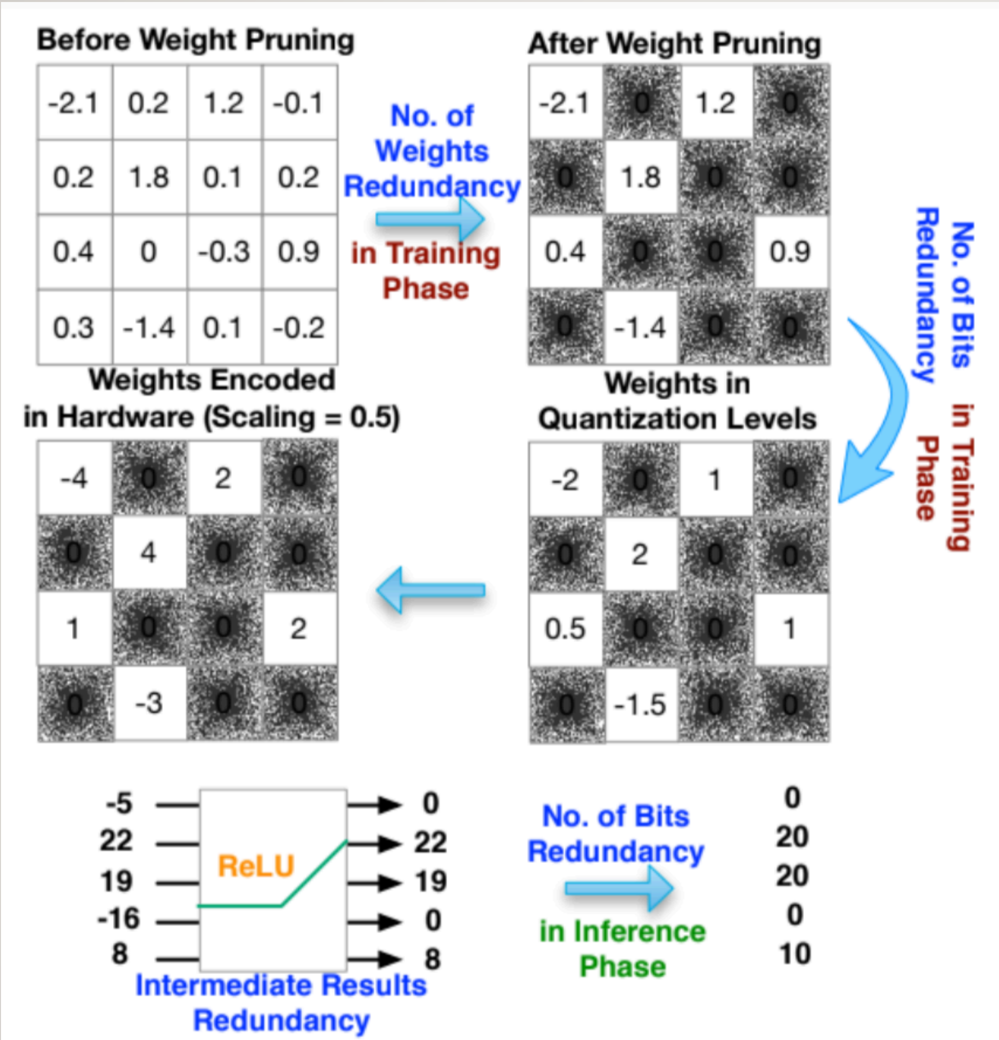
Emerging use of ADMM In Deep learning

Shaokai Ye

advisor: Prof. Yanzhi Wang

January 6, 2019

Sources of redundancy in Neural Networks



- Redundancy in the number of weights
- Redundancy in the bit representation of weights
- Redundancy in the intermediate results
- Redundancy in the bit representation of intermediate results

Opportunity: There is a lack of a unifying training framework to systematically exploit those redundancies all together.

ADMM(Alternating Direction Method of Multipliers) is able to train neural networks with combinatorial constrains with promising results.

Use of figure is authorized by Prof. Yanzhi Wang

What's ADMM?

- An effective mathematical optimization, by decomposing an original problem into two subproblems that can be solved separately and efficiently
- Consider the optimization problem

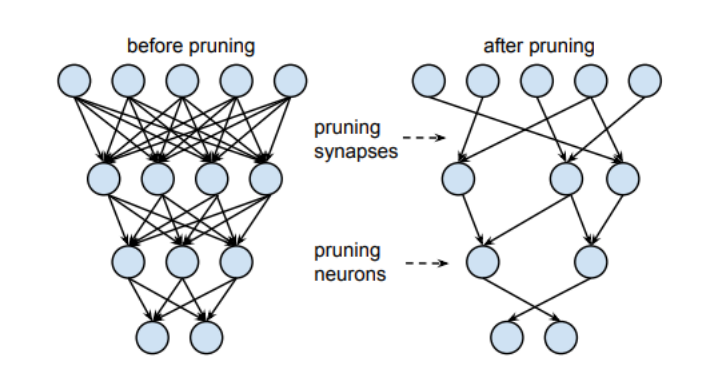
$$\min_{\mathbf{x}} \; f(\mathbf{x}) + g(\mathbf{x}).$$

- The problem can be first re-written into

$$\min_{\mathbf{x},\mathbf{z}} \; f(\mathbf{x}) + g(\mathbf{z}), \quad \text{subject to} \;\; \mathbf{x} = \mathbf{z}.$$

Weight Pruning
Redundancy source: weight

Fig. from [S. Han et al., NeuralPS 2015]:



Formulation of weight pruning:

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f\big(\{\mathbf{W}_i\},\{\mathbf{b}_i\}\big)+\sum_{i=1}^{N} g_i(\mathbf{W}_i),$$

$$g_i(\mathbf{W}_i)=\begin{cases}0 & \text{if } \operatorname{card}(\mathbf{W}_i)\le l_i,\\ +\infty & \text{otherwise.}\end{cases}$$

The original pruning problem is not differentiable, thus not applicable through backpropagation

ADMM formulation

$$\begin{aligned}&\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f\big(\{\mathbf{W}_i\},\{\mathbf{b}_i\}\big)+\sum_{i=1}^{N} g_i(\mathbf{Z}_i),\\ &\text{subject to} \quad \mathbf{W}_i=\mathbf{Z}_i,\ i=1,\dots,N.\end{aligned}$$

Augmented Lagrangian of ADMM formulation

$$L_\rho\big(\{\mathbf{W}_i\},\{\mathbf{b}_i\},\{\mathbf{Z}_i\},\{\mathbf{\Lambda}_i\}\big)=f\big(\{\mathbf{W}_i\},\{\mathbf{b}_i\}\big)+\sum_{i=1}^{N} g_i(\mathbf{Z}_i) +\sum_{i=1}^{N}\frac{\rho_i}{2}\|\mathbf{W}_i-\mathbf{Z}_i+\mathbf{U}_i\|_F^2-\sum_{i=1}^{N}\frac{\rho_i}{2}\|\mathbf{U}_i\|_F^2.$$

Solving 2 sub-problems iteratively

$$\begin{aligned}\{\mathbf{W}_i^{k+1},\mathbf{b}_i^{k+1}\} &:= \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\arg\min} \quad L_\rho\big(\{\mathbf{W}_i\},\{\mathbf{b}_i\},\{\mathbf{Z}_i^k\},\{\mathbf{U}_i^k\}\big)\\ \{\mathbf{Z}_i^{k+1}\} &:= \underset{\{\mathbf{Z}_i\}}{\arg\min} \quad L_\rho\big(\{\mathbf{W}_i^{k+1}\},\{\mathbf{b}_i^{k+1}\},\{\mathbf{Z}_i\},\{\mathbf{U}_i^k\}\big)\\ \mathbf{U}_i^{k+1} &:= \mathbf{U}_i^k+\mathbf{W}_i^{k+1}-\mathbf{Z}_i^{k+1},\end{aligned}$$

ADMM: Iteratively solving two sub-problems

$$\{\mathbf{W}_i^{k+1}, \mathbf{b}_i^{k+1}\} := \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\arg\min} \quad L_\rho\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}, \{\mathbf{Z}_i^k\}, \{\mathbf{U}_i^k\}\big)$$

$$\{\mathbf{Z}_i^{k+1}\} := \underset{\{\mathbf{Z}_i\}}{\arg\min} \quad L_\rho\big(\{\mathbf{W}_i^{k+1}\}, \{\mathbf{b}_i^{k+1}\}, \{\mathbf{Z}_i\}, \{\mathbf{U}_i^k\}\big)$$

$$\mathbf{U}_i^{k+1} := \mathbf{U}_i^k + \mathbf{W}_i^{k+1} - \mathbf{Z}_i^{k+1},$$

sub-problem (1)
sub-problem (2)

SGD to solve sub-problem(1)

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}\big) + \sum_{i=1}^{N} \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Z}_i^k + \mathbf{U}_i^k\|_F^2,$$

Euclidean projection to solve sub-problem(2)

$$\mathbf{Z}_i^{k+1} = \mathbf{\Pi}_{\mathbf{S}_i}(\mathbf{W}_i^{k+1} + \mathbf{U}_i^k),$$

$$\|\mathbf{W}_i^{k+1} - \mathbf{Z}_i^{k+1}\|_F^2 \ \le \epsilon_i, \quad \|\mathbf{Z}_i^{k+1} - \mathbf{Z}_i^k\|_F^2 \ \le \epsilon_i.$$

Empirically, around 10-20 iterations, this condition is satisfied.

After the condition is satisfied, we use Euclidean projection (mapping) to guarantee weights are truly sparse.
Then we apply masked retraining to retrain nonzero weights.

Structured pruning

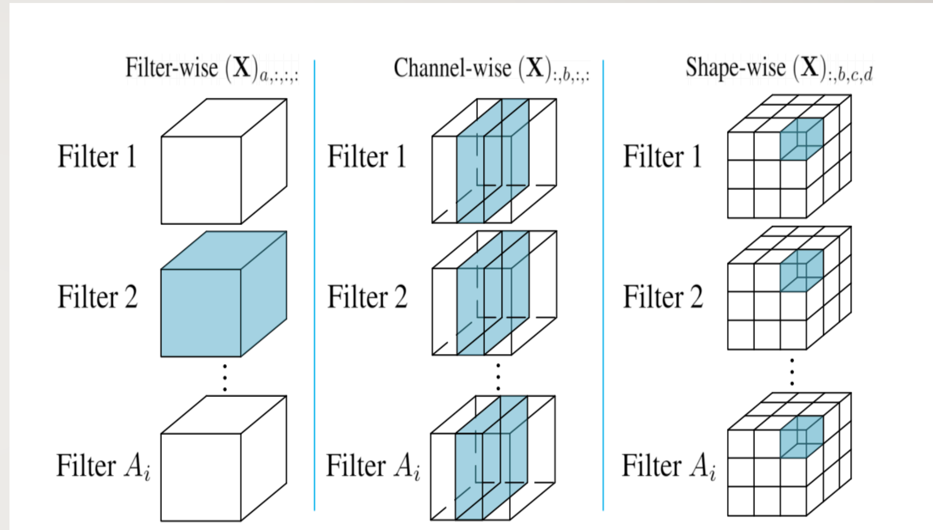
Redundancy Source: Weight in a structured form

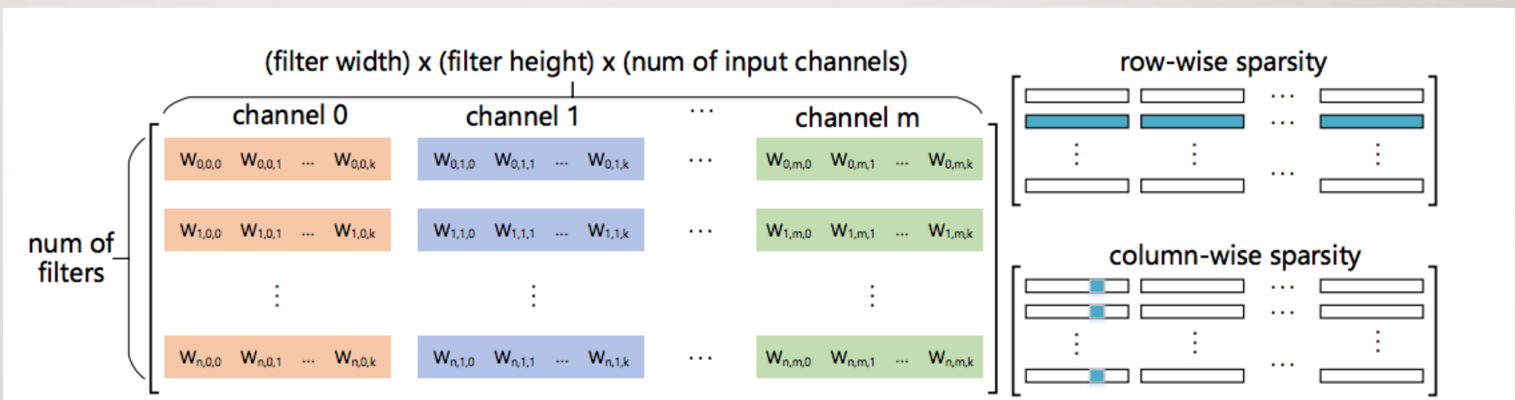
Pros: leverage GPUs, direct speed up for GEMM

Cons: Less storage reduction

Opportunity:

For tiny network such as Mobile-net, 90% computation is taken by 1x1 kernel, which is essentially done in GEMM





Tianyun Zhang*, Kaiqi Zhang*, Shaokai Ye*, Jiayu Li, Jian Tang, Wujie Wen, Xue Lin, Makan Fardad, Yanzhi Wang. "ADAM-ADMM: A Unified Systematic Framework of Structured pruning for DNNs."

Results on Structured Weight Pruning for CaffeNet

Structured pruning, with no accuracy loss

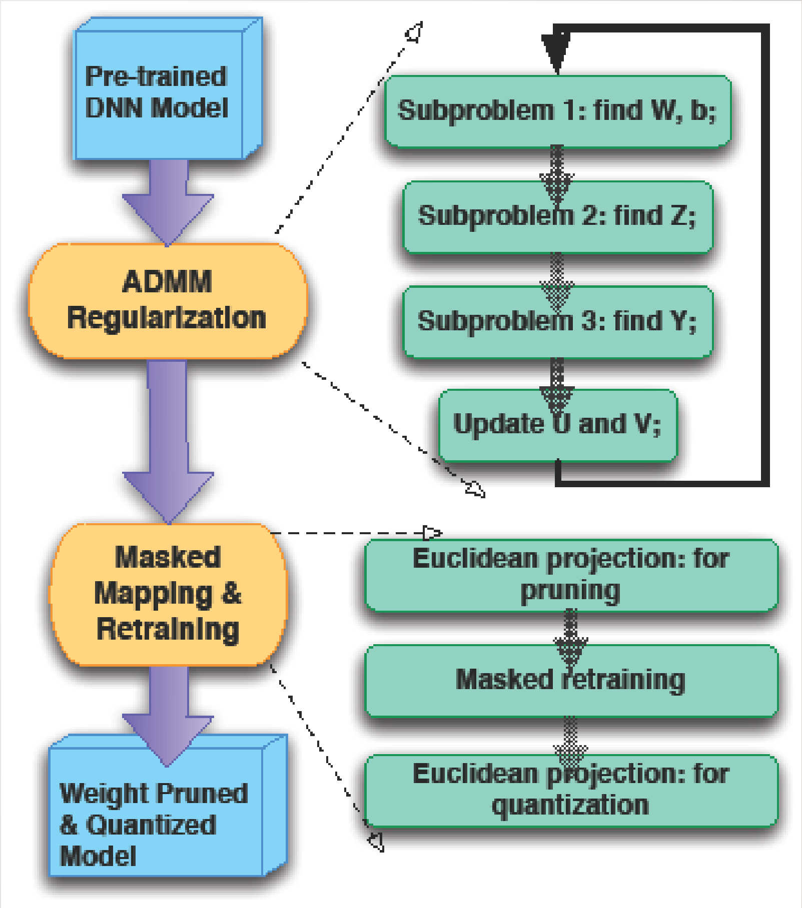
Method	Top1 error	Statistcs	conv1	conv2	conv3	conv4	conv5	conv2-5[§]
SSL [6]	42.53%	column sparsity	0.0%	20.9%	39.7%	39.7%	24.6%	1.5×
our method	40.96%	column sparsity	0.0%	20.9%	39.7%	39.7%	24.6%	1.5×
our method	42.53%	column sparsity	0.0%	70.0%	77.0%	85.0%	81.0%	4.8×
		GPU1×	1.00	2.27	3.35	3.64	1.04	
		GPU2×	1.00	2.83	3.92	4.63	3.22	

Structured pruning, within 2% accuracy loss

Method	Top1 error	Statistcs	conv1	conv2	conv3	conv4	conv5	conv2-5[§]
SSL [6]	44.66%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%	6.4×
		row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%	
our method	43.35%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%	6.4×
		row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%	
		CPU×	1.05	2.76	6.28	8.64	3.92	
		GPU1×	1.00	1.25	4.10	1.49	1.19	
		GPU2×	1.00	2.29	6.52	5.94	3.25	
our method	44.67%	column sparsity	0.0%	87.1%	90.0%	90.0%	88.1%	13.2×
		row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%	
		CPU×	1.05	7.75	14.68	13.55	6.02	
		GPU1×	1.00	2.32	5.34	1.82	1.59	
		GPU2×	1.00	4.77	12.55	7.97	4.70	

ADMM-NN: An integrated Framework

- We develop an integrated framework of *ADMM regularization* and *masked mapping & retraining* steps
- We guarantee solution feasibility (satisfying all constraints) and provide high quality (maintaining test accuracy)



Weight quantization + Weight Pruning

Redundancy Source: Weight and Weight representation

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N g_i(\mathbf{Z}_i) + \sum_{i=1}^N h_i(\mathbf{Y}_i),$$
$$\text{subject to} \quad \mathbf{W}_i = \mathbf{Z}_i,\ \mathbf{W}_i = \mathbf{Y}_i,\ i = 1, \dots, N.$$

-1.01	1.00	0.00	0.88
0.00	0.17	0.00	-0.02
0.56	0.00	0.38	0.00
0.00	-0.49	-0.95	0.00

-1.00	1.00	0.00	1.00
0.00	0.50	0.00	-0.50
0.50	0.00	0.50	0.00
0.00	-0.50	-1.00	0.00

-2	2	0	2
0	1	0	-1
1	0	1	0
0	-1	-2	0

Implication:

- Highly desirable for FPGA
- Multiple ADMM regularizations

Shaokai Ye*, Tianyun Zhang*, Kaiqi Zhang, Jiayu Li, Jiaming Xie, Yuan Liang, Sijia Liu, Xue Lin & Yanzhi Wang. "A Unified Framework of DNN Weight Pruning and Weight/Clustering/Quantization Using ADMM."

Results on Joint Weight Pruning and Quantization for AlexNet

Model	Accuracy degradation	No. of weights	CONV weight bits	FC weight bits	Total data size/ Compress ratio	Total model size (including index)/ Compress ratio
AlexNet Baseline	0.0%	60.9M	32	32	243.6MB	243.6MB
Iterative pruning (Han, Mao, and Dally 2016)	0.0%	6.7M	8	5	5.4MB / 45×	9.0MB / 27×
Binary quant. (Leng et al. 2017)	3.0%	60.9M	1	1	7.3MB / 32×	7.3MB / 32×
Ternary quant. (Leng et al. 2017)	1.8%	60.9M	2	2	15.2MB / 16×	15.2MB / 16×
Our Method (Clustering)	0.1%	2.47M	5	3	1.16MB / 210×	2.7MB / 90×
Our Method (Quantization)	0.2%	2.47M	5	3	1.16MB / 210×	2.7MB / 90×

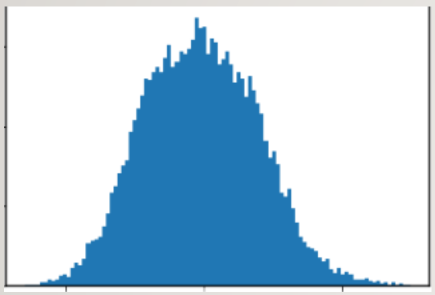
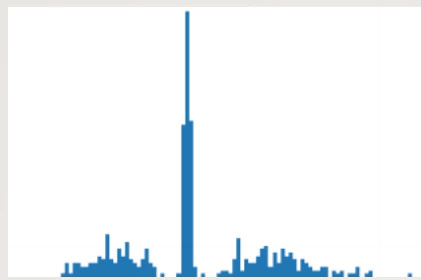
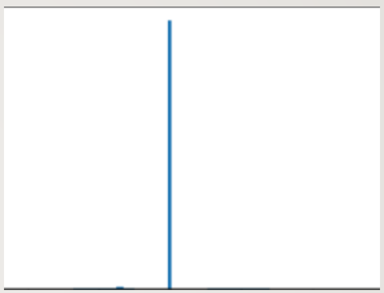
Shortcomings of ADMM

1. Sensitive to the choice of penalty parameter
2. Increasing length of training time when regularization target is far away from the original weights, making it hard to converge.

Our solution: Progressive ADMM

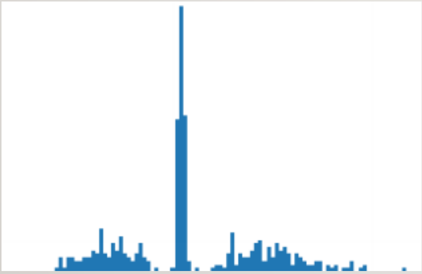
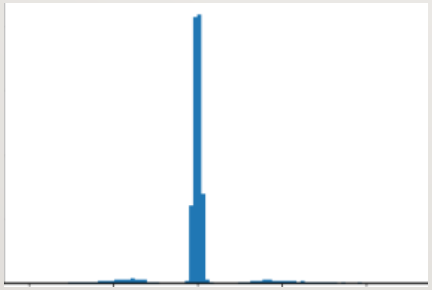
Direct ADMM

initial state after ADMM pruned

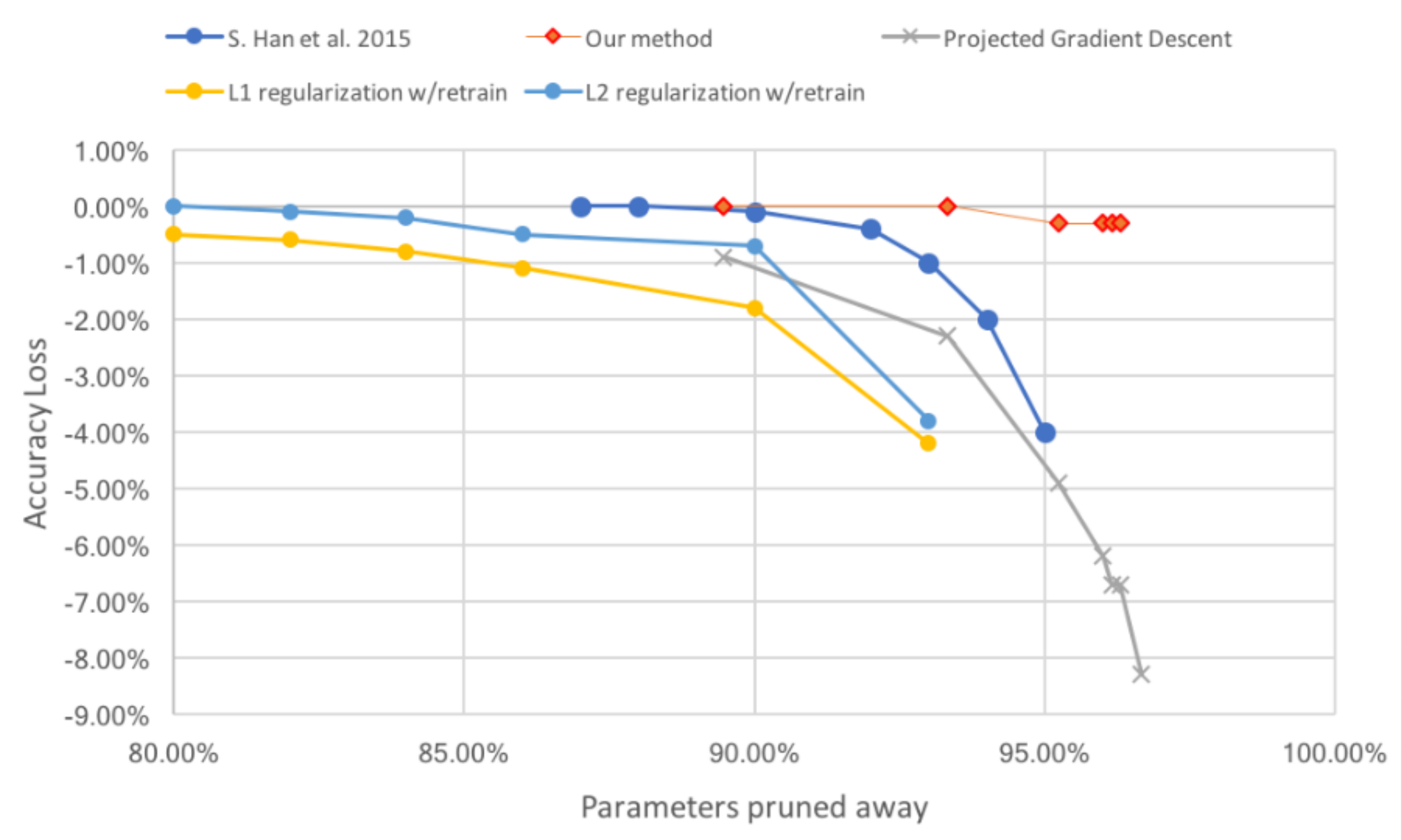
Progressive ADMM

initial state after ADMM pruned


Tianyun Zhang*, Shaokai Ye*, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, Yanzhi Wang. "A systematic DNN weight pruning framework using alternating direction method of multipliers."

Progressive ADMM makes pruning stable even pruning rates are extra high



Shaokai Ye*, Tianyun Zhang*, Kaiqi Zhang*, Xue Lin, Yanzhi Wang and et al. "Progressive weight pruning of Deep Neural Networks using ADMM"

Progressive ADMM makes highest pruning rates ever

- Weight pruning ratio and accuracy on LeNet-5 (MNIST data set)

Method	Accuracy	No. Para.	Rate
Uncompressed	99.2%	431K	1×
Network Pruning (Han et al., 2015)	99.2%	36K	12.5×
ADMM Pruning (Zhang et al., 2018b)	99.2%	6.05K	71.2×
Optimal Brain Surgeon (Dong et al., 2017)	98.3%	3.88K	111×
Our Proposed Method	99.0%	2.58K	167×

new version achieves 243X

- Weight pruning ratio and accuracy on AlexNet (ImageNet dataset)

Method	Top-5 Acc.	No. Para.	Rate
Uncompressed	80.27%	61.0M	1×
Network Pruning (Han et al., 2015)	80.3%	6.7M	9×
Optimal Brain Surgeon (Dong et al., 2017)	80.0%	6.7M	9.1×
Low Rank and Sparse Decomposition (Yu et al., 2017)	80.3%	6.1M	10×
Fine-Grained Pruning (Mao et al., 2017)	80.4%	5.1M	11.9×
NeST (Dai et al., 2017)	80.2%	3.9M	15.7×
Our Proposed Method (BVLC Model)	80.0%	2.0M	31×
Our Proposed Method (CaffeNet Model)	80.0%	2.0M	31×

Progressive ADMM makes highest pruning rates ever

- Weight pruning ratio and accuracy on VGGNet (ImageNet dataset)

Method	Top-5 Acc.	No. Para.	Rate
Uncompressed	88.7%	138M	1×
Network Pruning (Han et al., 2015)	89.1%	10.6M	13×
Optimal Brain Surgeon (Dong et al., 2017)	89.0%	10.3M	13.3×
Low Rank and Sparse Decomposition (Yu et al., 2017)	89.1%	9.2M	15×
Our Proposed Method	88.7%	4.6M	30×
Our Proposed Method	88.2%	4.1M	34×

- Weight pruning ratio and accuracy on ResNet-50 (ImageNet dataset)

Method	Acc. Loss	No. Para.	Rate
Uncompressed	0%	25.6M	1×
Fine-grained Pruning (Mao et al., 2017)	0%	9.8M	2.6×
AMC (He et al., 2018)	0%	5.1M	5×
Our Method	0%	2.8M	9.2×
Our Method	0.7%	1.47M	17.4×

Other results

- The gain is more significant for CONV layers
  –Which are more computationally intensive than FC layers
  –For example, we achieve 13.1x weight reduction in CONV layers of AlexNet without accuracy degradation, whereas prior work is only 2.7x
- Test accuracy even increases with moderate pruning rates
  –Using example of AlexNet (original accuracy 80.2%)

Pruning Rate	Our Proposed Method
18×	80.9%
21×	80.8%
30×	80.2%

Binary Quantization

Scenario: performance degradation when quantizing first and last layer.
More than 10 points accuracy drop is observed by other methods(ResNet-18).

However, using progressive ADMM, only around 6 points accuracy drop is observed.

Ongoing work

Co-training for both adversarial robustness and model compression using ADMM(Results in Lenet-5)

Table 1: Filter Pruning from robust pretrained model

Pruning Rate	Nat. Acc	Adv. Acc
1×	98.53%	94.52%
2×	98.65%	93.73%
3.25×	98.53%	93.34%

Contribution

Unlike prior work, state-of-art pruning and robustness can be achieved at the same time.