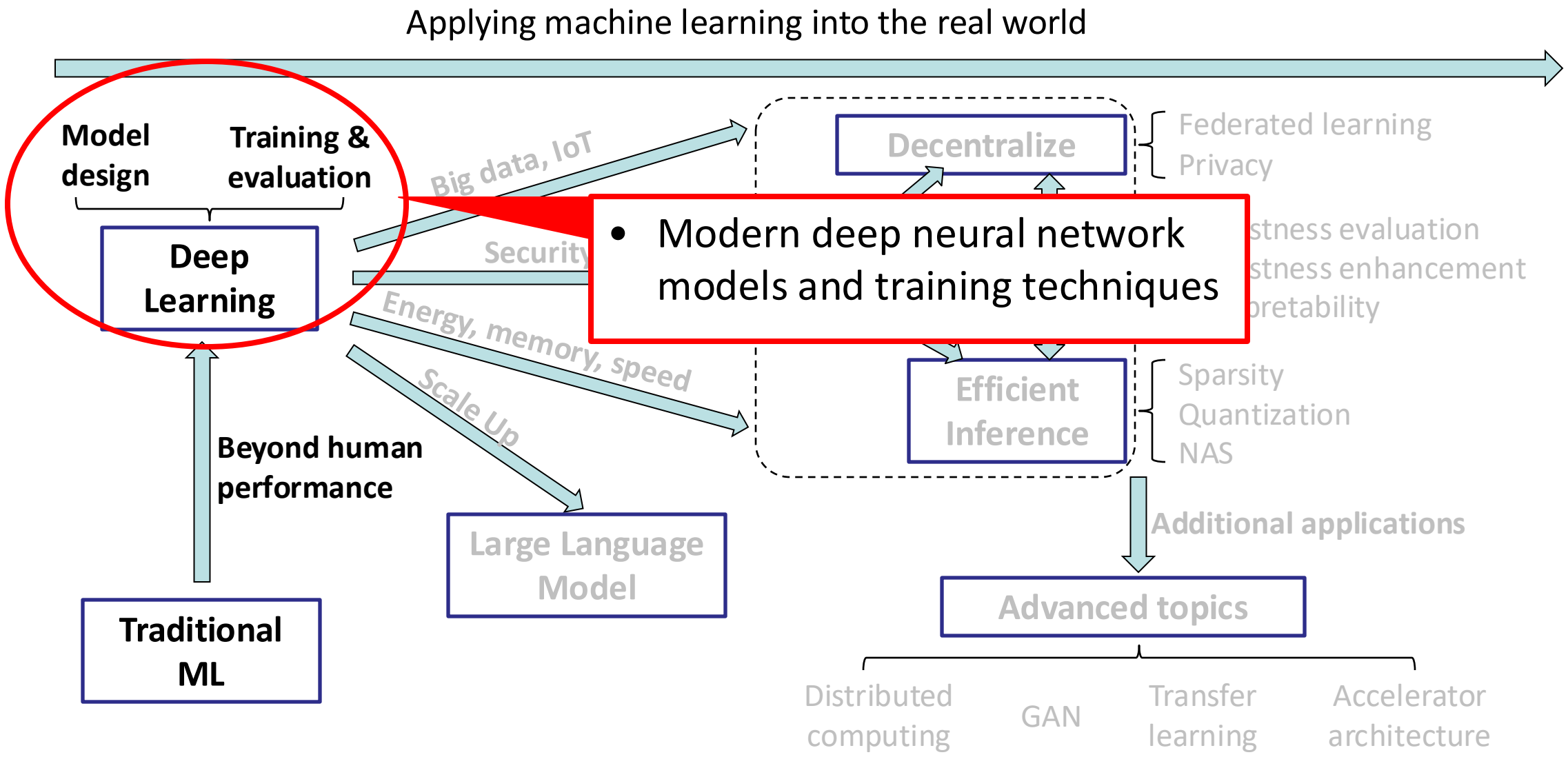
ECE 661 COMP ENG ML & DEEP NEURAL NETS

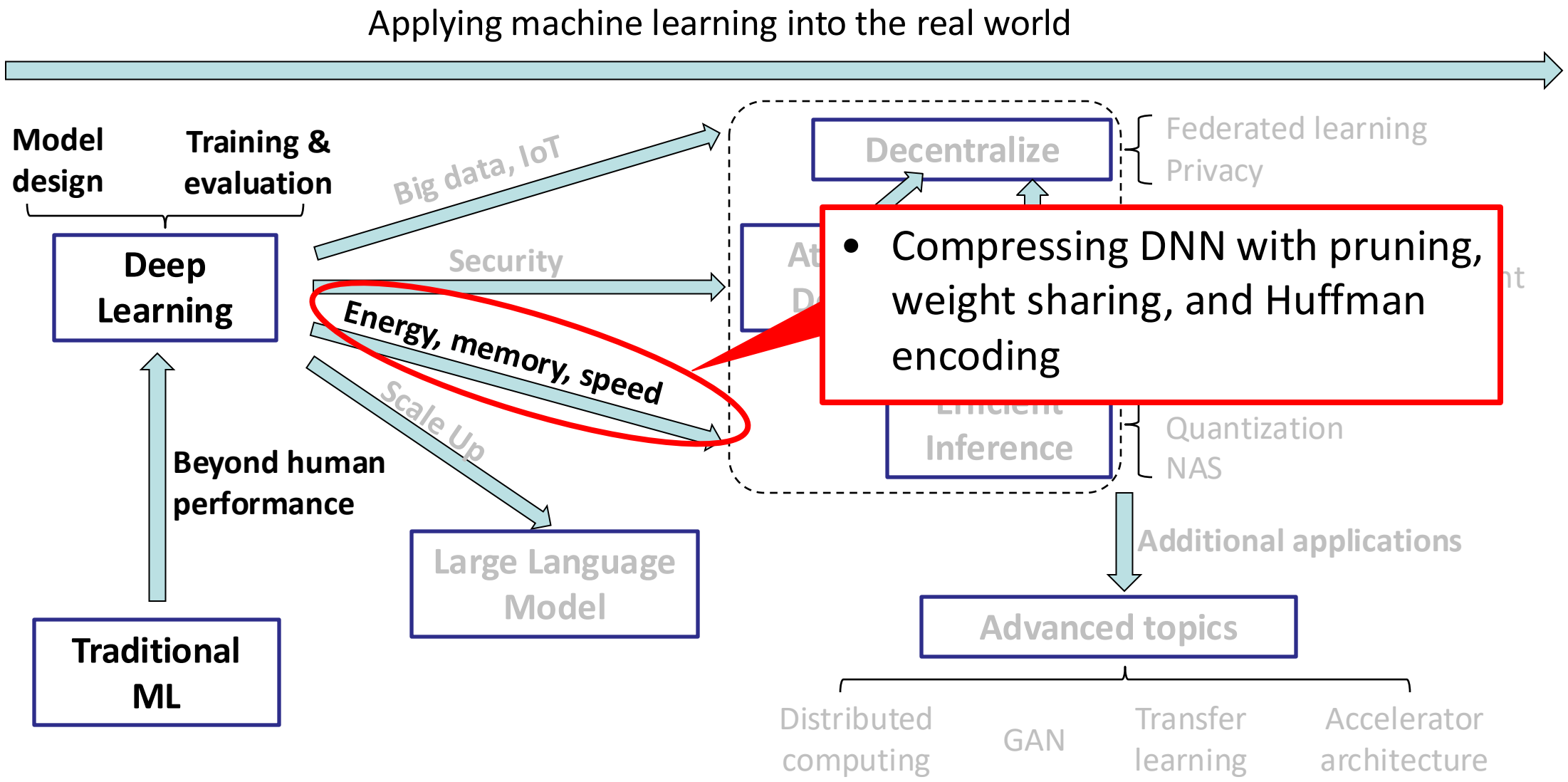
14. COMPRESSION OF THE DNN MODELS

HAI "HELEN" LI, SPRING 2025

Previously


Applying machine learning into the real world
Model design
Training & evaluation
Deep Learning
Traditional ML
Beyond human performance
Big data, IoT
Security
Energy, memory, speed
Scale Up
Large Language Model
Modern deep neural network models and training techniques
Decentralize
Federated learning
Privacy
stness evaluation
stness enhancement
pretability
Efficient Inference
Sparsity
Quantization
NAS
Additional applications
Advanced topics
Distributed computing
GAN
Transfer learning
Accelerator architecture

This lecture



Why is model compression needed?

- State-of-the-art DNNs are getting larger and requiring more computations

Model	Year	Top-5 Error	Param (MB)	FLOPs
AlexNet	2012	19.20%	233	727 M
ResNet-152	2016	6.70%	230	11 G
SENet	2017	4.47%	440	21 G

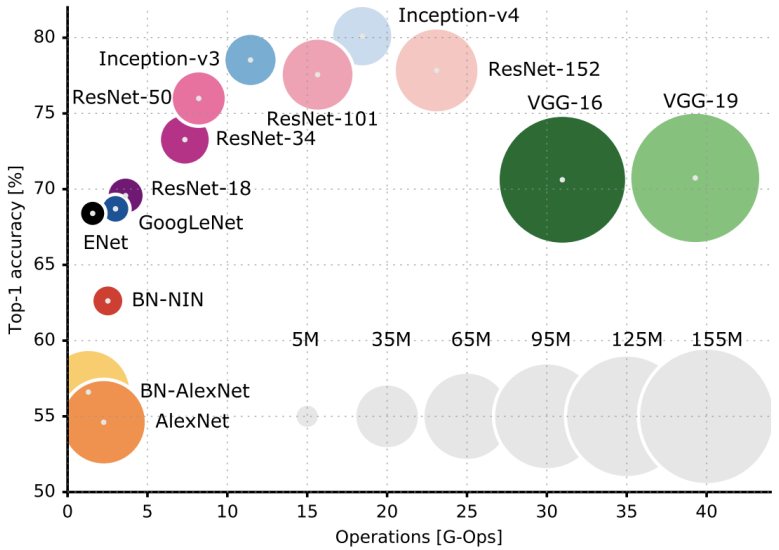


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

Why is model compression needed?

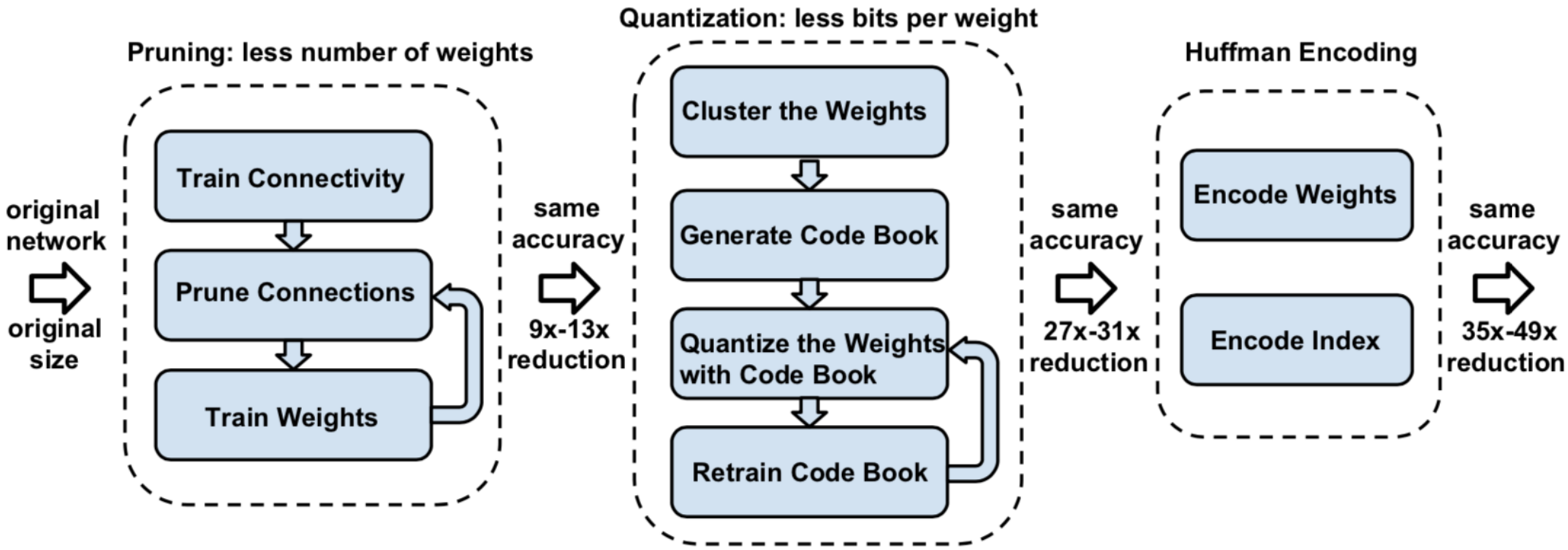
- Problems brought by large DNNs
  - More storage space and memory access, higher hardware cost
  - Larger computation load, longer latency and higher energy consumption
  - Harder to share and distribute models online
- Solutions
  - Reducing the number of parameters in a DNN (pruning, low rank)
  - Reducing the number of bits representing each parameter (quantization, coding)
  - Improving hardware efficiency (SW/HW co-design)

Model compression methods

- Reducing the number of parameters
  - Sparse optimization (pruning)
  - Sparsity-inducing regularizations
  - Low-rank decomposition
- Reducing the bit-width of parameters
  - Weight sharing
  - Fixed-point quantization
  - Encoding
- Finding novel efficient model architectures
  - Neural architecture search

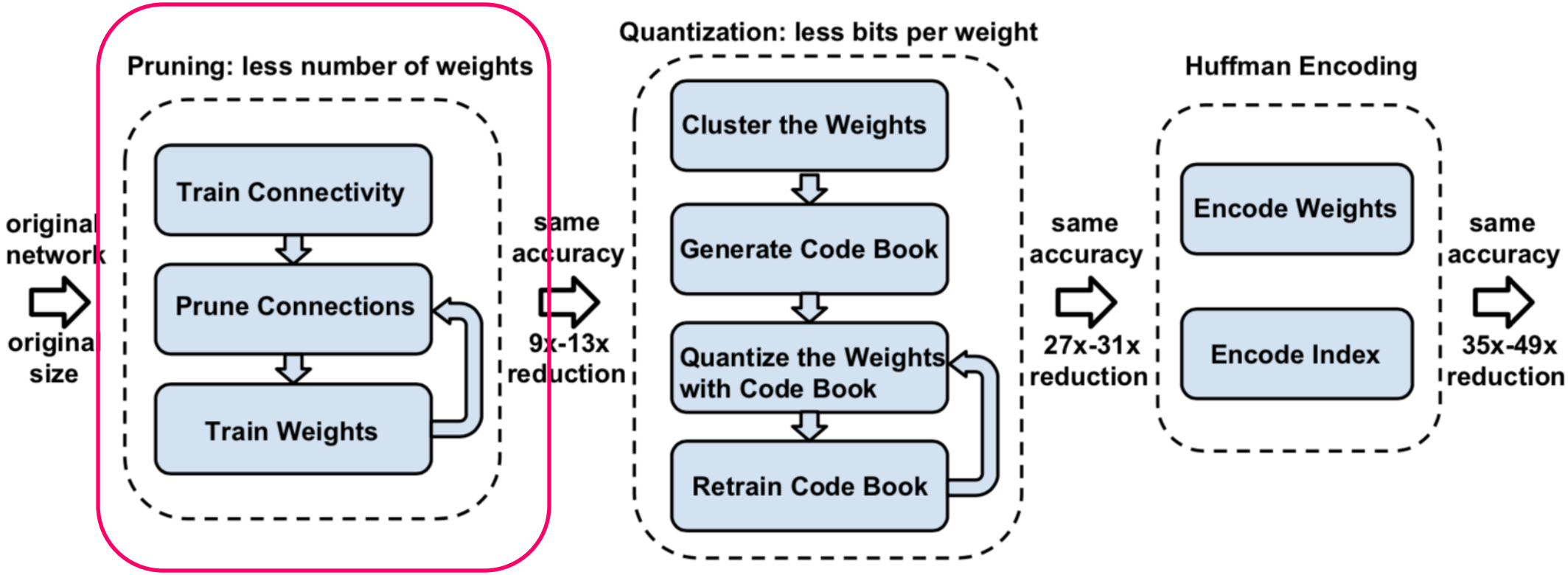
Let’s start with a straightforward model compression pipeline:

Deep Compression



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Deep Compression: Pruning neural networks

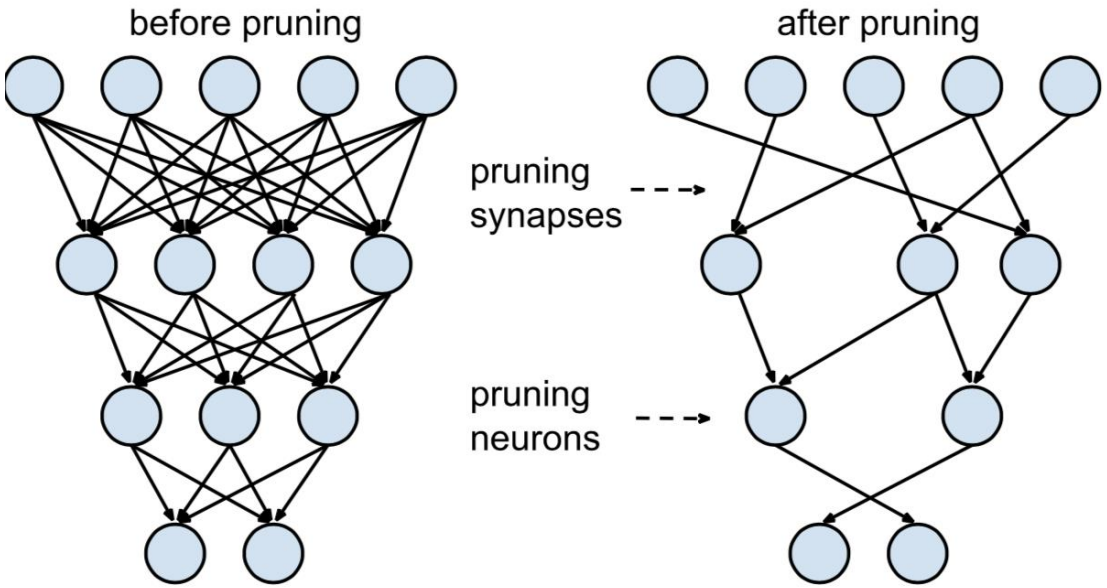


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Pruning neural networks

- Forcing more zeros (higher sparsity) in network weights

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems.* 2015.



- What can zero weights bring us?
  - Compressed storage (CSC format)
  - Less computation (sparse matrix multiplication)
  - Need specialized hardware to support, will introduce later

Achieving sparse neural network

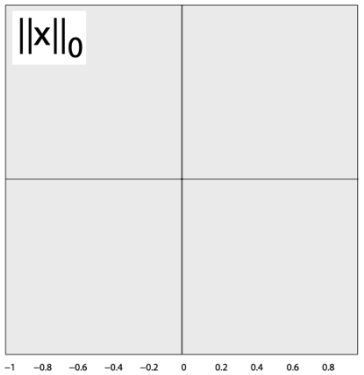
- Objective: Find a set of weights $\hat{\theta} \in \mathbb{R}^N$
  - Can achieve the optimal performance
  - Has at most n nonzero elements with $n \ll N$
- Mathematically, ℓ_0 norm $\left|\left|\cdot\right|\right|_0$ is used to represent the number of nonzero elements in a vector or a matrix
- Formal objective:

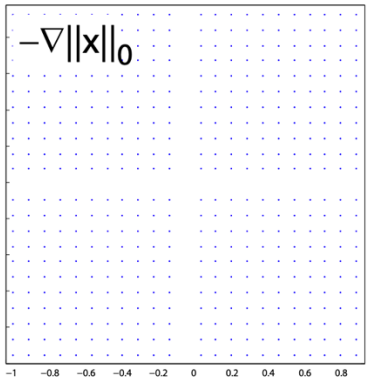
$$\hat{\theta} \coloneqq argmin_{\theta} \mathcal{L}(\theta)\ s.t. \left|\left|\theta\right|\right|_0 \leq n$$

Achieving sparse neural network

$$\hat{\theta} \coloneqq argmin_{\theta}\mathcal{L}(\theta)\ s.t.\ \left|\left|\theta\right|\right|_0 \leq n$$

- We are familiar with the blue part (SGD)
- What about the red part?
  - The ℓ_0 norm has only zero or infinity gradients, no useful gradient information





- New optimization methods are needed

Krishnan, Dilip, Terence Tay, and Rob Fergus. "Blind deconvolution using a normalized sparsity measure." *CVPR 2011*. IEEE, 2011.

Achieving sparse neural network

$$\hat{\theta} \coloneqq argmin_{\theta} \mathcal{L}(\theta) \; s.t. \left|\left|\theta\right|\right|_0 \leq n$$

- Projected gradient descent (PGD)
  - The red part, as a constraint, defines a *Feasible Set* for valid solutions of $\hat{\theta}$
  - At step l, perform gradient descent on the training objective (blue part) normally, then check:
    - If the resulting $\theta^{(l)}$ is within the feasible set, **continue**
    - If $\theta^{(l)}$ is outside of the feasible set, **project** it to the set by finding the valid solution $\theta'^{(l)}$ closest to $\theta^{(l)}$:

$$\theta'^{(l)} \coloneqq argmin_{\theta} \left|\left|\theta - \theta^{(l)}\right|\right|_2 \; s.t. \left|\left|\theta\right|\right|_0 \leq n$$

PGD is very useful for constrained optimization, we will see it again in adversarial attacks later this semester.

Achieving sparse neural network

$$\theta'^{(l)} := argmin_{\theta} \left|\left|\theta - \theta^{(l)}\right|\right|_2, s.t. \left|\left|\theta\right|\right|_0 \leq n$$

- The constraint still exists in the projection step, but the objective is much simpler
- Closed-form solution? YES!
  - $\theta'^{(l)}$ can be achieved by keep **pruning the smallest nonzero element** in $\theta^{(l)}$, until only n nonzero elements left

Suppose $\theta^{(l)} = [4 \quad 2 \quad 3 \quad 1]$, $n = 2$, then
$\theta'^{(n)} = [4 \quad 0 \quad 3 \quad 0]$, where 1 and 2 are pruned

Achieving sparse neural network

$$\hat{\theta} := argmin_{\theta}\mathcal{L}(\theta)\ s.t.\left|\left|\theta\right|\right|_0 \leq n$$

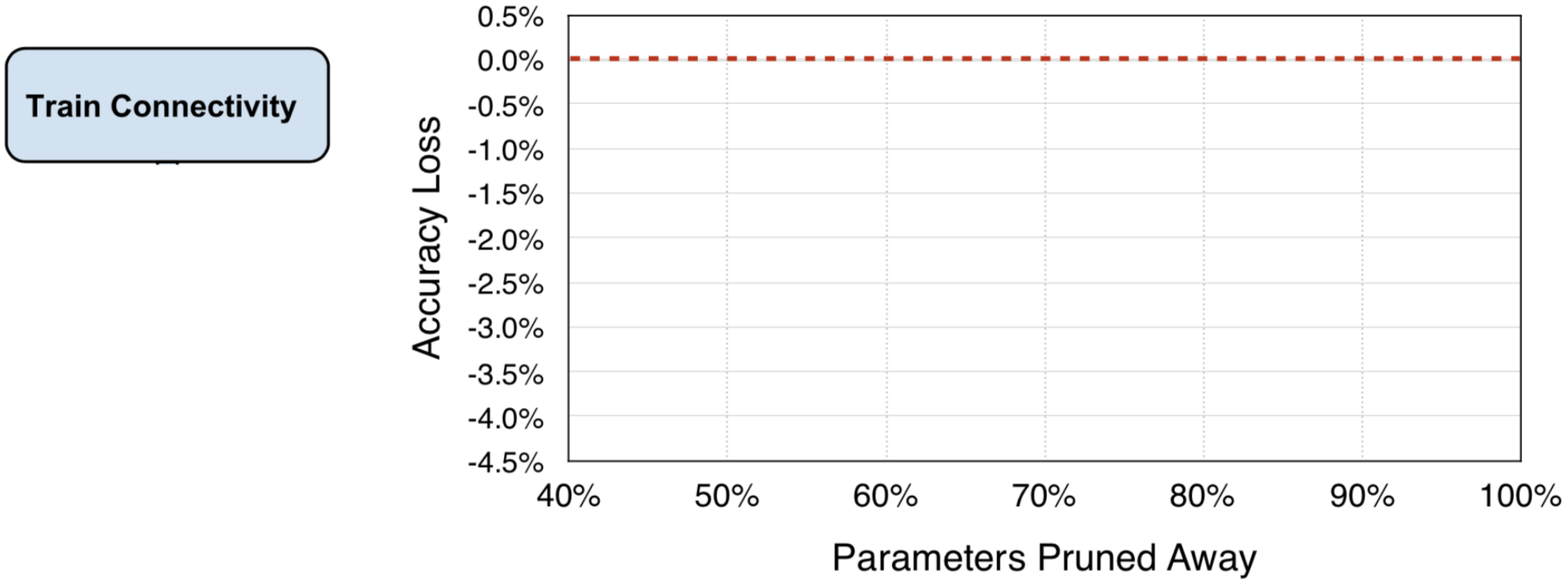
- PGD for sparse neural network
  - At the beginning of step l, current weight is $\theta'^{(l-1)}$: the pruned weight from step $l-1$
  - Training step: BP and SGD, note the loss is computed with the pruned weight

$$\theta^{(l)} = \theta'^{(l-1)} - \alpha\nabla_{\theta}\mathcal{L}\left(\theta'^{(l-1)}\right)$$

  - Projection step: pruning

$$\theta'^{(l)} = argmin_{\theta}\left|\left|\theta - \theta^{(l)}\right|\right|_2\ s.t.\left|\left|\theta\right|\right|_0 \leq n$$

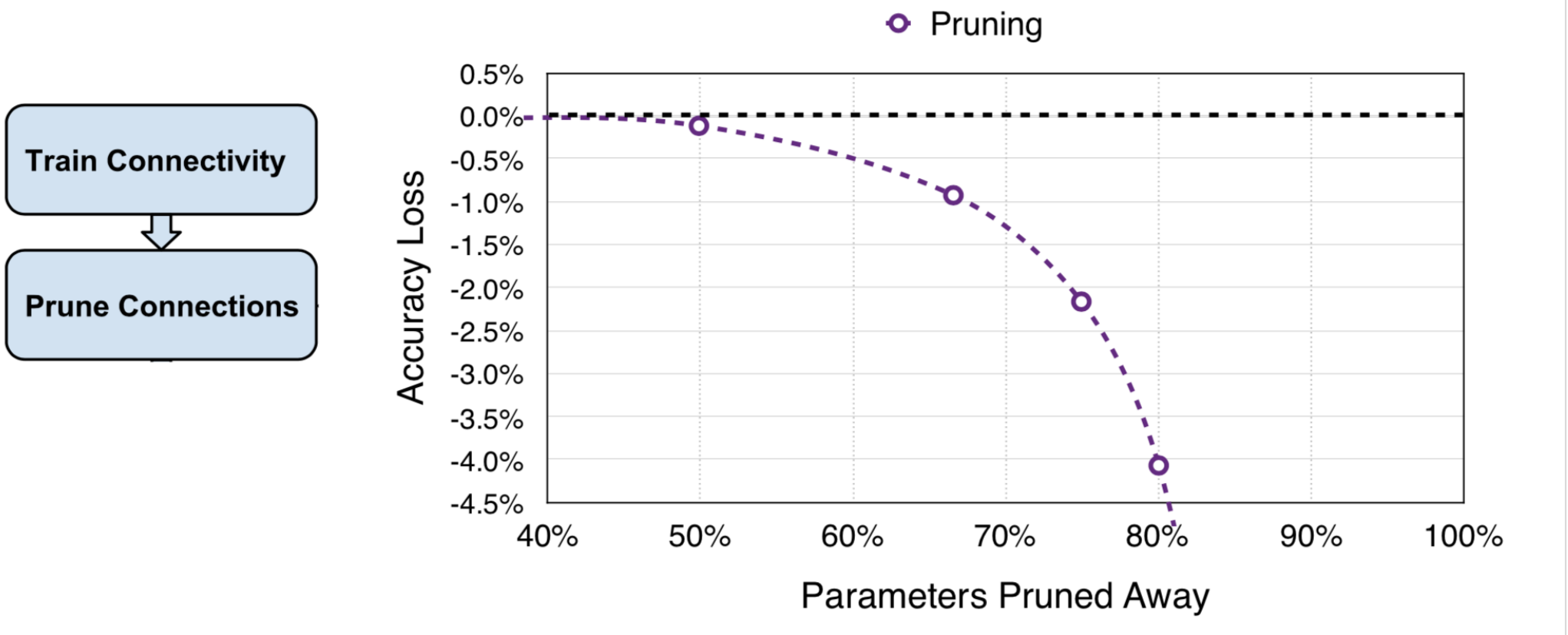
Iterative pruning and retraining



Here we start from a pretrained optimal model

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.
Figure from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf, credit to Song Han

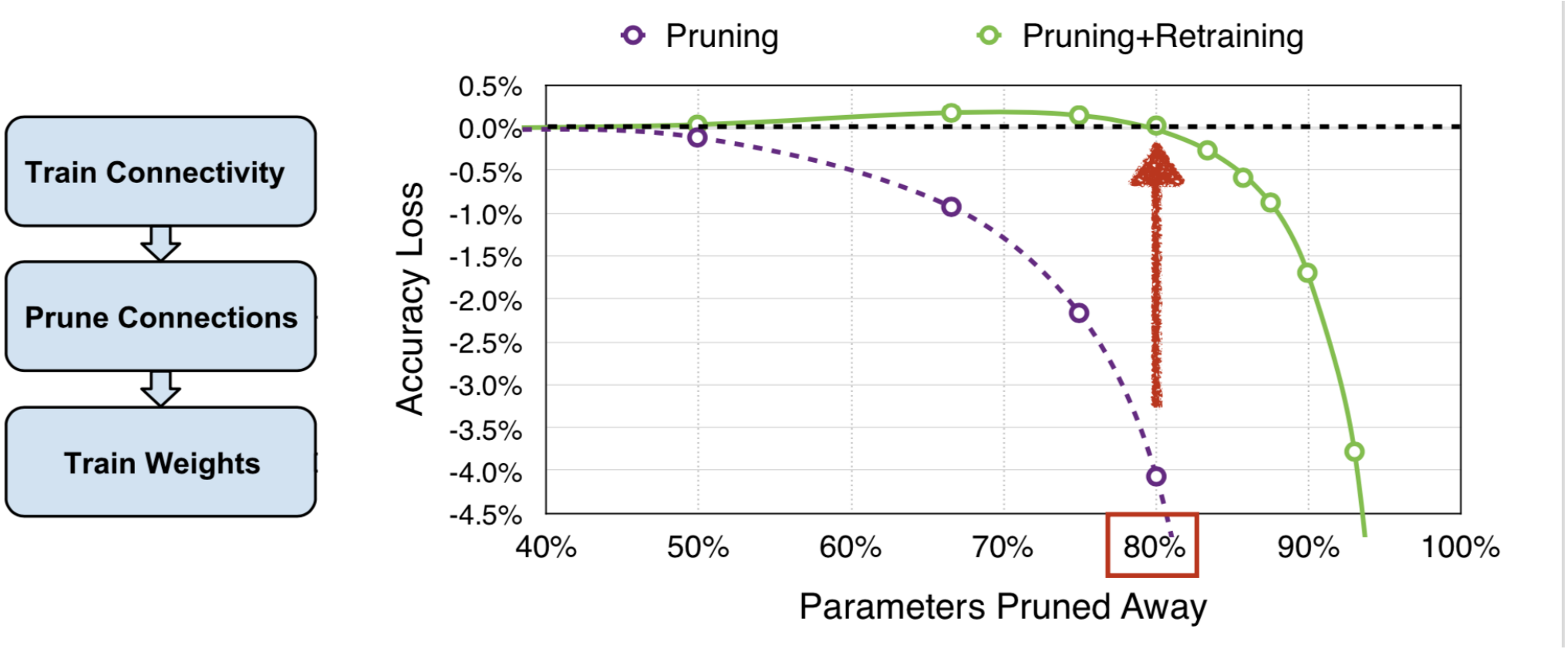
Iterative pruning and retraining



Pruning small weights away will hurt accuracy

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.
Figure from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf, credit to Song Han

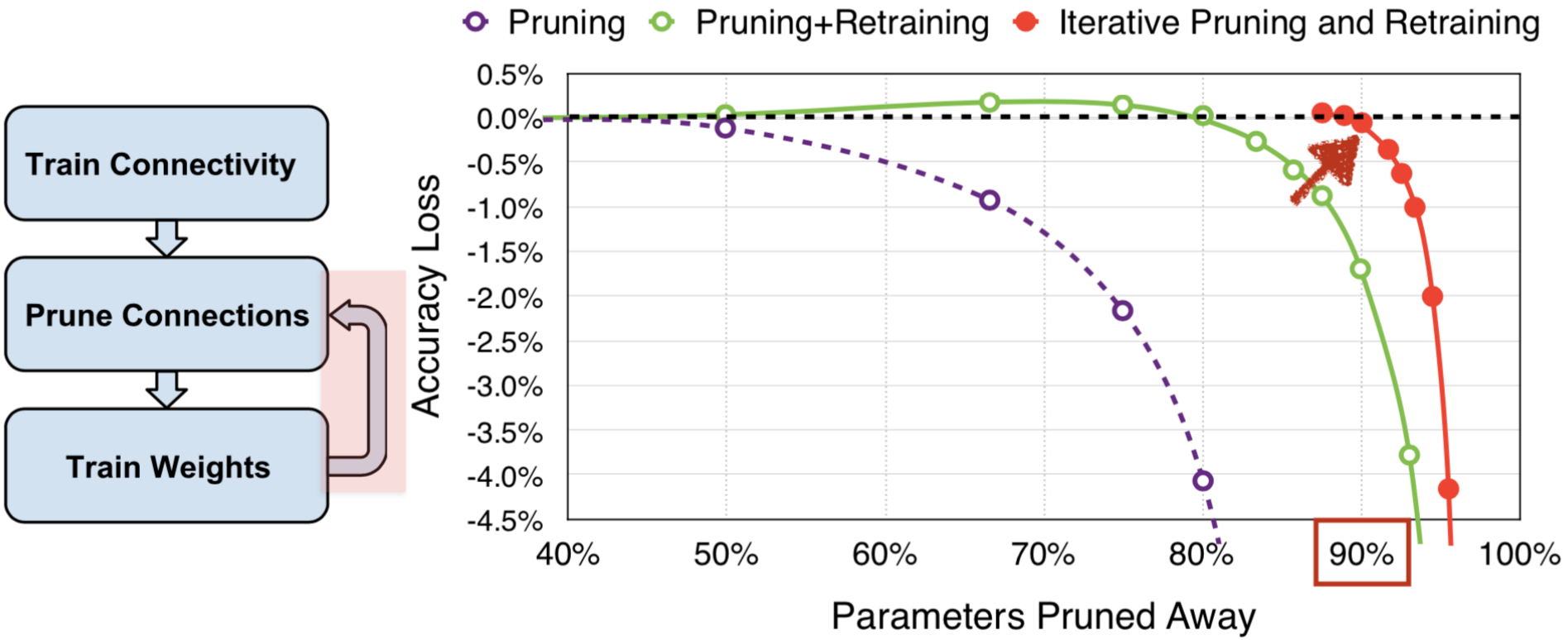
Iterative pruning and retraining



Directly retraining the pruned model with zero elements fixed can help recover accuracy

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.
Figure from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf, credit to Song Han

Iterative pruning and retraining



Iteratively pruning and retraining the weights (PGD) can further improve the performance

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.
Figure from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf, credit to Song Han

Common practice for threshold selection

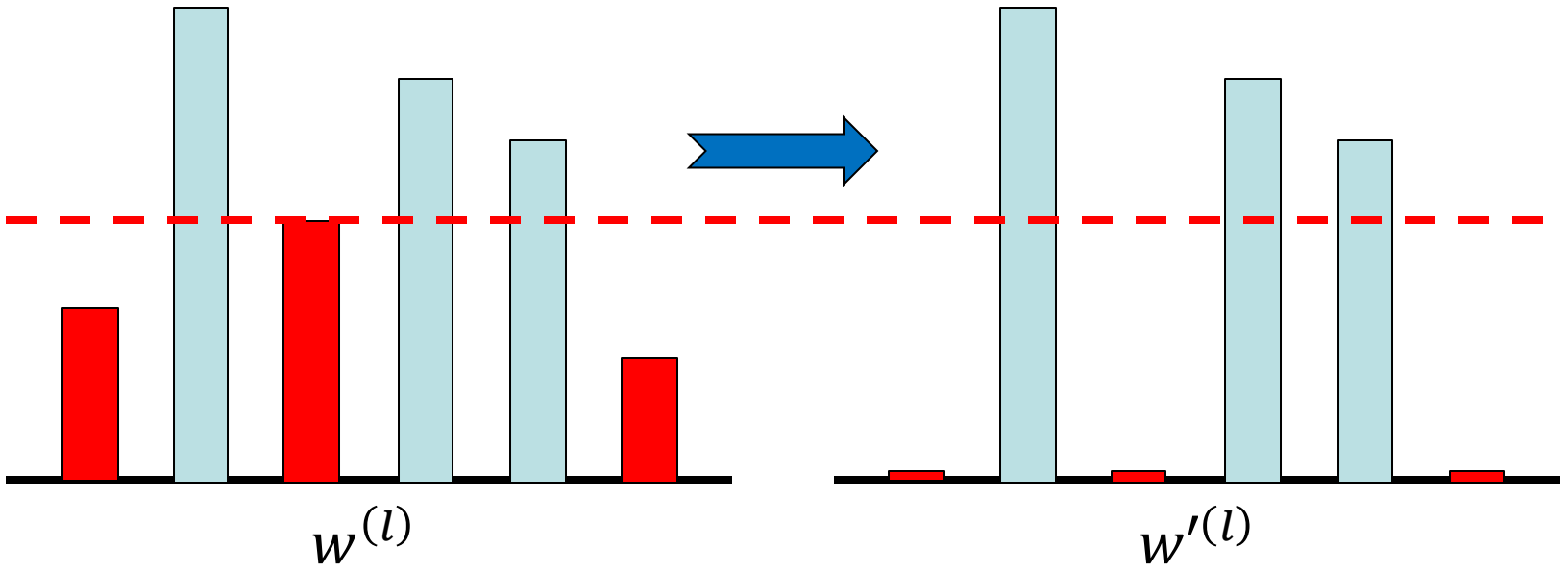
- Pruning threshold is the parameter to tune for trading off sparsity and performance
- Prune by percentage: prune a fixed percentage of weights in each layer
  - Original method proposed in Deep Compression
  - Directly determining the sparsity level in the final pruned model
  - Need a “ranking” process of all the weight parameters, may not be efficient

Common practice for threshold selection

- Prune by standard deviation (std): set pruning threshold to have constant ratio to the std of the parameters in each layer
  - A commonly applied approach in practice
  - Since the weight are distributed as Gaussian, a fixed ratio to std also infers a steady percentage
  - Works better with sparsity-inducing regularizers, to be introduced later

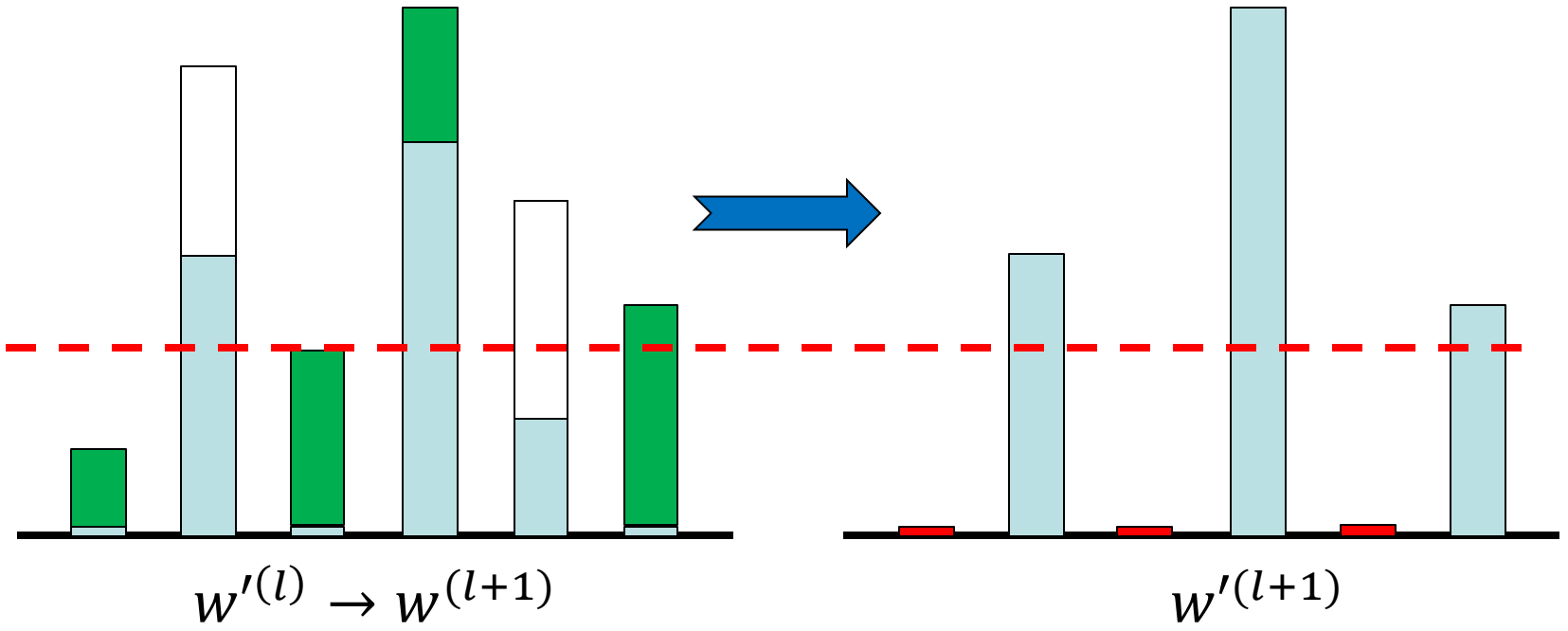
Intuition behind iterative pruning

- Larger weight elements have larger effects on the activation, therefore more important
- Pruning step:
  - Identify and remove unimportant elements (suppose we want to set the sparsity to 50%)



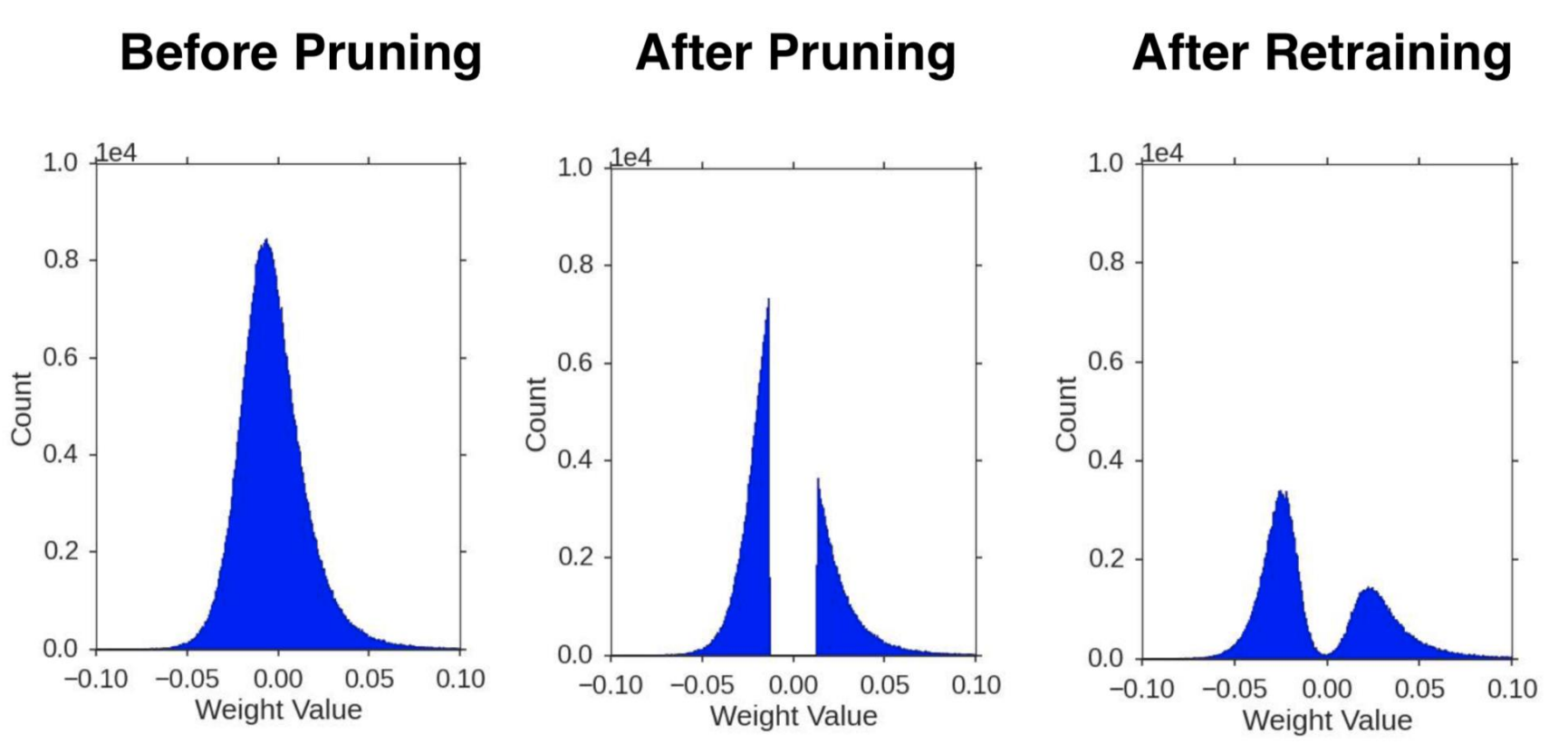
Intuition behind iterative pruning

- Next training step:
  - Gradient descent, adjusting parameters
  - Reidentifying important elements
  - Make remaining weights better fit to pruned model



Intuition behind iterative pruning

- Changes in nonzero weight distribution



Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.
Figure from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf, credit to Song Han

Effectiveness of iterative pruning

- Reduce parameter amount without performance drop

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**×
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**×
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**×
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13**×

- Effective comparing to previous methods
  - Evaluated on the AlexNet model

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
Baseline Caffemodel [26]	42.78%	19.73%	61.0M	1×
Data-free pruning [28]	44.40%	-	39.6M	1.5×
Fastfood-32-AD [29]	41.93%	-	32.8M	2×
Fastfood-16-AD [29]	42.90%	-	16.4M	3.7×
Collins & Kohli [30]	44.40%	-	15.2M	4×
Naive Cut	47.18%	23.23%	13.8M	4.4×
SVD [12]	44.02%	20.56%	11.9M	5×
Network Pruning	**42.77%**	**19.67%**	**6.7M**	**9**×

Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.

Inference with the pruned network

- Iterative connection pruning will lead to non-structured sparse weight matrices
- May not bring much speedup on traditional platforms like GPU, even with specifically designed sparse matrix multiplication algorithm

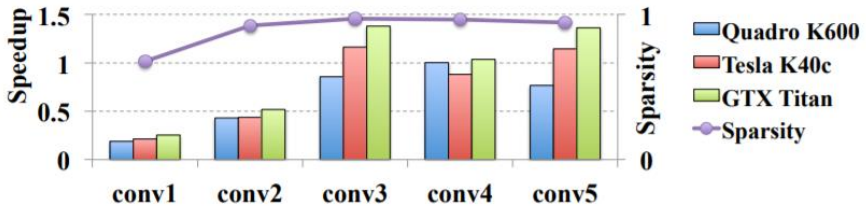
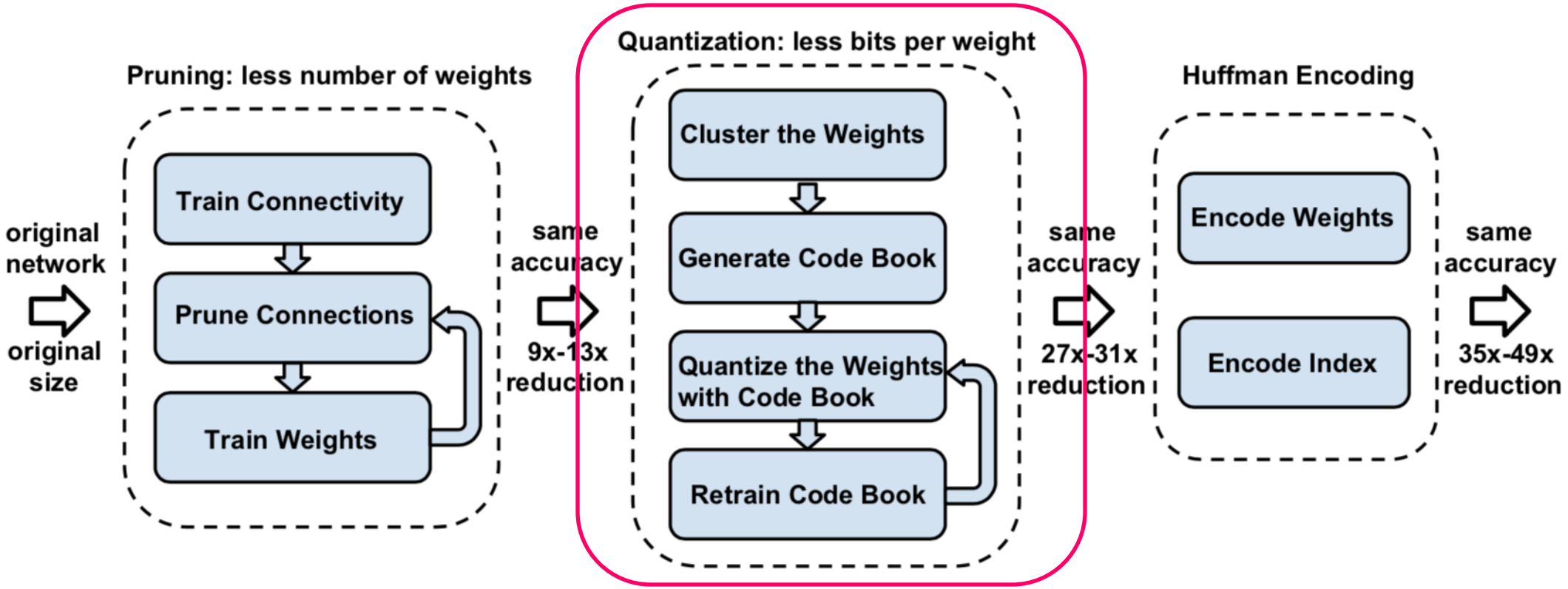


Figure 1: Evaluation speedups of AlexNet on GPU platforms and the sparsity. conv1 refers to convolutional layer 1, and so forth. Baseline is profiled by GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

- Specialized hardware accelerator should be designed to store and compute non-structured sparse weights

Wen, Wei, et al. "Learning structured sparsity in deep neural networks." *Advances in neural information processing systems*. 2016.

Deep Compression: Weight sharing



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

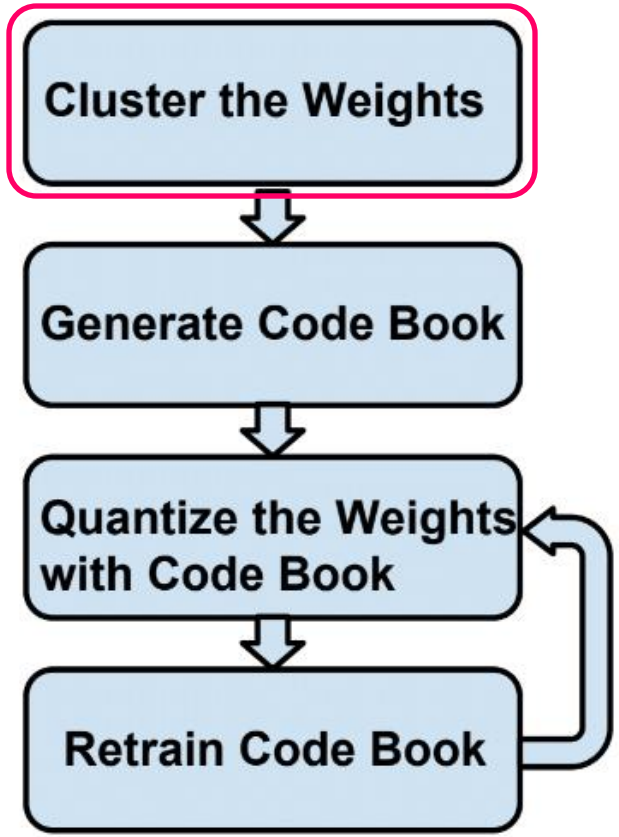
Representing weights with fewer bits

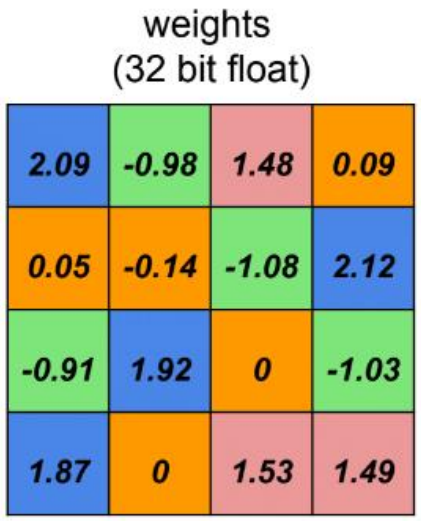
- After pruning: fewer weight elements to be stored
- Each weight element is represented as a float-point number (32 bits), still quite large
- Use less bits for representation?
  - Represent each weight element with lower precision (as low as 1 bit). Will discuss later
  - All the weight elements share a small amounts of different values (centroids), while cluster index can be used to represent weights

Weight Sharing

Weight sharing procedure

- Clustering
  - Determine how many different values we want to have in the end (e.g., 4)
  - For each layer, define *initial centroids* and cluster the weight of each layer into groups with k-means algorithm

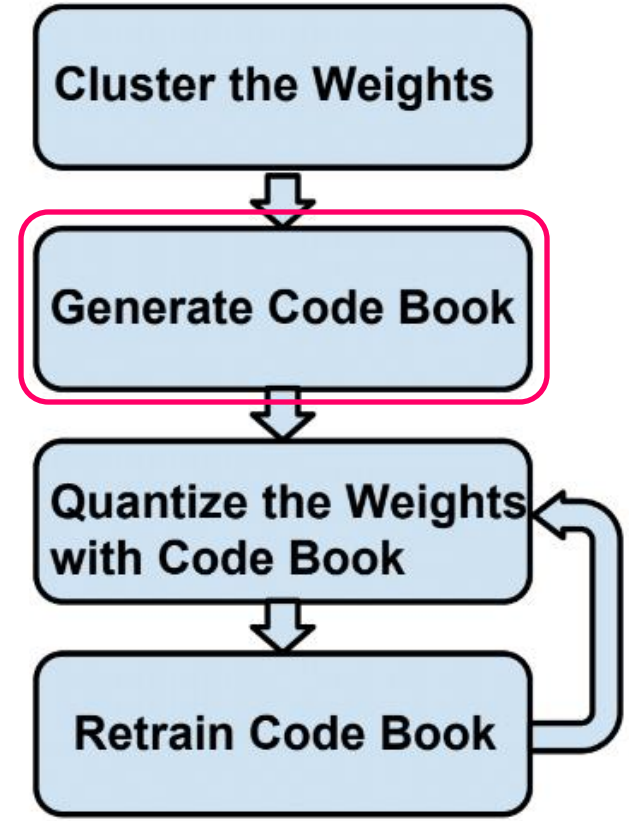


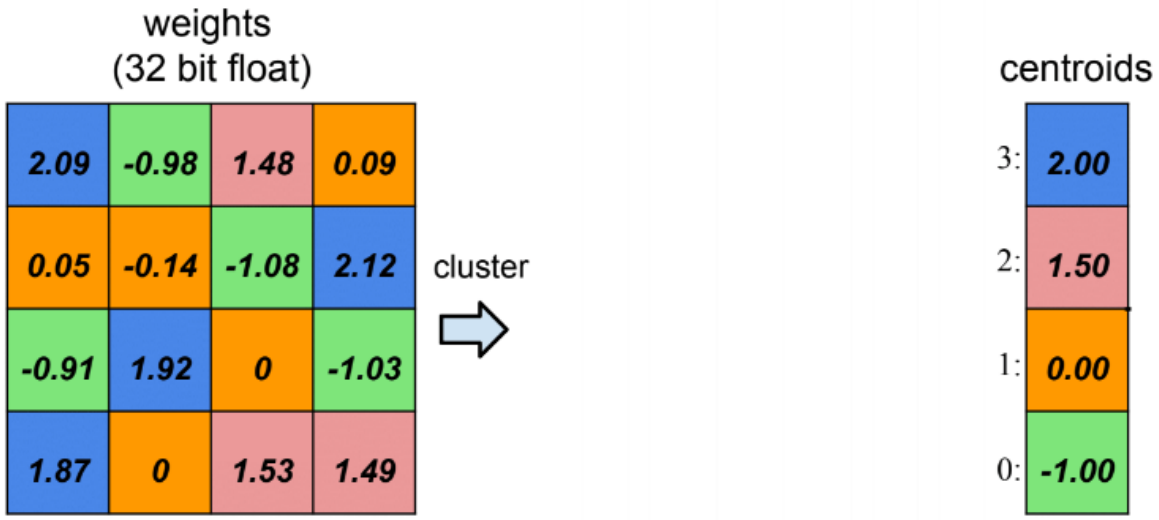


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Weight sharing procedure

- Code book generation
  - Gather the centroid of each group
  - Assign a cluster index to each centroid



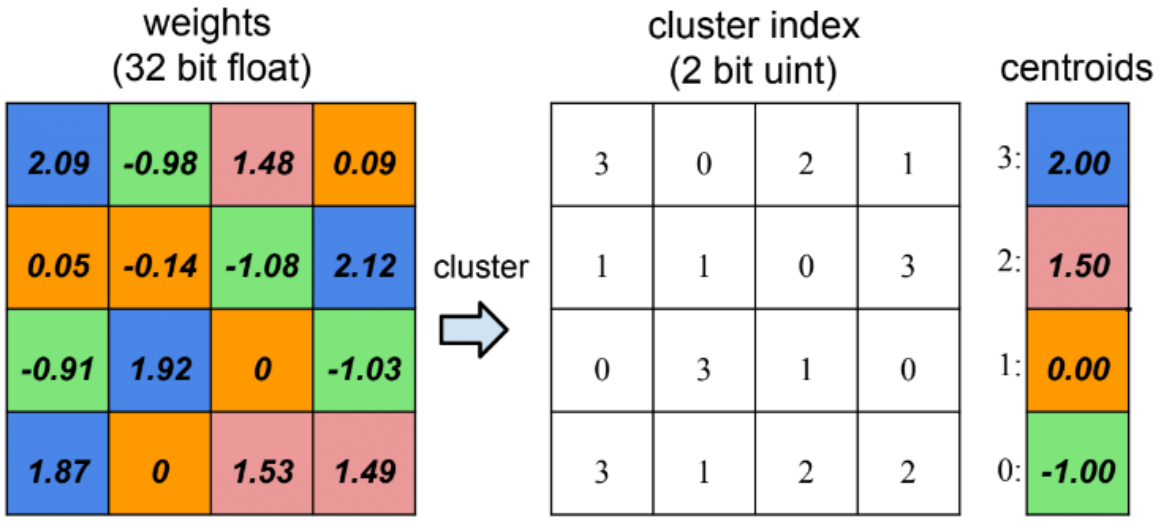


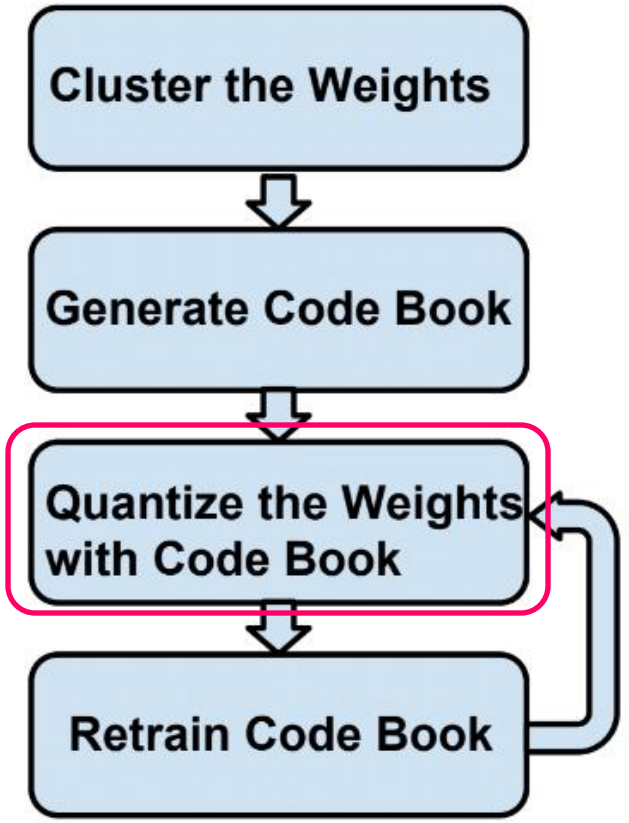
Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Weight sharing procedure

- Quantize
  - Replace the weight value with corresponding cluster index
  - Saving on N elements with 4 clusters: $\mathrm{N} \times 32\ bits \rightarrow N \times 2 + 4 \times 32\ bits$

Almost 16 times compression with large N



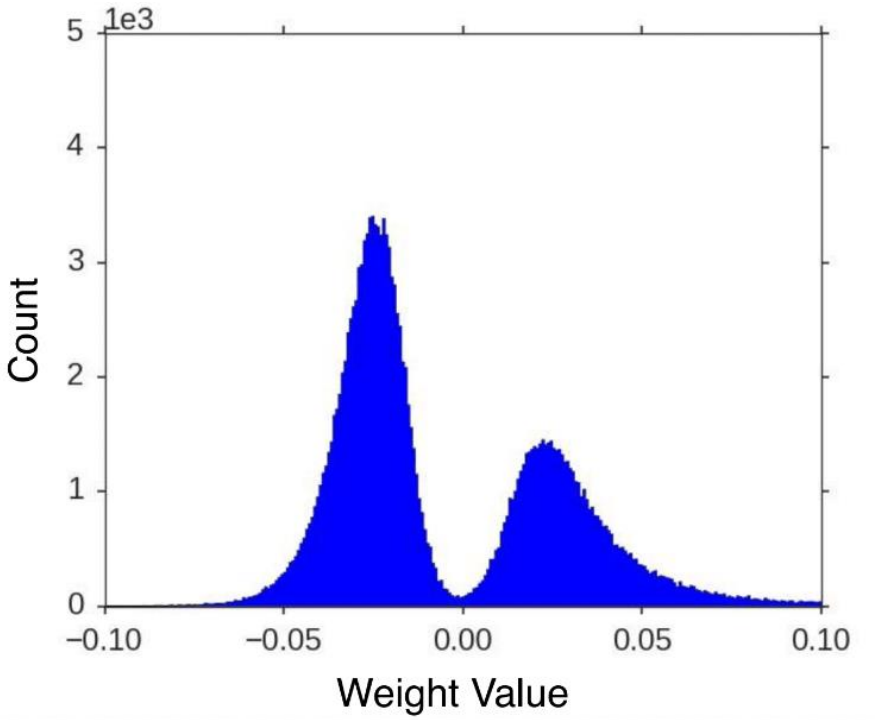


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

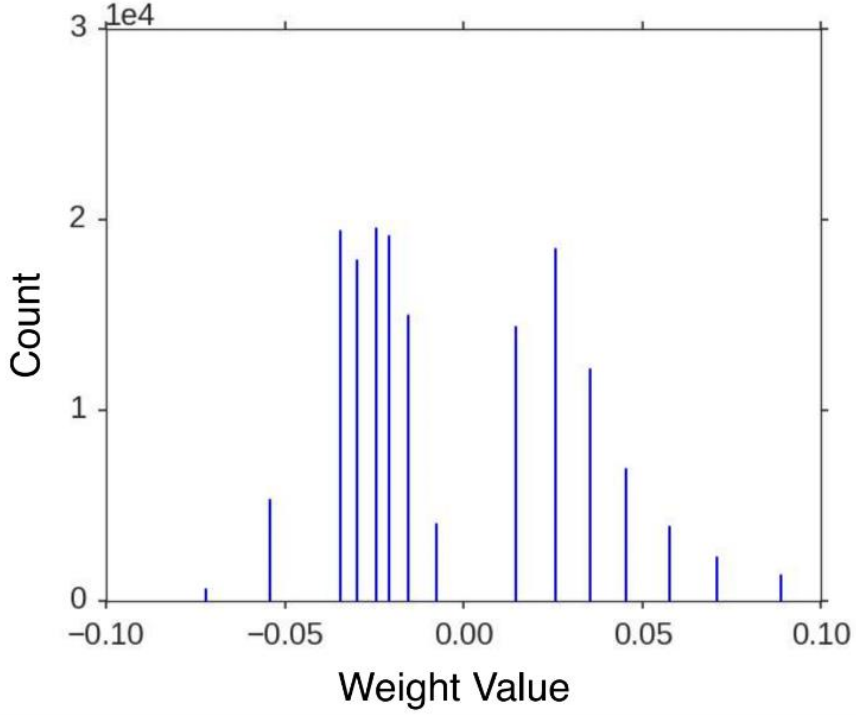
Weight sharing procedure

- Quantize
  - Convert continuous weights to limited amounts of discrete values

Before quantization



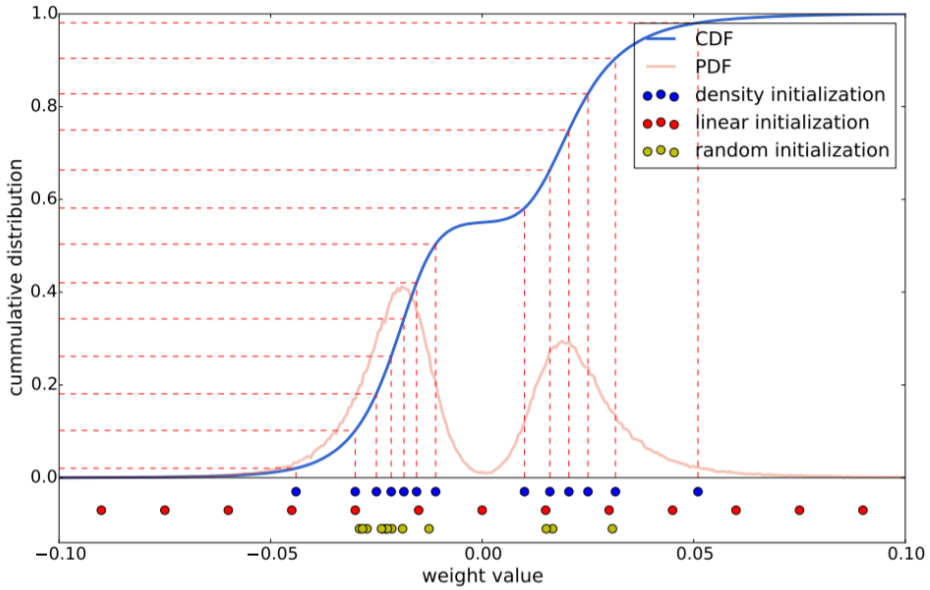
After quantization



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Centroid initialization

- Centroid initialization impacts the quality of clustering
- Random
  - Randomly choose k values as centroids (green dots)
- Density-based
  - Linearly space the y-axis of the CDF, and find the x value as centroids (blue dots)
- Linear
  - Linearly space in the range of original weights (red dots)

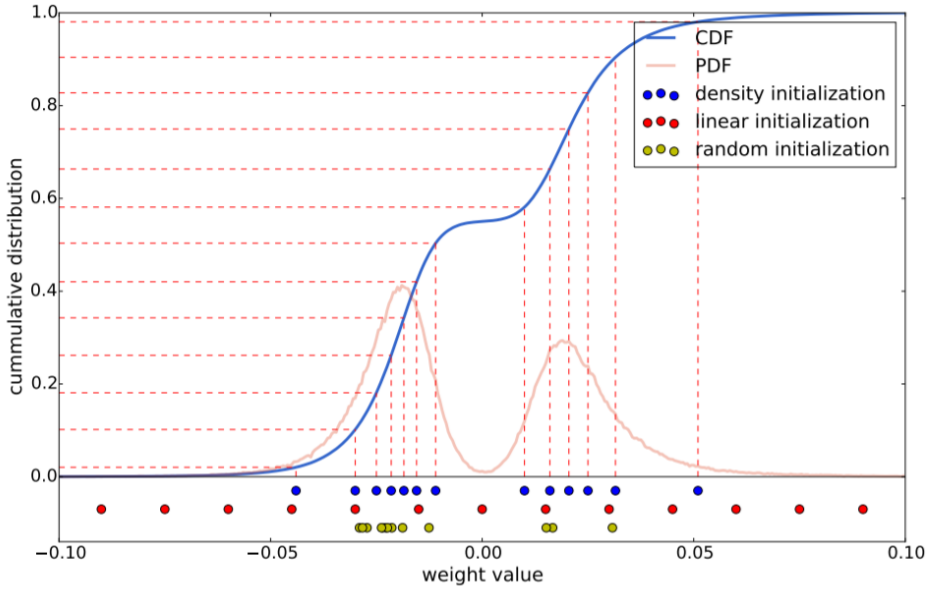


Which is better?
Make a guess!

Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Centroid initialization

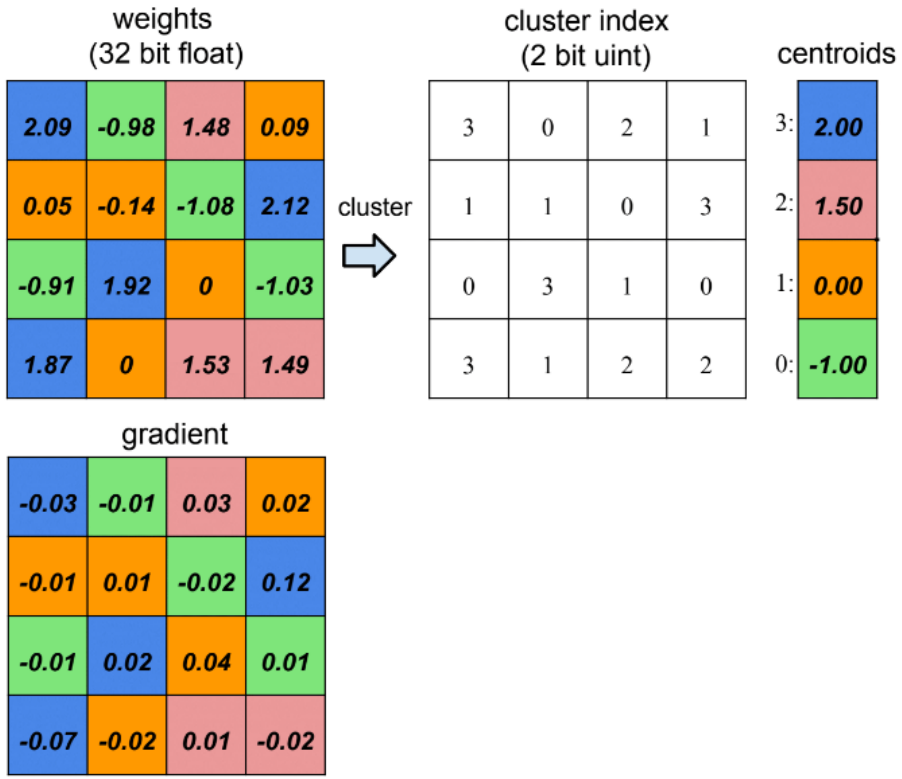
- Larger weights are more important!
- There are fewer large weights, so they will be represented poorly with random or density-based initialization
- **Linear initialization** is preferred for k-means weight clustering

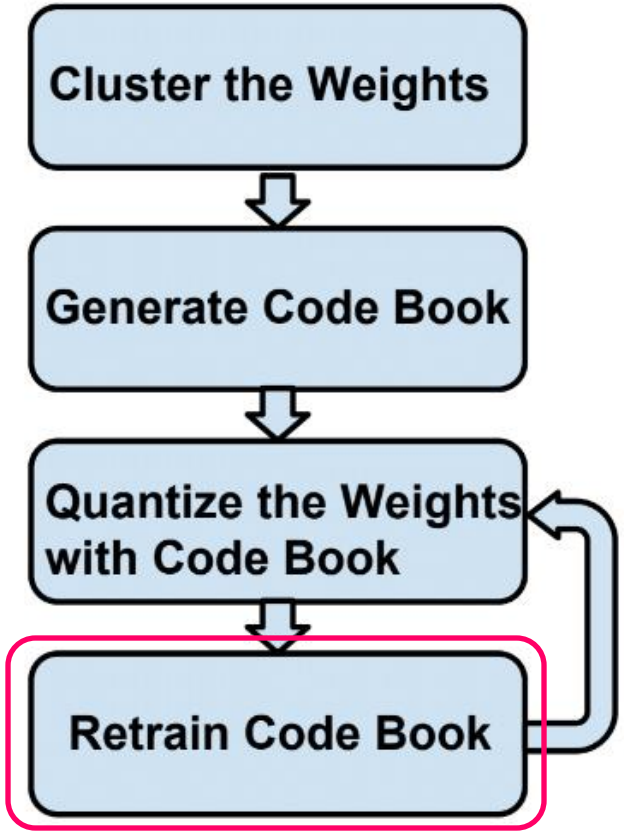


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Centroid finetuning

- Fix the cluster, update the centroids
- Gradient computation at all locations

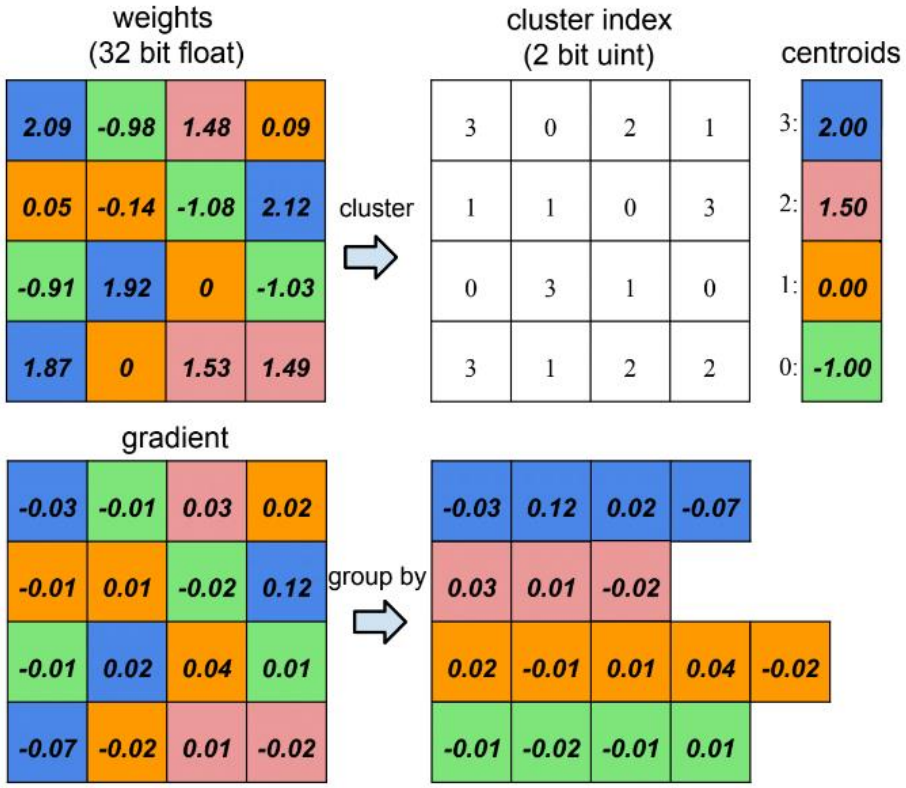


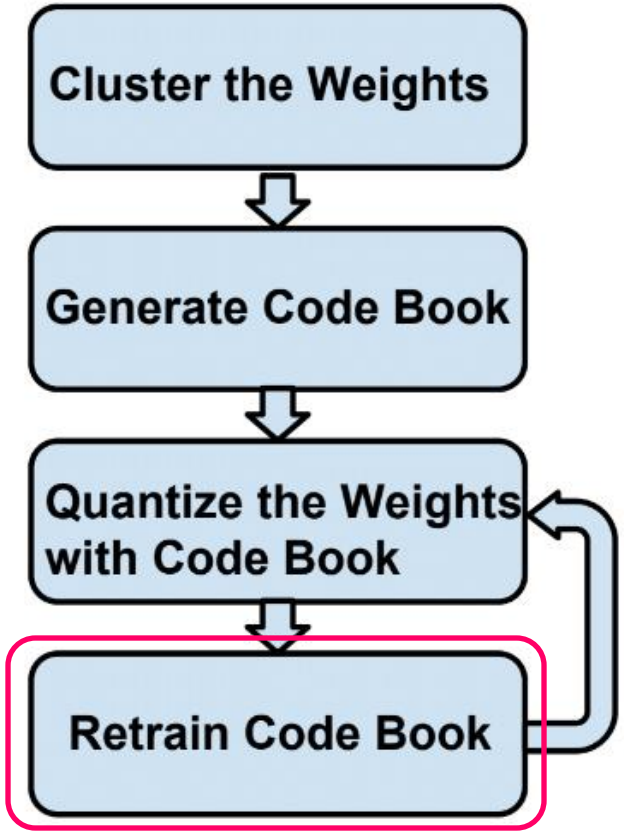


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Centroid finetuning

- Fix the cluster, update the centroids
- Cluster-based grouping

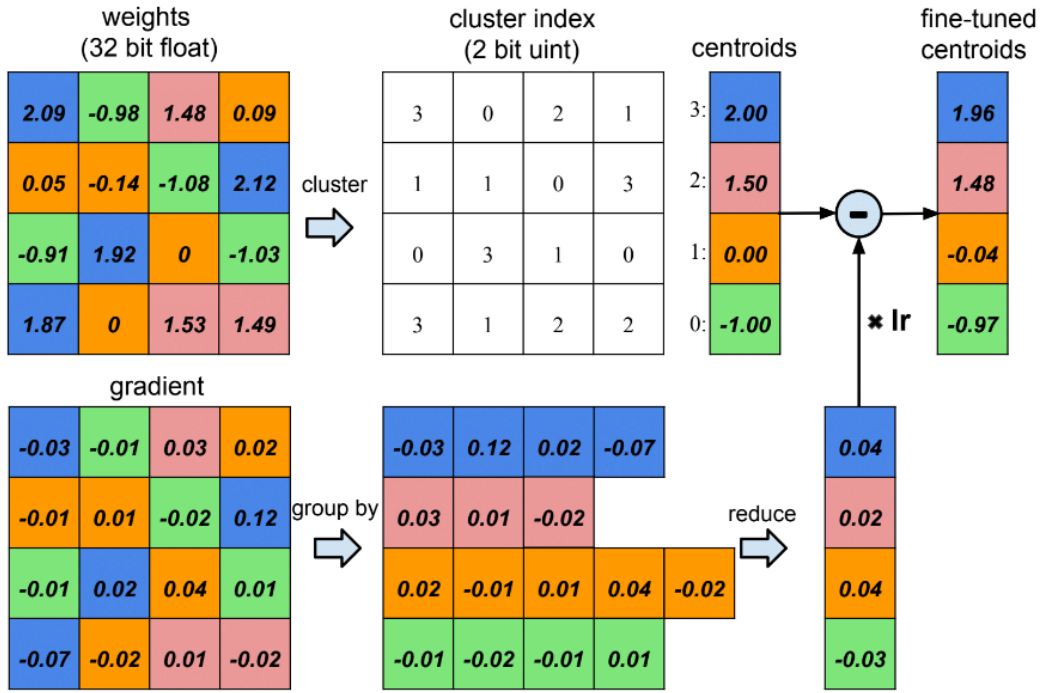




Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Centroid finetuning

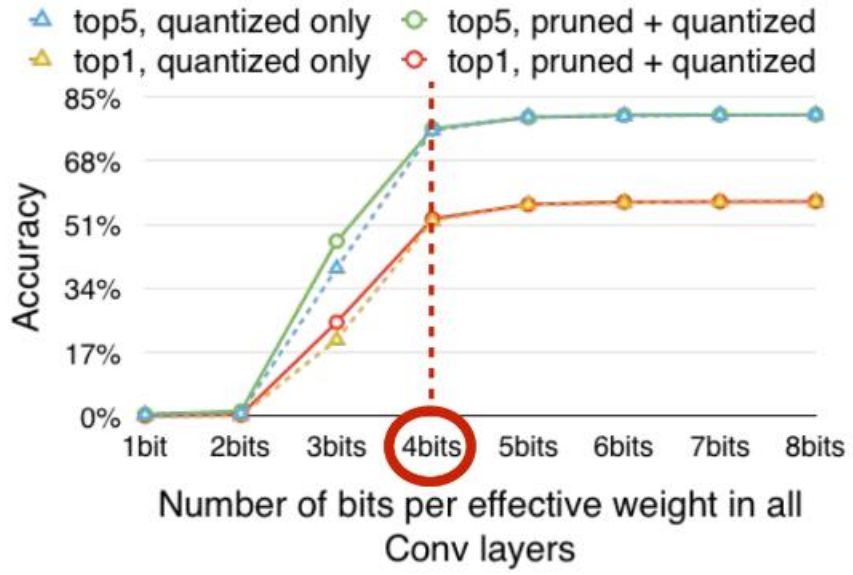
- Fix the cluster, update the centroids
- Sum and update
  - The gradient w.r.t. a centroid is the sum of all gradients w.r.t. the weights within that cluster

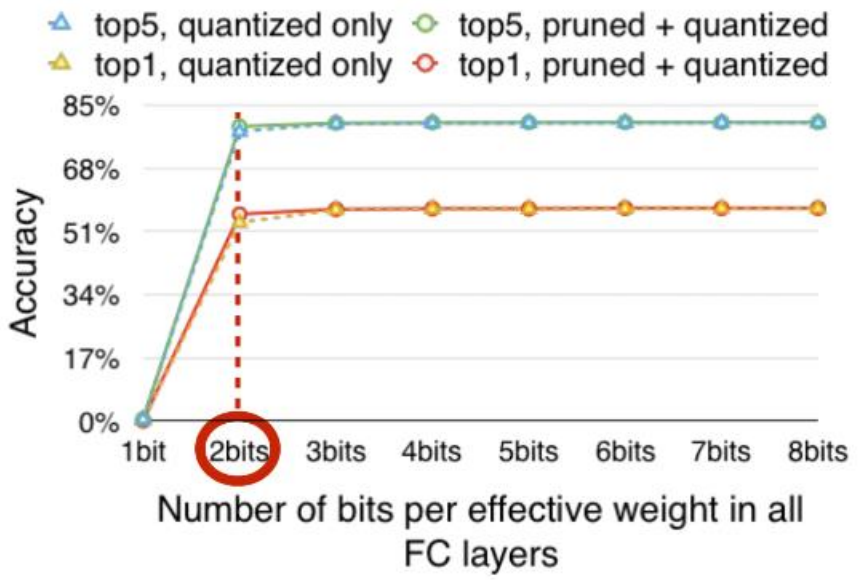


Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

How many bits are needed?

- 4 bits for conv layers, 2 bits for FC layers (AlexNet)
  - May be different on other models, generally FC layers requires fewer bits
- Pruning doesn't hurt quantization

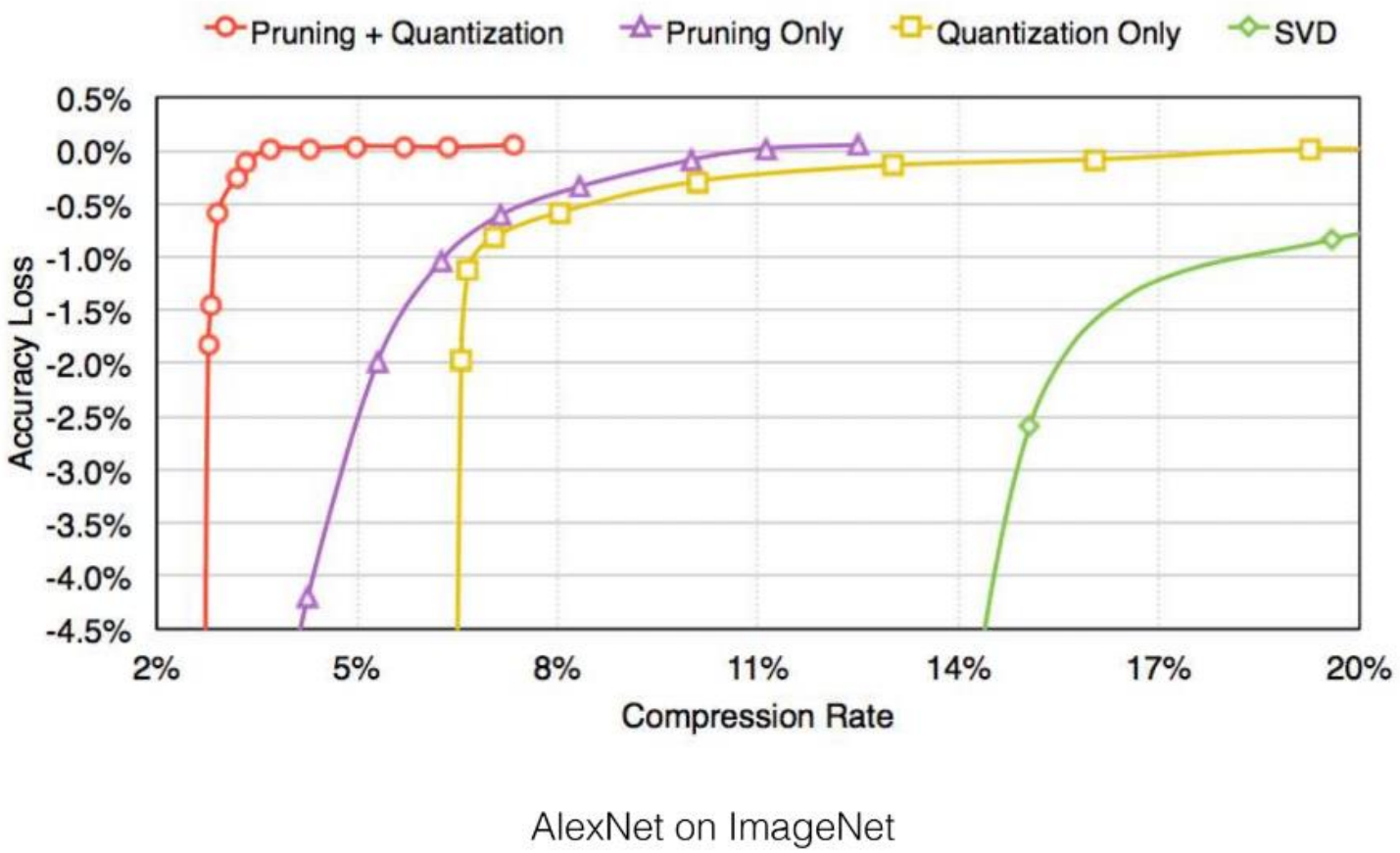




Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Combining pruning and quantization

- Further improving the compression rate

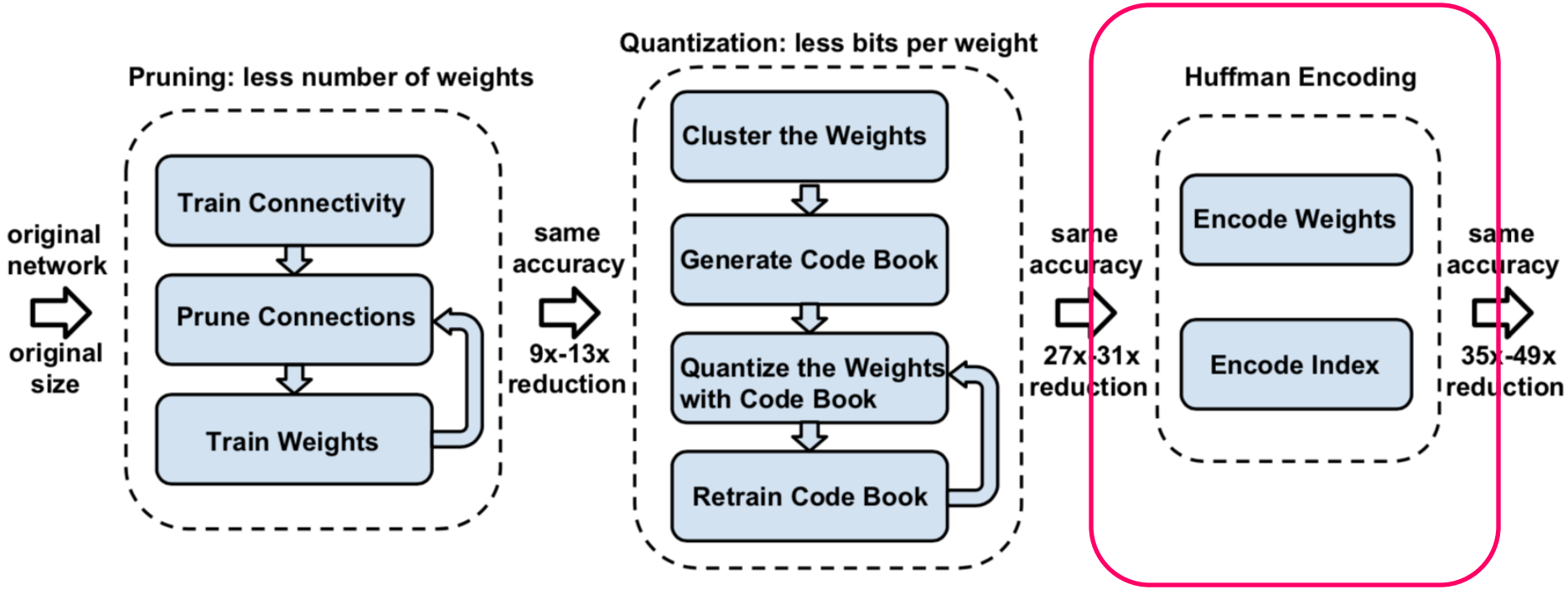


Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems*. 2015.

Efficient inference with weight sharing

- Weight sharing largely reduces the memory consumption of storing DNN
- Can it also help on inference efficiency?
- Since we only have limited weight values, we can save potential multiplication results in a look-up table (LUT), then just look up the result when inferencing
- Reading from a small LUT is more efficient than performing floating point multiplication

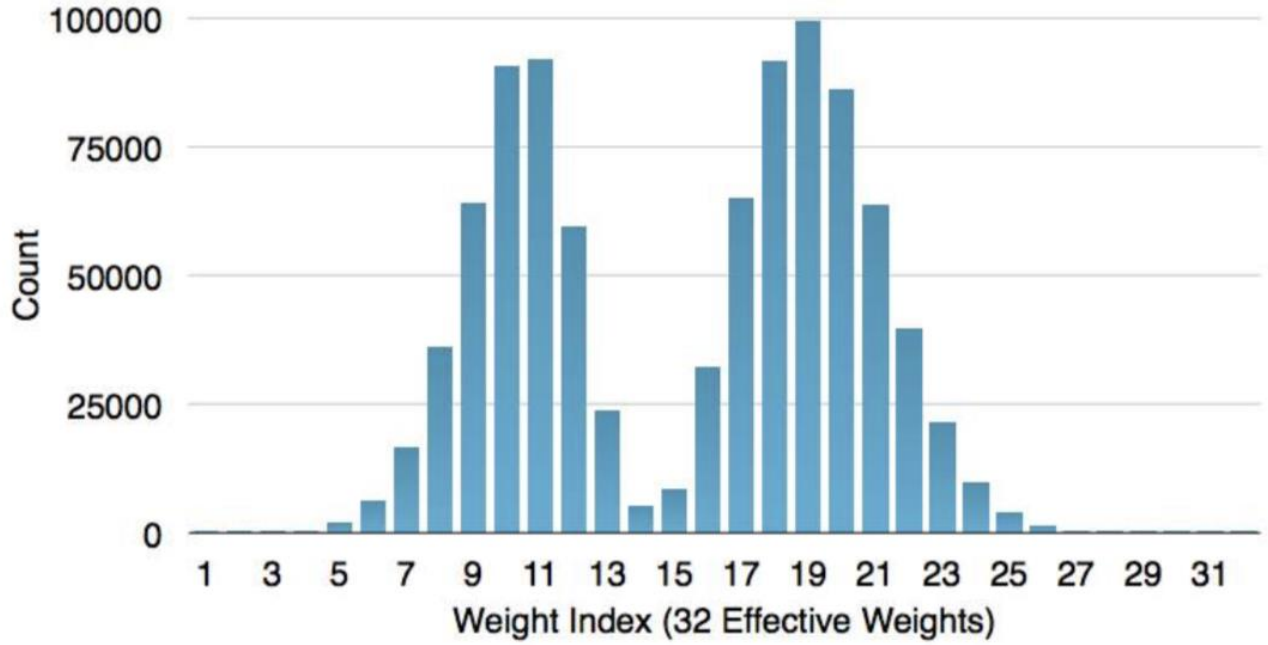
Deep Compression: Huffman encoding



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Encoding the weight index

- The weight index occurrence is unbalanced



- Would be efficient if we can use shorter code to represent frequent weights and longer for infrequent ones

Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Huffman code

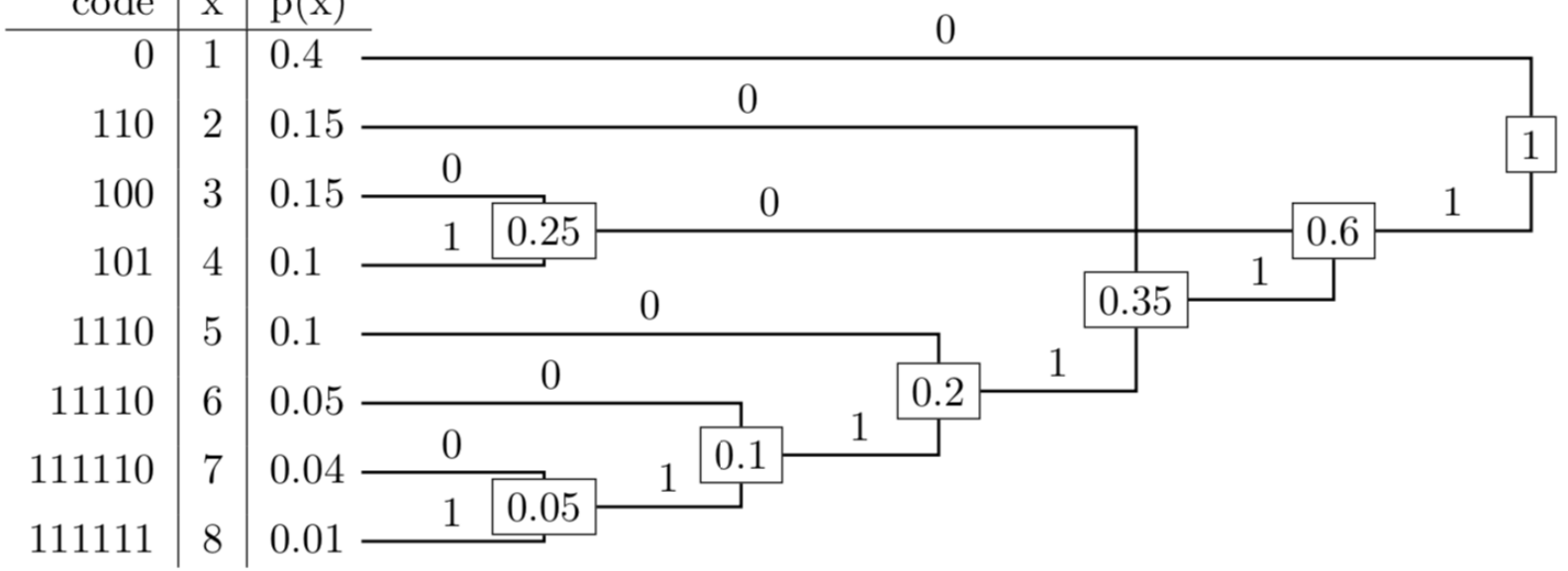
- A valid coding scheme should satisfy:
  - Nonsingular: different symbols have different codes
  - Uniquely decodable: any combination of different symbols have different concatenated code (counter example {0, 1, 01})
  - Prefix-free: no codeword is the prefix of another code (counter example {0, 01})
- Binary index (i.e. {00,01,10,11}) is valid, but not efficient for unbalanced distribution
- Huffman code is a valid coding scheme with the guaranteed shortest expected length for any symbol distribution

Huffman code

- Constructing a Huffman code
  - Codewords are assigned from the last bit
  - Take the two least probable symbols. These are assigned the longest codewords which have equal length and differ only in the last digit
  - Merge these two symbols into a new symbol with combined probability mass and repeat.

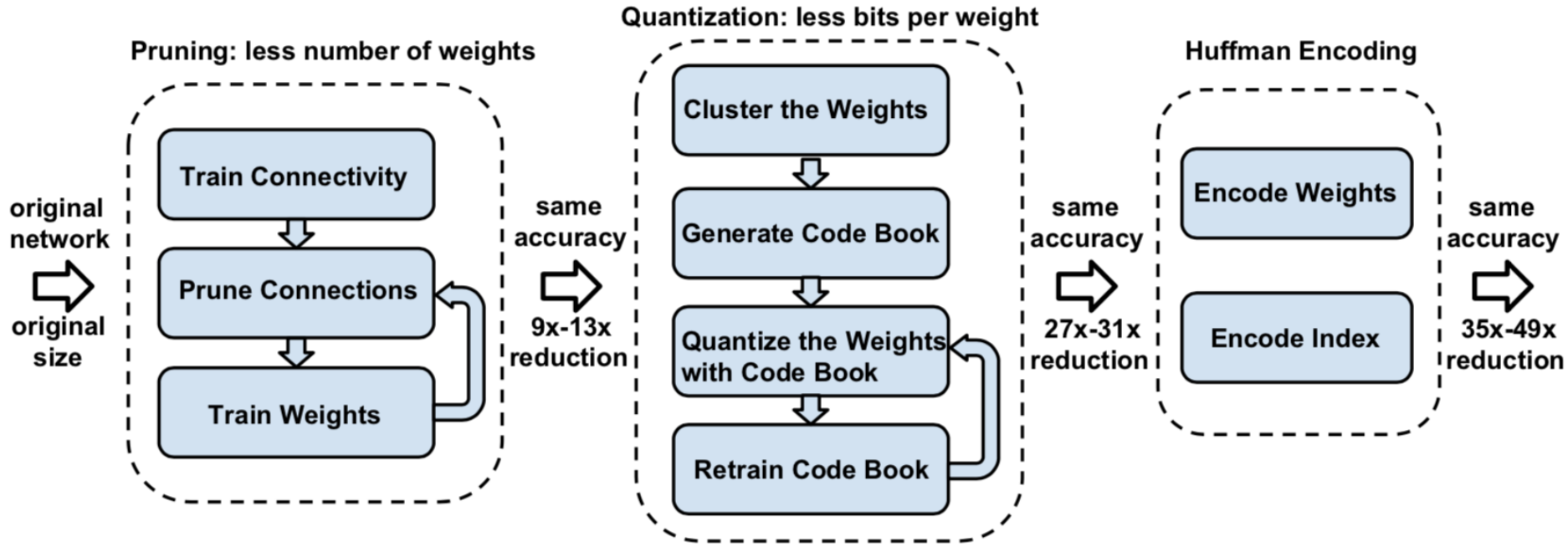
- **Example:** Consider the following source distribution.

code	x	p(x)
0	1	0.4
110	2	0.15
100	3	0.15
101	4	0.1
1110	5	0.1
11110	6	0.05
111110	7	0.04
111111	8	0.01



Example credit to Prof. Galen Reeves, check his ECE 587 course for more on information theory and coding

Summary of Deep Compression



- Achieving 35-49X memory reduction through the whole pipeline
- Requires specifically designed accelerator to fully utilize the benefits

Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

Reading Material

- Song Han. “EFFICIENT METHODS AND HARDWARE FOR DEEP LEARNING.”