# Lecture 9: Models Compression Techniques

AC215

Pavlos Protopapas
SEAS/Harvard

# Motivation

We want to process data (ideally a lot) and we do not have enough computing resources. For example:

1. Your phone can't run GoogleNet to assist you in some tasks

2. You can't compress enormous number of images coming from space (8Kx8K pixels from 3K satellites)

Using machine learning is resource intensive:

i. Computing power to train millions of parameters or predict for many observations

ii. Limited bandwidth

**So what?** Model compression techniques

Hannah Peterson and George Williams, *An Overview of Model Compression Techniques for Deep Learning in Space*, August 2020

# What is Model Compression?

The main idea is to simplify the model without diminishing accuracy. A simplified model means reduced in size and/or latency from the original. Both types of reduction are desirable.

- Size reduction can be achieved by reducing the model parameters and thus using less RAM.
- Latency reduction can be achieved by decreasing the time it takes for the model to make a prediction, and thus lowering energy consumption at runtime (and carbon footprint).

Karen Hao, *Training a single AI model can emit as much carbon as five cars in their lifetimes*, June 2019

# Compression Techniques

- Knowledge distillation
- Pruning
- Quantization
- {Low-rank approximation and sparsity}

# Compression Techniques

- **Knowledge distillation**
- Pruning
- Quantization
- {Low-rank approximation and sparsity}

# Compression Technique: Distillation

# Compression Technique: Distillation

# Compression Technique: Distillation

## Problem:

- During training, a model does not have to operate in real time and does not necessarily face restrictions on computational resources, as its primary goal is simply to extract as much structure from the given data as possible.
- But latency and resource consumption do become of concern if it is to be deployed for inference.

**So what?** we must develop ways to compress model for inference.

# Compression Technique: Distillation

## Idea:

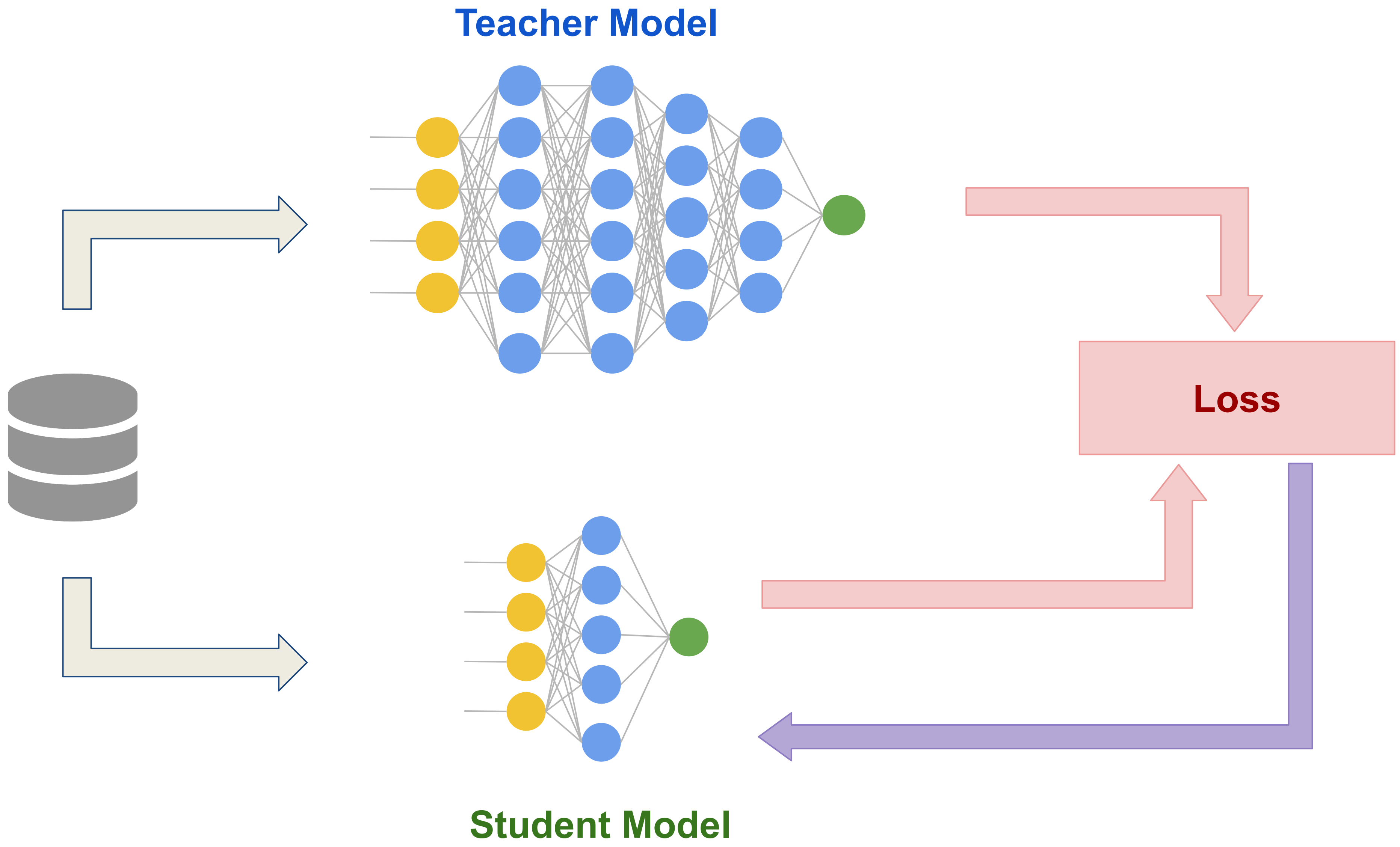
- In 2006, Buciluă et al. showed that it was possible to transfer knowledge from a large trained model (or ensemble of models) to a smaller model for deployment by **training it to mimic the larger model's output.**

- In 2014 Hinton et al generalized the process and gave the name **Distillation.**

Main idea of distillation is that **training and inference are 2 different tasks;** thus **a different model should be used**.

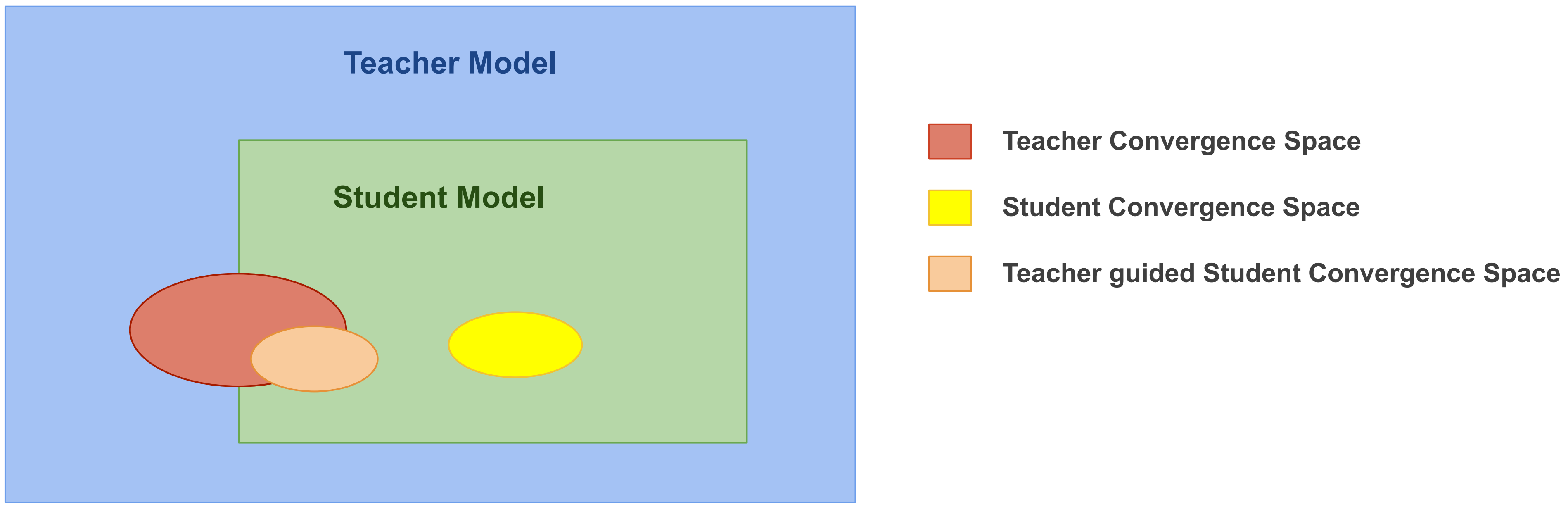
Buciluă et al., *Model Compression*, 2006
Hinton et al., *Distilling the Knowledge in a Neural Network*, 2014

# Distillation: Teacher Student



# Distillation: Teacher Student

**Assumption:** if we can achieve similar convergence using a smaller network, then the convergence space of the Teacher Network should overlap with the solution space of the Student Network.



# Distillation: Teacher Student Loss

Modified softmax function with Temperature:

$$q_i = \frac{\exp \frac{z_i}{T}}{\sum_j \exp \frac{z_j}{T}}$$

**An example of hard and soft targets**

| cow | dog | cat | car | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | original hard targets |
| $10^{-6}$ | .9 | .1 | $10^{-9}$ | output of geometric ensemble |
| .05 | .3 | .2 | .005 | softened output of ensemble |

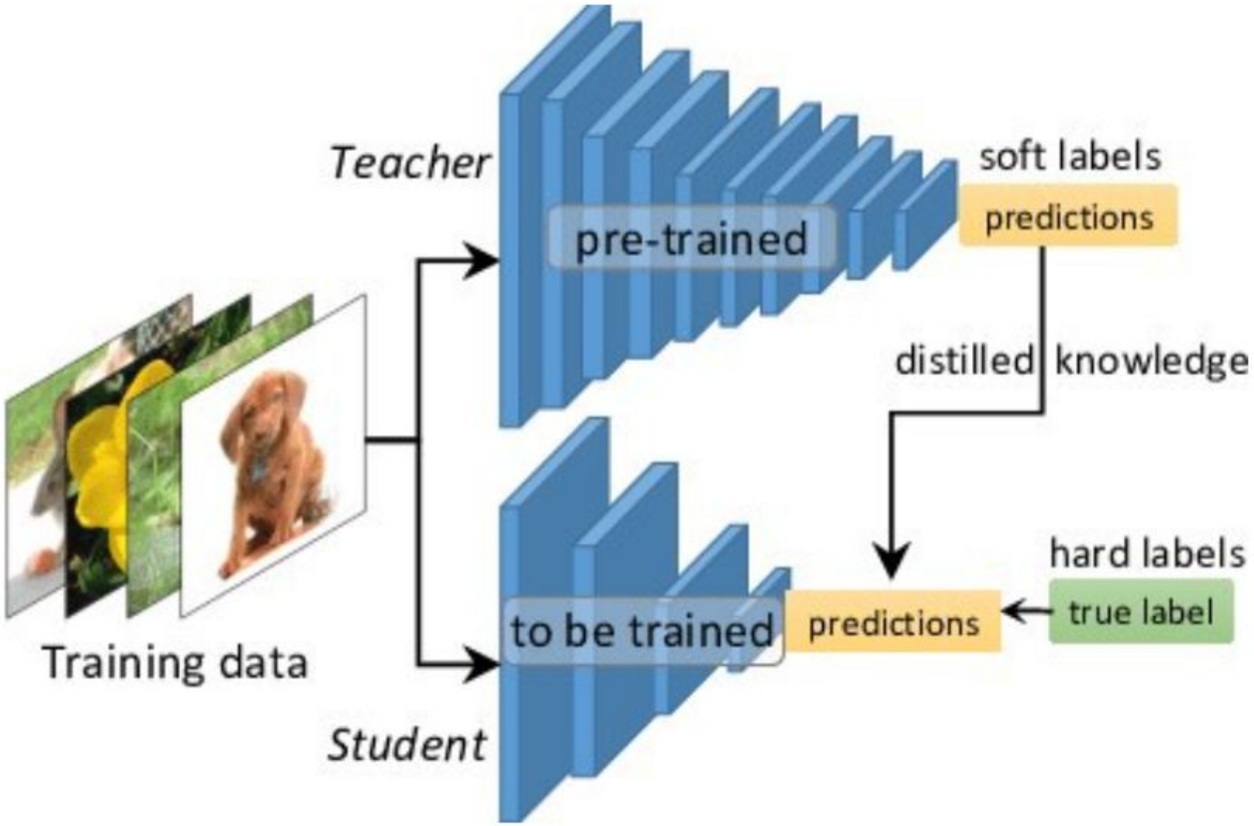
Softened outputs reveal the dark knowledge in the ensemble.

Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

# Distillation: Teacher Student Training

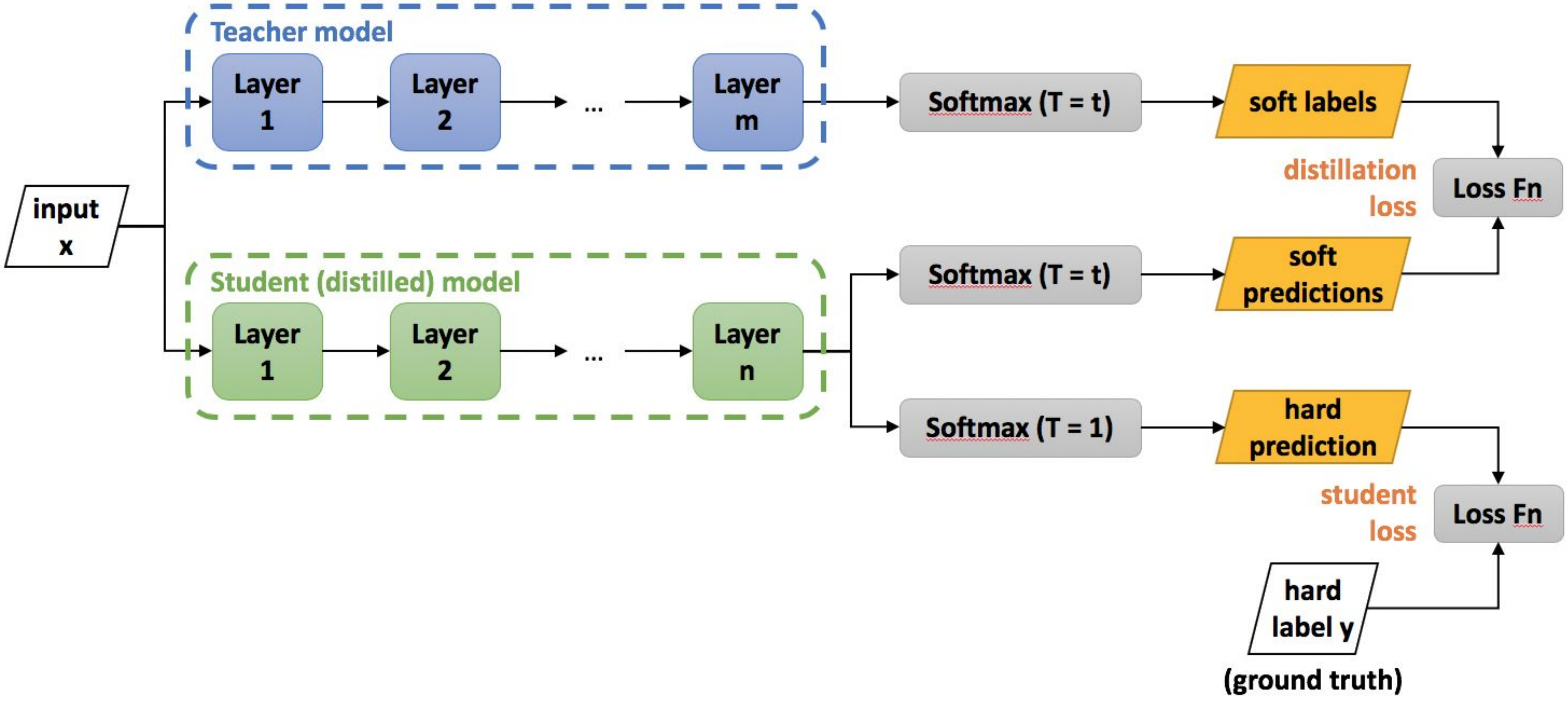
Trained to minimize the sum of two different cross entropy functions:

- one involving the original hard labels obtained using a softmax with $T$=1
- one involving the softened targets, T>1




Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

# Distillation: Teacher Student Training




$$L = \lambda L_{\text{student}} + (1 - \lambda) L_{\text{distillation}}$$

# Tutorial: Model Distillation

[Colab Notebook]

# What is next in Distillation?

**1:** Multiple teachers (i.e. converting an ensemble into a single network).

**2:** Introducing a teaching assistant (the teacher first teaches the TA, who then in turn teaches the student) etc.

**3:** Quite young field

A **drawback** of knowledge distillation as a compression technique, therefore, is that there are many decisions that must be made up-front by the user to implement it (student network doesn't even need to have a similar structure to the teacher).

# Compression Techniques

- Knowledge distillation
- **Pruning**
- Quantization
- {Low-rank approximation and sparsity}

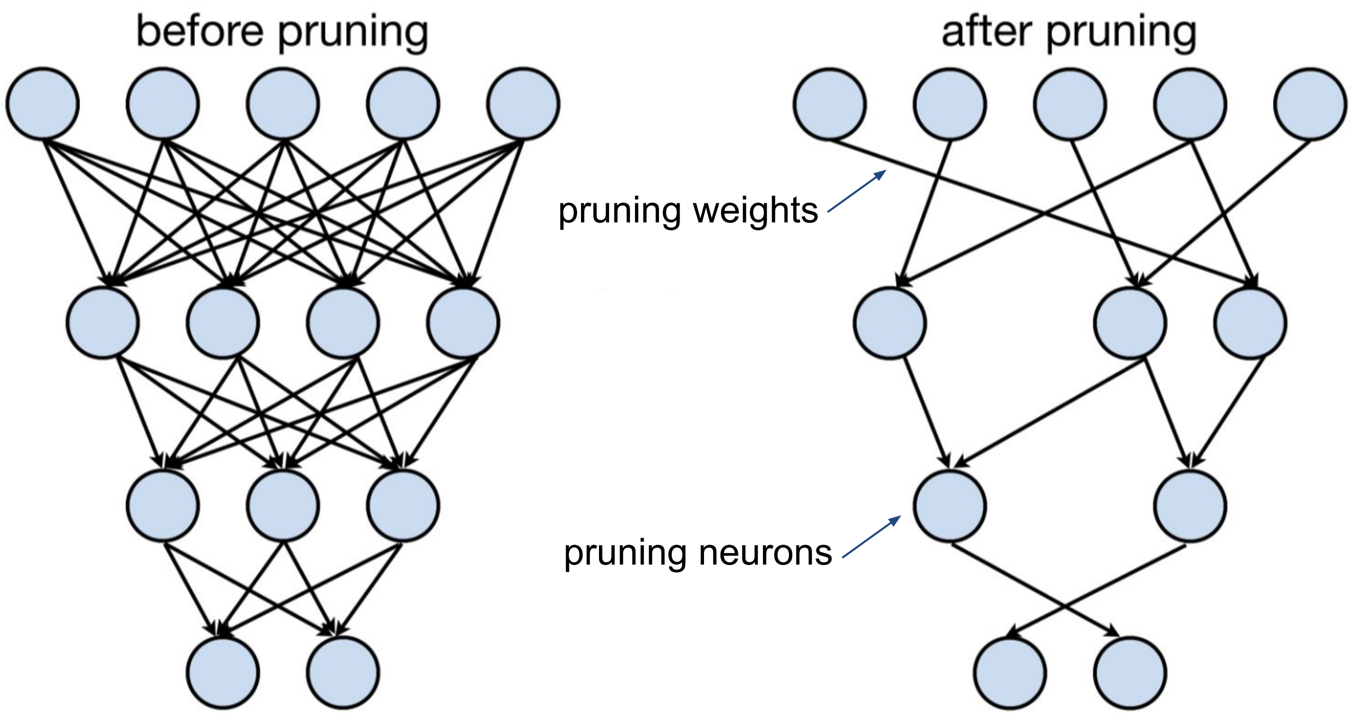
# Compression Technique: Pruning

The main idea is to remove less impactful features of the neural network.

Pruning involves removing connections between neurons, and neurons from a trained network. To prune a connection, we set a weight in the matrix to zero.

**Two types of pruning:**

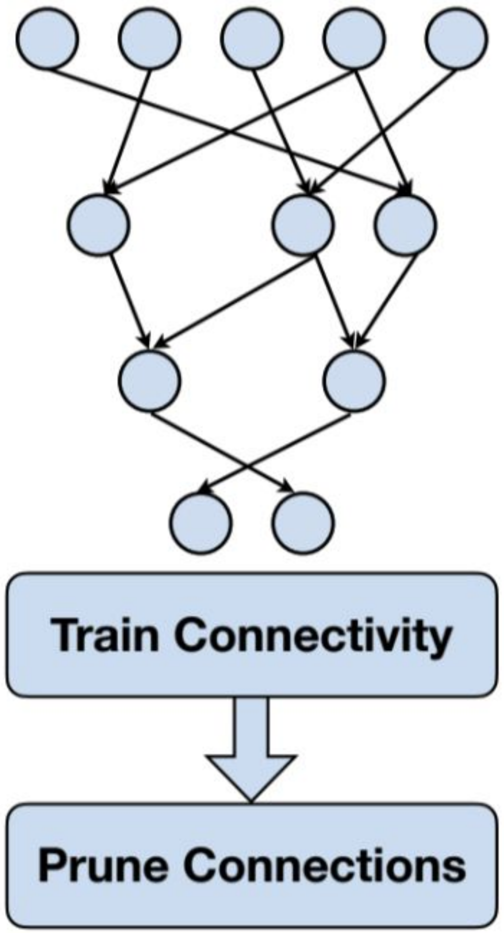
- Pruning weights
- Pruning neurons

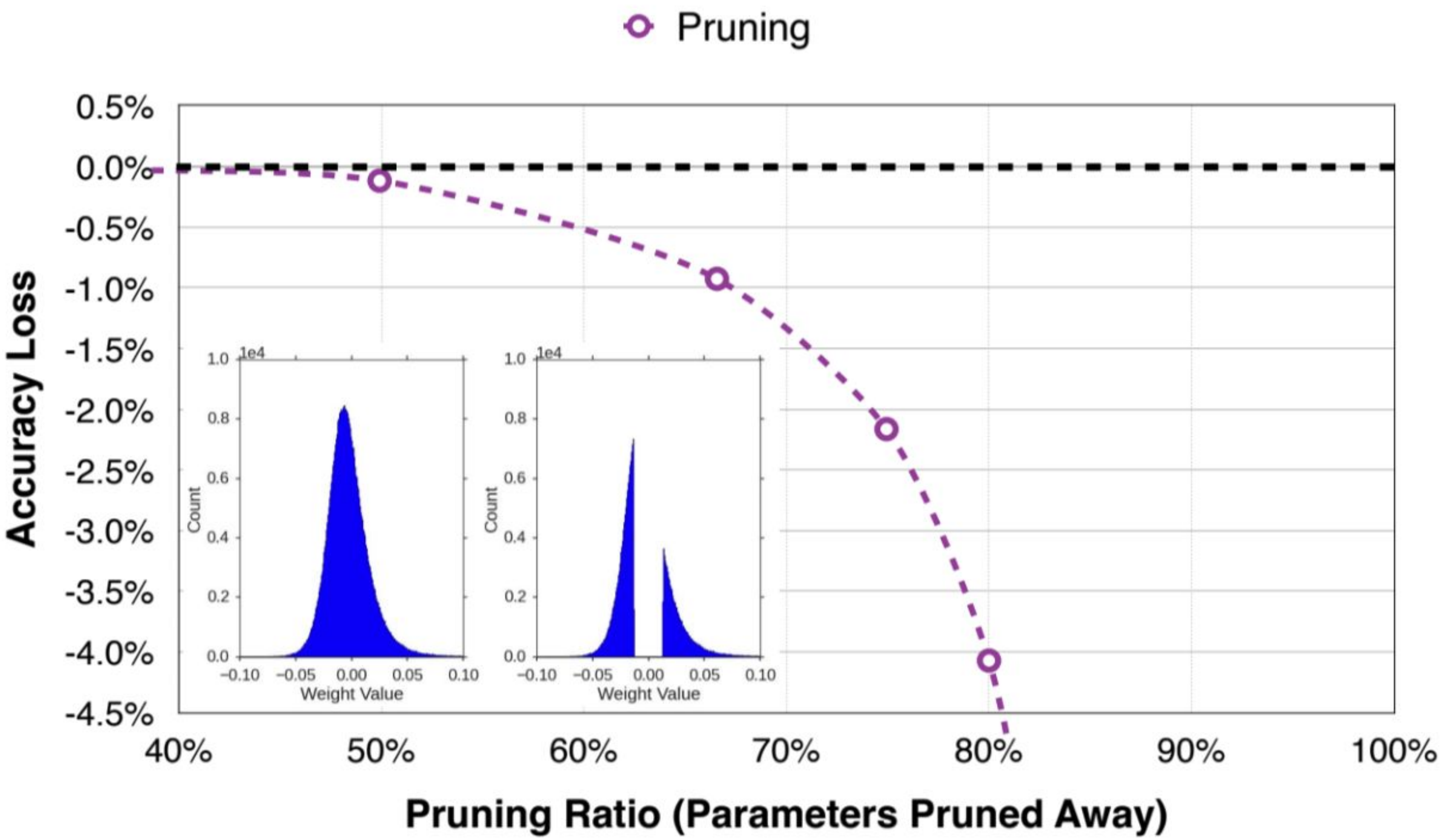



Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]

# Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons

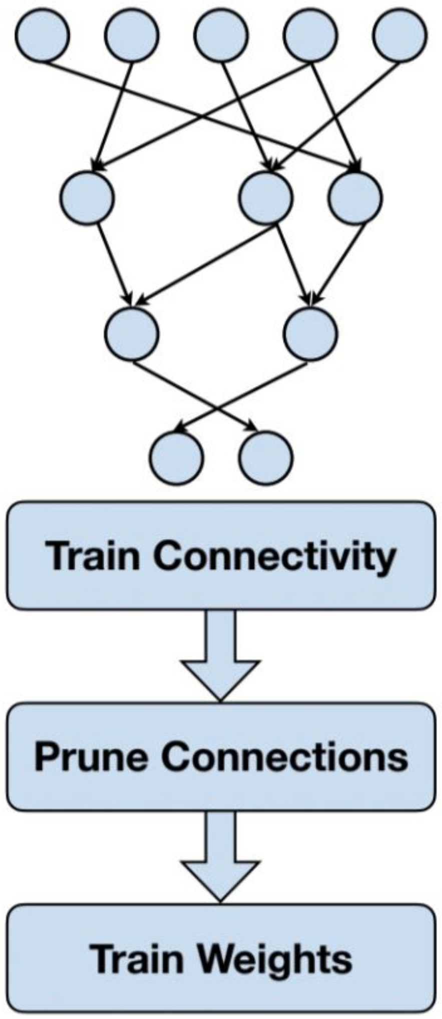


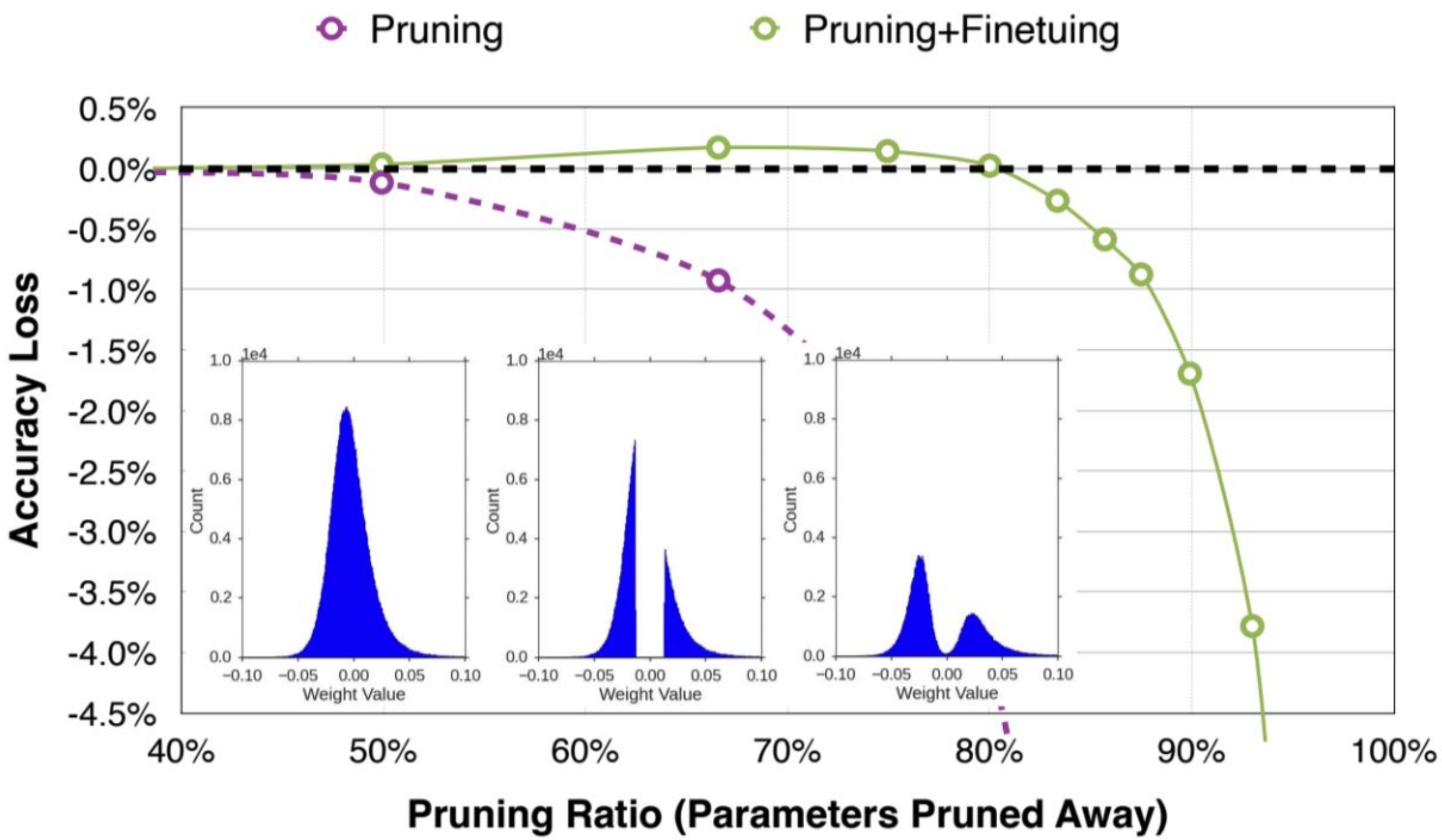




Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

# Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons

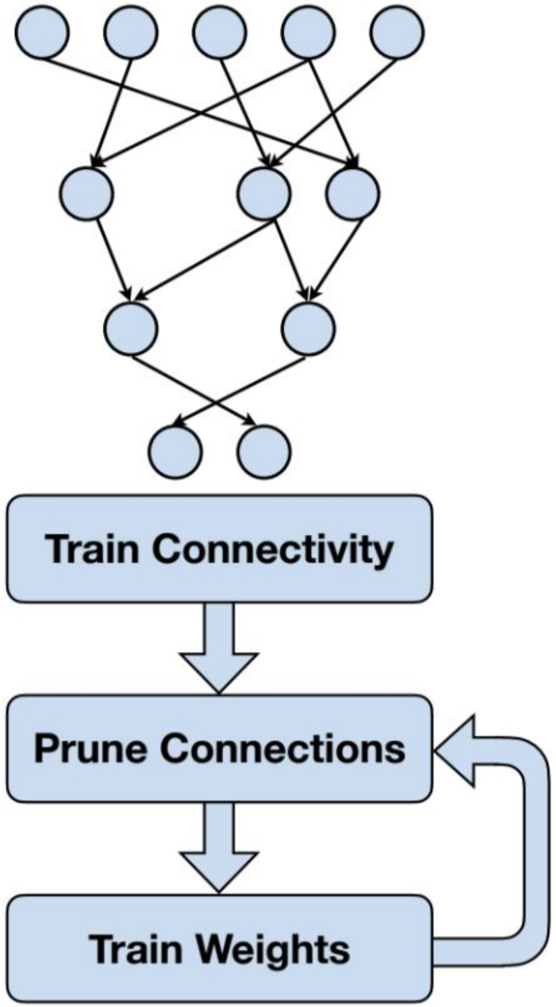


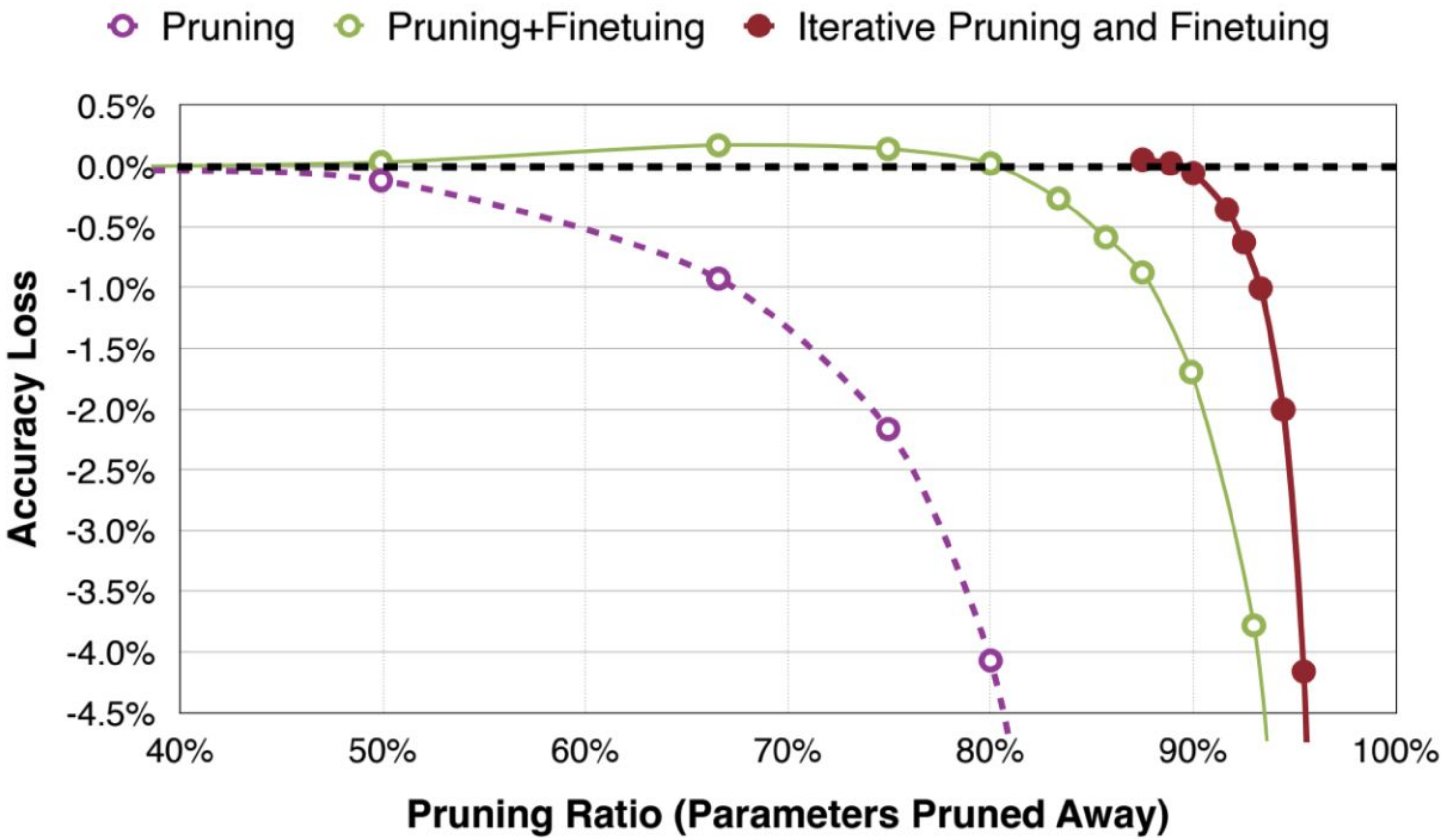




Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

# Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons



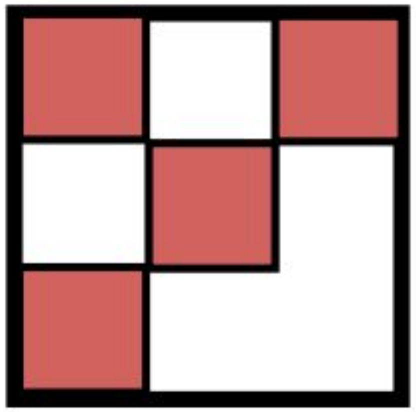




Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

# Compression Technique: Pruning

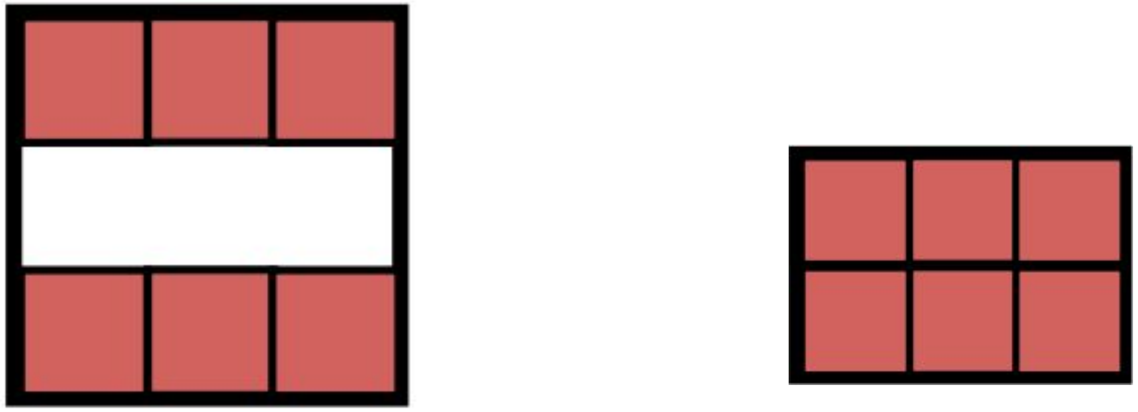
## Pruning Granularities

## Simple 2-D weight matrix example



**Fine-grained/Unstructured**

- More flexible pruning index choice
- Hard to accelerate (hardware limitations)



**Coarse-grained/Structured**

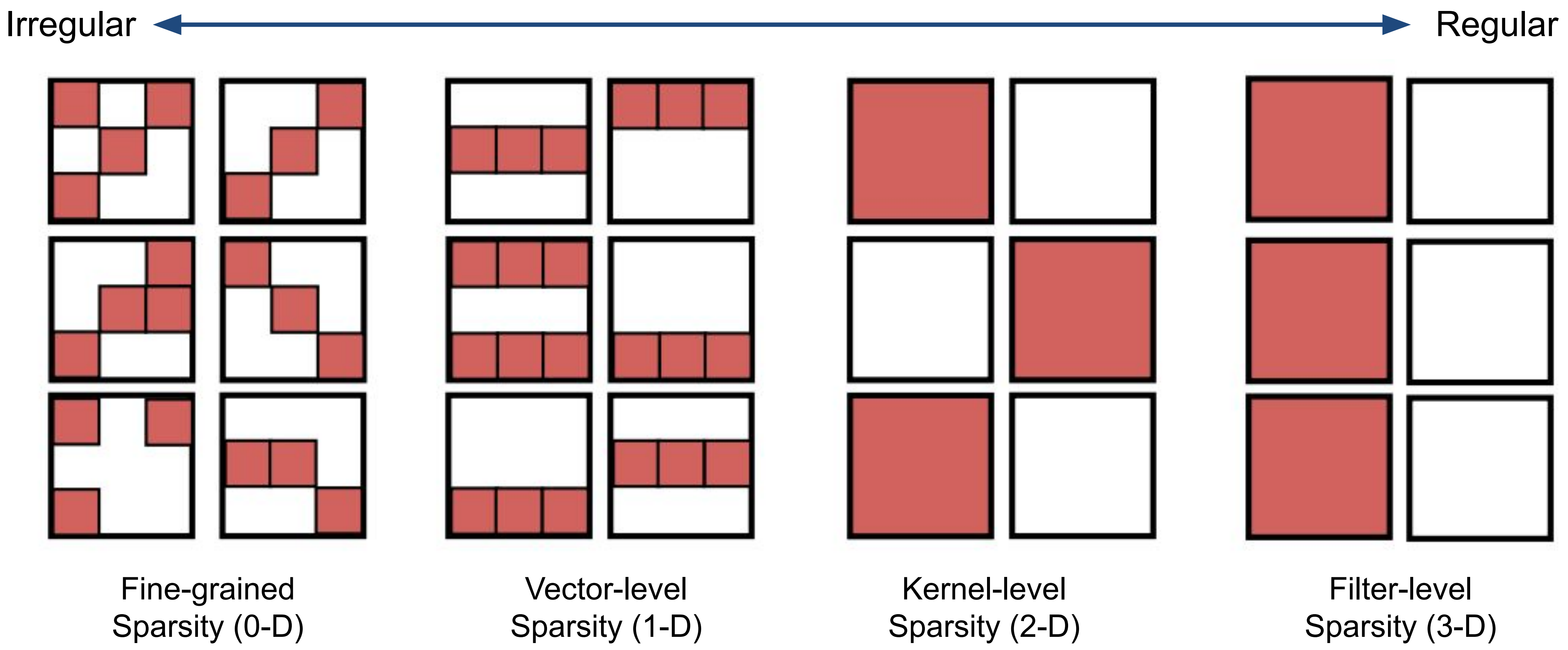
- Less flexible pruning idex choice (a subset of the fine-grained case)
- Easy to accelerate (just a smaller matrix!)

Exploring the Regularity of Sparse Structure in Convolutional Neural Networks
[Mao et al., CVPR-W]

# Compression Technique: Pruning

## Example using convolutional layers

- Commonly used pruning granularities




Exploring the Regularity of Sparse Structure in Convolutional Neural Networks
[Mao et al., CVPR-W]

# Compression Technique: Pruning

**Drawbacks** of neural network pruning:

- **Optimal Pruning Challenge**: determining the optimal neurons or weights to prune without detrimentally impacting model performance can be complex and time consuming in practise.
- **Fine-Tuning Requirement**: after pruning, models typically require additional fine-tuning to recover potential losses in predictive accuracy, which might consume additional training resources and time.
- **Hardware Dependency**: pruned models might not always translate to proportional computational or energy savings due to hardware inefficiencies or dependencies in exploiting sparsity.
- **Model Robustness**: excessive or imprecise pruning may lead to a substantial decrease in model accuracy or robustness, especially when encountering unseen or out-of-distribution data.

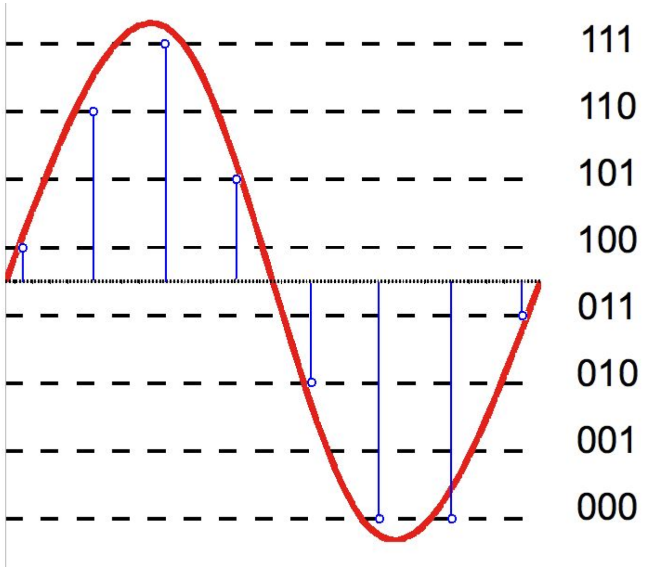
# Tutorial: Model Pruning

[Colab Notebook]

# Model Compression Technique: Quantization

Main idea is to map values from a continuous or a large discrete set to values in a smaller discrete set without losing too much information in the process.

Reducing a continuous signal, while maintaining the signal information.




Reducing the number of pixels, while maintaining the image content.

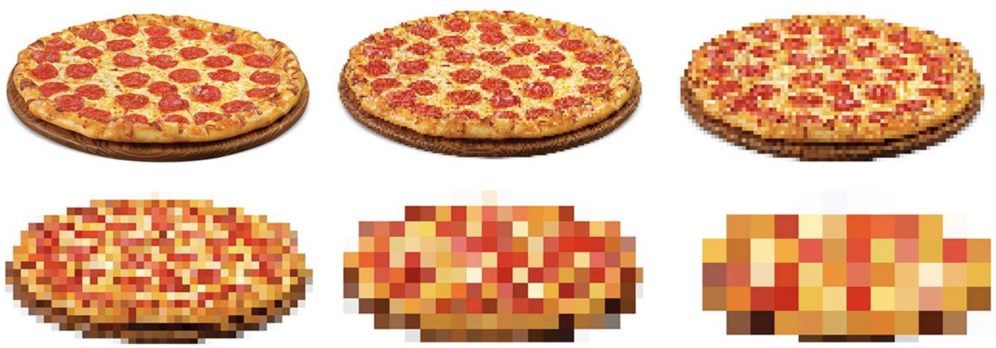


Image is in the public domain. “Analog to digital converter”

To implement quantization with Tensorflow: MC.AI, *Quantization in Deep Learning using TensorFlow 2.X*, May 2020

# Model Compression Technique: Quantization

## Size of a Deep Neural Network (in bytes)

deep_nn_size ~ num_parameters * parameter_size

num_parameters = total number of weights and biases of NN
parameter_size = size of parameter in bytes (e.g. 4 bytes for float32)

## Bottlenecks in Training and Inference of Deep Neural Networks

Memory Limitations: Memory demands for training large models, storing parameters, and handling intermediary computations challenge available GPU memory, restricting model complexity, and training efficiency.

Data Transfer Bottlenecks: Substantial data movement between storage (e.g., disks), CPUs, and GPUs, generates bottlenecks that slow down training and make real-time processing strenuous.

I/O Constraints: Input/output operations, involving reading and preprocessing data, often become bottlenecks, affecting GPU utilization and elongating training times, especially with voluminous and complex datasets.

# Neural Networks Quantization

## Weight Quantization Techniques

- **K-Means-based Quantization**

| | | | |
|---|---|---|---|
| 2.09 | -0.98 | 1.48 | 0.09 |
| 0.05 | -0.14 | -1.08 | 2.12 |
| -0.91 | 1.92 | 0 | -1.03 |
| 1.87 | 0 | 1.53 | 1.49 |

| | | | |
|---|---|---|---|
| 3 | 0 | 2 | 1 |
| 1 | 1 | 0 | 3 |
| 0 | 3 | 1 | 0 |
| 3 | 1 | 2 | 2 |

| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 1: | 0.00 |
| 0: | -1.00 |

| | **Full Model** | **K-Means Quantized Model** |
|---|---|---|
| **Storage** | Floating-Point Weights | Integer Weights; Floating-Point Codebook |
| **Computation** | Floating-Point Arithmetic | Floating-Point Arithmetic |

- Linear Quantization
- Binary/Ternary Quantization

Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Neural Networks Quantization

## Weight Quantization

weights 32-bit float

| | | | |
|---|---|---|---|
| **2.09** | **-0.98** | **1.48** | **0.09** |
| **0.05** | **-0.14** | **-1.08** | **2.12** |
| **-0.91** | **1.92** | **0** | **-1.03** |
| **1.87** | **0** | **1.53** | **1.49** |

~~2.09, 2.12, 1.92, 1.87~~

2.0

Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Neural Networks Quantization

## K-Means-based Weight Quantization

weights 32-bit float

| | | | |
|---|---|---|---|
| 2.09 | -0.98 | 1.48 | 0.09 |
| 0.05 | -0.14 | -1.08 | 2.12 |
| -0.91 | 1.92 | 0 | -1.03 |
| 1.87 | 0 | 1.53 | 1.49 |

Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Neural Networks Quantization

## K-Means-based Weight Quantization

weights (32-bit float)

| | | | |
|---|---|---|---|
| 2.09 | -0.98 | 1.48 | 0.09 |
| 0.05 | -0.14 | -1.08 | 2.12 |
| -0.91 | 1.92 | 0 | -1.03 |
| 1.87 | 0 | 1.53 | 1.49 |

→ cluster

cluster index (2-bit int)

| | | | |
|---|---|---|---|
| 3 | 0 | 2 | 1 |
| 1 | 1 | 0 | 3 |
| 0 | 3 | 1 | 0 |
| 3 | 1 | 2 | 2 |

centroids

| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 1: | 0.00 |
| 0: | -1.00 |

reconstructed weights (32-bit float)

| | | | |
|---|---|---|---|
| 2.00 | -1.00 | 1.50 | 0.00 |
| 0.00 | 0.00 | -1.00 | 2.00 |
| -1.00 | 2.00 | 0.00 | -1.00 |
| 2.00 | 0.00 | 1.50 | 1.50 |

quantization error

| | | | |
|---|---|---|---|
| 0.09 | 0.02 | 0.02 | 0.09 |
| 0.05 | 0.14 | 0.08 | 0.12 |
| 0.09 | 0.08 | 0 | 0.03 |
| 0.13 | 0 | 0.03 | 0.01 |

**storage**

32 bit * 16
= 512 bit = 64B

2 bit * 16
= 32 bit = 4B

+

32 bit * 4
=128 bit = 16B

= 20B

3.2 x smaller

Assume N-bit quantization, and num_parameters = M >> $2^N$.

32 bit * M
= 32M bit

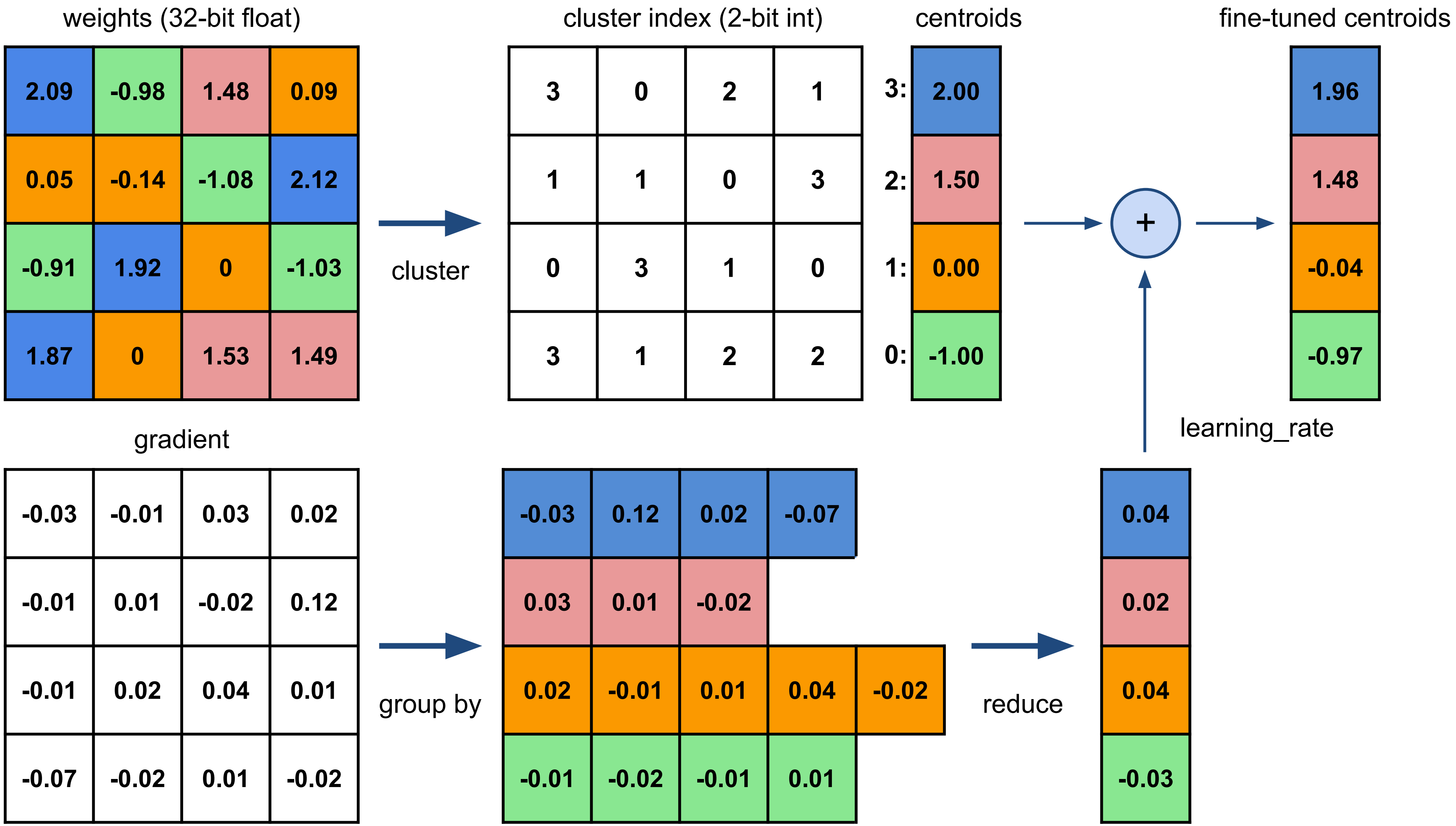
N bit * M
= NM bit

~~32 bit * $2^N$ = $2^{N+5}$ bit~~

32/N x smaller

# Neural Networks Quantization

## K-Means-based Weight Quantization



# Neural Networks Quantization

## K-Means-based Weight Quantization: Centroids Initialization

Centroid initialization impacts the quality of clustering and thus the network's prediction accuracy.

**Three types of initialization:**

Forgy (random): Randomly choose k observations from the data set and use these as initial centroids.
- Tends to concentrate around the highest mass of the weights' PDF

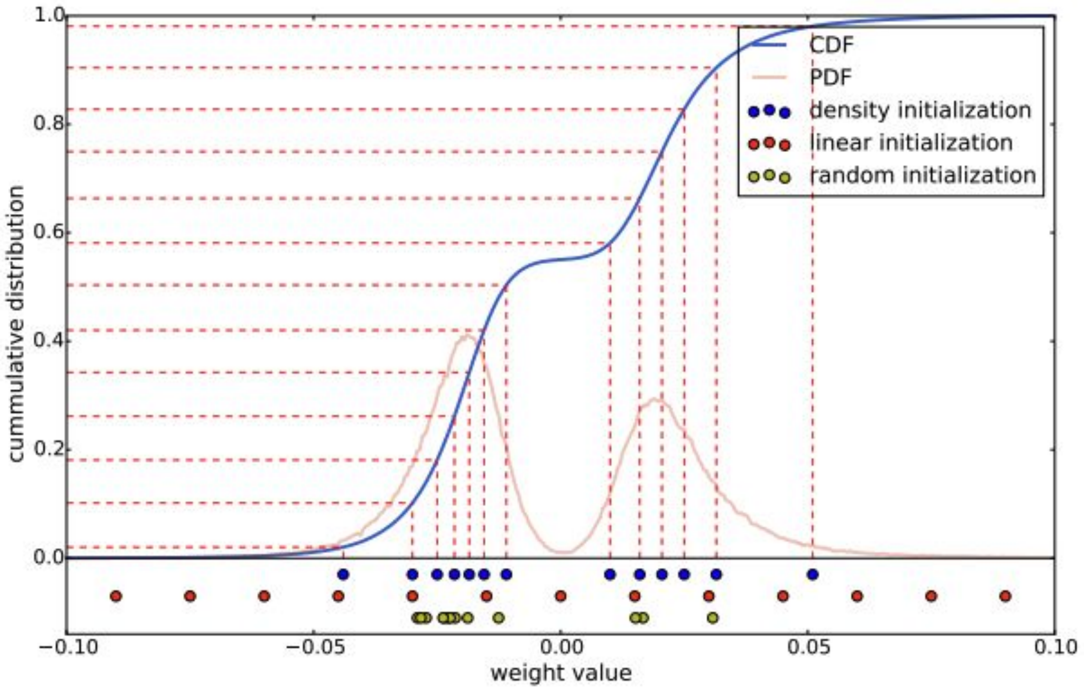
Density-based: Space points linearly on the range of CDF values [0, 1]. Then finds horizontal intersection with the CDF of the weights' distribution.
- Makes the centroids dense around the highest mass, but more scattered than Forgy

Linear: Space points linearly on the range of weights' values [min_weight_val, max_weight_val].
- This method is invariant to the distribution of the weights - most scattered

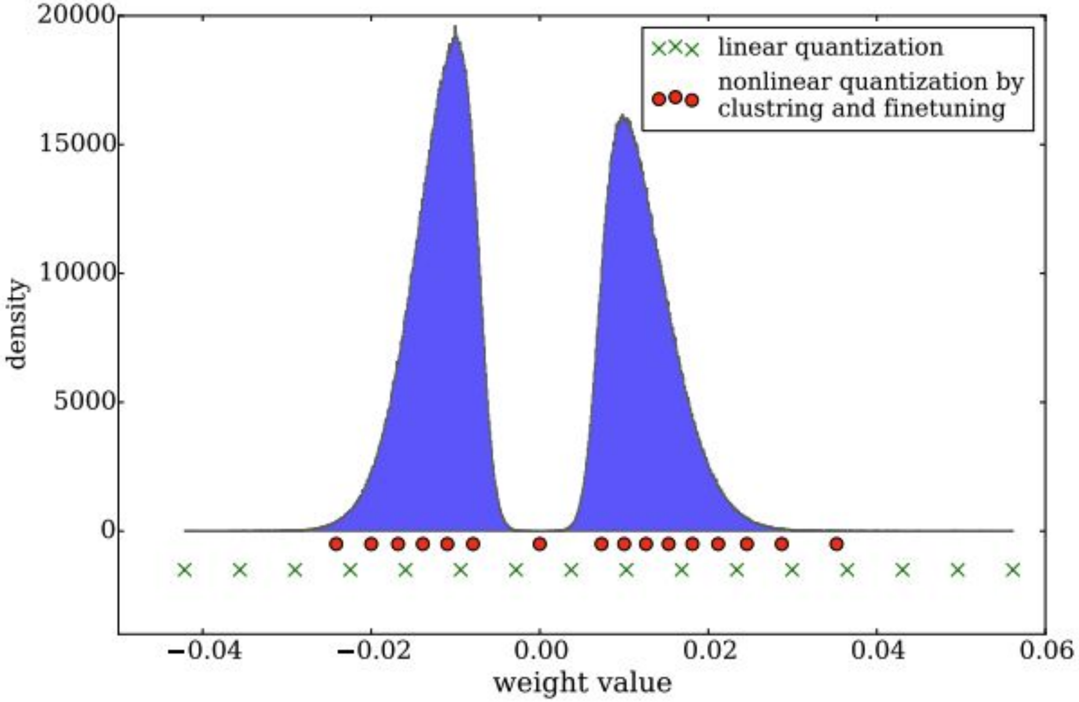
Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Neural Networks Quantization

## K-Means-based Weight Quantization: Centroids Initialization




Three different methods for centroids initialization




Initial distribution of weights, distribution of codebook before fine-tuning (greencross), and after fine-tuning (red dot)

Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Neural Networks Quantization

## K-Means-based Weight Quantization

quantized weights (32-bit float)

| | | | |
|---|---|---|---|
| 2.00 | -1.00 | 1.50 | 0.00 |
| 0.00 | 0.00 | -1.00 | 2.00 |
| -1.00 | 2.00 | 0.00 | -1.00 |
| 2.00 | 0.00 | 1.50 | 1.50 |

**During Computation**

decode

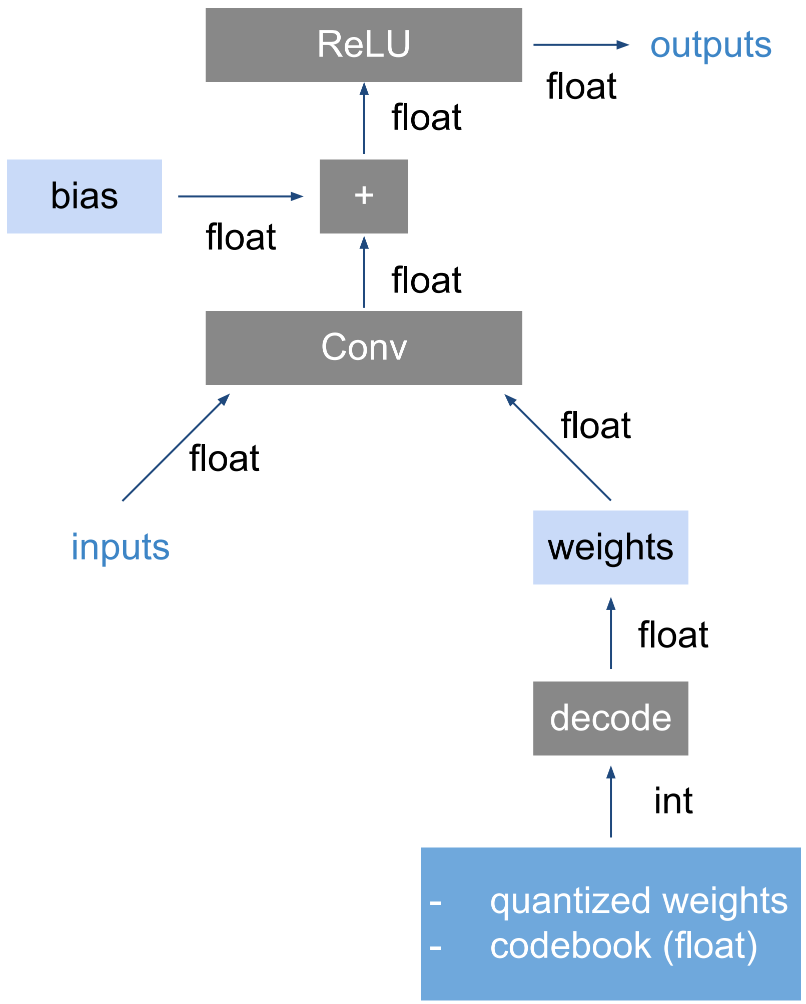
cluster index (2-bit int)

| | | | |
|---|---|---|---|
| 3 | 0 | 2 | 1 |
| 1 | 1 | 0 | 3 |
| 0 | 3 | 1 | 0 |
| 3 | 1 | 2 | 2 |

centroids

| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 0: | 0.00 |
| 1: | -1.00 |

**In Storage**




Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

# Compression Technique: Quantization

**Drawbacks** of neural network quantization:

- **Proficiency in Hardware Architecture**: Necessitates a in-depth understanding of hardware intricacies and bitwise computations.
- **Hardware Limitations**: Efficiency gains are intrinsically linked to the characteristics and capabilities of the utilized hardware.
- **Optimization Difficulties**: Maintaining balance between reducing model size through quantization and preserving predictive capabilities could be tricky requiring careful management of the trade-off between model size and accuracy.

# THANK YOU