Neural Network Compression

國立清華大學

黃柏瑜

Current Methods

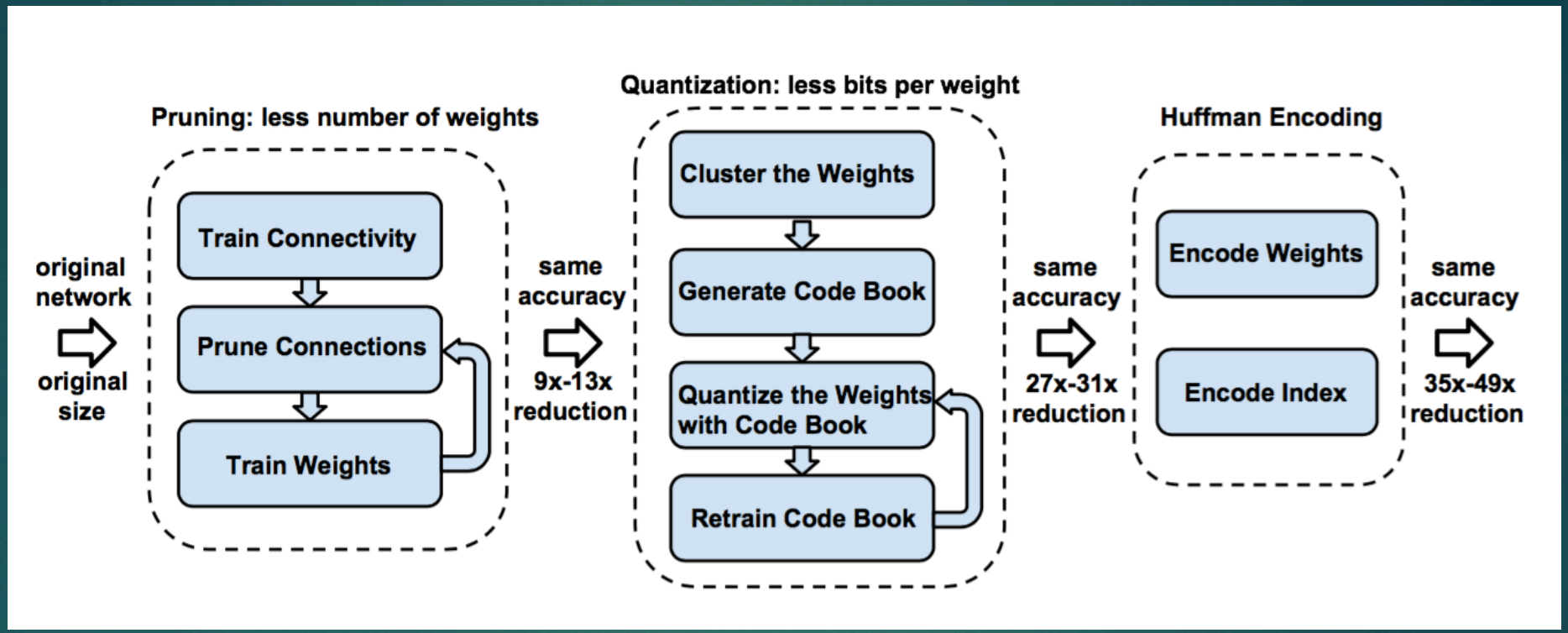
- Compressing pre-trained deep networks
  - Speed up at testing time
- Design compact layers
  - Speed up at both testing and training
- Quantizing parameters
  - 32 -> 16 -> 1 bit(s)

DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING

SONG HAN, STANFORD UNIVERSITY, STANFORD, CA 94305, USA
HUIZI MAO, TSINGHUA UNIVERSITY, BEIJING, 100084, CHINA
AWILLIAM J. DALLY, STANFORD UNIVERSITY, STANFORD, CA 94305, USA

Abstract

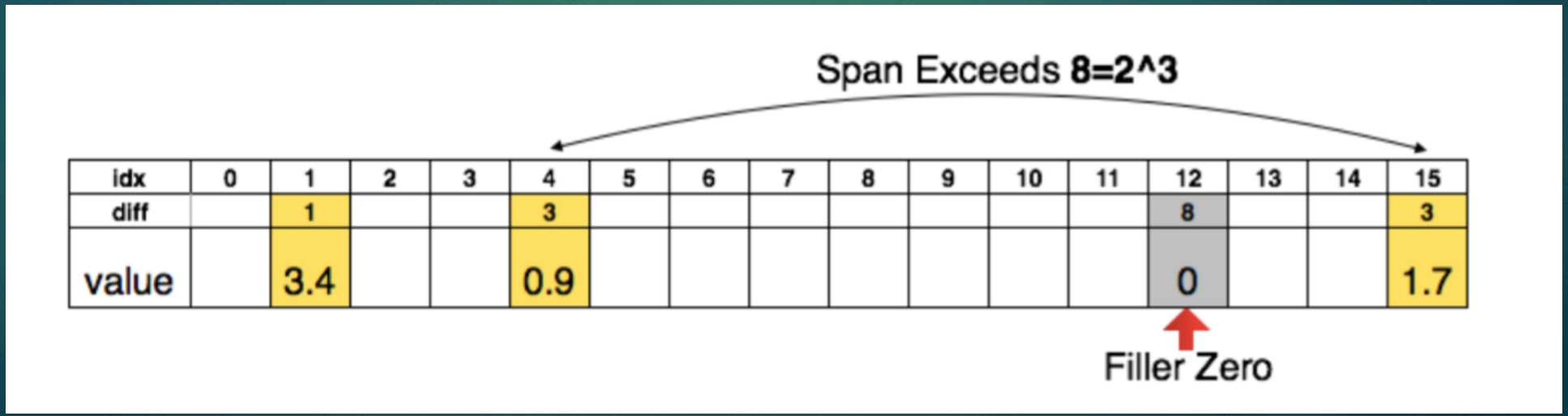
- Three stage pipeline of Deep Compression
  - Pruning
  - Quantization
  - Huffman code
- AlexNet reduce 35x
- VGG-16 reduce 49x
- Compress with no loss of accuracy


Pruning: less number of weights
Train Connectivity
Prune Connections
Train Weights
original network
original size
same accuracy
9x-13x reduction
Quantization: less bits per weight
Cluster the Weights
Generate Code Book
Quantize the Weights with Code Book
Retrain Code Book
same accuracy
27x-31x reduction
Huffman Encoding
Encode Weights
Encode Index
same accuracy
35x-49x reduction

Network Pruning

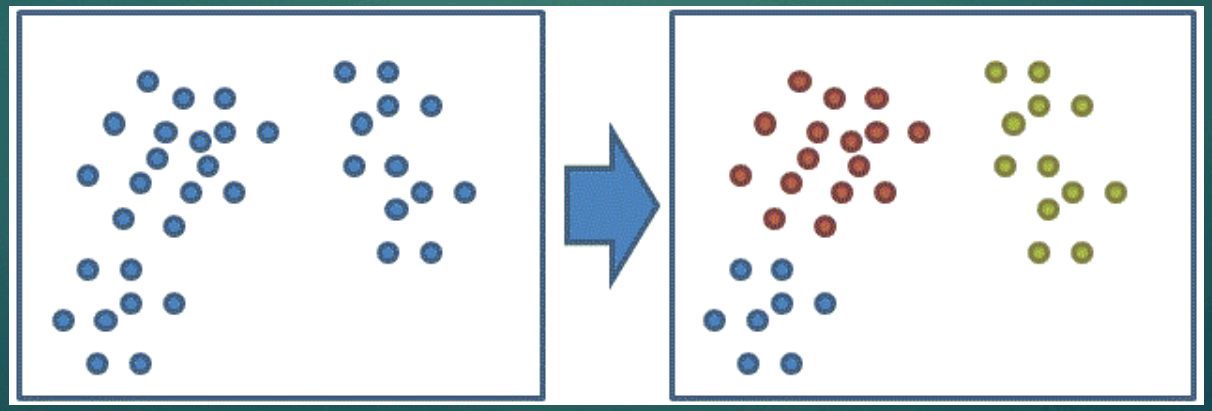
- Prune weights < threshold
- Retrain the network
- Representing the matrix sparsity with relative index

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

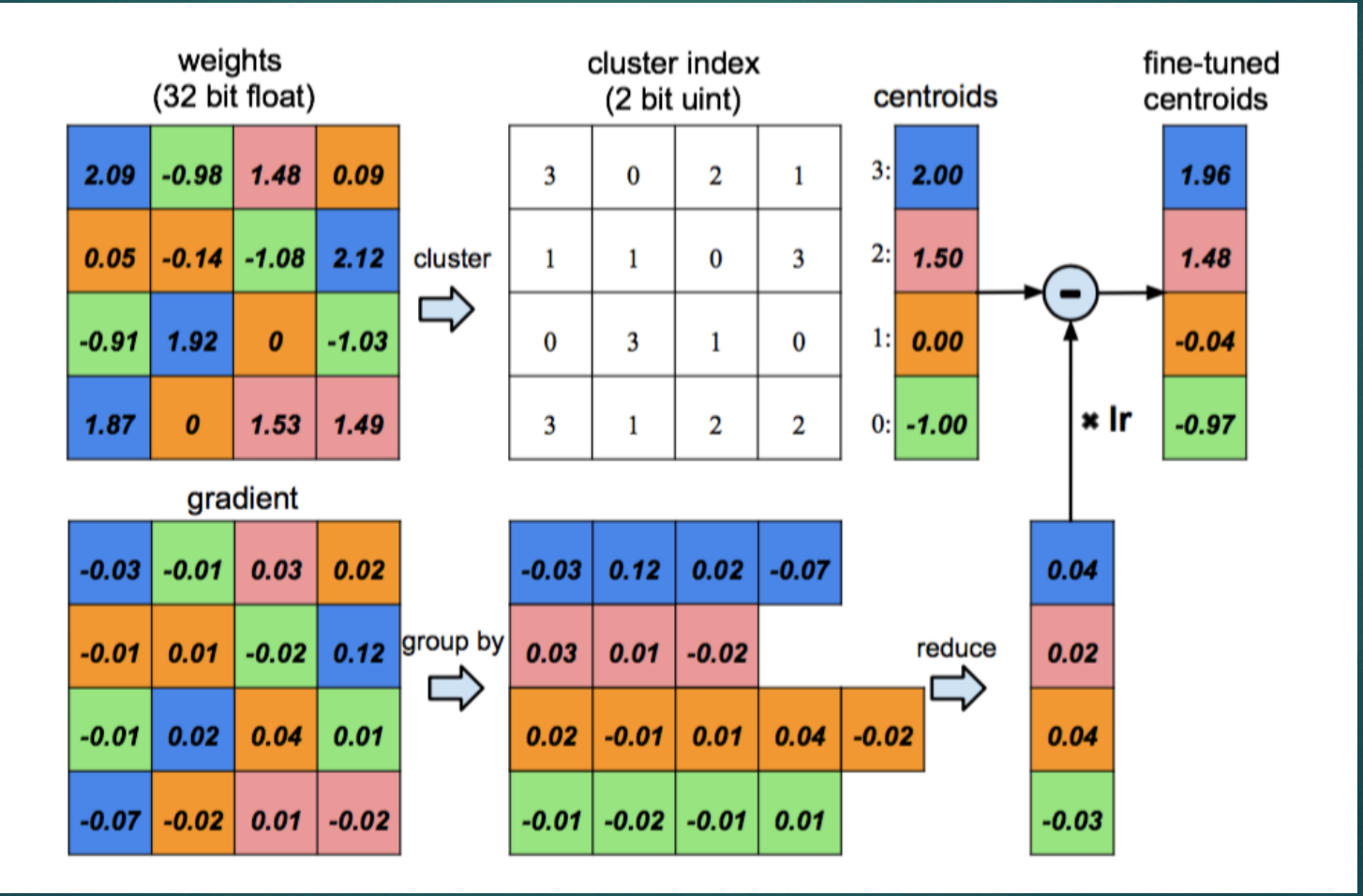


Quantization & Weight Sharing

- Use k-means to cluster the weights
- We only need to store k effective weight(n*32 bits) and n*log2(k)



Quantization & Weight Sharing



Quantization & Weight Sharing

- 1M*32/(1M*8+256*32) = 3.99

$$r = \frac{nb}{nlog_2(k) + kb}$$

Huffman Coding

- More common symbols are represented with fewer bits

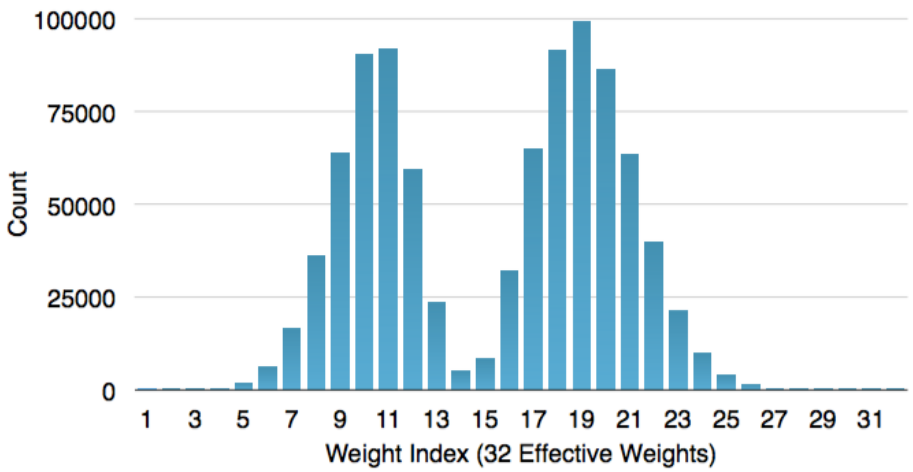


Figure 5: Distribution for weight (Left) and index (Right). The distribution is biased.

Experiment

- Implement on different network (Mnist)

Table 1: The compression pipeline can save 35× to 49× parameter storage with no loss of accuracy.

Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	**40**×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39**×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35**×
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49**×

Experiment

- Layer-wise compress result

Table 5: Compression statistics for VGG-16. P: pruning, Q:quantization, H:Huffman coding.

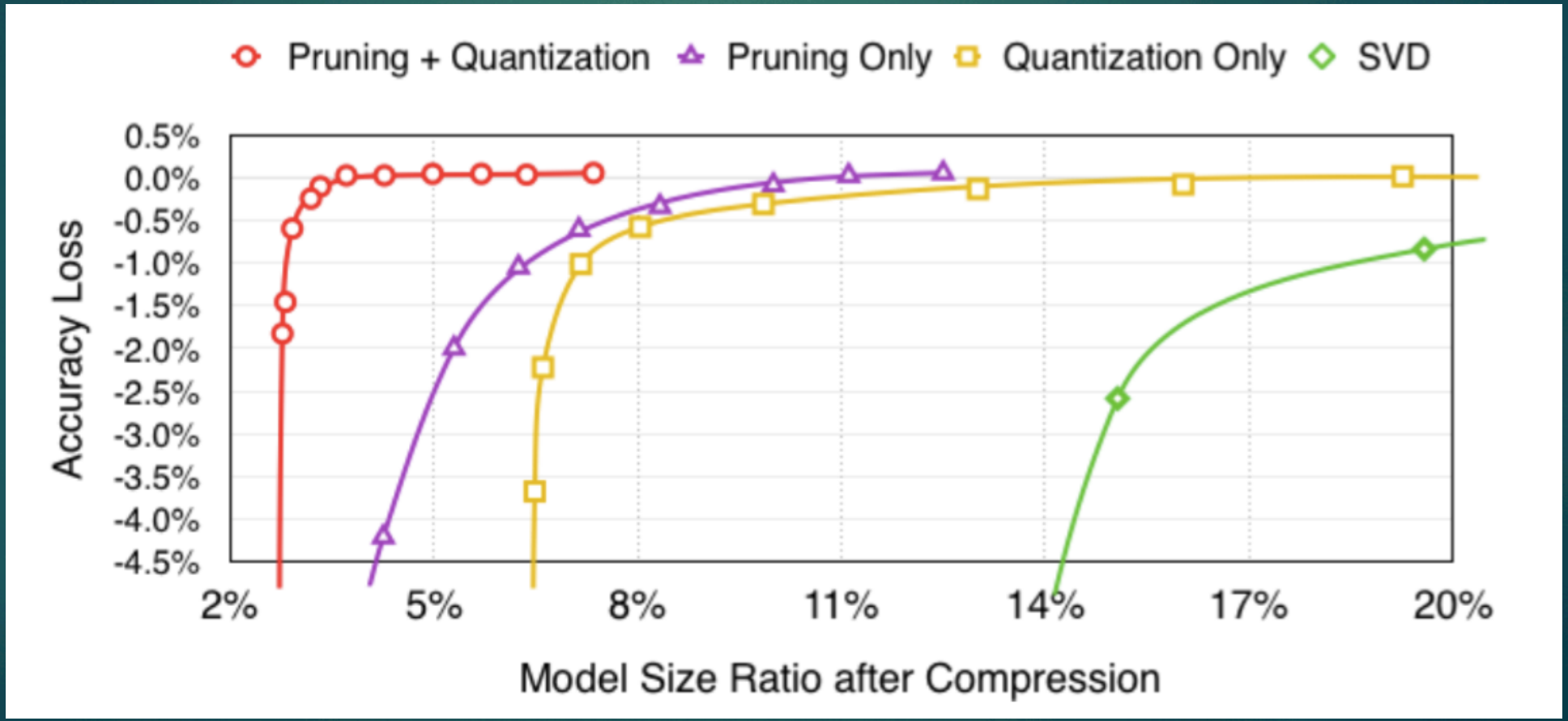
Layer	#Weights	Weights% (P)	Weigh bits (P+Q)	Weight bits (P+Q+H)	Index bits (P+Q)	Index bits (P+Q+H)	Compress rate (P+Q)	Compress rate (P+Q+H)
conv1_1	2K	58%	8	6.8	5	1.7	40.0%	29.97%
conv1_2	37K	22%	8	6.5	5	2.6	9.8%	6.99%
conv2_1	74K	34%	8	5.6	5	2.4	14.3%	8.91%
conv2_2	148K	36%	8	5.9	5	2.3	14.7%	9.31%
conv3_1	295K	53%	8	4.8	5	1.8	21.7%	11.15%
conv3_2	590K	24%	8	4.6	5	2.9	9.7%	5.67%
conv3_3	590K	42%	8	4.6	5	2.2	17.0%	8.96%
conv4_1	1M	32%	8	4.6	5	2.6	13.1%	7.29%
conv4_2	2M	27%	8	4.2	5	2.9	10.9%	5.93%
conv4_3	2M	34%	8	4.4	5	2.5	14.0%	7.47%
conv5_1	2M	35%	8	4.7	5	2.5	14.3%	8.00%
conv5_2	2M	29%	8	4.6	5	2.7	11.7%	6.52%
conv5_3	2M	36%	8	4.6	5	2.3	14.8%	7.79%
fc6	103M	4%	5	3.6	5	3.5	1.6%	1.10%
fc7	17M	4%	5	4	5	4.3	1.5%	1.25%
fc8	4M	23%	5	4	5	3.4	7.1%	5.24%
Total	138M	7.5%(13×)	6.4	4.1	5	3.1	3.2% (**31**×)	2.05% (**49**×)

Discussion

- Pruning + Quantize
- Centroid initialization
- Speedup and energy efficiency

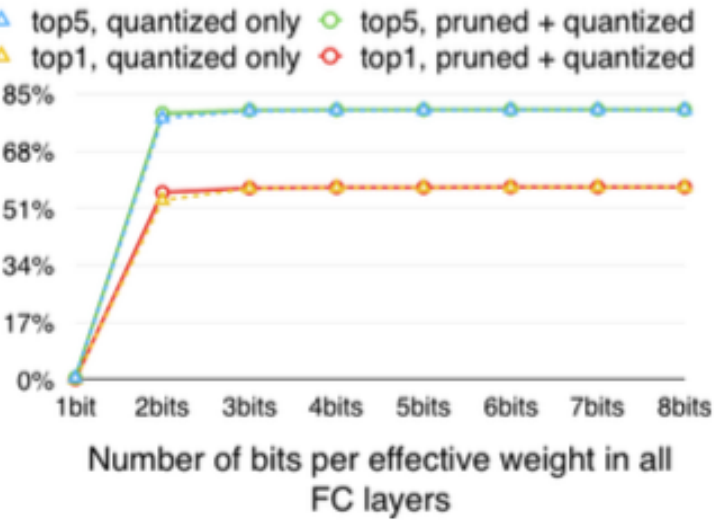
Pruning + Quantize

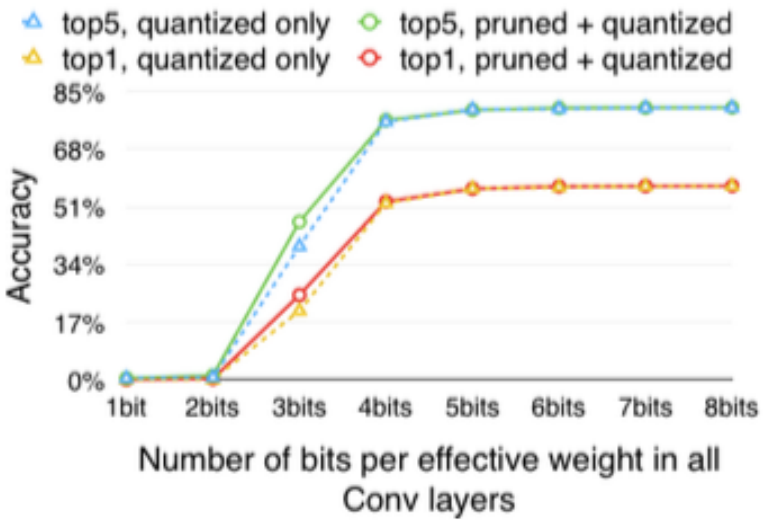
- P+Q is better

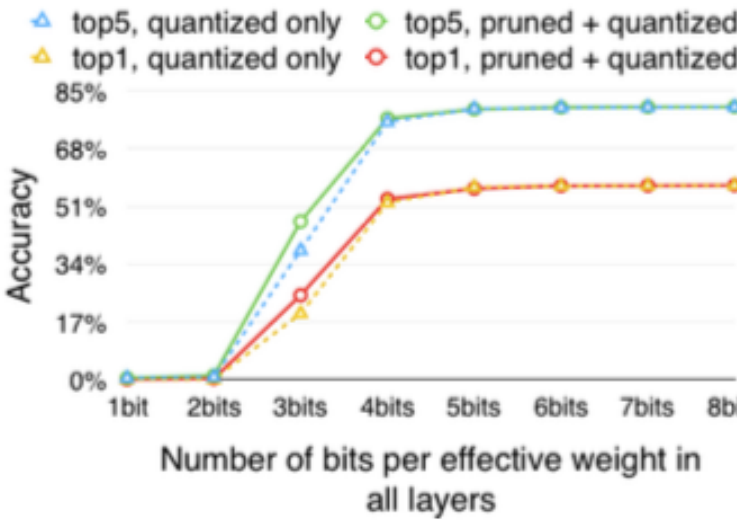


Pruning + Quantize

- pruning doesn't hurt quantiztion
- FC (3 bits)
- CONV(5 bits)







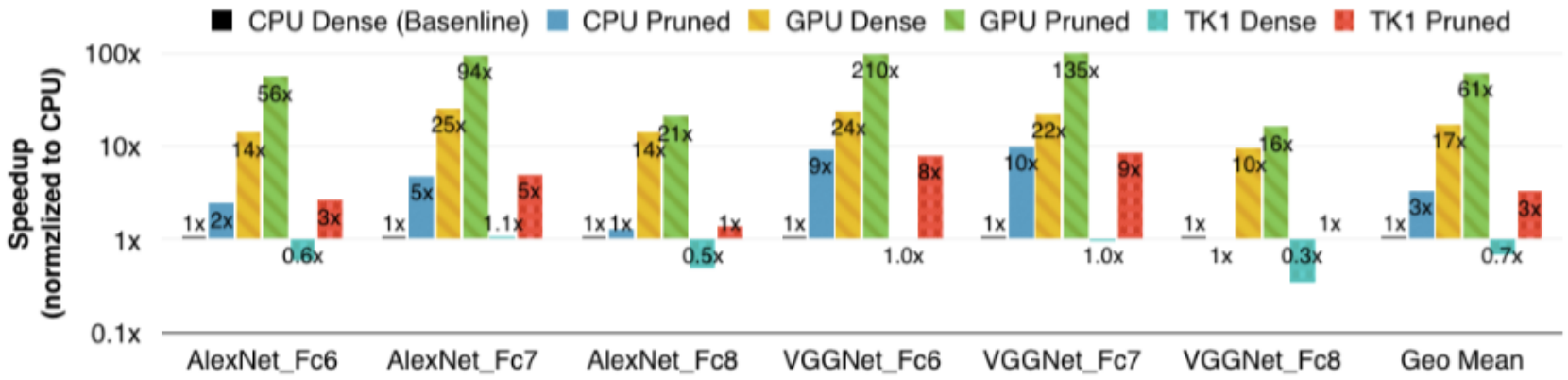


Figure 9: Compared with the original network, pruned network layer achieved 3× speedup on CPU, 3.5× on GPU and 4.2× on mobile GPU on average. Batch size = 1 targeting real time processing. Performance number normalized to CPU.

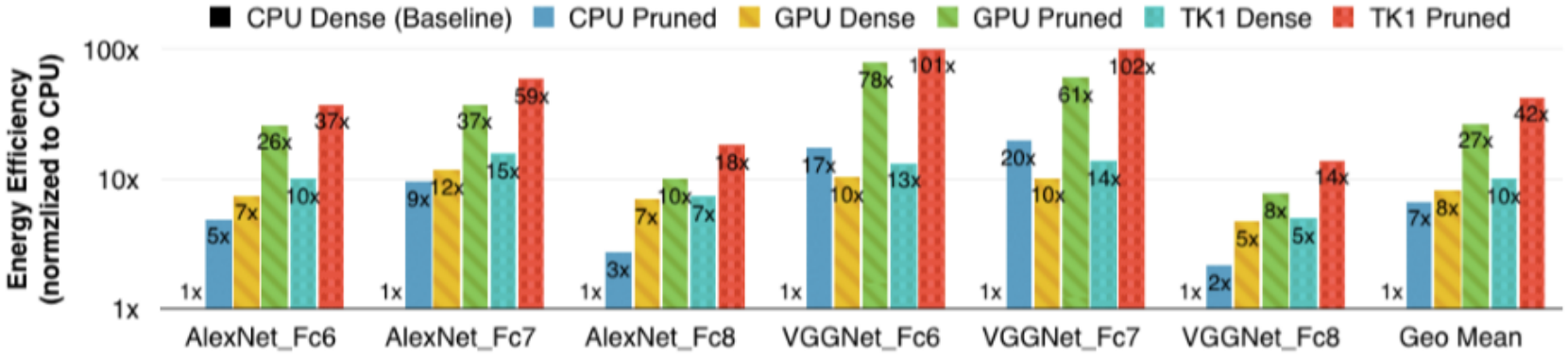
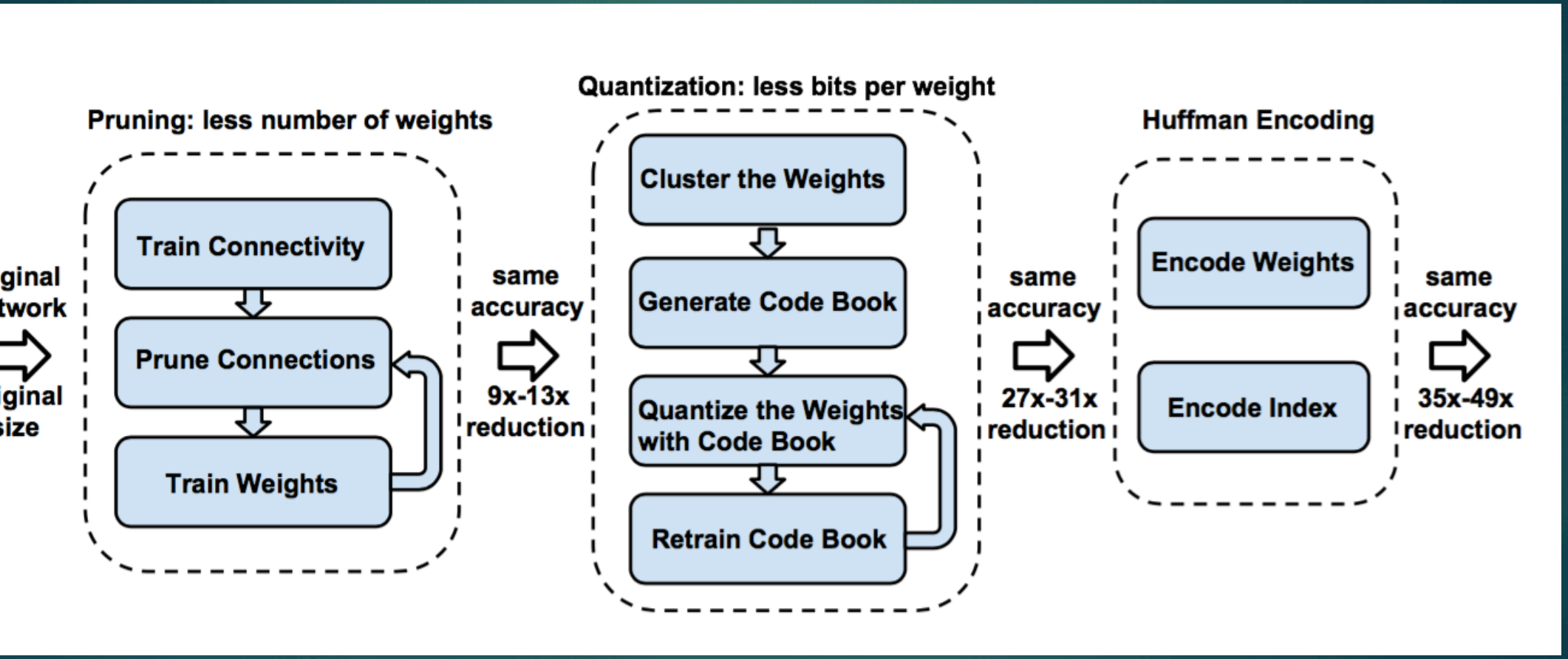


Figure 10: Compared with the original network, pruned network layer takes 7× less energy on CPU, 3.3× less on GPU and 4.2× less on mobile GPU on average. Batch size = 1 targeting real time processing. Energy number normalized to CPU.

Concept

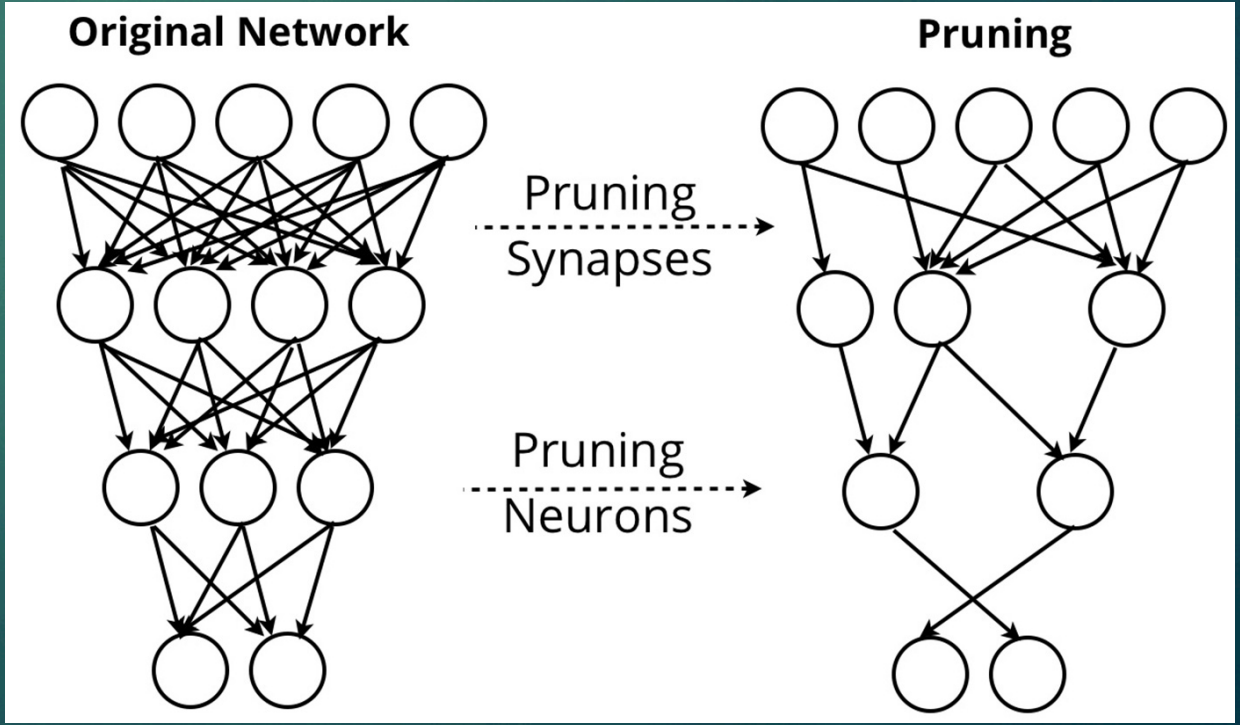

ginal
twork
ginal
size
Pruning: less number of weights
Train Connectivity
Prune Connections
Train Weights
same
accuracy
9x-13x
reduction
Quantization: less bits per weight
Cluster the Weights
Generate Code Book
Quantize the Weights
with Code Book
Retrain Code Book
same
accuracy
27x-31x
reduction
Huffman Encoding
Encode Weights
Encode Index
same
accuracy
35x-49x
reduction

Sparsifying Neural Network Connections for Face Recognition

YI SUN, XIAOGANG WANG, XIAOOU TANG

Abstract

- Pruning weight depends on correlation to output
- The correlation between two connected neurons are defined by the magnitude of the correlation between their neural activations.

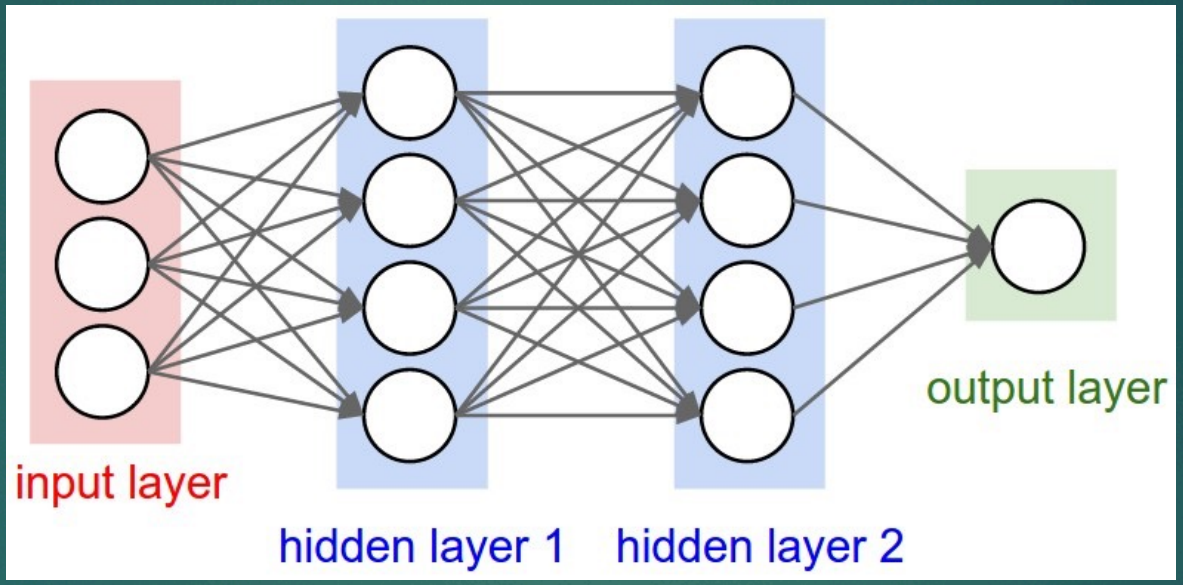


Sparsity connections

- Sample S* | W |
- Keep high correlated and drop weakly correlated connections.
- Keep a small portion weakly correlated connections is also helpful
- Fully-connected and convolutional

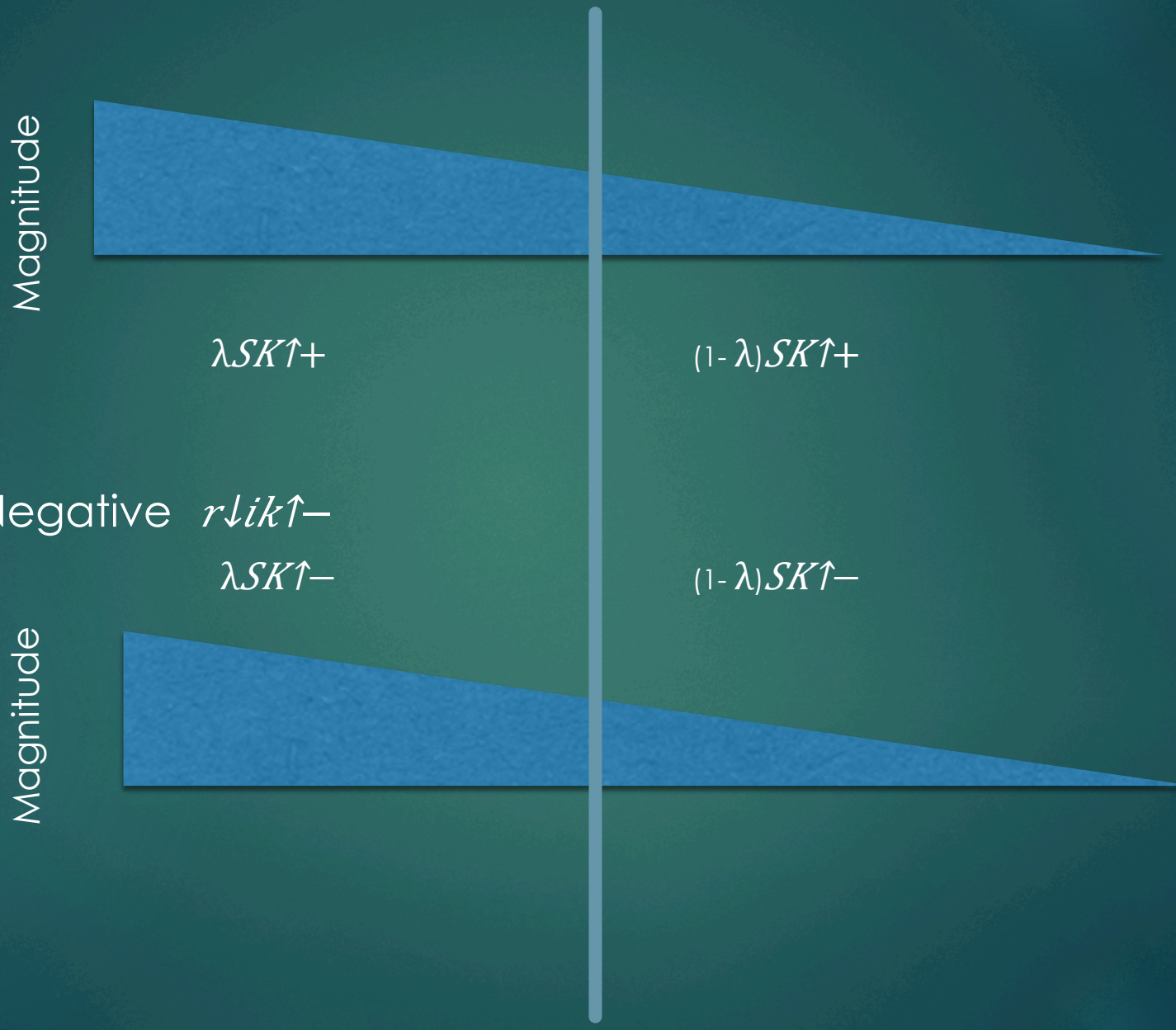
Sparsity connections

- fully- and locally-connected



$$r_{ik} = \frac{E[a_i - \mu_{a_i}][b_{ik} - \mu_{b_{ik}}]}{\sigma_{a_i}\sigma_{b_{ik}}}$$

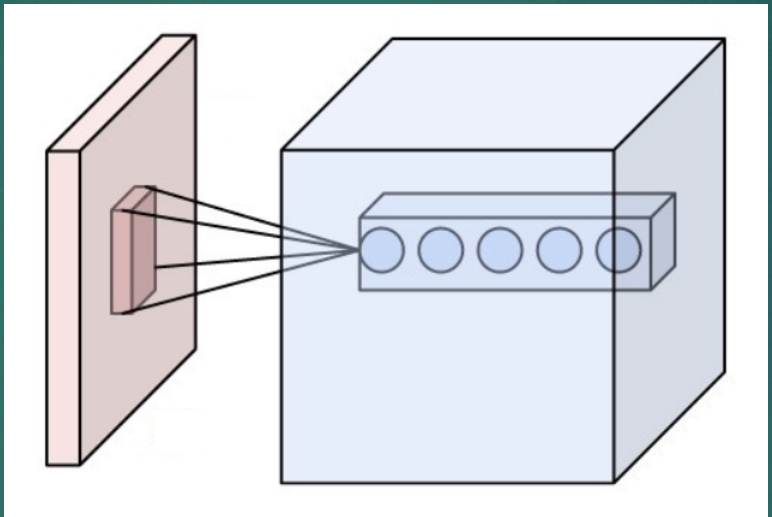
- Positive $r\downarrow lik\uparrow +$



- Negative $r\downarrow lik\uparrow -$

Sparsity connections

- Convolutional layer



$$r_{ik} \triangleq \sum_{m=1}^{M} \left| \frac{E[a_{im} - \mu_{a_{im}}][b_{mk} - \mu_{b_{mk}}]}{\sigma_{a_{im}} \sigma_{b_{mk}}} \right| .$$

Experiment

- Sparsity improves performance

sparse structure	accuracy	compression ratio
	0.9895	1
1/256-f	0.9898	0.96
1/256-f 1/128-5b	0.9923	0.76
1/256-f 1/128-5b 1/2-4b	**0.9930**	0.74
1/256-f 1/128-5b 1/2-5a	0.9922	0.43
1/256-f 1/128-5b 1/32-5a	0.9898	0.12

Experiment

- Correlation guided weight selection

sparse structure	accuracy	corr before	corr after
1/256-f	**0.9898**	0.147	0.494
1/256-f-r	0.9893	0.114	0.511
1/256-f-h	0.9893		
1/256-f 1/128-5b	**0.9923**	0.120	0.275
1/256-f 1/128-5b-r	0.9910	0.089	0.272
1/256-f 1/128-5b-h	0.9902		
1/256-f 1/128-5b 1/2-4b	**0.9930**	0.075	0.079
1/256-f 1/128-5b 1/2-4b-r	0.9922	0.067	0.073
1/256-f 1/128-5b 1/2-4b-h	0.9925		

Experiment

- Why do we need a denser network?

  sparse network is easier to get stuck at a local minimum

sparse structure	if pre-trained	accuracy	train error
1/256-f	yes	0.9898	0.0207
	no	0.9887	0.0229
1/256-f 1/128-5b	yes	0.9923	0.0302
	no	0.9845	0.0423
1/256-f 1/128-5b 1/2-4b	yes	0.9930	0.0299
	no	0.9833	0.0463

Experiment

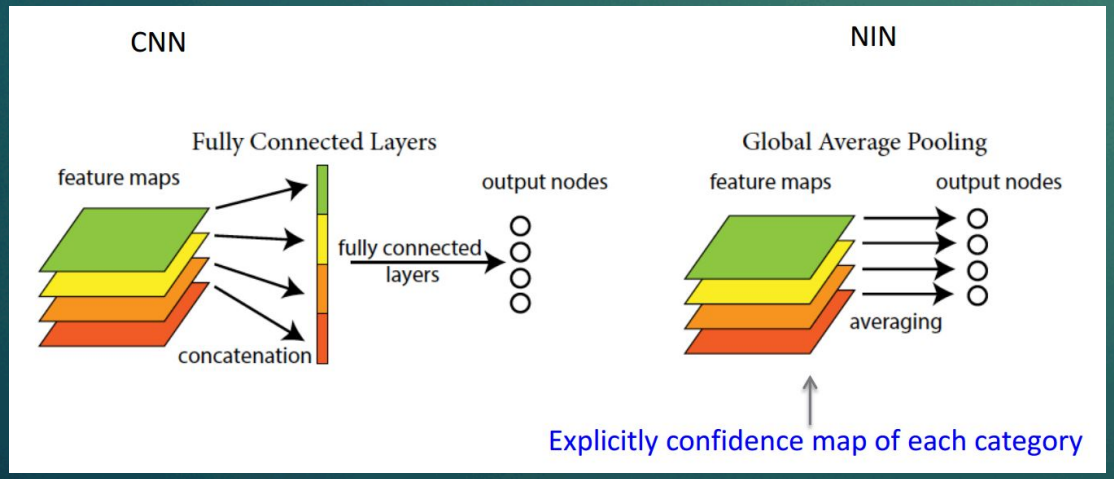
- Method comparison

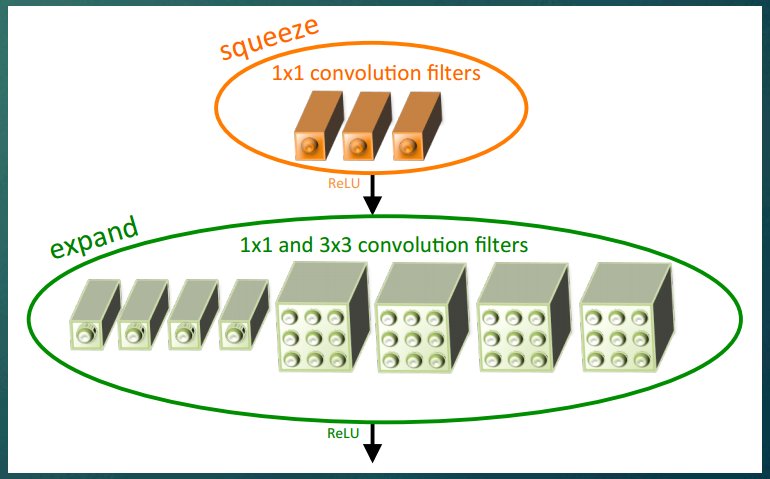
sparse structure	OBD	magnitude	BRP	correlation
1/256-f	**0.9905**	0.9888	0.9863	0.9898
1/128-5b	0.9903	0.9912	0.9902	**0.9923**
1/2-4b	0.9920	0.9925	0.9913	**0.9930**

Design compact layers

- Network in Network
  replace the fully connected layer with global average pooling

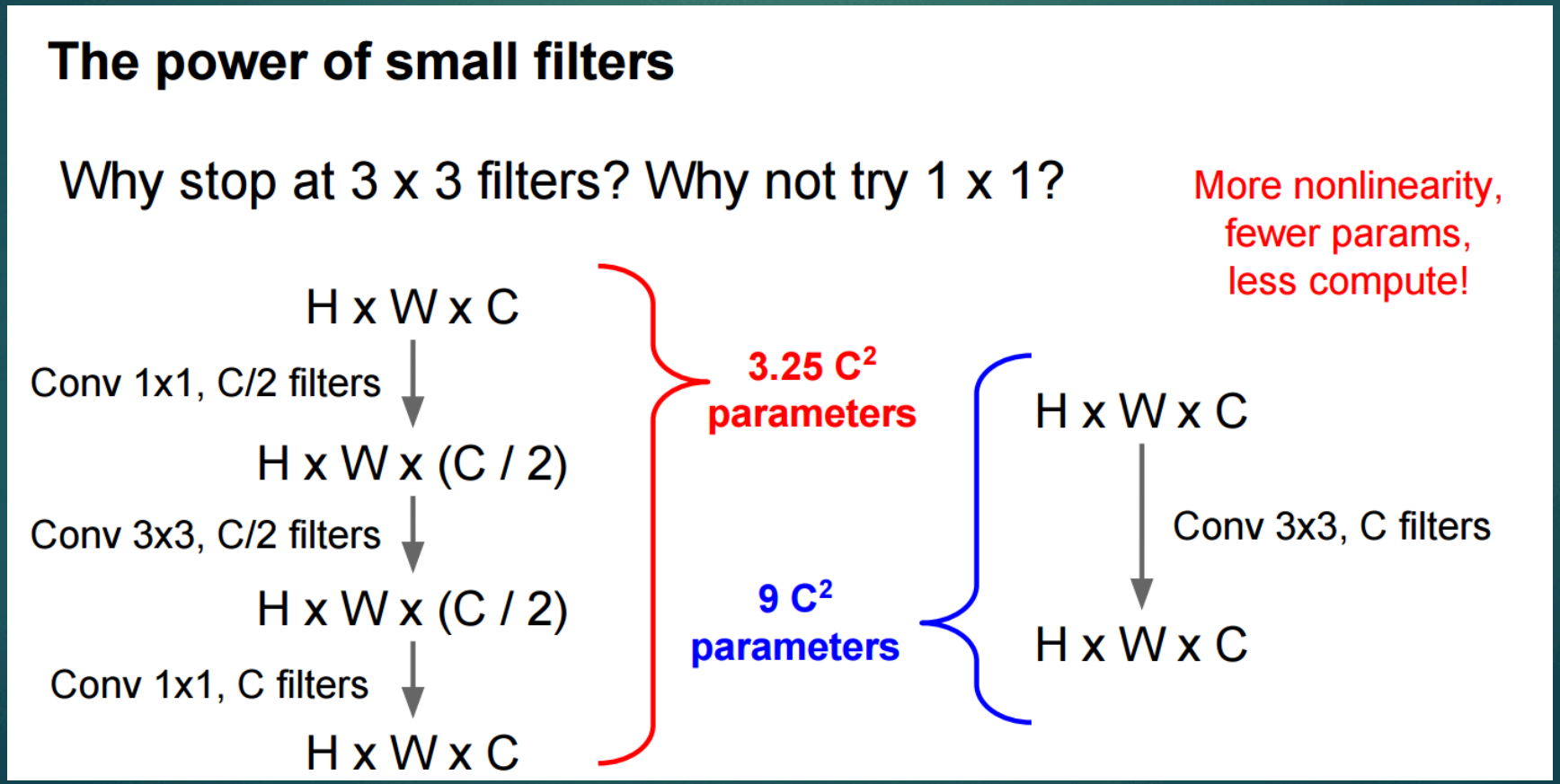
- SqueezeNet 50X model size save
  replace 3*3 convolution to 1*1 convolution reduce dimension





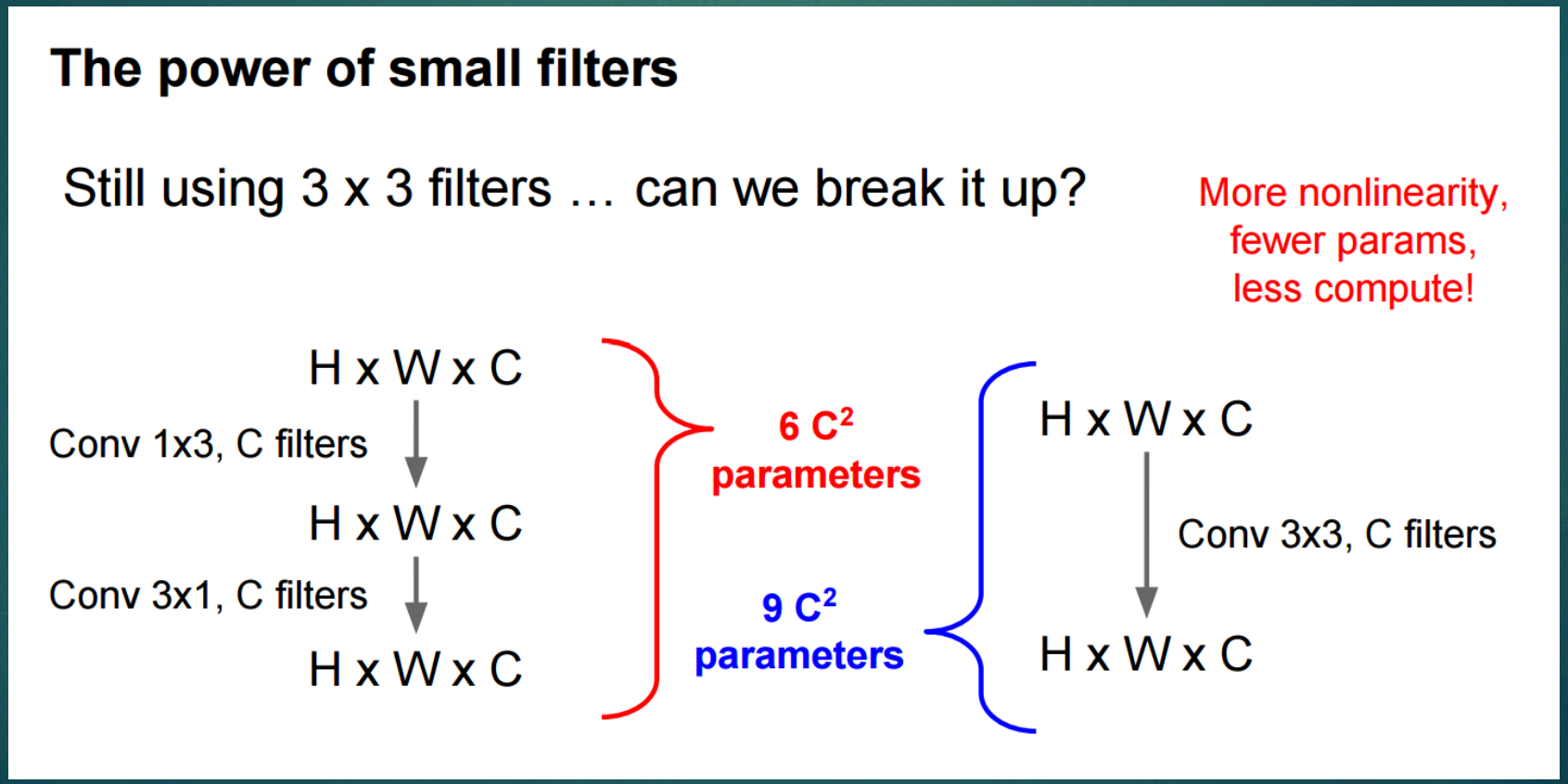
Design compact layers

- Reduce dimension



Design compact layers

- Decompose 3*3 convolution

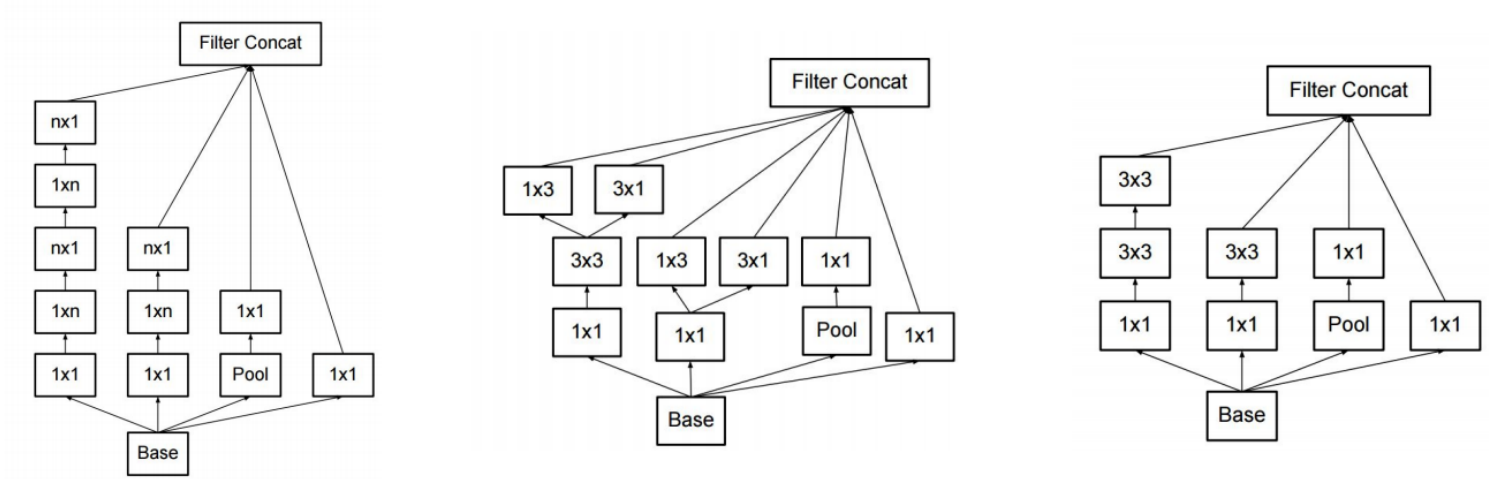


Design compact layers

- GoogLeNet

The power of small filters

Latest version of GoogLeNet incorporates all these ideas



Szegedy et al, “Rethinking the Inception Architecture for Computer Vision”

Quantizing parameters

- 32 bit “single” precision is typically used for CNNs for performance
- Nvidia support “half” precision

- BinaryNet

- Xnor-Net

- DoReFa-Net

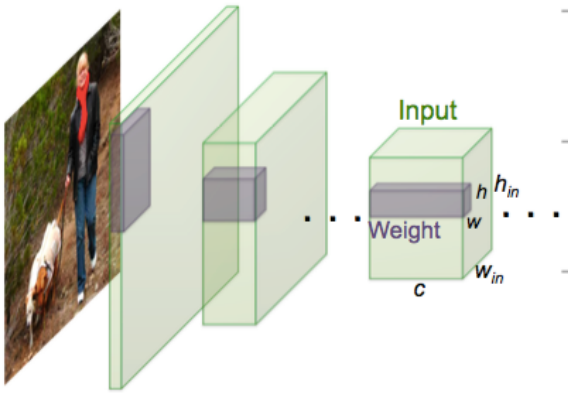
XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

MOHAMMAD RASTEGARI†, VICENTE ORDONEZ†, JOSEPH REDMON⇤, ALI FARHADI

ALLEN INSTITUTE FOR AI†, UNIVERSITY OF WASHINGTON⇤

Abstract

- Binary-Weight-Networks
  The filter are approximated with binary values. save 32x memory
- XNOR-Networks
  both the filters and the input to convolutional layers are binary.
  58x faster & save 32x memory



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Time Saving on CPU (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%53.8
BinaryWeight Binary Input **(XNOR-Net)**	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

Binary Convolutional Neural Network

$$\mathbf{I} = \mathcal{I}_{l(l=1,\ldots,L)}$$

$$\mathcal{W}_{lk(k=1,\ldots,K^l)}$$

$$\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \oplus \mathbf{B})\,\alpha$$

$$\langle \mathcal{I}, \mathcal{W}, * \rangle$$

$$\langle \mathcal{I}, \mathcal{B}, \mathcal{A}, \oplus \rangle$$

$$\mathcal{W}_{lk} \approx \mathcal{A}_{lk}\mathcal{B}_{lk}$$

Binary Convolutional Neural Network

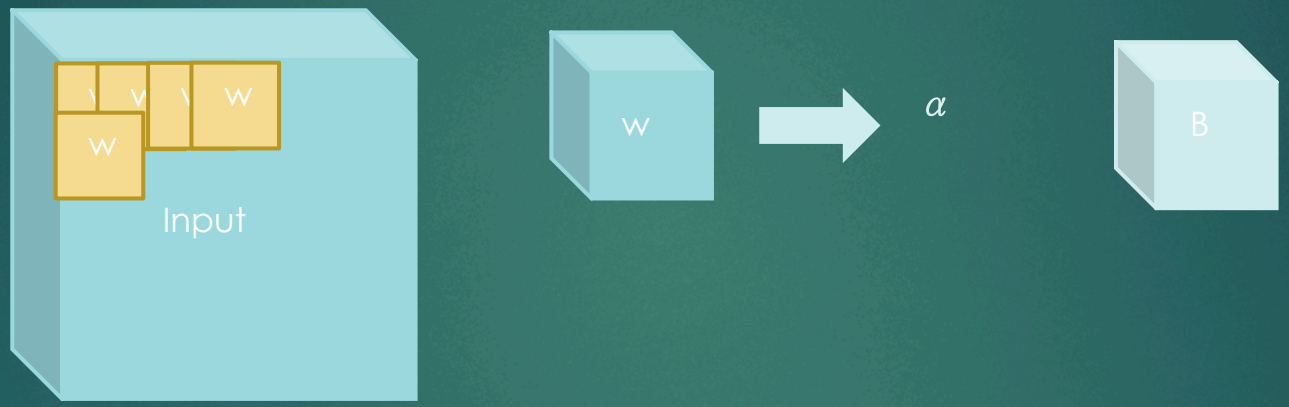
$$J(\mathbf{B}, \alpha) = \|\mathbf{W} - \alpha\mathbf{B}\|^2$$

$$\alpha^*, \mathbf{B}^* = \underset{\alpha, \mathbf{B}}{\operatorname{argmin}} J(\mathbf{B}, \alpha)$$

$$\mathbf{B}^* = \operatorname{sign}(\mathbf{W})$$

$$\alpha^* = \frac{\mathbf{W}^\mathsf{T} \operatorname{sign}(\mathbf{W})}{n} = \frac{\sum |\mathbf{W}_i|}{n} = \frac{1}{n}\|\mathbf{W}\|_{\ell 1}$$

Forward Convolution



Learning Algorithm

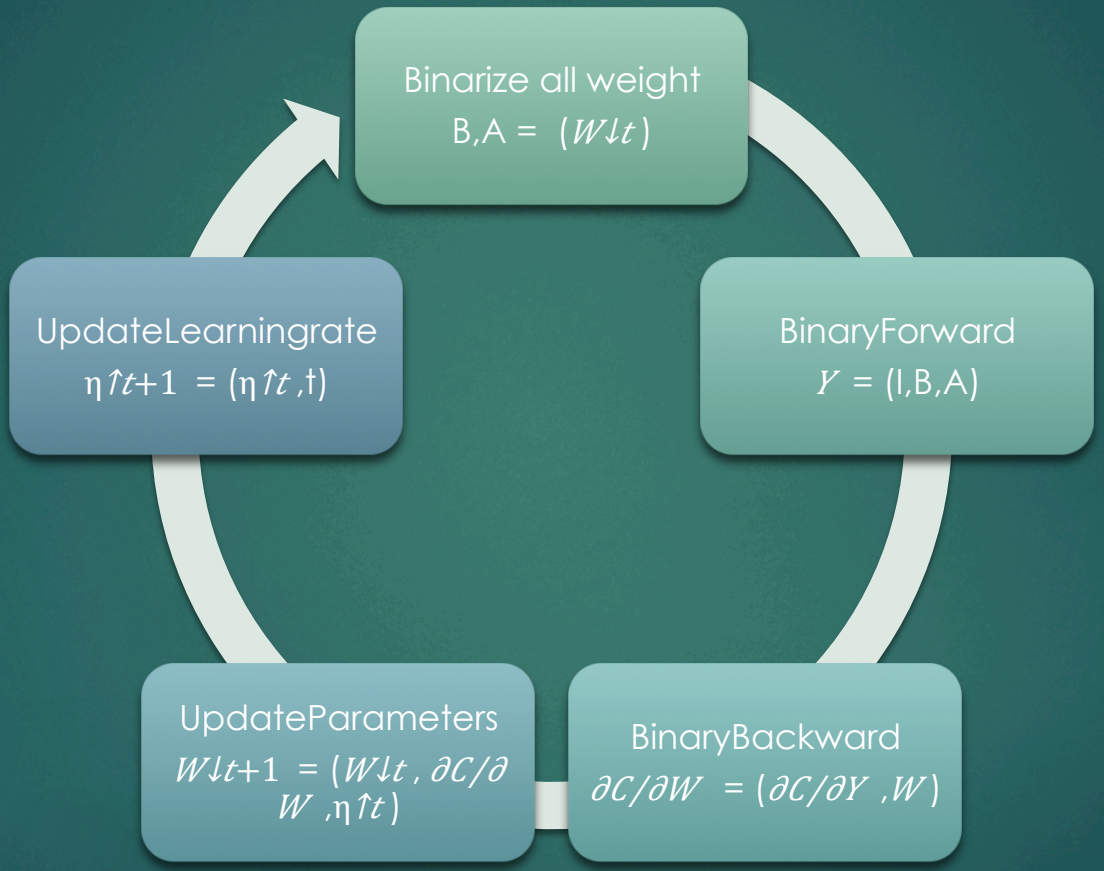
Algorithm 1 Training an L-layers CNN with binary weights:

Input: A minibatch of inputs and targets $(\mathbf{I}, \mathbf{Y})$, cost function $C(\mathbf{Y}, \hat{\mathbf{Y}})$, current weight $\mathcal{W}^t$ and current learning rate η^t.

Output: updated weight $\mathcal{W}^{t+1}$ and updated learning rate η^{t+1}.

Binarizing weight filters:
for $l = 1$ to L **do**
  for k^{th} filter in l^{th} layer **do**
    $\mathcal{A}_{lk} = \frac{1}{n}\|\mathcal{W}^t_{lk}\|_{\ell 1}$
    $\mathcal{B}_{lk} = \text{sign}(\mathcal{W}^t_{lk})$
    $\widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}$
$\hat{\mathbf{Y}} =$ **BinaryForward**$(\mathbf{I}, \mathcal{B}, \mathcal{A})$ // standard forward propagation except that convolutions are computed using equation 1 or 11
$\frac{\partial C}{\partial \widetilde{\mathcal{W}}} =$ **BinaryBackward**$(\frac{\partial C}{\partial \hat{\mathbf{Y}}}, \widetilde{\mathcal{W}})$ // standard backward propagation except that gradients are computed using $\widetilde{\mathcal{W}}$ instead of $\mathcal{W}^t$
$\mathcal{W}^{t+1} =$ **UpdateParameters**$(\mathcal{W}^t, \frac{\partial C}{\partial \widetilde{\mathcal{W}}}, \eta_t)$ // Any update rules (*e.g.*,SGD or ADAM)
$\eta^{t+1} =$ **UpdateLearningrate**(η^t, t) // Any learning rate scheduling function

Learning Algorithm



XNOR-Networks

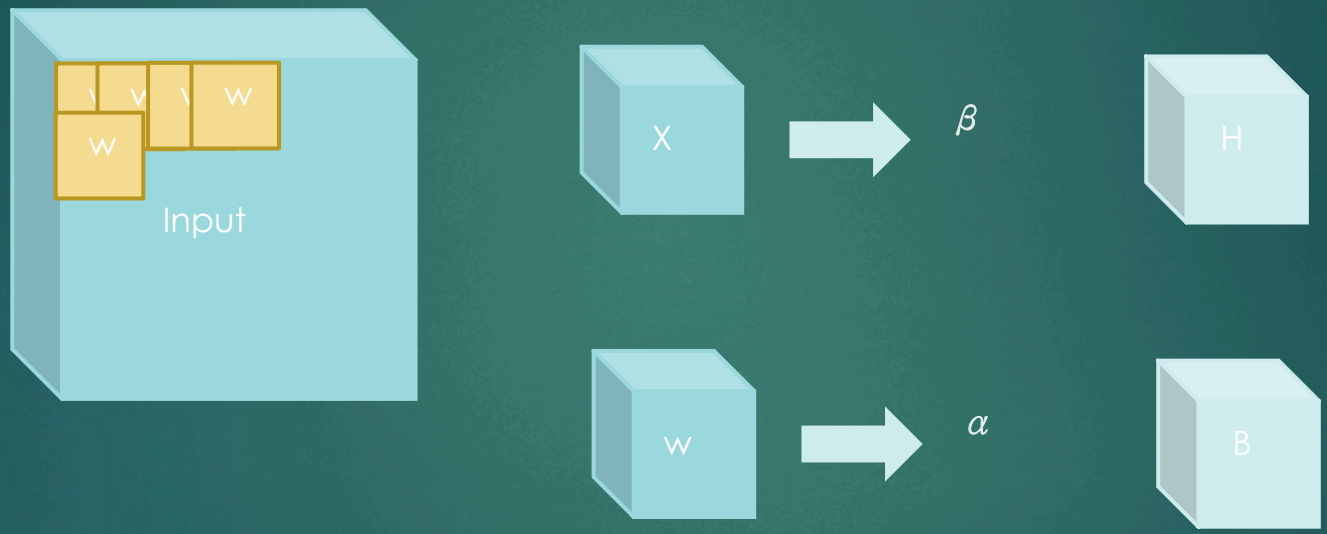
$$\mathbf{X}^\mathsf{T}\mathbf{W} \approx \beta\mathbf{H}^\mathsf{T}\alpha\mathbf{B}$$

$$\alpha^*, \mathbf{B}^*, \beta^*, \mathbf{H}* = \underset{\alpha,\mathbf{B},\beta,\mathbf{H}}{\operatorname{argmin}} \|\mathbf{X}^\mathsf{T}\mathbf{W} - \beta\alpha\mathbf{H}^\mathsf{T}\mathbf{B}\|$$

$$\mathbf{C}^* = \operatorname{sign}(\mathbf{Y}) = \operatorname{sign}(\mathbf{X}^\mathsf{T}) \operatorname{sign}(\mathbf{W}) = \mathbf{H}^{*\mathsf{T}}\mathbf{B}^*$$

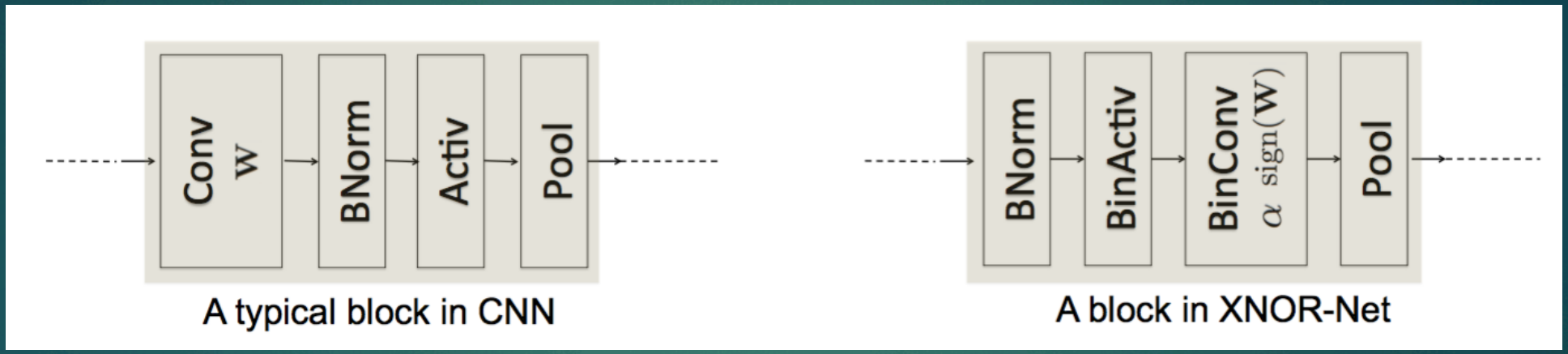
$$\gamma^* = \frac{\sum|\mathbf{Y}_i|}{n} = \frac{\sum|\mathbf{X}_i||\mathbf{W}_i|}{n} \approx \left(\frac{1}{n}\|\mathbf{X}\|_{\ell 1}\right)\left(\frac{1}{n}\|\mathbf{W}\|_{\ell 1}\right) = \beta^*\alpha^*$$

Forward Convolution



Layer order

- We normalize the input before binarization



Efficiency Analysis

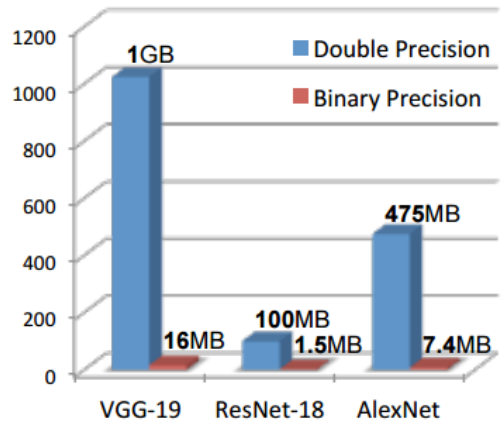
- we can perform 64 binary operations in one clock of CPU

$$\mathbf{I} * \mathbf{W} \approx (\text{sign}(\mathbf{I}) \circledast \text{sign}(\mathbf{W})) \odot \mathbf{K}\alpha$$

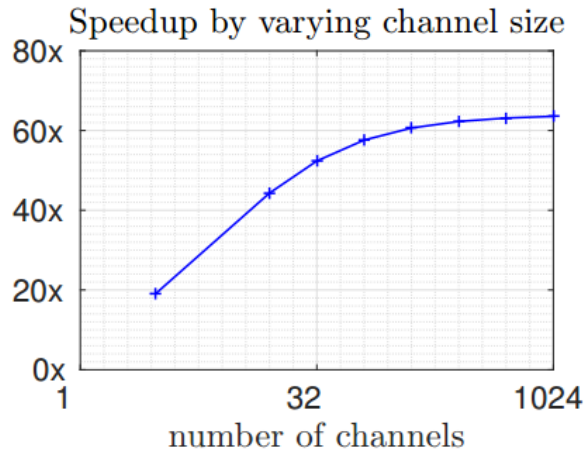
$$S = \frac{cN_{\mathbf{W}}N_{\mathbf{I}}}{\frac{1}{64}cN_{\mathbf{W}}N_{\mathbf{I}} + N_{\mathbf{I}}} = \frac{64cN_{\mathbf{W}}}{cN_{\mathbf{W}} + 64}$$

Efficiency Analysis

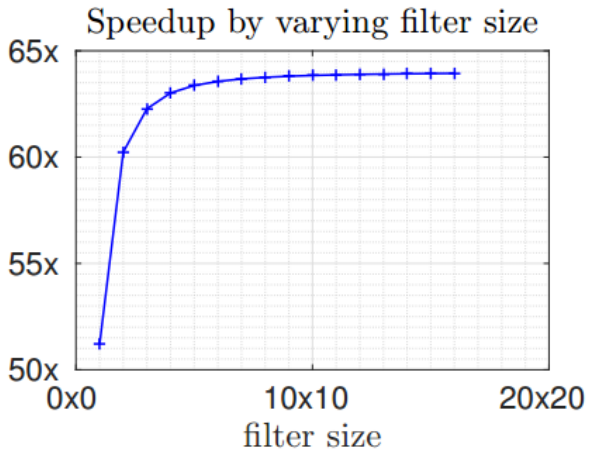
- c = 256, $n\downarrow I$ = 14↑2 , $n\downarrow W$ = 3↑2



(a)



(b)



(c)

- Avoid binarization at the first and last XNOR-Net

Discussion

Binary-Weight-Network		
Strategy for computing α	top-1	top-5
Using equation 6	56.8	79.4
Using a separate layer	46.2	69.5

XNOR-Network		
Block Structure	top-1	top-5
C-B-A-P	30.3	57.5
B-A-C-P	44.2	69.2

Accuracy Analysis

- AlexNet compare with other network

Classification Accuracy(%)									
Binary-Weight				Binary-Input-Binary-Weight				Full-Precision	
BWN		BC[11]		XNOR-Net		BNN[11]		AlexNet[1]	
Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
56.8	**79.4**	35.4	61.0	**44.2**	**69.2**	27.9	50.42	56.6	80.2

Concept

- Use Xnor to implement convolution speedup the inference procedure.
- First binarized network testing on imagenet but falling about 10% accuraccy.
- Introduce new strategy to compute α scalar and layer order.

DoReFa-Net

國立清華大學

黃柏瑜

Abstract

- Generalize the binarized neural network. Let CNN has arbitrary bitwise in weights, activations and gradients
- Using bit convolution kernels
- Explore the configureation space of bitwidth for weights, activation and gradients

Bit Convolution

$$\mathbf{x} \cdot \mathbf{y} = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} 2^{m+k} \ \text{bitcount}[\text{and}(c_m(\mathbf{x}), c_k(\mathbf{y}))],$$

$$c_m(\mathbf{x})_i, c_k(\mathbf{y})_i \in \{0, 1\} \, \forall i, \ m, \ k.$$

Quantization

- quantize$_k$ $r_i \in [0,1]$

$$\textbf{Forward: } r_o = \frac{1}{2^k - 1} \text{round}((2^k - 1) r_i)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}.$$

- Weight – Normal distribution, use non-uniform distribution to quantize

$$\textbf{Forward: } r_o = f_\omega^k(r_i) = 2\,\text{quantize}_k\left(\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}\right) - 1.$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i}\frac{\partial c}{\partial r_o}$$

Quantization

- Activation

$$f_{\alpha}^{k}(r) = \mathrm{quantize_k}(r).$$

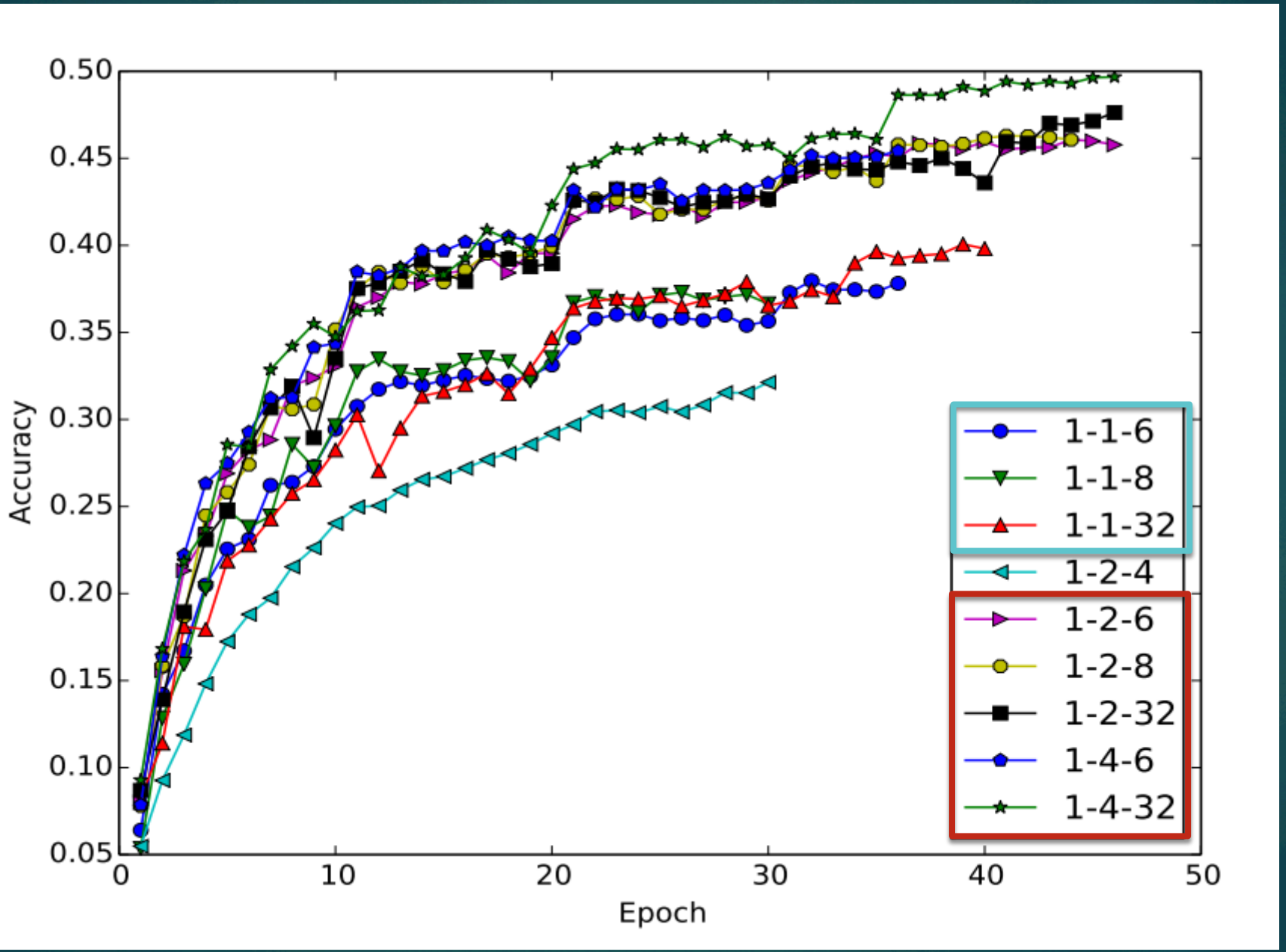
- Gradient – stochastic quantization is necesarry

$$f_{\gamma}^{k}(\mathrm{d}r) = 2\max_0(|\mathrm{d}r|)\left[\mathrm{quantize}_k\left[\frac{\mathrm{d}r}{2\max_0(|\mathrm{d}r|)} + \frac{1}{2} + N(k)\right] - \frac{1}{2}\right)$$

SVHN

W	A	G	Training Complexity	Inference Complexity	Storage Relative Size	Model A Accuracy	Model B Accuracy	Model C Accuracy	Model D Accuracy
1	1	2	3	1	1	0.934	0.924	0.910	0.803
1	1	4	5	1	1	0.968	0.961	0.916	0.846
1	1	8	9	1	1	0.970	0.962	0.902	0.828
1	1	32	-	-	1	0.971	0.963	0.921	0.841
1	2	2	4	2	1	0.909	0.930	0.900	0.808
1	2	3	5	2	1	0.968	0.964	0.934	0.878
1	2	4	6	2	1	0.975	0.969	0.939	0.878
2	1	2	6	2	2	0.927	0.928	0.909	0.846
2	1	4	10	2	2	0.969	0.957	0.904	0.827
1	2	8	10	2	1	0.975	0.971	0.946	0.866
1	2	32	-	-	1	0.976	0.970	0.950	0.865
1	3	3	6	3	1	0.968	0.964	0.946	0.887
1	3	4	7	3	1	0.974	0.974	0.959	0.897
1	3	6	9	3	1	0.977	0.974	0.949	0.916
1	4	2	6	4	1	0.815	0.898	0.911	0.868
1	4	4	8	4	1	0.975	0.974	0.962	0.915
1	4	8	12	4	1	0.977	0.975	0.955	0.895
2	2	2	8	4	1	0.900	0.919	0.856	0.842
8	8	8	-	-	8			0.970	0.955
32	32	32	-	-	32	0.975	0.975	0.972	0.950

W	A	G	Training Complexity	Inference Complexity	Storage Relative Size	AlexNet Accuracy
1	1	6	7	1	1	0.395
1	1	8	9	1	1	0.395
1	1	32	-	1	1	0.279 (BNN)
1	1	32	-	1	1	0.442 (XNOR-Net)
1	1	32	-	1	1	0.401
1	1	32	-	1	1	0.436 (initialized)
1	2	6	8	2	1	0.461
1	2	8	10	2	1	0.463
1	2	32	-	2	1	0.477
1	2	32	-	2	1	0.498 (initialized)
1	3	6	9	3	1	0.471
1	3	32	-	3	1	0.484
1	4	6	-	4	1	0.482
1	4	32	-	4	1	0.503
1	4	32	-	4	1	0.530 (initialized)
8	8	8	-	-	8	0.530
32	32	32	-	-	32	0.559


0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Accuracy
0
10
20
30
40
50
Epoch
1-1-6
1-1-8
1-1-32
1-2-4
1-2-6
1-2-8
1-2-32
1-4-6
1-4-32

Concept

- First quantize the graidient down to 6 bits and get comparable result.
- Train time, inference time, model size, accuracy are trade-off. There is no definite conclusion which combination of (W,A,G) should choose.
- Sensitivity of bit size : G>A>W

Conclusion

- Compress test stage only
  shallow network
  prune (popular)

- Compress train & test stage
  compact layer
  quantization

Reference

- CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.stanford.edu/
- SqueezeNet : https://arxiv.org/pdf/1602.07360v3.pdf
- Network in Network : https://arxiv.org/pdf/1312.4400v3.pdf
- Xnor-Net : http://arxiv.org/pdf/1603.05279v4.pdf
- DoReFa-Net : https://arxiv.org/pdf/1606.06160v2.pdf
- Deep Compression : https://arxiv.org/pdf/1510.00149v5.pdf

Reference

- Convolutional Neural Networks using Logarithmic Data Representation https://arxiv.org/pdf/1603.01025.pdf
- https://github.com/andyhahaha/Convolutional-Neural-Network-Compression-Survey/blob/master/README.md