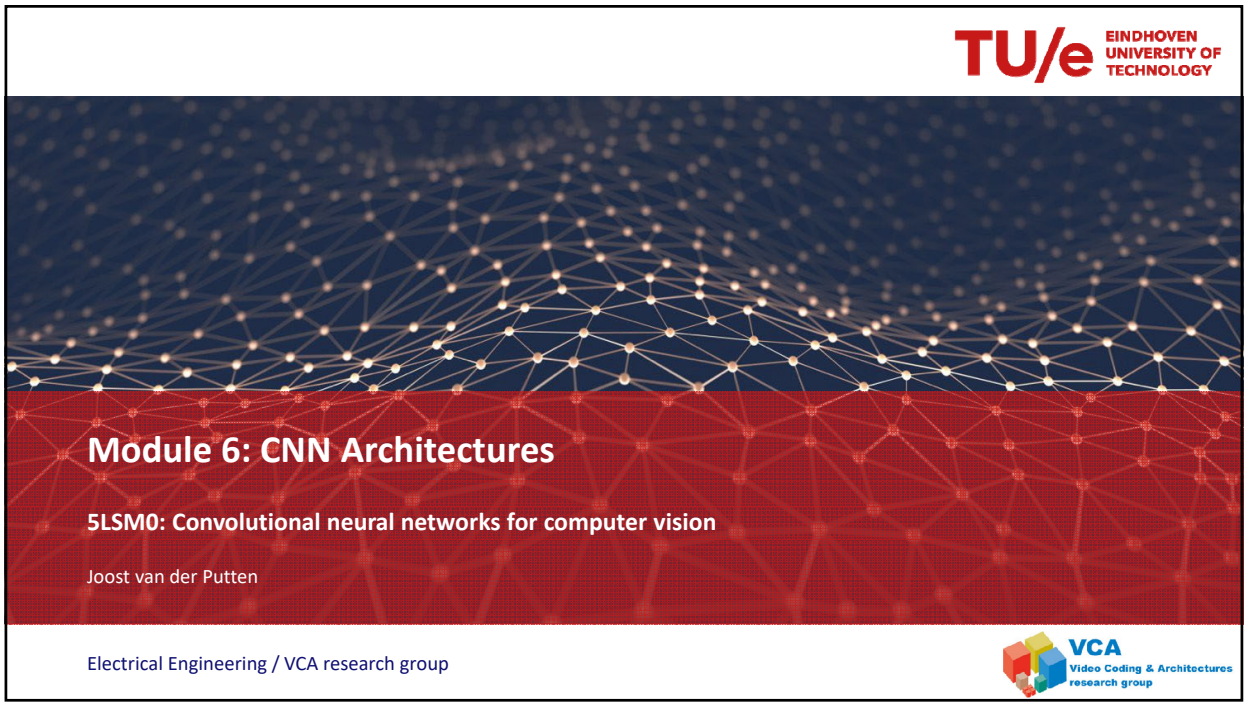


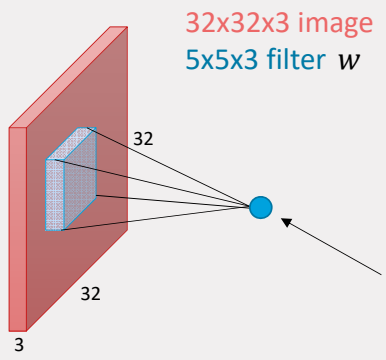
Module 6: CNN Architectures

5LSM0: Convolutional neural networks for computer vision

Joost van der Putten

Electrical Engineering / VCA research group

Last time – convolutional layer

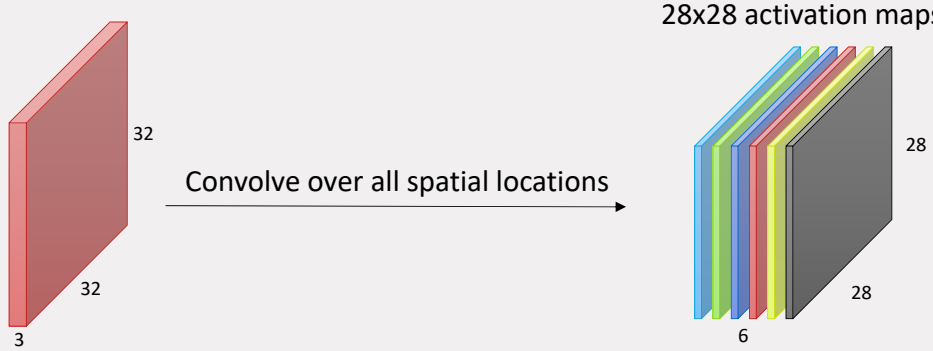


1 number:
The result of taking a dot product between the filter and a small 5x5x3 chunk of the image

$$w^T x + b$$

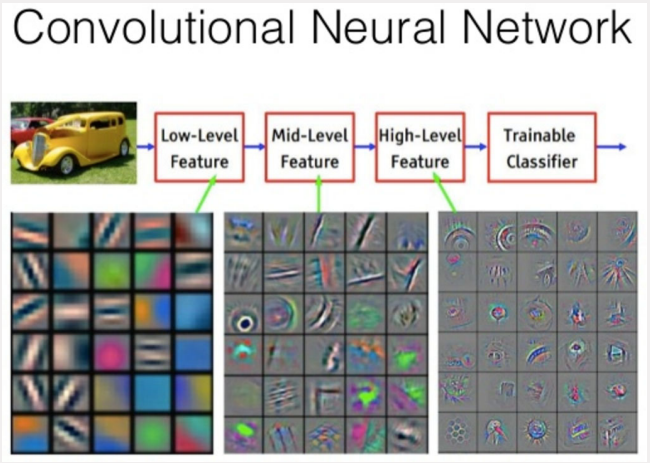
Last time – convolutional layer

With 6 5x5 filters, we obtain 6 28x28 activation maps

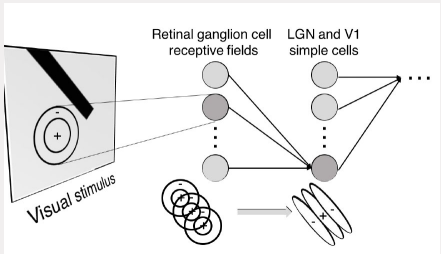


Stack the activation maps to obtain a "new image" of size 28x28x6

Last time – ConvNets

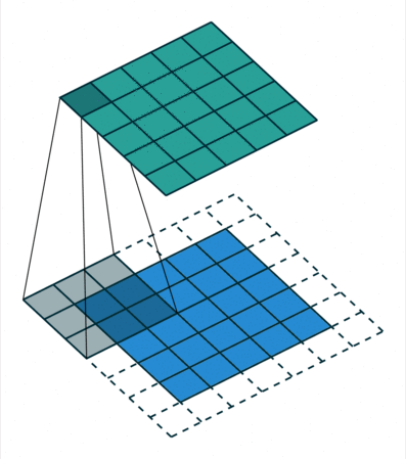


Hubel & Wiesel



Convolutional neural network learns hierarchical structure on its own

Last time – convolutional layers

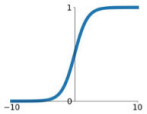


- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters **K**
  - Their spatial extent **F**
  - The stride **S**
  - The amount of padding **P**
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
  - $H_2 = \frac{H_1 - F + 2P}{S} + 1$
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume the **d**-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the **d**-th filter over the input volume with a stride of S, and then offset by **d**-th bias.

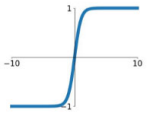
Last time – Activation functions

Activation Functions

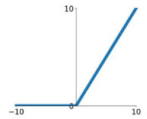
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



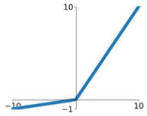
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

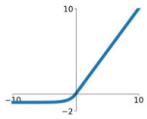


Leaky ReLU
$\max(0.1x, x)$



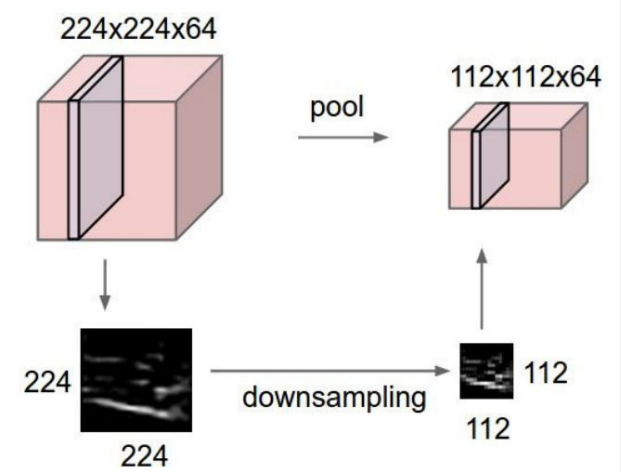
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Last time - Pooling layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



Module outline

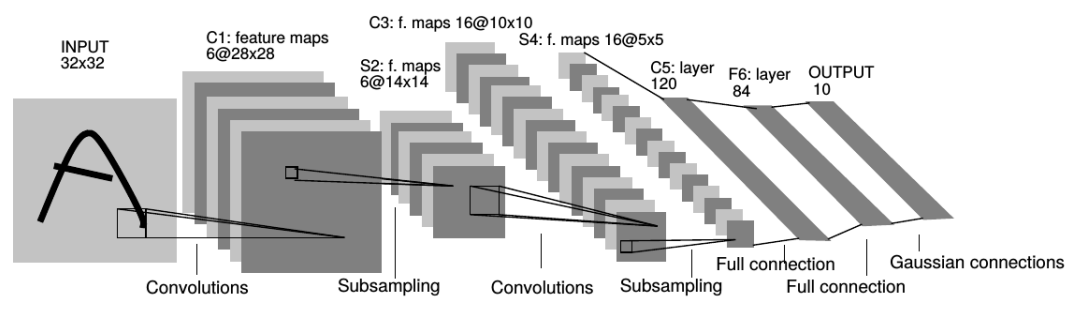
Case studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also…

- Network in Network
- Wide ResNet
- ResNeXT
- DenseNet
- FractalNet
- Stochastic Depth
- Squeeze-and-Exitation network
- SqueezeNet

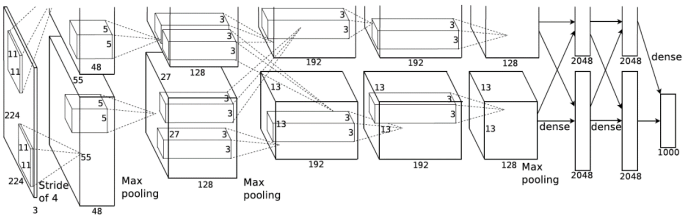
Review: LeNet-5



Architecture: CONV-POOL-CONV-POOL-FC-FC
Filters were 5x5, applied at stride 1

Case study: AlexNet

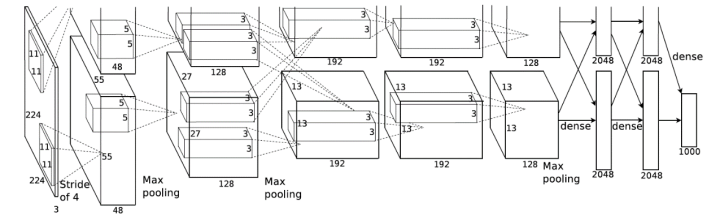
[Krizhevsky et al. 2012]



Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM1
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

Case study: AlexNet

[Krizhevsky et al. 2012]



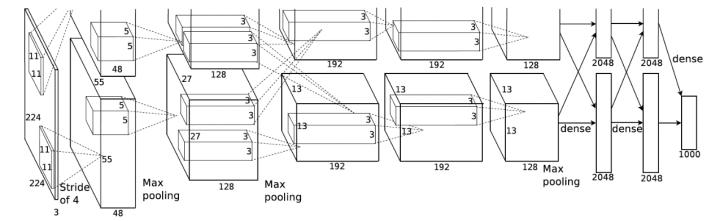
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

Q: what is the output volume size? Hints: (227-11)/4+1=55

Case study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

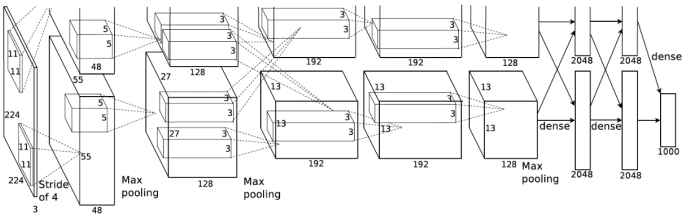
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

Parameters: (11*11*3)*96=**35k**

Case study: AlexNet

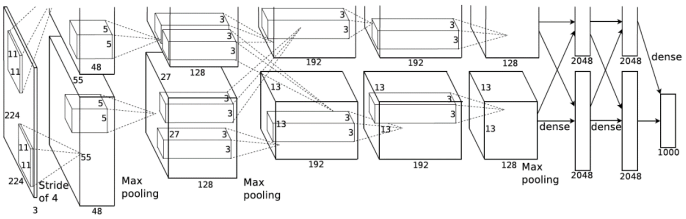
[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96
Second layer (Pool1): 3x3 filters applied at stride 2
Output volume **[55x55x96]**
Q: what is the output volume size? Hint: (55-3)/2+1=27

Case study: AlexNet

[Krizhevsky et al. 2012]

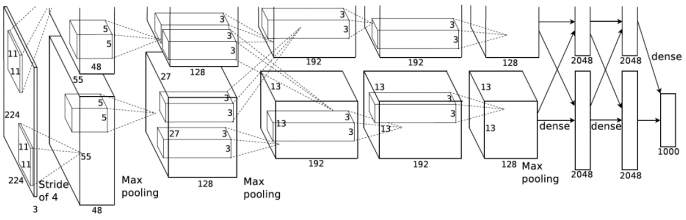


Input: 227x227x3 images
After CONV1: 55x55x96
Second layer (Pool1): 3x3 filters applied at stride 2
Output volume **[27x27x96]**
Q: What is the number of parameters?

Parameters: 0!

Case study: AlexNet

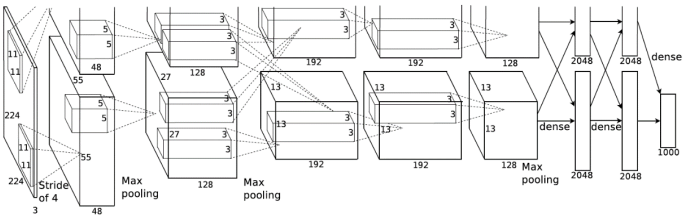
[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96
After POOl1: 27x27x96
....

Case study: AlexNet

[Krizhevsky et al. 2012]

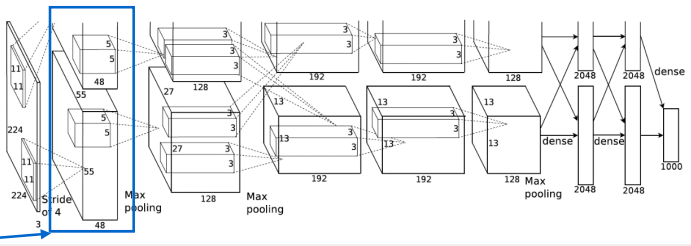


Architecture:
[227x227x3] INPUT
[55x55x96] CONV1 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1 3x3 filters at stride 2
[27x27x256] NORM1 Normalization layer
[13x13x256] CONV2 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2 3x3 filters at stride 2
[13x13x256] NORM1 Normalization layer
[13x13x384] CONV3 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3 3x3 filters at stride 2
[4096] FC6 4096 neurons
[4096] FC7 4096 neurons
[1000] FC8 4096 neurons

Details
- First use of ReLU
- Used Norm layers (not common anymore)
- Heavy data augmentation
- Dropout 0.5
- Batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by factor 10 manually when validation accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Case study: AlexNet

[Krizhevsky et al. 2012]



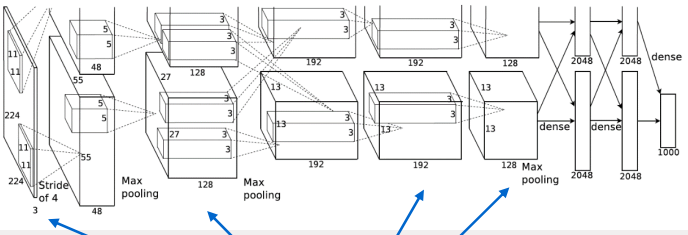
[55x55x48] x 2

Architecture:
[227x227x3] INPUT
[55x55x96] ~~CONV1~~ 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1 3x3 filters at stride 2
[27x27x256] NORM1 Normalization layer
[13x13x256] CONV2 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2 3x3 filters at stride 2
[13x13x256] NORM1 Normalization layer
[13x13x384] CONV3 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3 3x3 filters at stride 2
[4096] FC6 4096 neurons
[4096] FC7 4096 neurons
[1000] FC8 4096 neurons

Note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

Case study: AlexNet

[Krizhevsky et al. 2012]

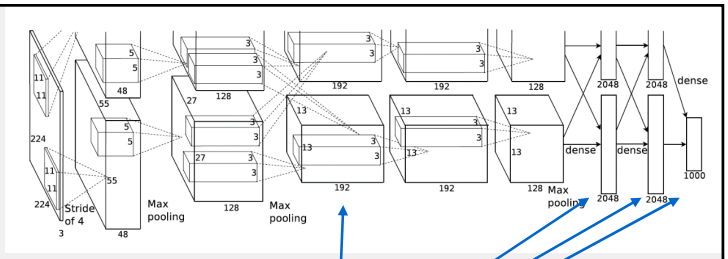


Architecture:
[227x227x3] INPUT
[55x55x96] CONV1 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1 3x3 filters at stride 2
[27x27x256] NORM1 Normalization layer
[13x13x256] CONV2 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2 3x3 filters at stride 2
[13x13x256] NORM1 Normalization layer
[13x13x384] CONV3 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3 3x3 filters at stride 2
[4096] FC6 4096 neurons
[4096] FC7 4096 neurons
[1000] FC8 4096 neurons

CONV1, CONV2, CONV4, CONV5: Connections only with feature maps on same GPU

Case study: AlexNet

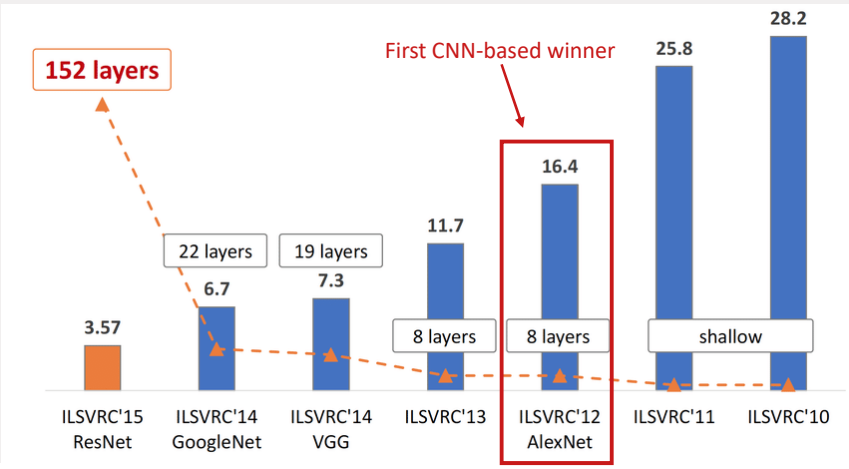
[Krizhevsky et al. 2012]



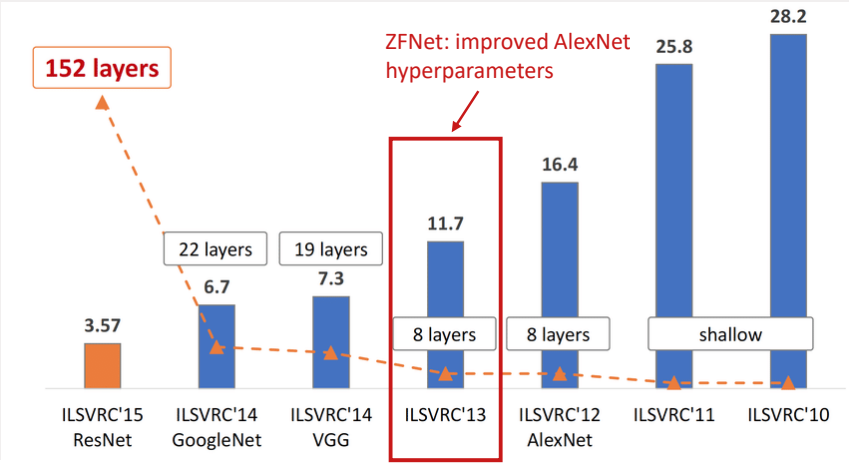
Architecture:
[227x227x3] INPUT
[55x55x96] CONV1 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1 3x3 filters at stride 2
[27x27x256] NORM1 Normalization layer
[13x13x256] CONV2 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2 3x3 filters at stride 2
[13x13x256] NORM1 Normalization layer
[13x13x384] CONV3 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3 3x3 filters at stride 2
[4096] FC6 4096 neurons
[4096] FC7 4096 neurons
[1000] FC8 4096 neurons

CONV3, FC6, FC7, FC8: Connections with all feature maps in preceding layer, communication across GPUs

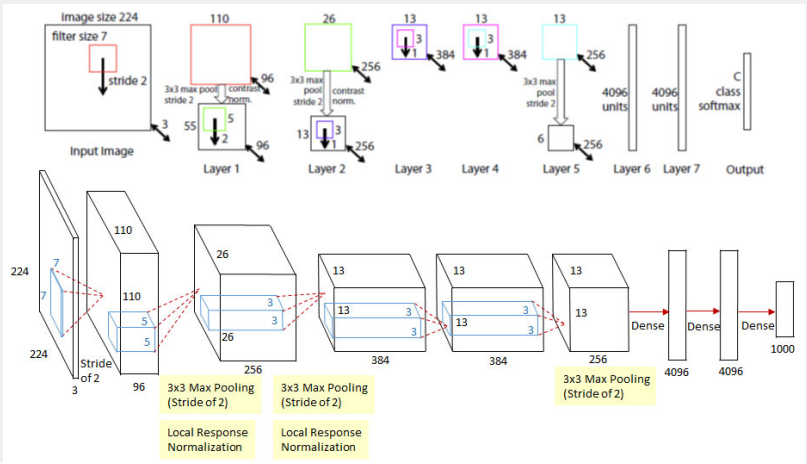
ImageNet Large Scale Visual Recognition Challenge winners



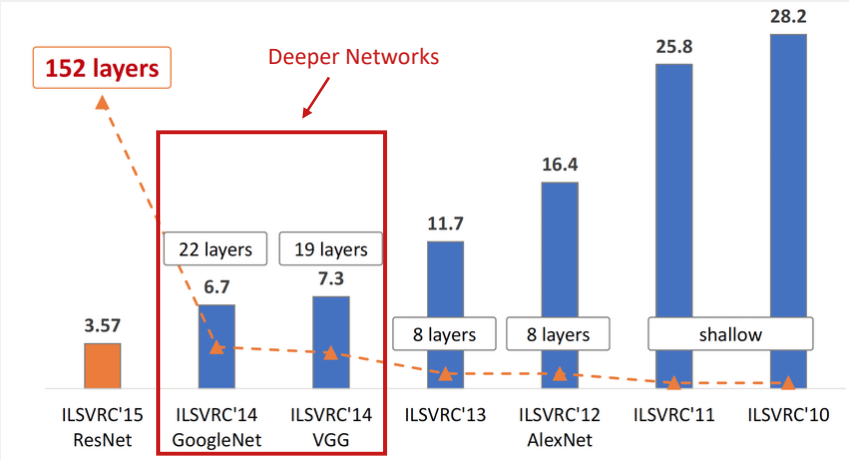
ImageNet Large Scale Visual Recognition Challenge winners



ZFNet



ImageNet Large Scale Visual Recognition Challenge winners



Case Study: VGG

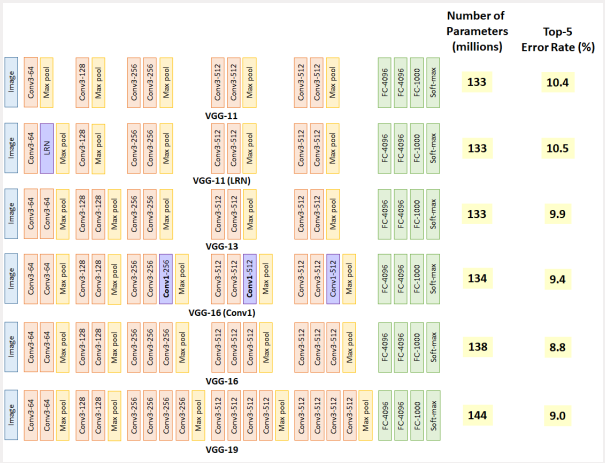
[Simonyan and Zisserman, 2014]

Smaller filters, deeper networks

From 8 layers (alexnet) to up to 19 layers (VGG19)

Only 3x3 CONV stride 1, pad 1 and 2x2 MaxPool stride 2

11.7% top 5 error in ILSVRC'13 (ZFNet) -> 7.3% top error in ILSVRC'14



Case Study: VGG

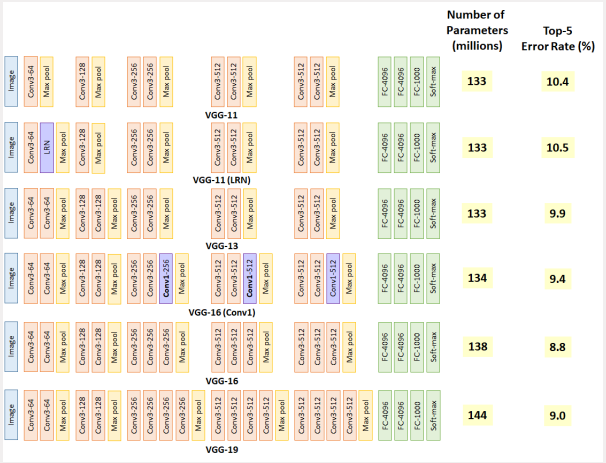
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters?

Stack of three 3x3 conv (stride 1) layers has the same **effective receptive field** as one 7x7 conv layer

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

[7x7]



Case Study: VGG

[Simonyan and Zisserman, 2014]

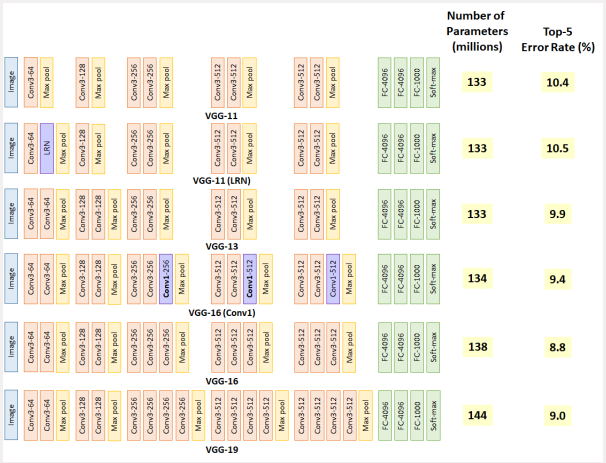
Q: Why use smaller filters?

Stack of three 3x3 conv (stride 1) layers has the same **effective receptive field** as one 7x7 conv layer but deeper, and has more non-linearities

Q: What about the number of parameters?

Fewer parameters as well:

$3 * \frac{(3^2C^2)}{7^2C^2} \approx$ half of the parameters



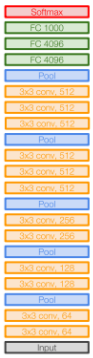
Case Study: VGG

[Simonyan and Zisserman, 2014]

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



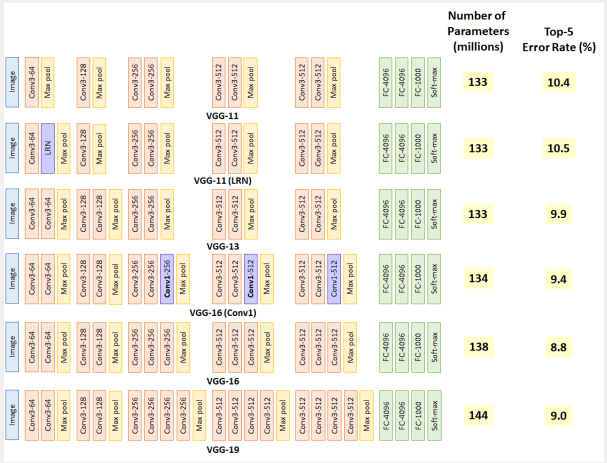
VGG16

Case Study: VGG

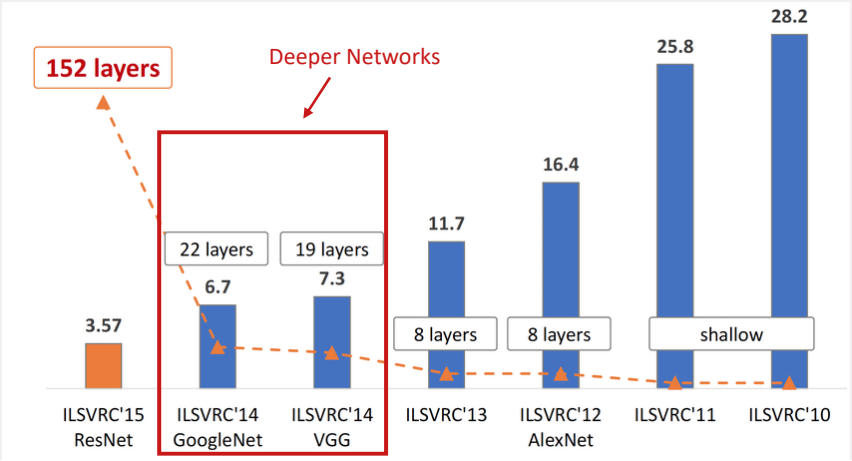
[Simonyan and Zisserman, 2014]

Details:

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky in 2012
- No local response normalization
- Use VGG16 or VGG19 (slightly better but uses more memory)
- FC features generalizes well to other tasks



ImageNet Large Scale Visual Recognition Challenge winners

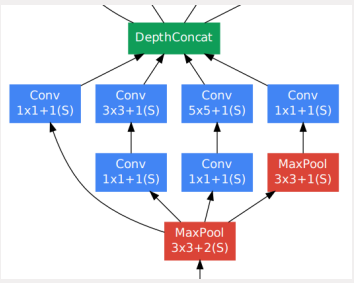


Case study: GoogLeNet

[Szegedy et al, 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient 'inception module
- No FC layers
- Only 5 million parameters! 12 times less than AlexNet
- ILSVRC'14 classification winner (6.7% top 5 error)

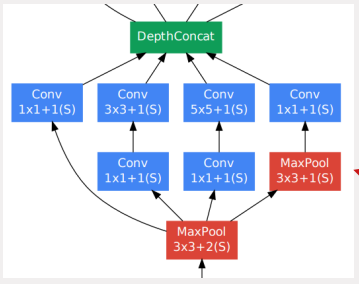


Inception module

Case study: GoogLeNet

[Szegedy et al, 2014]

'Inception module': design a good local network topology (network within a network) and then stack these modules on top of each other



Inception module

Case study: GoogLeNet

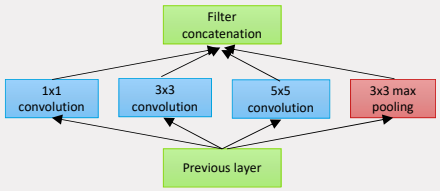
[Szegedy et al, 2014]

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolutions (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?
[Hint: Computational complexity]



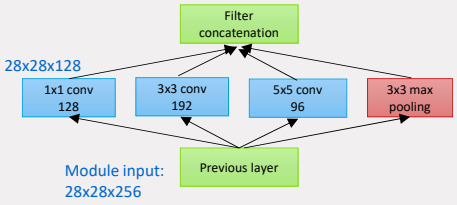
Naïve Inception module

Case study: GoogLeNet

[Szegedy et al, 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example: Q1: What is the output size of the 1x1 conv, with 128 filters?



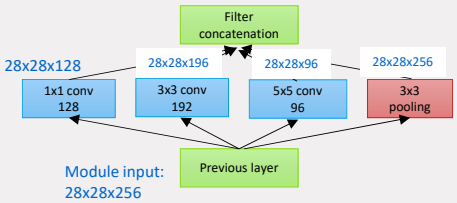
Naïve Inception module

Case study: GoogLeNet

[Szegedy et al, 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example: Q2: What about the other filters?



Naïve Inception module

Case study: GoogLeNet

[Szegedy et al, 2014]

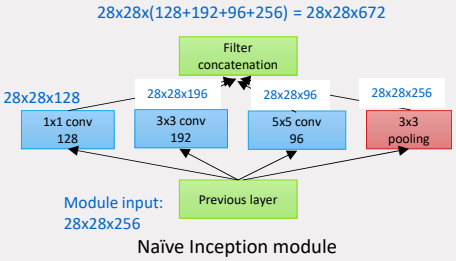
Q: What is the problem with this?
[Hint: Computational complexity]

Conv ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer

Example: Q2: What is the output size after filter concatenation?



Naïve Inception module

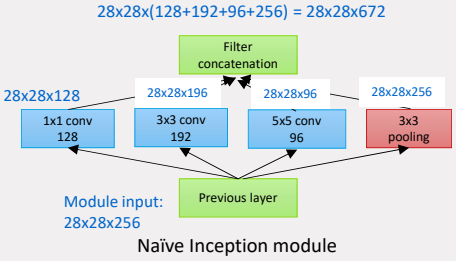
Case study: GoogLeNet

[Szegedy et al, 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

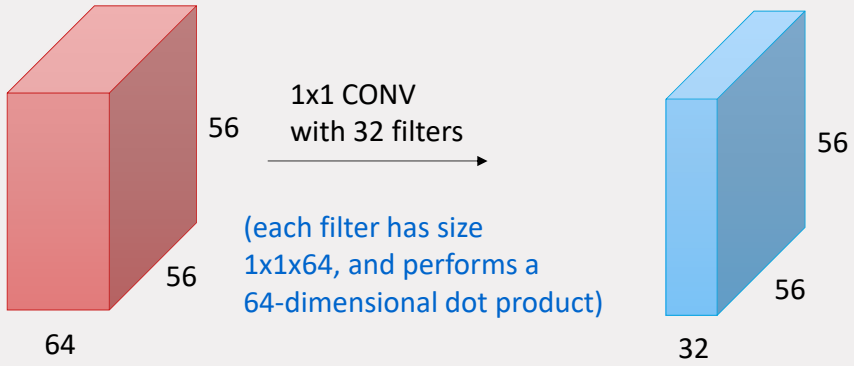
Solution: 'bottleneck layers that use 1x1 convolutions to reduce feature depth

Example: Q2: What is the output size after filter concatenation?



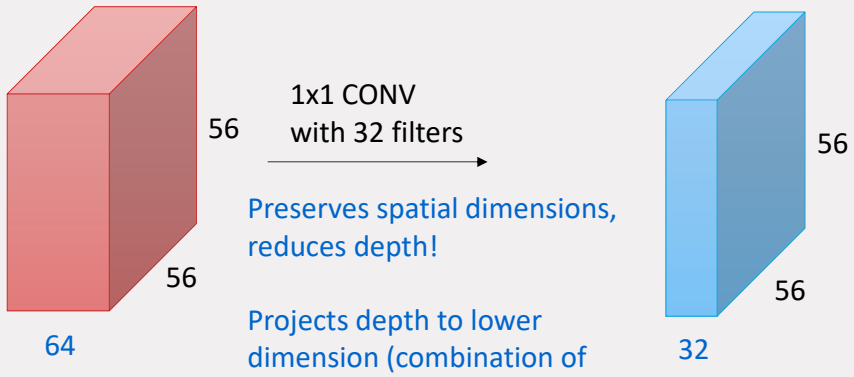
Naïve Inception module

Reminder: 1x1 convolutions



(each filter has size 1x1x64, and performs a 64-dimensional dot product)

Reminder: 1x1 convolutions

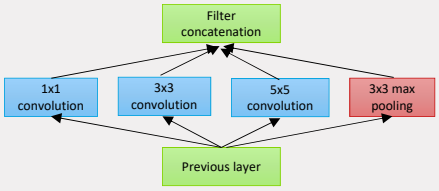


Preserves spatial dimensions, reduces depth!

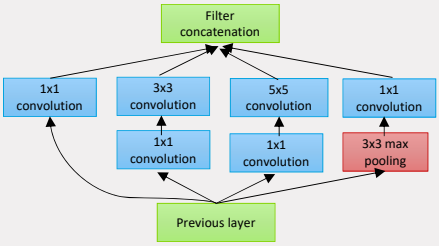
Projects depth to lower dimension (combination of feature maps)

Case study: GoogLeNet

[Szegedy et al, 2014]



Naïve Inception module

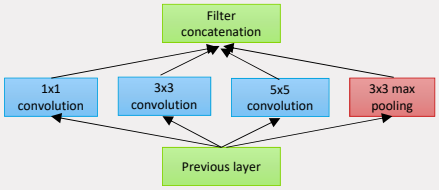


Inception module with dimension reduction

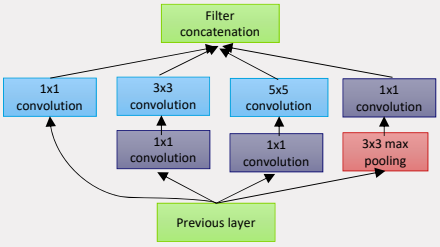
Case study: GoogLeNet

[Szegedy et al, 2014]

1x1 conv 'bottleneck' layer



Naïve Inception module



Inception module with dimension reduction

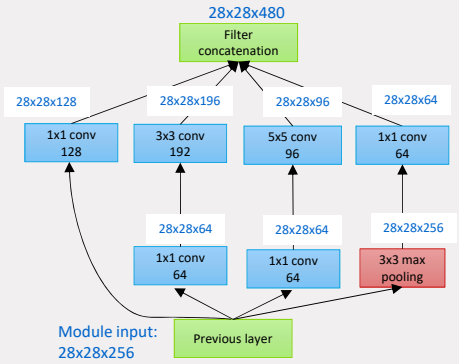
Case study: GoogLeNet

[Szegedy et al, 2014]

Using same parallel layers as naïve example, and adding '1x1 conv, 64 filter' bottlenecks:

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 naïve version
Bottleneck can also reduce depth after pooling layer

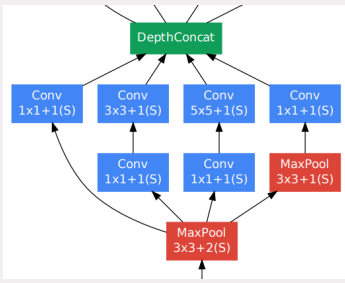


Inception module with dimension reduction

Case study: GoogLeNet

[Szegedy et al, 2014]

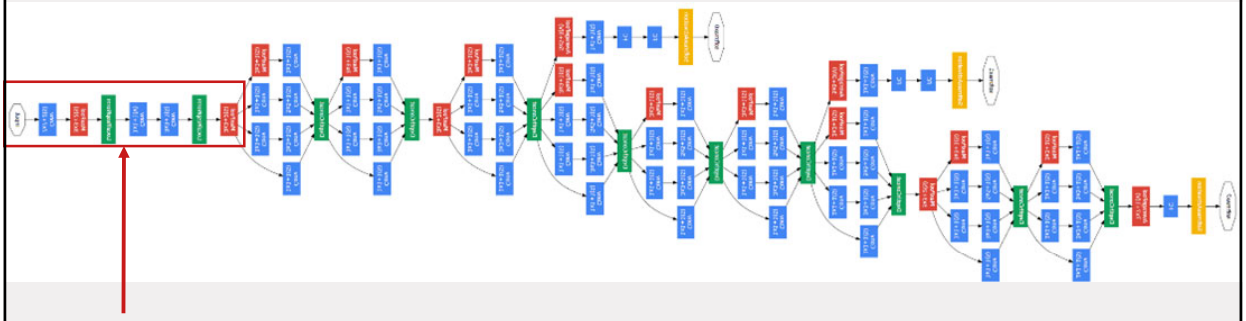
Stack Inception modules with dimension reduction on top of each other



Inception module

Case study: GoogLeNet

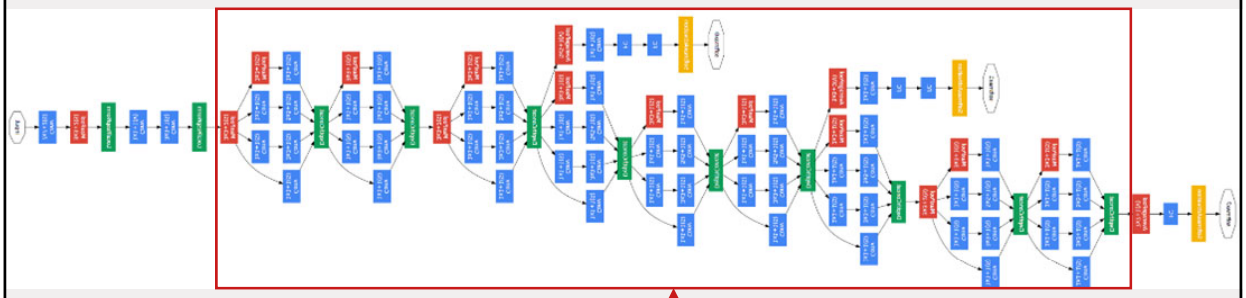
[Szegedy et al, 2014]



Stem Network: Conv-Pool-Conv-Conv-Pool

Case study: GoogLeNet

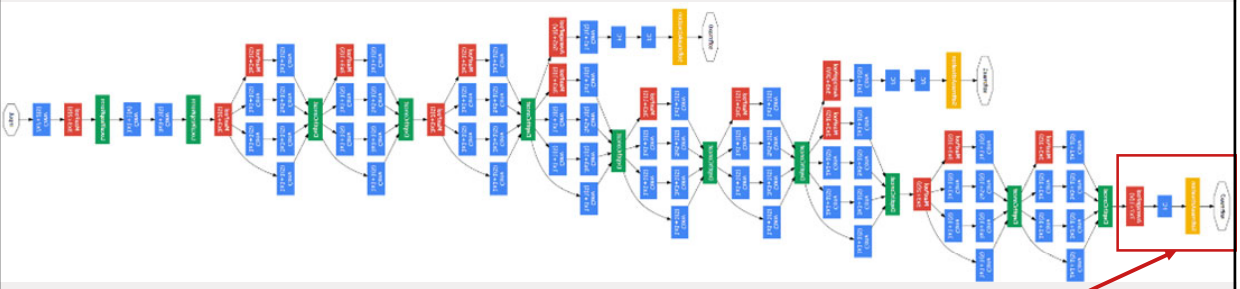
[Szegedy et al, 2014]



Stacked Inception Modules

Case study: GoogLeNet

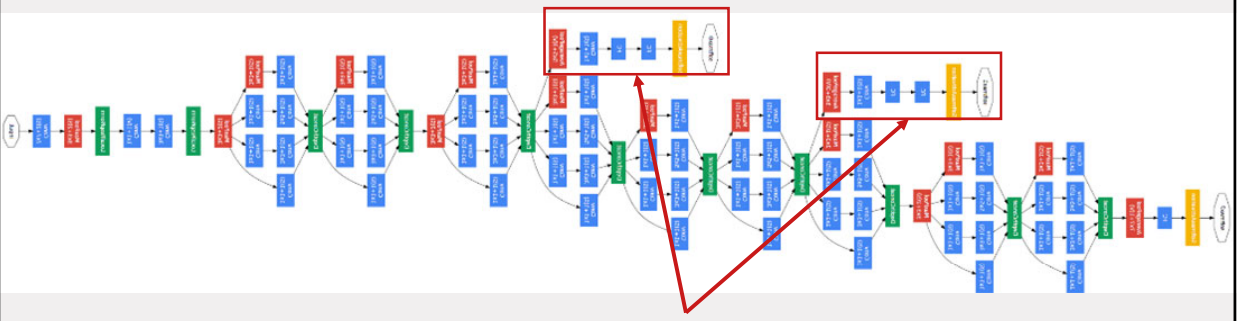
[Szegedy et al, 2014]



Classifier output
(removed expensive FC layers!)

Case study: GoogLeNet

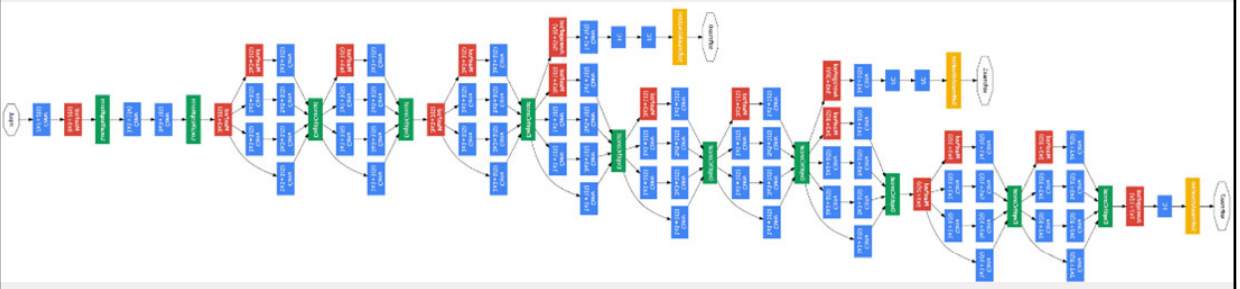
[Szegedy et al, 2014]



Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1conv-FC-FC-Softmax)

Case study: GoogLeNet

[Szegedy et al, 2014]



22 total layers with weights

(Parallel layers count as 1 layer -> 2 layers per inception module. Don't count auxiliary output layers)

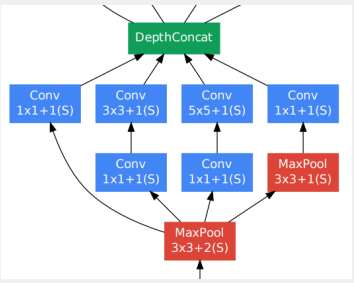
5LSM0 Module 6: CNN architectures

Case study: GoogLeNet

[Szegedy et al, 2014]

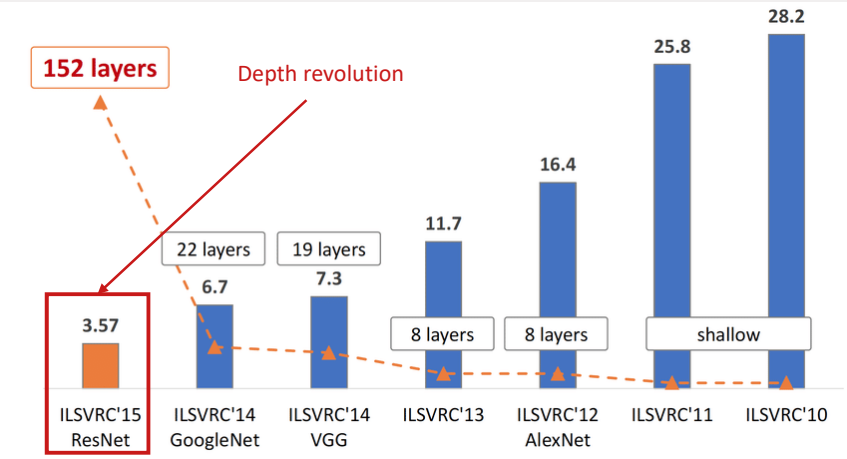
Deeper networks, with computational efficiency

- 22 layers
- Efficient 'inception module
- No FC layers
- Only 5 million parameters!
  12 times less than AlexNet
- ILSVRC'14 classification winner
  (6.7% top 5 error)



Inception module

ImageNet Large Scale Visual Recognition Challenge winners

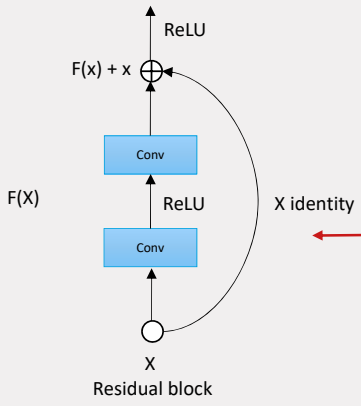


Case study: ResNet

[He et al, 2015]

Very deep networks using residual connections

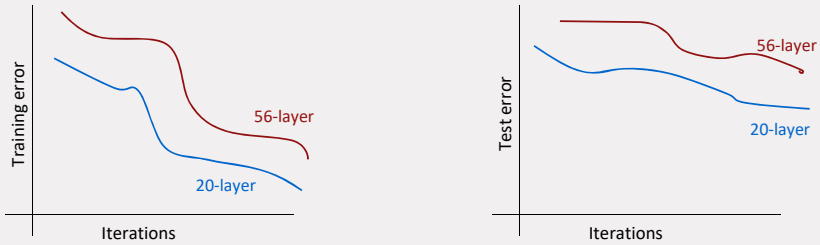
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15



Case study: ResNet

[He et al, 2015]

What happens when we continue stacking deeper layers on a 'plain' convolutional neural network?

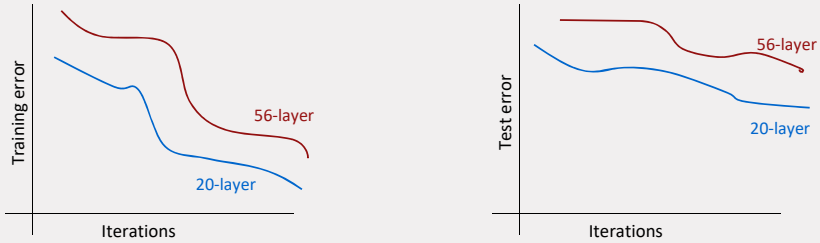


Q: What's strange about these training and test curves?

Case study: ResNet

[He et al, 2015]

What happens when we continue stacking deeper layers on a 'plain' convolutional neural network?



56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it is not caused by overfitting!

Case study: ResNet

[He et al, 2015]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

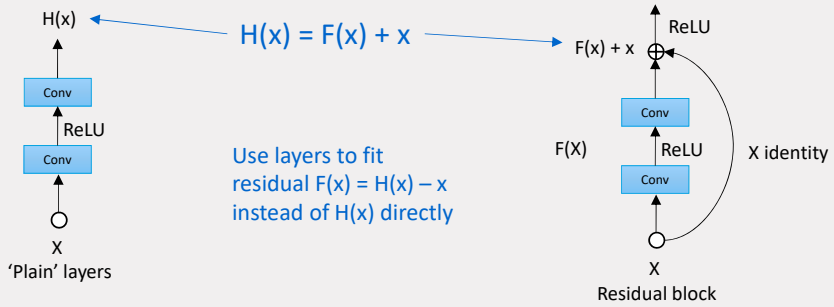
The deeper model should be able to perform at least as well as the shallower model.

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping

Case study: ResNet

[He et al, 2015]

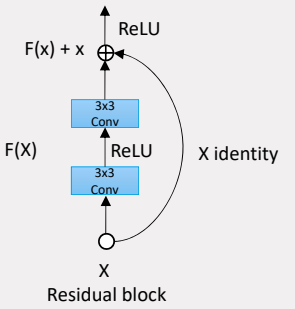
Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



Case study: ResNet

[He et al, 2015]

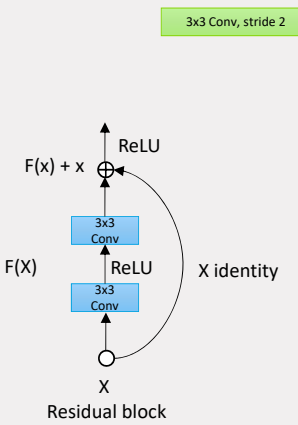
Full ResNet architecture
- Stack residual blocks
- Severy residual block has two 3x3 conv layers

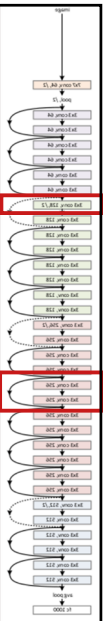


Case study: ResNet

[He et al, 2015]

Full ResNet architecture
- Stack residual blocks
- Severy residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)

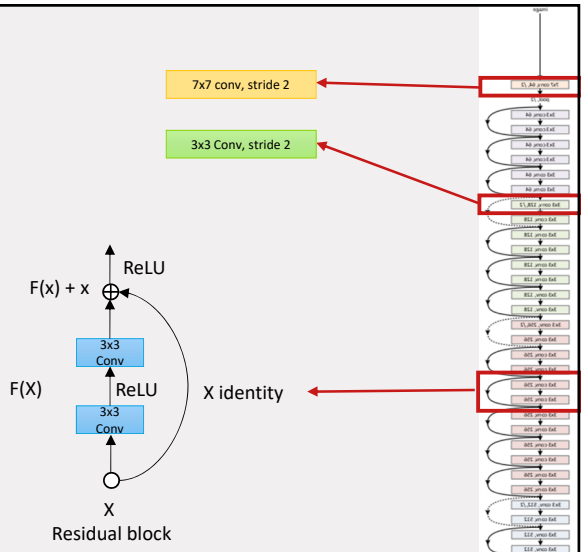




Case study: ResNet

[He et al, 2015]

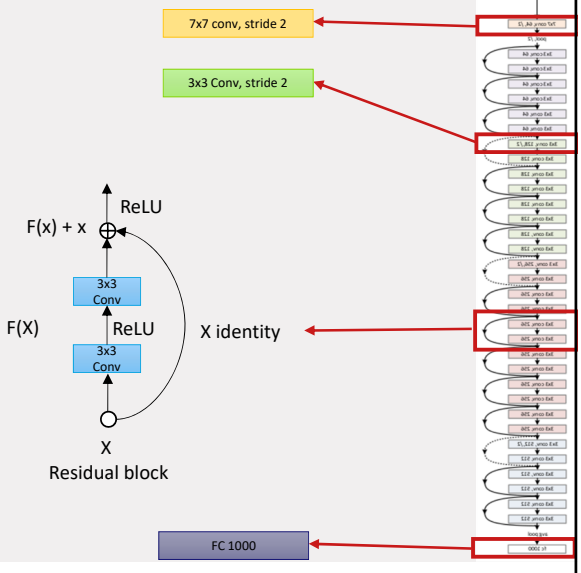
Full ResNet architecture
- Stack residual blocks
- Severy residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning



Case study: ResNet

[He et al, 2015]

Full ResNet architecture
- Stack residual blocks
- Severy residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)



Case study: ResNet

[He et al, 2015]

Total depths of 34, 50, 101, or 152 layers for ImageNet

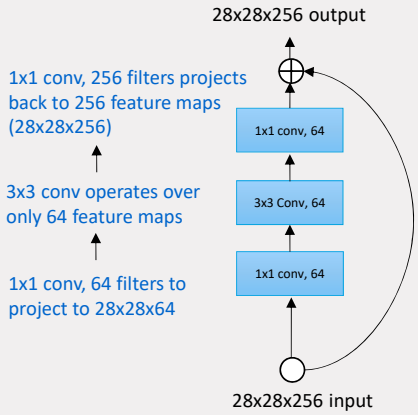
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

TU/e

Case study: ResNet

[He et al, 2015]

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



TU/e

Case study: ResNet

[He et al, 2015]

Training ResNet in practice:

- Batch Normilization after every CONV layer
- Xavier initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

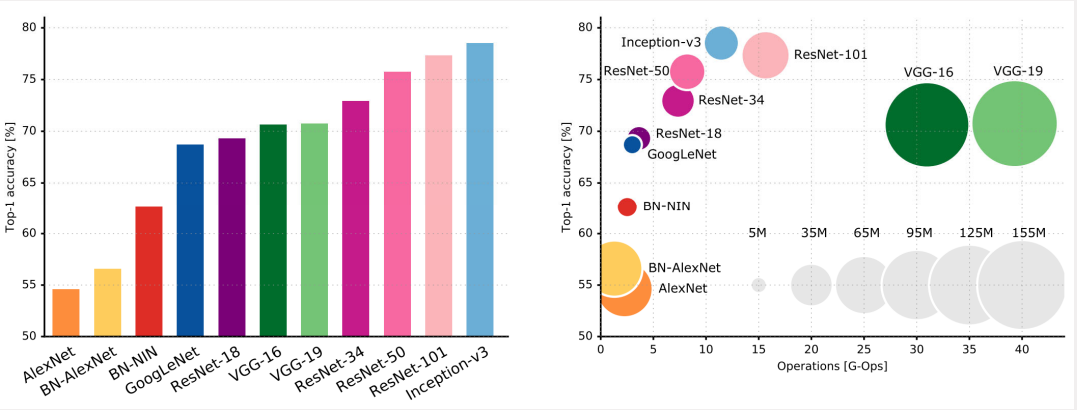
Case study: ResNet

[He et al, 2015]

Experimental results

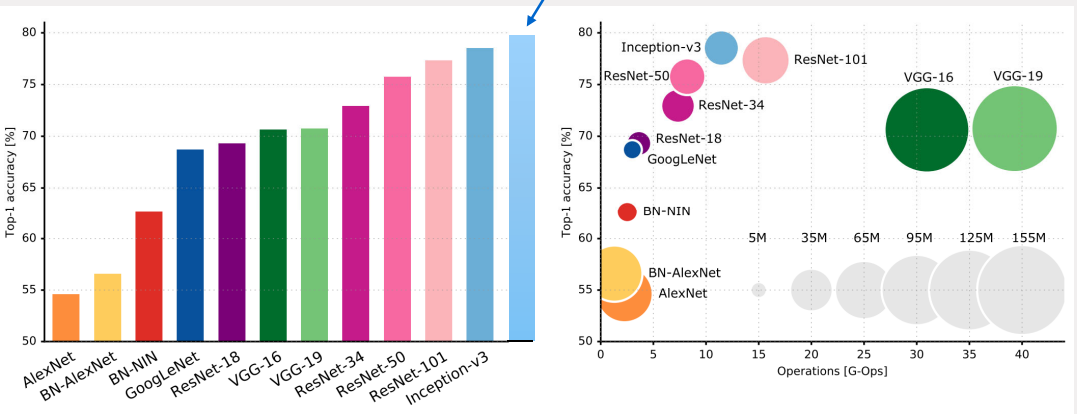
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

Comparing complexity



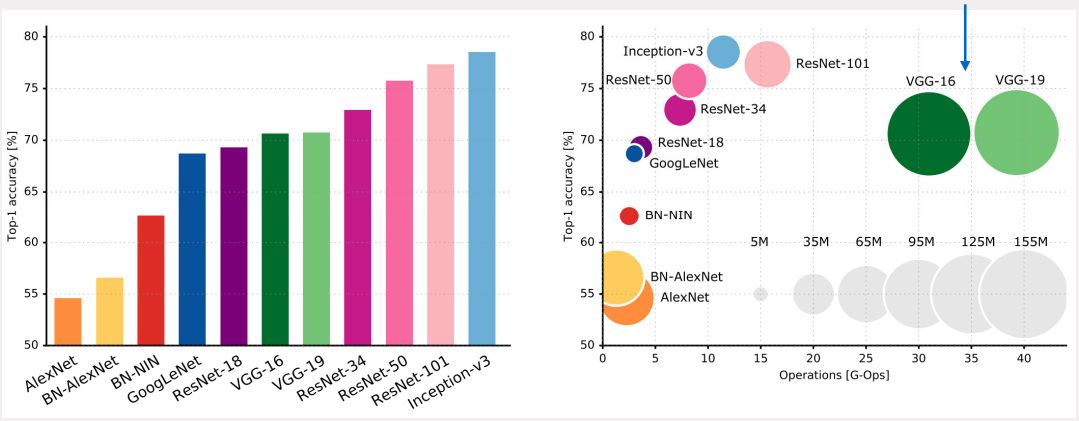
Comparing complexity

Inception-v4: Resnet + Inception!



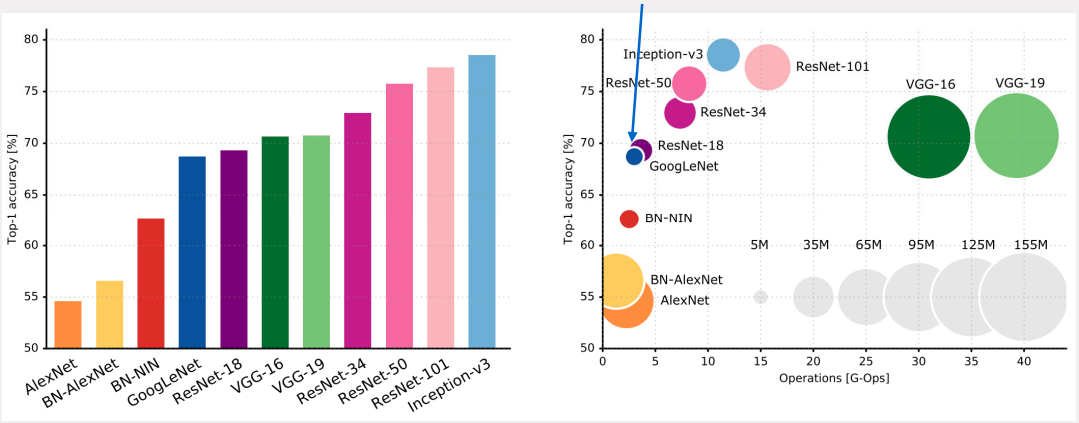
Comparing complexity

VGG: highest memory, most operations



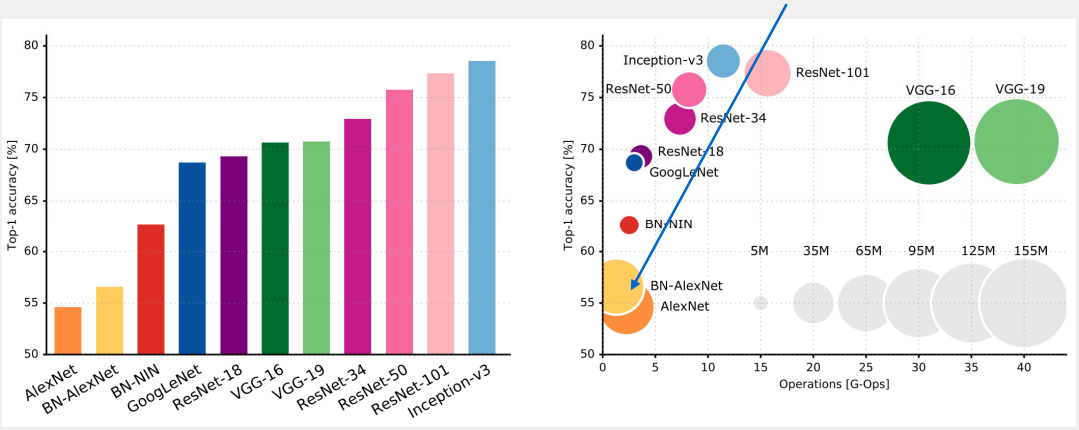
Comparing complexity

GoogLeNet: most efficient



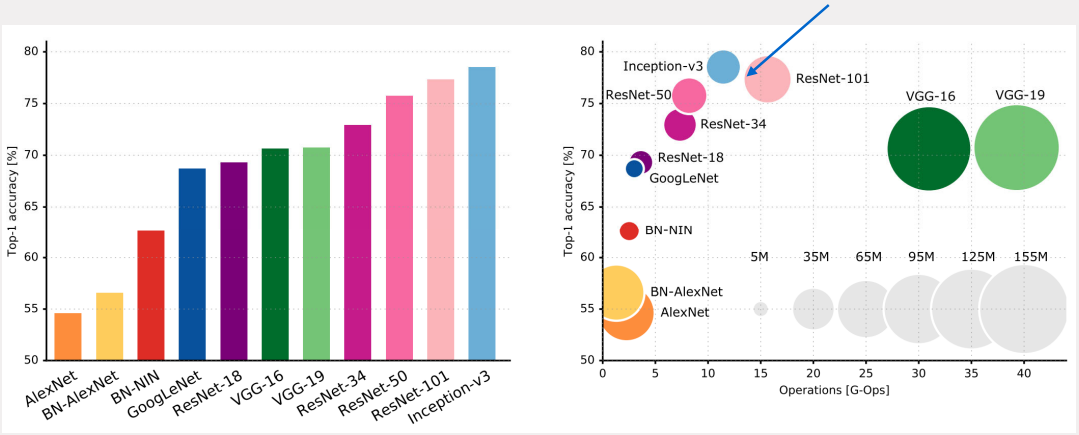
Comparing complexity

AlexNet: Smaller compute,
still memory heavy, lower accuracy

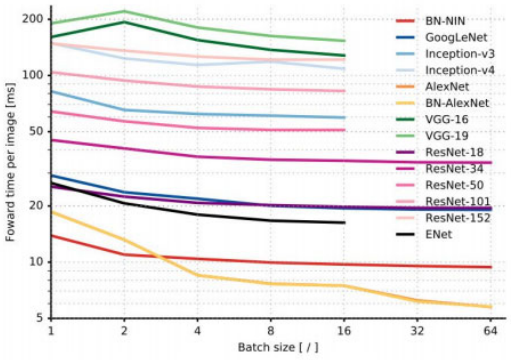


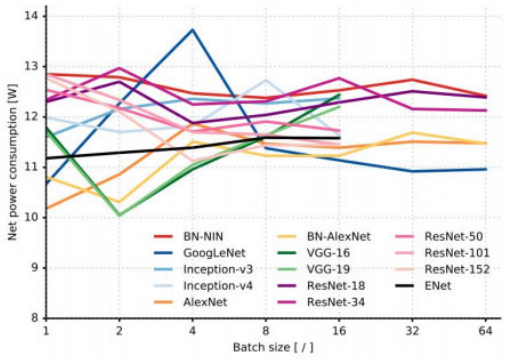
Comparing complexity

ResNet: Moderate efficiency depending
on model, highest accuracy



Comparing complexity



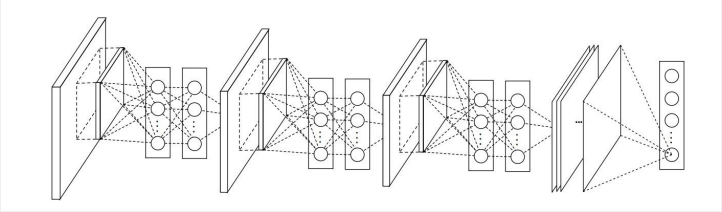


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Other architectures: Network in Network (NiN)

[Lin et al, 2014]

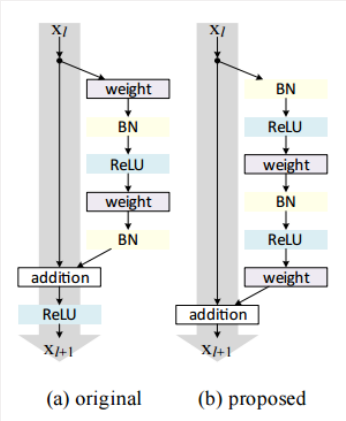
- MLPConv layer with 'micronetwork' within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet 'Bottleneck' layers
- Philosophical inspiration for GoogLeNet



Other architectures: Identity mapping for ResNet

[He et al, 2016]

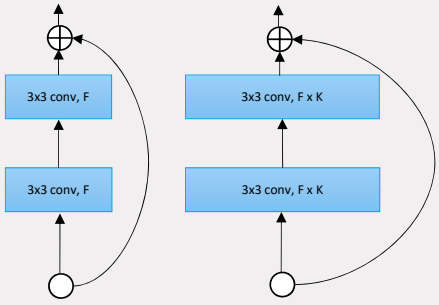
- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout the network (moves activation to residual mapping pathway)
- Gives better performance



Other architectures: Wide ResNets

[Zagoruyko et al, 2016]

- Argues that residuals are the important factor, not depth
- Uses 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

Other architectures: 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

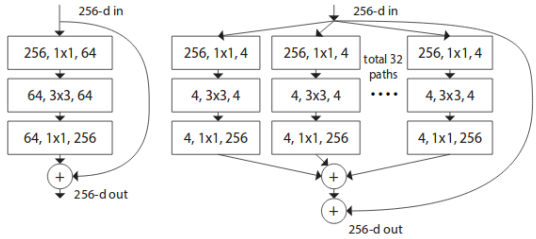
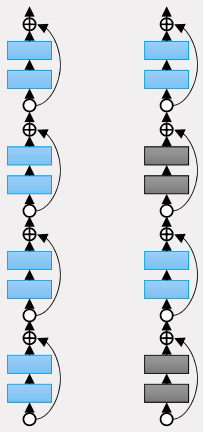


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Other architectures: Deep Networks with Stochastic Depth

[Huang et al, 2016]

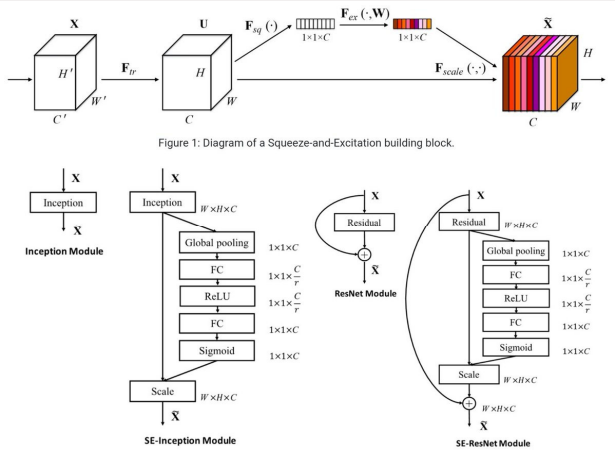
- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



Other architectures: Squeeze and Exiation Network (SENet)

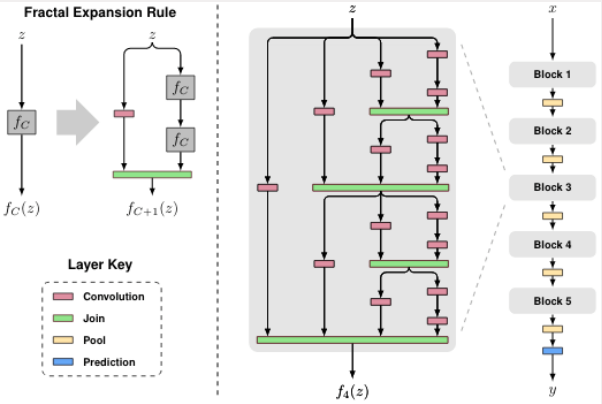
[Hu et al, 2017]

- Add a 'feature recalibration' module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC'17 classification winner (using ResNeXt-152 as a base architecture)

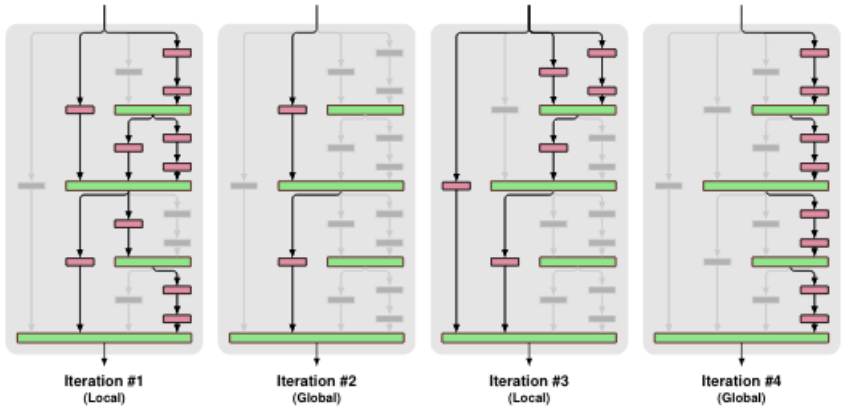


Other architectures: FractalNet: Ultra-deep networks without residuals [Hu et al, 2017]

- 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



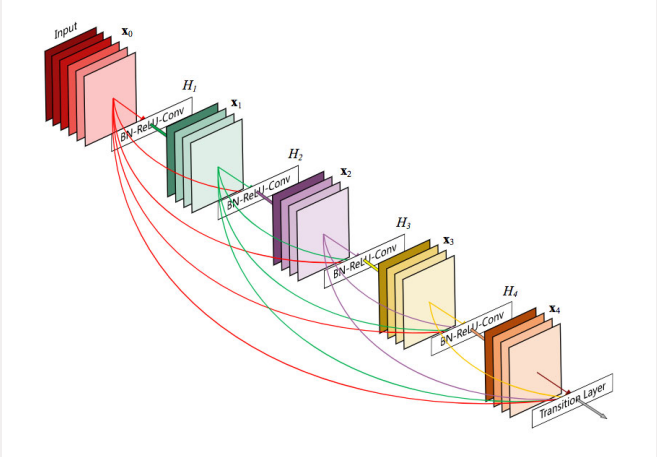
Other architectures: FractalNet: Ultra-deep networks without residuals [Hu et al, 2017]



Other architectures: DenseNet

[Huang et al, 2017]

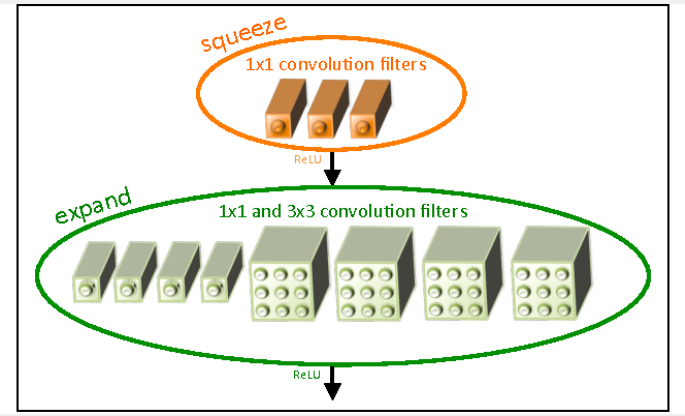
- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



Other architectures: SqueezeNet

[Iandola et al, 2017]

- AlexNet-level Accuracy with 50x fewer parameters and <0.5Mb model size
- Fire modules consisting of a 'squeeze' layer with 1x1 and 3x3 filters



Summary

Case studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also…

- Network in Network
- Wide ResNet
- ResNeXT
- DenseNet
- FractalNet
- Stochastic Depth
- Squeeze-and-Exitation network
- SqueezeNet

Summary: CNN Architectures

- VGG, GoogLeNet. ResNet all in wide use, available in model zoos
- ResNet current best default, aslo consider SENet when available
- Trend towards extremely deep networks
- Significant research centers around design of layer/skip connections and improving gradient flow
- Efferts to investigate necessity of depth vs. width and residual connections