Learning Structured Sparsity in Deep Neural Networks

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li
University of Pittsburgh
{wew57, chw127, yaw46, yic52, hal66}@pitt.edu
Acknowledgement: Sheng Li and Jongsoo Park, Intel Parallel Computing Lab

Evolutionary Intelligence Lab

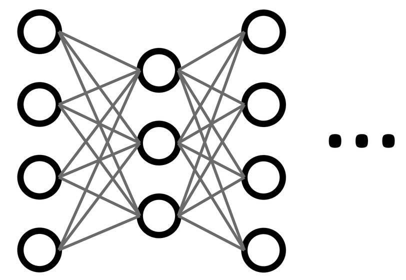
Code in GitHub

About Me

Introduction

Goal

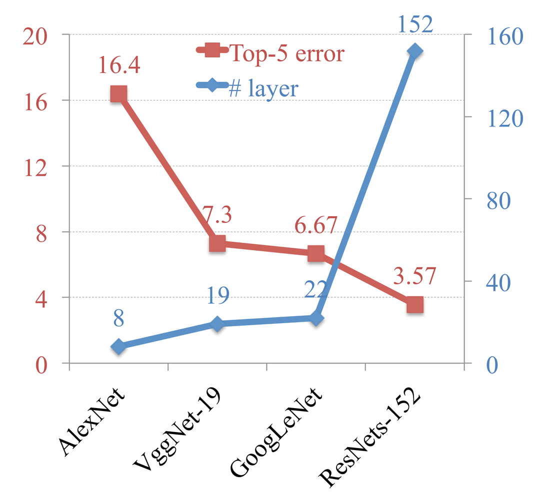
- ✓ Speedup the testing of DNNs deployed in resource-constrained systems, *e.g.*, mobile devices, embedded systems, *etc*.
- ✓ Focus on convolutional layers in deep neural networks.



Deep Neural Networks (DNNs)

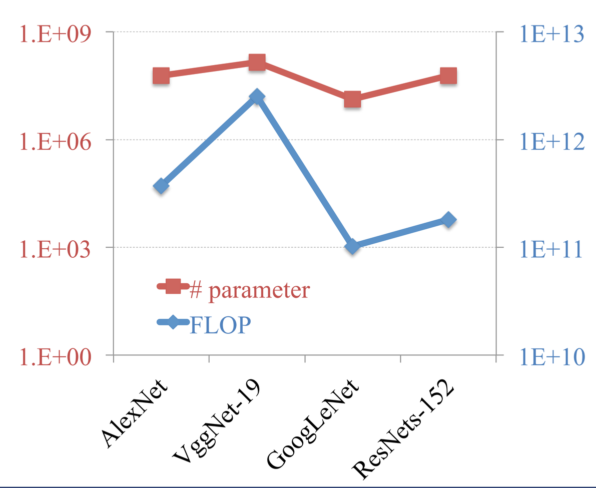
Trends

- ✓ Higher classification performance -- human-level performance @ ImageNet.
- ✓ Deeper neural networks -- several layers to hundreds or thousands of layers.
- ✓ Larger-scale neural networks.
- ✓ More complex computation.



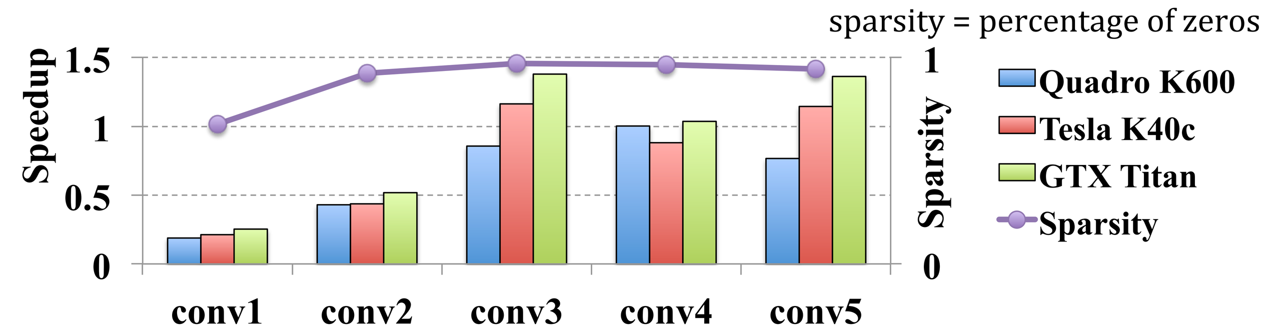
Computation Reduction

- ✓ FLOP: FLoating-point Operations Per test image.
- ✓ FLOP is positively related to the number of parameters -- reducing parameters can reduce computation.
- ✓ Methods: Connection pruning, L1 regularization, low-rank decomposition, *etc*.



Inefficiency of Sparse DNNs

sparsity = percentage of zeros

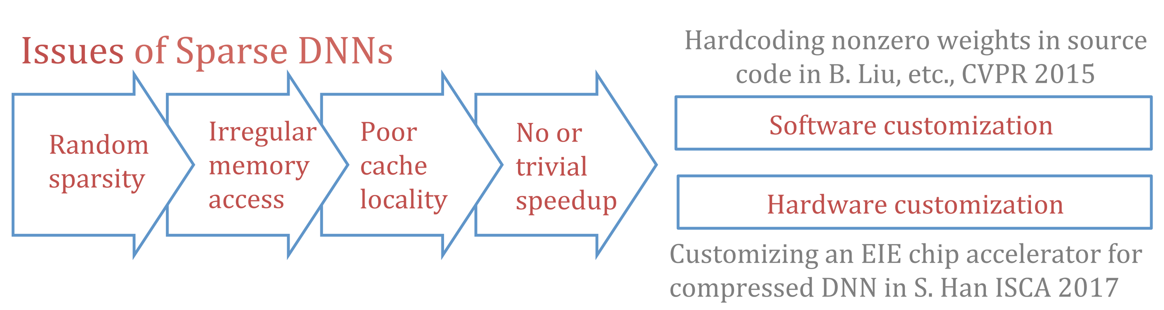


Speedup of testing of *AlexNet* on Nvidia GPUs using L1 regularization or connection pruning. The original dense *AlexNet* is benchmarked by cuda BLAS. The sparse weight matrixes of sparse *AlexNet* are stored in the format of Compressed Sparse Row and accelerated by cuSPARSE library.

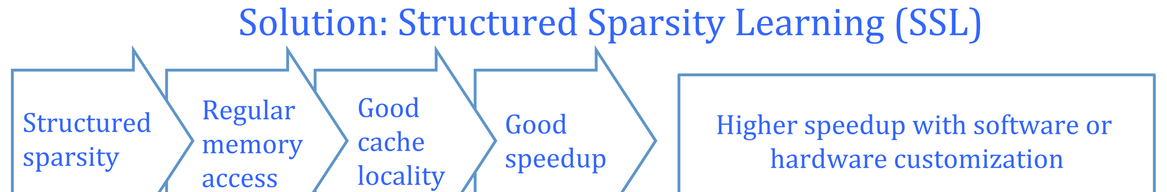
Results of state-of-the-art sparse DNNs

- ✓ 90% sparsity on average with 2% accuracy loss.
- ✓ Small speedup when sparsity is as high as 95% (conv3 and conv5)
- ✓ Slowing down in some cases (conv1, conv2 and conv4)

Issues of Sparse DNNs

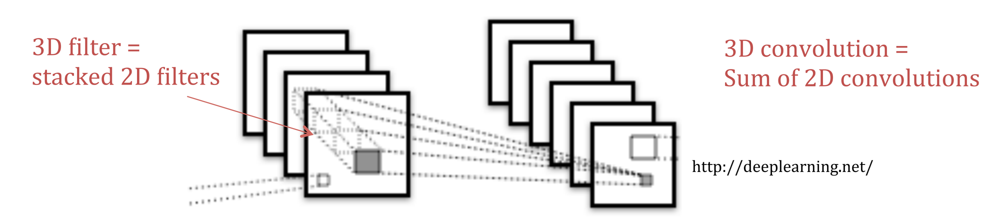


Solution: Structured Sparsity Learning (SSL)

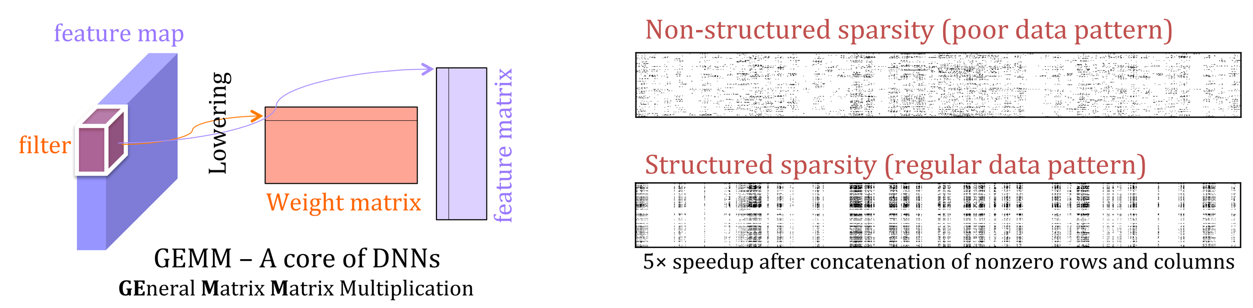


Efficiency of Structurally Sparse DNNs

Example 1: Structurally removing 2D filters = directly reducing 2D convolutions



Example 2: Removing rows/cols in weight matrices = reducing the dimensions of GEMM



Structured Sparsity Learning (SSL)

Structured sparsity learning by group Lasso regularization

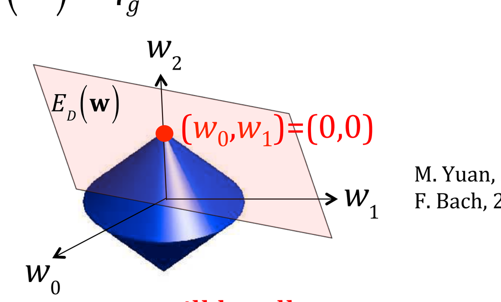
$$\arg\min_{\mathbf{w}}\left\{E(\mathbf{w})\right\}=\arg\min_{\mathbf{w}}\left\{E_D(\mathbf{w})+\lambda_g\cdot R_g(\mathbf{w})\right\}$$

$$R_g(\mathbf{w})=\sum_{g=1}^{G}\|\mathbf{w}^{(g)}\|_g \qquad \arg\min_{\mathbf{w}}\left\{E(\mathbf{w})\right\}=\arg\min_{\mathbf{w}}\left\{E_D(\mathbf{w})\right\}$$

$$\|\mathbf{w}^{(g)}\|_g=\sqrt{\sum_{i=1}^{N}\left(w_i^{(g)}\right)^2} \qquad s.t.\ R_g(\mathbf{w})\le\eta_g$$

Example:

$$R_g\big(\underbrace{w_0,w_1}_{\text{group 1}},\underbrace{w_2}_{\text{group 2}}\big)=\sqrt{w_0^2+w_1^2}+\sqrt{w_2^2}\le\eta_g$$



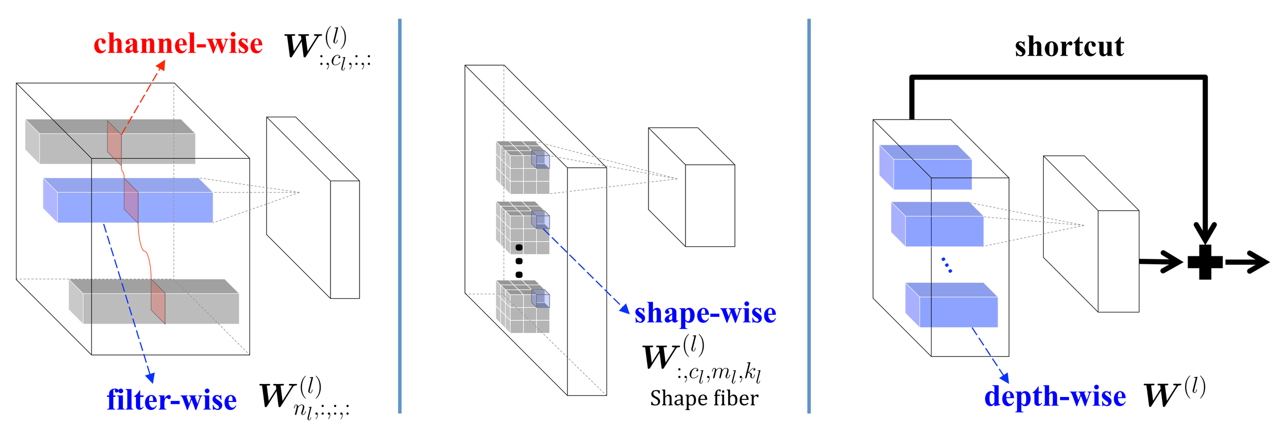
M. Yuan, 2006
F. Bach, 2012

Many groups will be all-zeros

Structured sparsity learning in DNNs:

$$E(\boldsymbol{W})=E_D(\boldsymbol{W})+\lambda\cdot R(\boldsymbol{W})+\lambda_g\cdot\sum_{l=1}^{L}R_g\left(\boldsymbol{W}^{(l)}\right)$$

Learned structured sparsity is determined by the way of splitting groups



Penalize unimportant filters and channels | **Learn filter shapes** | **Learn the depth of layers**

$$E(\boldsymbol{W})=E_D(\boldsymbol{W})+\lambda_n\cdot\sum_{l=1}^{L}\left(\sum_{n_l=1}^{N_l}\|\boldsymbol{W}^{(l)}_{n_l,:,:,:}\|_g\right)+\lambda_c\cdot\sum_{l=1}^{L}\left(\sum_{c_l=1}^{C_l}\|\boldsymbol{W}^{(l)}_{:,c_l,:,:}\|_g\right).$$

$$E(\boldsymbol{W})=E_D(\boldsymbol{W})+\lambda_s\cdot\sum_{l=1}^{L}\left(\sum_{c_l=1}^{C_l}\sum_{m_l=1}^{M_l}\sum_{k_l=1}^{K_l}\|\boldsymbol{W}^{(l)}_{:,c_l,m_l,k_l}\|_g\right).$$

$$E(\boldsymbol{W})=E_D(\boldsymbol{W})+\lambda_d\cdot\sum_{l=1}^{L}\|\boldsymbol{W}^{(l)}\|_g.$$

$\boldsymbol{W}^{(l)}$:weight tensor in the l-th layer with axes in the order of (filter #, channel #, kernel height, kernel width)

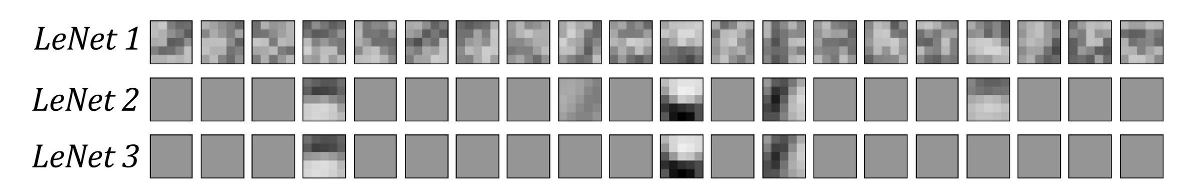
Learning filter, channel and neuron

Learning the number of filters and channels:

Table 1: Results after penalizing unimportant filters and channels in *LeNet*

LeNet #	Error	Filter # §	Channel # §	FLOP §	Speedup §
1 (*baseline*)	0.9%	20—50	1—20	100%—100%	1.00×—1.00×
2	0.8%	5—19	1—4	25%—7.6%	1.64×—5.23×
3	1.0%	3—12	1—3	15%—3.6%	1.99×—7.44×

§ In the order of *conv1—conv2*

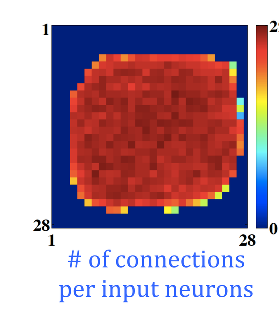


Fewer but smoother feature extractors

Learning the number of neurons:

MLP #	Error	Neuron # per layer §	FLOP per layer §
1 (*baseline*)	1.43%	784–500–300–10	100%–100%–100%
2	1.34%	469–294–166–10	35.18%–32.54%–55.33%
3	1.53%	434–174–78–10	19.26%–9.05%–26.00%

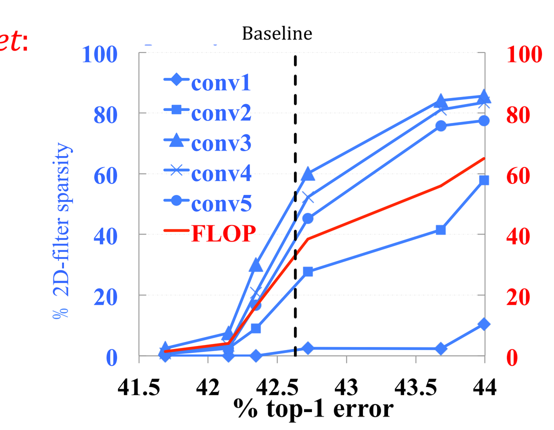
§ In the order of *input layer–hidden layer 1–hidden layer 2–output layer*



# of connections per input neurons

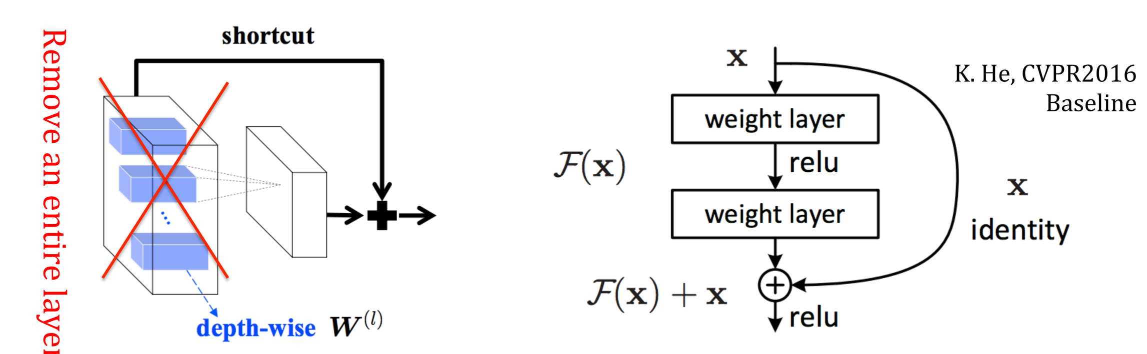
Learning the number of 2D filters @ *AlexNet*:

- ✓ Save FLOP by structurally removing 2D filters
- ✓ Save 30%–40% FLOP without accuracy loss
- ✓ Save 60%–70% FLOP with <1.5% accuracy loss
- ✓ Deeper layer has higher sparsity
- ✓ Reduce the error of *AlexNet* by ~1%



Learning the depth of DNNs

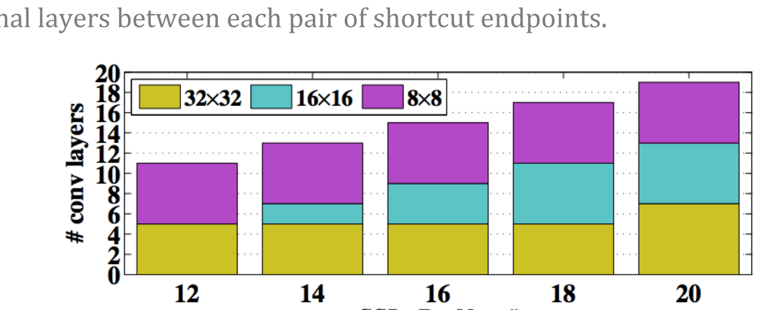
Experiments of ResNets on CIFAR-10



	# layers	error	# layers	error
ResNet	20	8.82%	32	7.51%
SSL-ResNet	14	8.54%	18	7.40%

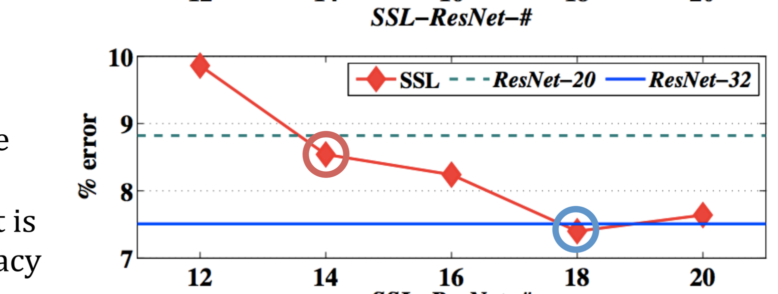
ResNet-20/32: baseline with 20/32 layers.
SSL-ResNet-#: Ours with # layers after learning depth of ResNet-20. Group Lasso regularization is only enforced on the convolutional layers between each pair of shortcut endpoints.

32×32 indicates the convolutional layers with an output map size of 32×32, and so forth.



SSL-ResNet

- ✓ Less layers with similar accuracy
- ✓ Higher accuracy with the same number of layers
- ✓ The depth of SSL-ResNet is still important for accuracy



Learning weight matrix dimensions

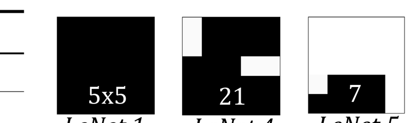
In GEMM computation

Removing filters = removing rows in weight matrices (filter-wise sparsity = row-wise sparsity)
Removing shape fibers = removing columns in weight matrices (shape-wise sparsity = column-wise sparsity)
↓
Concatenating non-zero rows and columns to a smaller dense weight matrix to save computation

Table 2: Results after learning filter shapes in *LeNet*

LeNet #	Error	Filter size §	Channel #	FLOP	Speedup
1 (*baseline*)	0.9%	25—500	1—20	100%—100%	1.00×—1.00×
4	0.8%	21—41	1—2	8.4%—8.2%	2.33×—6.93×
5	1.0%	7—14	1—1	1.4%—2.8%	5.19×—10.82×

§ The sizes of filters after removing zero shape fibers, in the order of *conv1—conv2*



Learned shapes of conv1 filters

Learned shape of conv2 filters @ *LeNet 5*, 3D 20x5x5 filters is regularized to 2D filters!

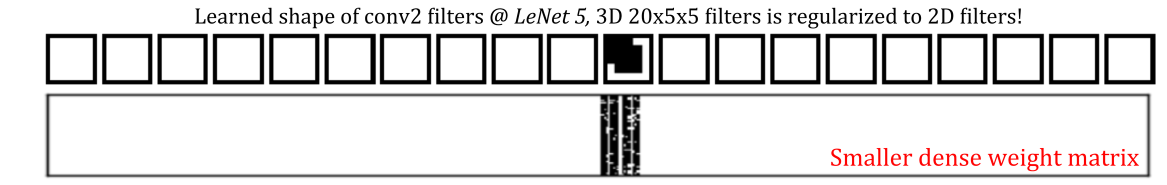


Table 3: Learning row-wise and column-wise sparsity of *ConvNet* on CIFAR-10

ConvNet #	Error	Row sparsity §	Column sparsity §	Speedup §
1 (*baseline*)	17.9%	12.5%–0%–0%	0%–0%–0%	1.00×–1.00×–1.00×
2	17.9%	50.0%–28.1%–1.6%	0%–59.3%–35.1%	1.43×–3.05×–1.57×
3	16.9%	31.3%–0%–1.6%	0%–42.8%–9.8%	1.25×–2.01×–1.18×

§ in the order of *conv1–conv2–conv3*

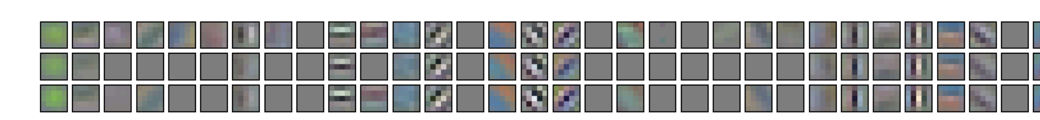
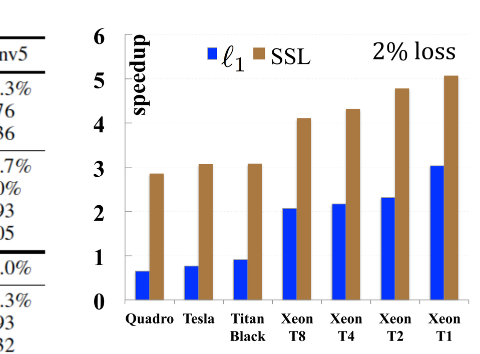


Figure 5: Learned *conv1* filters in *ConvNet 1* (top), *ConvNet 2* (middle) and *ConvNet 3* (bottom)

Table 4: Sparsity and speedup of *AlexNet* on ILSVRC 2012

	#	Method	Top1 err.	Statistics	conv1	conv2	conv3	conv4	conv5
2% loss	1	ℓ_1	44.67%	sparsity	67.6%	92.4%	97.2%	96.6%	94.3%
				CPU ×	0.80	2.91	4.84	3.83	2.76
				GPU ×	0.25	0.52	1.38	1.04	1.36
	2	SSL	44.66%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%
				row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%
				CPU ×	1.05	3.37	6.27	9.73	4.93
				GPU ×	1.00	2.37	4.94	4.03	3.05
	3	pruning[6]	42.80%	sparsity	16.0%	62.0%	65.0%	63.0%	63.0%
No loss	4	ℓ_1	42.51%	sparsity	14.7%	76.2%	85.3%	81.5%	76.3%
				CPU ×	0.34	0.99	1.30	1.10	0.93
				GPU ×	0.08	0.17	0.42	0.30	0.32
	5	SSL	42.53%	column sparsity	0.00%	20.9%	39.7%	39.7%	24.6%
				CPU ×	1.00	1.27	1.64	1.68	1.32
				GPU ×	1.00	1.25	1.63	1.72	1.36

[6] S. Han, NIPS 2016



L1 regularization is adopted since it obtains higher sparsity than connection pruning in conv layers.

- ✓ On average, layer-wise 5.1× /3.1× on CPUs/GPUs with 2% accuracy loss.
- ✓ On average, layer-wise 1.4× on both CPU and GPU w/o accuracy loss.
- ✓ Non-structured sparsity method even slows down the computation in some layers.
- ✓ Sparse *AlexNet* with structured sparsity gets 2× speedups of the one with non-structured sparsity.

Conclusion

- ✓ We propose a Structured Sparsity Learning (SSL) method to regularize filter, channel, neuron, filter shape, and depth structures in Deep Neural Networks (DNN).
- ✓ SSL can enforce DNNS to dynamically learn more compact structures without accuracy loss.
- ✓ The structured sparsity in DNNs achieves significant speedups for the DNN evaluation both on CPU and GPU with off-the-shelf libraries.
- ✓ A variant of SSL can be performed as structure regularization to improve classification accuracy of state-of-the-art DNNs.
- ✓ SSL may achieve higher speedups when combining with hardware/software customization.

Acknowledgments

This work was supported in part by NSF XPS-1337198 and NSF CCF-1615475. The authors thank Drs. Sheng Li and Jongsoo Park for valuable feedback on this work.