CNN Architectures

DNN_08

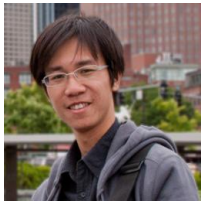
Lets Revisit the Networks

- Lenet - 5 (1998) by Yann Lecunn currently at Facebook AI Research

- Alexnet (2012) by Alex Krizhvesky currently helping other DL researchers and left his job at Google in 2017 as he lost his interest in his work

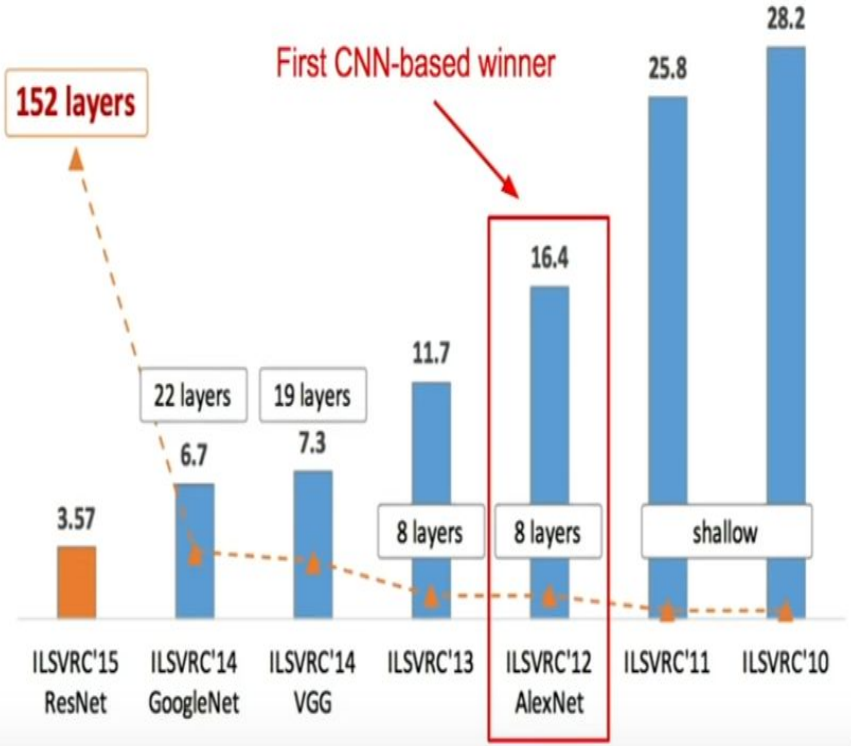
- VGG -16 (16 layers) in 2014 by karen and zisserman as a part of Oxfords Visual Geometry Group
- Inception(22 layers) in 2014 by Google

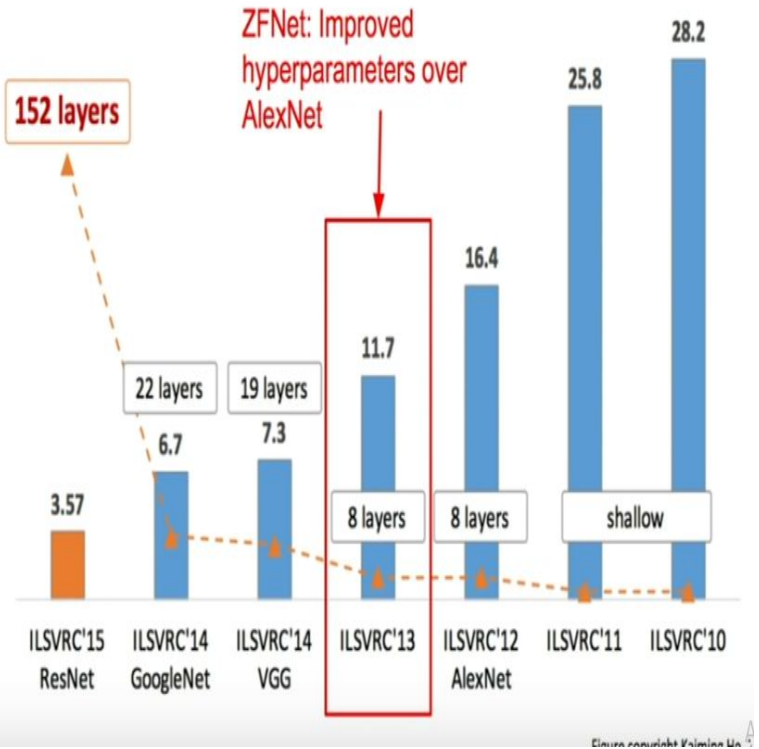
Google



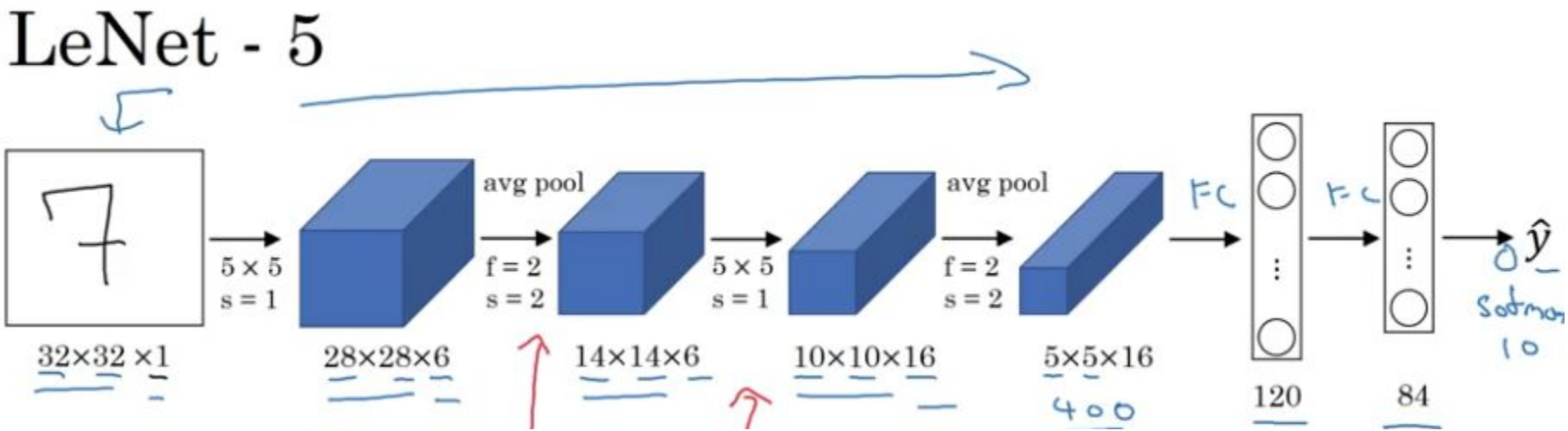
- Resnet (152 layers) in 2015 by “kaiming He” as a part of Microsoft team currently at Facebook AI

Timeline ..


First CNN-based winner
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10


ZFNet: Improved hyperparameters over AlexNet
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10

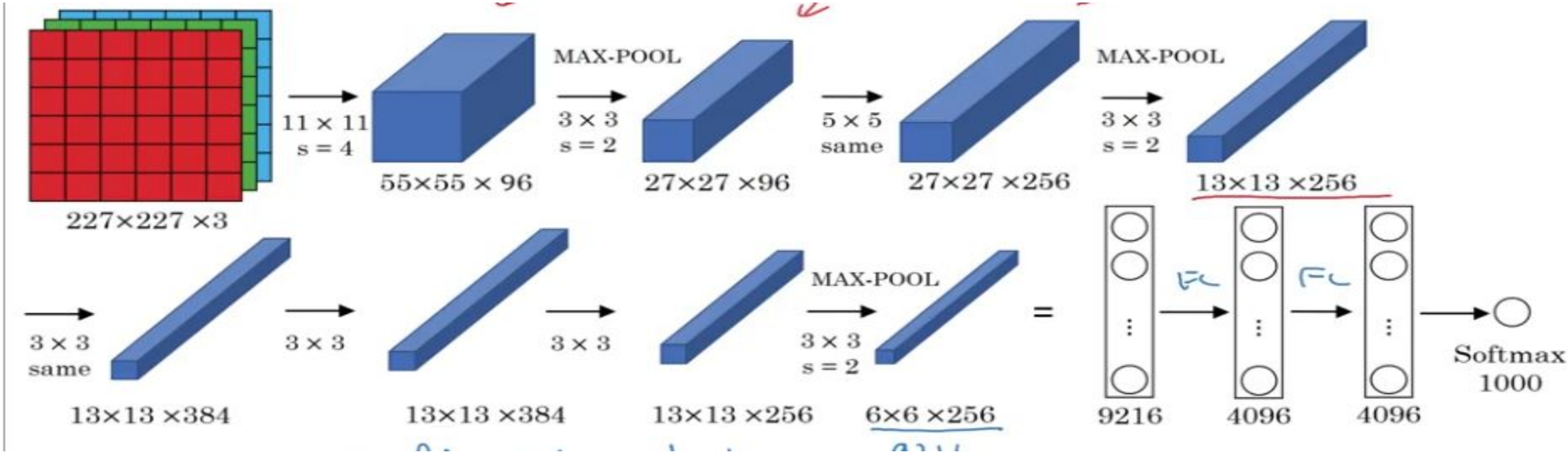
Lenet



Key features :

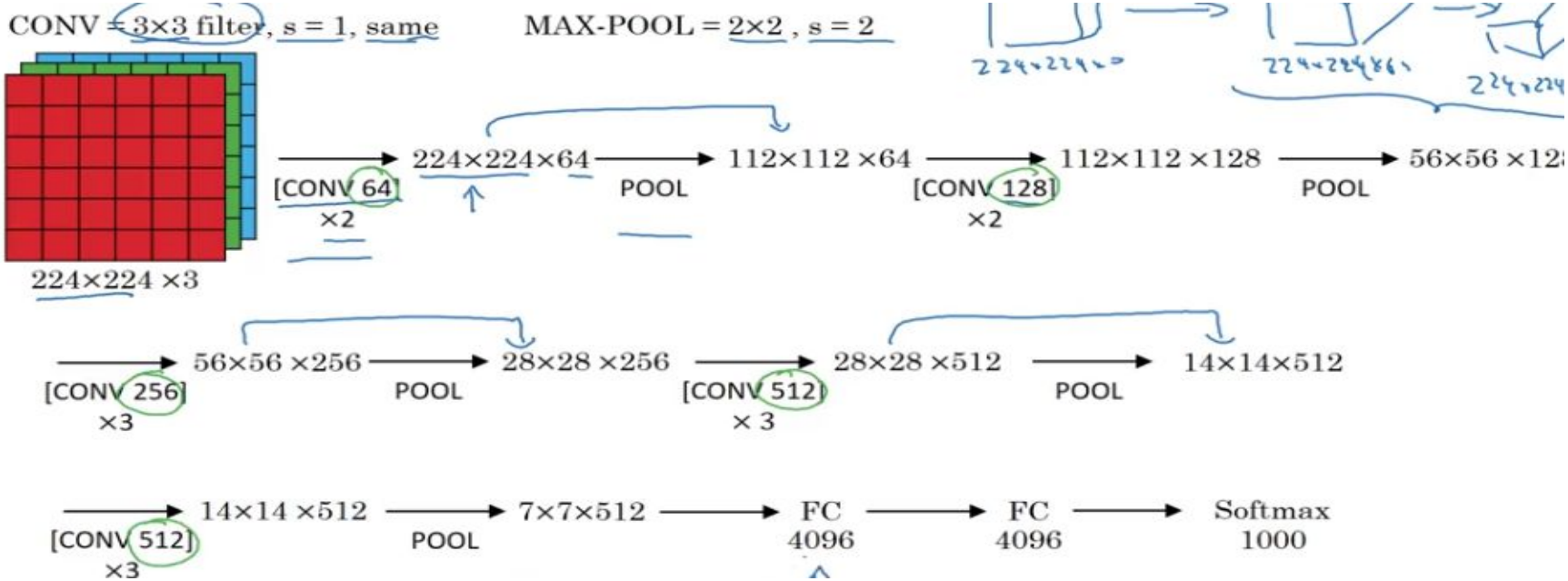
- Convolution in different way by skipping few channels in depth
- 5 layers (3 conv and 2 FC)
- Used on Handwritten numbers for cheques

Alexnet (named after Alex Krizhevsky)



- Deeper than Lenet and more parameters 60 million
- Relu was used
- Training on multiple GPUs in parallel
- Local Response Normalization (LRU) : to disable highly activated neurons but no such big improvement found later

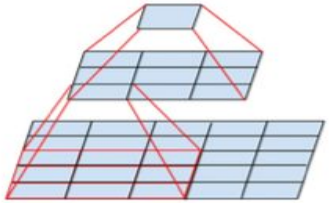
VGG - 16



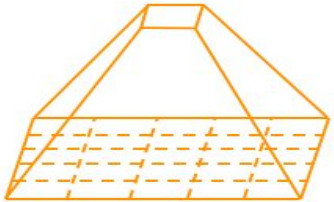
Key Features :

- Consistent and deeper network with multiple filters of smaller size
- H & W decreases while Depth increases as we go deeper in the network bcz of multiple filter used

Receptive Field



two successive
3x3 convolutions

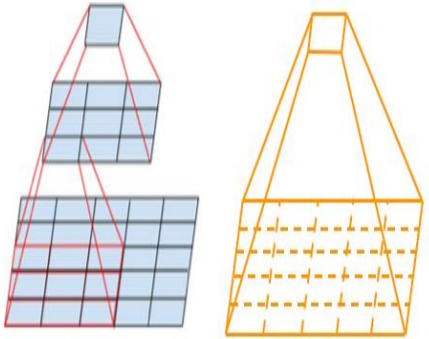


5x5 convolution

Whereas a $5 \times 5 \times c$ filter requires $25c$ parameters, two $3 \times 3 \times c$ filters only require $18c$ parameters. In order to most accurately represent a 5x5 filter, we shouldn't use any nonlinear activations between the two 3x3 layers. However, it was discovered that "linear activation was always inferior to using rectified linear units in all stages of the factorization."

- **Receptive field (**area covered under a filter**)**of one (7*7) filter and 3 (3*3) filter is same
- It will form a pyramid shape because of padding for the second filter will happen on edges(3*3,5*5,7*7)
- Less parameter (3*9*3) vs (1*49*3)

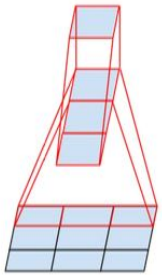
Receptive Field cont..



two successive
3x3 convolutions

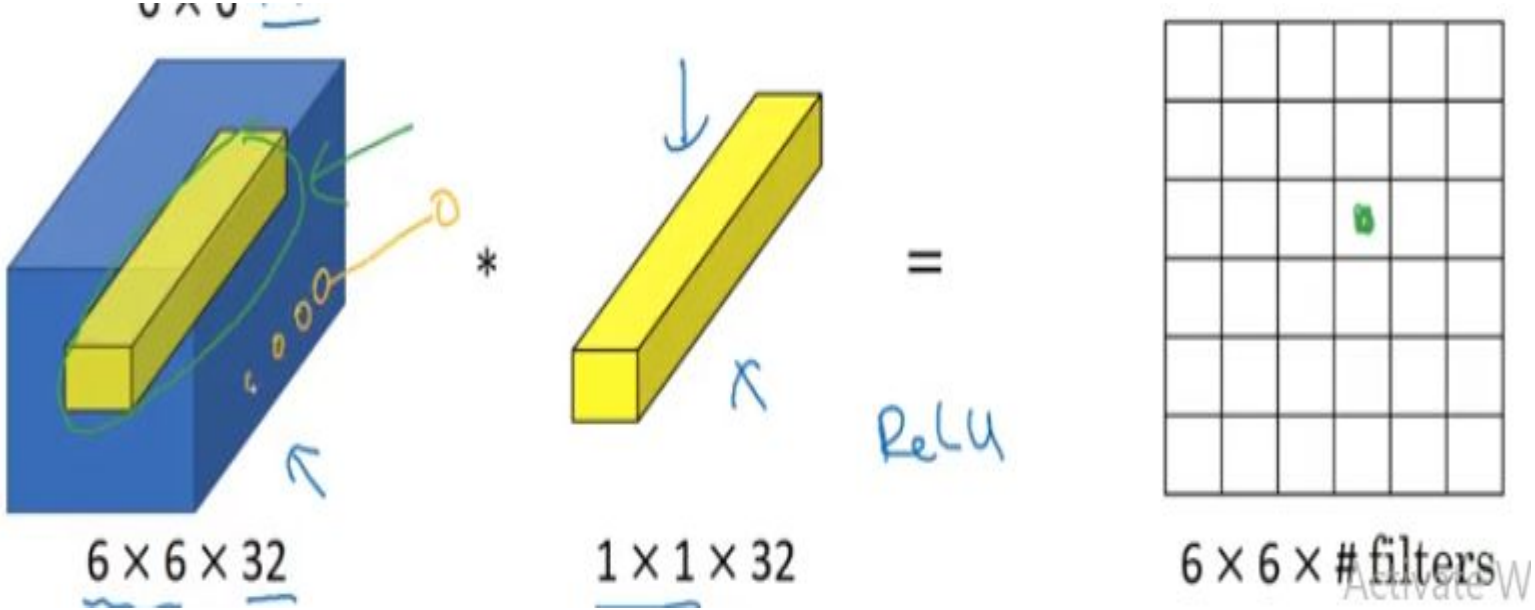
5x5 convolution

It was also shown that 3x3 convolutions could be further deconstructed into successive 3x1 and 1x3 convolutions.



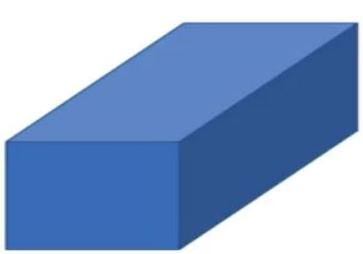
Generalizing this insight, we can more efficiently compute an $n \times n$ convolution as a $1 \times n$ convolution followed by a $n \times 1$ convolution.

1*1 Convolution (Network in Network)



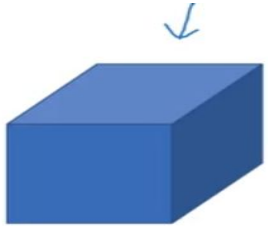
- To reduce image dimensions
- To make size of output layers same
- Used in FC layer also

1*1 Conv to reduce Parameters



28 × 28 × 192

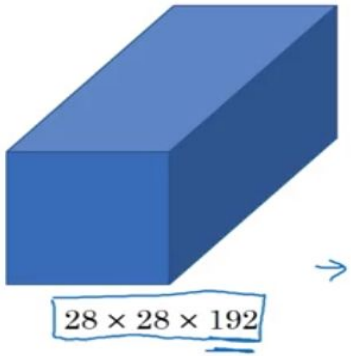
CONV
5 × 5,
same,
32



28 × 28 × 32

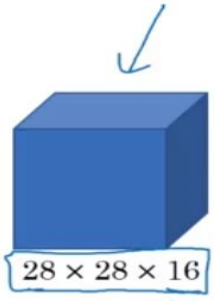
32 filters. filter one 5×5×192.

28×28×32 × 5×5×192 = 120M.



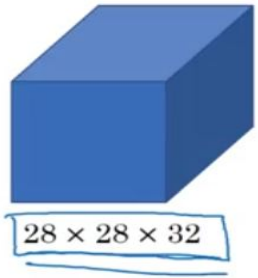
28 × 28 × 192

CONV
1 × 1,
16,
1 × 1 × 192



28 × 28 × 16

CONV
5 × 5,
32,
5 × 5 × 16



28 × 28 × 32

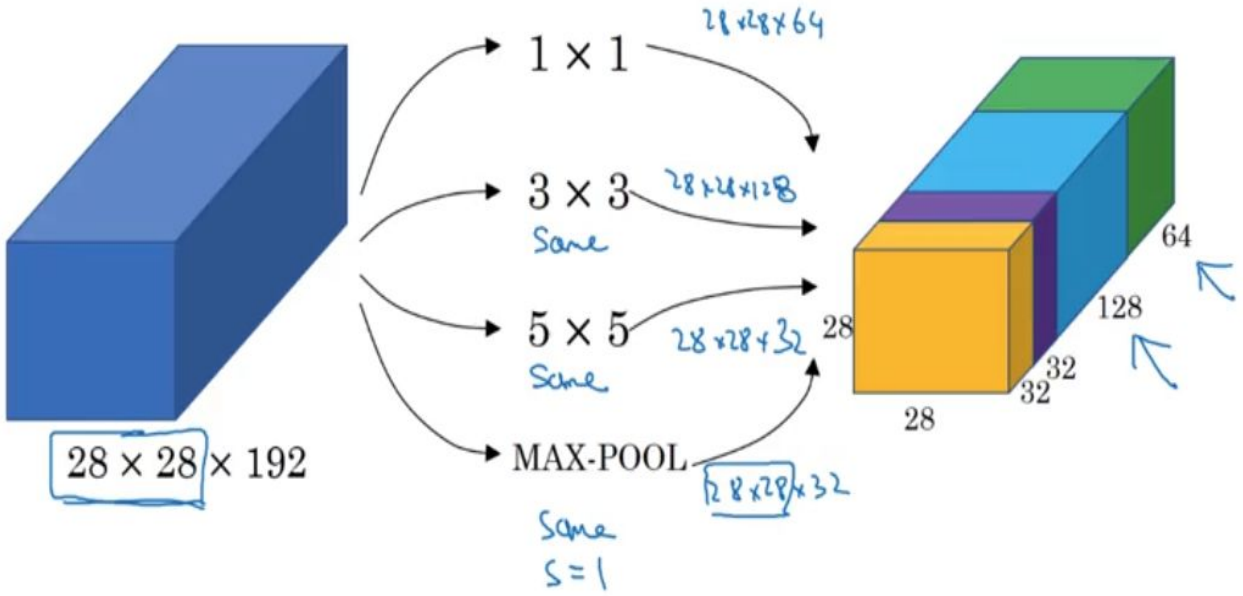
28×28×16 × 192 = 2.4M

28×28×32 × 5×5×16 = 10.0M

12.4M

120M

Inception Block

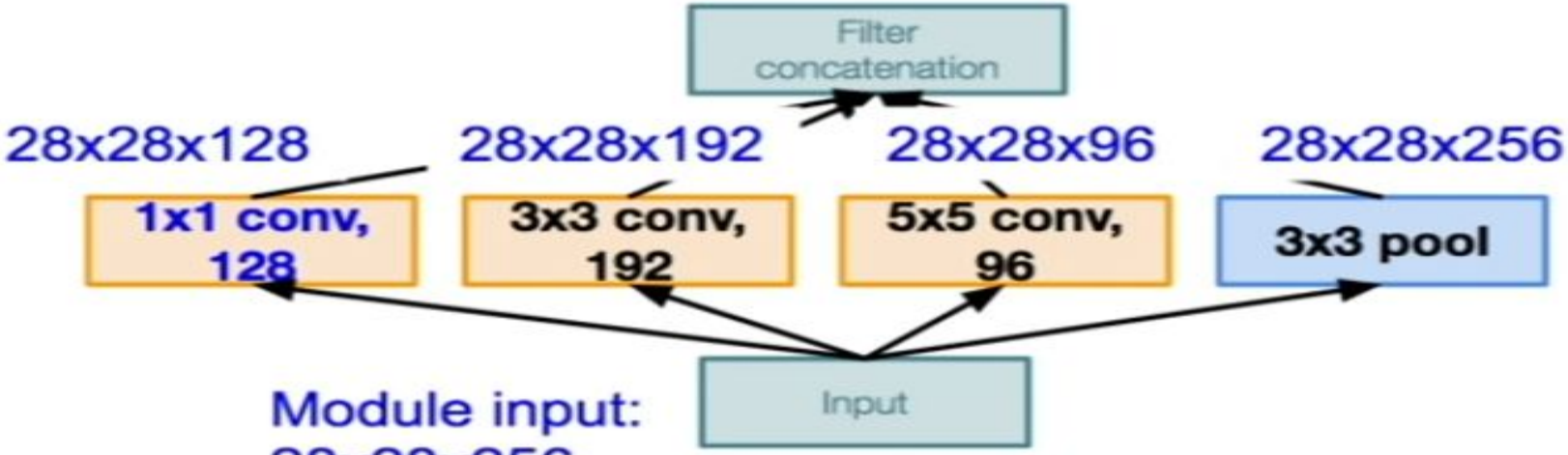


- Same convolution applied to keep dimensions same
- Features from between layers also passed as input to output layer(features learned from in between layers)

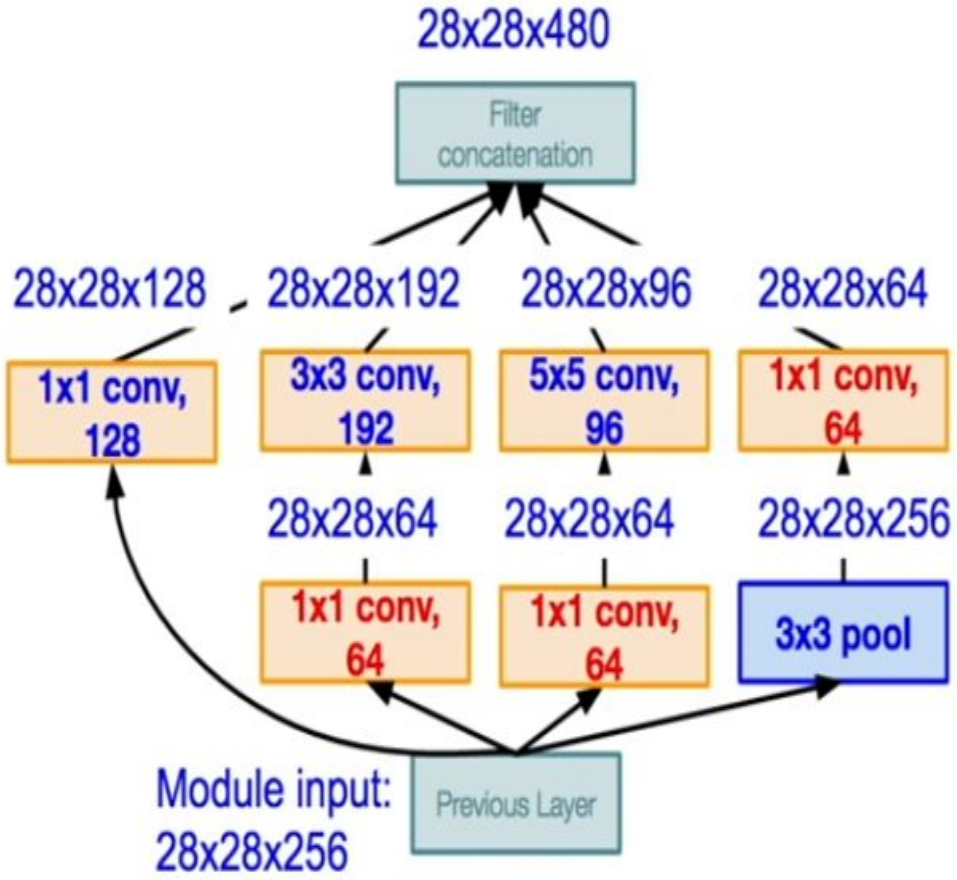
Example:

Q3:What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Naive Inception module



Inception module with dimension reduction

Conv Ops:

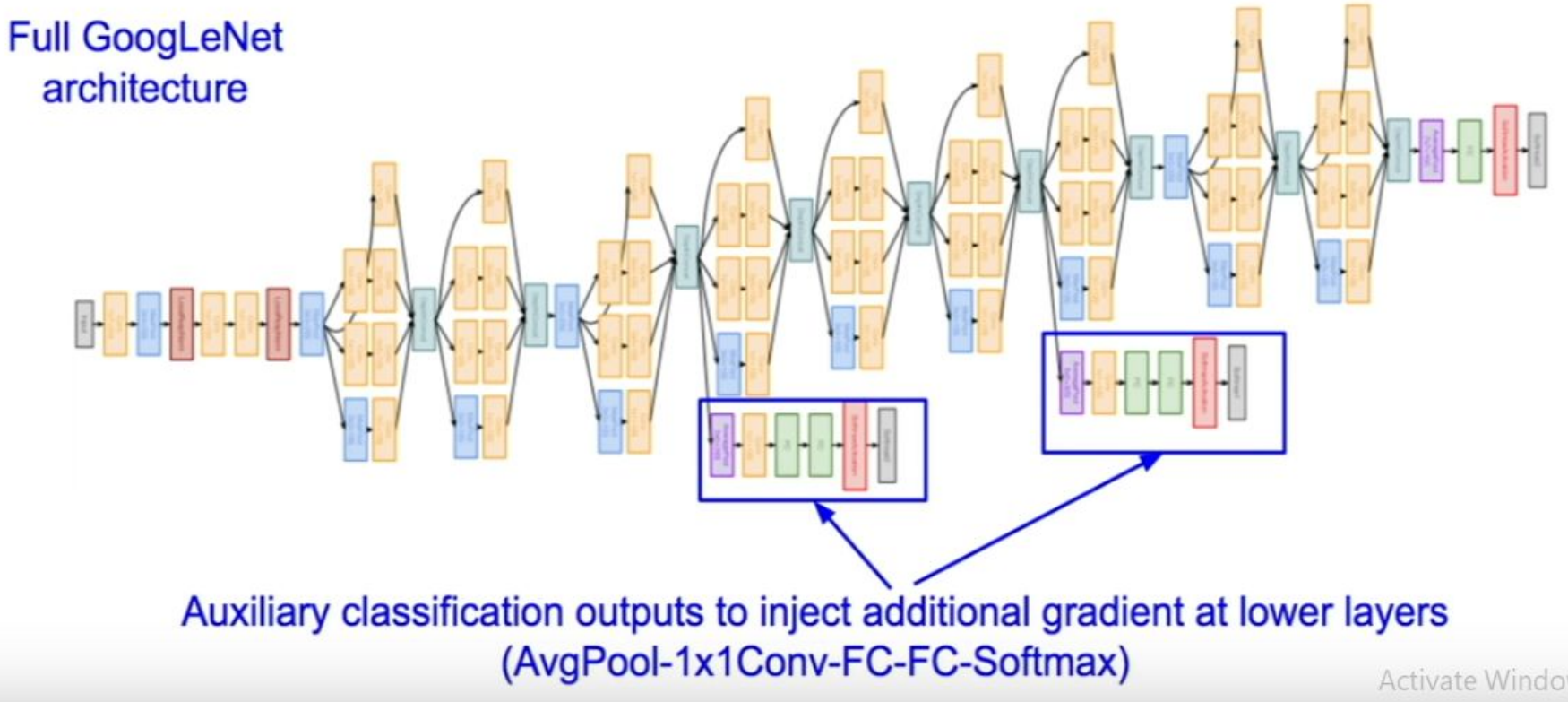
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

Compared to 854M ops for naive version

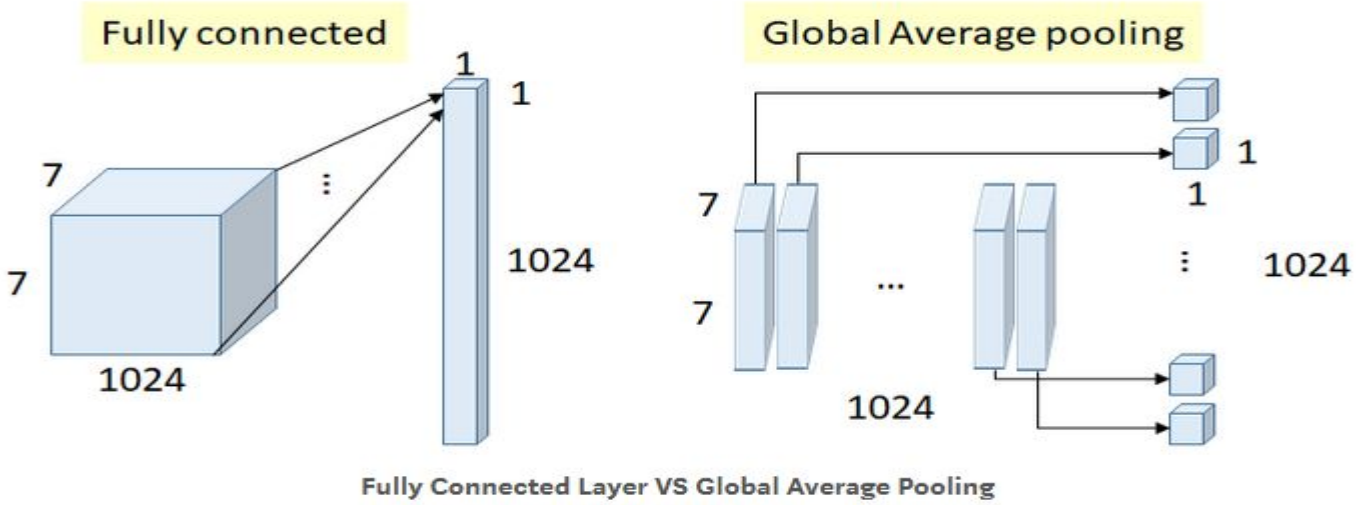
Bottleneck can also reduce depth after pooling layer

Auxiliary Output



- Add features of initial layers and improves gradient for backprop
- Reduce overfitting also (regularization)

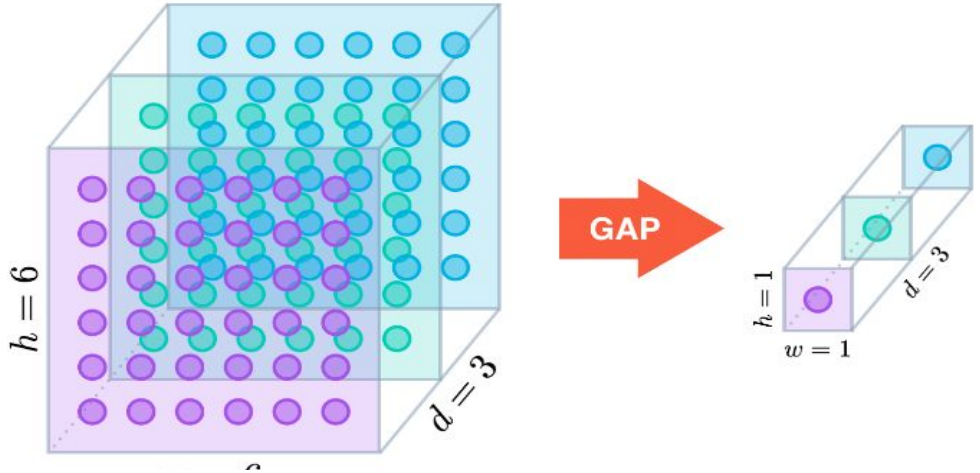
Global Average Pooling

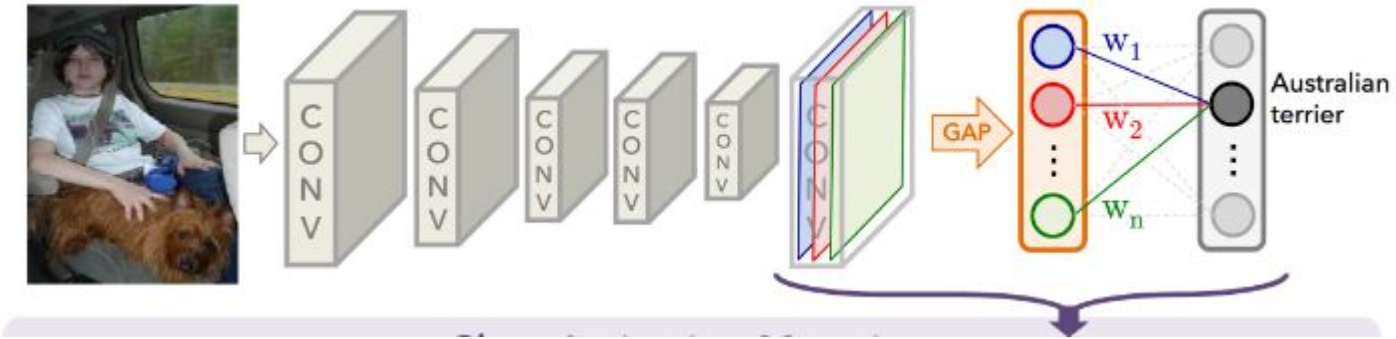


Fully Connected Layer VS Global Average Pooling

- GAP is mainly used to optimize the FC layer which we earlier done by Flattening the last conv layer
- Irrespective of size of output vector take average on each channel and generate a single number
- In that way size of layer will always remain same
- Like (7*7*128) => take average of 7*7 and generate vector of 128 size
- (5*5*128) => take average of 5*5 and it will also generate vector of size 128 , so it helps in keeping the output dimensions same

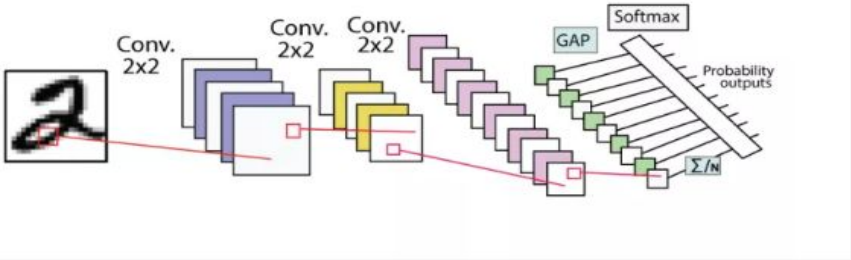
GAP cont..

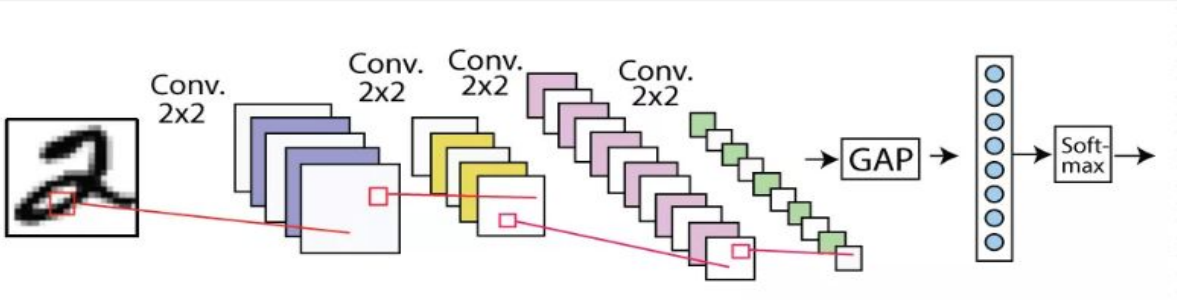




- GAP is used in most of network now to handle image of different dimesions

GAP cont..





- GAP can be used directly as last layer in Network or we can place few FC layer(by using 1*1 conv) as well after that

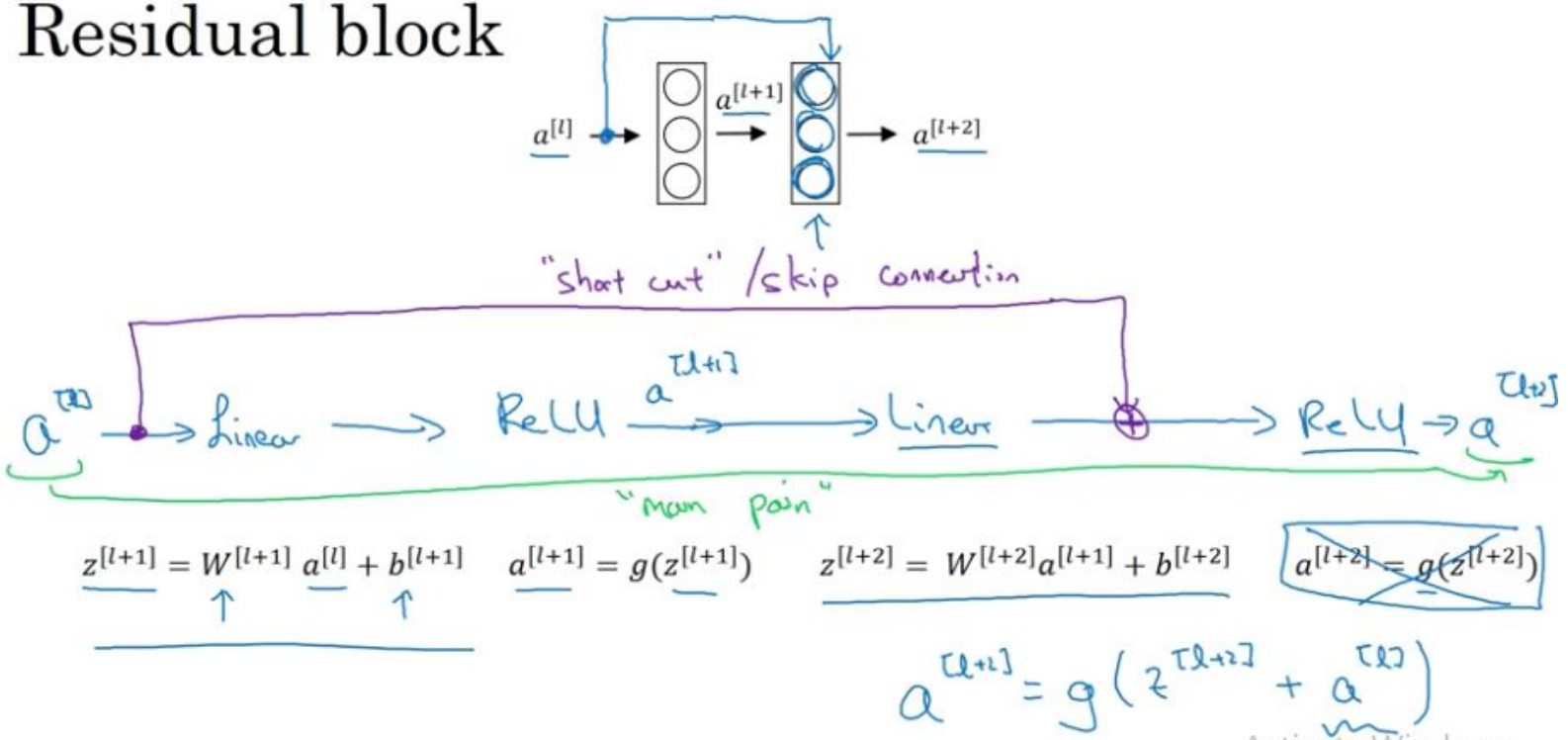
Resnet

- If network with N layers performs P , then network with N+1 should not perform less than P
- If above case was fine, then what was the problem in training deep networks!!
- Problem with deep network is **gradient vanishing and exploding**
- Residual block helps in training the deeper network
- Resnet is just a layered architecture of these residual blocks
- Based on the concept that **direct mappings** are hard to learn , so instead of that make the network learn the **residue** only
- It also helps in backprop by handling vanishing gradient problem

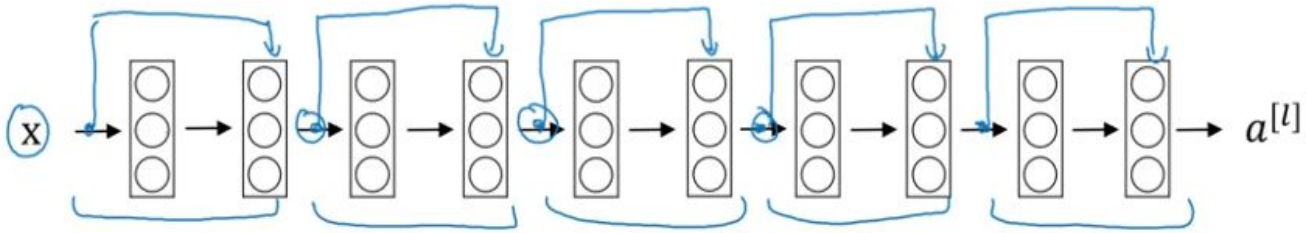
Gradient Vanishing /Exploding

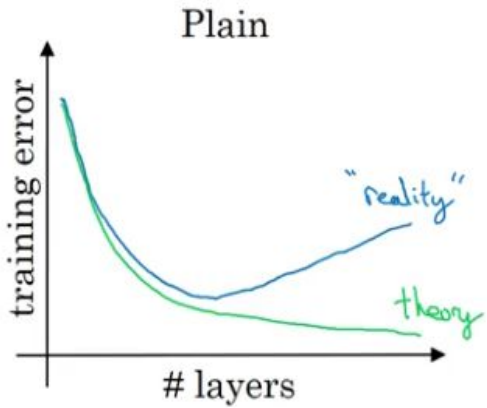
- Gradient which is calculated during backprop to adjust the weights and make network learn
- Gradient is calculated at time of backprop and weight are adjusted w.r.t gradient
- During backprop gradient becomes too small which makes negligible impact on the weights of initial layers and this is **Gradient vanishing and** vice versa
- Initial layers are important bcz they learn the basic features of NN and act as a building block to the network
- Sigmoid and Tanh activations are not used bcz of that now

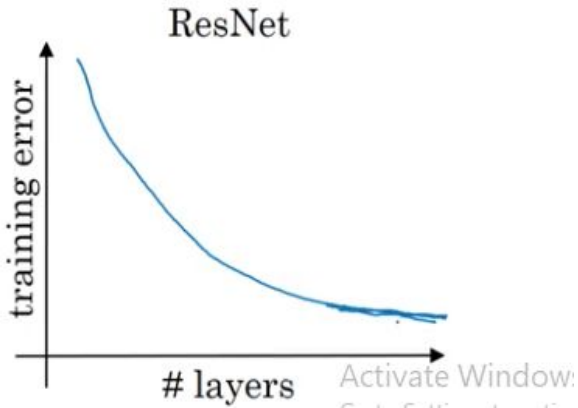
Residual Block



Plain Net V/S Resnet

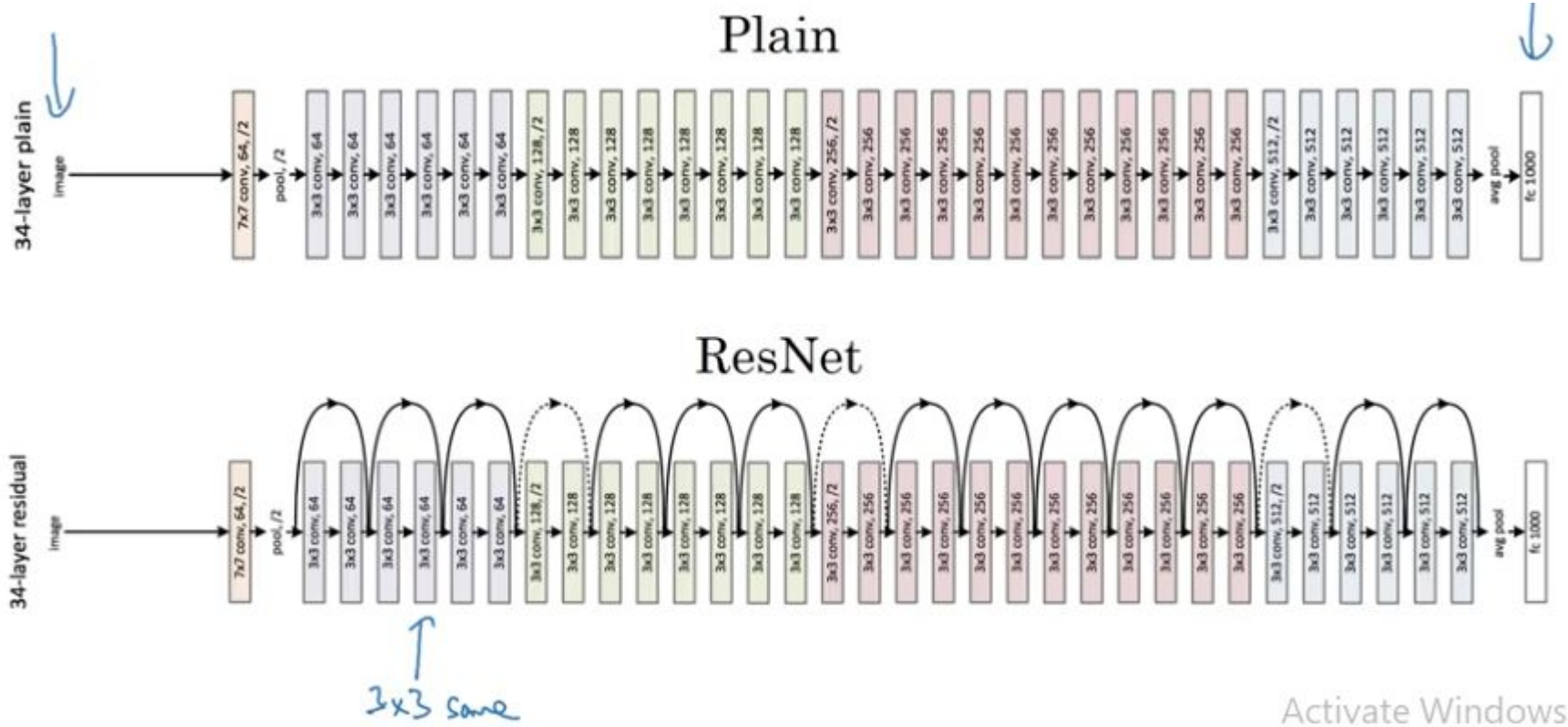






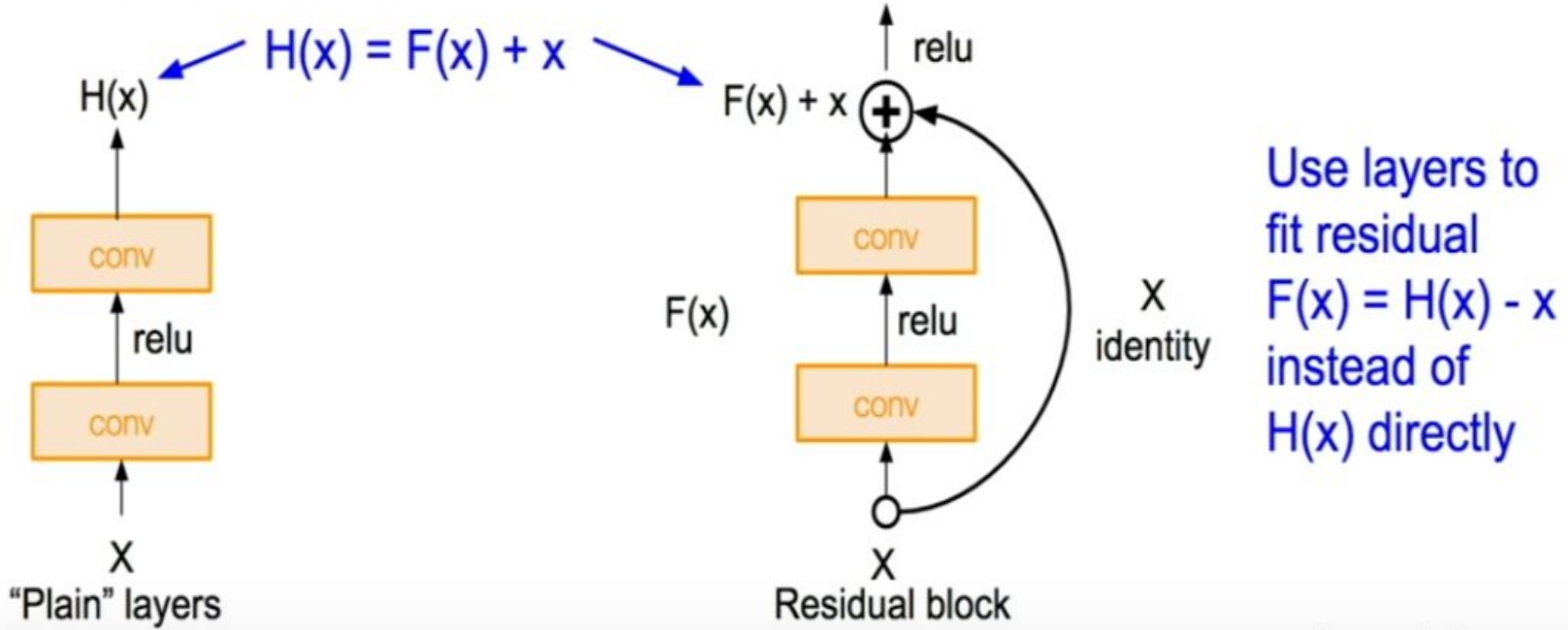
- Plain network with skip connection is Resnet
- In practical error increases with deeper network but not in Resnet

Resnet Architecture



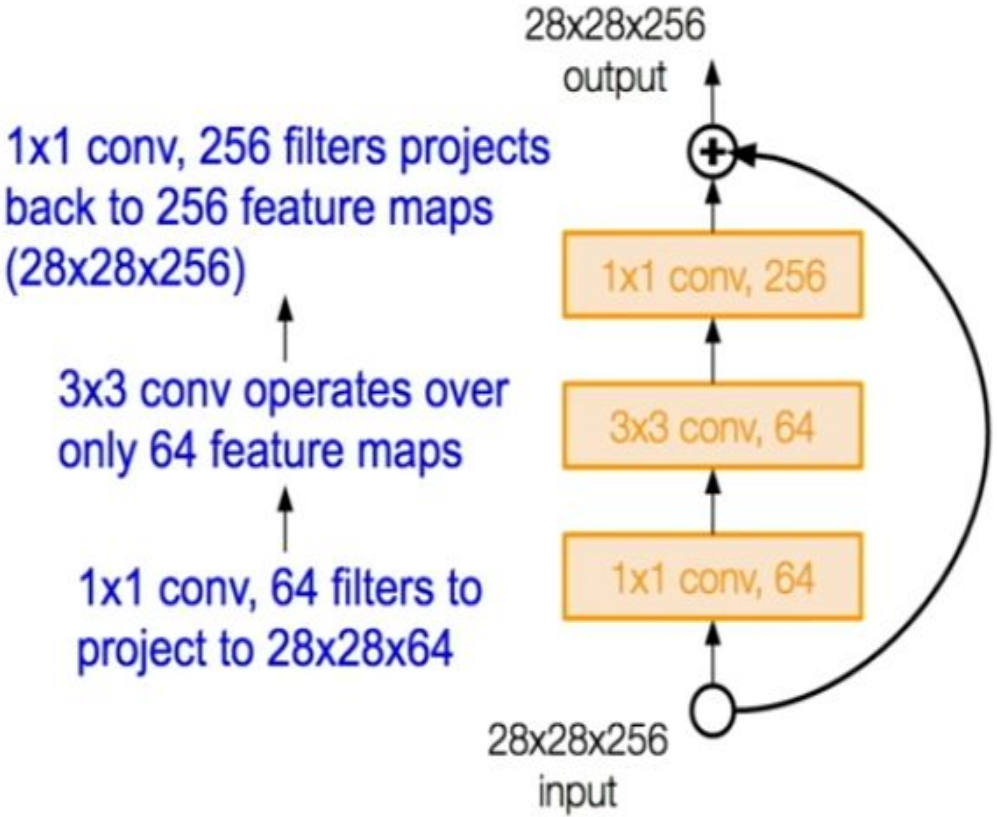
How to train a deep network

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

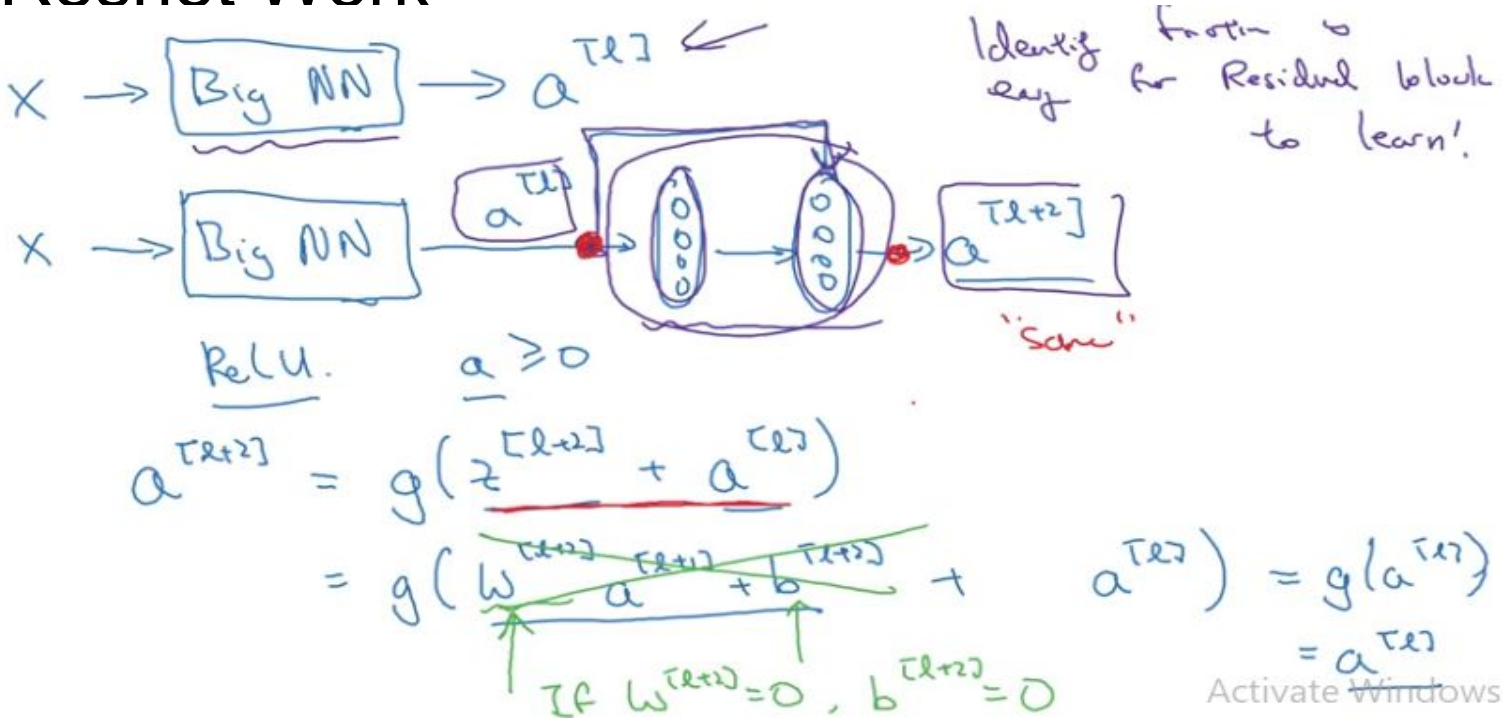


Bottleneck with residual block

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



How Resnet Work

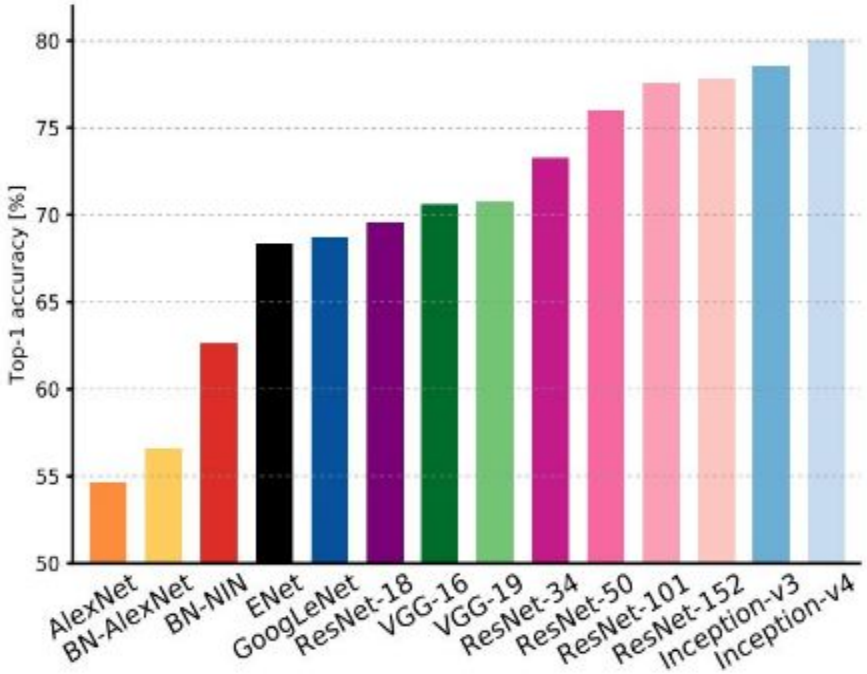


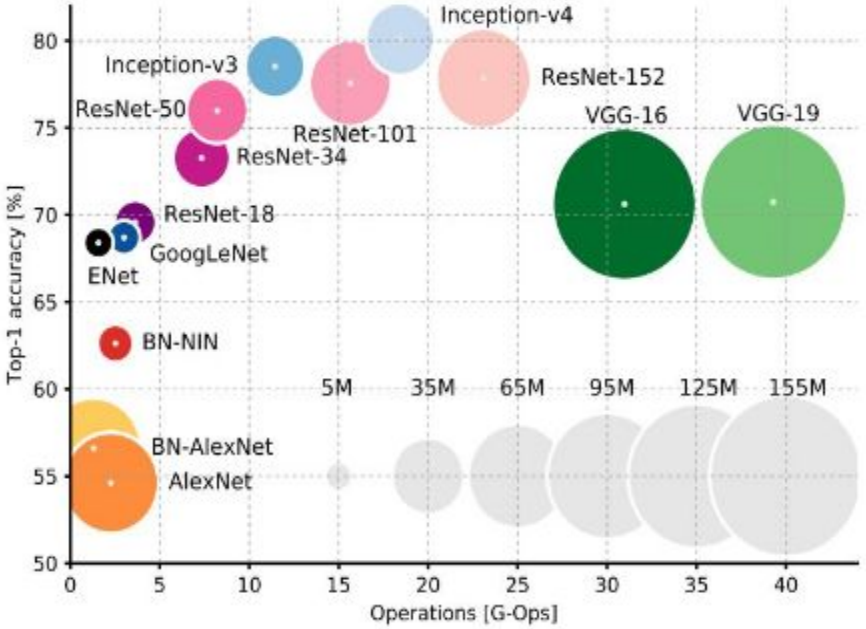
- By learning Identity function , as a[l] is residue in above case
- Same convolution is used to add the residue in further layers
- Residual block improves training by stopping the deeper layers from degrading the training

Overview

Year	CNN	Developed By	Error rates	No. of parameters
1998	LeNet	Yann LeCun et al		60 thousand
2012	AlexNet	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	15.3%	60 million
2013	ZFNet	Matthew Zeiler, Rob Fergus	14.8%	
2014	GoogLeNet	Google	6.67%	4 million
2014	VGGNet	Simonyan, Zisserman	7.3%	138 million
2015	ResNet	Kaiming He	3.6%	

Overview Cont..





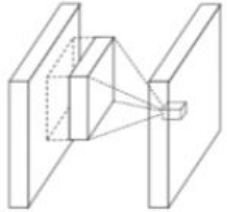
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Other Networks (NiN)

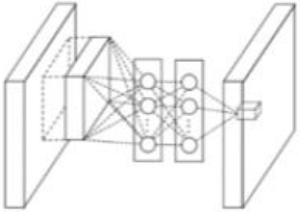
Network in Network (NiN)

[Lin et al. 2014]

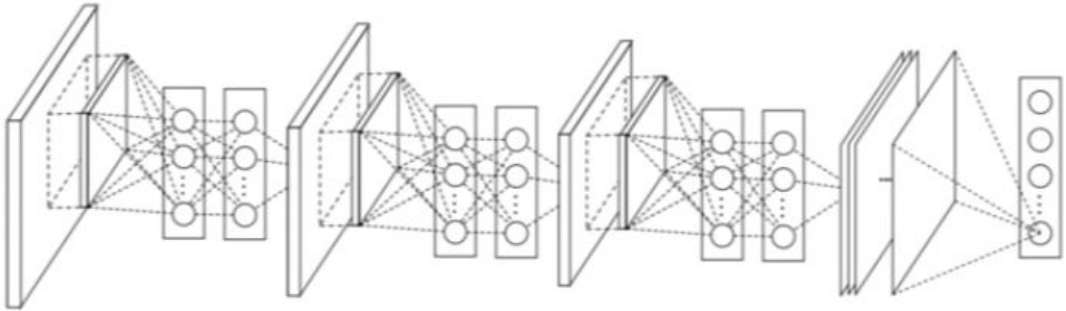
- Mlpconv layer with "micronetwork" within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet "bottleneck" layers
- Philosophical inspiration for GoogLeNet



(a) Linear convolution layer

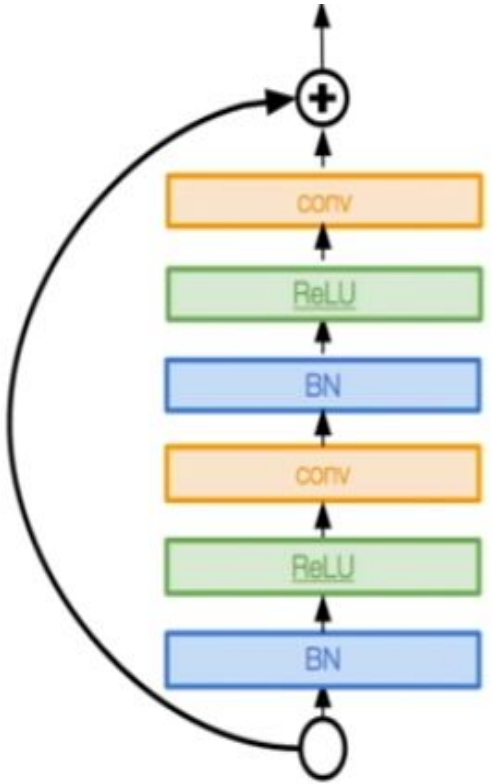


(b) Mlpconv layer



Improved version of Resnet (Residual block)

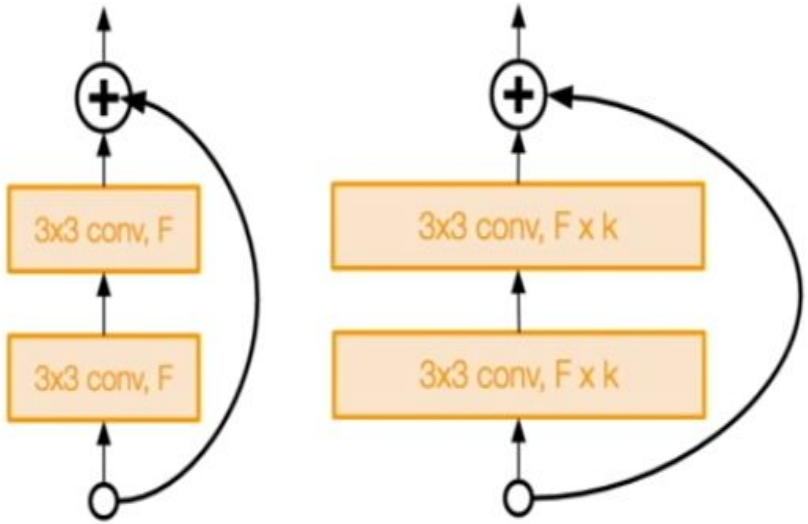
- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance



Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



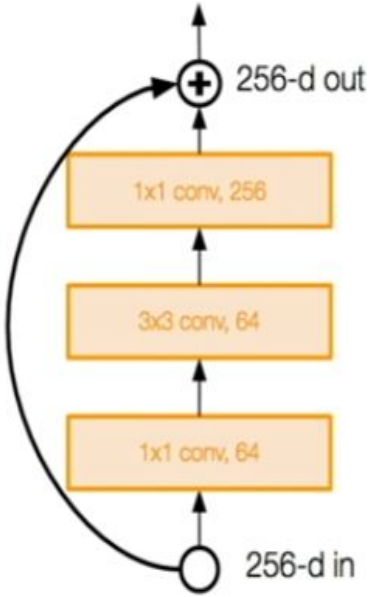
Basic residual block

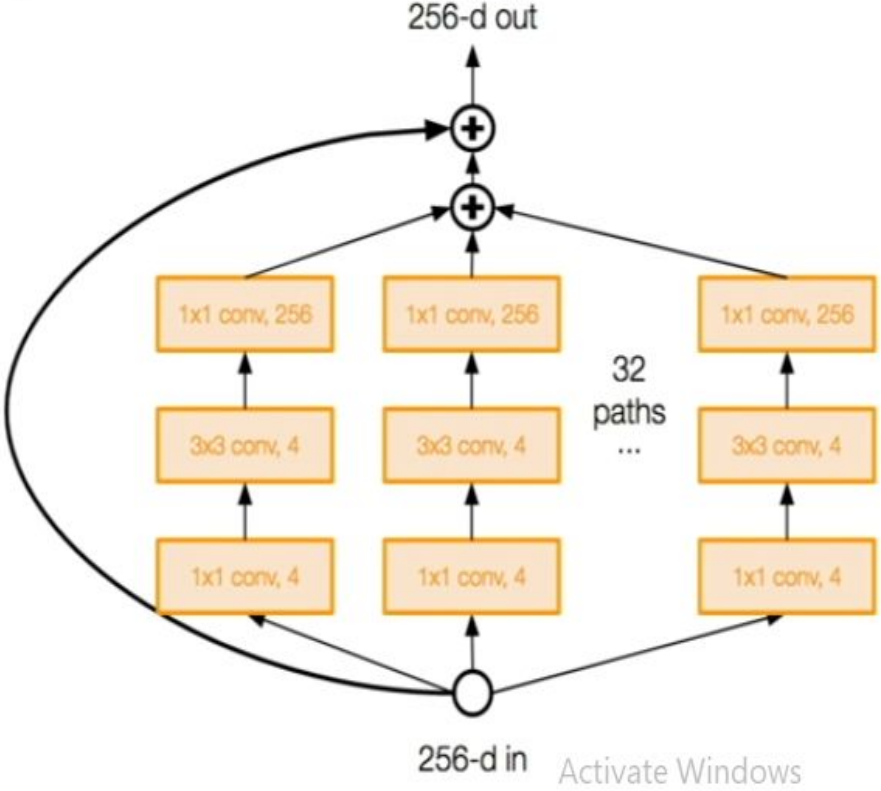
Wide residual block

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module

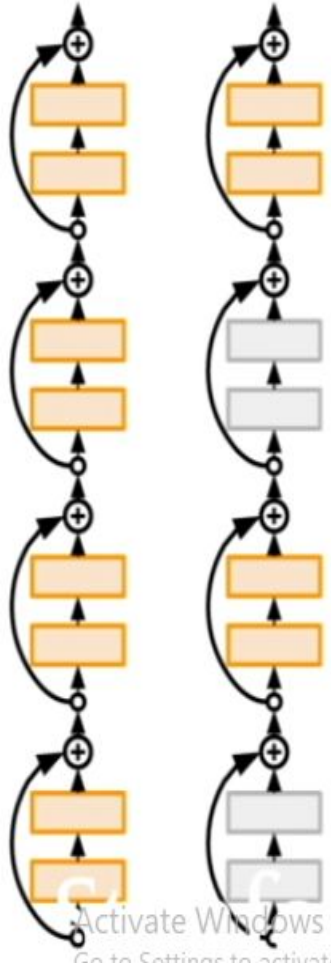




Deep Networks with Stochastic Depth

[Huang et al. 2016]

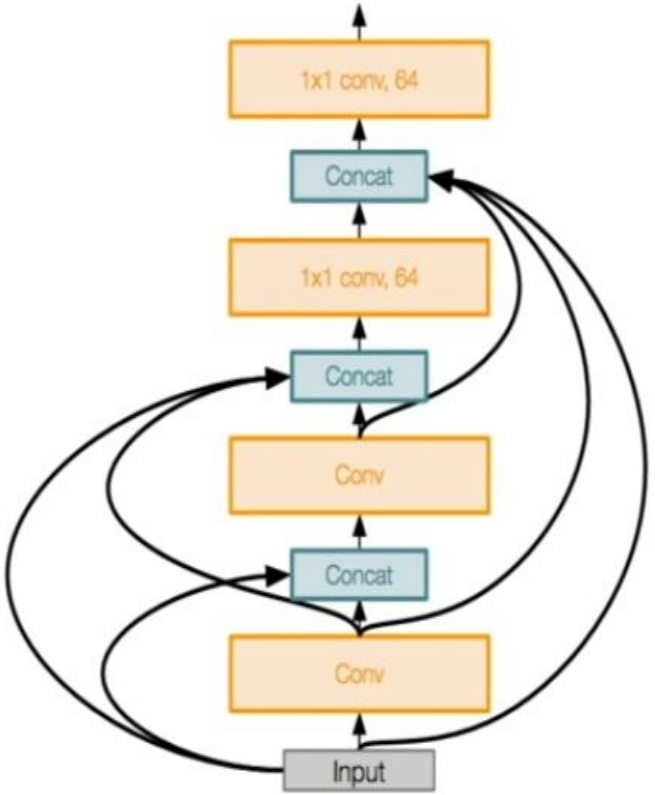
- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse

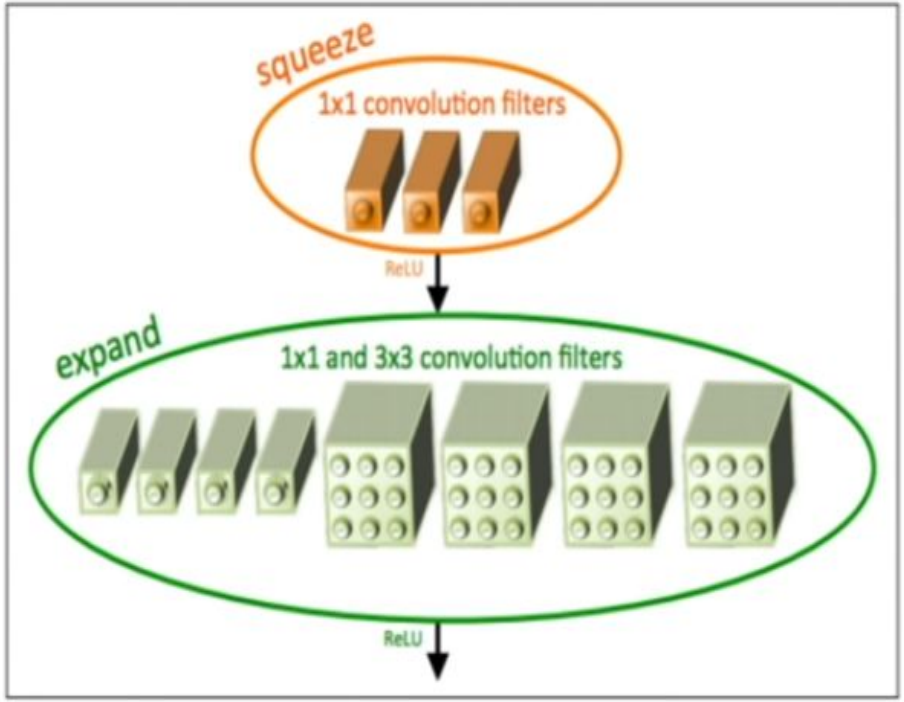


Dense Block

SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size

[Iandola et al. 2017]

- Fire modules consisting of a 'squeeze' layer with 1x1 filters feeding an 'expand' layer with 1x1 and 3x3 filters
- AlexNet level accuracy on ImageNet with 50x fewer parameters
- Can compress to 510x smaller than AlexNet (0.5Mb)



FractalNet: Ultra-Deep Neural Networks without Residuals

[Larsson et al. 2017]

- Argues that key is transitioning effectively from shallow to deep and residual representations are not necessary
- Fractal architecture with both shallow and deep paths to output
- Trained with dropping out sub-paths
- Full network at test time