CNN Intuition and working

Dr. Sagarika B

Recap: Session 1,2,3

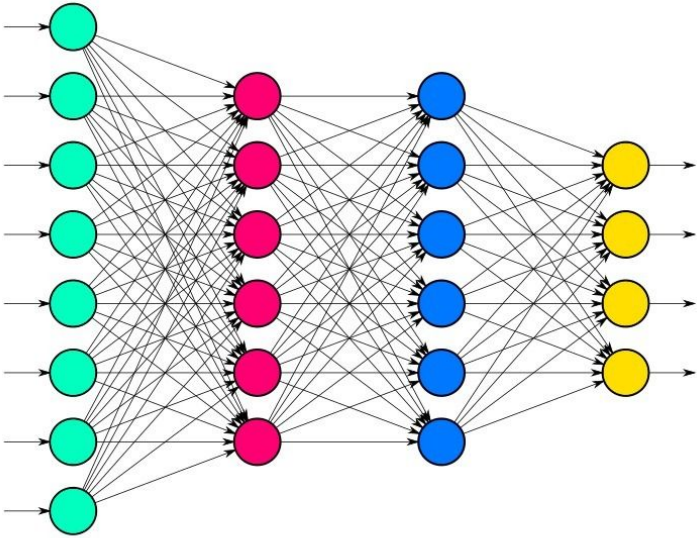
- ANN Training
  - Forward propagation
  - Back propagation
- ANN performance improvement
  - Vanishing Gradient problem
  - Under fitting
  - Overfitting
  - SLow training
- Hyper parameter tuning
  - Activation function
  - Model architecture
  - Weight initialization
  - Regularization
  - Optimizers
  - Learning rate

Agenda: Session 4

- Issues faced by ANN
- CNN intro
- Layers in CNN
- How learning happens in CNN?

What are the problems in ANN ?

What is a Dense layer ?



Input layer
Hidden layer
Output layer

Which one is Dense layer ?

Limitations of a dense layer !!

- More number of Number of weights and biases leads to overfitting

  This mostly happens in images (every pixel as an input)

- Spatial info is lost (Neighborhood information)

Nose !!

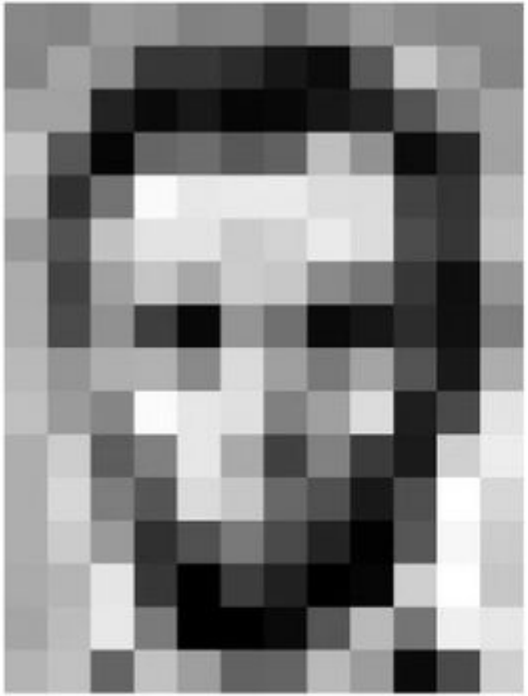
Not Nose !!

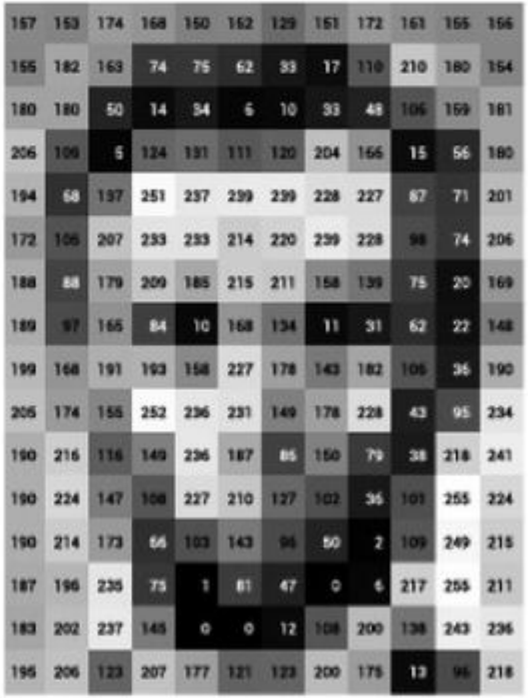
135	135	129	133	130	134	134	137
133	133	132	132	135	127	123	119
132	127	129	115	121	87	96	110
110	104	115	109	120	103	129	160
105	112	136	162	173	201	219	231
167	187	202	223	216	231	240	238
221	231	240	223	214	216	218	219
224	217	222	214	215	217	219	220



......

One solution could be to Reduce the image resolution !!





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

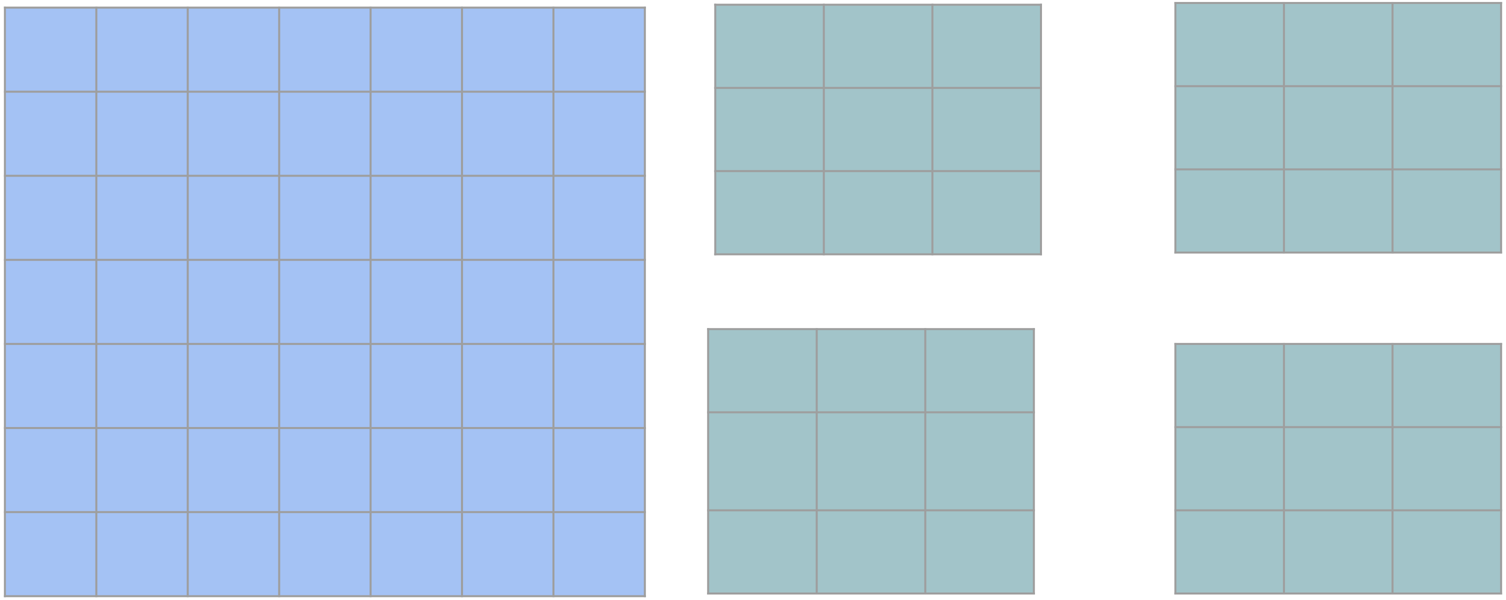
This may lead to underfitting !!

How to solve these problems !!

Convolutional Layers

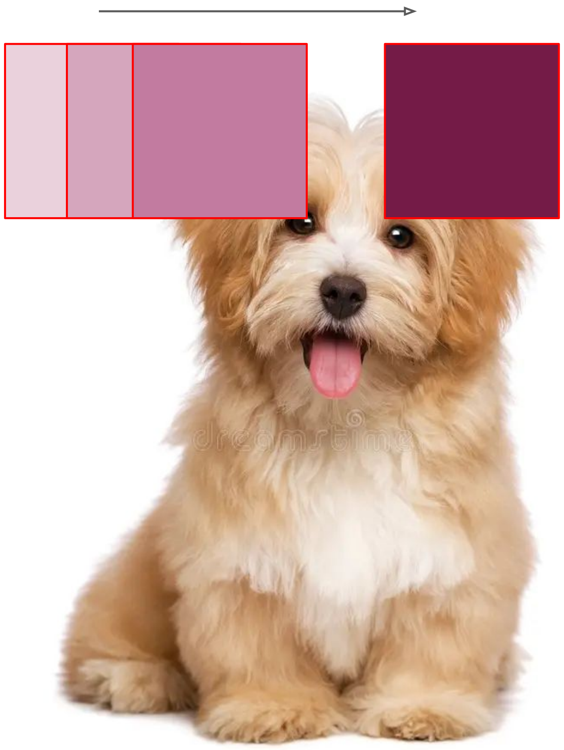
- Reduce the number of weights and biases
- Neighborhood information is captured !!

Look at the image part by part !!



Convolution Operation

Convolve = Sliding window !!



kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23				

$=(1x2)+(2x4)+(5x1)+(3x0)+(1x0)+(2x0)+(2x1)$
$+(3x2)+(1x1)$

$=\mathbf{23}$

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28			

=(1x4)+(2x5)+(6x1)+(1x0)+(2x0)+(3x0)+(3x1)
+(1x2)+(3x1)

=28

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32		

=(1x5)+(2x6)+(2x1)+(2x0)+(3x0)+(5x0)+(1x1)
+(3x2)+(6x1)

=**32**

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	

=(1x6)+(2x2)+(3x1)+(3x0)+(1x0)+(2x0)+(3x1)
+(6x2)+(0x1)

=**28**

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36				

$$=(1\times3)+(2\times1)+(2\times1)+(2\times0)+(3\times0)+(1\times0)+(7\times1)+(7\times2)+(8\times1)$$

$$=\mathbf{36}$$

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35			

$=(1x1)+(2x2)+(3x1)+(3x0)+(1x0)+(2x0)+(7x1)$
$+(8x2)+(4x1)$

=35

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35		

=(1x2)+(2x3)+(5x1)+(1x0)+(3x0)+(6x0)+(8x1)
+(4x2)+(6x1)

=35

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	

$=(1x3)+(2x4)+(7x1)+(3x0)+(6x0)+(0x0)+(4x1)$
$+(6x2)+(2x1)$

=36

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	

=(1x3)+(2x4)+(7x1)+(3x0)+(6x0)+(0x0)+(4x1)
+(6x2)+(2x1)

=36

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31				

$$=(1x2)+(2x3)+(1x1)+(7x0)+(7x0)+(8x0)+(8x1)$$
$$+(4x2)+(6x1)$$

=31

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33			

$$=(1x3)+(2x1)+(3x1)+(7x0)+(8x0)+(4x0)+(4x1)+(6x2)+(9x1)$$

$$=\mathbf{33}$$

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33		

=(1x1)+(2x3)+(6x1)+(8x0)+(4x0)+(6x0)+(6x1)
+(9x2)+(4x1)

=33

kernel

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33	41	

=(1x3)+(2x6)+(0x1)+(4x0)+(6x0)+(2x0)+(9x1)
+(4x2)+(8x1)

=41

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

kernel

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33	41	
	44				

$$=(1x7)+(2x7)+(8x1)+(8x0)+(4x0)+(6x0)+(3x1)$$
$$+(4x2)+(6x1)$$

$$=44$$

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

kernel

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33	41	
	44	51			

=(1x7)+(2x8)+(4x1)+(4x0)+(6x0)+(9x0)+(4x1)
+(6x2)+(1x8)

=51

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

kernel

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33	41	
	44	51	54		

=(1x8)+(2x4)+(6x1)+(6x0)+(9x0)+(4x0)+(6x1)
+(8x2)+(10x1)

=**54**

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

kernel

filter

1	2	1
0	0	0
1	2	1

	23	28	32	28	
	36	35	35	36	
	31	33	33	41	
	44	51	54	50	

Feature map

=(1x4)+(2x6)+(2x1)+(9x0)+(4x0)+(8x0)+(8x1)+(10x2)+(4x1)

=50

Why convolution ?

Example a kid learning to draw,

He will learn with lines, curves……

What is a filter ?

A filter, or kernel, in a CNN is a small matrix of weights that slides over the input data (such as an image), performs element-wise multiplication with the part of the input it is currently on, and then sums up all the results into a single output pixel.

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

Image
6x6

1	2	1
0	0	0
1	2	1

Filter
3x3

How many weights to learn ?

9

1 filter = 1 neuron

Size of a filter : Hyperparameter (usually small size)

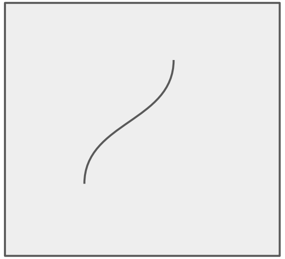
Sliding the Filter: The filter slides across the input data, moving by a certain number of pixels each time, defined by the “stride”

Feature Extraction: Filters are responsible for feature extraction in CNNs.

For example, some filters might become specialized to detect horizontal edges in an image, others might detect vertical edges, colors, textures, etc.

As the model becomes deeper, the filters can recognize more complex patterns.

I am looking for a feature which looks like this —--



High value in the feature map after convolution means strong presence of the feature.

0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	0	0	0
0	0	1	0	0

Kernel

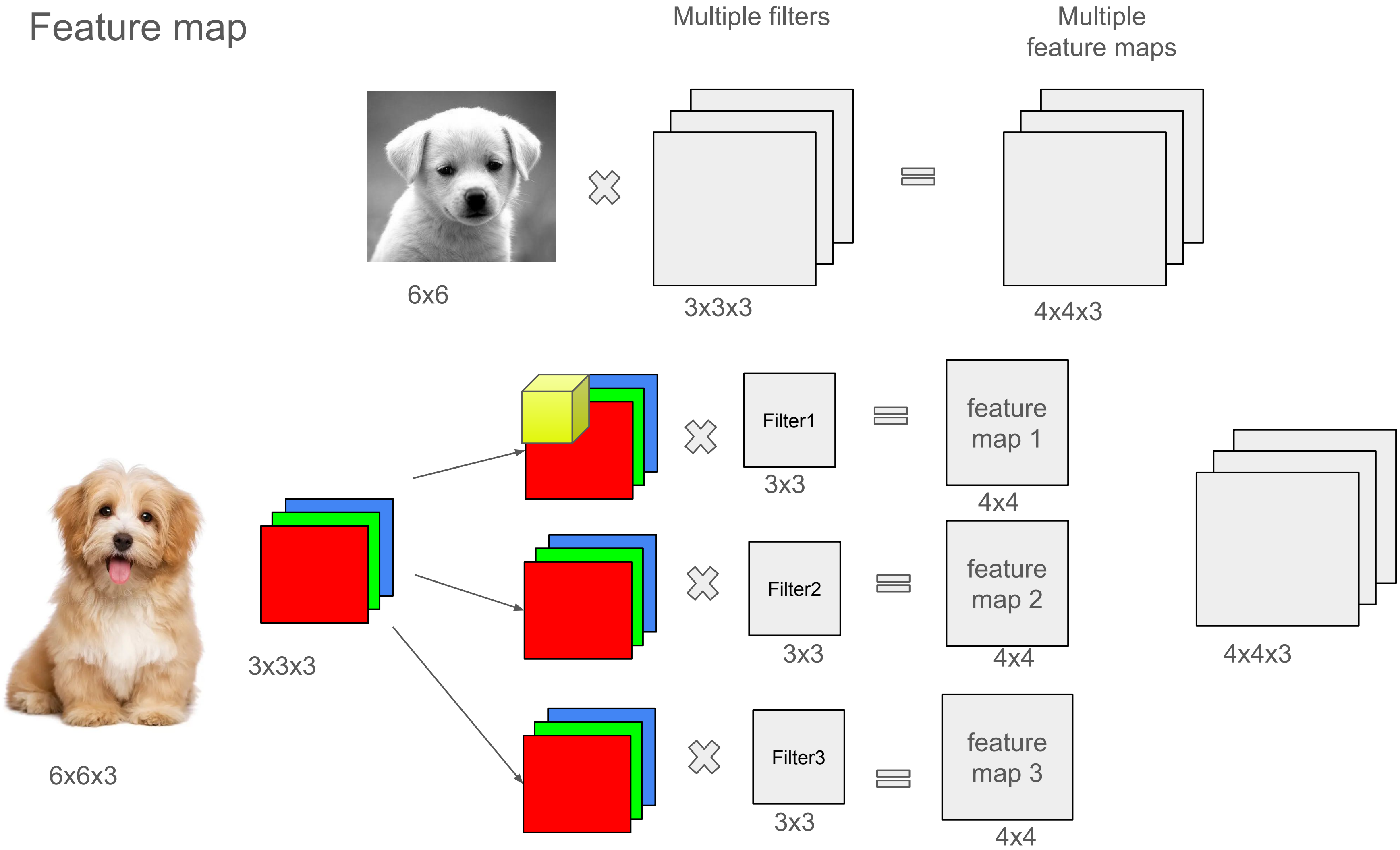
Edge detection

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Sharpen

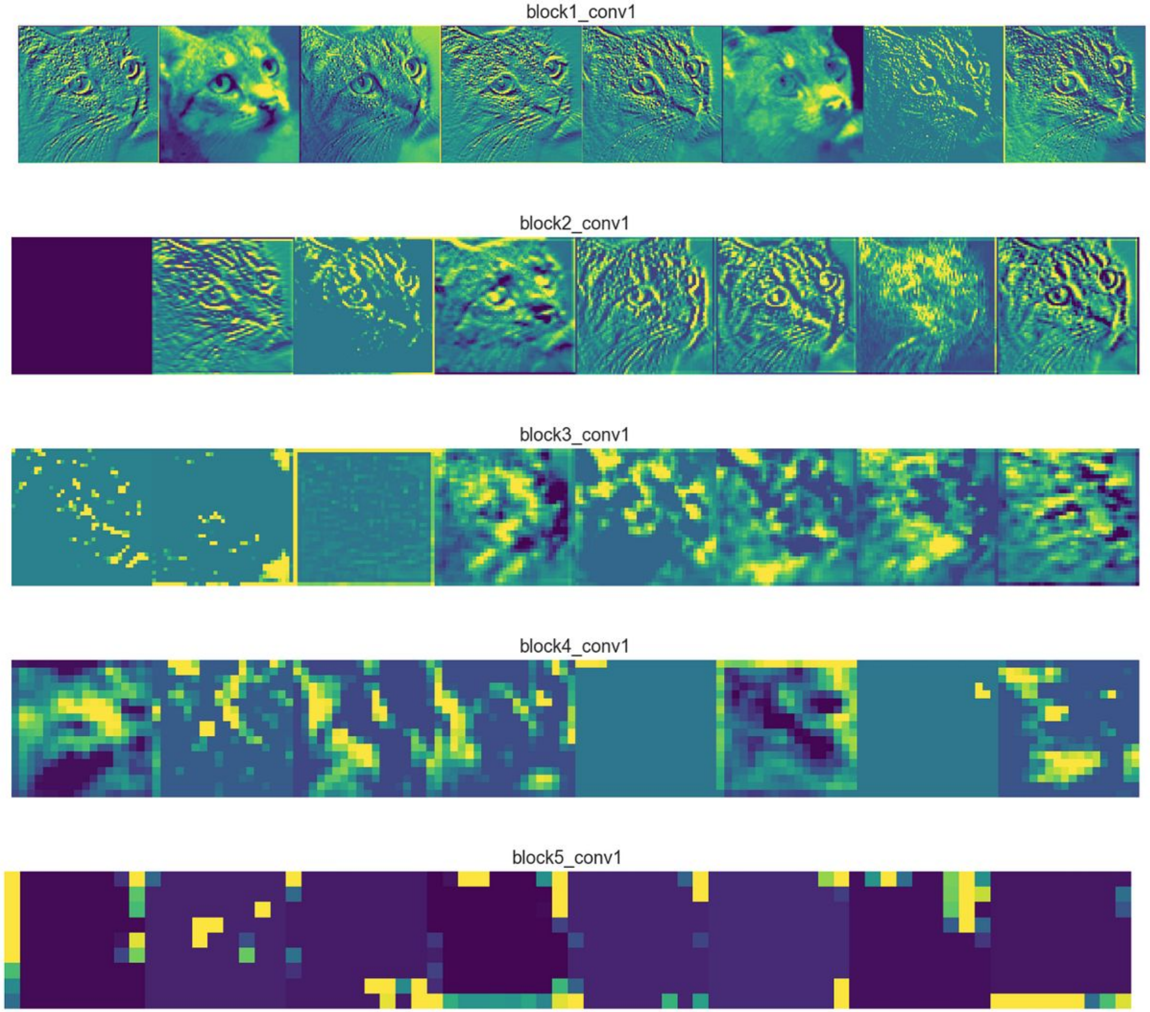
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

Feature map



Multiple filters
Multiple
feature maps
6x6
3x3x3
4x4x3
6x6x3
3x3x3
Filter1
3x3
feature
map 1
4x4
Filter2
3x3
feature
map 2
4x4
Filter3
3x3
feature
map 3
4x4
4x4x3

Visualize feature maps



Stride

- The number says how many pixels will slide from left to right during convolution.

Stride =(1,1)

2	4	5	6	2	3
3	1	2	3	5	7
2	3	1	3	6	0
7	7	8	4	6	2
8	4	6	9	4	8
3	4	6	8	10	4

Stride value High
Resulting output feature map size decreases

Stride value Low
Output feature map size will be larger.

This can help us to reduce overfitting !!!

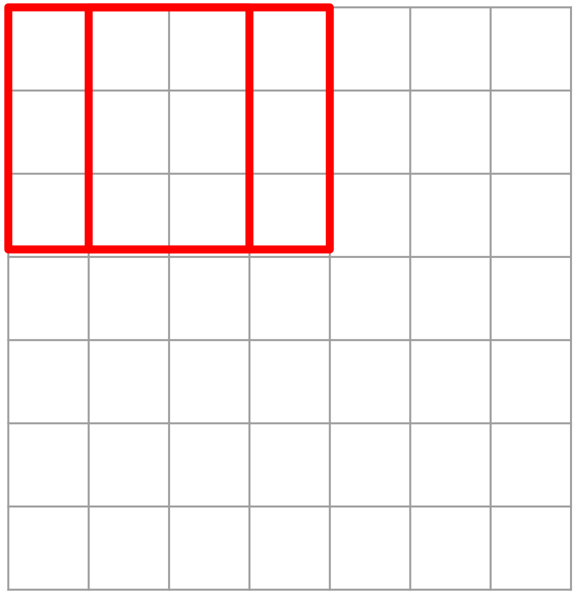


Image size = 7x7
Filter size = 3x3
stride = 1

Output feature map size=
5x5

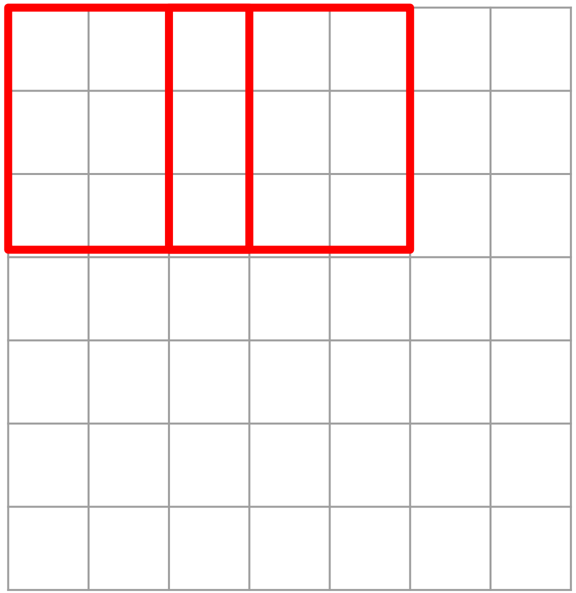


Image size = 7x7
Filter size = 3x3
stride = 2

Output feature map size=
3x3

FM= (N-F)/S +1

FM= feature map size
N= Image size
F= filter size
S= Stride size

Padding

A padding layer is typically added to ensure that the outer boundaries of the input layer doesn’t lose its features when the convolution operation is applied. It is also done to adjust the size of the input

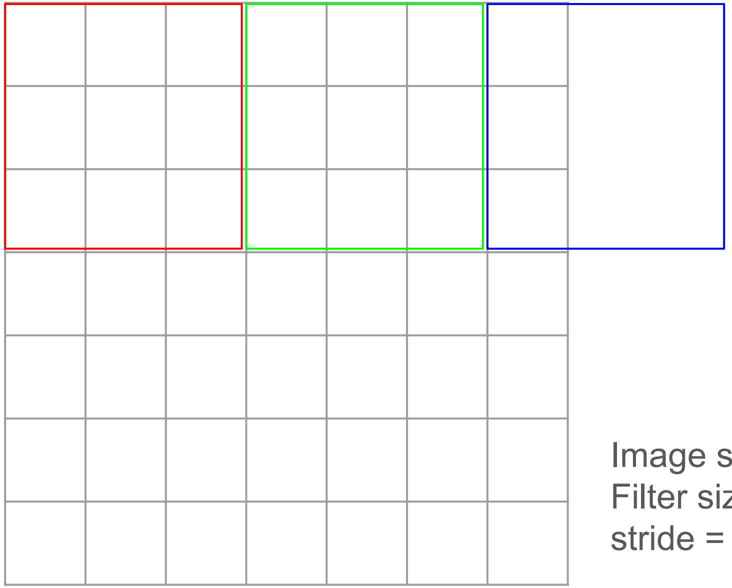
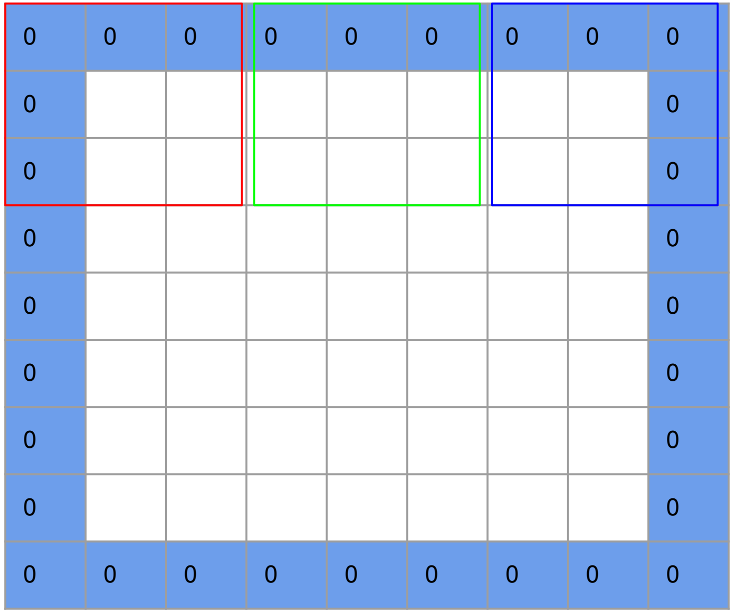


Image size = 7x7
Filter size = 3x3
stride = 3

Does not fit !!

Zero padding



What are the types of padding ?

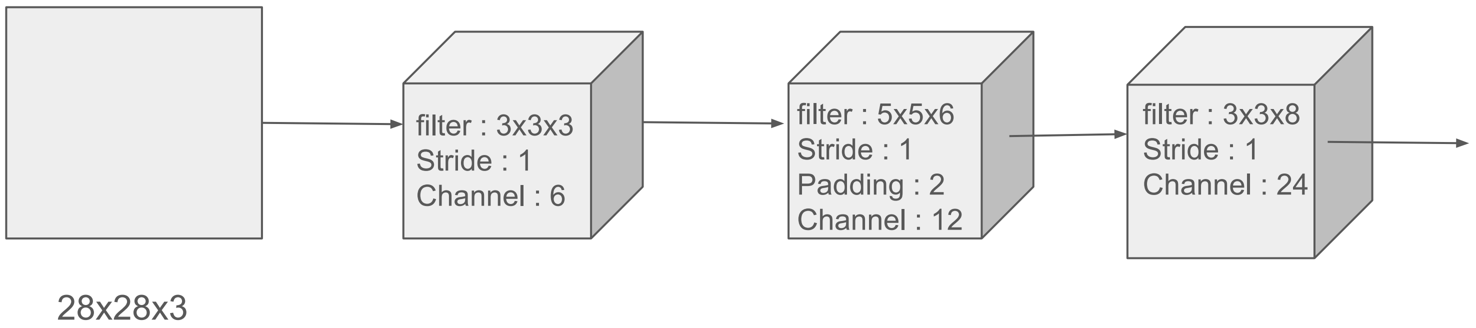
Valid Padding: This type of padding involves no padding at all. The convolution operation is performed only on the valid overlap between the filter and the input. As a result, the output dimensions will be smaller than the input dimensions.

Same Padding: In this approach, padding is added to the input so that the output dimensions after the convolution operation are the same as the input dimensions. This is typically achieved by adding an appropriate number of zero-value pixels around the input.

Feature map size

$$FM = (N+2P-F)/S+1$$

FM= feature map size
N= Image size
F= filter size
S= Stride size
P= Padding size

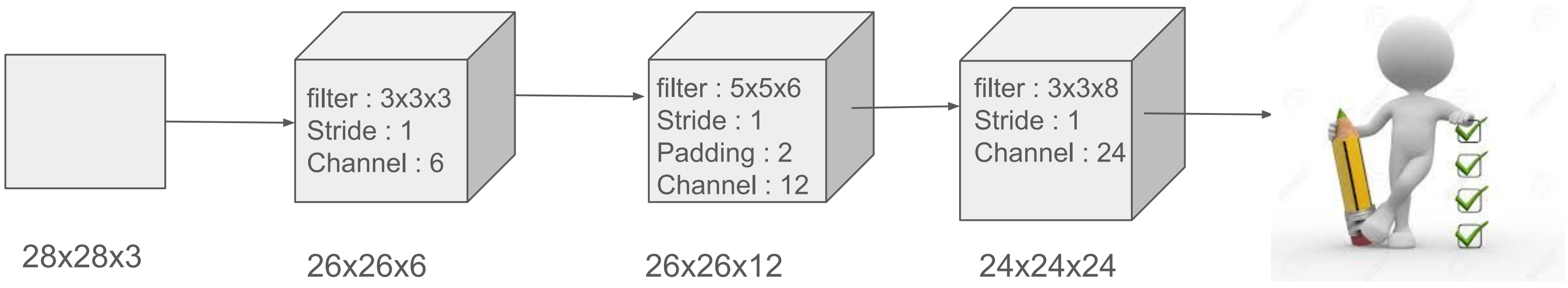


Feature map size in every layer ?

$FM = (N+2P-F)/S+1$

How many weights and biases ?

Number of weights=Number of filters×(Filter height×Filter width×Input channels)
Number of biases=Number of filters



1 neuron= multiple output

Conv layer 1: 6x(3x3x3)+6=168
Conv layer 2: 12x(5x5x6)+12=1812
Conv layer 3: 24x(3x3x12)+24=2616

Total learnable parameters: 4596

Pooling Layer in CNN

Down sampling

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

Pool size = (2,2)
Stride = 2

6	5
8	5

Max pooling

Pooling Layer in CNN

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

Pool size = (2,2)
Stride = 2

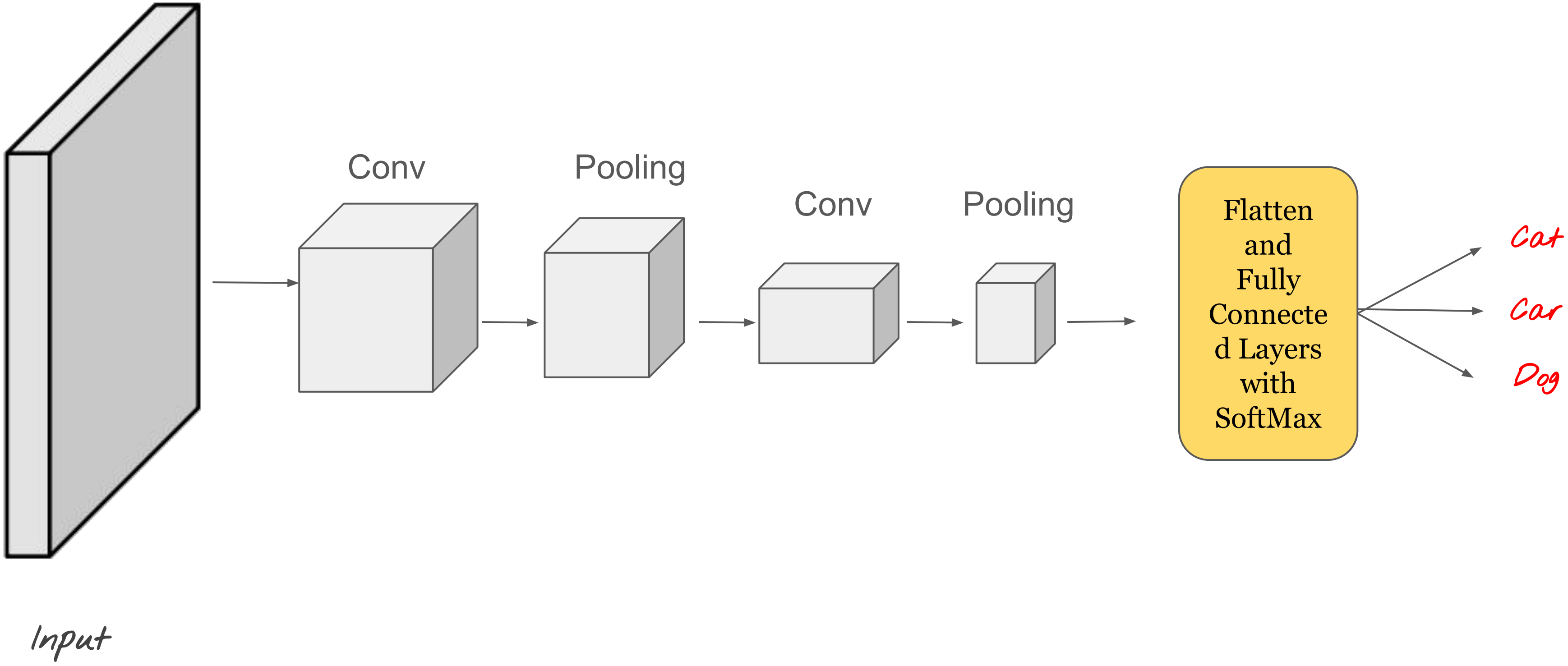
4	4.2
5.5	3

Average/Mean pooling

Fully connected layer or Dense layer

- We need to add a flatten layer to create a 1D array from the last pooling layer

- The Dense layer is an important component in convolutional neural networks (CNNs) because it transforms the multi-dimensional output of the convolutional and pooling layers into a one-dimensional array, which can then be fed into fully connected (dense) layers for classification or regression tasks.


Conv
Pooling
Conv
Pooling
Flatten and Fully Connected Layers with SoftMax
Cat
Car
Dog
Input

CNN Architecture for CIFAR-10

1. **Input Layer**: 32x32x3 (color image)
2. **Conv2D Layer**: 32 filters, kernel size 3x3, stride 1, padding 'same'
3. **MaxPooling2D Layer**: pool size 2x2, stride 2

4. **Conv2D Layer**: 64 filters, kernel size 3x3, stride 1, padding 'same'
5. **MaxPooling2D Layer**: pool size 2x2, stride 2

6. **Flatten Layer**
7. **Dense Layer**: 128 units
8. **Dense Layer**: 10 units (output layer for classification)

1. Conv2D Layer (32 filters, 3x3 kernel)

- **Input shape**: 32x32x3
- **Number of filters**: 32
- **Kernel size**: 3x3
- **Padding**: 'same' (output shape will be 32x32x32)

2. MaxPooling2D Layer (2x2 pool size, stride 2)

- **Input shape**: 32x32x32
- **Pool size**: 2x2
- **Stride**: 2
- **Output shape**: 16x16x32

3. Conv2D Layer (64 filters, 3x3 kernel)

- **Input shape**: 16x16x32
- **Number of filters**: 64
- **Kernel size**: 3x3
- **Padding**: 'same' (output shape will be 16x16x64)

4. MaxPooling2D Layer (2x2 pool size, stride 2)

- **Input shape**: 16x16x64
- **Pool size**: 2x2
- **Stride**: 2
- **Output shape**: 8x8x64

5. Flatten Layer

- **Input shape**: 8x8x64
- **Output shape**: 4096 (8 \times 8 \times 64)

6. Dense Layer (128 units)

- **Input shape**: 4096
- **Number of units**: 128

How learning happens in CNN?

- Forward propagation
- Backpropagation
  - Gradient descent

ANN	vs	Convolution
Requires one dimensional input, no neighbourhood info		Can work with multi-dimensional data, uses neighbourhood info
Large number of weights		Smaller number of weights
One output per neuron		Multiple outputs per neuron
Less calculations		More calculations

Good materials !

https://medium.com/advanced-deep-learning/cnn-operation-with-2-kernels-resulting-in-2-feature-mapsunderstanding-the-convolutional-filter-c4aad26cf32

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

https://caffe.berkeleyvision.org

https://distill.pub/2017/feature-visualization/

https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

https://poloclub.github.io/cnn-explainer/

Backpropagation in CNN

https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Visualize CIFAR10 training

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html