Network Compression

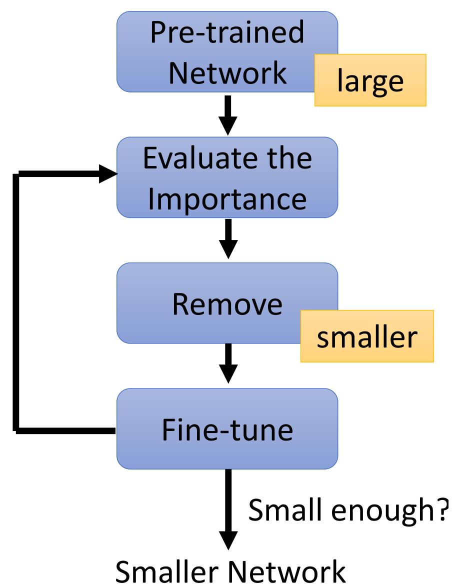
Less parameters.
Purpose: deploying ML models in resource- constrained environments.
Latency issue. Circumstances might need immediate results e.g. self driving.
Privacy.

Outline

Network Pruning
Knowledge Distillation
Parameter Quantization • Architecture Design
Dynamic Computation
We will not talk about hard-ware solution today.

Network Pruning

Networks are typically over-parameterized (there is significant redundant weights or neurons), and we want to prune them.



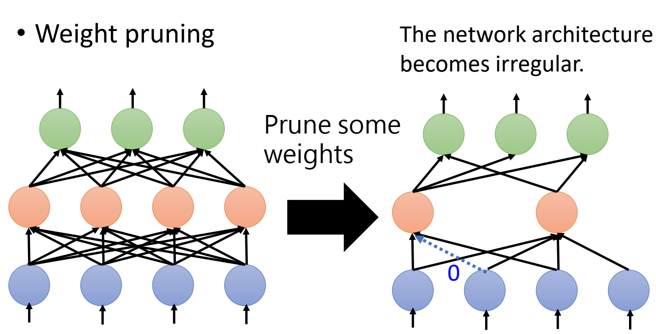
Parameters or Neuron as basic pruning unit

Evaluate the importance of a parameter or neuron:
Importance of a weight: absolute values, method from life long learning v_i.
Importance of a neuron:m the number of times it wasn't zero on a given data set ......

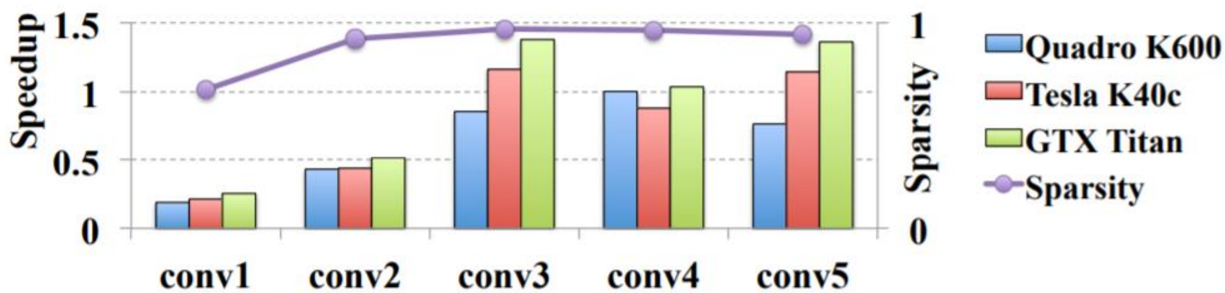
After pruning, **the accuracy will drop** (hopefully not too much)
To save the accuracy, we need to fine-tuning on training data for recover.
Re-evaluate the parameters and reprune the model.
Don't prune too much at once, or the network won't recover.

Pruning by weights (parameters):

The network architecture becomes irregular.



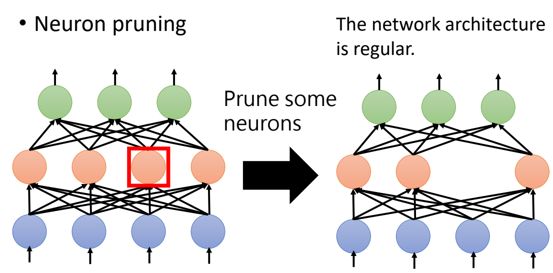
Hard to implement and hard to speed up. In implement, let the parameters be 0 but the size of the model does not shrink too much.



Speed becomes slower after weight pruning.

Network Pruning

Easy to implement and easy to speed up.



How about simply train a smaller network?

It is widely known that smaller network is more difficult to learn successfully.

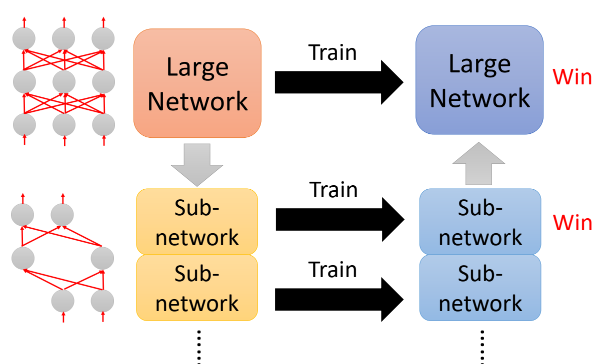
Larger network is easier to optimize? There is a hypothesis: Lottery Ticket Hypothesis

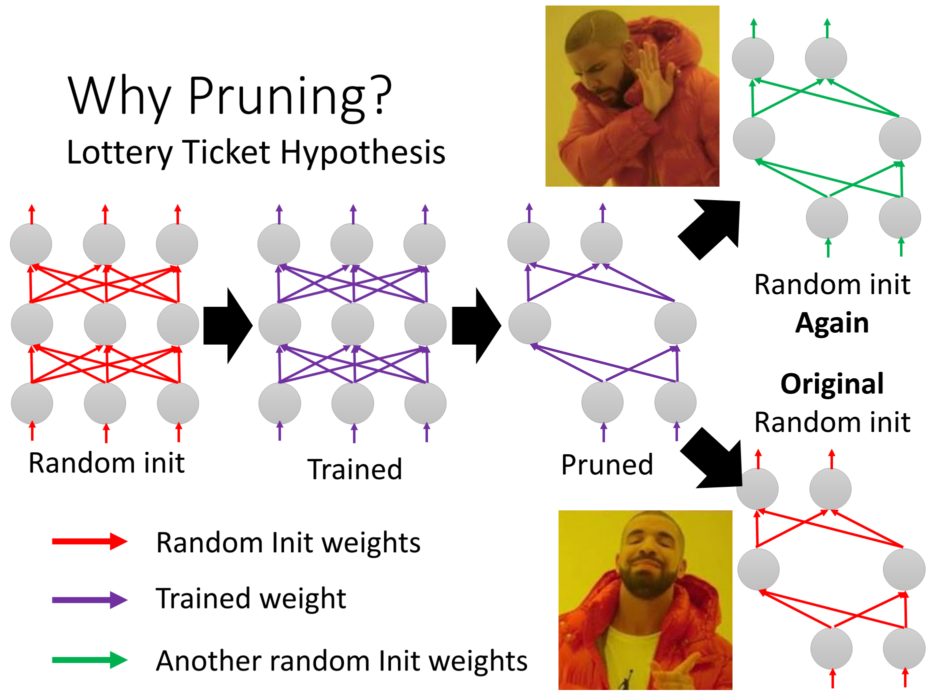
---

Lottery Ticket Hypothesis

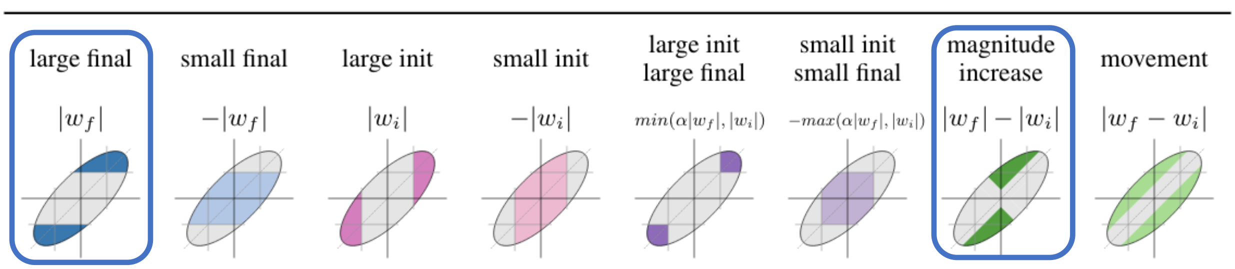
Large network can be seen the combination of small sub-networks.

How to exam lottery ticket hypothesis:





After pruning, we have “lucky” parameters.
Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask
Different pruning strategy
https://arxiv.org/abs/1905.01067



“sign-ificance” of initial weights: Keeping the sign is critical. (+ or -)
0.9, 3.1, -9.1, 8.5 ……→ $+\alpha$, $+\alpha$, - α, $+\alpha$ ……
Pruning weights from a network with random weights.
Parameters which can do classification are already in the large network. We just remove unwanted parameters.
Weight Agnostic Neural Networks https://arxiv.org/abs/1906.04358

Rethinking the Value of Network Pruning

New random initialization, not original random initialization in “Lottery Ticket Hypothesis”
Limitation of “Lottery Ticket Hypothesis” (**small learning rate, unstructured data**)

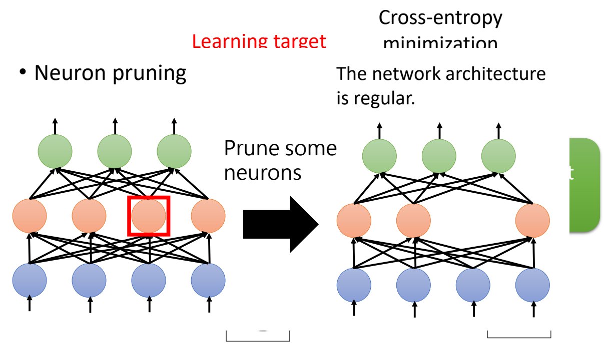
How about do more epoch?
When we have small learning rate, and unstructured data, we have higher probability to observe Lottery Ticket Hypothesis

Dataset	Model	Unpruned	Pruned Model	Fine-tuned	Scratch-E	Scratch-B
CIFAR-10	VGG-16	93.63 (±0.16)	VGG-16-A	93.41 (±0.12)	93.62 (±0.11)	**93.78** (±0.15)
	ResNet-56	93.14 (±0.12)	ResNet-56-A	92.97 (±0.17)	92.96 (±0.26)	**93.09** (±0.14)
			ResNet-56-B	92.67 (±0.14)	92.54 (±0.19)	**93.05** (±0.18)
	ResNet-110	93.14 (±0.24)	ResNet-110-A	93.14 (±0.16)	**93.25** (±0.29)	93.22 (±0.22)
			ResNet-110-B	92.69 (±0.09)	92.89 (±0.43)	**93.60** (±0.25)
ImageNet	ResNet-34	73.31	ResNet-34-A	72.56	72.77	**73.03**
			ResNet-34-B	72.29	72.55	**72.91**

Knowledge Distillation

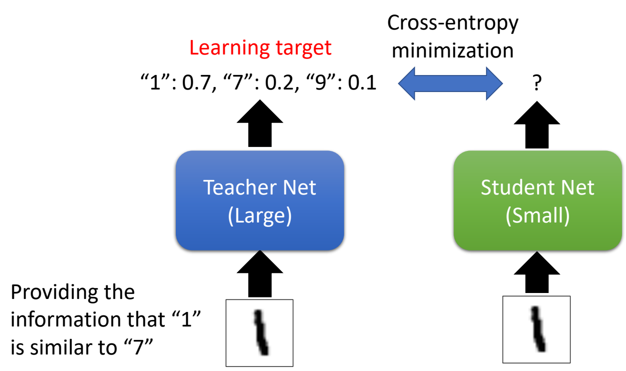
It is widely known that smaller network is more difficult to train than large networks.

1. Train a huge network (Teacher Network)
2. Create student network (small) by teacher network via **cross-entropy minimization.**



Review: Cross entropy
The cross-entropy of the distribution q relative to a distribution p over a given set is defined as follows:
$H(p,q) = -\mathbb{E}_p[\log q]$
For discrete probability:
$$H(p,q) = -\sum_{x\in\chi} p(x)\log q(x)$$
For continuous probability:
Let P and Q be probability density functions of p and q with respect to r. Then

$$H(p,q) = -\int_{\chi} P(x)\log Q(x)dr(x)$$



Export of student net need to follow teacher net’s export **even it is wrong**.
Because teacher network may provide extra information to student network.
Student network even and learn from teacher network without seeing some of the data.

Teacher network can be **Ensemble** of more networks. (Voting or average)

Combine Ensemble network to a small student work to reduce the computation cost.

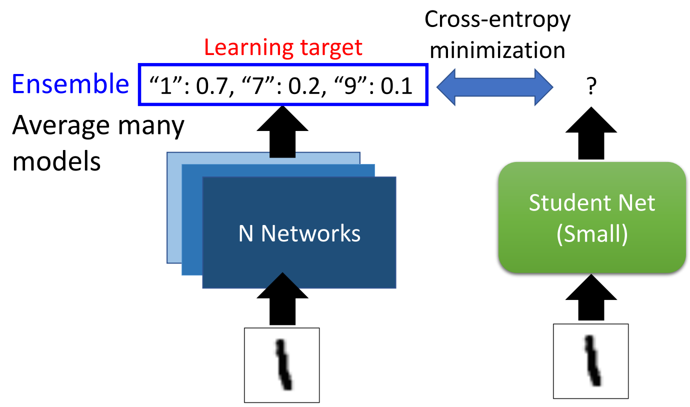
Temperature for softmax:

$$y_i' = \frac{exp(y_i)}{\sum_j exp(y_i)}$$

Temperature T: Convert sharp distributions to "smoother" distribution.

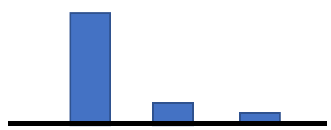
$$y_i' = \frac{exp(y_i/T)}{\sum_j exp(y_i/T)}$$

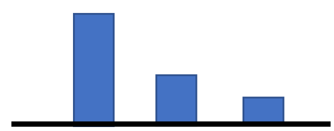
Temperature is a hyper-parameter.



- Temperature for softmax

$$y_i' = \frac{exp(y_i)}{\sum_j exp(y_j)} \quad \xrightarrow[T = 100]{} \quad y_i' = \frac{exp(y_i/T)}{\sum_j exp(y_j/T)}$$



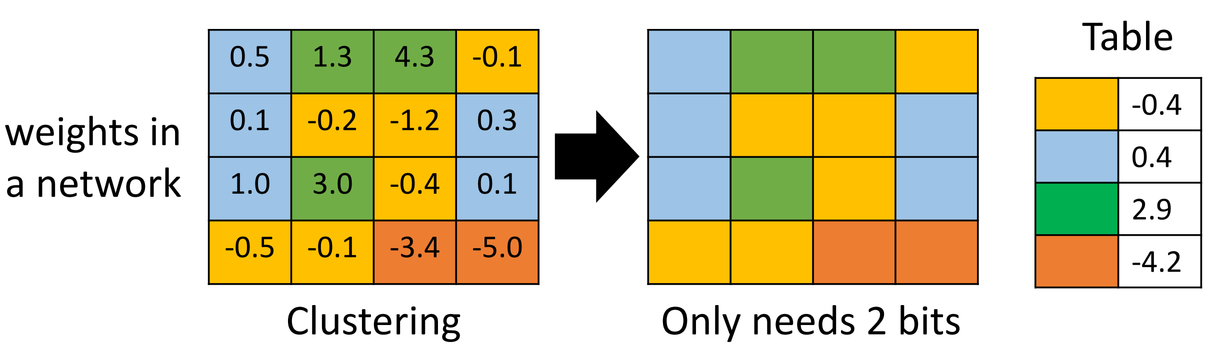


$$y_1 = 100 \qquad y_1' = 1$$
$$y_2 = 10 \qquad y_2' \approx 0$$
$$y_3 = 1 \qquad y_3' \approx 0$$

$$y_1/T = 1 \qquad y_1' = 0.56$$
$$y_2/T = 0.1 \qquad y_2' = 0.23$$
$$y_3/T = 0.01 \qquad y_3' = 0.21$$

Parameter Quantization

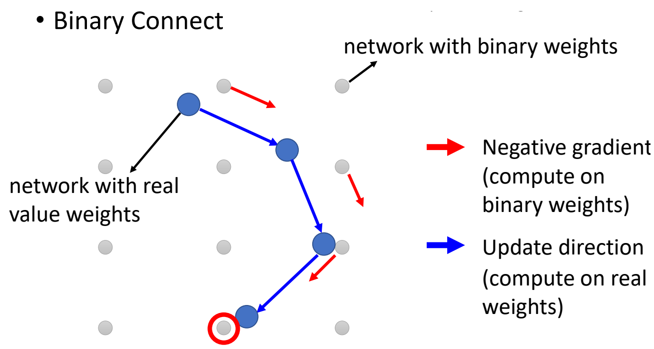
Using less bits to represent a value. - Use less space or bit to store a parameter.
Weight clustering



Easy implement: after training then clustering. But may have problem.
Solution: When training, let parameters be more close to each other.
Represent frequent clusters by less bits, represent rare clusters by more bits
e.g. **Huffman encoding**
Describe Moore general thing by less bits, rare things for more bits.

Binary Weights
Your weights are always +1 or -1
Binary Connect
Binary Network
XNOR-net

Explanation: Using Binary weight can prevent overfitting.

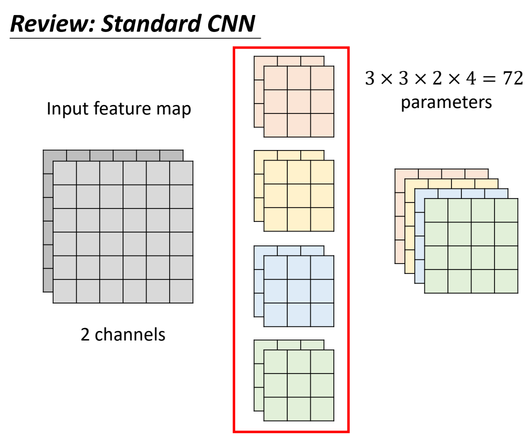


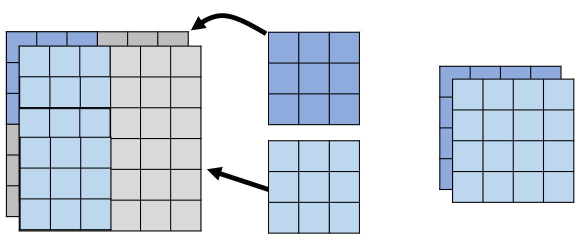
Architecture Design

Depthwise Separable Convolution

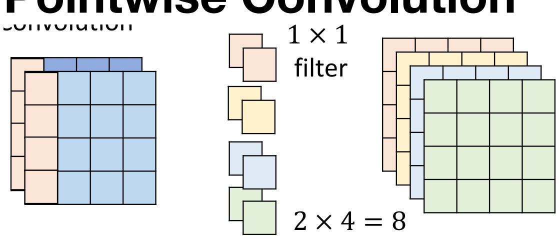
Depthwise Convolution

How many channels, how many filters.
Filter number = Input channel number
Each filter only considers one channel.
The filters are $k \times k$ matrices. One filter only have 1 channel.
There is no interaction between channels.
Can not read features cross channels.





Pointwise Convolution



Force the filter size is 1 x 1. Pointwise Convolution only consider the interaction between channels.

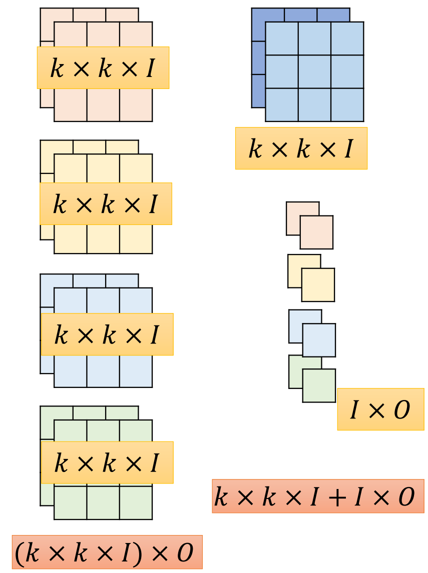
I: number of input channels

O: number of output channels (It's usually big.)

$k \times k$: kernel size

$$\frac{k \times k \times I + I \times O}{k \times k \times I \times O} = \frac{1}{O} + \frac{1}{k \times k}$$

It's more related to $\frac{1}{k \times k}$



Why it is useful? - Low rank approximation

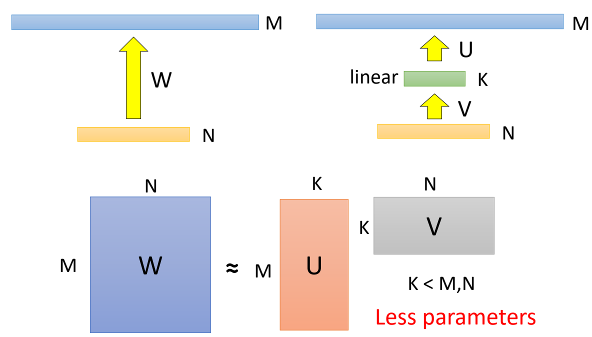
Original parameter number:

$W = N \times M$

Then we can insert a liner layer to reduce the parameters.

$M * N \sim M \times K + K \times N$

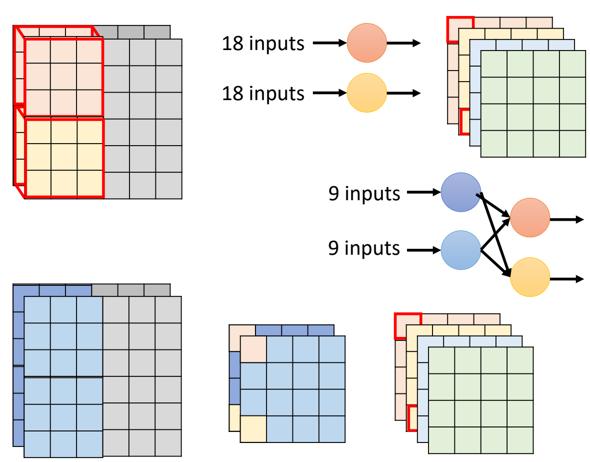
K < M, N



Depthwise + Pointwise Convolution reduces parameters in a similar way.

Original CNN: $18\ inputs \rightarrow 1\ output$

DPC: $9 \times 2\ inputs \rightarrow 2 \rightarrow 1\ output$



More…

- SqueezeNet
- https://arxiv.org/abs/1602.07360
- MobileNet
- https://arxiv.org/abs/1704.04861
- ShuffleNet
- https://arxiv.org/abs/1707.01083
- Xception
- https://arxiv.org/abs/1610.02357
- GhostNet
- https://arxiv.org/abs/1911.11907

Dynamic Computation

The network adjusts the computation it need.

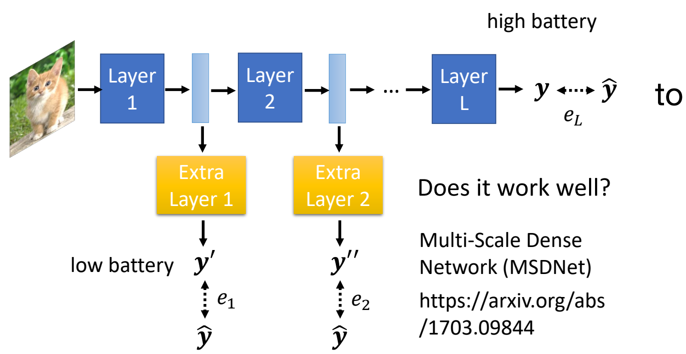
1. Because we would like to run the same model on different devices with different resources.
2. Even the same device may need different computation resource.

Dynamic Depth

Add extra layer between layers. The function of the extra layer is determine the class based on the output of previous hidden layer.

$$L = e_1 + e_2 + \ldots + e_L$$

(Currently best: Multi-Scale Dense Network (MSDNet))

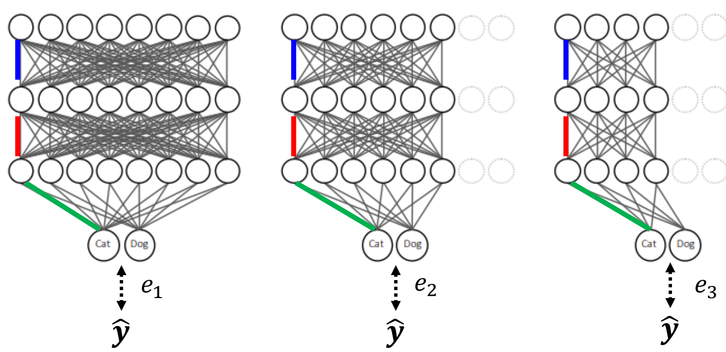


Dynamic Width

The same network can choose different width.

Slimmable Neural Networks
https://arxiv.org/abs/1812.08928

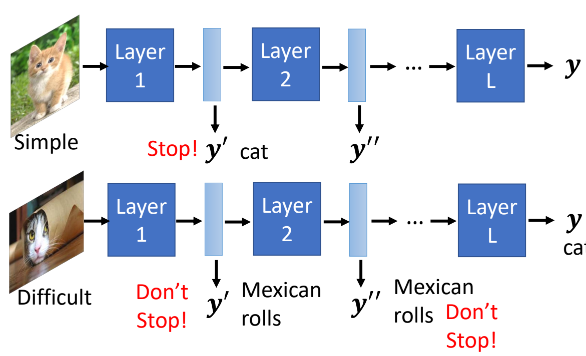
$$L = e_1 + e_2 + e_3$$



Computation based on Sample Difficulty

Let network to decide the width and depth.

SkipNet: Learning Dynamic Routing in Convolutional Networks
RuntimeNeuralPruning
BlockDrop: Dynamic Inference Paths in Residual Networks



Concluding Remarks

• Network Pruning
• Knowledge Distillation
• Parameter Quantization • Architecture Design
• Dynamic Computation

Those skills are not mutual. You can use all or some of them.