Model Compression: Pruning and Quantization

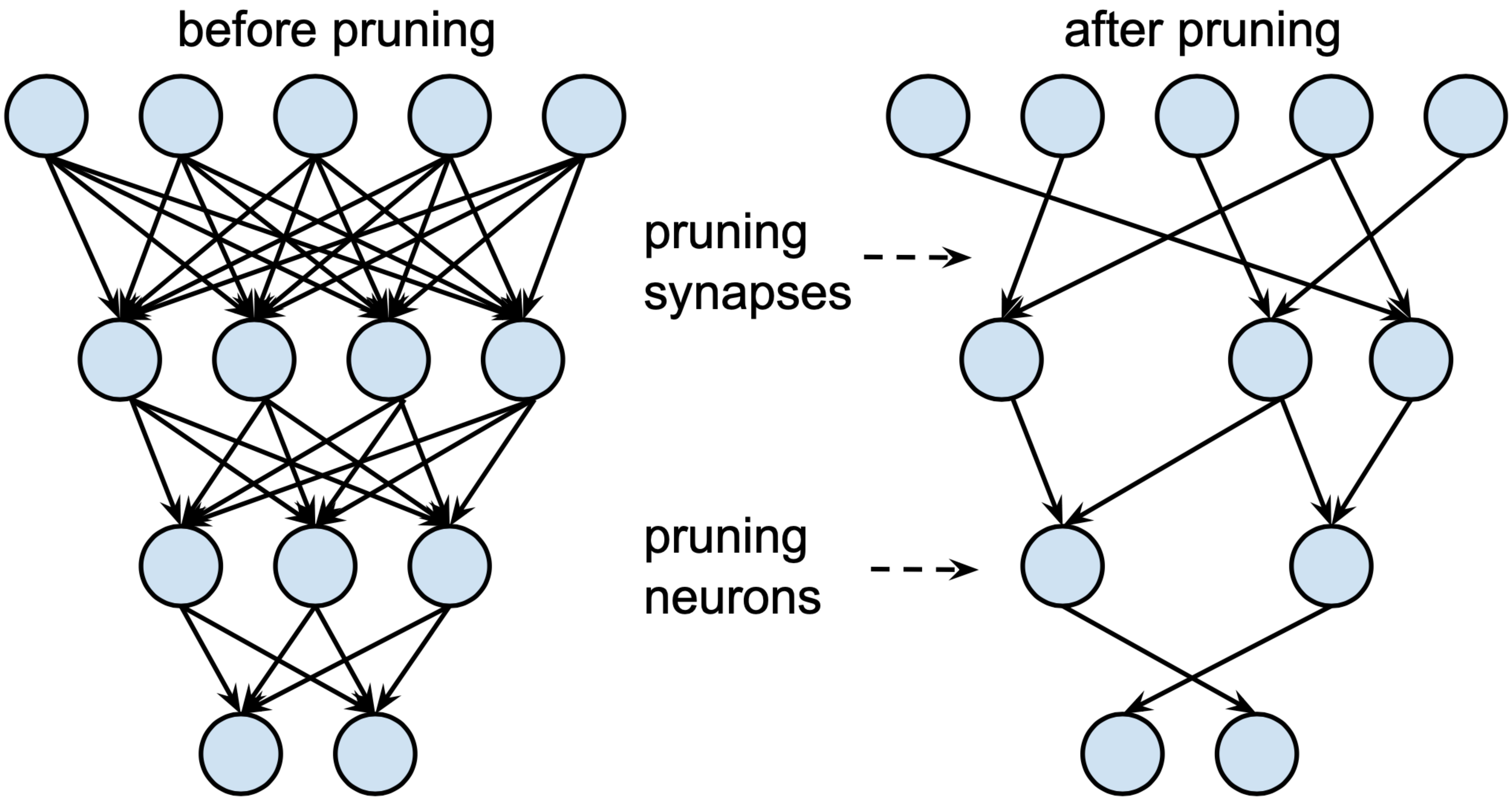
Pooyan Jamshidi
UofSC

The slides are mainly based on a NeurIPS'15 tutorial by William Dally

Reducing Size of Network Reduces Work and Storage

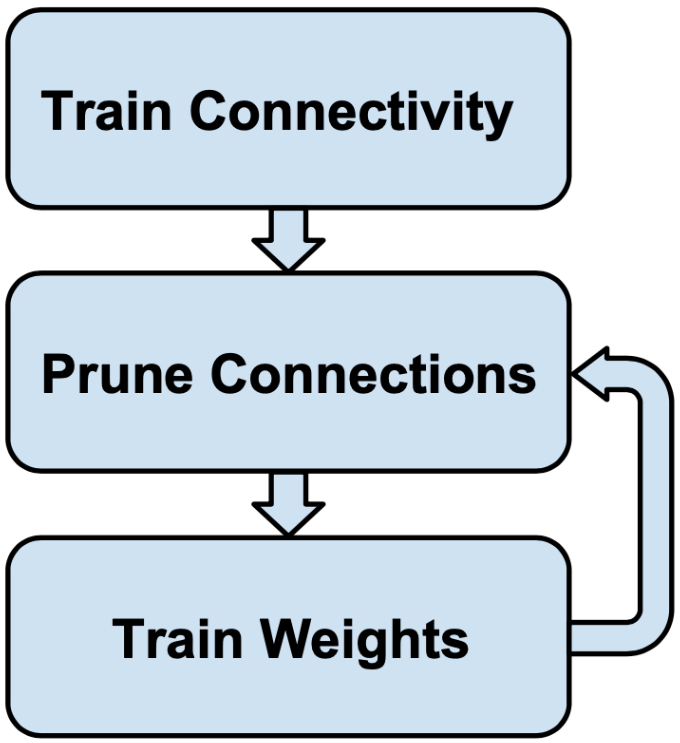
Prune Unneeded Connections

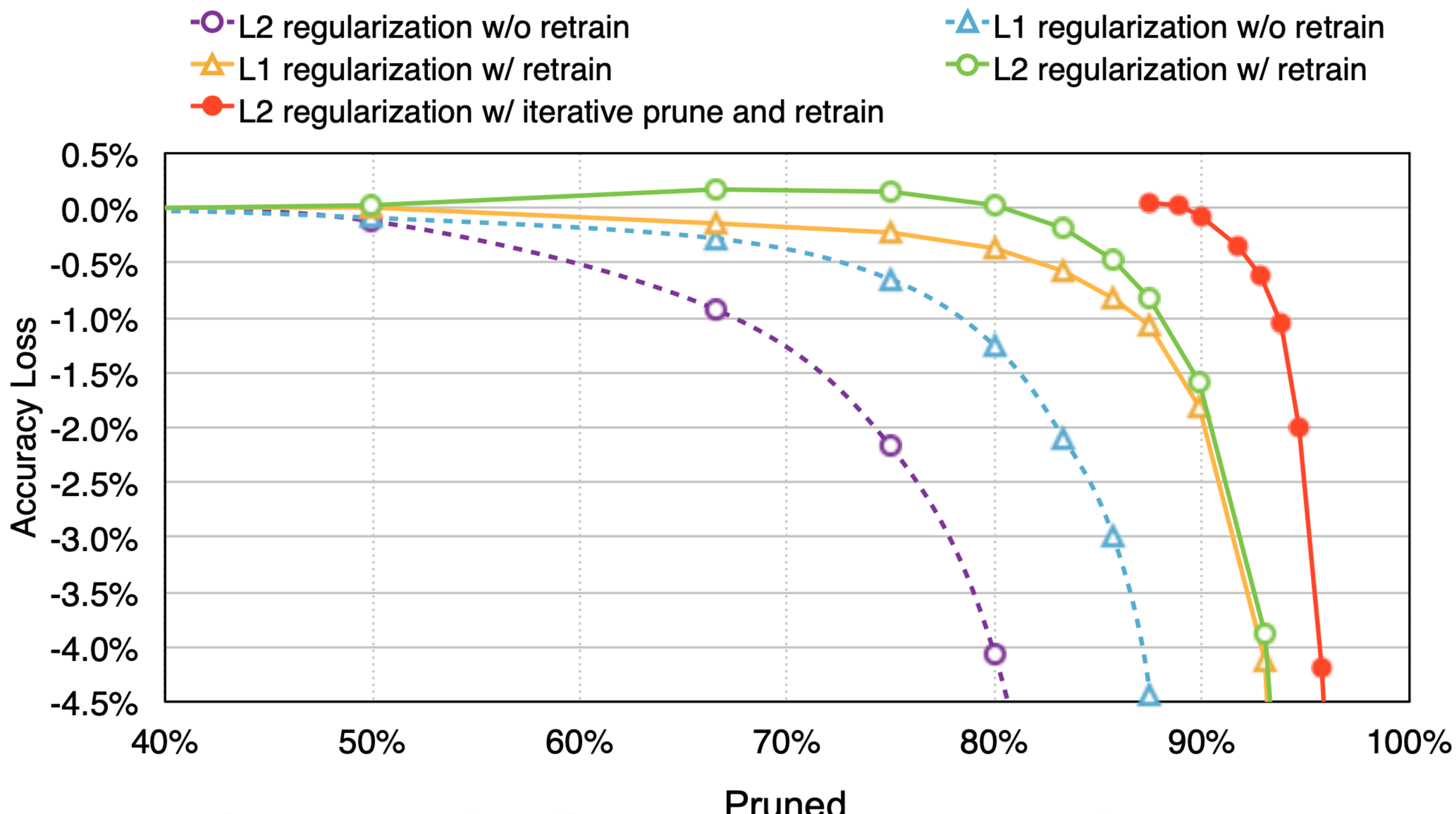
Pruning



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

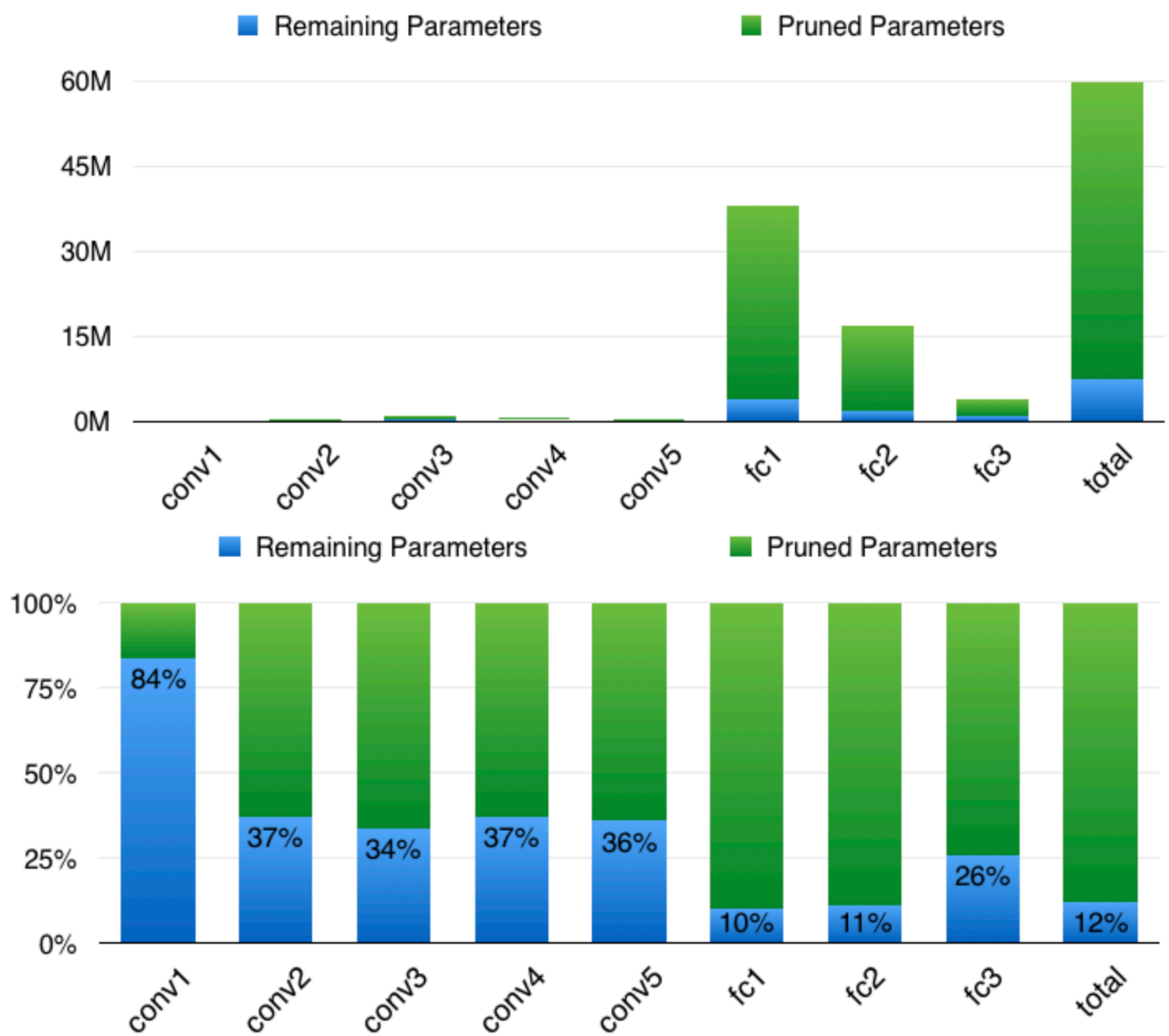
Retrain to Recover Accuracy



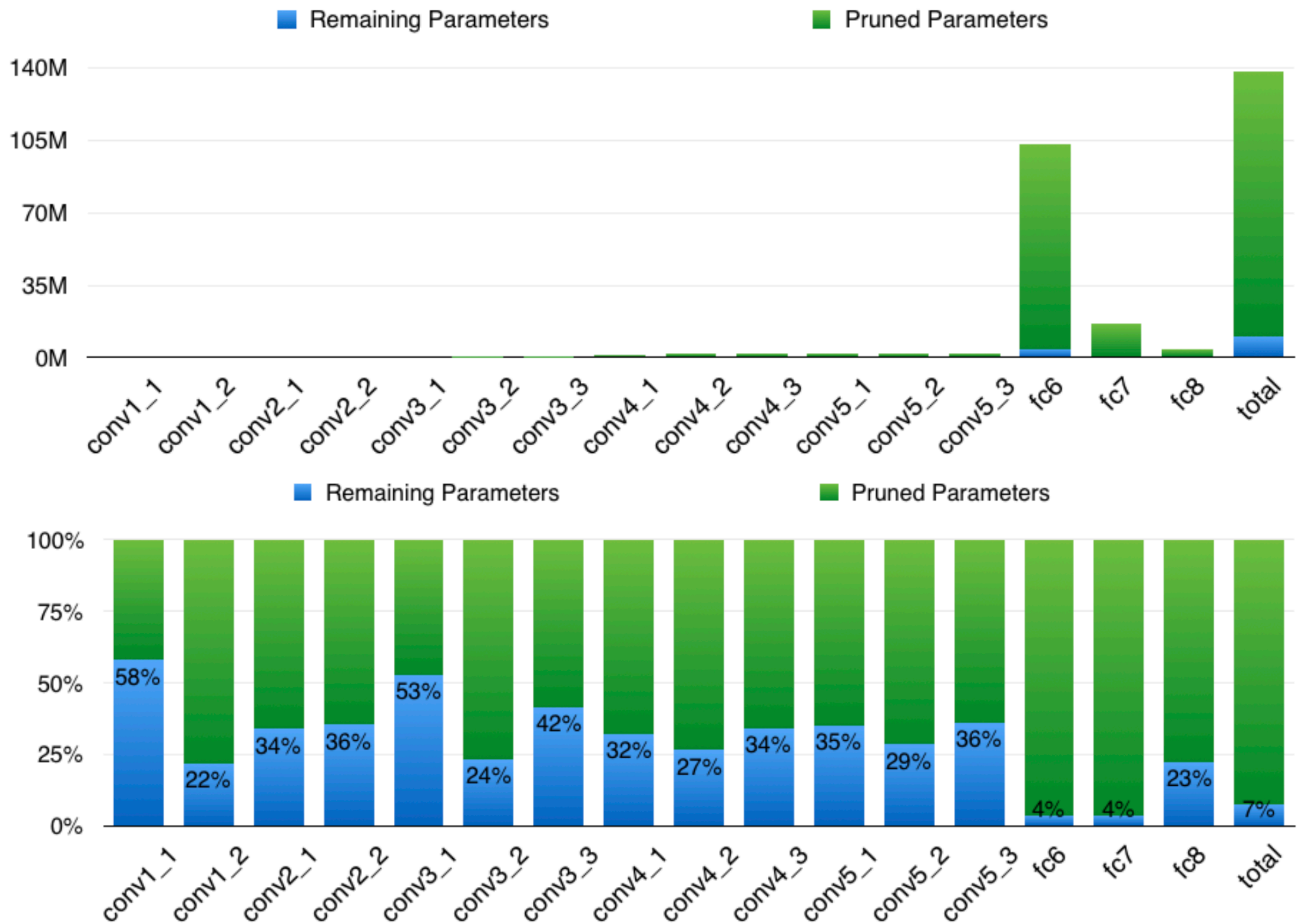


Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

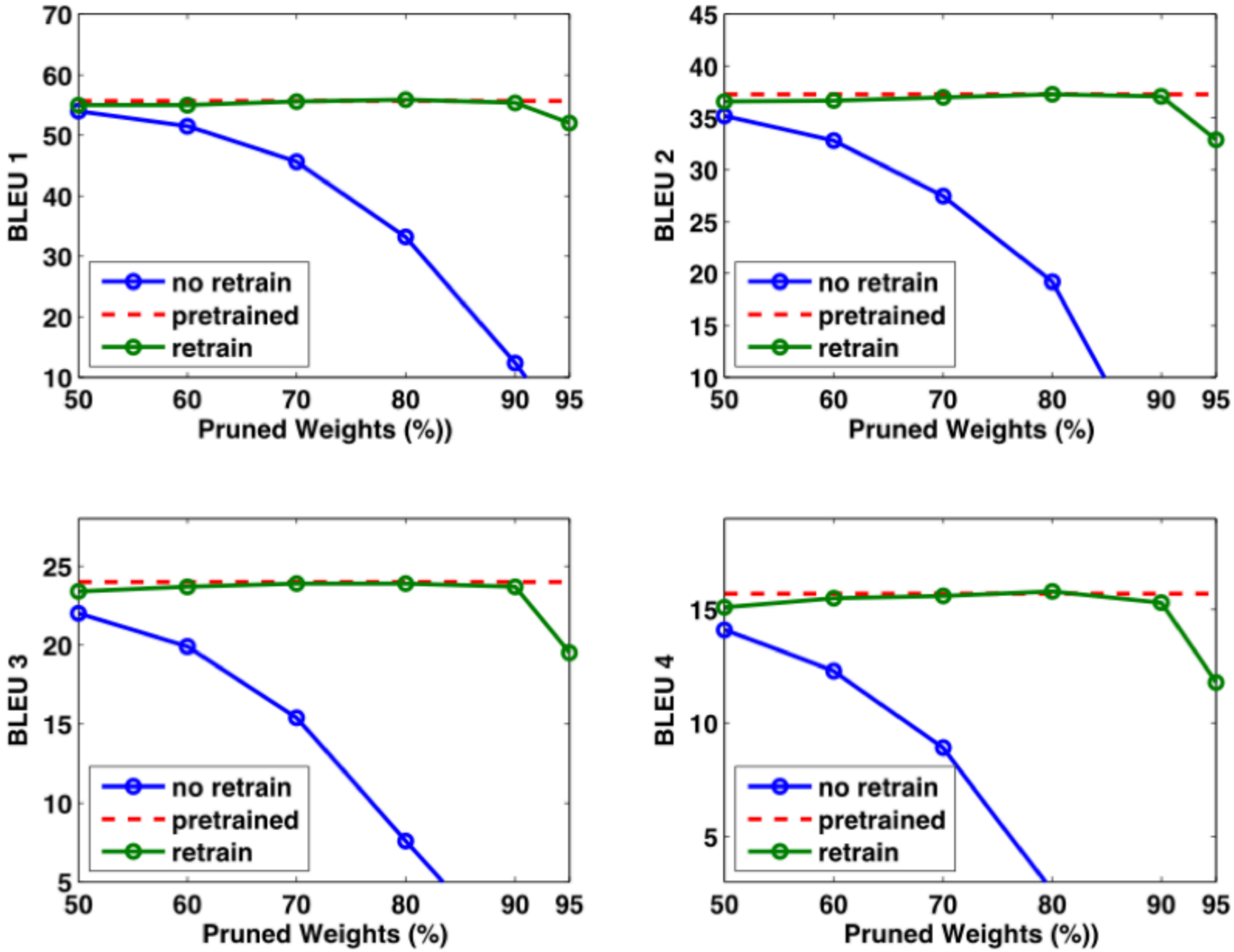
Pruning of AlexNet



Pruning of VGG-16



Pruning Neural Talk and LSTM



Pruning Neural Talk and LSTM

- **Original**: a basketball player in a white uniform is playing with a ball
- **Pruned 90%**: a basketball player in a white uniform is playing with a basketball

- **Original** : a brown dog is running through a grassy field
- **Pruned 90%**: a brown dog is running through a grassy area

- **Original** : a man is riding a surfboard on a wave
- **Pruned 90%**: a man in a wetsuit is riding a wave on a beach

- **Original** : a soccer player in red is running in the field
- **Pruned <u>95%</u>**: a man in a red shirt and black and white black shirt is running through a field

Speedup of Pruning on CPU/GPU

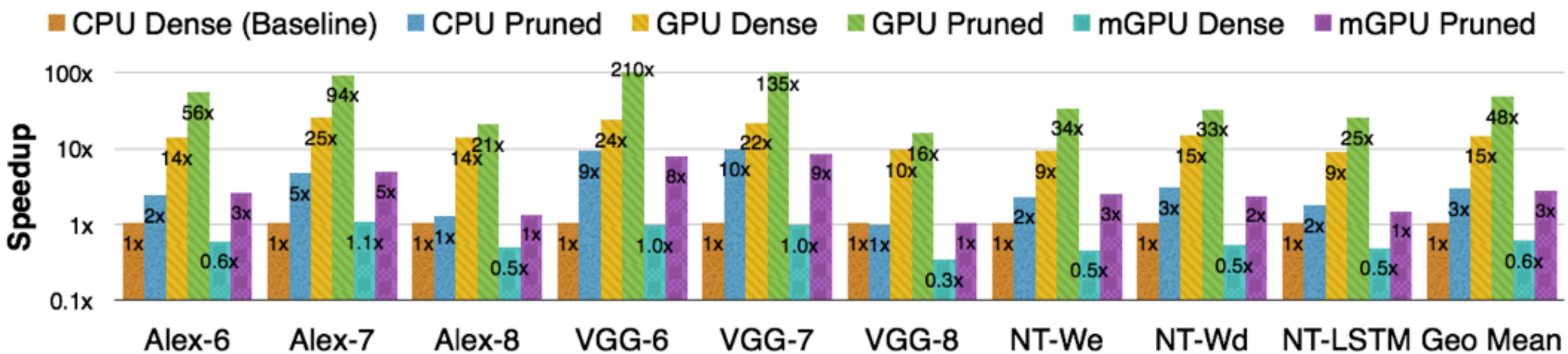


Figure 9: Compared with the original network, pruned network layer achieved 3× speedup on CPU, 3.5× on GPU and 4.2× on mobile GPU on average. Batch size = 1 targeting real time processing. Performance number normalized to CPU.

Intel Core i7 5930K: MKL CBLAS GEMV, MKL SPBLAS CSRMV
NVIDIA GeForce GTX Titan X: cuBLAS GEMV, cuSPARSE CSRMV
NVIDIA Tegra K1: cuBLAS GEMV, cuSPARSE CSRMV

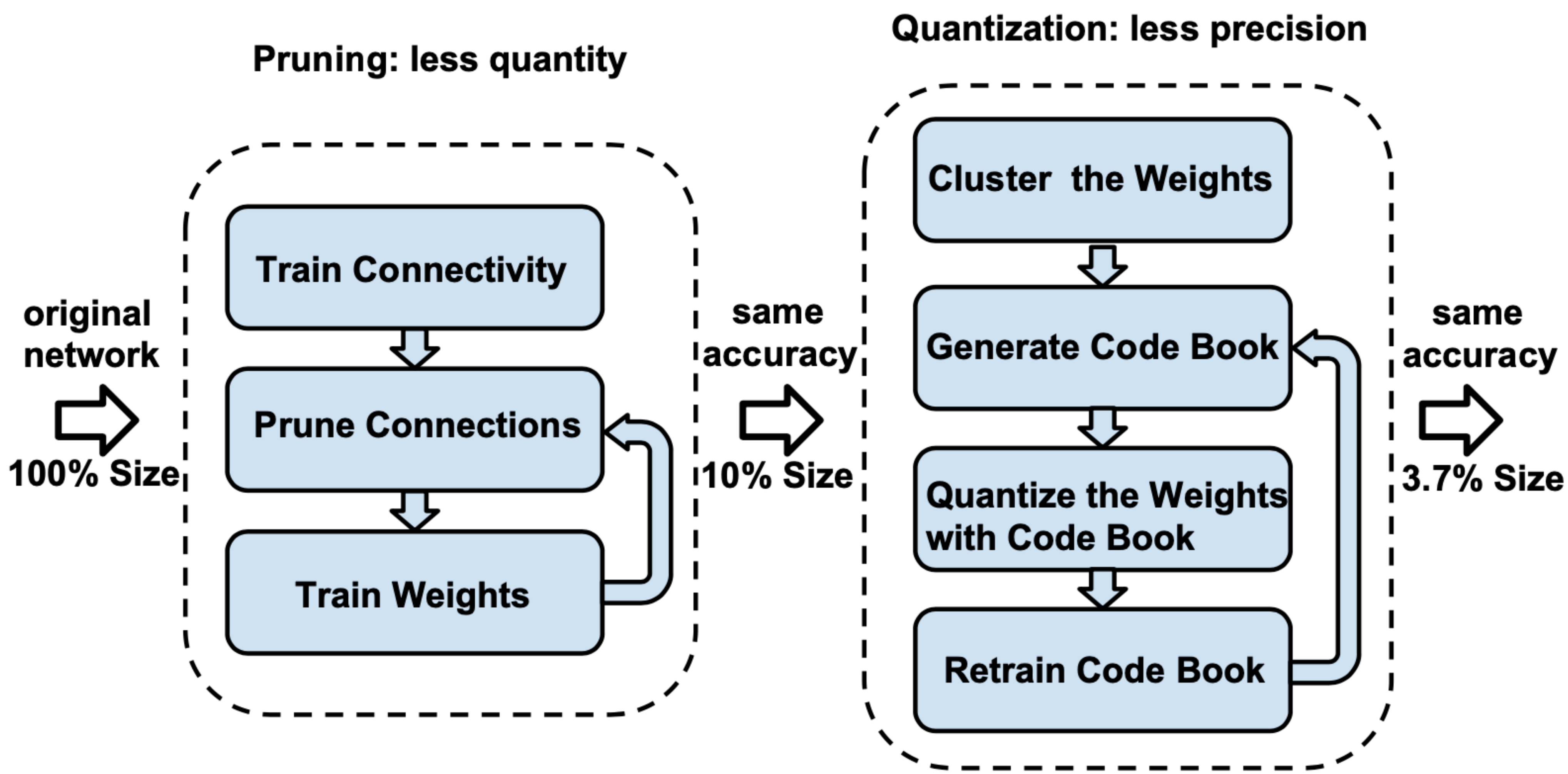
History of Pruning

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

Babak Hassibi, David G Stork, et al. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, pages 164–164, 1993.

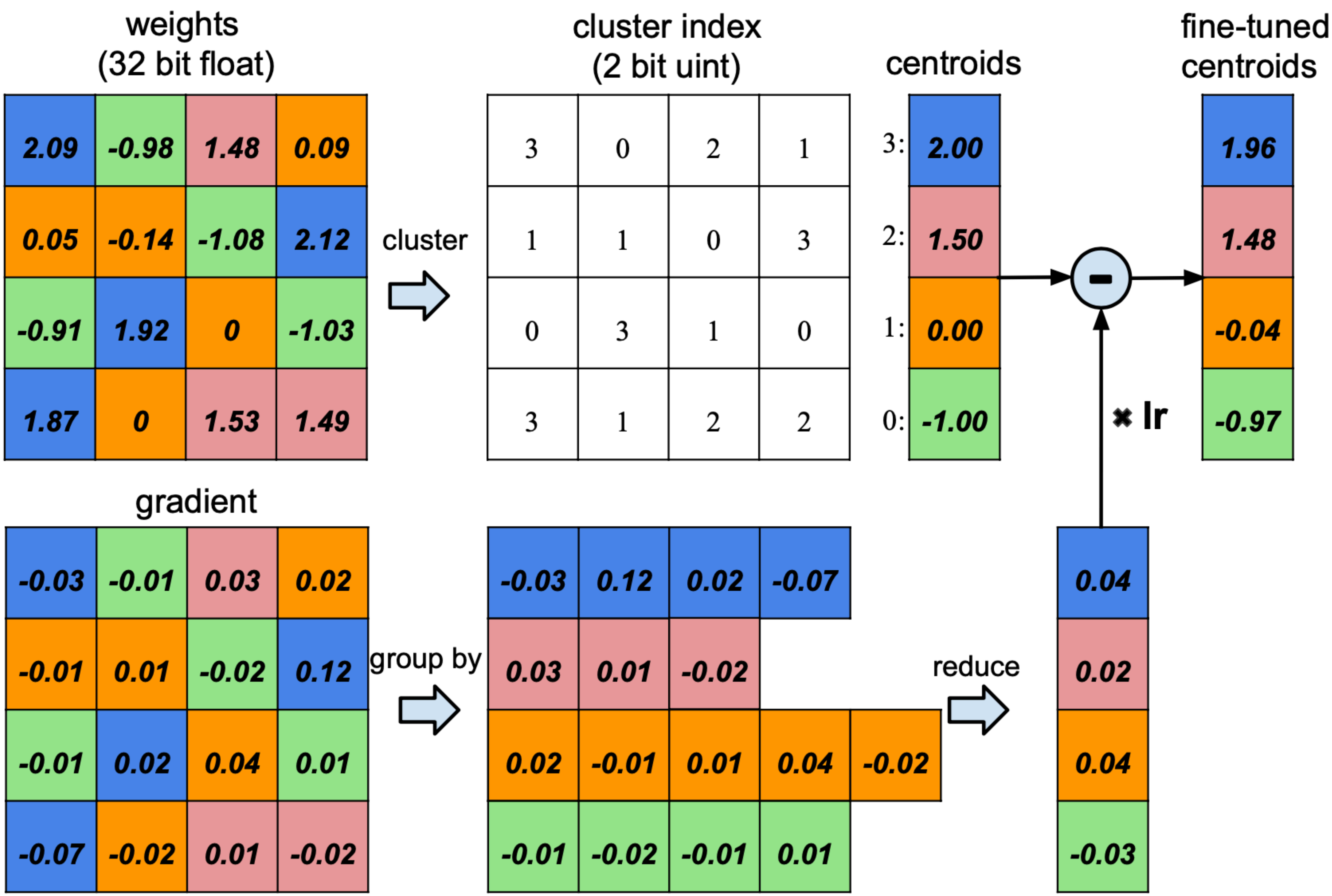
Reduce Storage for Each Remaining Weight

Trained Quantization (Weight Sharing)

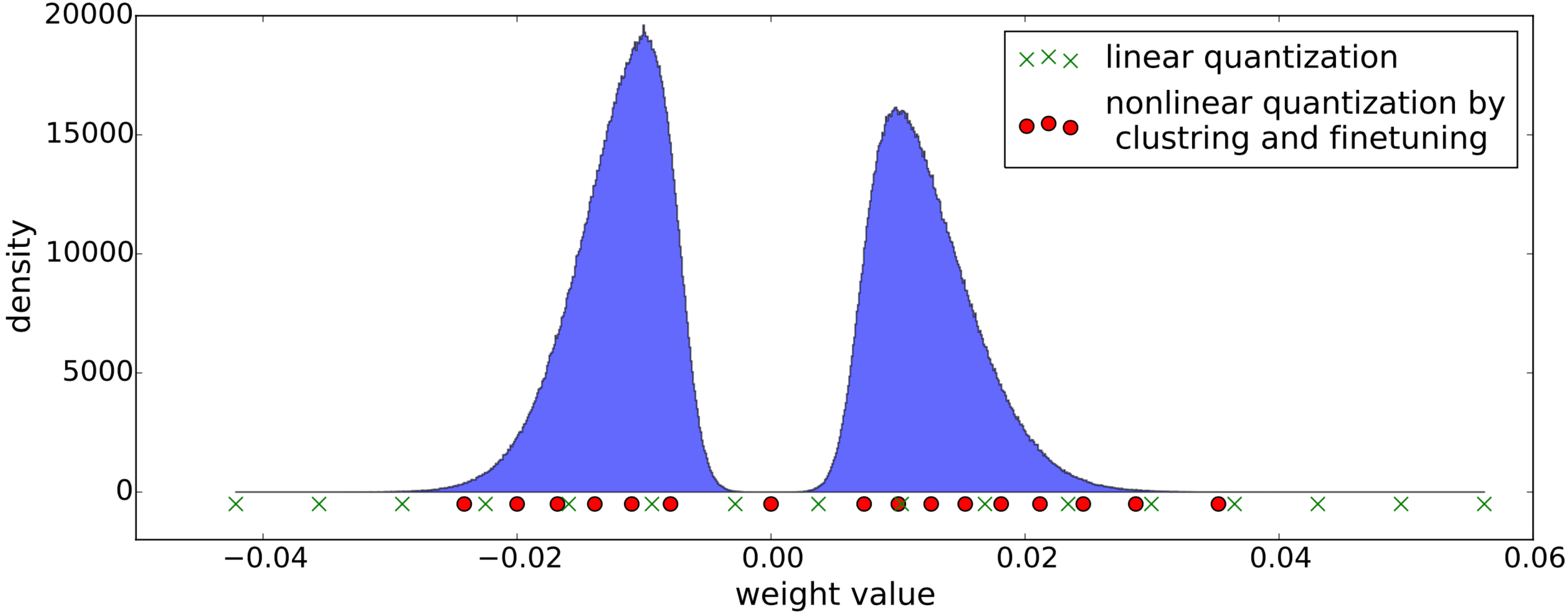


Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv 2015

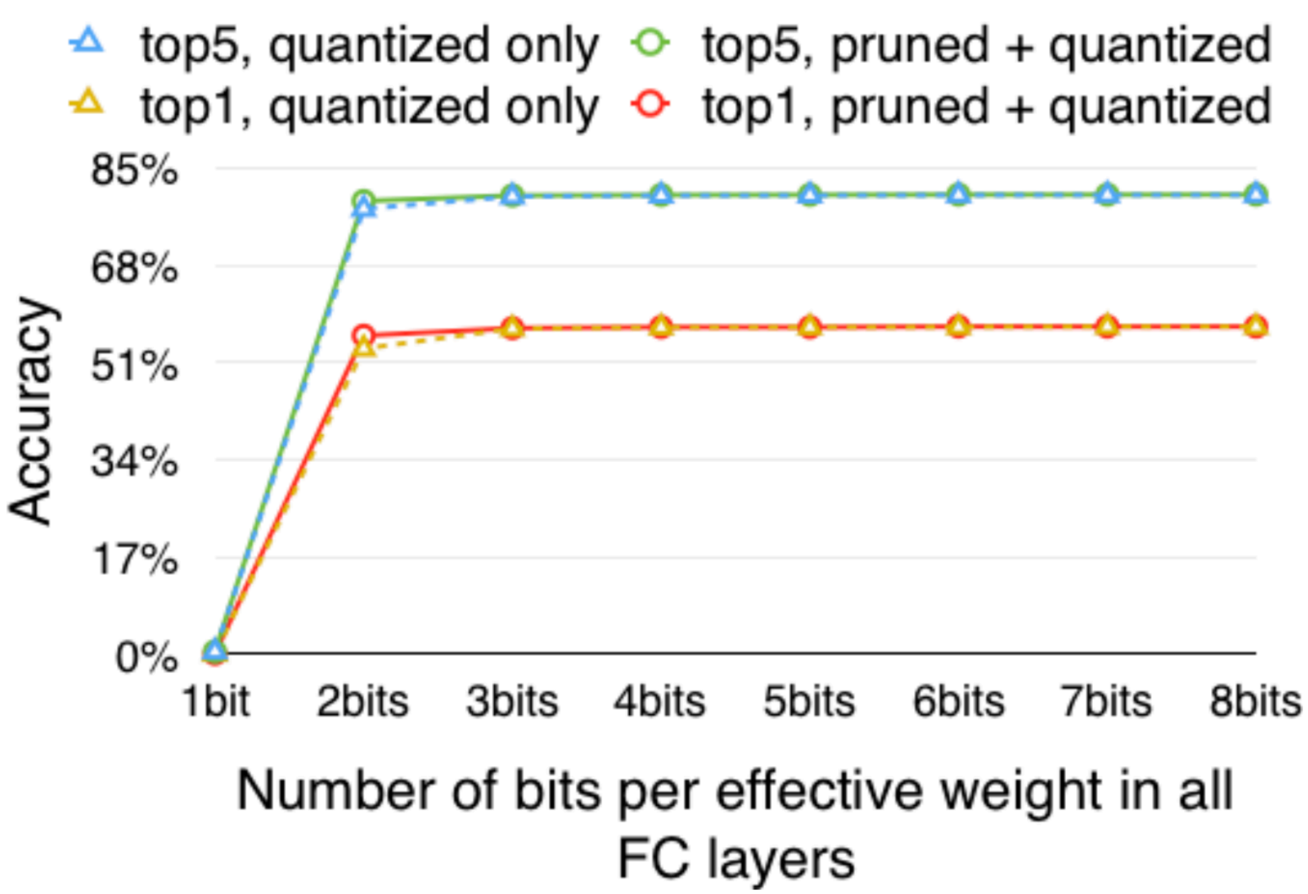
Weight Sharing via K-Means

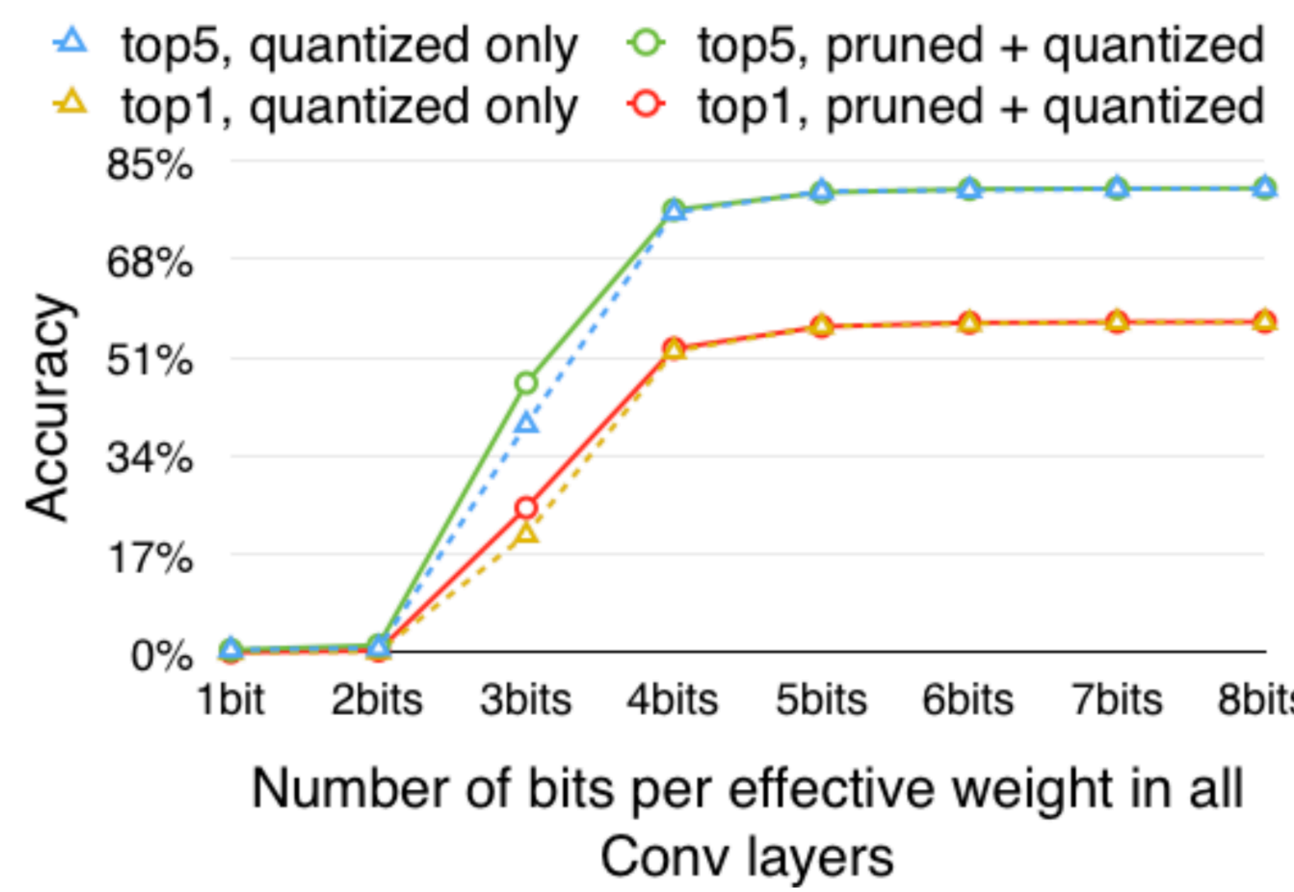


Trained Quantization

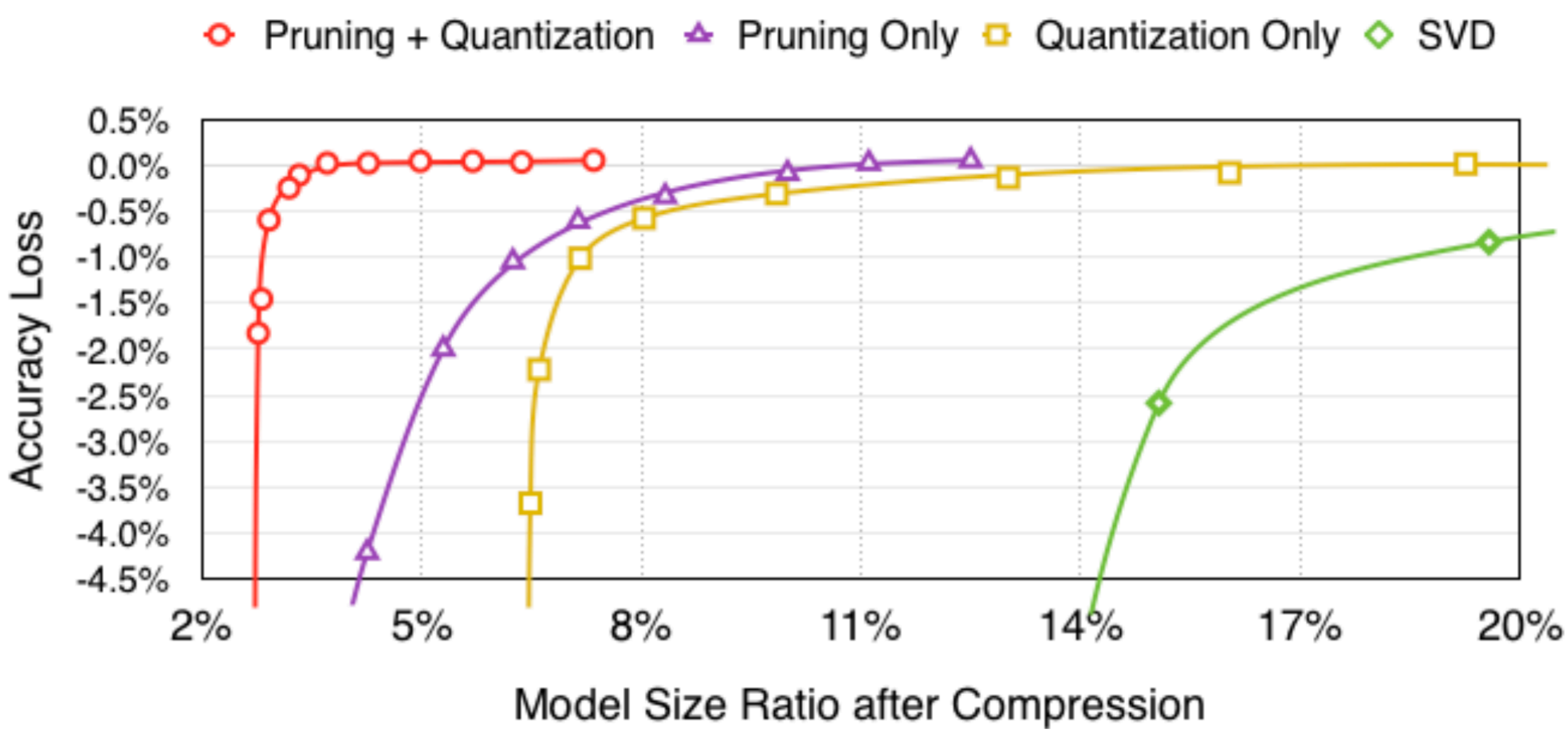


Bits per Weight





Pruning + Trained Quantization



Summary of Compression

Table 1: The compression pipeline can save $35\times$ to $49\times$ 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**$\times$
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39**$\times$
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35**$\times$
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49**$\times$

Compress neural networks without affecting accuracy by:

1. Pruning the unimportant connections =>
2. Quantizing the network and enforce weight sharing =>
3. Apply Huffman encoding

30x – 50x Compression Means

- Complex DNNs can be put in mobile applications (<100MB total):
  - 1GB network (250M Weights) become 20-30 MB

- Memory bandwidth reduced by 30-50x:
  - Particularly for FC layers in real-time applications with no reuse

- Memory working set fits in on-chip SRAM
  - 5pJ/word access vs 640pJ/word