CS 4104
APPLIED MACHINE LEARNING

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

CONVOLUTIONAL NEURAL NETWORK

CNN

- Neural Networks that use convolution in place of general matrix multiplication in at least one layer

- There are three types of layers in the convolutional network,

  - **Convolution layer (Conv)**
  - **Pooling layer (Pool)**
  - **Fully connected layer (FC)**

CNN

$n \times n$ image, $f \times f$ filter

padding p, stride s

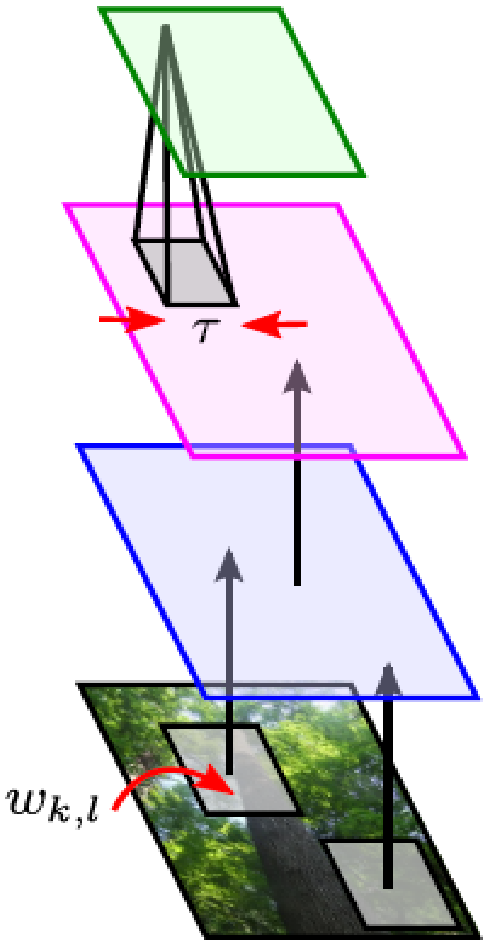
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$$\lfloor z \rfloor = \mathrm{floor}(z)$$

If the fraction is not an integer, then we will get floor of the result.

CNN … STEPS

CNN … Steps



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \mathrm{sigmoid}(a)$

non-linear stage

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

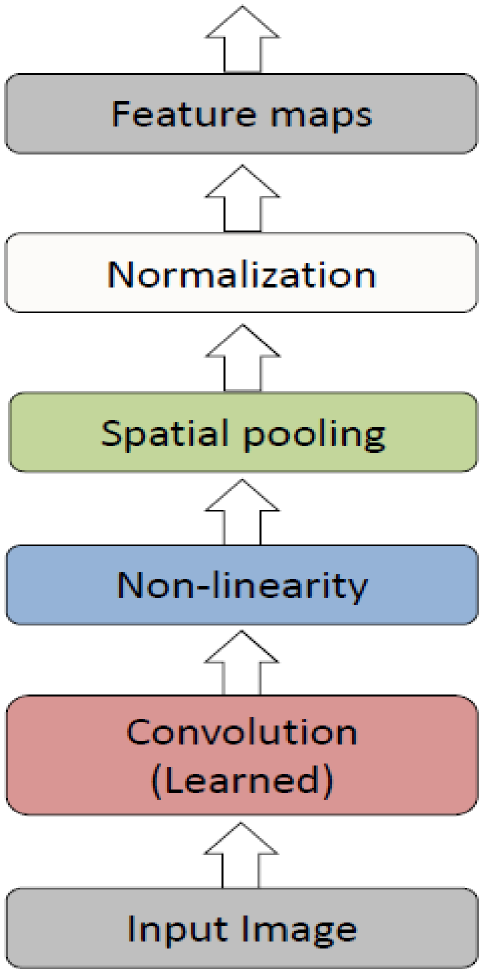
only parameters

convolutional stage

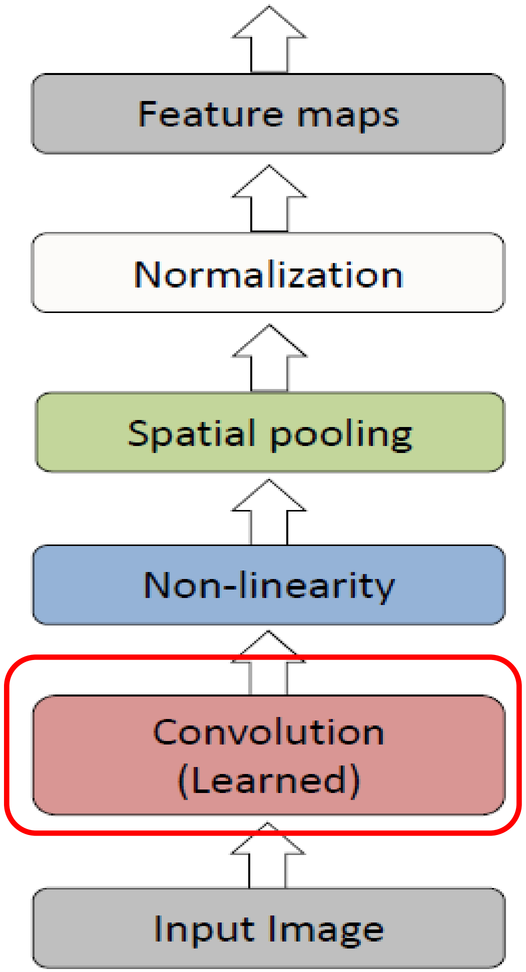
$z_{i,j}$

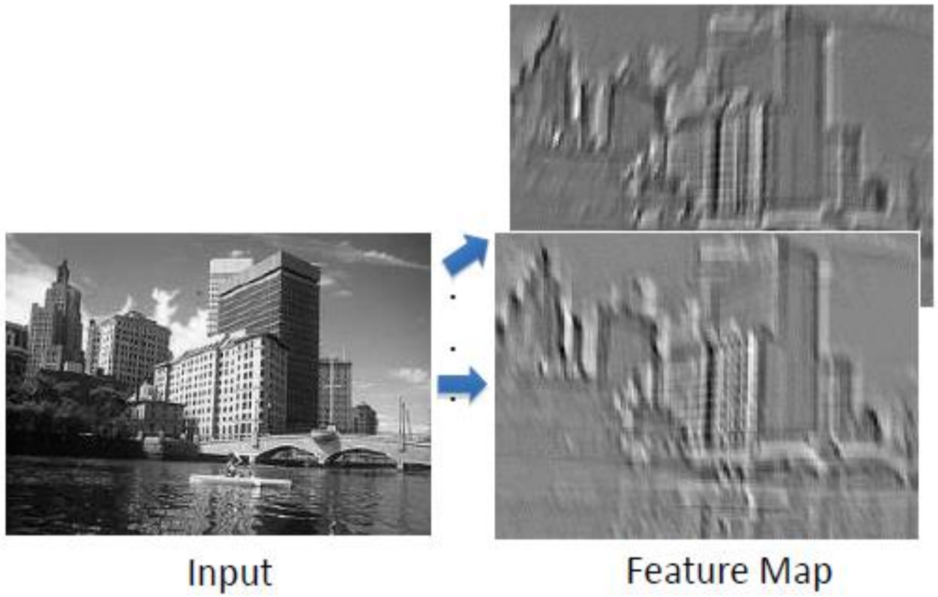
input image

CNN … Steps

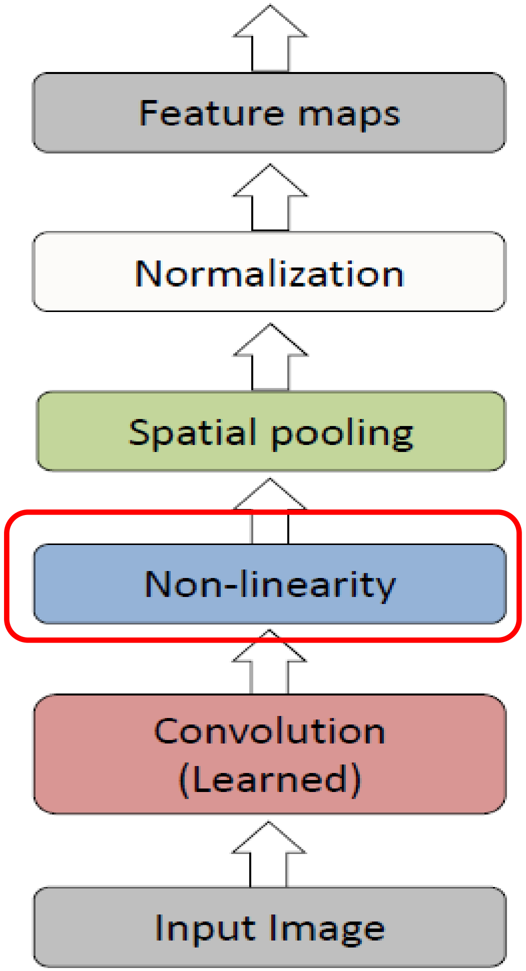

Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution
(Learned)
Input Image

CNN … Steps

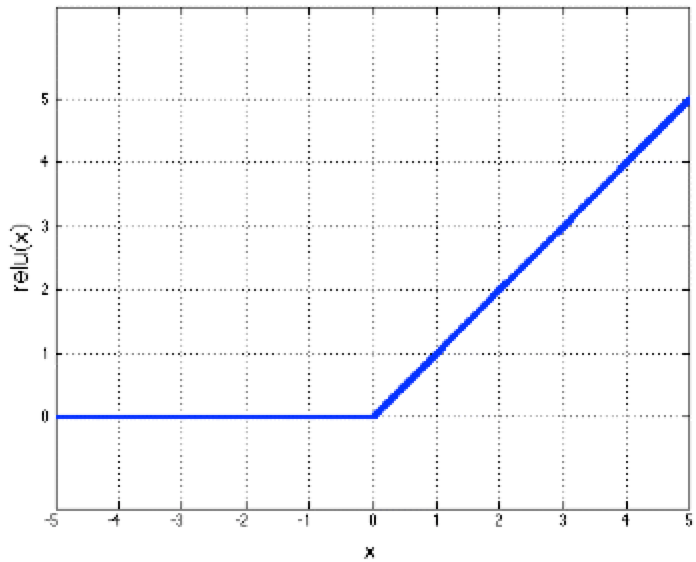

Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution
(Learned)
Input Image


Input
Feature Map

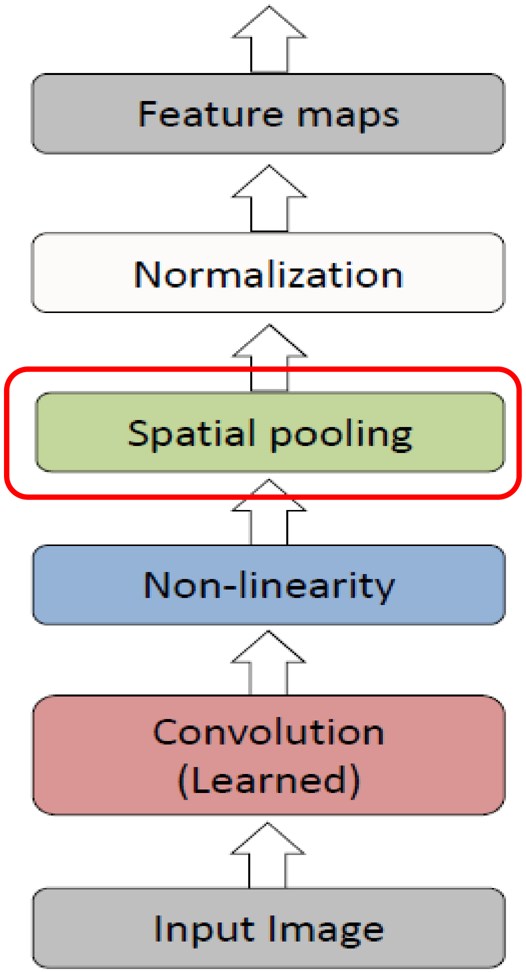
CNN … Steps

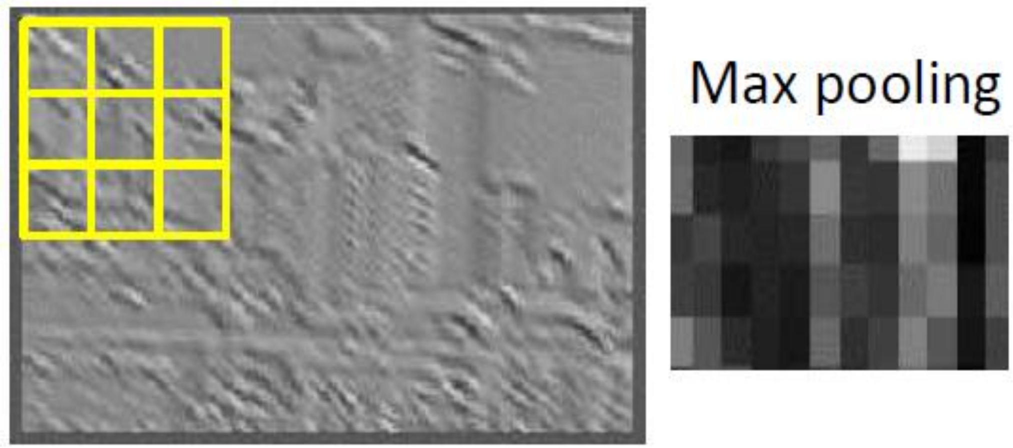


Rectified Linear Unit (ReLU)

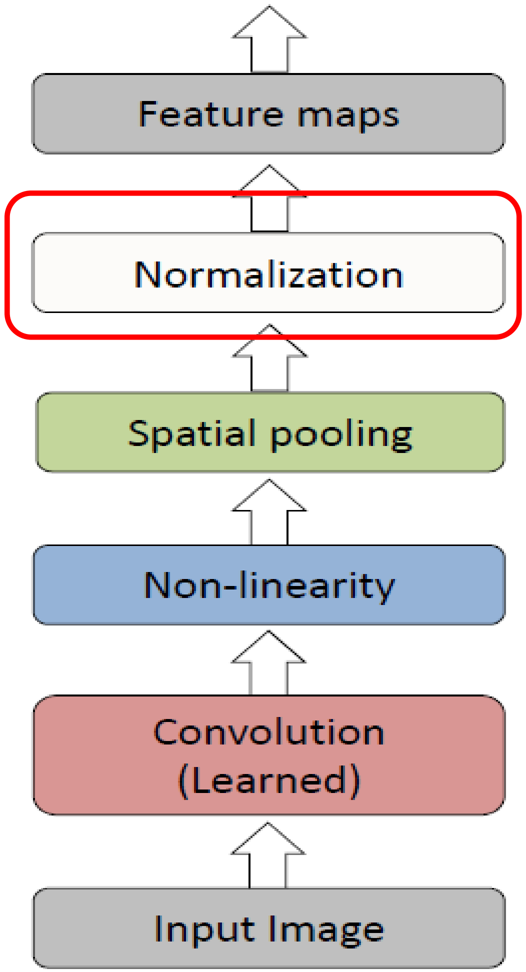


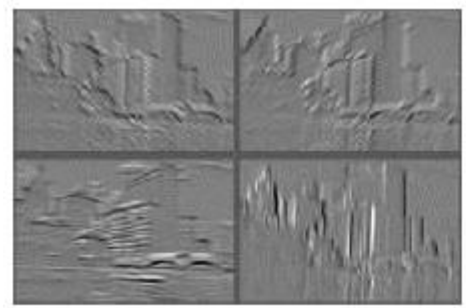
CNN … Steps


Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution
(Learned)
Input Image

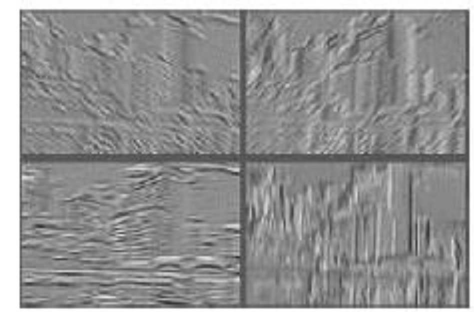

Max pooling

CNN … Steps



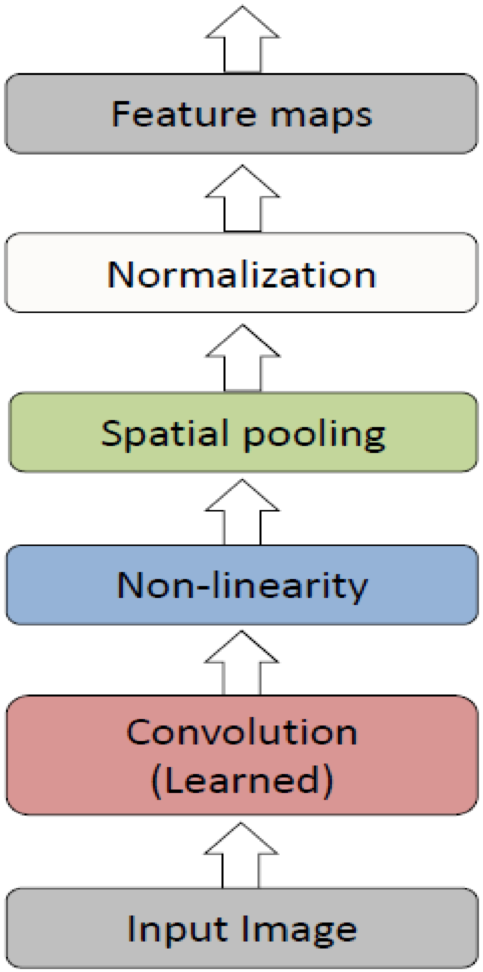


Feature Maps



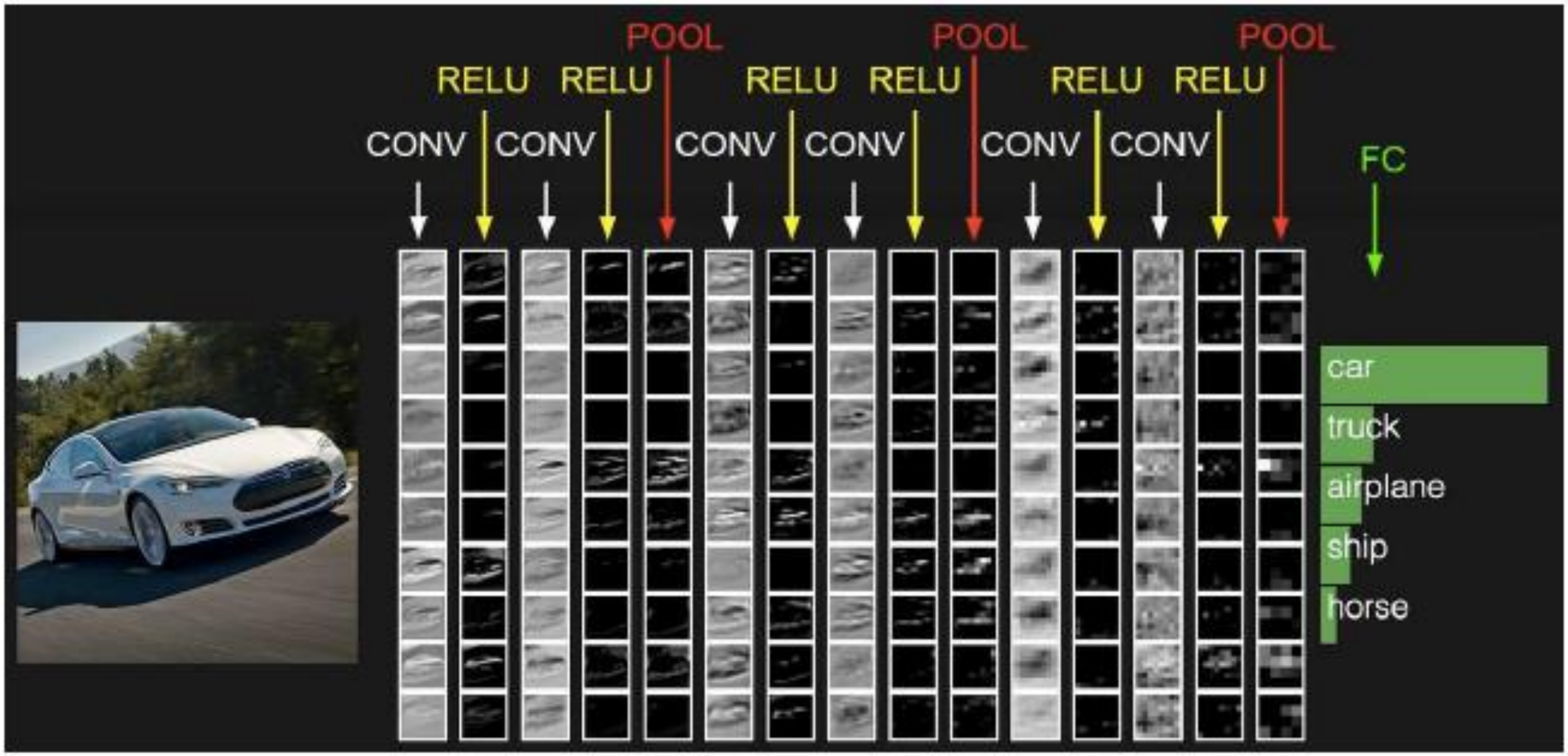
Feature Maps
After Contrast
Normalization

CNN … Steps



Convolutional filters are trained in a supervised manner by back-propagating classification error

CNN … Example


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

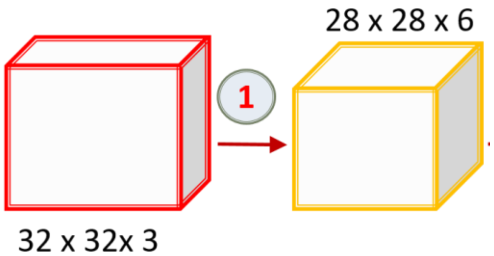
WHY CONVOLUTION?

CNN … Example

Construct the CNN with the following specifications,

- The RGB image of size 32×32 is given as input
- **Layer 1-Conv1:** There are 6 filters of size 5×5, stride $= 1$ and no padding,
- **Layer 1-Pool1:** There are 6 filters of size 2×2, stride $= 2$ and no padding,
- **Layer 2-Conv2:** There are 16 filters of size 5×5, stride $= 1$ and no padding,
- **Layer 2-Pool2:** There are 16 filters of size 2×2, stride $= 2$ and no padding,
- **Layer 3-FC3:** There are 120 neurons in FC3.
- **Layer 4-FC4:** There are 84 neurons in FC4.
- **Output Layer:** with 10 classes.

CNN … Example 3



1 **Layer 1-Conv1:** There are 6 filters of size 5×5, stride = 1 and no padding,

$$\frac{n + 2p - f}{s} + 1$$

$$= \frac{32 + 0 - 5}{1} + 1 = 28$$

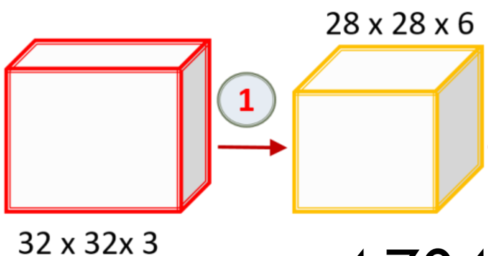
28 x 28 x 6

Why Convolution?

$$\frac{n+2p-f}{s}+1$$
$$=\frac{32+0-5}{1}+1=28$$

28 x 28 x 6

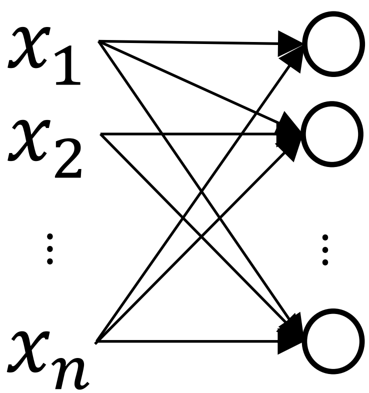
- Consider the same example,



4,704

3,072

1 **Layer 1-Conv1:** There are 6 filters of size 5×5, stride $= 1$ and no padding,



3,072 4,704

No. of Parameters

ANN	CNN
	Filter: $5\times5\times3=75$
$3072\times4{,}704$ =**14 million**	$(5\times5\times3+1)\times6$ = **456**

Why Convolution?

- The number of parameters: $(5\times5\times3+1)\times6 = 456$
- This is based on the following equation,

$$(f^{[l]} \times f^{[l]} \times n_c^{[l-1]} + 1) \times n_c^{[l]}$$

- Where,
    - $(f^{[l]} \times f^{[l]})$ is filter height and width
    - $n_c^{[l-1]}$ is the number of channels in the previous layer
    - $n_c^{[l]}$ is the number of channels in the current layer
    - "1" is the bias term.

Why Convolution?

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 x 6

*

1	0	-1
1	0	-1
1	0	-1

3 x 3

=

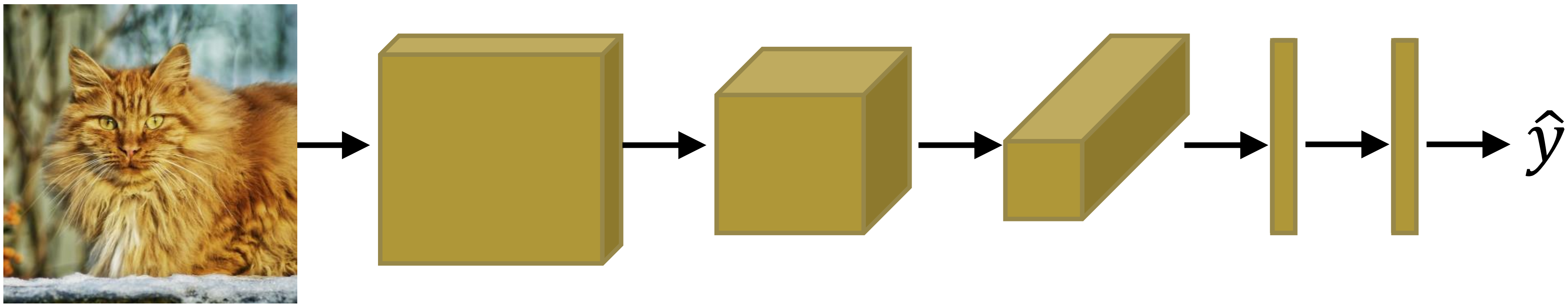
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4 x 4

- **Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.
- **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Putting all together

Training set: $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.



Cost: $J = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

- Use gradient descent to optimize parameters to reduce J

CLASSIC NETWORKS

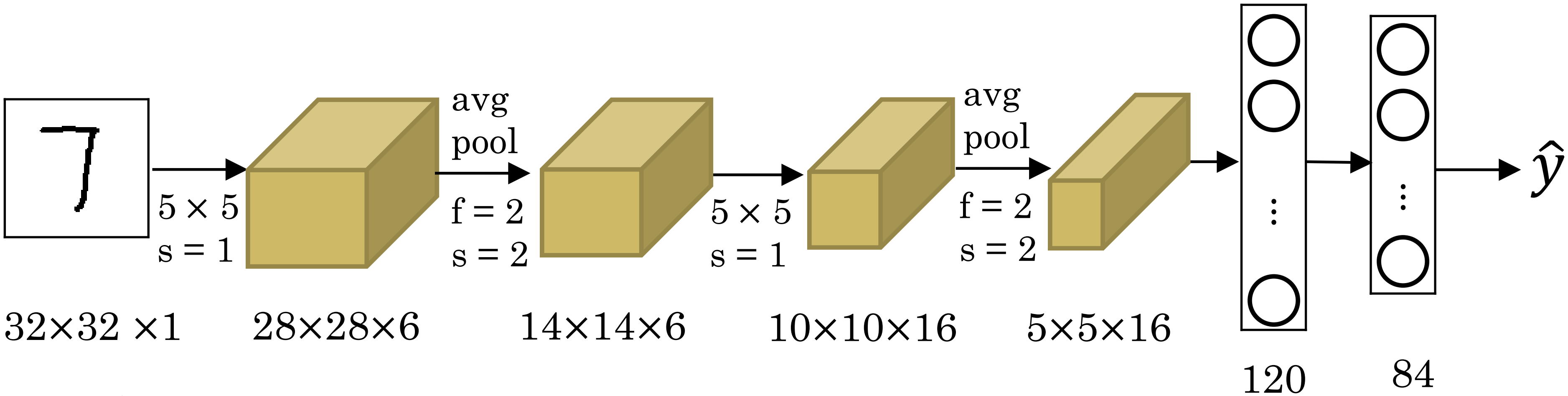
Classic Networks

Some Classic Networks:

- LeNet-5
- AlexNet
- VGG
- ResNet

LENET-5

LeNet-5



60K parameters

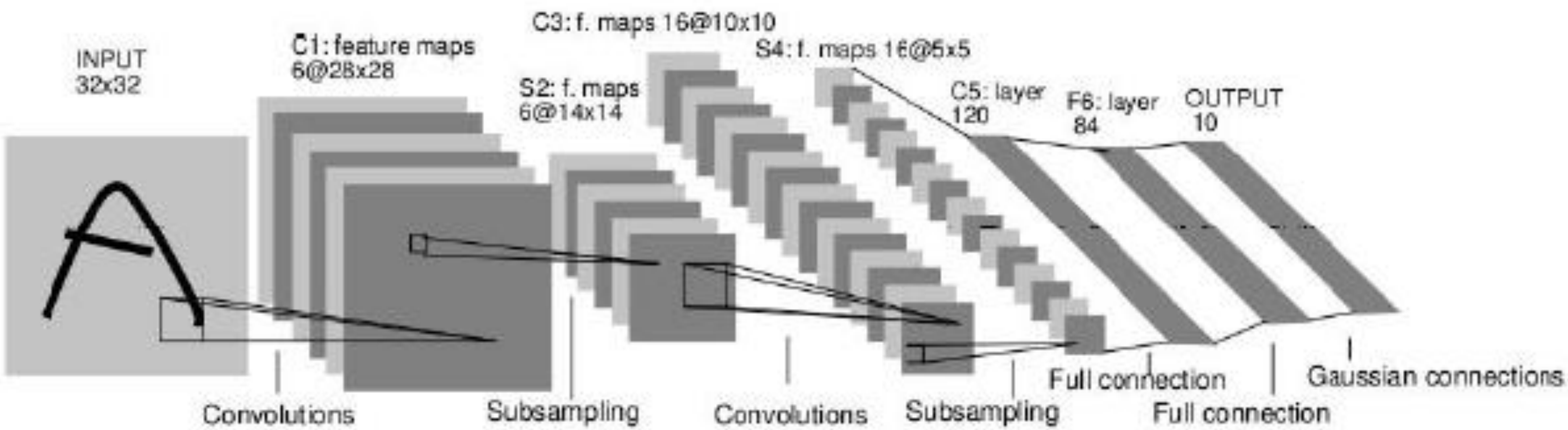
The size of n_H and n_W are decreasing.

The size of n_c is increasing.

For Linearity, they used sigmod/tanh and used after the pooling layer (Not recommended in modern age)

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

LeNet-5

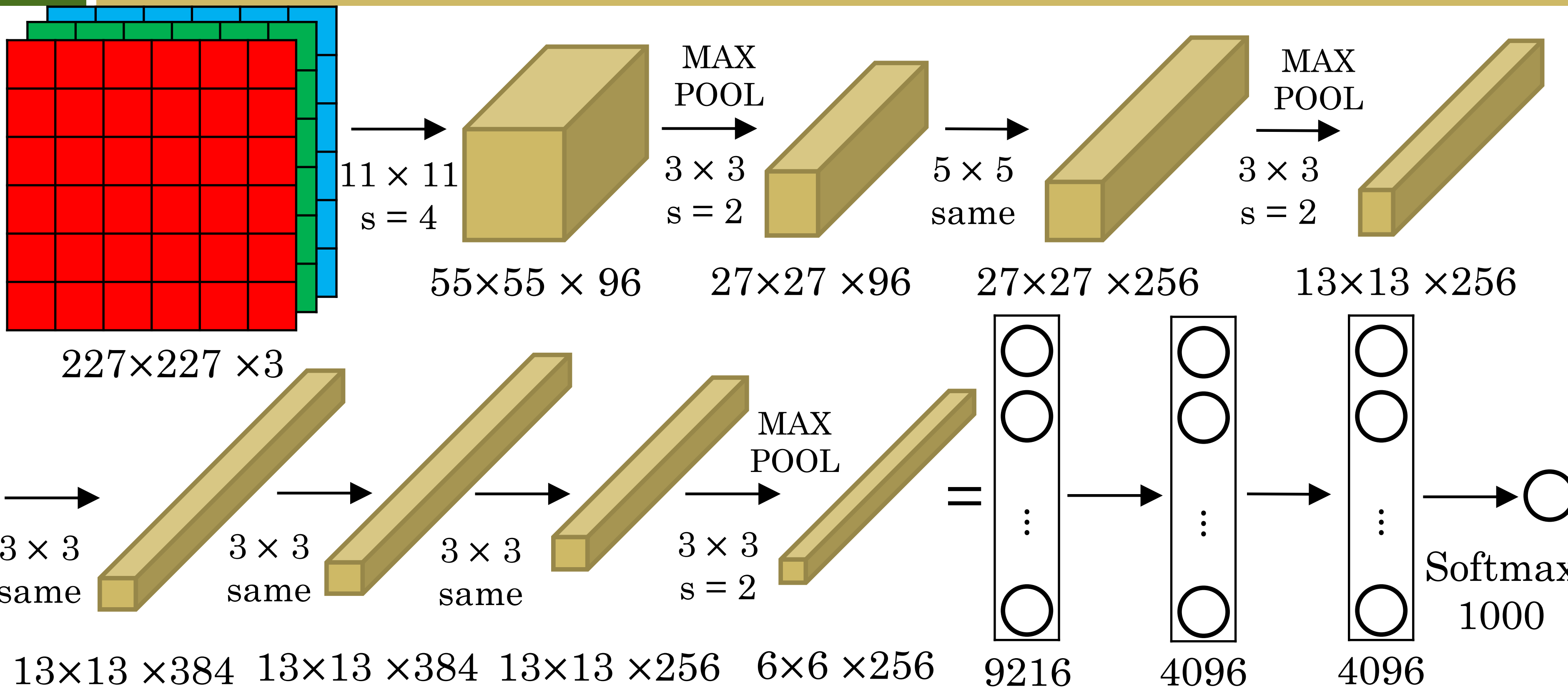


- Filters are of size 5×5, stride 1
- Pooling is 2×2, with stride 2

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

ALEXNET

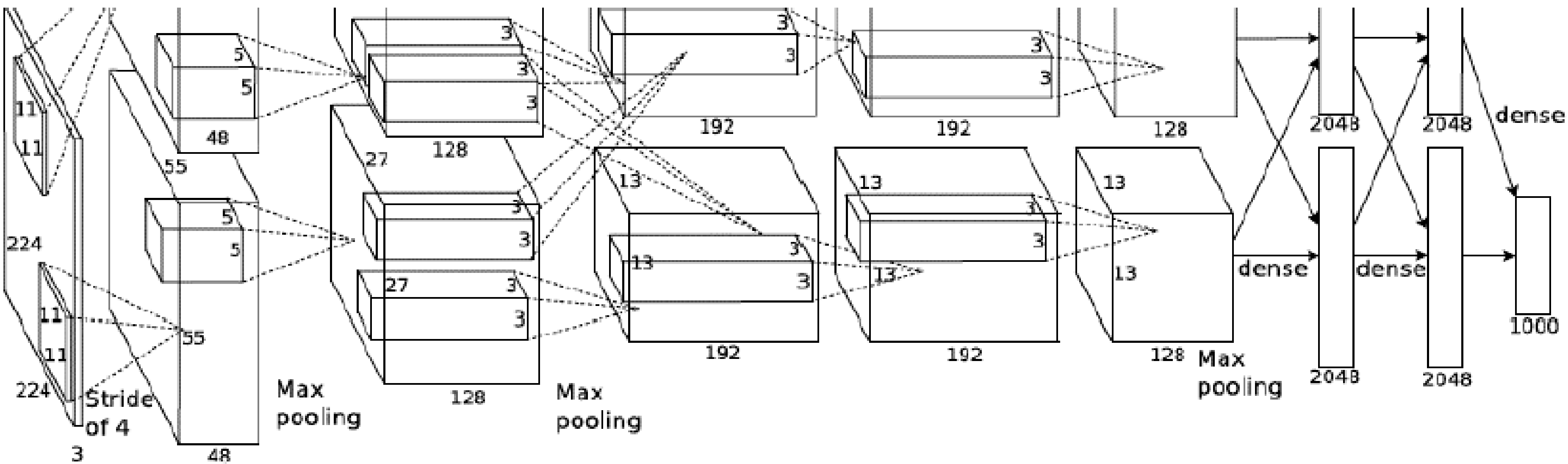
AlexNet



60M parameters

For Linearity, they used ReLU

AlexNet



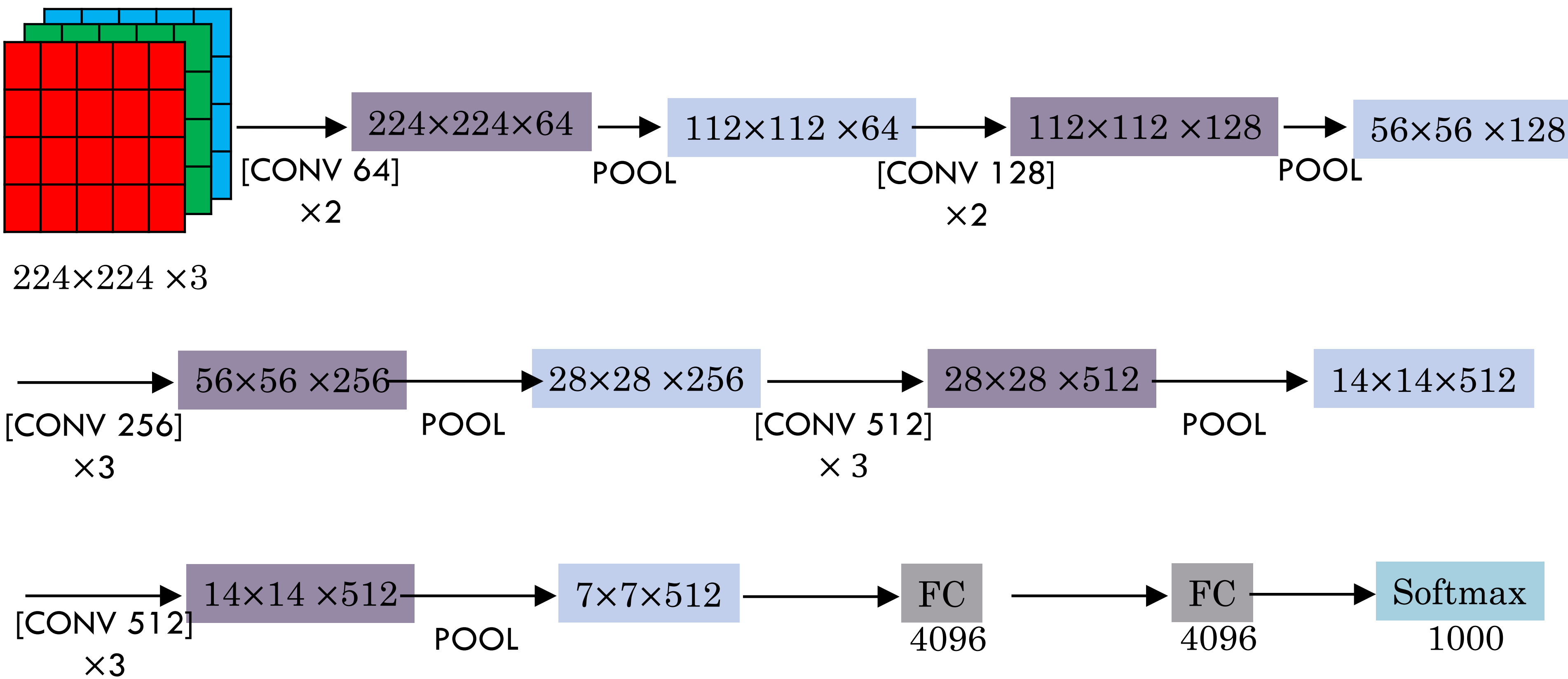
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

VGG: VISUAL GEOMETRY GROUP

VGG-16

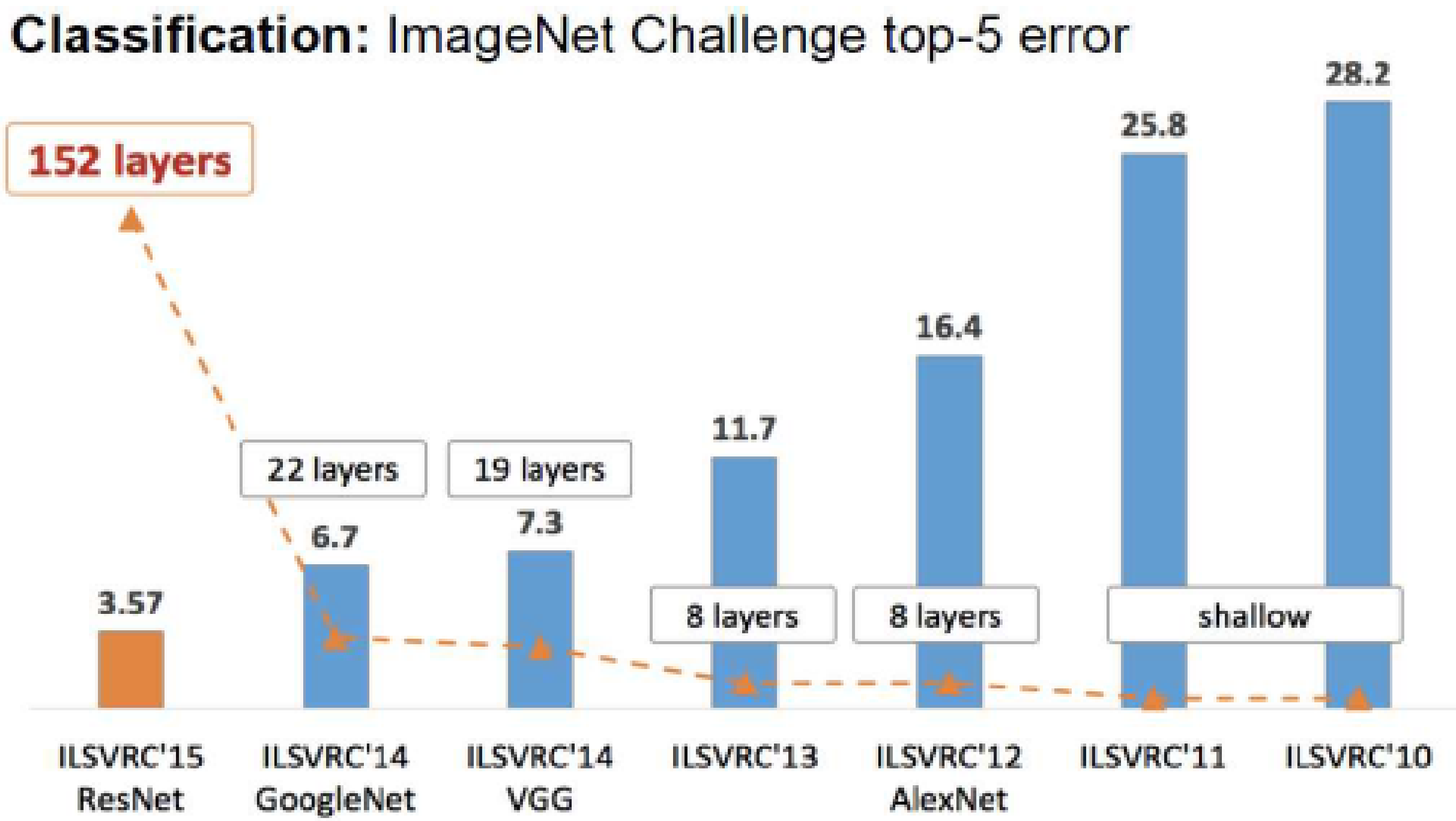
CONV = 3×3 filter, s = 1, same

MAX-POOL = 2×2 , s = 2



~138M parameters

ImageNet Challenge



Acknowledgements

Stuart J. Russell and Peter Norvig, Tom M. Mitchell, Jiwon Jeong, Floydhub, Andrej Karpathy