NIPS

Neural Information Processing Systems

Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding

Song Han[1], Huizi Mao[2], William J. Dally[1,3]
[1] Stanford University [2] Tsinghua University [3] NVIDIA
{songhan, dally}@stanford.edu mhz12@mails.tsinghua.edu.cn

Motivation: Make DNN Smaller

ML Researchers are Happy

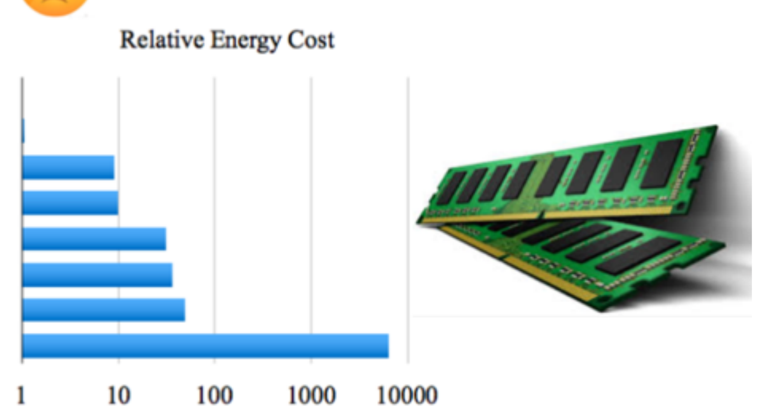
GoogLeNet:	5M parameters
AlexNet:	60M parameters
Deepface:	120M parameters
VGG-16:	130M parameters

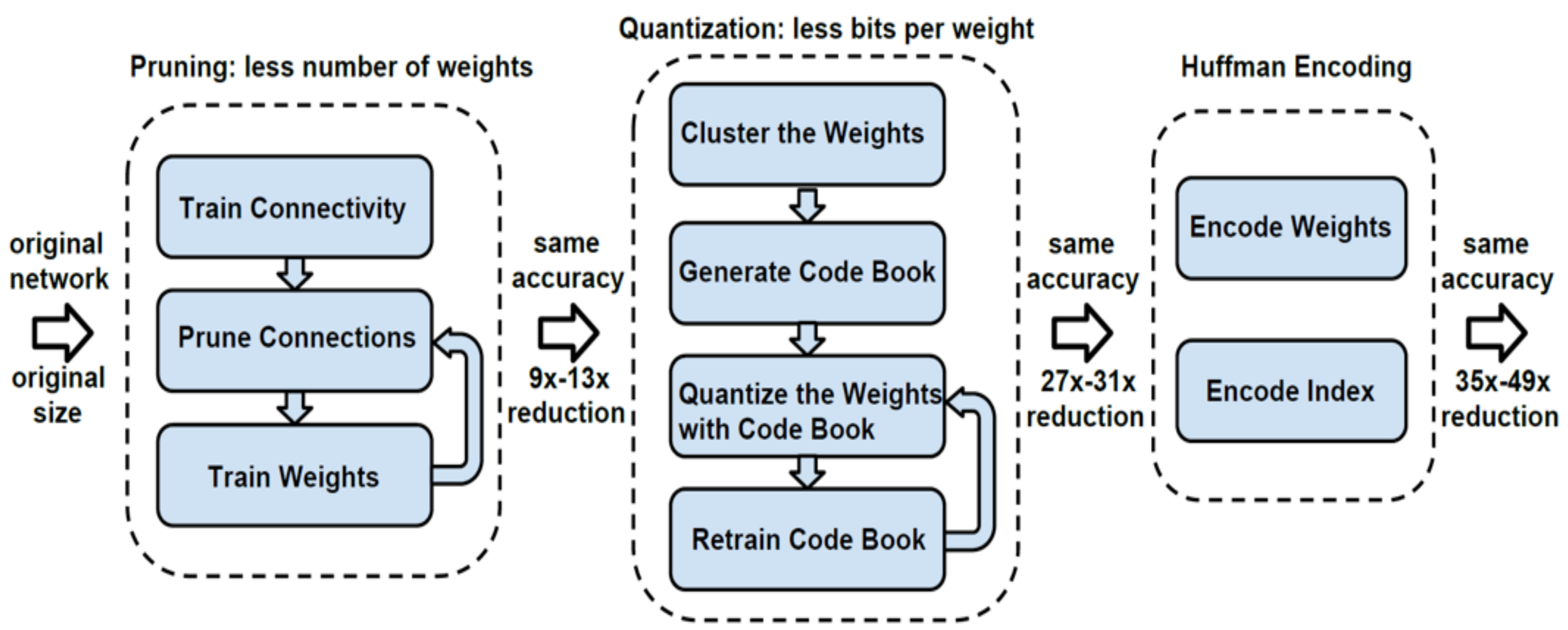
Software Engineer Suffers

"Apps above 100 MB will not download until you connect to Wi-Fi "

Hardware Engineer Suffers

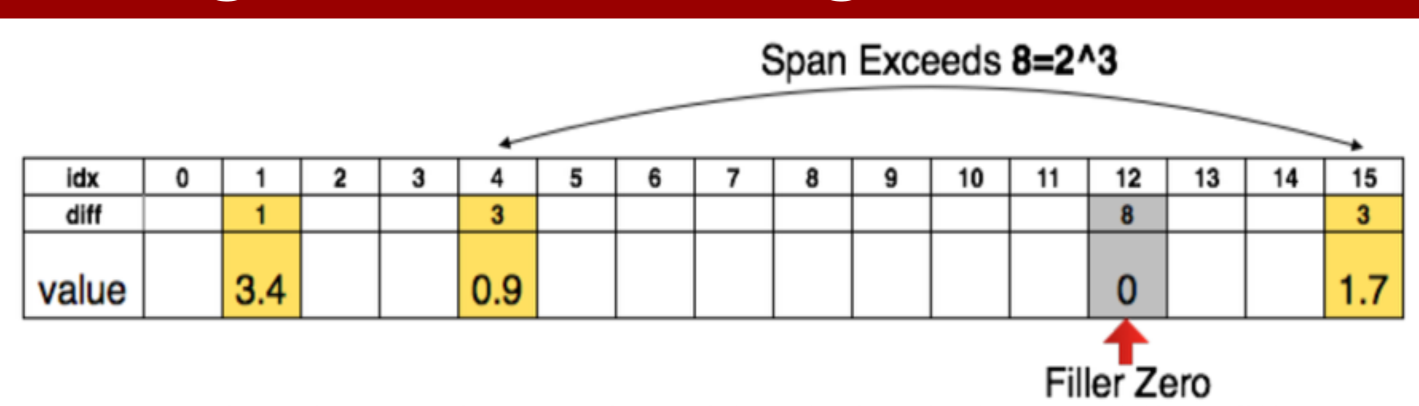
Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**



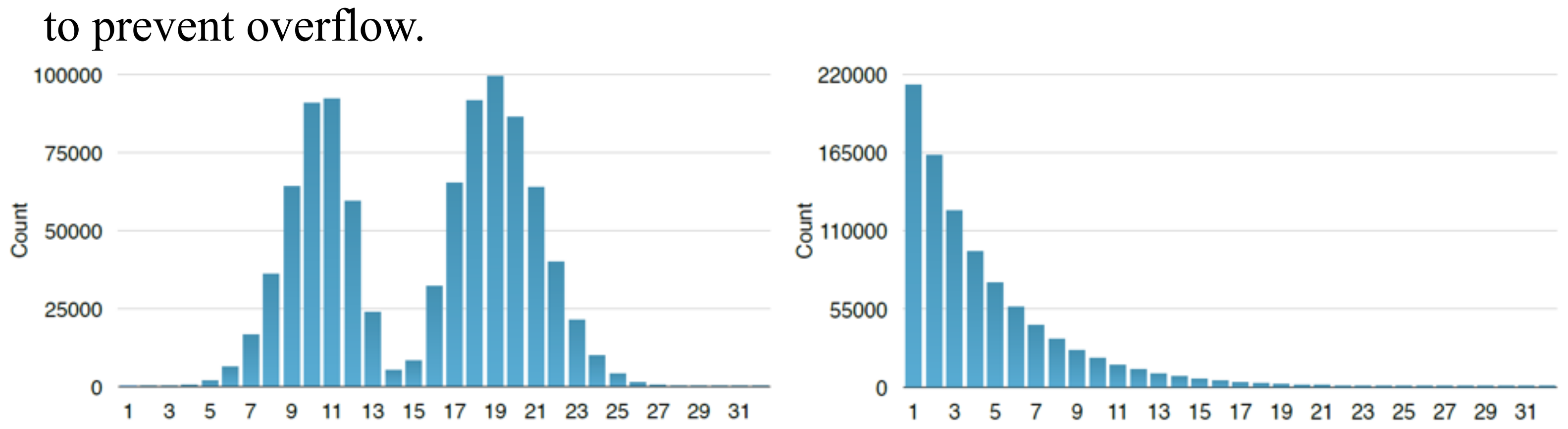


Deep Compression is a three stage compression pipeline: pruning, quantization and Huffman coding. Pruning reduces the number of weights by 10x, quantization further improves the compression rate between 27x and 31x. Huffman coding gives more compression: between 35x and 49x. The compression rate already included the meta-data for sparse representation. *Deep Compression* doesn't incur loss of accuracy.

Pruning, Relative Indexing & Huffman Coding

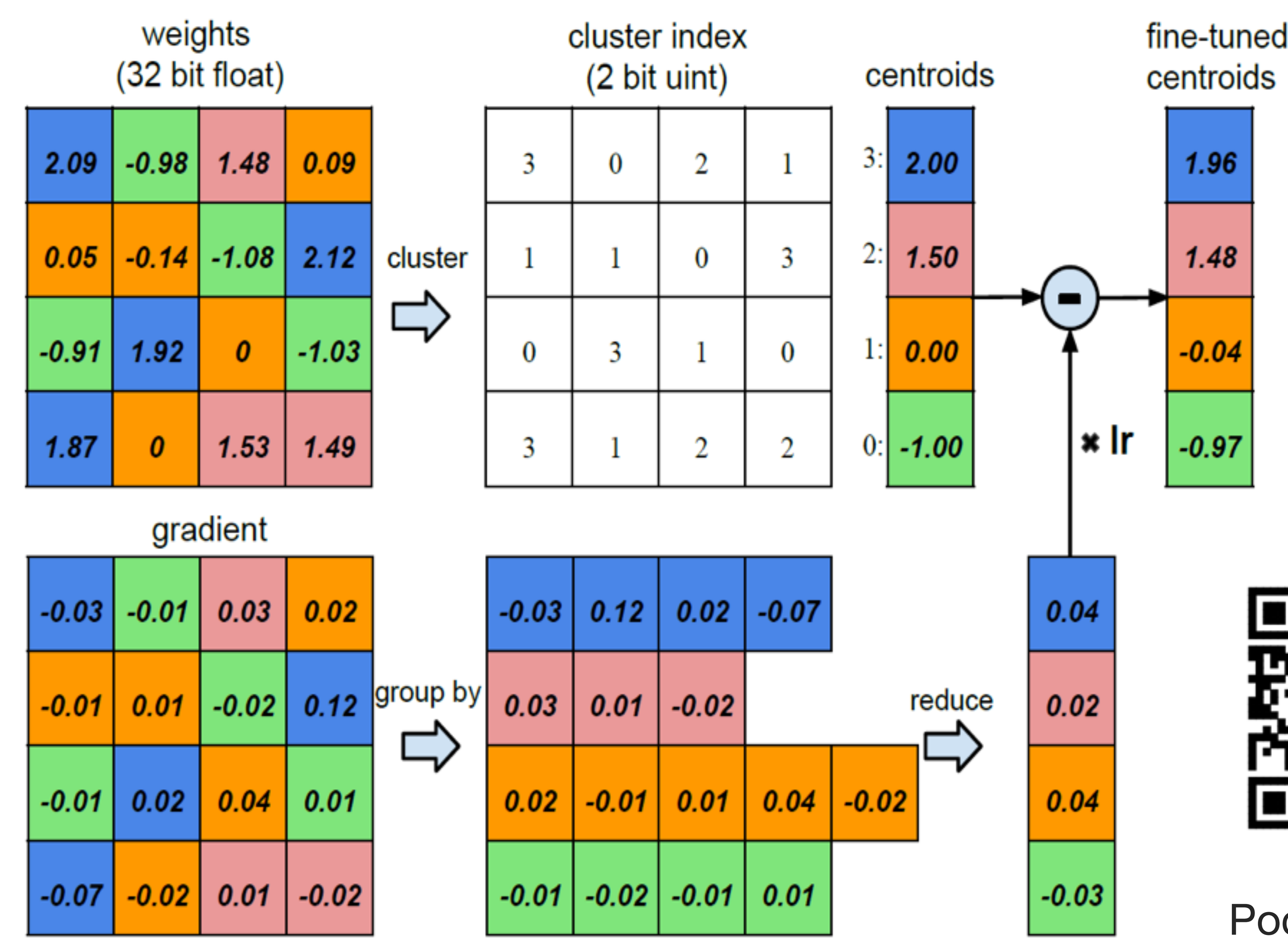


Representing the matrix sparsity with relative index. Padding filler zero to prevent overflow.



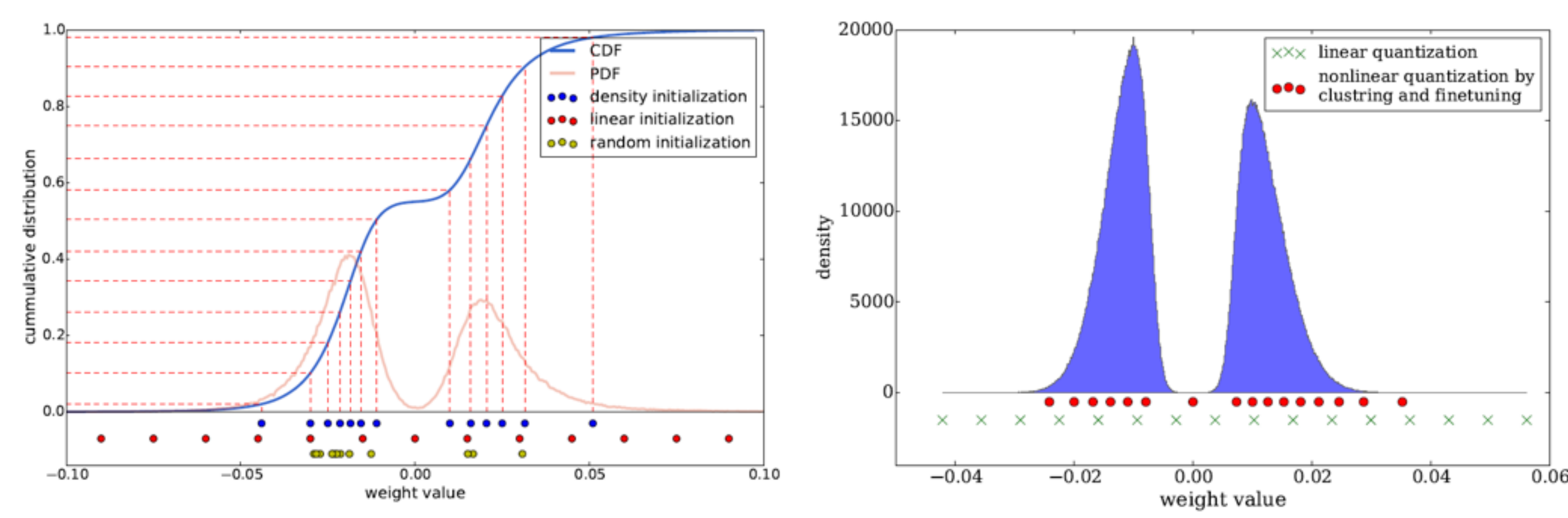
Weight / Index distribution is biased. Represent frequent occurring weight with less number of bits, less frequent occurring weight with more bits.

Weight Sharing and Quantization

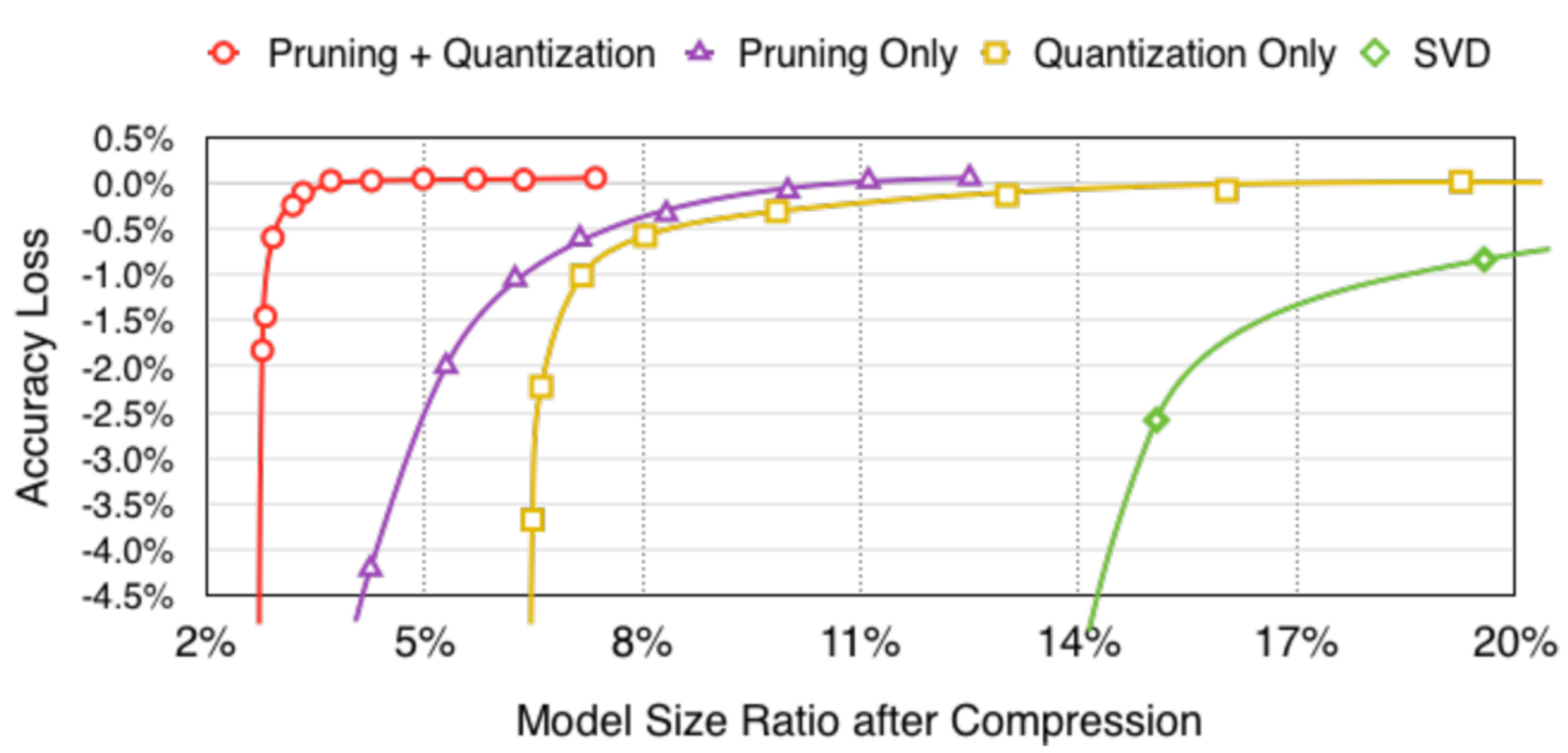


Demo: Pocket AlexNet

Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom).



Left: Three different methods for centroids initialization. Right: Distribution of weights (blue) and distribution of codebook before (green cross) and after fine-tuning (red dot).

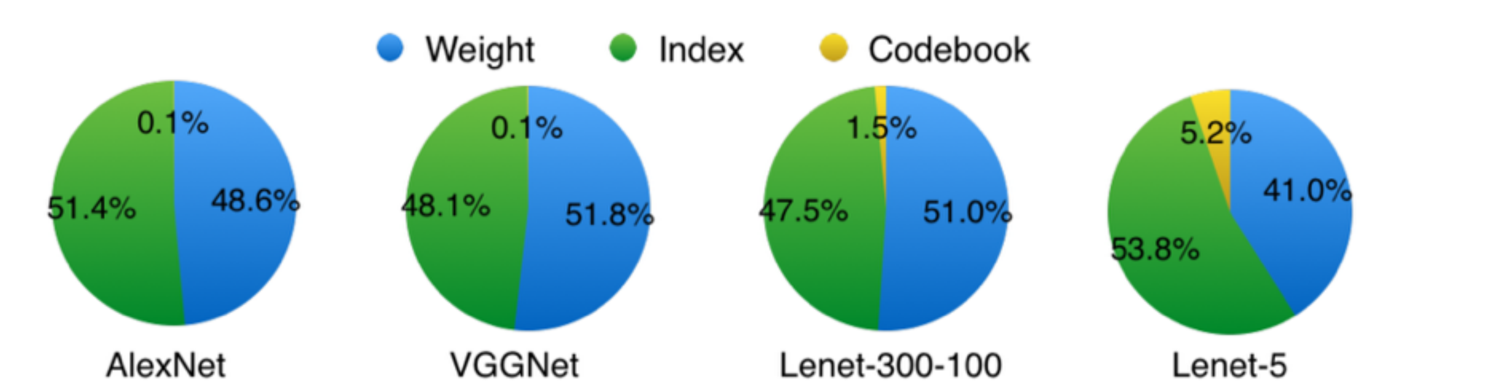


Accuracy v.s. compression rate under different compression methods. Pruning and quantization works best when combined.

Results

We pruned, quantized, and Huffman encoded four networks: Lenet-5, Lenet-300-100 on MNIST and AlexNet, VGG-16 on ImageNet. The compression pipeline saves network storage by 35× to 49× across different networks without loss of accuracy. The total size of AlexNet decreased from 240MB to 6.9MB, which is small enough to be put into on-chip SRAM, eliminating the need to store the model in energy-consuming DRAM memory.

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×



Storage ratio of weight, index and codebook.

Compression statistics for AlexNet. P: pruning, Q: quantization, H:Huffman coding.

Layer	#Weights	Weights% (P)	Weight 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	35K	84%	8	6.3	4	1.2	32.6%	20.53%
conv2	307K	38%	8	5.5	4	2.3	14.5%	9.43%
conv3	885K	35%	8	5.1	4	2.6	13.1%	8.44%
conv4	663K	37%	8	5.2	4	2.5	14.1%	9.11%
conv5	442K	37%	8	5.6	4	2.5	14.0%	9.43%
fc6	38M	9%	5	3.9	4	3.2	3.0%	2.39%
fc7	17M	9%	5	3.6	4	3.7	3.0%	2.46%
fc8	4M	25%	5	4	4	3.2	7.3%	5.85%
Total	61M	11%(9×)	5.4	4	4	3.2	3.7% (**27**×)	2.88% (**35**×)

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**×)