Lecture 11: Models Compression Techniques

AC215

Pavlos Protopapas
SEAS/Harvard

Announcements / Reminders

- **Milestone 2 Due - 10/18 9PM EST**

Please add Staff GitHub Account - @ac2152024 (or ac215.2024@gmail.com) as collaborator (See Ed post)

Upcoming –

- **Guest Lecture - Modal Labs 10/24**

- **Milestone 3 - Midterm Presentations - 10/31** (Note: For some groups, presentations may begin and end 15-30 minutes later due to room availability.)

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)

3. You cannot run “big” LLMs locally.

Using machine learning is resource intensive:

i. Computing power to train millions/billions 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

Compression Techniques

- **Knowledge distillation**
- Pruning

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**.

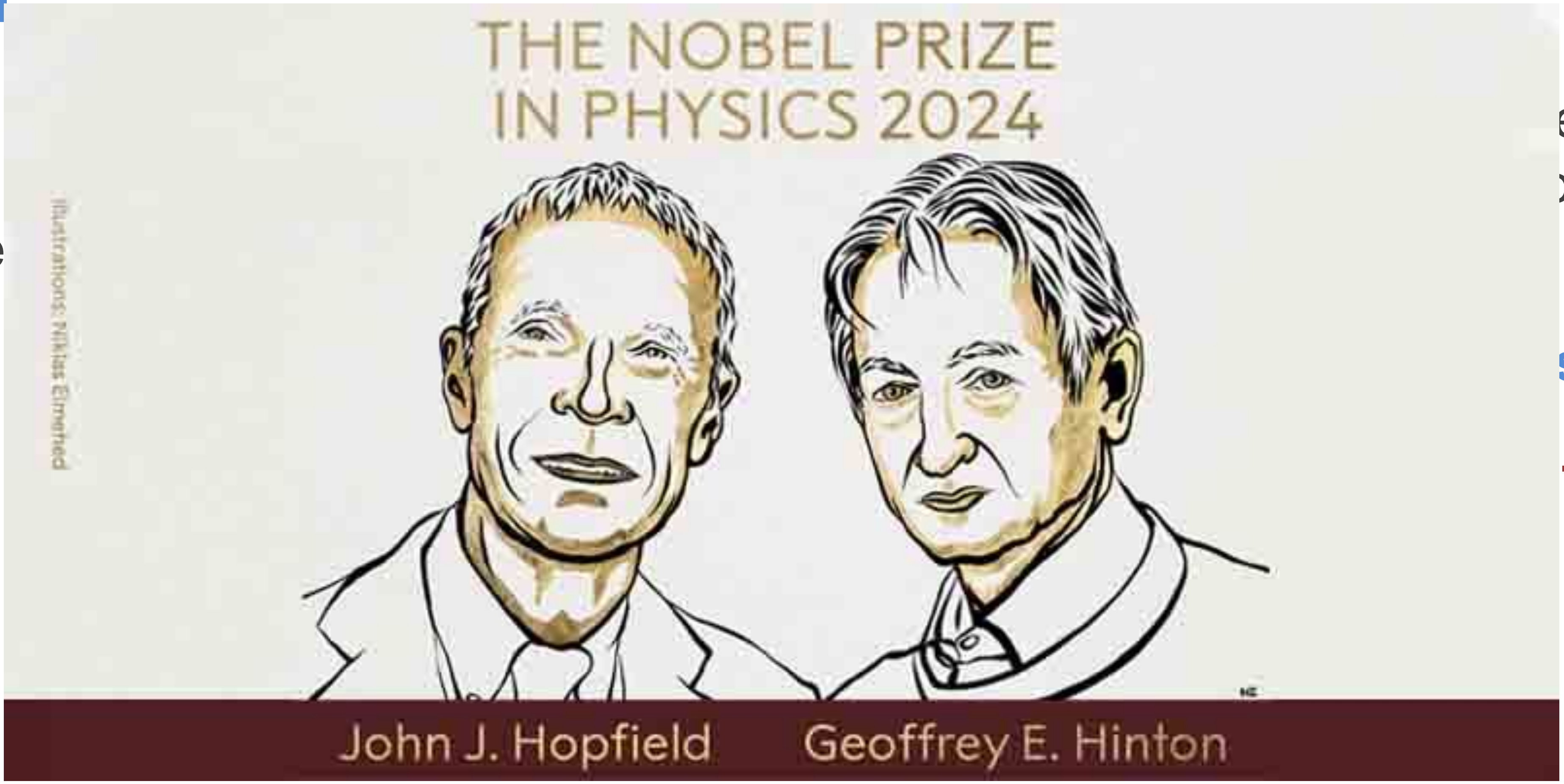
Buciluă et al., *Model Compression*, 2006
Hinton et al., *Distilling the Knowledge in a Neural Network*, 2014

Compression Technique: Distillation

Idea:

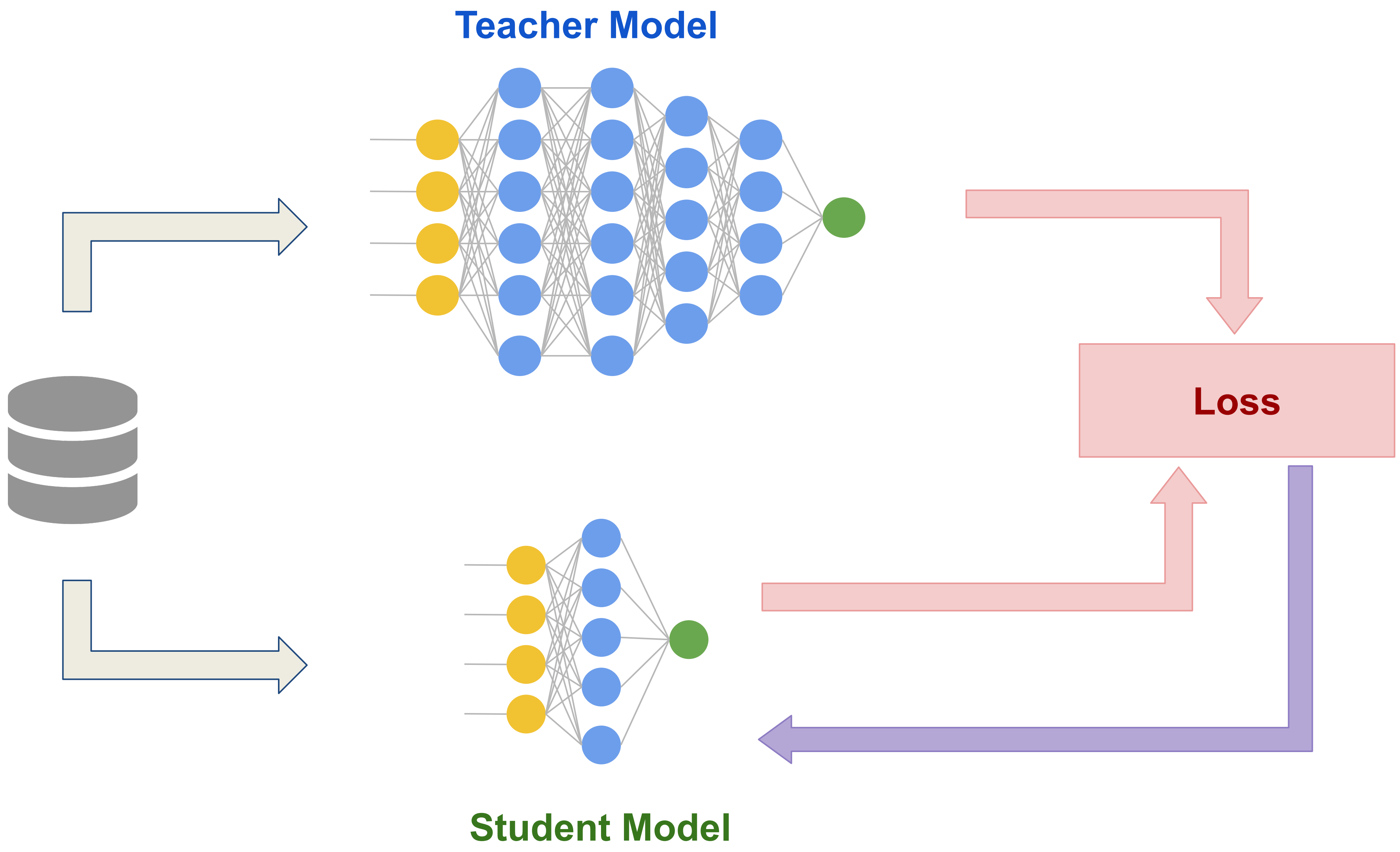
- In edge from a l or de
- In **stillation.**

Main **tasks;** thus



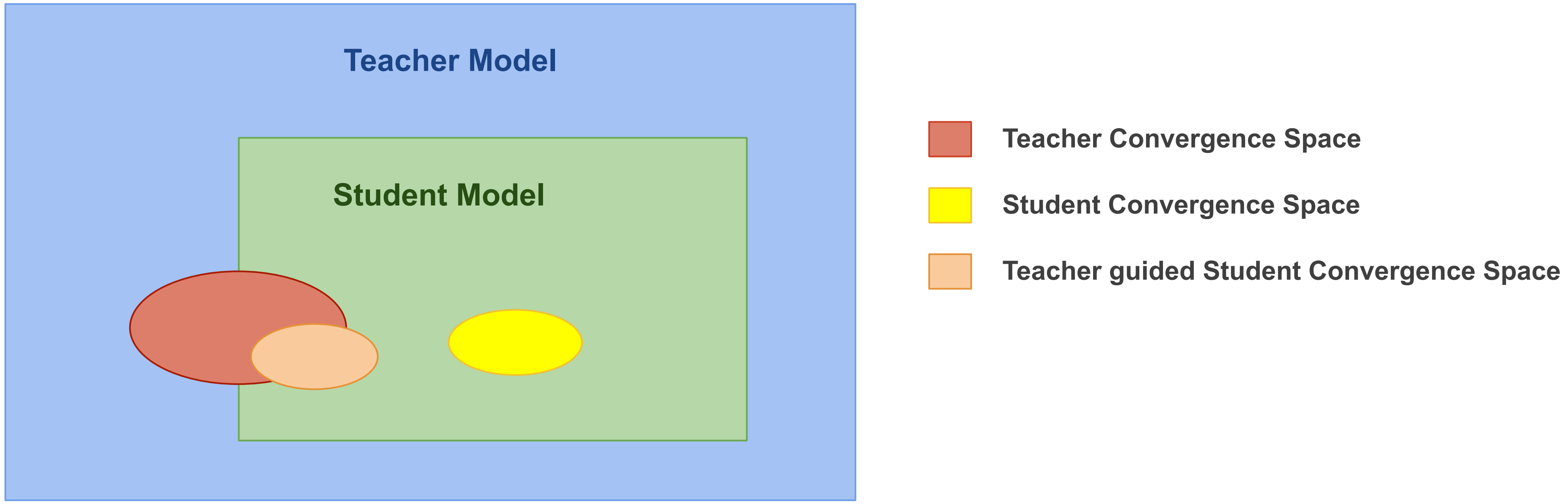
Buciluă et al., *Model Compression*, 2006
Hinton et al., *Distilling the Knowledge in a Neural Network*, 2014

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.


Teacher Model
Student Model
Teacher Convergence Space
Student Convergence Space
Teacher guided Student Convergence Space

Distillation: Soft targets

Training on one-hot encoded labels only give information about a specific class with complete certainty.

This ignores the rich taxonomy that might exist on a dataset.

Models trained in this setting, will be overconfident in the predictions, hurting the prediction on unseen data.

J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, ImageNet: A large-scale hierarchical image database, 2009

Distillation: Soft targets

While training, our confidence in the labels must be high.

Otherwise the data would not be present in the training set.

For example:

$$\text{This is a Dog} \longrightarrow \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Distillation: Soft targets

While training, our confidence in the labels must be high.

Otherwise the data would not be present in the training set.

For example:

$$\text{This is also a dog} \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Distillation: Soft targets

While training, our confidence in the labels must be high.

Otherwise the data would not be present in the training set.

For example:

$$\text{This is also a dog} \longrightarrow \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Distillation: Soft targets

While training, our confidence in the labels must be high.

Otherwise the data would not be present in the training set.

For example:

$$\text{This is a Cat} \longrightarrow \begin{pmatrix}0\\ \vdots\\ 0\\ 1\\ \vdots\\ 0\end{pmatrix}$$

Distillation: Soft targets

While training, our confidence in the labels must be high.

Otherwise the data would not be present in the training set.

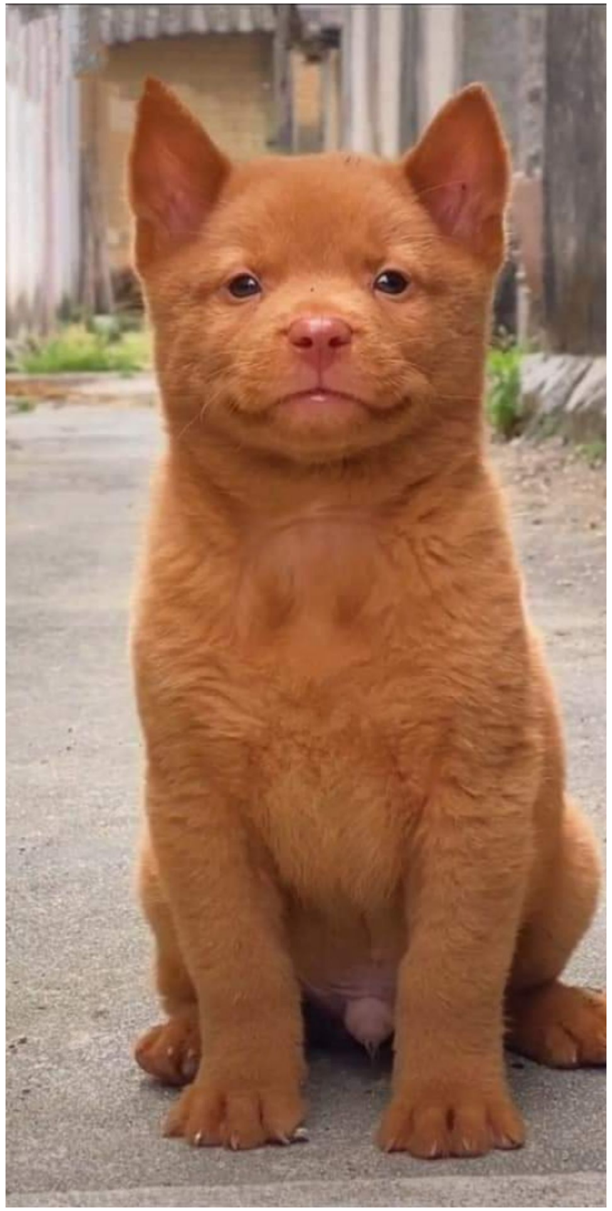
For example:

$$\text{This is also a Cat} \longrightarrow \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

Distillation: Soft targets

What about this example?

This is a … $\begin{pmatrix} 0 \\ \vdots \\ 0.5 \\ 0.5 \\ \vdots \\ 0 \end{pmatrix}$?



Distillation: Soft targets

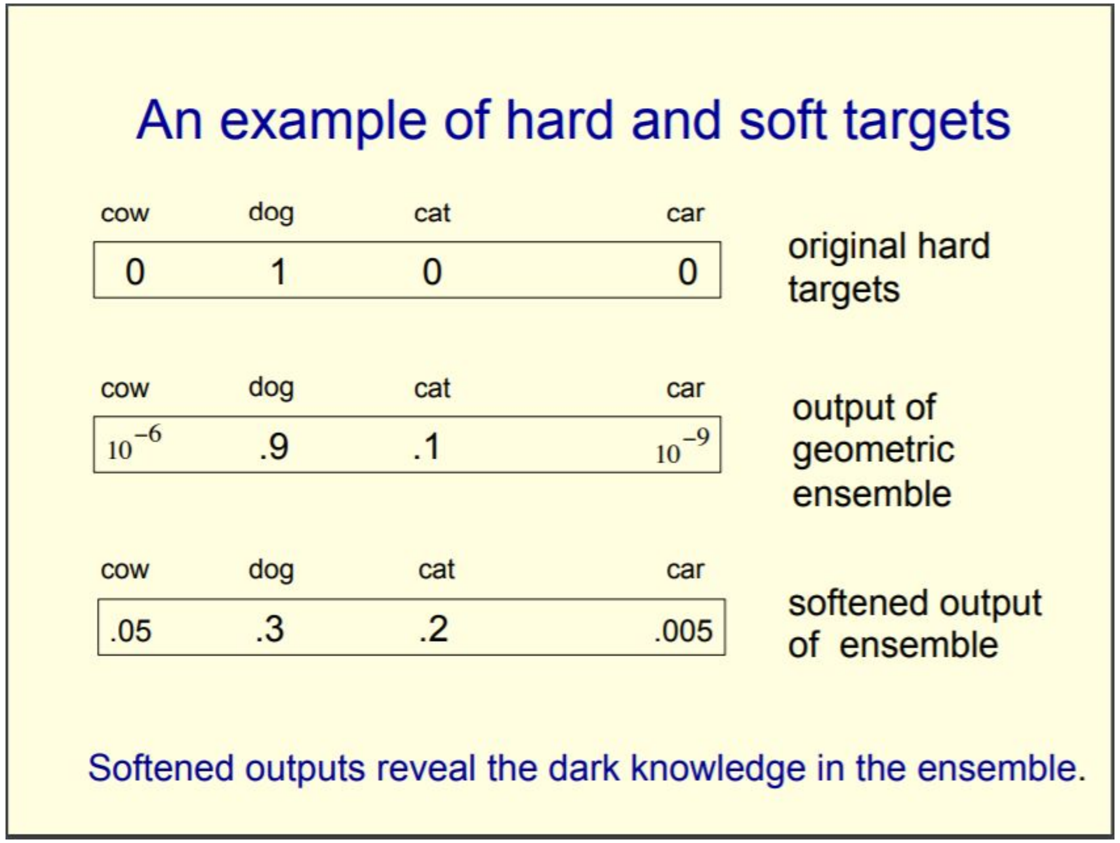
In the real world, the distinction is never that clear.

Ideally, the training labels must be soft, to learn about the nuanced relations across the elements of the dataset.

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

The teacher shows the student the relations between the different classes, based on what it learned during its training stage.

From a dataset containing the original categories, the teaching outputs the normalized probabilities for each example.

cow	dog	cat	car	
0	1	0	0	Original labels
10^{-6}	0.9	0.1	10^{-9}	Probabilities

Adapted from:
Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

Distillation: Teacher - Student

To enhance the student’s learning, the teacher softens its output *z* even more.

To do this, introduces the softmax function with temperature *T*:

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

cow	dog	cat	car	
0	1	0	0	Original labels
10^{-6}	0.9	0.1	10^{-9}	Softmax probabilities
0.05	0.3	0.2	0.005	Softened probabilities

Adapted from:
Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

Distillation: Teacher - Student

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

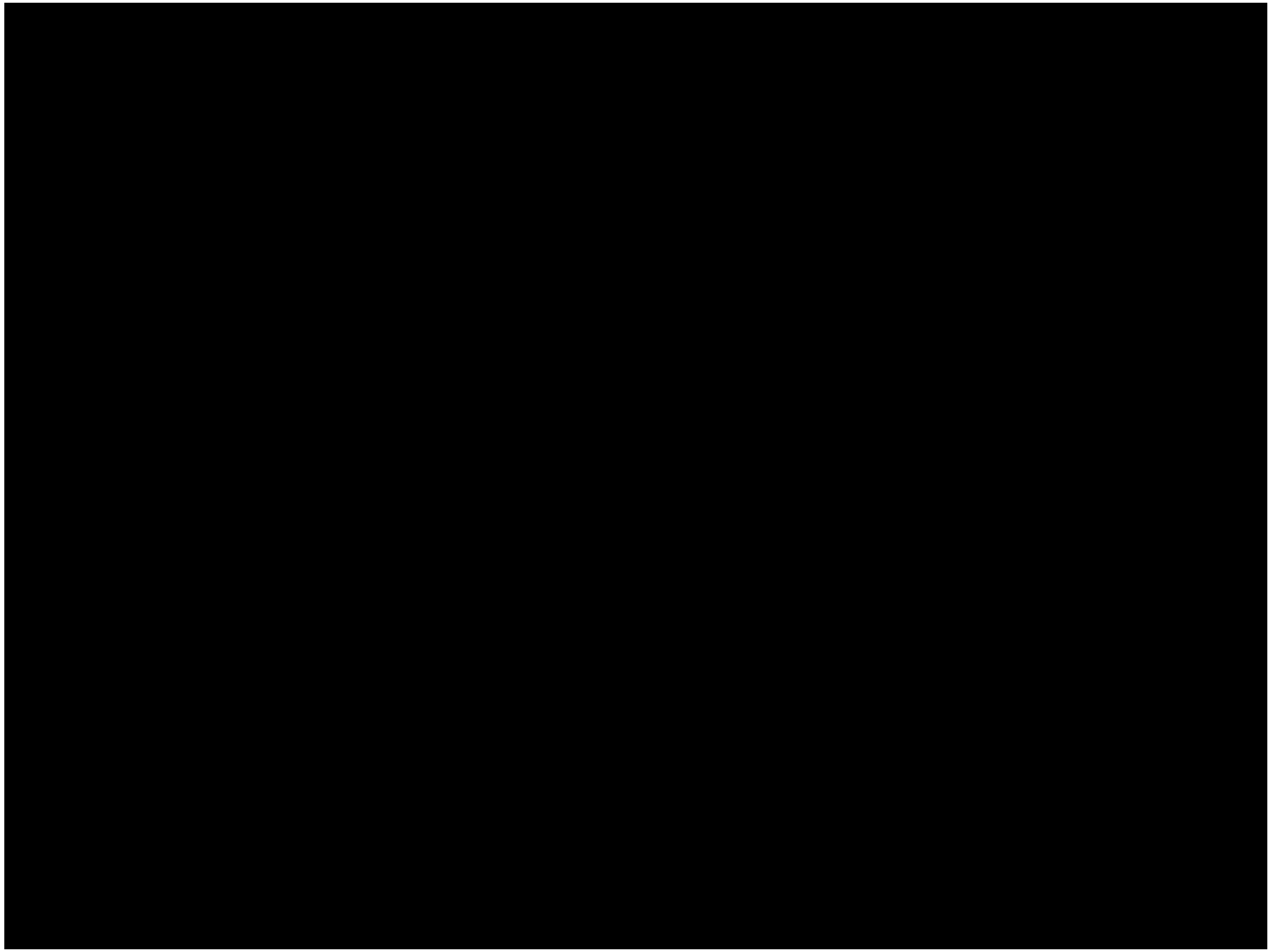
A high value of *T* will dilute the information contained in the probabilities.

- The standard softmax is recovered for *T*=1.
- For T>1, the probabilities are softened. Typical values range between 2 and 5.
- For T<1, it approaches to the original one-hot labels.

cow	dog	cat	car	
0	1	0	0	Original labels
10^{-6}	0.9	0.1	10^{-9}	Softmax probabilities
0.05	0.3	0.2	0.005	Softened probabilities

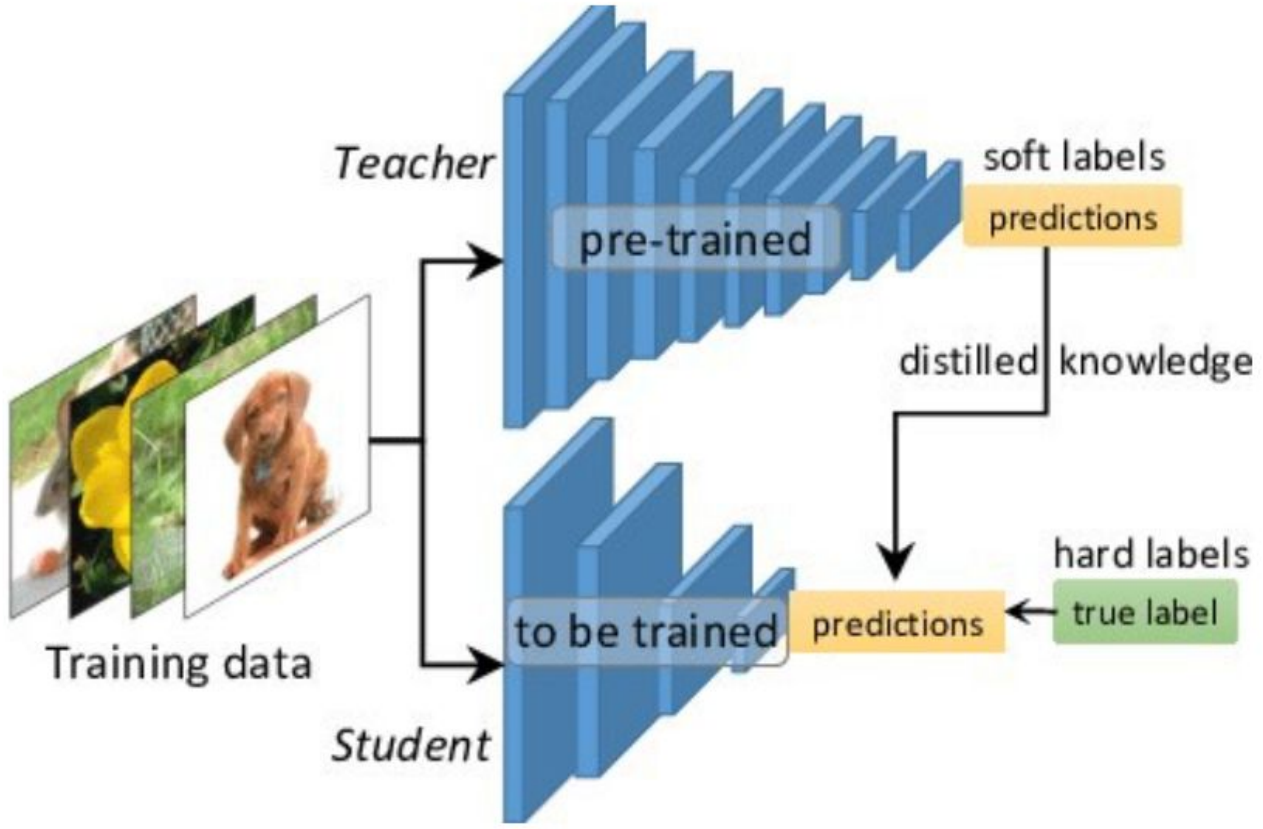
Adapted from:
Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

Temperature explanation



Distillation: Teacher Student Training

The student is trained to solve the original problem, and to replicate the softened probabilities from the teacher.



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

Distillation: Teacher Student Training

Scenario 1:

Train on the same data as the teacher model. DistilBert was trained on this example.

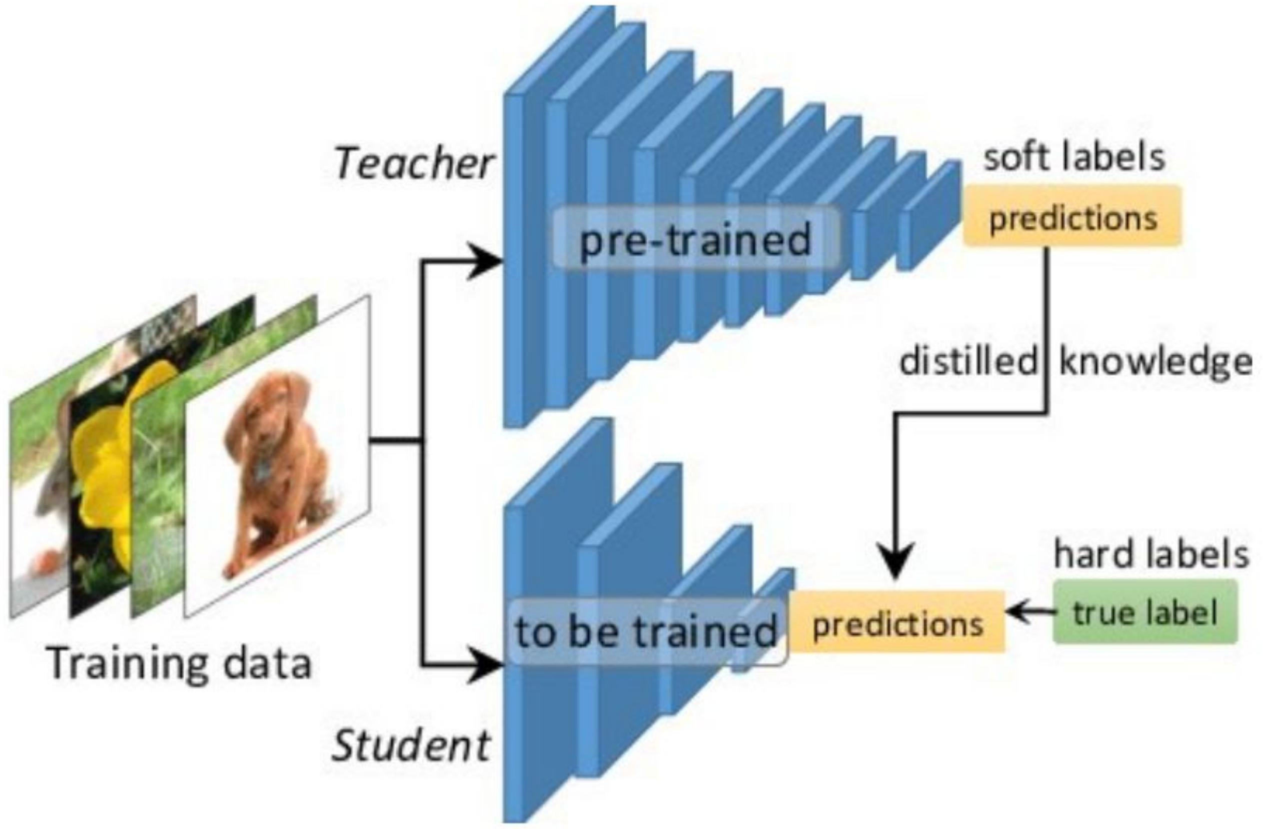
Scenario 2:

Train on a reduce dataset. This is usually done when you have the model but not the original dataset or resources.

Distillation: Teacher Student Training

It minimizes 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 probabilities with T>1

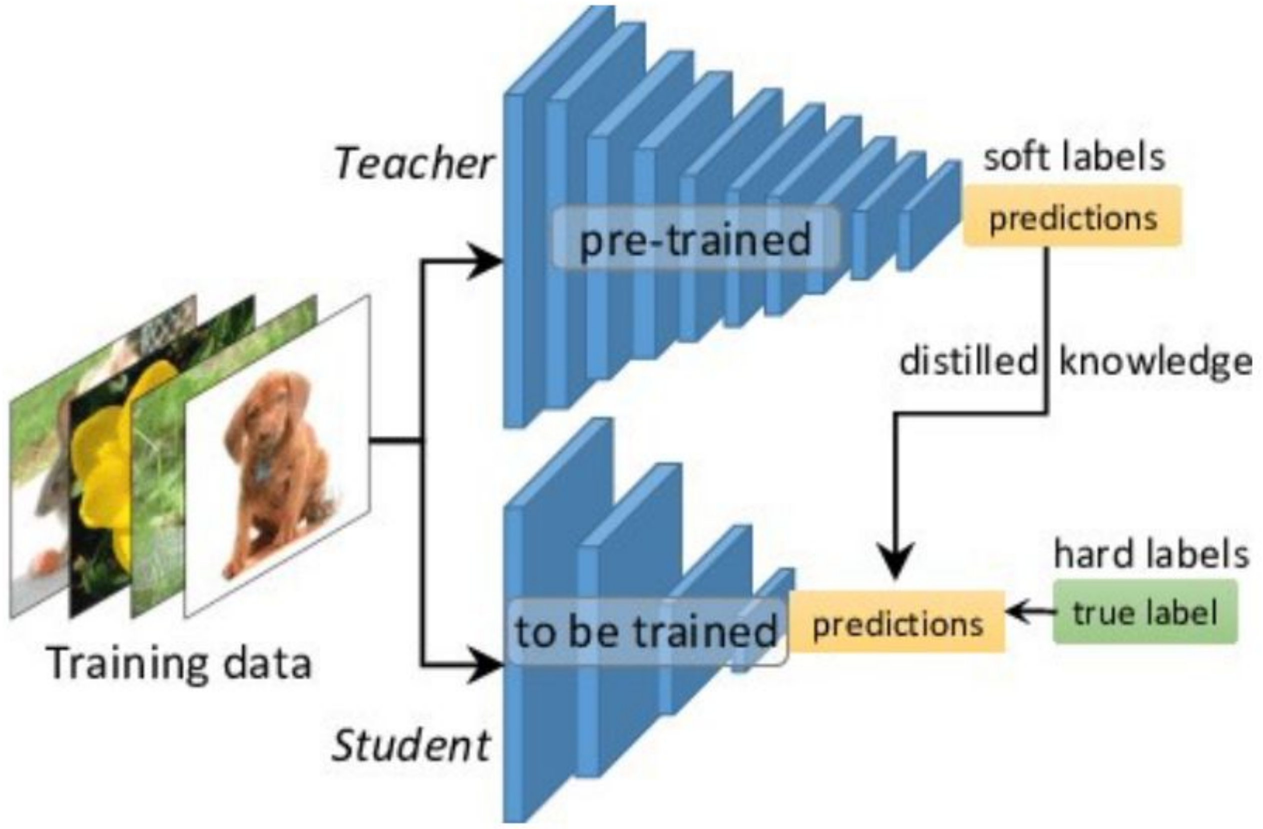


Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

Distillation: Teacher Student Training

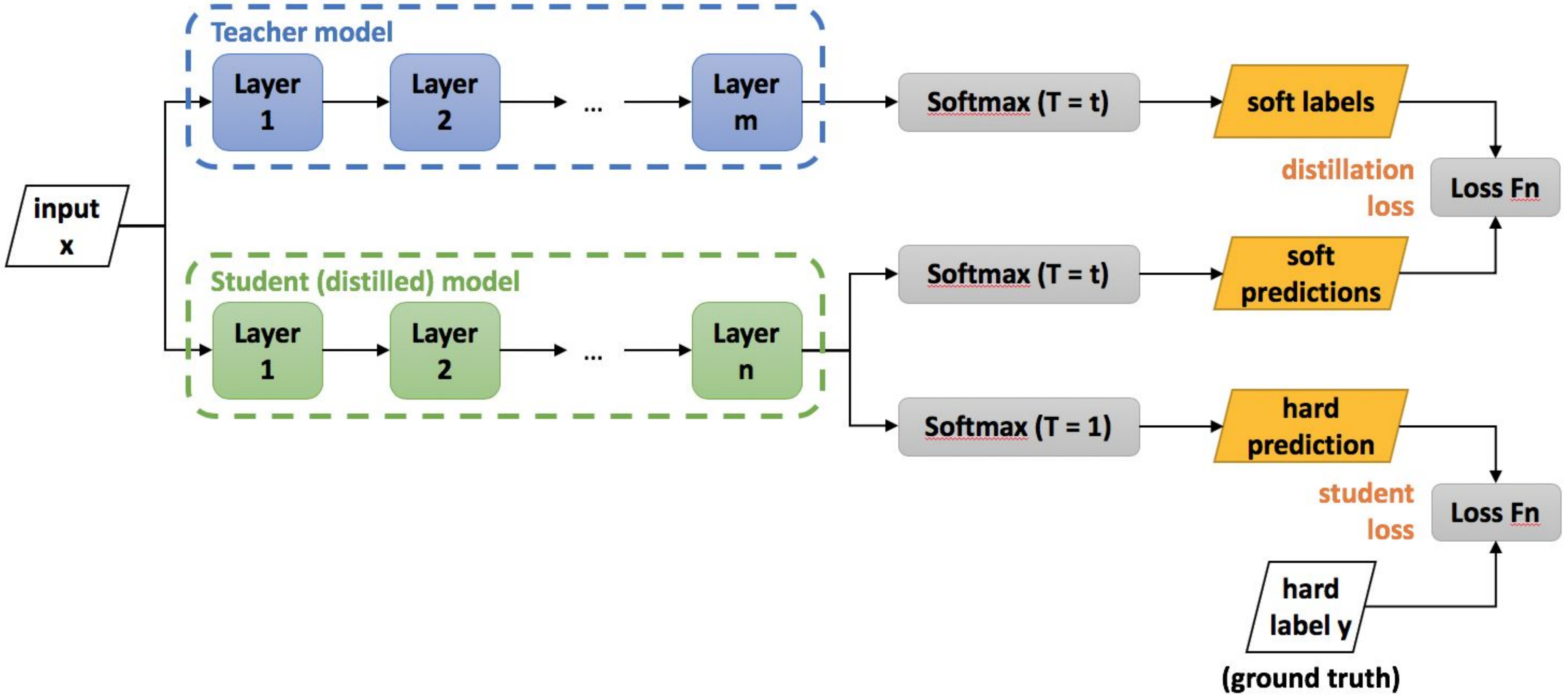
Trained to minimize the sum of two different cross entropy functions:

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



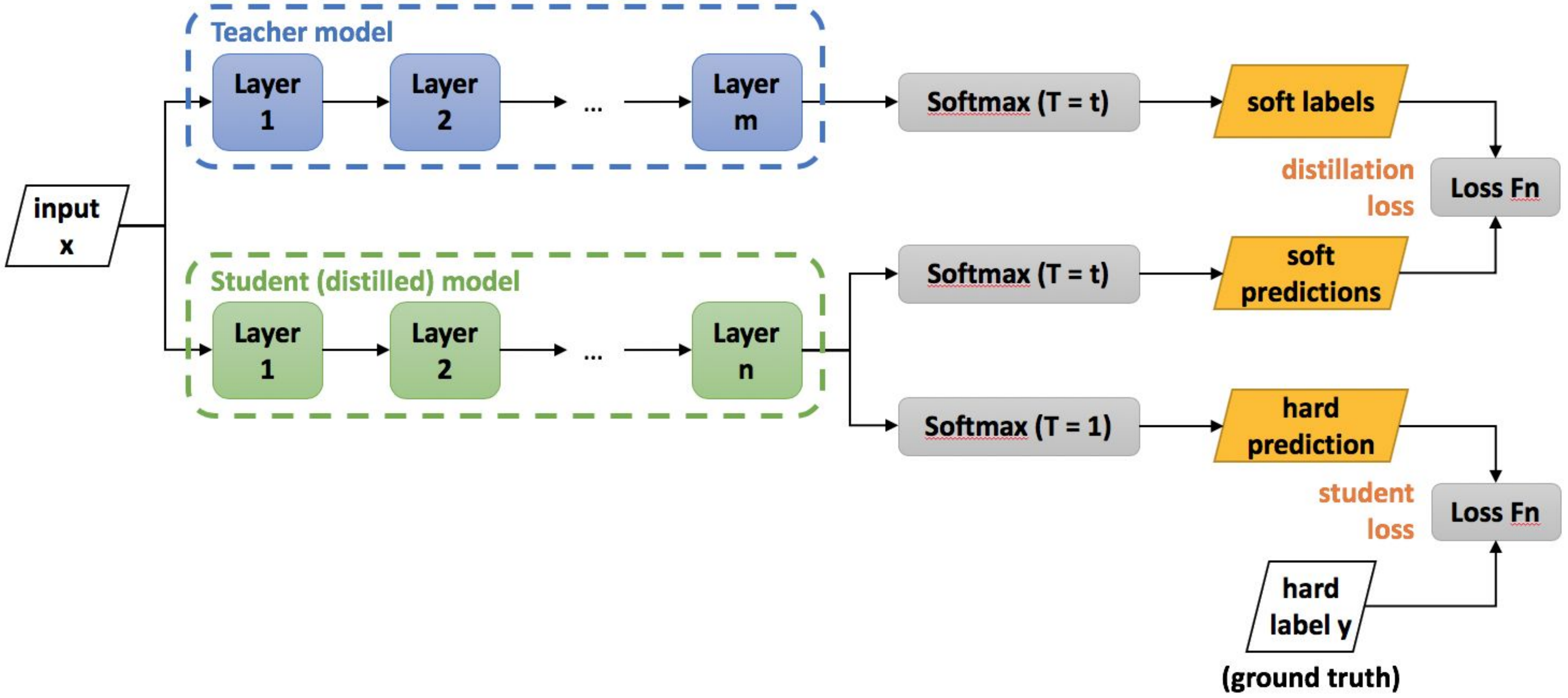
Geoffrey Hinton, Oriol Vinyals & Jeff Dean, *Dark Knowledge*

Distillation: Teacher Student Training



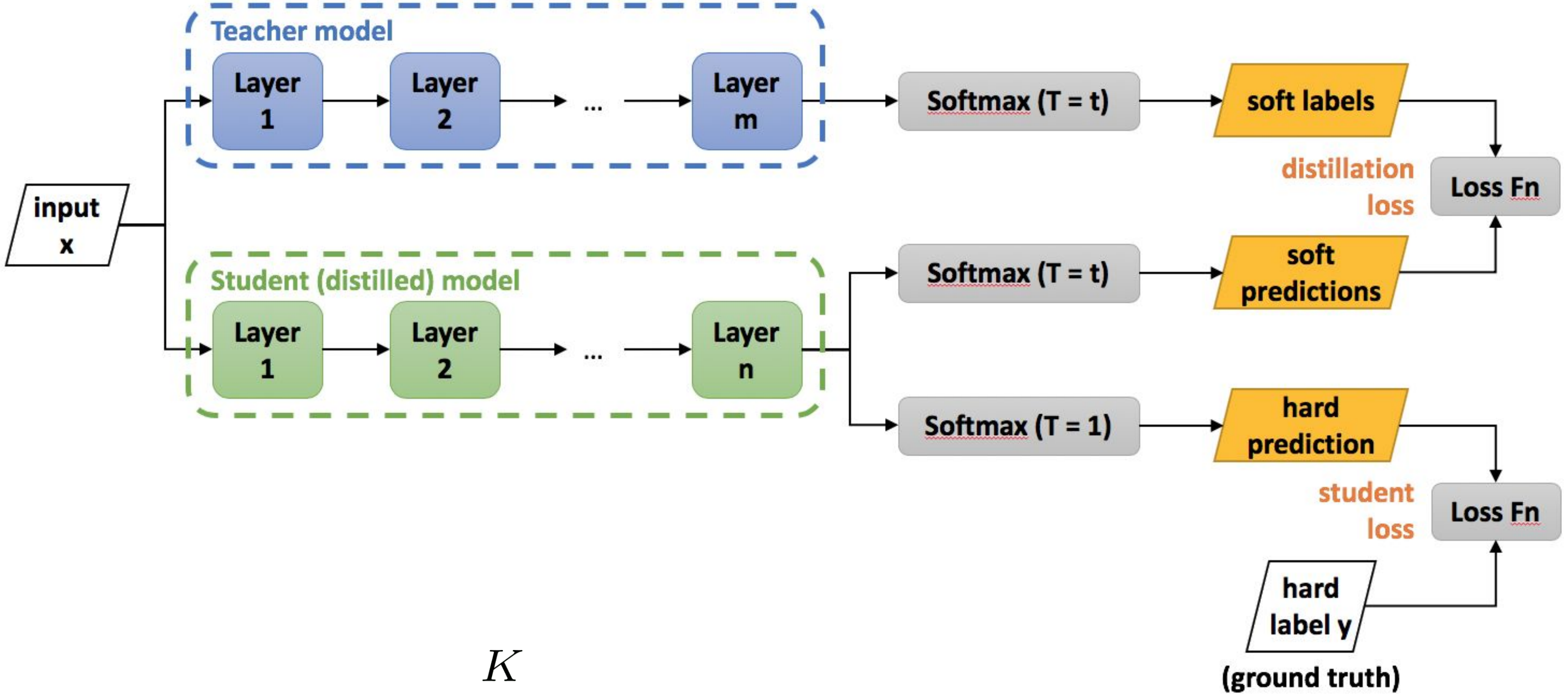
$$L = \lambda L_{\text{student}} + (1 - \lambda) L_{\text{distillation}}$$

Distillation: Teacher Student Training



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

Distillation: Teacher Student Training



$$L_{Distillation} = -\sum_{k}^{K} p_T^k(x,T)\log\left(q_S^k(x,T)\right)$$

Distillation: Teacher Student Training

$$L_{Distillation} = -\sum_{k}^{K} p_T^k(x,T)\log\left(q_S^k(x,T)\right)$$

In our example, for the example *n* in the dataset.

$$p_T(x,T) = [0.05, 0.3, 0.2, 0.005]$$

And our predictions

$$q_T(x,T) = [0.1, 0.45, 0.30, 0.15]$$

The resulting loss value for this example is:

$$-[0.05\cdot\log(0.1) + 0.3\cdot\log(0.45) + 0.2\cdot\log(0.30) + 0.005\cdot\log(0.15)]$$

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: Learning not only the output, but the intermediate representations.

4: Quite a *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).

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**

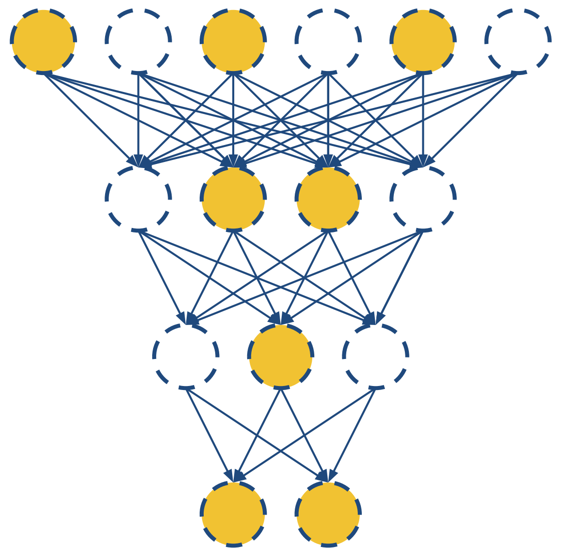
Compression Technique: Pruning

Pruning is based on the idea of the Lottery Ticket Hypothesis.

A randomly-initialized neural network contains a subnetwork that is initialized such that - when trained in isolation- it can match the test accuracy of the original network after training for at most the same number of iterations.

Inside each network, we might find a winning ticket where a small subset of the neurons are non-zero.

In principle, this network will be use less memory, perform faster during inference without losing performance.



The lottery ticket hypothesis: Finding sparse, trainable neural networks
[Frankle, J., & Carbin, M. (2018)]

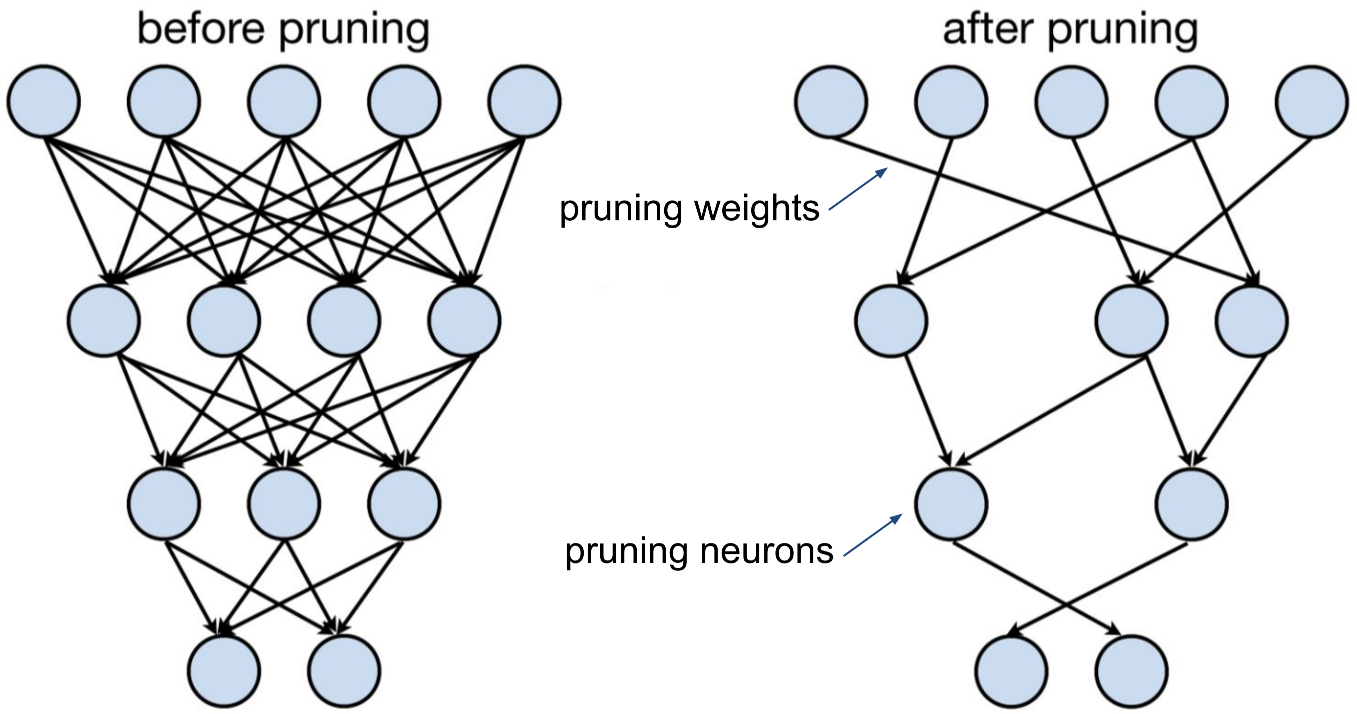
Compression Technique: Pruning

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 structure

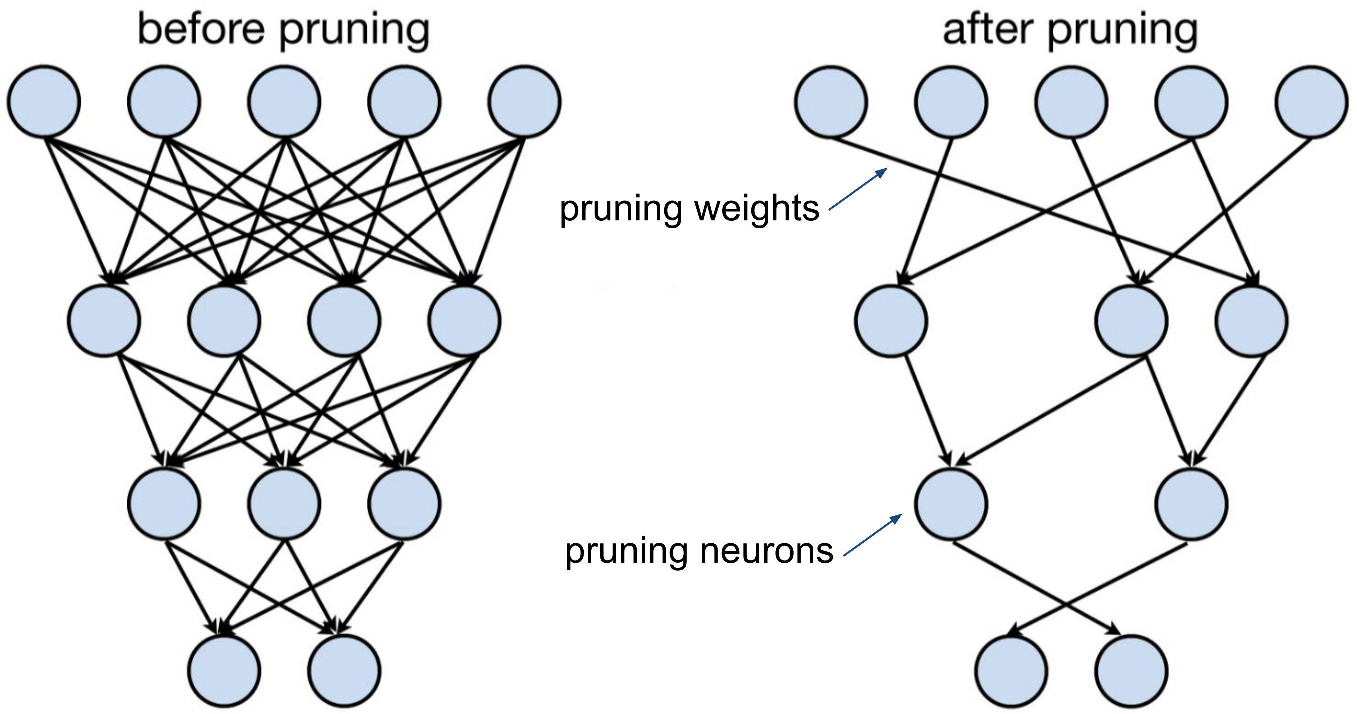


Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]

Compression Technique: Pruning

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

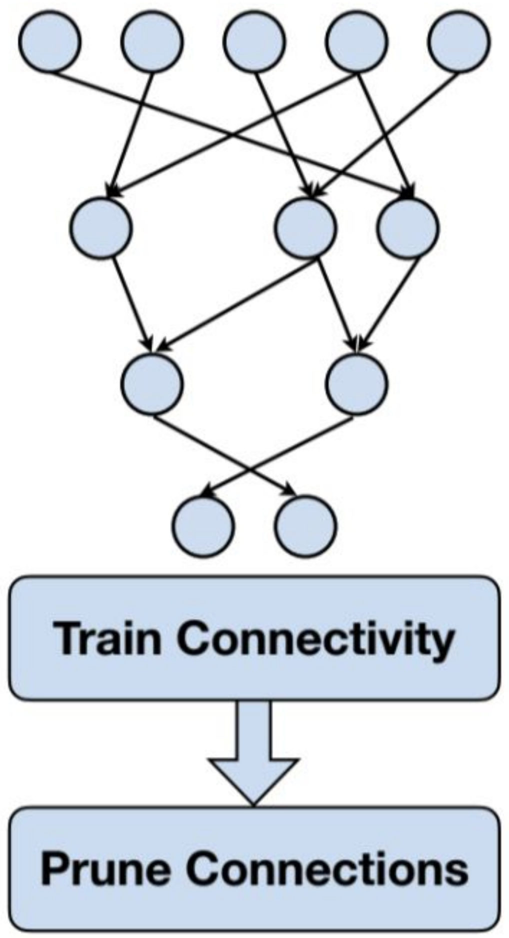
The criteria to select the rank the relevant features might depend on the weights absolute value, on the activations magnitudes, among other values.

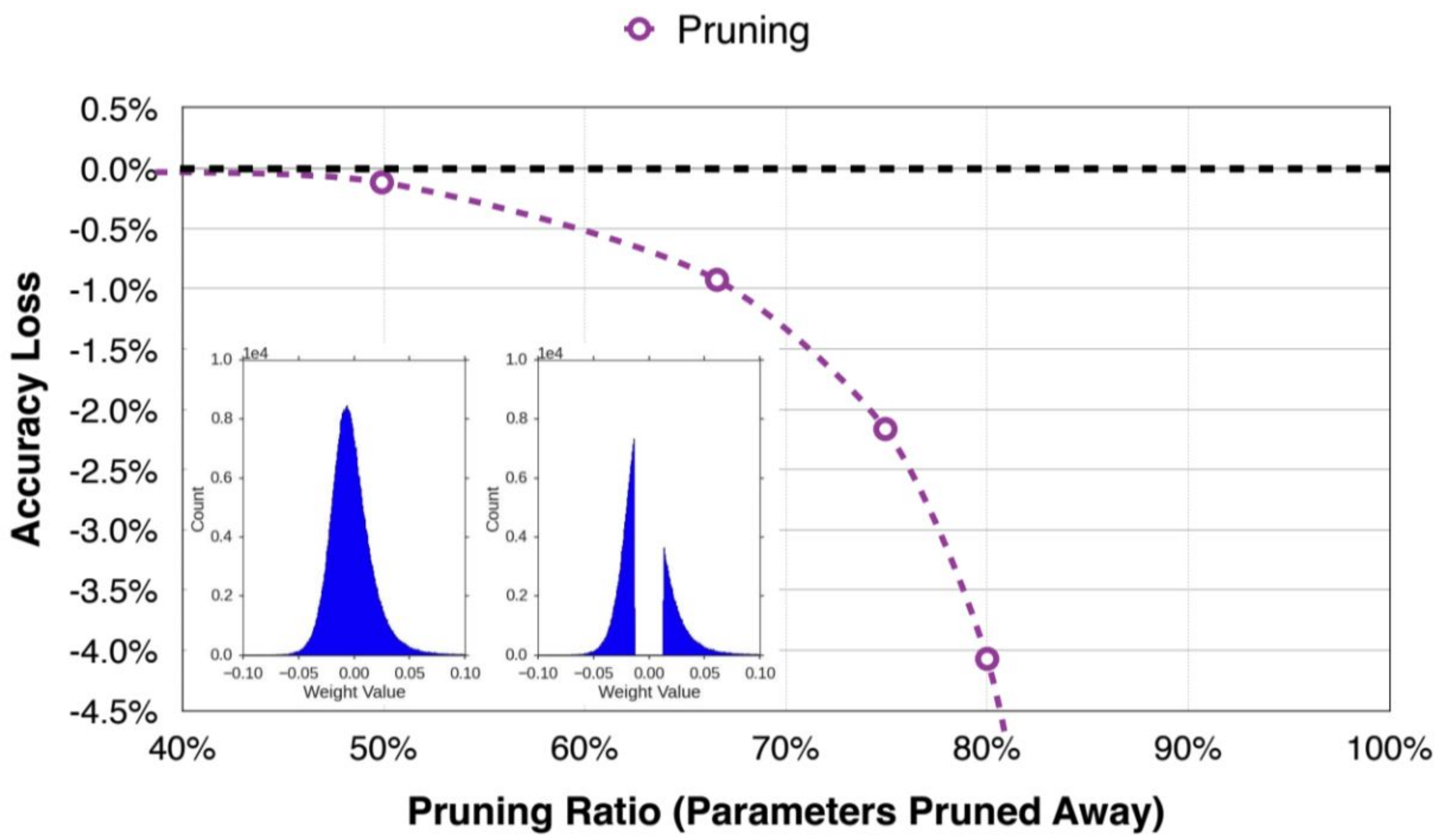


Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]

Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons

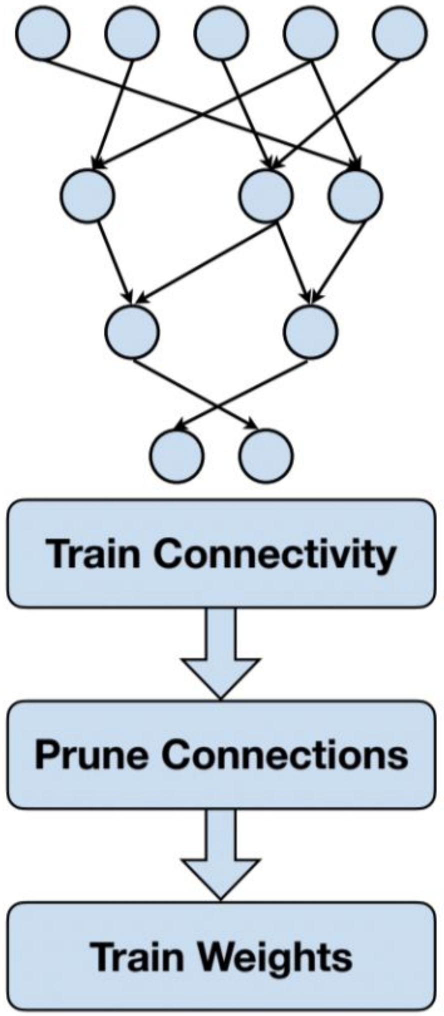


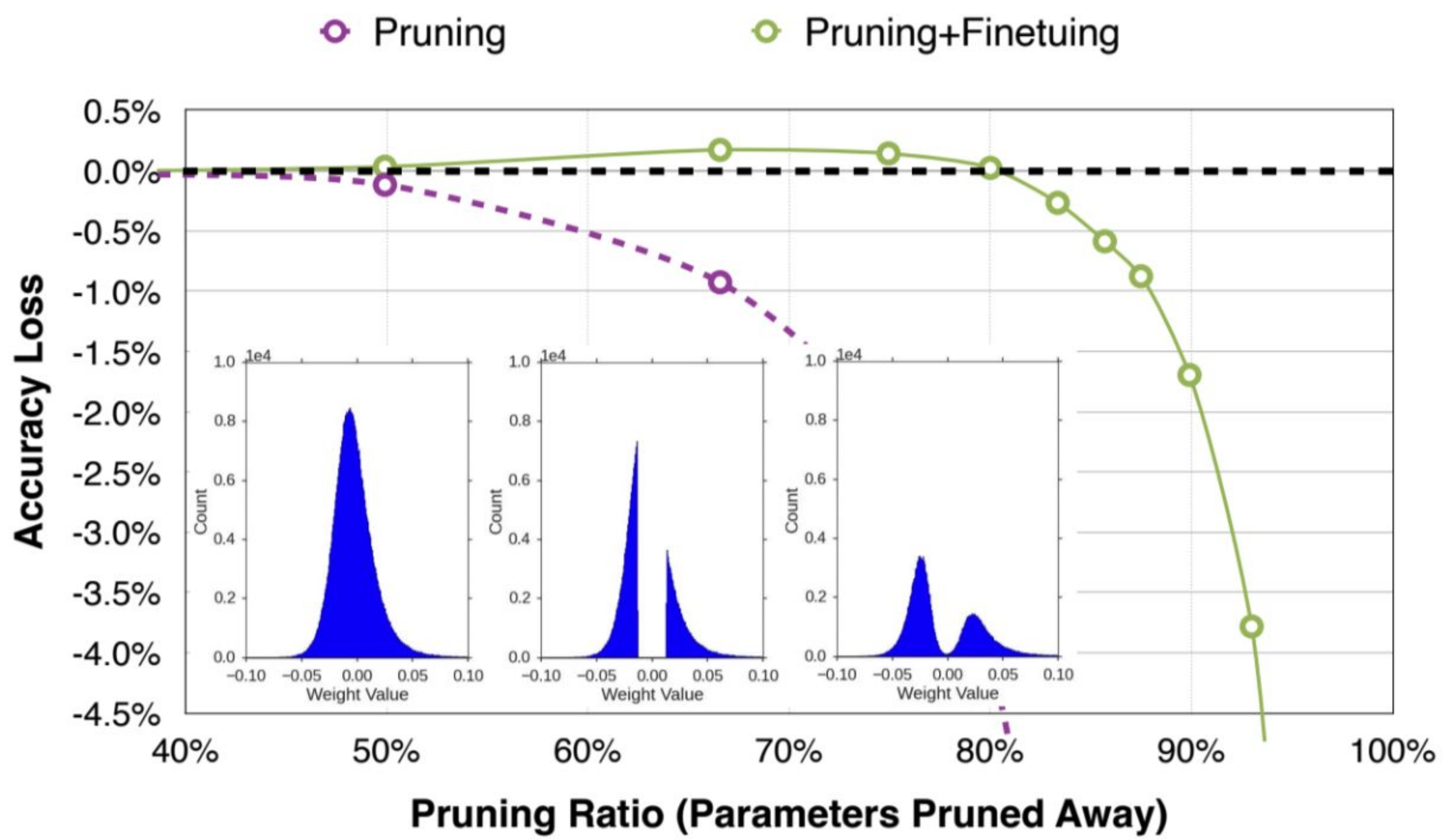


Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons

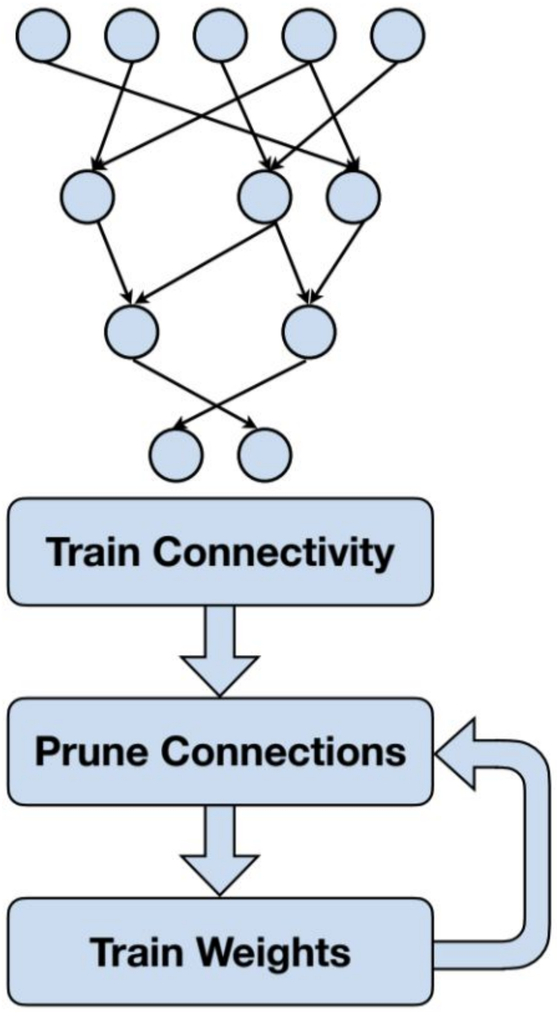


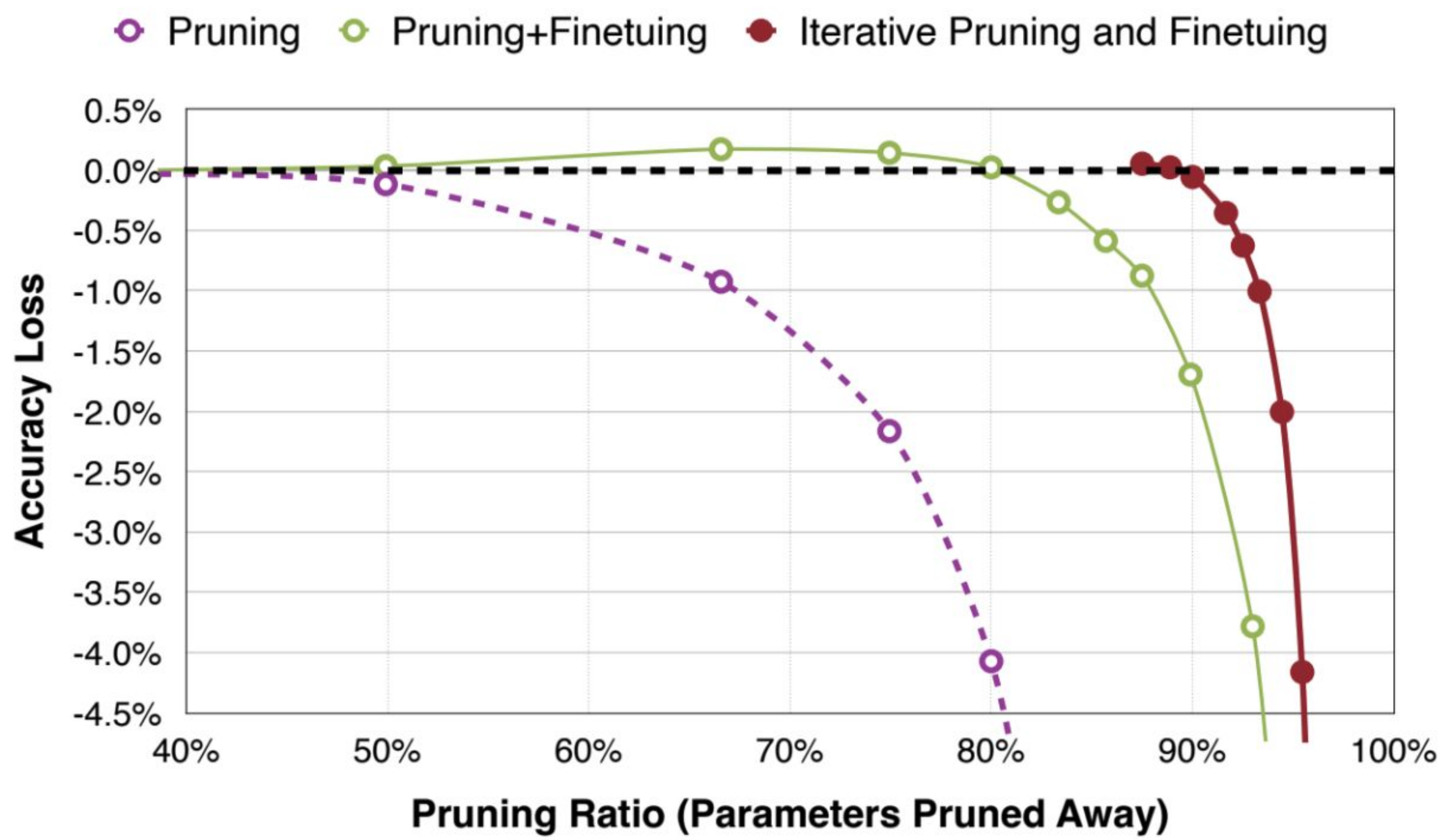


Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

Compression Technique: Pruning

Make neural networks smaller by removing weights and neurons



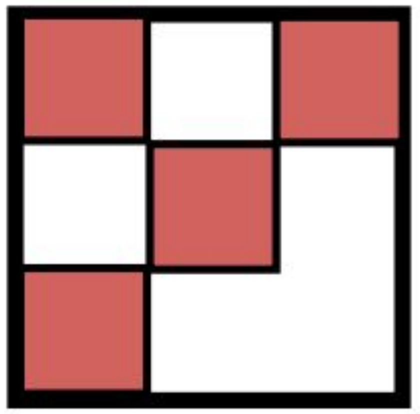


Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
MIT EfficientML.ai: Pruning and Sparsity

Compression Technique: Pruning

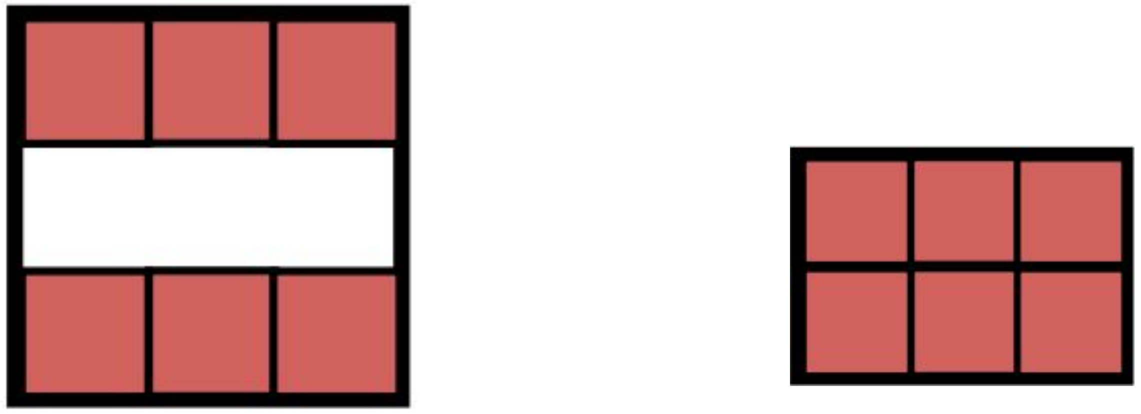
Pruning Granularities

Simple 2-D weight matrix example



Fine-grained/Unstructured

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



Coarse-grained/Structured

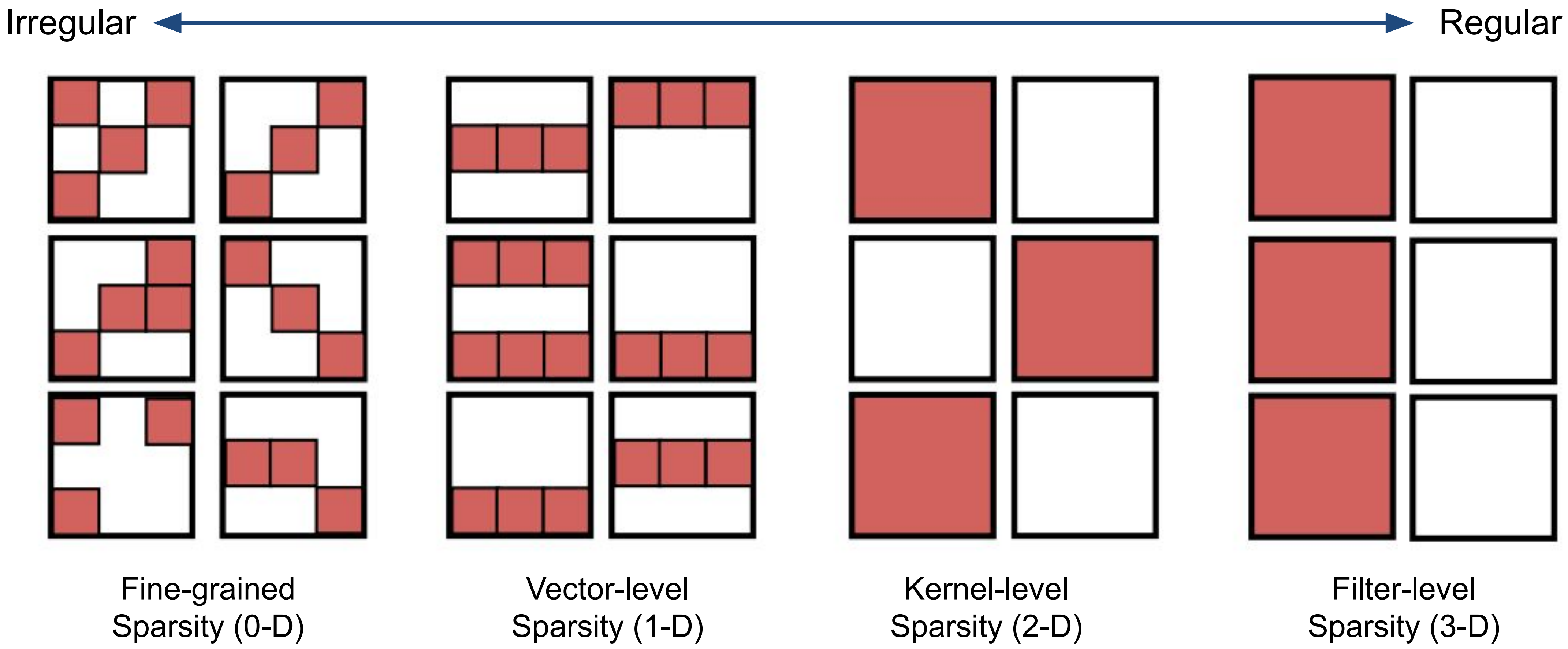
- Less flexible pruning idex choice (a subset of the fine-grained case)
- Easy to accelerate (just a smaller matrix!)

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

Compression Technique: Pruning

Example using convolutional layers

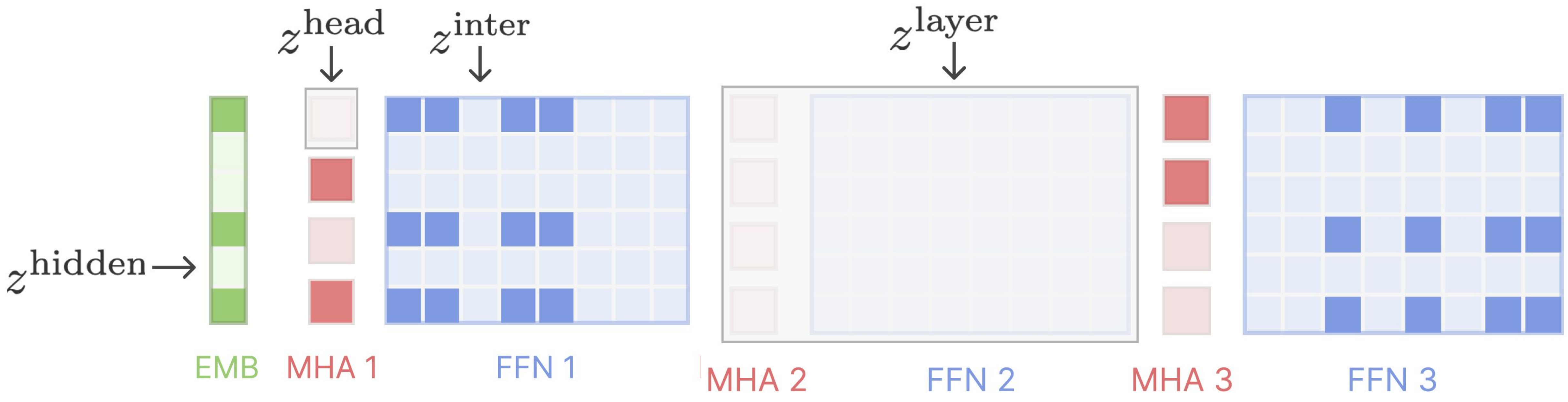
- Commonly used pruning granularities


Irregular
Regular
Fine-grained Sparsity (0-D)
Vector-level Sparsity (1-D)
Kernel-level Sparsity (2-D)
Filter-level Sparsity (3-D)

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

Compression Technique: Pruning

Pruning on transformers can be applied at different levels



Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning [Xia et al.,2024]

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.

Compression Technique: Next steps

Things to consider:

- Quantization, distillation and pruning applied alone can reduce the models complexity.
- But together, these techniques achieve state-of-the-art performance while keeping a "*reduced size*".
- Distilled LLMs can still be huge, with ~7 billion parameters.
- The choice of methodology depends on the dataset, tasks and models.
- Finetuning is always necessary to regain performance losses. LoRA is extremely relevant.
- There is no general methodology that will work out of the box. Experiment!

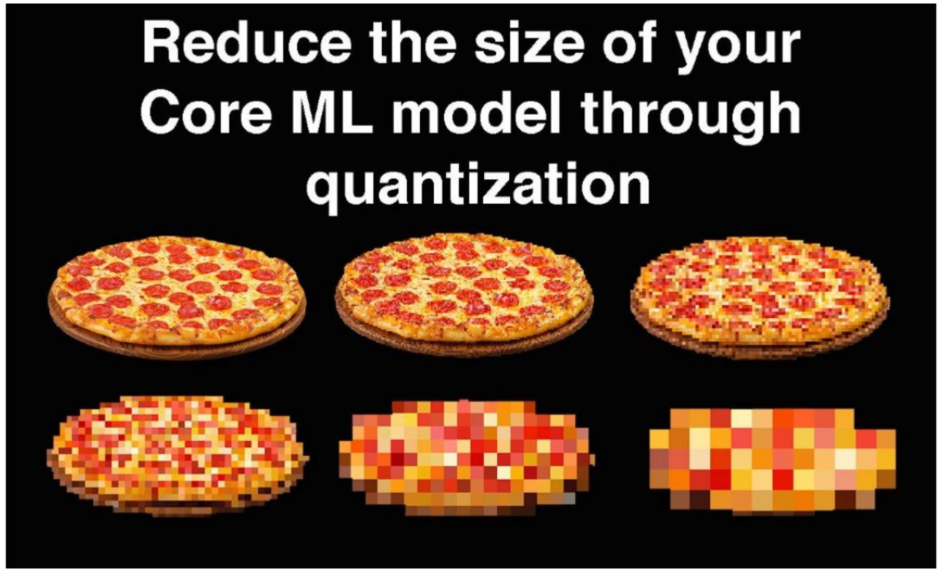
THANK YOU

Tutorial: Model Pruning

Colab Notebook

Compression Technique: Quantization

Main idea is to map values from a large set to values in a smaller set without losing too much information in the process. So by reducing the number of pixels, the image below should still be clear.

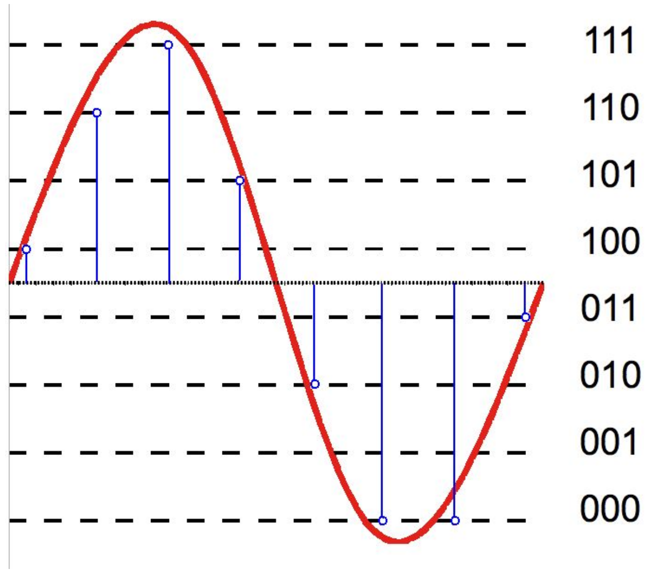


To implement quantization with Tensorflow: MC.AI, *Quantization in Deep Learning using TensorFlow 2.X*, May 2020

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.

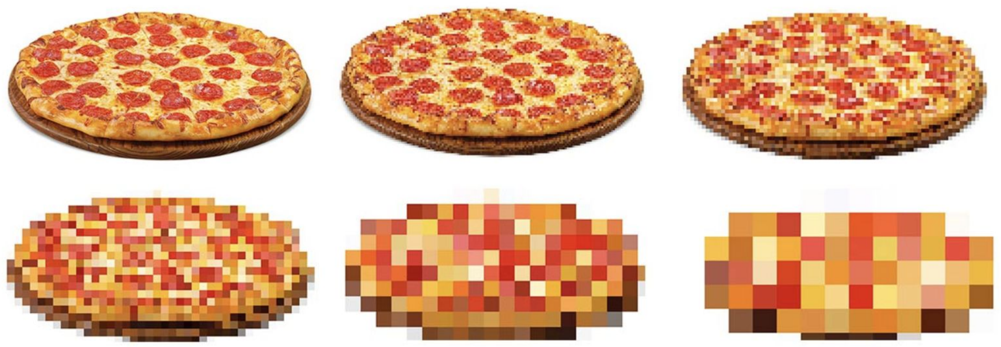


Image is in the public domain. "Analog to digital converter"

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**

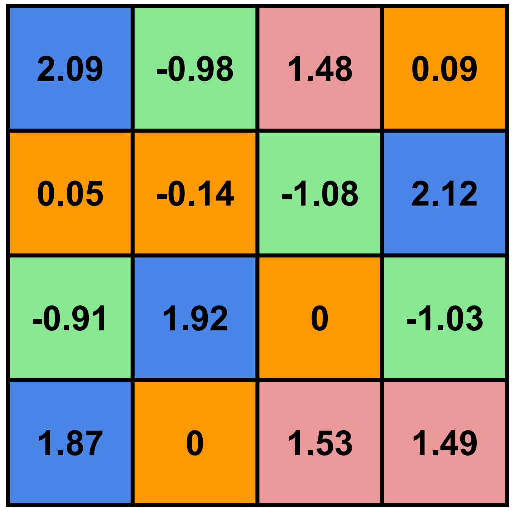
$$\begin{tikzcd} \cancel{2.09,\ 2.12,\ 1.92,\ 1.87} \arrow[d] \\ 2.0 \end{tikzcd}$$

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

$\xrightarrow{\text{cluster}}$

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

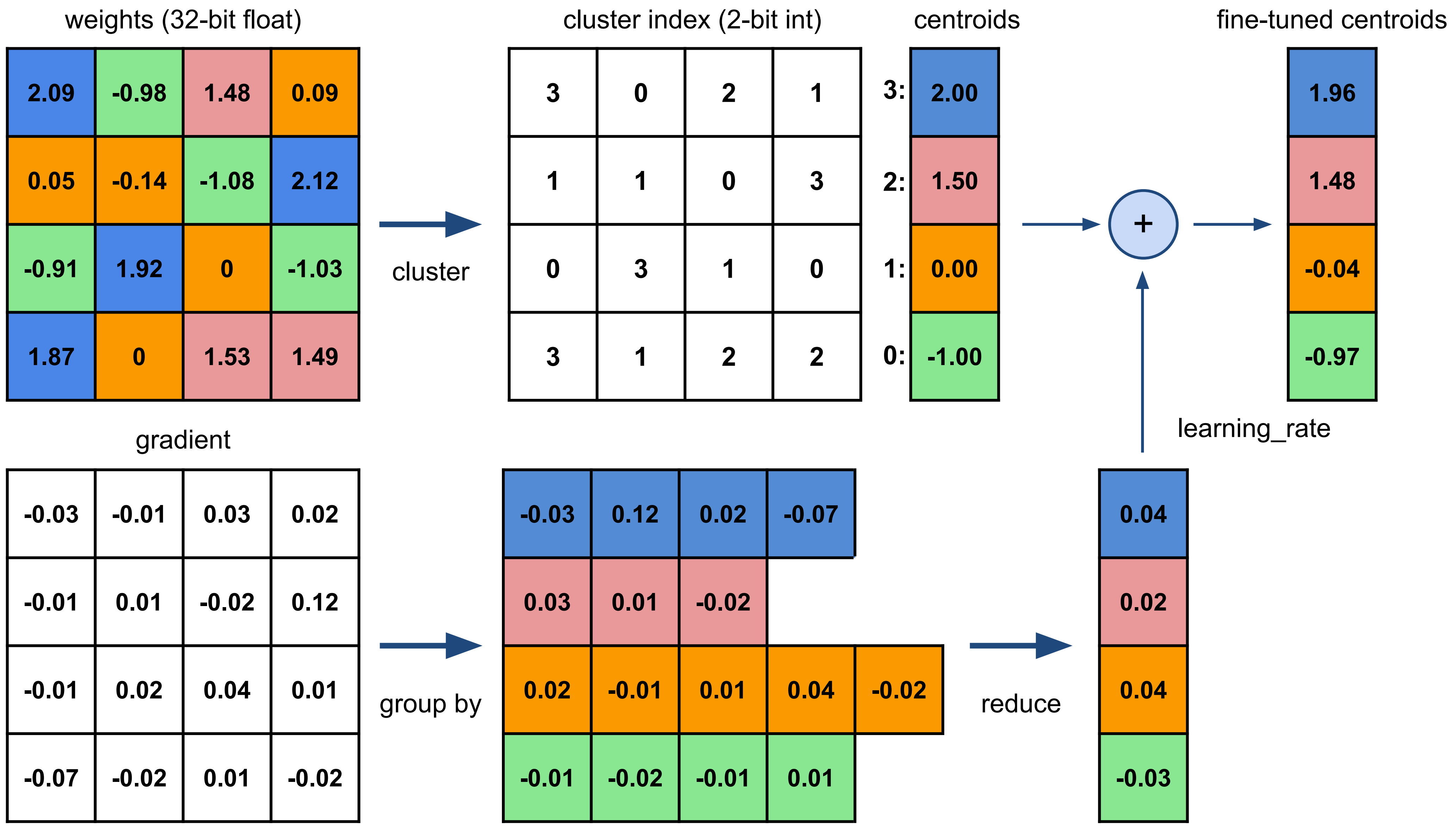
N bit * M
= NM bit

~~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

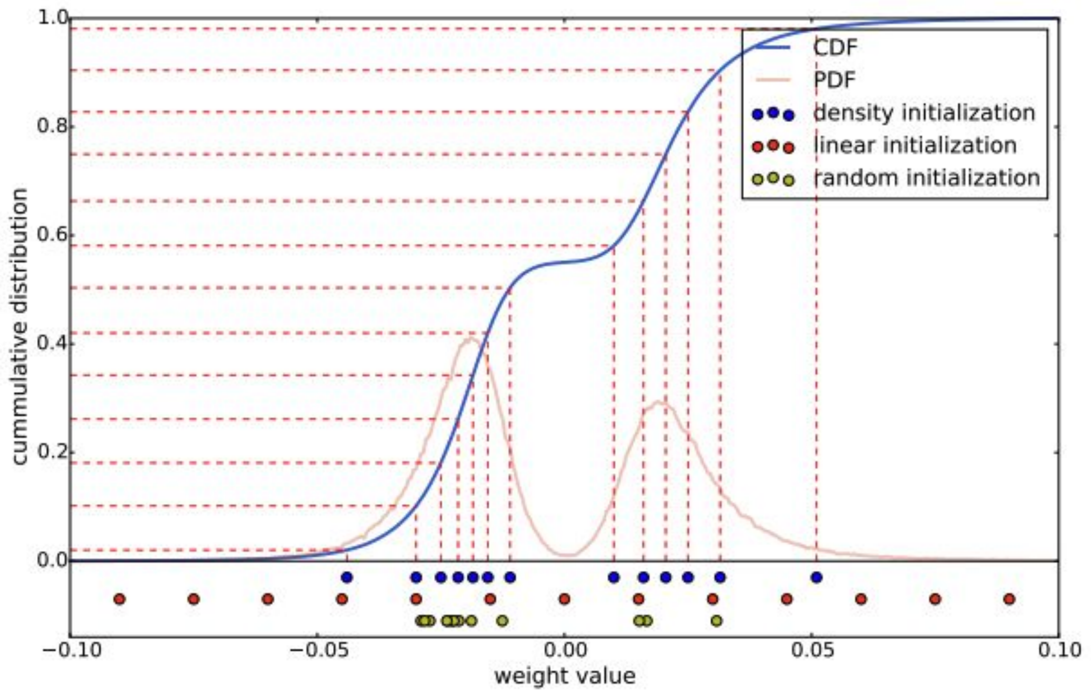
Linear: Space points linearly on the range of weights’ values [min_weight_val, max_weight_val].

- This method is invariant to the distribution of the weights - most scattered

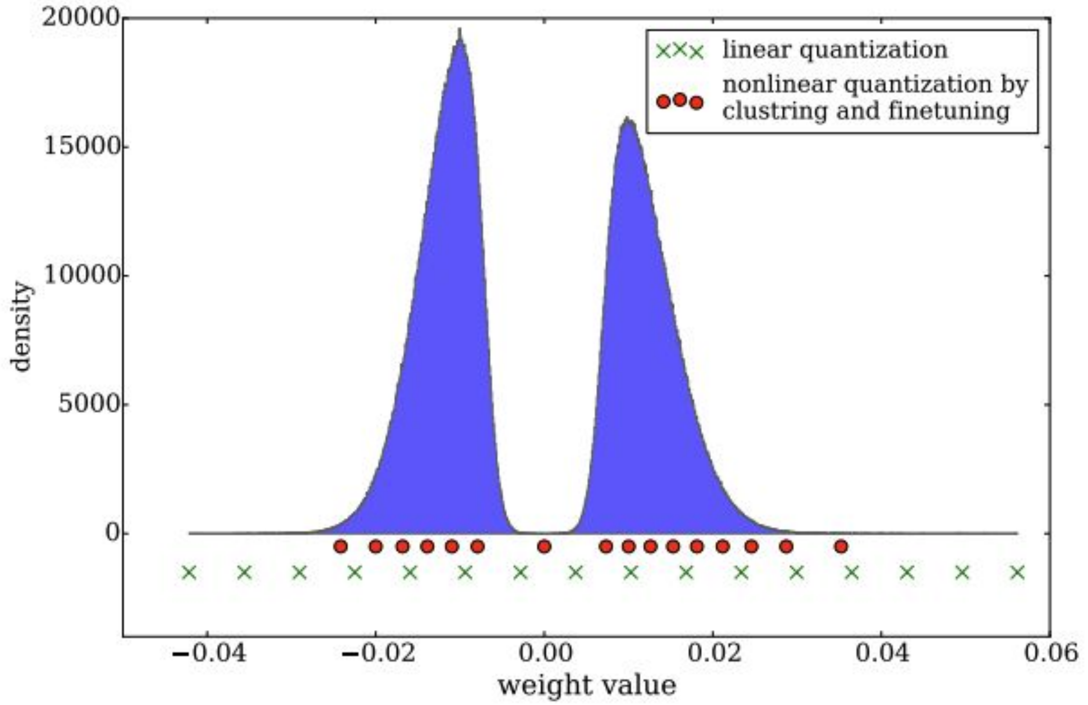
Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

Neural Networks Quantization

K-Means-based Weight Quantization: Centroids Initialization



Three different methods for centroids initialization

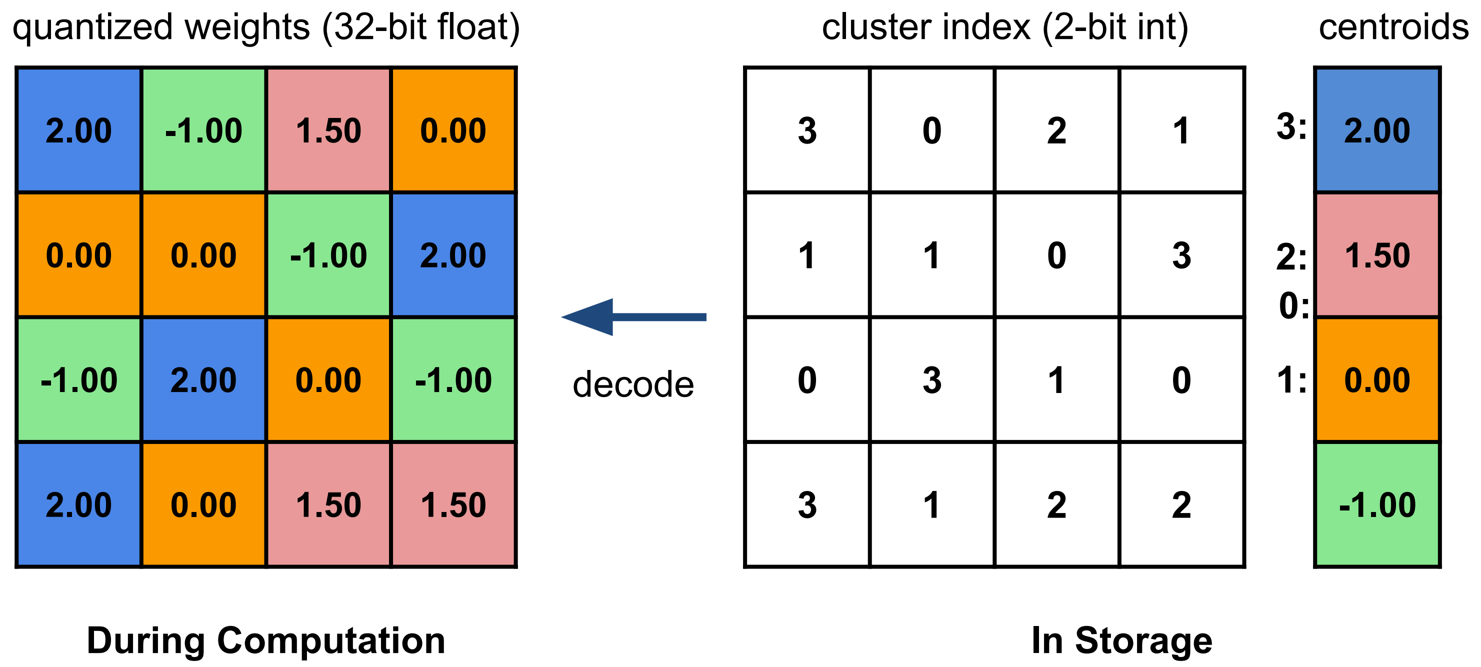


Initial distribution of weights, distribution of codebook before fine-tuning (greencross), and after fine-tuning (red dot)

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)

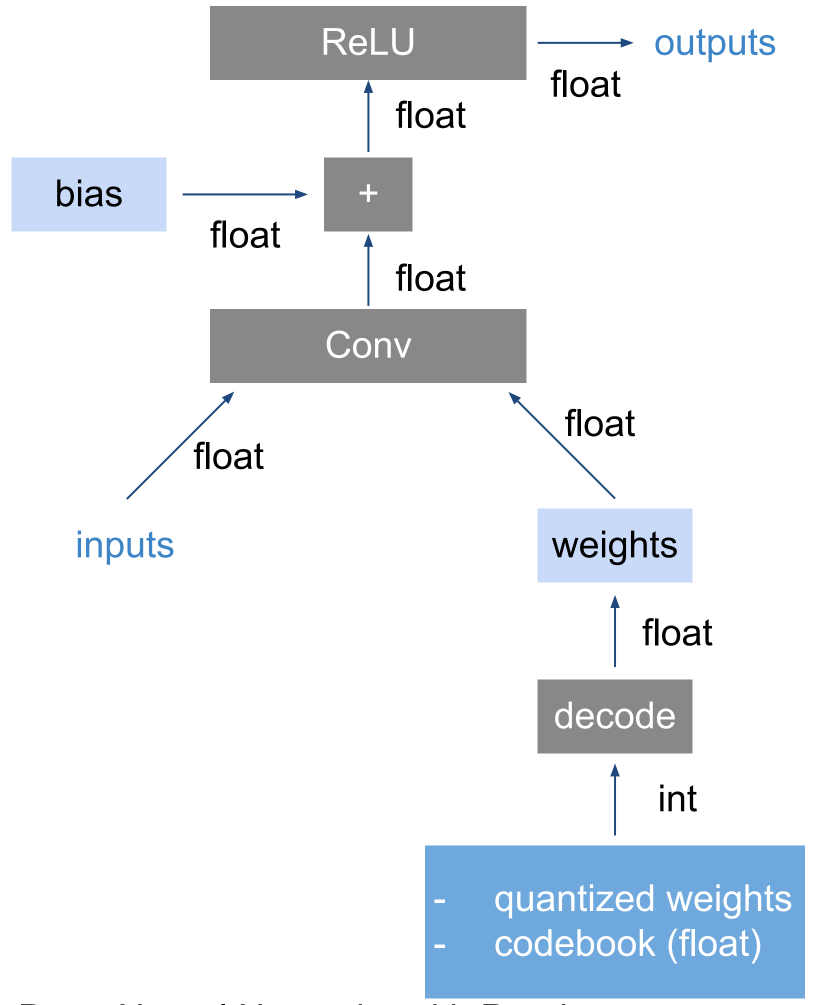
2.00	-1.00	1.50	0.00
0.00	0.00	-1.00	2.00
-1.00	2.00	0.00	-1.00
2.00	0.00	1.50	1.50

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: 1:	0.00
	-1.00



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.

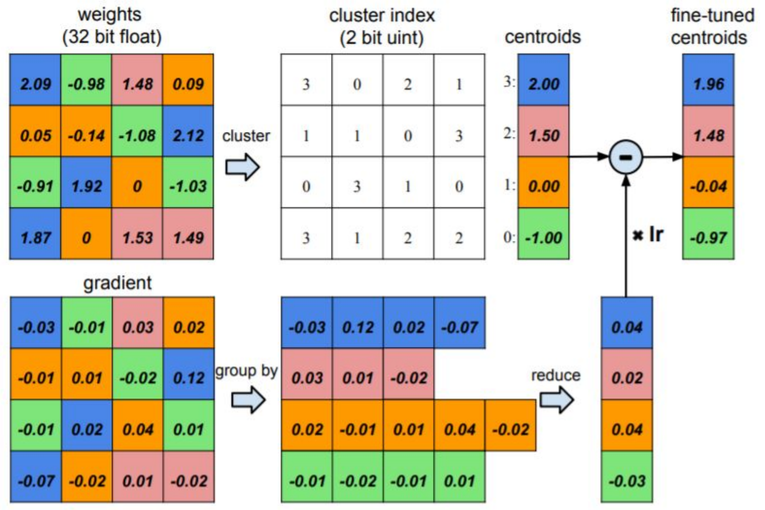
Compression Technique: Quantization

Quantization-Aware Training:

There could be an accuracy loss in a post-training model quantization and to avoid this and if you don't want to compromise the model accuracy we do quantization aware training.

This technique ensures that the forward pass matches precision for both training and inference.

https://www.tensorflow.org/model_optimization/guide/quantization/training

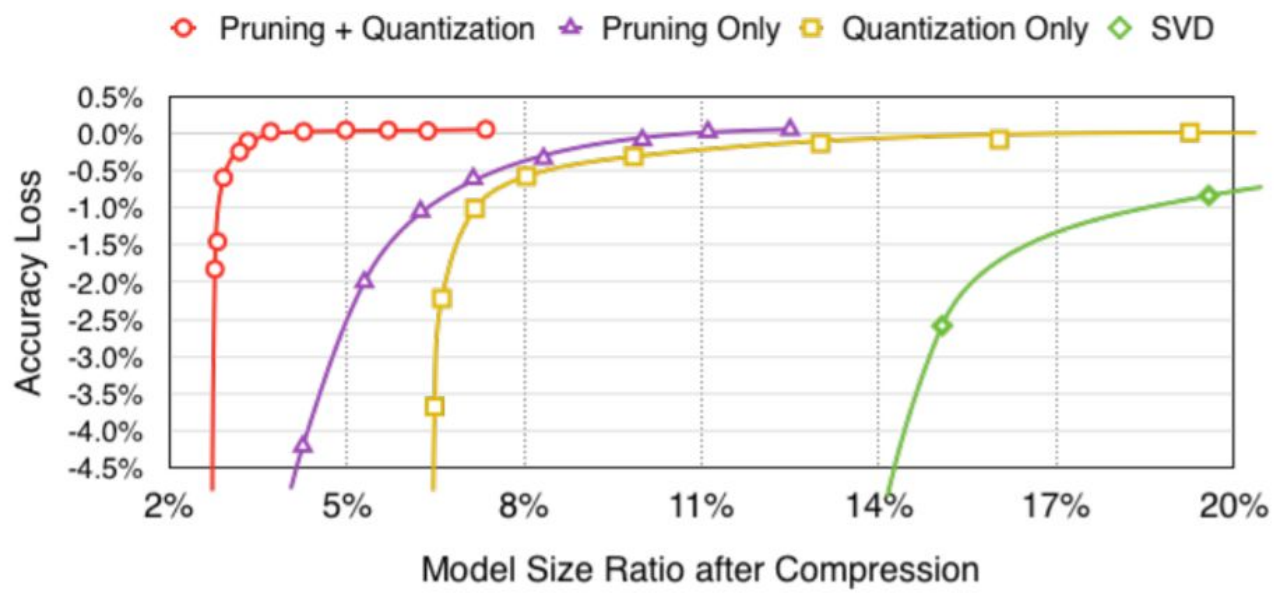


Han S. et al, *Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding*, 2016

Compression Technique: Quantization

Quantization can be **tricky:**

- Requires having a decent **understanding of hardware and bitwise computations**
- **Savings are tied to the features of the hardware** being used



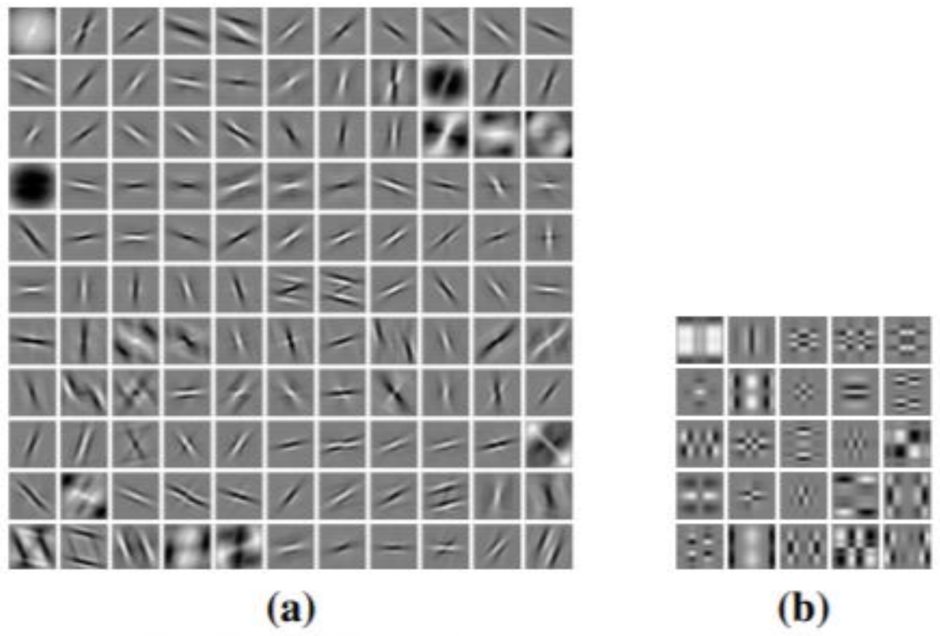
Han S. et al, *Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding*, 2016

Compression Technique: Low Rank Approximation

Main idea is to **approximate the redundant filters of a layer** using a linear combination of fewer filters. Compressing layers in this way reduces the network's memory footprint, the computational complexity of convolutional operations and can yield significant **speedups**.

Examples:

- Singular Value Decomposition
- Tucker decomposition
- Canonical Polyadic decomposition

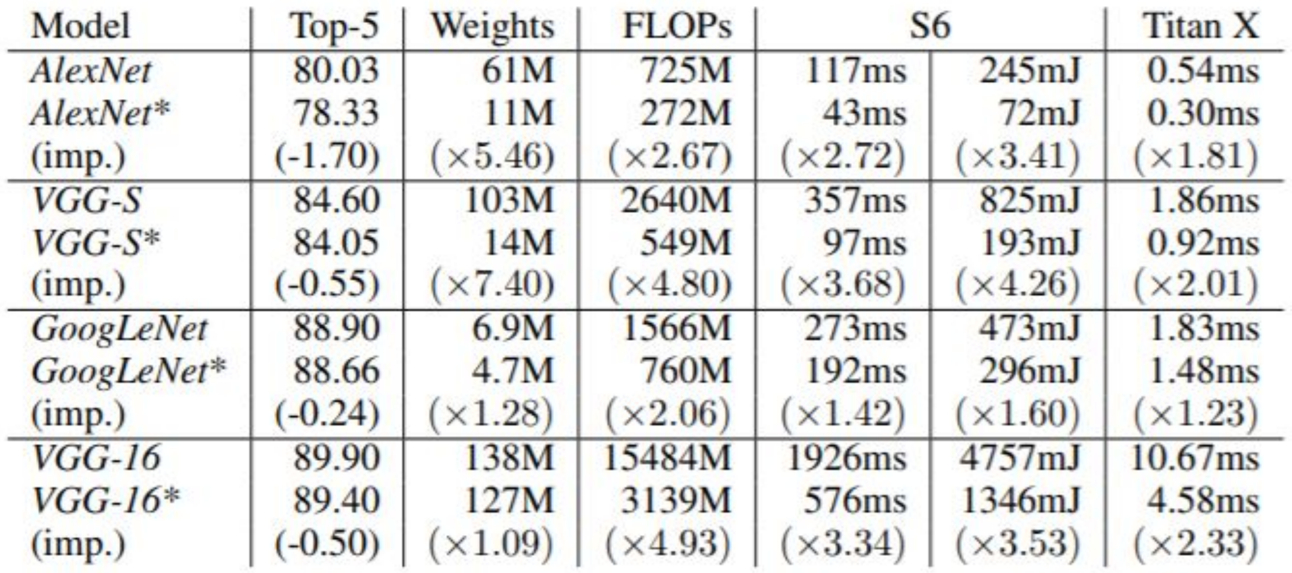


Rigamonti R. et al., *Learning Separable Filters*, 2013

Compression Technique: Low Rank Approximation

Kim et al. use Tucker decomposition to determine the ranks that the compressed layers should have. They apply the compression to various models for image classification tasks and run them on both a Titan X and Samsung Galaxy S6 phone*:

- Low-rank approximation achieve significant size and latency reductions
- Prove potential deployment on mobile devices
- Reduce parameters simplifying model structure
- Does not require specialized hardware to implement



Model	Top-5	Weights	FLOPs	S6		Titan X
AlexNet	80.03	61M	725M	117ms	245mJ	0.54ms
*AlexNet**	78.33	11M	272M	43ms	72mJ	0.30ms
(imp.)	(-1.70)	(×5.46)	(×2.67)	(×2.72)	(×3.41)	(×1.81)
VGG-S	84.60	103M	2640M	357ms	825mJ	1.86ms
*VGG-S**	84.05	14M	549M	97ms	193mJ	0.92ms
(imp.)	(-0.55)	(×7.40)	(×4.80)	(×3.68)	(×4.26)	(×2.01)
GoogLeNet	88.90	6.9M	1566M	273ms	473mJ	1.83ms
*GoogLeNet**	88.66	4.7M	760M	192ms	296mJ	1.48ms
(imp.)	(-0.24)	(×1.28)	(×2.06)	(×1.42)	(×1.60)	(×1.23)
VGG-16	89.90	138M	15484M	1926ms	4757mJ	10.67ms
*VGG-16**	89.40	127M	3139M	576ms	1346mJ	4.58ms
(imp.)	(-0.50)	(×1.09)	(×4.93)	(×3.34)	(×3.53)	(×2.33)

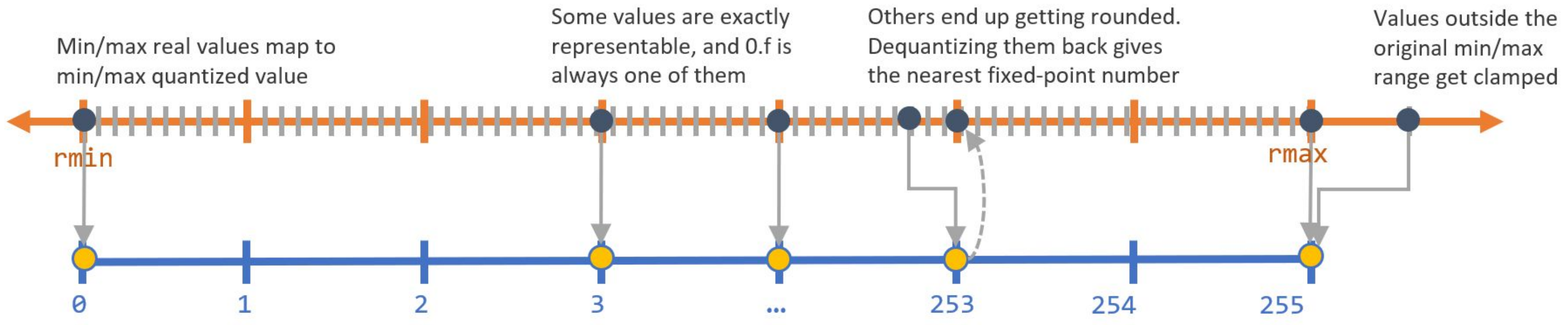
* S6 has a GPU with 35× lower computing ability and 13× smaller memory bandwidth than Titan

Kim et al, *Compression of deep convolutional neural networks for fast and low power mobile applications*, 2016

Compression Technique: Quantization

Quantization can be achieved by changing the output or NN architecture:

- **Post Training Quantization:** reducing the size of the weights stored (e.g. from 32-bit floating point numbers to 8-bit)


Min/max real values map to min/max quantized value
Some values are exactly representable, and 0.f is always one of them
Others end up getting rounded. Dequantizing them back gives the nearest fixed-point number
Values outside the original min/max range get clamped
rmin
rmax
0
1
2
3
...
253
254
255

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