Convolutional Neural Network Architectures

Girish Gore
www.linkedin.com/in/datascientistgirishgore

Computer Vision Foundations

- Basic Computer Vision
  - Computer Vision an Introduction
  - Fundamentals of Image Processing
- Convolutional Neural Networks
- CNN Architectures
- Transfer Learning & Applications

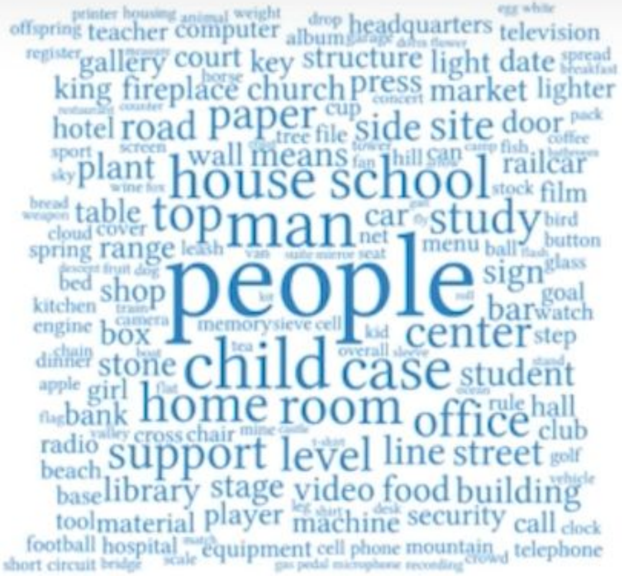
Pre Requisites

- Machine Learning Overall Process Understanding
- Deep Learning Foundations

greatlearning
Learning for Life

Image Classification Data Sets

MNIST: handwritten digits

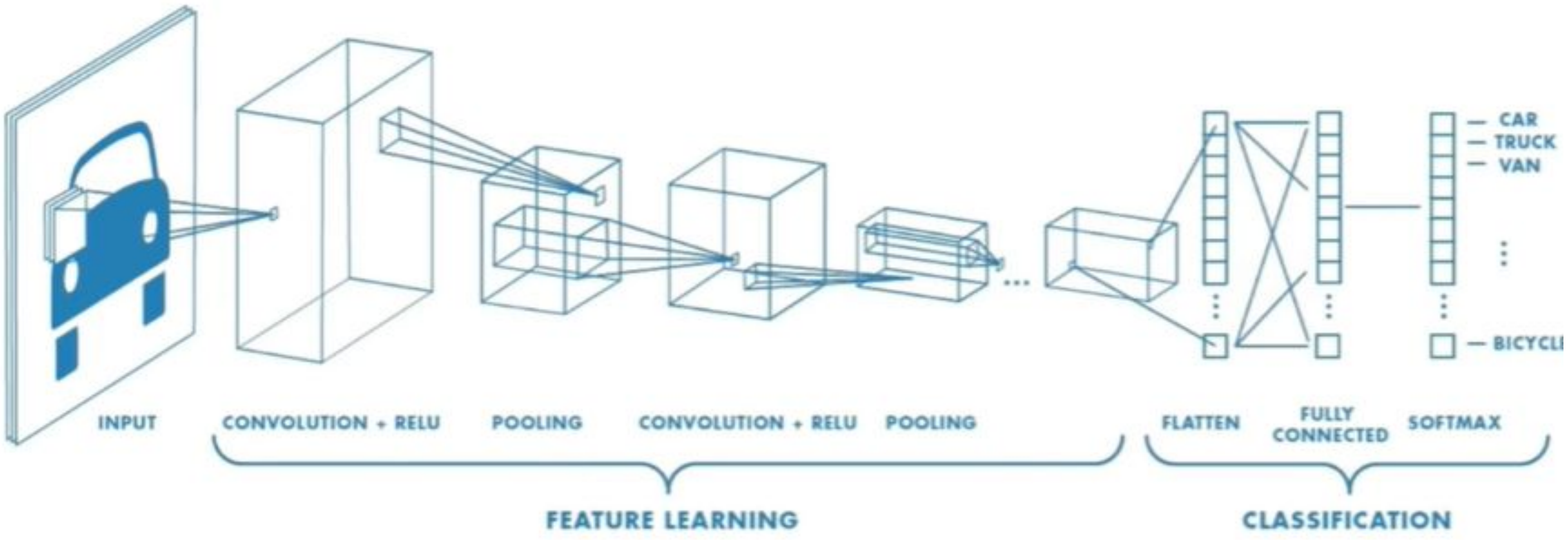


ImageNet: WordNet hierarchy

CIFAR-10(0): tiny images

Places: natural scenes

Summary : Typical CNN's for Object Classification



$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

- Learn weights for
  - Filters in Convolutional Blocks for Feature Learning
  - Fully Connected Layers for Classification
- Backpropagating the cross entropy loss across both

$$J(\theta) = \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

ImageNet (ILSRC)

- 14 M images from (21841) 20k categories !
- Ex:18800 fruit images with 1409 banna , 1206 granny smith apples etc.

Classification task: produce a list of object categories present in image. 1000 categories.
"Top 5 error": rate at which the model does not output correct label in top 5 predictions

Top-5 error on 100k test images

- You get 5 guesses to get the correct label

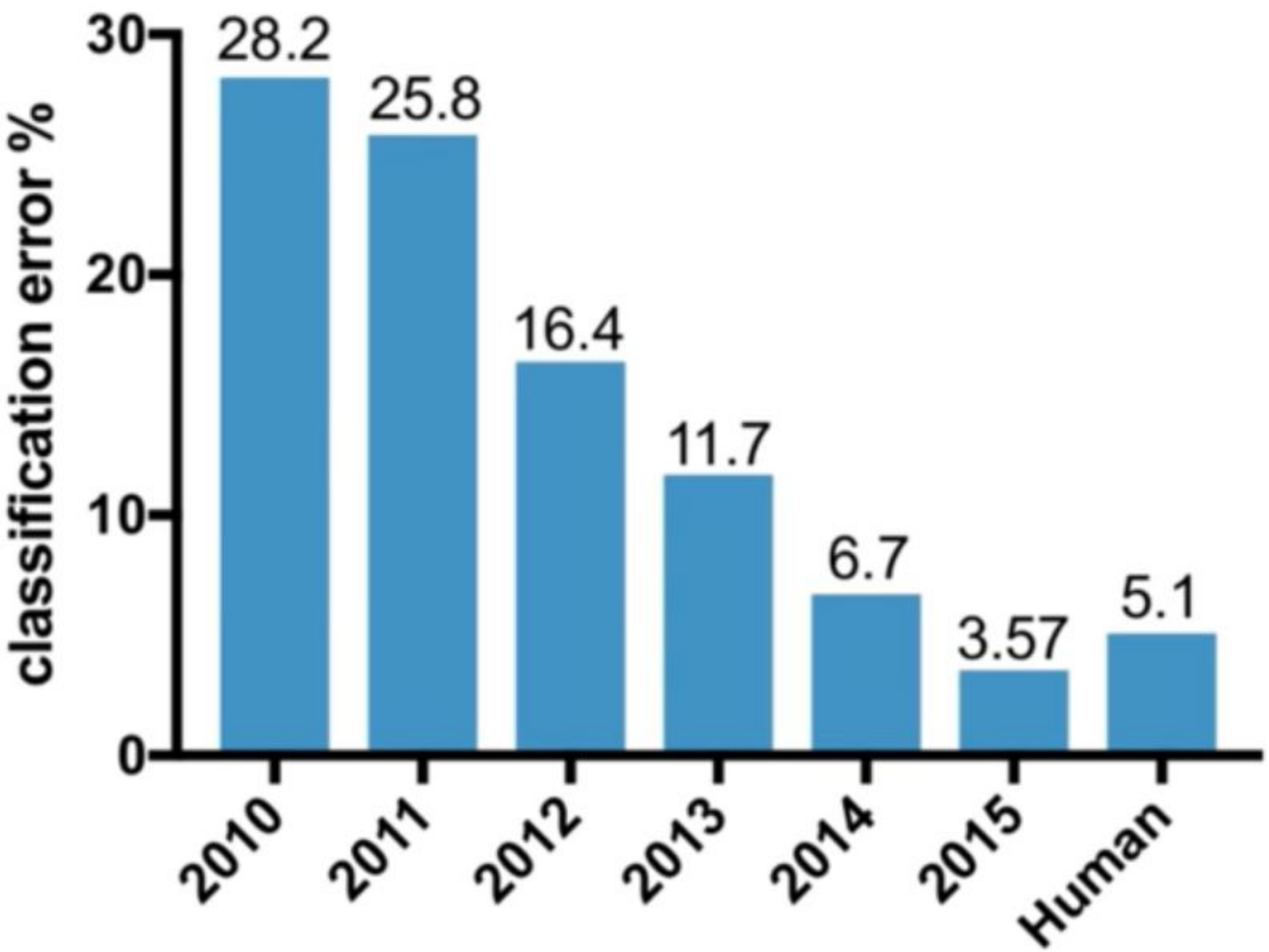
Steel drum

Output:
Scale
T-shirt
<u>Steel drum</u>
Drumstick
Mud turtle

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

$$\text{Error} = \frac{1}{100{,}000} \sum_{\substack{100{,}000 \\ \text{images}}} 1[\text{incorrect on image } i]$$

ImageNet Trained Architectures & Best Practices



2012: AlexNet. First CNN to win.
- 8 layers, 61 million parameters

2013: ZFNet
- 8 layers, more filters

2014: VGG
- 19 layers

2014: GoogLeNet
- "Inception" modules
- 22 layers, 5million parameters

2015: ResNet
- 152 layers

AlexNet(2012)

- Around 2011, a good ILSVRC classification error rate was **25.8%**. In 2012, AlexNet achieved **16.4%**, a watershed moment!
- Since then, the Computer Vision field has completely changed for one!

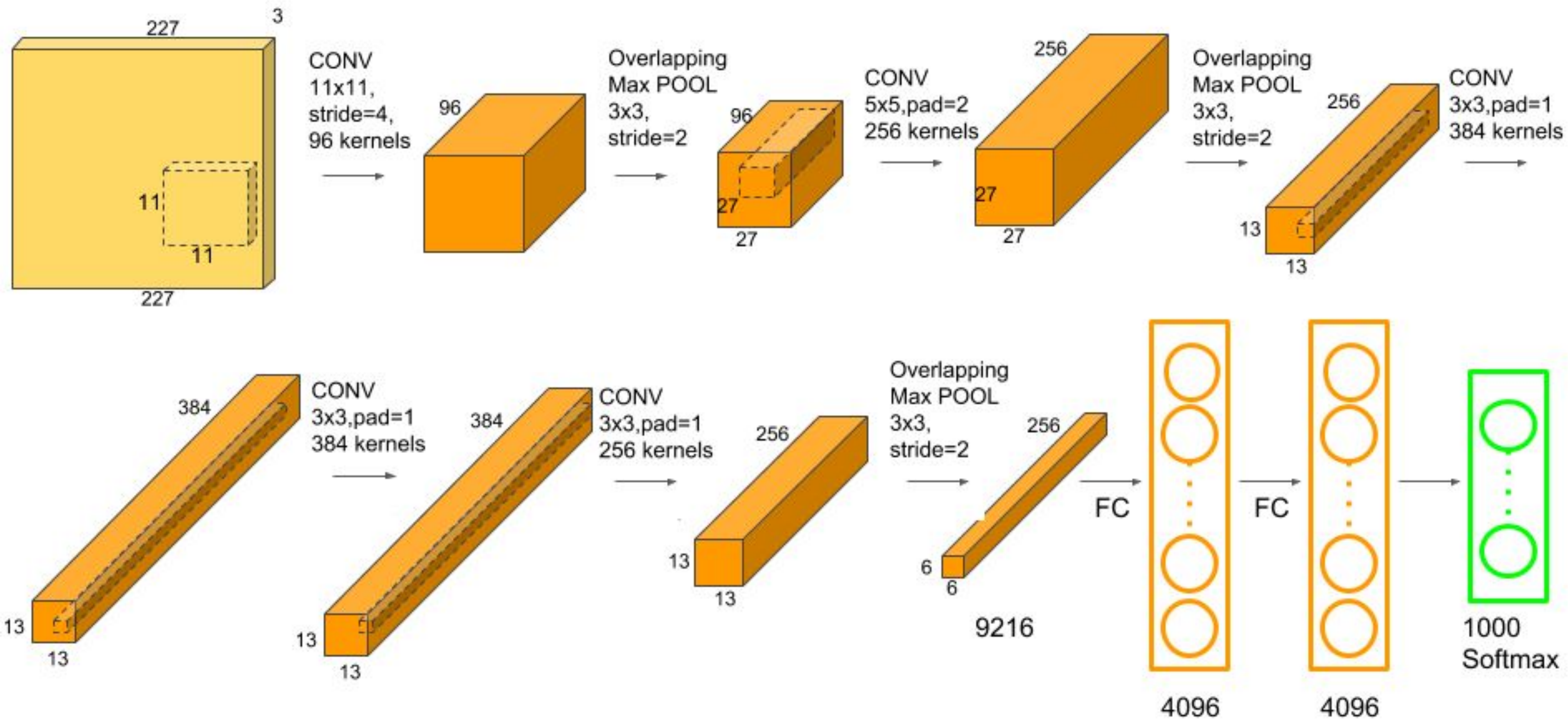
  Computer Vision == CNN's (Deep :) ofcourse)

- Compared to the state of the art DL architectures in 2012, AlexNet had a deep architecture (5 Conv layers, 3 Fully connected layers)

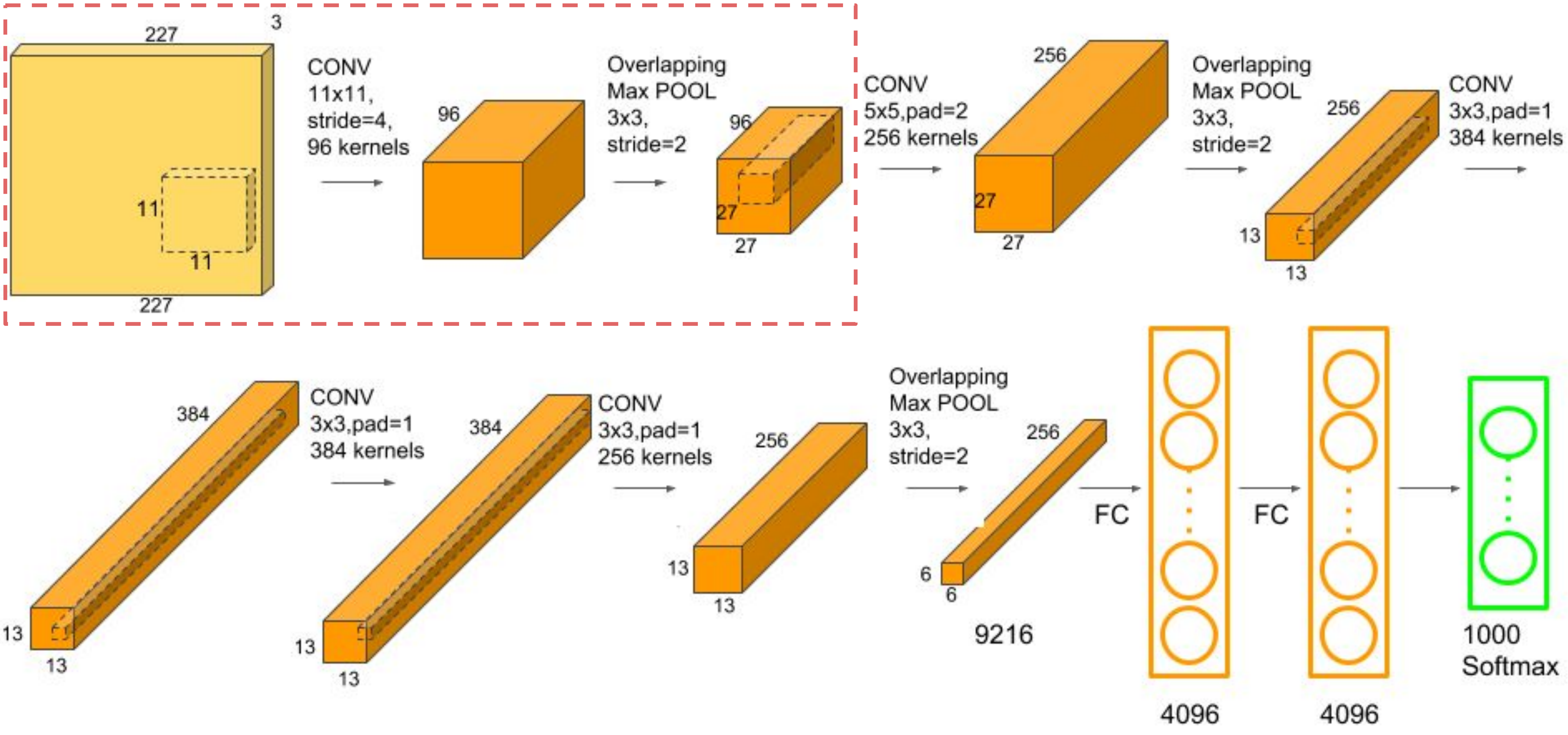
A peek into AlexNet ...
first successful CNN architecture

1. AlexNet architecture
2. Deep dive block by block
3. Overlapping max pooling
4. ReLu
5. Dropouts
6. Cropping
7. Data Augmentation
8. Inference Augmentation

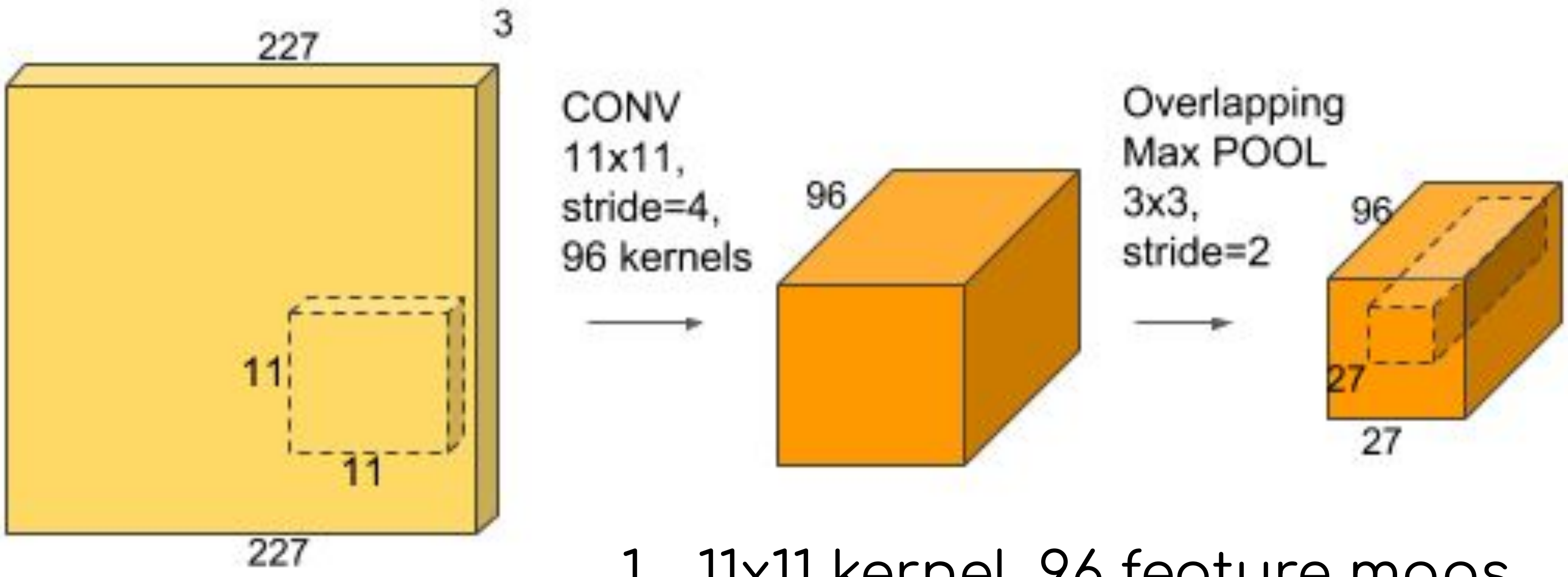
AlexNet (2012) : 8 Layers(5+3)



Lets Step in..

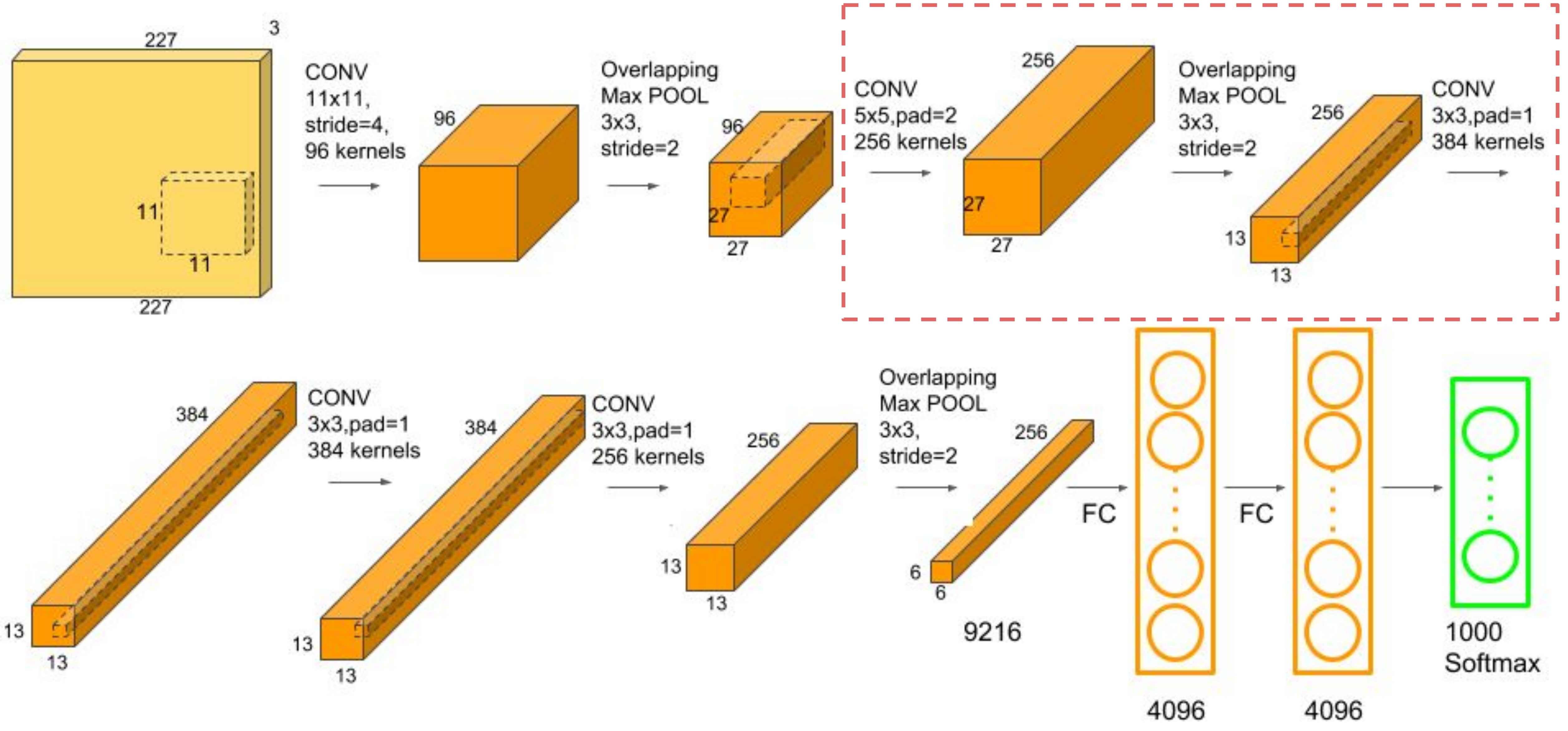

227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
96
Overlapping
Max POOL
3x3,
stride=2
96
27
27
CONV
5x5,pad=2
256 kernels
256
27
27
Overlapping
Max POOL
3x3,
stride=2
256
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
256 kernels
256
13
13
Overlapping
Max POOL
3x3,
stride=2
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Lets go block by block - First Block

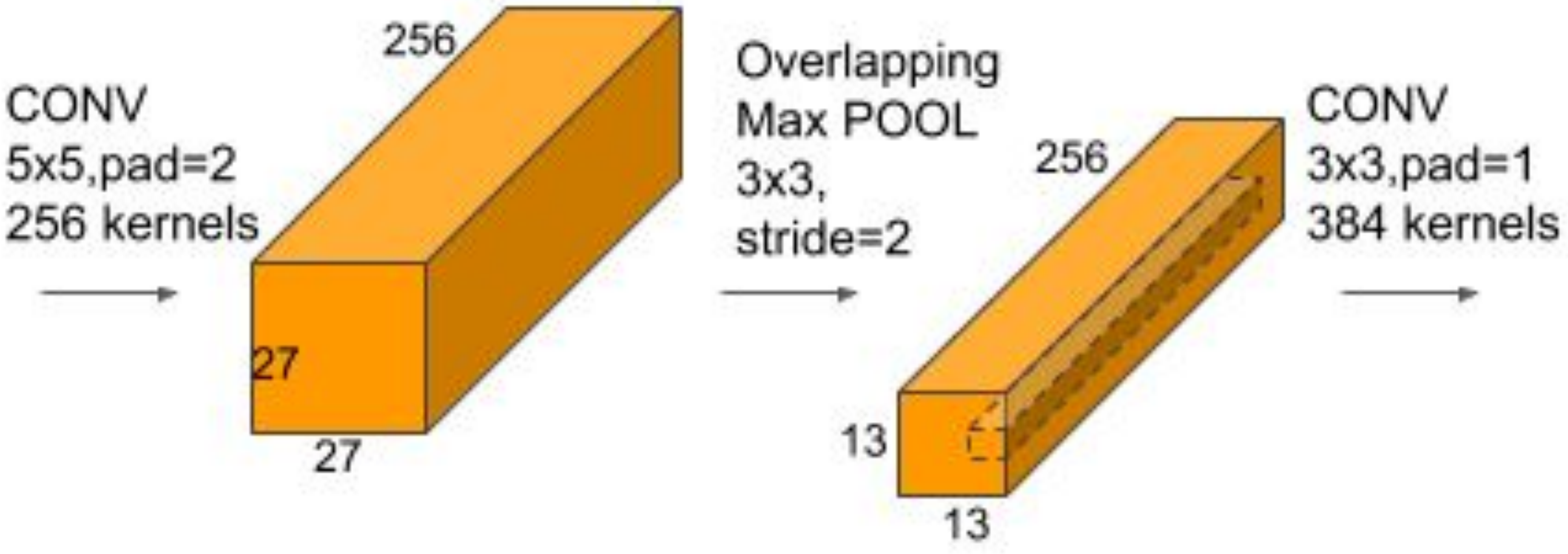


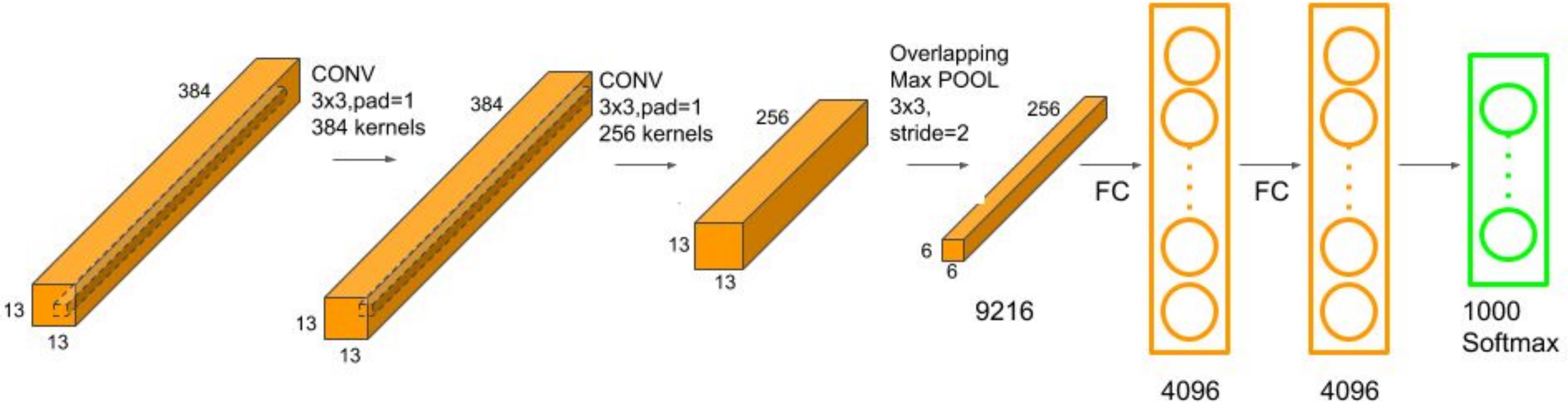
1. 11x11 kernel, 96 feature maps
2. Large Stride=4
3. Formula for output size - **(W+2P-F)/S+1**
4. Formula for no of parameters (ignore bias)- **W*D*k*k**
5. Maxpool, 3x3, s=2

Lets Step in..


227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
96
Overlapping
Max POOL
3x3,
stride=2
96
27
27
CONV
5x5,pad=2
256 kernels
256
27
27
Overlapping
Max POOL
3x3,
stride=2
256
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
256 kernels
256
13
13
Overlapping
Max POOL
3x3,
stride=2
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Second block





1. Flatten layer
2. FC layer
3. Softmax
4. #parameters in FC layers?

Overlapping Max Pooling

greatlearning
Learning for Life

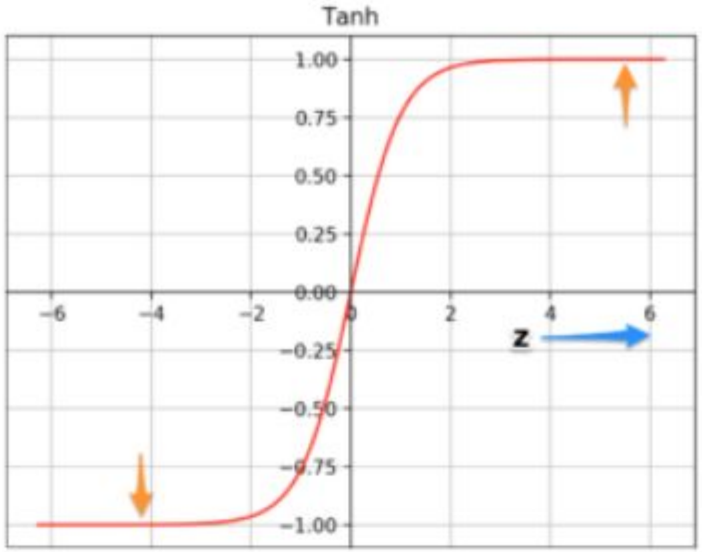
(3x3, stride 2)

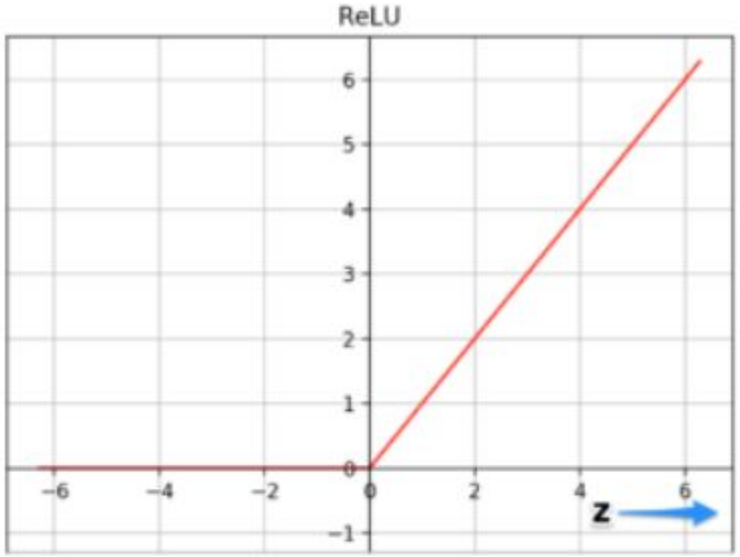
1	4	5	2	7
5	3	6	3	6
7	2	1	1	4
3	9	4	6	7
4	2	5	1	2

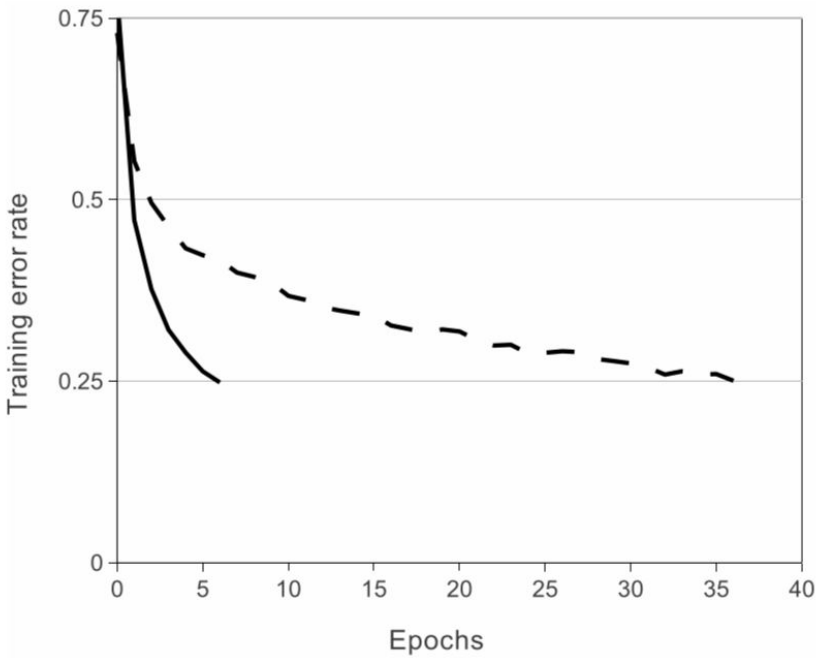
7	7
9	7

Moderate performance gain reported by authors

ReLU instead of tanh







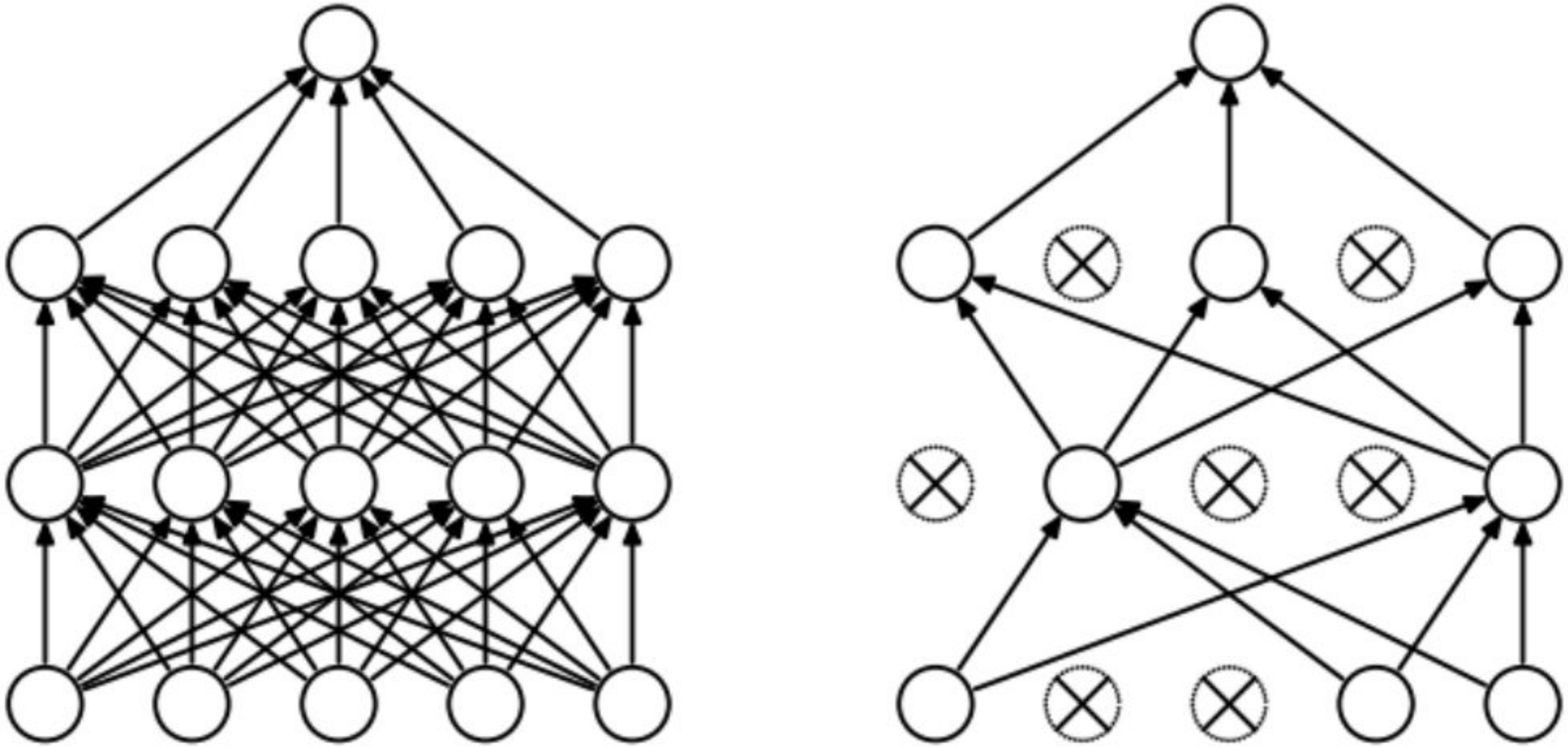
tanh

ReLU

Faster convergence

Previously, Vanishing Gradients was an issue networks couldn't go deeper, ReLU

Dropout in fully connected layers



(a) Standard Neural Net

(b) After applying dropout.

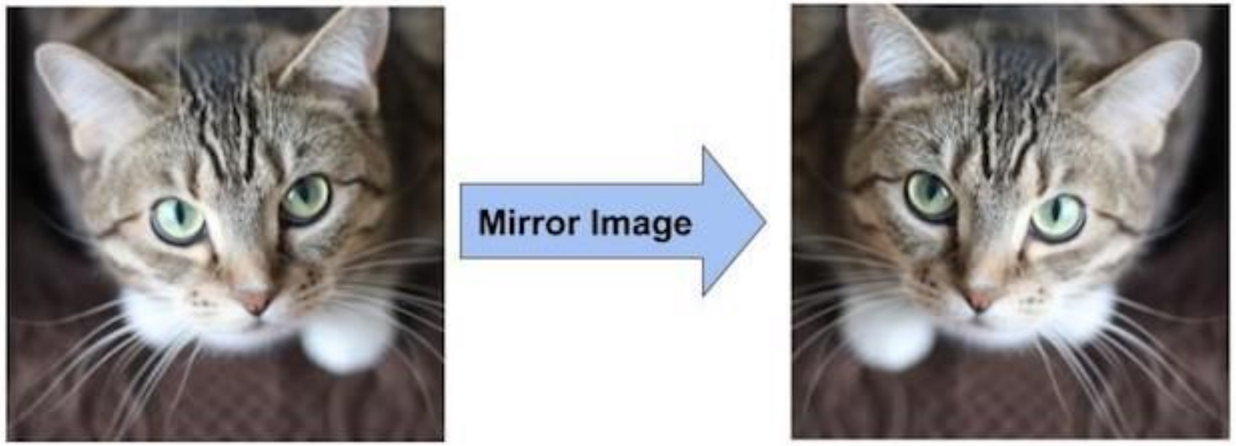
Regularization key due to huge #parameters in FC layer
Dropout as high as 0.5 used.

Input Images

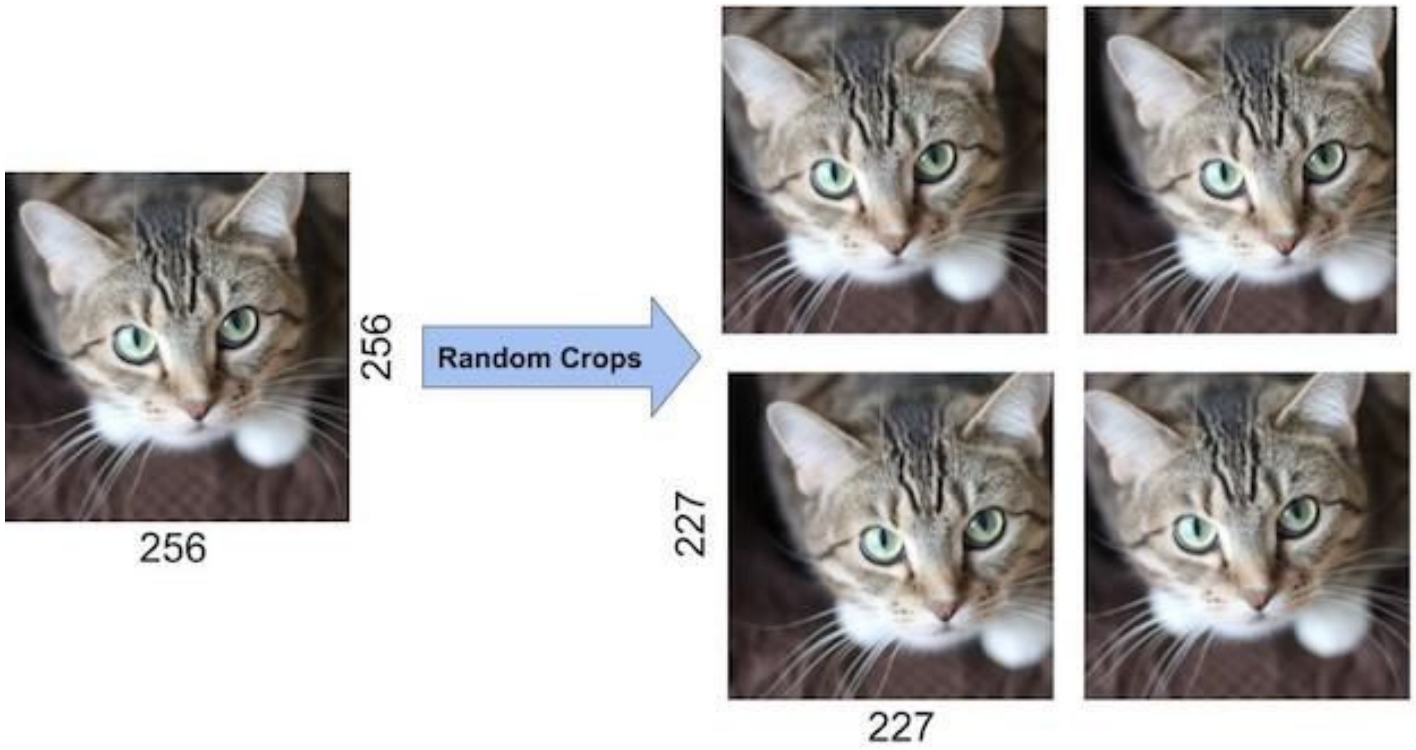
Resize smaller side to 256 and crop larger side to get 256x256

Get close to object of interest

Data augmentation

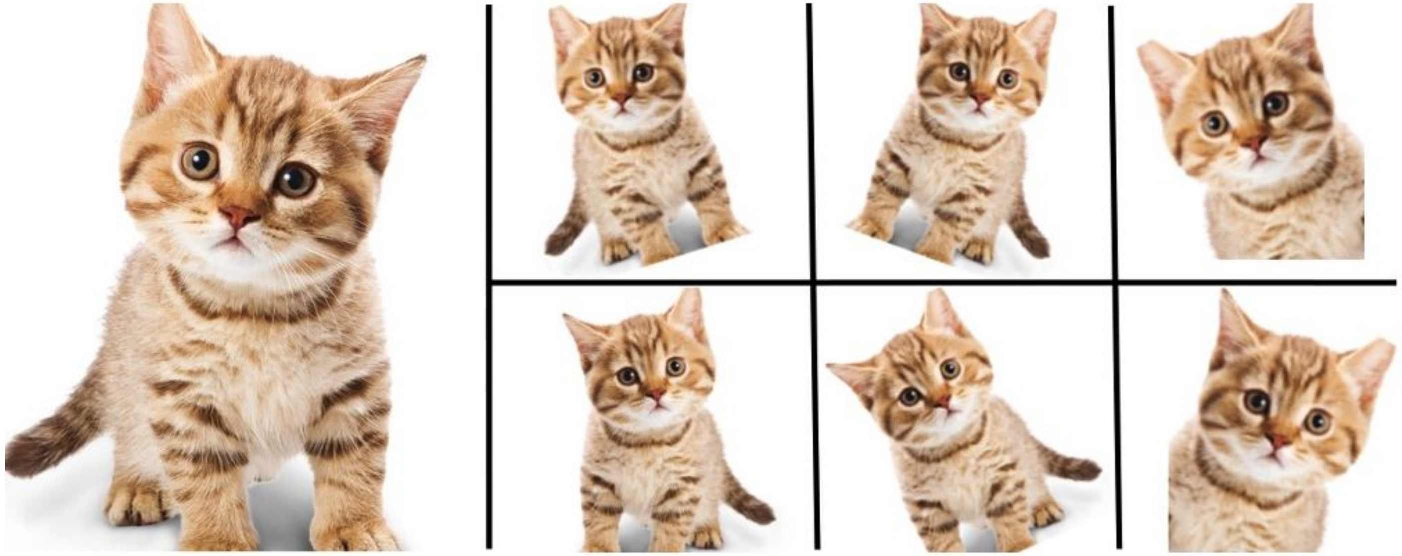


Flipping , Gittering , Cropping



Random crop of 227x227 from 256x256 images

Inference Augmentation



Predict for each image using learnt model

Average prediction

Test image

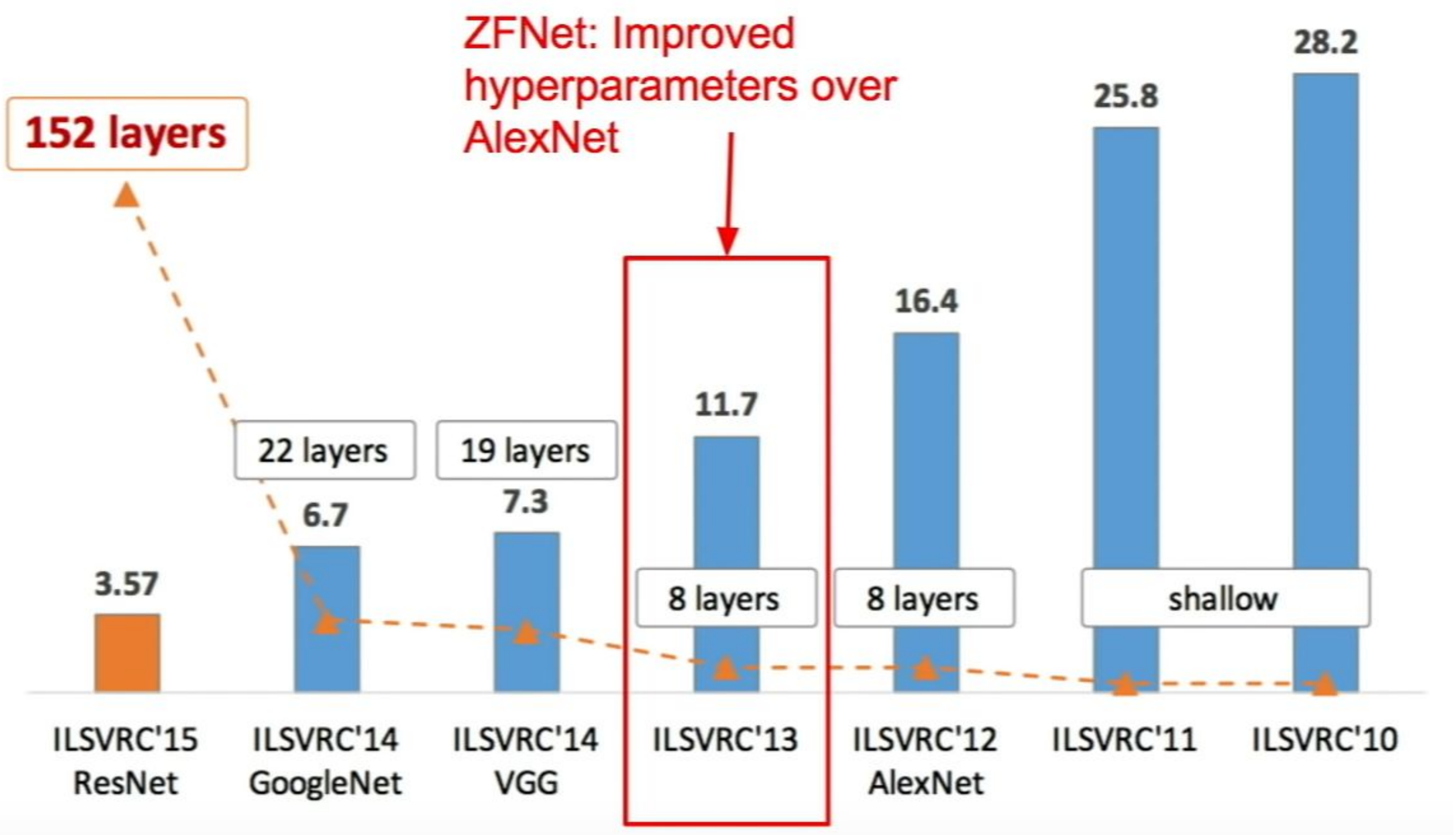
Augmented versions of test image

Is seen to improve accuracy moderately in many applications

Summary

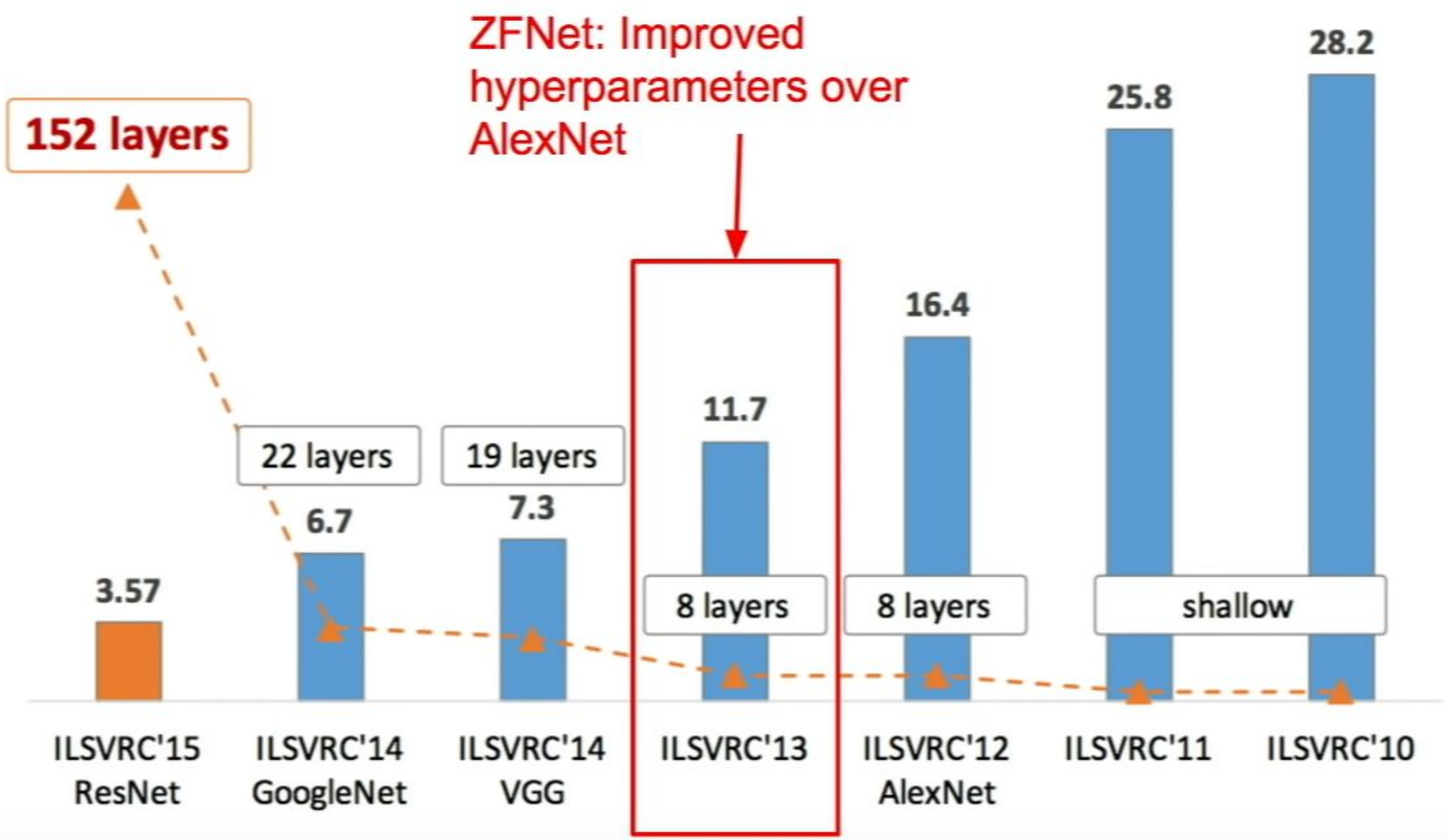
- Trained the network on ImageNet data
- Overlapping Maxpool
- Used ReLU as the nonlinearity functions over tanh (first time)
- Data augmentation: image translations, horizontal reflections, and patch extractions.
- Inference/Test-time augmentation
- SGD Momentum set to 0.9 & Batch Size 128
- Dropout (0.5) in fully connected layers
- 7 CNN Ensembles : Error Rate Drop from 18.2% to 15.4%
- Trained on two GTX 580 , 3GB memory GPUs for five to six days
- Norm Layers (not commonly used now)

ZFNet(2013)



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

ZFNet(2013)



Hyper Parameters in CNN (sample list)

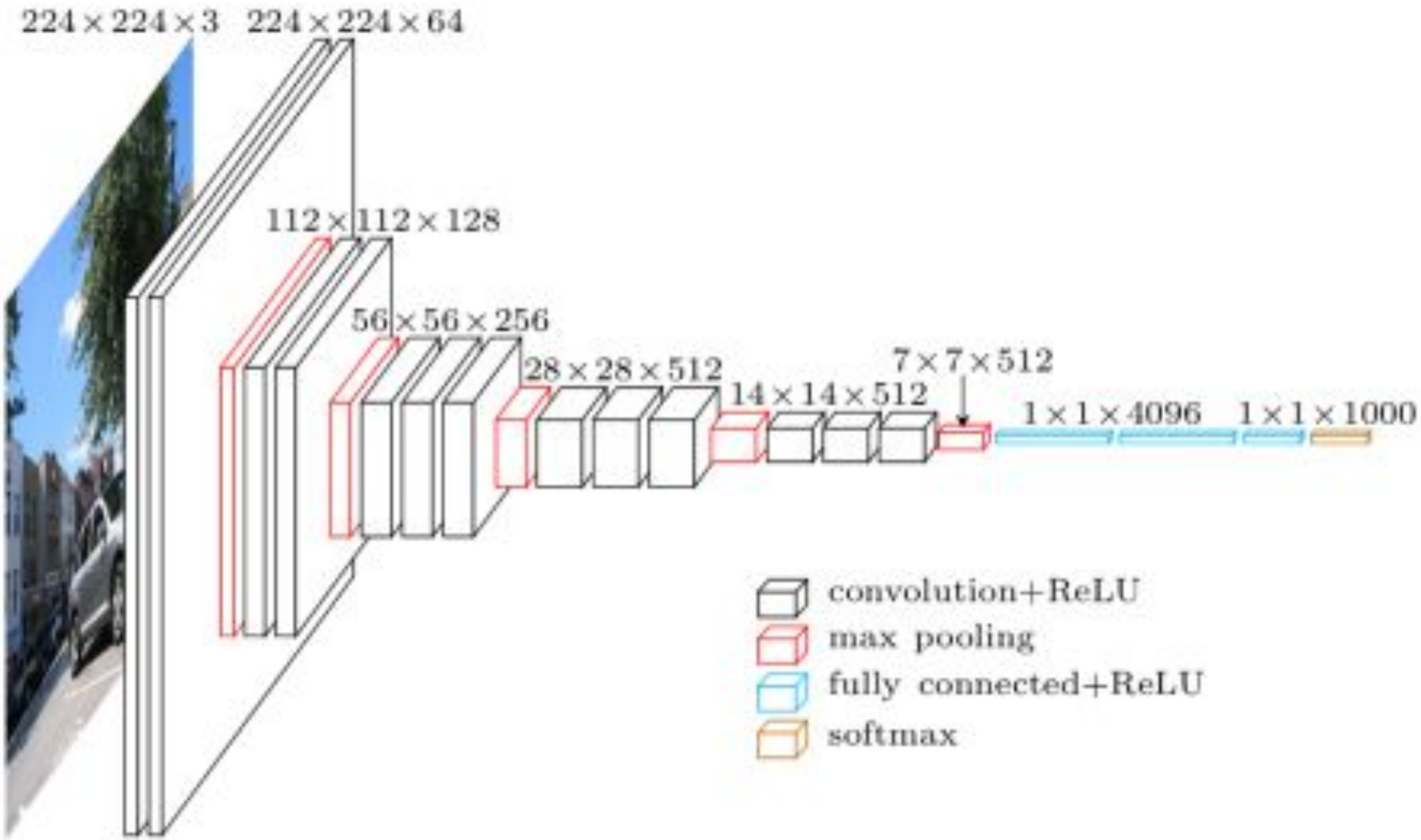
- Image Sizes
- Strides
- Pooling
- Dropouts
- Network weights initializations
- Choice of activations
- Optimization Algo
    - Learning Rate
    - Momentum
    - Epochs
    - Batch Size
- Number of layers & hidden units

VGG (2014)

Another influential work is VGG which brought the ImageNet error down below **10% i.e. from 16.8% to 7.3%** precisely**)**

Compared to Alexnet : Smaller Filters & Much Deeper
Use of 3x3 filters is mimicked by most works today

Scale Augmentation at Train and Test time is another key addition



Notice the increasing Filter depth in each layer?

Key Points

- Use of 3×3 Filters instead of large-size filters (such as 11×11, 7×7)
- Different VGG architectures
- Increasing Filter depth
- Multi-Scale Training/ Testing
- Model Ensembling

Need for large filters and challenges

In images, non-local or wide range pixel interactions is important to capture

Thus a wide receptive field is important

Need
Smaller
Receptive
Field

Need
Larger
Receptive
Field

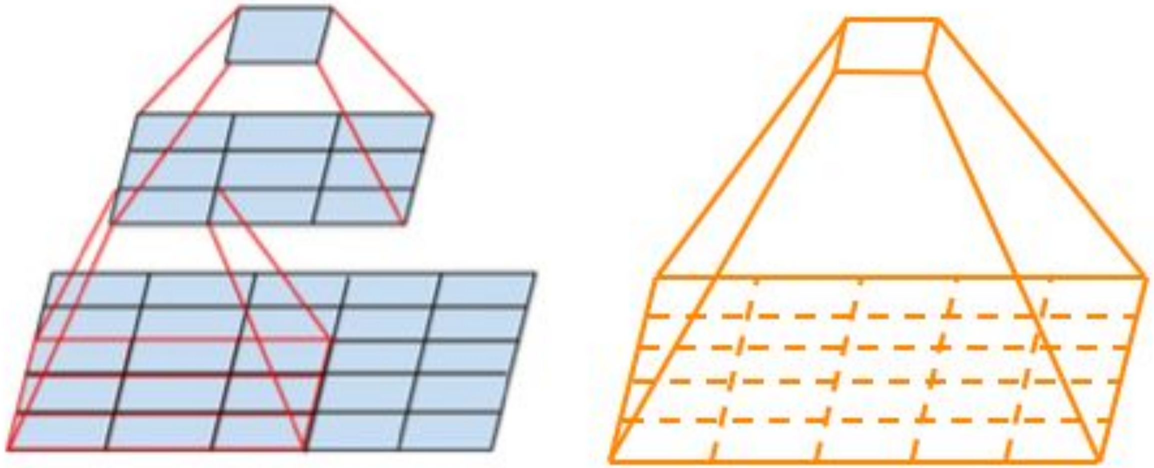
- Larger kernels (7x7, 11x11), Maxpooling are possibilities
- With pooling, information loss is a risk
- Larger kernels mean more parameters/compute

Remember:
M- inputs of dimension D x D,
N - outputs,
KxK kernels take

MxNxKxK parameters &
MxNxKxKxDxD operations

Use of 3x3 filters

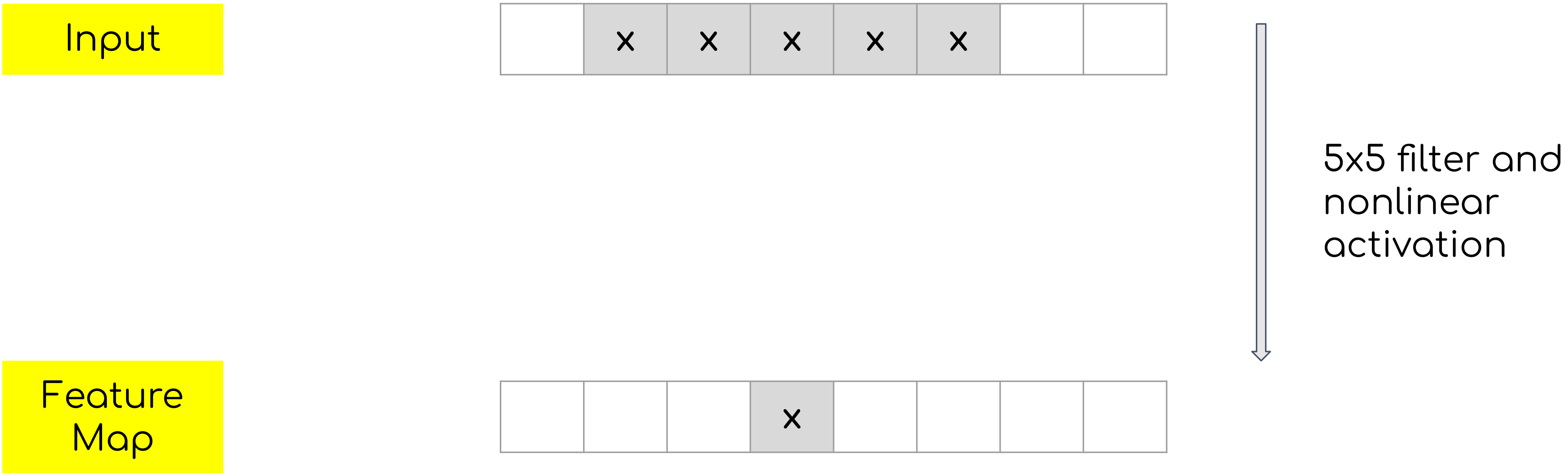
Use of multiple layers of 3x3 filters instead of 1 layer of 5x5 or 7x7 or 11x11



two successive
3x3 convolutions

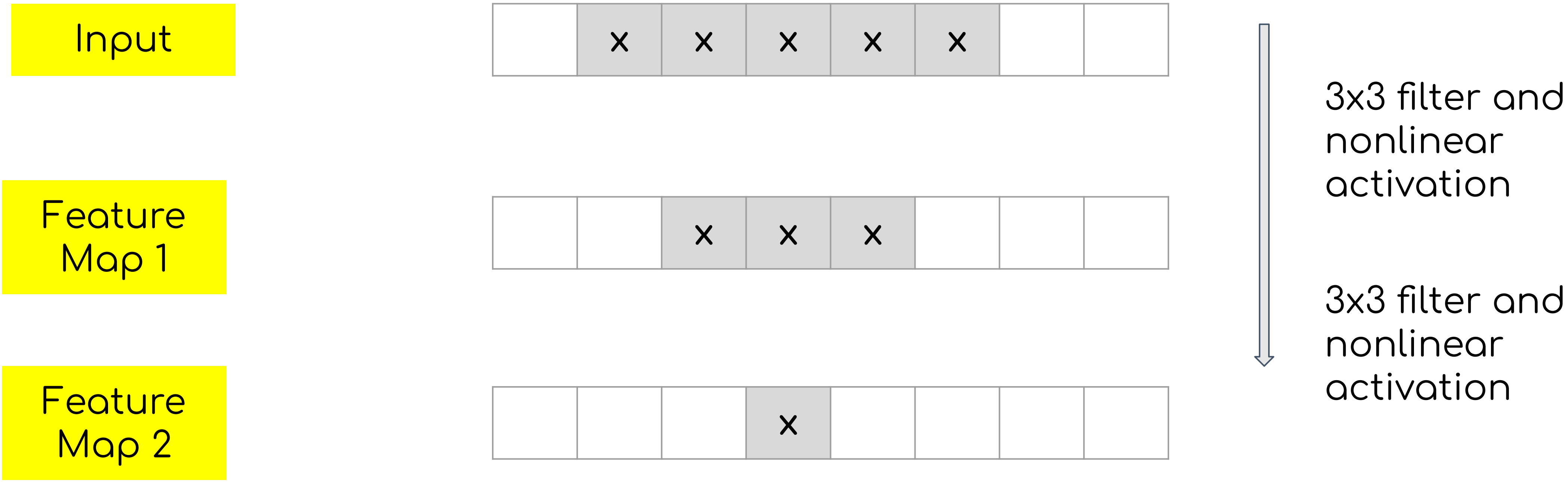
5x5 convolution

5x5 layer receptive field



Receptive field

Stacked 3x3 layer receptive field



Same Receptive field and more non-linearity

Parameters/Computations

What is the number of parameters and receptive field in the following two cases

Input channels = 32
conv-32, k=3x3, s=1,'relu'
conv-32, k=3x3, s=1,'relu'

Input channels = 32
conv-32, k=5x5, s=1,'relu'

Parameters/Computations

What is the number of parameters and receptive field in the following two cases

Input channels = 32

conv-32, k=3x3, s=1,'relu'

conv-32, k=3x3, s=1,'relu'

Input channels = 32

conv-32, k=5x5, s=1,'relu'

32x32x3x3 + 32x32x3x3 = 32x32x18

32x32x5x5 = 32x32x25

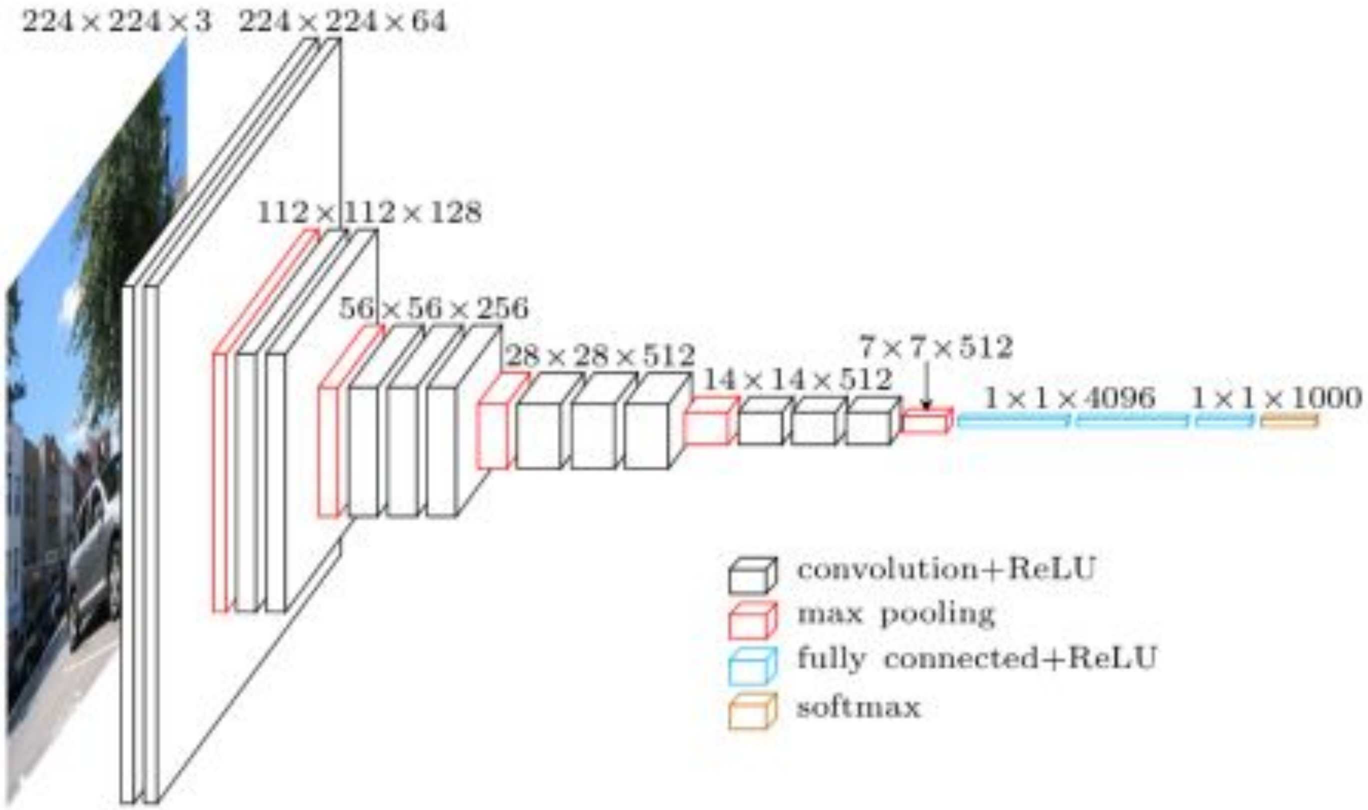
Remember both have same receptive field of 5 x 5 !

Different VGG architectures

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					

Architectures used in the VGG work

Increasing Filters with Depth



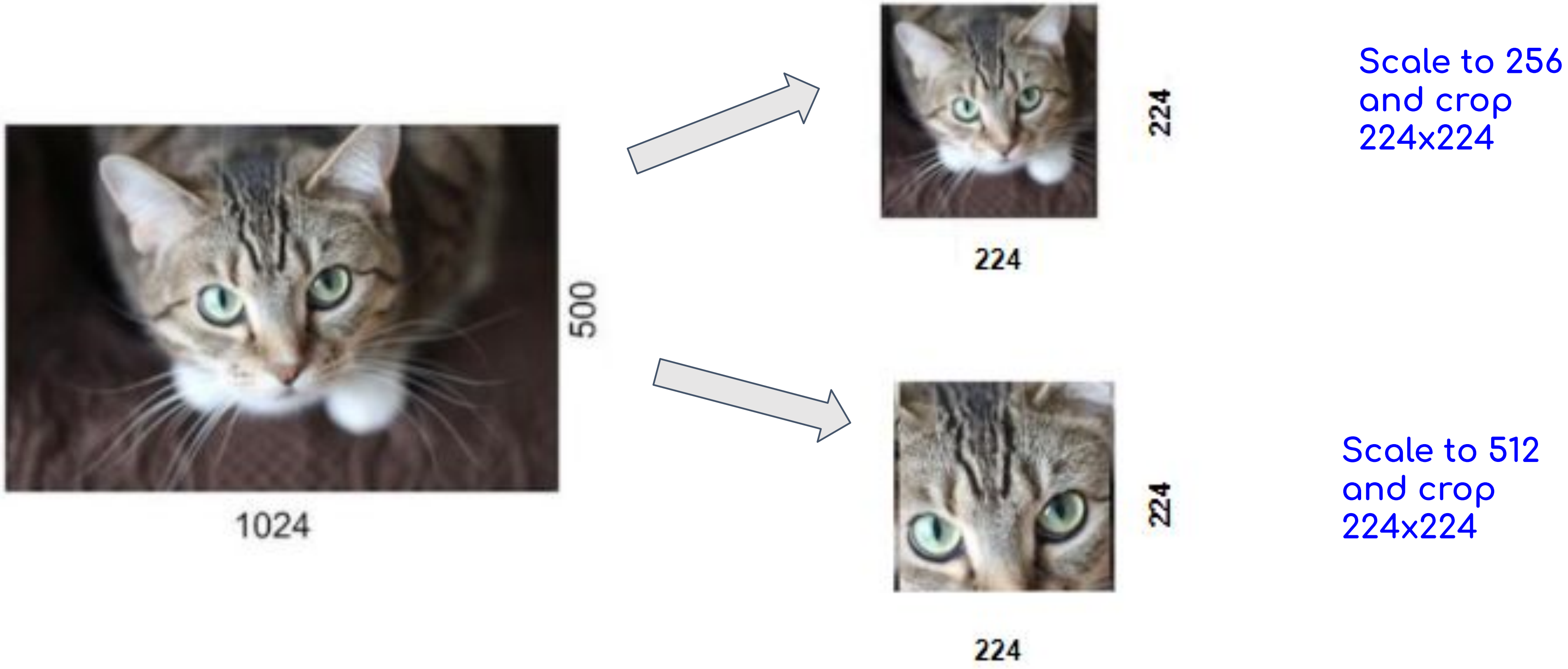
Initial layers encode low-level information, more spatial resolution, less depth
Upper layers encode high-level info, less spatial resolution, more depth

Maintain information content with decreasing spatial resolution

Rectangular Input Images: Cropping

Resize smaller side to 256 and crop at center to get 224x224

Multi-scale augmentation at train/test time



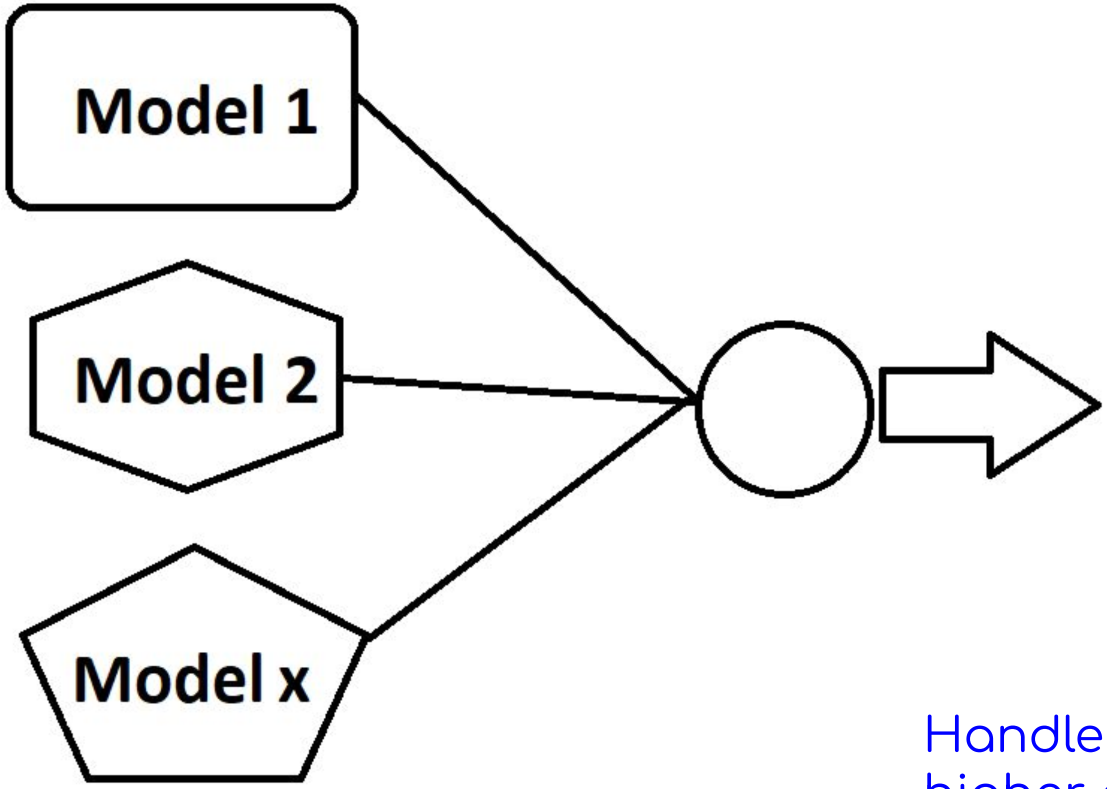
Resize smaller side to multiple scales in [256,512] and crop to get 224x224

Multi-scale augmentation at train/test time

Same image gives different scaled and cropped/shifted versions

Model ensembling

Average prediction probabilities from multiple models (VGG-16, VGG-19)



Handle Overfitting issues, higher accuracy

Summary

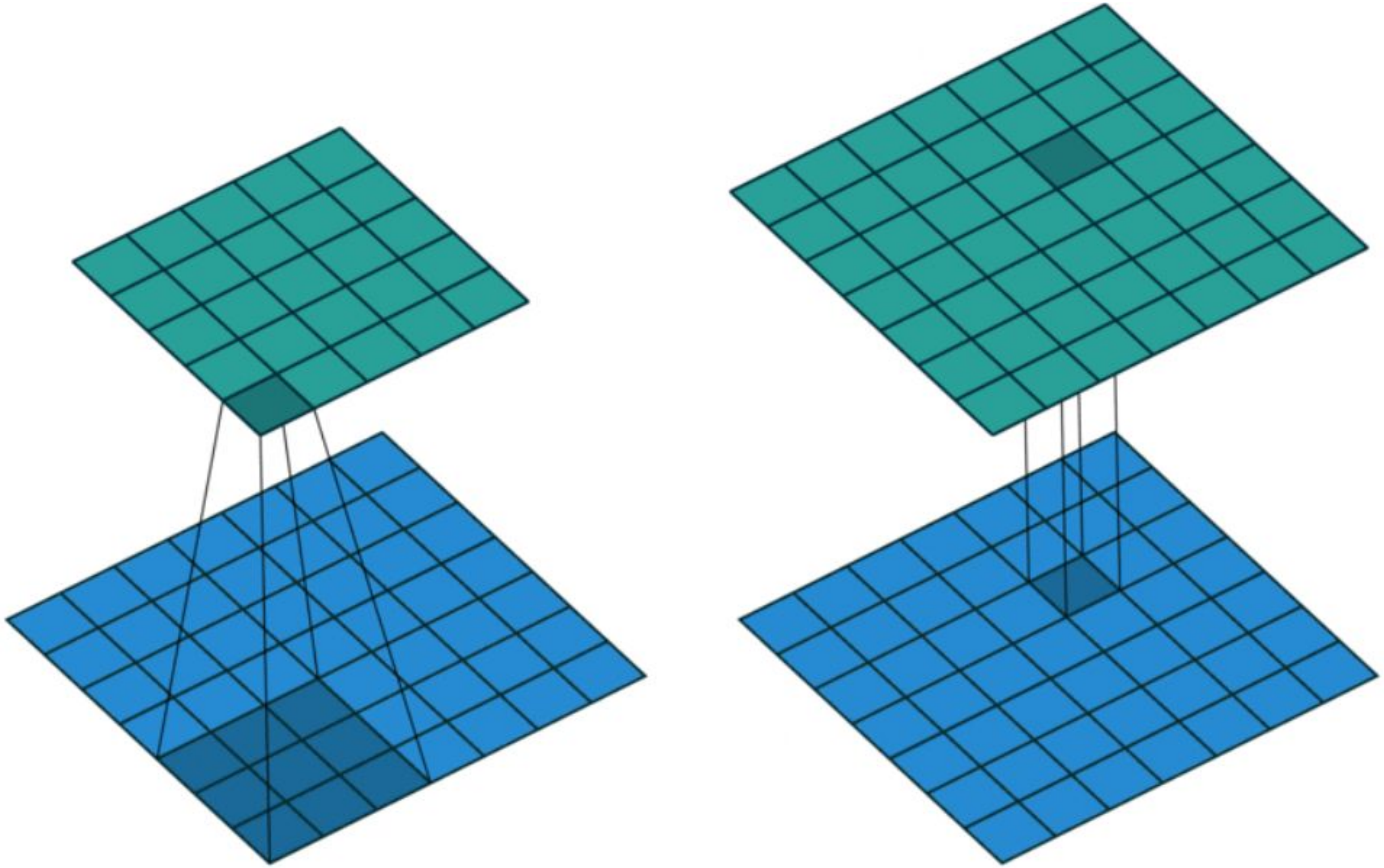
- The use of only 3x3 sized filters as against AlexNet's 11x11 filters in the first layer.
- Increasing filters with depth
- Used scale variation as one data augmentation technique during training and testing.
- Model ensembling for best results
- The top-5 test error on ImageNet was 7.3%

GoogLeNet/Inception-v1

Key Features

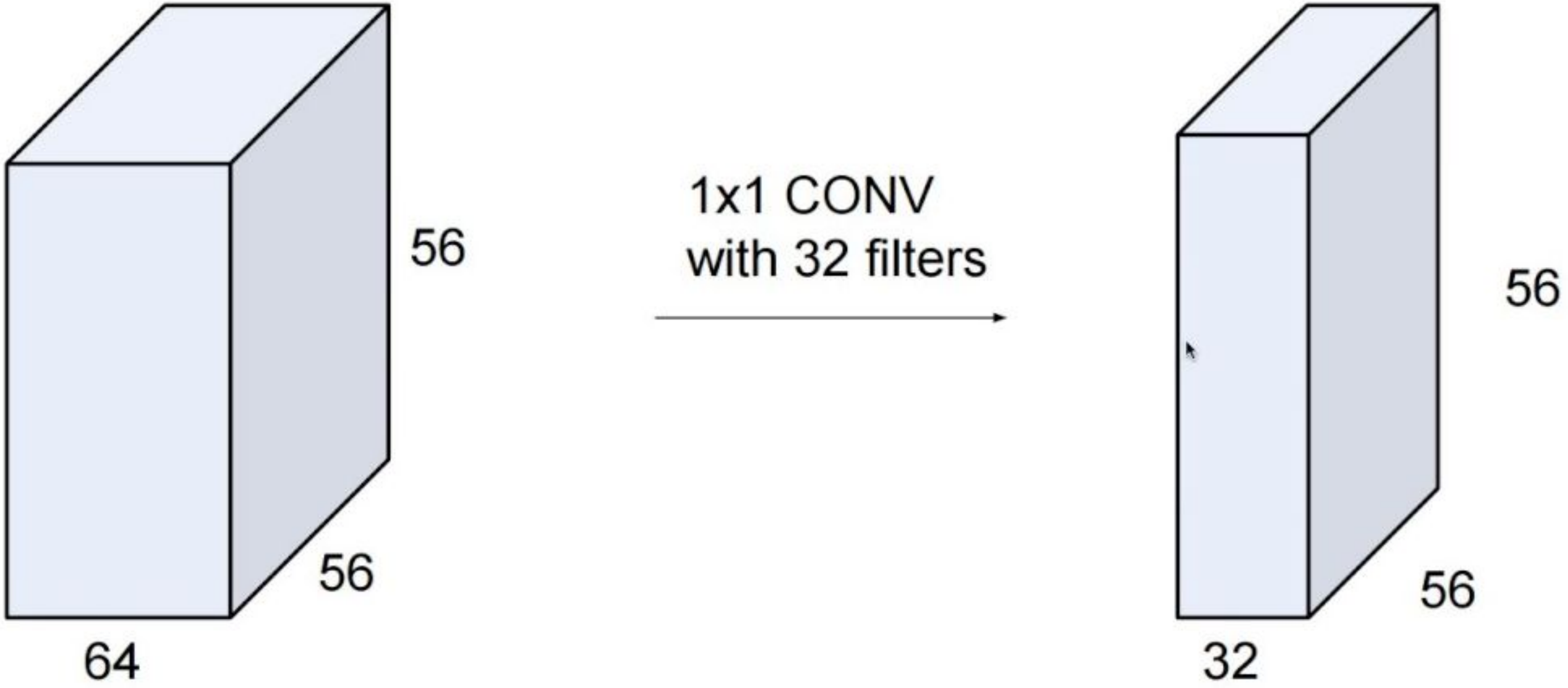
- Deeper Network with 22 layers
- Efficiently designed “Inception Module”
- No FC layers
- Only 5Million parameters (12x less than AlexNet)
- 6.7% Top 5 error in ILSVRC

1x1 Convolution filters



1x1 filters

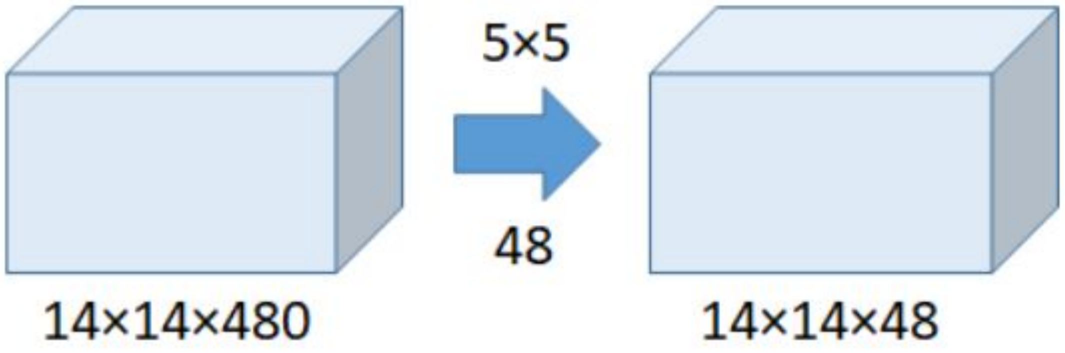
Each output feature map is a linear combination of input feature maps followed by non-linear activation



H = W = 56
Input Depth =64

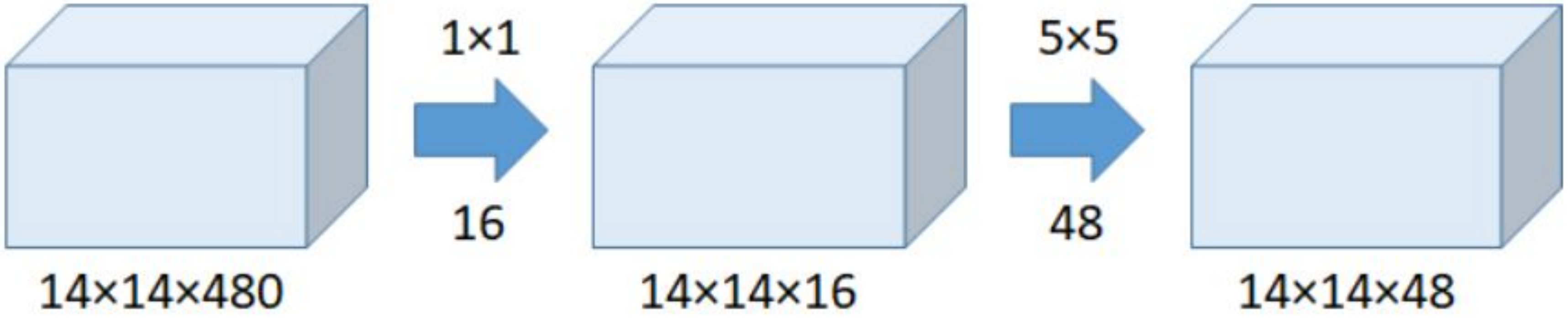
- Easy way to get Feature reduction/increase, additional non-linearity
- Preserves Spatial Dimensions & Reduces the depth

Reduction of parameters



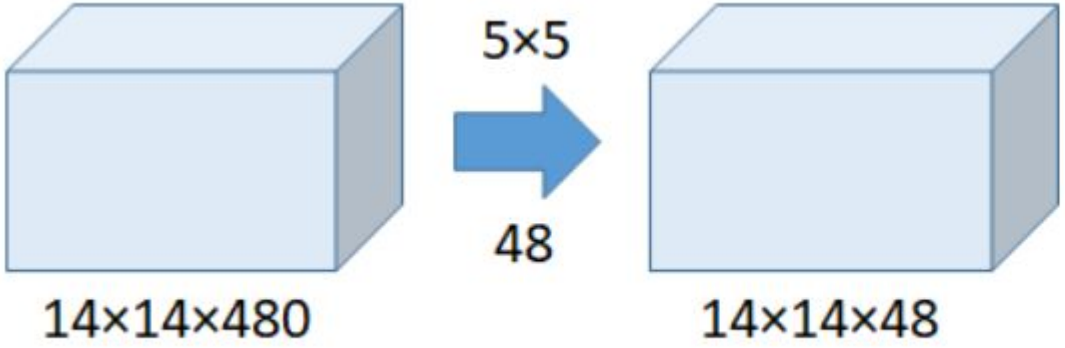
Without the Use of 1×1 Convolution

#Parameters - 48 x 480 x 5 x 5 = **0.5 M**
#OPs - 14x14x480x5x5x48 = **113M**



With the Use of 1×1 Convolution

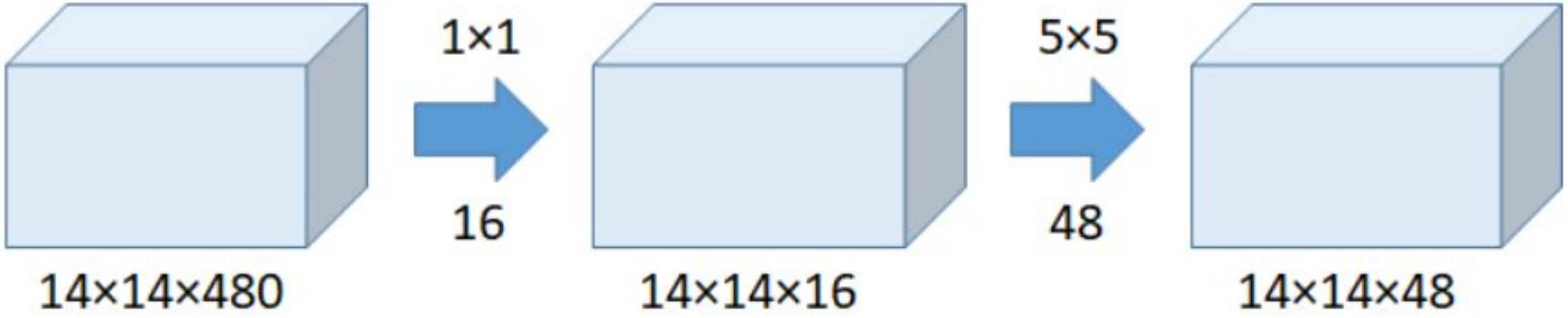
#OPs for the below network - ?



Without the Use of 1×1 Convolution

#Parameters - 48 x 480 x 5 x 5 = **0.5 M**

#OPs - 14x14x480x5x5x48 = **113M**

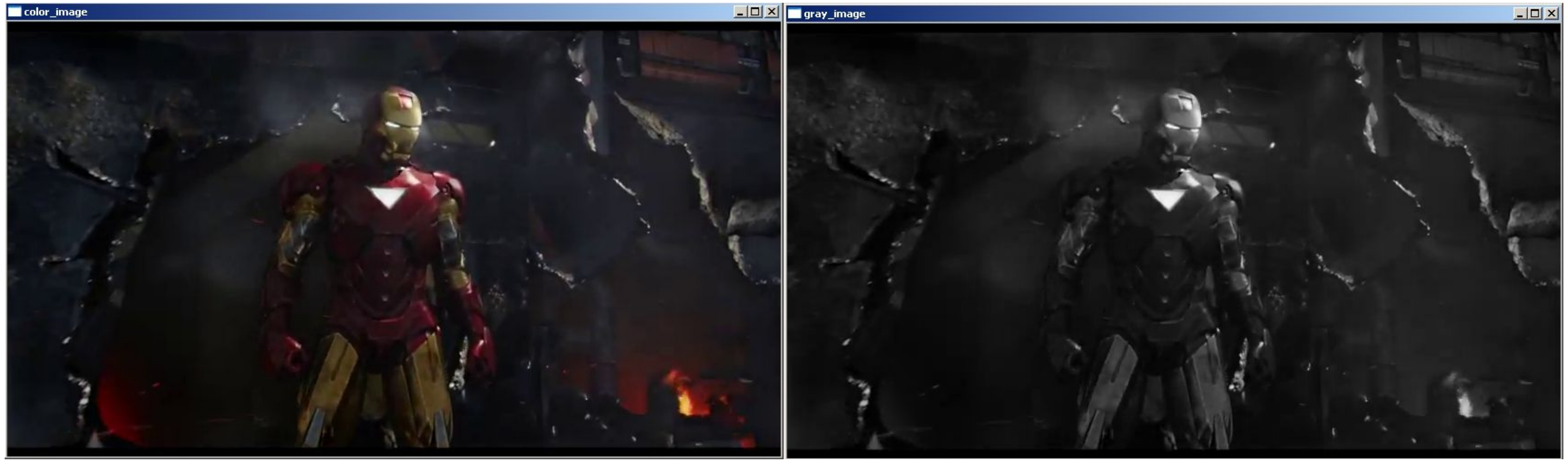


With the Use of 1×1 Convolution

#OPs - 14x14x480x16 + 14x14x16x5x5x48 = **5.3 M**

Possible ways to derive the Output feature map

The Object is identifiable by just a linear combination of input features/channels



Objects in an image is small requiring small kernel size

224

224

Objects could be bigger requiring a larger sized kernel

224

224

Do we need to focus on a lower resolution or same resolution for classification ?

Do we need Pooling or not?

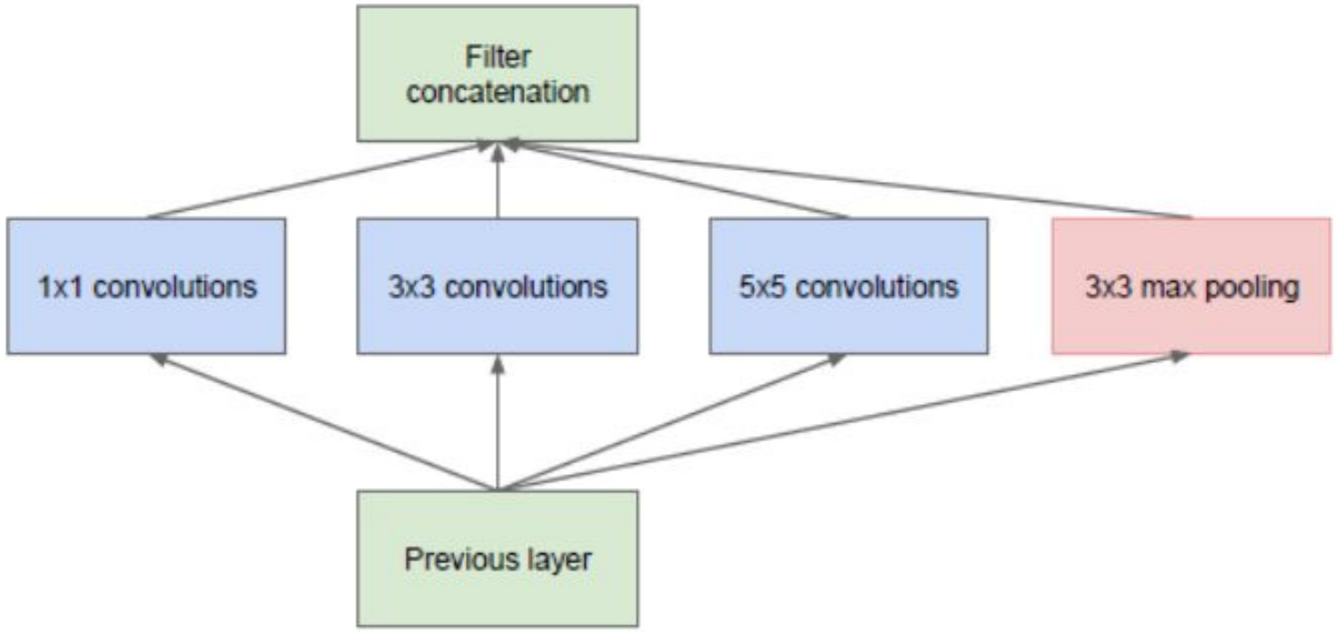
Thus, at every layer, there is a design choice on

a) linear combination of input maps
b) size of kernels
c) Whether or not to do Pooling

Can we use data/optimization to choose on what is important for a layer ?

Inception Block !

Inception Block



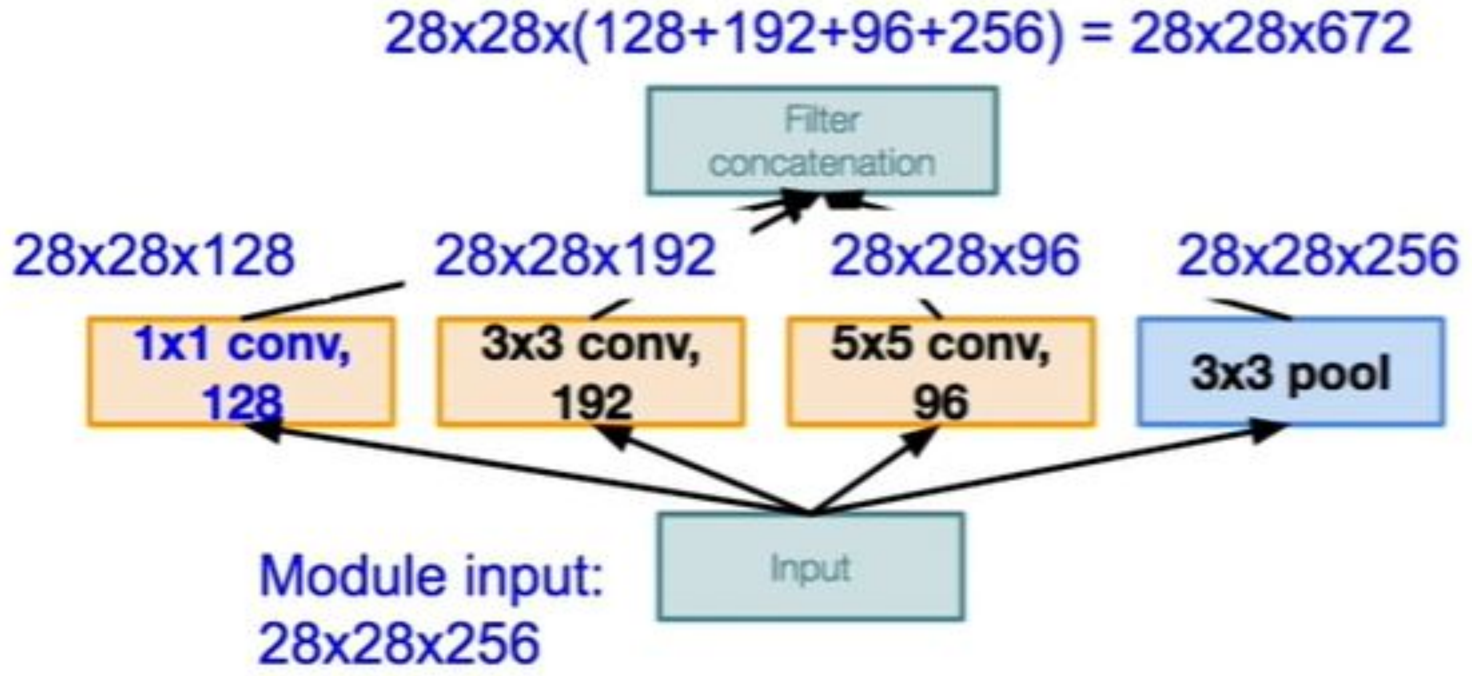
(a) Inception module, naïve version

Local Topology (Network within a Network)

Offers design choice on

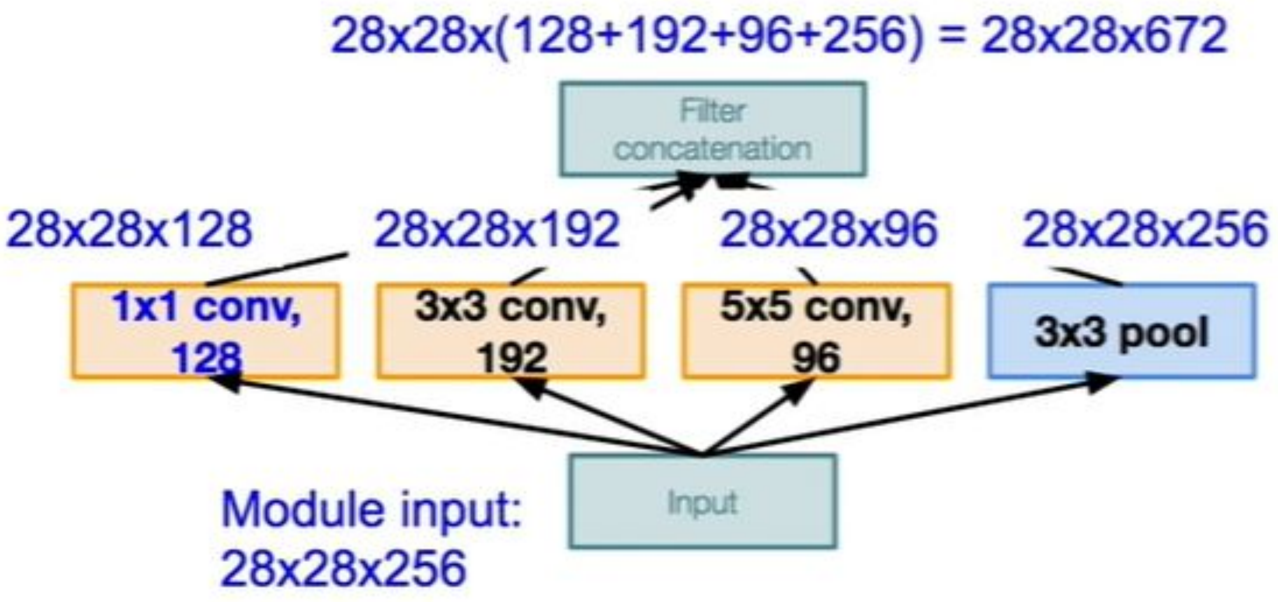
a) linear combination of input maps
b) size of kernels 3x3 or 5x5 or combinations
c) Whether or not to do Pooling

Many parameters and Expensive



Naive Inception module

Many parameters and Expensive



Naive Inception module

Conv Ops:

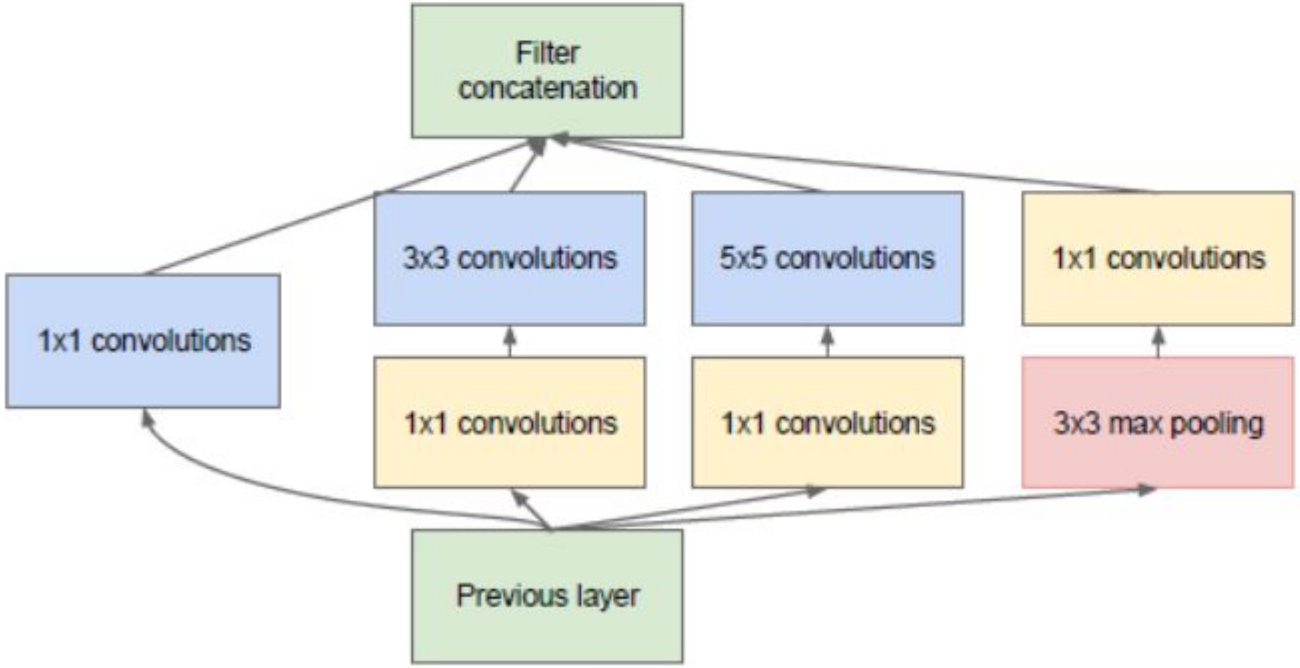
[1x1 conv, 128] 28x28x128x1x1x256

[3x3 conv, 192] 28x28x192x3x3x256

[5x5 conv, 96] 28x28x96x5x5x256

Total: 854M ops

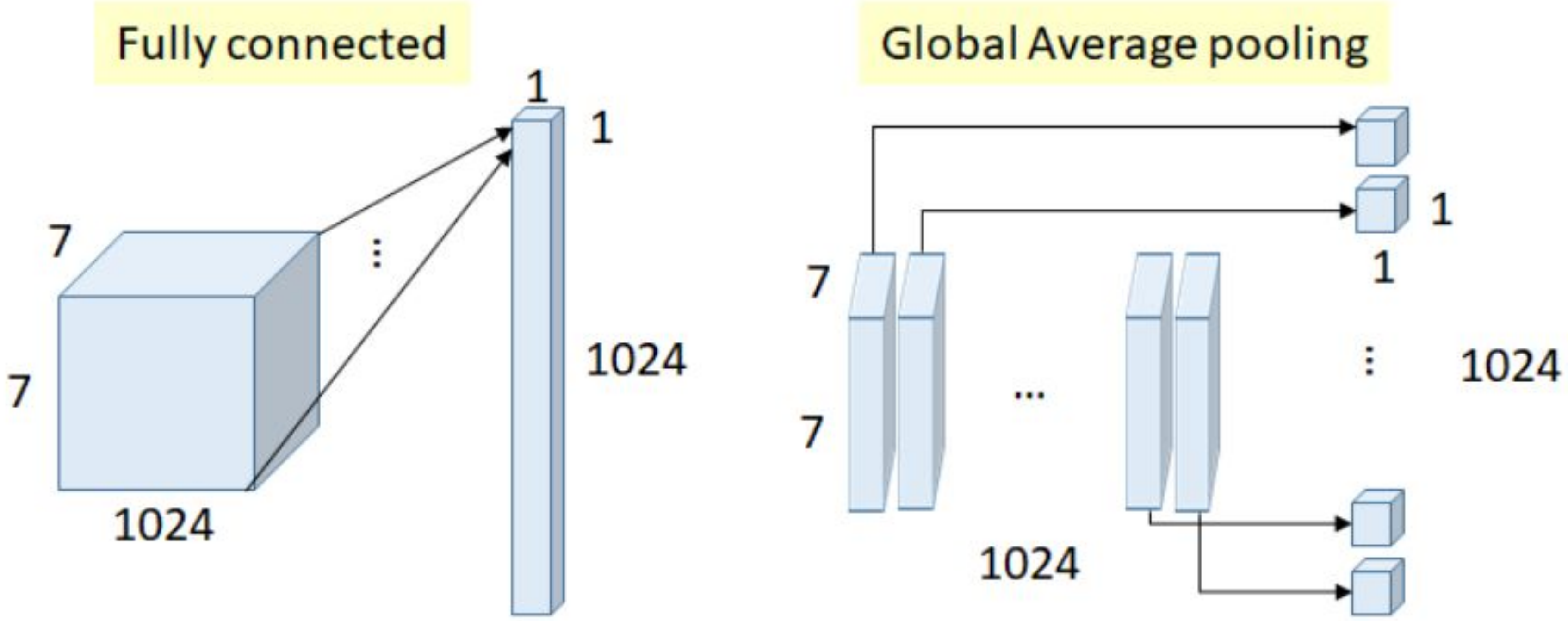
Use 1x1 conv to reduce parameters & speed



(b) Inception module with dimensionality reduction

Modified

Global Average Pooling

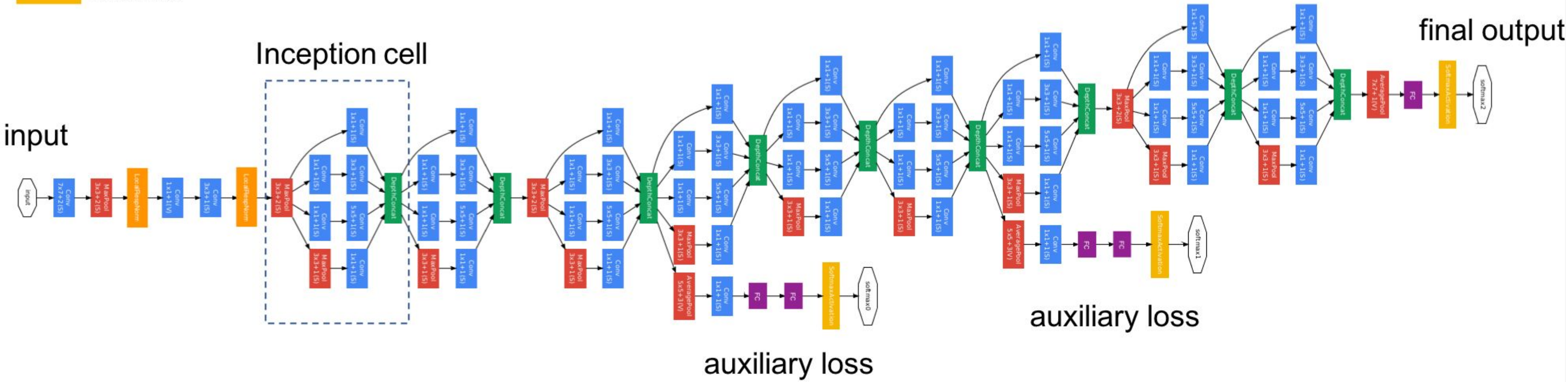


Fully Connected Layer VS Global Average Pooling

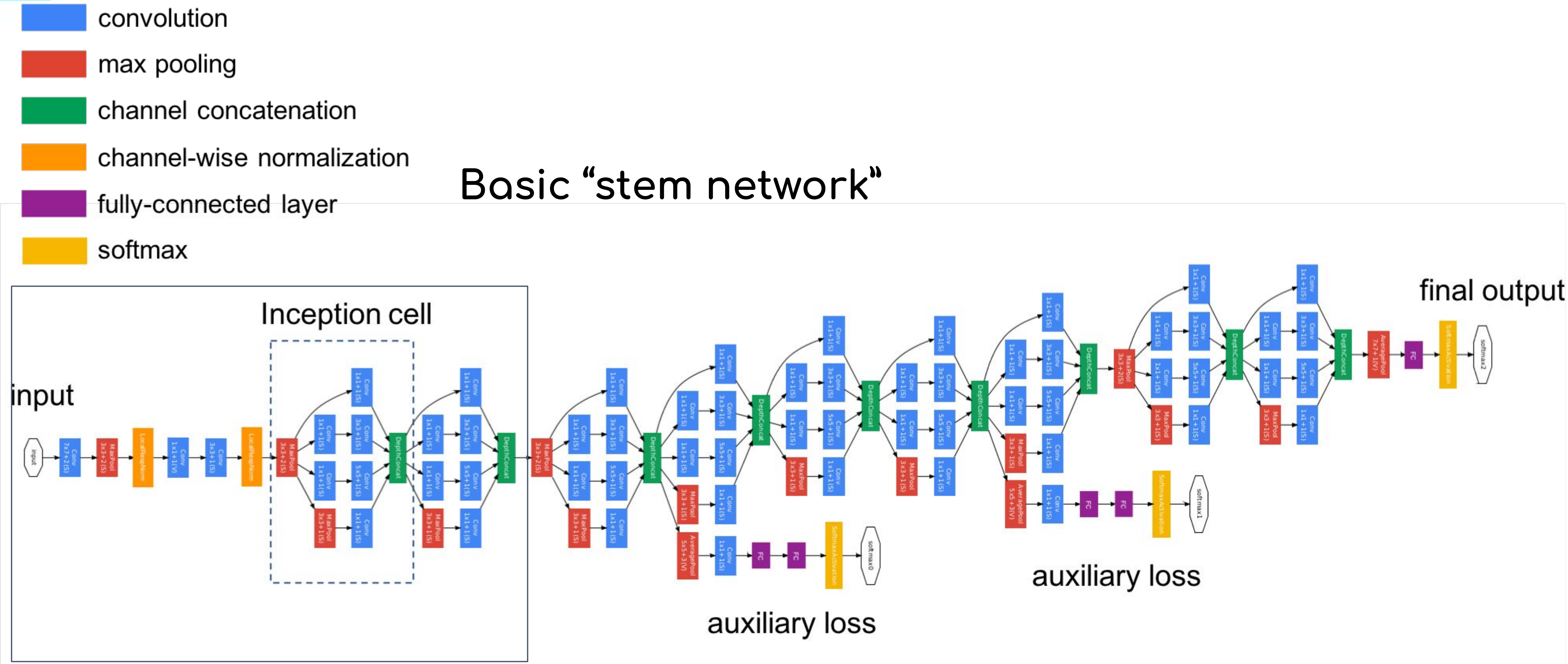
FC parameters: 7x7x1024x1024 = **50M,**
GAP parameters: **0**
Much less parameters using GAP, less overfitting!

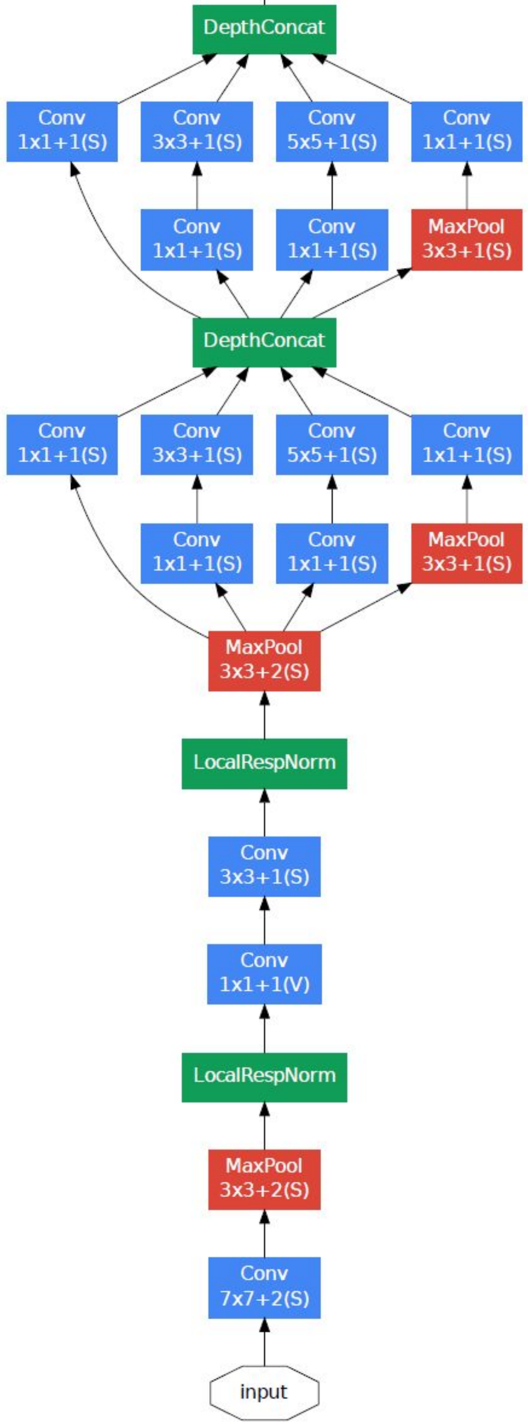
Overall Architecture

- convolution
- max pooling
- channel concatenation
- channel-wise normalization
- fully-connected layer
- softmax



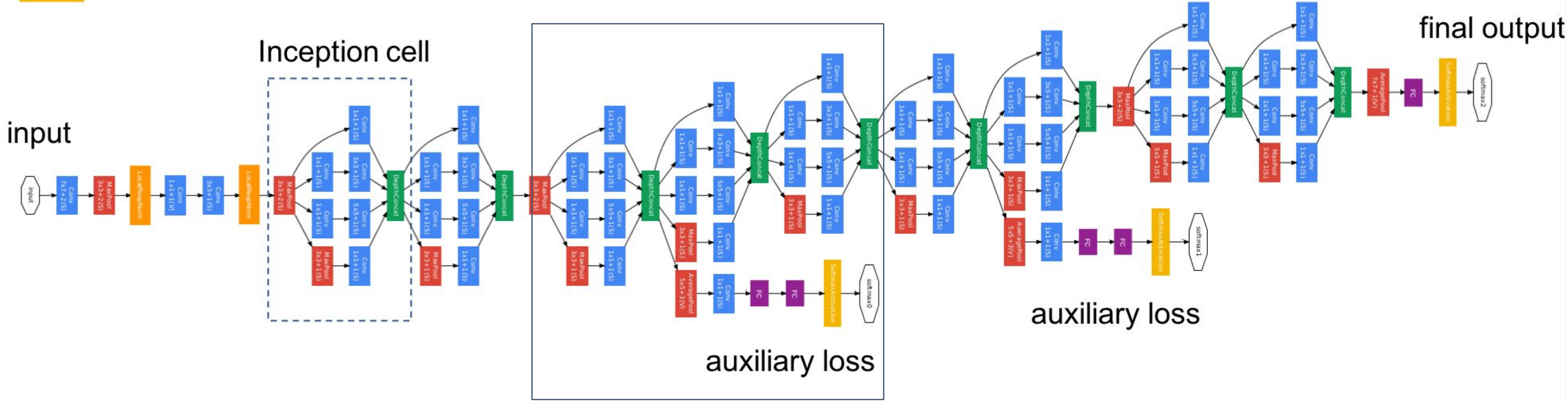
Overall Architecture

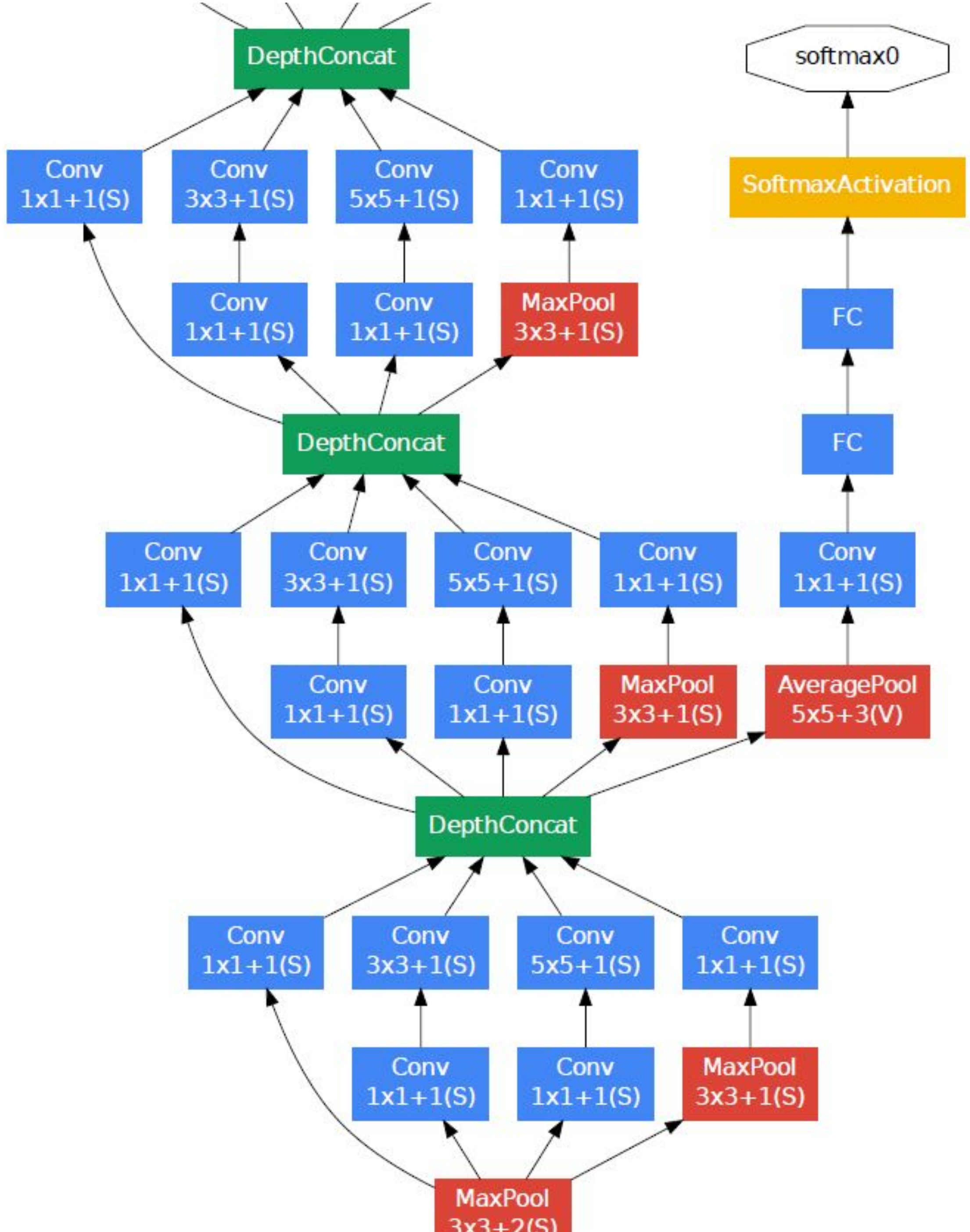



DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 3x3+1(S)
Conv 1x1+1(V)
LocalRespNorm
MaxPool 3x3+2(S)
Conv 7x7+2(S)
input

Overall Architecture

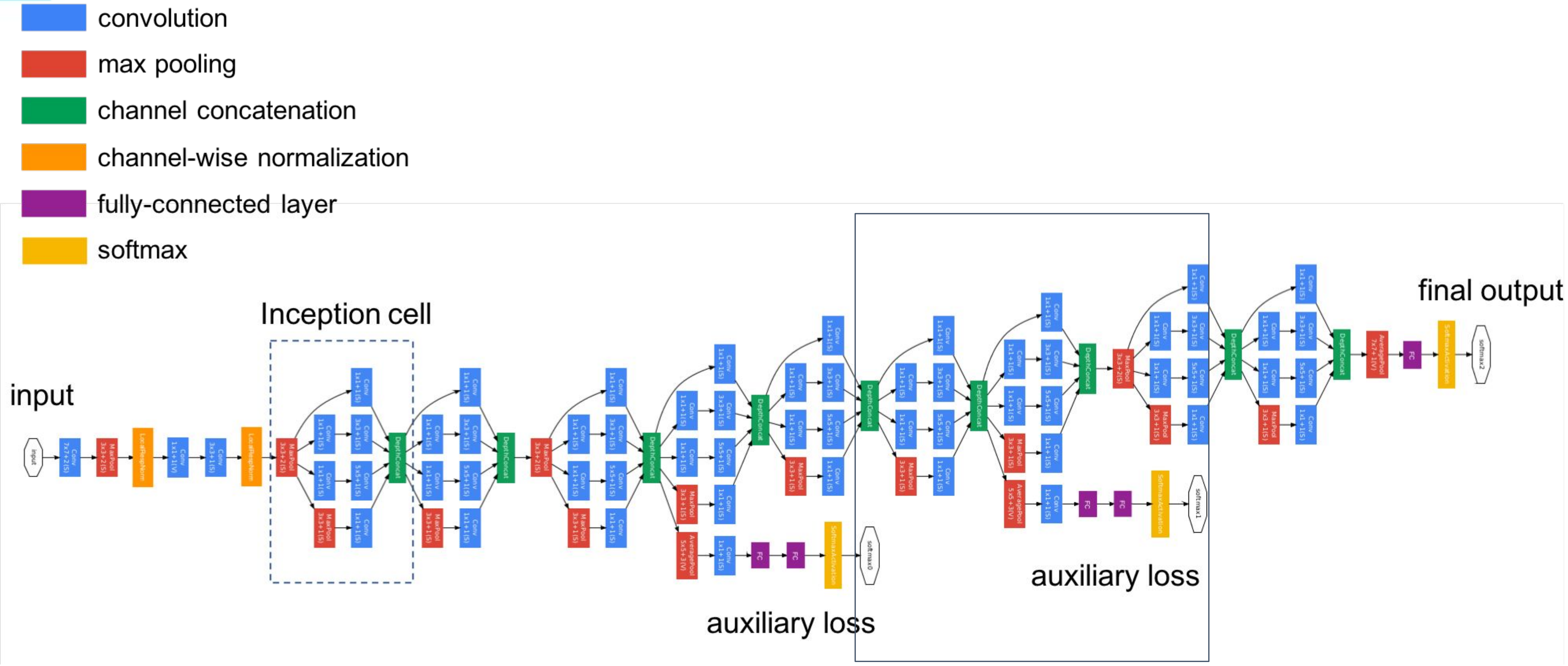
- convolution
- max pooling
- channel concatenation
- channel-wise normalization
- fully-connected layer
- softmax





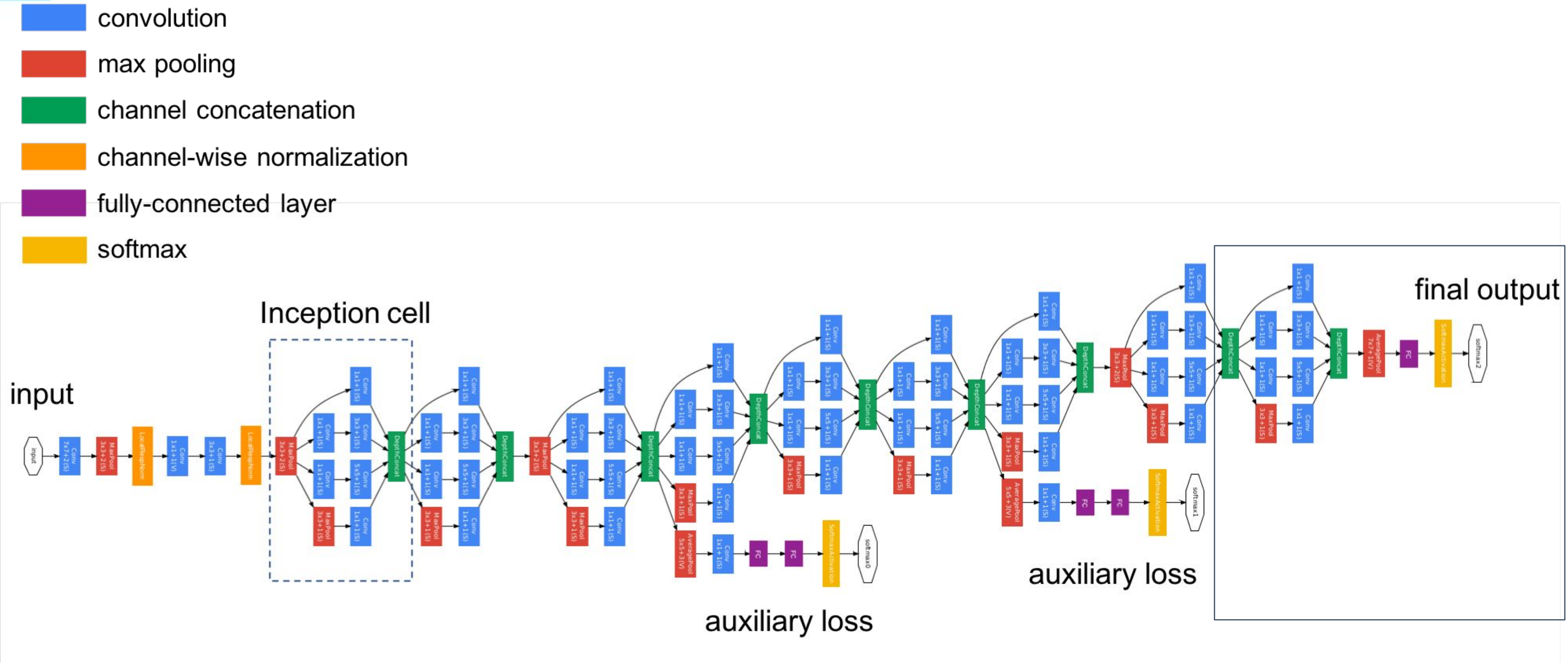
Use this to avoid VG effect in middle layers

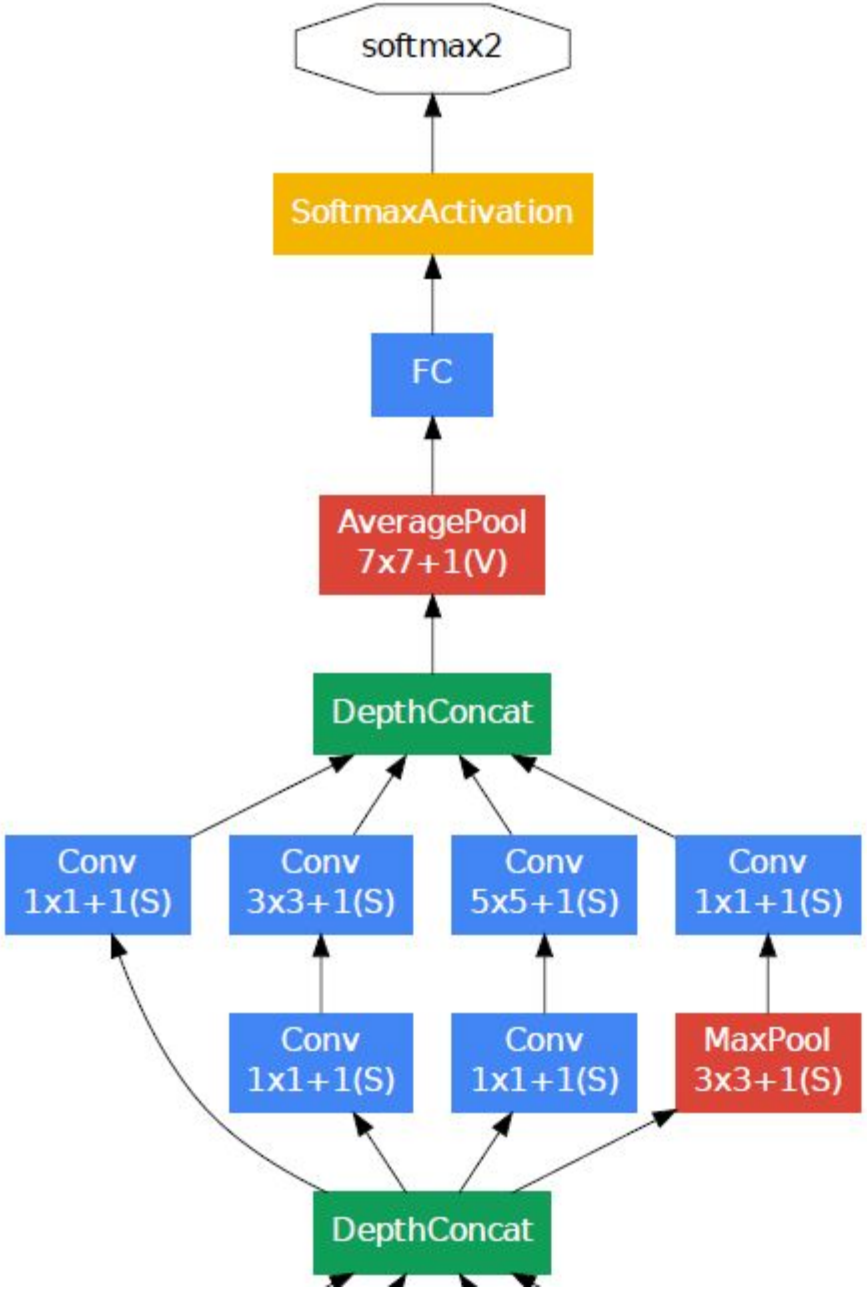
Overall Architecture




DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
softmax1
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
SoftmaxActivation
MaxPool 3x3+2(S)
FC
DepthConcat
FC
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
softmax0

Overall Architecture





softmax2
SoftmaxActivation
FC
AveragePool
7x7+1(V)
DepthConcat
Conv
1x1+1(S)
Conv
3x3+1(S)
Conv
5x5+1(S)
Conv
1x1+1(S)
Conv
1x1+1(S)
Conv
1x1+1(S)
MaxPool
3x3+1(S)
DepthConcat

Summary

- Used 9 Inception modules, 100 layers in total
- No use of fully connected layers. This saves a huge number of parameters.
- Uses 12x fewer parameters than AlexNet.
- During testing, multiple crops of the same image were created, fed into the network, and the softmax probabilities were averaged to give us the final solution.
- Trained on “a few high-end GPUs **within a week**”.
- Imagenet Top-5 error rate down to **6.66%** from **7.32 %** (VGG)

Residual Networks

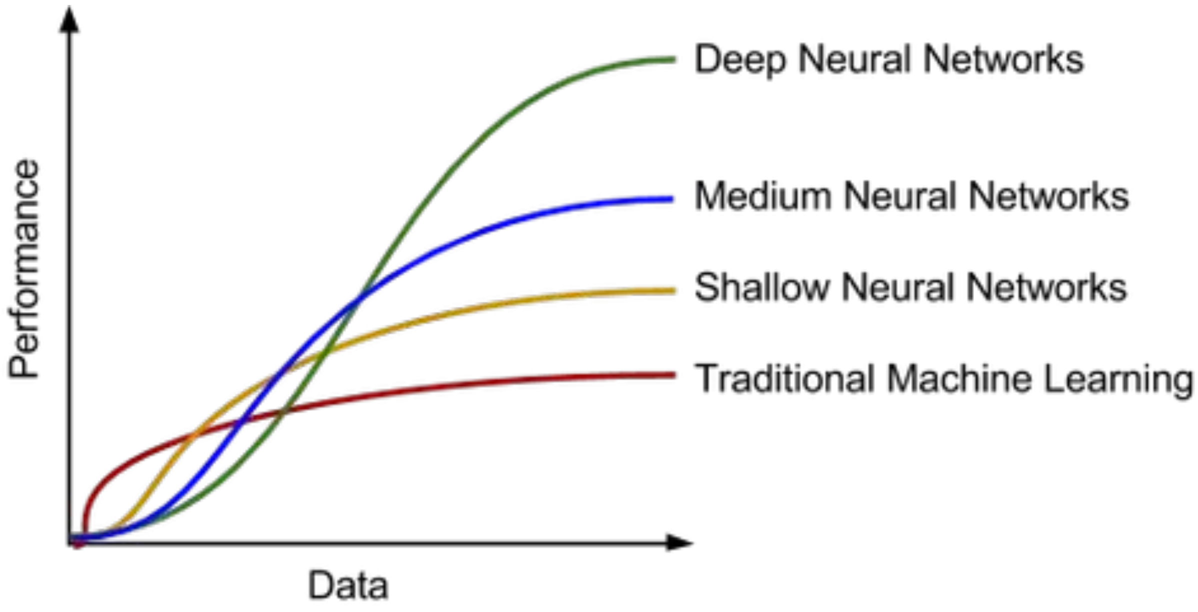
Were the first to train really deep networks (150 layers, 1000 layers)

Imagenet error rate down to **3.57%** from **7.32 %** (VGG)

Very key idea of Residual connections

Deep Residual Networks

- Neural Networks with just 1 hidden layer are universal approximators
- Efficient representation is important for managing computational requirements, robust learning and preventing overfitting
- An important element of representation is depth of the network
- The benefit of depth has been successfully demonstrated previously in AlexNet, VGG



Advantages of greater Depth

- Representation complexity grows exponentially w.r.t hidden units compared to shallow networks
- Thus for same number of parameters, Deep networks allow for more complex representation
- Deep CNN networks with small filters (e.g. 3x3, 1x1) have lesser parameters/faster compute for same receptive field

Challenges of Deep Networks-Hard to train

- Vanishing gradients issue
  - RELU alleviates this issue to some extent

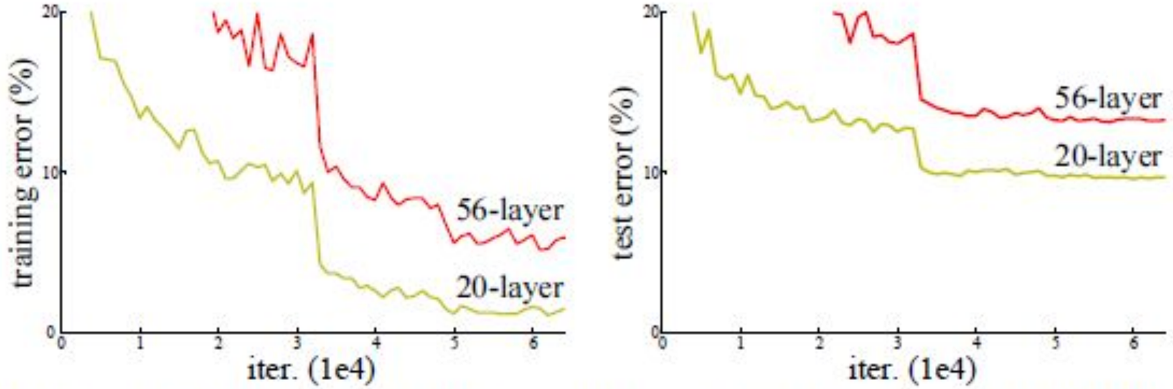


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Degradation in training
  - Increased non-convexity, harder to train
  - Simple maps like the identity map hard to converge

Residual Block: Skip connection

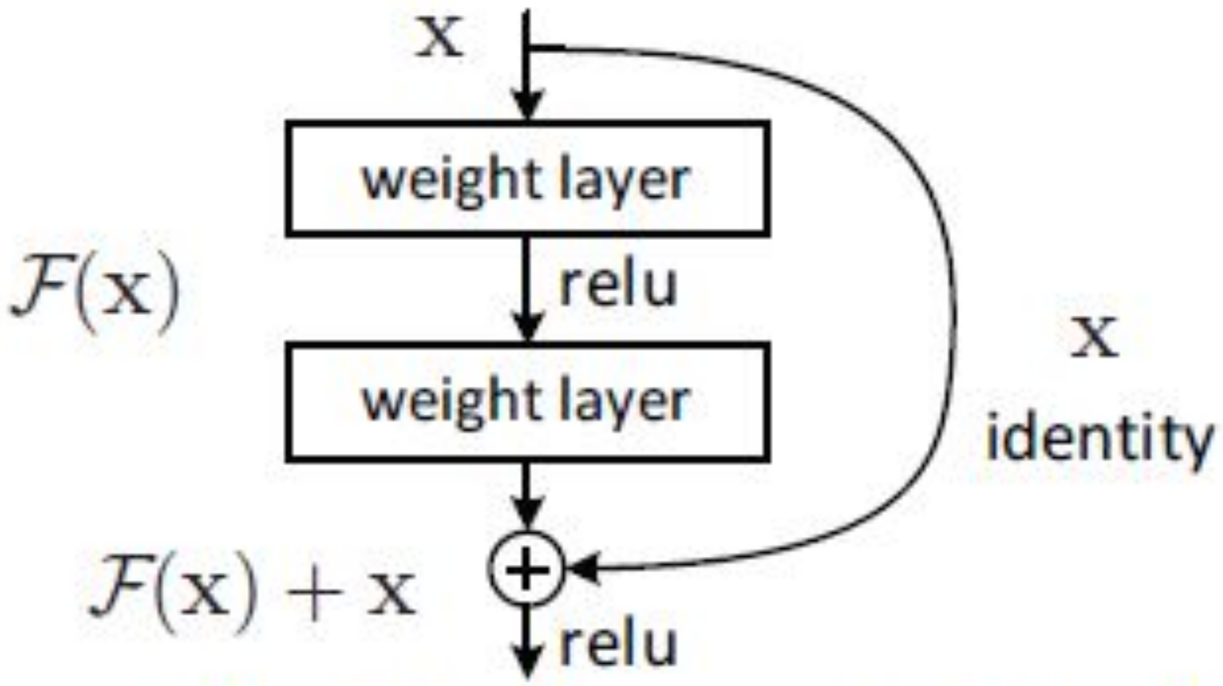
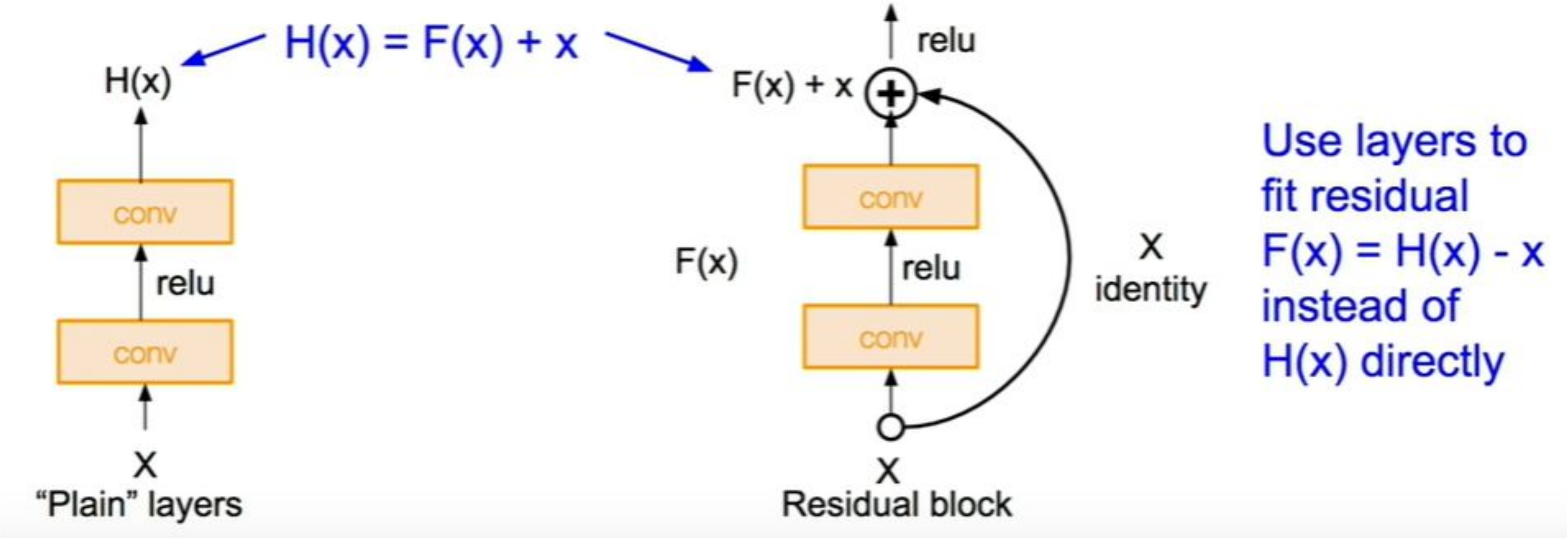


Figure 2. Residual learning: a building block.

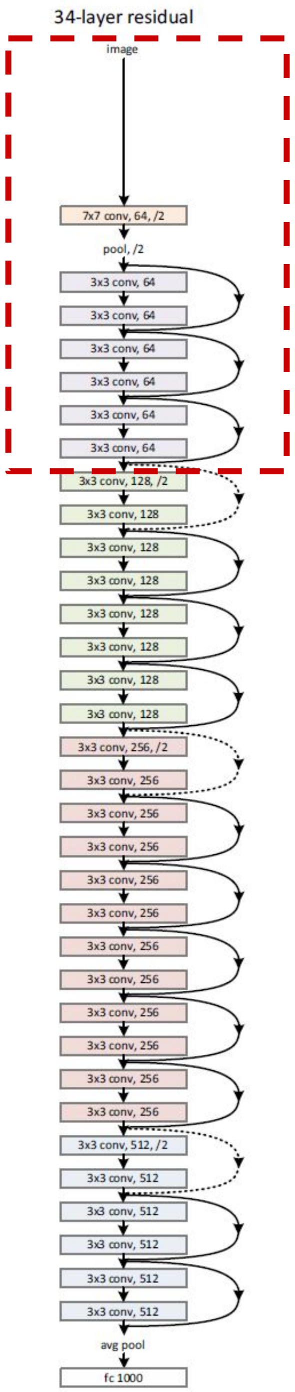
In layers, learn Residuals rather than the original function
As motivated earlier, easier to optimize since derivatives are well behaved

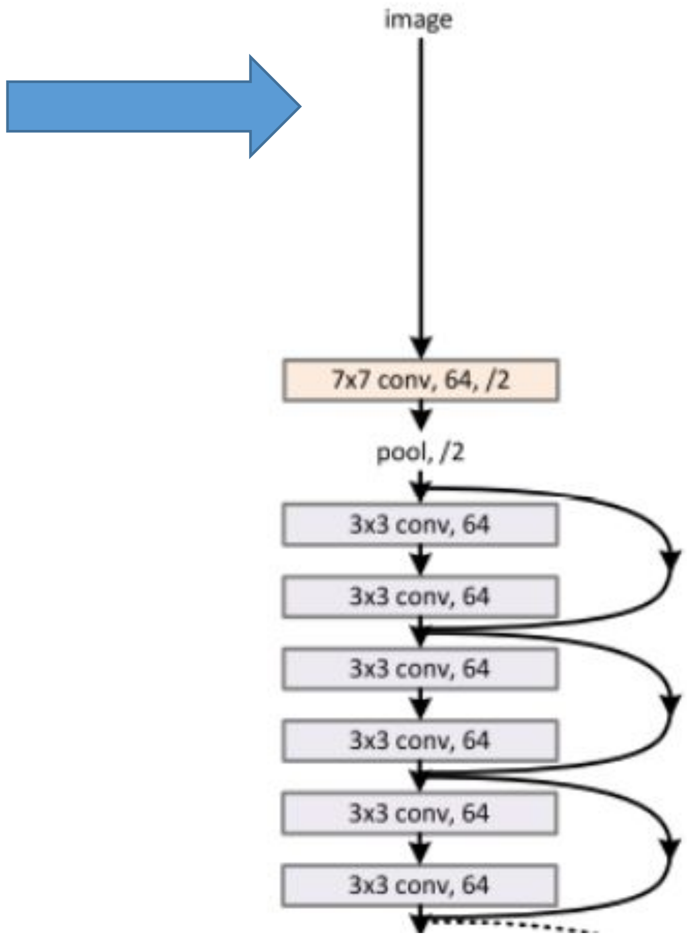
Residual Block: Skip connection

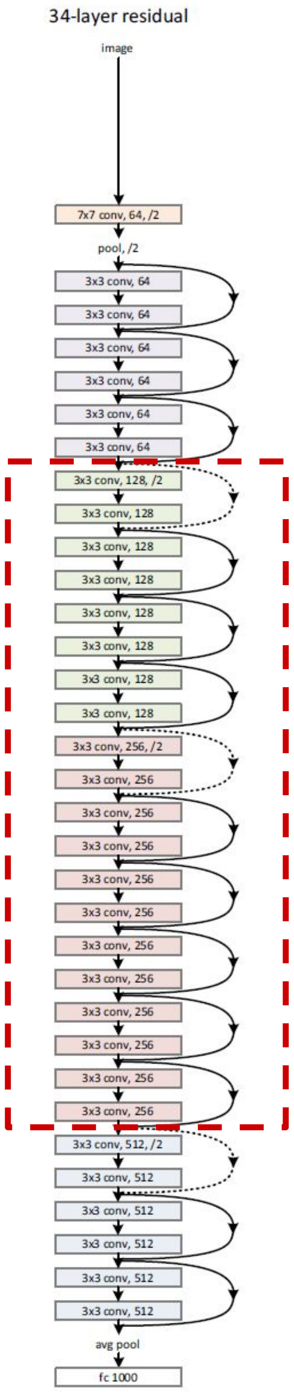


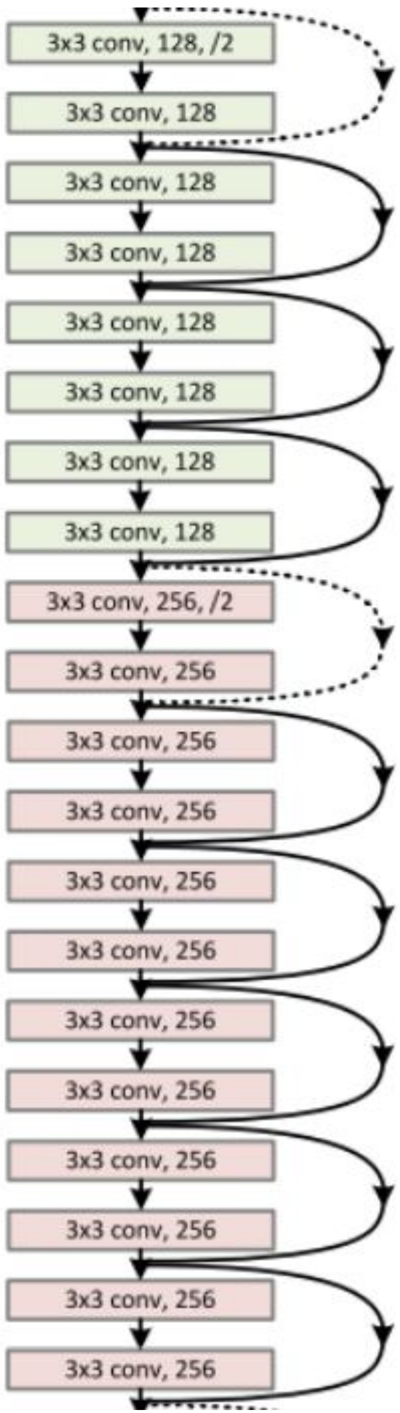
Typically the Residuals are small...
Thus, by stacking more layers, worst case is, we learn the identity. Earlier, the entire layer would collapse!

Basic Architechure- Resnet 34



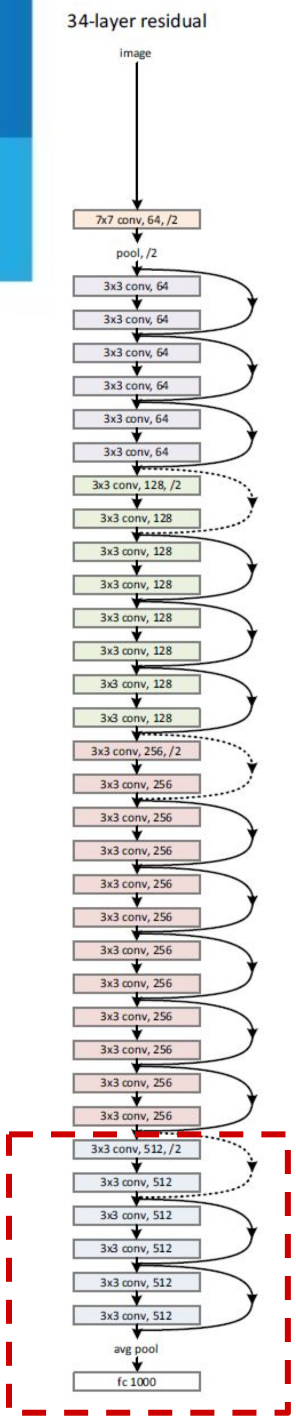


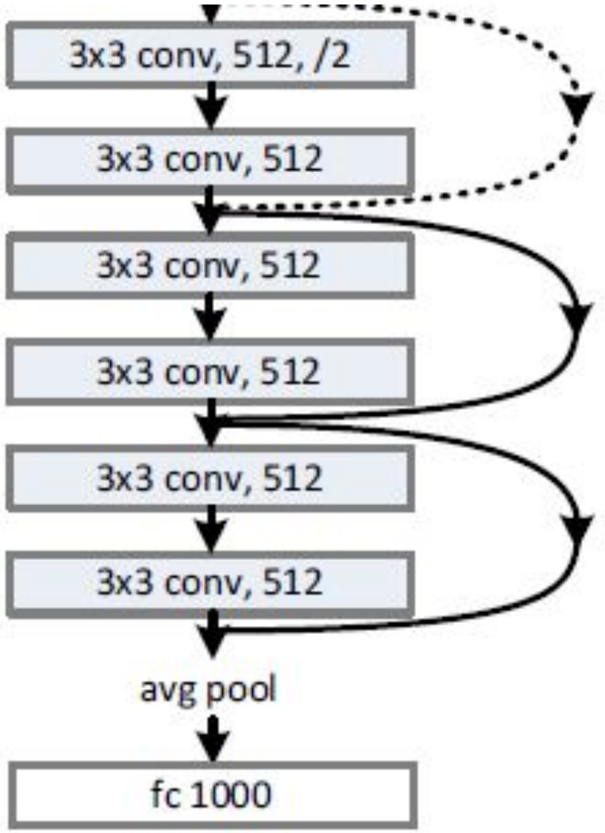




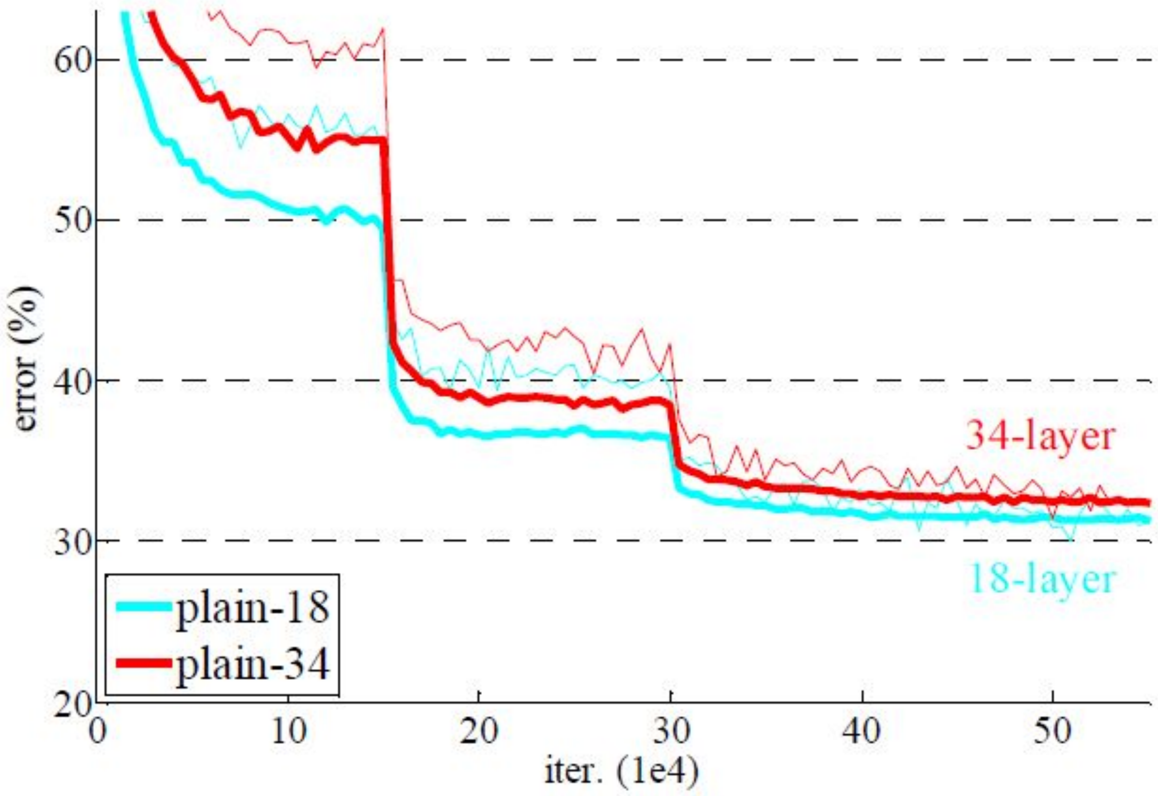
Basic Architechure- Resnet 34

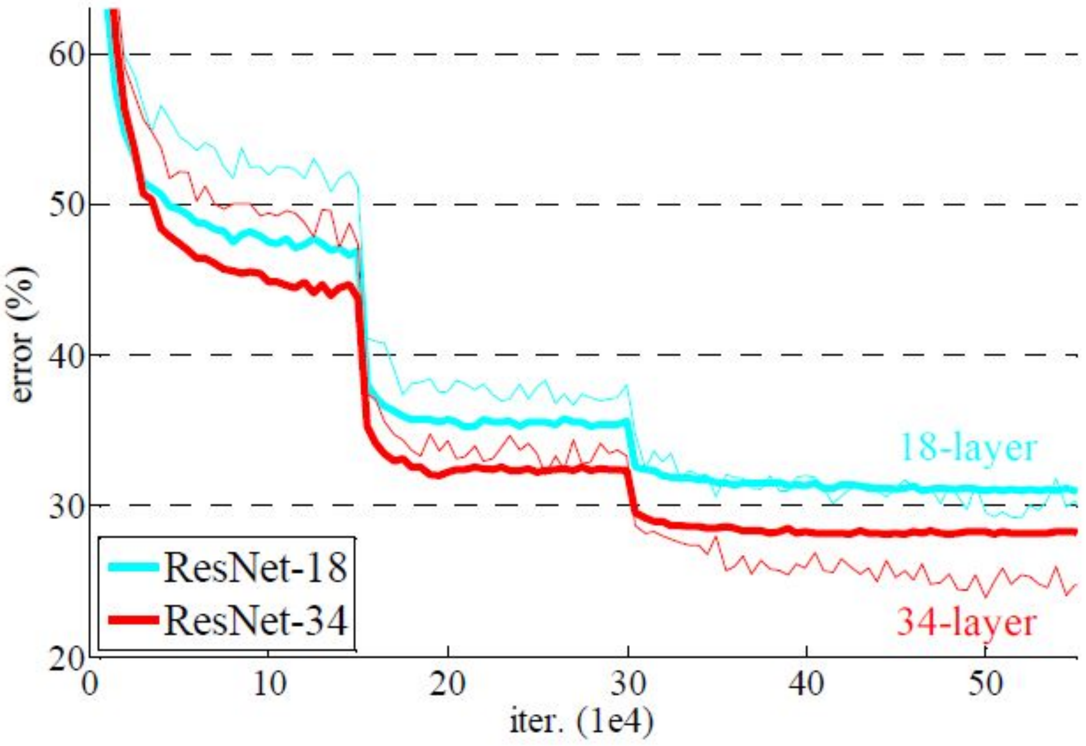
Basic Architecture- Resnet 34





Improved Training and Test Accuracy





Summary

- Skip connections for training very deep networks
- Scale/horizontal flip data augmentation
- Batch normalization
- Dropout not used
- Fully convolutional output
- Multi-crop/multi-scale prediction and averaging testing
- Imagenet error rate down to **3.57%** from **7.32 %** (VGG)

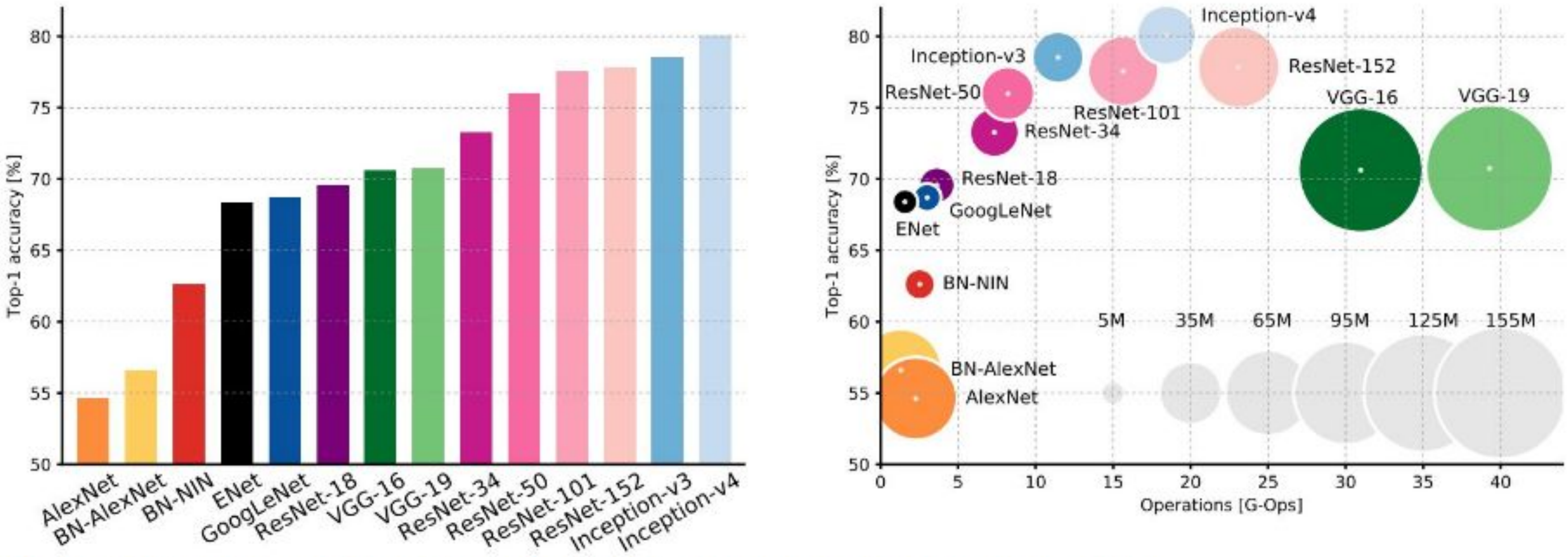
Key Achievements by Resnet

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

State of Art CNN architectures (Recap)

Performance trends (ImageNet (https://en.wikipedia.org/wiki/ImageNet)



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