

Machine Learning & Deep Learning

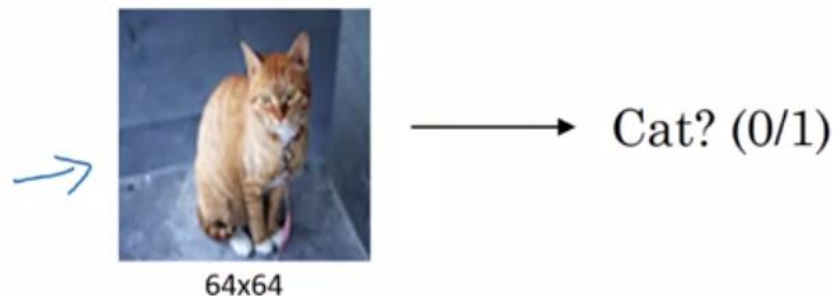
Week-22_23

Instructor: *Engr. Najam Aziz*

National Center for Big Data & Cloud Computing (NCBC), UET Peshawar

Computer Vision Problems

Image Classification



Object detection



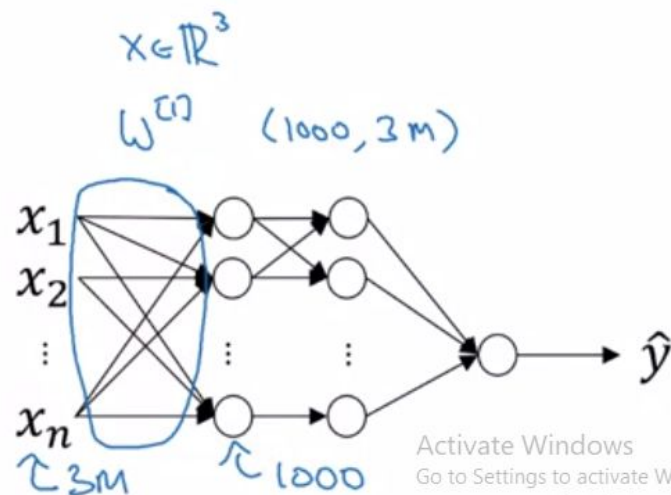
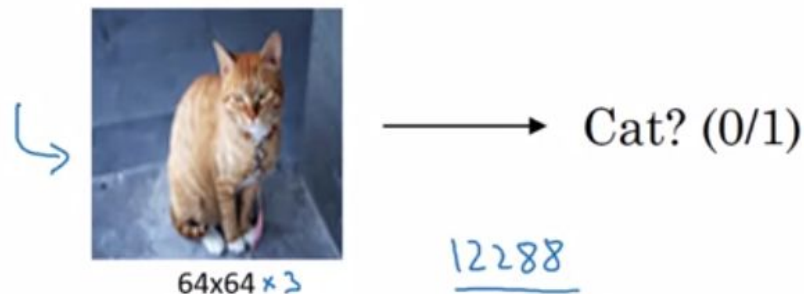
Neural Style Transfer



Activate Windows
Go to Settings to activate Windows.

Andrew Ng

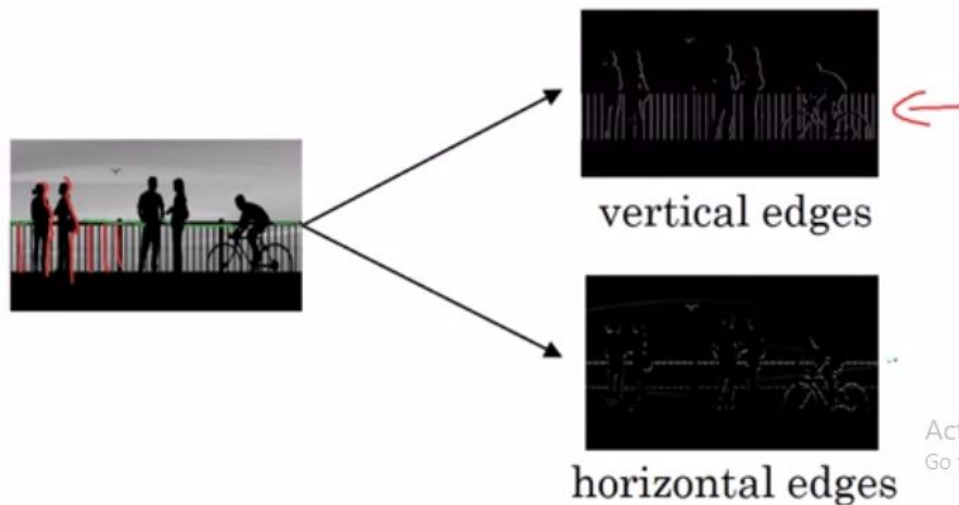
Deep Learning on large images



Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Computer Vision Problem



Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

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

6x6

"convolution"
↓
*

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

3x3
filter

=

-5			

4x4

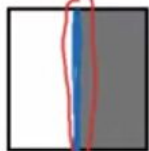
Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0

6x6



*

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

3x3

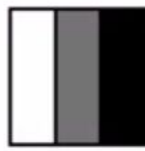
=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4x4



*



↑ ↑ ↑

Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0




0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10




*

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




=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0




*

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



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

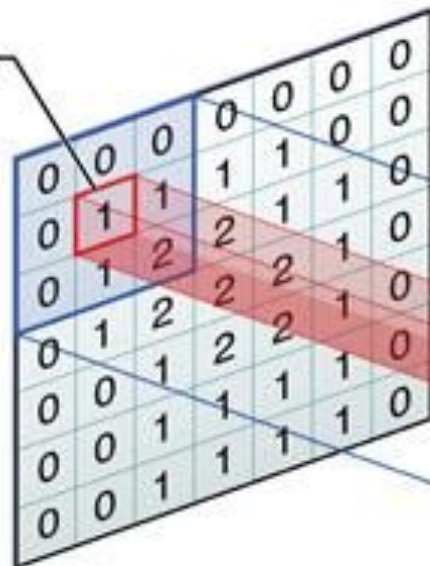


Activate Windows
Go to Settings to activate Windows.

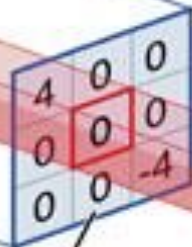
Andrew Ng

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

Source pixel

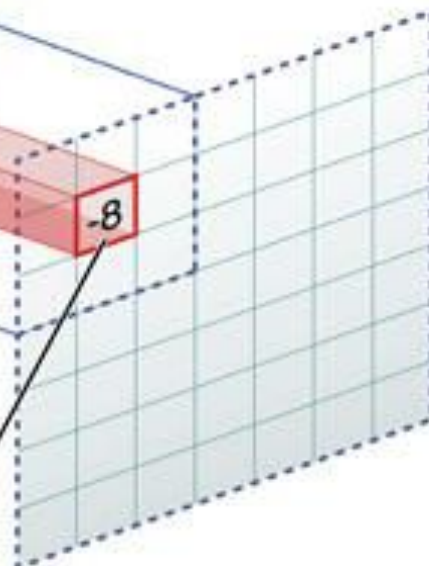


Convolution



New pixel value (destination pixel)

$$\begin{array}{r}
 (4 \times 0) \\
 (0 \times 0) \\
 (0 \times 0) \\
 (0 \times 0) \\
 (0 \times 1) \\
 (0 \times 1) \\
 (0 \times 0) \\
 (0 \times 1) \\
 (0 \times 1) \\
 + (-4 \times 2) \\
 \hline
 -8
 \end{array}$$



Filters/Kernel

- Every layer of filters is there to capture patterns. For example, the first layer of filters captures patterns like edges, corners, dots etc. Subsequent layers combine those patterns to make bigger patterns (like combining edges to make squares, circles, etc.).
- Now as we move forward in the layers, the patterns get more complex; hence there are larger combinations of patterns to capture. That's why we increase the filter size in subsequent layers to capture as many combinations as possible.

Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

6x6

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Learning to detect edges

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



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

→

1	0	-1
2	0	-2
1	0	-1

Sobel filter



convolution
*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3x3

=

45°
70°
73°

3	0	-3
10	0	-10
3	0	-3

Scharr filter



Activate Windows

Go to Settings to activate Windows.

Andrew Ng

Padding

Downsides of Convolution:

- Image shrinks, but sometime we don't want to shrink it after every convolution or step. Bcoz in a very deep NN, the image will shrink to very small image.
- The pixels on the corners or edges use much less in the output than the pixels at center. Which means throwing away lot of information near the edges of an image.

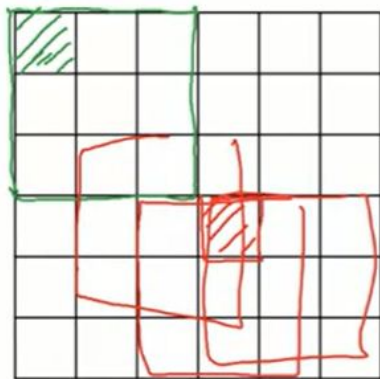
To solve both these problem, we need Padding.

In order to build deep neural network one modification to the basic convolutional operation that we need to really use, is padding. We pad the image before convolutional operation.

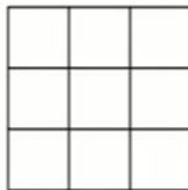
Downside of Convolution

Padding

- shrinky output
- throw away info from edge



*



=

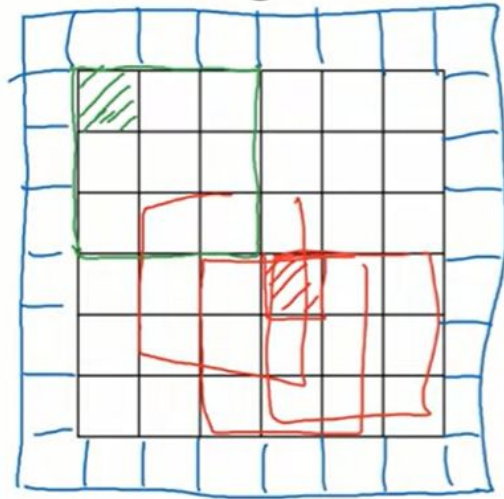
3×3
 $f \times f$

$\frac{6 \times 6}{n \times n}$

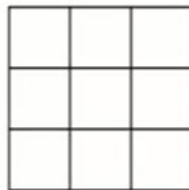
$n - f + 1 \times n - f + 1$
 $6 - 3 + 1 = 4$

$\longrightarrow \underline{\underline{4 \times 4}}$

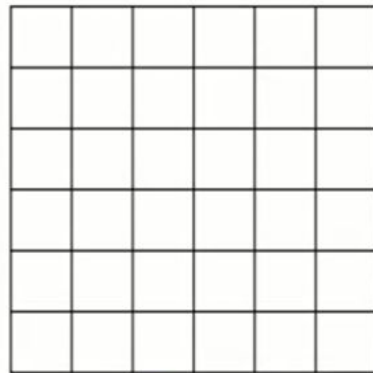
Padding



- shunt output
- throw away info from edge


$$\begin{array}{c} 3 \times 3 \\ f \times f \end{array}$$

==

 6×6

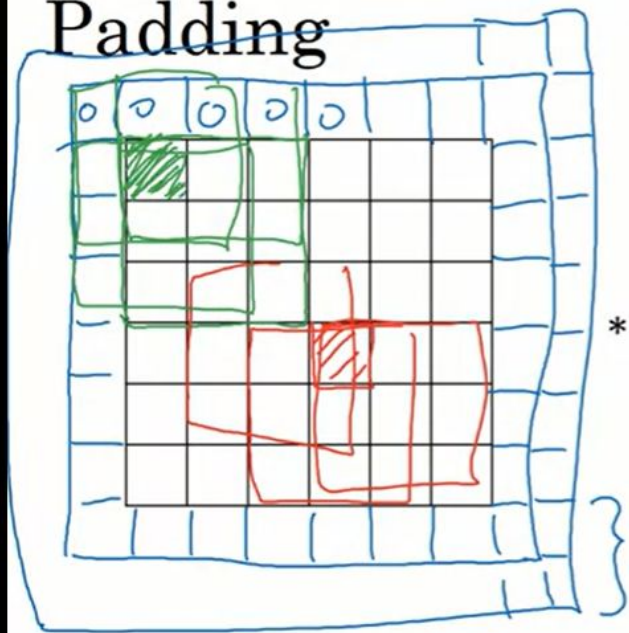
$$\begin{array}{c} 6 \times 6 \rightarrow 8 \times 8 \\ \hline n \times n \end{array}$$

$$n-f+1 \times n-f+1$$
$$6-3+1=4$$

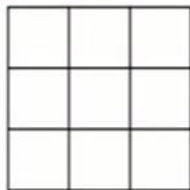
→ 4x4

Padding

- shrinky output
- throw away info from edge

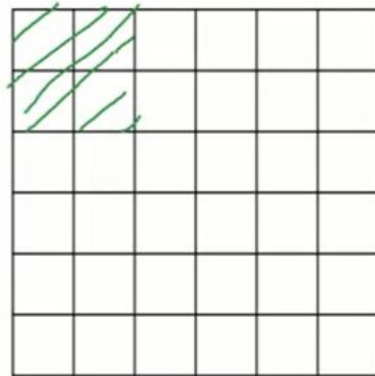


*



3x3
f x f

=



6x6

6x6 \rightarrow 8x8
 $n \times n$

$n - f + 1 \times n - f + 1$
 $6 - 3 + 1 = 4$

$p = \text{padding} = \underline{1}$

$n + 2p - f + 1 \times n + 2p - f + 1$
 $6 + 2 - 3 + 1 \times \underline{\underline{4}} = 6 \times 6$

Valid and Same convolutions

→ no padding

"Valid": $n \times n \quad * \quad f \times f \quad \rightarrow \quad \underline{n - f + 1} \times n - f + 1$
 $6 \times 6 \quad * \quad 3 \times 3 \quad \rightarrow \quad 4 \times 4$

"Same": Pad so that output size is the same as the input size.

$$n + 2p - f + 1 \times n + 2p - f + 1$$
$$\cancel{n + 2p - f + 1} = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$$

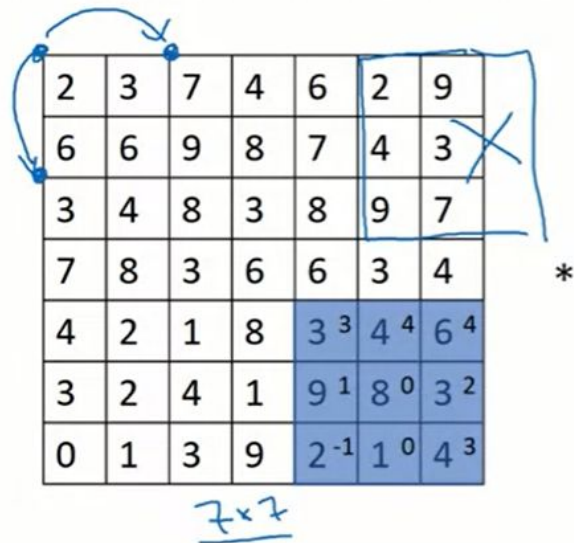
$$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad \left| \quad \begin{matrix} 5 \times 5 \\ f=5 \end{matrix} \right. \quad p=2$$

f is usually odd



1x1
3x3
5x5
7x7
.

Strided convolution



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

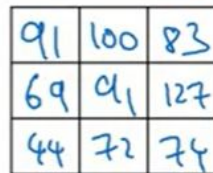
7x7

3	4	4
1	0	2
-1	0	3

3x3

stride = 2

=



91	100	83
69	91	127
44	72	74

3x3

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

$n \times n$ * $f \times f$
 padding p stride s
 $s=2$

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

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

Summary of convolutions

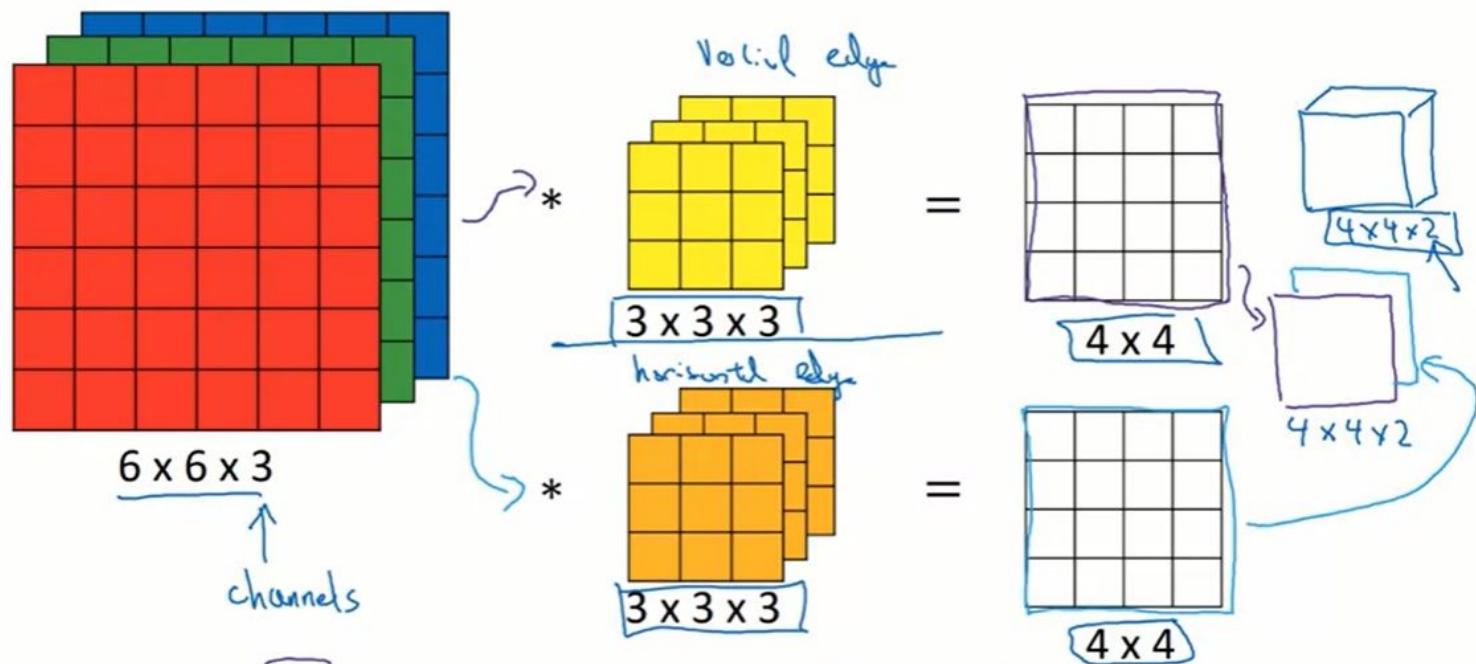
$n \times n$ image $f \times f$ filter

padding p stride s

Output Size:

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

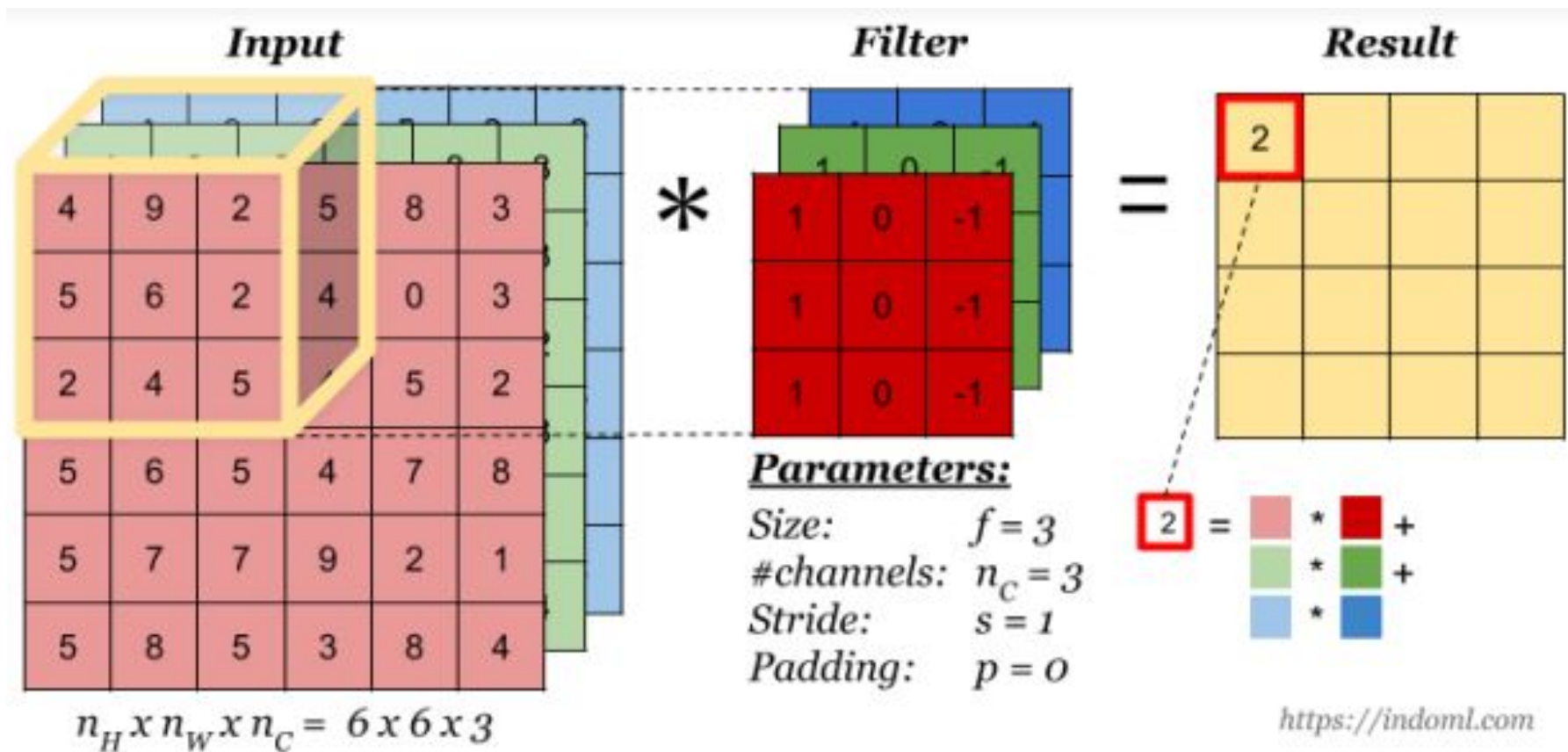
Multiple filters



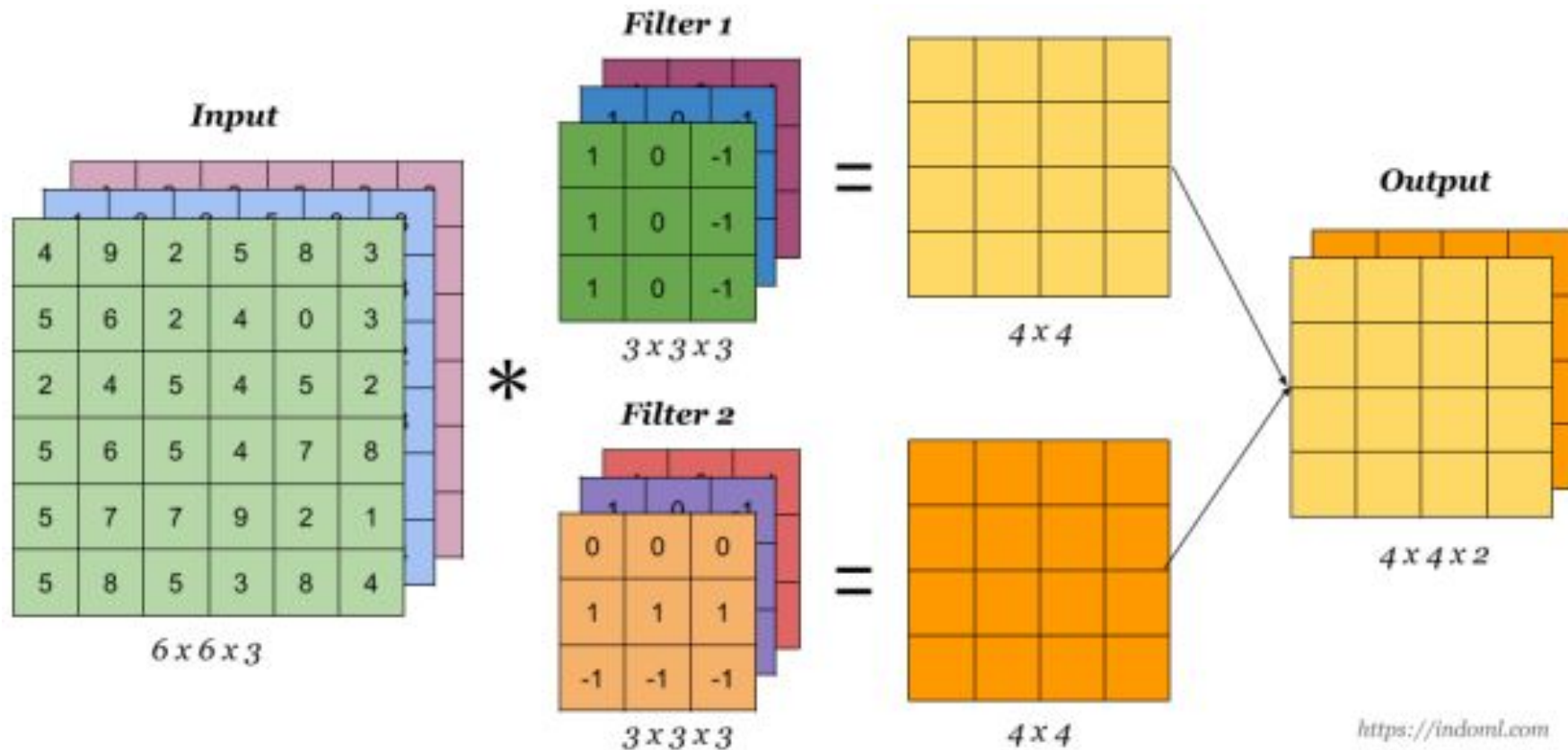
Summary: $n \times n \times n_c \times f \times f \times n_c \rightarrow \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times \frac{n_c}{2}$

$6 \times 6 \times 3 \times 3 \times 3 \times 3 \rightarrow 4 \times 4 \times 2$ # filters

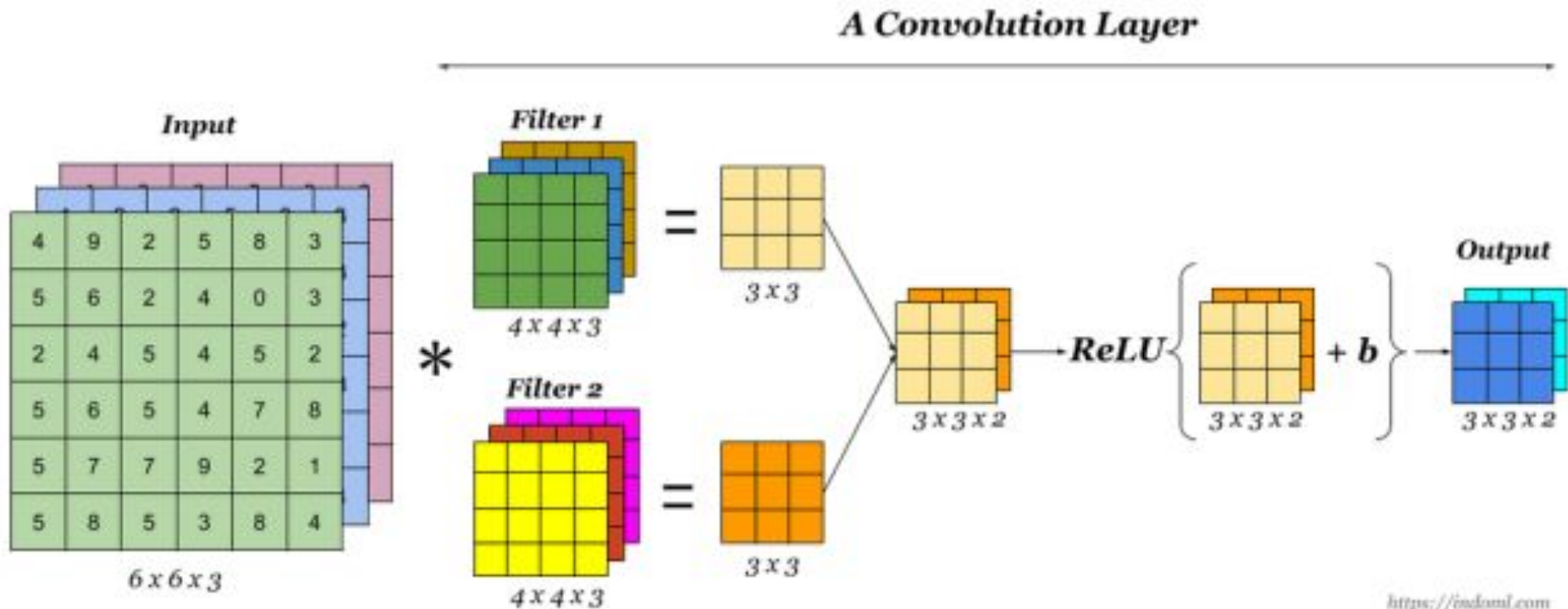
Convolution on RGB images



Convolution with multiple filters



A Convolutional Layer



Limitation of the feature map output of convolutional layers

- A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.
- A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task.
- Down sampling can be achieved with convolutional layers by changing the stride of the convolution across the image. A more robust and common approach is to use a pooling layer.

Pooling

- A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer; for example the layers in a model may look as follows:

Input Image - Convolutional Layer - Nonlinearity - Pooling Layer

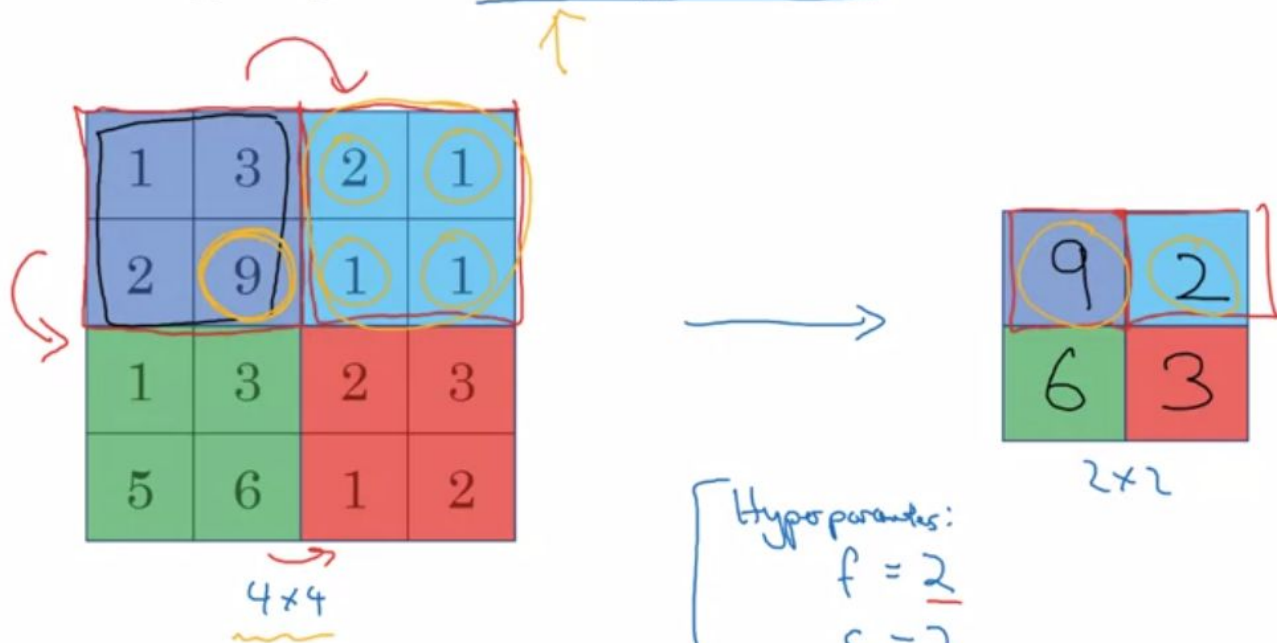
- The pooling operation is specified, rather than learned. (Hence have No learnable parameters). It creates feature maps that summarize the presence of those features in the input. Two common functions used in the pooling operation are:

Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

Average Pooling: Calculate the average value for each patch on the feature map.

The result of using a pooling layer and creating down-sampled or pooled feature maps is a summarized version of the features detected in the input.

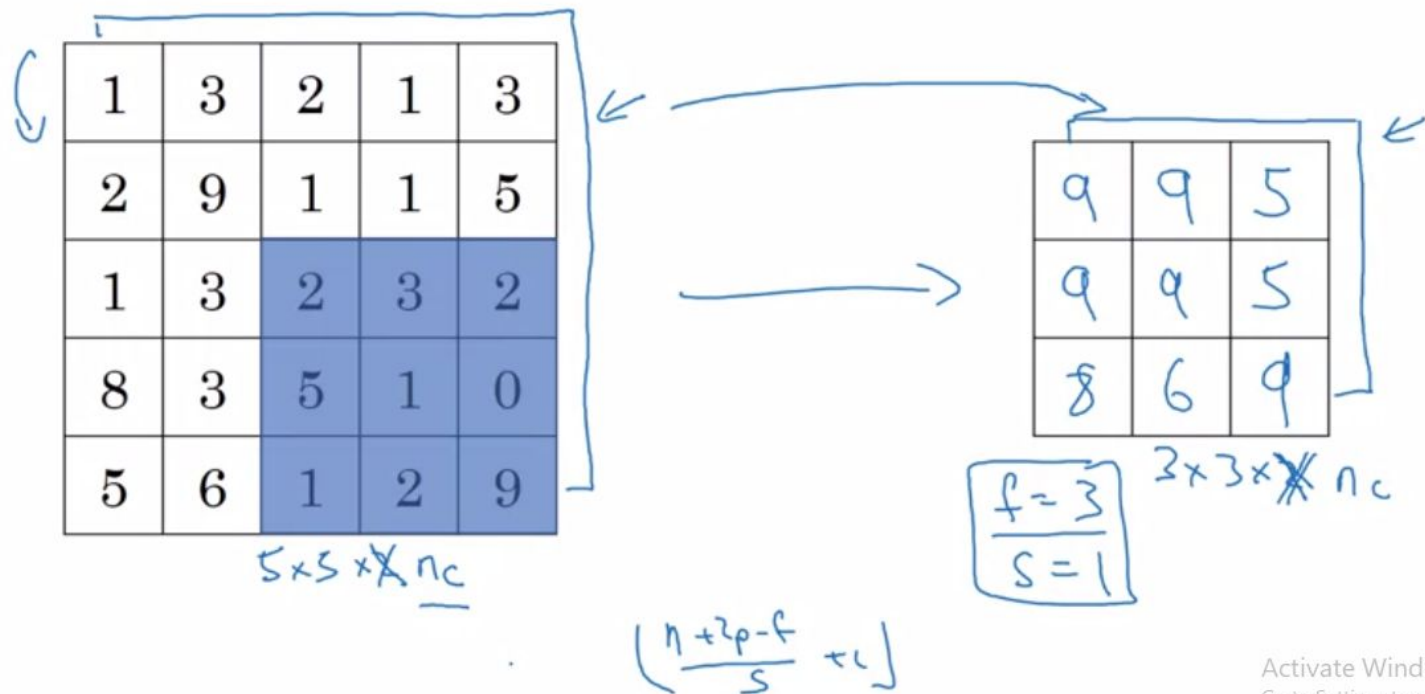
Pooling layer: Max pooling



Hypoparameters:
 $f = \underline{2}$
 $s = \underline{2}$

No parameters!

Pooling layer: Max pooling



Activate Windows
Go to Settings to activate Windows.

Andrew Ng

Pooling layer: Average pooling

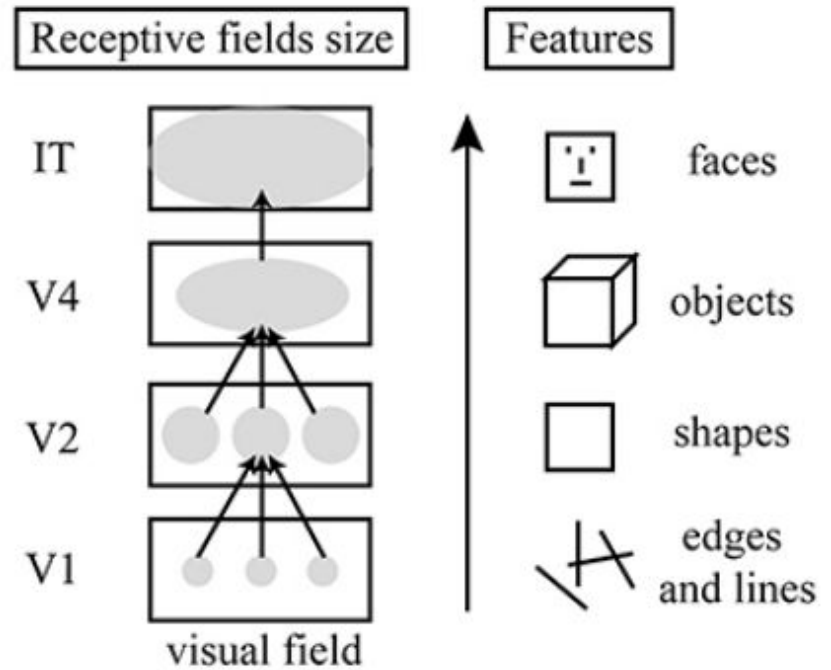
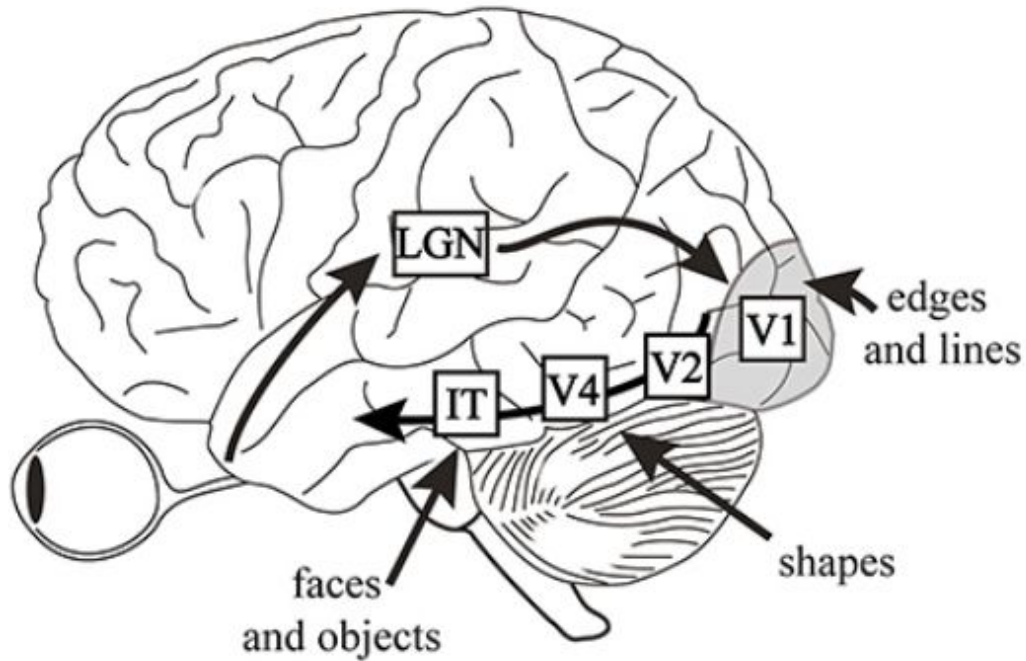
1	3	2	1
2	9	1	1
1	4	2	3
5	6	1	2



3.75	1.25
4	2

$$f=2$$
$$s=2$$

Human Visual Cortex





What Human see, What Computer See!



What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

Problems with MLP

Too Many parameters:

MLPs (Multilayer Perceptron) use one perceptron for each input (e.g. pixel in an image) and the amount of weights rapidly becomes unmanageable for large images. It includes too many parameters because it is fully connected. Each node is connected to every other node in next and the previous layer, forming a very dense web — resulting in redundancy and inefficiency. As a result, difficulties arise whilst training and overfitting can occur which makes it lose the ability to generalize.

Not translation invariant:

MLPs react differently to an input (images) and its shifted version — they are not translation invariant. For example, if a picture of a cat appears in the top left of the image in one picture and the bottom right of another picture, the MLP will try to correct itself and assume that a cat will always appear in this section of the image.

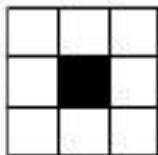
Spatial information:

MLPs are not the best idea to use for image processing. One of the main problems is that spatial information is lost when the image is flattened(matrix to vector) into an MLP.

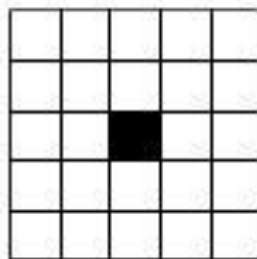
Filter Size



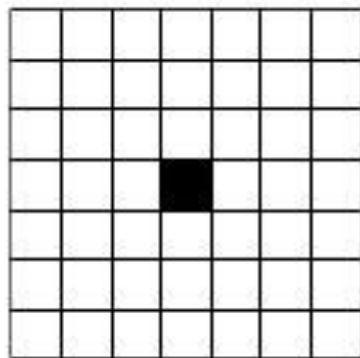
1×1



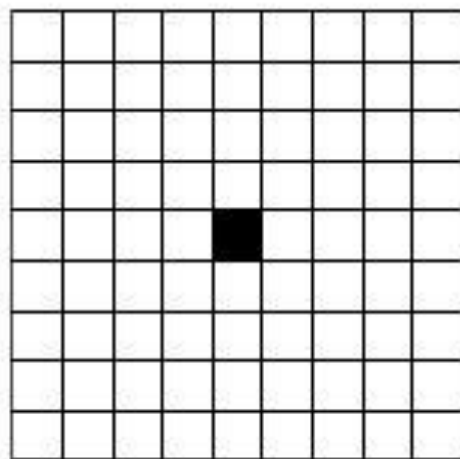
3×3










5×5



7×7



9×9

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

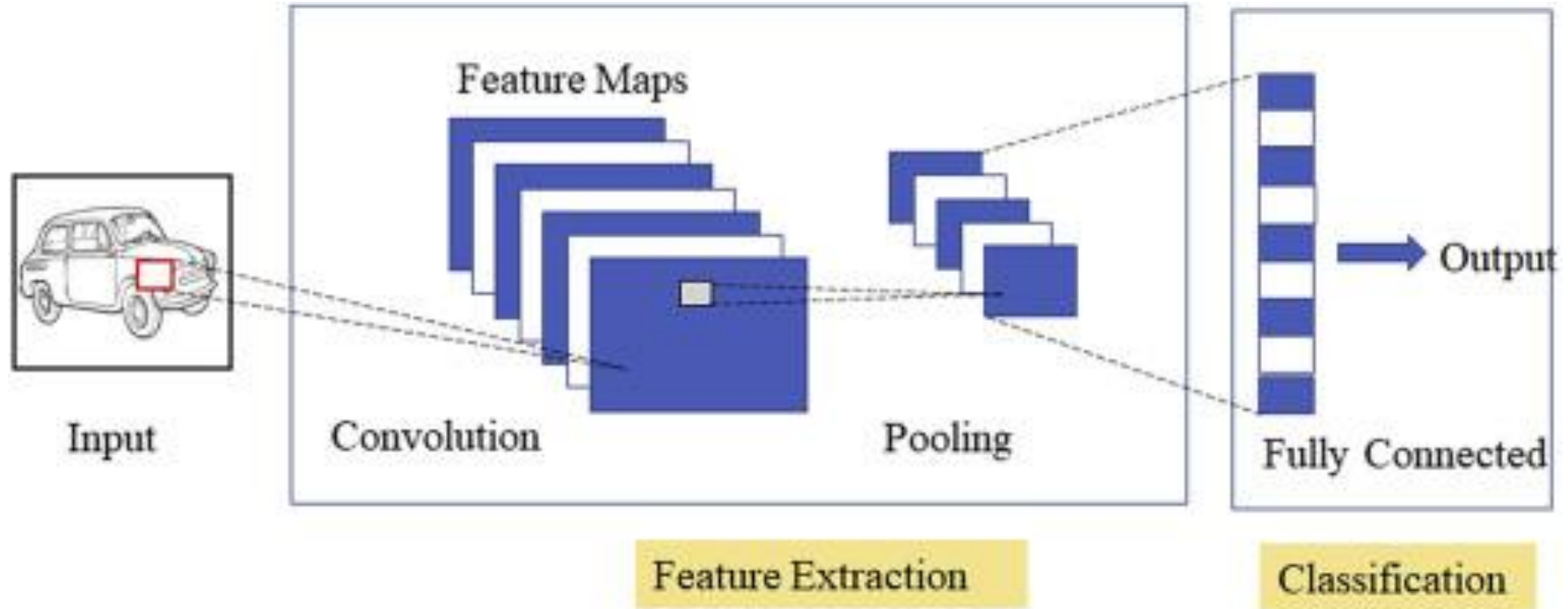
CNN

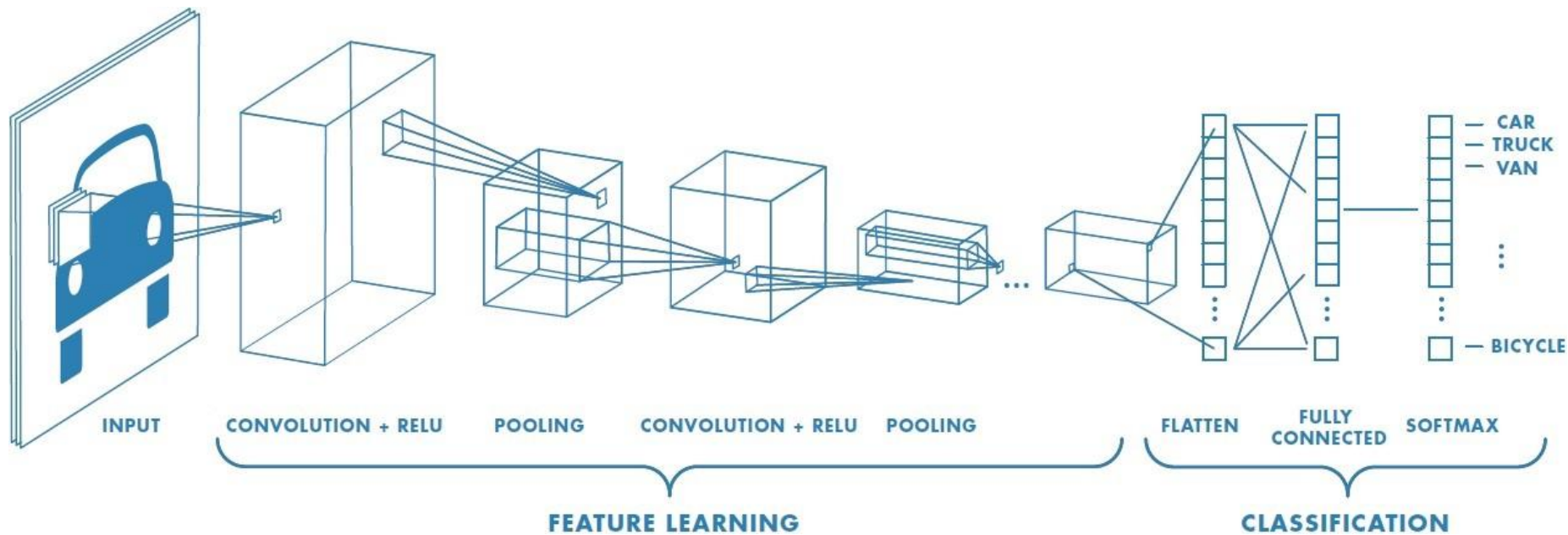
- We can think of a CNN as an artificial neural network that has some kind of specialization for being able to pick out or detect patterns and make sense of them , this pattern detection is what makes CNN so useful for image analysis so if a CNN is just some form of an artificial Neural network, what differentiates it from just a standard multilayer perceptron or MLP.
- Well CNN has HIDDEN Layers called CONVOLUTIONAL Layers and these layers are precisely what makes a CNN. CNN's can and usually do have other Non_Convolutional layers as well but the basis of a CNN is CONVOLUTIONAL layers
- Although image analysis has been the most widespread use of CNN's they can also be used for other data analysis or classification problems as well. More generally, CNNs work well with data that has a spatial relationship. Therefore CNNs are go-to method for any type of prediction problem involving image data as an input.

Convolutional Neural Networks have 2 main components.

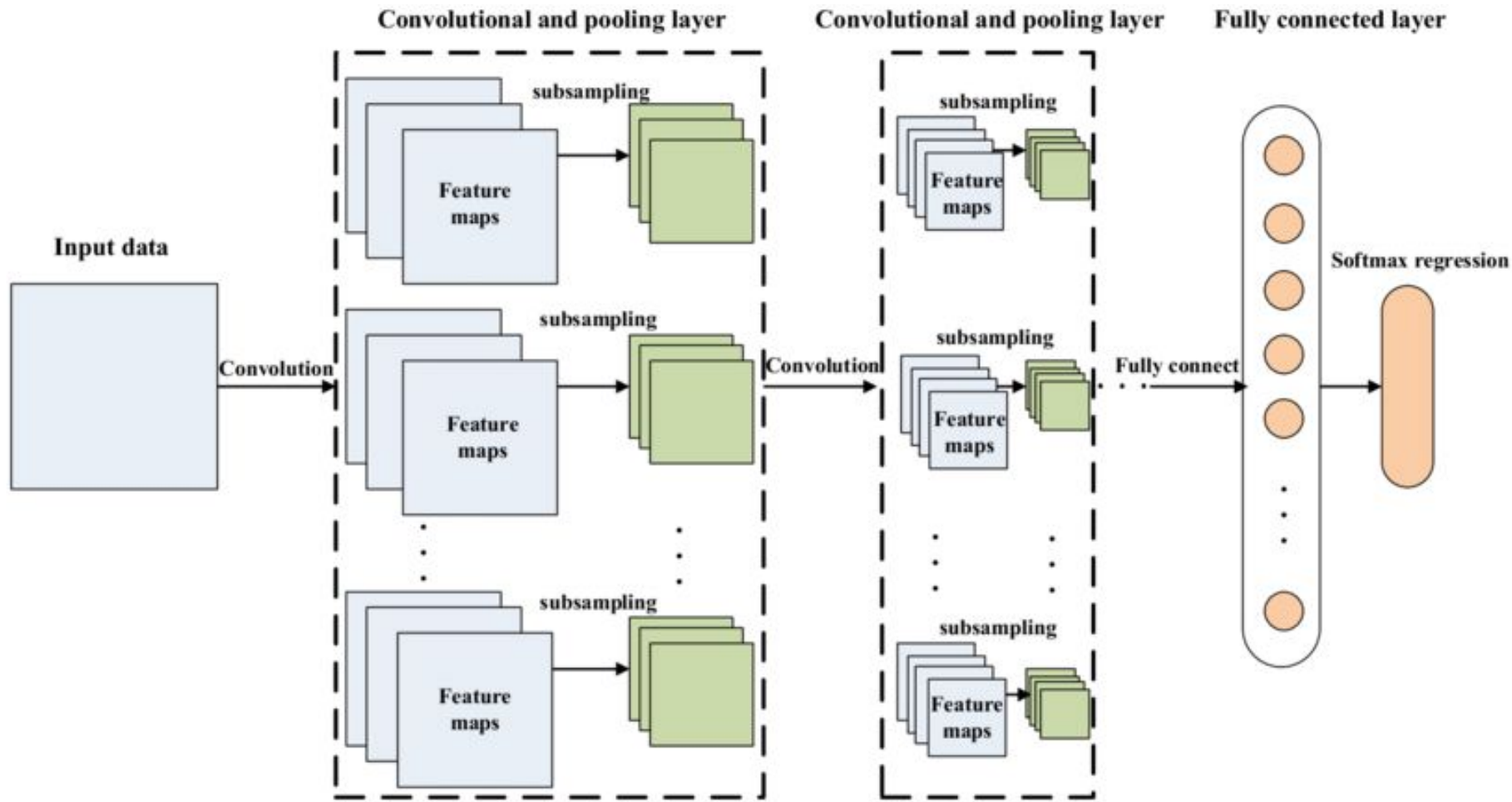
Feature learning: you can see convolution, ReLU, Pooling layer phases here. Edges, shades, lines, curves, in this Feature learning step are get extracted.

Classification: you see Fully Connected(FC) layer in this phase. They will assign a probability for the object on the image being what the algorithm predicts it is.





Ideally, CONV + Pooling is termed as a layer.



Conv Layer

The result of Convolved feature depends on the three parameters

Summary. To summarize, the Conv Layer:

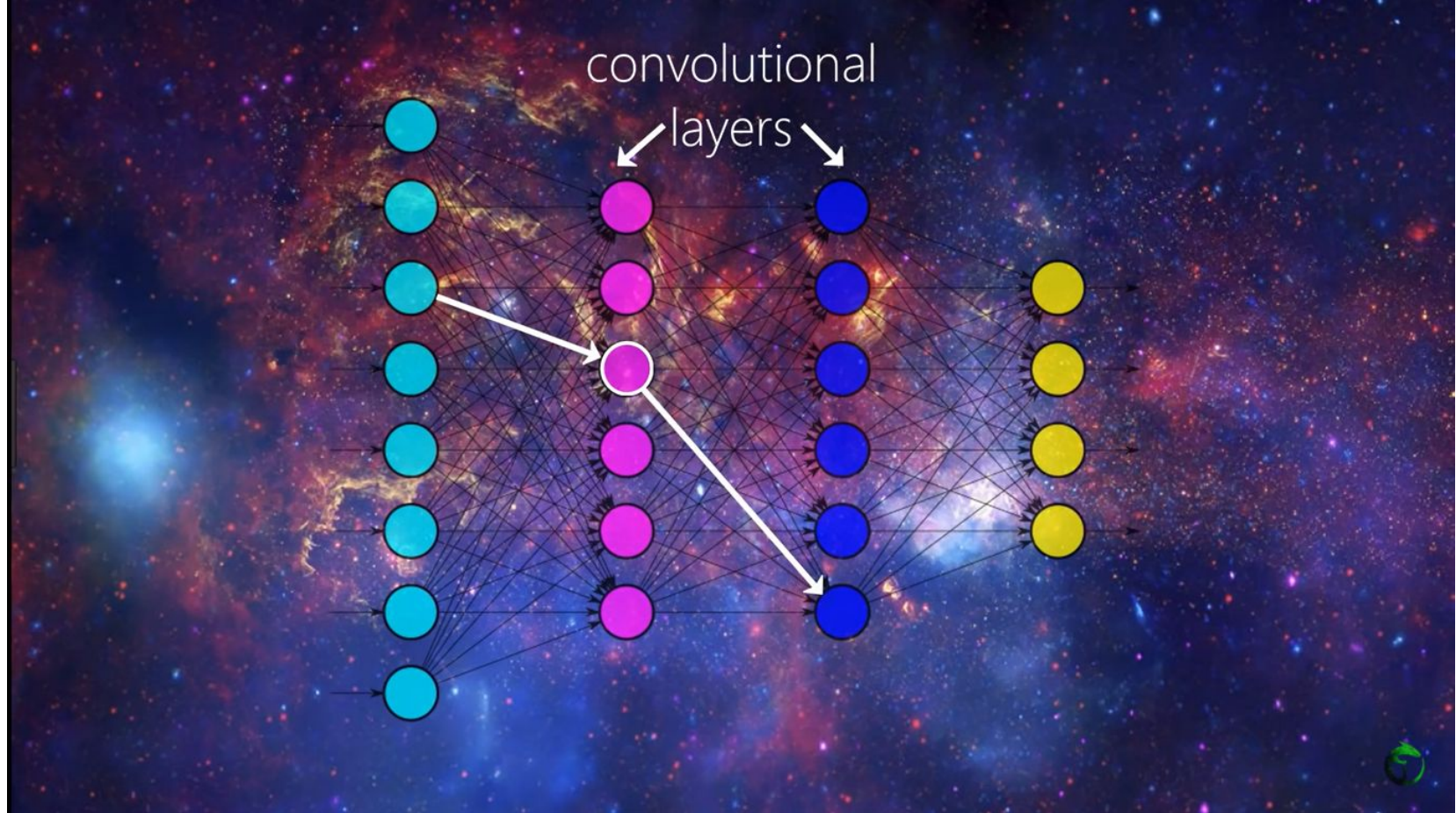
Accepts a volume of size $W1 \times H1 \times D1$

Requires four hyperparameters:

- Number of filters K ,
- their filter size F ,
- the stride S ,
- the amount of zero padding P .

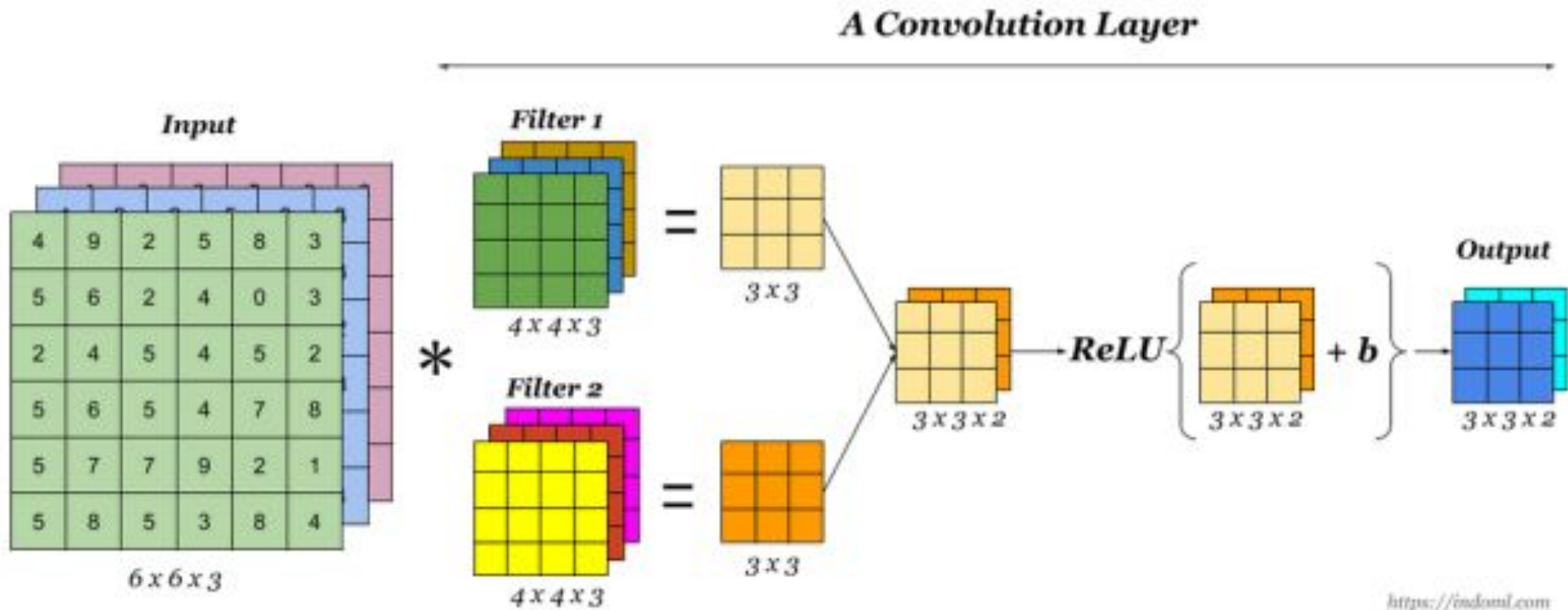
The first layer of a Convolutional Neural Network is always a Convolutional Layer.

A convolutional neural network consists of an input layer, hidden layers and an output layer.



- With each convolutional (Hidden) layer we need to specify the number of filters the layers should have.
- We also specify the size of filter, and the values within the filter are initialized with random numbers.

A Convolutional Layer



Relu Effect

Input Feature Map



ReLU



Rectified Feature Map

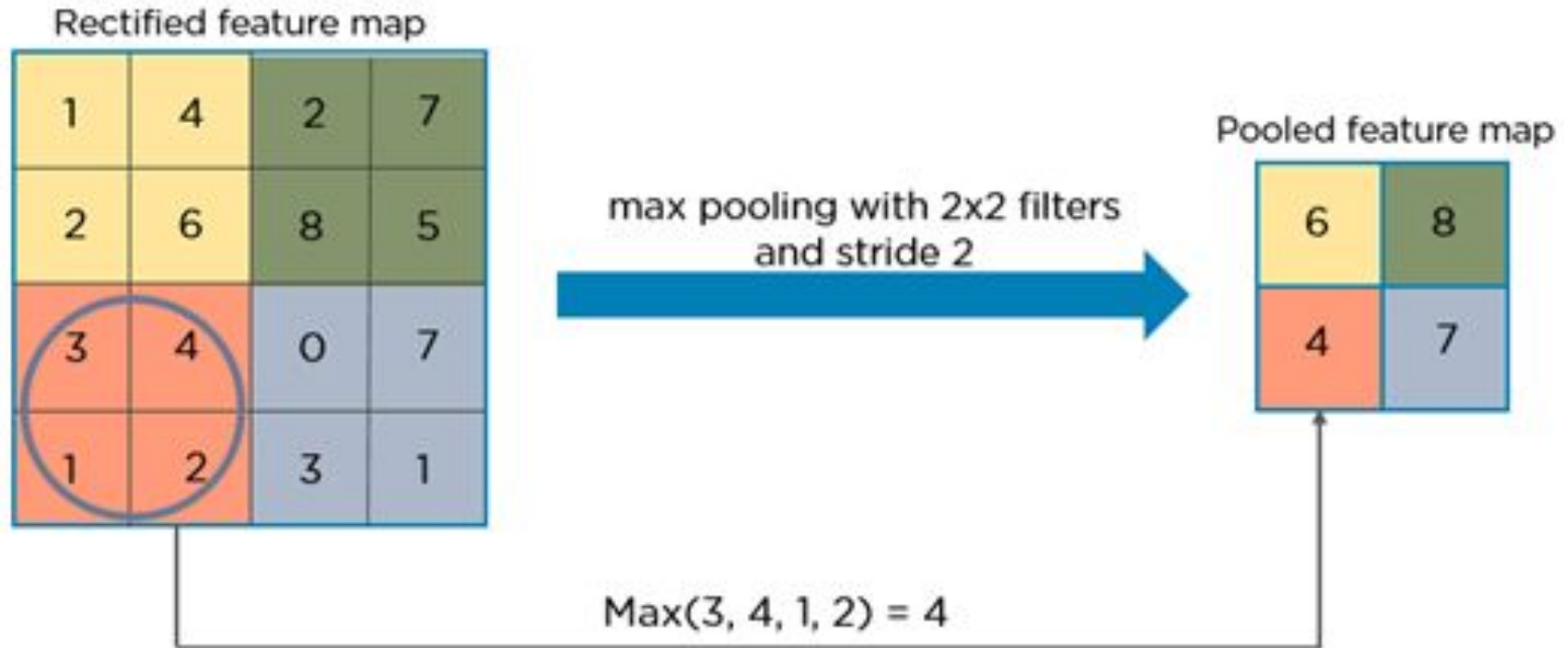


Left: before relu applied,

Right: after ReLU applied

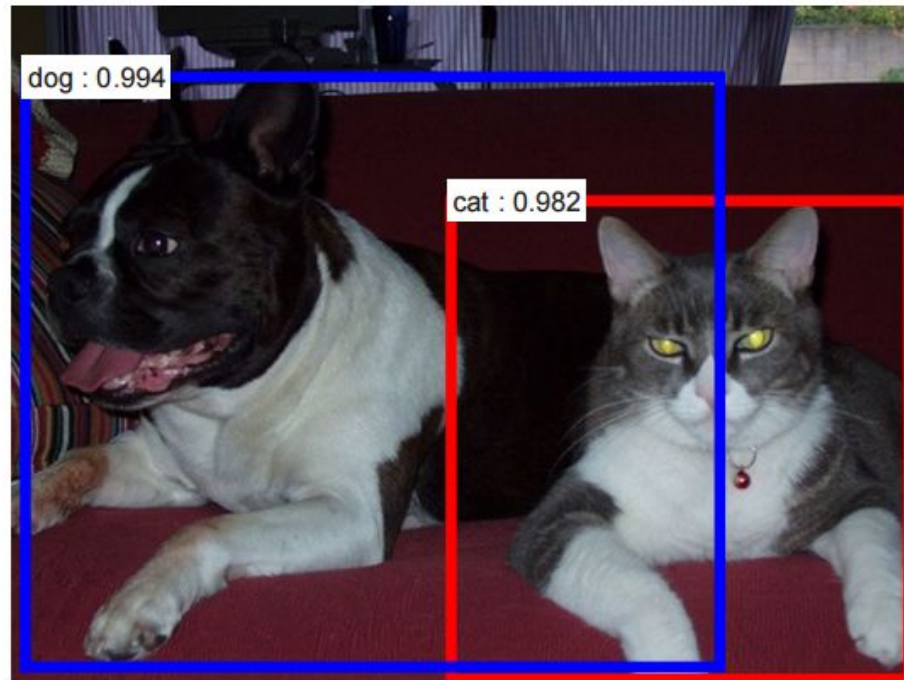
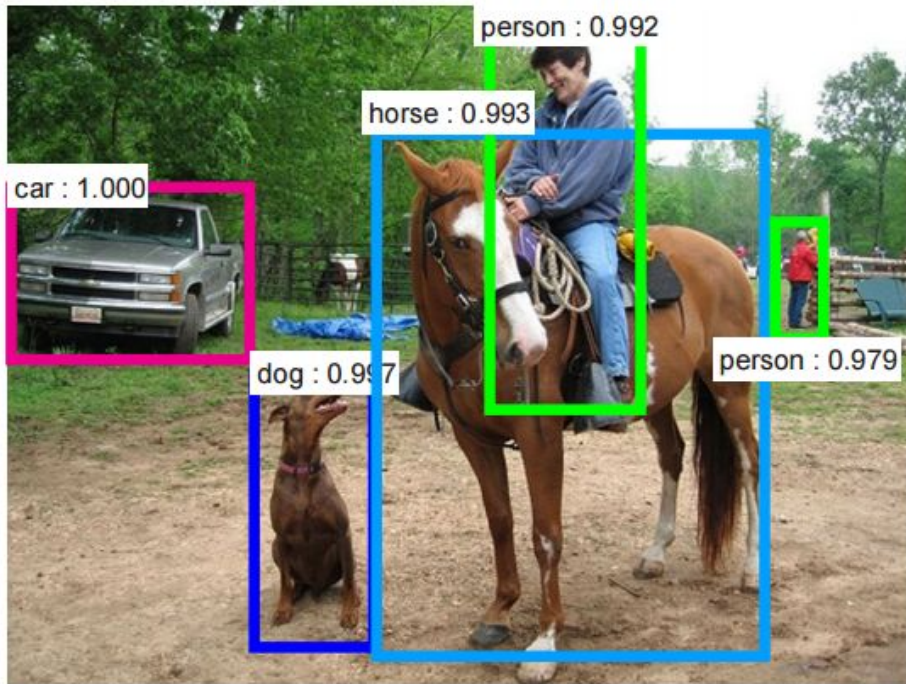
Pooling layer

In this phase the dimensionality of ConvLayer or feature map gets reduced keeping the important information. Have no Learnable parameters.



CNN Applications

Object Detection



Edge Detection



Semantic Segmentation

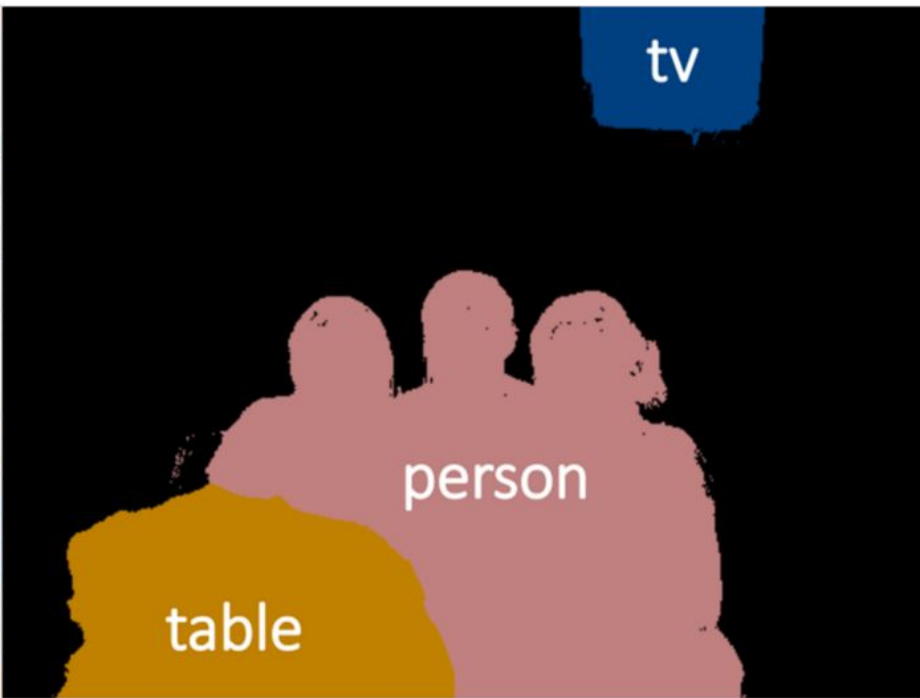


Image Captioning



man in black shirt is playing guitar.



construction worker in orange safety vest is working on road.



two young girls are playing with lego toy.



boy is doing backflip on wakeboard.

Question & Answering



What kind of store is this?	bakery	art supplies
	bakery	grocery
	pastry	grocery
Is the display case full as it could be?	no	no
	no	yes
	no	yes



How many bikes are there?	2	3
	2	4
	2	12
What number is the bus?	48	4
	48	46
	48	number 6

Object tracking

Video classification

style transfer

And many more.

How CNN Learn - Training

Implementation of the CNN model

1. Model Construction

Code:

```
model = Sequential()

model.add(Conv2D(28,(3,3),strides=(1,1),input_shape=X_train.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D((2,2)))

model.add(Flatten()) # flattens the input
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Python Copy

2. Model Training

```
# model compilation
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# model fitting
history = model.fit(x=X_train,y=Y_train, epochs=10)
history
```

[Python](#) [Copy](#)

3. Model Evaluation

Let's evaluate the performance of our model on test data

```
model.evaluate(X_test, Y_test)
```

Output:

Loss - 0.05

Accuracy - 98.73%

```
from keras import models
from keras import layers

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

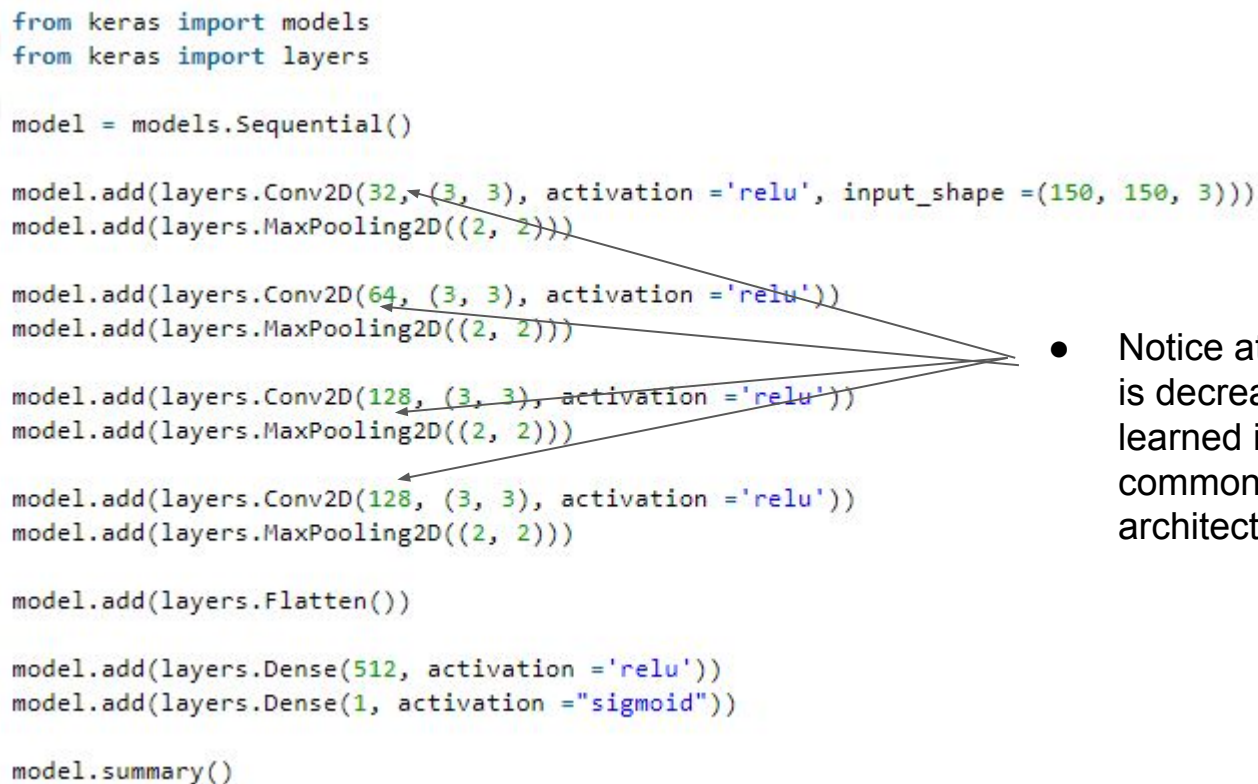
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation="sigmoid"))

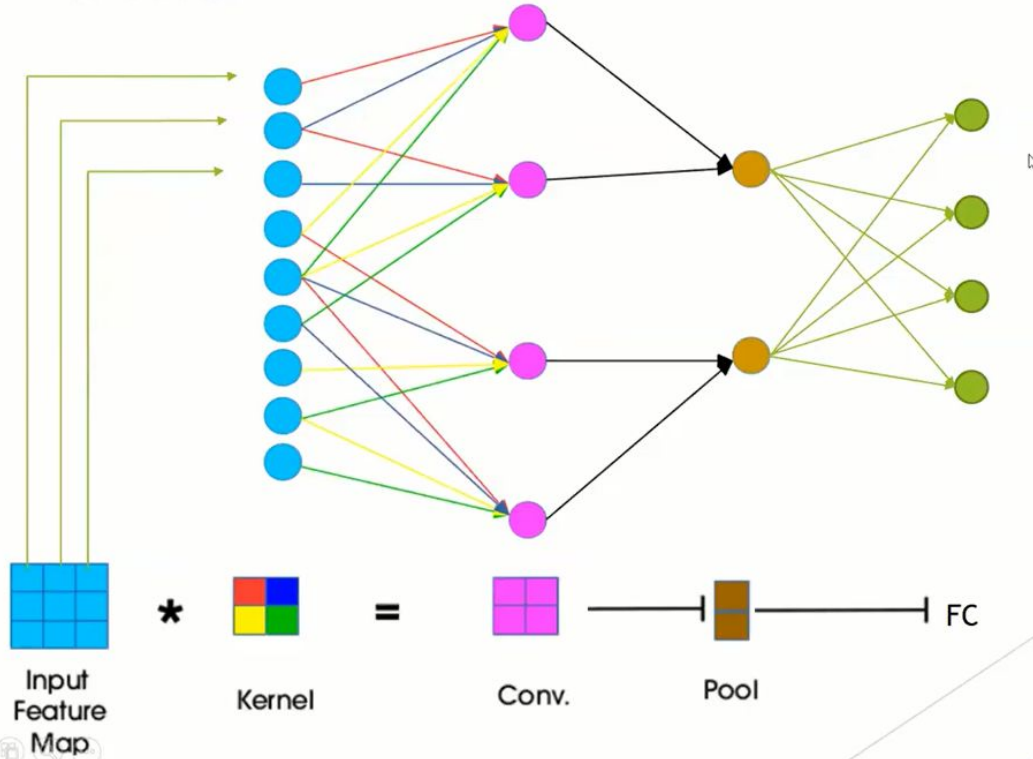
model.summary()
```



- Notice that as our output spatial volume is decreasing our number of filters learned is increasing — this is a common practice in designing CNN architectures

Visualizing CNN as MLP/DNN

CNN as a DNN ?



Back Propagation w.r.t weight: ($l = 2$)

$$\frac{dJ}{dw^{[2]}_{1,1}} = \frac{dJ}{da^{[2]}_1} \frac{da^{[2]}_1}{dz^{[2]}_1} \frac{dz^{[2]}_1}{dw^{[2]}_{1,1}}$$

Repeat for
M examples

$$\frac{dJ}{da^{[2]}_1} = \text{whatever is in previous slide}$$

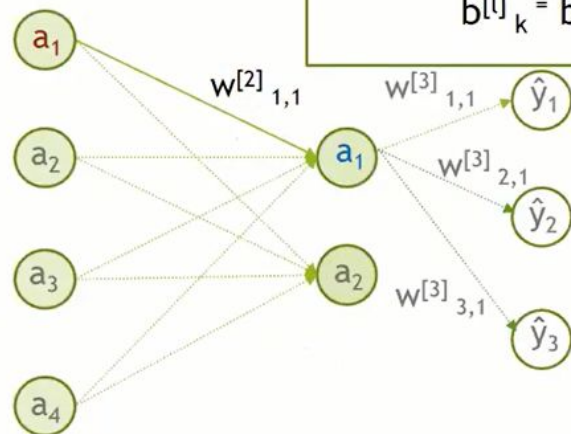
$$\frac{da^{[2]}_1}{dz^{[2]}_1} = a^{[2]}_1(1 - a^{[2]}_1)$$

$$\frac{dz^{[2]}_1}{dw^{[2]}_{1,1}} = a^{[1]}_1$$

for $l = L : 1$

$$\begin{aligned} \frac{dJ}{dw^{[l]}_{i,j}} &= \frac{dJ}{db^{[l]}_k} - \sum_{m=1}^M \frac{dJ}{dw^{[l]}_{i,j}} \\ b^{[l]}_k &= b^{[l]}_k - \sum_{m=1}^M \frac{dJ}{db^{[l]}_k} \end{aligned}$$

Backward
Propagation



$$J = \sum_{k=1}^{n^{[3]}} (\hat{y}_k - y_k)^2$$

$l = 0$

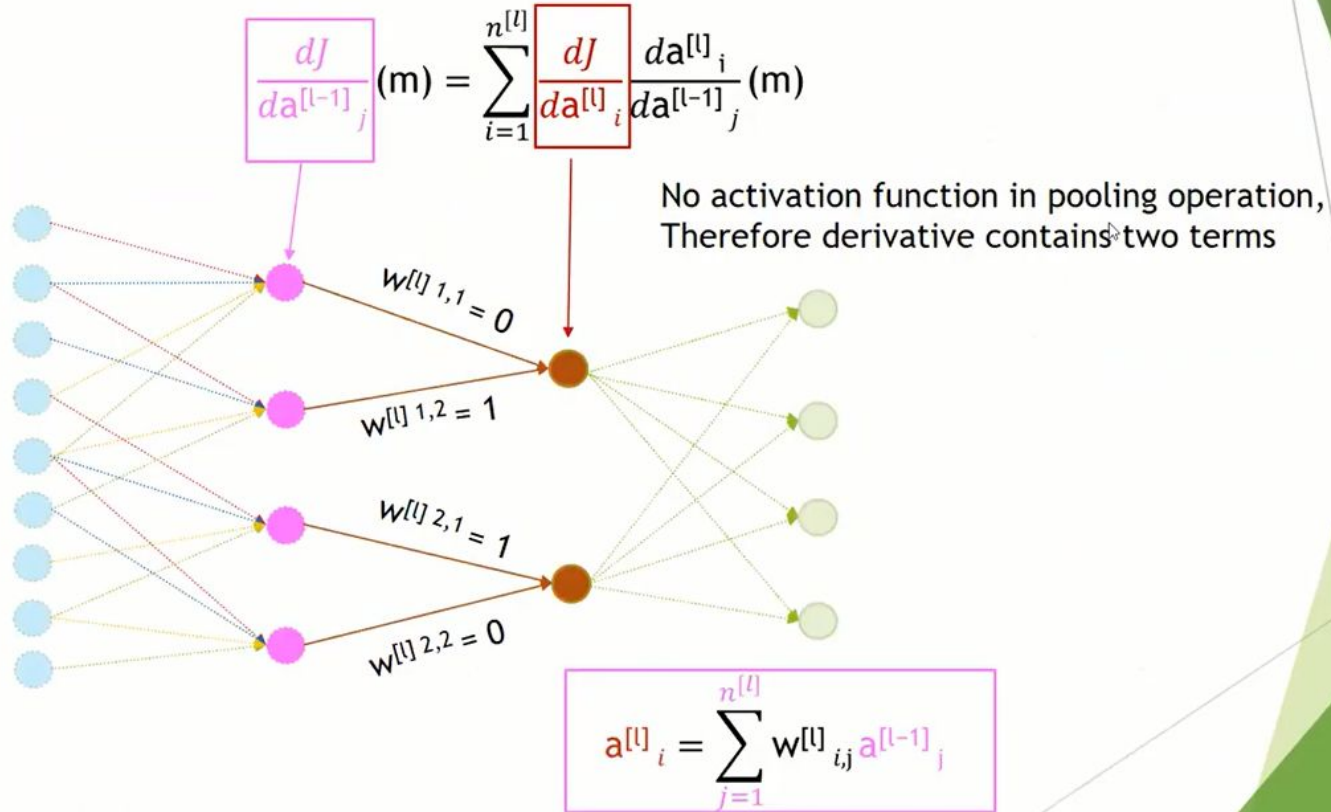
$l = 1$

$l = 2$

$l = 3$

$l \in [1, \dots, L]$

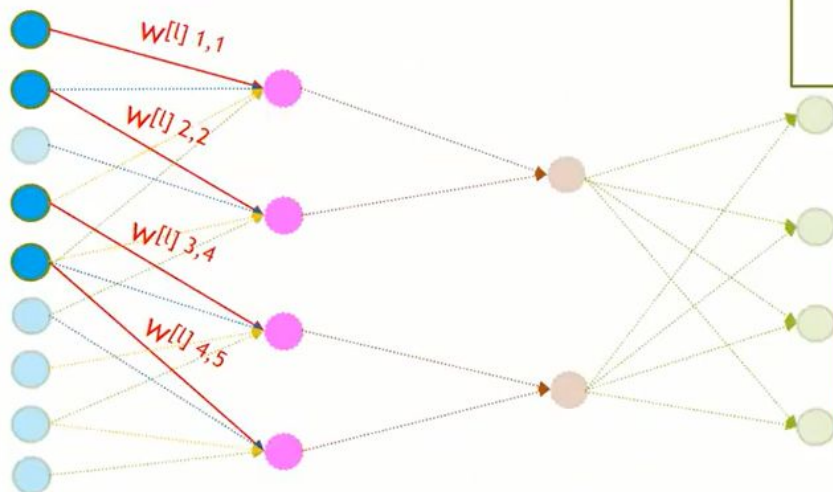
From Pooling to Conv layer ...



From Conv layer to input layer ... updating filter coefficients

K^{th} filter coefficient at layer l

$$\frac{dJ}{dw^{[l]}_{i,j}} = \frac{dJ}{da^{[l]}_i} \frac{da^{[l]}_i}{dz^{[l]}_i} \frac{dz^{[l]}_i}{dw^{[l]}_{i,j}}$$



for $l = L : 1$

$$f^{[l]}_k = f^{[l]}_k \cdot \sum_{m=1}^M \frac{dJ}{df^{[l]}_k}$$

Backward
Propagation

Repeat for
M examples

$$\frac{dJ}{df^{[l]}_k} = \frac{dJ}{dw^{[l]}_{1,1}} + \frac{dJ}{dw^{[l]}_{1,2}} + \frac{dJ}{dw^{[l]}_{3,4}} + \frac{dJ}{dw^{[l]}_{4,5}}$$

Xtra

How CNN reduce weights? - Weight Sharing

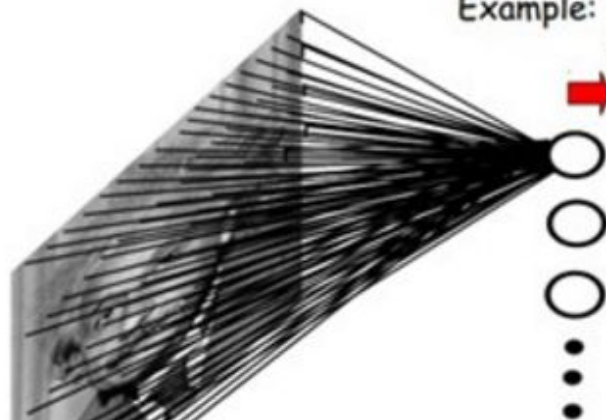
The neural network (in MLP) will learn different interpretations for something that is possibly the same. But in CNN, the number of weights is dependent on the kernel size (see Weight sharing) instead of the input size which is really important for images. So, by forcing the shared weights among spatial dimensions which drastically reduces the number of parameters, the convolution kernel acts as a learning framework.

That's how convolutional layers reduce memory usage and compute faster.

Local connection: the features needed to distinguish different kinds of pictures are only some local areas in the whole picture, so the convolution kernel (receptive field) used in convolution operation can be just a few different small areas, instead of using the convolution kernel (full connection) of the size of the whole picture. This can not only express different features better, but also reduce parameters. For example, in the figure below, on the the left is the network using full connection neural network, and on the right is the network using local connection convolution kernel.

FULLY CONNECTED NEURAL NET

Example: 1000x1000 image
1M hidden units
→ 10^{12} parameters!!!

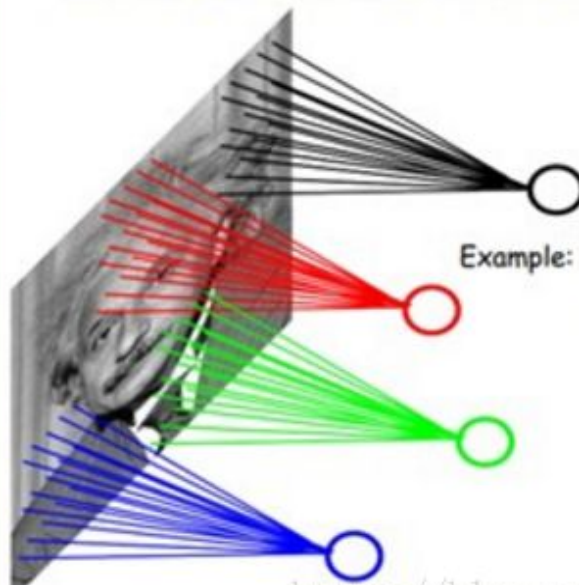


- Spatial correlation is local
- Better to put resources elsewhere!

59

LOCALLY CONNECTED NEURAL NET

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters



Ranzoi

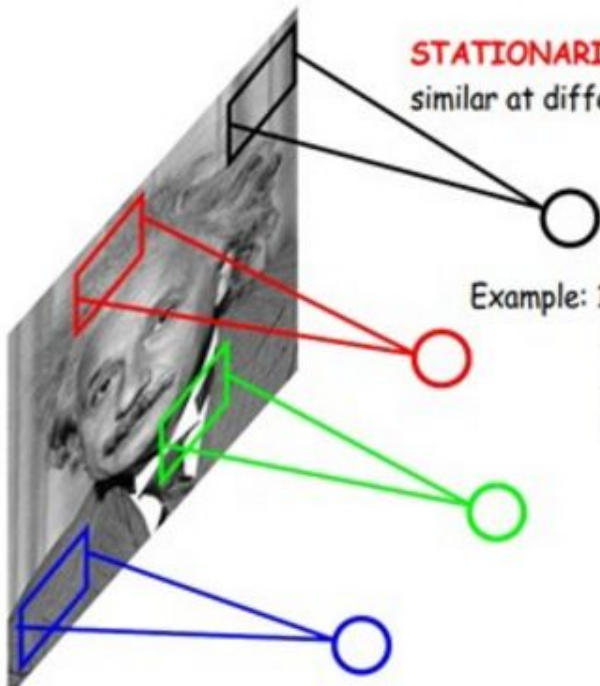
https://blog.csdn.net/comway_Li

Weight sharing: for a class of pictures, they have similar features, but the position of features in each picture may be shifted. For example, the position of eyes in different face photos may change, and rarely two photos have the same eye position. When a picture is convoluted, there can be multiple convolution kernels to extract different features, but the weight of a convolution kernel remains unchanged in the process of moving (of course, the weight of different convolution kernels is not shared). This can not only ensure that the feature extraction is not affected by the location, but also reduce the number of parameters.

LOCALLY CONNECTED NEURAL NET

STATIONARITY? Statistics is similar at different locations

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

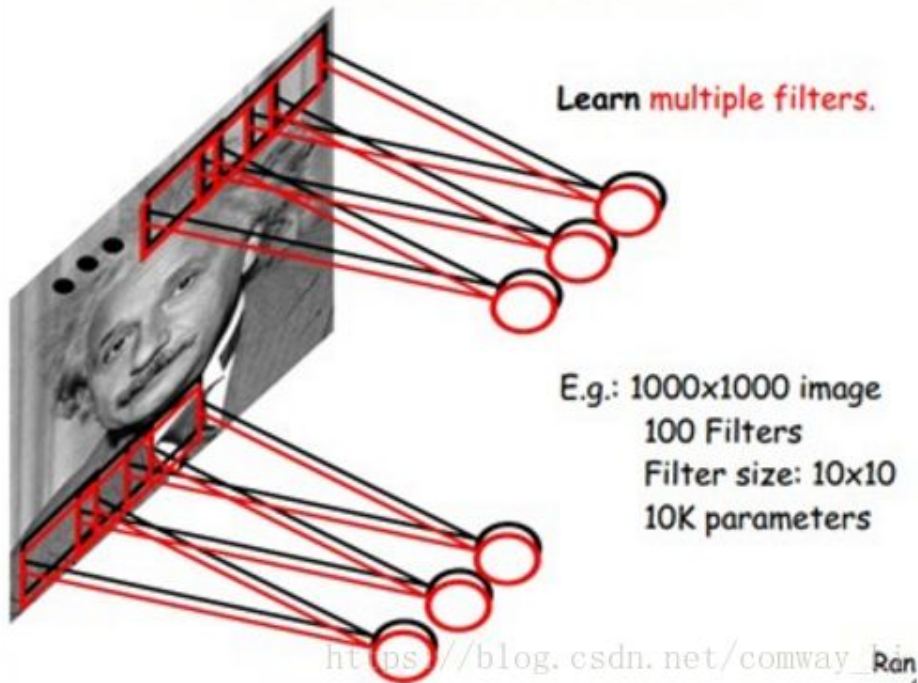


Ranza

CONVOLUTIONAL NET

Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters



https://blog.csdn.net/comway_Ran

Batch Normalization

Sometimes, one additional step called batch normalization is applied after certain layers in the CNN. Batch normalization is a technique used to increase the stability of a neural network. It helps our neural network to work with better speed and provide more efficient results. Let us say we have different parameters and the values of those parameters differ on a great scale i.e. if parameter A has value in range 1-10 and parameter B has values in range 1-million, then some issues might occur. This is where batch norm will help. It can help us by normalizing values in a certain range (lets say 0-1). Now all the values are changed and ranges b/w 0 and 1.