Computer Vision – CNN Architectures

Fundamental CNN Architectures and best practices

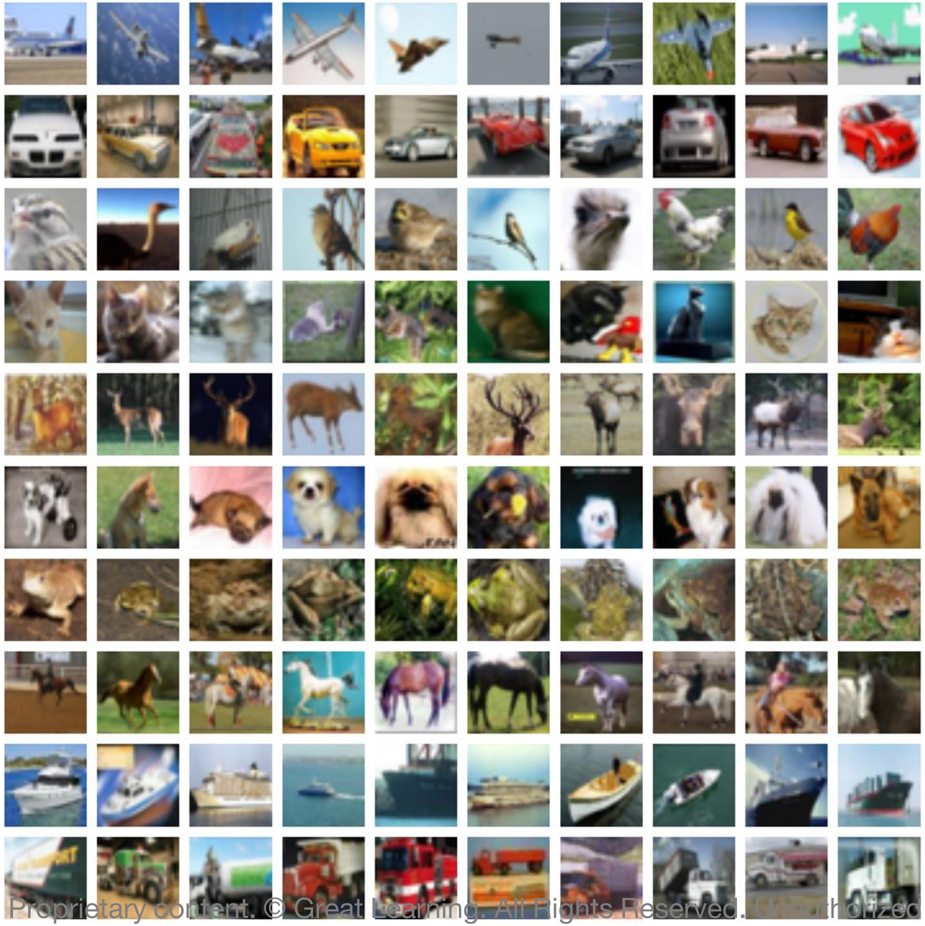
- Given an application, several architecture design choices
  - #layers, #filters, kernel size, pooling, fully connected layers, regularizers etc.

- Look at competitions in related domains and start with these network designs, follow best practices

- Are there ways to reuse these open source networks trained on related domains without having to train from scratch?

ImageNet (ILSRC)

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



40 M images from 20k categories !

Top-5 error on 100k test images

greatlearning
Learning for Life

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 and best practices

AlexNet
VGG
ResNet
GoogLeNet (Inception-v1)
Inception-v2, Inception-v3, Inception-v4
Inception-ResNet
ResNext
DenseNet
……

Deep dive today

AlexNet(2012)

1. Around 2011, a good ILSVRC classification error rate was **25%**. In 2012, AlexNet achieved **16%**, a watershed moment!

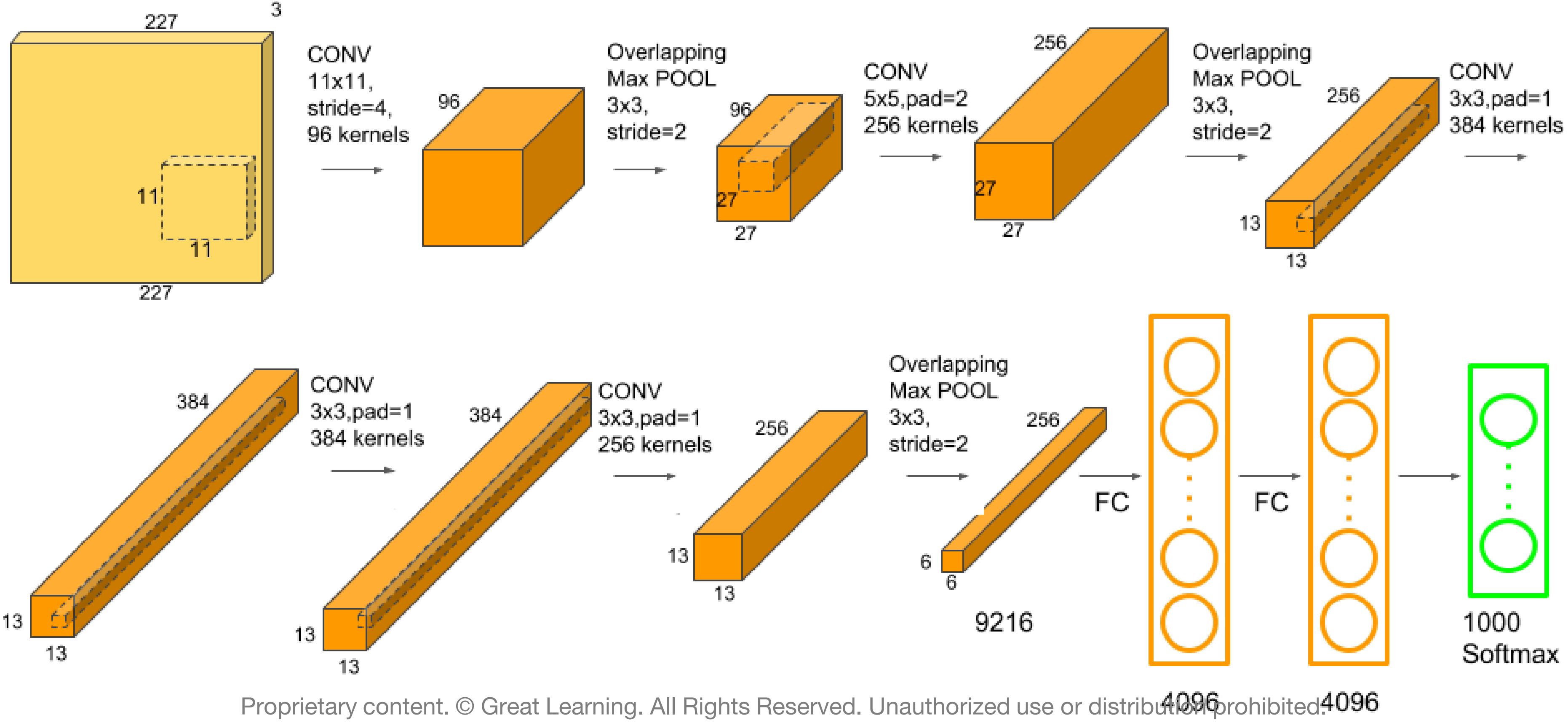
2. Since then, the Computer Vision field has completely changed for one!

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

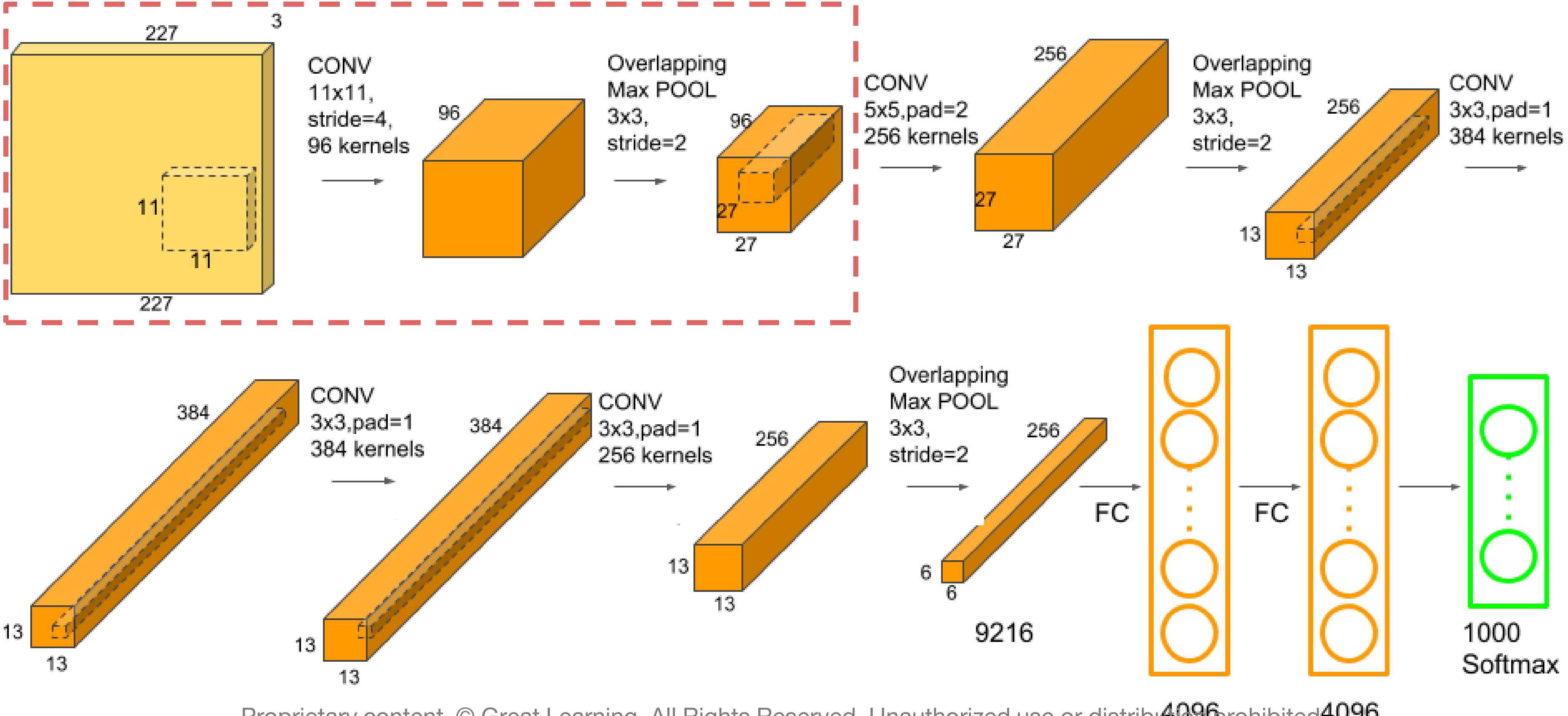
What made AlexNet successful?

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)

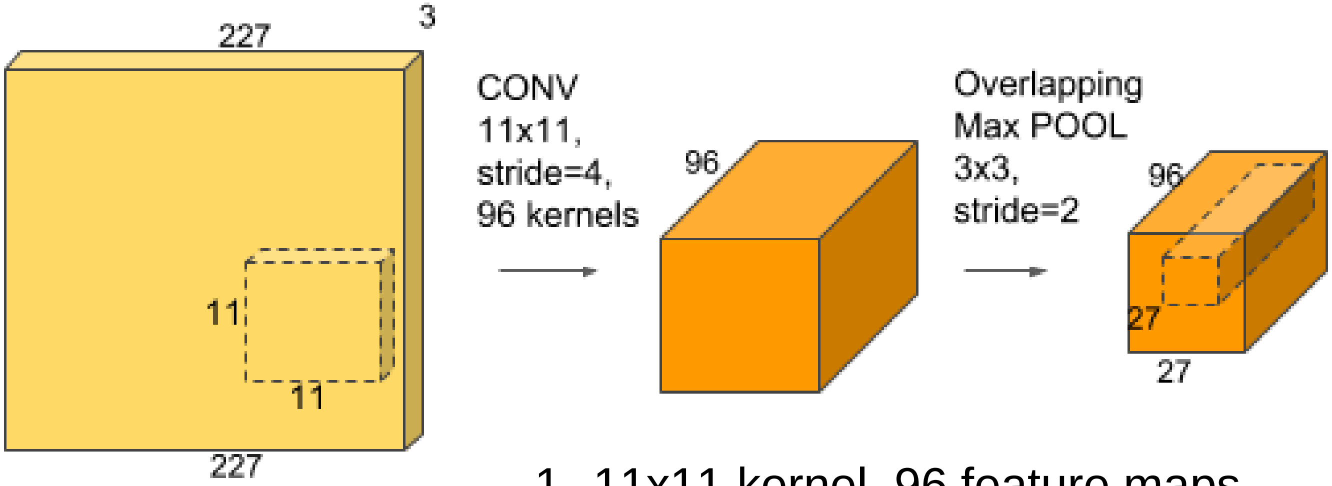


Lets Step in..



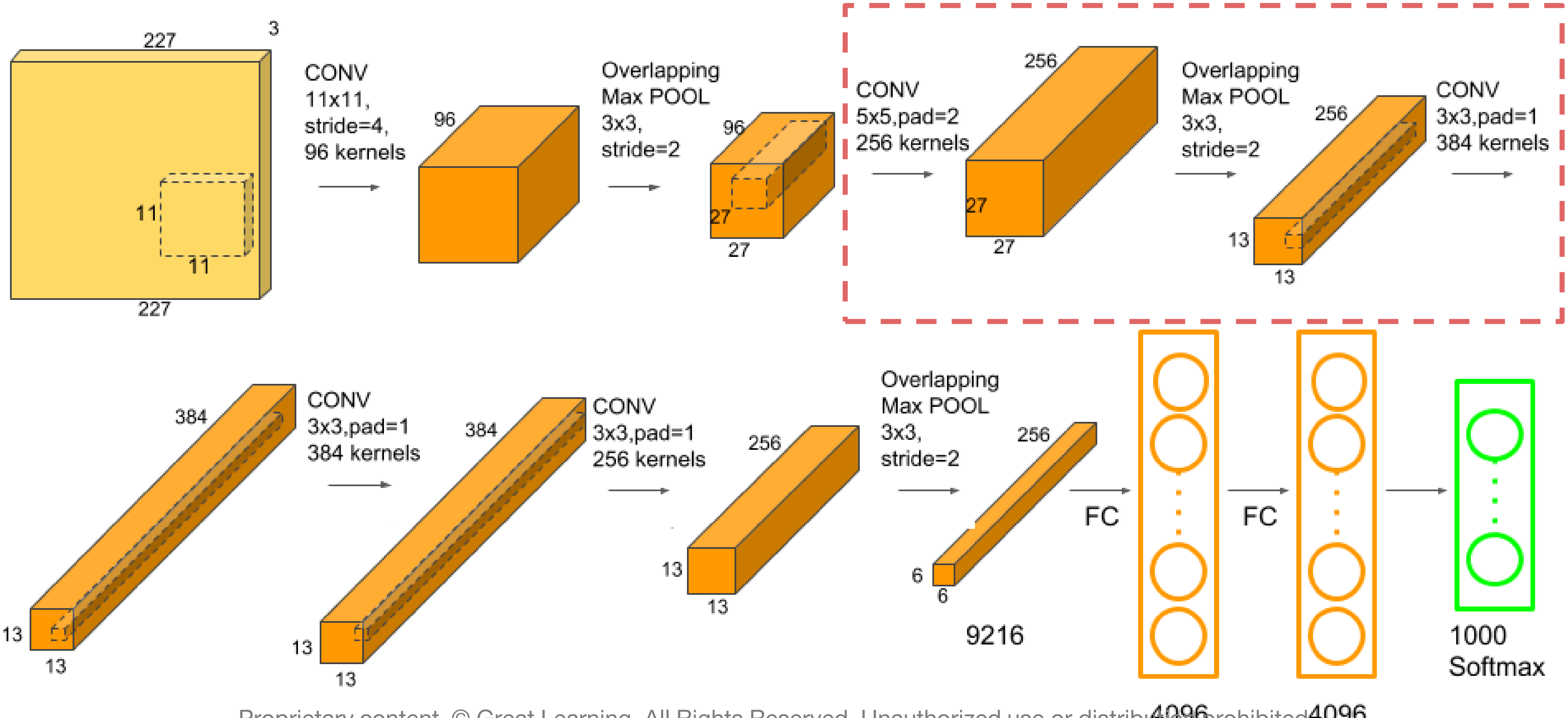
227
3
11
11
227
CONV
11x11,
stride=4,
96 kernels
96
Overlapping
Max POOL
3x3,
stride=2
96
27
27
CONV
5x5,pad=2
256 kernels
256
27
27
Overlapping
Max POOL
3x3,
stride=2
256
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
384 kernels
384
13
13
CONV
3x3,pad=1
256 kernels
256
13
13
Overlapping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

Lets go block by block - First Block

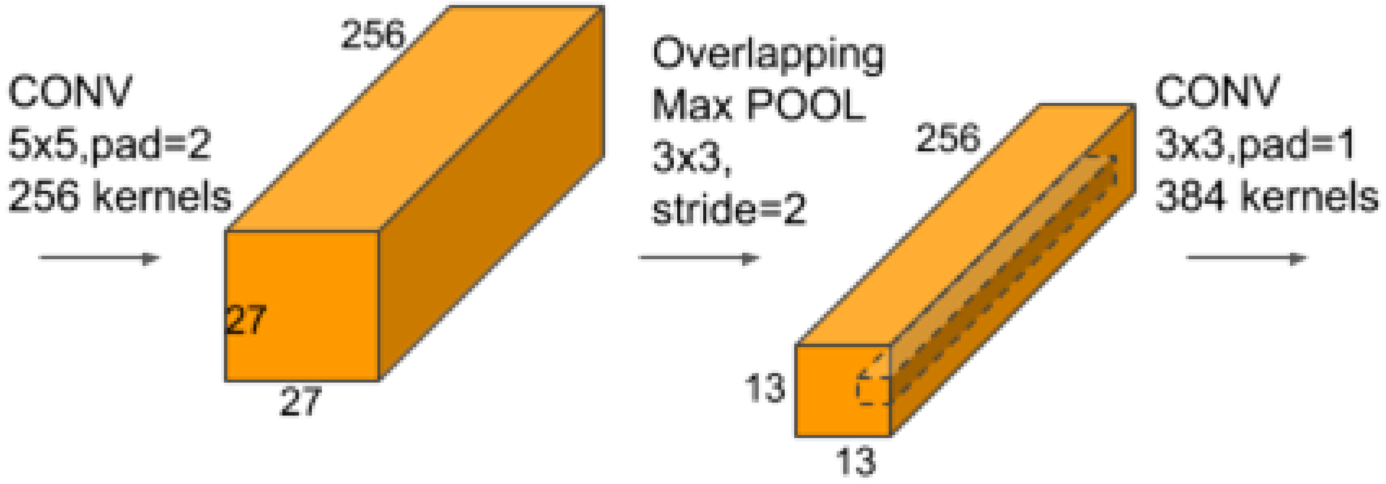


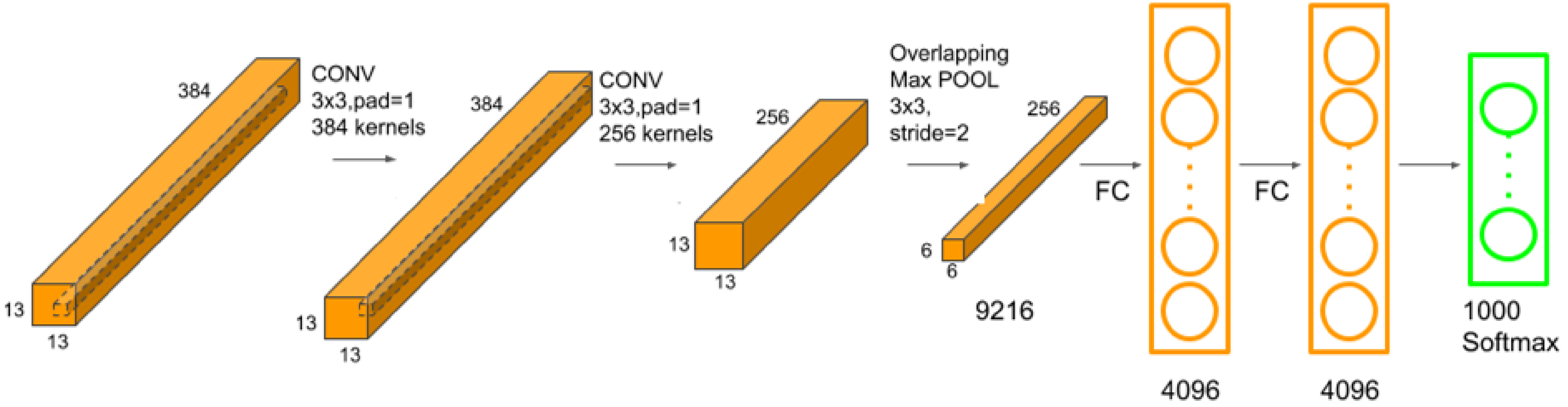
1. 11x11 kernel, 96 feature maps
2. Large Stride=4
3. Formula for output size - **(W+2P-F)/S+1**
4. Formula for no of parameters (ignore bias)- **MxNxkxk**
5. Maxpool, 3x3, s=2

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
FC
4096
4096
1000
Softmax

Second block





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

Overlapping Max Pooling

(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

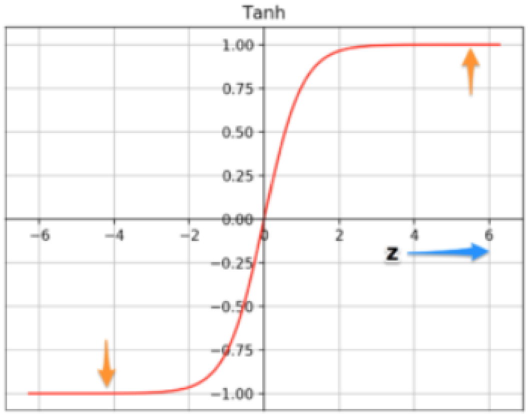
→

7	7
9	7

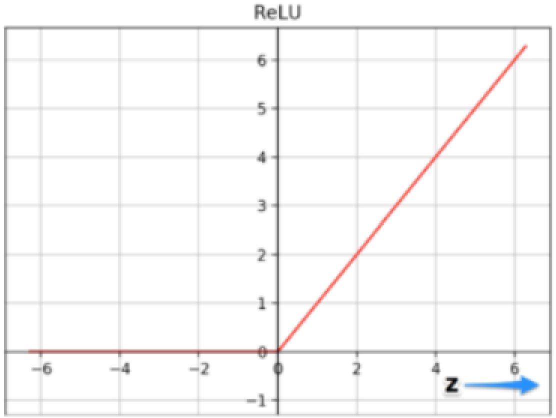
Moderate performance gain reported by authors

ReLU

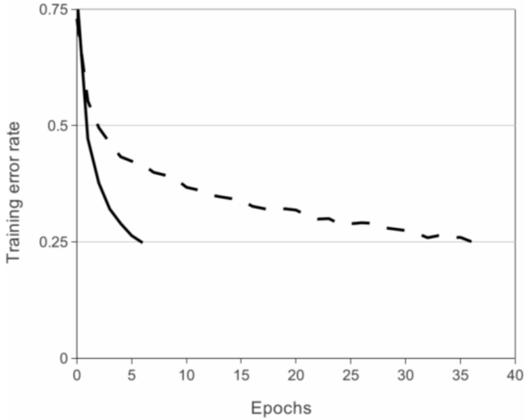
ReLU instead of tanh



tanh

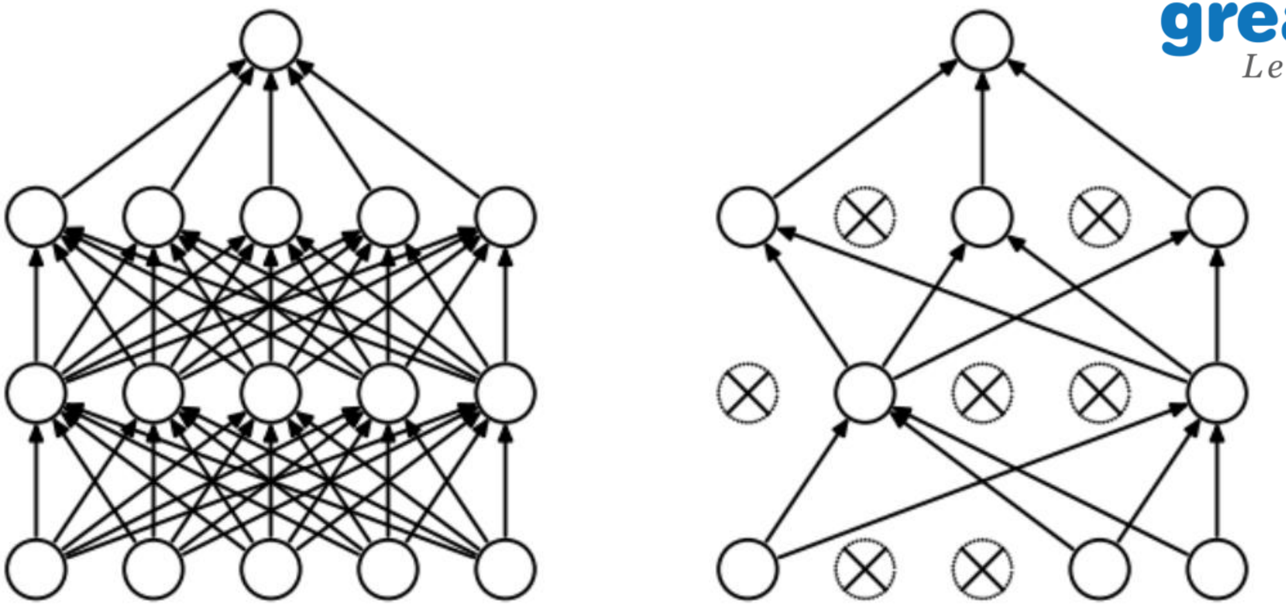


ReLU



Faster convergence

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



(a) Standard Neural Net (b) After applying dropout.

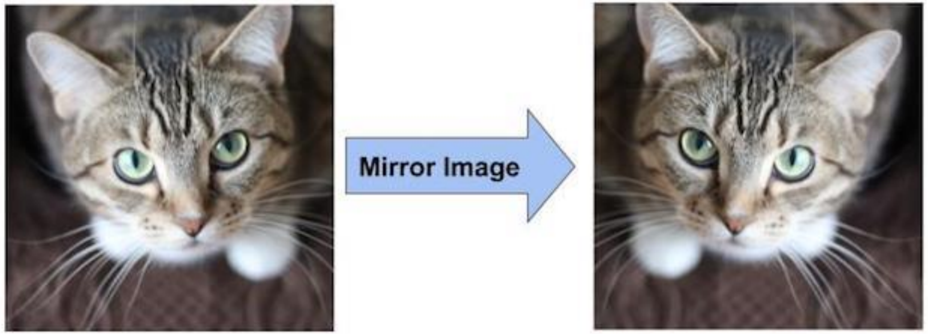
Regularization key due to huge #parameters in FC layer

Input Images

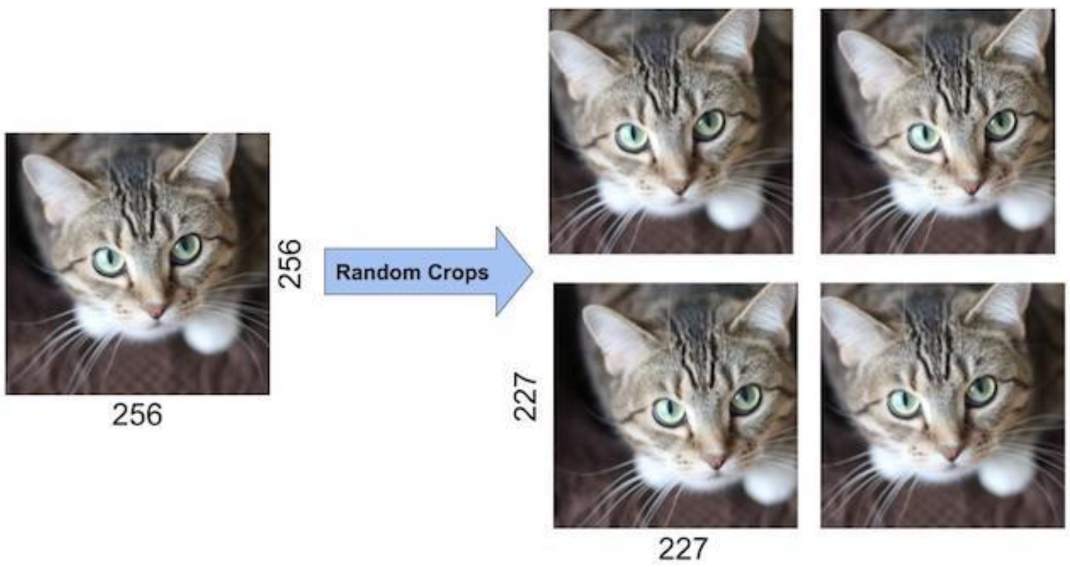
Resize smaller side to 256 and crop larger side to get 256x256

Get close to object of interest

Data augmentation

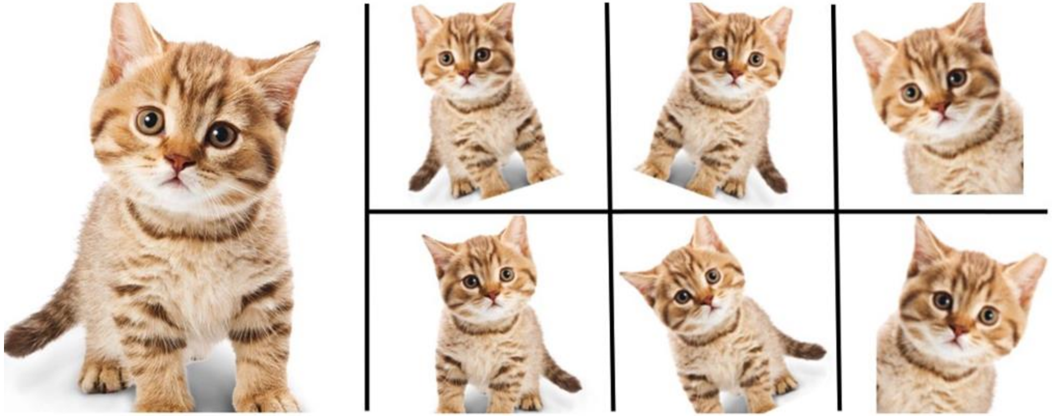


Horizontal flips



Random crop of 227x227 from 256x256 images

Inference Augmentation



Predict for each image using learnt model

Average prediction

Is seen to improve accuracy moderately in many applications

Test image

Augmented versions of test image

Summary

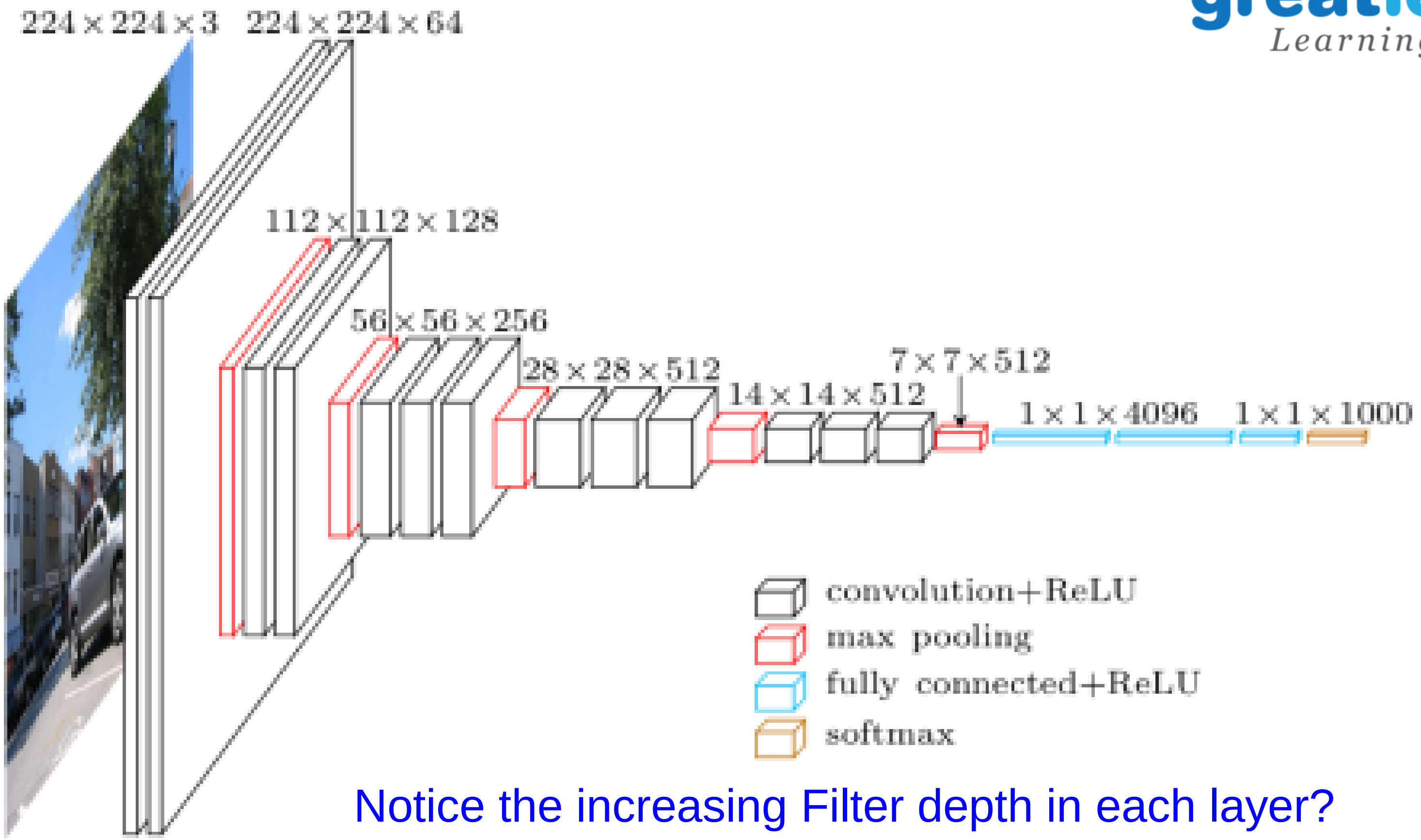
1. Trained the network on ImageNet data
2. Overlapping Maxpool
3. Used ReLU for the nonlinearity functions
4. Data augmentation: image translations, horizontal reflections, and patch extractions.
5. Inference/Test-time augmentation
6. Dropout in fully connected layers
7. Trained on two GTX 580 GPUs for five to six days

VGG (2014)

Another influential work is VGG which brought the ImageNet error down below **10% (7.3%** precisely**)**

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

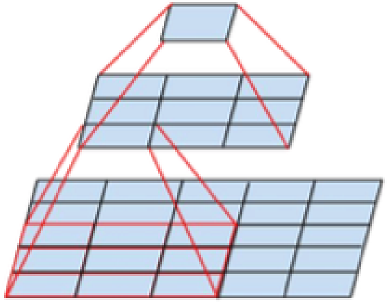
With pooling, information loss is a risk

Larger kernels mean more parameters/compute

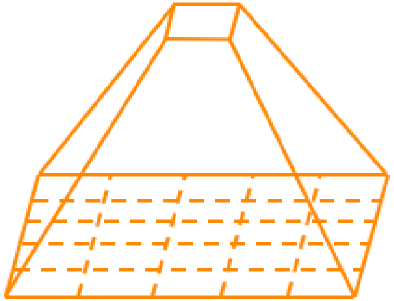
Remember: M- inputs of dimension D x D,, N - outputs, KxK kernels take MxNxKxK parameters and MxNxKxKxDxD operations

Use of 3x3 filters

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



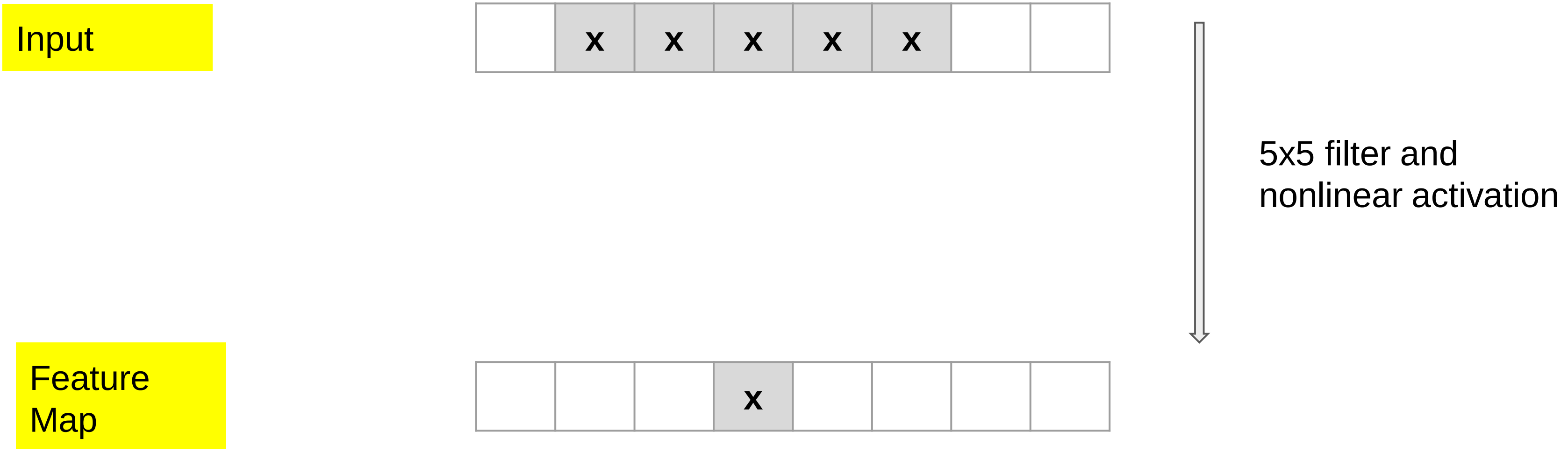
two successive
3x3 convolutions



5x5 convolution

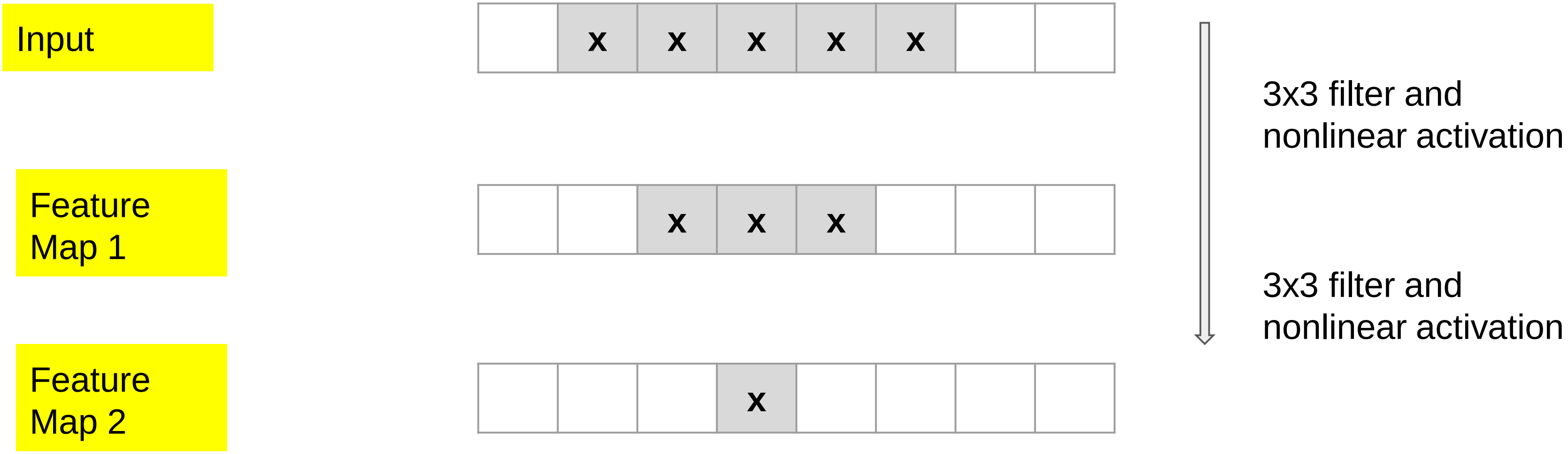
Receptive field

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

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

Case 2
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

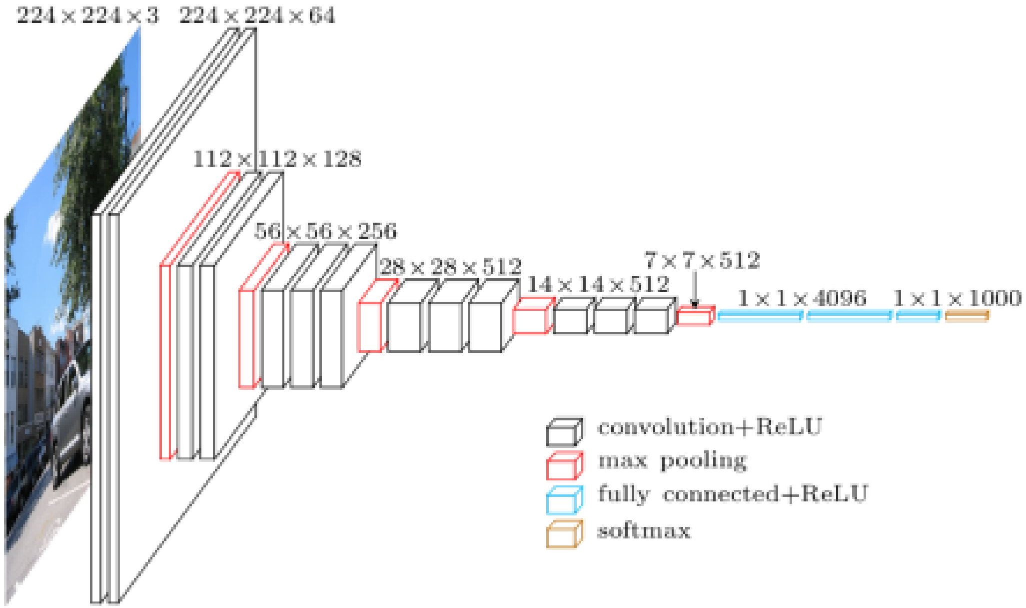
Remember both have same receptive field of 5 x 5 !

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 image)					
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 laye … ution, less depth

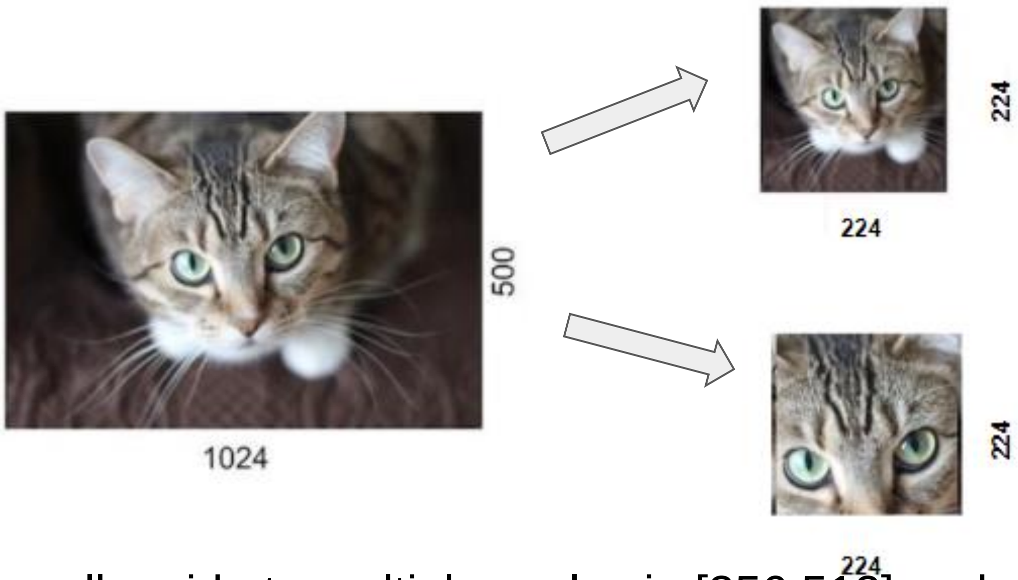
Upper layers encode high-level info, less spatial resolution, more depth

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



Scale to 256 and crop 224x224

Scale to 512 and crop 224x224

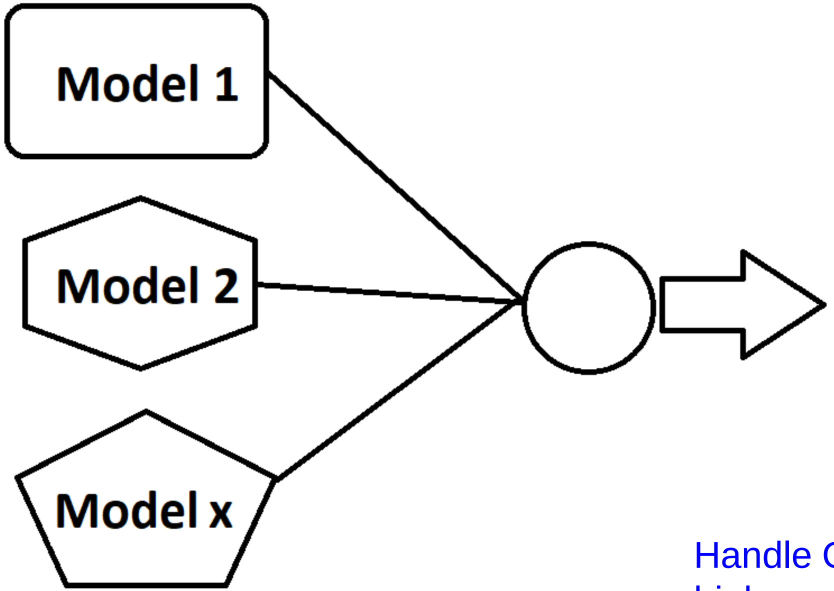
Resize smaller side to multiple scales in [256,512] and crop to get 224x224

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

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

Residual Networks

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

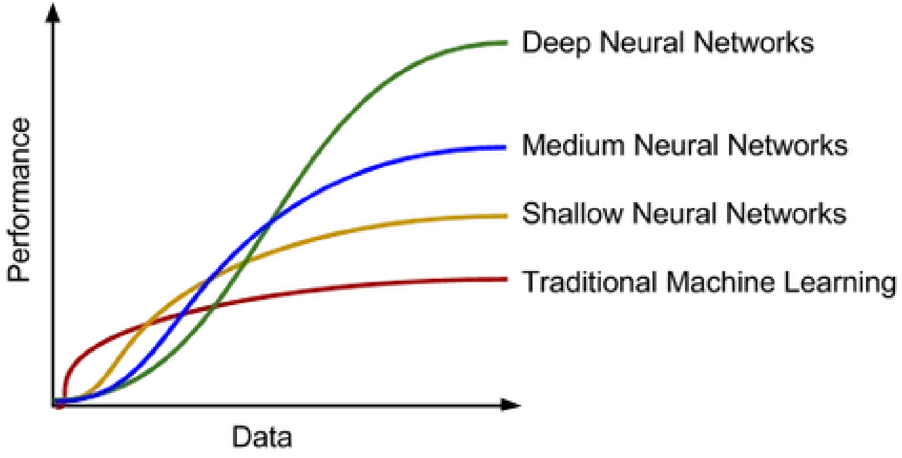
Imagenet error rate down to **3.57%** from **7.32 %** (VGG)

Very key idea of Residual connections

Deep Residual Networks

greatlearning
Learning for Life

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

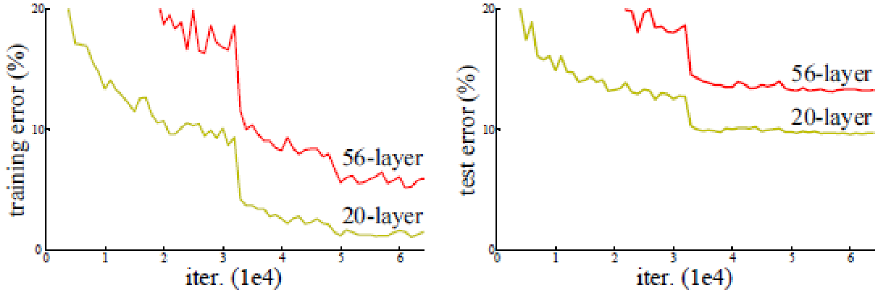


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Some Math Examples

$$P(x) = f(g(x))$$
$$P'(x) = f'(g(x))g'(x)$$
$$P''(x) = f''(g(x))(g'(x))^2 + f'(g(x))g''(x)$$

See the Vanishing Gradient problem?

Is the composition of 2 convex functions convex?

Some Math Examples

$$P(x) = f(g(x))$$

$$P'(x) = f'(g(x))g'(x)$$

$$P''(x) = f''(g(x))(g'(x))^2 + f'(g(x))g''(x)$$

Is the composition of 2 convex functions convex?

Not in general. e.g. e^{-x^2}

Convex i.e. the 2nd derivative is positive definitely if *f* is monotonic.

What if we use residuals ?

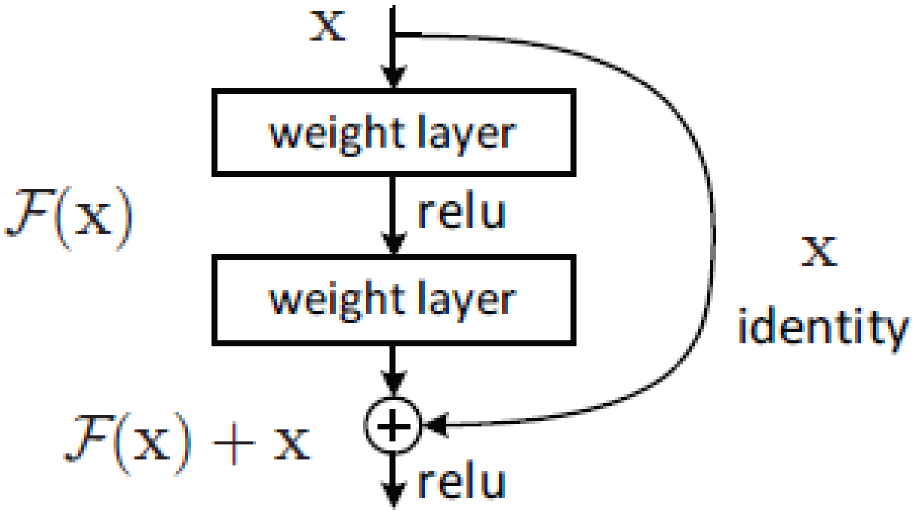
$$P(x) = f(x + g(x)) + x + g(x)$$

We define *f* and *g* as residuals

$$P(x) = (f'(x + g(x)) + 1)(g'(x) + 1)$$

The 1st derivative is better conditioned

Residual Block: Skip connection



In I

Figure 2. Residual learning: a building block.

Residual Block: Skip connection

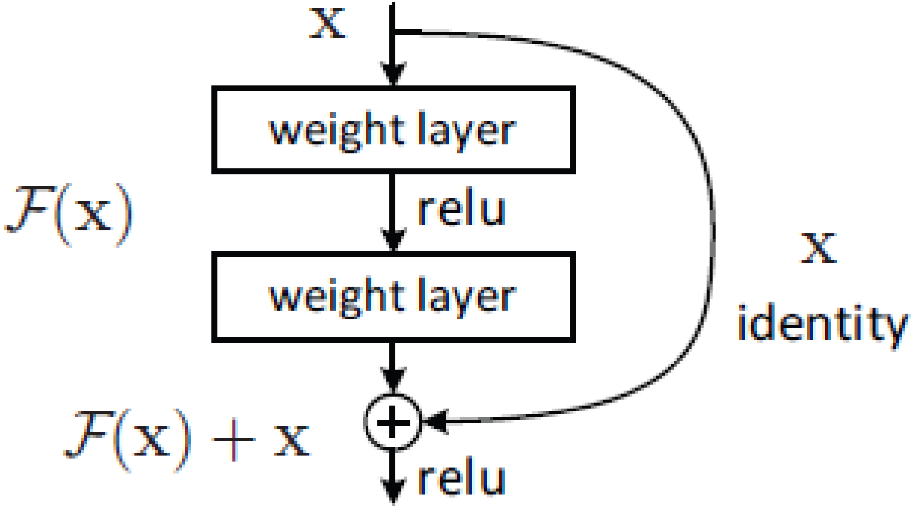


Figure 2. Residual learning: a building block.

As ell behaved

Residual Block: Skip connection

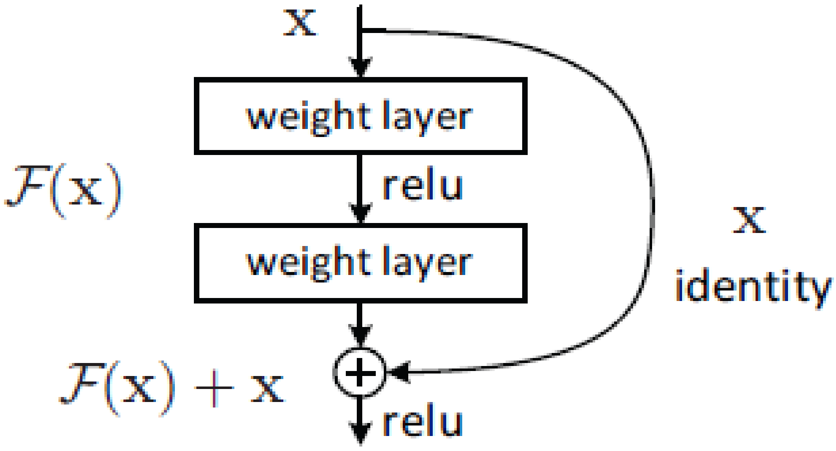
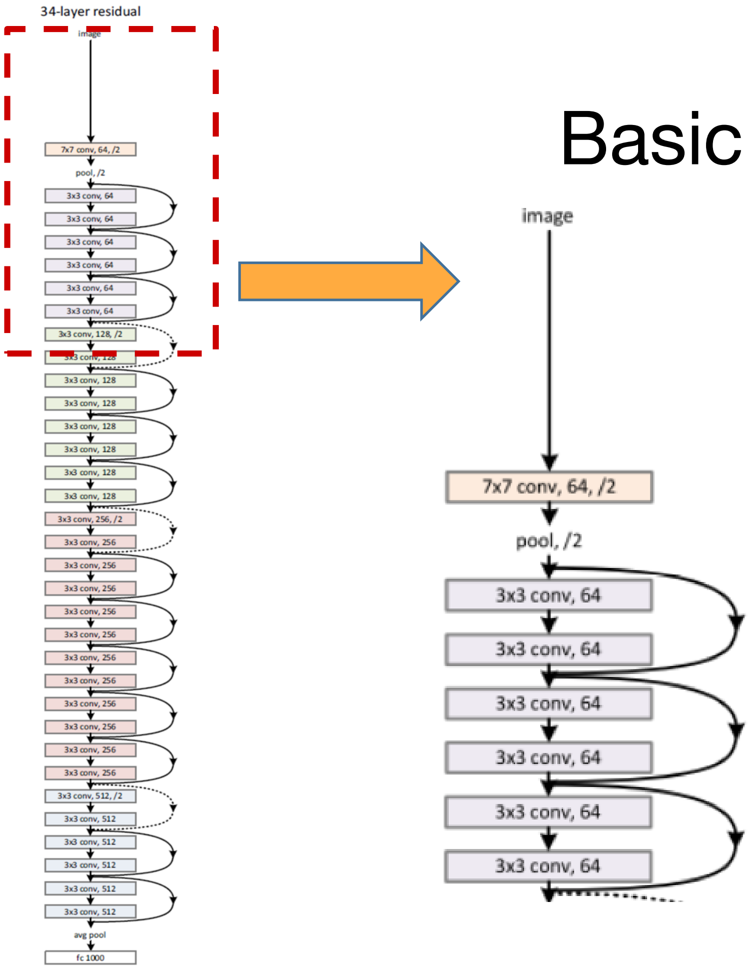


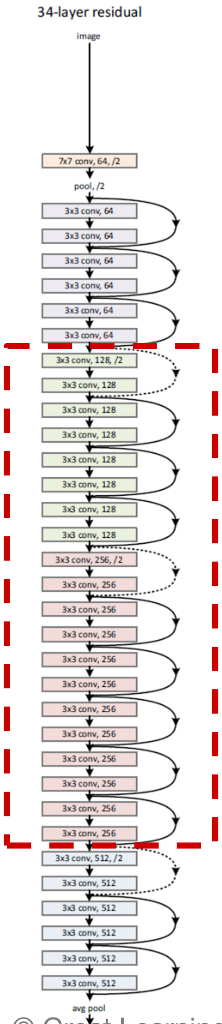
Figure 2. Residual learning: a building block.

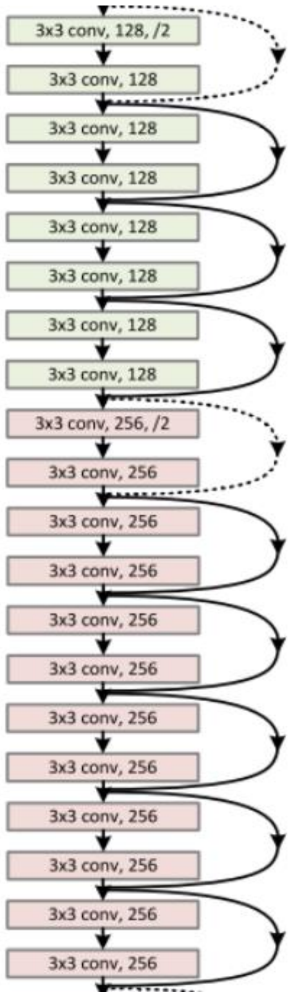
Typically the Residuals are small...

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

Basic Architechure- Resnet 34

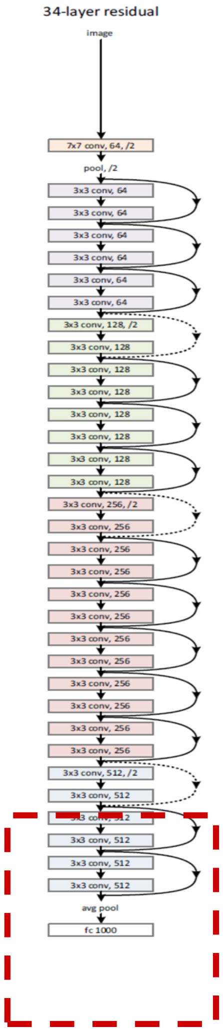




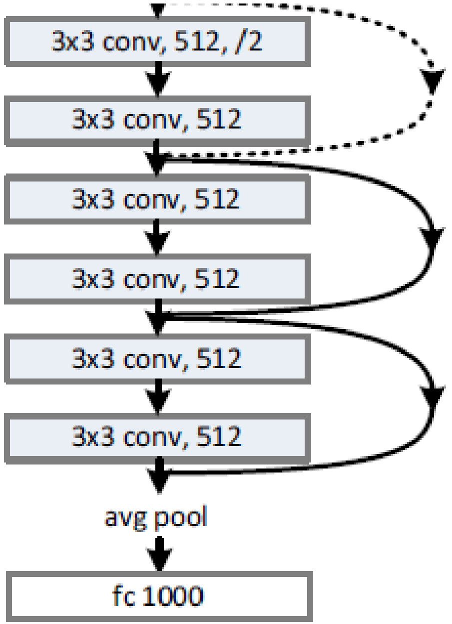


greatlearning
Learning for Life

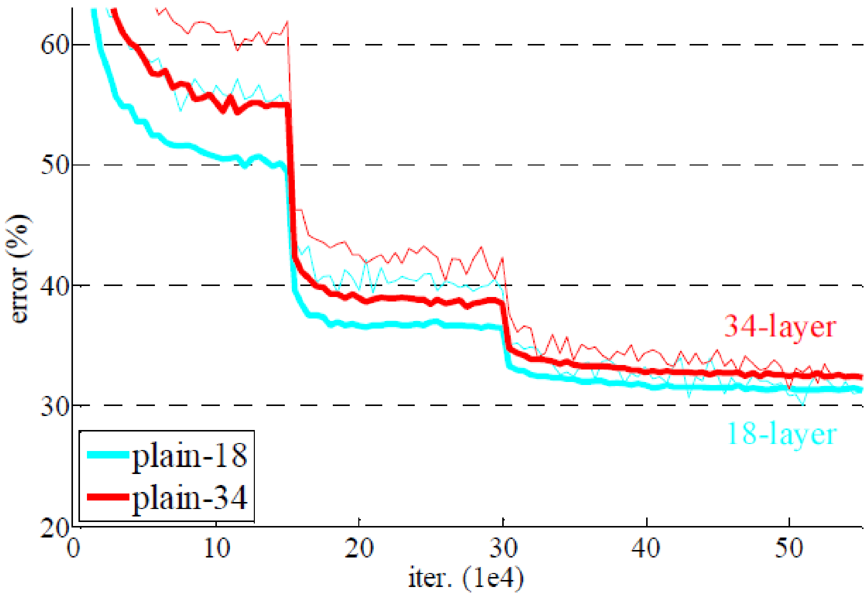
Basic Architechure- Resnet 34

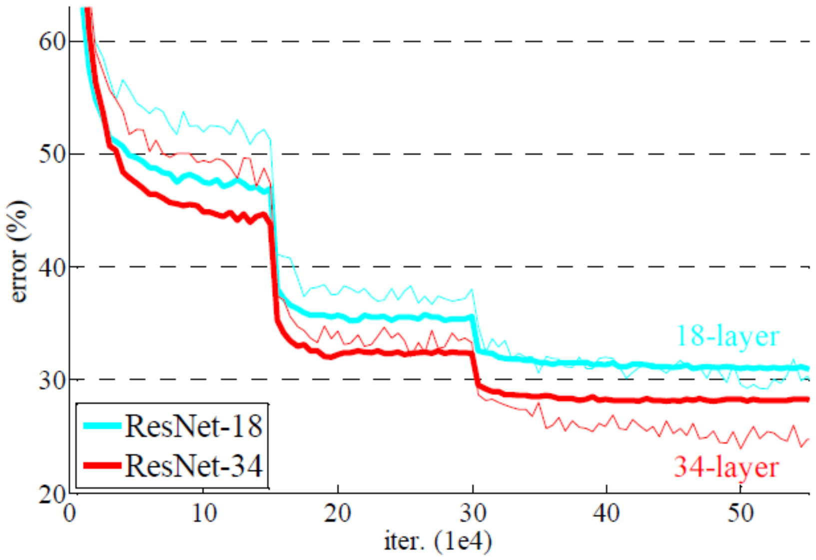


Basic Architechure- Resnet 34



Improved Training and Test Accuracy





Summary

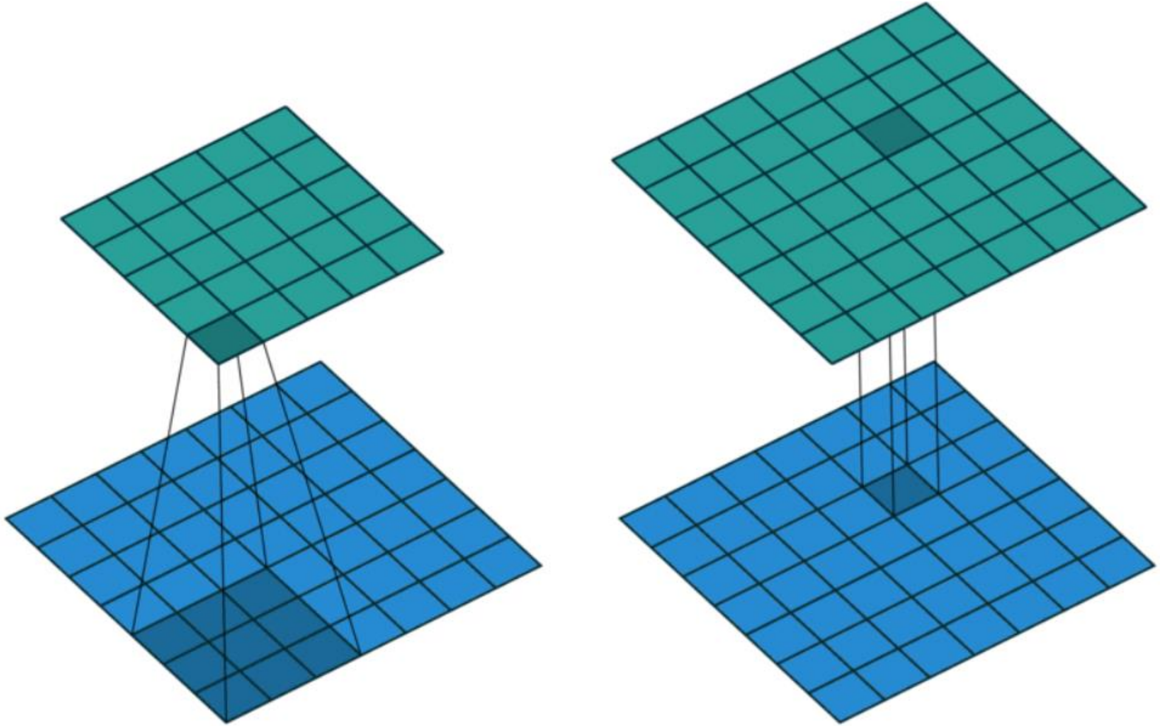
- Skip connections for training very deep networks
- Scale/horizontal flip data augmentation
- Batch normalization
- Dropout not used
- Fully convolutional output
- Multi-crop/multi-scale prediction and averaging testing
- Imagenet error rate down to **3.57%** from **7.32 %** (VGG)

GoogLeNet/Inception-v1

Topics we will look at:

1. The 1×1 Convolution
2. Inception Module
3. Global Average Pooling
4. Overall Architecture
5. Auxiliary Classifiers for Training
6. Testing Details

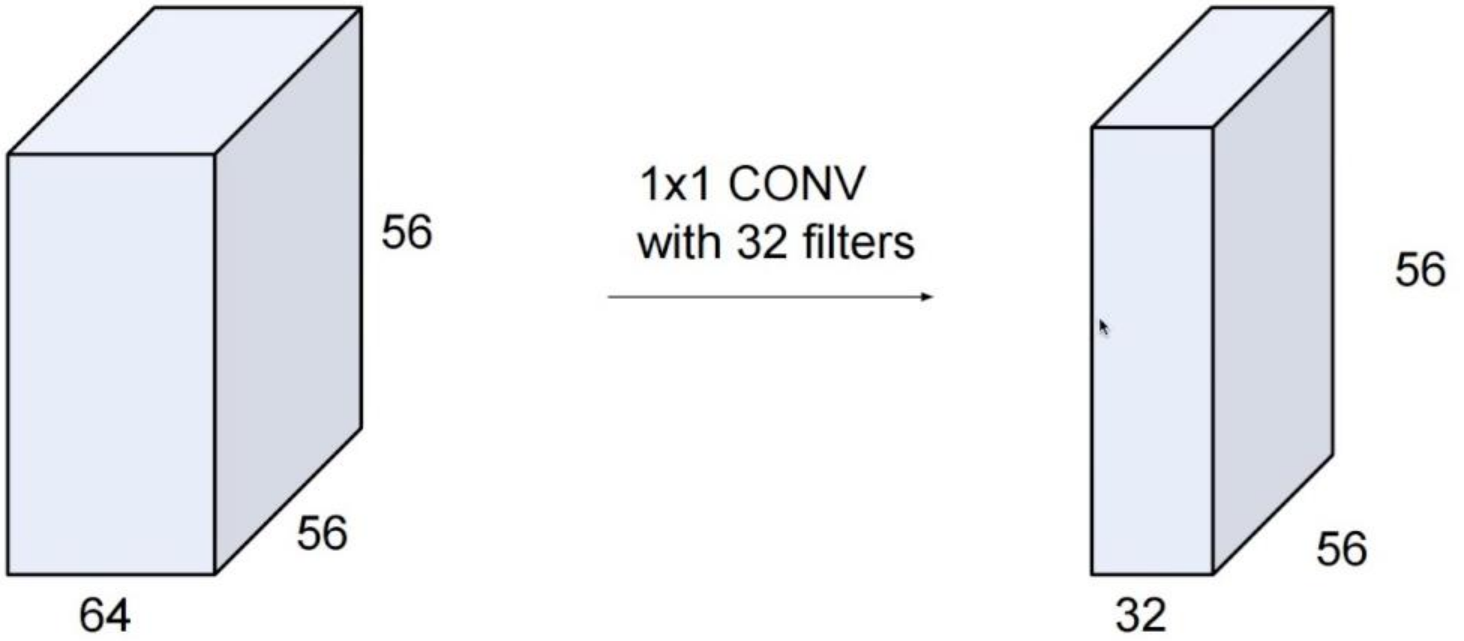
1x1 Convolution filters



1x1 filters

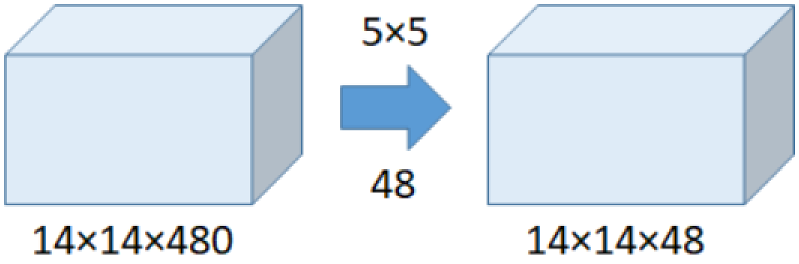
Eacl ture maps followed
by n

greatlearning
Learning for Life



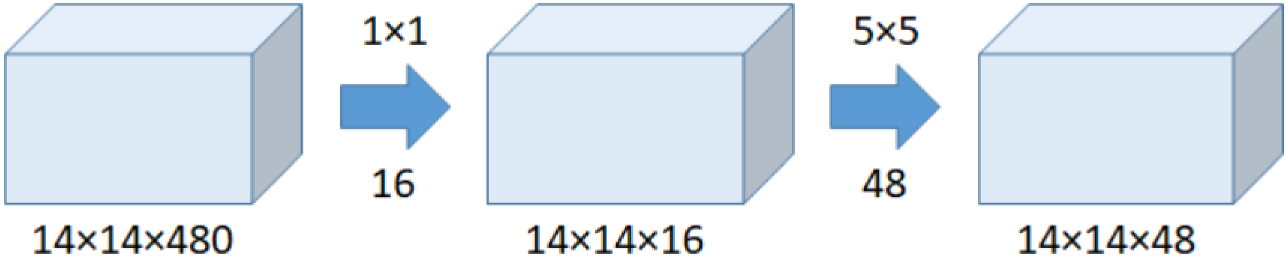
Easy way to get Feature reduction/increase, additional non-linearity

Reduction of parameters



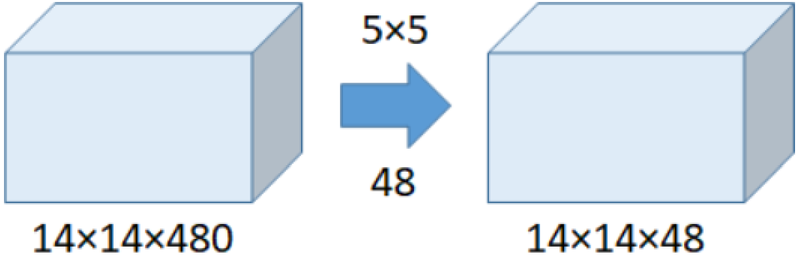
#Parameters - 48 x 480 x 5 x 5 = **0.5 M**
#OPs - 14x14x480x5x5x48 = **113M**

Without the Use of 1×1 Convolution



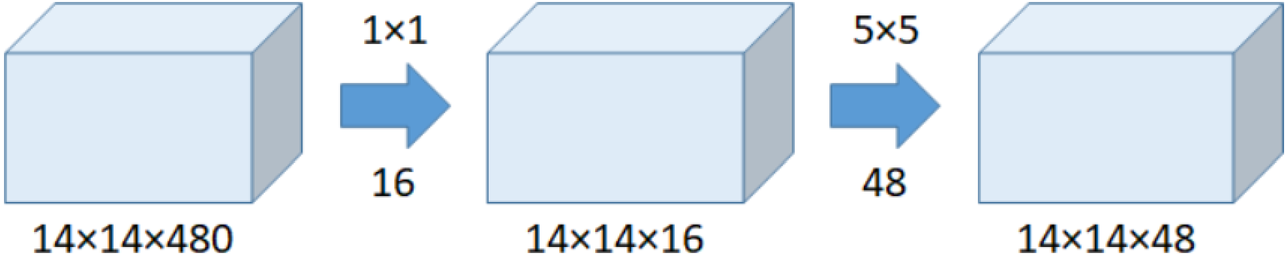
With the Use of 1×1 Convolution

#OPs for the below network - ?



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

Without the Use of 1×1 Convolution

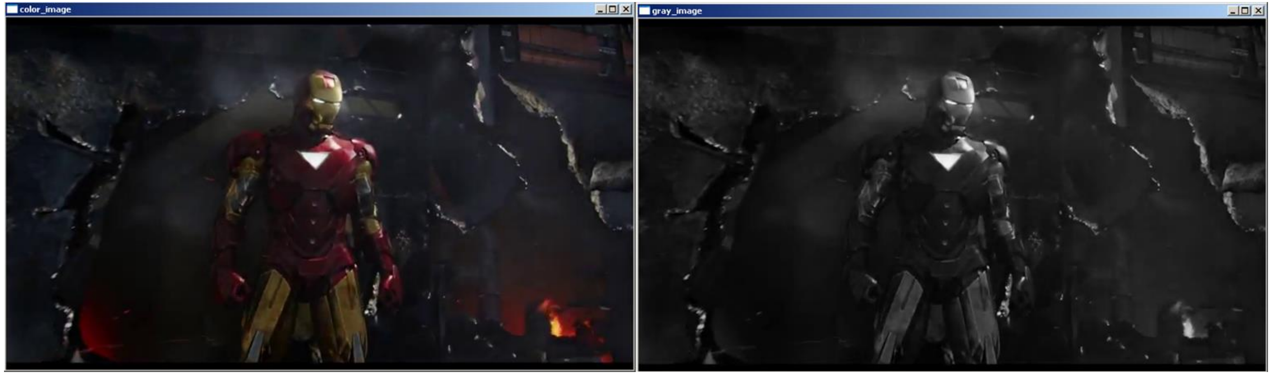


With the Use of 1×1 Convolution

#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

Objects could be bigg ... ırger sized kernel

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

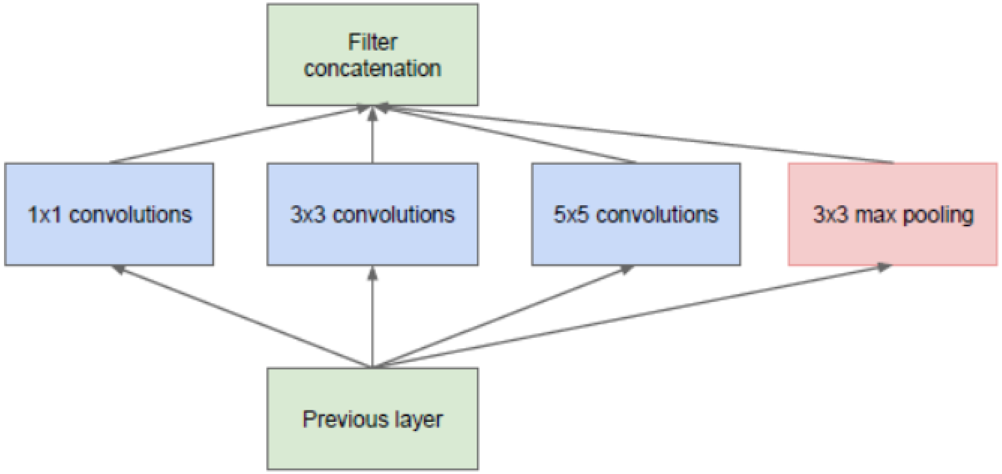
Do we need Pooling or not?

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 !

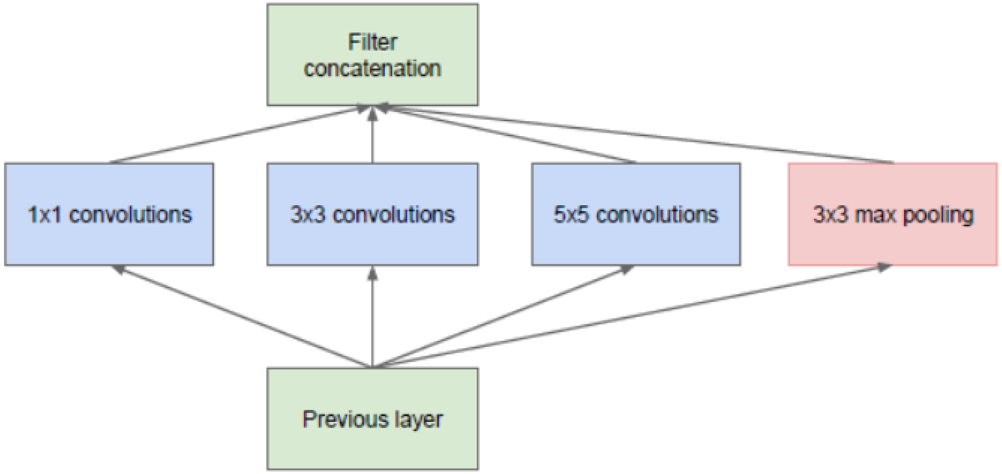
Inception Block



(a) Inception module, naïve version

Offe size of kernels
3x3 or 5x5 or combinations c) Whether or not to do Pooling

Too Many parameters and Expensive

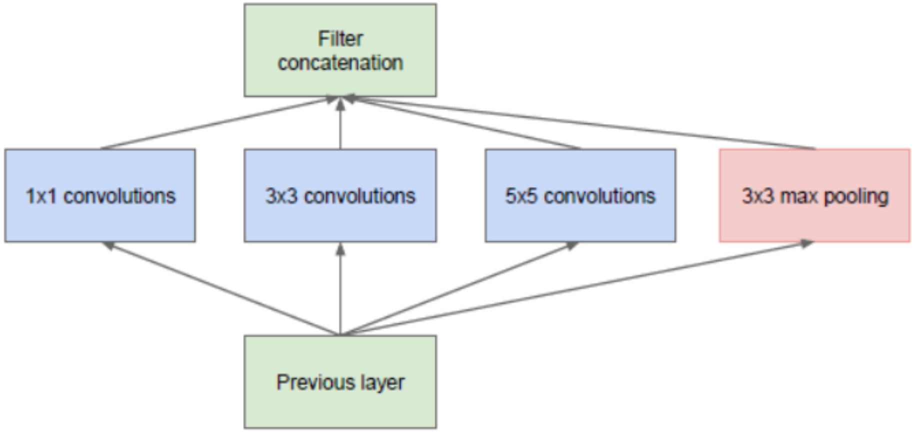


(a) Inception module, naïve version

Say input dim = 256x256x4. Can you compute the #parameters and #OPs for Output = 4 features for each path

e.g. 1x1 conv - #Param: 4x4 #OPs: 256x256x4x4

Too Many parameters and Expensive



(a) Inception module, naïve version

Say inpu   rs and #OPs for
Output =

1x1 conv - #Param: 32x32 #OPs: 256x256x32x32

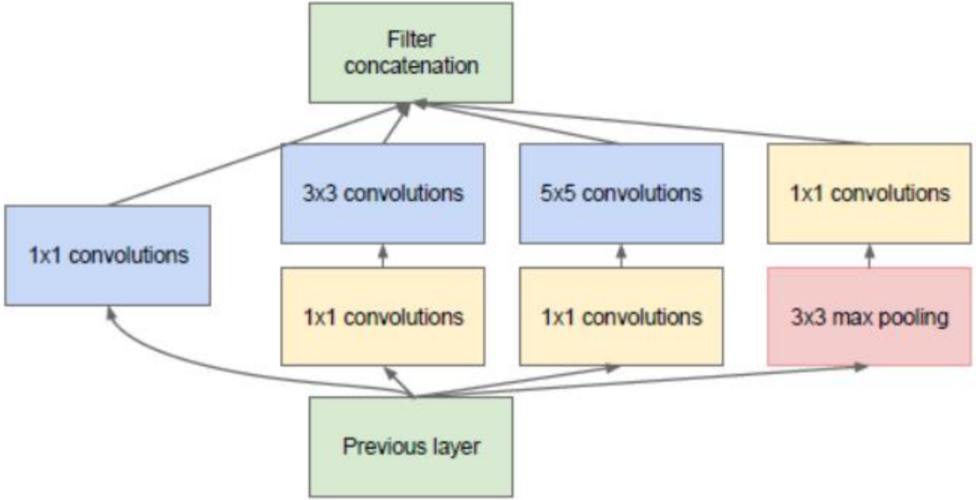
So What to do?

3x3 conv - #Param: 32x32x3x3 #OPs: 256x256x32x32x3x3

5x5 conv - #Param: 32x32x5x5 #OPs: 256x256x32x32x5x5

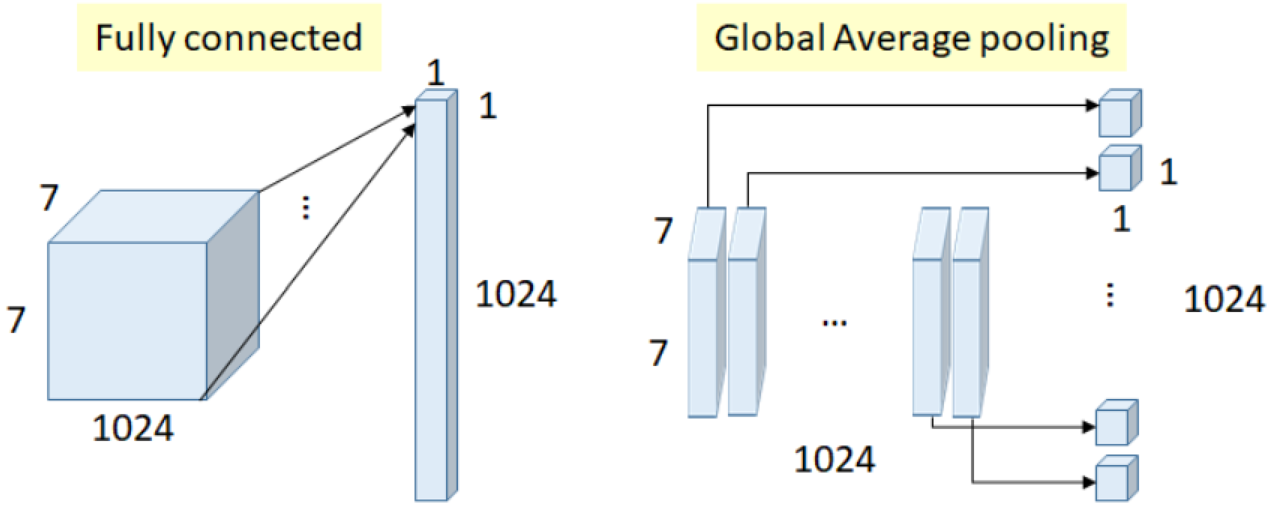
3x3 pool - #Param: 0 #OPs: 128x128x32x3x3

Use 1x1 conv to reduce parameters and speed



(b) Inception module with dimensionality reduction

Global Average Pooling



Fully Connected Layer VS Global Average Pooling

FC parameters: 7x7x1024x1024 = **50M**,

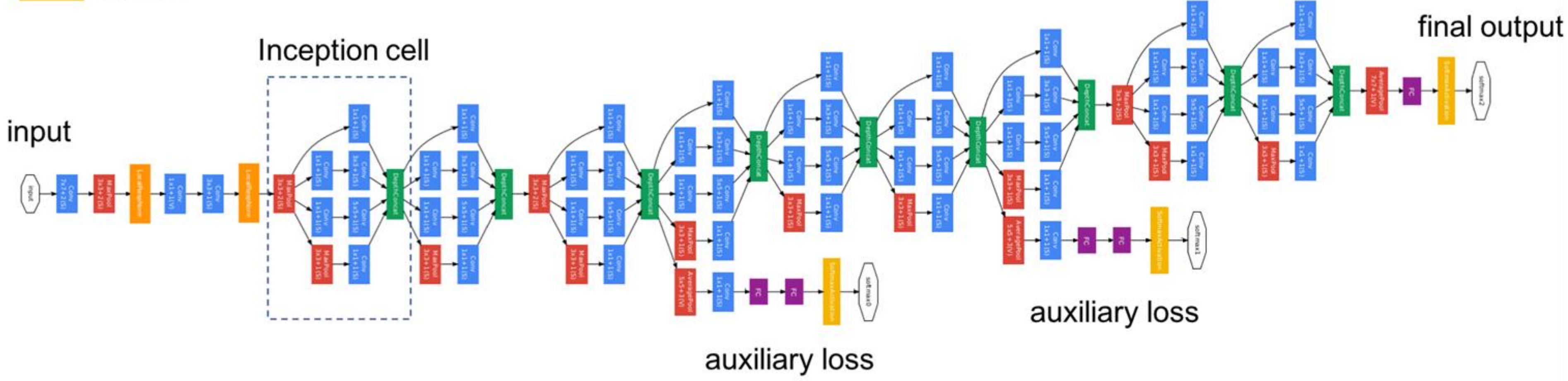
GAP parameters: **0**

Much less parameters using GAP, less overfitting!

Overall Architecture

greatlearning
Learning for Life

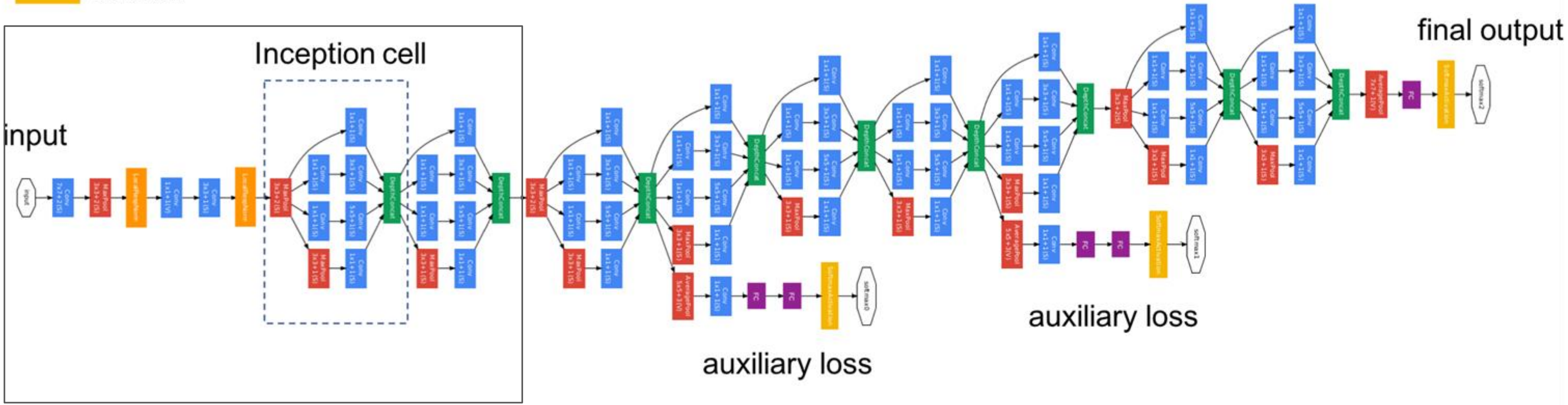
- convolution
- max pooling
- channel concatenation
- channel-wise normalization
- fully-connected layer
- softmax

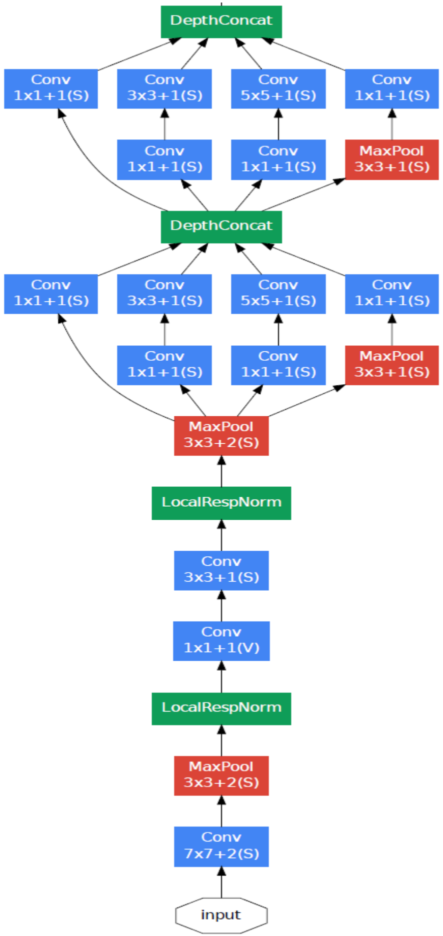


Overall Architecture

greatlearning
Learning for Life

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

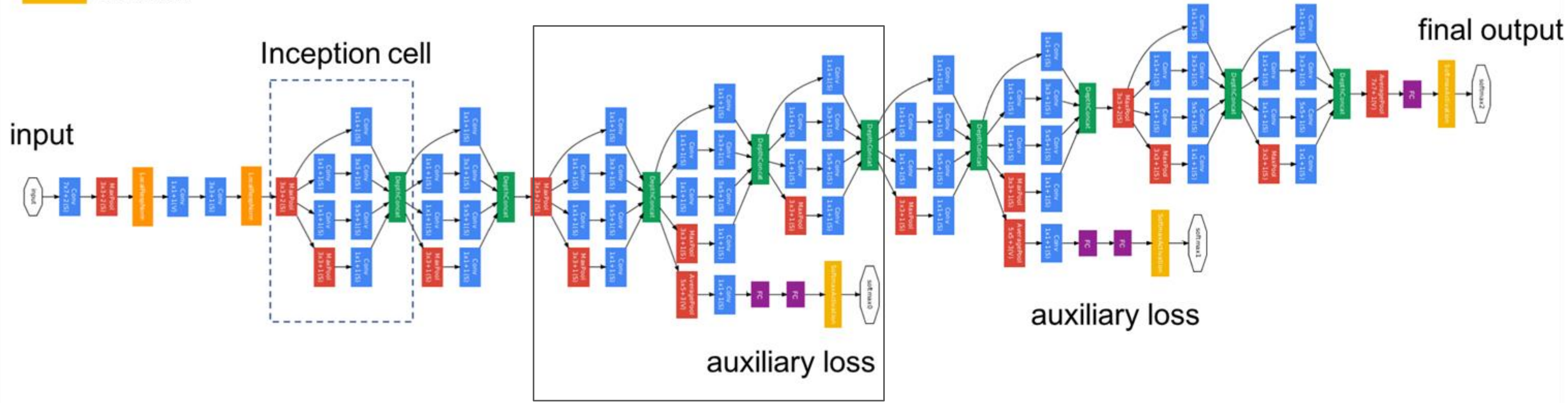


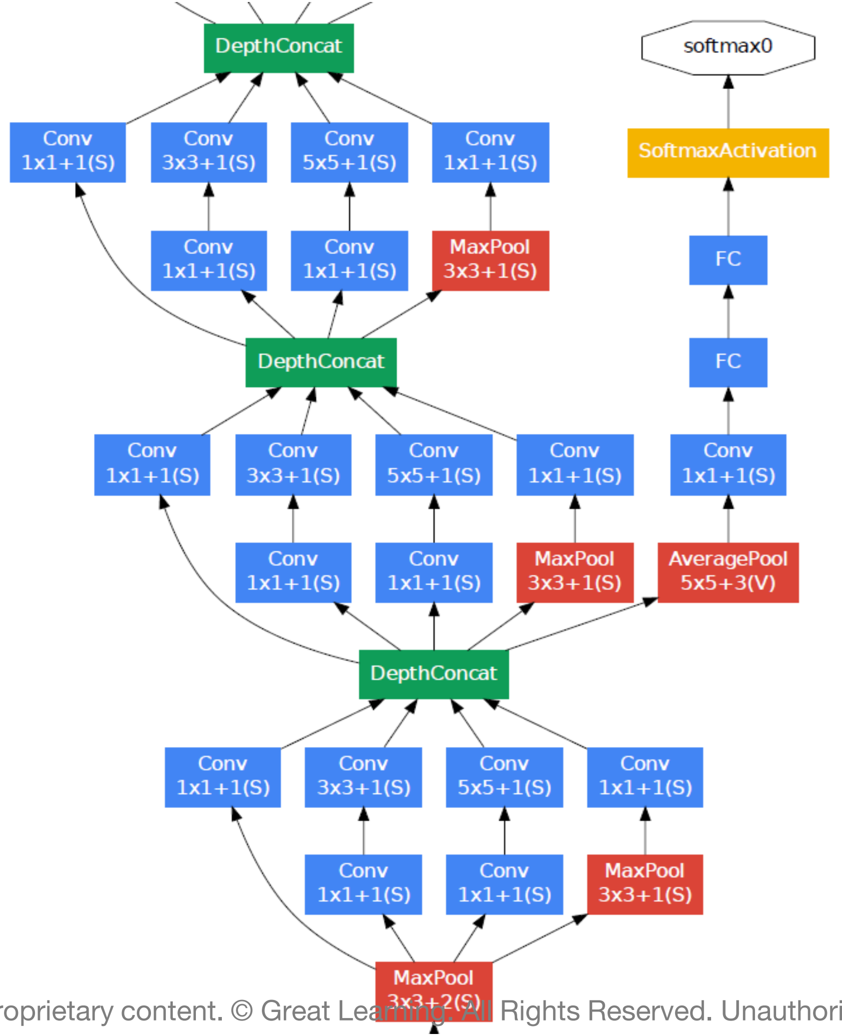

DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 3x3+1(S)
Conv 1x1+1(V)
LocalRespNorm
MaxPool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

Overall Architecture

greatlearning
Learning for Life

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

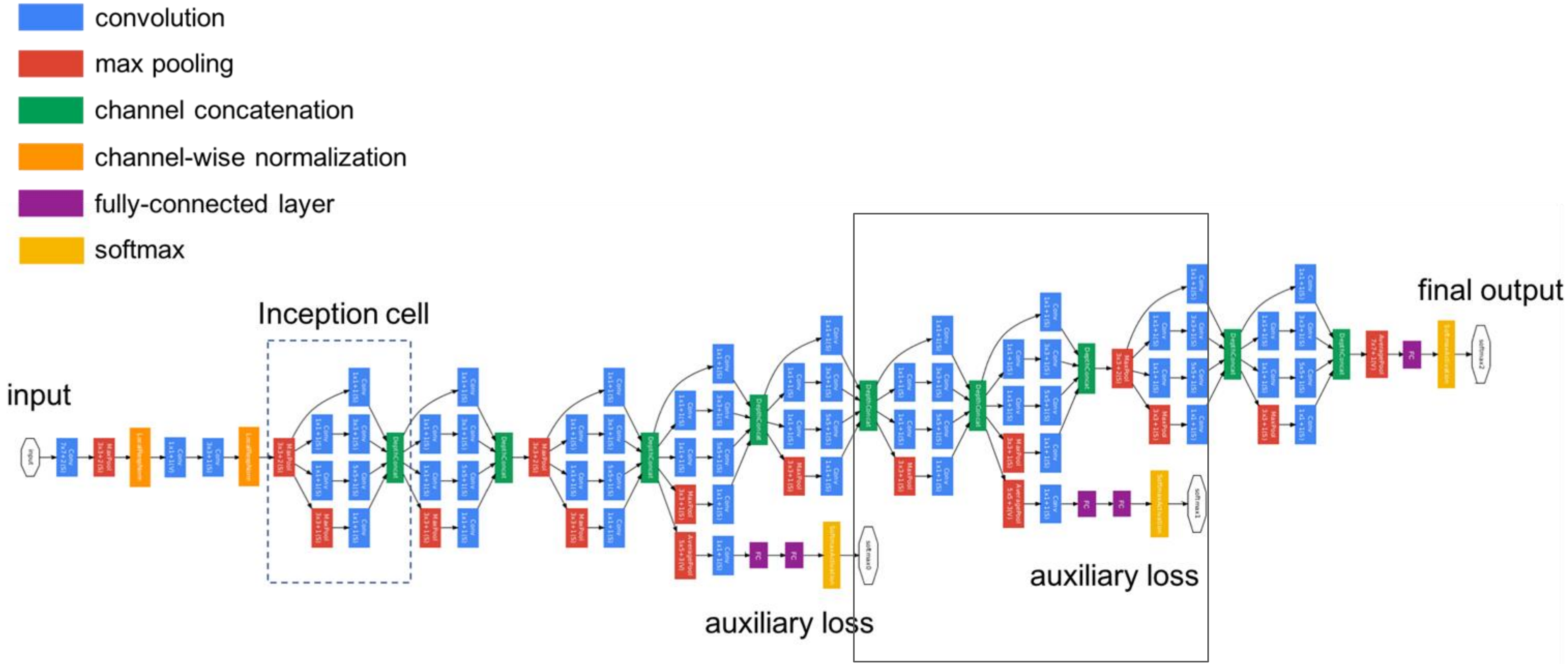




greatlearning
Learning for Life

Use this to
avoid VG effect
in middle layers

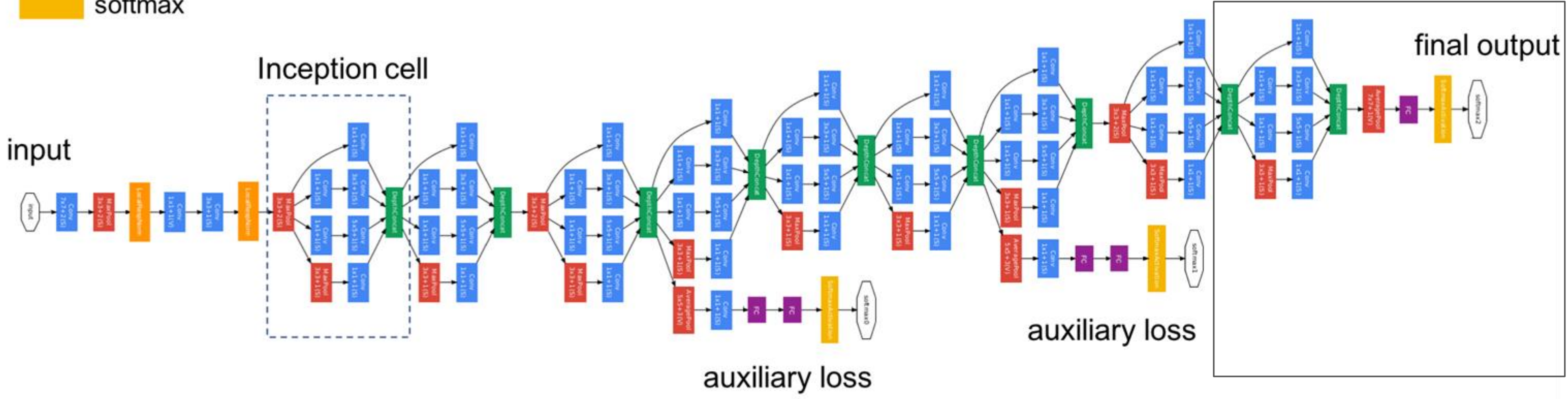
Overall Architecture

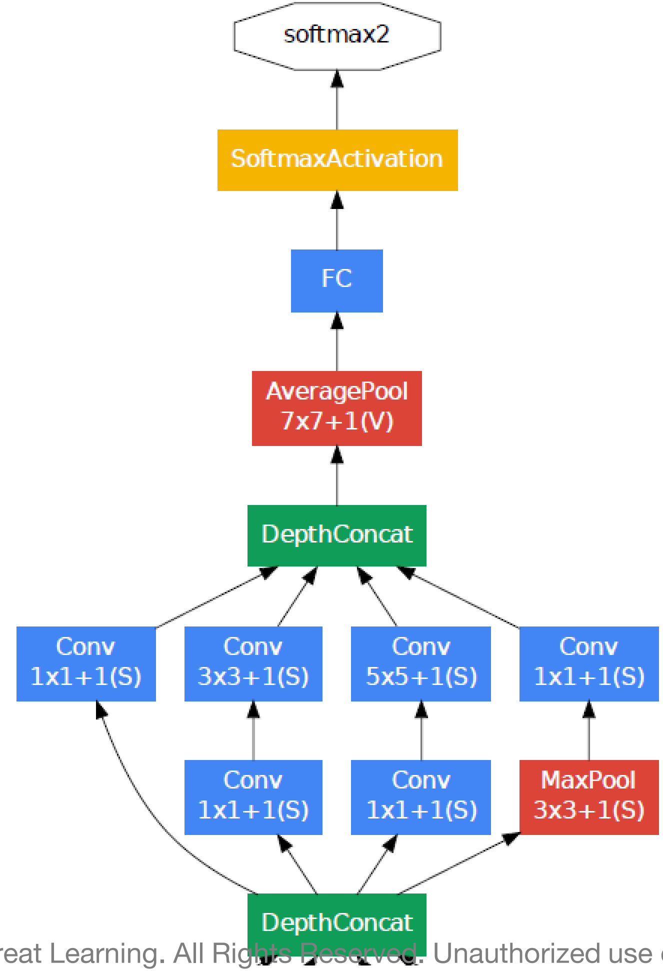


greatlearning
Learning for Life

Overall Architecture

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



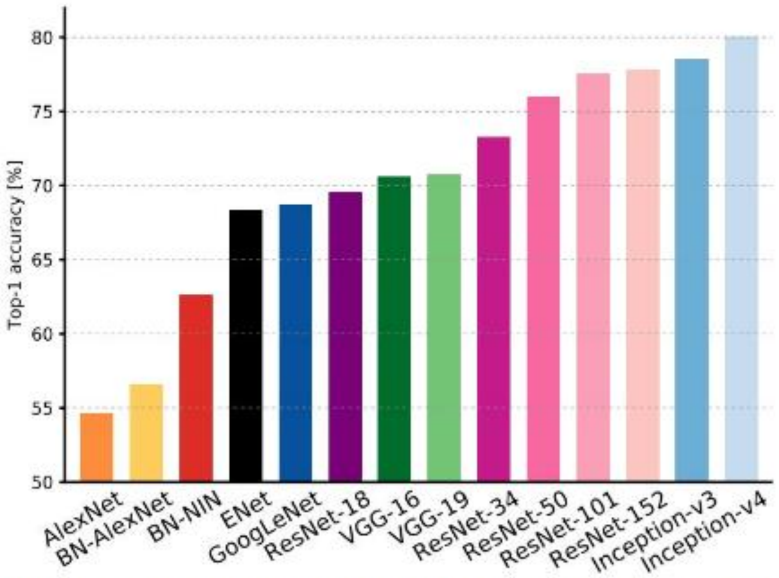

softmax2
SoftmaxActivation
FC
AveragePool
7x7+1(V)
DepthConcat
Conv
1x1+1(S)
Conv
3x3+1(S)
Conv
5x5+1(S)
Conv
1x1+1(S)
Conv
1x1+1(S)
Conv
1x1+1(S)
MaxPool
3x3+1(S)
DepthConcat

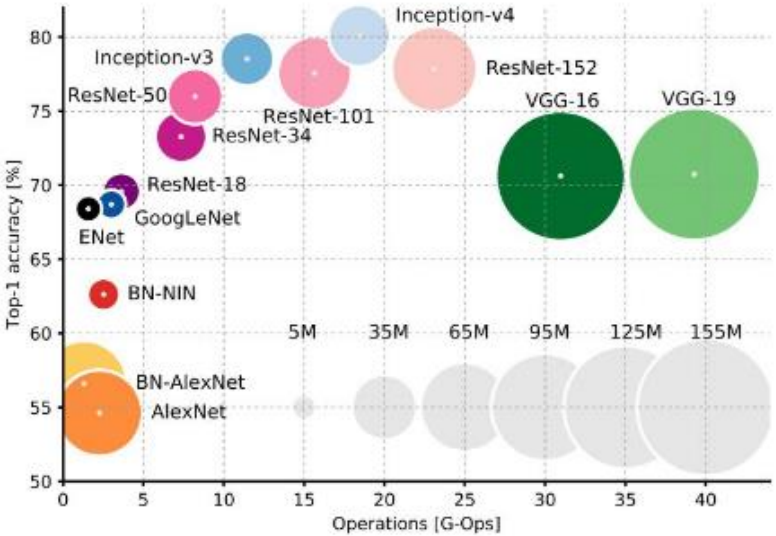
Summary

- 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)

State of Art CNN architectures

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





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