Case Studies

Wednesday, September 9, 2020 11:52 AM

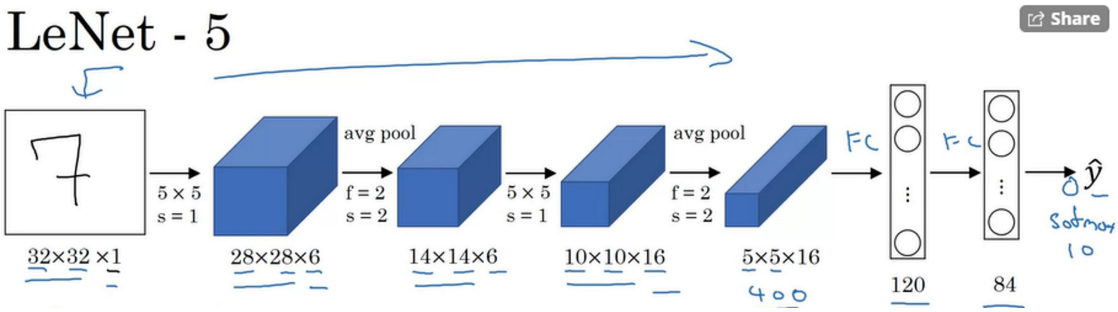
① Why case studies

→ Classic Networks
- LeNet-5
- AlexNet
- VGG

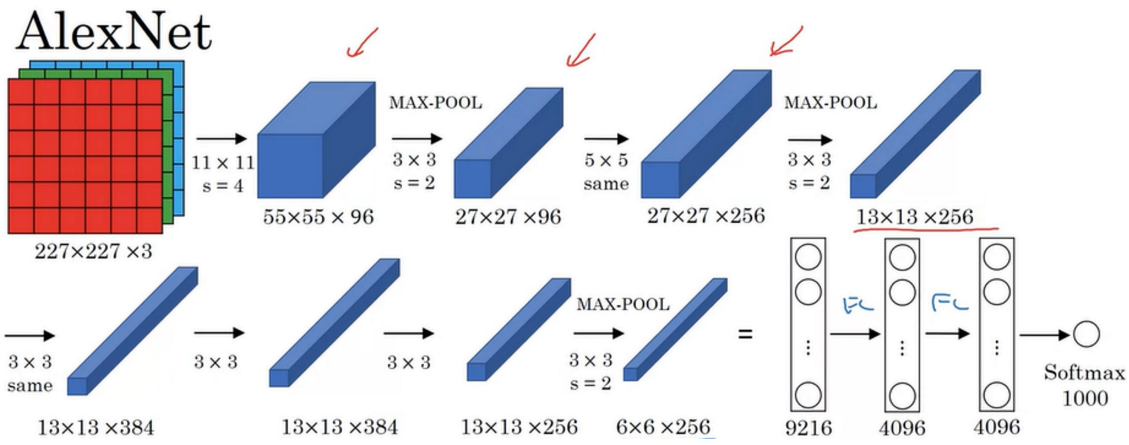
→ ResNet network
→ Inception Network

② Classic Networks

→ LeNet-5
- To recognise handwritten digits
- 32×32×1 images (gray scale image)
- Padding wasn't used
- Small for today's standards (~60K parameters)
- n_H n_W ↓ & n_C ↑ as we go deeper
- conv+pool → conv+pool → fc → fc → output



→ AlexNet
- Similar to LeNet, but much bigger
- 60M parameters
- used ReLU activation function
- multiple GPUs were used in training

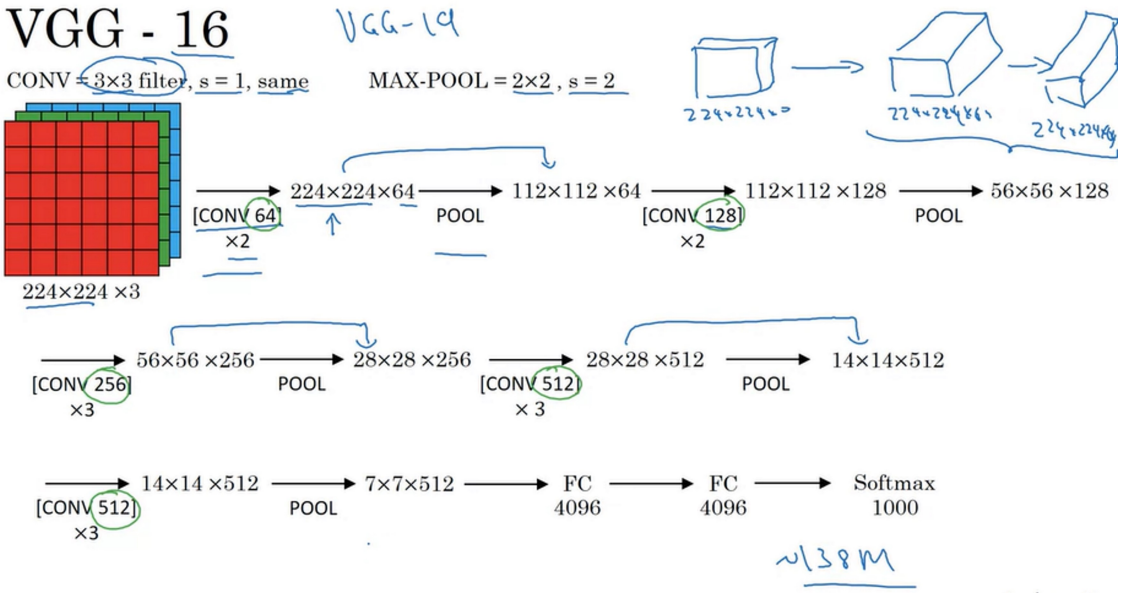


→ VGG-16

13×13 ×384	13×13 ×384	13×13 ×256	6×6 ×256	9216	4096	4096

→ VGG-16

- Deeper than previous networks
- ~138M parameters
- Filters were doubled after every POOL
- The uniformity of this network made it attractive

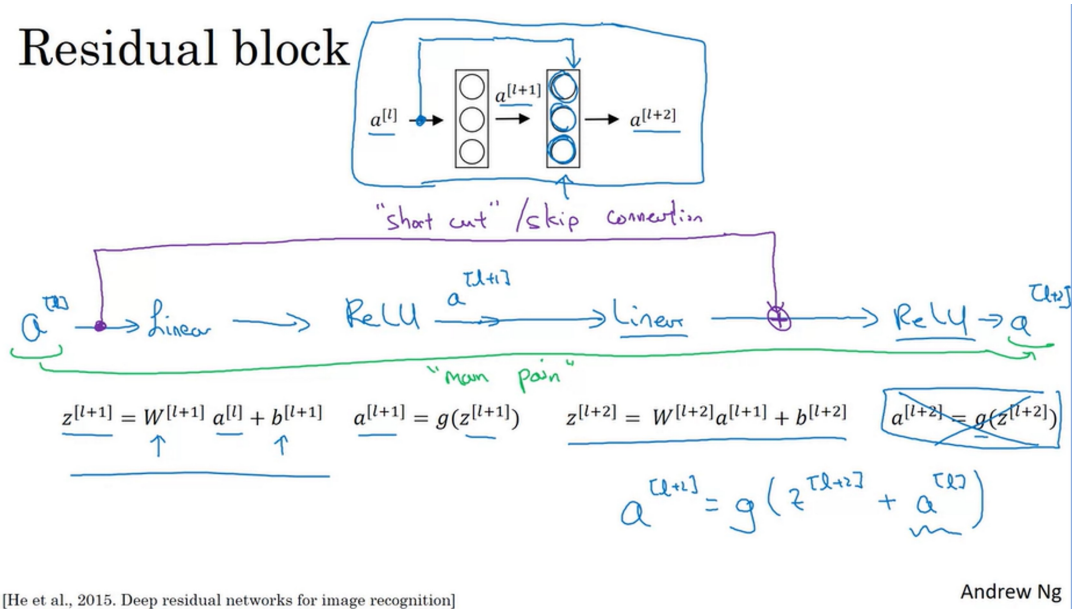


[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng

③ ResNets

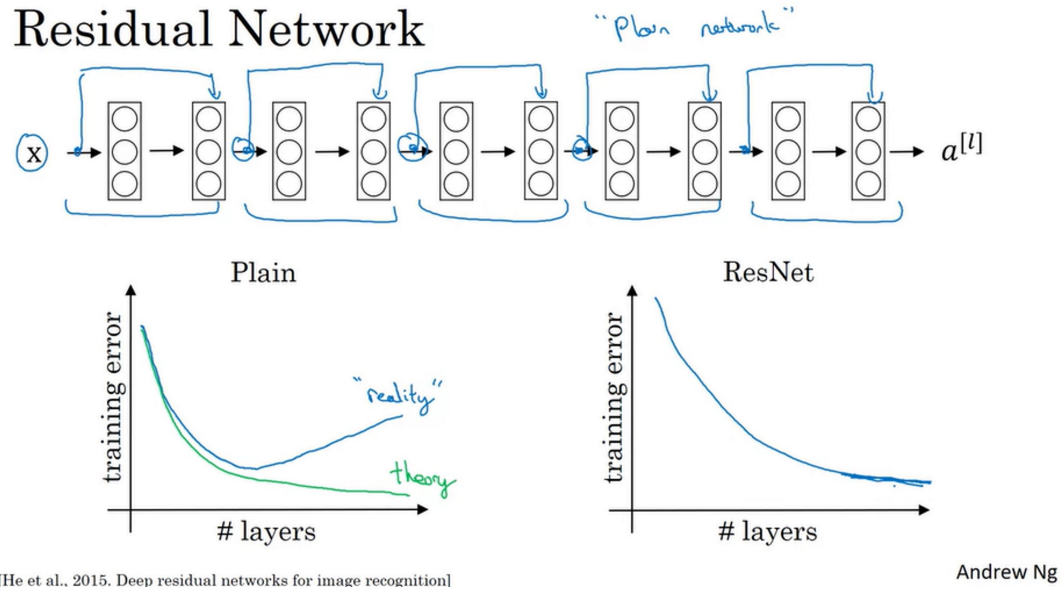
- Using skip connections



[He et al., 2015. Deep residual networks for image recognition]

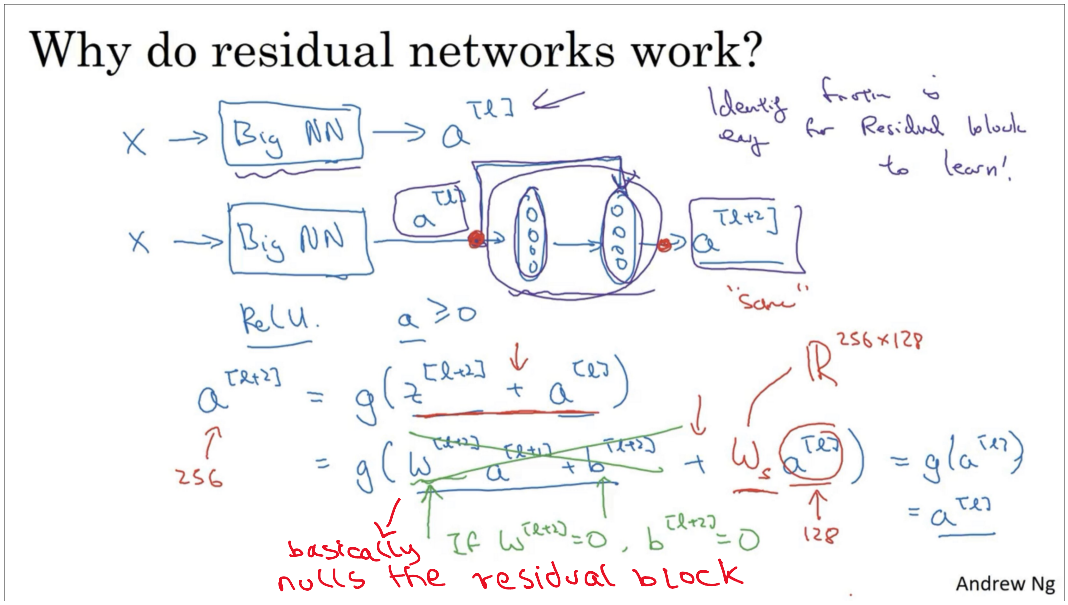
Andrew Ng

- Residual Network



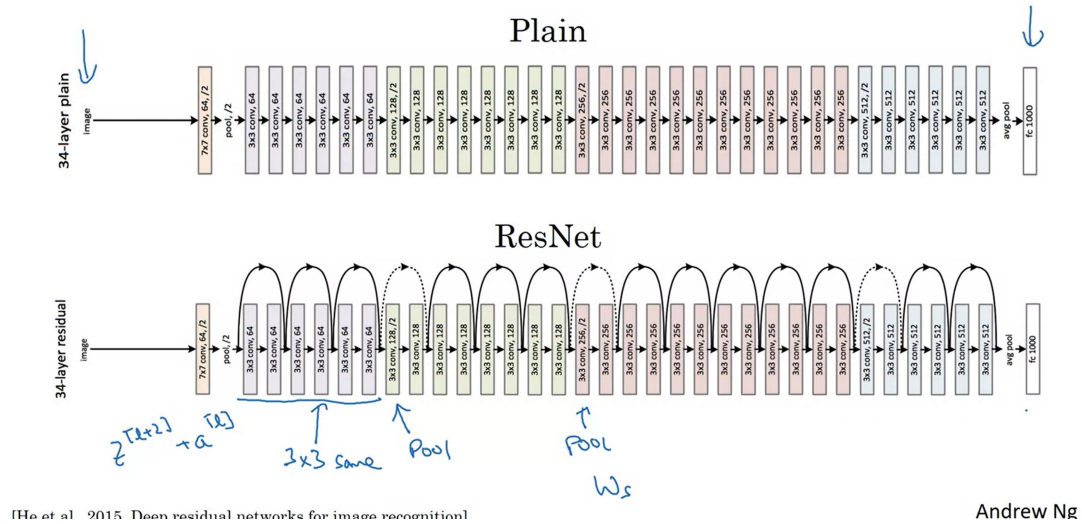
④ Why ResNets work

→ Usually deeper networks hurt performance But this isn't true for ResNets



- Using a residual block never hurts performance as it can easily learn the identity fn.
- This is particularly helpful in large networks where lot of parameters make it hard for the network to learn identity function

→ ResNet Paper

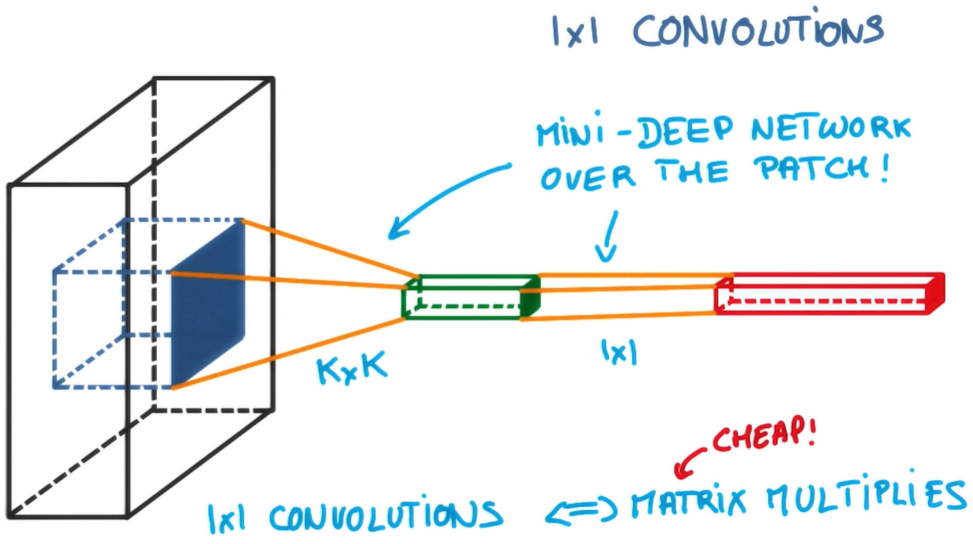


⑤ Networks in Networks and 1x1 Convolutions

→ When using 1x1 convolution,
- Input: 6x6x1
  Conv: 1x1x1 (one filter)
  Output: 6x6x1
- Input: 6x6x32
  Conv: 1x1x32 (5 filters)
  Output: 6x6x5

→ It is useful when
- we want to shrink number of channels called feature transformation
- This can help in saving lot of computations
- If we use same number of filters as number of channels, output contains same number of channels.
- The 1x1 conv will act like a non-linearity and will learn non linearity operator

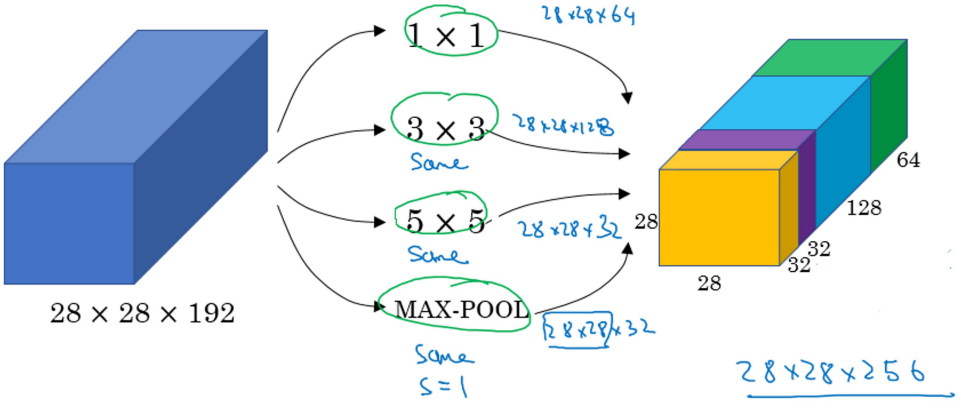
→ We could replace a fully connected layer with 1x1 conv as they are essentially the same



→ Cheap way to make model deeper without adding lot of parameters

⑥ Inception Network Motivation

→ use multiple convolutions and let model decide weights



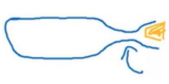
→ Problem of computational cost

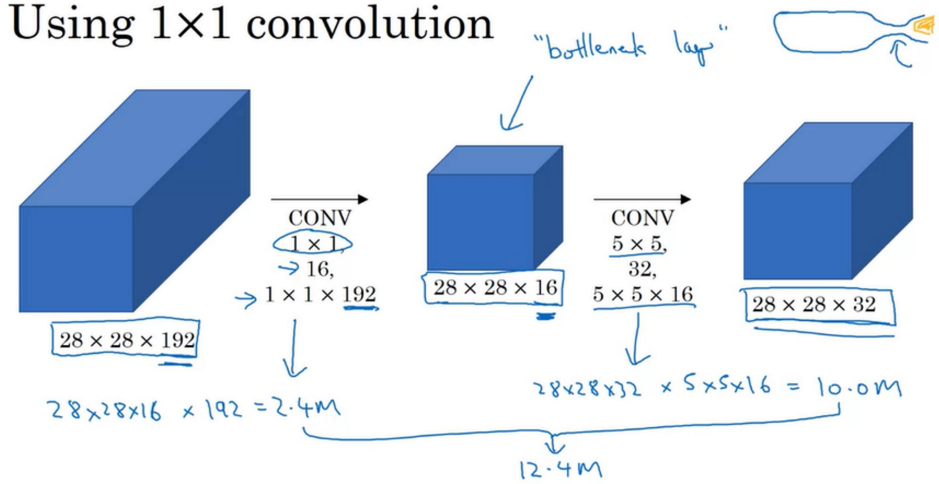
- Since we use multiple filters the no. of operations is very high
  consider just 5×5 conv.
  we use 32 filters

∴ No. of operations
= no. of outputs * f * f * input dimensions
= (28 * 28 * 32) * (5 * 5) * 192
= 120 Million

- 120 Million is large even for modern computers
- Using 1×1 conv.

Using 1×1 convolution "bottleneck layer"



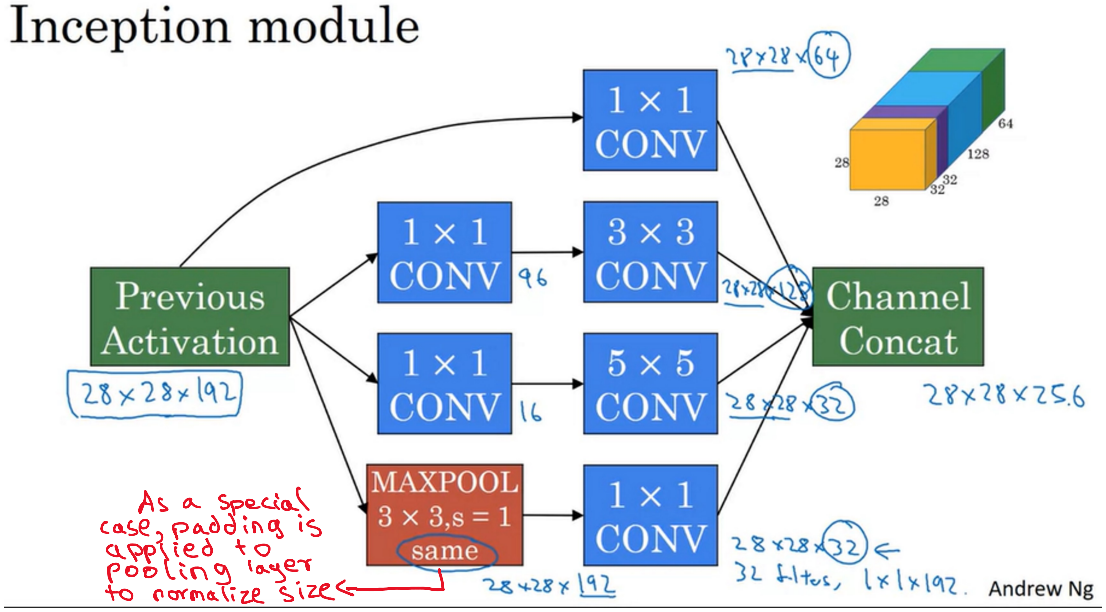


we have reduced parameters from 120M to 12.4 M by using a 1x1 "bottleneck layer"

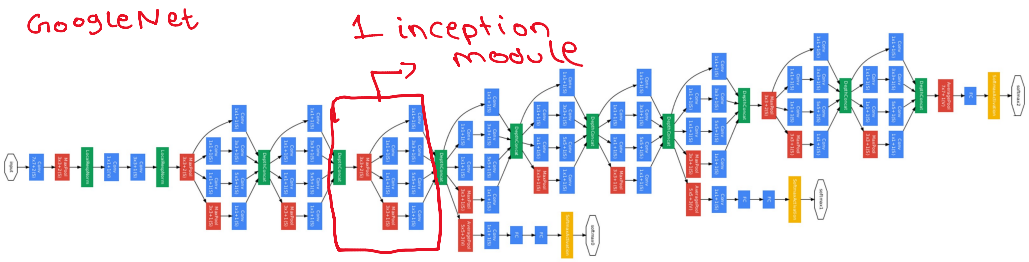
→ shrinking the representation size doesn't affect performance of model as long as done within reason

⑦ Inception Network

→ Inception Module



→ The inception network combines a lot of these modules to make a deep network


GoogleNet
1 inception module