Skoltech

Skolkovo Institute of Science and Technology

Speeding up and compression of Convolutional Neural Networks

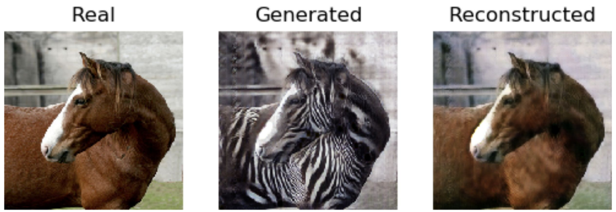
Konstantin Sobolev, Skoltech

Outline

1. Introduction
    a. Motivation
    b. Preliminaries
    c. CNN reminder
2. Efficient architecture design
    a. MobileNet
    b. ShuffleNet
    c. GhostNet
3. Model compression
    a. Weight decomposition
    b. Fine-grained pruning
    c. Structural pruning
    d. Quantization
    e. Knowledge Distillation

Motivation

Increasing importance and number of practical applications of CNN applications:



Source:

1) https://medium.com/@ismailou.sa
2) https://machinelearningmastery.com/cyclegan-tutorial-with-keras/
3) https://www.internetandtechnologylaw.com/bias-facial-recognition-flaws

Motivation

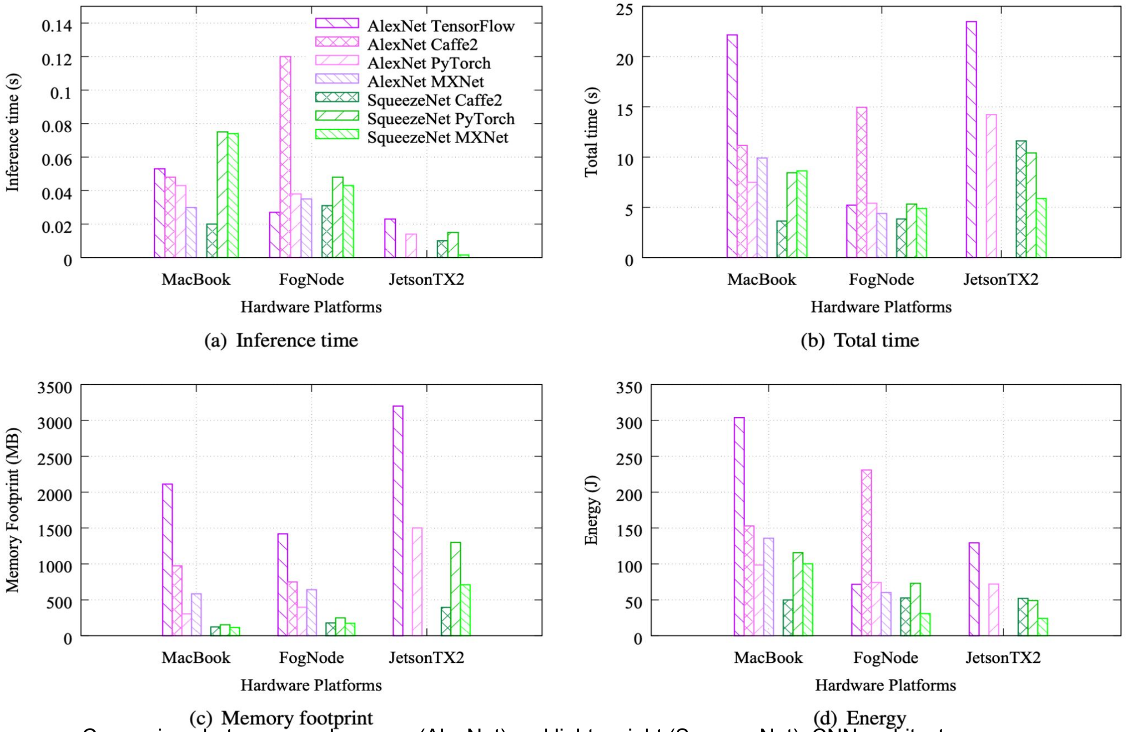
DL model limitations:

- High memory consumption
- Huge computational requirements
- Great power consumption

Difficult to deploy on portable devices

(e.g. laptops and smartphones)

Efficient architecture design is required



Comparison between cumbersome (AlexNet) and light-weight (SqueezeNet) CNN architectures on different edge platforms (MacBook, FogNode and JetsonTX2) and frameworks (TensorFlow, Caffe2, PyTorch and MXNet)

Source: https://www.usenix.org/system/files/conference/hotedge18/hotedge18-papers-zhang.pdf

Metrics to optimize

Real metrics:

- Inference time
- Memory consumption
- Battery consumption

Proxy metrics:

- **FLOP**s - number of computational operations required for inference
- **N** parameters - number of trained parameters

Efficient architecture design methods

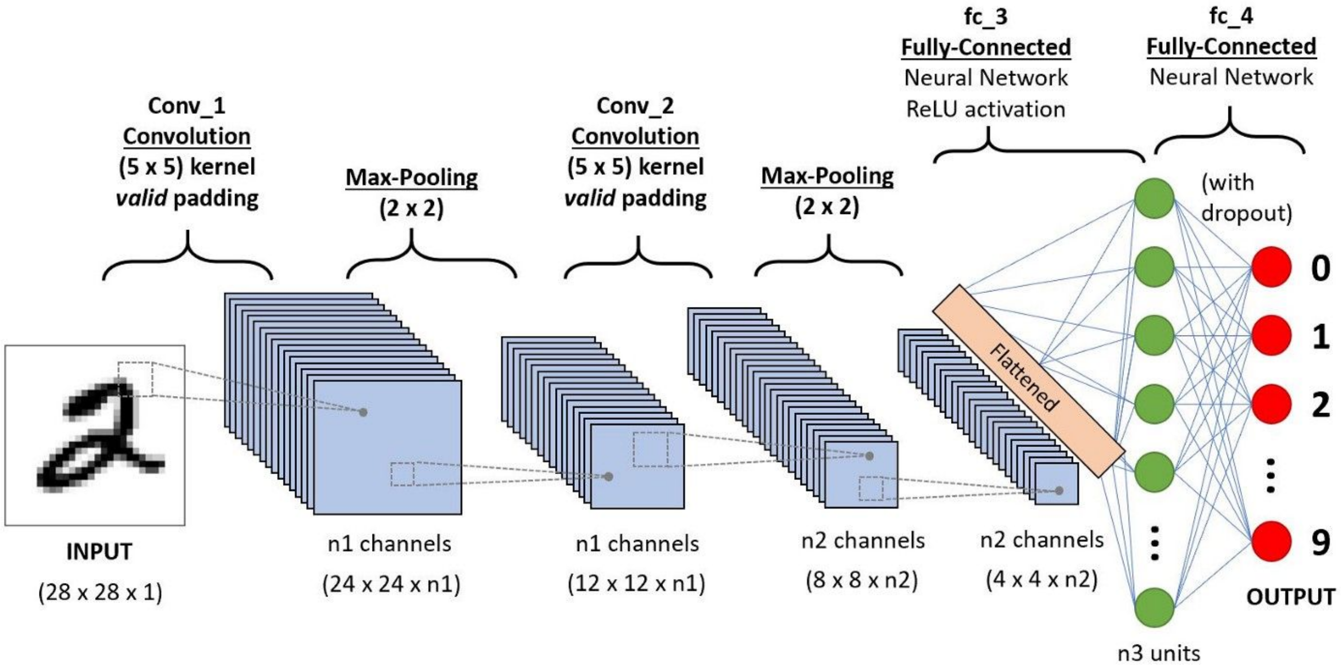
Efficient neural architecture design

- MobileNet
- ShuffleNet
- GhostNet

Neural network compression

- Weight Decomposition
- Unstructured Pruning
- Structured Pruning
- Quantization
- **Knowledge Distillation**

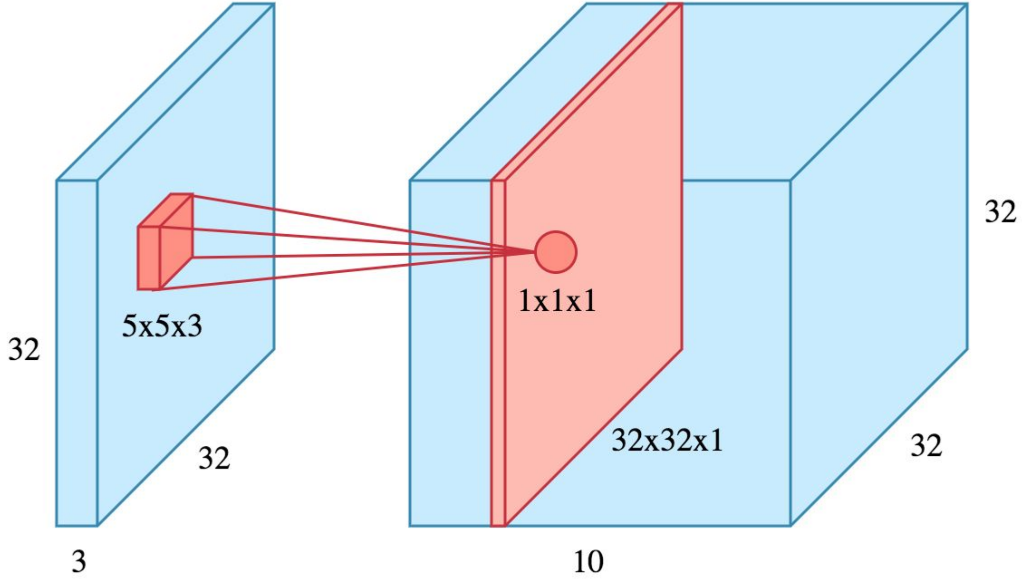
Convolutional Neural Network reminder



Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional Neural Network reminder

Most computations are concentrated in convolutional layer



Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Design efficient models from scratch

MobileNet

Idea: Replace ordinary convolution by depth-wise separable convolution

Original convolution complexity: $D_k * D_k * C_{in} * D_f * D_f * C_{out}$

Depth-wise separable convolution complexity: $(D_k * D_k + C_{out}) * C_{in} * D_f * D_f$

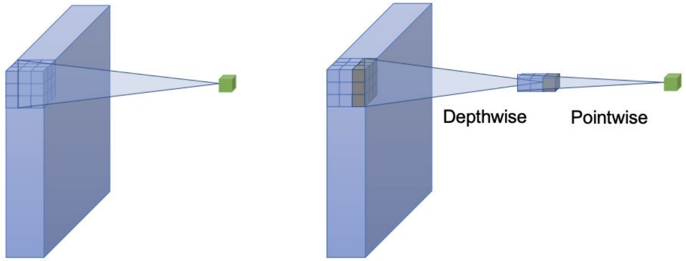
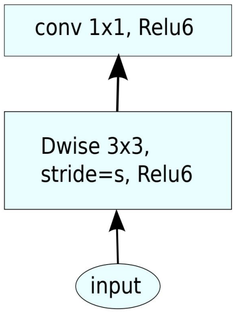
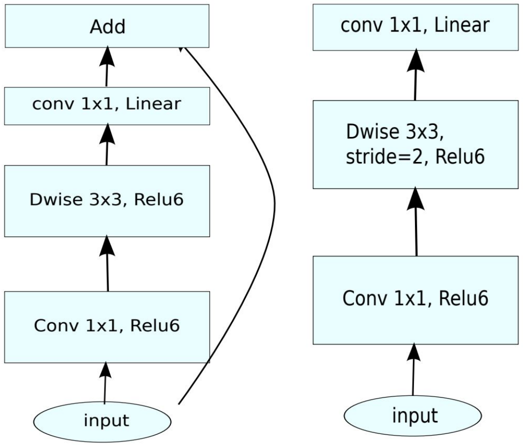


Figure 3: Standard convolution and depthwise separable convolution.



MobileNet block



Stride=1 block

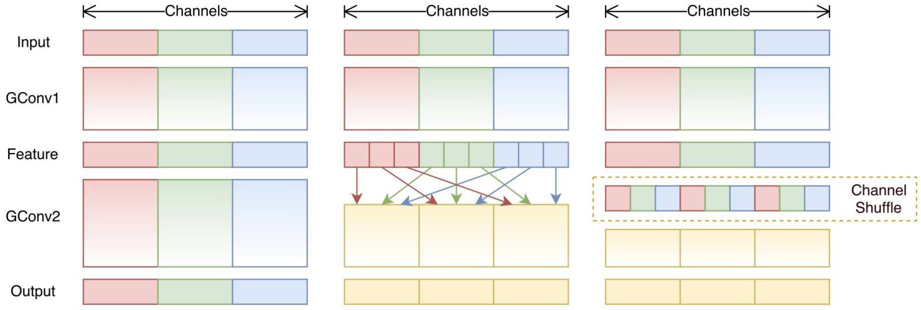
Stride=2 block

MobileNet v2 block

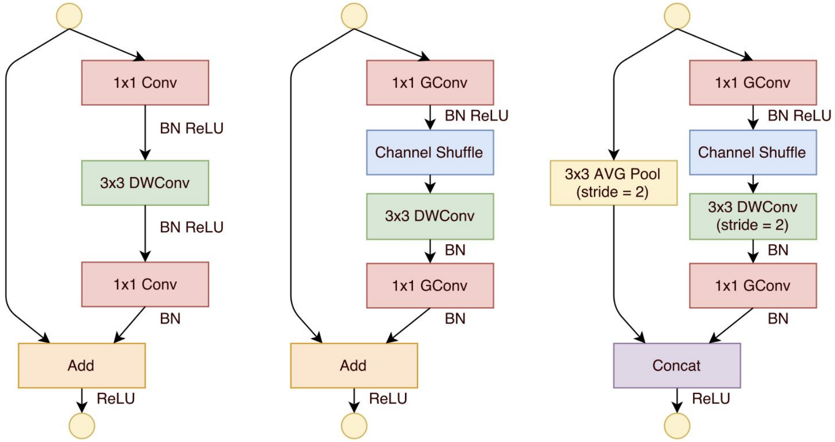
Source: https://arxiv.org/pdf/1704.04861.pdf
https://arxiv.org/pdf/1801.04381.pdf
https://arxiv.org/pdf/1905.02244.pdf
https://habr.com/ru/post/352804/

ShuffleNet

Idea: 1x1 Convolution is still an expensive operation. It can be accelerated by channel shuffling operation.



ShuffleNet operation visualization

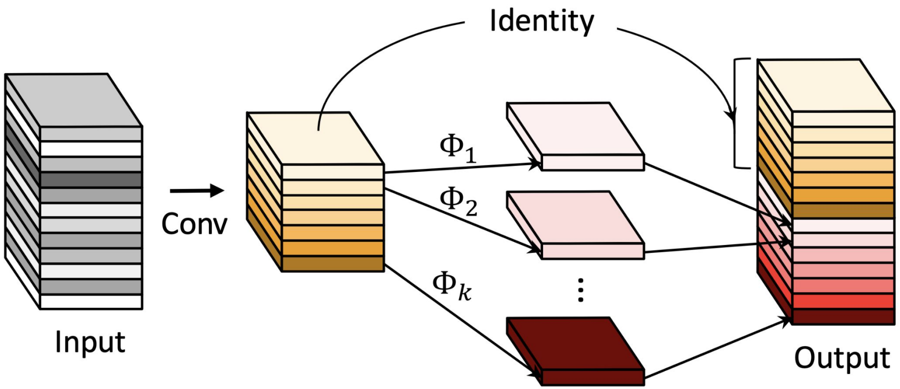


MobileNet block Shufflenet block ShuffleNet block

Source: https://arxiv.org/pdf/1707.01083.pdf

GhostNet

Idea: Generate redundancy in featuremap by cheap operations



(b) The Ghost module.

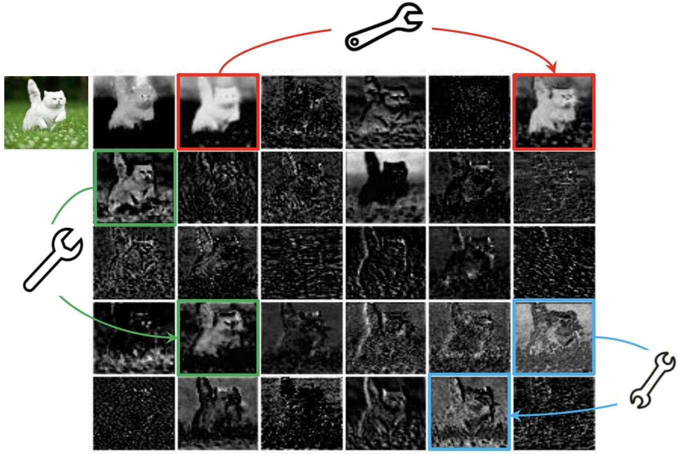


Figure 1. Visualization of some feature maps generated by the first residual group in ResNet-50, where three similar feature map pair examples are annotated with boxes of the same color. One feature map in the pair can be approximately obtained by transforming the other one through cheap operations (denoted by spanners).

Source: https://arxiv.org/pdf/1911.11907.pdf

Results

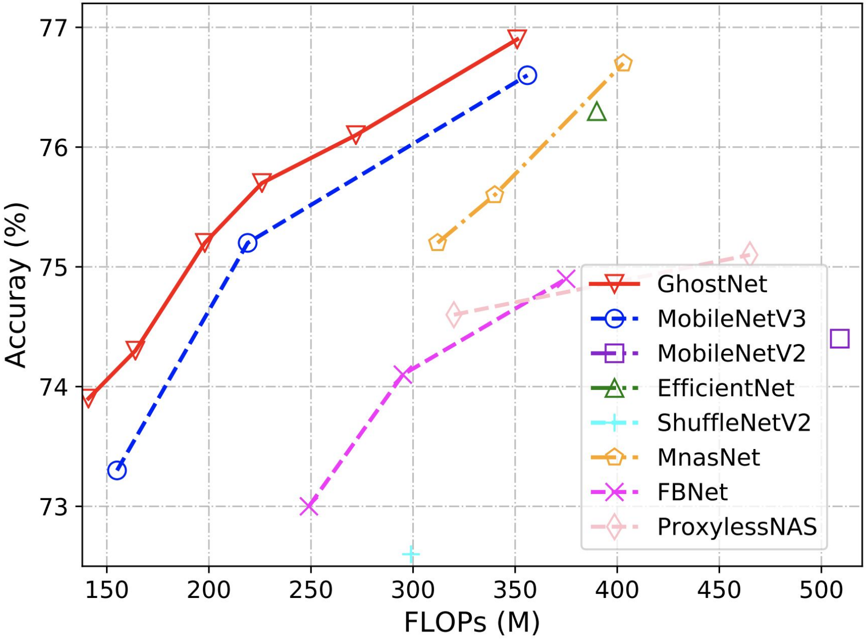


Figure 6. Top-1 accuracy *v.s.* FLOPs on ImageNet dataset.

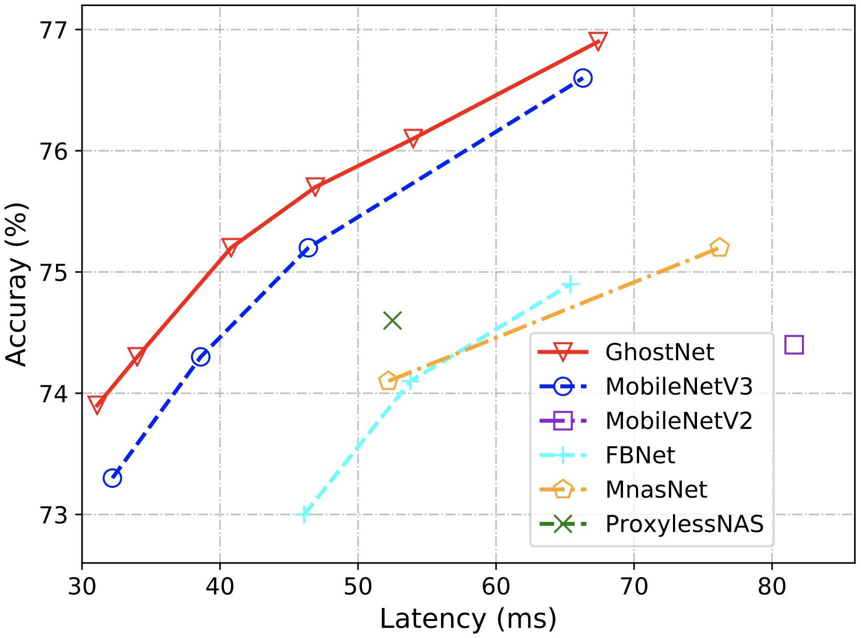


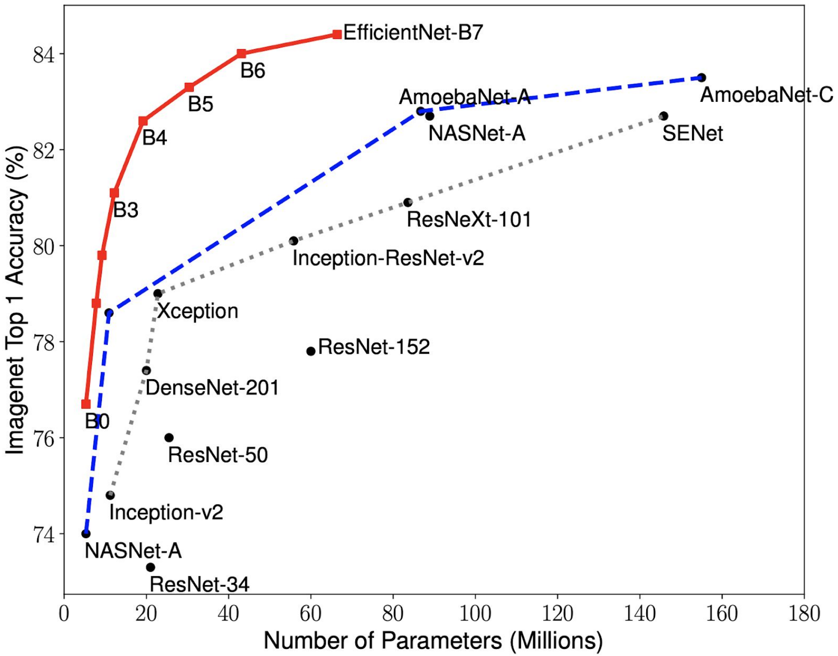
Figure 7. Top-1 accuracy *v.s.* latency on ImageNet dataset.

Source: https://arxiv.org/pdf/1911.11907.pdf

Model compression

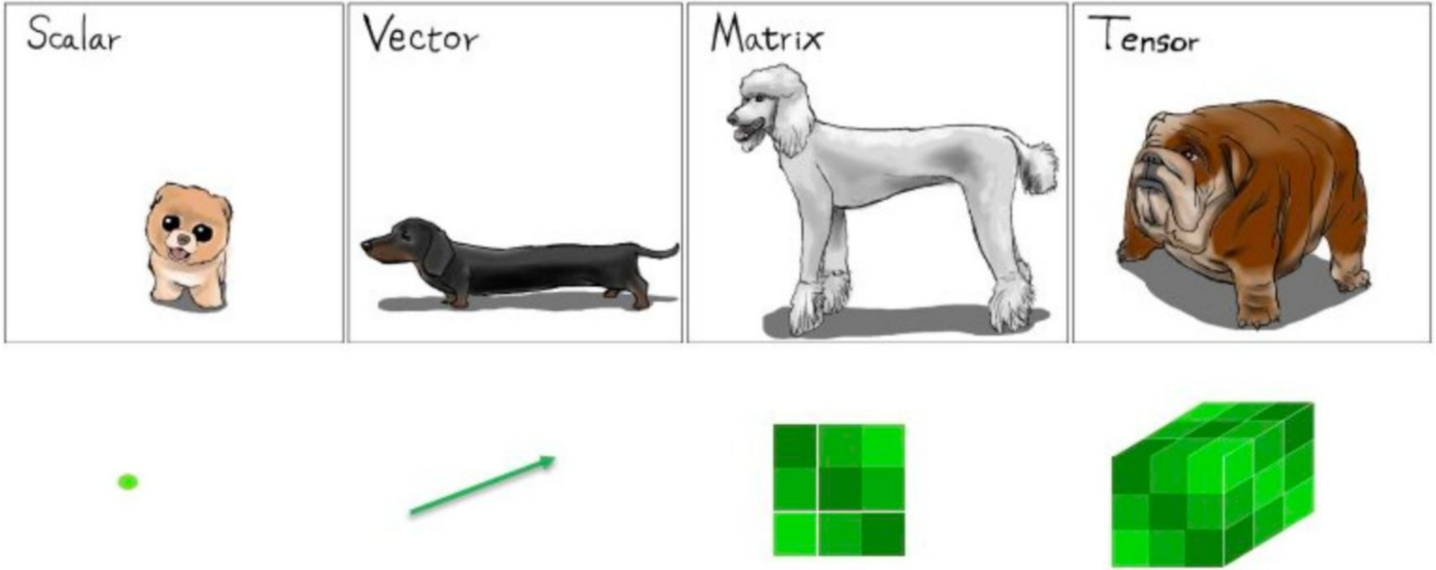
Model compression

- Most neural networks are redundant
- Redundancy helps to achieve a better performance
- Redundancy helps during training but harms during inference



Source: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Weight Decomposition: Preliminaries



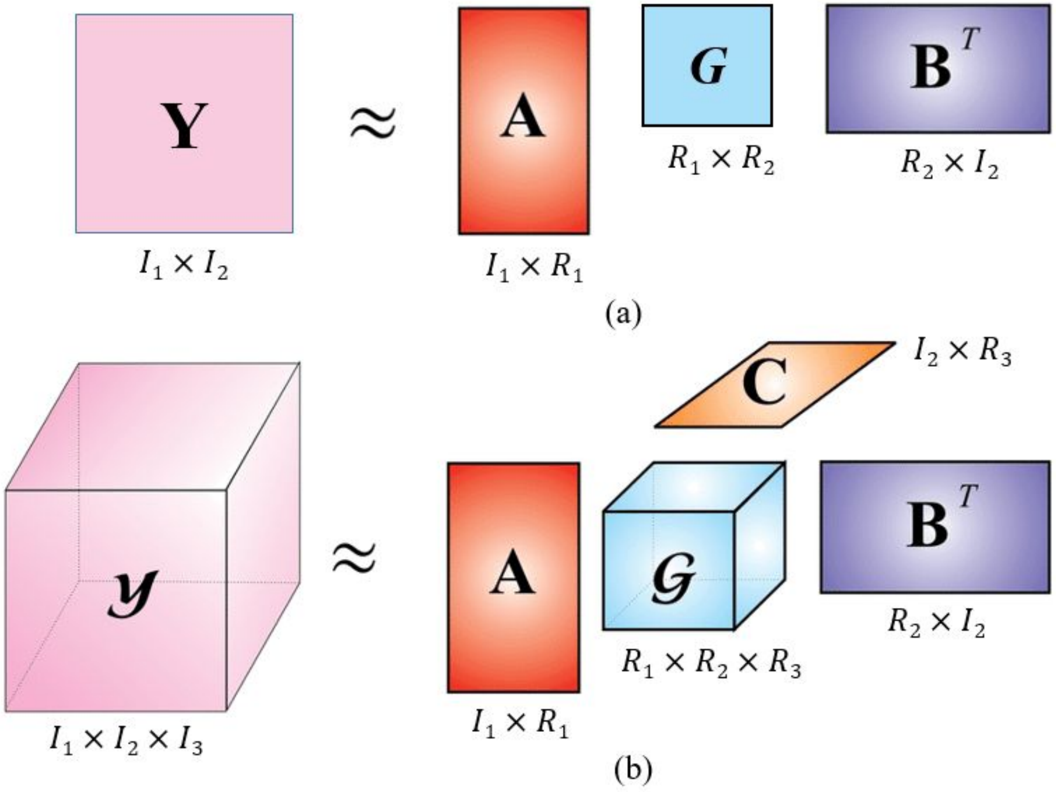
Scalar-level 1960's → Matrix-level 1980's → Block-Matrix level 2000's → Tensor level Now

Level of thinking in algebra field

Weight Decomposition: Preliminaries

Matrix decomposition:

Tensor decomposition:

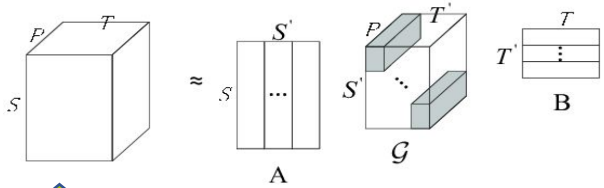


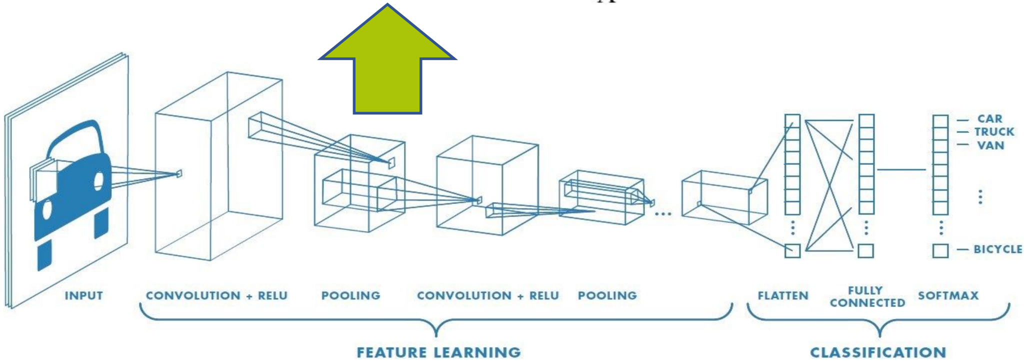
Source: https://www.researchgate.net/publication/359367892_electronics-11-009451/figures?lo=1&utm_source=google&utm_medium=organic

Weight Decomposition

Pipeline:

1. Extract convolutional kernel
2. Decompose it into factors: A, G, B
3. Replace initial layer by sequence of layers with factors as kernels
4. Fine-tune network



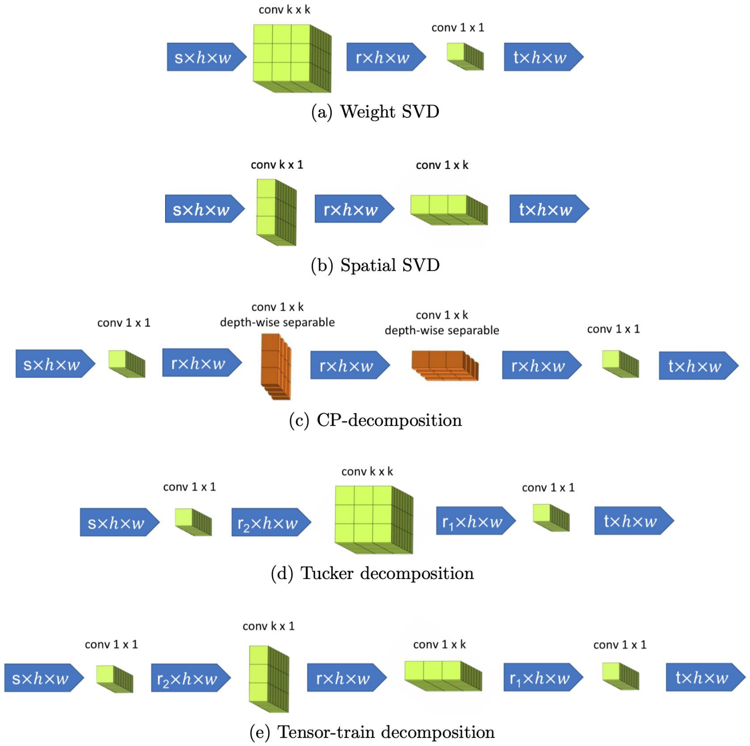


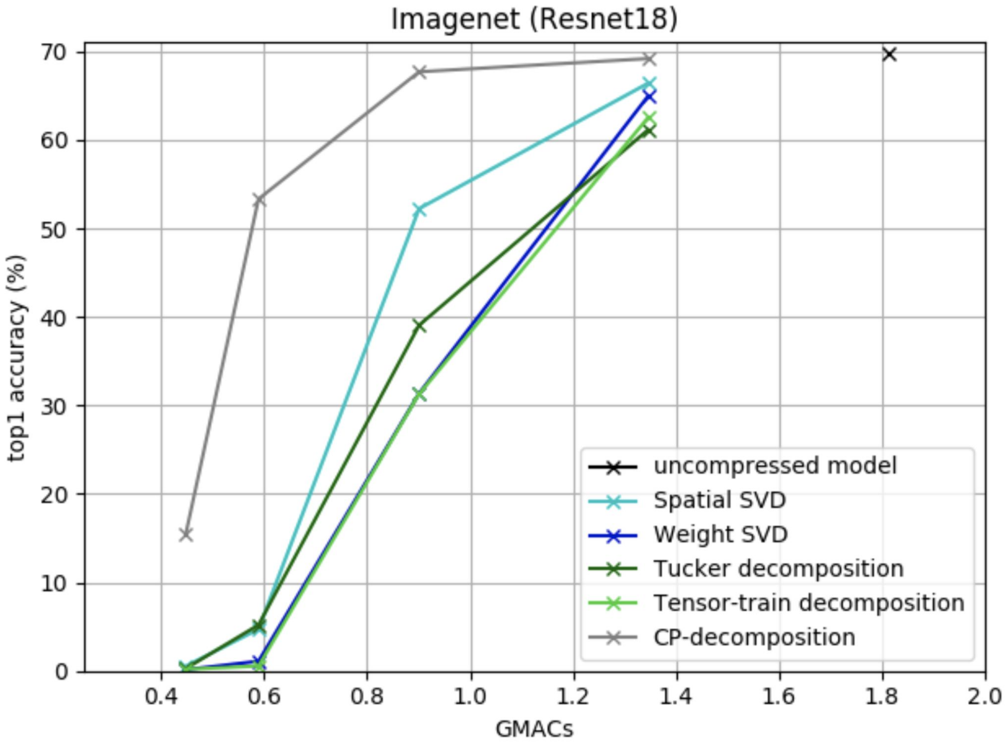
Result:

1. Faster inference
2. Lower memory consumption
3. Longer battery life

Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Weight Decomposition





Source: https://arxiv.org/pdf/1912.09802.pdf

Weight Decomposition

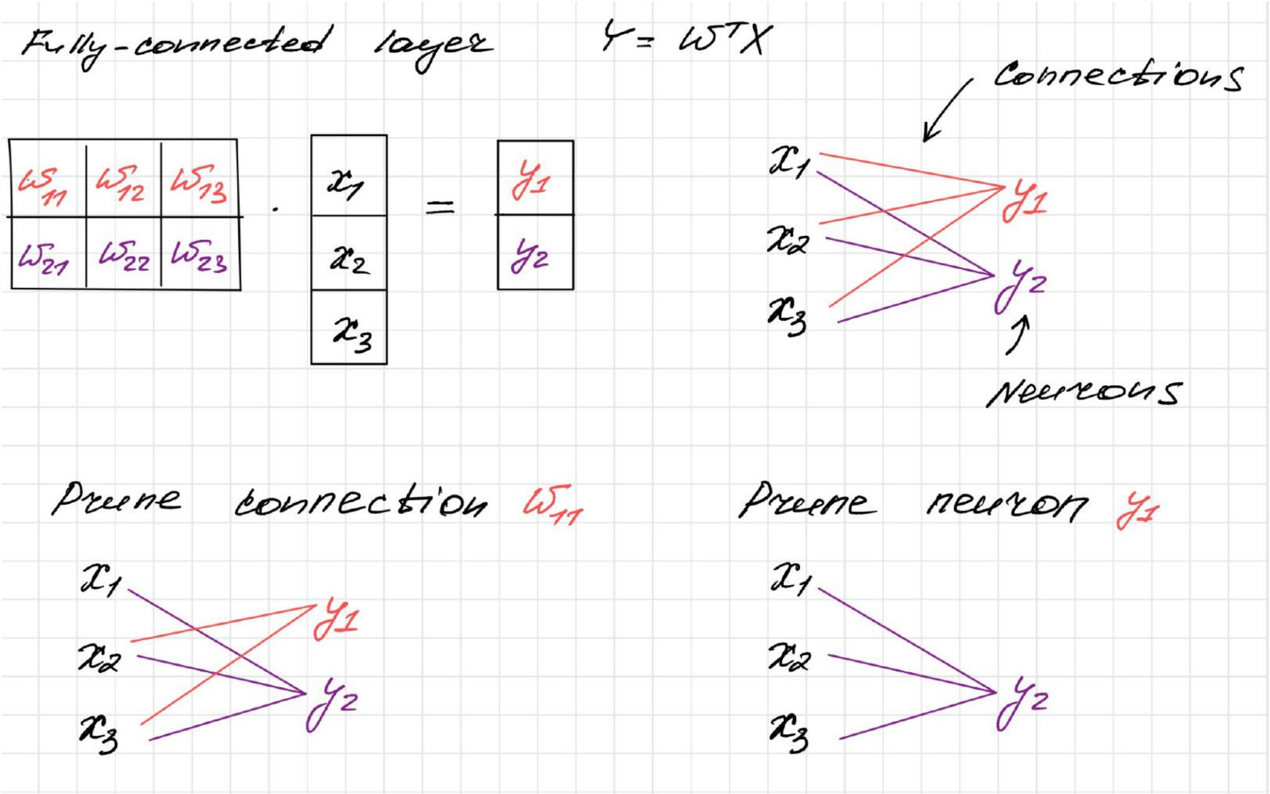
Pros:

- Achieves high theoretical speedup with low performance degradation
- Does not require additional hardware support

Cons:

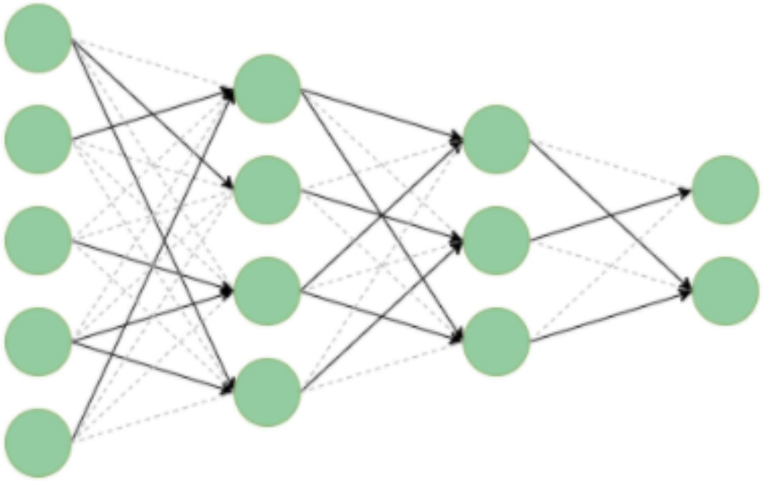
- Poorly works with non-standard convolutions (PW, DW, Group)
- Requires careful rank selection

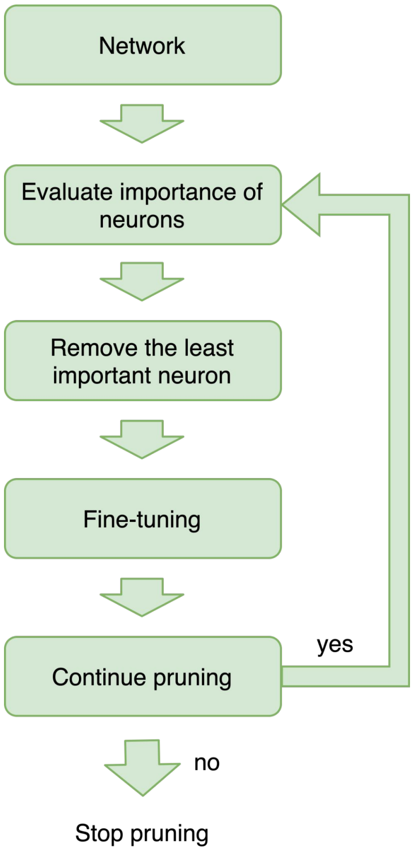
Pruning: Preliminaries



Unstructured Pruning

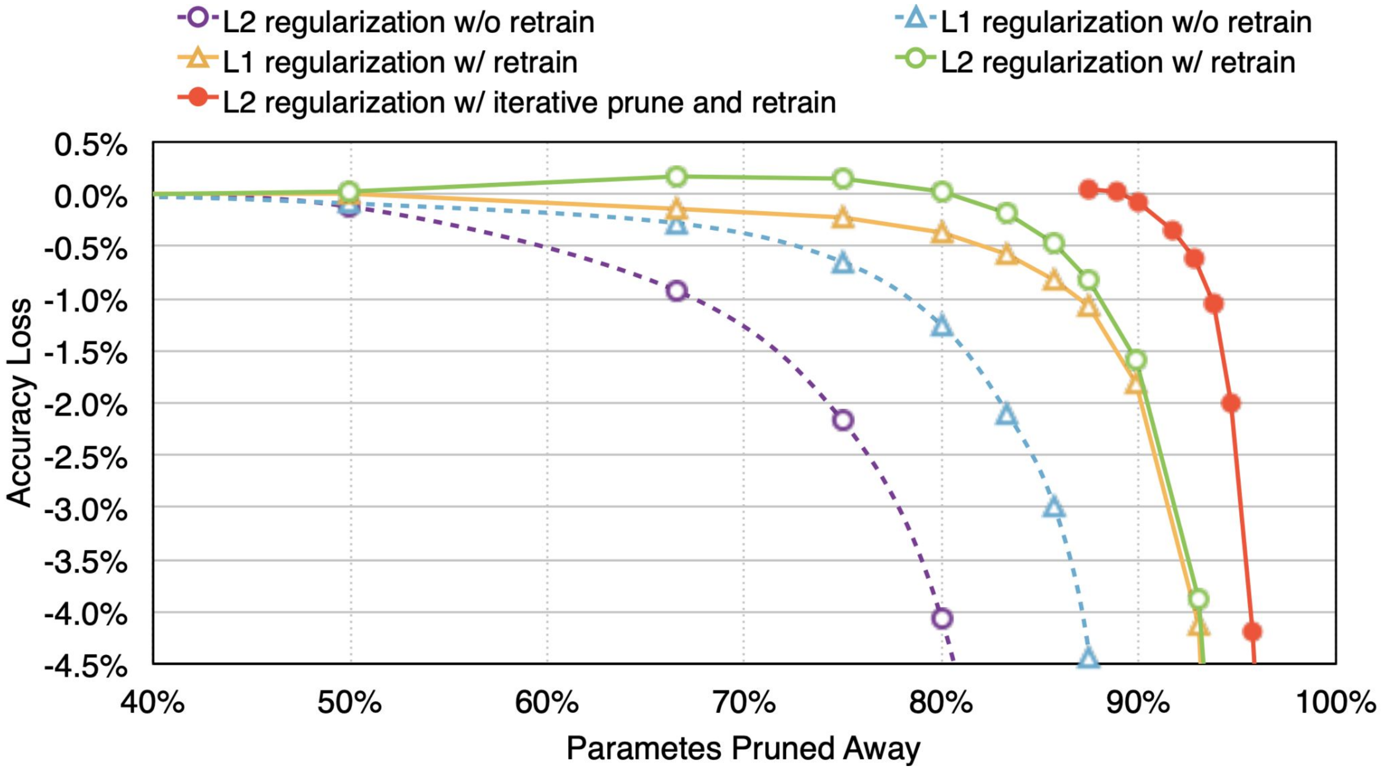
aka Fine-Grained Pruning aka Weight Sparsification





Source: https://arxiv.org/pdf/1506.02626.pdf

Unstructured Pruning

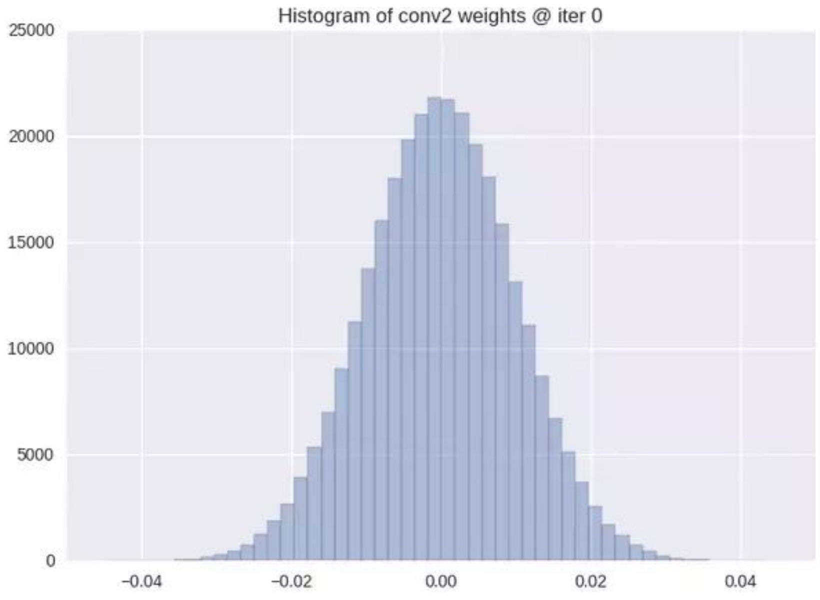


Source: https://arxiv.org/pdf/1506.02626.pdf

Unstructured Pruning

List of possible criterions:

- Weight-based criteria (L1/L2 norm)
- Gradient-based criteria

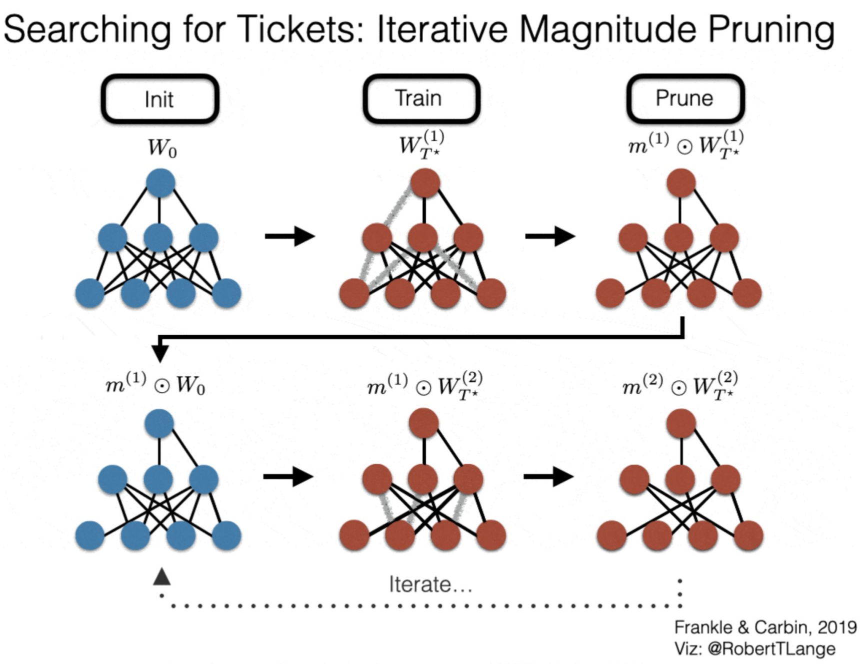


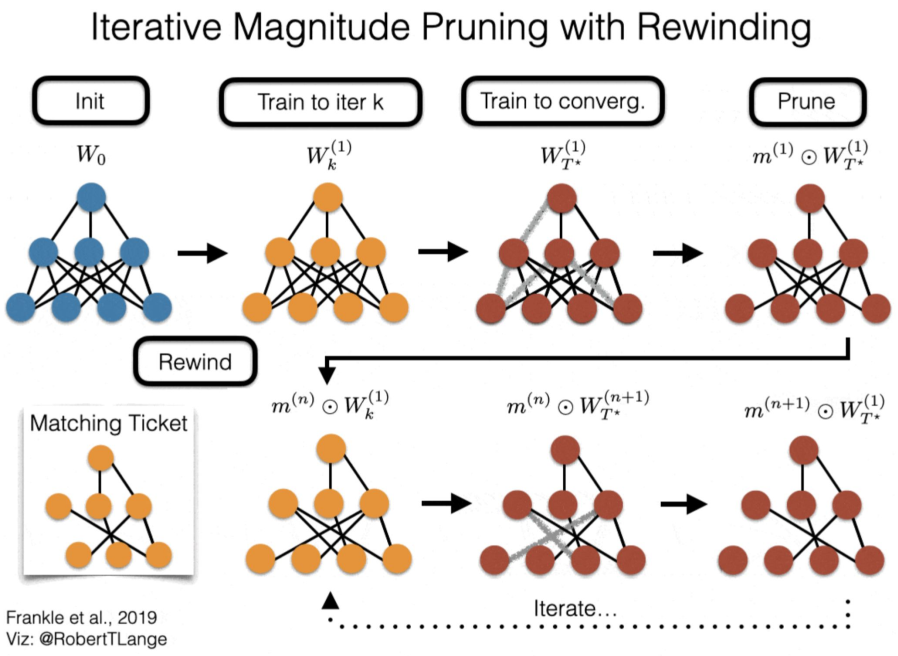
$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(||\delta U||^3)$$

$$g_i = \frac{\partial E}{\partial u_i} \quad \text{and} \quad h_{ij} = \frac{\partial^2 E}{\partial u_i \partial u_j}$$

Source: https://openaccess.thecvf.com/content_CVPR_2020/papers/He_Learning_Filter_Pruning_Criteria_for_Deep_Convolutional_Neural_Networks_Acceleration_CVPR_2020_paper.pdf

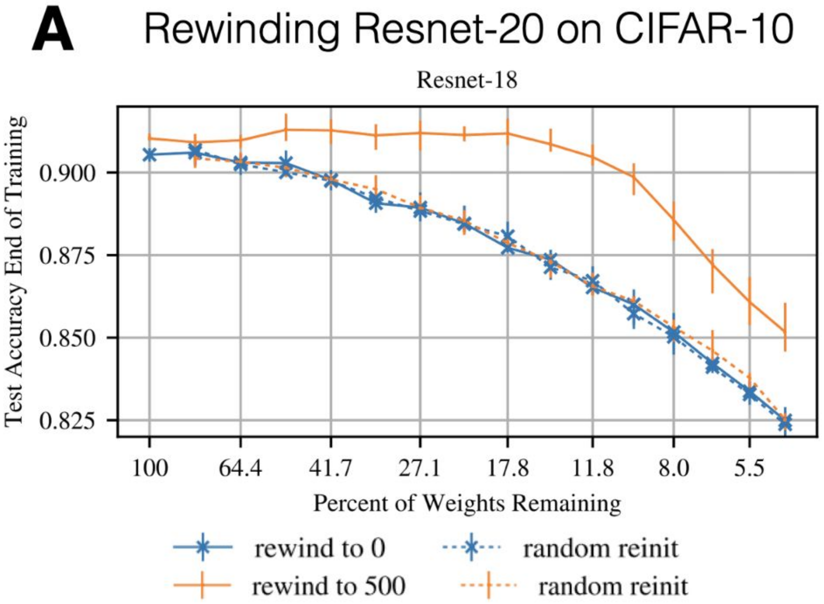
Unstructured Pruning: Lottery Ticket Hypothesis

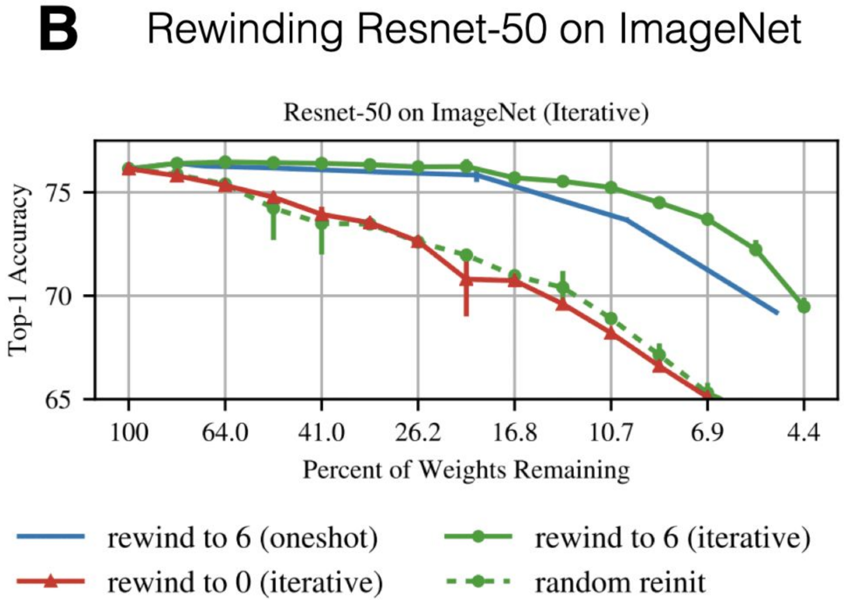




Source: https://towardsdatascience.com/the-lottery-ticket-hypothesis-a-survey-d1f0f62f8884
https://arxiv.org/pdf/1803.03635.pdf
https://arxiv.org/pdf/1903.01611.pdf

Unstructured Pruning: Lottery Ticket Hypothesis





Source: https://towardsdatascience.com/the-lottery-ticket-hypothesis-a-survey-d1f0f62f8884

Unstructured Pruning

Pros:

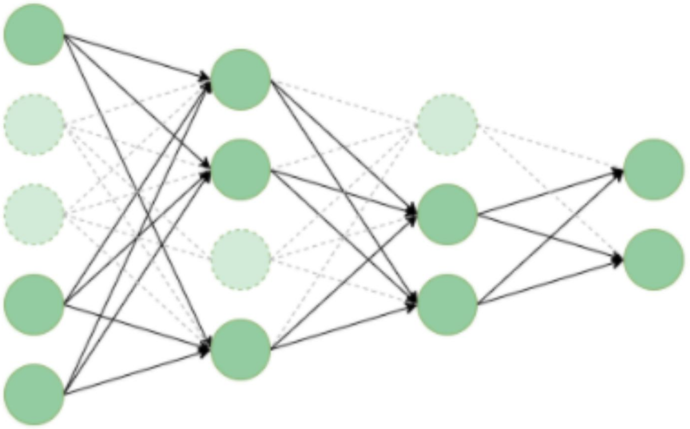
- Achieves high rates of weight reduction without acc. drop (~ 95% of weights removed)

Cons:

- Requires hardware support for sparse computation speedup
- Hard to find sparsity level for all layers of the network

Structured Pruning

Removing structural parts instead of individual weights



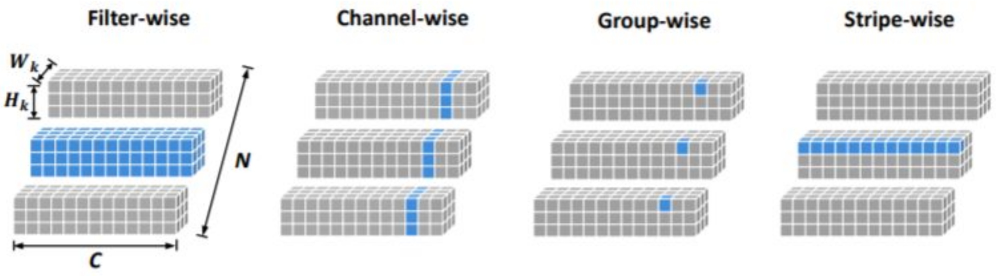


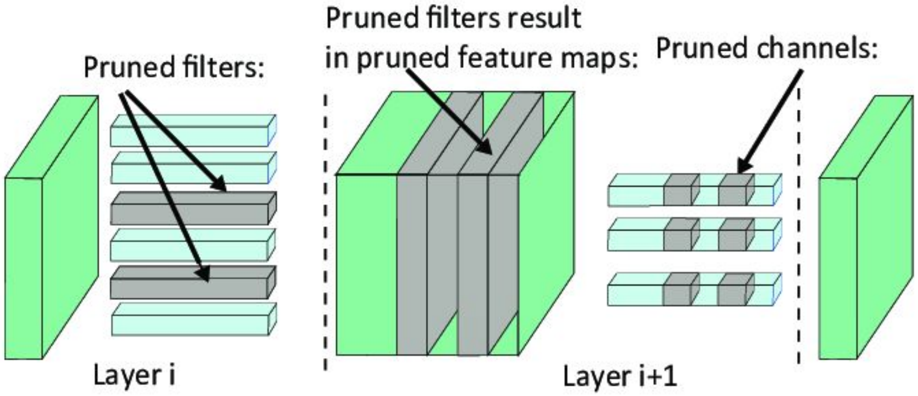
Figure 2: The visualization of different types of pruning.

Source: https://heartbeat.fritz.ai/neural-network-pruning-research-review-2020-bc21a77f0295

Structured Pruning

List of possible criterions:

- Weight-based criteria:
  - L1/L2 norm
  - Scaling parameter in BN
- Activation-based criteria:
  - PCA of activations
- Gradient-based criteria
- Greedy and One-shot Pruning



Source: https://openaccess.thecvf.com/content_CVPR_2020/papers/He_Learning_Filter_Pruning_Criteria_for_Deep_Convolutional_Neural_Networks_Acceleration_CVPR_2020_paper.pdf

Structured Pruning

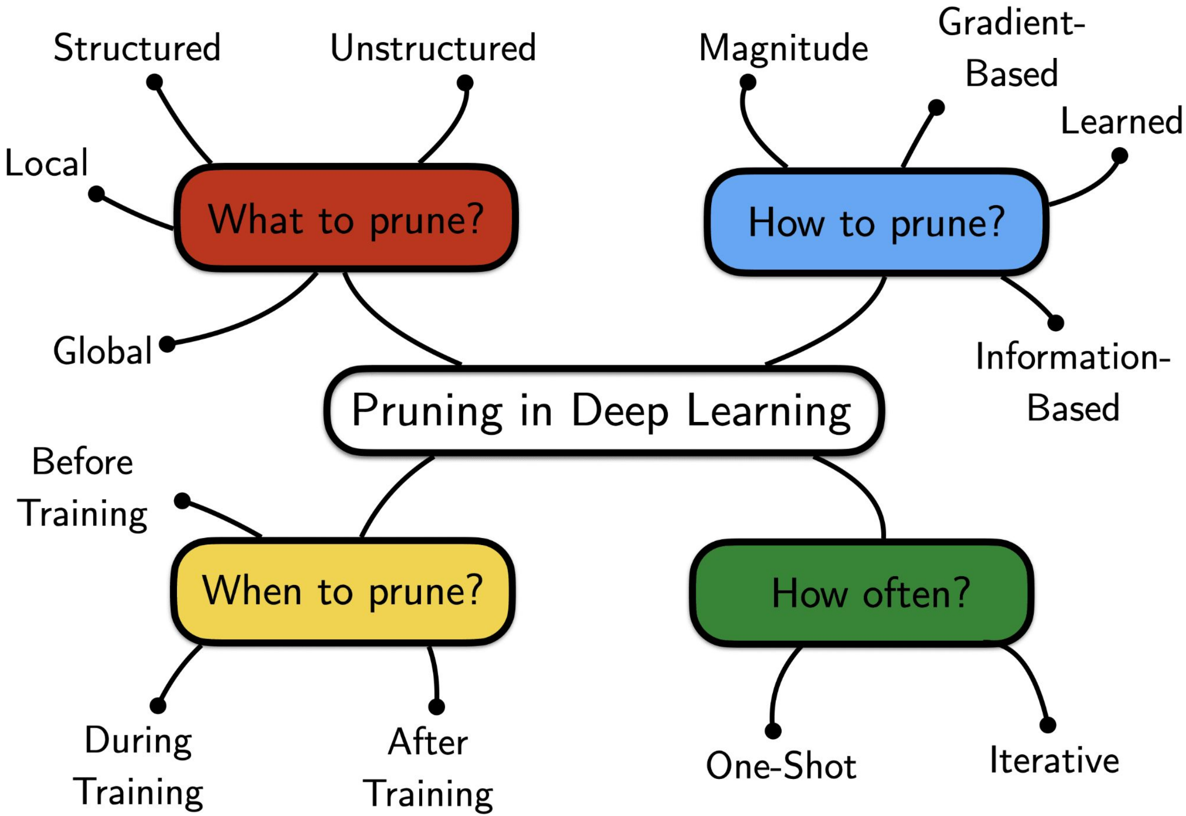
Pros:

- Efficiently accelerates model
- No need of hardware/software support (pruned model is structurally equivalent to initial model)

Cons:

- Pruning channels/filters in one layer affects previous/subsequent layers
- Hard to find best filter configuration for the whole network

Pruning

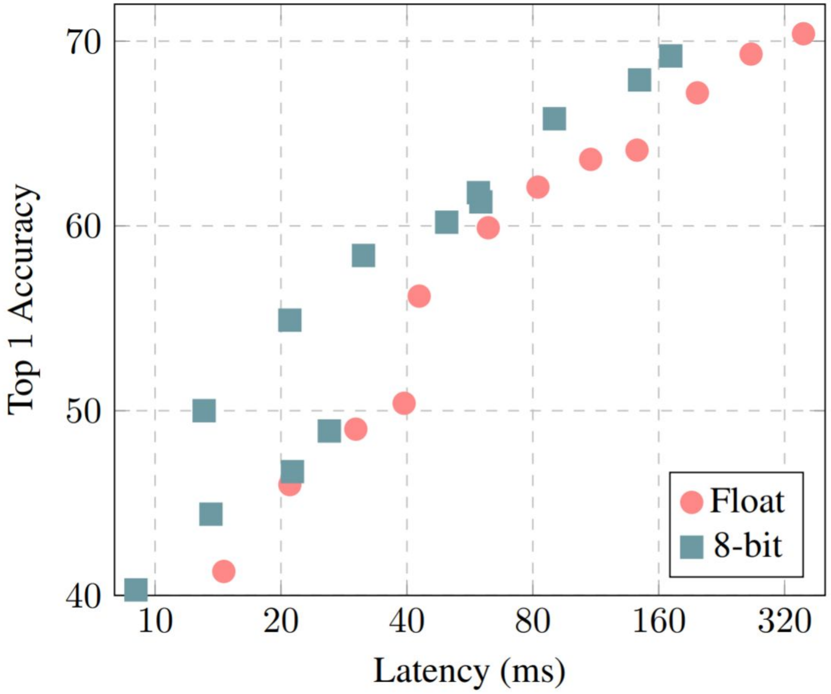


Pruning in Deep Learning
What to prune?
Structured
Unstructured
Local
Global
How to prune?
Magnitude
Gradient-Based
Learned
Information-Based
When to prune?
Before Training
During Training
After Training
How often?
One-Shot
Iterative

Quantization

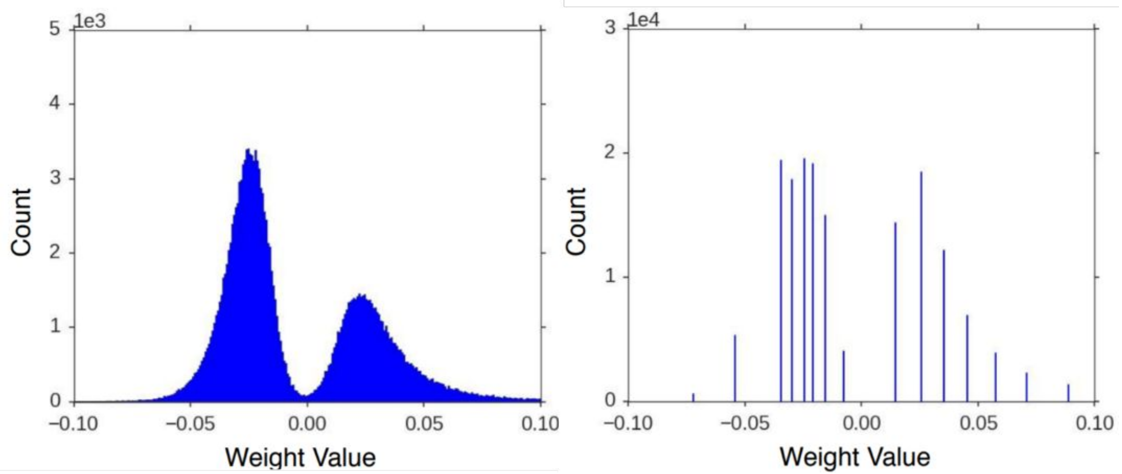
Reduce redundancy in weight numerical representation

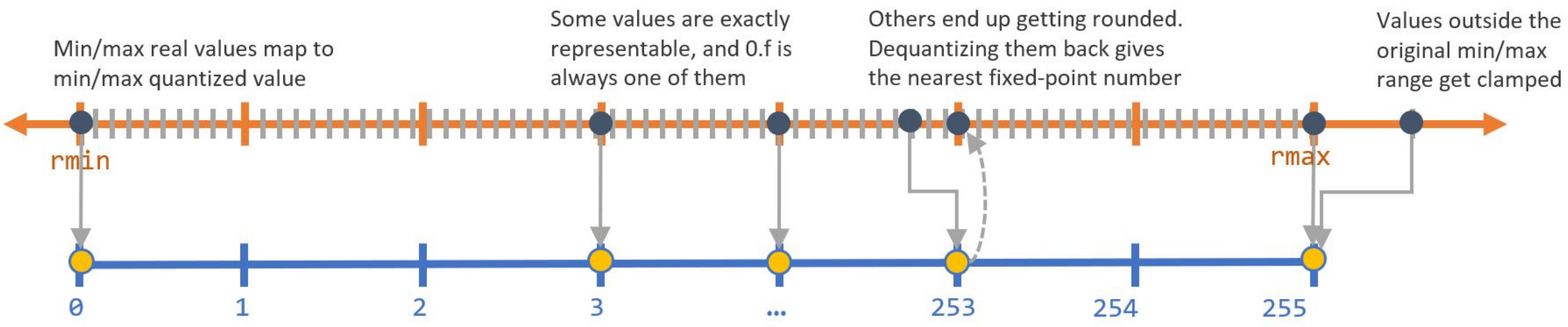
- DNNs are known to be quite robust to noise and other small perturbations
- Weights and activations by a particular layer often tend to lie in a small range
- Arithmetic with lower bit-depth is faster
- Reduces memory consumption (e.g. in moving from 32-bits to 8-bits, we get (almost) 4x reduction in memory)



Source: https://heartbeat.fritz.ai/8-bit-quantization-and-tensorflow-lite-speeding-up-mobile-inference-with-low-precision-a882dfcafbbd
https://arxiv.org/pdf/1712.05877.pdf

Quantization





Source: https://heartbeat.fritz.ai/8-bit-quantization-and-tensorflow-lite-speeding-up-mobile-inference-with-low-precision-a882dfcafbbd
https://arxiv.org/pdf/1510.00149.pdf

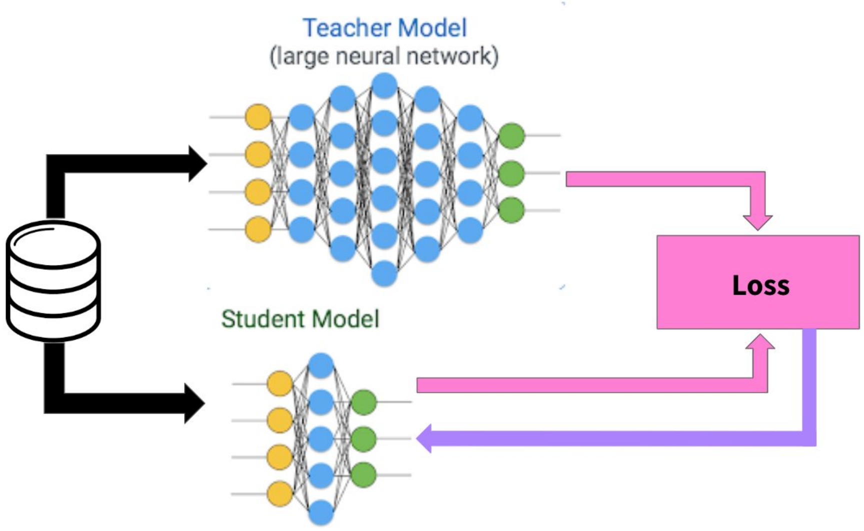
Quantization

Pros:

- Easy combine with other methods (low-rank, pruning, etc.)
- Easy to apply (supported in modern NN frameworks: pytorch, tensorflow)
- Efficient model acceleration and compression
- Few quantization options => easy to find the best

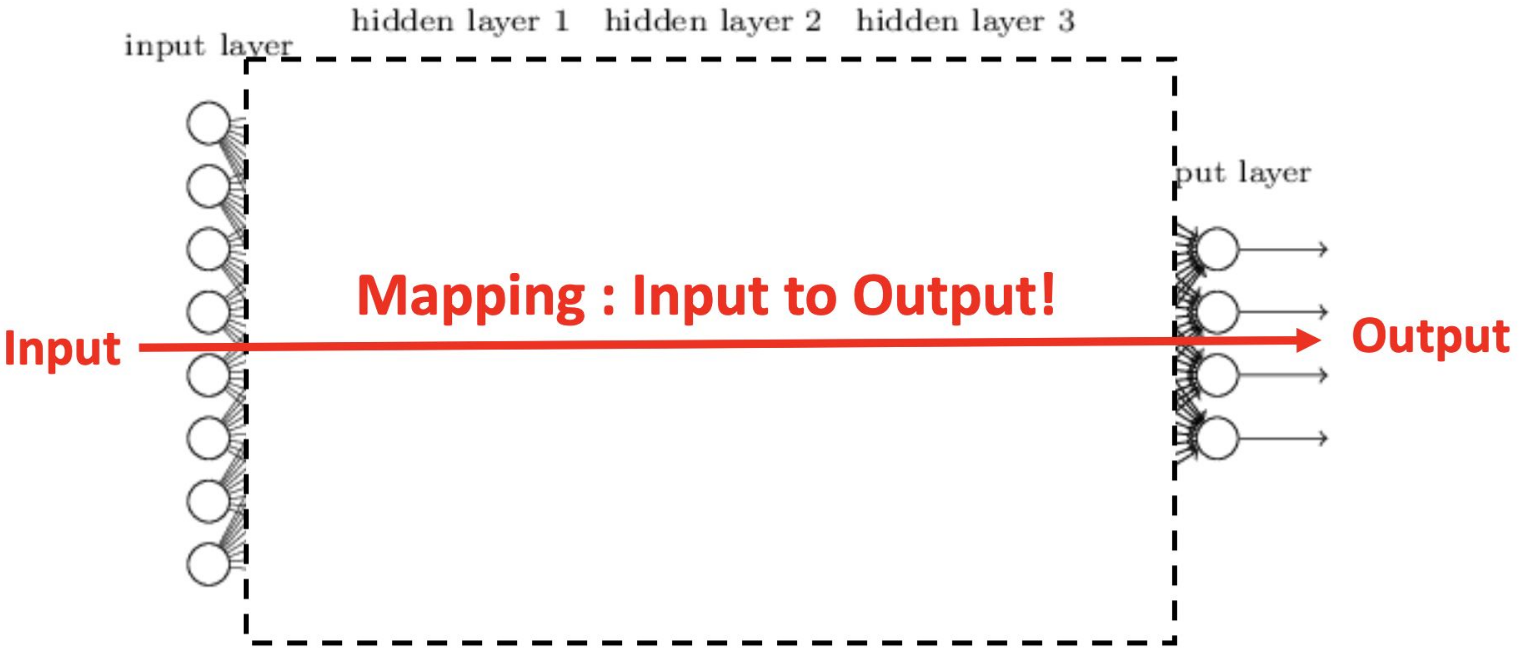
Knowledge Distillation

Knowledge distillation is a process of distilling or transferring the knowledge from a (set of) large, cumbersome model(s) to a lighter, easier-to-deploy single model



Credits: http://ir.hit.edu.cn/~xiachongfeng/slides/Knowledge%20Distillation.pdf
https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Knowledge Distillation

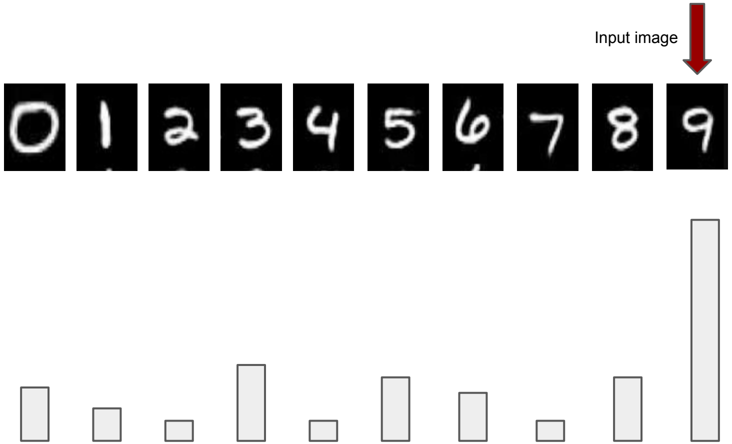


A more abstract view of the **knowledge**, that frees it from any particular instantiation, is that it is a **learned mapping from input vectors to output vectors**.

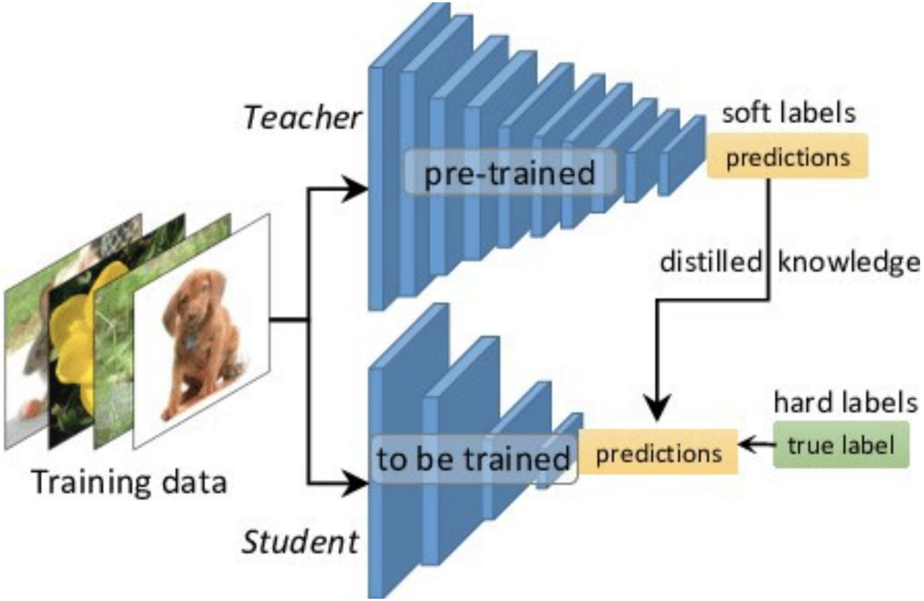
Credits: http://ir.hit.edu.cn/~xiachongfeng/slides/Knowledge%20Distillation.pdf

Knowledge Distillation

Idea: Train less redundant model by using knowledge from big models



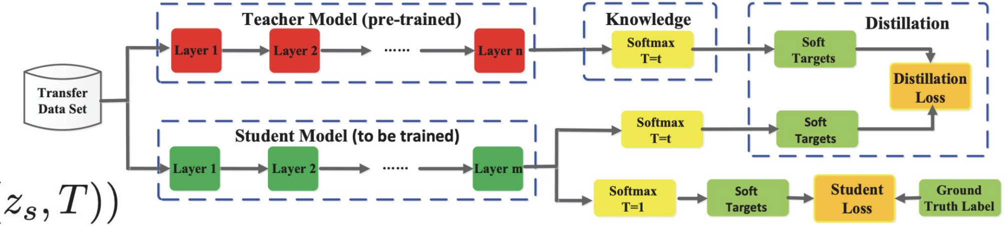
Output probability distribution



Source: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Knowledge Distillation

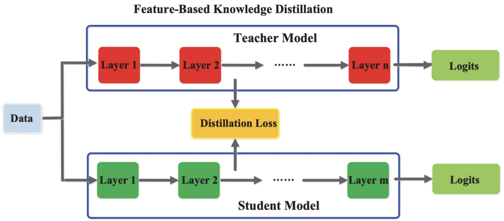
- Response-Based Knowledge



$$L_{ResD}(p(z_t,T),p(z_s,T)) = \mathcal{L}_R(p(z_t,T),p(z_s,T))$$

$$p(z_i,T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

- Feature-Based Knowledge

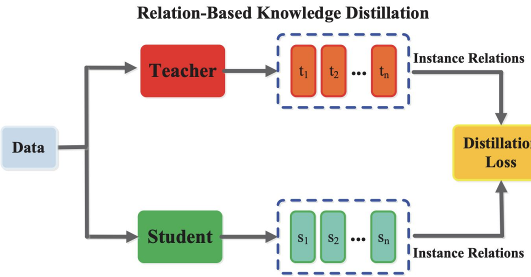


$$L_{FeaD}\big(f_t(x), f_s(x)\big) = \mathcal{L}_F\big(\Phi_t(f_t(x)), \Phi_s(f_s(x))\big)$$

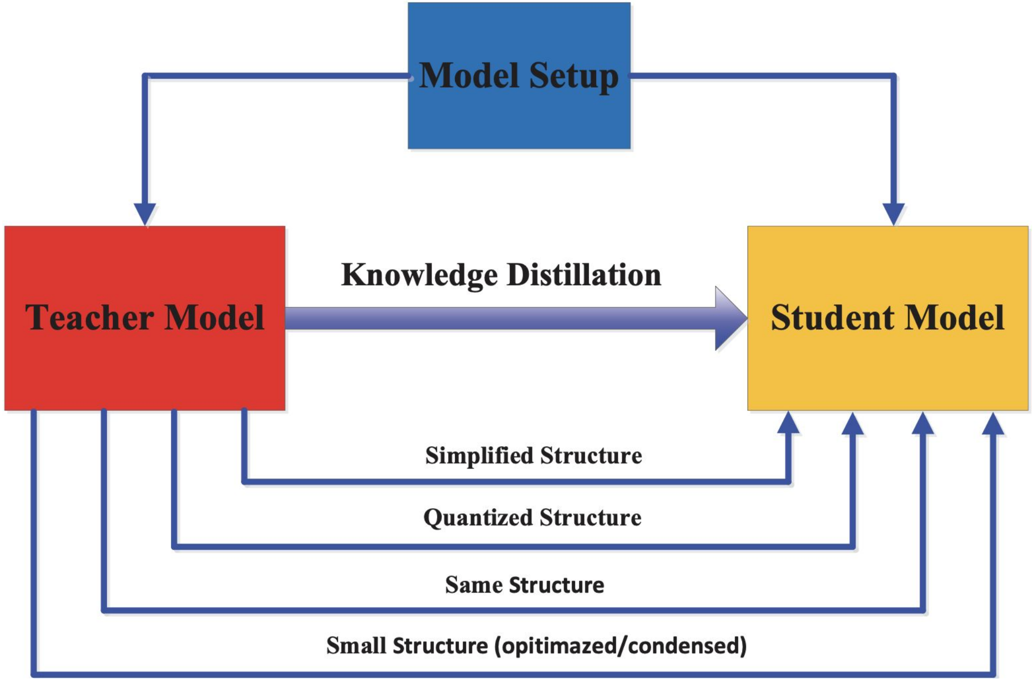
- Relation-Based Knowledge

$$L_{RelD}(f_t, f_s) = \mathcal{L}_{R^1}\big(\Psi_t(\hat{f}_t, \check{f}_t), \Psi_s(\hat{f}_s, \check{f}_s)\big)$$

$$L_{RelD}(F_t, F_s) = \mathcal{L}_{R^2}\big(\psi_t(t_i, t_j), \psi_s(s_i, s_j)\big)$$



Knowledge Distillation



Knowledge Distillation

Pros:

- Improves model performance

Cons:

- Capacity gap
- Controversy of the method

Summary

Efficient neural architecture design

- MobileNet
- ShuffleNet
- GhostNet

Neural network compression

- Weight Decomposition
- Unstructured Pruning
- Structured Pruning
- Quantization
- **Knowledge Distillation**

Thanks for your attention!