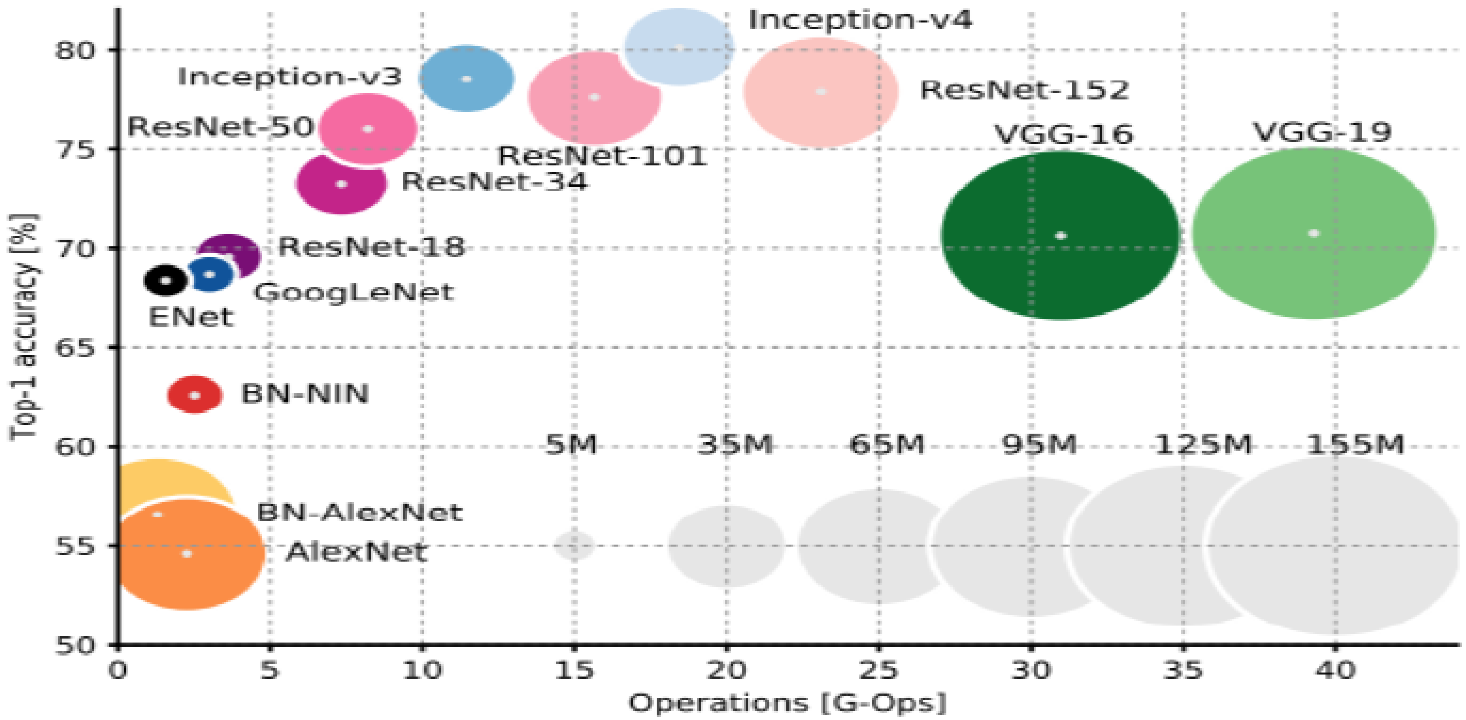
CSDS503 / COMP552 – Advanced Machine Learning

Faizad Ullah

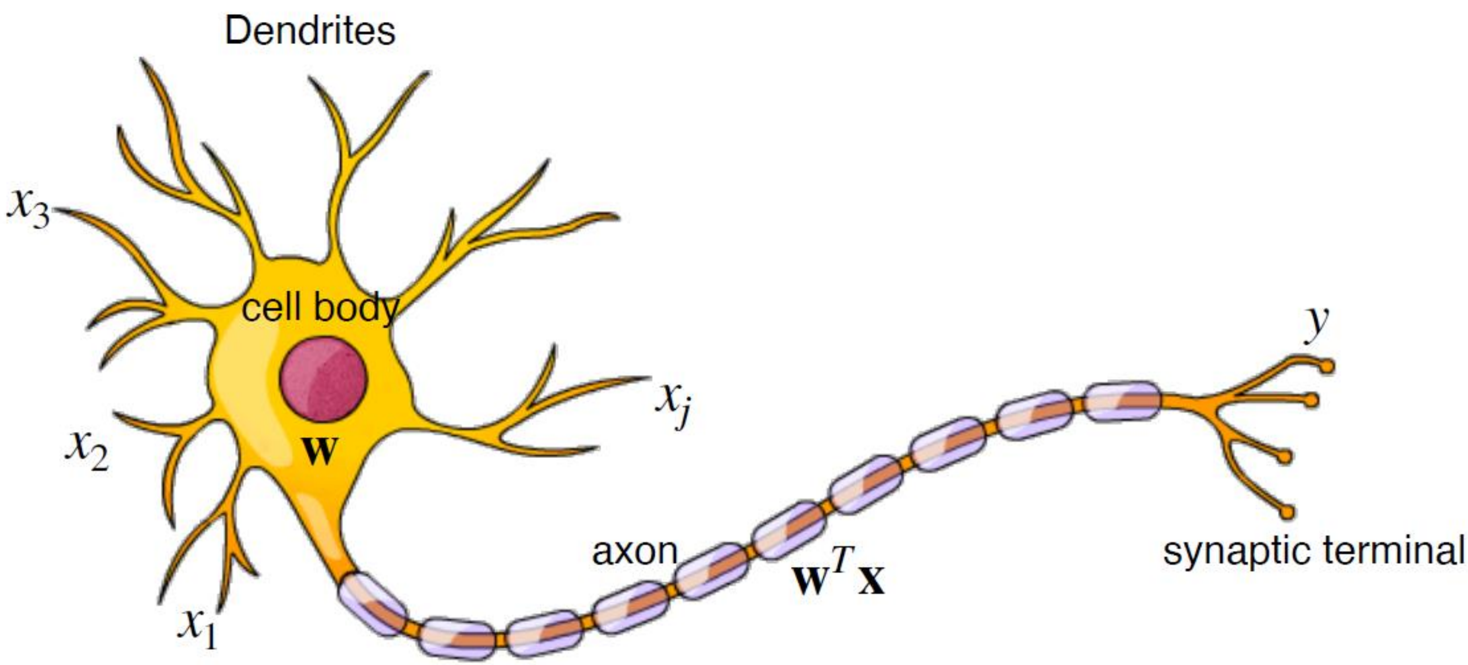
Network Compression

Breakthroughs in Deep Learning

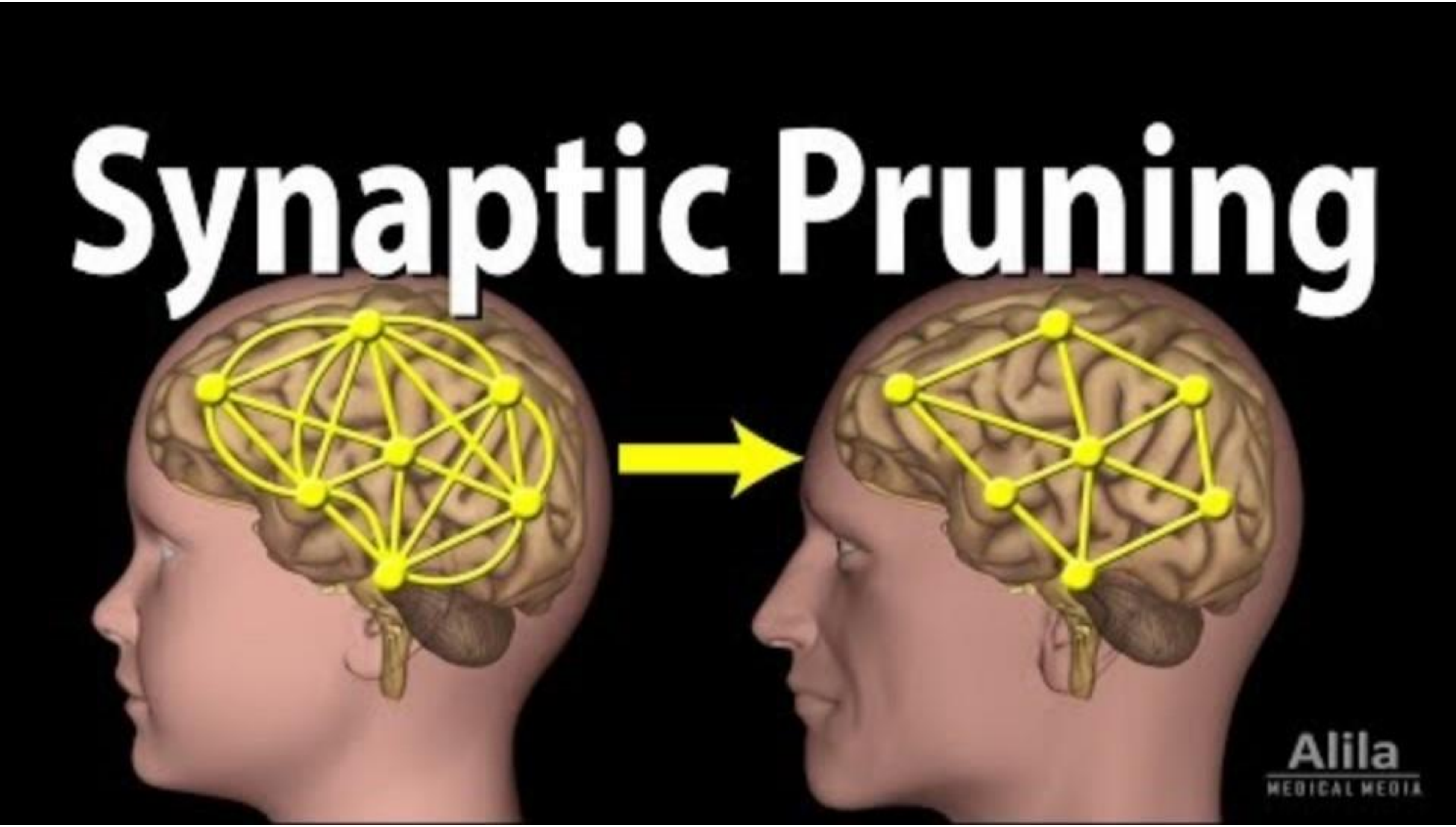


Shortcoming

?



Synaptic Pruning

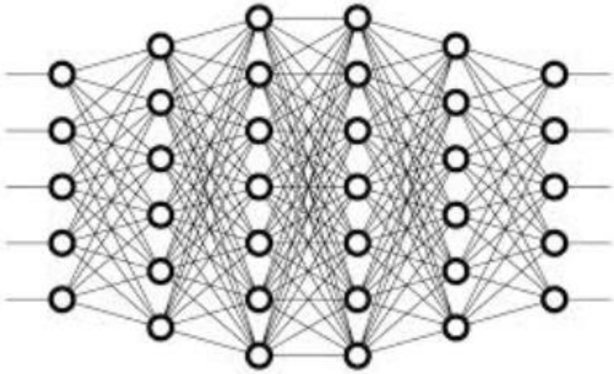


Network Compression

- Complex networks require a huge amount of memory and compute

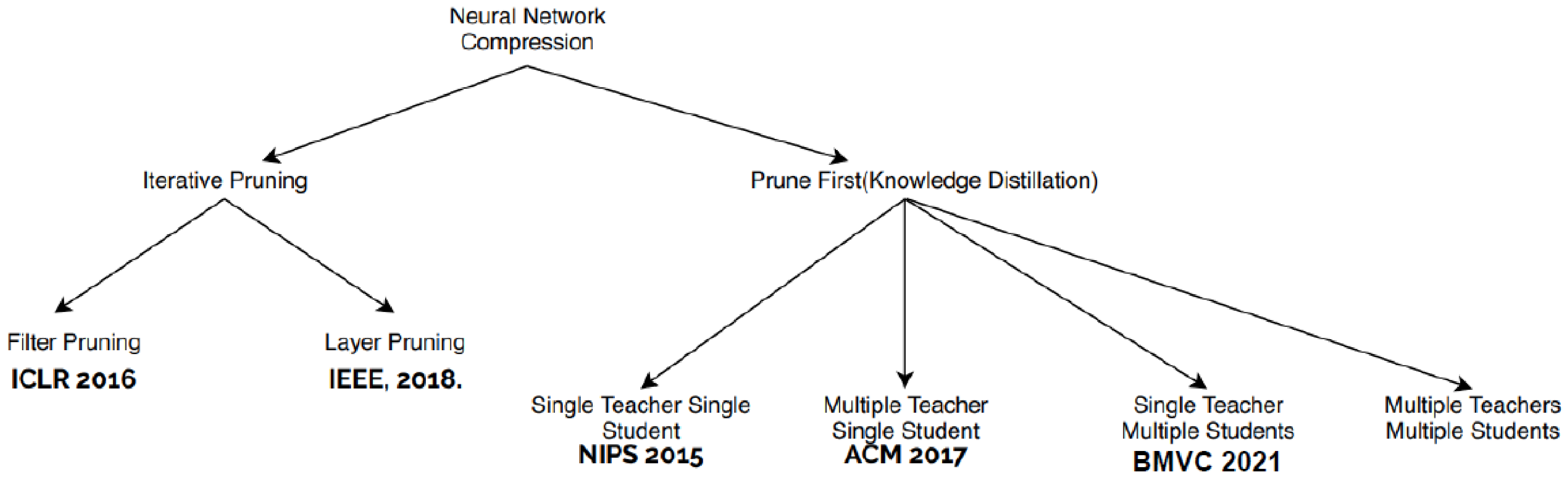
- Portable devices have very **limited resources** in terms of **memory** and **compute**

- **Deploying** complex networks on portable devices is **almost practically impossible** since portable devices are **not capable enough**



Structured Pruning

Network Compression



Network Compression

- Unstructured Pruning
  - Weights
  - L1 Regularisation
  - Random Dropout
- Structured Pruning
  - Filter Pruning
  - Layer Pruning

Network Compression

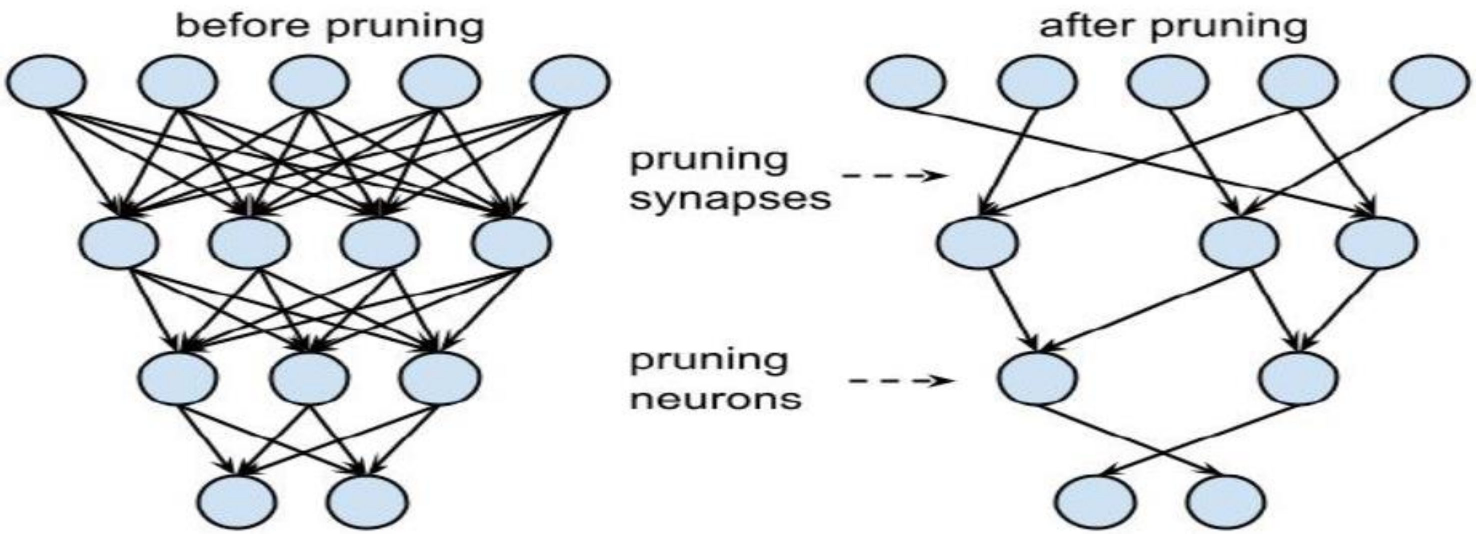
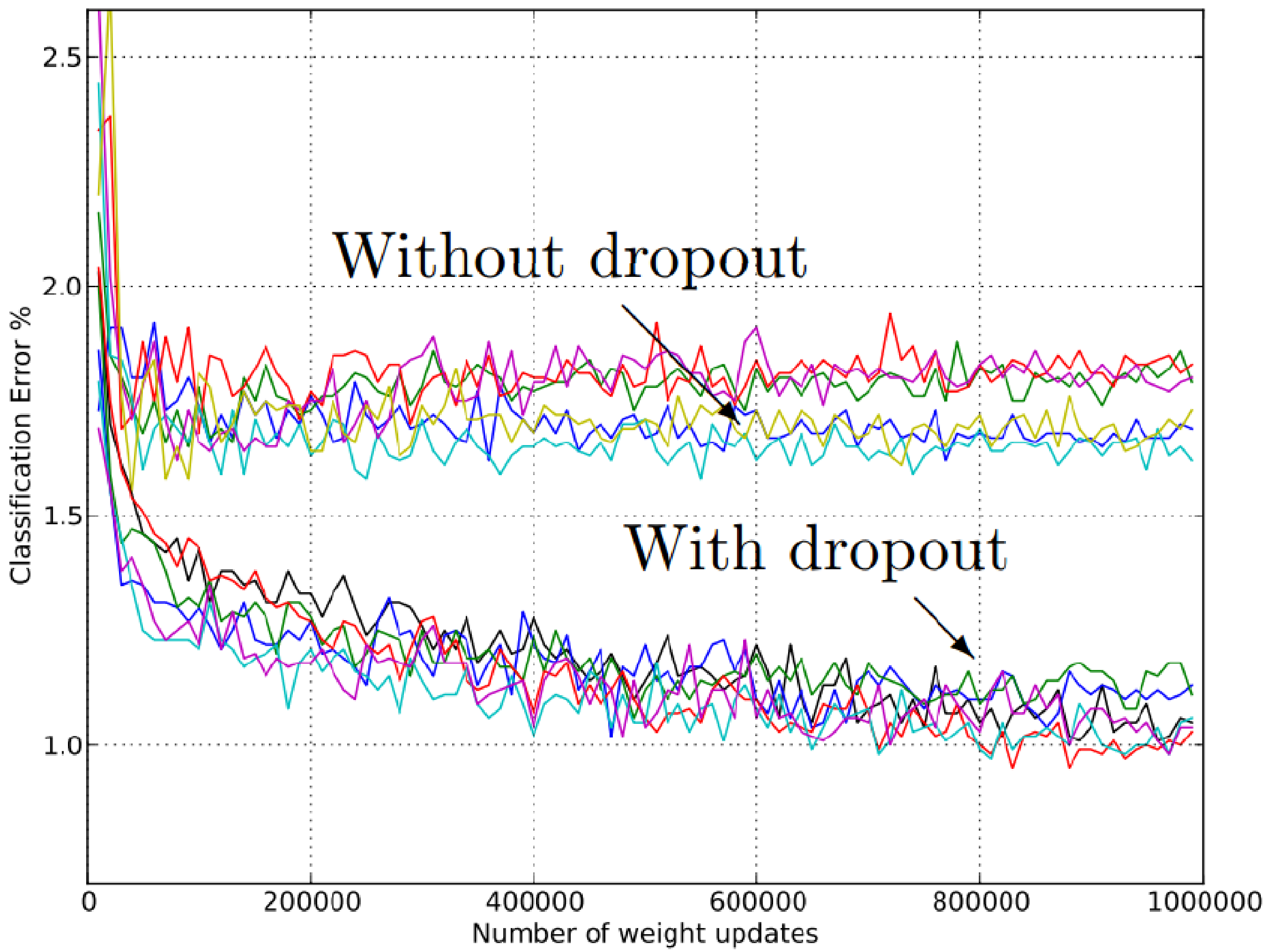


Image courtesy of Song Han

Network Compression

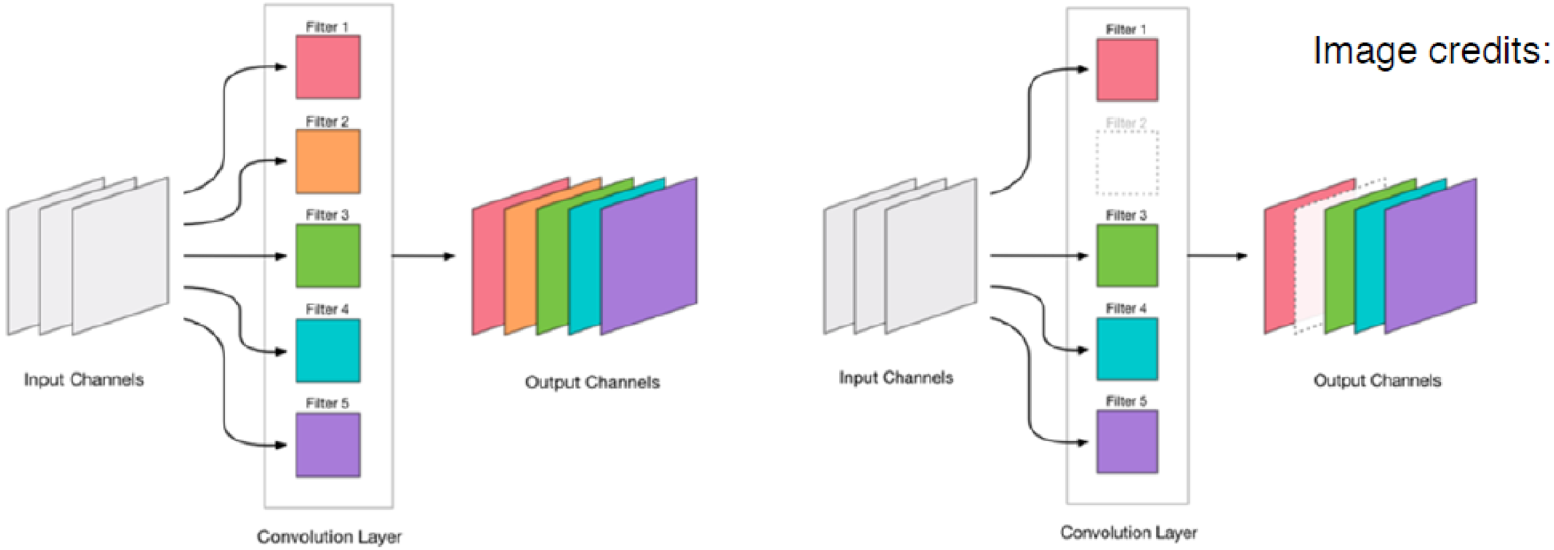


N. Srivastava et al. 2014

Network Compression

- Structured Networks & Structured Pruning

Image credits: Google



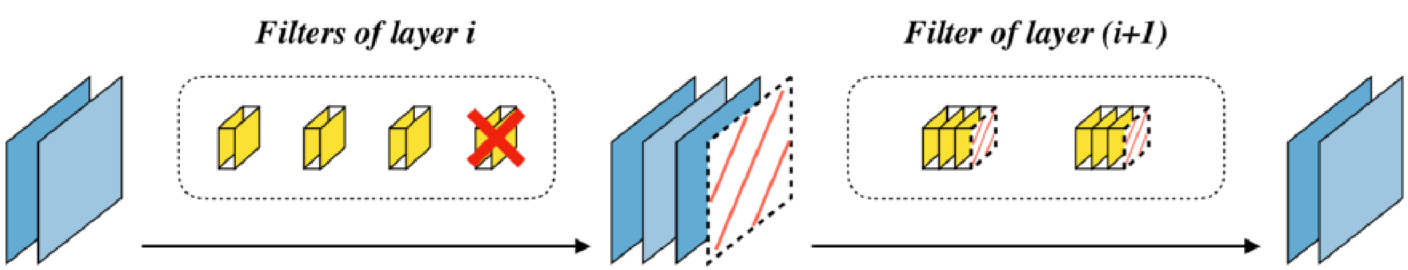


Figure 1: An illustration of filter pruning. The i-th layer has 4 filters (*i.e.* channels). If we remove one of the filters, the corresponding feature map will disappear, and the input of the filters in the $(i+1)$-th layer changes from 4 channels to 3 channels.

Image credits: Gate Decorator Z. You et. al. 2019

Network Compression

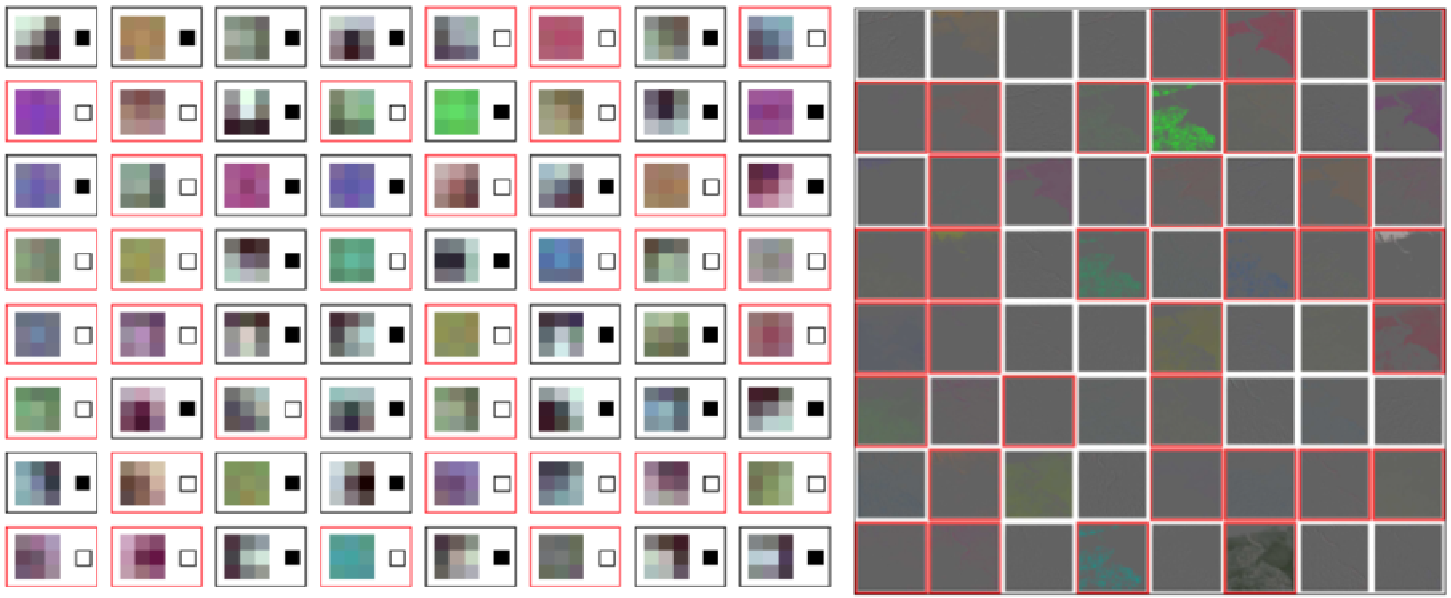
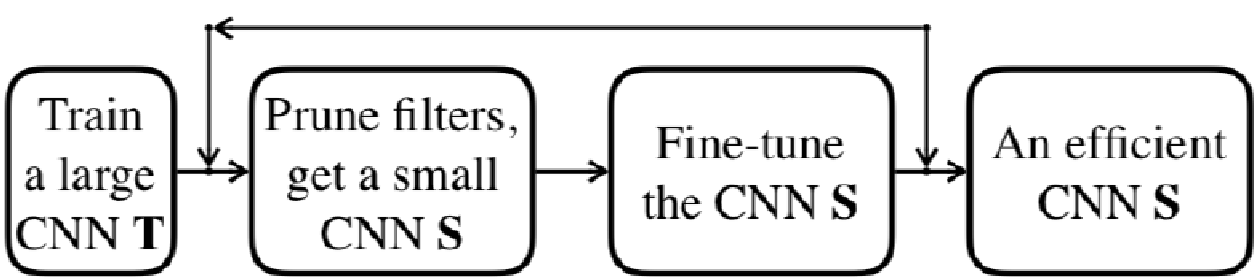


Image credits: S. Lin et al. IJCAI 2018

Global Filter Importance Ranking



(a) The Traditional Pruning Paradigm

Image credits: X. Dong et. al. NeurIPS2019

Network Compression

Correlation Based Filter Pruning

- 1. Remove highly correlated filter, fine tune
- 2. Train by introducing filter correlation term in loss function. This will result in network with highly correlated filters. Remove highly correlated filter, fine tune
- 3. Sparse Subspace Clustering of highly correlated filters

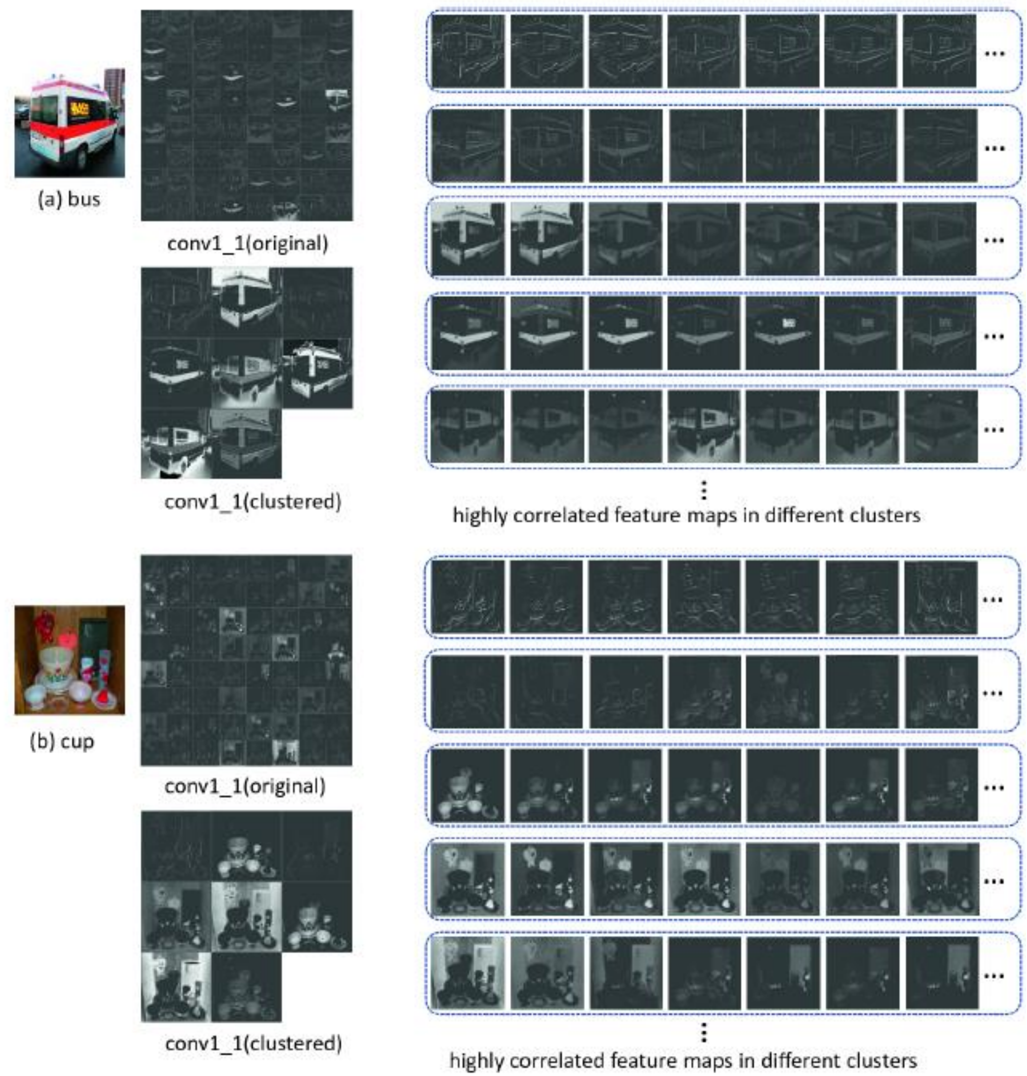


Image credits: D. Wang et. al., ICIG 2019

Network Compression

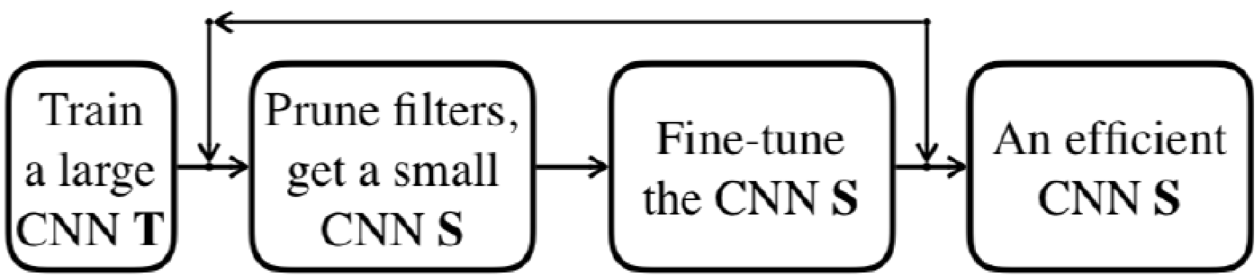
Pruning Filters for Efficient ConvNets (ICLR 2017)

- Procedure
  - 1. For each filter $F_{i,j}$, calculate the sum of its absolute kernel weights $s_j = \sum |F_{i,j}|$ i.e. its L1-norm
  - 2. Sort the filters by s_j.
  - 3. Prune m filters with the smallest sum values and their corresponding feature maps. The kernels in the next convolutional layer corresponding to the pruned feature maps are also removed.
  - 4. A new kernel matrix is created for both the i-th and i+1-th layers, and the remaining kernel weights are copied to the new model

Network Compression

- All of the previous approaches of filter pruning had a major shortcoming of **not being able to do pruning while training**, all of them had to **stop training to prune** and then start again and repeat this cycle many times.

- This may result in a pruned network that fails to achieve the same accuracy as compared to the original network.



(a) The Traditional Pruning Paradigm

Image credits: X. Dong et. al. NeurIPS2019

Network Compression

Prune while training

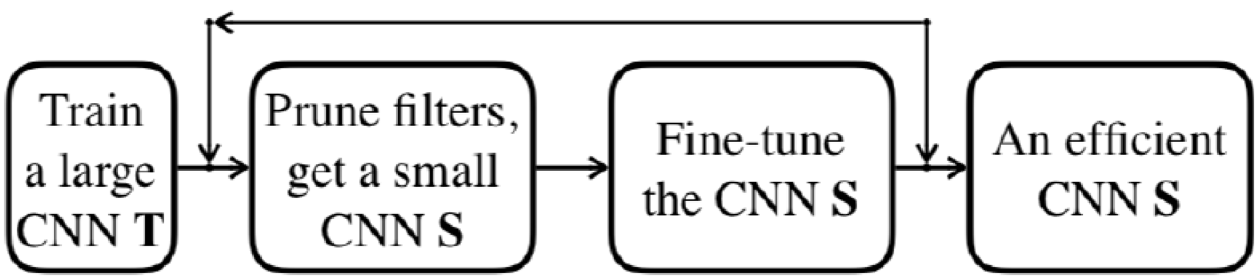
- Design a specialised network that can be pruned while training
- This network will have additional connections or on/off switches with filters and layers
- The on/off decision is a learnable parameter itself

- Filter Pruning
  - Usually a custom dropout layer is added
  - Either the filter or its activation is multiplied with switch state (0, 1, between 0-1)
  - Structured Sparsity Regularization
- Layer Pruning
  - Usually a residual layer is added

Network Compression

- Criteria
  - Magnitude of filter
  - Magnitude of activations
  - Clustering of filters (to remove redundancy)

- Implementation of criteria via loss function
  - Loss = Error + λRegulariser



(a) The Traditional Pruning Paradigm

Image credits: X. Dong et. al. NeurIPS2019

Network Compression

Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks, NeurIPS 2019

Problem Definition

- Let $\mathscr{L}(X, Y; \theta)$ denotes loss function
- X: Input data, Y: Output label,
- θ: model parameters, θ_k^-: removed params, θ_k^+: remaining params
- $\mathcal{K}$: set of all filters of the network

- Filter Pruning: The key to global pruning methods is to solve the global filter importance ranking (GFIR) problem
- Using importance ranking, choose a subset of filters $k \subset \mathcal{K}$ and remove their parameters θ_k^- from the network
- To minimise the loss increase, choose k^* by solving

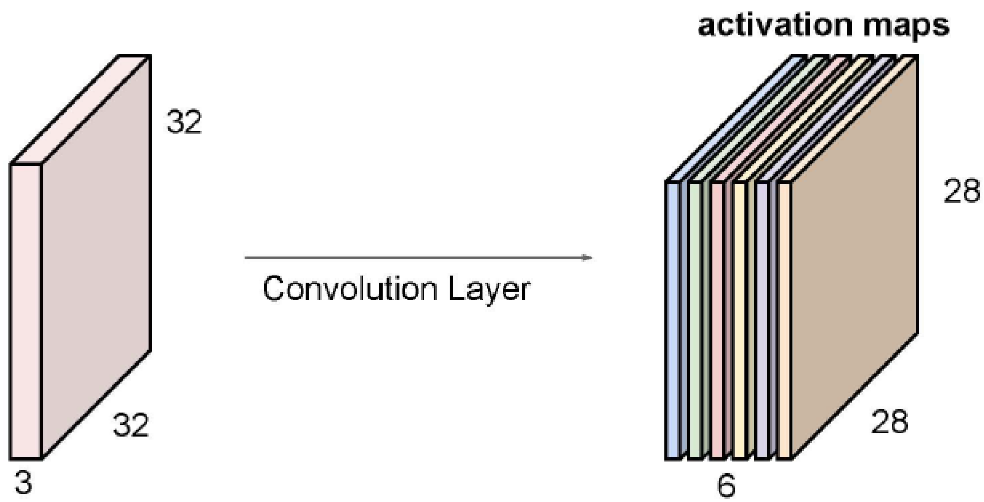
$$k^* = \arg\min_k \left| \mathcal{L}(X, Y; \theta) - \mathcal{L}(X, Y; \theta_k^+) \right| \quad s.t. \ \|k\|_0 > 0$$

Network Compression

Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks, NeurIPS 2019

$$k^* = \arg\min_k \left| \mathcal{L}(X, Y; \theta) - \mathcal{L}(X, Y; \theta_k^+) \right| \quad s.t.\ \|k\|_0 > 0$$

Assuming that feature map z is the output of the filter k, we multiply z by a trainable scaling factor $\phi \in \mathbb{R}$ and use $\hat{z} = \phi z$ for further calculations. When the gate ϕ is zero, it is equivalent to pruning the filter k.



$\{\phi_1, \phi_2, \cdots, \phi_6\}$

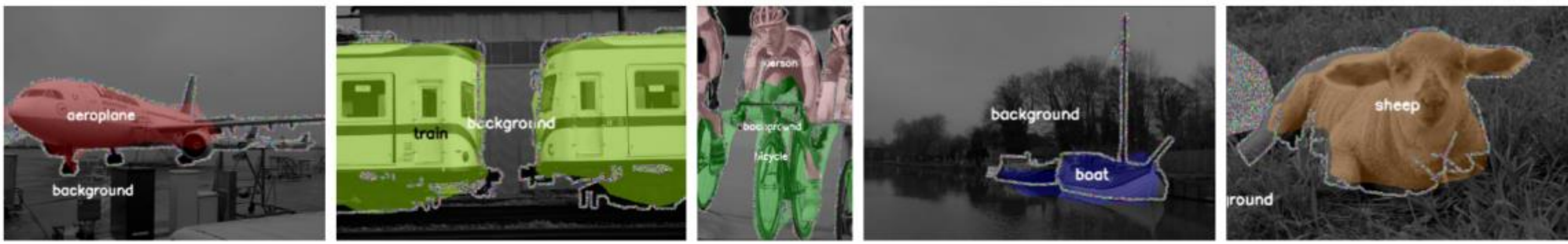
$$\frac{\partial L}{\partial \phi_i} = ?$$

Network Compression

GBN with Tick-Tock		
Param	Finetune	Scratch
69.0%	**74.6**	73.7
85.5%	**73.2**	73.0
94.7%	**71.2**	69.9

Network Compression

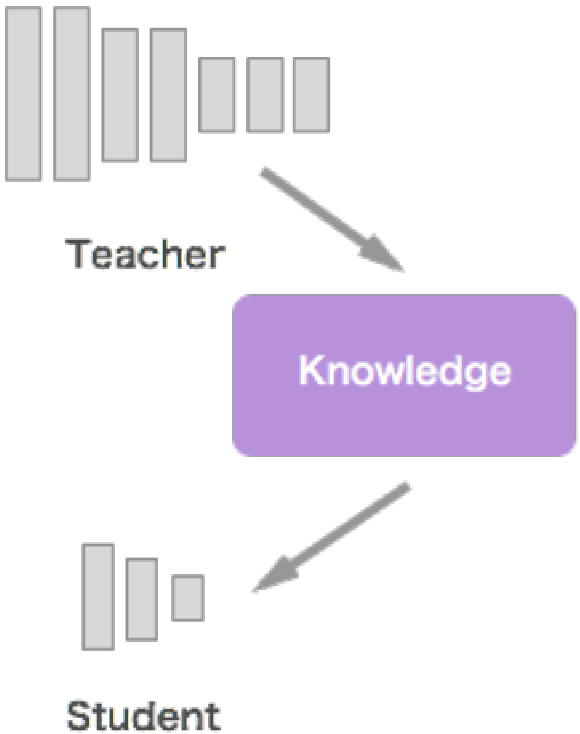
Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks, NeurIPS 2019

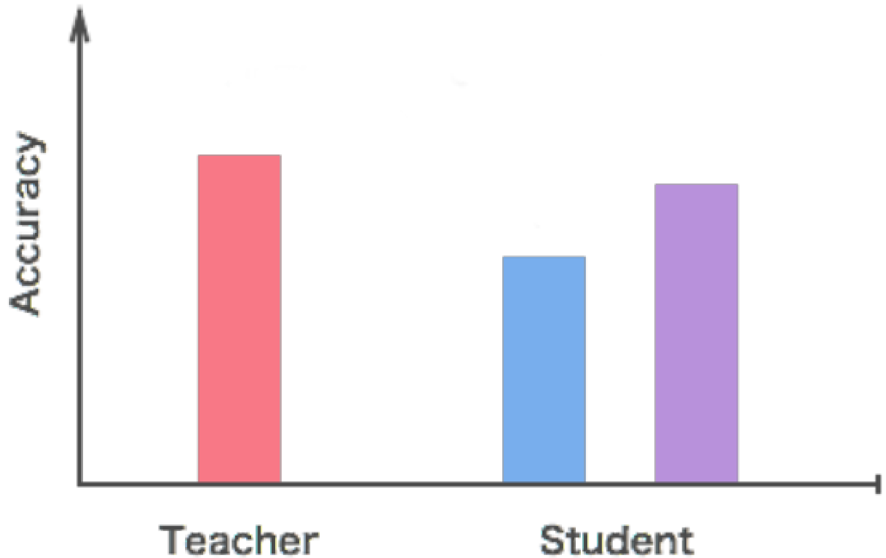


(a) Baseline (mIoU: 62.84)

(b) Pruned (mIoU: 62.88)

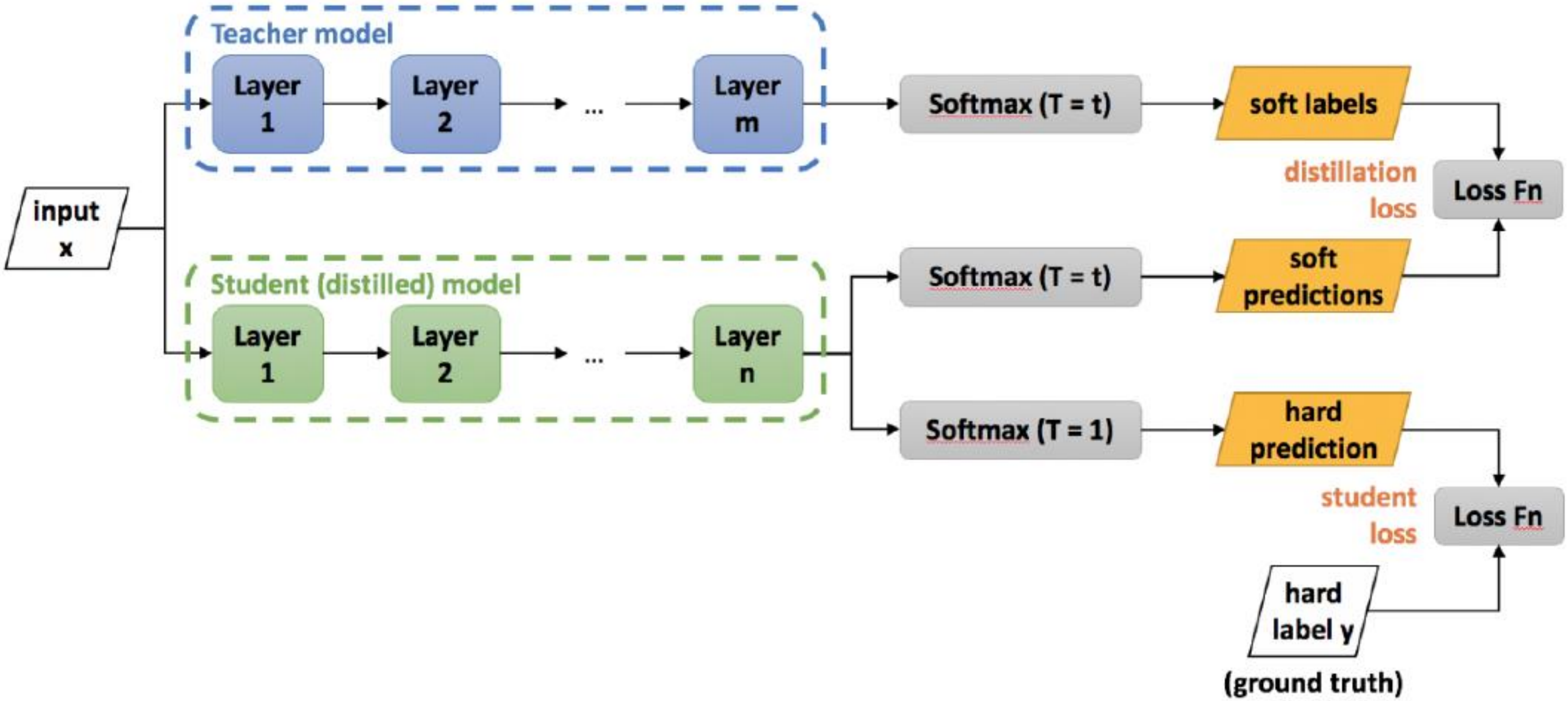
Network Compression





Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

Network Compression



Network Compression

- Neural networks typically produce class probabilities by using a "softmax" output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_i with the other logits.

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

- where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes.

Network Compression

$$q_i = \frac{exp(z_i)}{\sum exp(z_j)} \quad q_i = \frac{exp(z_i/T)}{\sum exp(z_j/T)}$$

cow	dog	cat	car
10^{-6}	.9	.1	10^{-9}

cow	dog	cat	car
.05	.3	.2	.005

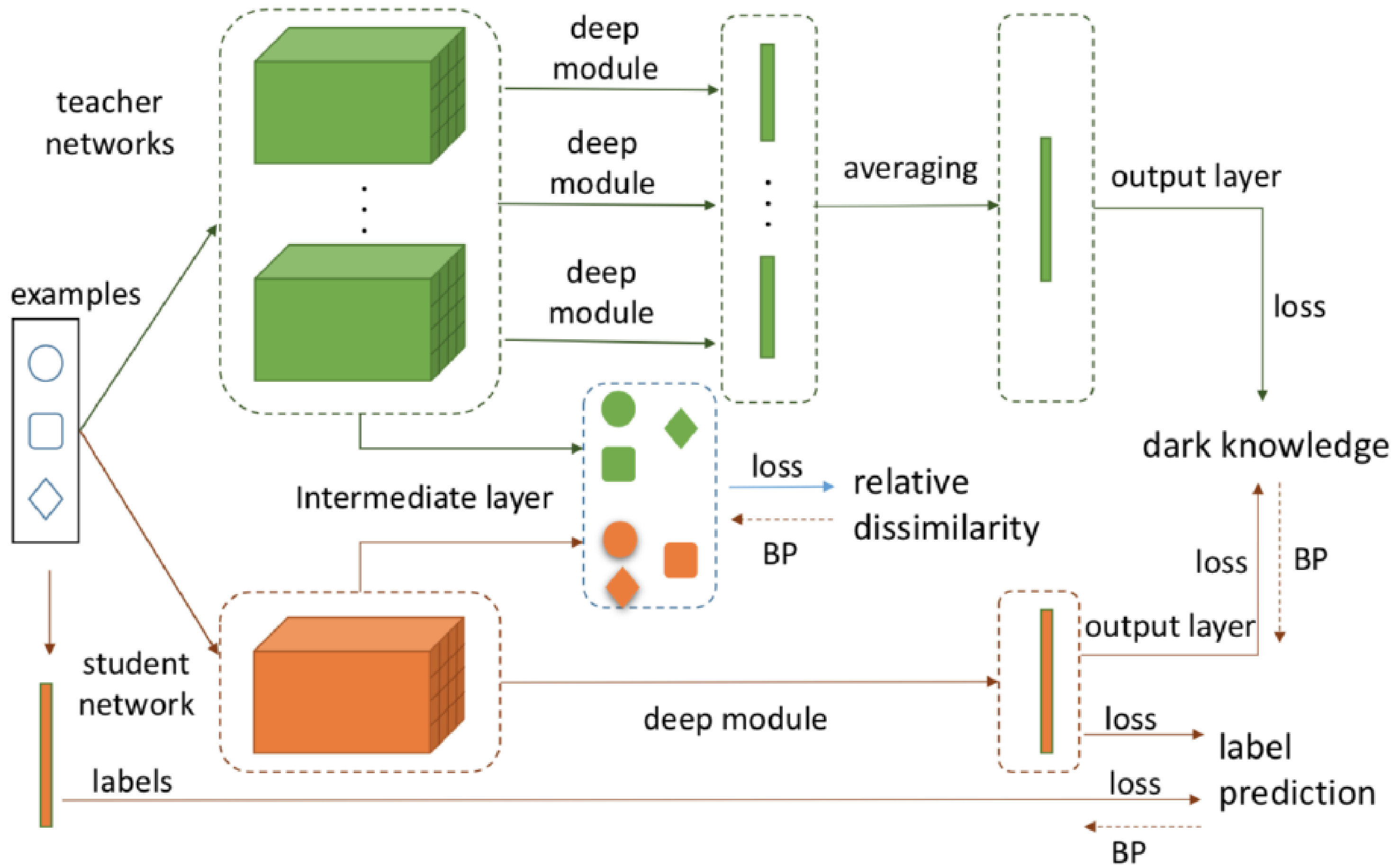
Network Compression

- Knowledge distillation otherwise also called **student-teacher** network refers to the idea of **model compression** by **teaching a smaller network**, step by step, exactly what to do using a **bigger already trained network**

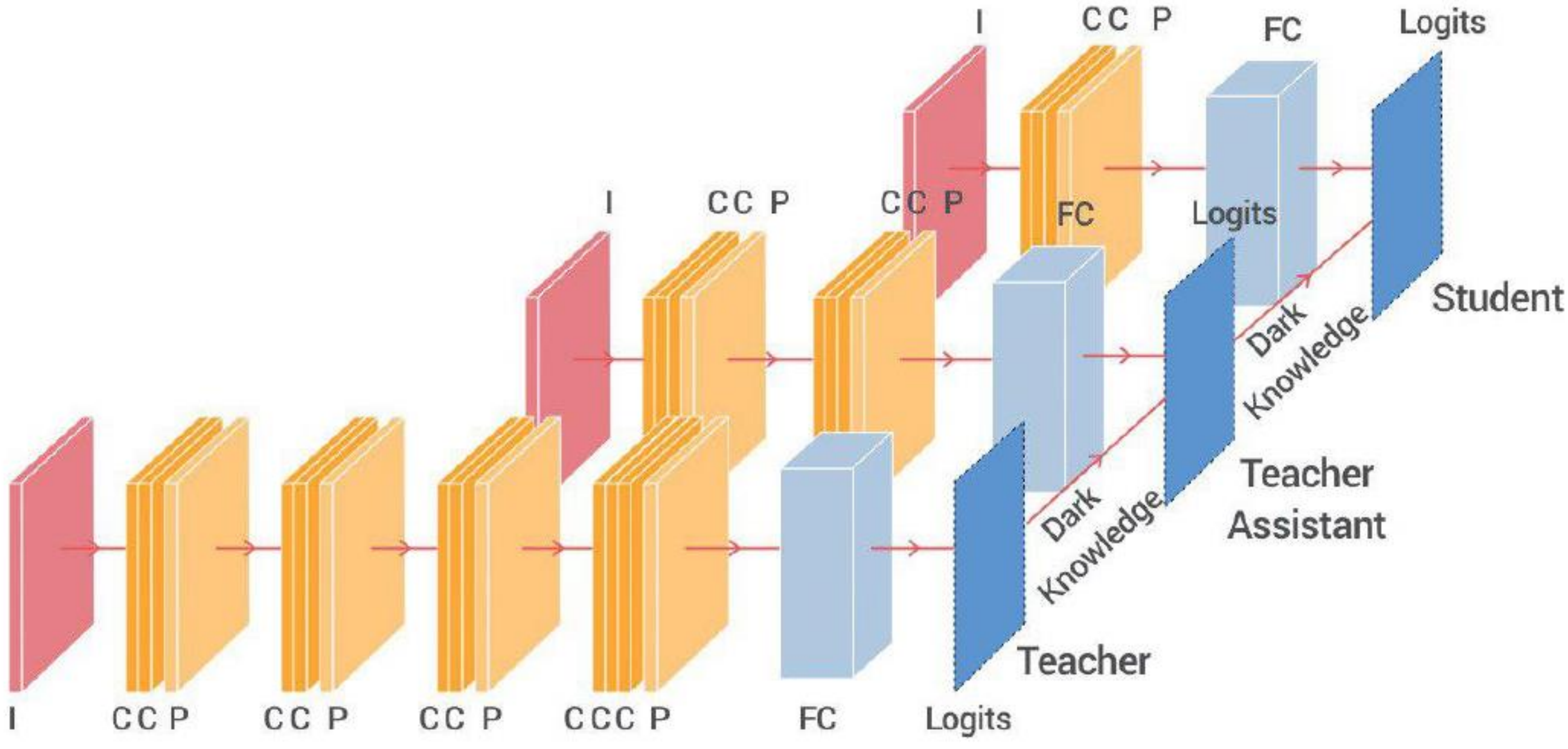
- Knowledge distillation **enables** the **smaller network to learn complex features**, that the teacher has already gone through the effort of extracting

- Knowledge distillation **transfers knowledge** to the smaller model by training it on a **transfer set that is obtained from the teacher network**, it can further be improved when the ground truth is known

Network Compression



Network Compression



Network Compression