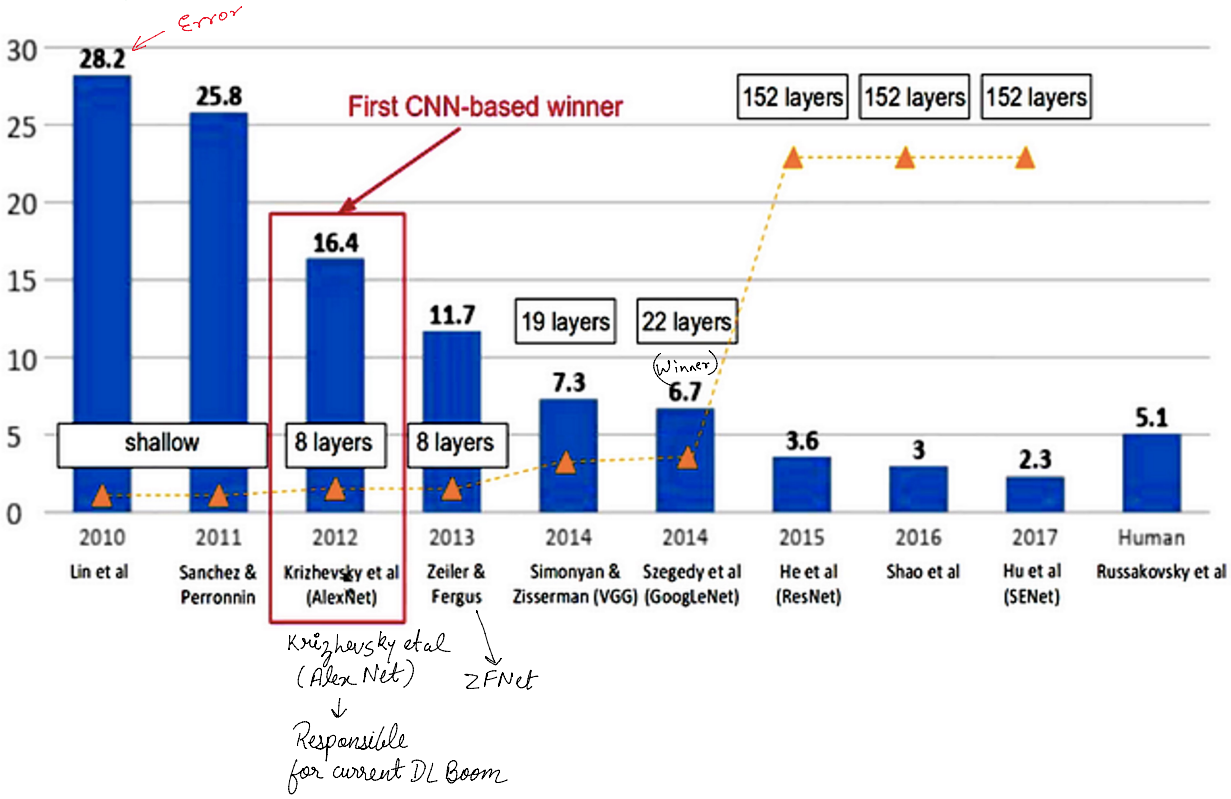
CNN Part - 3

13 September 2023 10:09 AM

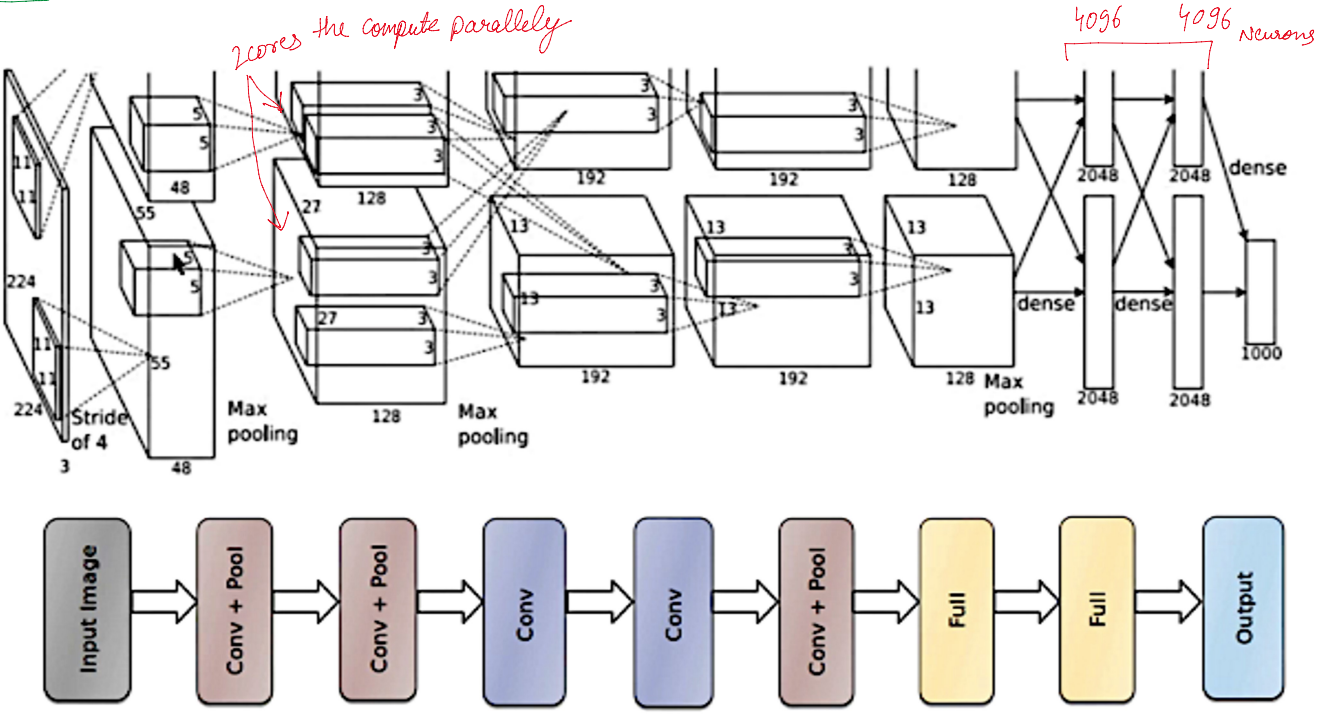
Image Net challenge



Krizhevsky et al (Alex Net) → Responsible for current DL Boom

ZFNet

Alex Net



→ Input Image Size = 224 × 224 × 3

→ first layer filter = 11 × 11 × 3
Stride = 4

Train set size = 1.2 M images & 60 M parameters.
Optimization algo = mini batch G.D.

96 such filters
↓
computation was divided across 2 cores
↙ 48 filters ↘ 48 filters

Trained on → NVIDIA GTX 560 GPU (1 week to train)
→ Activation function → ReLU

⭐ AlexNet introduced ReLU as an activation function.

In AlexNet, convolution layers accounted for about 90-95% computation but contained only about 5% of total parameters.
↳ This is because of weight sharing, the same filter is used on the entire image ⇒ less parameters, but every time it is applied it is convoluted ⇒ more computations as it happens on entire image.

In Dense layer the parameters are more → (4096 × 4096) Number of connections b/w first 2 Dense layers.

ZF-Net

It is similar to AlexNet, with some slight modifications
↳ Conv-1: Changed from (11x11, 4 stride) to → (7x7, 2 stride)
{Filter in AlexNet}

↳ Conv-3,4,5: Instead of using 384, 384, 256 filters respectively, ZF-Net used 512, 1024, 512 filters respectively.

ZF-Net was made by performing really good hyperparameter tuning on Alex Net.

Google Net

(won the 2014 Image Net Challenge)

Important traits of Google Net:

1.) 22 layer architecture

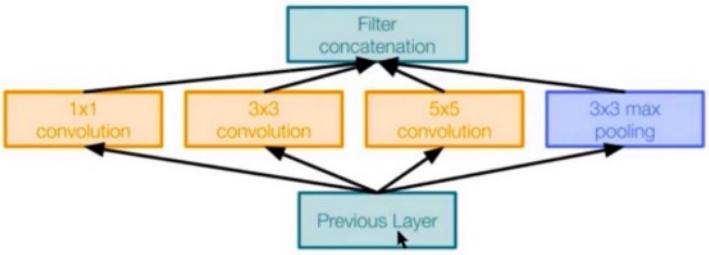
2.) No fully connected layers → Because of a large number of parameters

3.) Deeper Network with focus on efficiency.
↳ Here the complexity of the model is also taken into consideration, which ensures that there is no overfitting.

4.) 5 million parameters.

Google Net was able to do all this and utilized the concept of Inception Module.

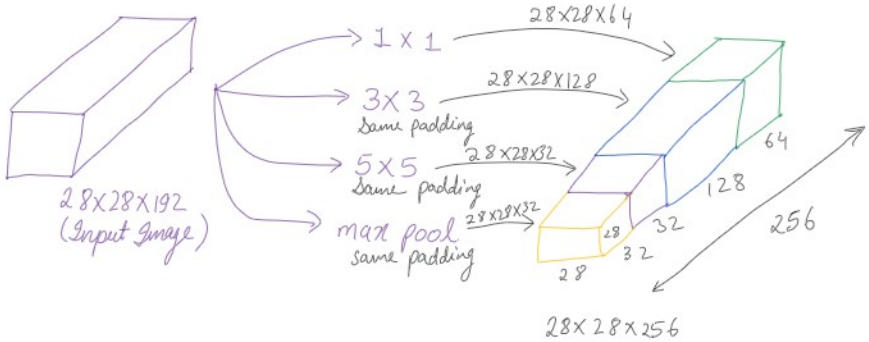
Inception Module



⭐ Inception module is helping us to pay attention to different features of different sizes by using multiple filter/conv. layers.

⭐ The basic idea is that rather than selecting one filter why not try all of them.

Eg

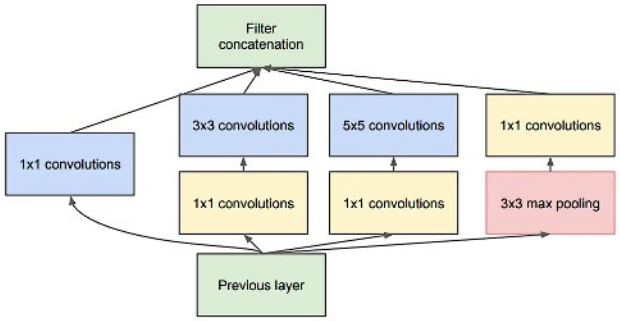


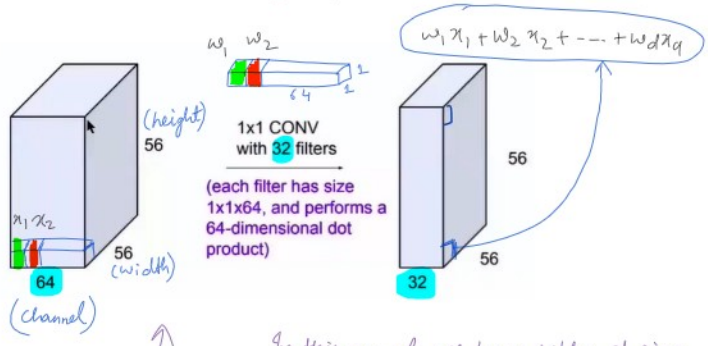
⭐ The final results are all concatenated/stacked together.

Problems with Inception Module

↳ Still computationally very expensive

Solution → Use 1x1 bottleneck layers to reduce the channel size before the expensive convolution operation.

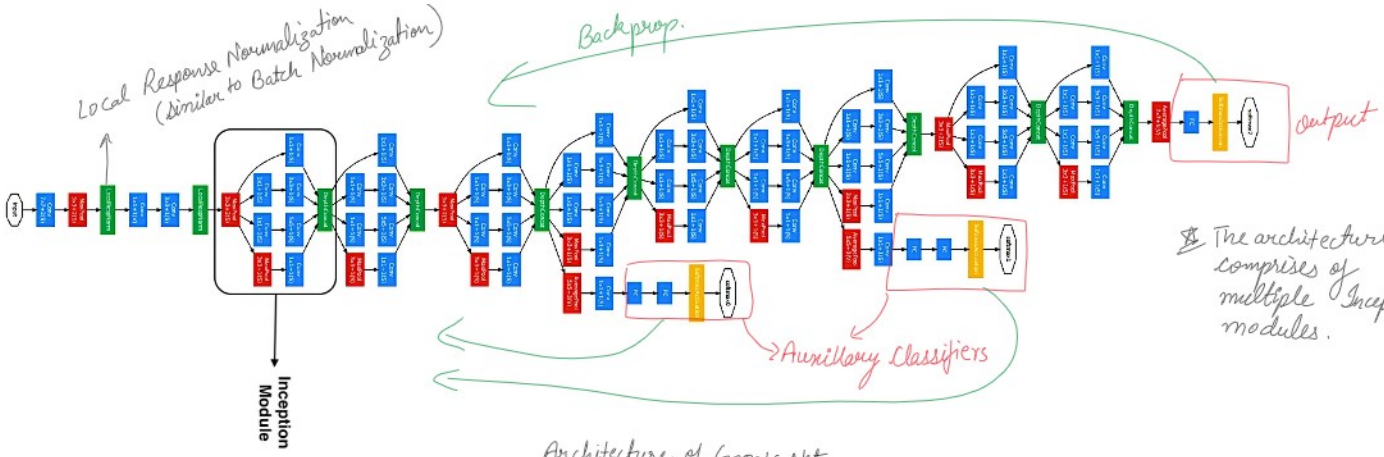




In this example we have filter of size (1x1x64) and there are 32 such filters.

1x1 convolution layer → Bottle neck layer

* Only the last output layer is fully connected layer.



* The architecture comprises of multiple Inception modules.

Architecture of GoogLe Net.

Q. What is the need for auxillary classifiers?

Ans When performing backpropagation, since the depth of the network is too large, the gradient doesn't reach back and we run into the problem of Vanishing Gradient. In order to counter this, for the layers that are very far from the output, they not only recieve gradient from the output layer but from the auxillary classifier as well, thereby preventing vanishing gradient. This happens by adding all the gradients recieved from the auxillary classifier + the final output layer.

↳ (Ref. Backprop. in Depth)

Q. In the inception module, why does the 1x1 conv. layer come after max pool layer and not before?

Ans Performing convolution is similar to getting an output pixel which is a linear combination of pixels like $w_1x_1 + w_2x_2 + \dots + w_dx_d$ (Refer Pic. above), now post convolution if we perform maxpooling all the unique important features in the channels of input image can't be filtered out.

So, by performing max pooling first we extract the important features and then by applying 1x1 conv. we can reduce the nb. of channels as in maxpooling no. of channels don't change.