EE655: Computer Vision & Deep Learning

Lecture 09

Koteswar Rao Jerripothula, PhD
Department of Electrical Engineering
IIT Kanpur

Back Propagation in Convolution Base

Convolution Layers

Pooling Layers

Backpropagation in Convolutional layer

$$dy = \frac{\partial C}{\partial y}$$

$$dx = \frac{\partial C}{\partial x}$$

$$dw = \frac{\partial C}{\partial w}$$

$$db = \frac{\partial C}{\partial b}$$

y=activation matrix of a convolution layer

x=input matrix

w=filter

b=bias weight

C=Loss

- ‘dw’ and ‘db’ are needed for updating CNN weights
- ‘dx’ is required to have ‘dy’ for the next convolutional layer (in backward direction)

Motivation

$$dx_{ij} = \frac{\partial C}{\partial x_{ij}} = \frac{\partial C}{\partial y} \bullet \frac{\partial y}{\partial x_{ij}}$$

$$dw_{uv} = \frac{\partial C}{\partial w_{uv}} = \frac{\partial C}{\partial y} \bullet \frac{\partial y}{\partial w_{uv}}$$

$$db = \frac{\partial C}{\partial b} = \frac{\partial C}{\partial y} \bullet \frac{\partial y}{\partial b}$$

$$\bullet = dot\_product$$

$$ij = indices\_in\_x - matrix; x_{ij}\_is\_an\_element\_in\_x$$

$$uv = indices\_in\_w - matrix; w_{uv}\_is\_an\_element\_in\_w$$

$$y = y - matrix\_convereted\_to\_a\_vector$$

Final Formulas

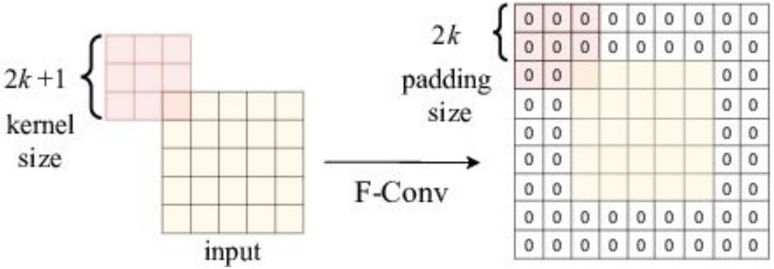
$$db = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij}$$

$$dw = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} * \begin{bmatrix} dy_{11} & dy_{12} & dy_{13} \\ dy_{21} & dy_{22} & dy_{23} \\ dy_{31} & dy_{32} & dy_{33} \end{bmatrix} = x * dy$$

$$dx = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & dy_{11} & dy_{12} & dy_{13} & 0 \\ 0 & dy_{21} & dy_{22} & dy_{23} & 0 \\ 0 & dy_{31} & dy_{32} & dy_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix} = dy\_0 * w'$$

*=convolution
dy_0=padded ‘dy’ (ensuring full convolution)
w’=180 degree rotated w

What's Full Convolution?



$$\begin{pmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 1 \\ 0 & 5 & 0 \\ 2 & 1 & 2 \end{pmatrix}$$

full same valid

$$\begin{bmatrix} 17 & 75 & 90 & 35 & 40 & 53 & 15 \\ 23 & 159 & 165 & 45 & 105 & 137 & 16 \\ 38 & 198 & 120 & 165 & 205 & 197 & 52 \\ 56 & 95 & 160 & 200 & 245 & 184 & 35 \\ 19 & 117 & 190 & 255 & 235 & 106 & 53 \\ 20 & 89 & 160 & 210 & 75 & 90 & 6 \\ 22 & 47 & 90 & 65 & 70 & 13 & 18 \end{bmatrix}$$

In pooling layers

- In a max-pooling layer, the gradient is simply passed to the input value that is maximum. The gradients for all other inputs are set as 0.

Let $y = \max(x_1, x_2, \ldots, x_n)$ be the max-pooling operation. The gradient is:

$$\frac{\partial L}{\partial x_i} = \begin{cases} \frac{\partial L}{\partial y} & \text{if } x_i = y \\ 0 & \text{otherwise} \end{cases}$$

- In a average-pooling layer, the gradient is evenly distributed to the input values

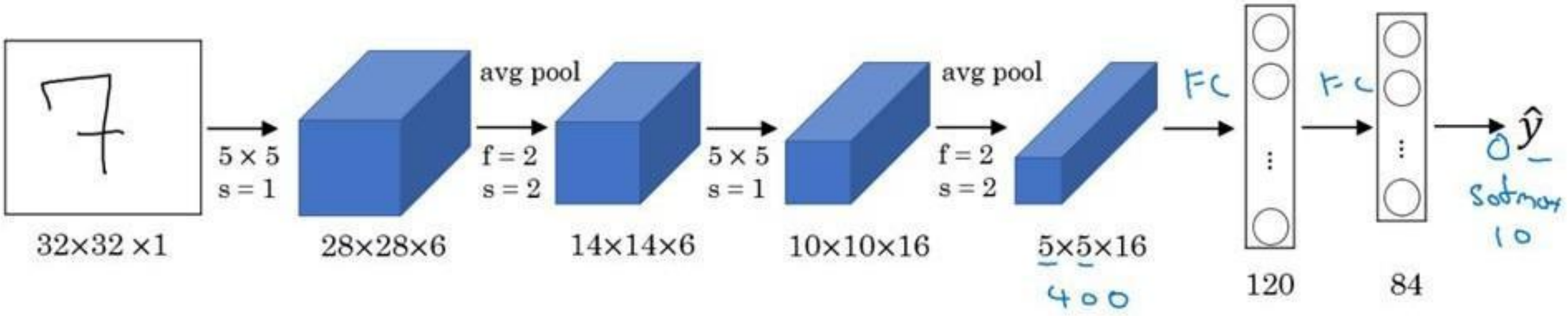
If $y = \frac{1}{n}\sum_{i=1}^{n} x_i$, then the gradient is:

$$\frac{\partial L}{\partial x_i} = \frac{1}{n}\frac{\partial L}{\partial y}$$

Sample CNNs

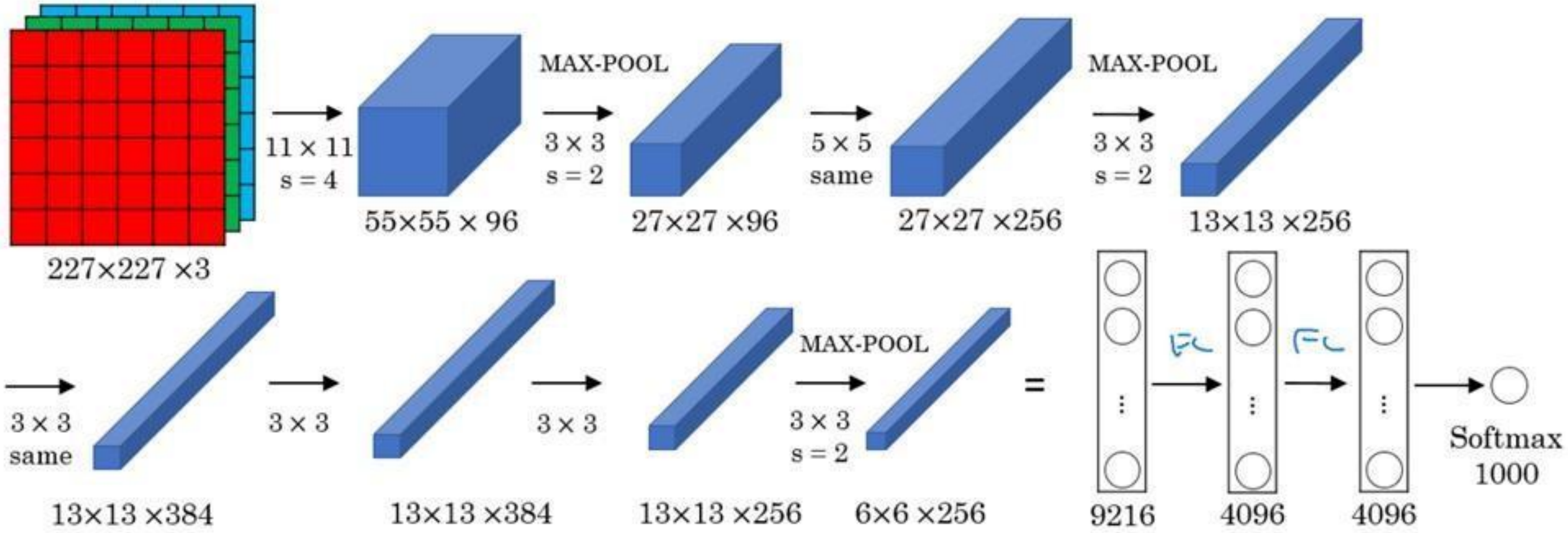
- LeNet
- AlexNet
- VGGNet

- LeNet-5
    - The goal for this model was to identify handwritten digits in a `32x32x1` gray image. Here are the drawing of it:



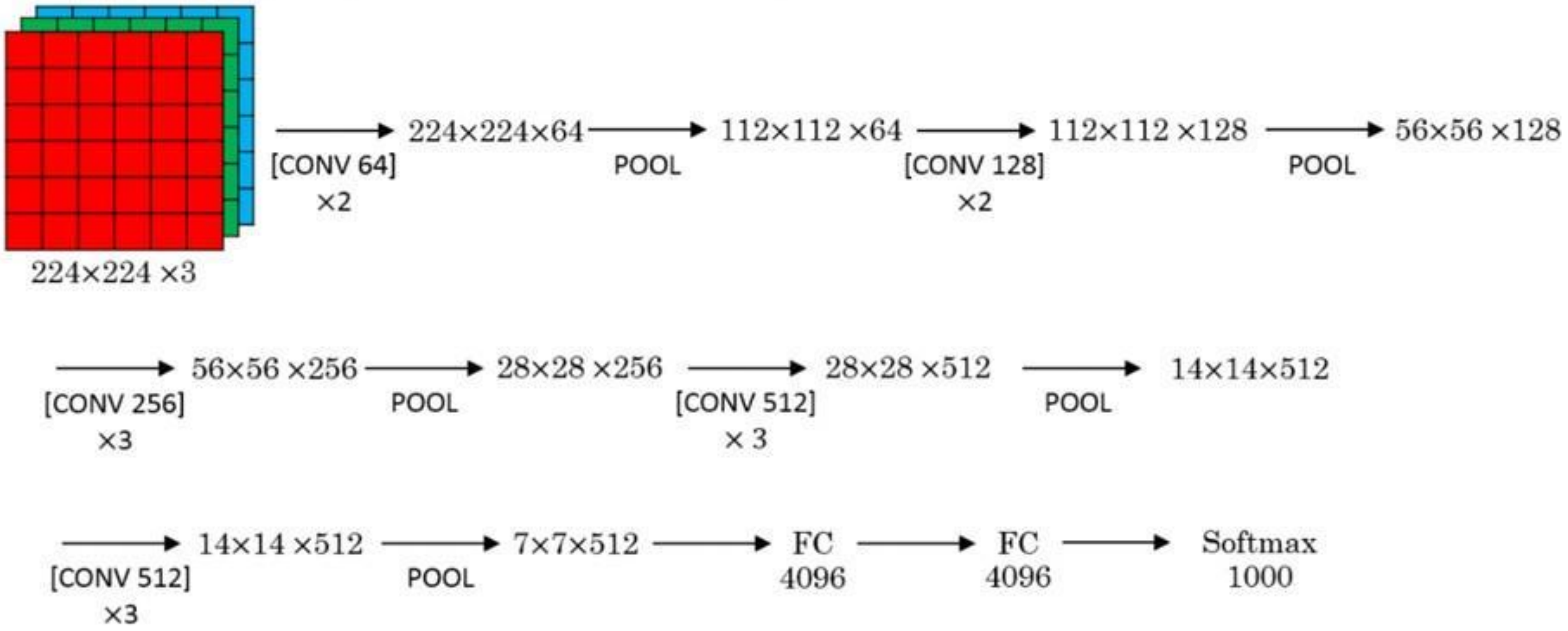
    -
    - This model was published in 1998. The last layer wasn't using softmax back then.
    - It has 60k parameters.
    - The dimensions of the image decreases as the number of channels increases.
    - `Conv ==> Pool ==> Conv ==> Pool ==> FC ==> FC ==> softmax` this type of arrangement is quite common.
    - The activation function used in the paper was Sigmoid and Tanh. Modern implementation uses RELU in most of the cases.
    - [LeCun et al., 1998. Gradient-based learning applied to document recognition]

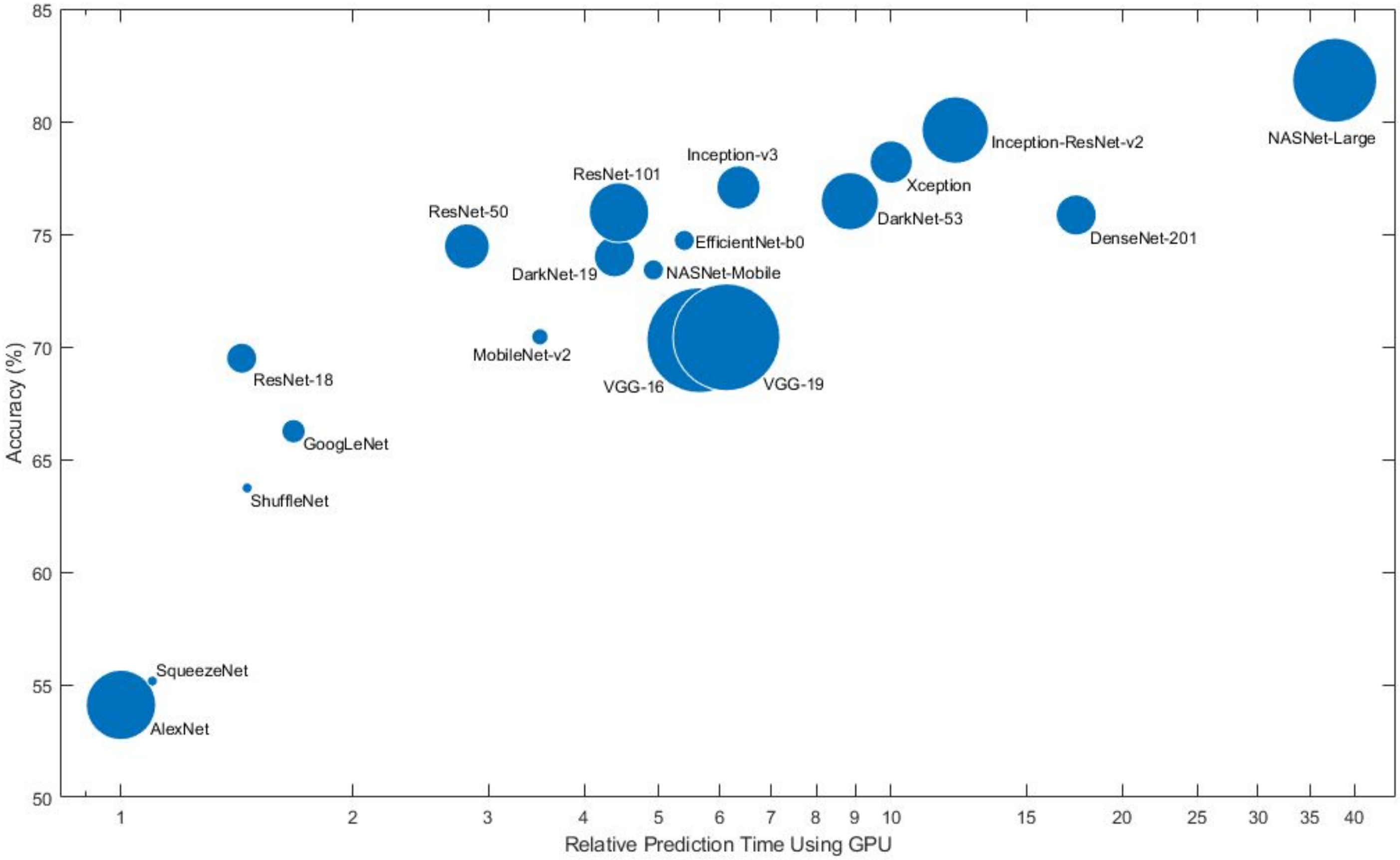
- AlexNet
  - Named after Alex Krizhevsky who was the first author of this paper. The other authors includes Geoffrey Hinton.
  - The goal for the model was the ImageNet challenge which classifies images into 1000 classes. Here are the drawing of the model:



  - Summary:
    - `Conv => Max-pool => Conv => Max-pool => Conv => Conv => Conv => Max-pool ==> Flatten ==> FC ==> FC ==> Softmax`

- VGG-16
  - A modification for AlexNet.
  - Instead of having a lot of hyperparameters lets have some simpler network.
  - Focus on having only these blocks:
    - CONV = 3 X 3 filter, s = 1, same
    - MAX-POOL = 2 X 2 , s = 2
  - Here are the architecture:



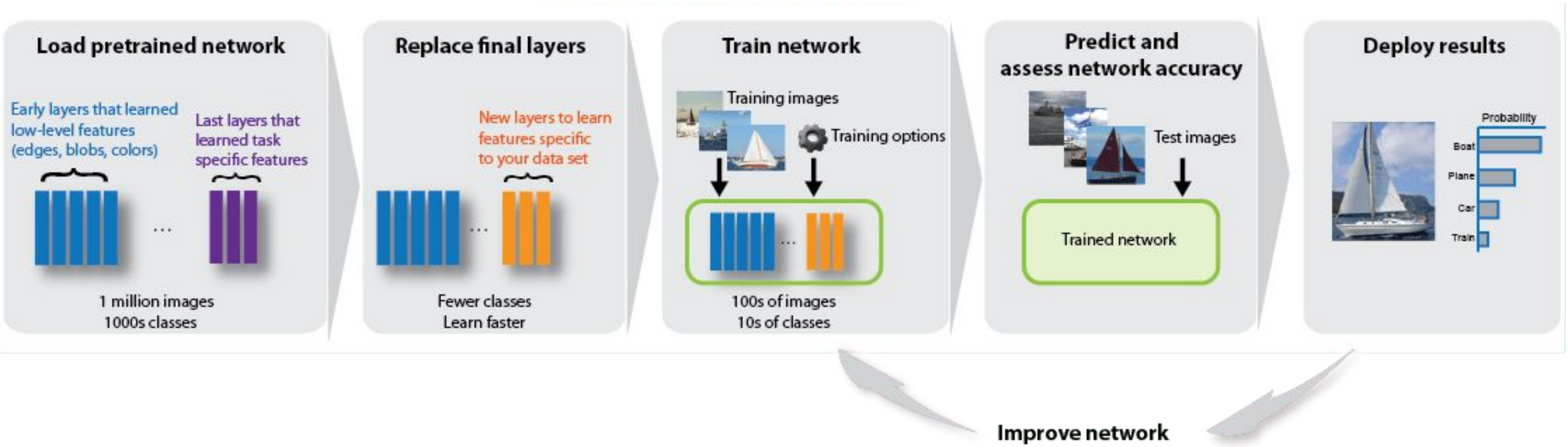


85
80
75
70
65
60
55
50
Accuracy (%)
NASNet-Large
Inception-ResNet-v2
Xception
Inception-v3
ResNet-101
DarkNet-53
DenseNet-201
ResNet-50
EfficientNet-b0
DarkNet-19
NASNet-Mobile
MobileNet-v2
ResNet-18
VGG-16
VGG-19
GoogLeNet
ShuffleNet
SqueezeNet
AlexNet
1
2
3
4
5
6
7
8
9
10
15
20
25
30
35
40
Relative Prediction Time Using GPU

Pre-trained networks Usage: Transfer Learning

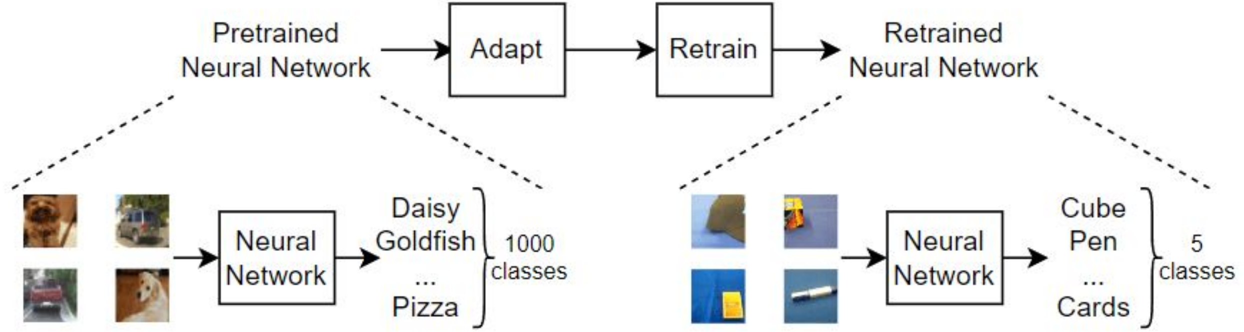
Networks that have been already trained on large amounts of data

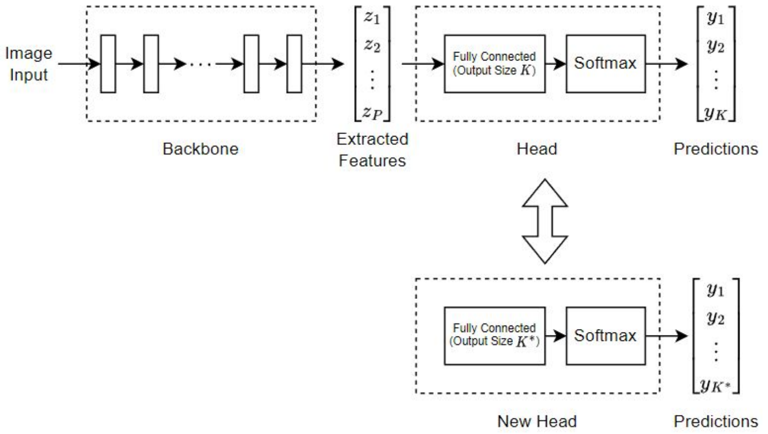
- ❖ **Fine-tuning:** Learning last few layers for solving a similar problem
- ❖ **Feature Extraction:** Using Pre-trained networks as feature extractor for solving a similar problem
- ❖ **Backbone:** Using Pre-trained networks as backbones for solving more complex problems

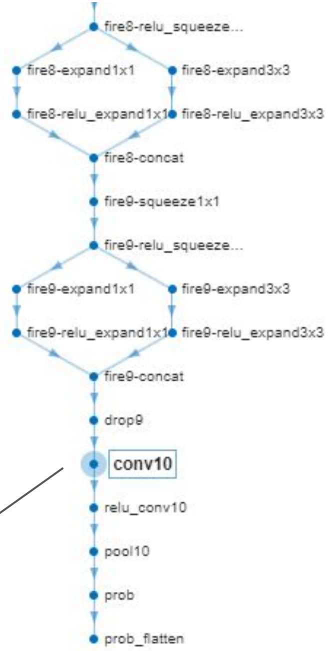


Load pretrained network
Early layers that learned low-level features (edges, blobs, colors)
Last layers that learned task specific features
1 million images
1000s classes
Replace final layers
New layers to learn features specific to your data set
Fewer classes
Learn faster
Train network
Training images
Training options
100s of images
10s of classes
Predict and assess network accuracy
Test images
Trained network
Deploy results
Probability
Boat
Plane
Car
Train
Improve network

How to fine-tune?

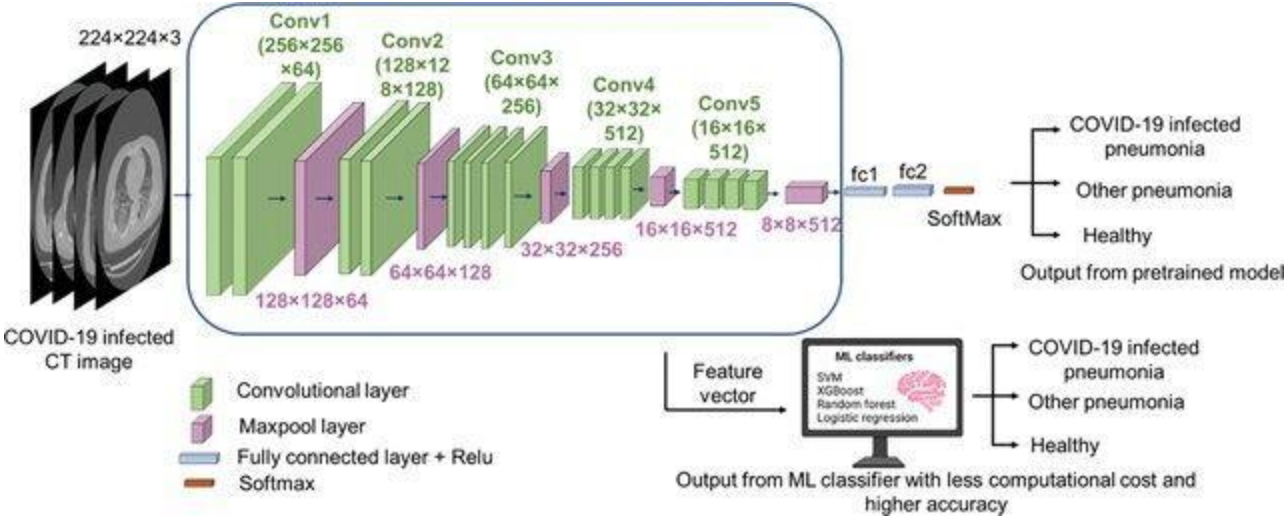






Replace it

We can also extract features from them and use a ML classifier



As a backbone