regularizer

1. The optimization problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{2}$$

2. Goal: Cancel entire neurons
3. traditional regularizers: ℓ_1 or ℓ_2
4. cannot cancel entire neurons because they control weights individually.

Model Selection: Regularizer

1. The optimization problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{2}$$

2. Goal: Cancel entire neurons
3. traditional regularizers: ℓ_1 or ℓ_2
4. cannot cancel entire neurons because they control weights individually.
5. Neurons are *groups* of parameters
6. weights $\Theta = [\theta_l, b_l]$ for layer l $\theta_l = [\theta_l^n]$ $1 \leq l \leq L$ *and* $1 \leq n \leq N_l$

Model Selection: Regularizer

1. The optimization problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{2}$$

2. Goal: Cancel entire neurons
3. traditional regularizers: ℓ_1 or ℓ_2
4. cannot cancel entire neurons because they control weights individually.
5. Neurons are *groups* of parameters
6. weights $\Theta = [\theta_l, b_l]$ for layer l $\theta_l = [\theta_l^n]$ $1 \leq l \leq L$ *and* $1 \leq n \leq N_l$
7. Use new regularizer: *group sparsity*

Model Selection: Group Sparsity

1. Parameters associated with a neuron are grouped together
2. Penalty on groups of weights instead of individual weights
3. parameters of each neuron in layer l are grouped in a vector of size P_l
4. New regularizer:

$$r(\Theta) = \sum_{l=1}^{L} \lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_n^l||_2 \tag{3}$$

5. θ_l^n are the parameters for neuron n in layer l
6. ℓ_2 norm followed by ℓ_1 norm
7. λ_l sets the influence of the penalty.

Model Selection: Group Sparsity

1. But does not lead to sparsity within a group

$$r(\Theta) = \sum_{l=1}^{L} (1-\alpha)\lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_n^l||_2 + \alpha\lambda_\ell ||\theta_\ell||_1 \qquad (4)$$

2. more general penalty that leads to sparsity both at and within group level.

Training: Proximal Gradient Descent

$$\textit{minimize } f(x) = g(x) + h(x) \tag{5}$$

proximal gradient algorithm:

$$x^{k+1} = \mathbf{prox}_{t_k h}\left(x^{k-1} - t_k(\nabla g(x^{k-1}))\right) \tag{6}$$

1. proximal operator:

$$\mathbf{prox}_h(x) = \arg\min_u h(u) + \frac{1}{2}||x - u||_2^2 \tag{7}$$

2.

$$x^{k+1} = \arg\min_u \left(h(u) + \frac{1}{2t}||u - x^{k-1} + t_k(\nabla g(x^{k-1}))||_2^2\right) \tag{8}$$

Training: Proximal Gradient Descent

1. The objective:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + r(\Theta) \tag{9}$$

2.

$$r(\Theta) = \sum_{l=1}^{L} (1-\alpha)\lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_n^l||_2 + \alpha\lambda_\ell ||\theta_\ell||_1 \tag{10}$$

3. loss function is $g(x)$ and regularizer $h(x)$ in proximal gradient algorithm

Training: Proximal Gradient Descent

1. Update:Take gradient of loss and apply proximal operator of the regularizer
2. $$\tilde{\theta}_l^n = \arg\min_{\tilde{\theta}_l^n} \frac{1}{2t} ||\tilde{\theta}_l^n - \hat{\theta}_l^n||_2^2 + r(\Theta) \tag{11}$$
   where $\hat{\theta}_l^n$ is update by gradient of loss function
3. This has a closed form solution:

$$\tilde{\theta}_l^n = \left(1 - \frac{t(1-\alpha)\lambda_l\sqrt{P_l}}{||S(\hat{\theta}_l^n, t\alpha\lambda_l)||_2)}\right)_+ S(\hat{\theta}_l^n, t\alpha\lambda_l)$$

$$(S(\mathbf{z}, \tau))_j = sign(z_j)(|\mathbf{z}_j| - \tau)_+$$

Experiments and Model Architectures

1. Dataset: ImageNet , Places2-401
2. Models:
   1. VGG-B Net:10 convolutional layers followed by three fully-connected layers
   2. DecomposeMe$_8$ (Dec$_8$): 16 Conv layers with 1D kernels

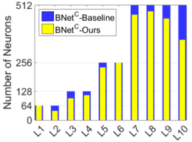
Experiments and Model Architectures

Table 1: Top-1 accuracy results for several state-of-the art architectures and our method on ImageNet.

Model	Top-1 acc. (%)
BNet	62.5
BNet^C	61.1
ResNet50[a] [He et al., 2015]	67.3
Dec$_8$	64.8
Dec$_8$-640	66.9
Dec$_8$-768	68.1

Model	Top-1 acc. (%)
Ours-Bnet$^C_{GS}$	62.7
Ours-Dec$_{8-GS}$	64.8
Ours-Dec$_8$-640$_{SGL}$	67.5
Ours-Dec$_8$-640$_{GS}$	68.6
Ours-Dec$_8$-768$_{GS}$	68.0

[a] Trained over 55 epochs using a batch size of 128 on two TitanX with code publicly available.



BNet^C on ImageNet (in %)

	GS
neurons	12.70
group param	13.59
total param	13.59
total induced	27.38
accuracy gap	1.6

Figure 1: Parameter reduction on ImageNet using BNet^C. (Left) Comparison of the number of neurons per layer of the original network with that obtained using our approach. (Right) Percentage of zeroed-out neurons and parameters, and accuracy gap between our network and the original one. Note that we outperform the original network while requiring much fewer parameters.