Less is More: Towards Compact CNNs

Hao Zhou[1], Jose M. Alvarez[2] and Fatih Porikli[2,3]
[1] University of Maryland, College Park, USA
[2] Data61/CSIRO, Canberra, Australia
[3] Australian National University, Canberra, Australia

Introduction

CNNs contain huge number of parameters, which leads to large memory footprint:
(1) Fewer test samples at once.
(2) Not suitable for Mobile devices.

Previous work:

1. Network distillation.
2. Memory efficient structures.
3. Parameter pruning.

We remove the number of neurons during training using sparse constraints. **removing neurons has advantages in**:

1. Do not rely on sparse data structure.
2. Also apply to Fourier domain.
3. Dimension reduction.

Contributions

1. Reducing number of neurons of CNNs **during training**.
2. Analyzing the importance of ReLU for sparse constraints.
3. Reducing significant amount of parameters for four well-known CNNs.
4. Easy to implement.

Formulation

Objective function:

$$\min_{\hat{\mathbf{W}}} \psi(\hat{\mathbf{W}}) + g(\hat{\mathbf{W}}). \quad (1)$$

$\hat{W}$: the parameter of the CNN.
$g(\hat{\mathbf{W}})$: the sparse constraints.
$\psi(\hat{\mathbf{W}})$: the objective function of training a CNN.

Normal Backprop is difficult as:
(1) gradient of $g(\hat{\mathbf{W}})$ is difficulty to compute.
(2) $g(\hat{\mathbf{W}})$ is non differentiable at sparse point.

Forward-backward splitting:

Algorithm 1 Forward-backward splitting
1: **while** Not reaching maximum number of iterations **do**
2: One step back-propagation for $\psi(\hat{\mathbf{W}})$ to get $\hat{\mathbf{W}}^{k*}$
3: $\hat{\mathbf{W}}^{k+1} = \arg\min_{\hat{\mathbf{W}}} g(\hat{\mathbf{W}}) + \frac{1}{2\tau^k}||\hat{\mathbf{W}} - \hat{\mathbf{W}}^{k*}||^2$
4: **end while**

one step = one epoch.

References

[1] S. Srinivas, R.V. Babu. Data-free Parameter Pruning for Deep Neural Networks In *BMVC '15*
[2] J. Liu, P. Musialski, P. Wonka and J. Ye Tensor Completion for Estimating Missing Values in Visual Data. PAMI '13

Sparse Constraints

Tensor Low Rank [2]:

$$g(\hat{\mathbf{W}}) = \lambda \sum_{(j,l)\in\Omega} \frac{1}{n} \sum_{i=1}^{n} ||\hat{\mathbf{w}}_{lj(i)}||_{tr}. \quad (3)$$

Group Sparsity:

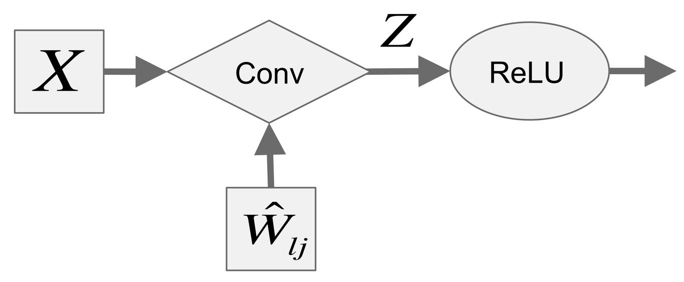
$$g(\hat{\mathbf{W}}) = \lambda \sum_{(j,l)\in\Omega} ||\hat{\mathbf{w}}_{lj}||, \quad (4)$$

Importance of ReLU

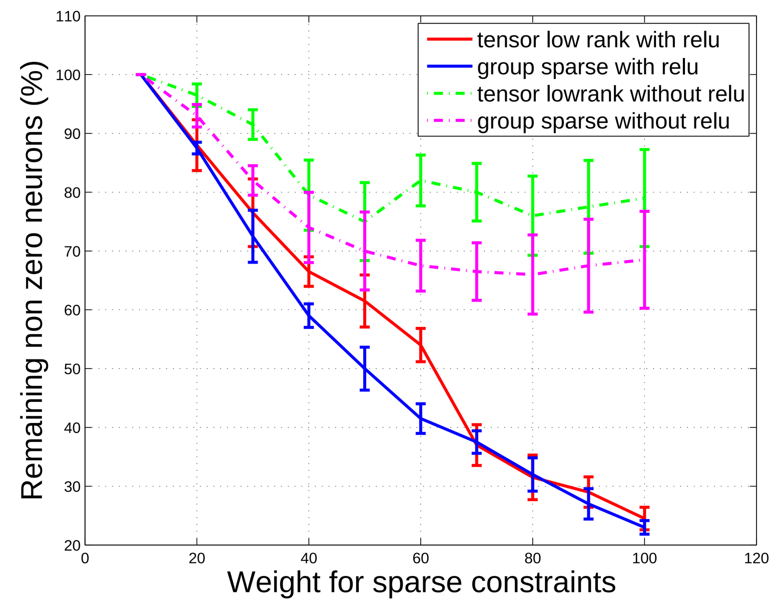
Considering ReLU as:

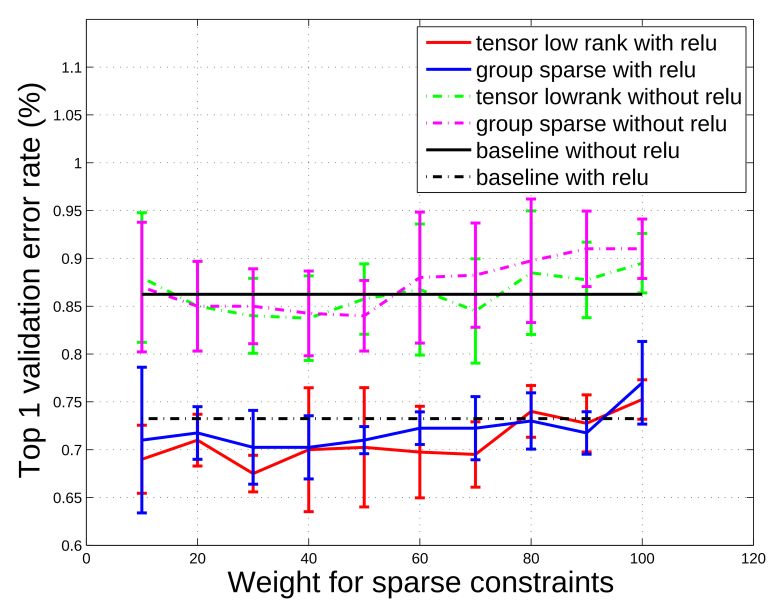
$$ReLU(x) = \begin{cases} x & \text{if } x > \epsilon \\ 0 & \text{if } x \le \epsilon. \end{cases} \quad (2)$$

then for a particular neuron $\hat{W}_{lj}$, **0 is its local minimum** if all other neurons are fixed.



Experiments about ReLU





Left: Percentage of nonzero neurons of conv2 for LeNet with and without ReLU layer.
Right: the corresponding top 1 validation error on MNIST.

Experiments

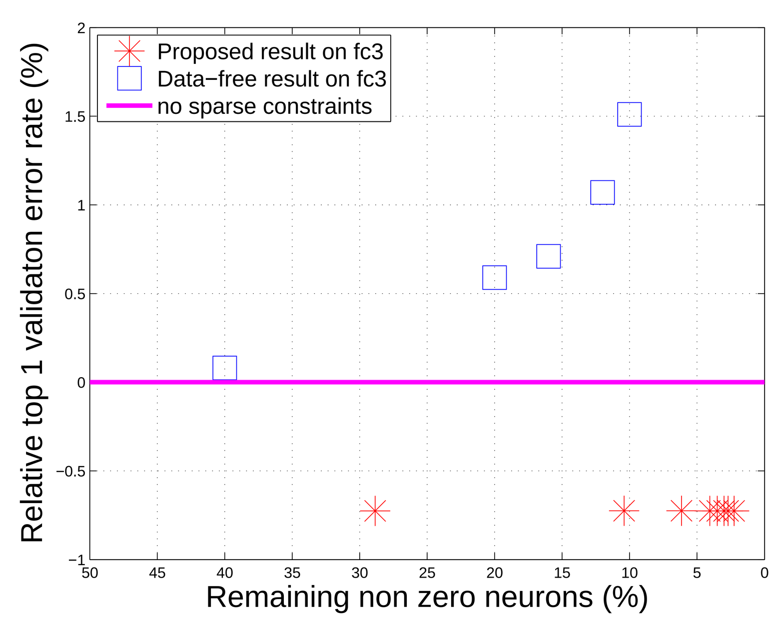
	τ		Neurons pruned(%)		Top-1 error (%)		parameter	memory
	conv2	fc3	conv2	fc3	absolute	relative	reduction (%)	reduced (MB)
LeNet	60	100	45.5	97.75	0.73	0.00	95.35	1.57
	80	100	56.5	97.75	0.77	0.04	96.31	1.58
	100	100	63.0	97.75	0.76	0.03	**96.79**	**1.59**
	τ		Neurons pruned (%)		Top 1 error (%)		parameters	memory
	conv3	fc4	conv3	fc4	absolute	relative	reduction(%)	reduced (KB)
cifar10-quick	220	280	31.25	70.31	22.21	-0.12	47.17	268.24
	240	280	46.88	71.86	22.73	0.4	**55.15**	**313.62**
	280	280	54.69	70.31	23.78	1.45	58.56	333.01
	τ		Neurons pruned (%)		Top 1 error (%)		parameters	memory
	fc6	fc7	fc6	fc7	absolute	relative	reduction (%)	reduced(MB)
AlexNet	40	35	48.46	56.49	44.58	-0.98	55.15	128.26
	45	30	77.05	60.21	46.14	0.57	**76.76**	**178.52**
	45	35	73.39	65.80	45.88	0.31	74.88	174.14

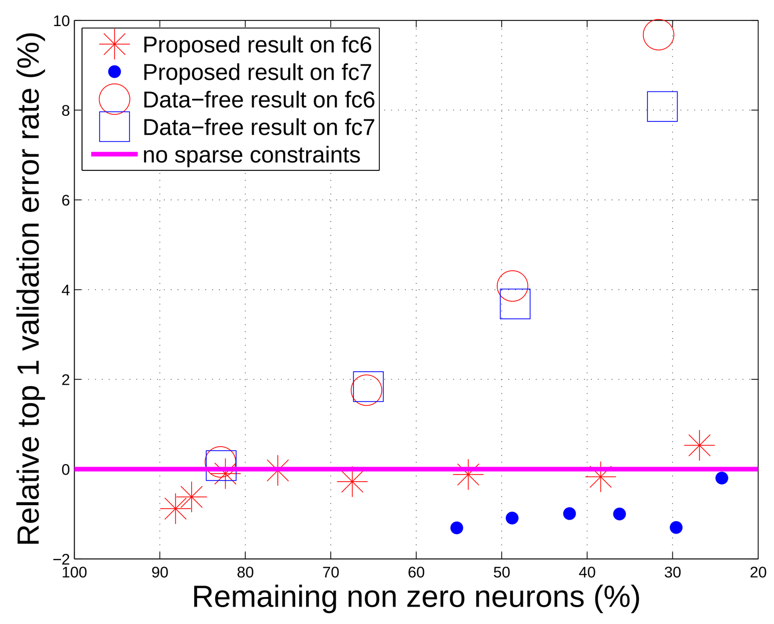
Results of LeNet on MNIST, cifar-10 quick on cifar-10 and AlexNet on ImageNet. Sparse constraints are added to two layers.

layer	τ	compression %		memory	top 1 error (%)	
		neurons	parameters	reduced (MB)	absolute	relative
fc1	5	39.04	35.08	178.02	38.30	0.80
fc1	10	49.27	44.28	224.67	38.54	1.04
fc1	20	76.21	61.30	311.06	39.26	1.76

Results of VGG-13 on ImageNet. Sparse constraints are added to one layer.

Compare With [1]





Left: compare with [1] on LeNet.
Right: compare with [1] on AlexNet.
[1] recursively combines similar neurons of a trained network.