Knowledge Distillation: Compact Model Performance for Scalable, Green, and Efficient Deployment

11.04.2025

Fadi Boutros

What will we cover today?

- ❑ Why do we need resource-efficient ML?
- ❑ How can we measure deep learning model efficiency?
- ❑ How can we enhance tiny model performance with Knowledge distillation?
- ❑ How can we distill the knowledge from large to small model?
- ❑ What are the main challenges in knowledge distillation?

Deep learning is Everywhere

https://news.mit.edu/2024/researchers-enhance-peripheral-vision-ai-models-030

Vision

https://www.earth.com/news/language-impact-perception-earth/

Language

https://educational-innovation.sydney.edu.au/teaching@sydney/multimodal-assessment-what-w

Multimodal

Deep learning is Everywhere

But, Highly accurate DL models are computationally costly!

https://news.mit.edu/2024/researchers-enhance-peripheral-vision-ai-models-030

Vision

https://www.earth.com/news/language-impact-perception-earth/

Language

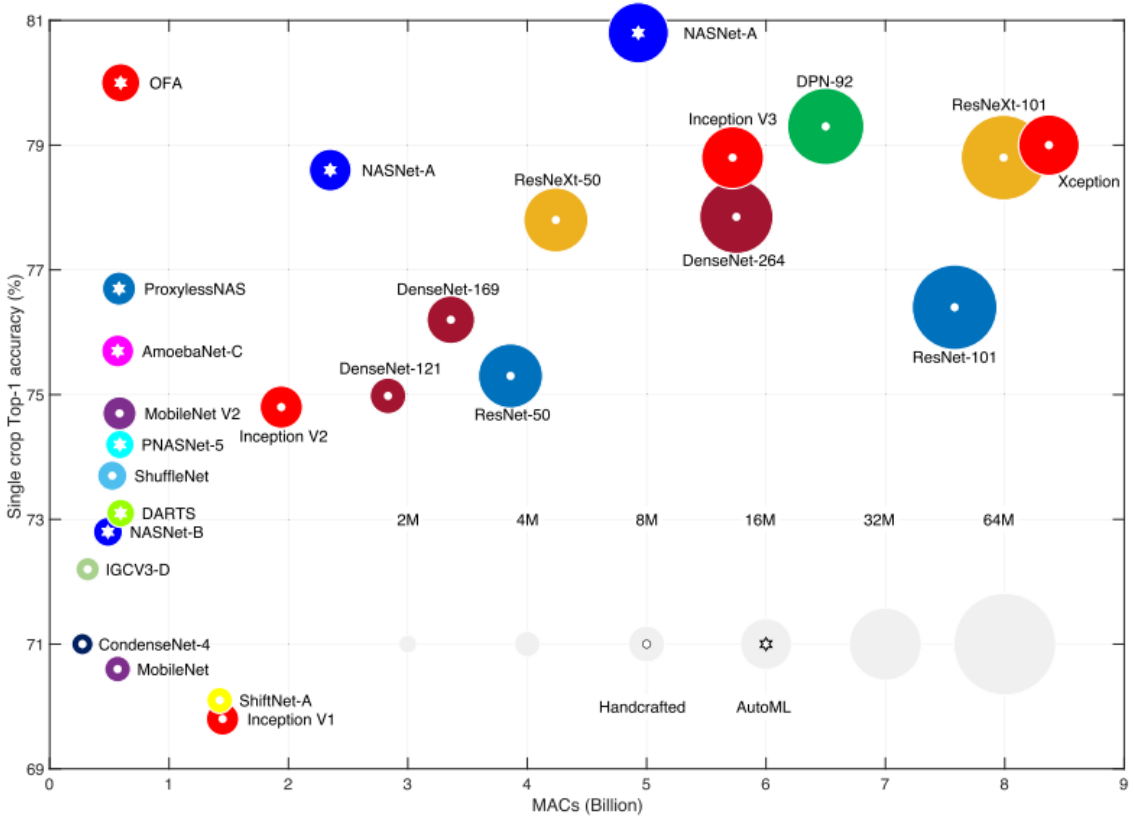
https://educational-innovation.sydney.edu.au/teaching@sydney/multimodal-assessment-what-w

Multimodal

Deep learning is Everywhere

But, Highly accurate DL models are computationally costly!

Vision



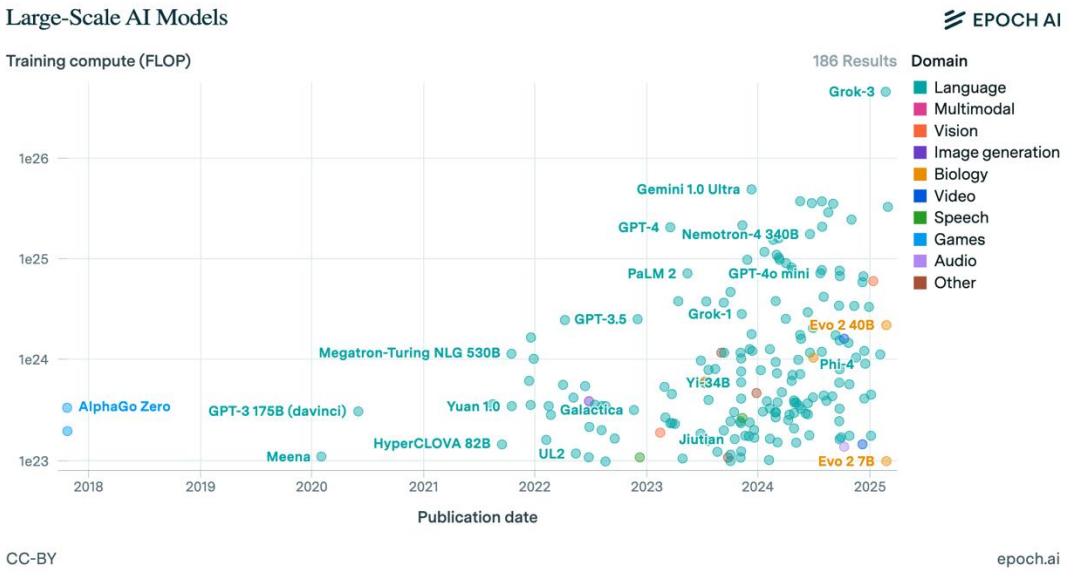
Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Dengetal.,IEEE 2020]

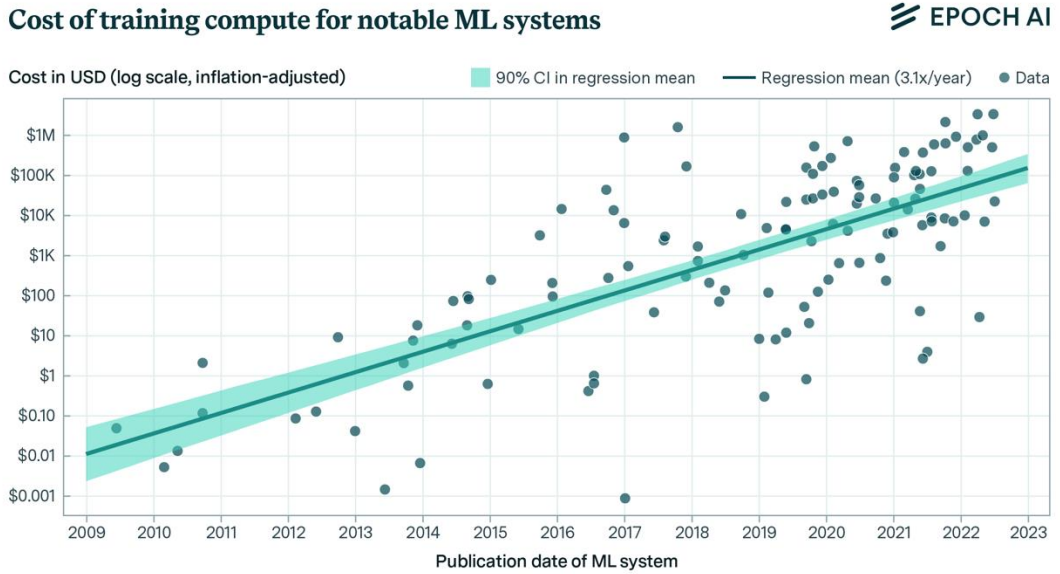
Multimodal

Recent Trends

Huge computations, data scientists, Tremendous training data

Deep learning
Computational Cost





Deep learning is Everywhere
Hosting Cost

Cost= development + deployment

Instance	vCPU(s)	RAM	GPU	Linux VM Price	Machine Learning Service Surcharge	Pay As You Go Total Price
NV6	6	56 GiB	1X M60	$832.200/month	$0/month	$832.200/month
NV12	12	112 GiB	2X M60	$1,664.400/month	$0/month	$1,664.400/month
NV24	24	224 GiB	4X M60	$3,328.800/month	$0/month	$3,328.800/month

https://azure.microsoft.com/en-us/pricing/details/machine-learning/#pricing

Deep learning is Everywhere

How to make them cheaper and faster? → Efficient ML

https://news.mit.edu/2024/researchers-enhance-peripheral-vision-ai-models-030

https://www.earth.com/news/language-impact-perception-earth/

https://educational-innovation.sydney.edu.au/teaching@sydney/multimodal-assessment-what-

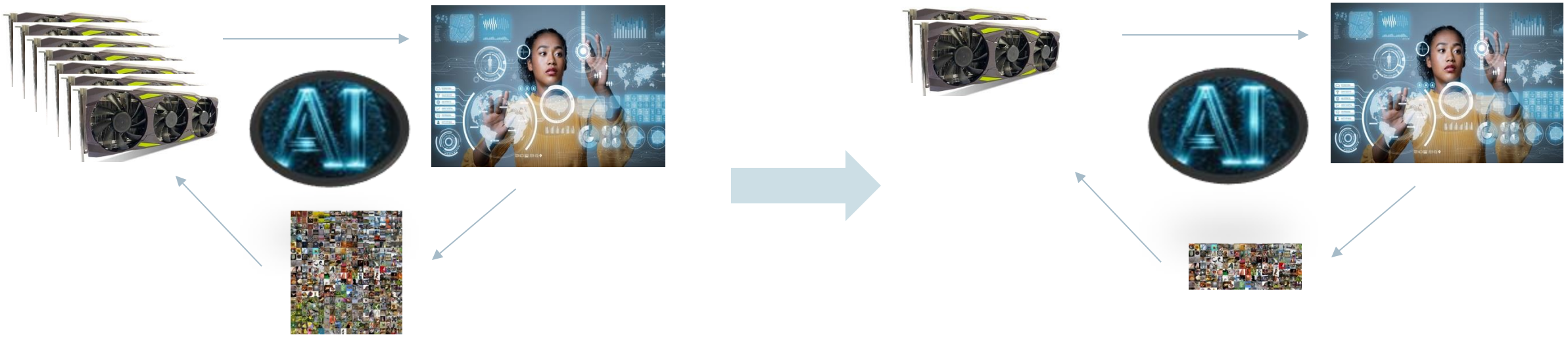
Vision

Language

Multimodal

Efficient ML

- Efficiency in this perspective is dealing with maintaining high accuracy without increasing the computational demands and, ideally, decreasing them



Efficient ML

- Efficiency in this perspective is dealing with maintaining high accuracy without increasing the computational demands and, ideally, decreasing them

3-15$ per million tokens

vs.

0.48$ per million tokens

Efficient ML

- Efficiency in this perspective is dealing with maintaining high accuracy without increasing the computational demands and, ideally, decreasing them
- Why?

Efficient ML

- Efficiency in this perspective is dealing with maintaining high accuracy without increasing the computational demands and, ideally, decreasing them

- Why?
  - Memory: often share memory with others
  - Latency: some requires realtime (translation, self-driving)
  - Cost: power machines are more expensive
  - Energy: both computation and accessing memory needs

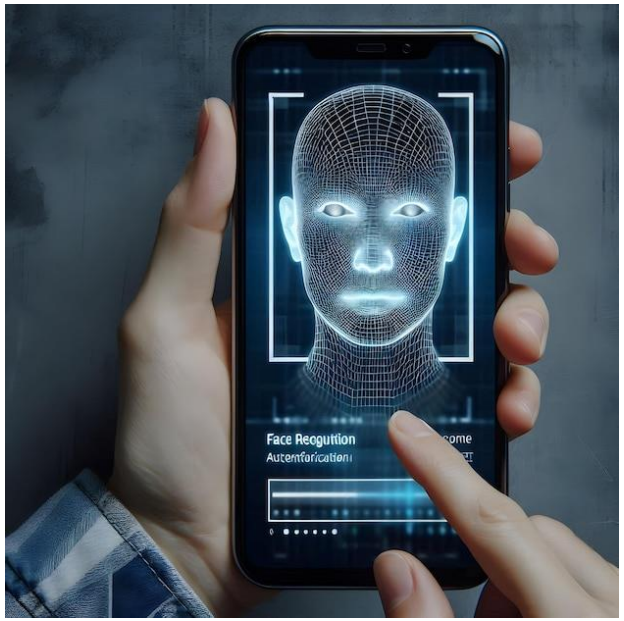
a significant amount energy, especially for devices powered by batteries

Why do we need efficient ML?

- From an application point of view, we need efficient ML for the following reasons:

1. Have limited resources

- E.g. smart phones…
- Limited memory to share with other applications
- Limited battery that we want to last longer
- Limited computational power that is used by other apps
- Move towards edge computing

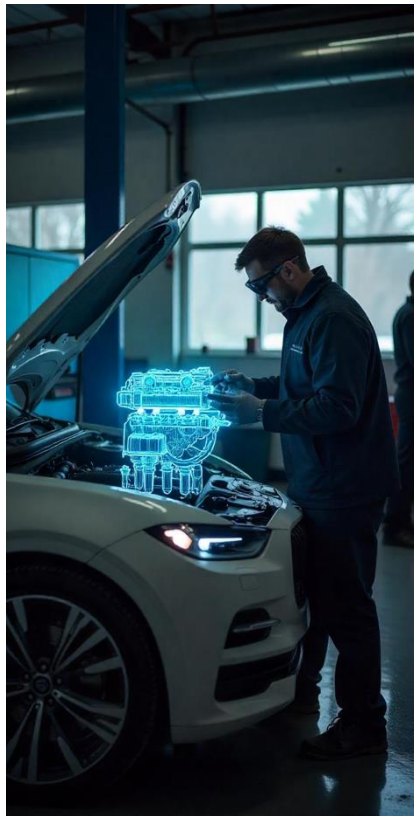


Why do we need efficient ML?

- From an application point of view, we need efficient ML for the following reasons:

2. Minimize resources cost

- E.g. Automotive domain
- Constant monitoring
- Attention + identity
- Minimize cost of special automotive
- electronics..

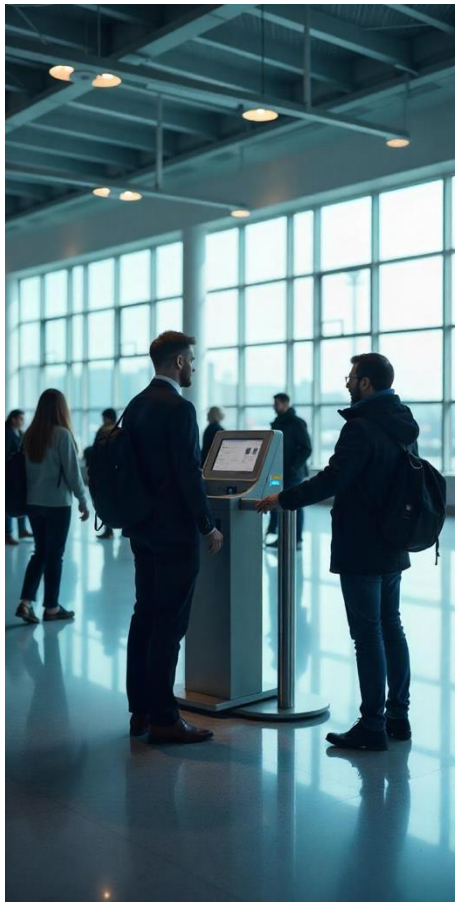


Why do we need efficient ML?

- From an application point of view, we need efficient ML for the following reasons:

3. Enable large scale application

  - E.g. security (border control, ID management)
  - Deployability is based on cost + resources
  - Efficiency = Deployability = Security
  - Deployability = larger scale

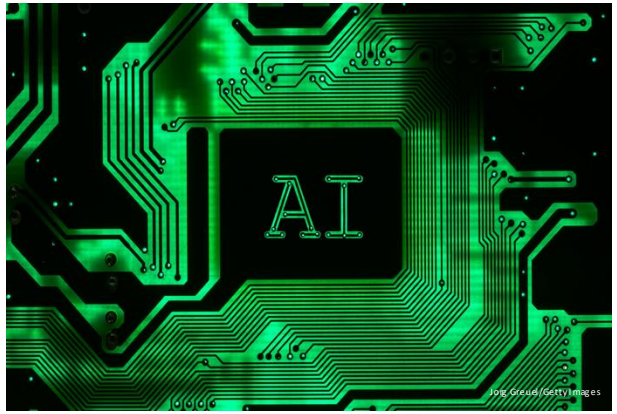


Why do we need efficient ML?

- From an application point of view, we need efficient ML for the following reasons:

4. Environment/power friendly operations (Green AI)

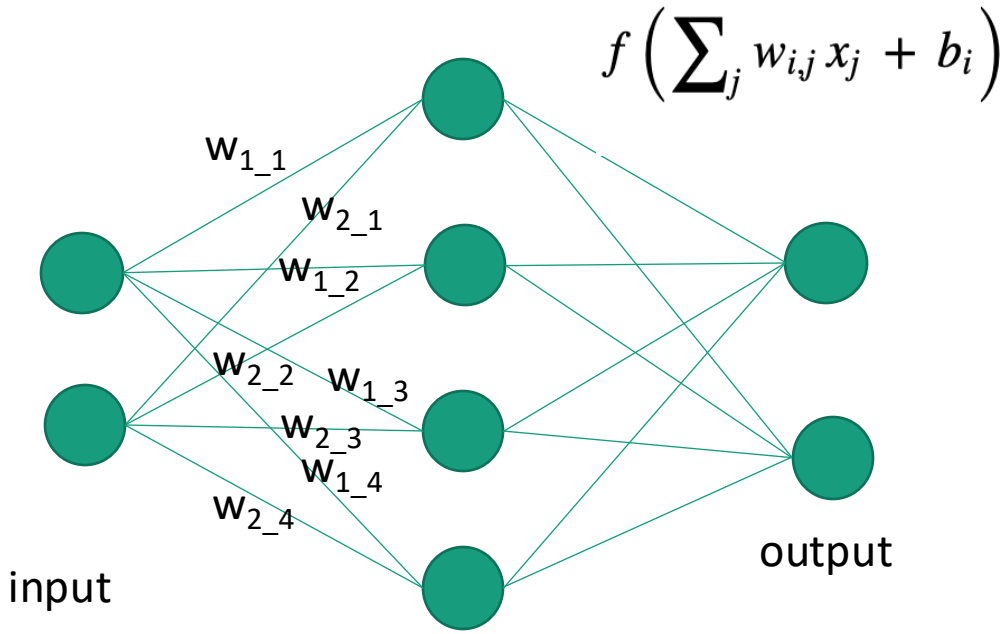
- E.g. from anything with a battery to huge cooled servers
- A 2018 blog post from OpenAI revealed that the amount of compute required for the largest AI training runs has increased by 300,000 times since 2012
- An average American is responsible for about 36,000 tons of CO2 emissions per year; training and developing one machine translation model that uses a technique called neural architecture search was responsible for an estimated 626,000 tons of CO2
- *Training Stable Diffusion costs $600,000 (256 A100s,150k hours)*



Model complexity estimation
Deployability

- Number of parameters → Memory footprint
- Floating Point Operations (FLOPs) --> Computational complexity in forward phase
- Inference time → Required time for inference (platform-dependent metric)

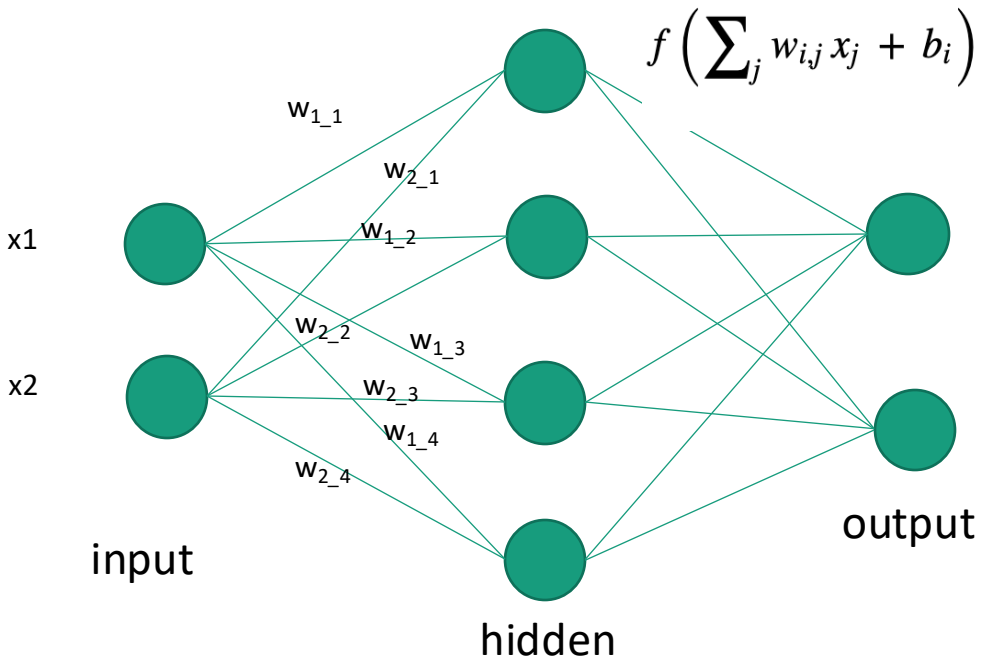
Number of parameters – toy example



- Input X of dimension 2
- 4-D hidden layer
- 2-D output layer

Number of parameters - toy example

The output of each neuron is calculated by multiplying the corresponding inputs with their weights, add bias offset, and pass them through activation function

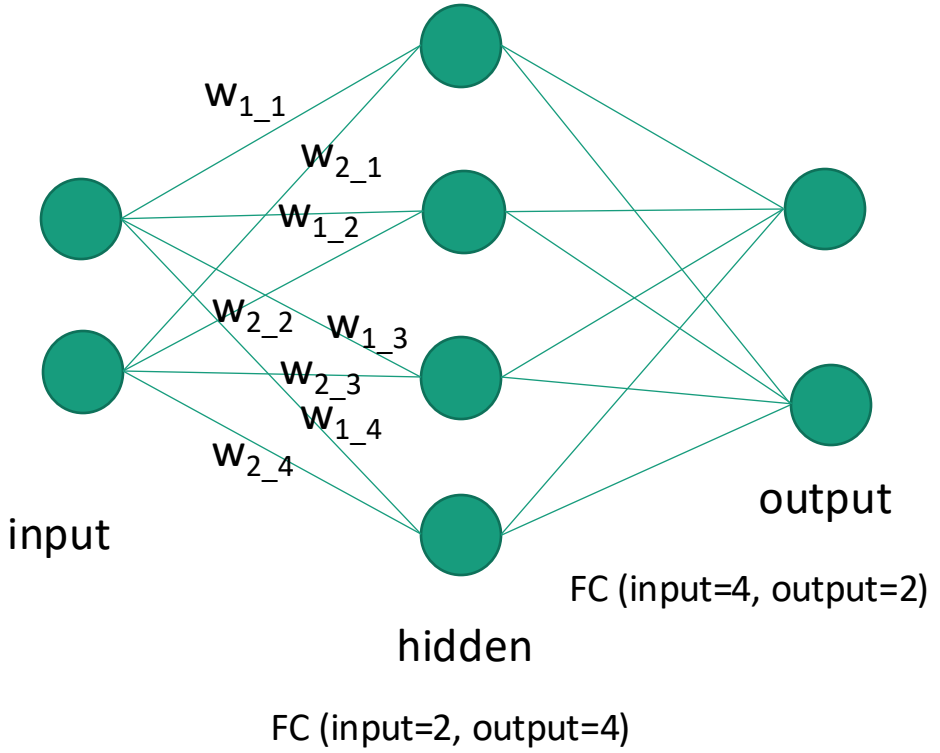


w: network weight
b: bias term
f: activation function

- ❑ Input X of dimension 2
- ❑ 4-D hidden layer
- ❑ 2-D output layer

Number of parameters - toy example

- Number of parameters= number of connection between layers + number of biases in each layer
- Parameters= O x I + O = (I+1) x O
  - I: input dimensionality
  - O: output dimensionality

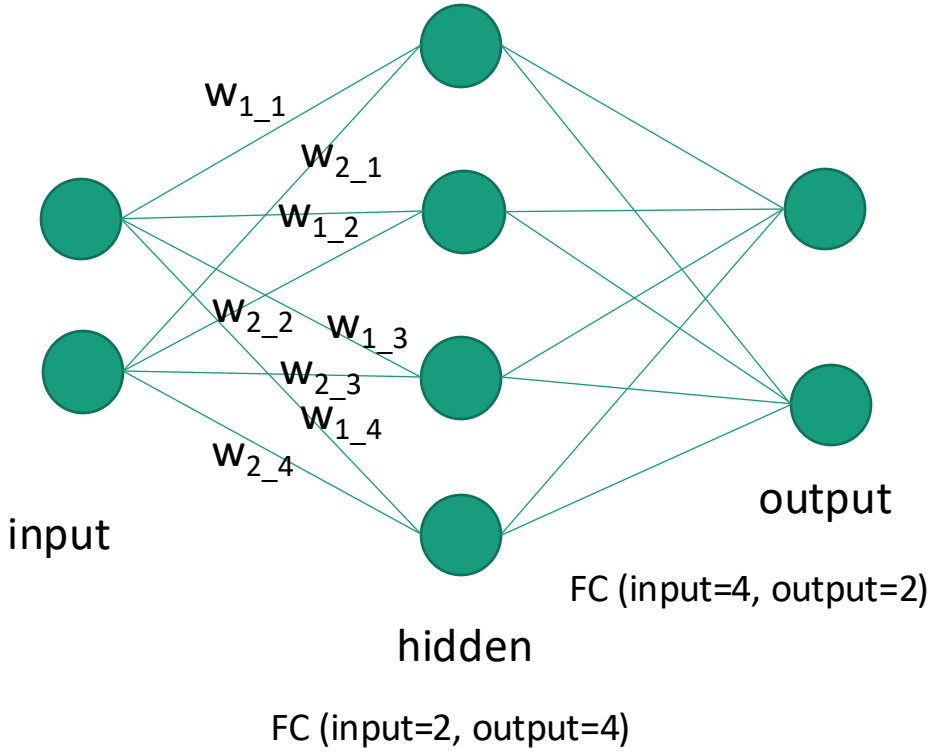


Number of parameters - toy example

- Number of parameters= number of connection between layers + number of biases in each layer
- Parameters= Ox I + O = (I+1) x O
  - I: input dimensionality
  - O: output dimensionality
- Example: Number of parameters

= (2 x 4 + 4 x 2)+ (4 +2)

= 22 parameters



Memory footprint

- Size of memory that is required by model during the inference phase
- Memory footprint= number of parameters x datatype size

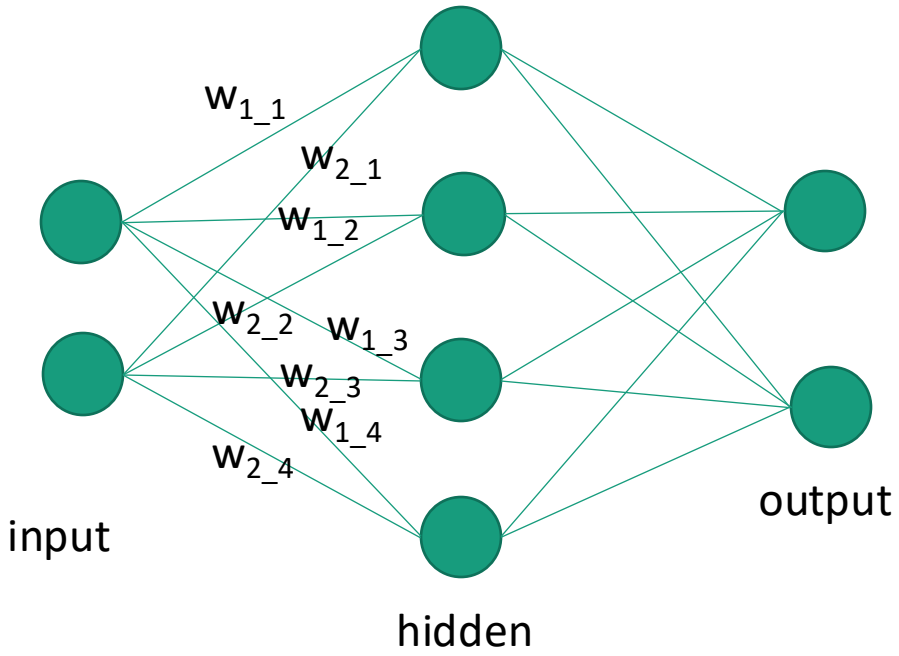
Data type	Description	Size (in Byte)
int8	8-bit integer	1
uint8	8-bit unsigned integer	1
int32	Integer	4
float32	single-precision float	4

Example of Python datatype with size in Byte

Toy example

Given that all weights and biases are represented using 4-byte float

Calculate the required memory footprint to run the following model:

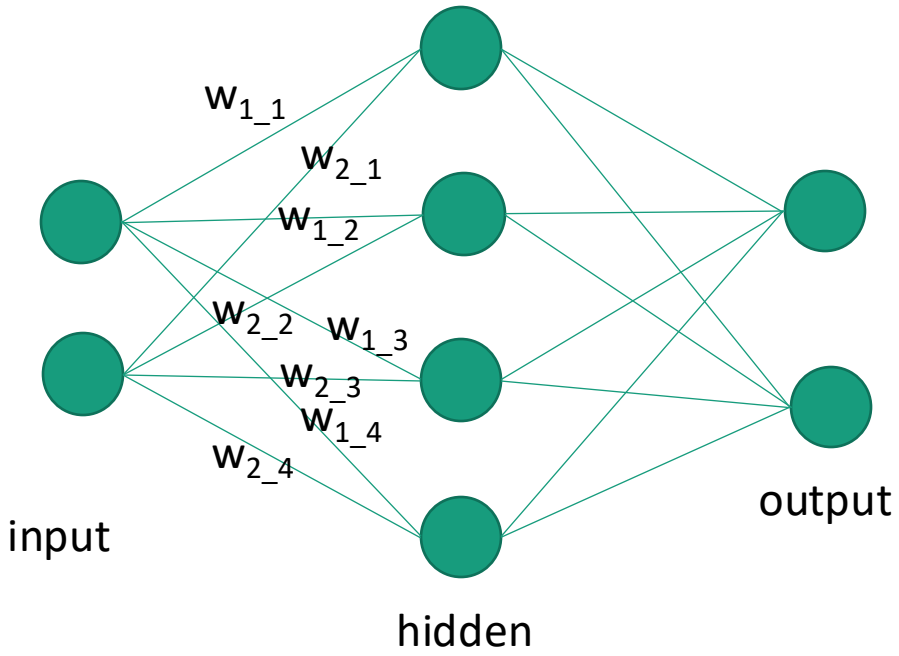


Toy example

Given that all weights and biases are represented using 4-byte float

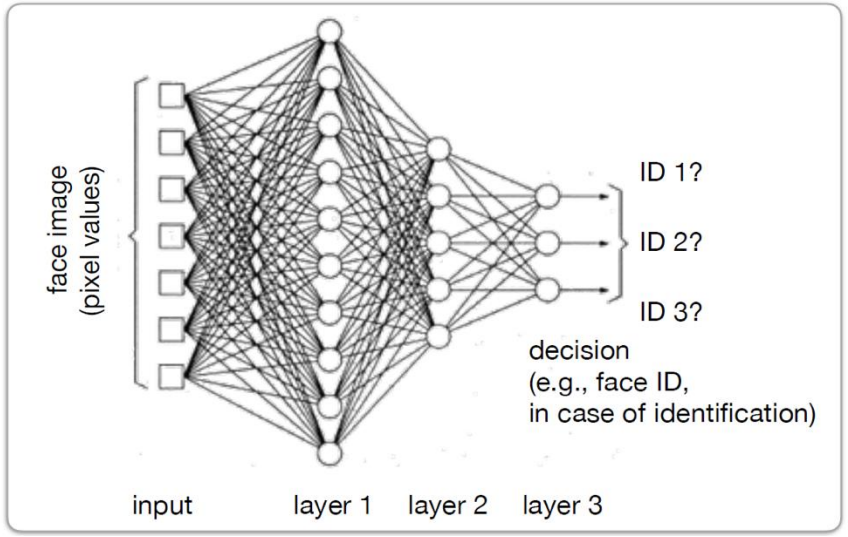
Calculate the required memory footprint to run the following model:

4 x 22= 88 Byte



Fully connected neural network for image classification

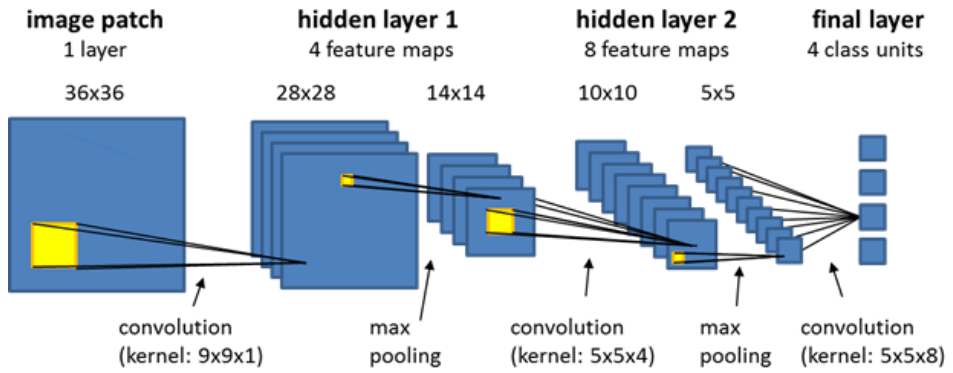
- Computational cost is very high
  - Example: When the input data has shape of 112x112x3 and the first hidden layer has 500 neurons → 112x112x3x500 = 18816000 weights ☹
- Required huge dataset for training



Convolutional neural networks

Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input

- Usually It consists of **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**
- Convolutional layer arrange the neurons in shape of width x hight, depth
  - It consists of learnable filters repeated n times
  - Each filter (kernel) has shape of $kernel_w$ x $kernel_h$
  - Each filter will produce an activation map



Convolutional neural networks

- The output of convolutional layer is computed by sliding the kernel across the input of the previous layer and compute the dot product between the entries of the filter and the input
- Convolutional layer hyperparameters:
  - Stride: Control how the kernel is moved in each step i.g. stride=1 the kernel is moved 1 pixel each time, stride=2 the kernel is jumped 2 pixel ..etc.
  - Depth: number of filter
  - Zero-padding: pad the input volume with zeros around the border

Convolutional neural networks

Pooling layer performs a down-sampling operation along the spatial dimensions (width, height)

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

output

Compute the dot product between the entries of the filter and the input at the first position

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4		

output

Compute the dot product between the entries of the filter and the input at the first position

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	

output

Sliding the kernel and compute the dot product between the entries of the filter and the input

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2		

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2	2	

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2	2	2

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2	2	2
5		

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2	2	2
5	3	

output

Example

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

1	1	1
0	0	0
1	0	1

3x3 kernel

4	2	3
2	2	2
5	3	2

output

$$h_i = (h_{i-1} - k_h + 2p)/s + 1$$

$$w_i = (w_{i-1} - k_w + 2p)/s + 1$$

Exercise – zero-padding

1	0	1	0	1
0	0	1	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0

5x5 image

Given zero-padding of 1,
What is the output?

1	1	1
0	0	0
1	0	1

3x3 kernel

?

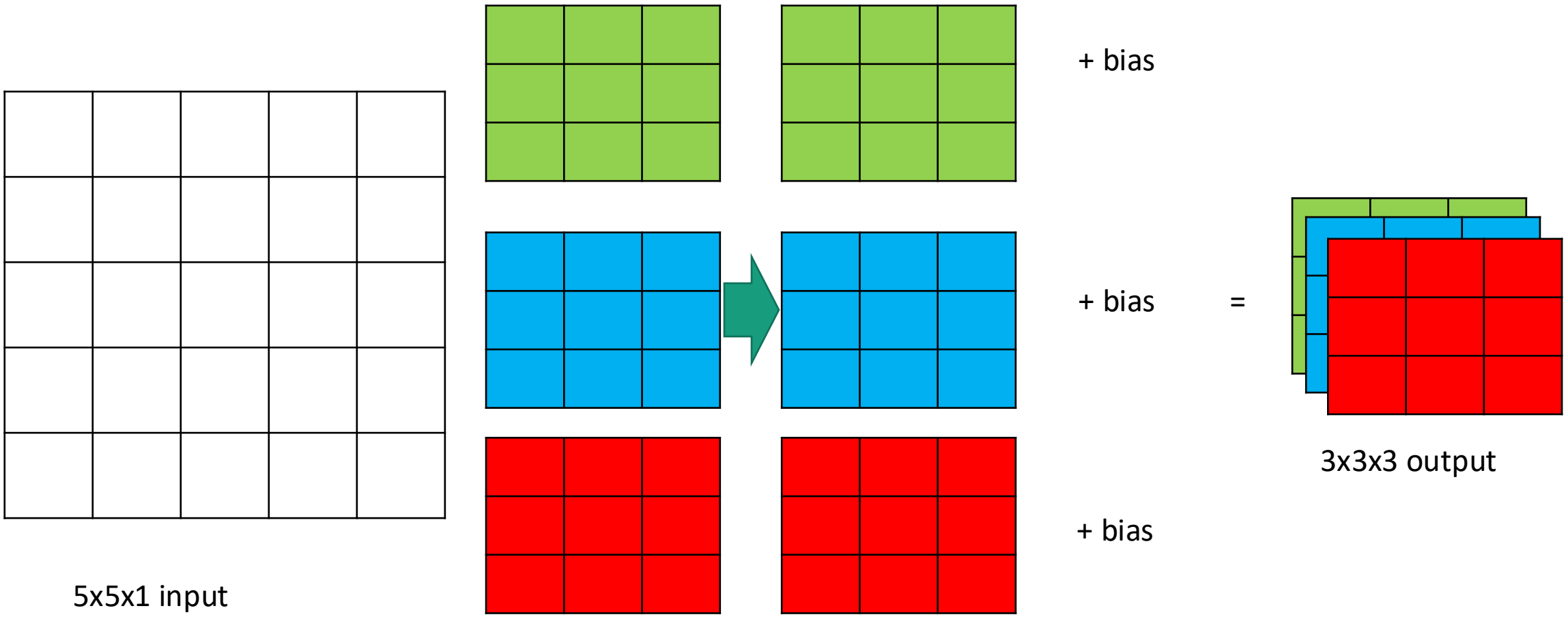
Calculating the number of parameters in convolutional layer

- The number of parameters in convolutional layer depends on:
  - Kernel size ($kernel_w$ x $kernel_h$)
  - Number of kernel ($depth_i$)
  - Depth of the previous layer ($depth_{i-1}$)
- Number of parameters= $depth_i$ x ($kernel_w$ x $kernel_h$) x $depth_{i-1}$ + biases
- Pooling layers are parameter free

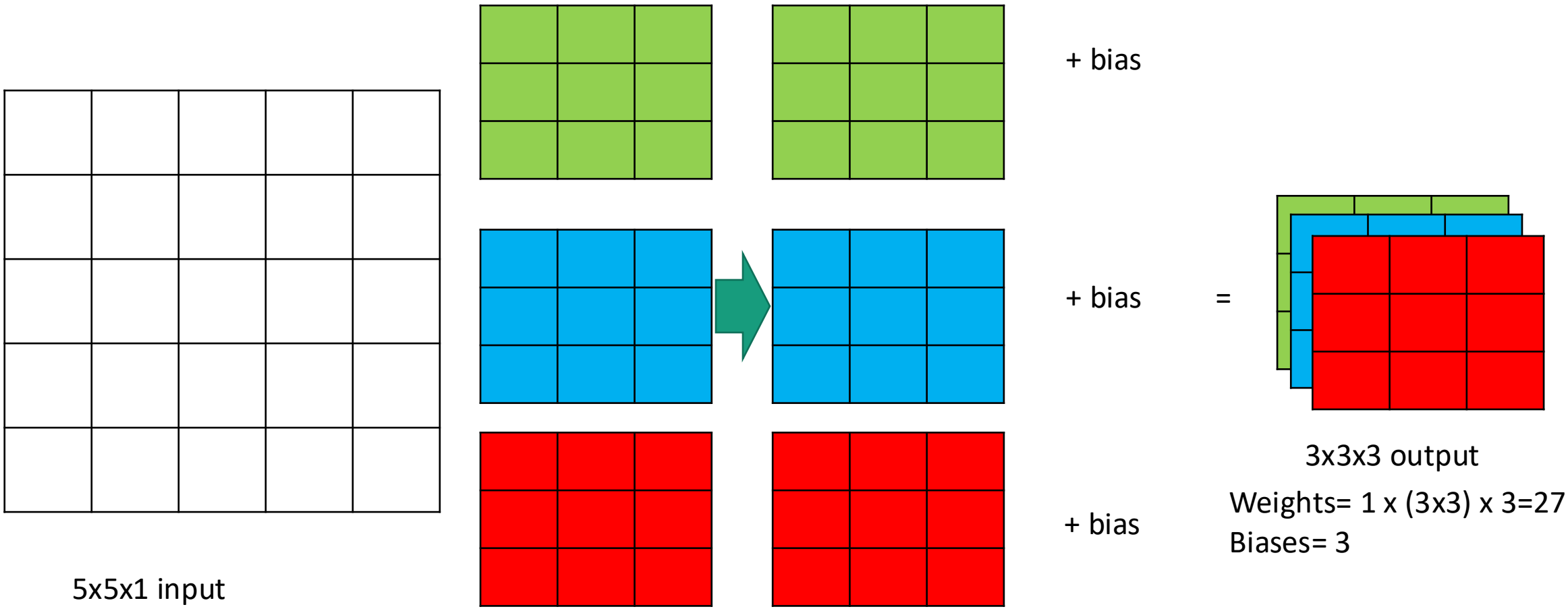
Calculating the number of parameters in convolutional layer

- Parameters in one filter of size(k,k)= k x k.
- The filter will convolve over all input channels concurrently(input depth). Thus, parameters in one filter will be k x k x d_{i-1}
  [filter size x input_data depth]
- Bias: One bias will be added to each filter
- Example: total parameters for one filter kernel of size (3,3) for depth 3=(3x 3 x 3)+1=28
- The total number of filters is d_i.

Example – number of parameters



Example – number of parameters



Floating Operation Points (FLOPs)

- Analysis the required number of floating operation point is essential to design efficient and low-complexity algorithms
- Floating Operation Points: the number of complex multiplications and summations required by for a single forward pass
- Typically, the FLOPs is typically computed as twice of multiply-adds (MACs) because a typical MAC operation (a * b + c) involves both a multiplication and an addition, each of which is a floating-point operation
- Example: The FLOPs of fully connected layer with input dimensionality of I and output dimensionality of O:
- FLOPs(FC)= (IxO) + O

Convolutional layer FLOPs

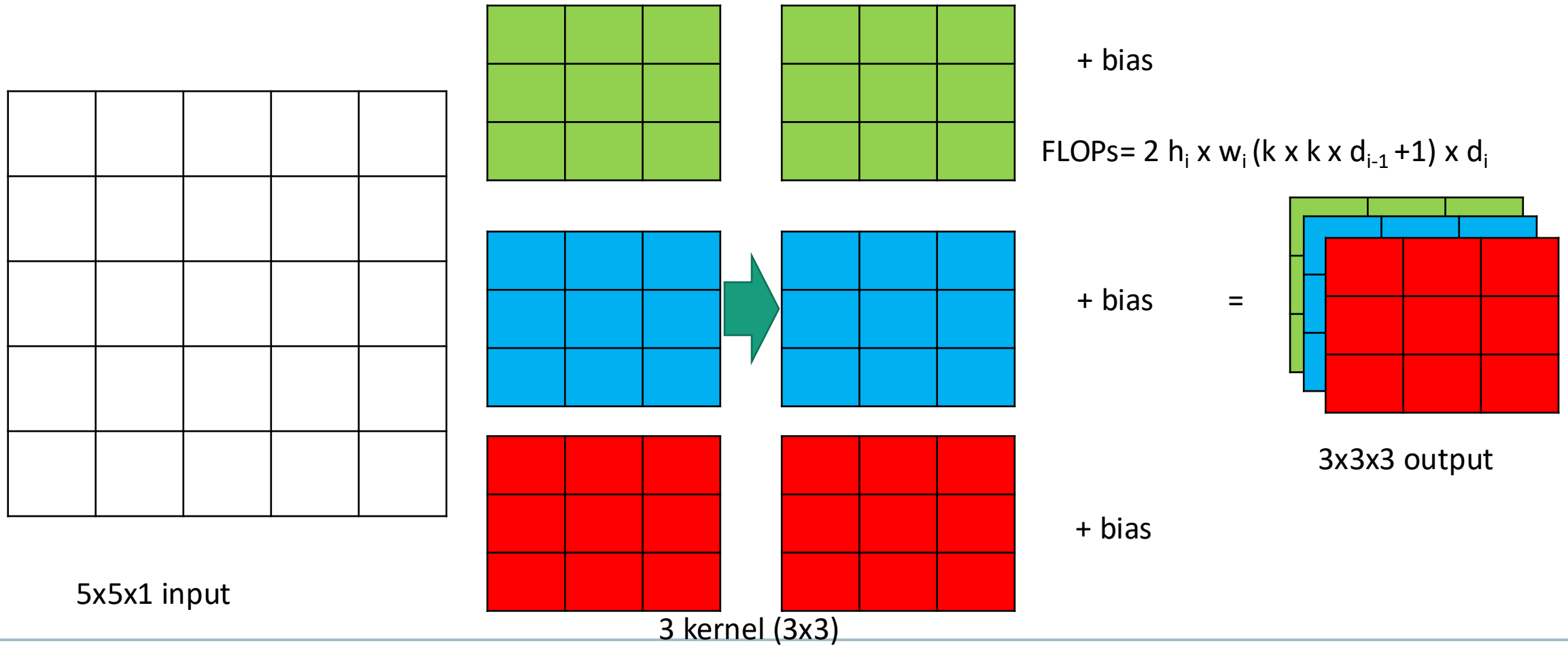
Given a kernel of size (k x k), output spatial width and high of (h_i x w_i), depth of previous layer d_{i-1} and depth of current layer d_i i.e., output feature maps, the number of FLOPs in this case is calculated as:

FLOPs= 2 x h_i x w_i x d_i x(k x k x d_{i-1} +1)

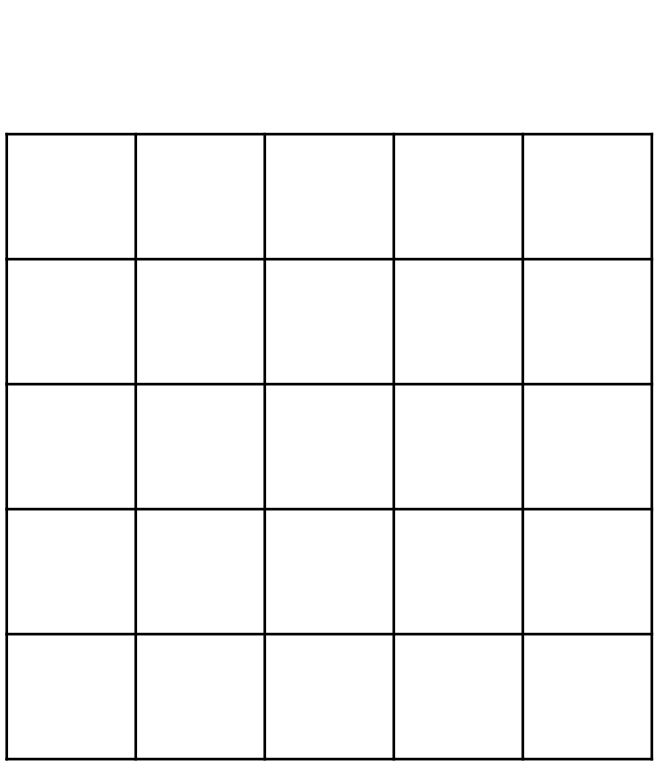
$$h_i = (h_{i-1} - k_h + 2p)/s + 1$$

$$w_i = (w_{i-1} - k_w + 2p)/s + 1$$

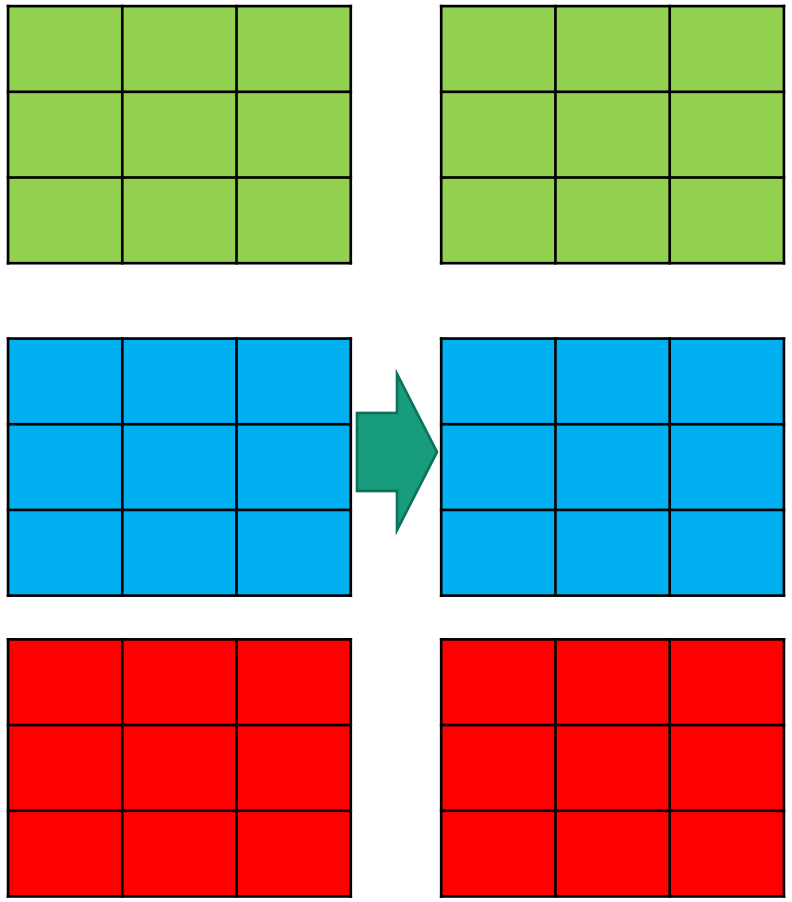
Example – FLOPs



Example – FLOPs



5x5x1 input

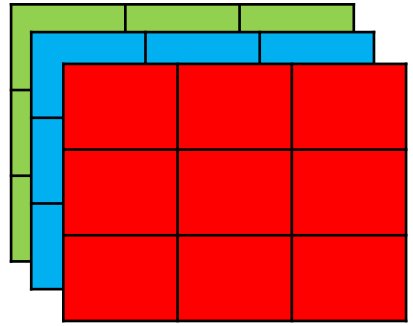


3 kernel (3x3)

+ bias

FLOPs= 2 h_i x w_i (k x k x d_{i-1} +1) x d_i

+ bias =



3x3x3 output

+ bias

FLOPs= 2x (3x3) x (1x3x3+1) x 3=540 FLOPs

How to enhance these measures (efficiency)?

- We do that keeping by pursuing three goals:

1. Make the model smaller….

   and/or

2. Make less computational operations…

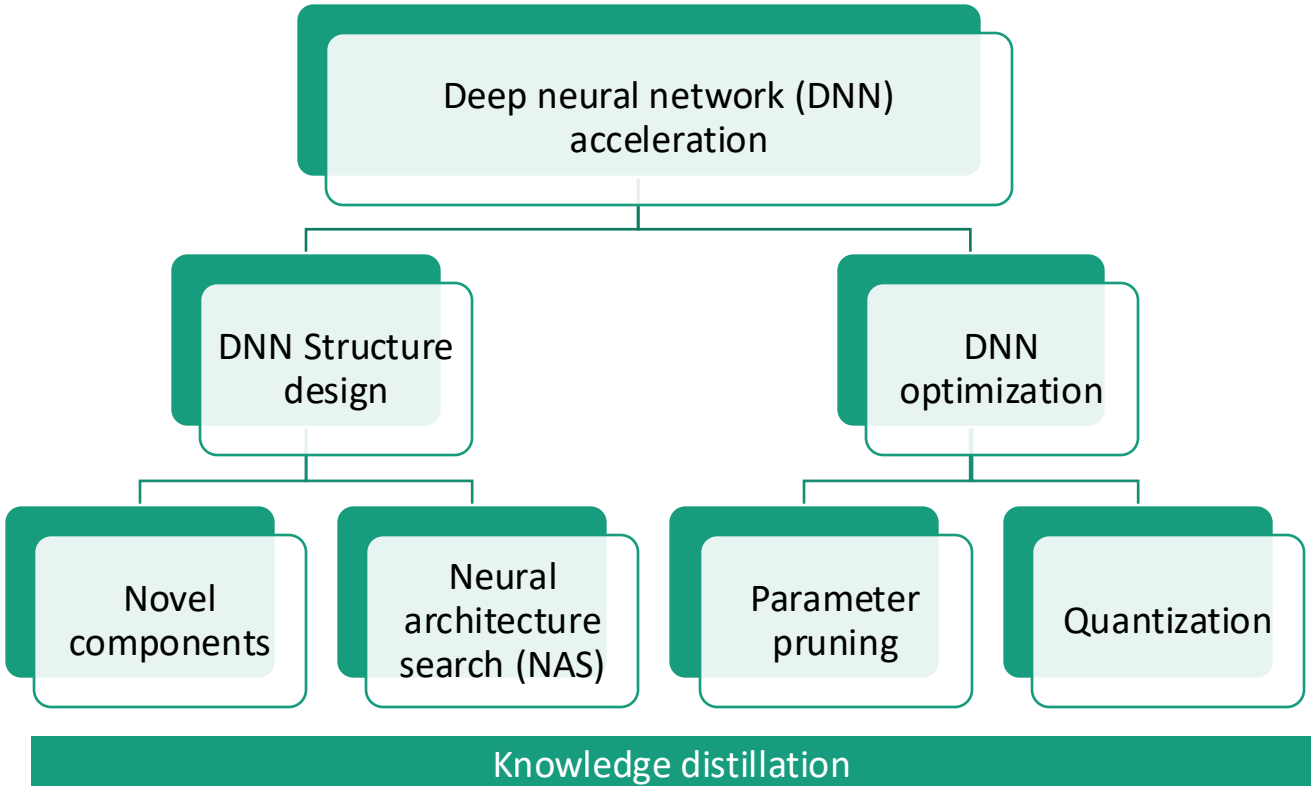
   But, larger models learn much better…. So we also need

3. Maintain some of the larger model wisdom

How to enhance these measures (efficiency)?

- Many ideas … and you can come up with new ones…
- Model efficiency by design
  - Using smaller feature maps, smaller kernels, few layers ..etc
  - Utilize neural architecture search to generate a model that has best trade-off between the accuracy and model size
- Model compression
  - Reducing the model size without significantly effect the model accuracy
- Enhance compact model performances → Today topic ☺

How to enhance these measures (efficiency)?



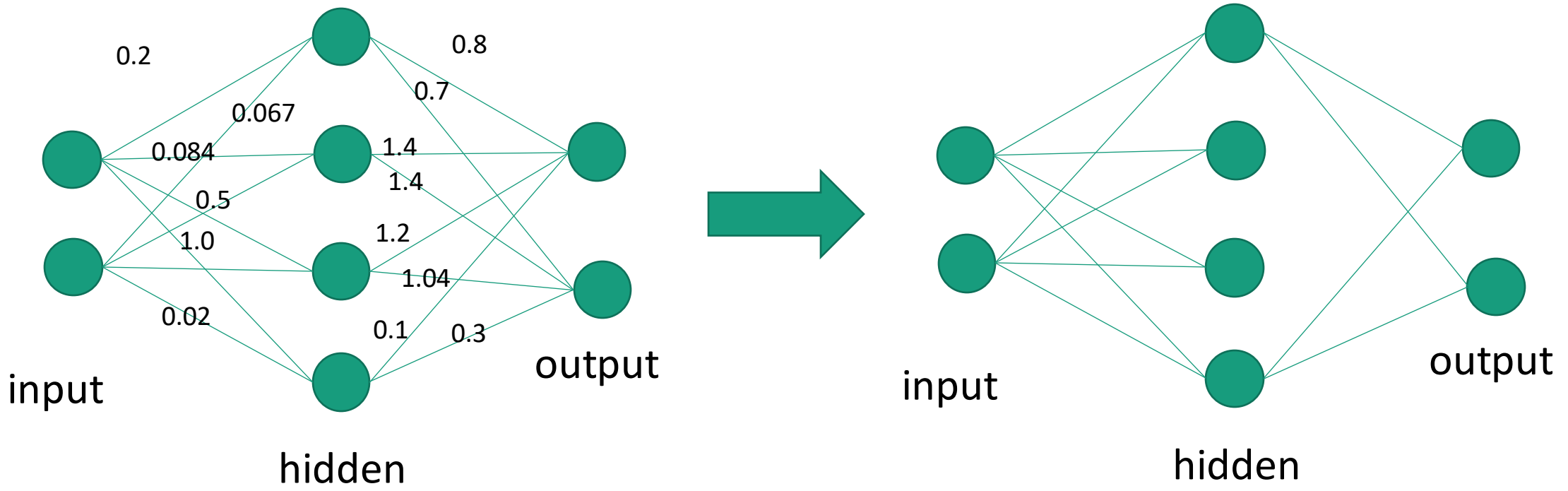
Parameter pruning

Parameter pruning is a model compression technique that reduces the number of parameters in a trained large network, resulting in a smaller network

Parameter pruning

- Many of neurons in deep neural network are redundant and they do not significantly contribute to the model accuracy, i.e. have fewer contribution to the final model output than other neurons
- Removing these neurons from the neural network, resulting in smaller and faster network
- Does parameter pruning effect the model accuracy?
  - To some degree, **Yes, however our goal here is to find the best trade-off between the model complexity and accuracy**

Parameter pruning



Parameter pruning - criteria

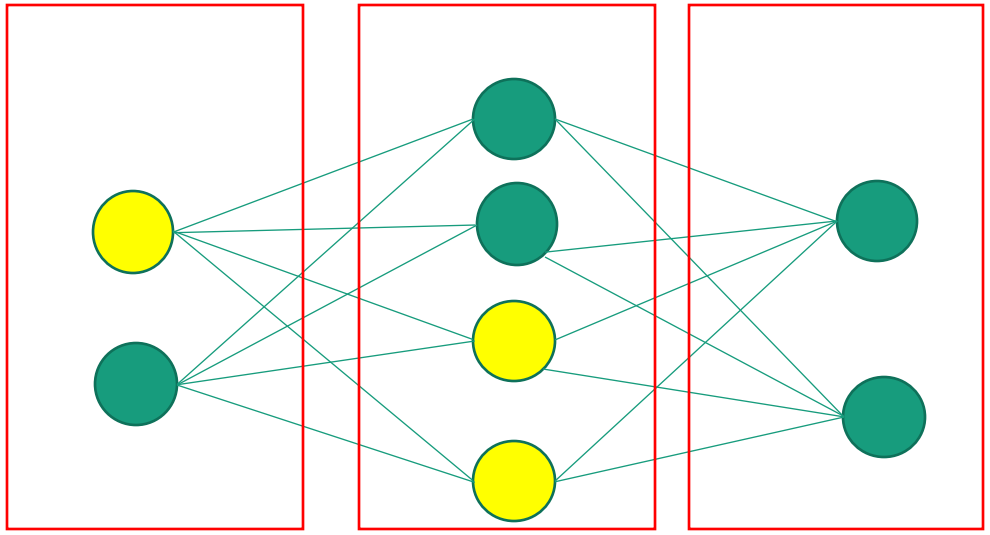
Challenge 1: How to select the neurons for removing them from the network?

- ❑ Randomly select subset of neurons
- ❑ Weights magnitude-based pruning: Trained weights with large values are more important than trained weights with smaller values. Rank the neurons based on their contribution to the model output i.e. L1/L2 mean of neuron weights
- ❑ Gradient based pruning: Consider the contributions of neurons to the gradient in the back-propagation. Accumulate gradients over a subset (minibatch) of training data and prune the neurons based on the product between this gradient and the corresponding weight of each parameter

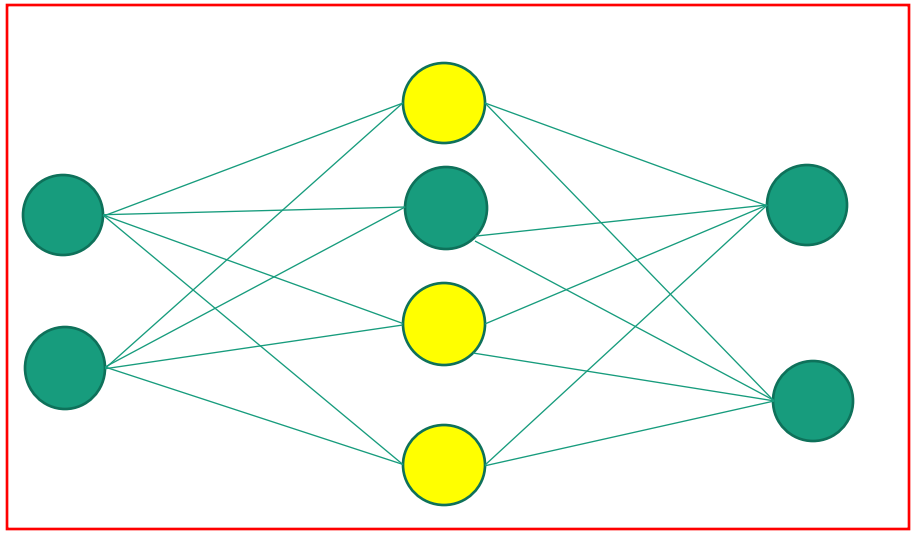
Parameter pruning - criteria

Challenge 2: where to prune?

- ❑ Local pruning: removing a fixed percentage of the neurons from each layer
- ❑ Global pruning: removing a fixed percentage of all neurons globally



Local pruning



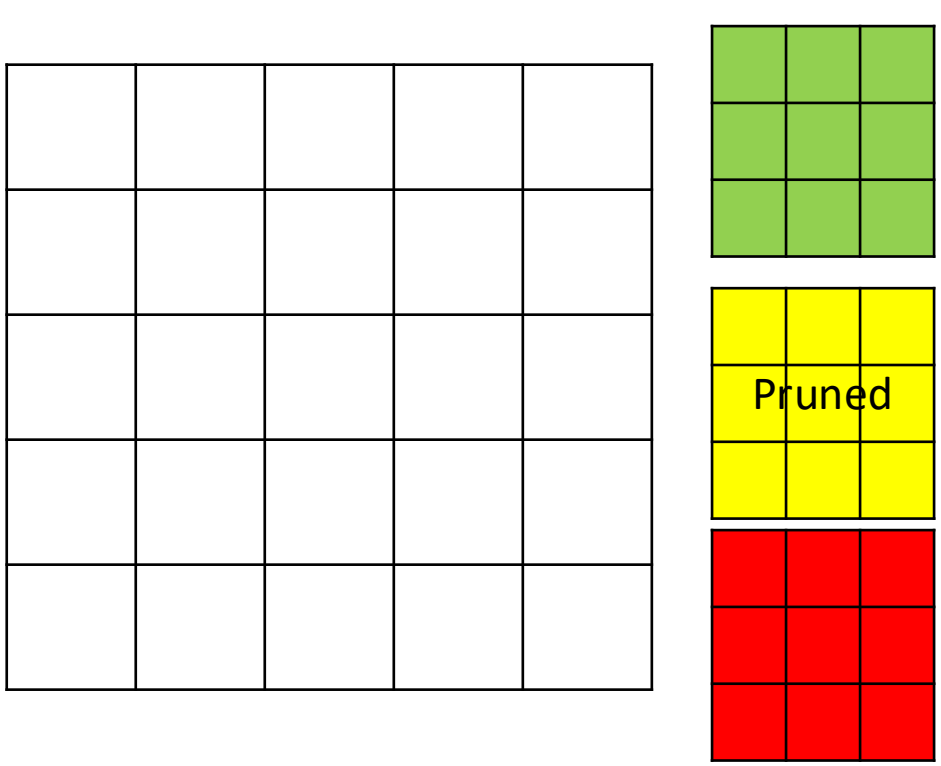
Global pruning

Pruning structures

1. Unstructured pruning: removing the less important neurons (the less salient connection) in the DNN wherever they are. It dose not consider the relationships between the selected neurons

2. Structured pruning: pruning weights in groups such as a layer, a channel or a layer i.e., selective pruning of a large part (a layer of a channel) of the DNN

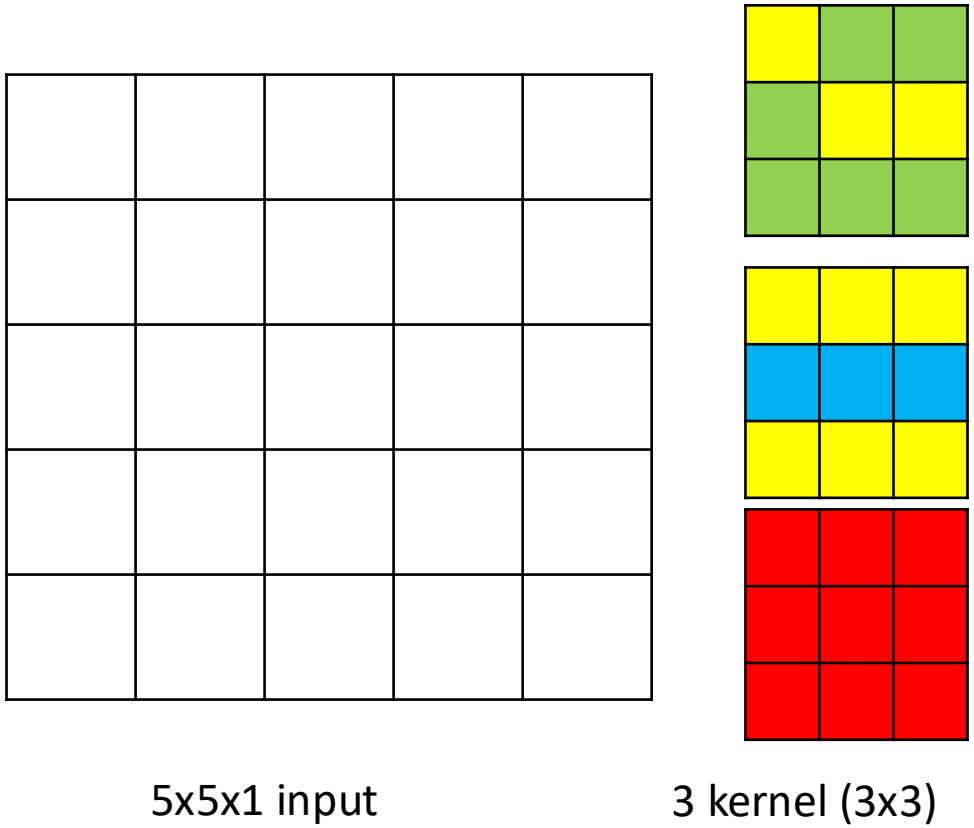
Pruning structures



5x5x1 input

3 kernel (3x3)

Structured



5x5x1 input

3 kernel (3x3)

Unstructured

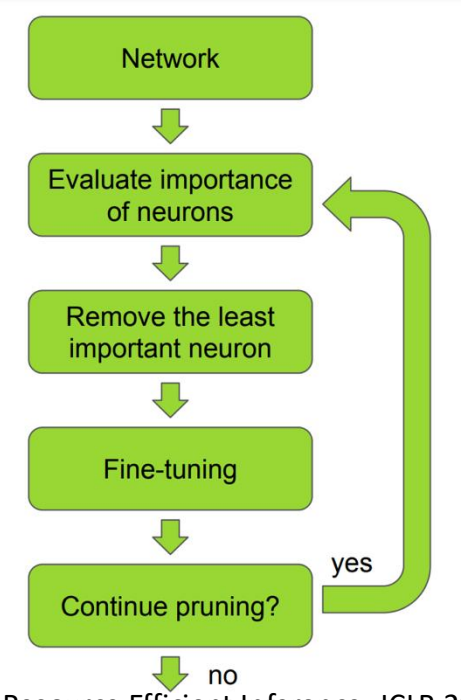
Pruning method

- One-shot pruning:
  - Initialize the network weights
  - Train the network
  - Prune
  - Fine-tune

Pruning method

Iterative parameters pruning

- Based on iteratively removing the least important parameters i.e. features map that have less weight
- Iterative parameters pruning method consist of three stage:
  - Fine-tune the network until convergence on the target task
  - Alternate iterations of pruning and further fine-tuning
  - Stop pruning after reaching the target trade-off between accuracy and pruning objective



Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, Jan Kautz: Pruning Convolutional Neural Networks for Resource Efficient Inference. ICLR 2017

Model quantization

- Model quantization is one of the model compression techniques that aim at reducing the required memory footprint of the deep learning model by reducing the number of bits required to represent each weight
- Run neural networks in lower precision e.g., uint8 or 4bit to reduce the network computational cost
  - Training free quantization
  - Quantization-Aware training / fine-tunning

Model quantization

- Quantization maps a real value (full precision) [B , a] to lie within the range value of low precision b-bit
- FP32 is full precision model datatype i.e., each parameter requires 4 byte
- b is the bit width of a low precision format

Model quantization - Low bit value range

- Given b-bit representation, an 2^b possible integer values can be represented using b-bit format
- The value range of a b-bit signed integer precision format is between

  $[-2^{b-1}, 2^{b-1}-1]$

- Example- 4bit: 16 value can be represented using 2^4 and the value range of 4 bit is between [-8, 7]

Model quantization - Quantization operator

- Quantization operator consists of two processes: value transformation and clipping process
  - Transformation process that maps x into x_q

$$T(x, s, z) = round(\frac{x}{s} - z)$$

    - z (zero-point): is s an integer zero-point which represents x corresponding to the real zero value
    - s is a real-valued scaling factor that divides the real value range into a number of fractions

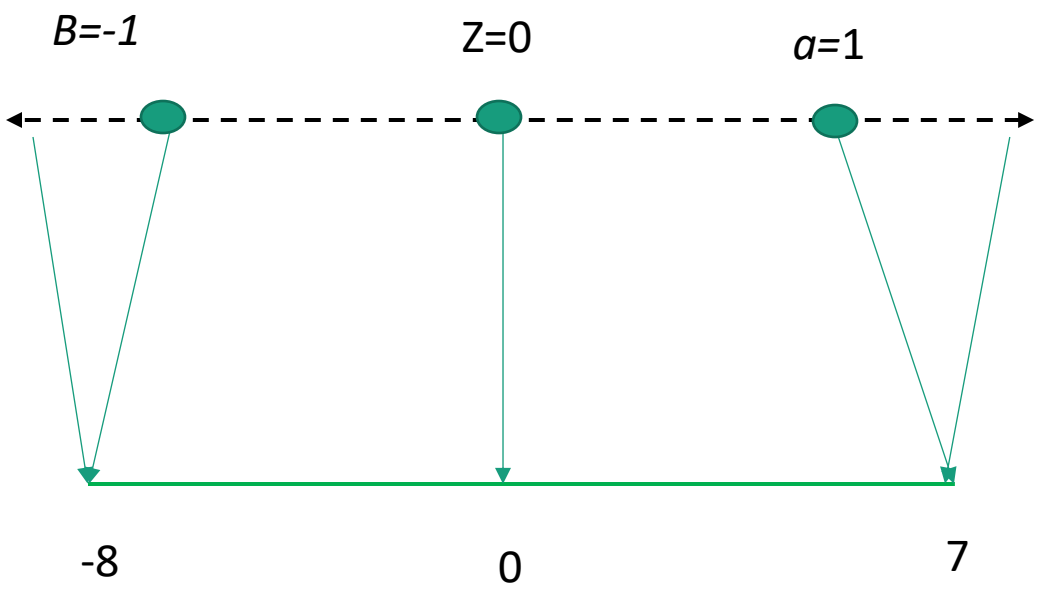
Model quantization - Quantization operator

The scaling factor s and zero-point are defined as follows:

$$s = \frac{\alpha - \beta}{2^b - 1}.$$

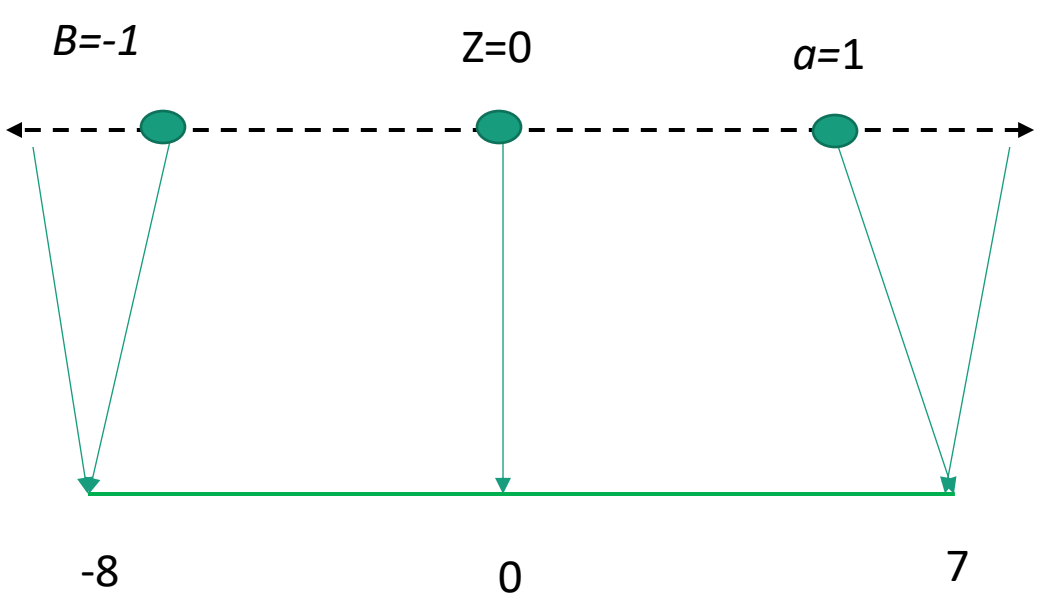
$$z = round(\beta . \frac{2^b - 1}{\alpha - \beta} + 2^{b-1}).$$

Model quantization - Symmetric quantization and asymmetric quantization

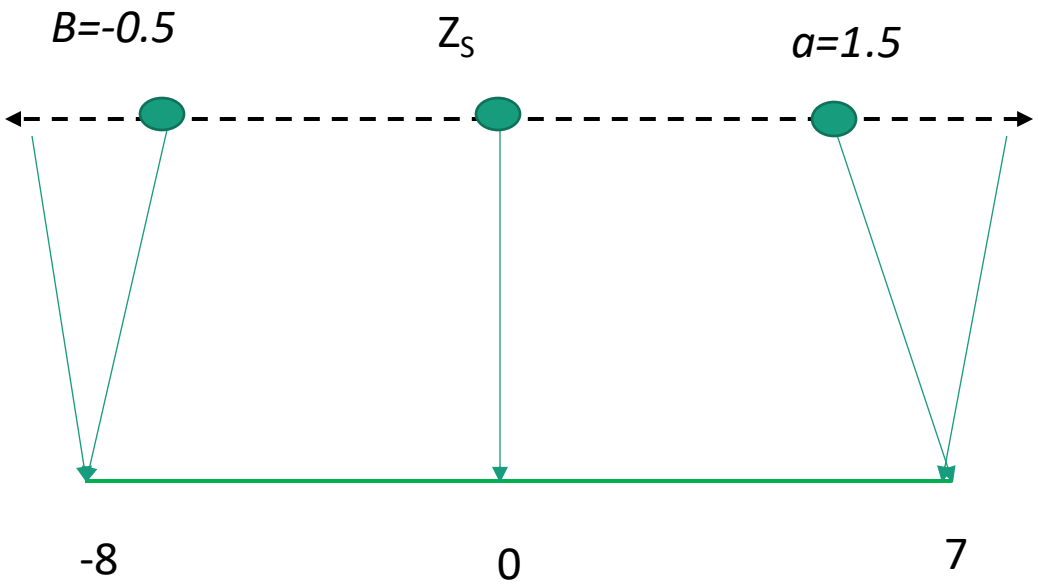


Symmetric quantization: *-a=B and z=0*
Usually *a* and *B* are min(r) and max(r)

Model quantization - Symmetric quantization and asymmetric quantization

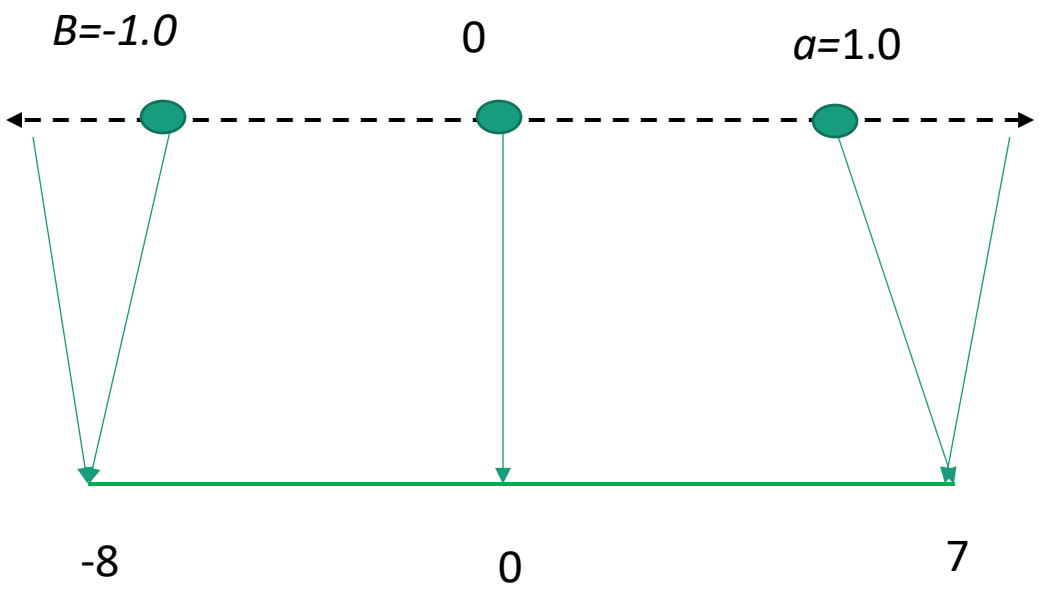


Symmetric quantization:
-a=B and z=0



Asymmetric quantization:
-a !=B and z=Z_S

Model quantization - Symmetric quantization: Exercise



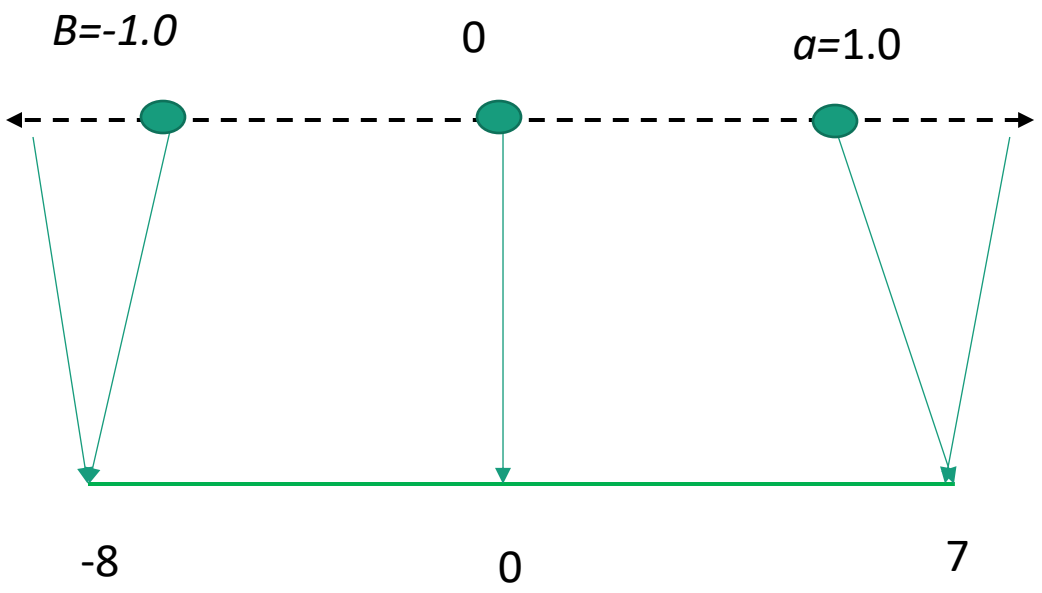
Given 4-bit representation,
a=1.0 and *B*=-1.0
What is the quantized values of
[-1.0,-0.5,0.0,0.5,1.0] ?

$$T(x, s, z) = round(\frac{x}{s} - z)$$

$$s = \frac{\alpha - \beta}{2^b - 1}.$$

Symmetric quantization: -*a*=*B*

Model quantization - Symmetric quantization: Exercise



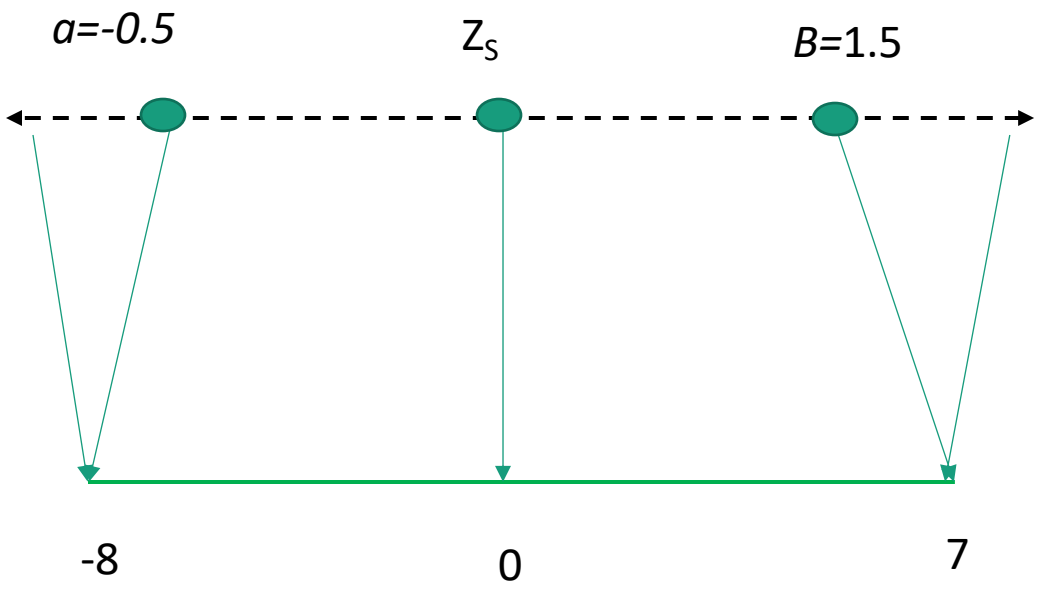
Given 4-bit representation,
a=1.0 and *B*=-1.0
What is the quantized values of
[-1.0,-0.5,0.0,0.5,1.0] ?
[-8, -4, 0, 4, 8]

$$T(x, s, z) = round(\frac{x}{s} - z)$$

$$s = \frac{\alpha - \beta}{2^b - 1}.$$

Symmetric quantization: -*a=B*

Model quantization - Asymmetric quantization: Exercise



Given 4-bit representation, a=-0.5
B=1.5
What is the quantized values of [-0.5,0.0,0.5,1.0] ?

$$T(x, s, z) = round(\frac{x}{s} - z)$$

$$s = \frac{\alpha - \beta}{2^b - 1}.$$

$$z = round(\beta . \frac{2^b - 1}{\alpha - \beta} + 2^{b-1}).$$

Asymmetric quantization: *-a != B*

Model quantization - Clipping

- Clipping operation: clip value x_t to lie within *l* and *u* i.e., min-max value of quantization value range

$$clip(x_t, l, u) = \begin{cases} l & \text{if } x_t < l \\ x & \text{if } l \leq x_t \leq u \\ u & \text{if } x > u, \end{cases}$$

- The quantization of real value x to a b-bit precision format is given by

$$Q(x, b) = clip(T(x, s, z), -2^{b-1}, 2^{b-1} - 1)$$

Model quantization

Discussion:

- Does the model quantization effect the model accuracy?

Yes, however, the gain in the model size and inference speed are extremely high

Example: Quantization aware training of MobileNetV2 trained on ImageNet

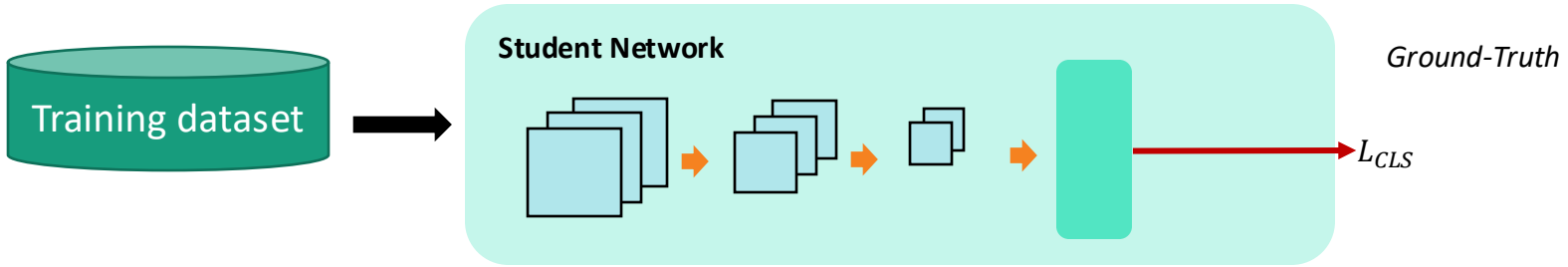
fp32 accuracy %	Int8 accuracy %	accuracy change	Required memory reduction
71.9	65.6	-6.3	4x

Knowledge Distillation

"KD: Transfer knowledge from teacher models to smaller student models that is more suitable for deployment"

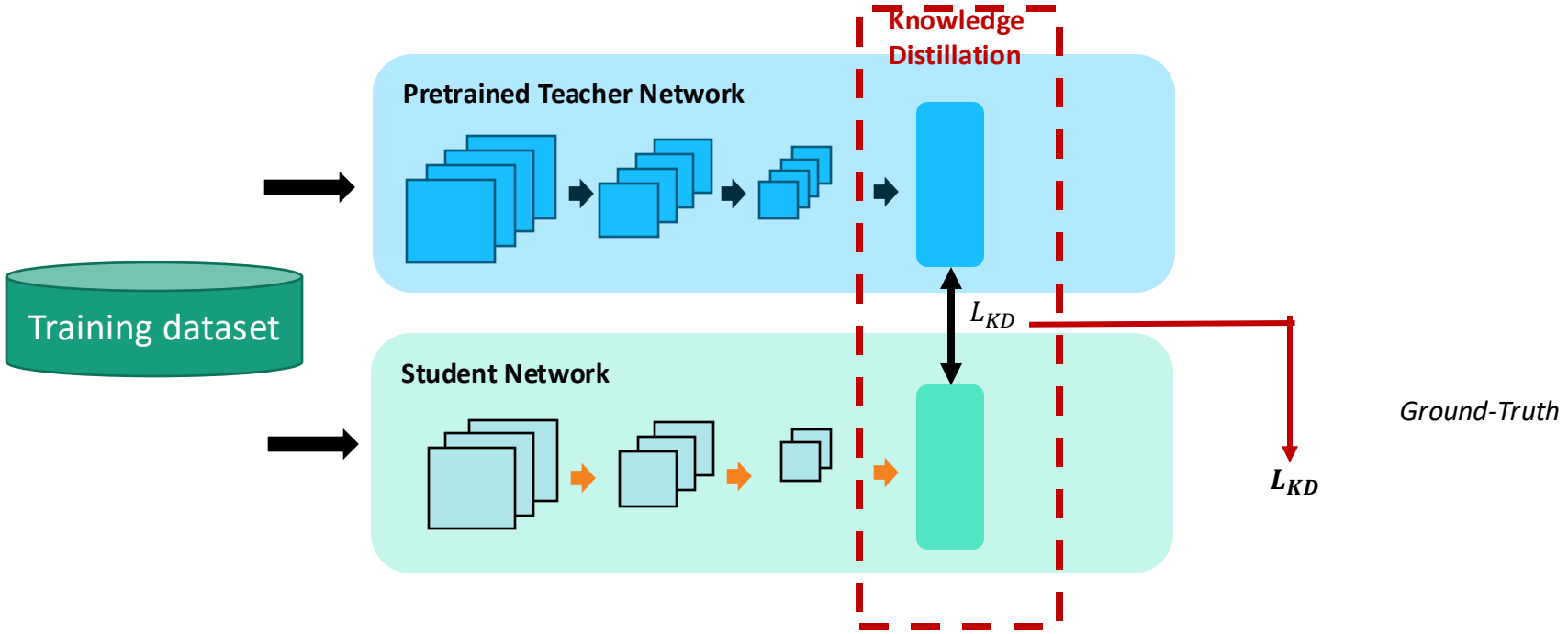
Knowledge Distillation

"KD: Transfer knowledge from teacher models to smaller student models that is more suitable for deployment"



Knowledge Distillation

"KD: Transfer knowledge from teacher models to smaller student models that is more suitable for deployment"



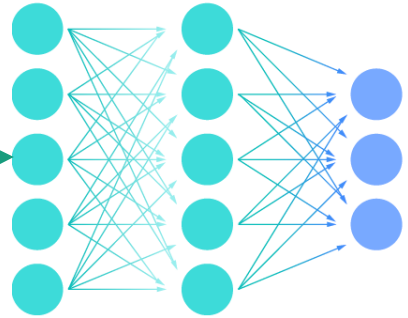
Types of Knowledge Distillation

- Response-based distillation: Using output e.g., soft targets, from the teacher model
- Feature-based distillation: Matching intermediate feature representations
- Relation-based distillation: Preserving relationships between data points or layers

Response-based distillation

Logit-based distillation

Student



1	0	0

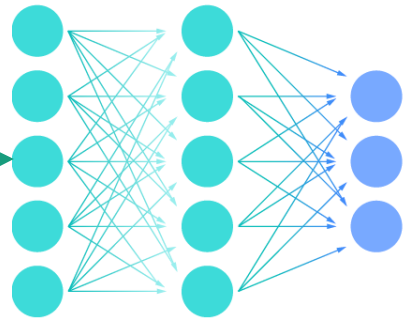
Ground truth

Label	Logit	probability
Cat	8	
Lion	1	
Horse	-2	

Response-based distillation

Logit-based distillation

Student



1	0	0

Ground truth

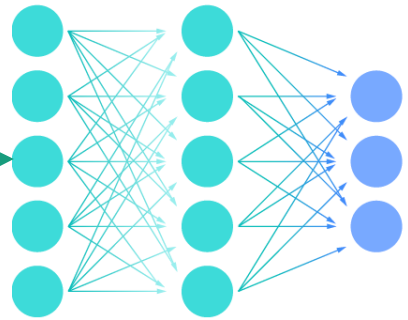
Label	Logit	probability
Cat	8	0.88
Lion	1	0.10
Horse	-2	0.02

$$p_s(i) = \frac{\exp(z_s^i)}{\sum_{j=1}^{C} \exp(z_s^j)}$$

Response-based distillation

Logit-based distillation

Student



1	0	0

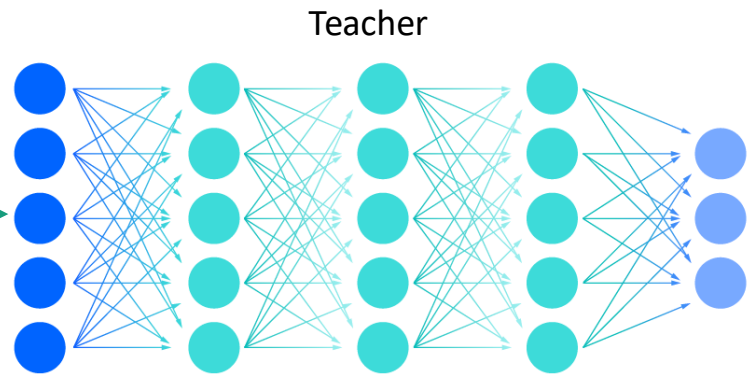
Ground truth

Label	Logit	probability
Cat	8	0.88
Lion	1	0.10
Horse	-2	0.02

$$CE = -log(\frac{\exp(z_s^i)}{\sum_{j=1}^{C}\exp(z_s^j)})$$

Response-based distillation

Logit-based distillation



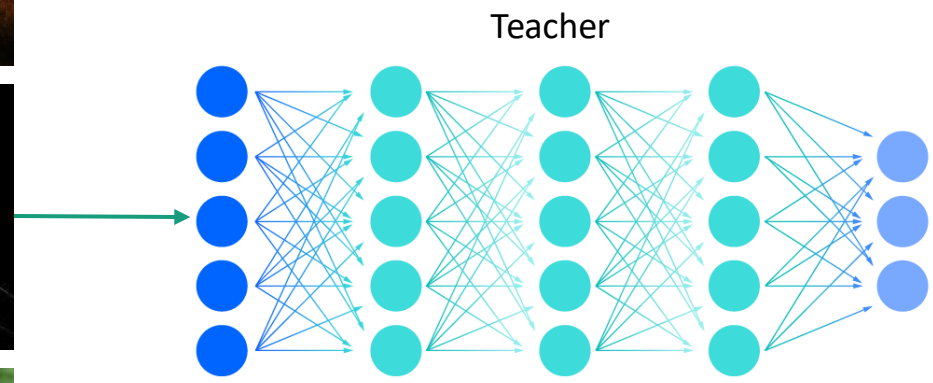
Label	Logit	probability
Cat	10	
Lion	2	
Horse	-1	

1	0	0

Ground truth

Response-based distillation

Logit-based distillation



$$p_t(i) = \frac{\exp(z_t^i)}{\sum_{j=1}^{C} \exp(z_t^j)}$$

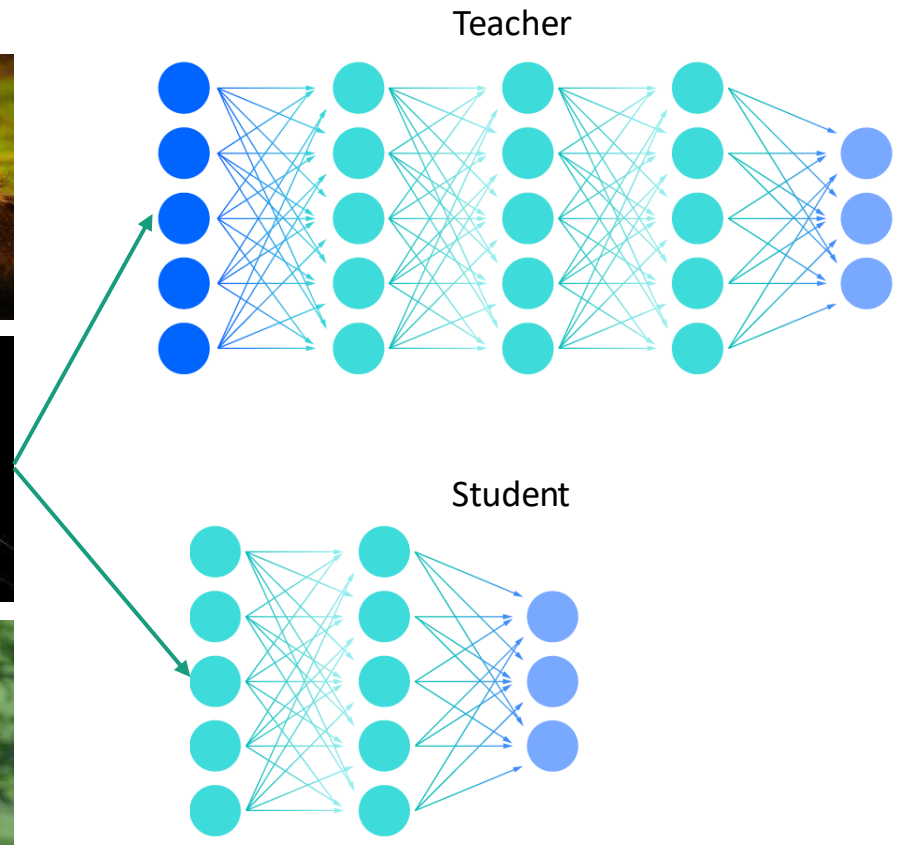
Label	Logit	probability
Cat	10	0.95
Lion	2	0.04
Horse	-1	0.01

1	0	0

Ground truth

Response-based distillation

Logit-based distillation



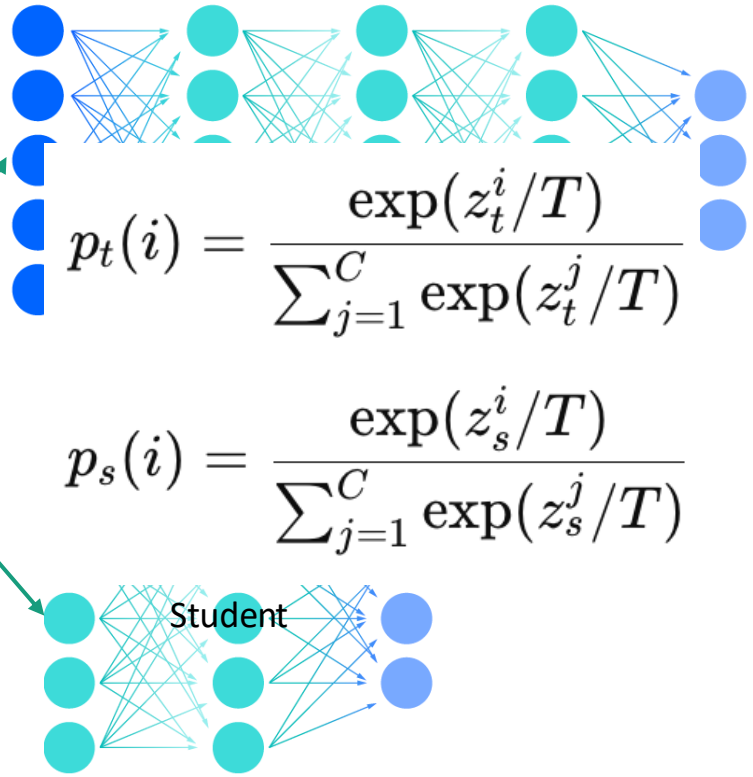
Label	Logit	probability
Cat	10	0.95
Lion	2	0.04
Horse	-1	0.01

Label	Logit	probability
Cat	8	0.88
Lion	1	0.10
Horse	-2	0.02

Response-based distillation

Logit-based distillation

❑ Softened probability with Temperature



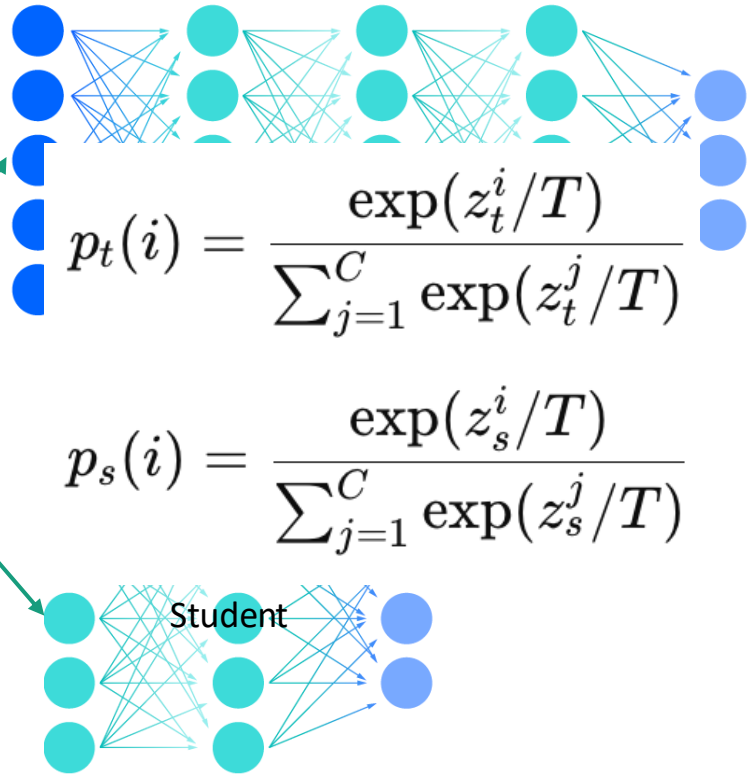
Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Response-based distillation

Logit-based distillation

- Soft labels reveal that the second class is **somewhat related** to the first class, which wouldn't be obvious from hard labels alone



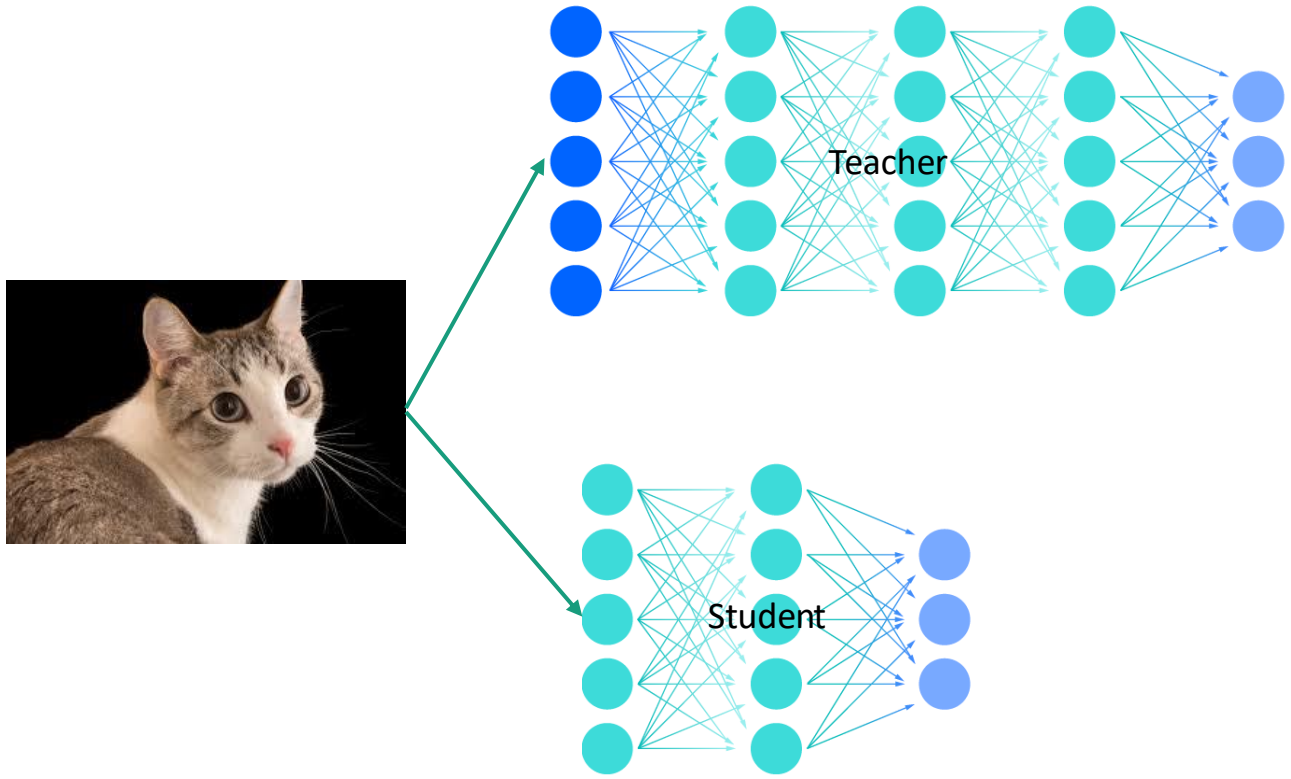
Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Response-based distillation

Logit-based distillation

- Soft targets from the teacher model contain richer information than hard labels → **Help the student to generalize better** and **learn inter-class relationships**



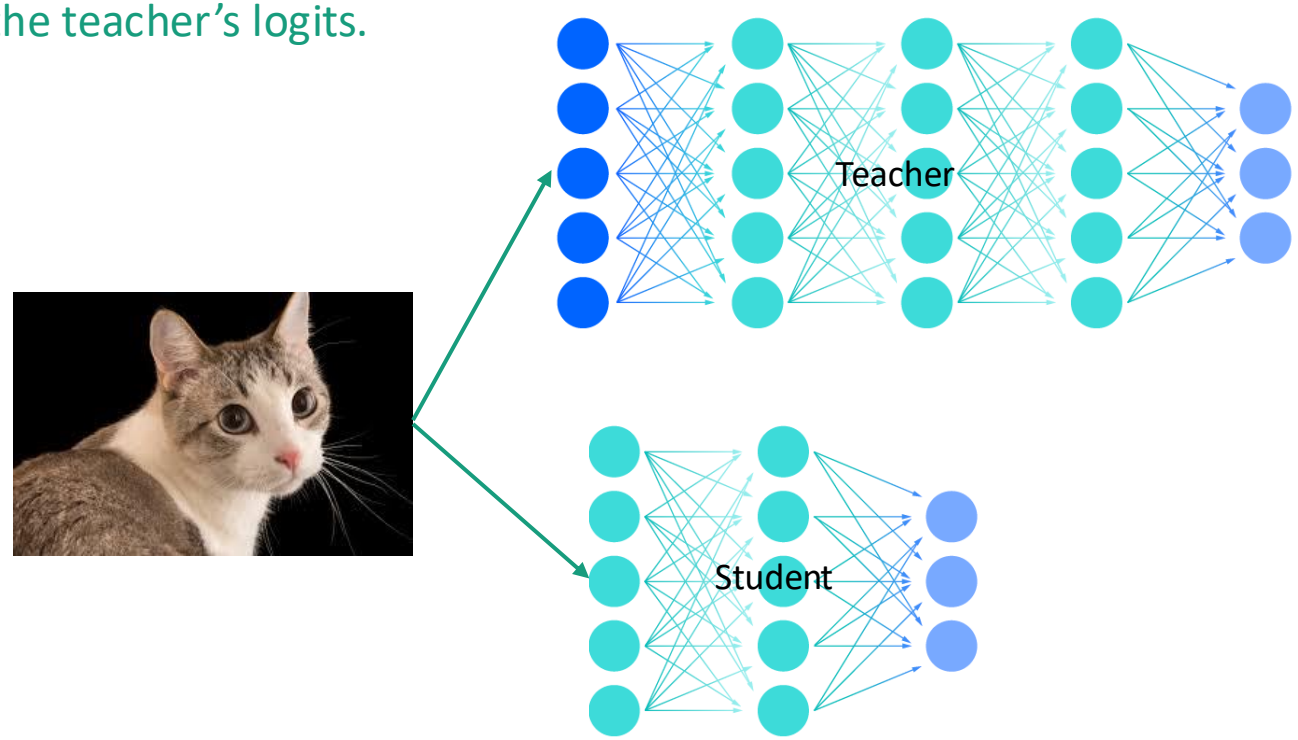
Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Response-based distillation

Logit-based distillation

Logit-based distillation, first introduced by Hinton et al. (2015), transfers knowledge from a **teacher model** to a **student model** using **soft targets**. Instead of using hard labels, the student learns from the softened probability distribution of the teacher's logits.



Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

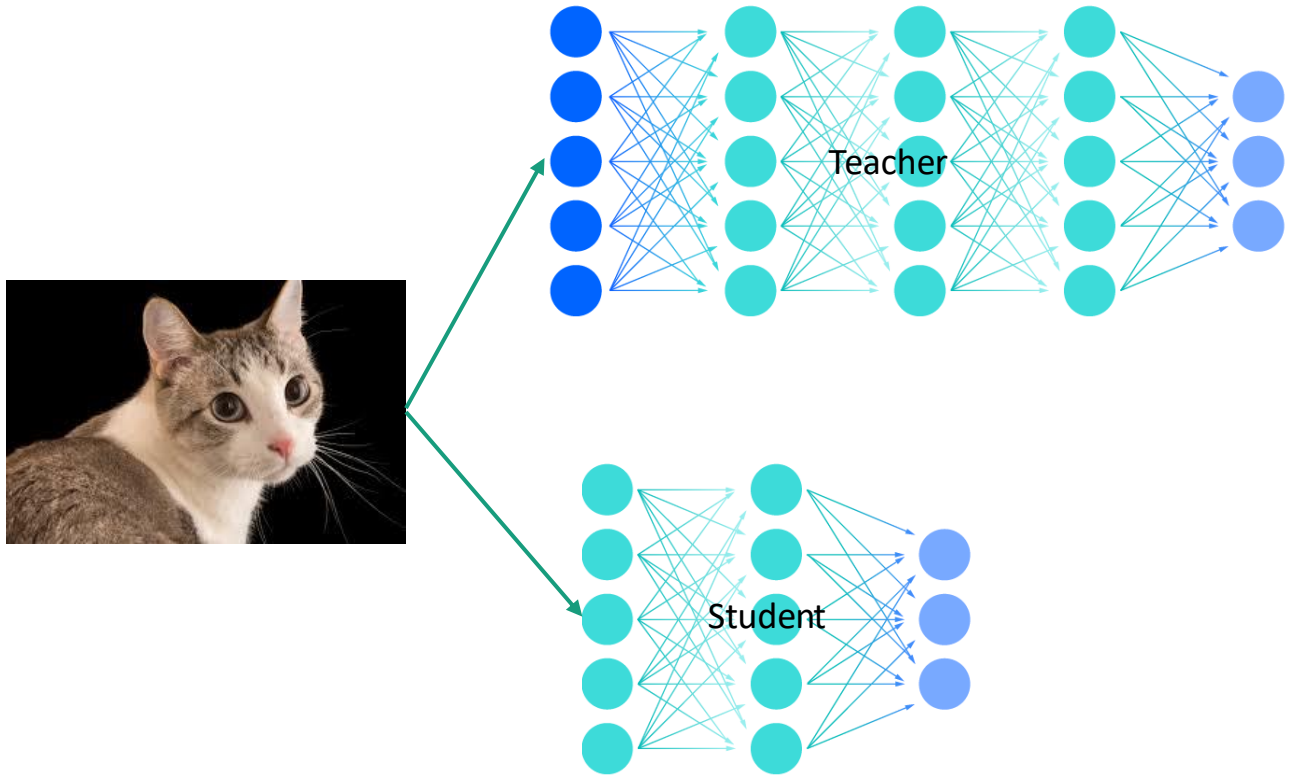
Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Geoffrey E. Hinton, Oriol Vinyals, Jeffrey Dean: **Distilling the Knowledge in a Neural Network.** CoRR abs/1503.02531 (2015)

Response-based distillation

Logit-based distillation

- How? Train the student model to learn to minimize its output probabilities distribution and the ones from the teacher model

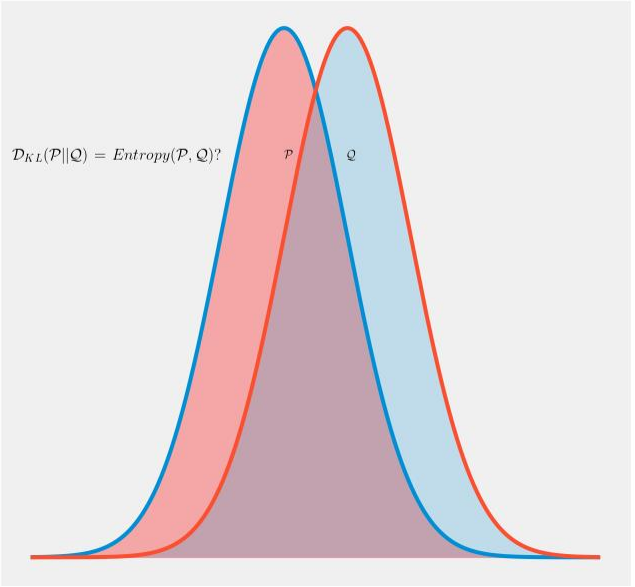


Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Kullback–Leibler divergence (KL divergence)

- KL divergence measures the **difference** between **two probability distributions**, typically used to compare a learned (approximate) distribution against a true (reference) distribution, i.e, quantify the difference between two probability distributions, often used to **assess how much information is lost when approximating one distribution with another**



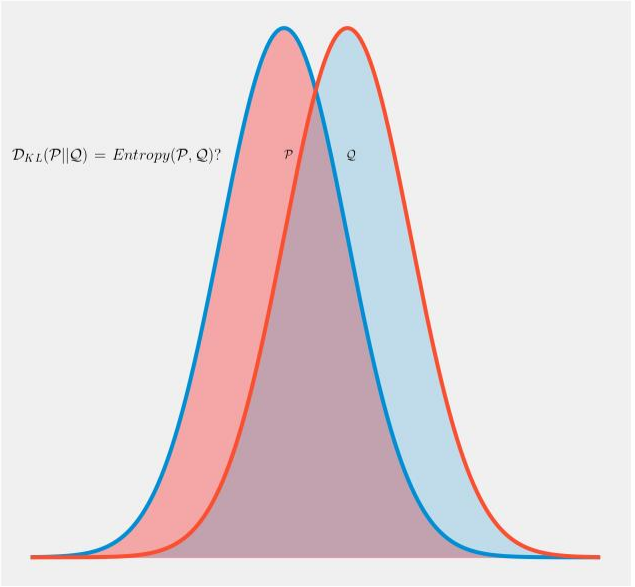
Kullback–Leibler divergence (KL divergence)

- KL divergence measures the **difference** between **two probability distributions**, typically used to compare a learned (approximate) distribution against a true (reference) distribution, i.e, quantify the difference between two probability distributions, often used to **assess how much information is lost when approximating one distribution with another**

- Given two probability distributions:
  - P: The true (or target) distribution
  - Q: The approximate (or learned) distribution

- The Kullback–Leibler divergence from P to Q is defined as:

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$



Kullback–Leibler divergence (KL divergence)

- The Kullback–Leibler divergence from P to Q is defined as:

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

- Intuition:
  - KL divergence quantifies how much information is lost when approximating P with Q
  - If P and Q are identical,

$$D_{\mathrm{KL}}(P||Q) = 0$$

  - KL divergence is asymmetric

$$D_{\mathrm{KL}}(P||Q) \neq D_{\mathrm{KL}}(Q||P)$$

Kullback–Leibler divergence (KL divergence)

- Example:
- P = (0.64, 0.27, 0.09) Q = (0.57, 0.30, 0.13)

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

Kullback–Leibler divergence (KL divergence)

- Example:
- P = (0.64, 0.27, 0.09) Q = (0.57, 0.30, 0.13)

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Label	Logit	probability	Softened
Cat	10	0.95	0.64
Lion	2	0.04	0.27
Horse	-1	0.01	0.09

Label	Logit	probability	Softened
Cat	8	0.88	0.57
Lion	1	0.10	0.30
Horse	-2	0.02	0.13

$$D_{\mathrm{KL}}(P||Q) = 0.64 \log \frac{0.64}{0.57} + 0.27 \log \frac{0.27}{0.30} + 0.09 \log \frac{0.09}{0.13}$$

Kullback–Leibler divergence (KL divergence)

Relation between Cross-Entropy and KL divergence

If we consider the class probabilities P as soft label (target), then the KL Loss can be considered as **a softmax cross entropy** between teacher (target) and student (prediction) class probabilities:

Kullback–Leibler divergence (KL divergence)

Relation between Cross-Entropy and KL divergence

Proof:

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log P(i) - \sum_i P(i) \log Q(i)$$

Cross-Entropy: Cross-entropy between P and Q is defined as:

$$H(P,Q) = -\sum_i P(i) \log Q(i)$$

Entropy: Entropy of P (the uncertainty in P) is: $H(P) = -\sum_i P(i) \log P(i)$

$$D_{\mathrm{KL}}(P||Q) = H(P,Q) - H(P)$$

Since H(P) is a constant with respect to Q, minimizing the cross-entropy H(P,Q) is equivalent to minimizing the KL divergence $D_{KL}(P||Q)$

Kullback–Leibler divergence (KL divergence)

Relation between Cross-Entropy and KL divergence

Intuition:

Minimizing **cross-entropy** forces the model's predicted distribution Q to be as close as possible to the true distribution P. Since **KL divergence** measures the discrepancy between these distributions, minimizing cross-entropy naturally reduces KL divergence.

•If P is **one-hot** (i.e., P=(0,1,0,…)), minimizing cross-entropy is equivalent to minimizing negative log-likelihood.

•If P is a soft distribution (e.g., in **knowledge distillation**, where a teacher model provides soft labels), minimizing cross-entropy ensures Q aligns with P

Response-based distillation
Logit-based distillation

- **Knowledge Distillation Loss:** The student model is trained using a combination of **cross-entropy loss** for ground truth labels and **KL divergence loss** for soft targets:

$$\mathcal{L}_{\mathrm{KD}} = (1-\lambda)\mathcal{L}_{\mathrm{CE}}(y, p_s) + \lambda T^2 \mathcal{L}_{\mathrm{KL}}(p_t, p_s)$$

  - $L_{CE}(y,p_s)$ is the standard cross-entropy loss for ground truth labels.
  - $L_{KL}(p_t,p_s)$ is the **Kullback-Leibler divergence** between teacher and student probabilities.
  - λ balances the contribution of hard labels vs. soft targets.
  - T^2 compensates for gradients being smaller at higher temperatures.

Response-based distillation
Logit-based distillation

❑ Why is T^2 Needed?

- ❑ Since both teacher and student probabilities are computed using a **softmax function** with temperature T, the logits z are divided by T, effectively reducing their magnitude. This makes the **softened probabilities smoother**, spreading probability mass across multiple classes. However, this also **reduces the gradient magnitudes** in backpropagation. To counteract this, we multiply the KL divergence term by $\mathbf{T^2}$

❑ Gradient Scaling Effect

- ❑ The **softened probabilities** provide better class relationships but inherently have smaller gradients.
- ❑ Multiplying by T^2 ensures that gradient updates remain significant when training the student network.
- ❑ This is important because if the gradients are too small, the student model may struggle to effectively learn from the teacher.

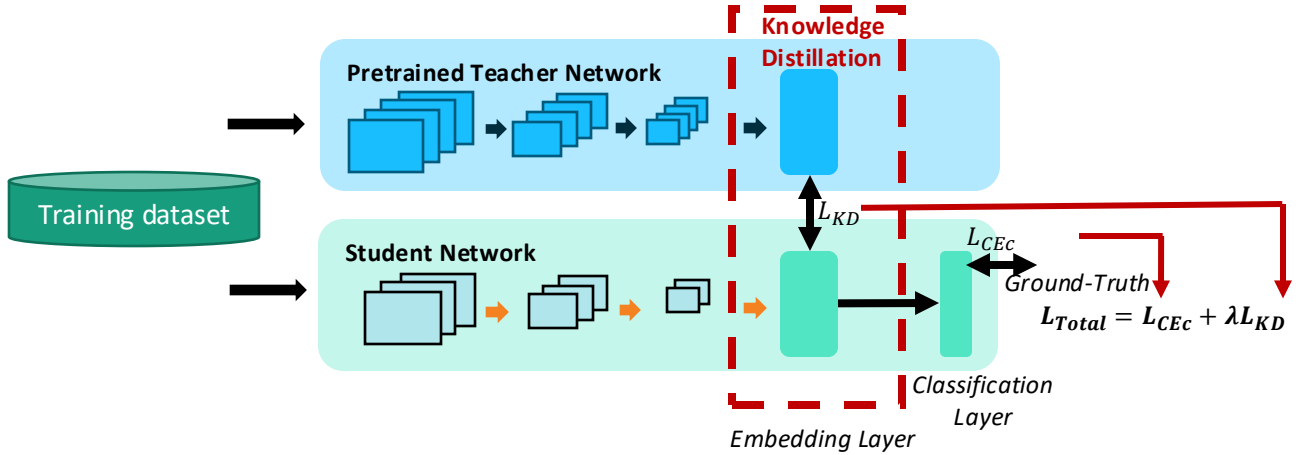
❑ Practical Takeaway

- ❑ **Higher T → Softer probabilities → Smaller gradients**
- ❑ **Multiplying by T^2 restores the gradient magnitude → Ensures effective learning from the teacher model**

Feature-Based Knowledge Distillation (FKD)

- Feature-Based Knowledge Distillation (FKD) focuses on transferring **intermediate feature representations** from a **teacher model** to a **student model** rather than using soft logits (as in logit-based distillation). This method ensures that the student network mimics how the teacher extracts and processes important features.

- FKD ensures that the feature representations extracted from the teacher and student networks are **aligned** in terms of both **content and structure**



Feature-Based Knowledge Distillation (FKD)

1. Pass batch of images through Teacher and Student models:

- Given an input image x, we extract feature representations from a specific layer in the teacher and student

$$F_T = \text{TeacherNetwork}(x)$$

$$F_S = \text{StudentNetwork}(x)$$

2. Align feature map dimensions

- Since the teacher model is usually **larger** than the student model, their feature maps might have different dimensions:

$$F_T \in \mathbb{R}^{C_T \times H_T \times W_T}$$

$$F_S \in \mathbb{R}^{C_S \times H_S \times W_S}$$

To match the dimensions, we apply a **learnable transformation function** g(·), which can be:

- A **1×1 convolutional layer**
- A **linear transformation (fully connected layer)**
- A **bilinear interpolation (upsampling/downsampling)**

Feature-Based Knowledge Distillation (FKD)

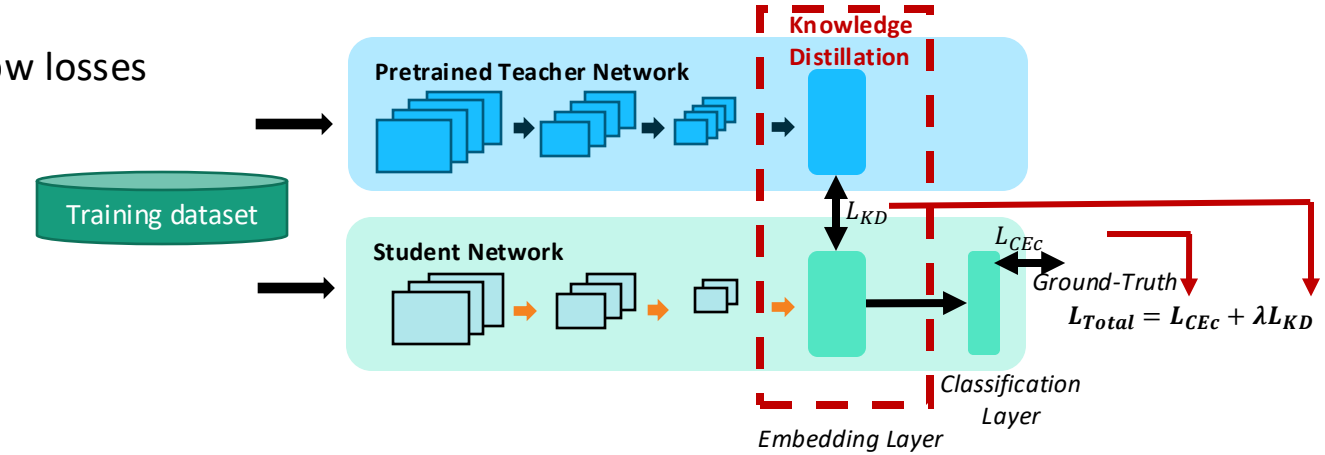
3. Compute feature-Based distillation loss

$$\mathcal{L}_{\text{Feature}} = \|F_T - F'_S\|^2$$

4. Train Student Model with Combined Loss

$$\mathcal{L}_{\text{Total}} = \lambda \mathcal{L}_{\text{Feature}} + (1 - \lambda)\mathcal{L}_{\text{CE}}$$

- λ is a weighting factor that balances the importance tow losses



Relation-Based Knowledge Distillation (RKD)

- **Recap:**
  - **Logit-based distillation** transfers **final class probabilities**
  - **Feature-based distillation** transfers **intermediate feature maps**
- **Relation-based knowledge distillation (RKD)** learns the **structural relationships** between different **data samples** or **layers**.
- Example: Face recognition
  - RKD ensures that the **relative** **distances and angles** between sample embeddings in the **student model** match those in the **teacher model**. This allows the student model to learn a **relational structure** rather than just mimicking feature values.

Relation-Based Knowledge Distillation (RKD)

❑ How Does RKD Work?

Let's assume we have a **batch** of N samples where the teacher model outputs feature embeddings:

$$F_T = \{f_T^1, f_T^2, ..., f_T^N\}, \quad f_T^i \in \mathbb{R}^d$$

and the student model produces its own embeddings:

$$F_S = \{f_S^1, f_S^2, ..., f_S^N\}, \quad f_S^i \in \mathbb{R}^d$$

RKD focuses on ensuring that **pairwise relations** (distances and angles) between embeddings in the student model are similar to those in the teacher model.

Relation-Based Knowledge Distillation (RKD)

RKD Loss Functions: **Euclidean distances** between embeddings in the student model match those in the teacher model.

$$\mathcal{L}_{\text{RKD-Dist}} = \sum_{i \neq j} \left(D_S^{(i,j)} - D_T^{(i,j)} \right)^2$$

where:

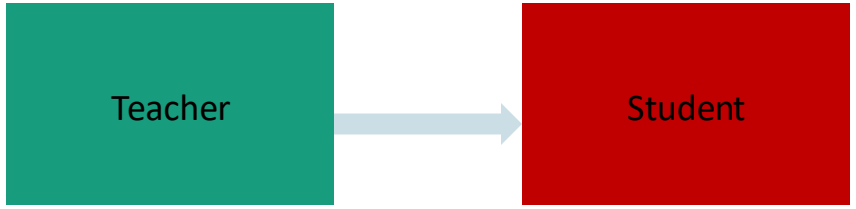
- $D_T^{(i,j)} = \| f_T^i - f_T^j \|_2$ is the distance between two embeddings in the teacher.
- $D_S^{(i,j)} = \| f_S^i - f_S^j \|_2$ is the corresponding distance in the student.

Types of Knowledge Distillation

Method	What is Distilled?	Loss Function	Best for
Logit-Based KD	Soft class probabilities	KL Divergence	Classification
Feature-Based KD	Intermediate feature maps	MSE, L1,	Face Recognition, Object Detection, Visual Representation learning
Relation-Based KD	Pairwise relations (distance & angles)	RKD-Distance	Face Verification, Metric Learning

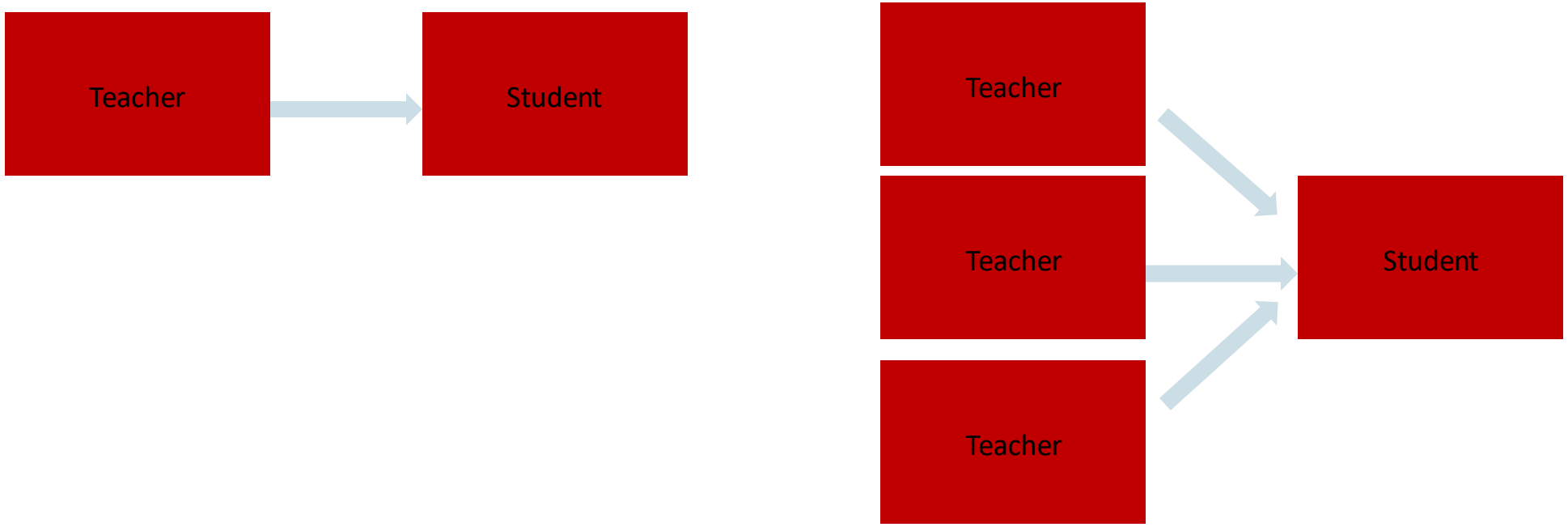
Knowledge Distillation Methods: Online, Offline, and Self-Distillation

1. Offline knowledge distillation: The teacher model is pre-trained and fixed before training the student model.



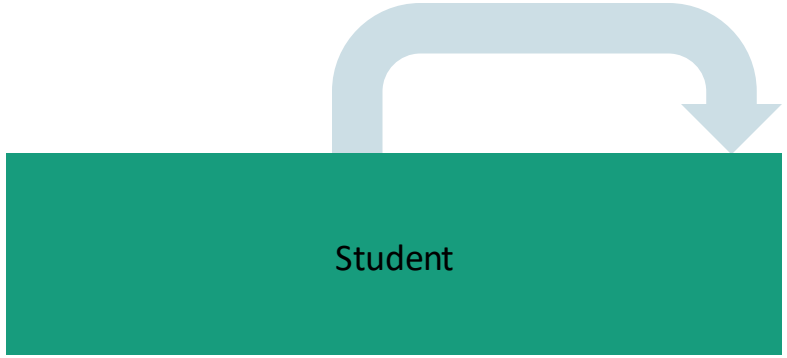
Knowledge Distillation Methods: Online, Offline, and Self-Distillation

1. Online knowledge distillation: The teacher and student are trained **simultaneously**, rather than using a pre-trained teacher.



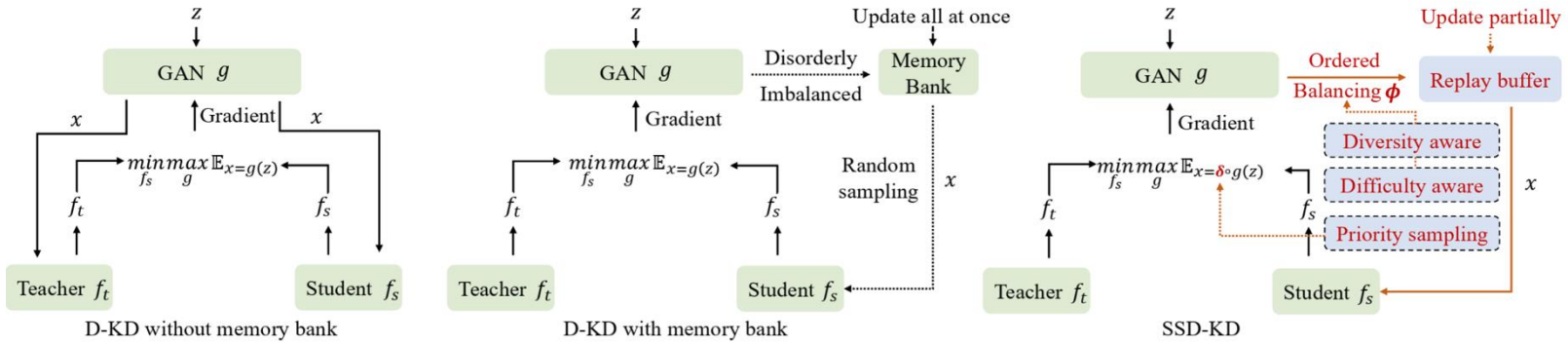
Knowledge Distillation Methods: Online, Offline, and Self-Distillation

1. Self-distillation: A single model acts as **both teacher and student**, learning from its own deeper layers.



Data-Free Knowledge Distillation

Data-Free Knowledge Distillation is a variant of traditional knowledge distillation in which the **original training data is not available** during the student model's training. Instead, the student learns to mimic the teacher using **synthetically generated data**.



Liu, He, et al. "Small scale data-free knowledge distillation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

Data-Free Knowledge Distillation

Instead of using real data xxx to compute soft targets from a teacher model, we **generate synthetic data** $\tilde{x}$ to train the student.

$$\mathcal{L}_{\mathrm{DFKD}} = D_{\mathrm{KL}}(T(\hat{x})||S(\hat{x}))$$

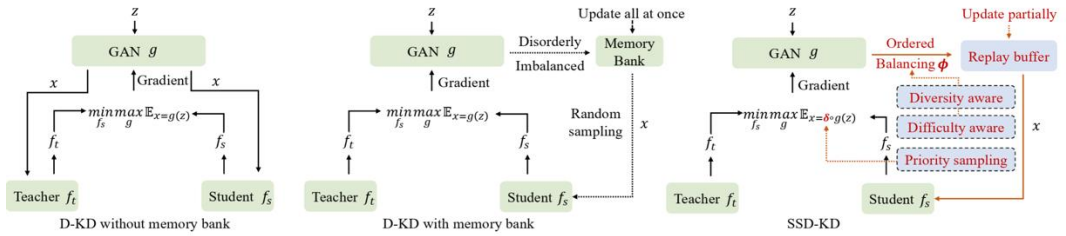
Where:

•T($\tilde{x}$) teacher prediction (soft labels)

•S($\tilde{x}$)student prediction

•$\tilde{x}$:generated data (e.g., from a generator or random noise)

•DKLD: KL divergence between teacher and student outputs



Liu, He, et al. "Small scale data-free knowledge distillation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

Data-Free Knowledge Distillation

How to generate data?

1. Generator Network (GAN-based)

Use a generator G(z), where z∼N(0,I) to produce synthetic samples $\tilde{x}$ =G(z). The generator is trained to **maximize** the response of the teacher (e.g., maximize entropy or diversity of outputs):

$$\max_{G} \mathcal{H}(T(G(z)))$$

Then, use the synthetic $\tilde{x}$ to train the student within KD framework.

Data-Free Knowledge Distillation

2. Deep Inversion

This method inverts the teacher model to generate images that match the internal batch norm statistics of the teacher.
Optimizes an output image $\tilde{x}$ to minimize:

$$\mathcal{L}_{\text{inv}} = \|\mu_{\text{BN}}^{\text{real}} - \mu_{\text{BN}}^{\text{synthetic}}\|^2 + \text{total variation}$$

Then uses these images to distill knowledge.

3. Random Noise + Activation Maximization

Start from random noise and optimize input to **maximize the confidence** or **entropy** of the teacher's predictions.
Encourages diverse synthetic samples.

Data-Free Knowledge Distillation

❑ Pros:

- ❑ No need for real data
- ❑ Maintains performance close to original distillation
- ❑ Useful in privacy-sensitive domains

❑ Cons:

- ❑ Synthetic data quality is crucial
- ❑ May require tuning of generator or inversion method

Challenges in Knowledge Distillation

Teacher-student capacity gap: A student model too small may not capture complex patterns.

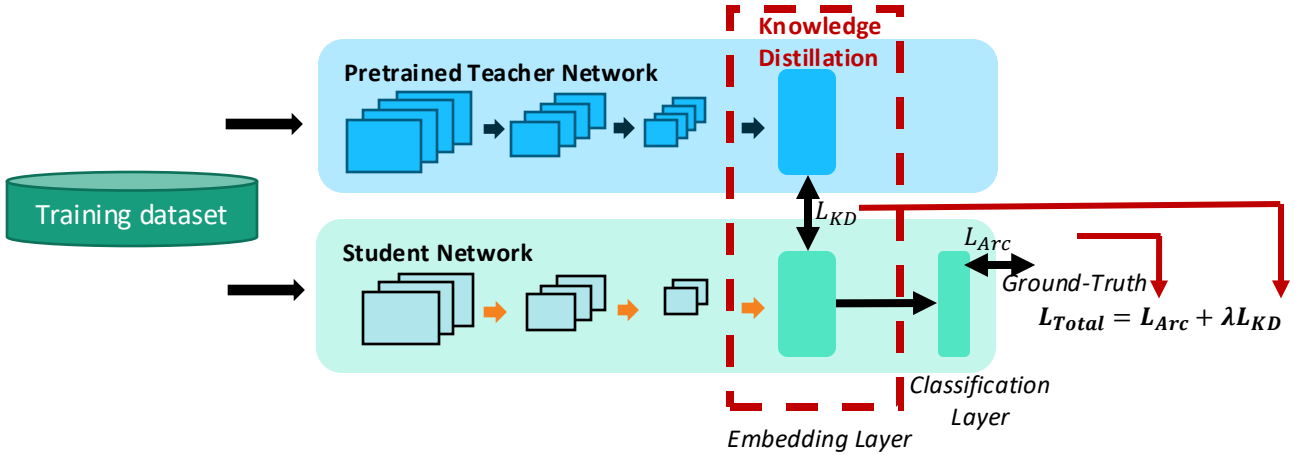
Overfitting on soft labels: Poor generalization if the student model relies too much on teacher predictions.

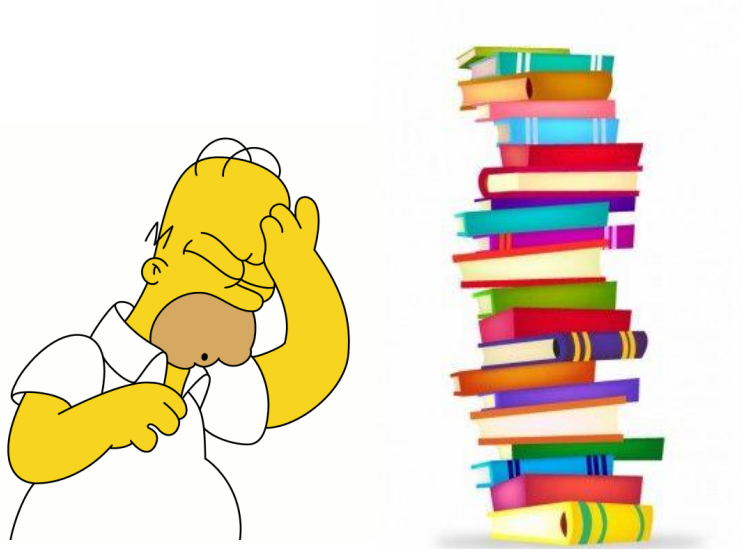
Balancing loss terms: Combining cross-entropy loss with KL divergence requires careful tuning.

Multi-Step knowledge distillation

Capacity Gap

Transform the knowledge from a very deep teacher model to a small student model is difficult when the gap in terms of network size between the teacher and the student model is large





Boutros et al. PocketNet: Extreme Lightweight Face Recognition Network Using Neural Architecture Search and Multistep Knowledge Distillation. IEEE Access 10: 46823-46833 (2022)

Multi-Step knowledge distillation

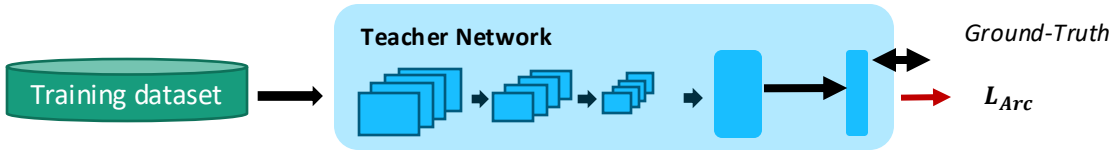
Capacity Gap

In **multi-step KD** the knowledge is distilled from the teacher model to the student model at **different stages of the training maturity**

Multi-Step knowledge distillation

Capacity Gap

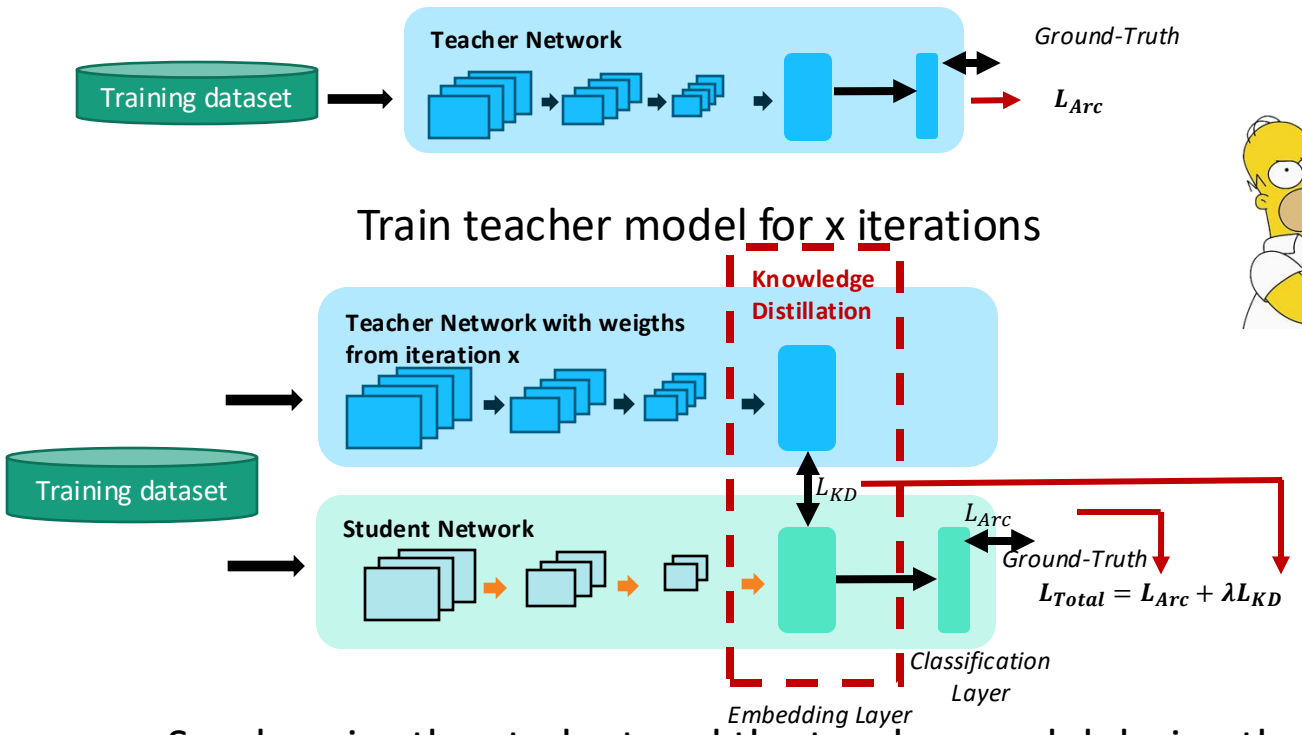
In **multi-step KD** the knowledge is distilled from the teacher model to the student model at **different stages of the training maturity**



Train teacher model for x iterations

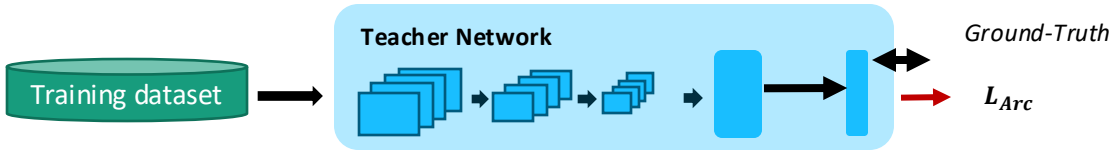
Multi-Step knowledge distillation

In **multi-step KD** the knowledge is distilled from the teacher model to the student model at **different stages of the training maturity**



Multi-Step knowledge distillation

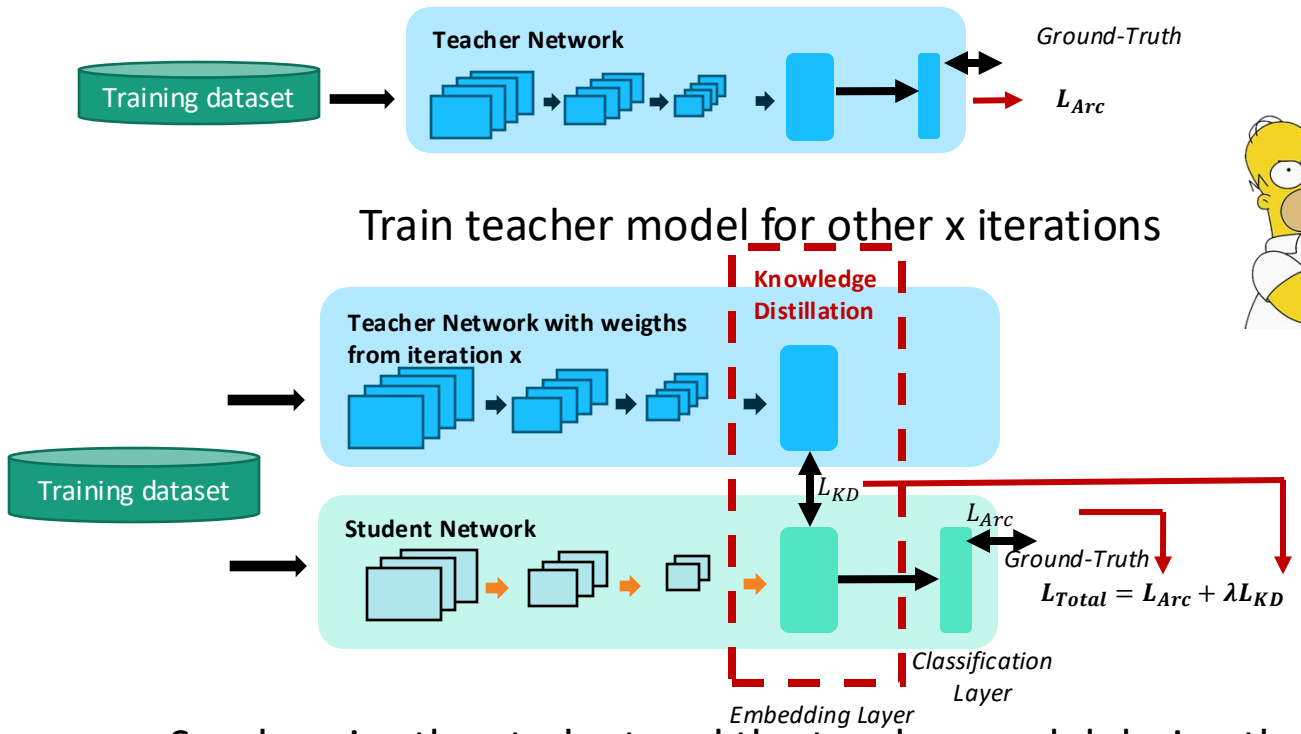
In **multi-step KD** the knowledge is distilled from the teacher model to the student model at **different stages of the training maturity**



Train teacher model for other x iterations

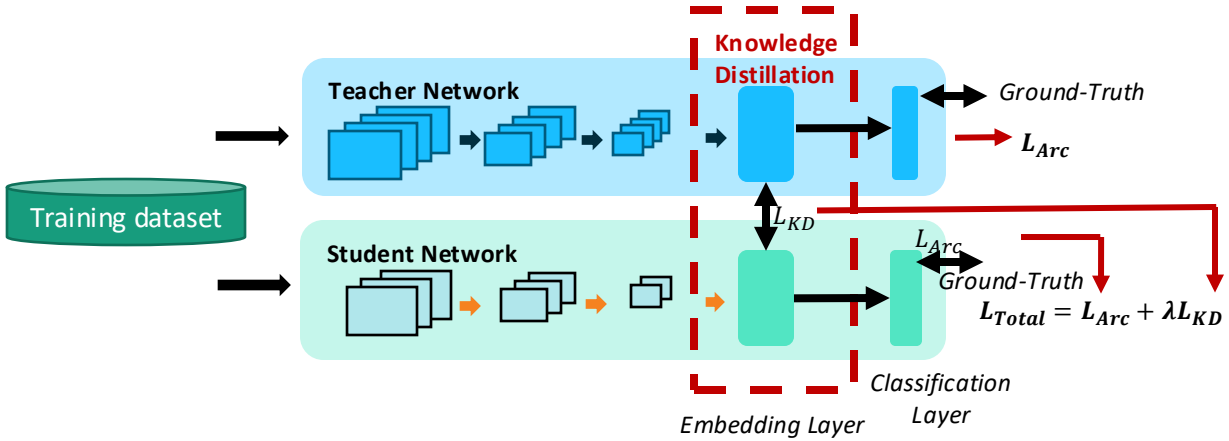
Multi-Step knowledge distillation

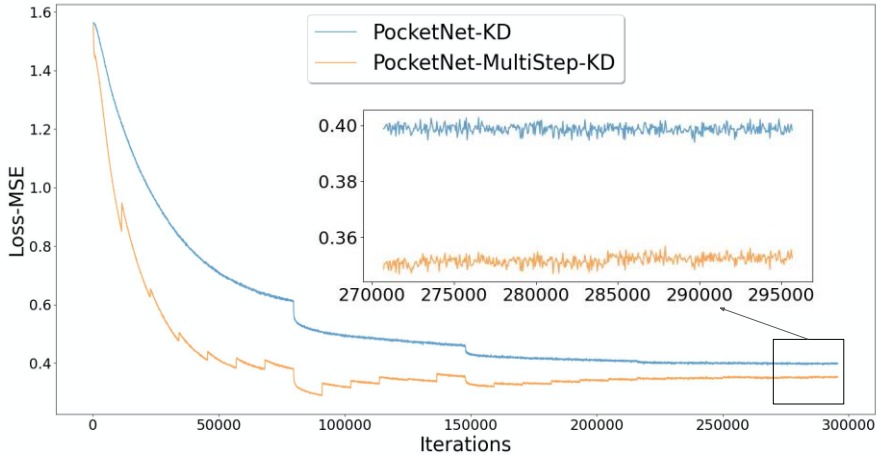
In **multi-step KD** the knowledge is distilled from the teacher model to the student model at **different stages of the training maturity**



Multi-Step knowledge distillation

- **Multi-step KD** guides the student model to **learn** feature representations that are **more similar to the feature produced by teacher**, in comparison to **conventional KD**
- Model trained with multi-step KD outperformed the model trained with conventional KD

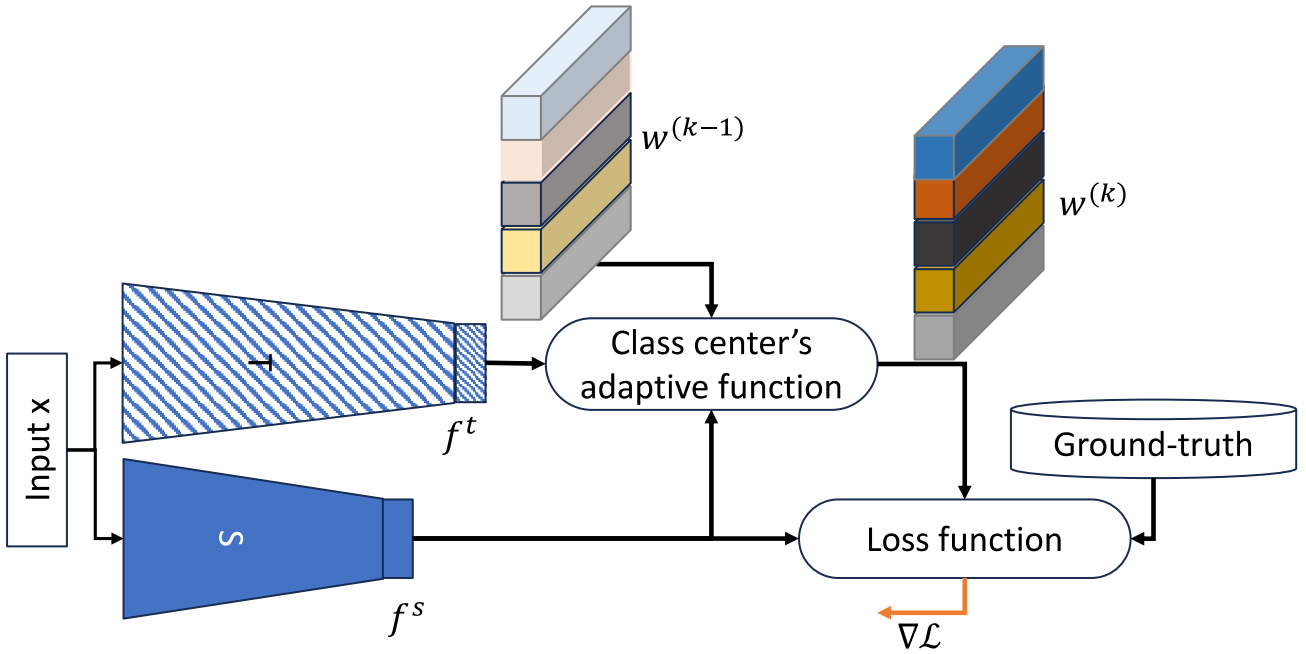




AdaDistill

Adaptive Knowledge distillation

AdaDistill embeds the **KD** concept into the **softmax** loss by training the student using a margin penalty softmax loss with **distilled class centers** from the teacher.



Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation

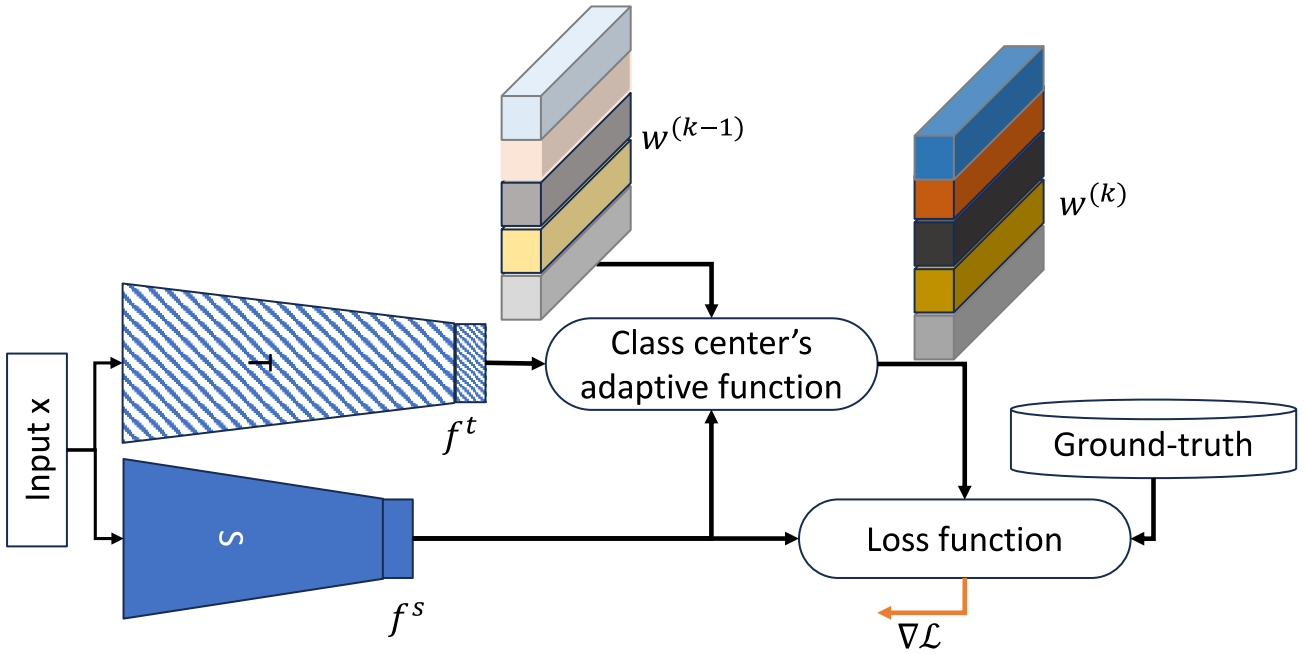
AdaDistill embeds the **KD** concept into the **softmax** loss by training the student using a margin penalty softmax loss with **distilled class centers** from the teacher.

- Feature distillation:

$$\mathcal{L}_{mse} = \frac{1}{N}\sum_{i\in n}\|f_i^s - f_i^t\|_2^2$$

- Multi-class classification loss:

$$\mathcal{L}_{AML} = \frac{1}{N}\sum_{i\in N} -log\frac{e^{s(cos(\theta_{y_i}+m1)-m2)}}{e^{s(cos(\theta_{y_i}+m1)-m2)} + \sum_{j=1,j\neq y_i}^{c} e^{s(cos(\theta_j))}}$$



Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation

AdaDistill embeds the **KD** concept into the **softmax** loss by training the student using a margin penalty softmax loss with **distilled class centers** from the teacher.

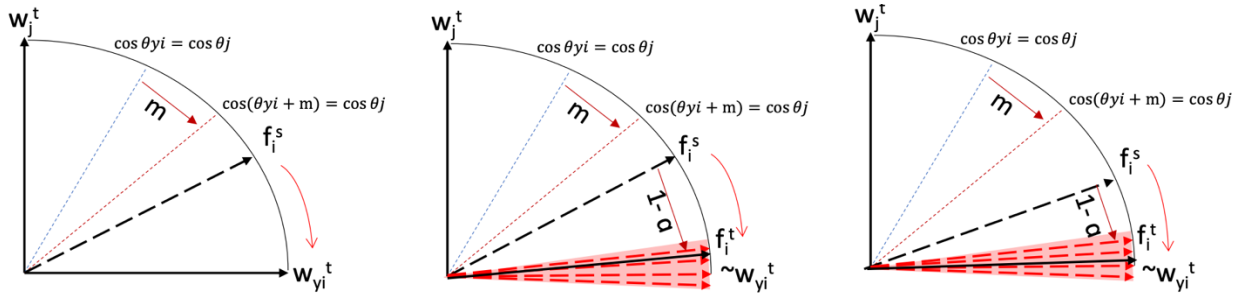
- Feature distillation through classification loss

$$\mathcal{L}_{AMLDistill} = \frac{1}{N}\sum_{i\in N} -log\frac{e^{s(cos(\theta^t_{y_i}+m1)-m2)}}{e^{s(cos(\theta^t_{y_i}+m1)-m2)} + \sum_{j=1,j\neq y_i}^{C} e^{s(cos(\theta^t_j))}}$$

- AdaDistill: Adaptive AMLDistill

$$w^{(k)}_{yi} = \alpha w^{(k-1)}_{yi} + (1-\alpha)(f^{t(k)}_i)^T$$

$$\alpha = \lfloor Cos(f_i, f^t_i) \rceil^1_0$$



(a) ArcDistill **(b)** AdaDistill (Early-stage) **(c)** AdaDistill (Later-stage)

Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation

AdaDistill embeds the **KD** concept into the **softmax** loss by training the student using a margin penalty softmax loss with **distilled class centers** from the teacher.

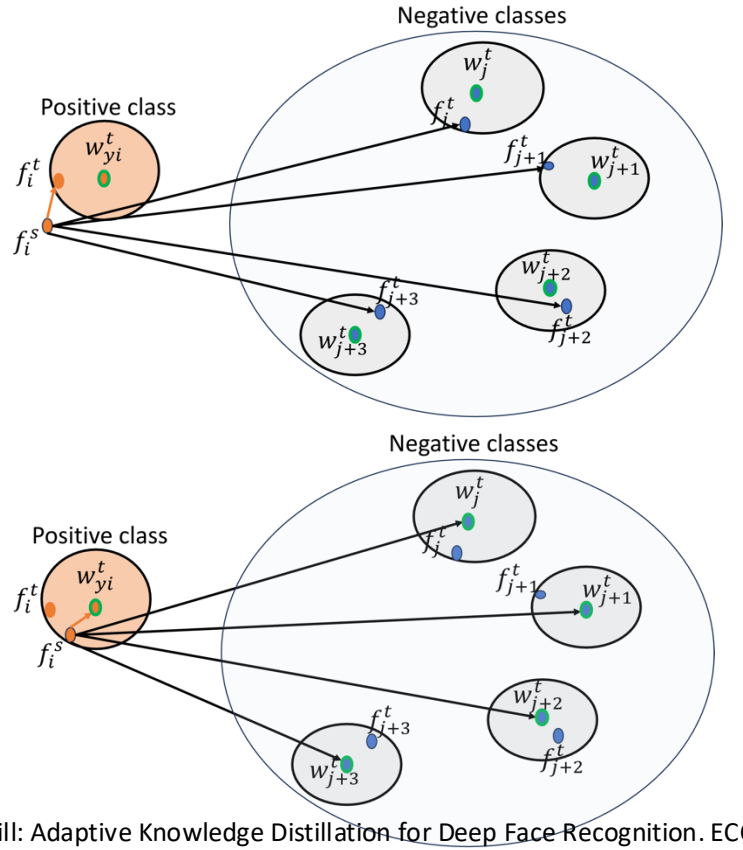
- Feature distillation through classification loss

$$\mathcal{L}_{AMLDistill} = \frac{1}{N}\sum_{i\in N} -log\frac{e^{s(cos(\theta^t_{y_i}+m1)-m2)}}{e^{s(cos(\theta^t_{y_i}+m1)-m2)} + \sum_{j=1,j\neq y_i}^{C} e^{s(cos(\theta^t_j))}}$$

- AdaDistill: Adaptive AMLDistill

$$w^{(k)}_{yi} = \alpha w^{(k-1)}_{yi} + (1-\alpha)(f^{t(k)}_i)^T$$

$$\alpha = \lfloor Cos(f_i, f^t_i) \rceil^1_0$$



Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation

AdaDistill embeds the **KD** concept into the **softmax** loss by training the student using a margin penalty softmax loss with **distilled class centers** from the teacher.

- Feature distillation through classification loss

$$\mathcal{L}_{AMLDistill} = \frac{1}{N}\sum_{i\in N} -log\frac{e^{s(cos(\theta^t_{y_i}+m1)-m2)}}{e^{s(cos(\theta^t_{y_i}+m1)-m2)} + \sum_{j=1,j\neq y_i}^{C} e^{s(cos(\theta^t_j))}}$$

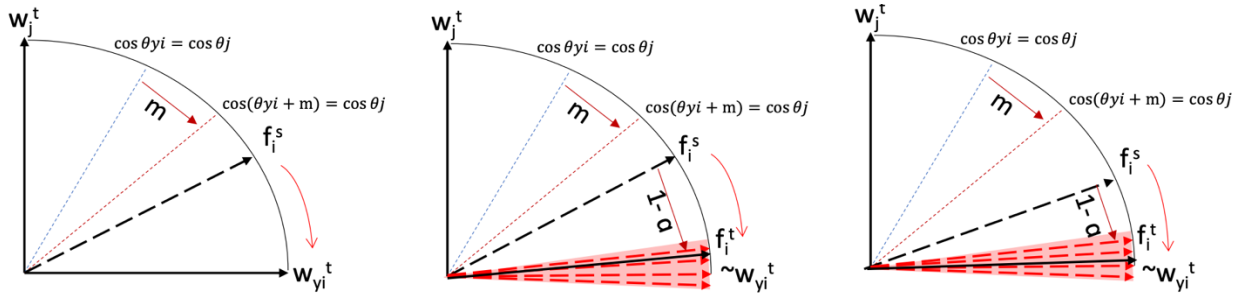
- AdaDistill: Adaptive AMLDistill

$$w^{(k)}_{yi} = \alpha w^{(k-1)}_{yi} + (1-\alpha)(f^{t(k)}_i)^T$$

$$\alpha = \lfloor Cos(f_i, f^t_i) \rceil^1_0$$

- Importance of hard mining

$$\alpha' = \lfloor Cos(f^s_i, f^t_i) \times Cos((w^{(k-1)}_{yi})^T, f^t_i) \rceil^1_0$$

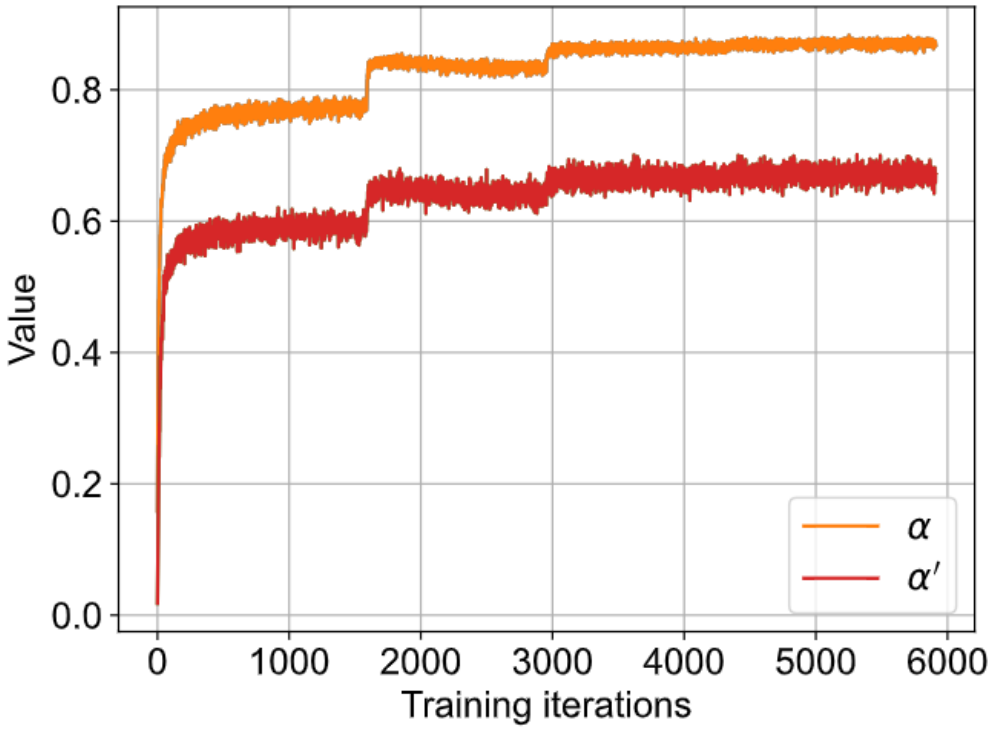


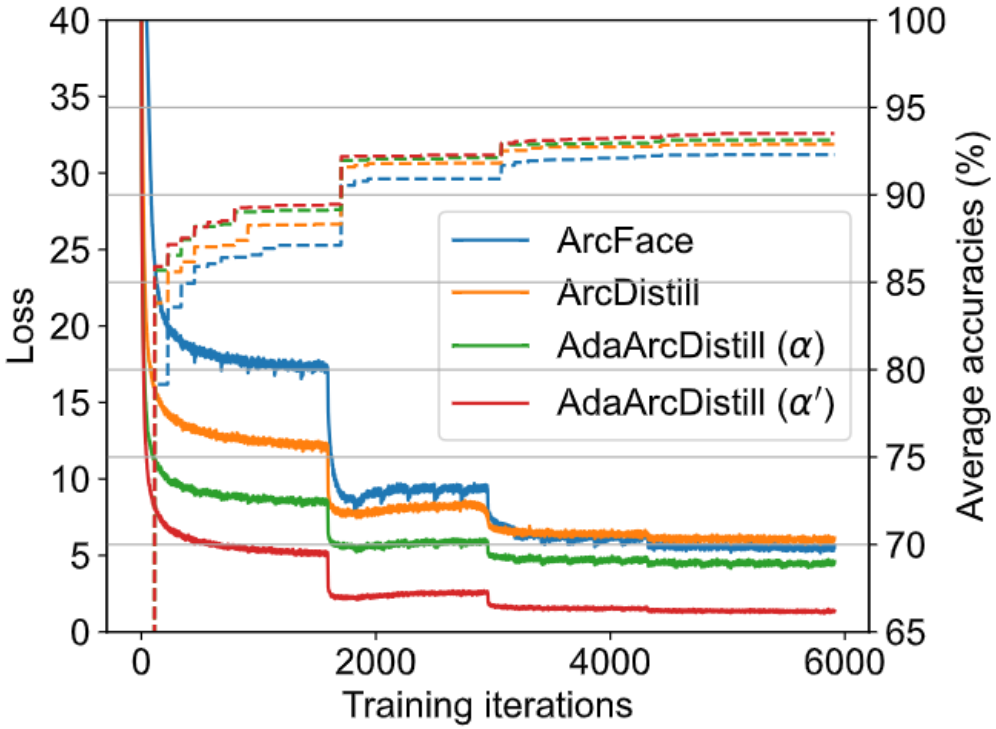
(a) ArcDistill **(b)** AdaDistill (Early-stage) **(c)** AdaDistill (Later-stage)

Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation





Alpha values over training iterations

Loss values over training iterations

Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

AdaDistill

Adaptive Knowledge distillation

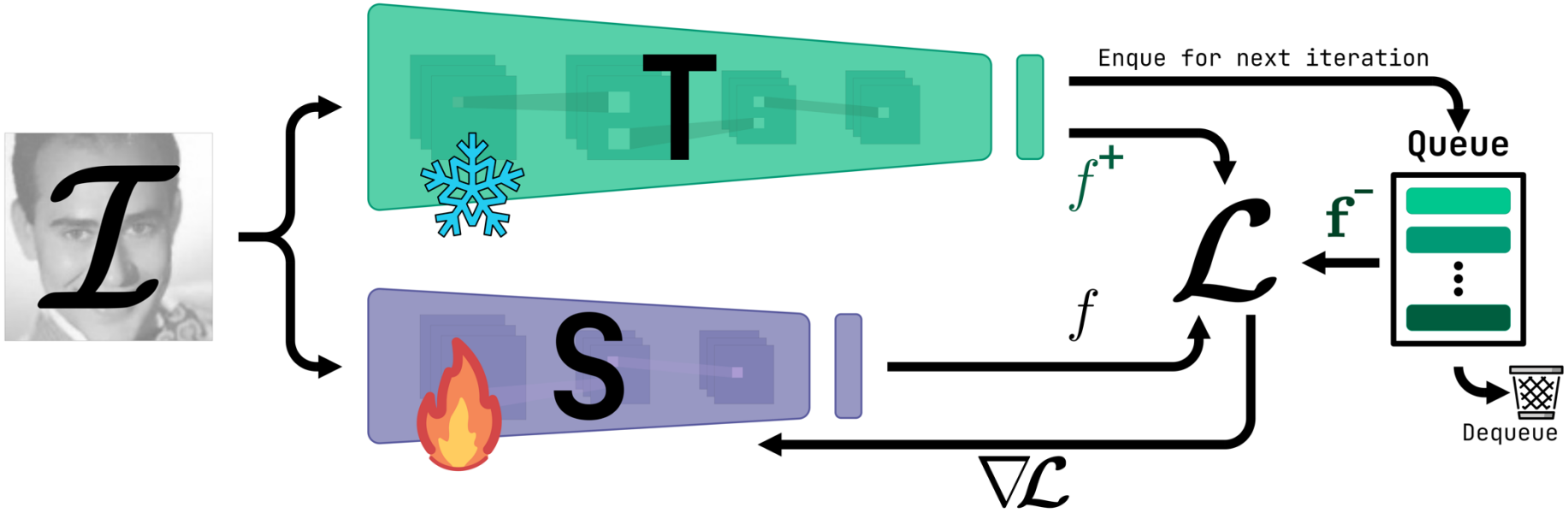
Ablation	Method	LFW	CFP-FP	AgeDB-30	CA-LFW	CP-LFW	Avg.	IJBC 1e-4	IJBC 1e-5
	ResNet50 (Teacher)	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN (Student)	99.52	91.66	95.82	95.12	87.93	94.01	89.13	81.65
1) KD method	MFN + Vanilla KD (Eq. 2)	99.57	93.77	96.97	95.70	89.01	95.00	91.29	79.79
	MFR + ArcDistill (Eq. 4)	99.63	93.83	96.82	95.48	89.20	94.99	91.70	84.57
	MFN + AdaArcDistill (α) (Eq. 4, 6 and 7)	99.60	94.00	96.88	95.65	89.63	95.15	92.56	88.23
	MFN + AdaArcDistill (α') (Eq. 4, 6 and 8)	99.60	95.04	96.78	95.58	90.08	**95.42**	**92.97**	**89.13**
2) Margin Penalty type and value	MFN + AdaArcDistill (α'), m =0.40	99.53	95.21	96.82	95.54	90.02	95.42	92.98	88.87
	MFN + AdaArcDistill (α'), m =0.45	99.58	95.24	96.78	95.55	90.02	**95.47**	**93.27**	**89.32**
	MFN + AdaArcDistill (α'), m =0.50	99.60	95.04	96.78	95.58	90.08	95.42	92.97	89.13
	MFN + AdaCosDistill (α'), m=30	99.50	94.96	95.95	95.20	90.32	95.19	92.92	**89.26**
	MFN + AdaCosDistill (α'), m=35	99.50	95.07	96.60	95.33	90.02	**95.30**	**93.05**	89.21
	MFN + AdaCosDistill (α'), m=40	99.60	93.98	96.57	95,45	89.35	94.99	93.12	89.09
3) Teacher architecture	ResNet18 (Teacher)	99.67	94.47	97.13	95.70	89.73	95,34	93.99	91.14
	MFN + AdaArcDistill(α'), m=45	99.60	94.30	96.43	95.35	89.13	94,96	92.27	88.10
	ResNet50 (Teacher)	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN + AdaArcDistill(α'), m=45	99.58	95.24	96.78	95.55	90.02	95.47	93.27	89.32
	ResNet100 (Teacher)	99.83	98.40	98.33	96.13	93.22	97,19	96.39	94.58
	MFN + AdaArcDistill(α'), m=45	99.63	94.26	96.97	95.60	89.28	95,15	92.54	87.48
	TransFace-B (Teacher)	99.85	99.17	98.53	96.20	92.92	97.33	96.55	94.15
	MF+ AdaArcDistill(α'), m=45	99.57	94.23	96.53	95.47	89.87	95.13	92.85	87.98

Fadi Boutros, Vitomir Struc, Naser Damer: AdaDistill: Adaptive Knowledge Distillation for Deep Face Recognition. ECCV (55) 2024: 163-182

QUD: Unsupervised Knowledge Distillation for Deep Face Recognition

Challenge: Labeled data is challanging to accquire

QUD: Unsupervised Knowledge Distillation for Deep Face Recognition



$$\mathcal{L} = -\log\left(\frac{\exp(f \cdot f^{+}/\tau)}{\exp(f \cdot f^{+}/\tau) + \sum_{f^{-} \in \mathbf{f}^{-}} \exp(f \cdot f^{-}/\tau)}\right)$$

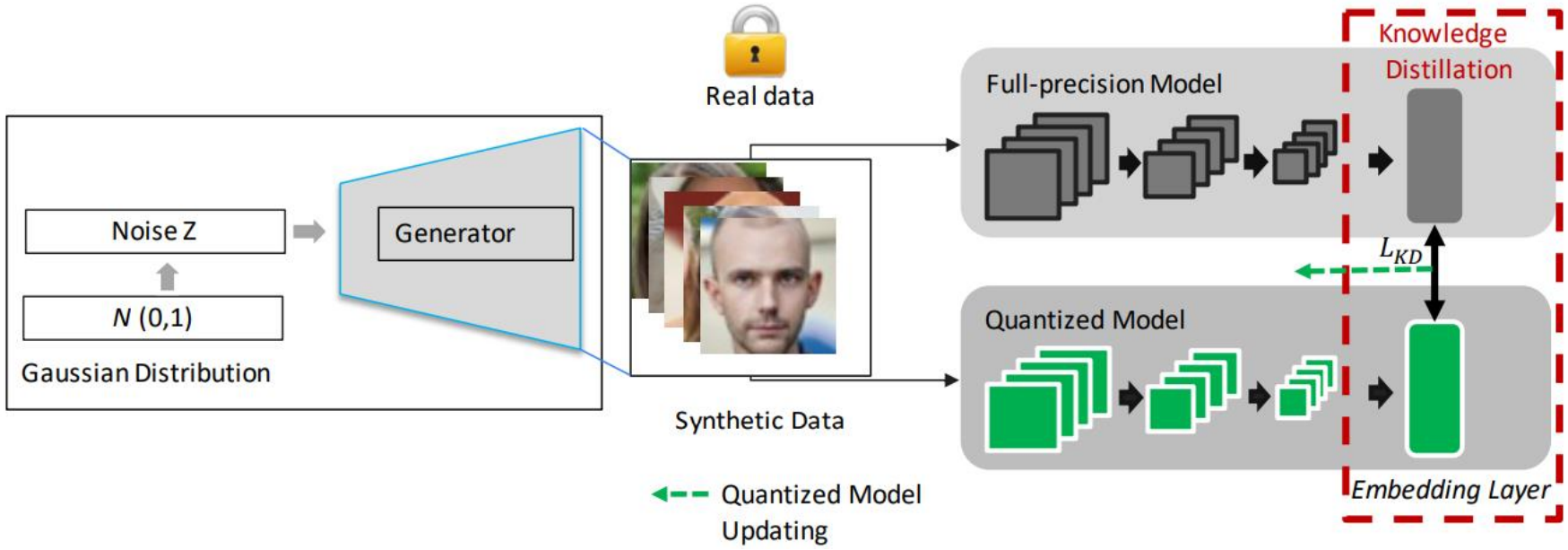
Jan Niklas Kolf, Naser Damer, Fadi Boutros: QUD: Unsupervised Knowledge Distillation for Deep Face Recognition. British Machine Vision Conference 2024.

QUD: Unsupervised Knowledge Distillation for Deep Face Recognition

Study	Method	Distillation Dataset	LFW	CFP-FP	AgeDB-30	CA-LFW	CP-LFW	Avg.	IJB-C 1e-4	IJB-C 1e-5
	ResNet-50 (Teacher)	-	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN (Student)	-	99.52	91.66	95.82	95.12	87.93	94.01	89.13	81.65
I) Temperature τ	MFN + QUD ($\tau = 0.06$, n= 1024)	CASIA-Webface [55]	99.17	93.26	94.12	93.87	89.30	93.94	88.69	75.75
	MFN + QUD ($\tau = 0.08$, n= 1024)		99.43	94.00	94.95	94.87	90.40	94.73	89.65	73.26
	MFN + QUD ($\tau = 0.1$, n= 1024)		99.55	94.14	95.67	94.78	90.50	94.93	**90.28**	**74.96**
	MFN + QUD ($\tau = 0.2$, n= 1024)		99.50	94.41	96.03	95.22	90.13	95.06	88.87	60.93
	MFN + QUD ($\tau = 0.4$, n= 1024)		99.52	94.20	95.77	94.93	89.63	94.81	89.03	65.72
	MFN + QUD ($\tau = 0.6$, n= 1024)		99.58	94.16	95.80	95.00	89.78	94.86	89.43	69.41
II) Queue Size n	MFN + QUD ($\tau = 0.1$, n= 512)	CASIA-Webface [55]	99.53	94.33	95.93	95.13	89.97	94.98	89.74	69.32
	MFN + QUD ($\tau = 0.1$, n= 1024)		99.55	94.14	95.67	94.78	90.50	94.93	**90.28**	**74.96**
	MFN + QUD ($\tau = 0.1$, n= 4096)		99.52	94.29	95.92	95.07	90.00	94.96	89.77	72.30
	MFN + QUD ($\tau = 0.1$, n= 8192)		99.53	94.36	95.60	95.03	90.45	94.99	90.10	71.01
	MFN + QUD ($\tau = 0.1$, n= 16384)		99.57	94.30	95.82	95.07	90.33	95.02	89.32	66.17
III) KD Method	ResNet-50 (Teacher)	-	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN + Feature-based KD	MS1MV2 [17, 21]	99.57	93.77	96.97	95.70	89.01	95.00	91.29	79.79
	MFN + QUD ($\tau = 0.1$, n= 1024)	MS1MV2 [17, 21]	99.58	93.89	96.73	95.65	90.47	95.26	**91.92**	**81.60**
IV) Teacher Architectures	ResNet-50 (Teacher)	-	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN + QUD ($\tau = 0.1$, n= 1024)	MS1MV2 [17, 21]	99.58	93.89	96.73	95.65	90.47	95.26	91.92	81.60
	ResNet-100 (Teacher)	-	99.83	98.40	98.33	96.13	93.22	97.19	96.39	94.58
	MFN + QUD ($\tau = 0.1$, n= 1024)	MS1MV2 [17, 21]	99.68	93.71	97.18	95.62	90.32	95.30	92.19	85.00
	TransFace-B (Teacher)	-	99.85	99.17	98.53	96.20	92.92	97.33	96.55	94.15
	MFN + QUD ($\tau = 0.1$, n= 1024)	MS1MV2 [17, 21]	99.58	93.36	96.65	95.72	90.15	95.09	92.80	87.45
V) Datasets	ResNet-50 (Teacher)	-	99.80	97.63	97.92	96.10	92.43	96.77	96.05	93.96
	MFN (Student)	-	99.52	91.66	95.82	95.12	87.93	94.01	89.13	81.65
	MFN + Feature-based KD	MS1MV2 [17, 21]	99.57	93.77	96.97	95.70	89.01	95.00	91.29	79.79
	MFN + QUD ($\tau = 0.1$, n= 1024)	MS1MV2 [17, 21]	99.58	93.89	96.73	95.65	90.47	95.26	**91.92**	**81.60**
	MFN (Student)	-	98.97	93.21	91.40	91.27	86.42	92.25	77.46	57.75
	MFN + Feature-based KD	CASIA-Webface [55]	99.45	93.90	95.82	94.95	89.48	94.72	**90.63**	**79.54**
	MFN + QUD ($\tau = 0.1$, n= 1024)	CASIA-Webface [55]	99.55	94.14	95.67	94.78	90.50	94.93	90.28	74.96
	MFN (Student)	-	97.05	79.16	81.88	89.37	78.10	85.11	22.19	3.43
	MFN + Feature-based KD	IDiff-Face [6]	99.30	88.79	91.93	93.53	85.40	91.79	60.01	17.31
	MFN + QUD ($\tau = 0.1$, n= 1024)	IDiff-Face [6]	99.33	89.57	92.50	93.73	85.65	92.16	**64.96**	**29.08**
	MFN + Feature-based KD	StyleGAN2-2M	99.32	88.30	92.67	93.82	84.52	91.73	22.56	3.02
	MFN + QUD ($\tau = 0.1$, n= 1024)	StyleGAN2-2M	99.42	90.21	93.88	94.13	86.13	92.76	**49.95**	**19.35**

QuantFace

Model Quantization with Knowledge Distillation



Boutros et al. QuantFace: Towards Lightweight Face Recognition by Synthetic Data Low-bit Quantization. ICPR 2022:

QuantFace

Model Quantization with Knowledge Distillation

Model	param	Quantization data	Bits	Size (MB)	LFW	CFP	AgeDb-30	CALFW	CPLFW	IJB-C	IJB-B
					Accuracy (%)					TAR@FAR 1e-4 (%)	
ResNet100	65.2M	-	FP32	261.22	99.83	98.40	98.33	96.13	93.22	96.50	95.25
		Real data	w8a8	65.31	**99.80**	**98.31**	**98.13**	**96.05**	**92.92**	**96.38**	**95.13**
		Synthetic data	w8a8	65.31	**99.80**	98.14	97.95	96.02	92.90	96.09	94.74
		Real data	w6a6	49.01	**99.55**	89.14	95.85	95.42	85.63	85.80	84.08
		Synthetic data	w6a6	49.01	99.45	**91.00**	**96.43**	**95.58**	**86.60**	**87.00**	**85.06**
ResNet50	43.6M	-	FP32	174.68	99.80	98.01	98.08	96.10	92.43	95.74	94.19
		Real data	w8a8	43.67	**99.78**	**97.70**	**98.00**	**96.00**	**92.17**	**95.66**	**94.15**
		Synthetic data	w8a8	43.67	**99.78**	97.43	97.97	95.87	92.08	95.18	93.67
		Real data	w6a6	32.77	**99.70**	95.00	97.17	**95.87**	90.17	**91.74**	**90.07**
		Synthetic data	w6a6	32.77	99.68	**95.17**	**97.43**	95.70	**90.38**	90.72	89.44
ResNet18	24.0M	-	FP32	96.22	99.67	94.47	97.13	95.70	89.73	93.56	91.64
		Real data	w8a8	24.10	**99.63**	**94.46**	97.03	**95.72**	89.48	**93.56**	**91.57**
		Synthetic data	w8a8	24.10	99.55	94.04	**97.07**	95.58	**89.53**	92.87	91.01
		Real data	w6a6	18.10	99.52	93.23	96.55	**95.58**	88.37	**93.03**	**91.08**
		Synthetic data	w6a6	18.10	**99.55**	**93.34**	**96.62**	95.32	**89.05**	92.36	90.38
MobileFaceNet	1.1M	-	FP32	4.21	99.47	91.59	95.62	95.15	87.98	90.88	88.54
		Real data	w8a8	1.10	**99.43**	**91.40**	**95.47**	**95.05**	**87.95**	**90.57**	**88.32**
		Synthetic data	w8a8	1.10	99.35	90.84	94.37	94.78	87.73	89.21	86.98
		Real data	w6a6	0.79	98.87	**87.69**	**93.03**	93.30	84.57	**83.13**	80.53
		Synthetic data	w6a6	0.79	**99.08**	87.64	91.77	**93.48**	**84.85**	82.94	**80.58**

Recap

What is the primary goal of knowledge distillation?

a) Increase the training time of a model

b) Transfer knowledge from a large model to a smaller model

c) Compress input data

d) Improve data labeling quality

Recap

Which of the following models is typically called the “teacher” in knowledge distillation?

a) A small and efficient model

b) A model with fewer parameters

c) A large, accurate model

d) Any model with dropout layers

Recap

Why do we use a temperature parameter in softmax during distillation?

a) To make the loss function sharper

b) To normalize the logits

c) To smooth the probability distribution of the outputs

d) To increase the learning rate

Recap

Which type of loss function is typically used in knowledge distillation?

a) Mean Squared Error

c) Kullback-Leibler Divergence

d) SmoothL1

Recap

What does a higher temperature in the softmax function produce?

a) More confident predictions

b) A uniform distribution

c) Softer probability distribution

d) Harder labels

Recap

True/False Questions

___ Knowledge distillation is only used in supervised learning scenarios.

___ One benefit of knowledge distillation is to reduce inference time on edge devices.

___ The student model is always trained using the same data as the teacher model.

___ The soft labels from the teacher model contain more information than hard labels.

___ Distillation can only be done once per model and cannot be repeated.

Recap

Softmax with Temperature Scaling

Softmax with Temperature Scaling

Given the logits from a teacher model:

$$z = [2.0, 1.0, 0.1]$$

Compute the softmax probabilities **with a temperature T=1** and **with T=2**.

$$p_t(i) = \frac{\exp(z_t^i / T)}{\sum_{j=1}^{C} \exp(z_t^j / T)}$$

$$p_s(i) = \frac{\exp(z_s^i / T)}{\sum_{j=1}^{C} \exp(z_s^j / T)}$$

Recap

Softmax with Temperature Scaling

```
import numpy as np
import math

def softmax(x):
"""Compute softmax values for each sets of scores in x."""
return np.exp(x) / np.sum(np.exp(x), axis=0)

# 1. Softmax with Temperature
def softmax_with_temperature(logits, T=1.0):
scaled_logits = np.array(logits) / T
return softmax(scaled_logits)

logits = [2.0, 1.0, 0.1]
print("Softmax (T=1):", softmax_with_temperature(logits, T=1))
print("Softmax (T=2):", softmax_with_temperature(logits, T=2))
```

Recap

Softmax with Temperature Scaling

Suppose the teacher's softened output probabilities are:

Pt=[0.7,0.2,0.1]

And the student outputs:

Ps=[0.6,0.3,0.1]

Compute the **Kullback–Leibler divergence** DKL(Pt‖Ps).

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Recap

Softmax with Temperature Scaling

```
# KL Divergence

def kl_divergence(P, Q):

return sum(p * math.log(p / q) for p, q in zip(P, Q) if p != 0)


P_teacher = [0.7, 0.2, 0.1]

P_student = [0.6, 0.3, 0.1]

kl = kl_divergence(P_teacher, P_student)

print("KL Divergence:", kl)
```

Recap

Softmax with Temperature Scaling

Distillation Loss Calculation

Given:

•Ground-truth label: class 1 , Student logits: [1.5,2.5,0.3], Teacher logits: [1.0,3.0,0.5] , Temperature T=2T = 2T=2

•Use cross-entropy with soft targets and include both hard and soft loss components

Calculate the **total distillation loss**:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{soft}} + (1 - \alpha) \cdot \mathcal{L}_{\text{hard}}$$

Assume α=0.7, and provide intermediate steps (softmax, soft loss, hard loss).

Recap

Softmax with Temperature Scaling

```
# 3. Distillation Loss
def distillation_loss(student_logits, teacher_logits, T=2.0, alpha=0.7, true_label=1):
teacher_soft = softmax_with_temperature(teacher_logits, T)
student_soft = softmax_with_temperature(student_logits, T)
# Soft loss (KL divergence)
soft_loss = kl_divergence(teacher_soft, student_soft)

# Hard loss (cross-entropy)
student_probs = softmax(student_logits)
hard_loss = -np.log(student_probs[true_label])

total_loss = alpha * soft_loss + (1 - alpha) * hard_loss
return {
"soft_loss": soft_loss,
"hard_loss": hard_loss,
"total_loss": total_loss
}

student_logits = [1.5, 2.5, 0.3]
teacher_logits = [1.0, 3.0, 0.5]
losses = distillation_loss(student_logits, teacher_logits, T=8.0, alpha=0.7, true_label=1)
print("Distillation Loss:", losses)
```

Softmax with Temperature Scaling

Sample code

https://colab.research.google.com/drive/1LElmWSVVDE7PiOop-dN6sji28hNG26Zd?usp=sharing

Fraunhofer IGD

Part 2- Handon

Handson

Setups:

- Required tools: Python, PyTorch (I am using Pytorch, however you are free to use any deep learning framework)
- You can use your own PC or setup Google Colab https://colab.research.google.com/ You only need a Gmail account
- Dataset: MNIST. The dataset is split between training and testing

Hands-on

Response-based Knowledge distillation

- ❑ Train a teacher model on MNIST with multi-class classification loss
- ❑ Optimizer: Adam
- ❑ Learning rate: 1e-3
- ❑ Number of epochs: 5
- ❑ Loss: Cross Entropy
- ❑ Evaluate on testing subset

```python
# Define the Teacher Model
class TeacherModel(nn.Module):
    def __init__(self):
        super(TeacherModel, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(64*7*7, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = x.view(-1, 64*7*7)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Hands-on

Response-based Knowledge distillation

- ❑ Train a teacher model on MNIST with multi-class classification loss
- ❑ Optimizer: Adam
- ❑ Learning rate: 1e-3
- ❑ Number of epochs: 5
- ❑ Loss: Cross Entropy
- ❑ https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html#torch.nn.CrossEntropyLoss
- ❑ Evaluate on testing subset: Teacher Model Accuracy: 99.14%

Hands-on

Response-based Knowledge distillation

- ❑ Train a student model on MNIST with multi-class classification loss
- ❑ Optimizer: Adam
- ❑ Learning rate: 1e-3
- ❑ Number of epochs: 5
- ❑ Loss: Cross Entropy
- ❑ Evaluate on testing subset: Standalone Student Model Accuracy: 98.91%

Hands-on

Response-based Knowledge distillation

- ❑ Implementing Knowledge Distillation
- ❑ Load pre-trained teacher model.
- ❑ Compute soft labels using temperature scaling.
- ❑ Train student model using a combination of cross-entropy loss and KD loss
- ❑ https://pytorch.org/docs/stable/generated/torch.nn.KLDivLoss.html
- ❑ Evaluate on testing subset
- ❑ Loss weight term: 0.5
- ❑ Temperature: T=3

Student Model Accuracy: 99.0%

Hands-on

Response-based Knowledge distillation

Solution:

https://colab.research.google.com/drive/1fTnzPHPi5Ol9Hbi33YFB-TfhVaeB4oO9?usp=sharing

Hands-on

Feature-based Knowledge distillation

- Using the same previous configuration retrain the student model with feature-based distillation.
- Modify the teacher and student architecture to return feature representation from last FC layer (layer before the classification layer)
- Minimize e.g., L2 loss between feature embedding from teacher and student
- Note: the dimensionality of teacher and student feature representation is not equal

Data-free Knowledge distillation

Given a pretrained teacher model:

- ❑ Design an approach to distill the knowledge from teacher to student without accessing the training dataset
- ❑ Scenario 1: You have access to conditional generative model (G) that can generate data, following MNIST distribution. The input for G is random vector of size 100 and class label in [0..9] and it outputs an images of size 1x28x28
- ❑ Scenario 2: You don't have access to data or pretrained generative model. Be creative ☺

Knowledge Distillation: Compact Model Performance for Scalable, Green, and Efficient Deployment

11.04.2025

Fadi Boutros

Fadi.boutros@igd.fraunhofer.de