Convolution Neural Networks

An Introduction to its concept and its code

ADiagnosis Team

Our Team:

Prof. Kourosh Parand
Board Director & Co-founder of the **NMS** Company

Elmira Mirzabeigi
CEO & Co-founder of the **NMS** Company

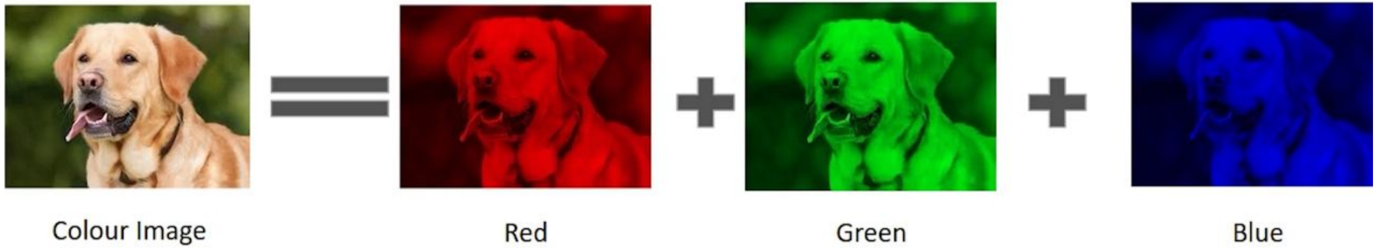
Sepehr Rezaee
CTO of the **NMS** Company

Amir Hossein Karami
ML Engineer of the **NMS** Company

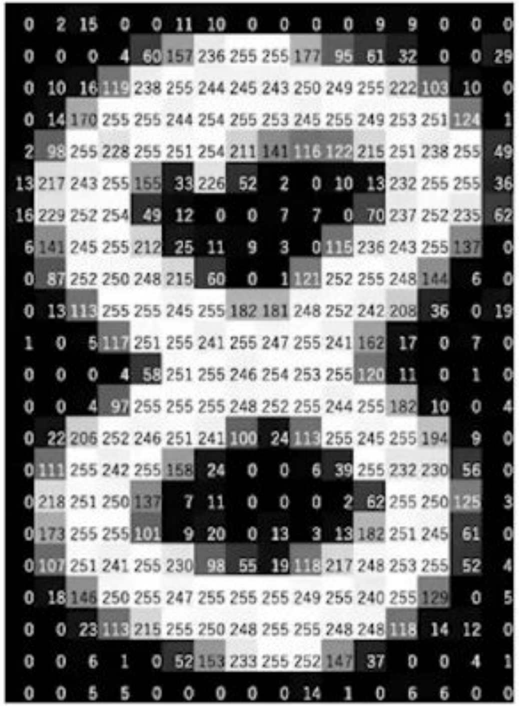
Mobina Mirzabeigi
Full-Stack Developer of the **NMS** Company

What is an image in terms of a computer

Image is a set of numbers



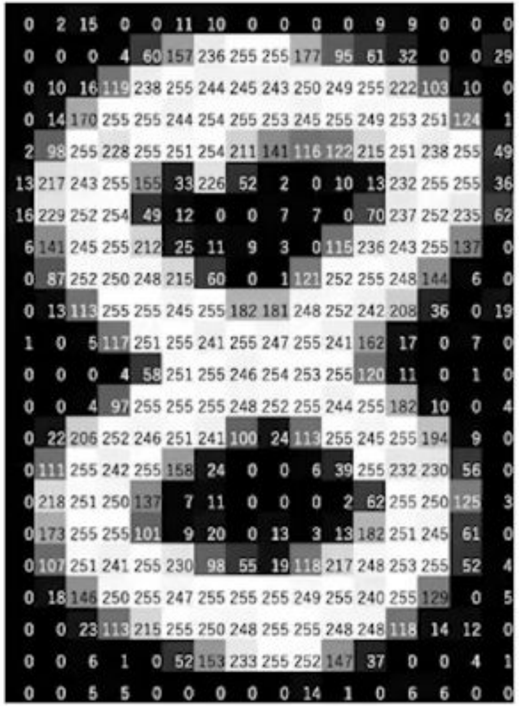
Tasks we can apply on images



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

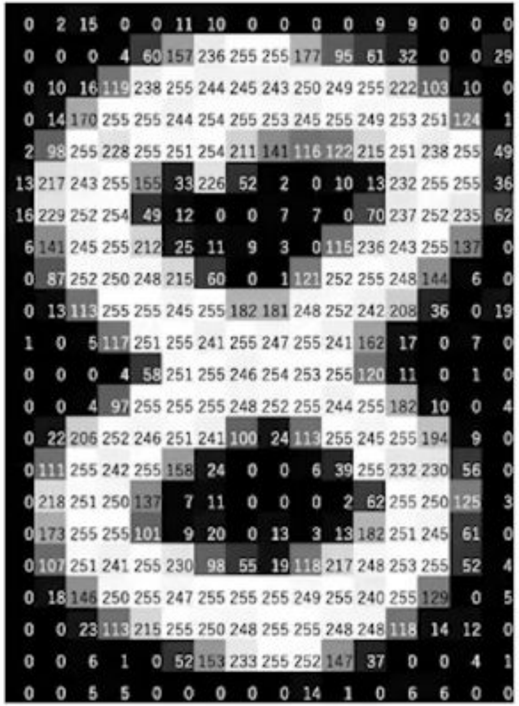
- **Regression:** Output variable takes continues value.
- **Classification:** Output variable takes class label.

Tasks: Classification

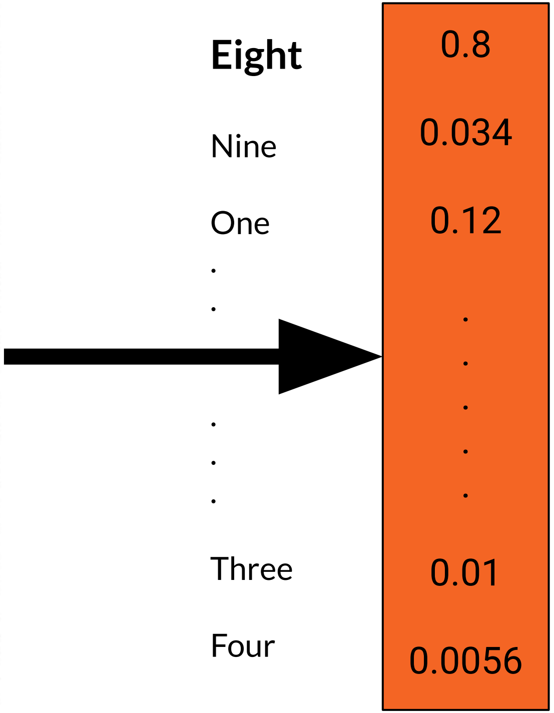


0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Tasks: Classification



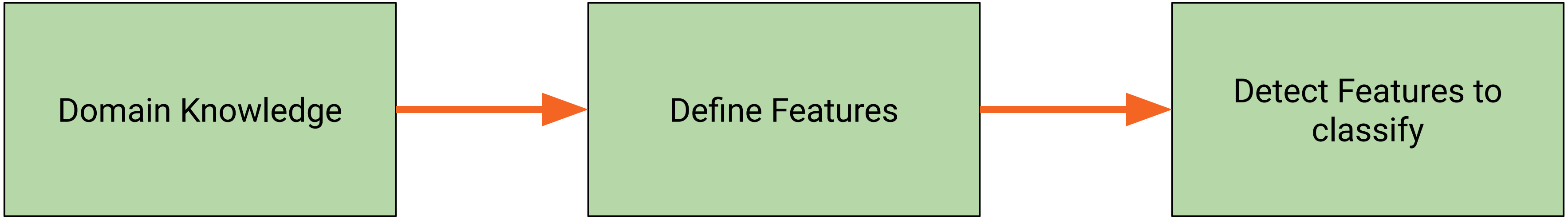
0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0



Eight	0.8
Nine	0.034
One	0.12
.	.
.	.
	.
	.
.	.
.	
.	
Three	0.01
Four	0.0056

Features of images to detect

Manual Feature Extractions



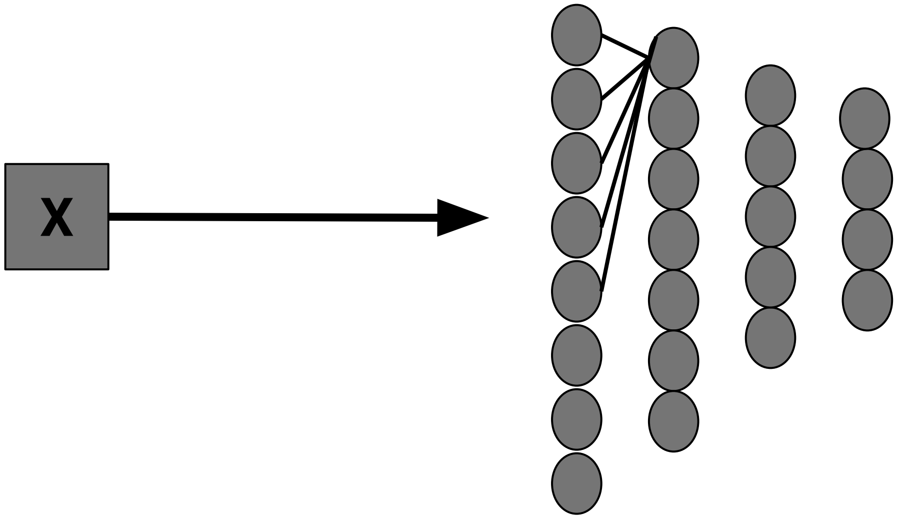
Domain Knowledge

Define Features

Detect Features to classify

Learning the Features

Learning with Fully Connected Perceptrons

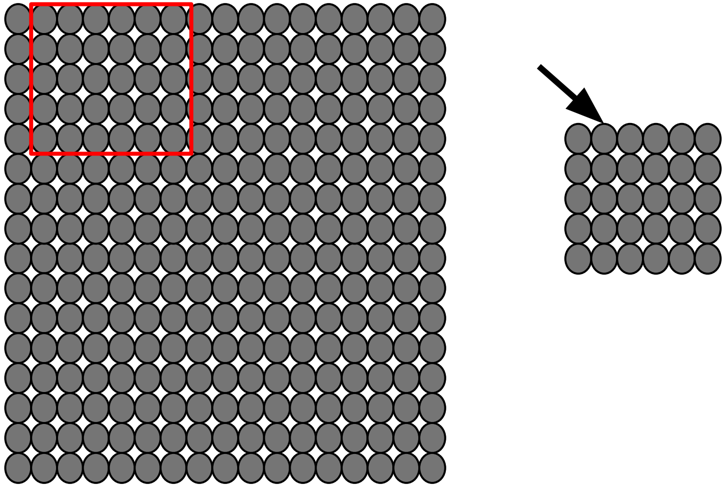


Learning with Fully Connected Perceptrons

Image with size of (100, 100, 1) we need (10000) perceptrons in first layer

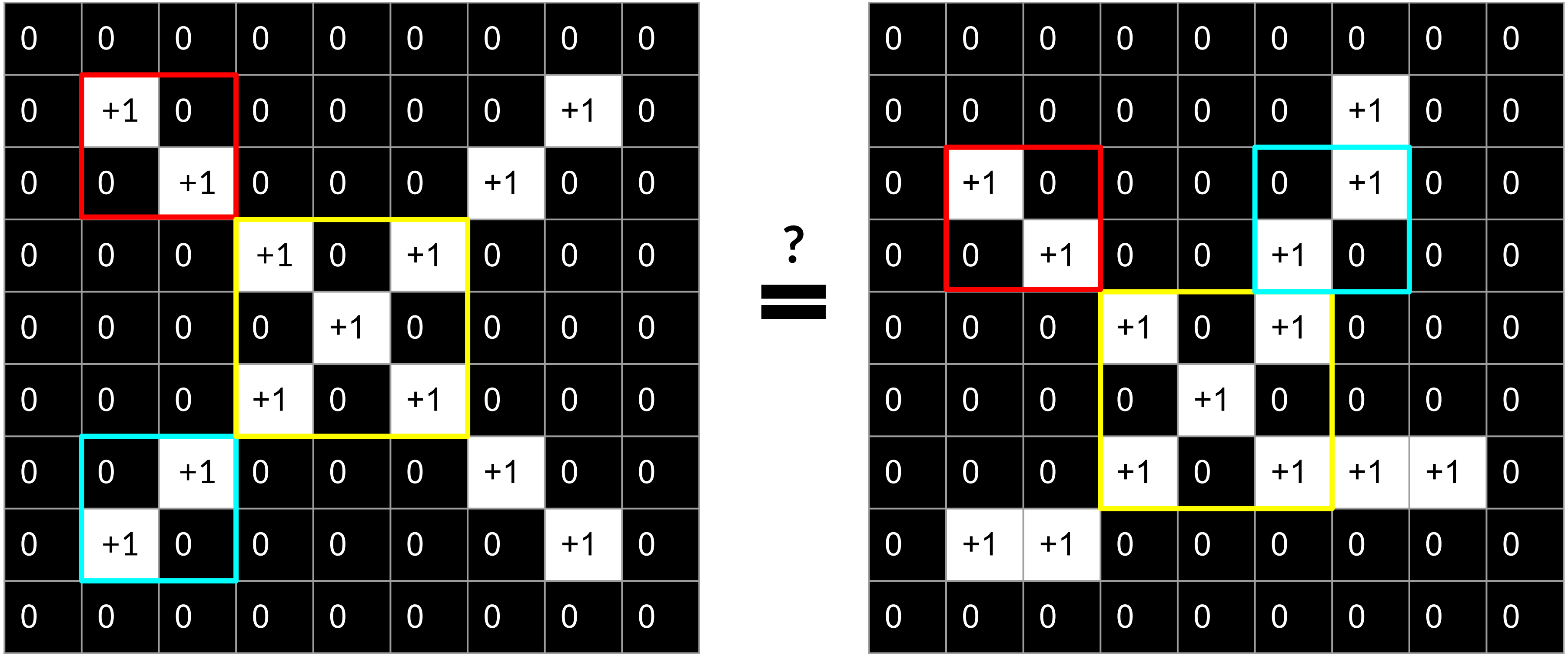
We ignore each spatial patterns

Use Spatial Structure



1. Apply a set of weights - a filter - to extract **local feature**
2. Use **multiple filters** to extract different features
3. **Spatially share** parameters of each filter

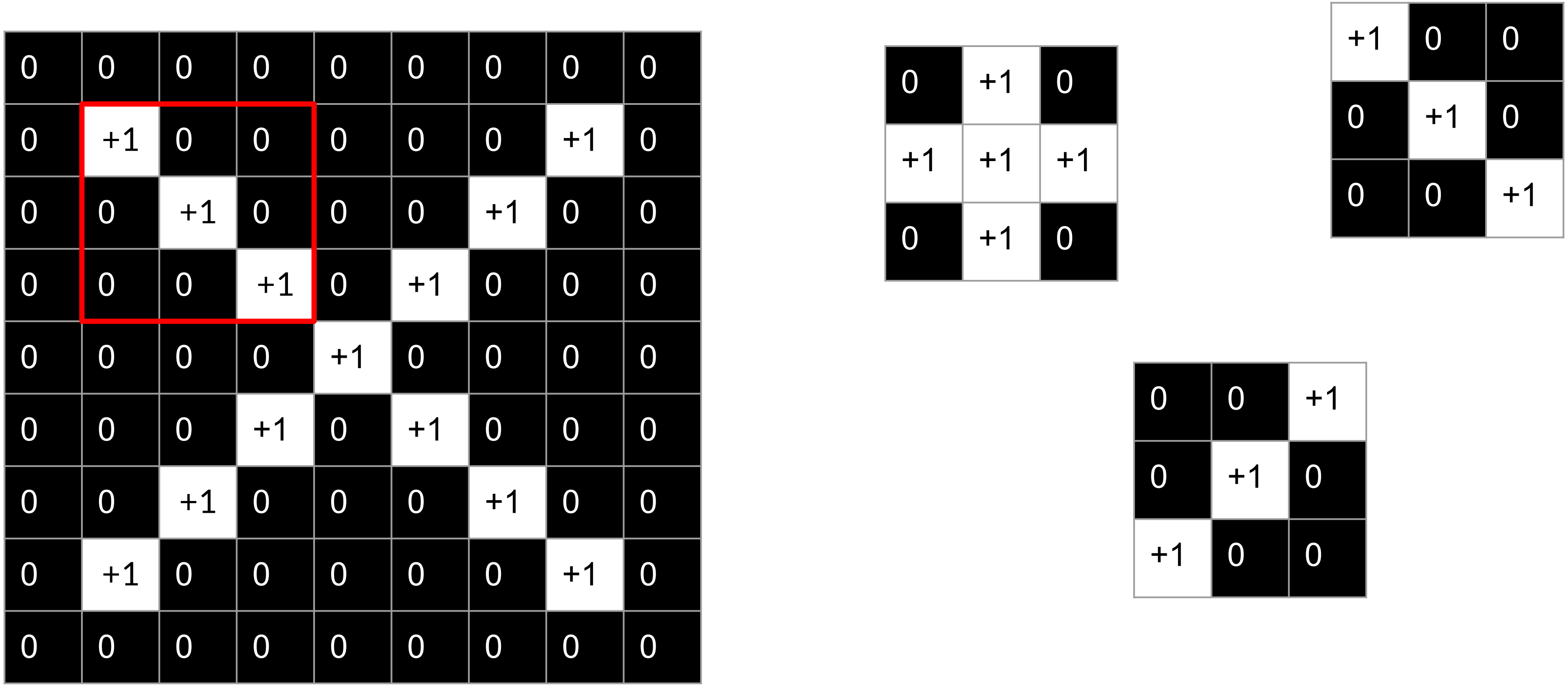
Convolution and Feature Extraction



0	0	0	0	0	0	0	0	0
0	+1	0	0	0	0	0	+1	0
0	0	+1	0	0	0	+1	0	0
0	0	0	+1	0	+1	0	0	0
0	0	0	0	+1	0	0	0	0
0	0	0	+1	0	+1	0	0	0
0	0	+1	0	0	0	+1	0	0
0	+1	0	0	0	0	0	+1	0
0	0	0	0	0	0	0	0	0

$\overset{?}{=}$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	+1	0	0
0	+1	0	0	0	0	+1	0	0
0	0	+1	0	0	+1	0	0	0
0	0	0	+1	0	+1	0	0	0
0	0	0	0	+1	0	0	0	0
0	0	0	+1	0	+1	+1	+1	0
0	+1	+1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0	0
0	+1	0	0	0	0	0	+1	0
0	0	+1	0	0	0	+1	0	0
0	0	0	+1	0	+1	0	0	0
0	0	0	0	+1	0	0	0	0
0	0	0	+1	0	+1	0	0	0
0	0	+1	0	0	0	+1	0	0
0	+1	0	0	0	0	0	+1	0
0	0	0	0	0	0	0	0	0

0	+1	0
+1	+1	+1
0	+1	0

+1	0	0
0	+1	0
0	0	+1

0	0	+1
0	+1	0
+1	0	0

Common Filters

1/16

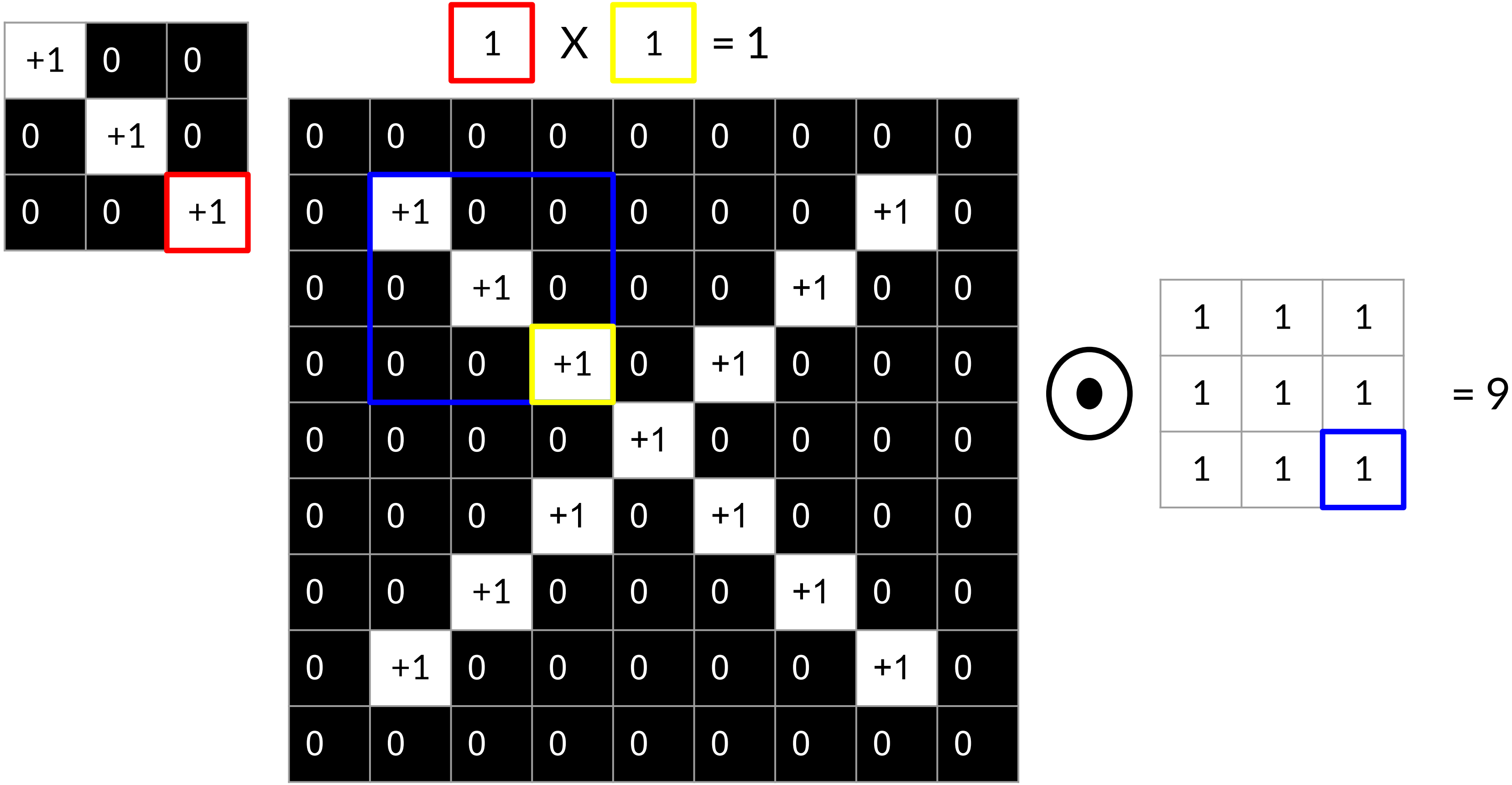
1	2	1
2	4	2
1	2	1

Gaussian Smoothing Filter

0	-1	0
-1	5	-1
0	-1	0

Gamma Filter

The Convolution Operator



$\boxed{1} \times \boxed{1} = 1$

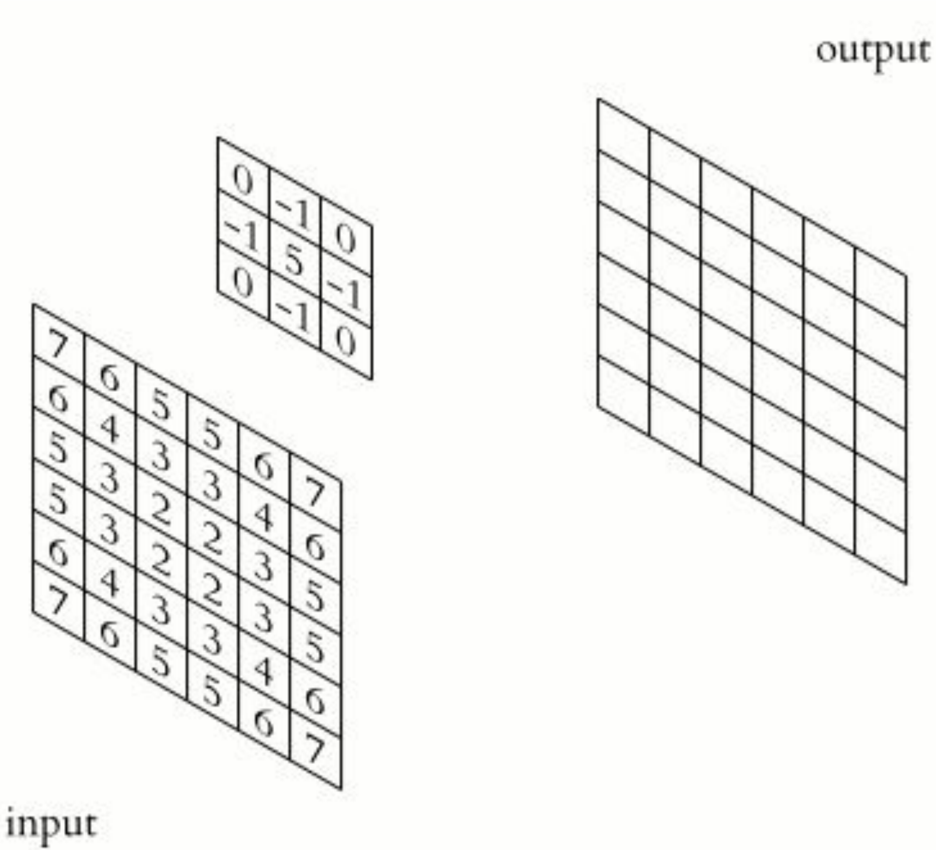
+1	0	0
0	+1	0
0	0	+1

0	0	0	0	0	0	0	0	0
0	+1	0	0	0	0	0	+1	0
0	0	+1	0	0	0	+1	0	0
0	0	0	+1	0	+1	0	0	0
0	0	0	0	+1	0	0	0	0
0	0	0	+1	0	+1	0	0	0
0	0	+1	0	0	0	+1	0	0
0	+1	0	0	0	0	0	+1	0
0	0	0	0	0	0	0	0	0

$\odot$

1	1	1
1	1	1
1	1	1

= 9



0	-1	0
-1	5	-1
0	-1	0

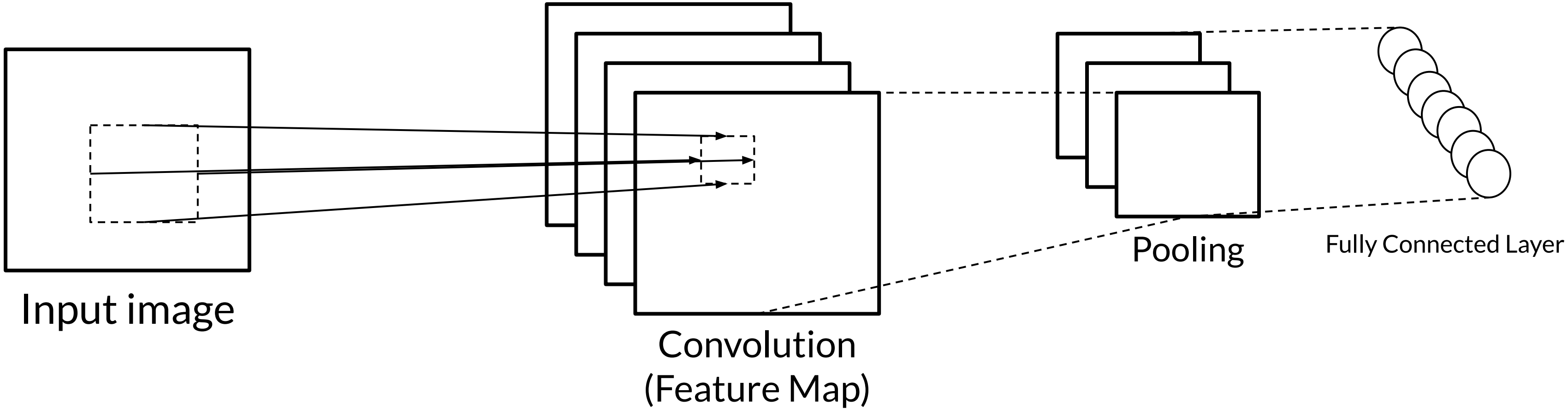
7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7

Feature Extraction with Convolution Operator

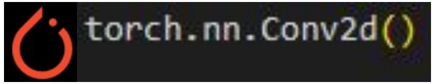
1. Apply a set of weights - a filter - to extract **local feature**
2. Use **multiple filters** to extract different features
3. **Spatially share** parameters of each filter

Convolution Neural Network

CNNs for Classification

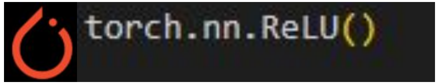


1. **Convolution:** apply filters to generate feature maps



```
torch.nn.Conv2d()
```

2. **Non-Linear activations:** e.g. ReLU

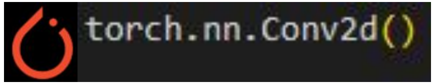


```
torch.nn.ReLU()
```

3. **Pooling:** Downstream operation to each feature map

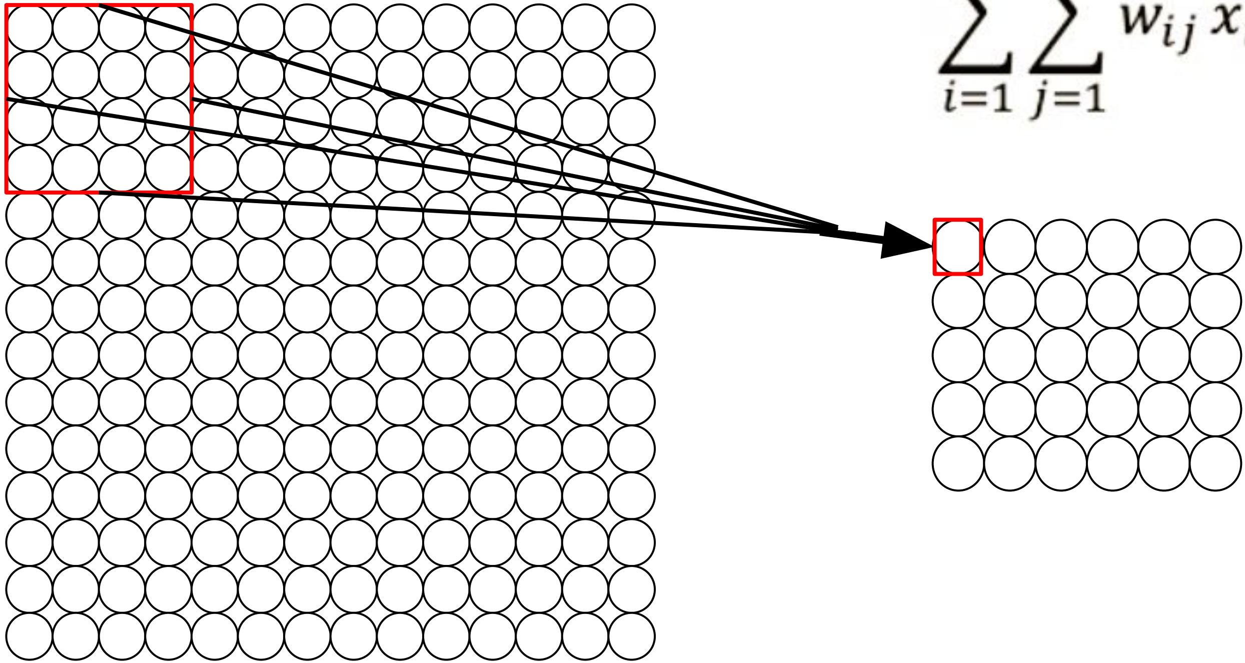
```
torch.nn.MaxPool2d()
```

Local Connectivity

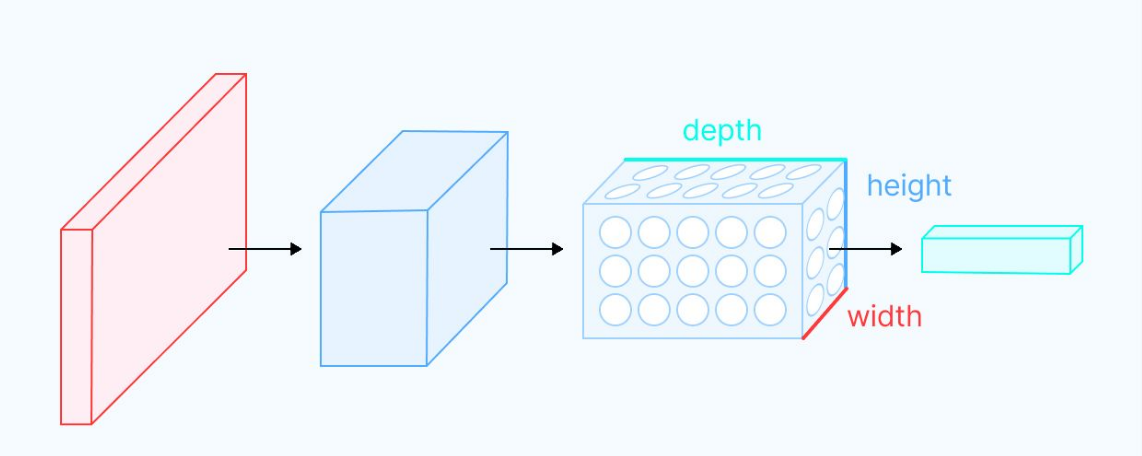


```
torch.nn.Conv2d()
```

$$\sum_{i=1}^{4}\sum_{j=1}^{4} w_{ij}\, x_{i+p,j+q} + b$$

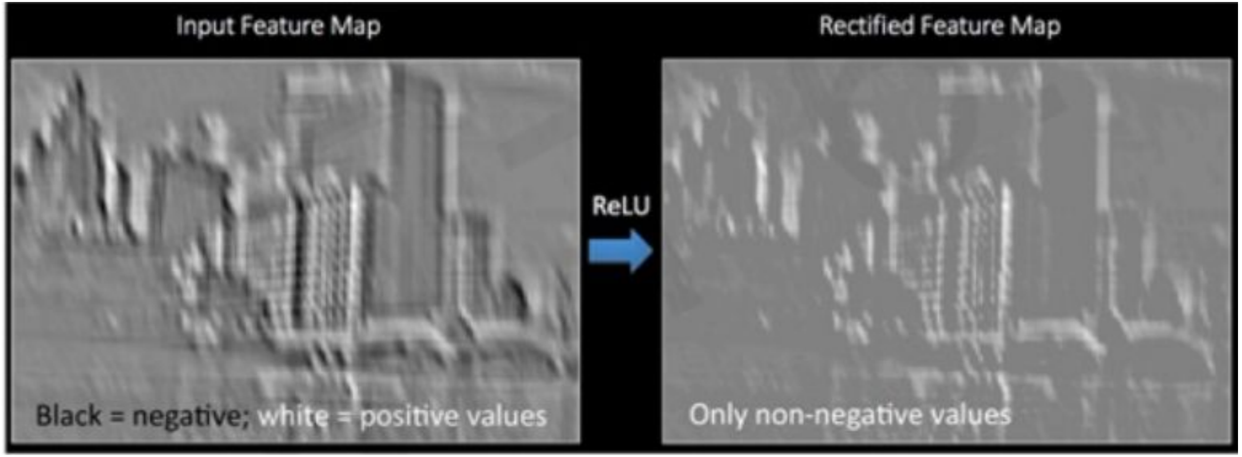


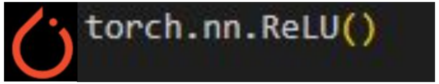
Spatial arrangement of output volume



Depth represent the number of filter applied

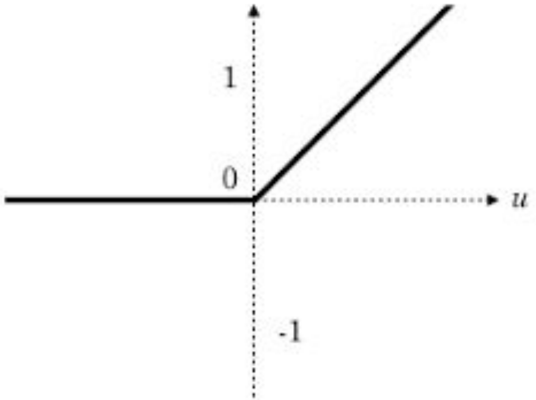
Introducing the non-linearity





```
torch.nn.ReLU()
```

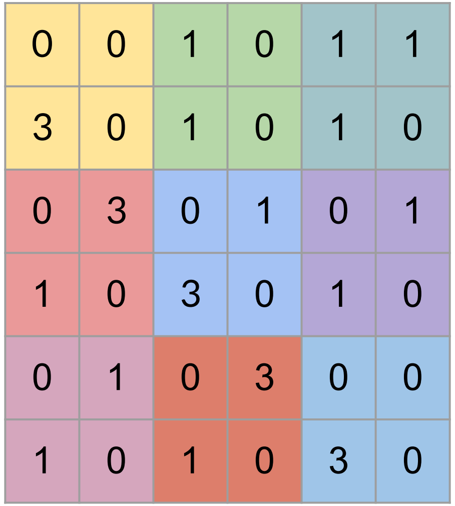
$$f(u) = \max(0, u)$$



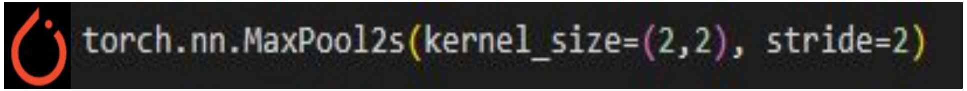
- Apply **after** each convolution operation
- **Pixel-by-pixel** replace all negative values by zero

Pooling

- Reduced dimensionality
- Sparse invariance

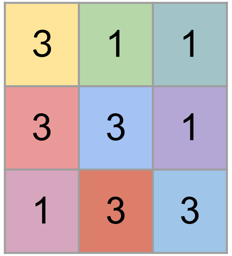


0	0	1	0	1	1
3	0	1	0	1	0
0	3	0	1	0	1
1	0	3	0	1	0
0	1	0	3	0	0
1	0	1	0	3	0



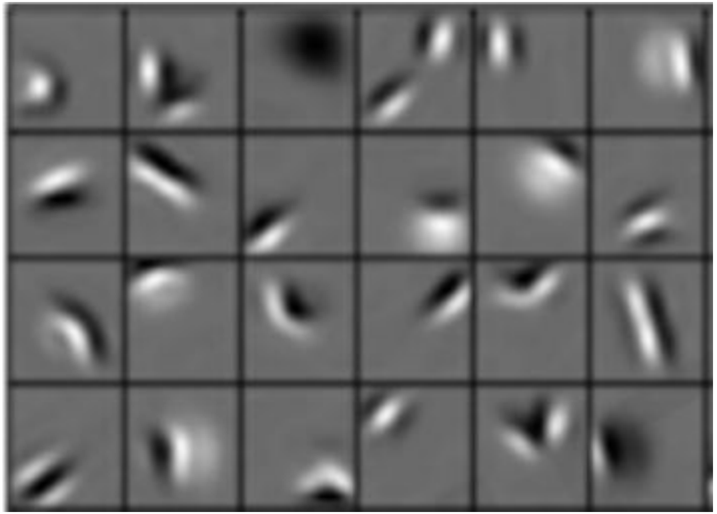
```
torch.nn.MaxPool2s(kernel_size=(2,2), stride=2)
```

Max Pool



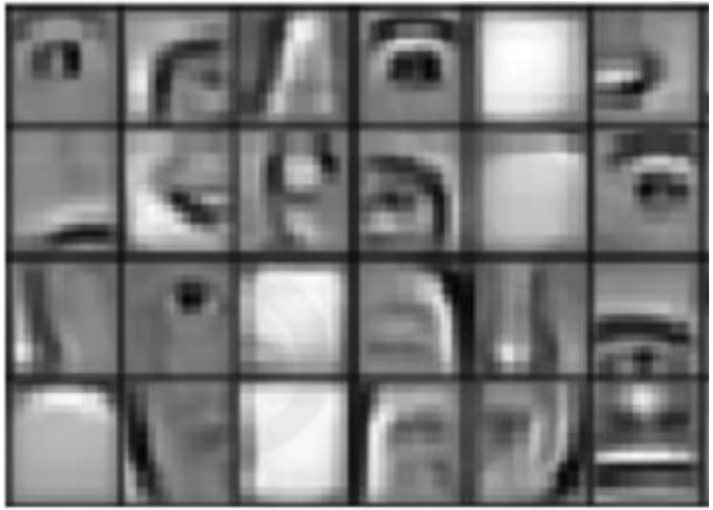
3	1	1
3	3	1
1	3	3

Representation Learnt Features in CNN



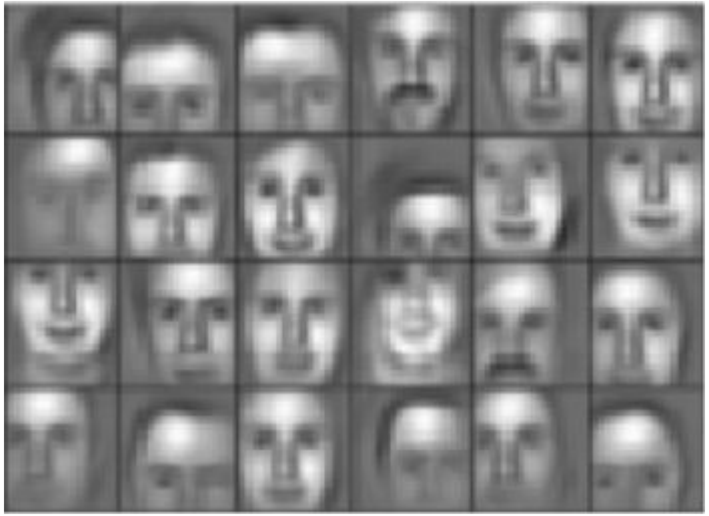
For example: after one Conv layer, we have just lines in the image. We call them **Low-level features obtained** from **pixel-level**

Representation Learnt Features in CNN



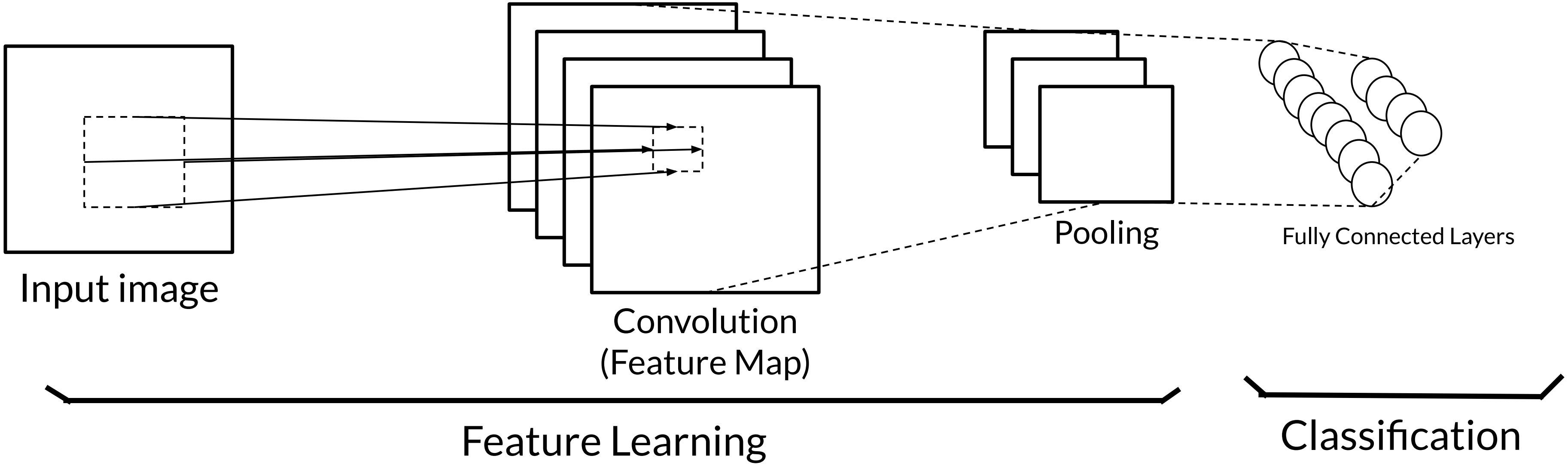
For example: after two Conv layer, we have just part of face like **eyes and noses** in the image. We call them **Mid-level features obtained** from **Feature Maps.**

Representation Learnt Features in CNN



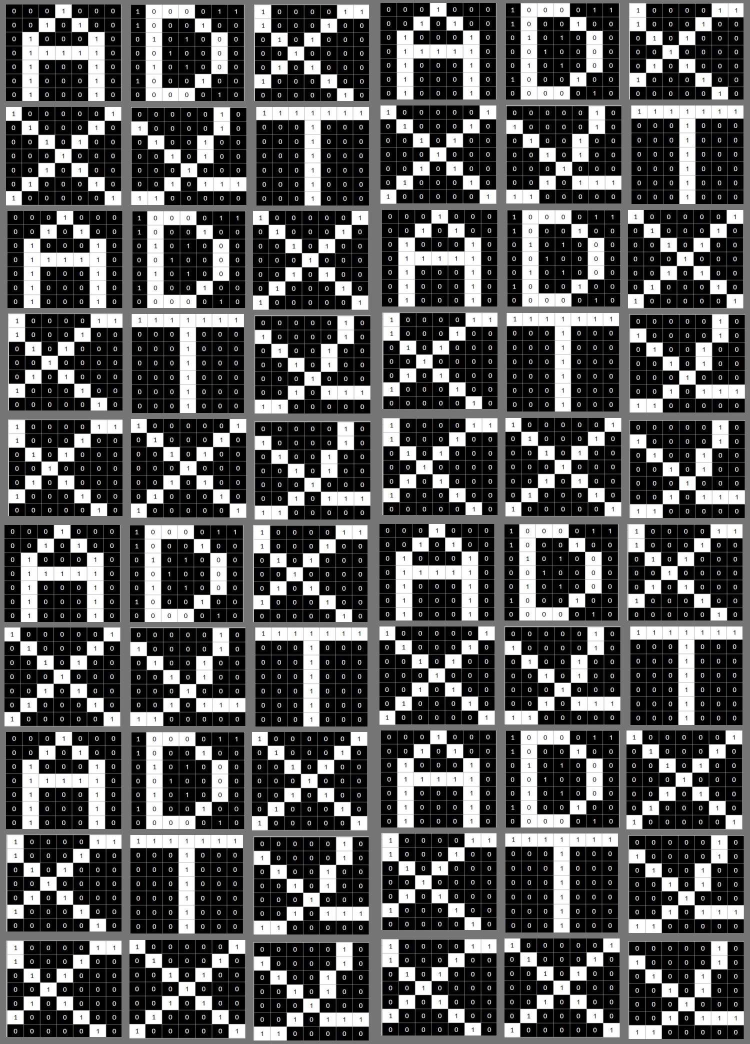
For example: after three Conv layer, we have just have **some faces**. We call them **High-level features** that are useful for **Classification** task

CNNs for Classification



1. Learn features in input images through **Convolution**
2. Introduce **non-linearity** through **activation function** (real-world patterns)
3. Reduce dimensionality and preserve spatial invariance with **Pooling**

An example

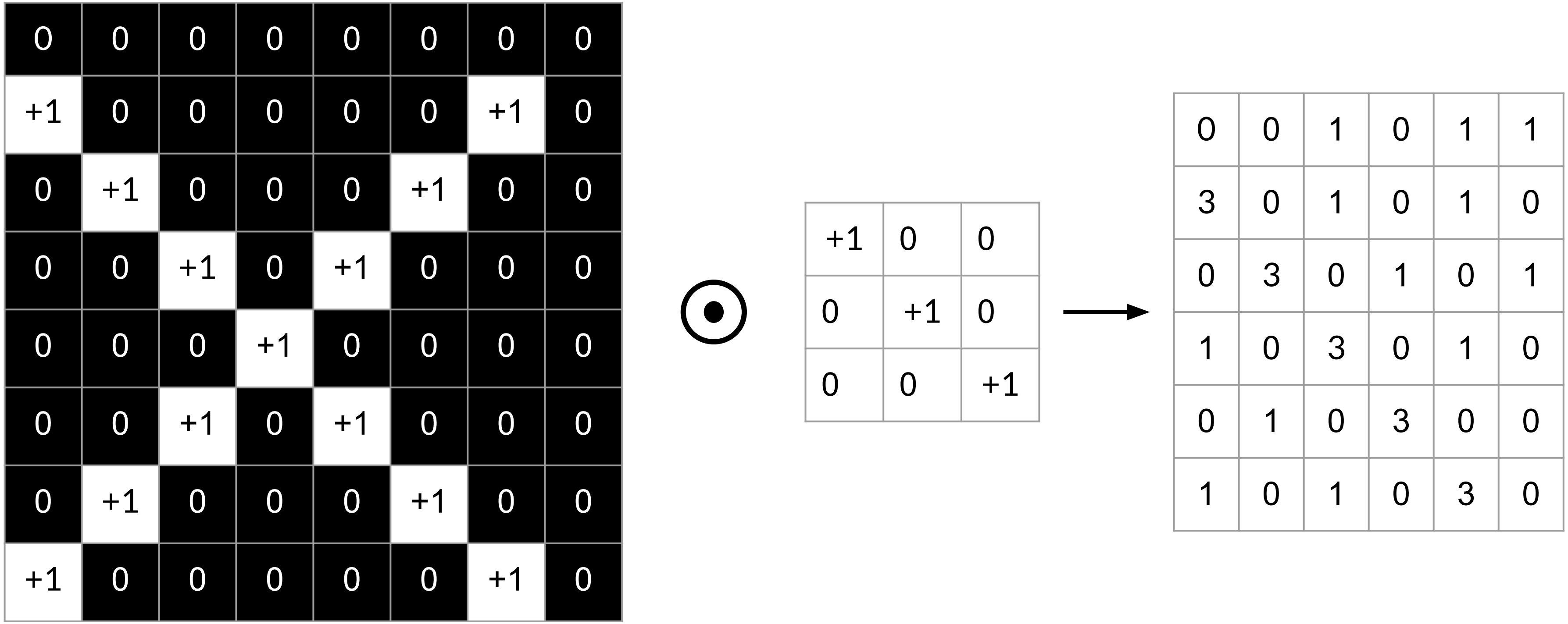


1	0	0	0	0	1	1
1	0	0	0	1	0	0
0	1	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
1	0	0	0	1	0	0
0	0	0	0	0	1	0

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

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

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



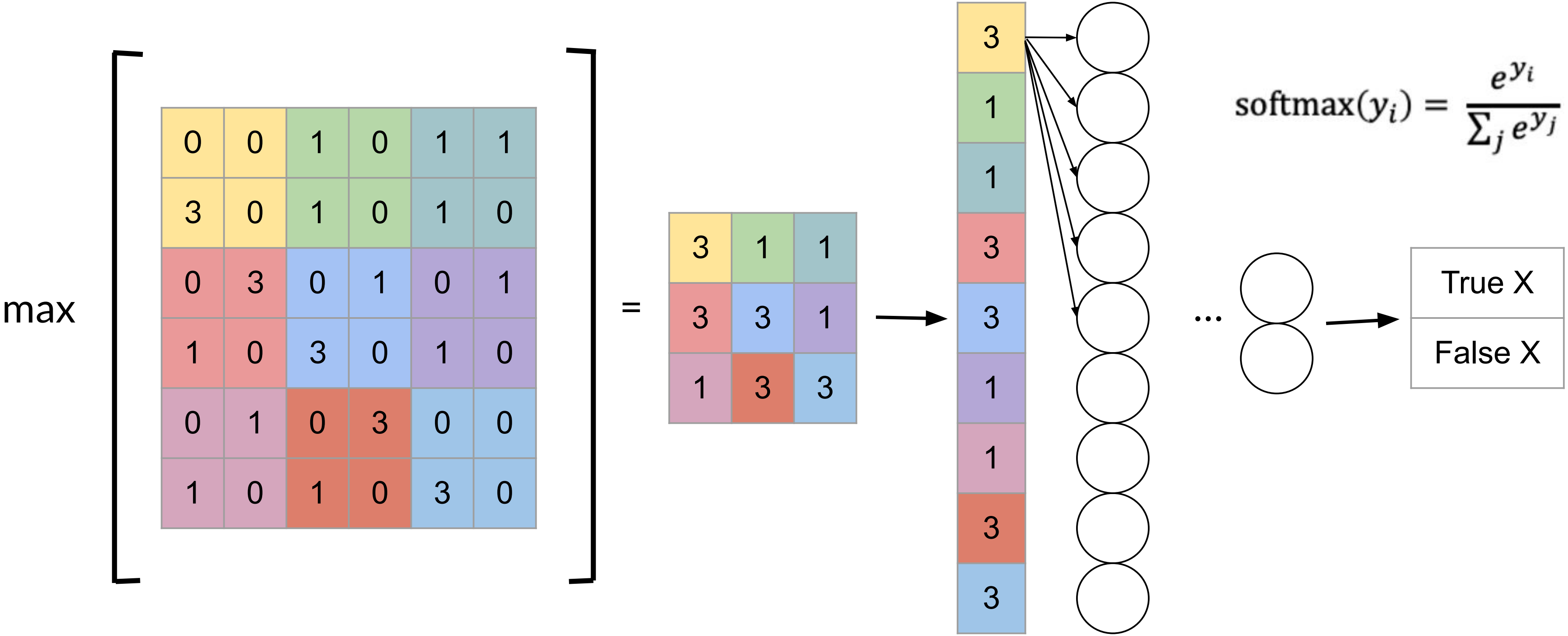
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	+1	0
0	+1	0	0	0	+1	0	0
0	0	+1	0	+1	0	0	0
0	0	0	+1	0	0	0	0
0	0	+1	0	+1	0	0	0
0	+1	0	0	0	+1	0	0
+1	0	0	0	0	0	+1	0

⊙

+1	0	0
0	+1	0
0	0	+1

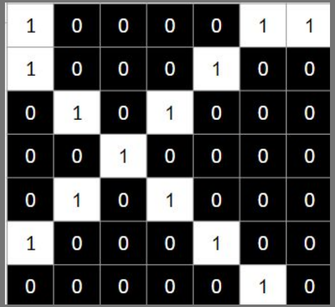
→

0	0	1	0	1	1
3	0	1	0	1	0
0	3	0	1	0	1
1	0	3	0	1	0
0	1	0	3	0	0
1	0	1	0	3	0



$$\max\begin{bmatrix}0&0&1&0&1&1\\3&0&1&0&1&0\\0&3&0&1&0&1\\1&0&3&0&1&0\\0&1&0&3&0&0\\1&0&1&0&3&0\end{bmatrix}=\begin{bmatrix}3&1&1\\3&3&1\\1&3&3\end{bmatrix}\rightarrow\begin{bmatrix}3\\1\\1\\3\\3\\1\\1\\3\\3\end{bmatrix}$$

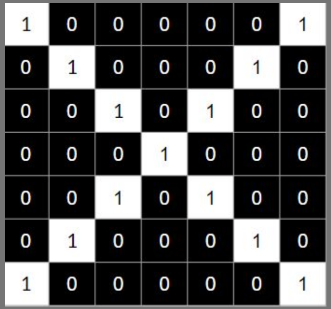
$$\text{softmax}(y_i)=\frac{e^{y_i}}{\sum_j e^{y_j}}$$



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

True X = 0.95

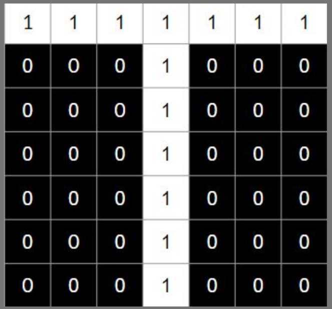
False X = 0.05



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

True X = 0.99

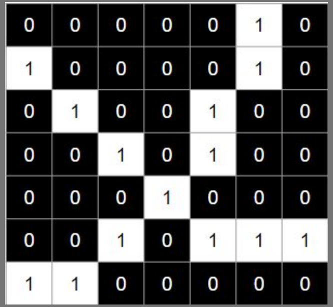
False X = 0.01



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

True X = 0.02

False X = 0.98



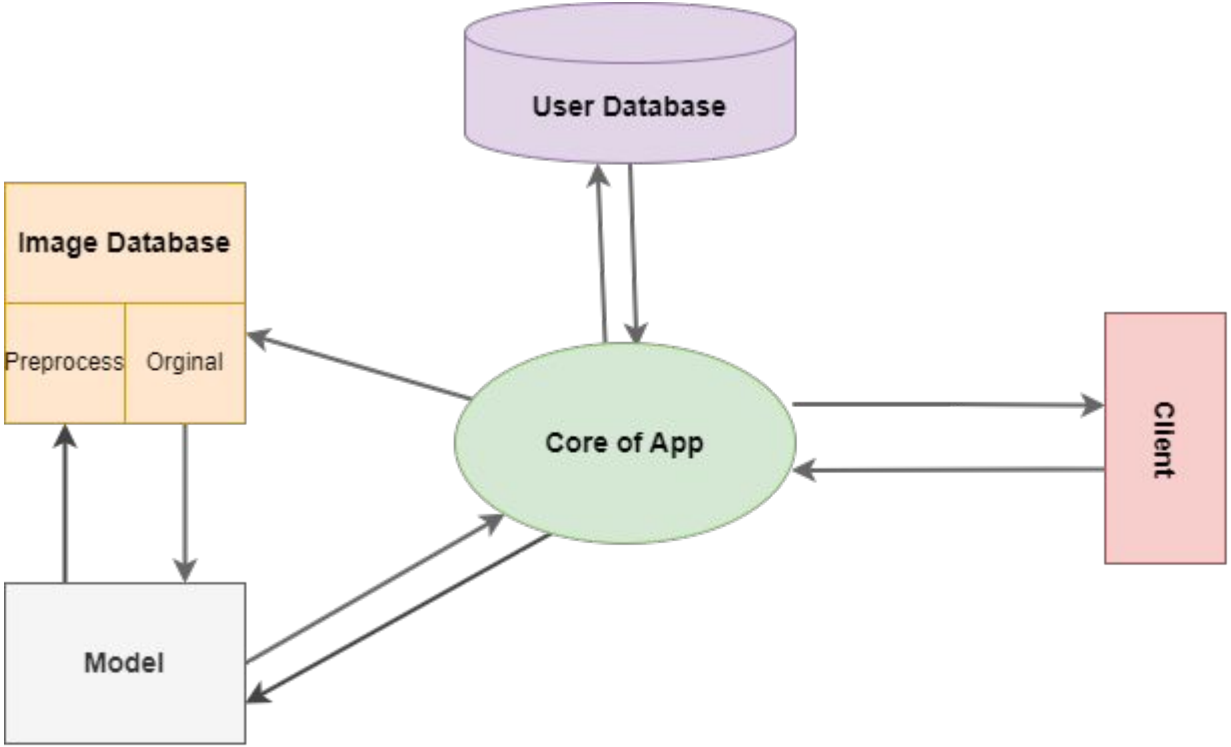
0	0	0	0	0	1	0
1	0	0	0	0	1	0
0	1	0	0	1	0	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	1
1	1	0	0	0	0	0

True X = 0.90

False X = 0.10

Our work on Alzheimer

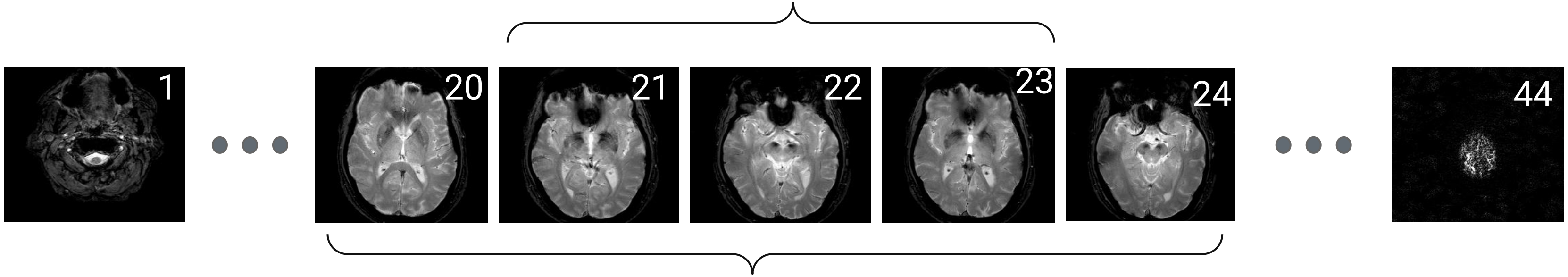
Our Software



The dataset we used

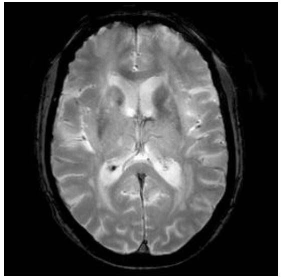
Each MRI Dicom file contains 44 slices of brain. The three slices in the middle of the brain are used

Used Slices

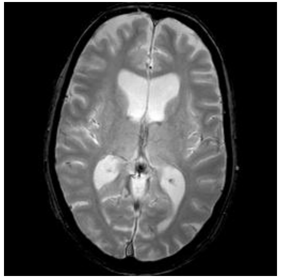


Able to be used

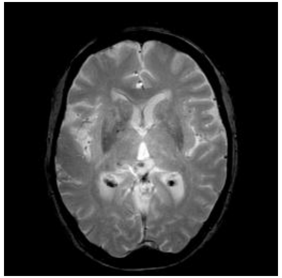
The labels of our dataset



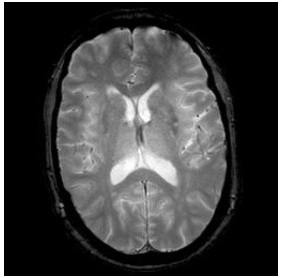
CN



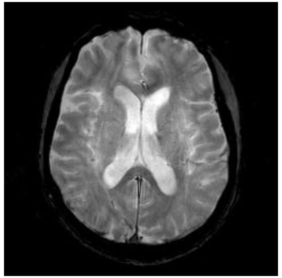
SMC



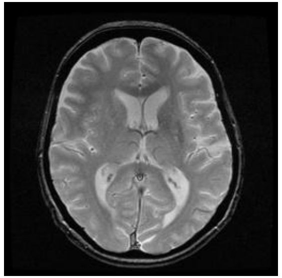
EMCI



MCI

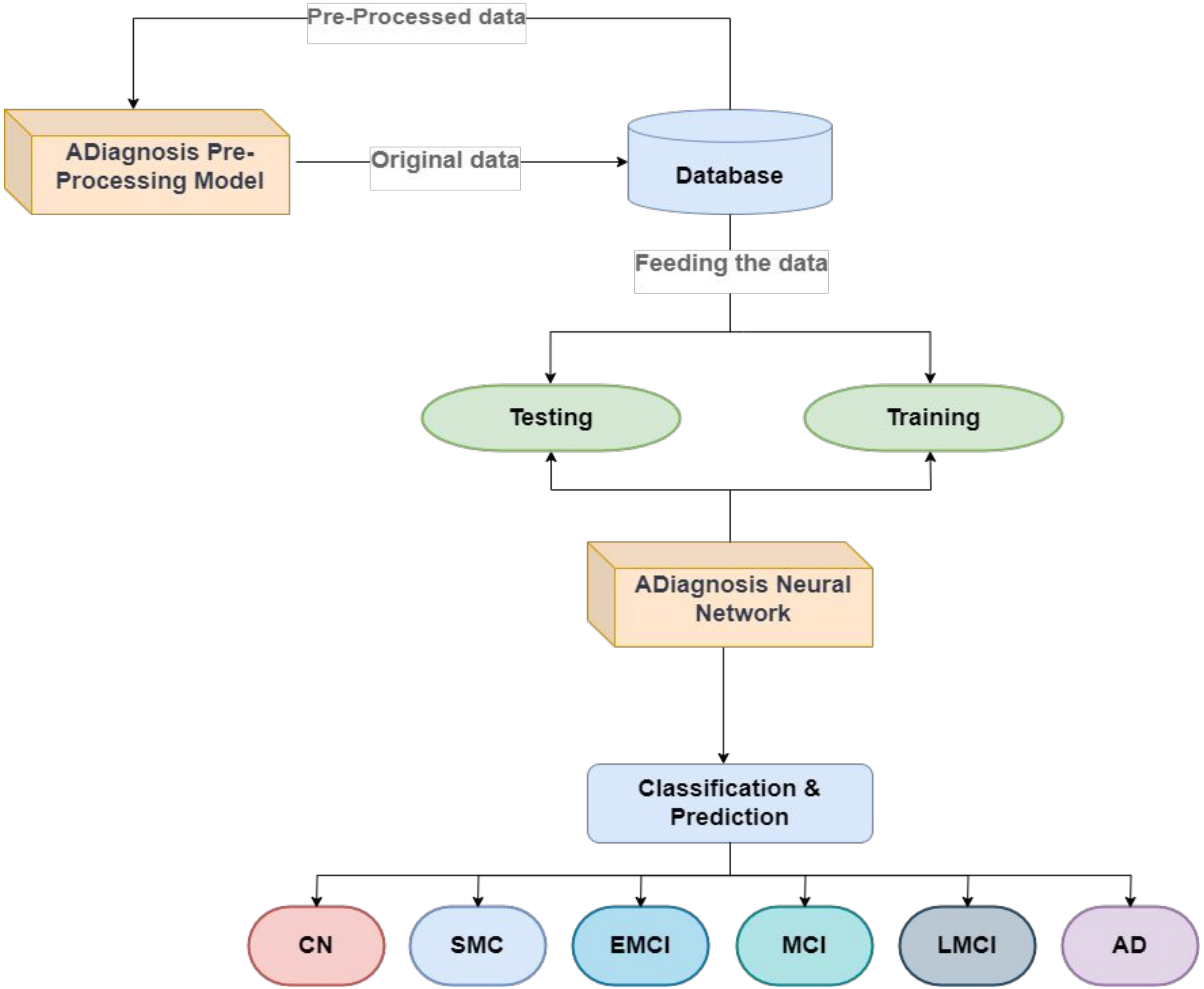


LMCI



AD

The **pipeline** of **Preprocessing** and **Classification**



Thanks for your attentions!!