AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Introduction to Deep Learning

CNN Architectures

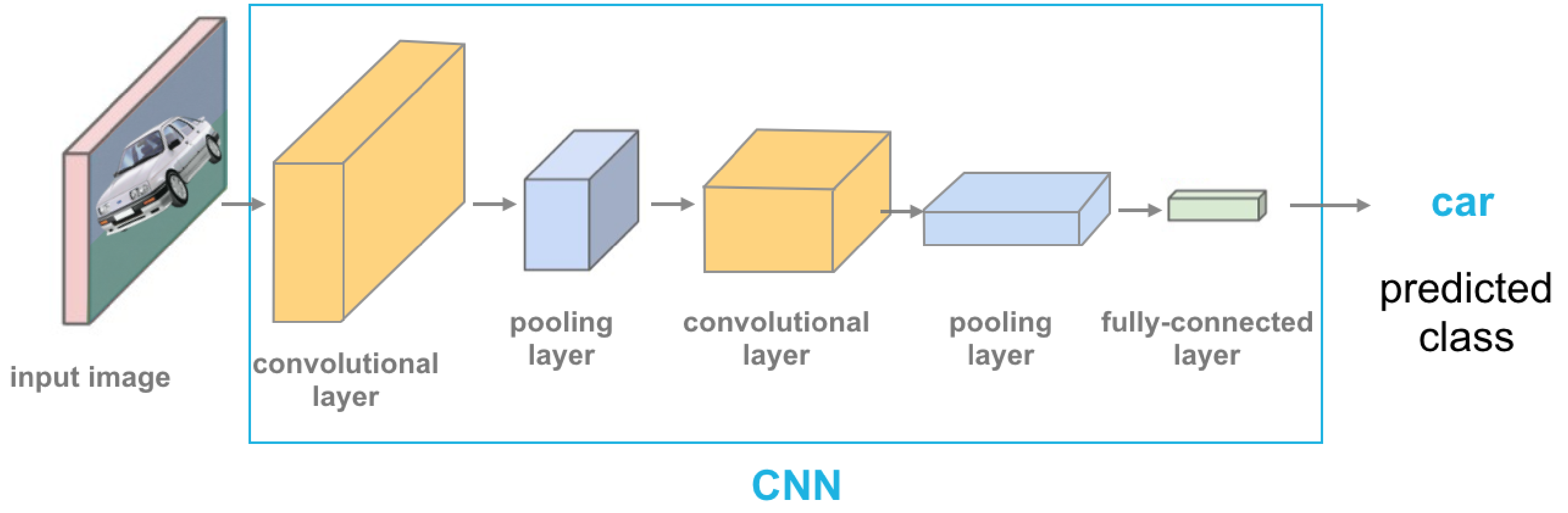
- LeNet
- AlexNet
- ZFNet
- VGG
- GoogLeNet
- ResNet

Convolutional Neural Network

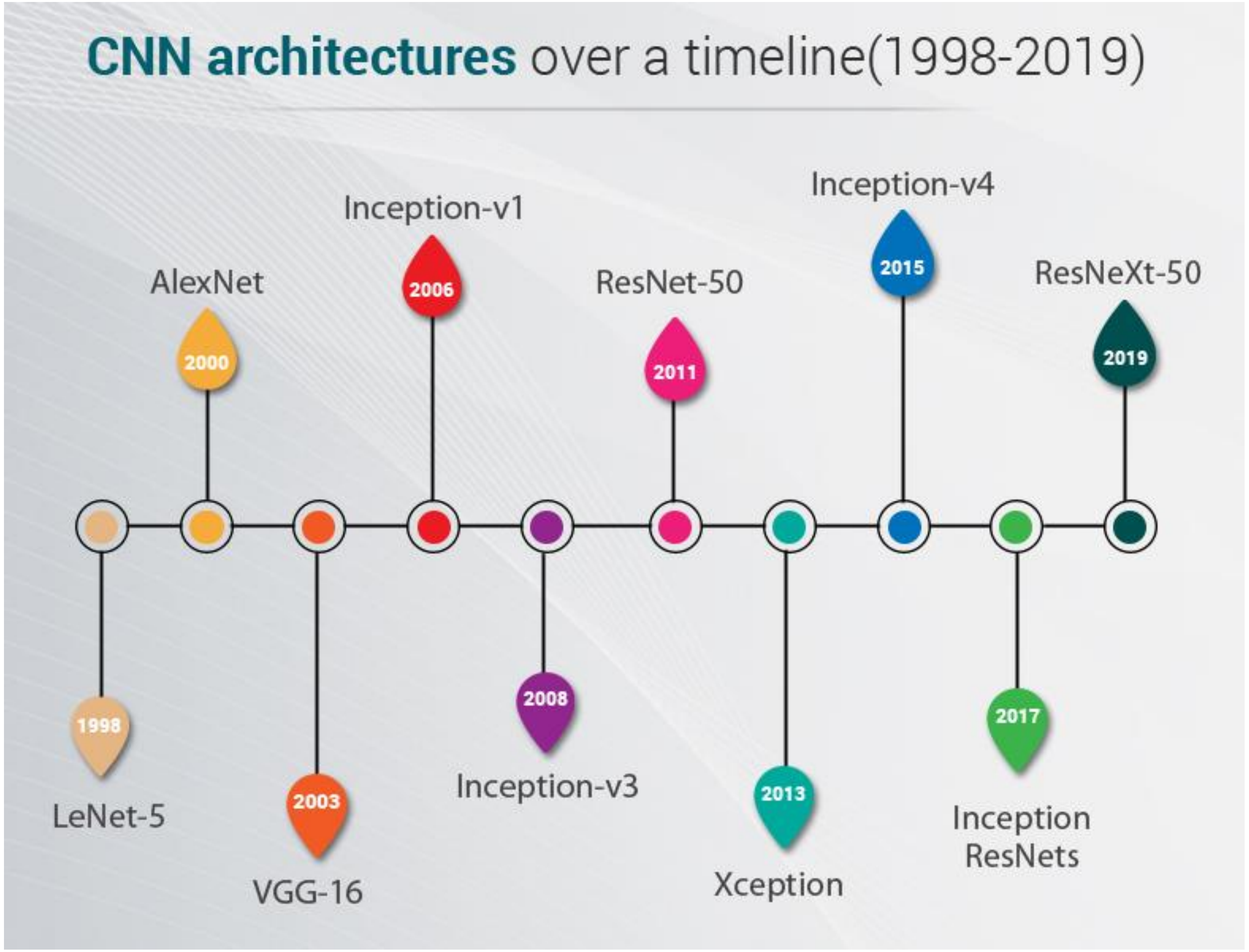
- In a convolutional network (ConvNet), there are basically three types of layers:
    1. Convolution layer
    2. Pooling layer
    3. Fully connected layer

After each convolution there is non-linearity applied using activation functions- Relu/Leaky Relu. H1=g(a1)

.



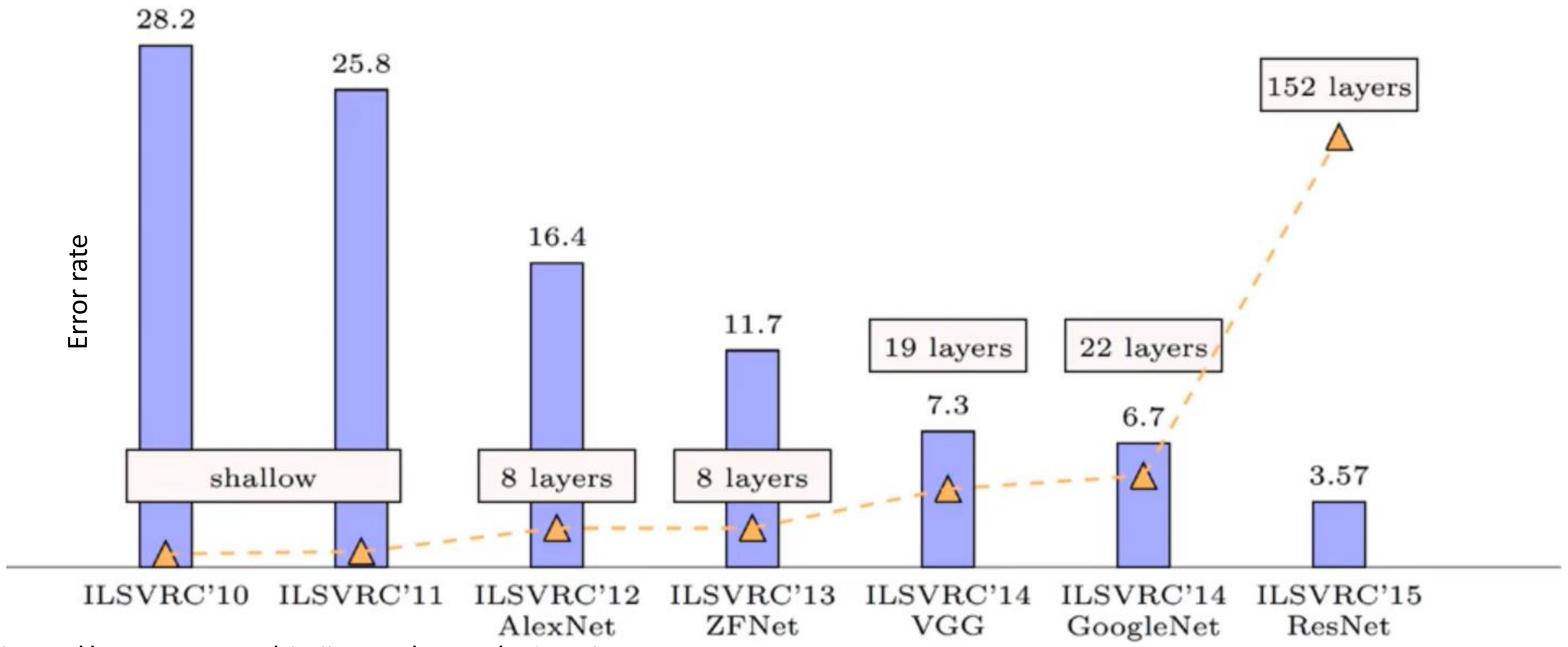
Different CNN Architectures



- **No: of Layers :**How many convolutional, max pooling fully connected layers?
- **No: of Filters in each layer**
- **Filter Size**
- **Max pooling:** What arrangement? 2 convolutional and then maxpooling or alternate convolutional and maxpooling?

Use standard tried and tested architectures!

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



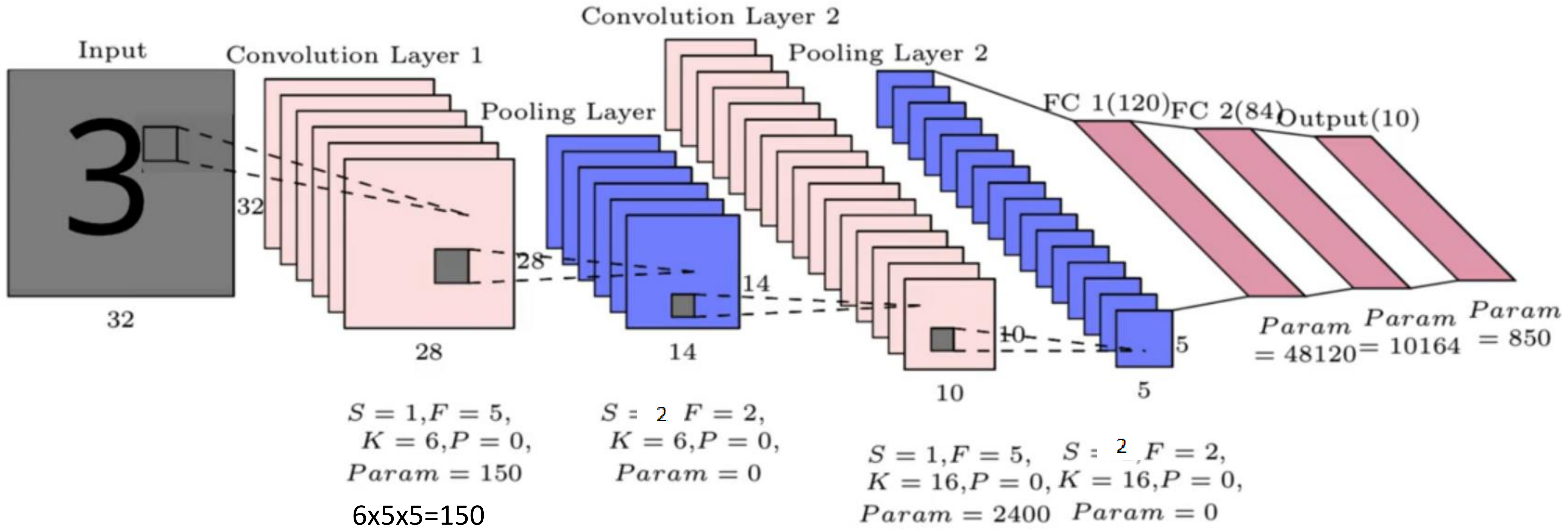
https://image-net.org/challenges/LSVRC/index.php

CNN Architecture

- LeNet

LeNet-5 – First CNN Architecture

Earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.



6x5x5=150

5x5x6x16=2400

Depth of filter is always depth of the input

After each convolution there is non-linearity applied using activation functions- Relu/Leaky Relu. h1=g(a1)

LeNet-5 – First CNN Architecture

Calculation of no of parameters

Conv Layer 1

$$S = 1, F = 5,$$
$$K = 6, P = 0,$$
$$Param = 150$$

S=1,F=5,K=6
W0=(32+0-5)/1+1=28,
As no of filters 6 we get 6 outputs of 28x28 (Filter size 5x5)
No: of parameters= 5x5x6=150

[Comparing with fully connected
Flatten 32x32
Flatten 28x28x6
Total parameters =
32x32x28x28x6]

Conv Layer 2

$$S = 1, F = 5,$$
$$K = 16, P = 0,$$
$$Param = 2400$$

S=1,F=5,K=16
W0=(14+0-5)/1+1=10,
As no of filters 16 we get 16 outputs of 10x10 from 6 input images(Filter size 5x5)

No: of parameters= 5x5x6x16=2400

[Comparing with fully connected
14x14x6x10x10x16]

FC1

Flatten 5x5x16=400 neurons
Hidden layer size=120
Bias from each node comes to 120
Parameters = 400x120+120=48120

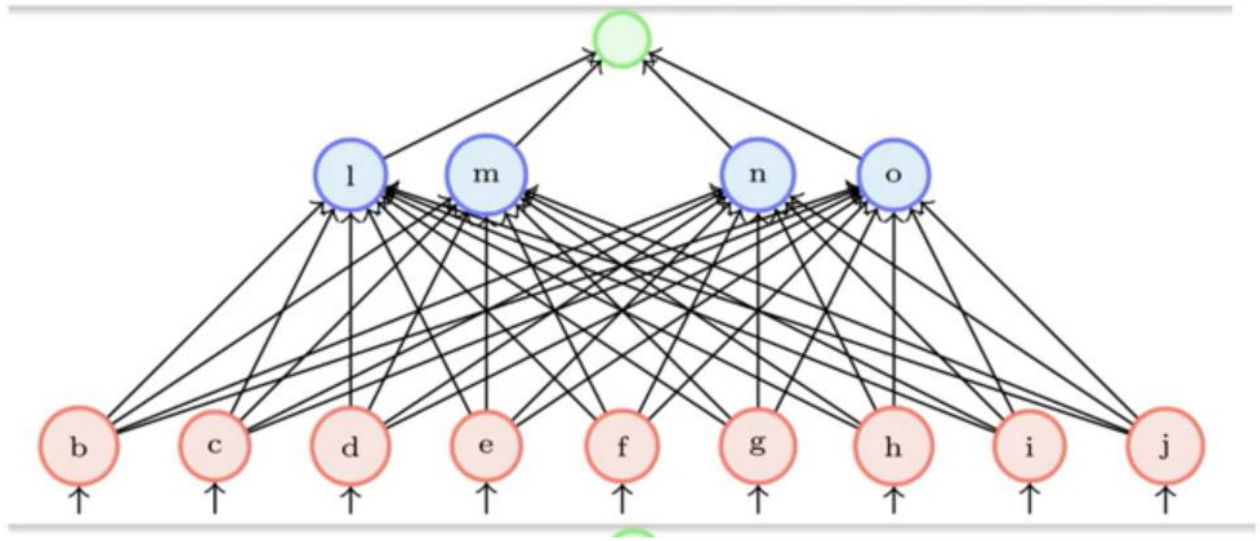
FC2

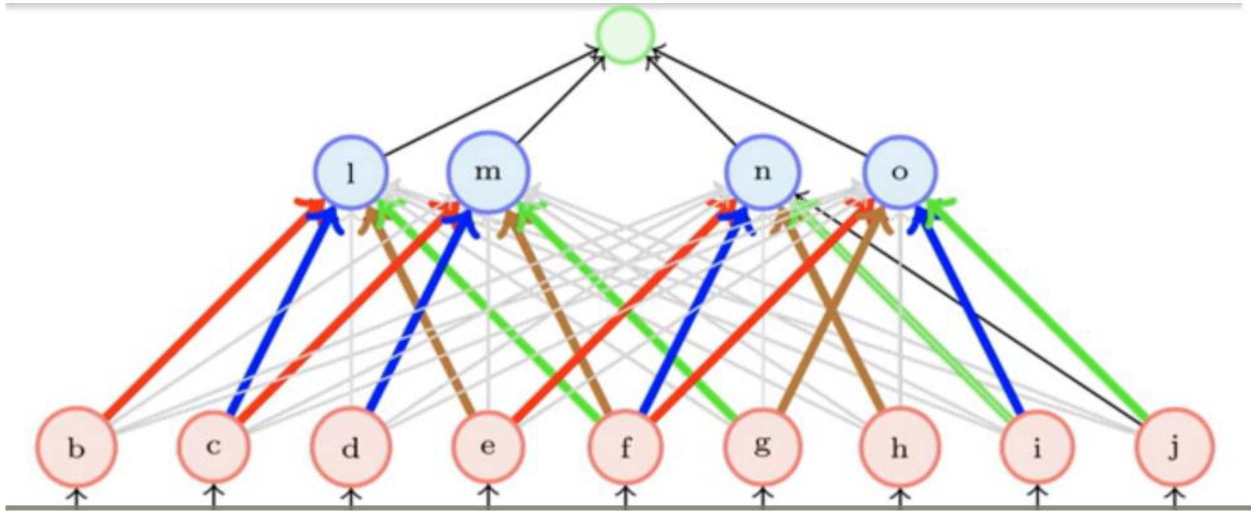
Hidden layer 1 size= 120
Hidden layer2 size =84
Bias from each of 84 hidden layer neuron=84
No of parameters=
120x84+84=10164

FC3

Hidden layer 1 size= 84
Hidden layer2 size =10
Bias from each of 10 hidden layer neuron=10
No of parameters= 84x10=850

How to train a Convolutional Neural Network





- A CNN can be implemented as a feedforward network
- wherein only a few weights (in color) are active
- the rest of the weights (in gray) are zero

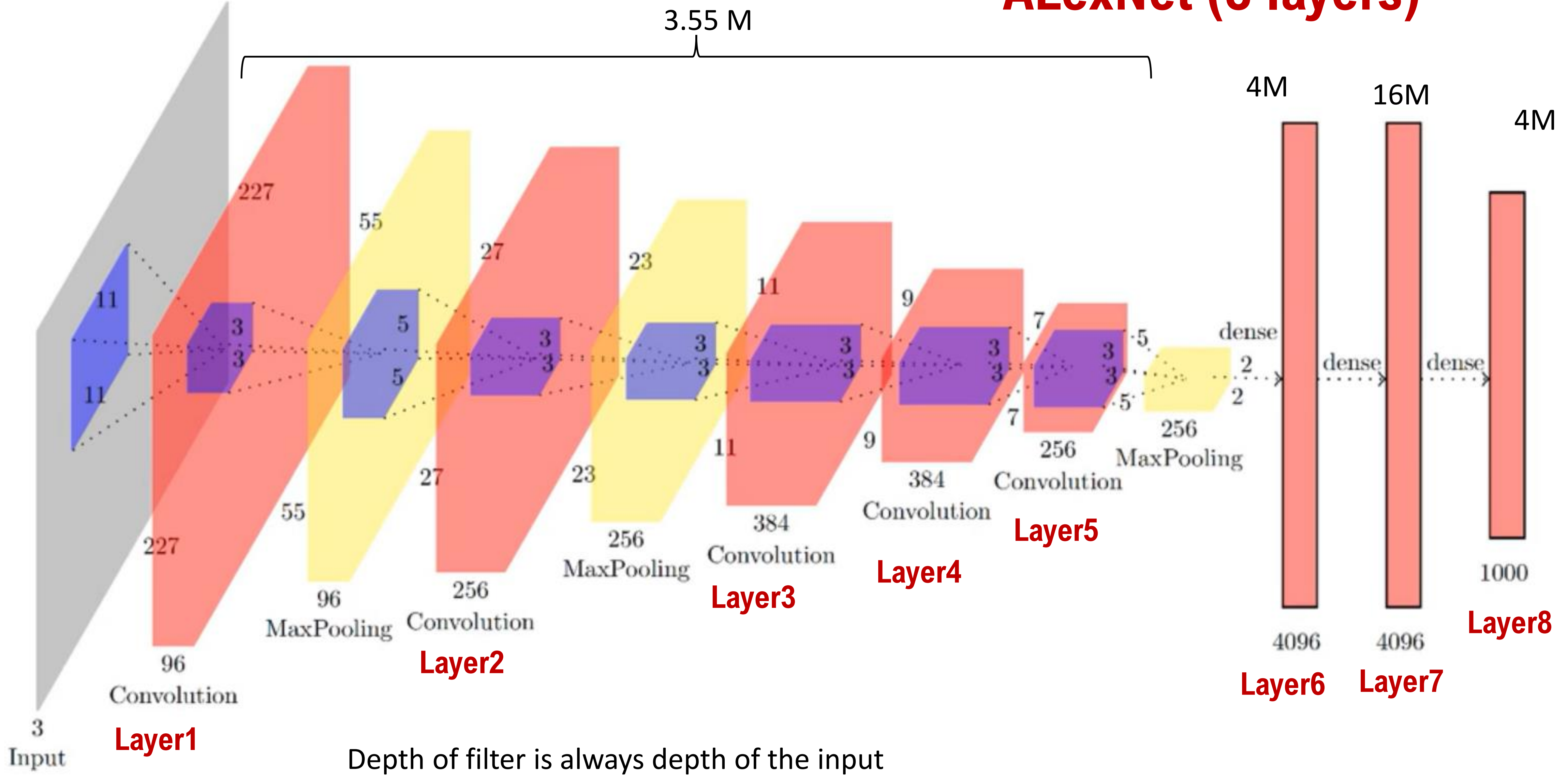
Deep learning frameworks have optimized codes which does not have to do the zero weight calculations or storage

Convolutional Neural Network Architectures

AlexNet

AlexNet is the name of a convolutional neural network (CNN) architecture, designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, who was Krizhevsky's Ph.D. advisor in the year2000

ALexNet (8 layers)



Depth of filter is always depth of the input

ALexNet (8 layers)

Input: $227 \times 227 \times 3$
Conv1: $K = 96, F = 11$
$S = 4, P = 0$
Output: $W_2 = 55,\ H_2 = 55$
Parameters: $(11 \times 11 \times 3) \times 96 = 34K$

Max Pool Input: $55 \times 55 \times 96$
$F = 3, S = 2$
Output: $W_2 = 27,\ H_2 = 27$
Parameters: 0

Input: $27 \times 27 \times 96$
Conv1: $K = 256, F = 5$
$S = 1, P = 0$
Output: $W_2 = 23,\ H_2 = 23$
Parameters: $(5 \times 5 \times 96) \times 256 = 0.6M$

Max Pool Input: $23 \times 23 \times 256$
$F = 3, S = 2$
Output: $W_2 = 11,\ H_2 = 11$
Parameters: 0

Input: $11 \times 11 \times 256$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 9,\ H_2 = 9$
Parameters: $(3 \times 3 \times 256) \times 384 = 0.8M$

Input: $9 \times 9 \times 384$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 7,\ H_2 = 7$
Parameters: $(3 \times 3 \times 384) \times 384 = 1.327M$

Input: $7 \times 7 \times 384$
Conv1: $K = 256, F = 3$
$S = 1, P = 0$
Output: $W_2 = 5,\ H_2 = 5$
Parameters: $(3 \times 3 \times 384) \times 256 = 0.8M$

Max Pool Input: $5 \times 5 \times 256$
$F = 3, S = 2$
Output: $W_2 = 2,\ H_2 = 2$
Parameters: 0

FC1
Parameters: $(2 \times 2 \times 256) \times 4096 = 4M$

FC1
Parameters: $4096 \times 4096 = 16M$

FC1
Parameters: $4096 \times 1000 = 4M$

Total Parameters: $27.55M$

Convolutional Neural Network Architectures

ZFNet(2013)

ZFNet (2013)

- The **ImageNet** project is a large visual database designed for use in visual object recognition software research. The ImageNet project runs an annual software contest, the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**, where software programs compete to correctly classify and detect objects and scenes.

- ILSVRC 2013 winner was a CNN which became known as ZFNet. It achieved a top-5 error rate of 14.8%
- It was mostly an achievement by tweaking the hyper-parameters of AlexNet while maintaining the same structure with additional Deep Learning elements
- Compared to AlexNet, the filter sizes are reduced and the stride of the convolutions are reduced.

The Top-5 error rate is the percentage of test examples for which the correct class was not in the top 5 predicted classes.

ZFNet (2013)

The ILSVRC 2013 winner was a CNN from Matthew Zeiler and Rob Fergus. It became known as **ZFNet**.
It improved on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller, going from 11 x 11 stride 4 in AlexNet to 7 x 7 stride 2 in ZFNet.

The intuition behind this was that by using **bigger filters we were losing a lot of pixel information** and a smaller filter size in the first convolution layer helps to retain a lot of the original pixel information.

The number of filters increase as we go deeper. This network also used **ReLUs** for their activation and trained using **batch stochastic gradient descent.**

Also, AlexNet was trained on 15 million images, while ZFNet was trained on only **1.3 million images**:
obtain a test error of **14.8%**, on ImageNet2012 dataset

https://arxiv.org/pdf/1311.2901v3.pdf
Source paper

BY Matthew D. Zeiler zeiler, Rob Fergus
Dept. of Computer Science, Courant Institute, New York University

ZFNet (2013)

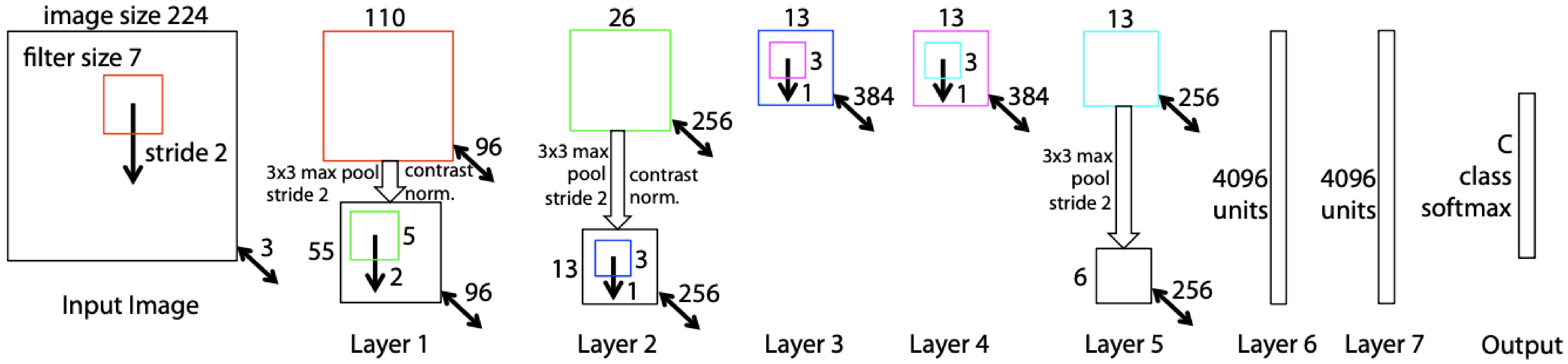
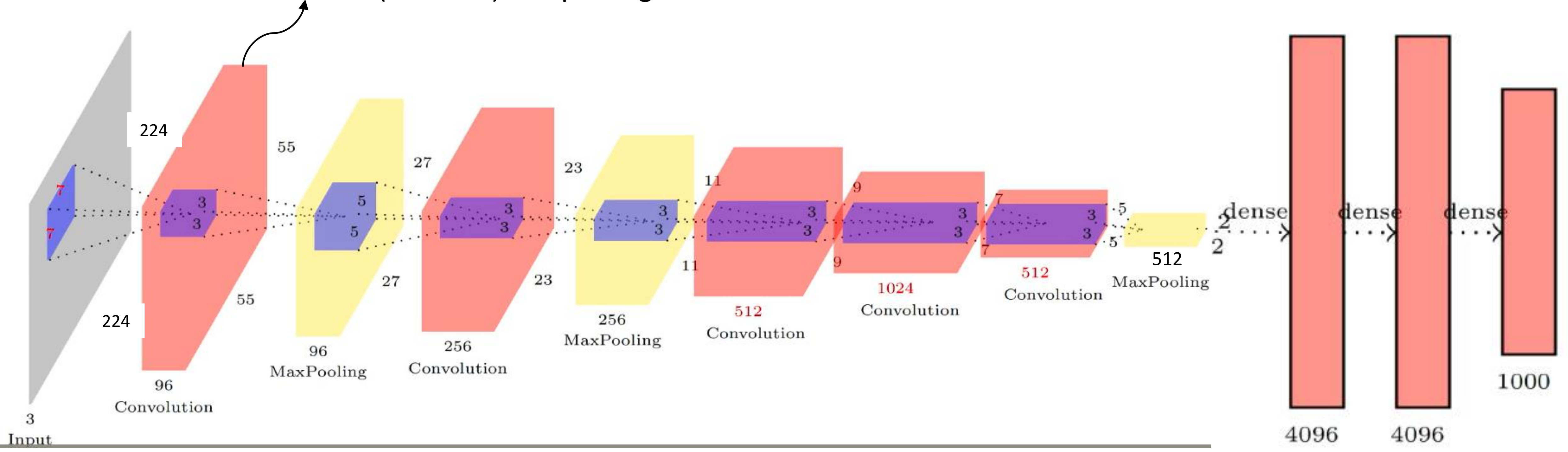


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

ZFNet Architecture

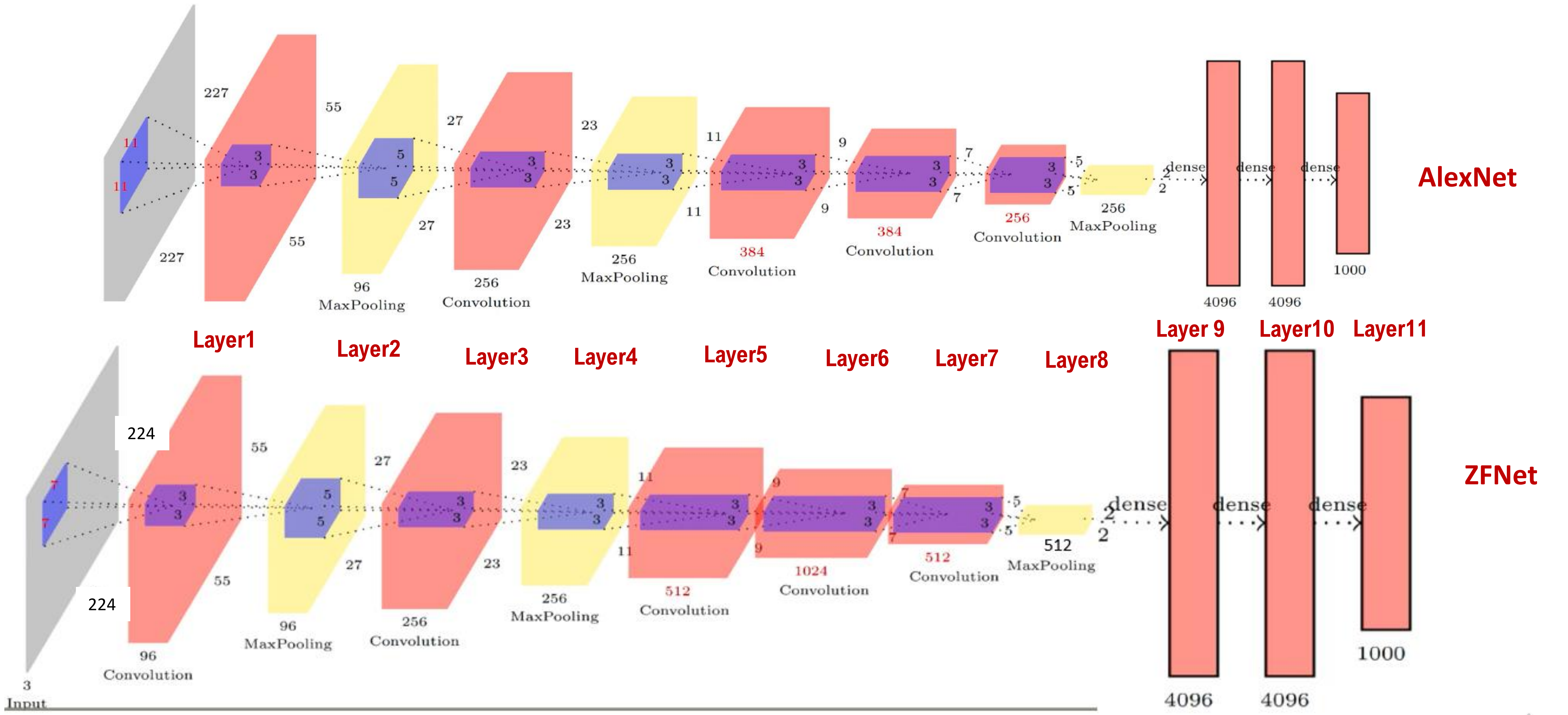
After convolution (110x110) and pooling



Layer I : 224x224 (3 channel) convolved with 96 filters of 7x7 size and stride =2, results in 96, 110x110 image
Wo=(Wi+2P-F) /S+1=(224+0-7)/2+1=110
This is then subjected to maxpooling resulting in 55x55

AlexNet-ZFNet Difference

Maxpooling also counted as layer for understanding purpose.



AlexNet-ZFNet Difference in parameters

Layer1: $F = 11 \rightarrow 7$
Difference in Parameters
$((11^2 - 7^2) \times 3) \times 96 = 20.7K$

11x11x3x96-7x7x3x96

Layer2: No difference

Layer3: No difference

Layer4: No difference

Layer5: $K = 384 \rightarrow 512$
Difference in Parameters
$(3 \times 3 \times 256) \times (512 - 384) = 0.29M$

3x3x256x512 – 3x3x256x384

Layer6: $K = 384 \rightarrow 1024$
Difference in Parameters
$(3 \times 3 \times ((384 \times 384) - (512 \times 1024)) = 0.8M$

Layer7: $K = 256 \rightarrow 512$
Difference in Parameters
$(3 \times 3 \times ((384 \times 256) - (1024 \times 512)) = 0.36M$

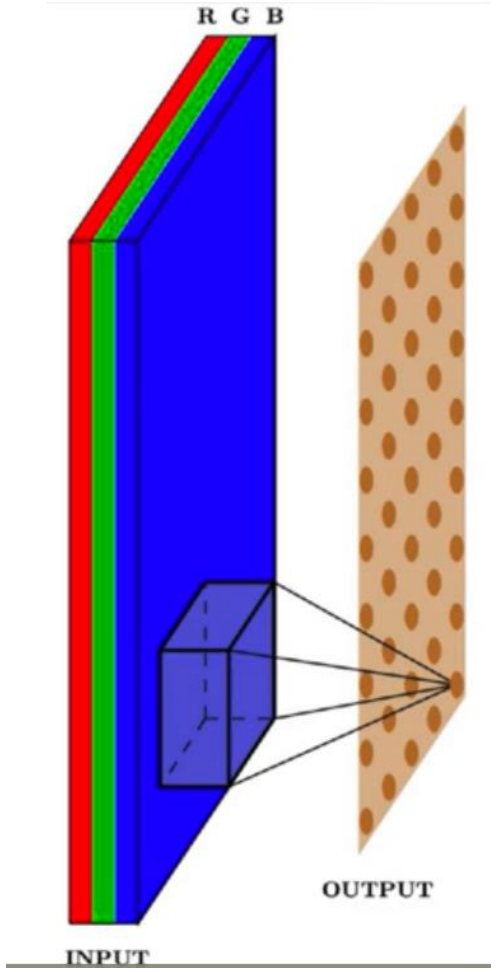
Layer8: No difference

Layer9: No difference

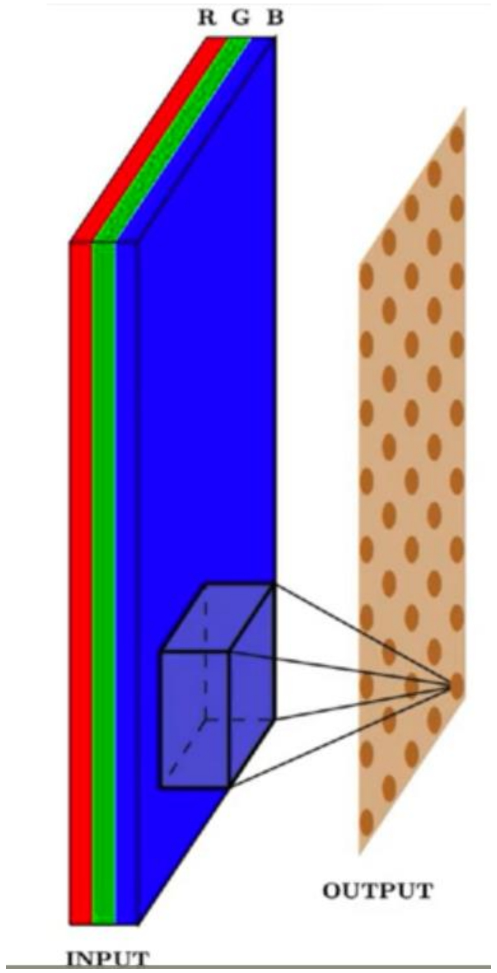
Layer10: No difference

Convolutional Neural Network Architectures GoogLeNet

No: of computations in CNN



No: of computations in CNN



Assume S=1, and with appropriate padding WI=Wo=W and HI=Ho=H
Each pixel need a computation of WIxHI

Each pixel takes FxFxD computations
There are WoxHo such pixels
So total computations WoxHoXFxFxD

Note that computation depends on D the depth (Intermediate layers may have bigger depth as no of filters)

GoogLeNet - Intuitions

Choice of filters- Parallel Convolutions and Max-pooling

Reduce the number of parameters- Average pooling

Reduce the computations – 1x1 Convolutions

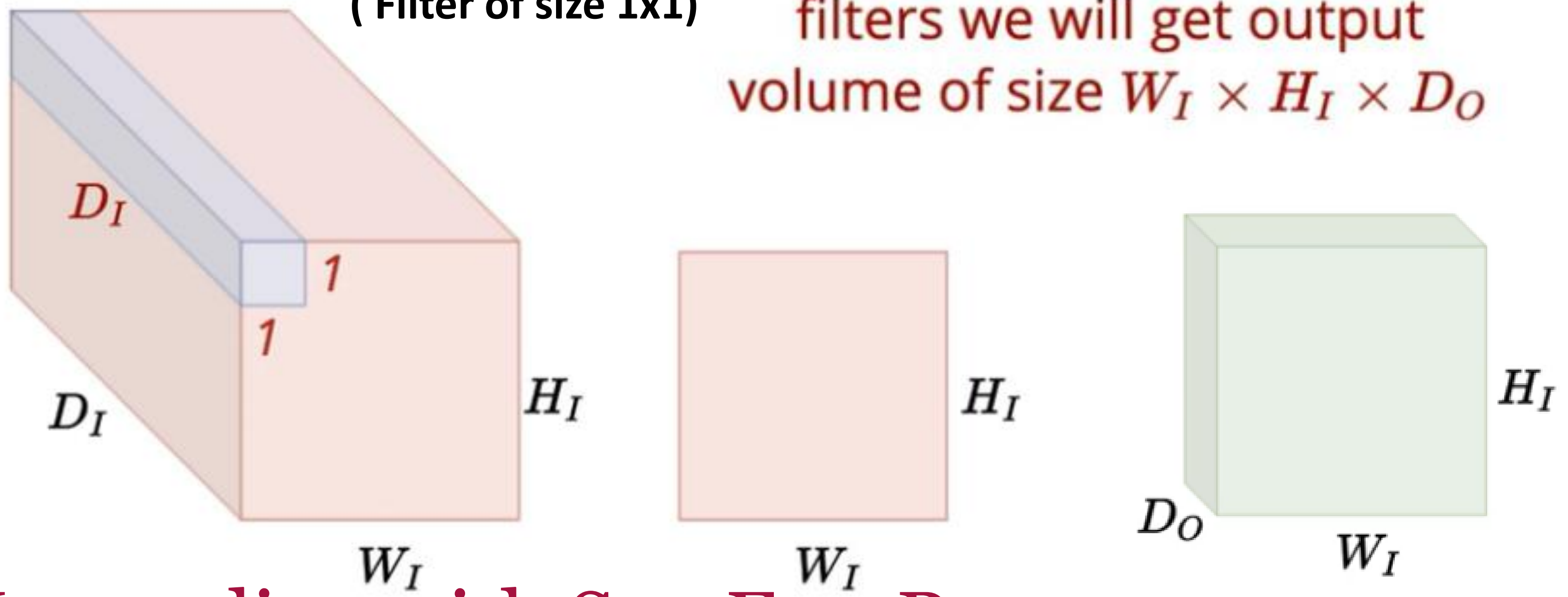
Deeper Networks

Intuitions behind GoogleNet

1x1 Convolution

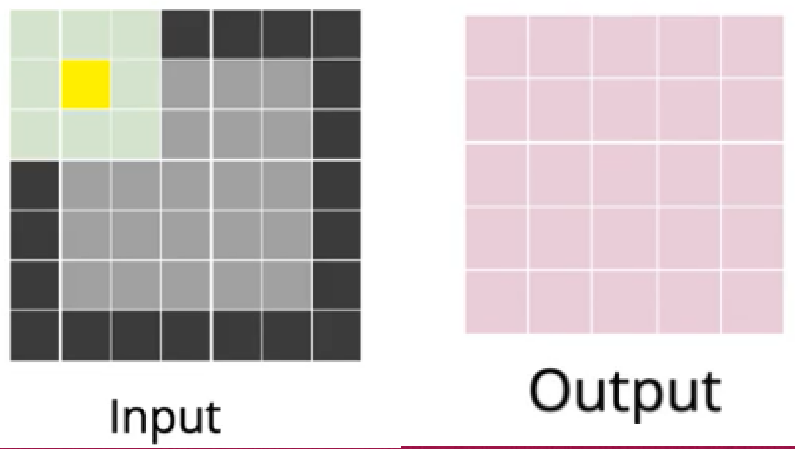
(Filter of size 1x1)

If we use $D_O (< D_I)$ such filters we will get output volume of size $W_I \times H_I \times D_O$



It is computationally intensive to use many filters K which increases the depth (D1) of output. 1x1 convolution which basically does weighted average helps in shrinking the volume from DI to Do(Do<D1) but at retaining the values to some extend. So this intermediate step helps in reducing the complexity by shrinking the intermediate inputs (Do kernels used)

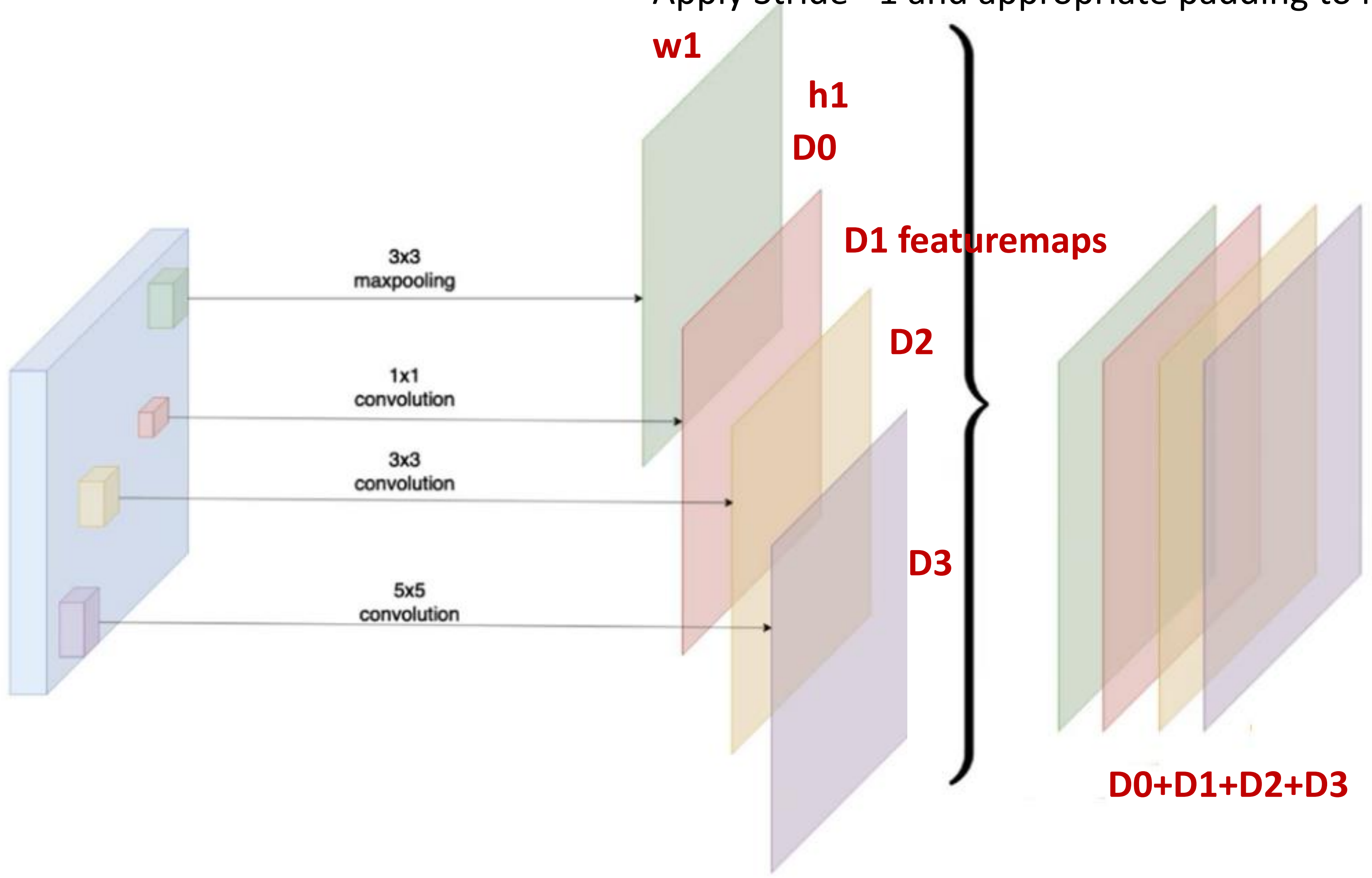
Maxpooling with S=1,F=3,P=1



When we do maxpooling with S=1 and P=1 when F=3 we get an output of same size.
Similarly S=1,P=2 when F=5
So a stride of 1 and appropriate padding help us in retaining size of output.

Intuition Behind GoogLeNet

Apply Stride =1 and appropriate padding to retain same output size as input



Until now, in the architectures we had seen variations in the hyper parameters .

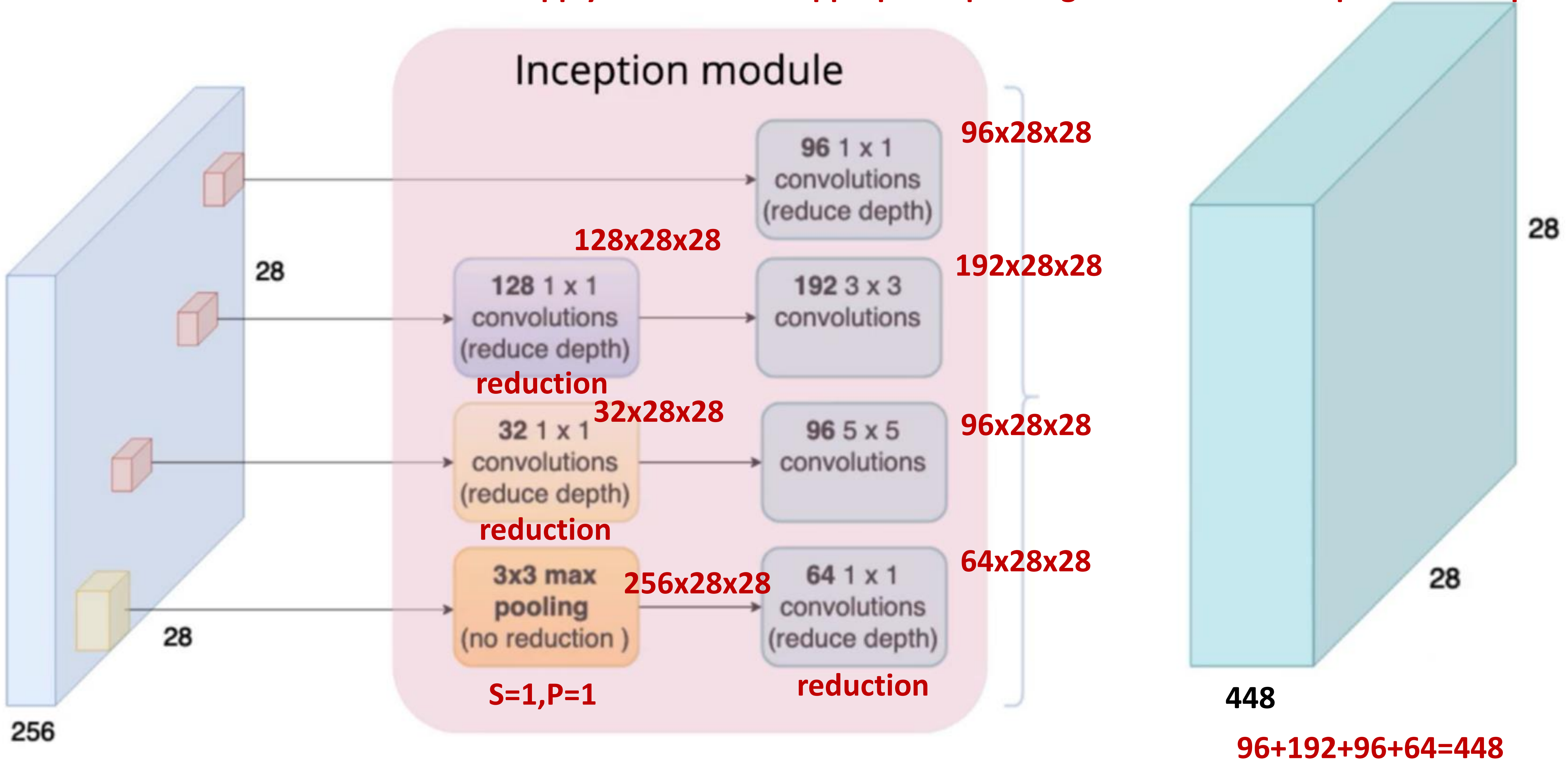
- How many filters to use and what should be filter size?
- What should be arrangement of max-pooling and convolutional layers?
- How many max pooling how many convolutional layers?

- **Intuition behind GoogLeNet-** Why to make that decision . Do everything on the same input and get a set of feature map outputs in one go- **Inception Module**

When we have back to back convolutions, filter size can be smaller(5x5 or 3x3)

Inception Module in GoogLeNet

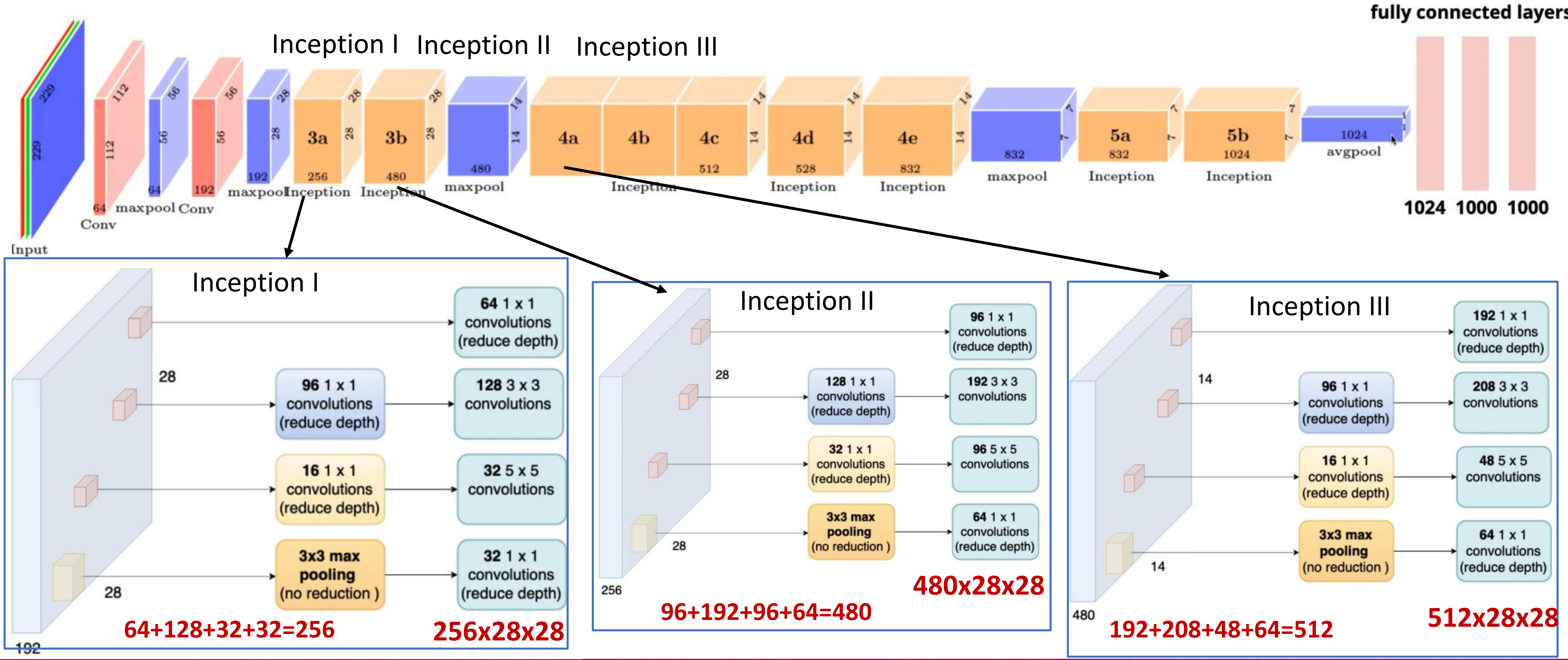
Apply Stride =1 and appropriate padding to retain same output size as input



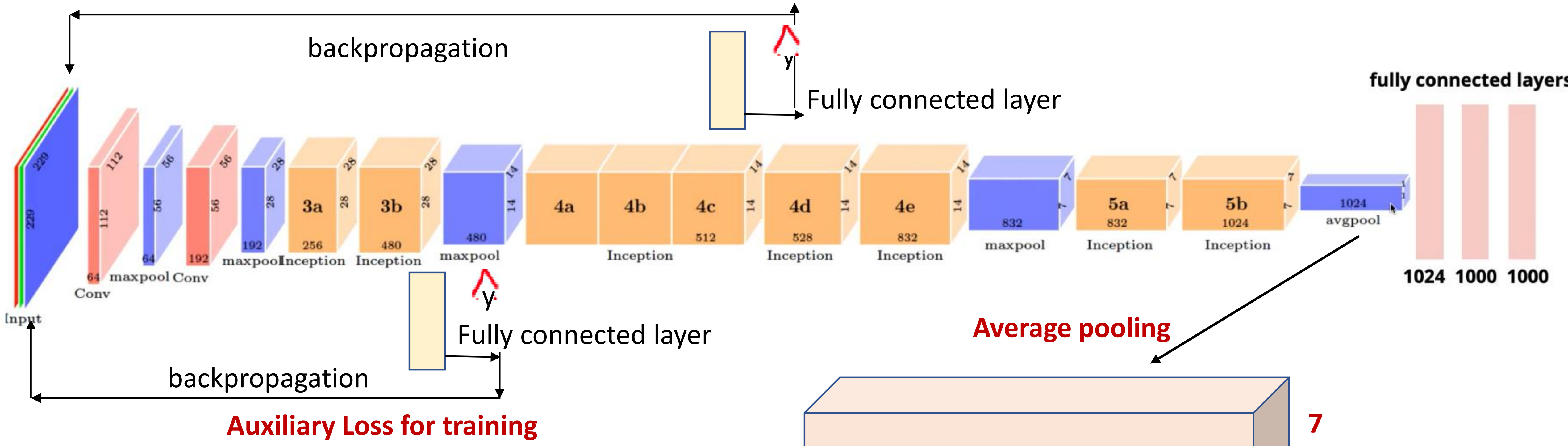
Inception module
96 1 x 1 convolutions (reduce depth)
96x28x28
128x28x28
128 1 x 1 convolutions (reduce depth)
192 3 x 3 convolutions
192x28x28
reduction
32 1 x 1 convolutions (reduce depth)
32x28x28
96 5 x 5 convolutions
96x28x28
reduction
3x3 max pooling (no reduction)
256x28x28
64 1 x 1 convolutions (reduce depth)
64x28x28
S=1,P=1
reduction
28
28
256
28
28
448

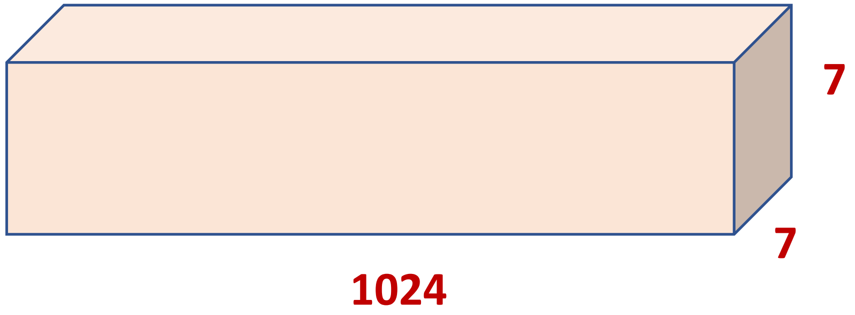
96+192+96+64=448

GoogLeNet Architecture



GoogLeNet Architecture- Average Pooling, Auxiliary Loss for training





The auxiliary loss **will add extra gradient flow during backpropogation**, thereby helping to reduce gradient vanishing problem, training stability

Calculate average of 7x7 2D image and do it depthwise resulting in a 1D vector of 1024

Conclusion

Choice of filters- Parallel Convolutions and Max-pooling

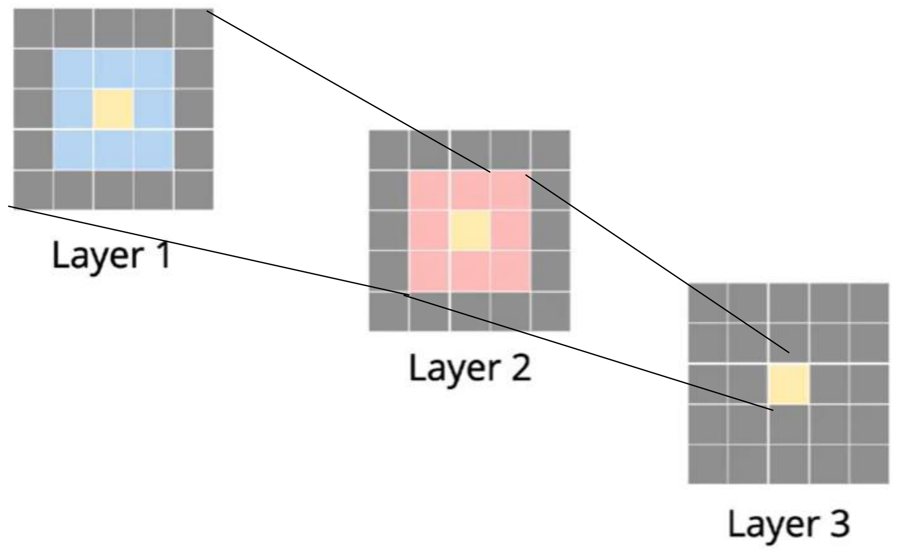
Reduce the number of parameters- Average pooling

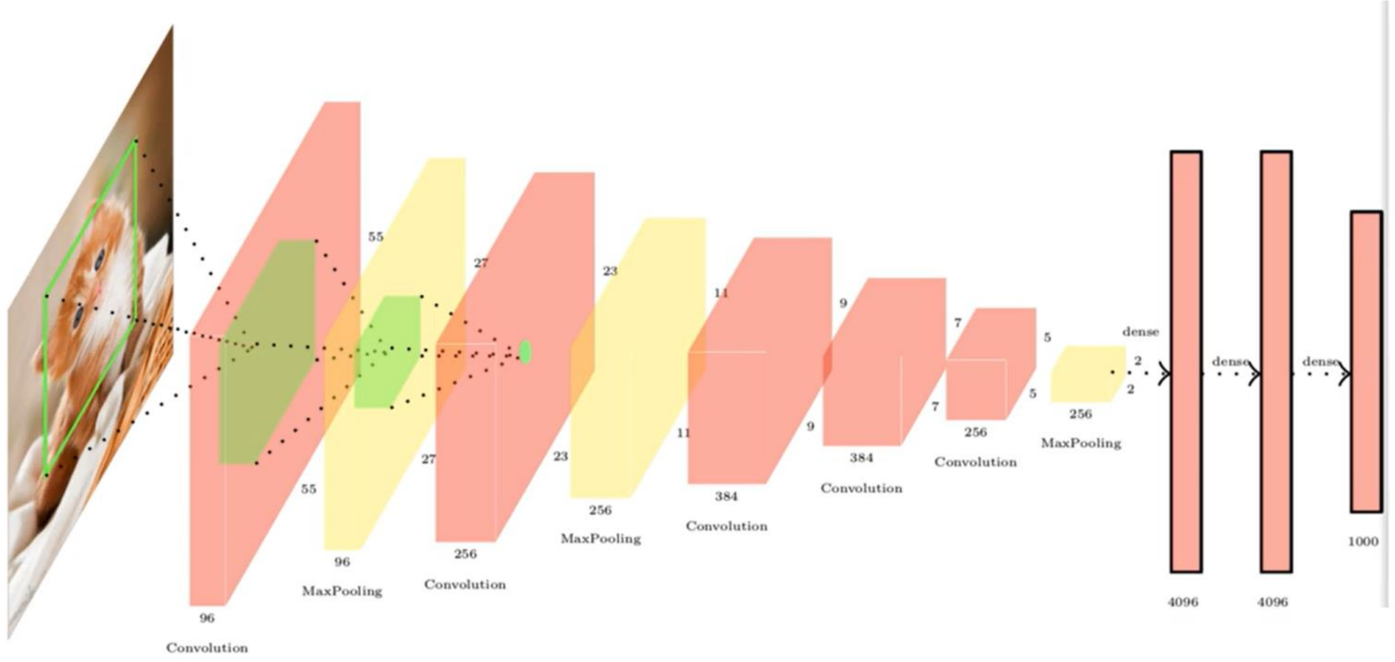
Reduce the computations – 1x1 Convolutions

Deeper Networks

Receptive Fields

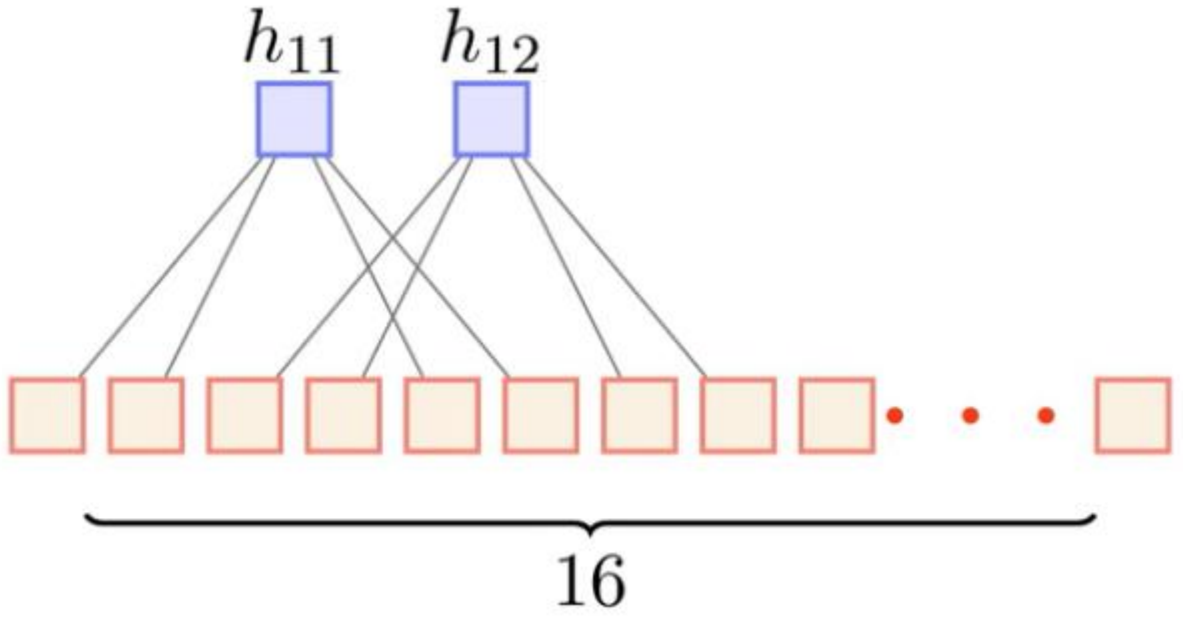
Layer 3 centre pixel is influenced by 3x3 shaded area in previous layer which is inturn influenced by 7x7 area of previous layer



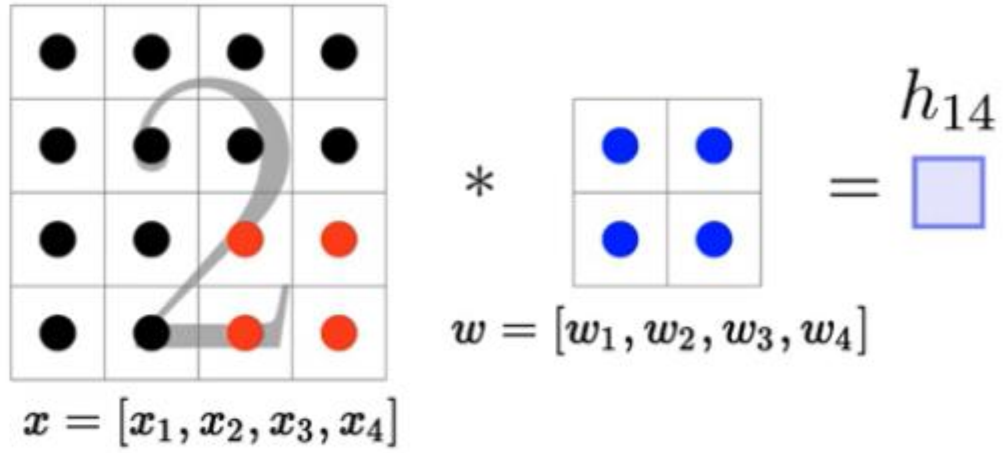


Check what is causing a neuron to fire??

Visualizing filters

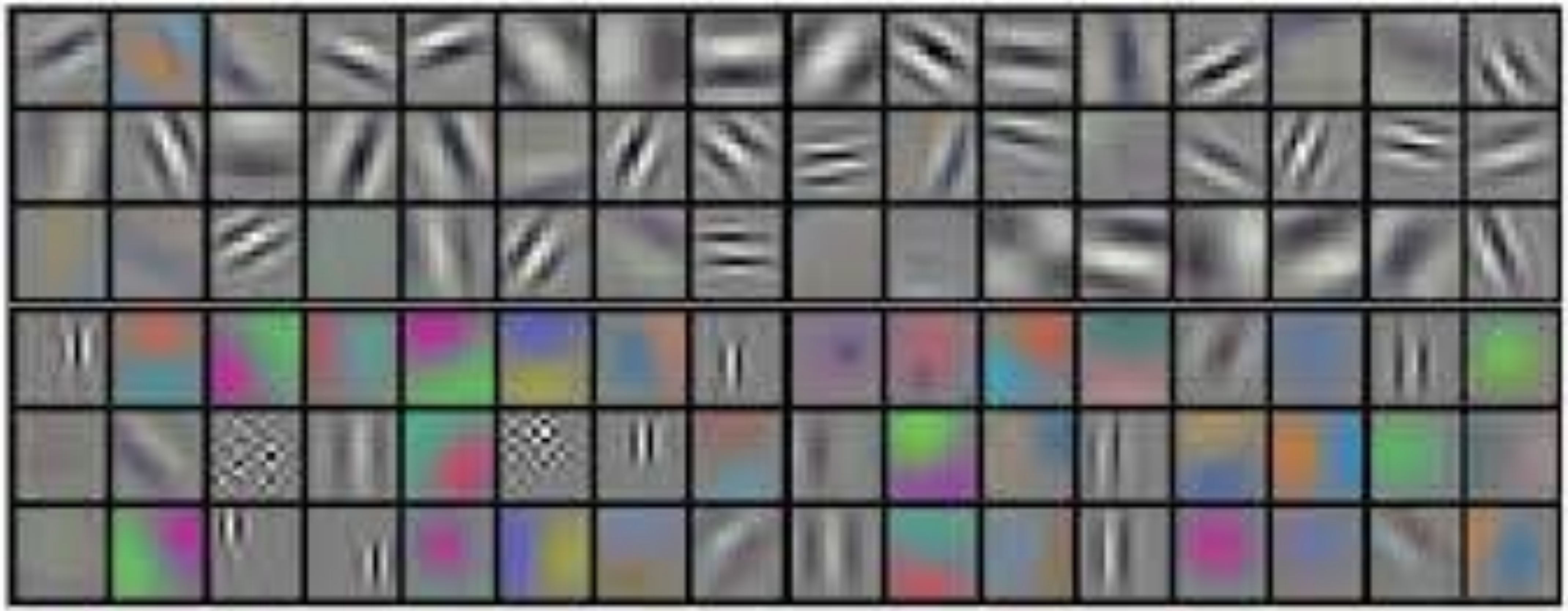


The neuron h_{14} will fire maximally when $\mathbf{x} = \frac{w}{||w||}$

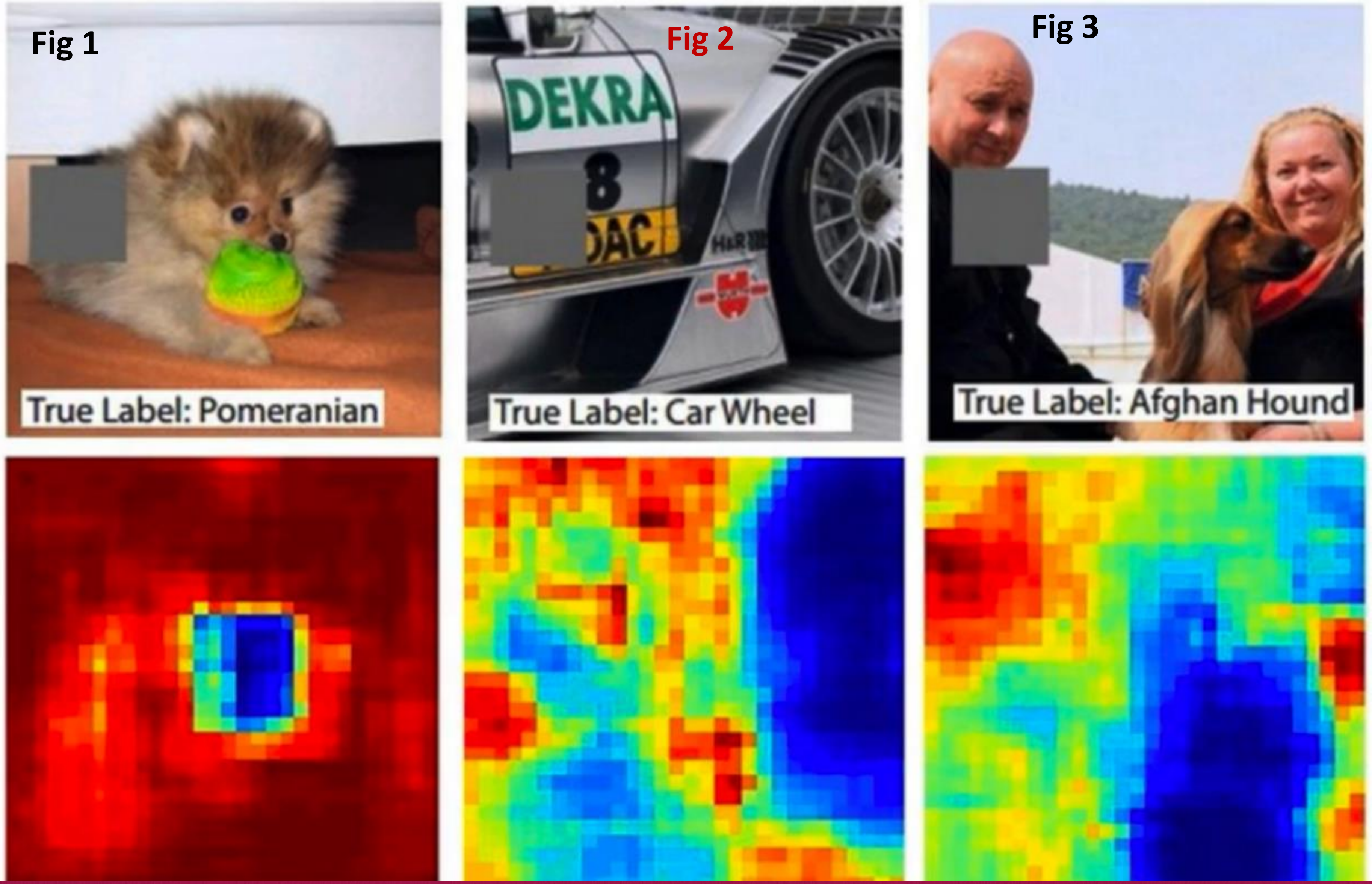


$$h_{14} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$
$$= \mathbf{w}^T\mathbf{x}$$
$$\propto cos\theta$$

H14 will be high when cos θ high, ie cos θ is 1
ie θ=0
ie w and x is aligned in same direction
wherever there is a match of filter pattern and image it fires and gives high response



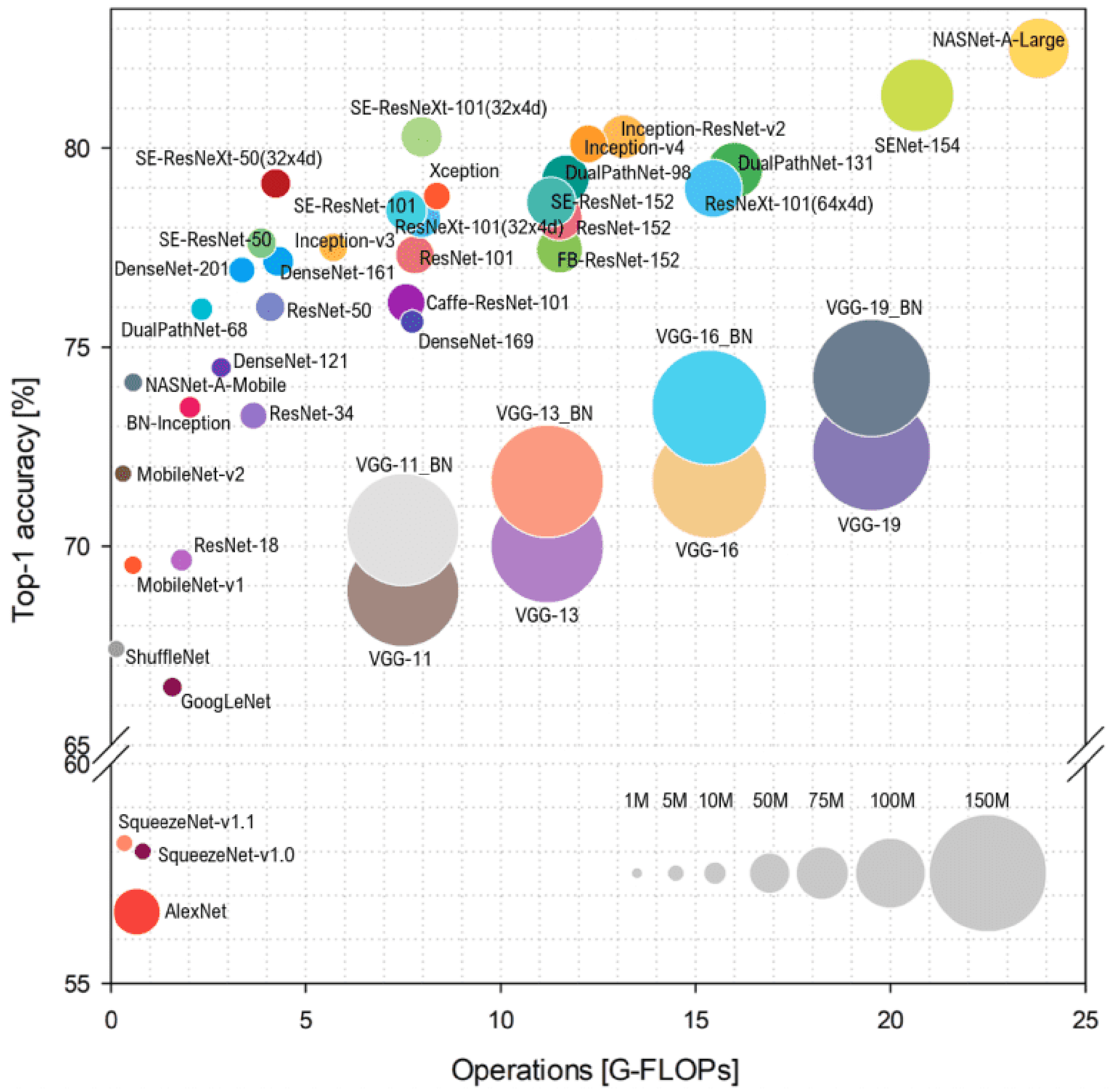
Visualization in CNN-

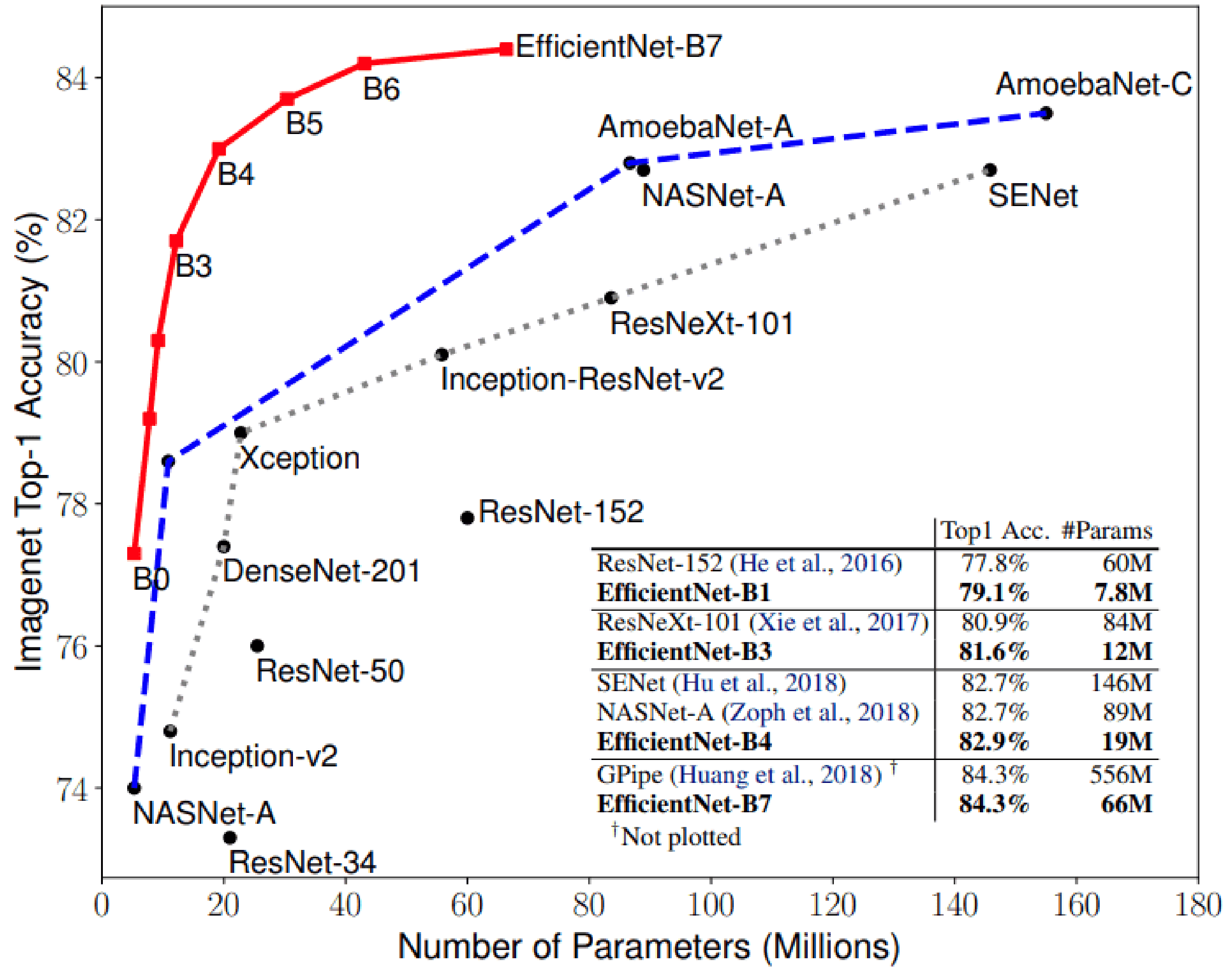


Which patch in the image contribute maximally to output

Heat Map shows the difference in the output y cap of original and after masking certain portions

- **Fig 1**: the face portion of Dog when masked resulted in substantial reduction in prediction
- **Fig2**: The wheel is maximally contributing
- **Fig 3**: The dog is maximally contributing. But lady also to small extend which is not right. May be all training sample had somebody holding the dog





	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Convolutional Neural Network Architectures

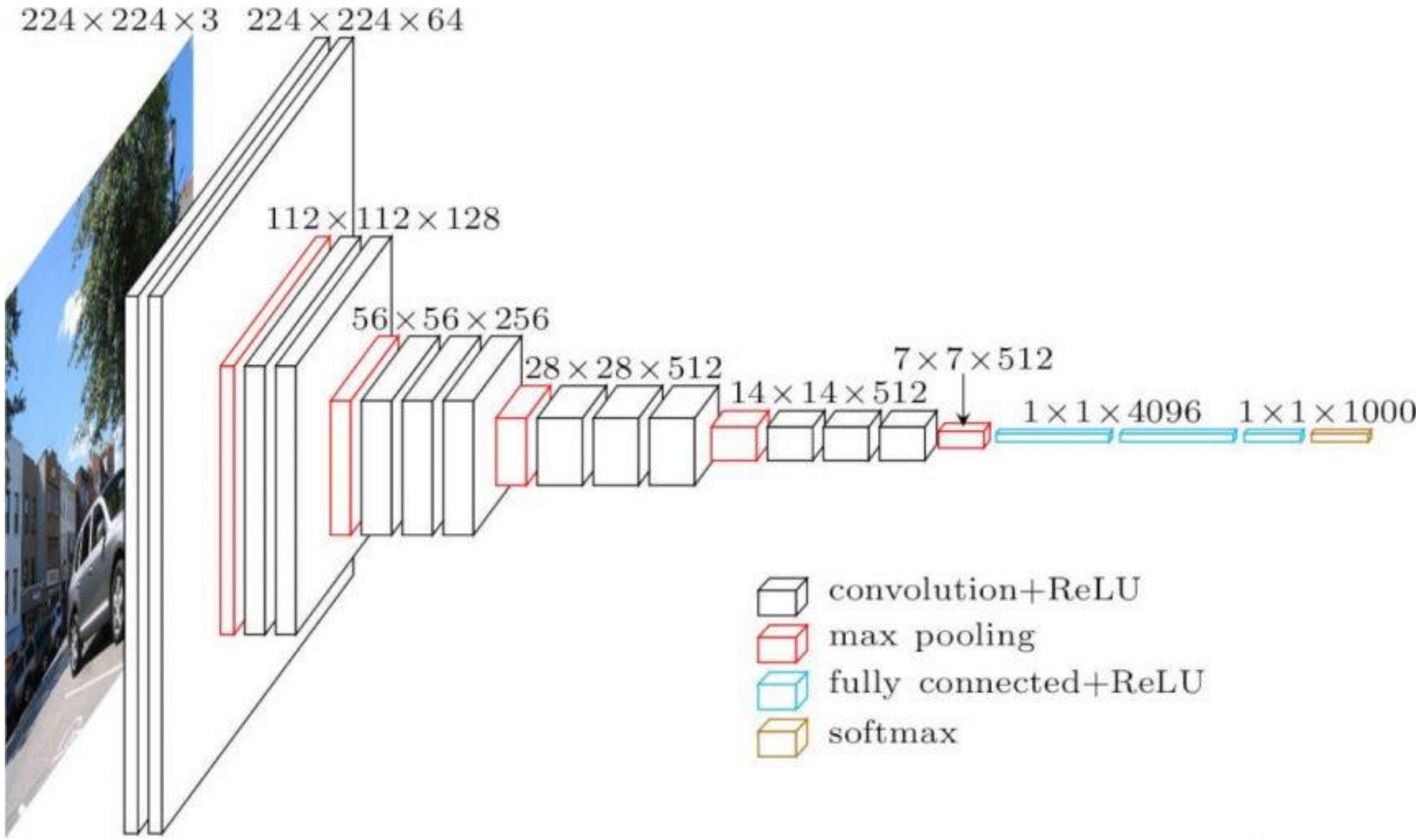
VGG 16

VGG16

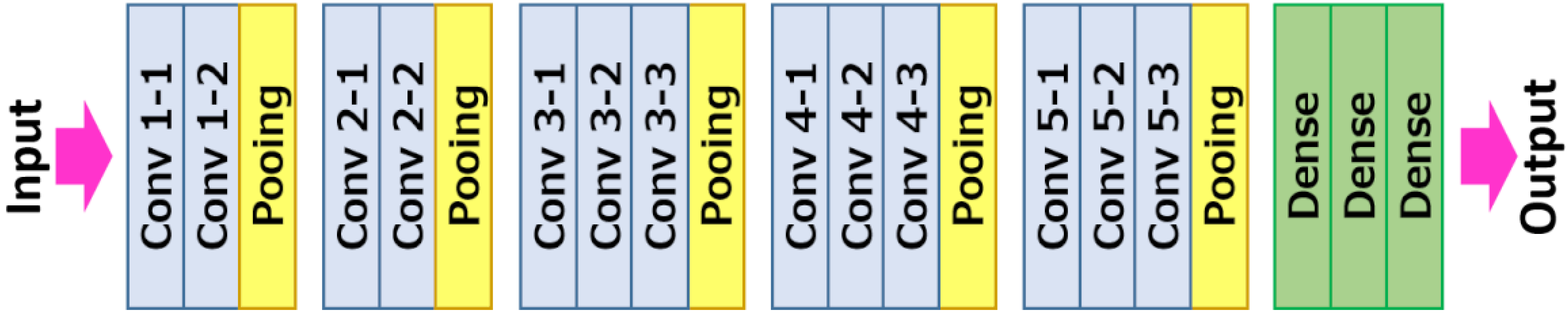
VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" in 2015. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

The authors give the intuition behind this that having two consecutive 2 consecutive 3x3 filters gives an effective receptive field of 5x5, and 3 – 3x3 filters give a receptive field of 7x7 filters,but using this we can use a far less number of hyper-parameters to be trained in the network.

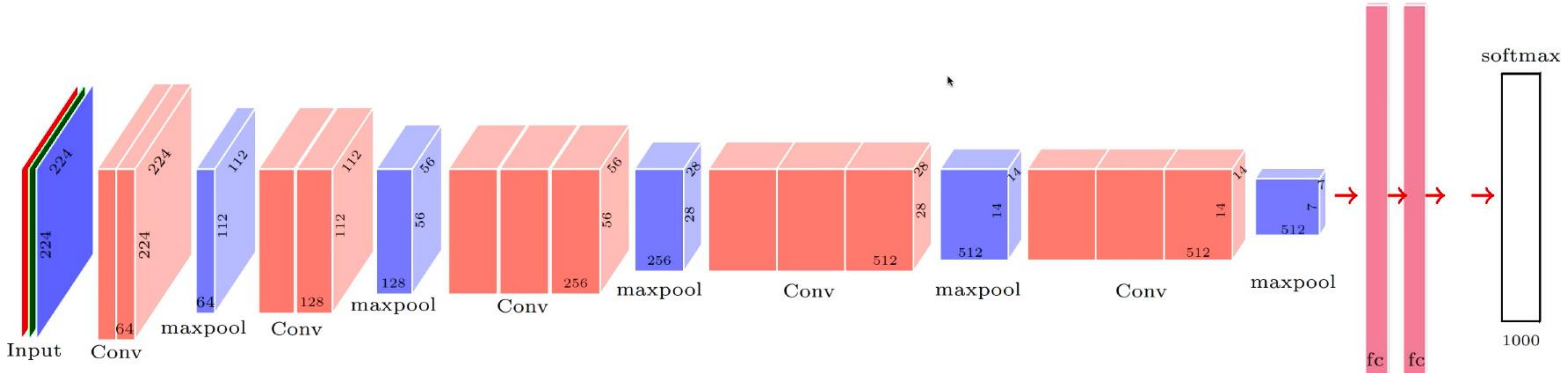
https://arxiv.org/pdf/1409.1556v4.pdf
Source paper


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

VGG-16


Input
Conv 1-1
Conv 1-2
Pooing
Conv 2-1
Conv 2-2
Pooing
Conv 3-1
Conv 3-2
Conv 3-3
Pooing
Conv 4-1
Conv 4-2
Conv 4-3
Pooing
Conv 5-1
Conv 5-2
Conv 5-3
Pooing
Dense
Dense
Dense
Output

VGG-16



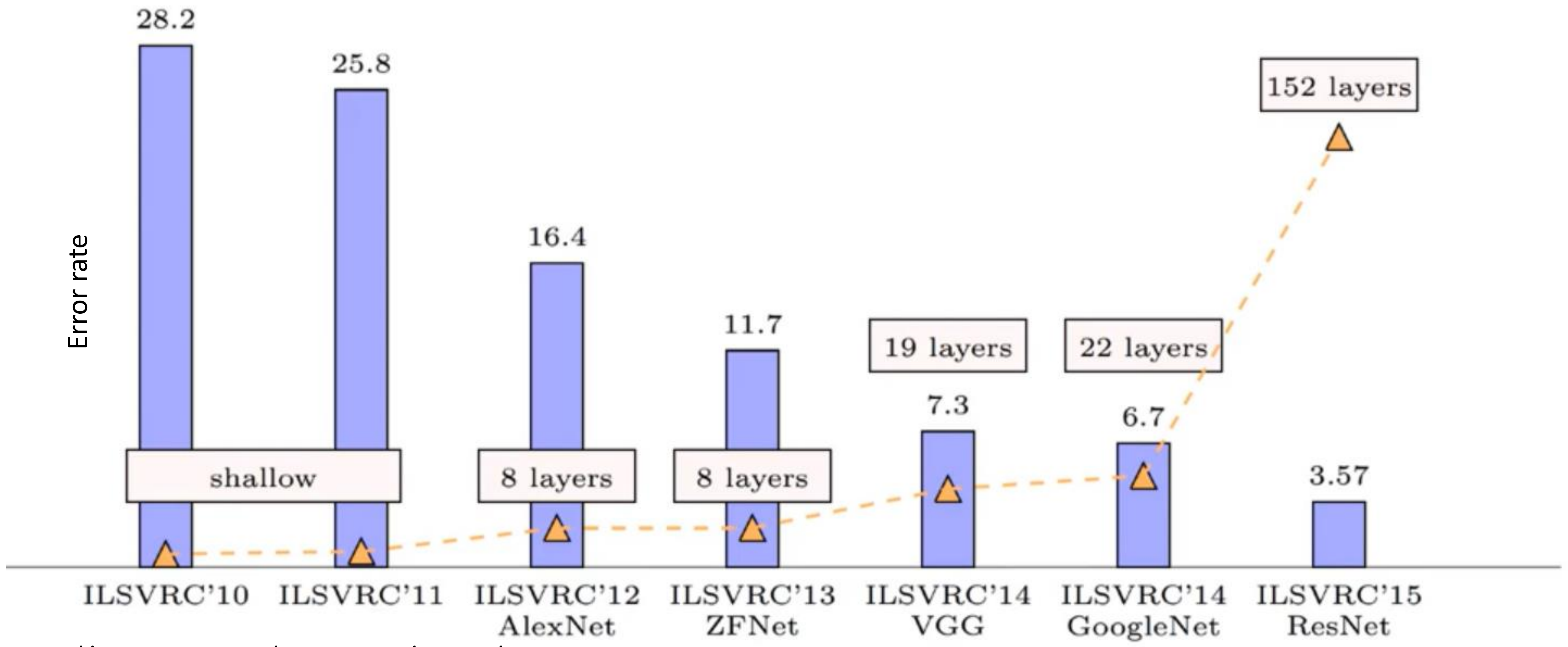
- Kernel size is 3×3 throughout
- Total parameters in non FC layers $= \sim 16M$
- Total Parameters in FC layers $= (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M$
- Most parameters are in the first FC layer ($\sim$ 102M)

Convolutional Neural Network Architectures

ResNet

— Winner of ILSVRC 2015 (Image Classification, Localization, Detection)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



https://image-net.org/challenges/LSVRC/index.php

ResNet

ResNet introduces skip connection (or **shortcut connection**) to fit the input from the previous layer to the next layer without any modification of the input. Skip connection enables to have deeper network and finally ResNet becomes the **Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation.** This is a **2016 CVPR** paper with **more than 19000 citations**

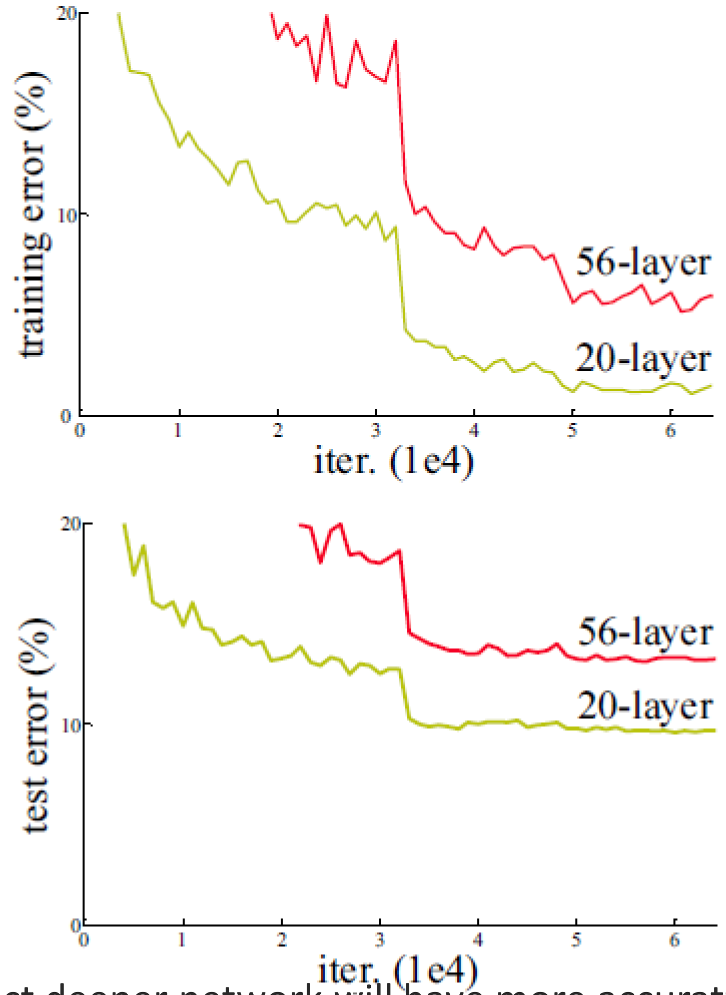
ImageNet, is a dataset of over 15 millions labeled high-resolution images with around 22,000 categories. ILSVRC uses a subset of ImageNet of around 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images and 100,000 testing images.

Background-Problems of Plain Network

For conventional deep learning networks (Plain Network), having conv layers then fully connected (FC) layers for classification task like AlexNet, ZFNet and VGGNet,
When the plain network is deeper (layers are increased), the problem of vanishing/exploding gradients occurs.

Vanishing/exploding gradients

- During backpropagation, when partial derivative of the error function with respect to the current weight in each iteration of training, this has the effect of **multiplying *n* of these small / large numbers to compute gradients** of the "front" layers in an *n*-layer network
- When the network is deep, and multiplying *n* of these small numbers will become zero (vanished).
- When the network is deep, and multiplying *n* of these large numbers will become too large (exploded).

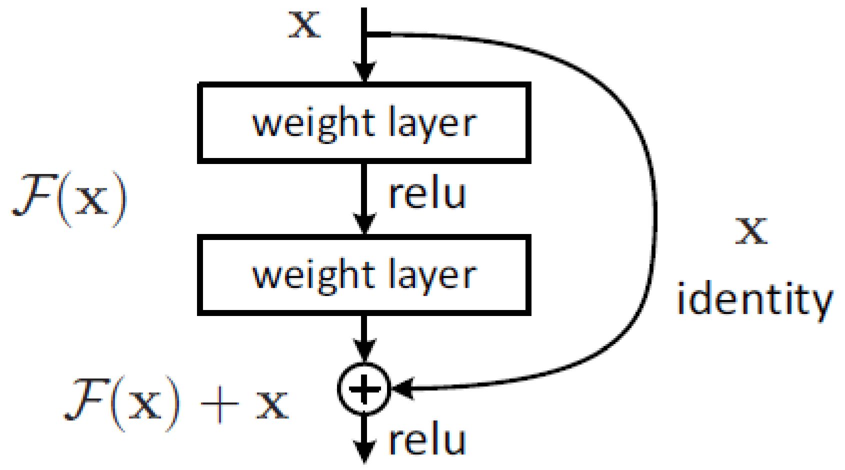


We expect deeper network will have more accurate prediction. However, above shows an example, **20-layer plain network got lower training error and test error than 56-layer plain network**, a degradation problem occurs due to vanishing gradients.

Skip / Shortcut Connection in Residual Network (ResNet)

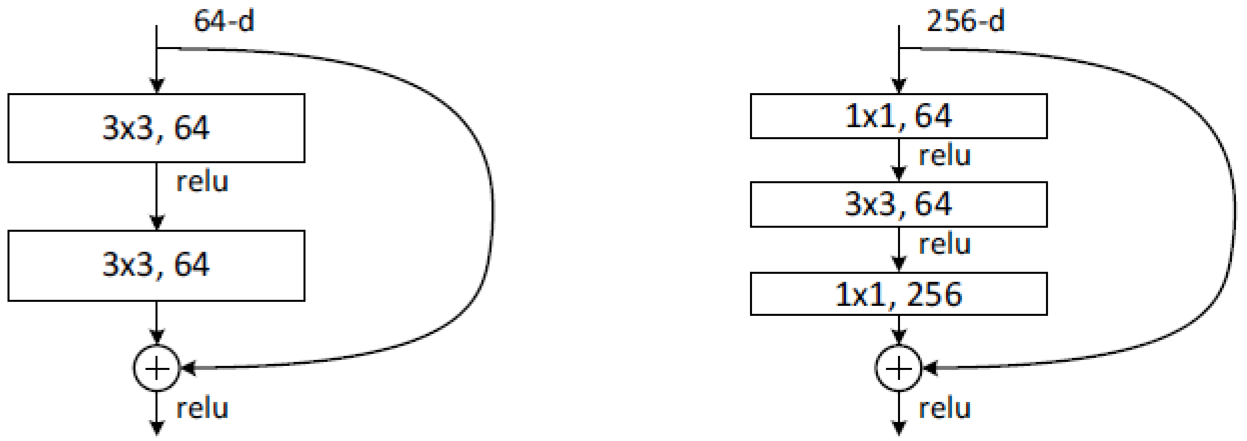
To solve the problem of vanishing/exploding gradients, a skip / shortcut connection is added to add the input *x* to the output after few weight layers as below:

A Building Block of Residual Network



x
weight layer
relu
weight layer
$\mathcal{F}(\mathrm{x})$
x
identity
$\mathcal{F}(\mathrm{x}) + \mathrm{x}$
relu

Bottleneck Design

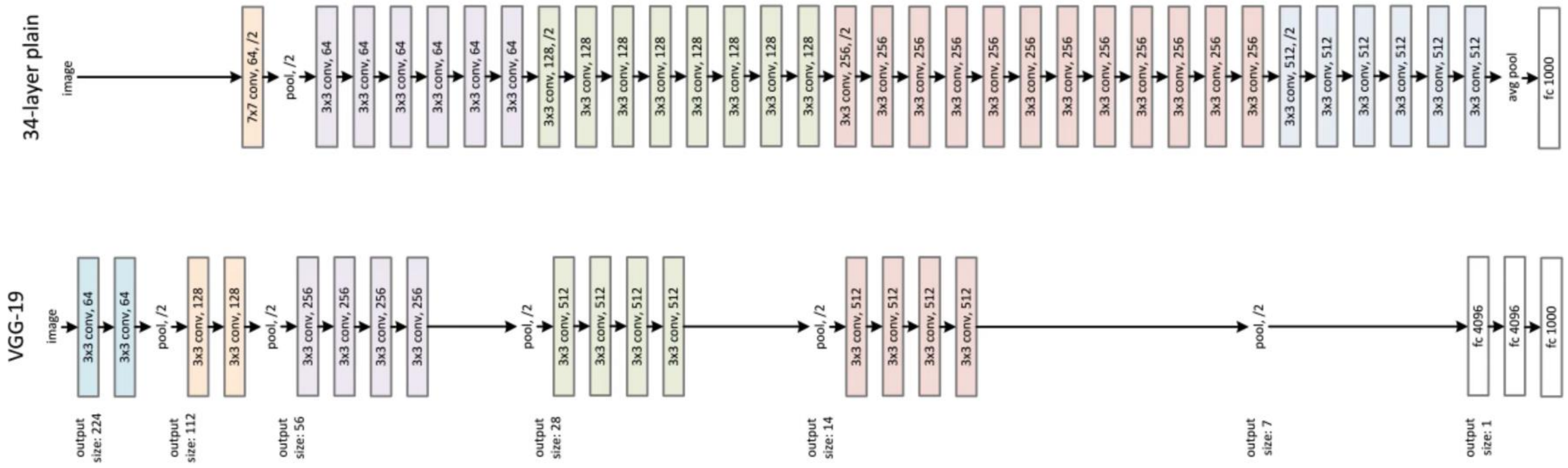


Hence, the output *H(x)= F(x) + x*. **The weight layers actually is to learn a kind of residual mapping: *F(x)=H(x)-x***.
Even if there is vanishing gradient for the weight layers, we always still have the identity *x* to transfer back to earlier layers.

The **1×1 conv layers are added to the start and end of network** as in the figure (right). This is a technique suggested in Network In Network and GoogLeNet (Inception-v1). It turns out that **1×1 conv can reduce the number of connections (parameters) while not degrading the performance of the network so much**

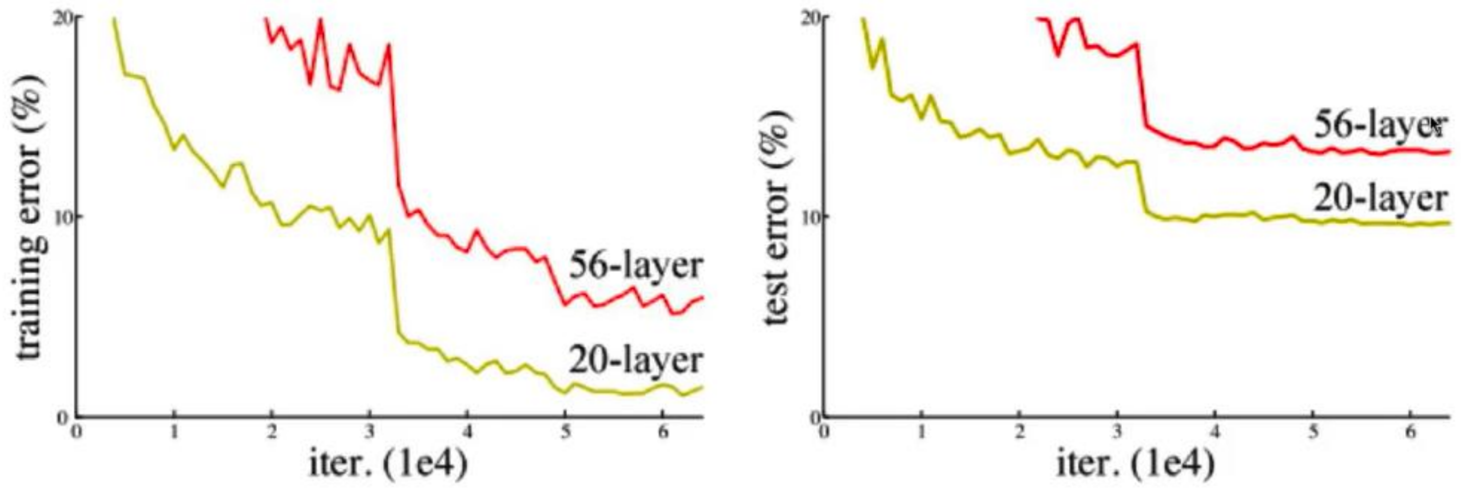
Towards ResNet Architecture

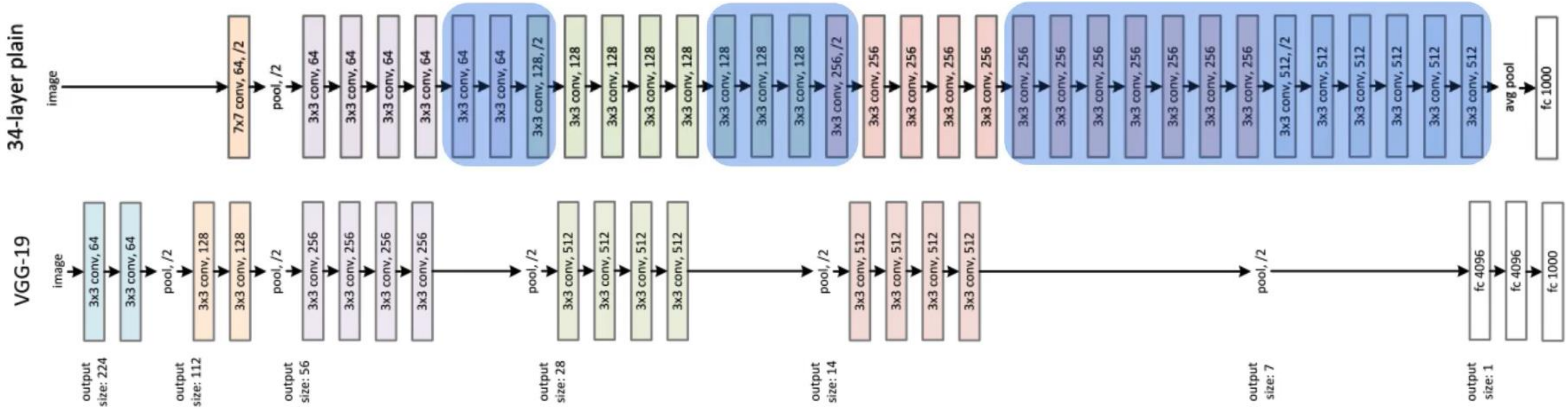
VGG -19 was added with more convolutional layers making it 34 layer, expecting better performance(being deeper) which it did not give as expected :-(. What went wrong??



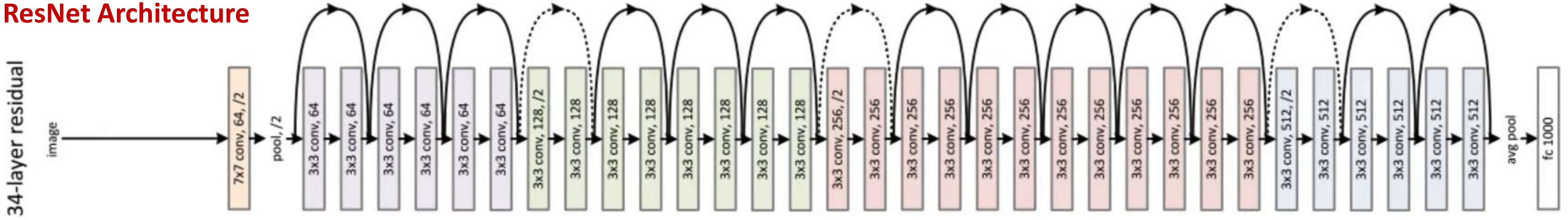
Towards ResNet Architecture

When network grows deeper input is getting transformed significantly and by the time it reaches the last layer, it is messing up the input due to many transformations

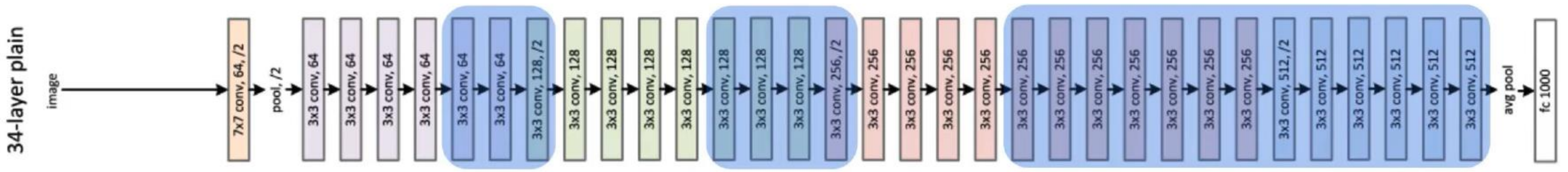




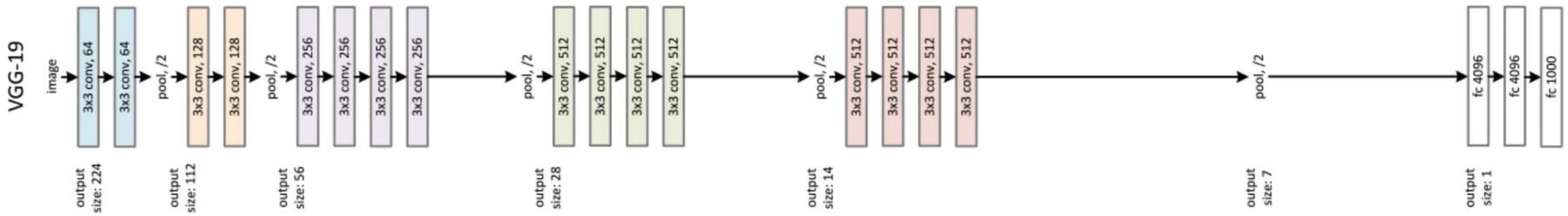
ResNet Architecture



34-layer residual network (ResNet) (top) is the plain one with addition of skip / shortcut connection.

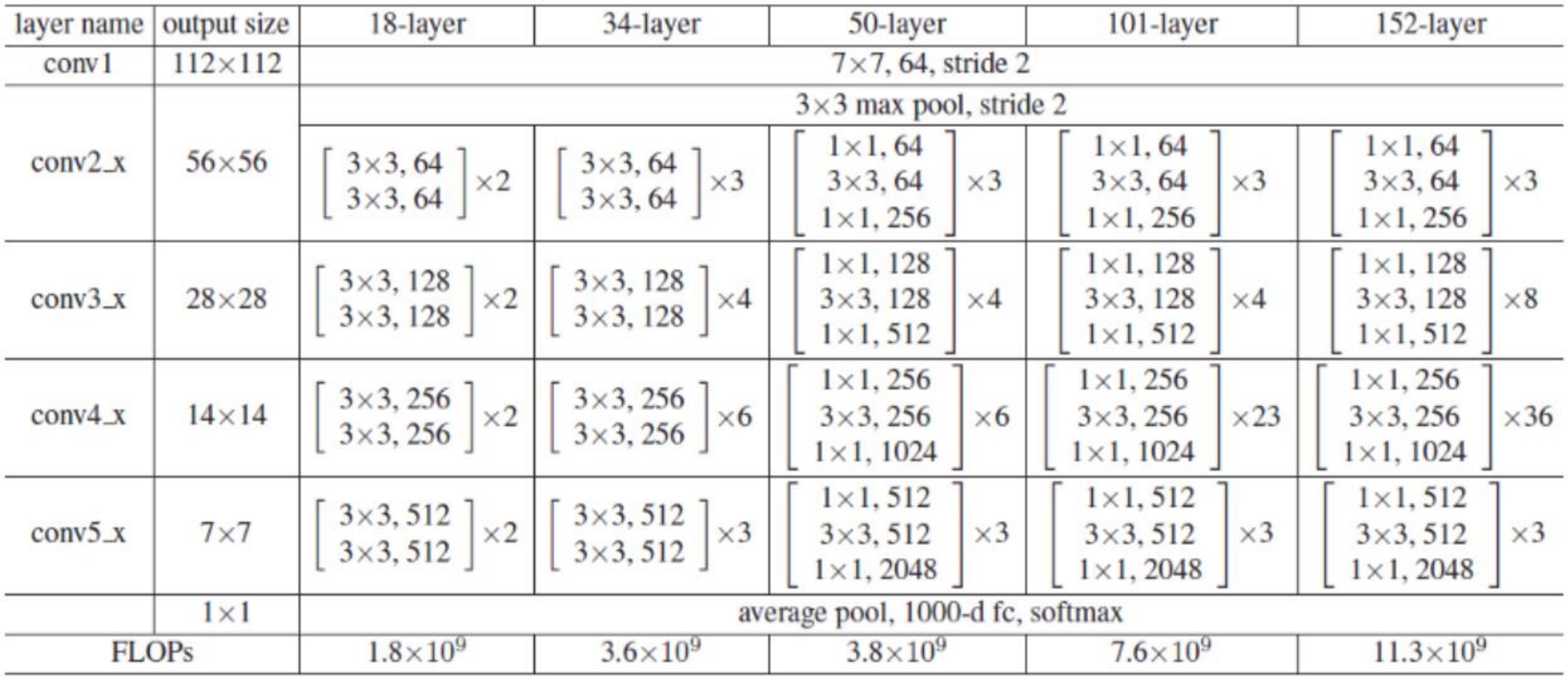


34-layer plain network (middle) is treated as the deeper network of VGG-19, i.e. more conv layers.



The **VGG-19 (bottom)** is a state-of-the-art approach in ILSVRC 2014.

With the bottleneck design, 34-layer ResNet become 50-layer ResNet. And there are **deeper network with the bottleneck design: ResNet-101 and ResNet-152.**

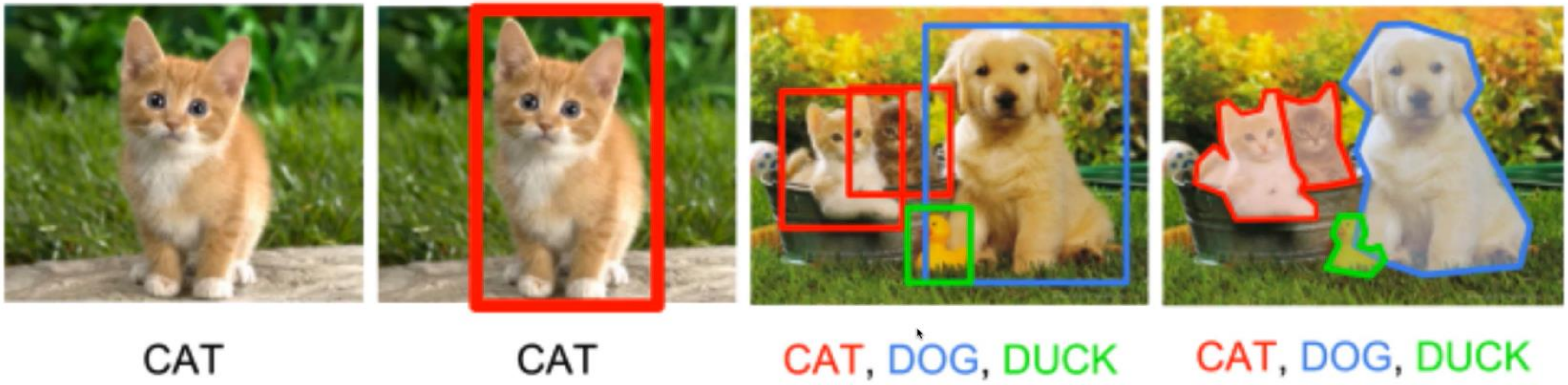


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
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

The overall architecture for all network

It is noted that **VGG-16/19 has 15.3/19.6 billion FLOPS. ResNet-152 still has lower complexity than VGG-16/19!!!!**

ResNet (Residual Network)– How does it perform across tasks



Winner on the 5 main tasks:

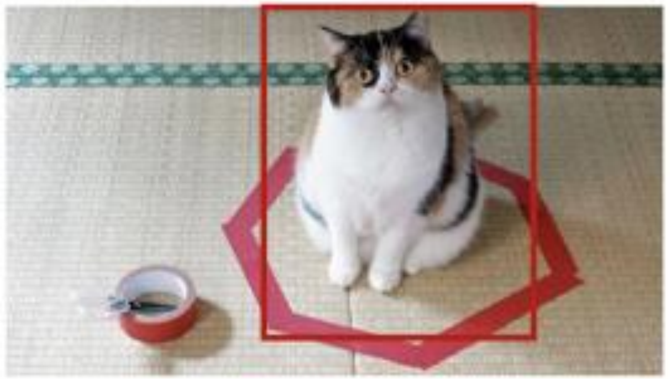
- ✓ ImageNet Classification
- ✓ ImageNet Localization*
- ✓ ImageNet Detection*
- ✓ Coco Detection*
- ✓ Coco Segmentation*

1

Is this image of Cat or not?

Image classification problem

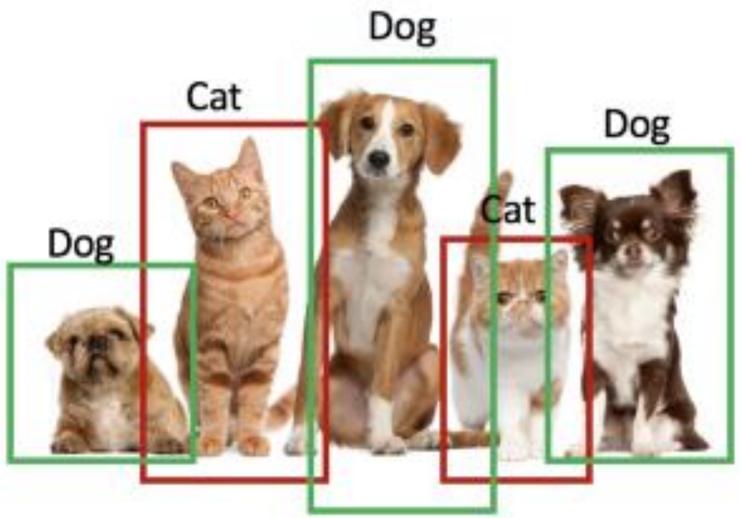
2



Where is Cat?

Classification with localization problem

3



Which animals are there in image and where?

Object detection problem

Image Classification

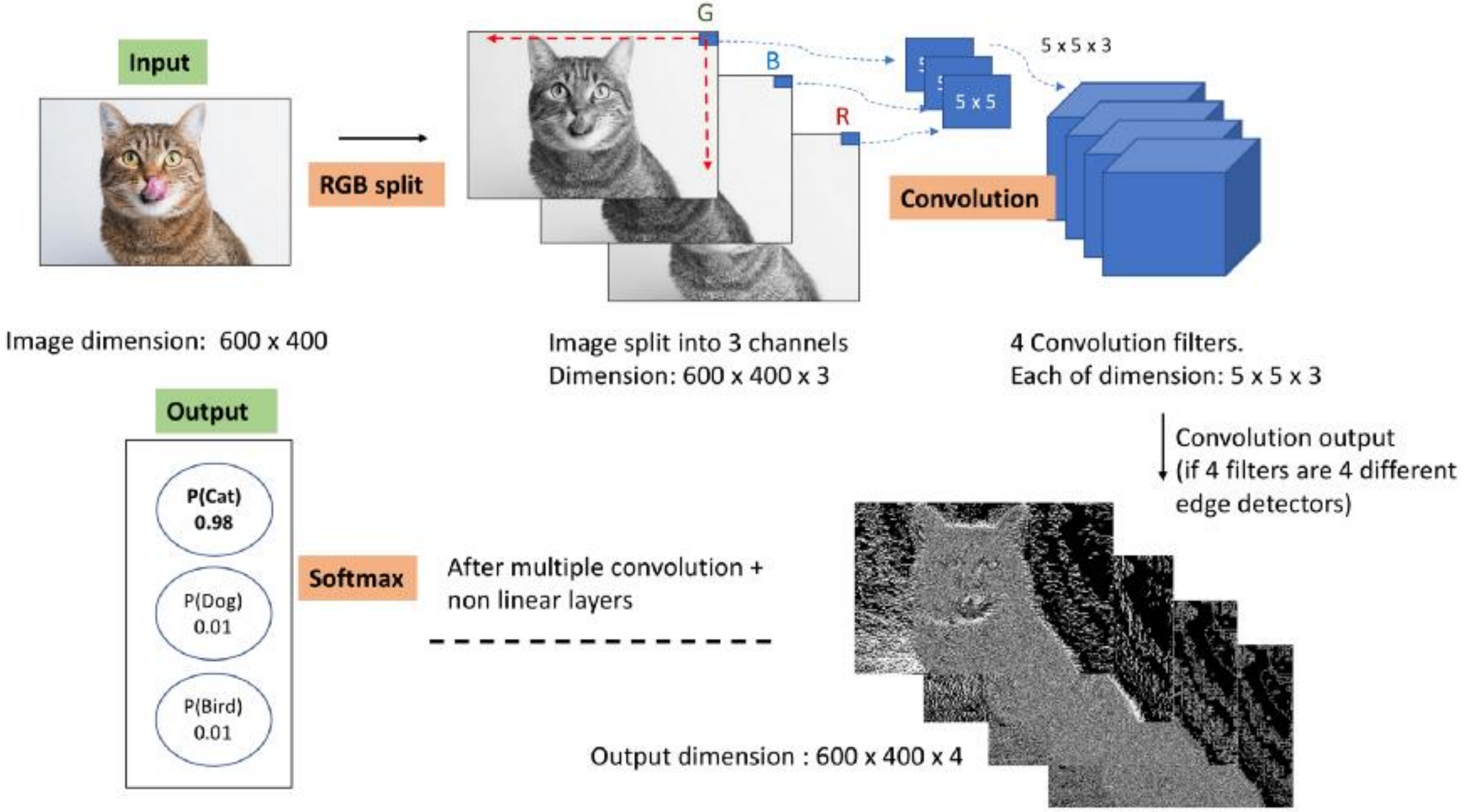
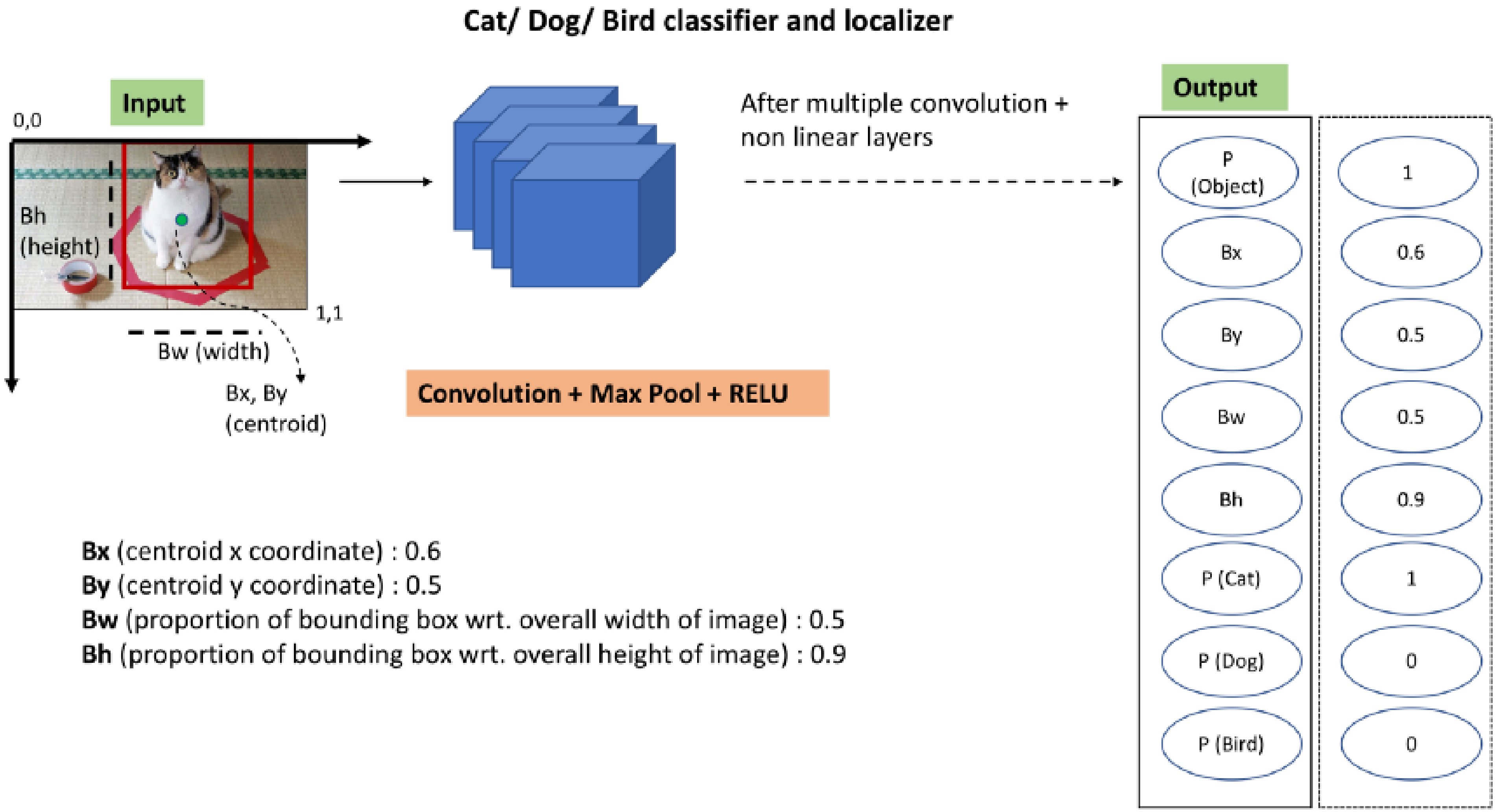
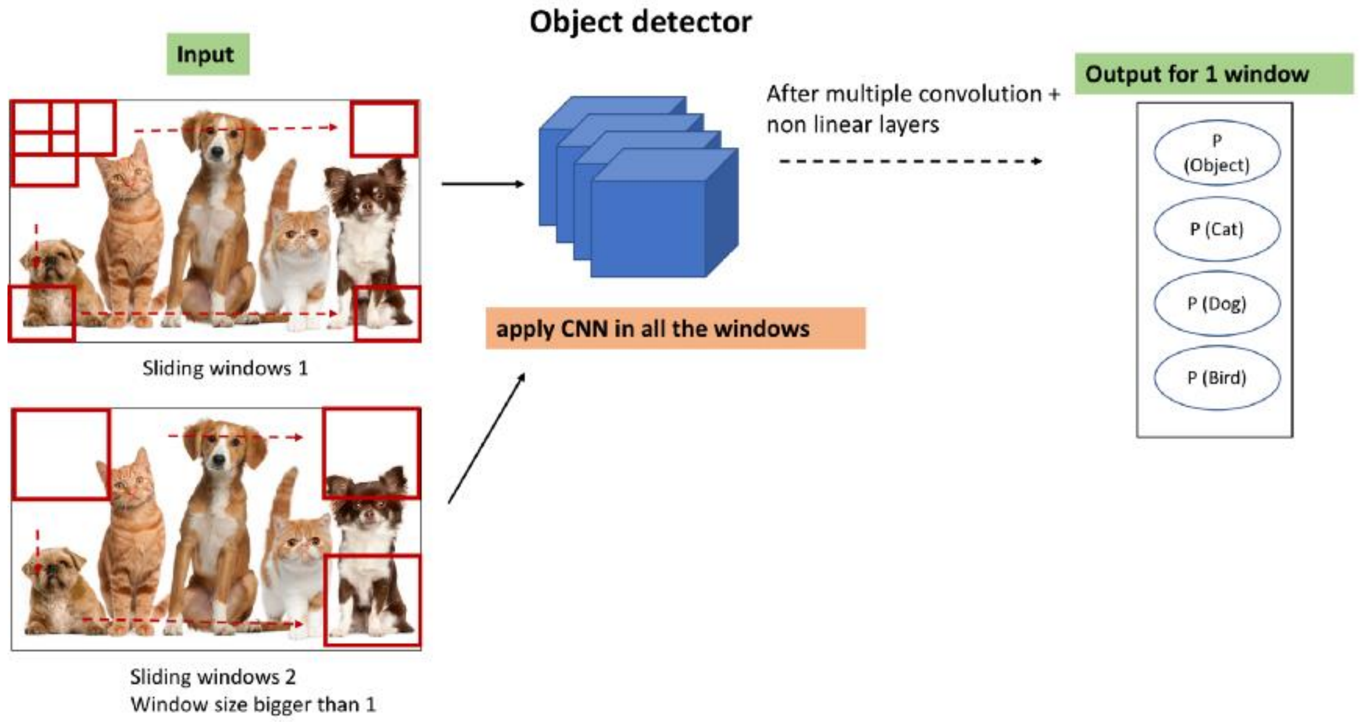


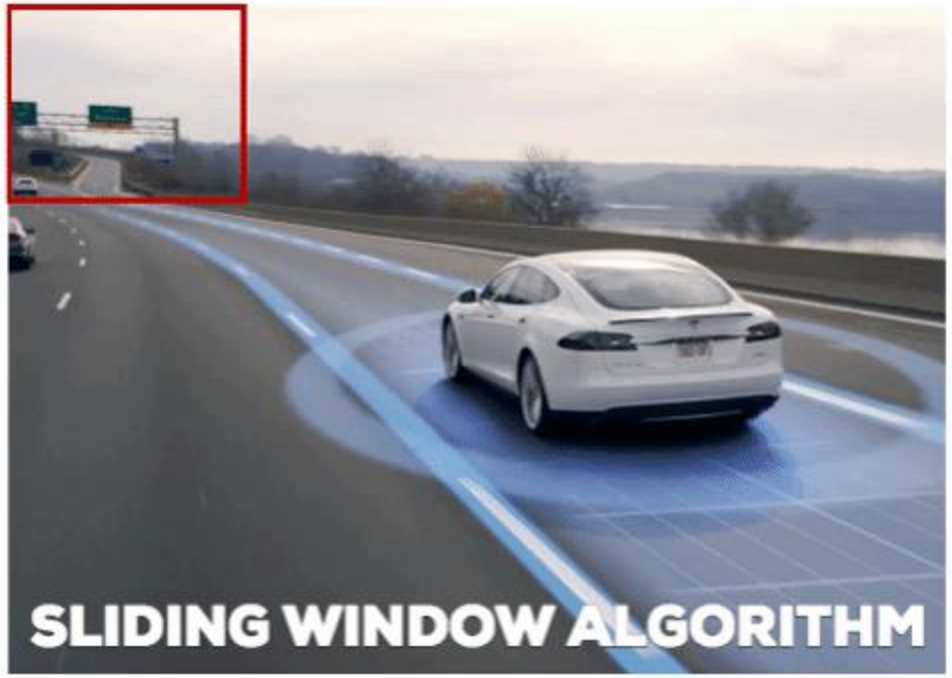
Image Classification with localisation



Bx (centroid x coordinate) : 0.6
By (centroid y coordinate) : 0.5
Bw (proportion of bounding box wrt. overall width of image) : 0.5
Bh (proportion of bounding box wrt. overall height of image) : 0.9

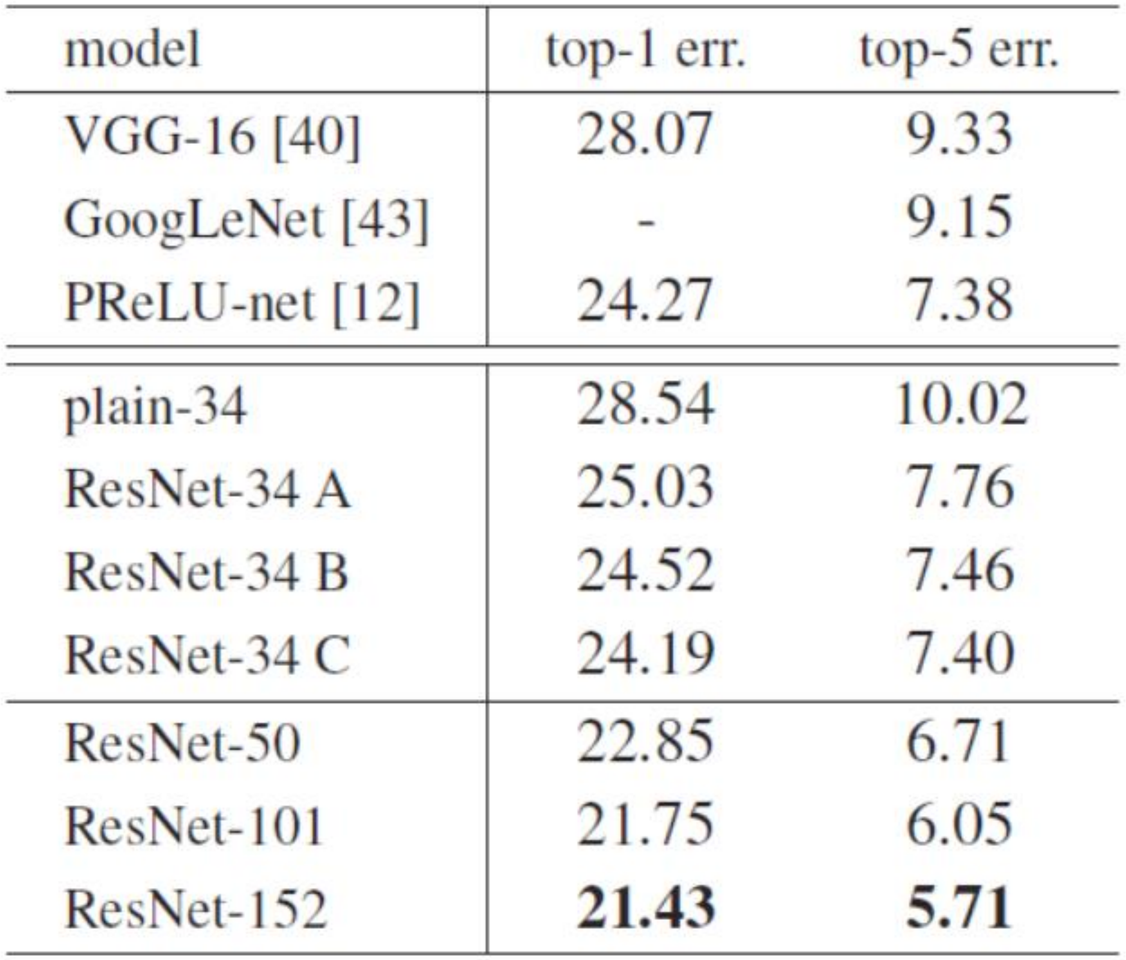
Multiple Object detection





Now we have better solutions- pretrained models for object detection- YOLO, R-CNN Faster RCNN, MaskRCNN. We will see those later

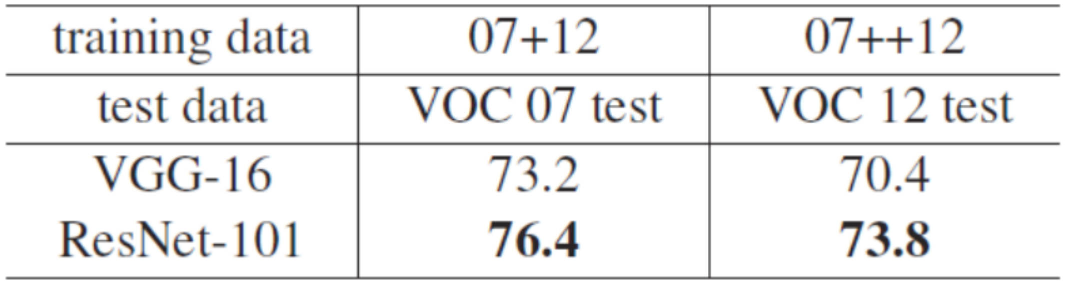
Comparison with State-of-the-art Approaches (Image Classification) ILSVRC



model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

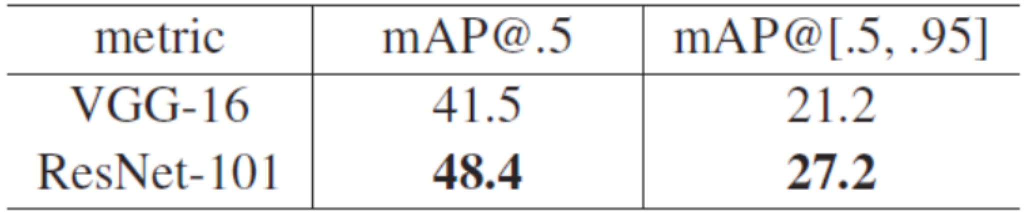
10-Crop Testing Results

Comparison with State-of-the-art Approaches (Object Detection) ILSVRC



training data	07+12	07++12
test data	VOC 07 test	VOC 12 test
VGG-16	73.2	70.4
ResNet-101	**76.4**	**73.8**

PASCAL VOC 2007/2012 mAP (%)



metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	**48.4**	**27.2**

Namah Shivaya