Deep Learning

Deep Learning

- Introduction to Deep Learning
- Introduction to Convolutional Neural Network (CNN)
- CNN Architecture and Layers
- Building Simple CNN Model for Classification
- Training and Testing the CNN Model

Introduction

In the past few years, *Deep Learning* has generated much excitement in Machine Learning

Many breakthrough results in *speech recognition, computer vision and text processing*

unstructured data.

What is Deep Learning?

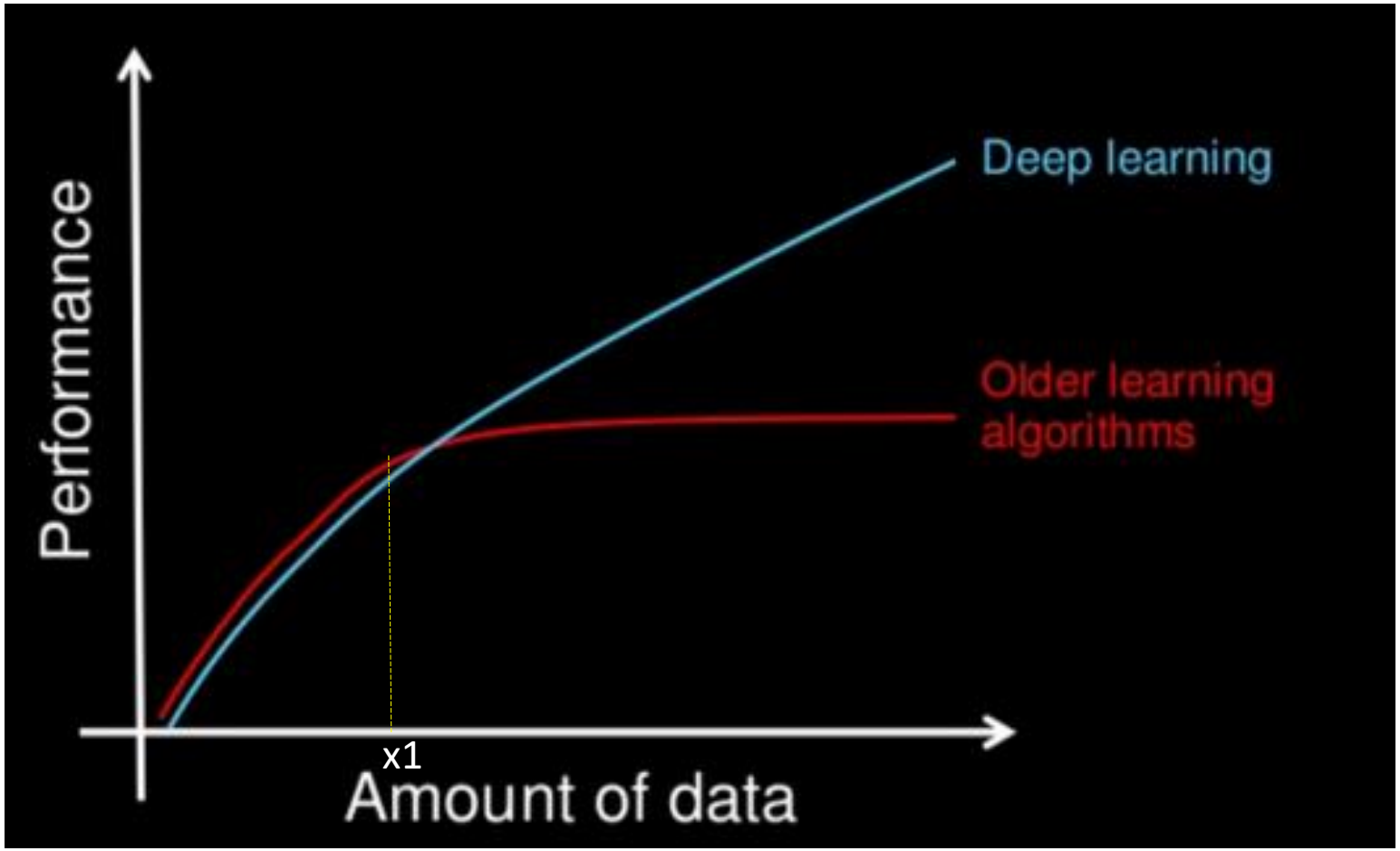
*Deep refers to the **number of layers***

Deep Learning is a deep neural networks generally.

“very large neural networks & huge amounts of data”

Why Deep Learning?

- Performance…



Important Property of Neural Networks

Results get better with

more data +

bigger models +

more computation

Deep Learning is successful…

… in recent years due to the availability of **inexpensive, parallel hardware** (GPUs, computer clusters) and **massive amounts of data**.

Deep Learning…

“ …is a pipeline of modules all of which are trainable.

… has multiple stages in the process of recognizing an object and

all of those stages are part of the training”

Types of deep learning models

- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Autoencoders and variational autoencoders
- Generative adversarial networks (GANs)
- Diffusion models
- Transformer models

Applications (1): Sentiment Analysis

The process of identifying and categorizing **opinions expressed in a piece of text**, to determine whether the writer's attitude towards a particular topic, product, etc. is **positive, negative, or neutral**.

The best way to hope for any chance of enjoying this film is by lowering your expectations.

The show starts out as competent but unremarkable and gradually grows into something of considerable power.

Applications (2): Question Answering

Yesterday Joy travelled to school.

Yesterday Rai went back to the beach.

This morning Joy travelled to market.

Bill went back to the cinema yesterday.

Rai went to the Hotel this morning.

Joy went back to the dining room this afternoon.

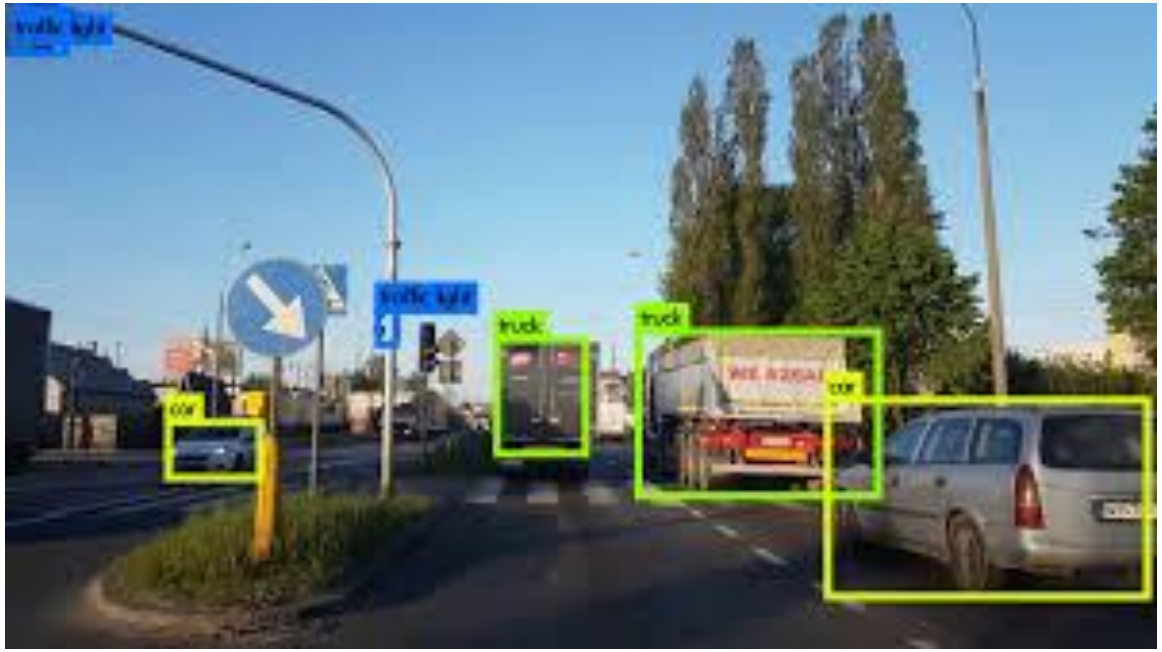
Question: Where was Rai before went to Hotel?

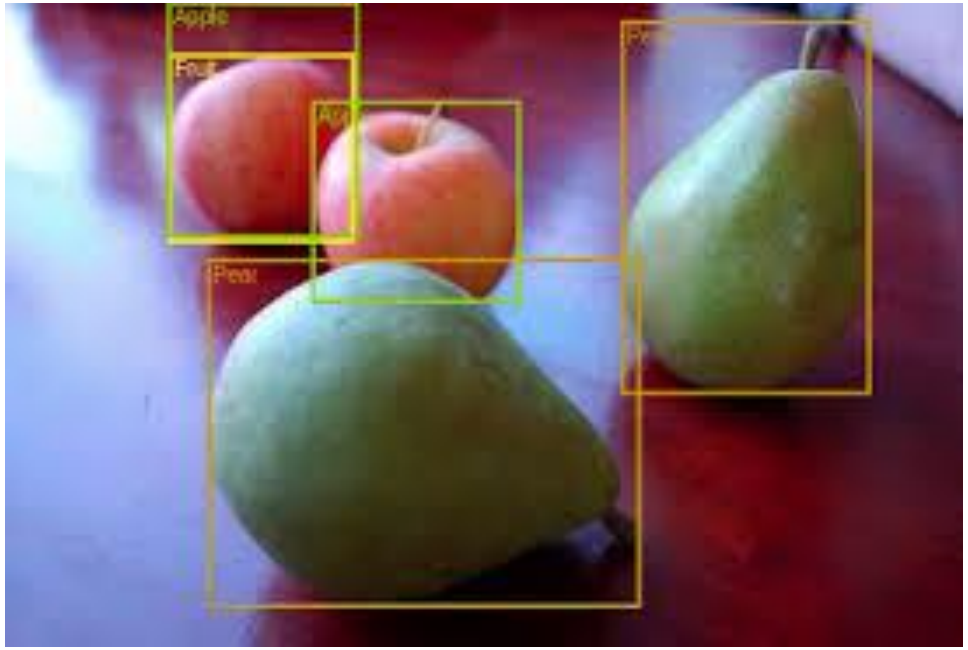
Applications (3): Language Translation

can a man live without food?

ಒಬ್ಬ ಮನುಷ್ಯನು ಆಹಾರವಿಲ್ಲದೆಯೇ ಜೀವಿಸಬಹುದೇ?

Applications (4): Object Detection





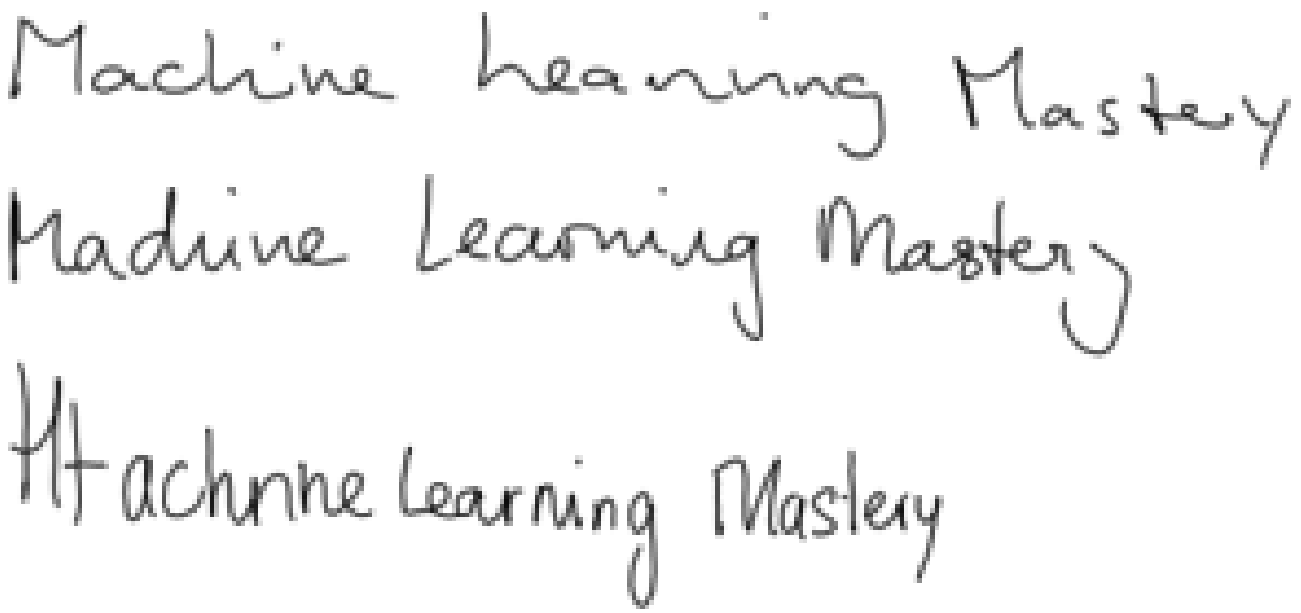
Applications (4): Automatic Caption Generation

"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."

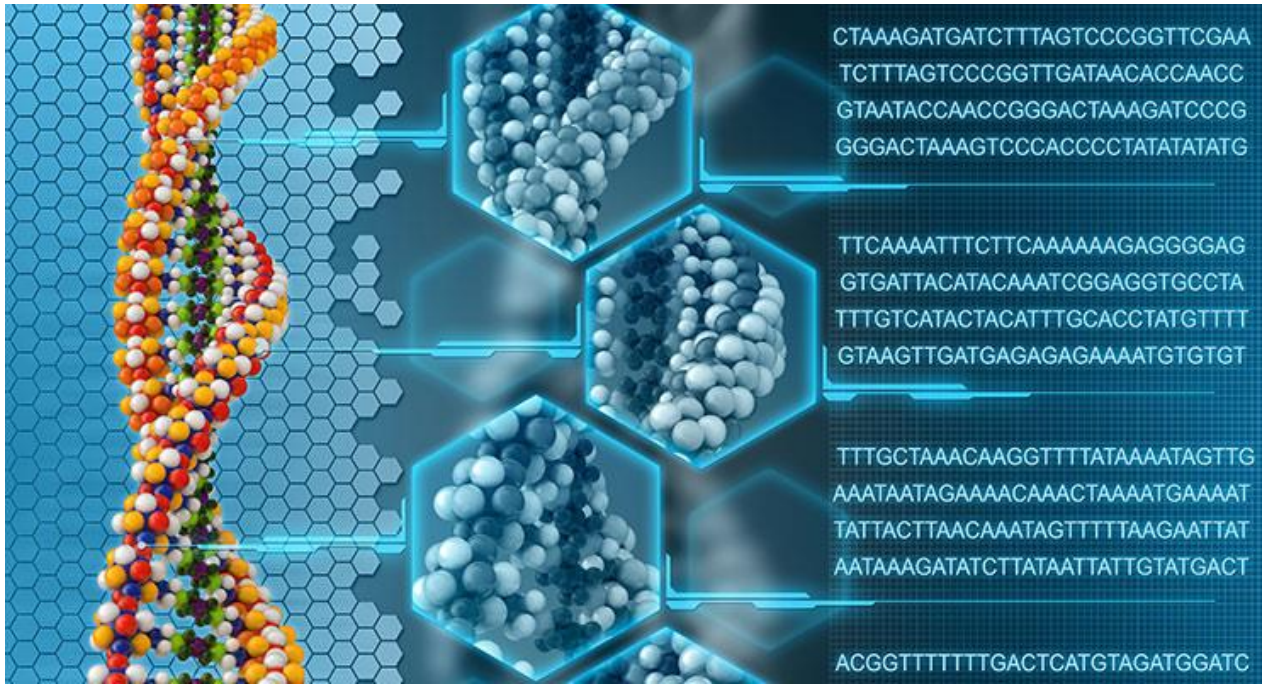
Applications (6): Automatic Handwriting Generation



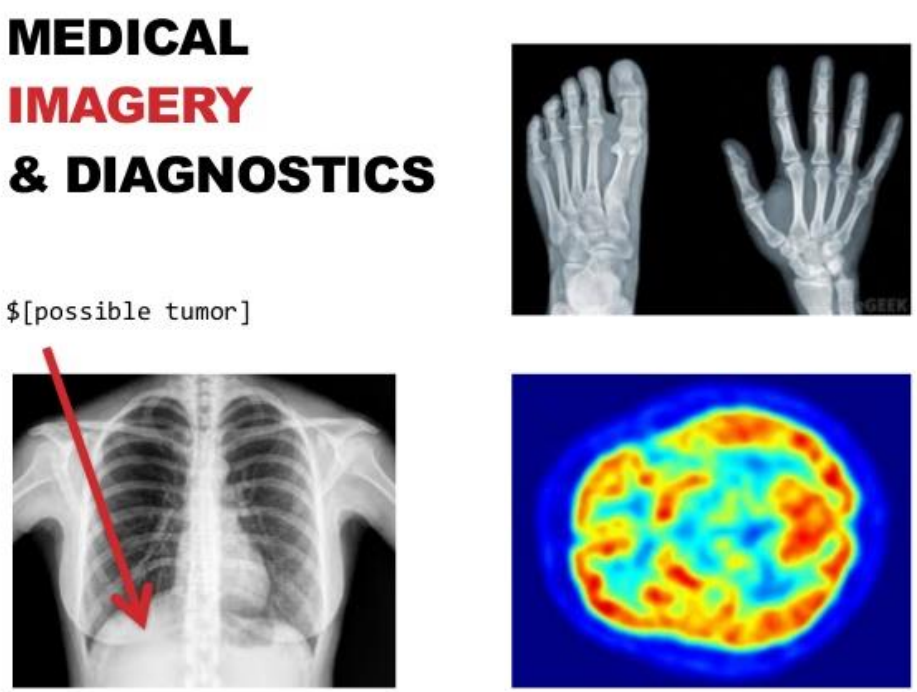
Machine Learning Mastery

Applications (7): Deep Learning in Medicine

Genomics for personalized medicine
Genome Sequencing for aging problem



Better and faster diagnoses –
MRIs, CT scans, X-rays



Applications (8): Recommender Systems

amazon

LinkedIn ebay YAHOO!

YouTube facebook

Applications (9): Self Driving Cars

+ Reduce traffic – communicating with other vehicles

+ Speed – within cities

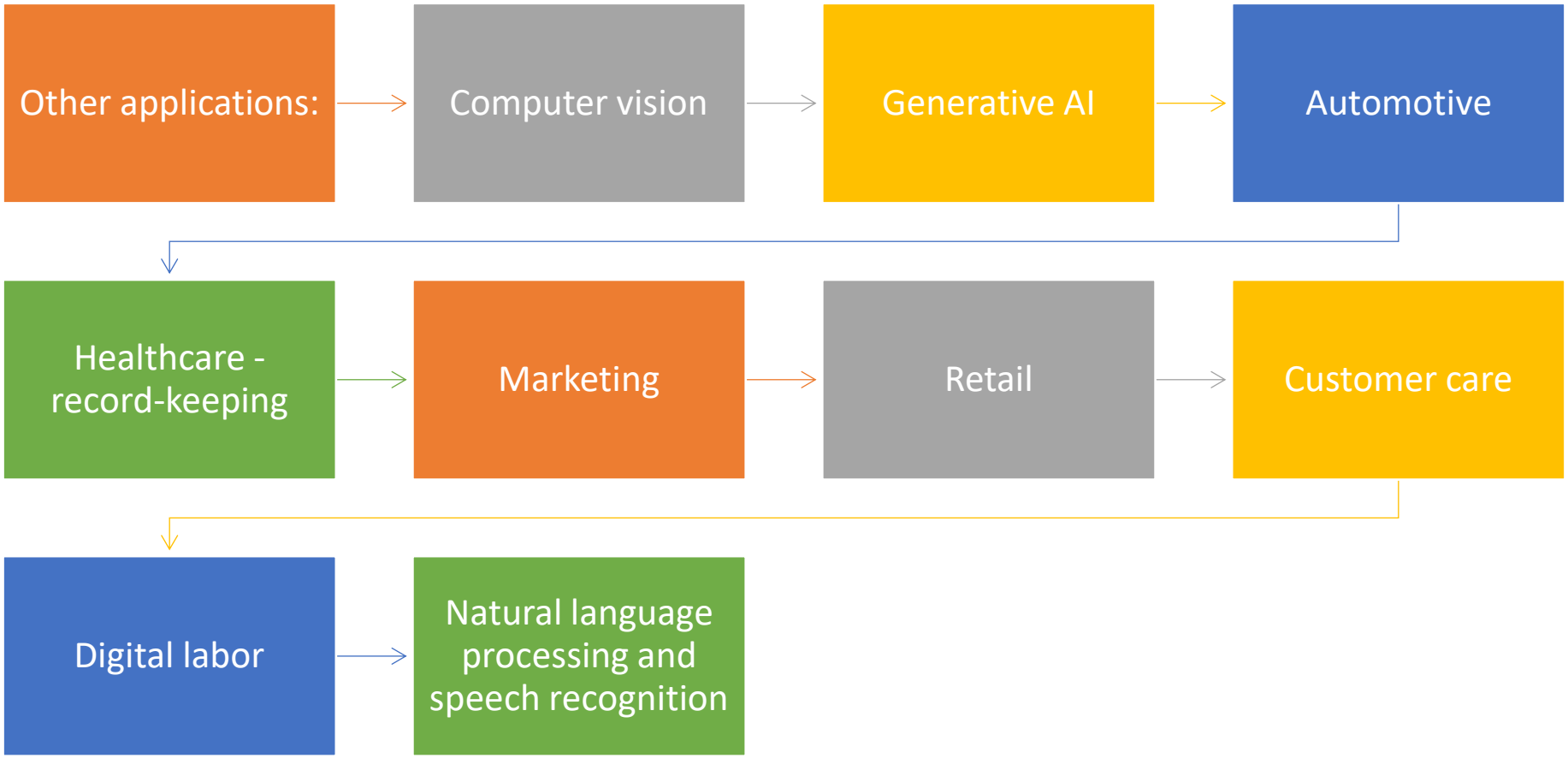
+ Reach destination – with in short time

+ No road accidents

+ No traffic jams

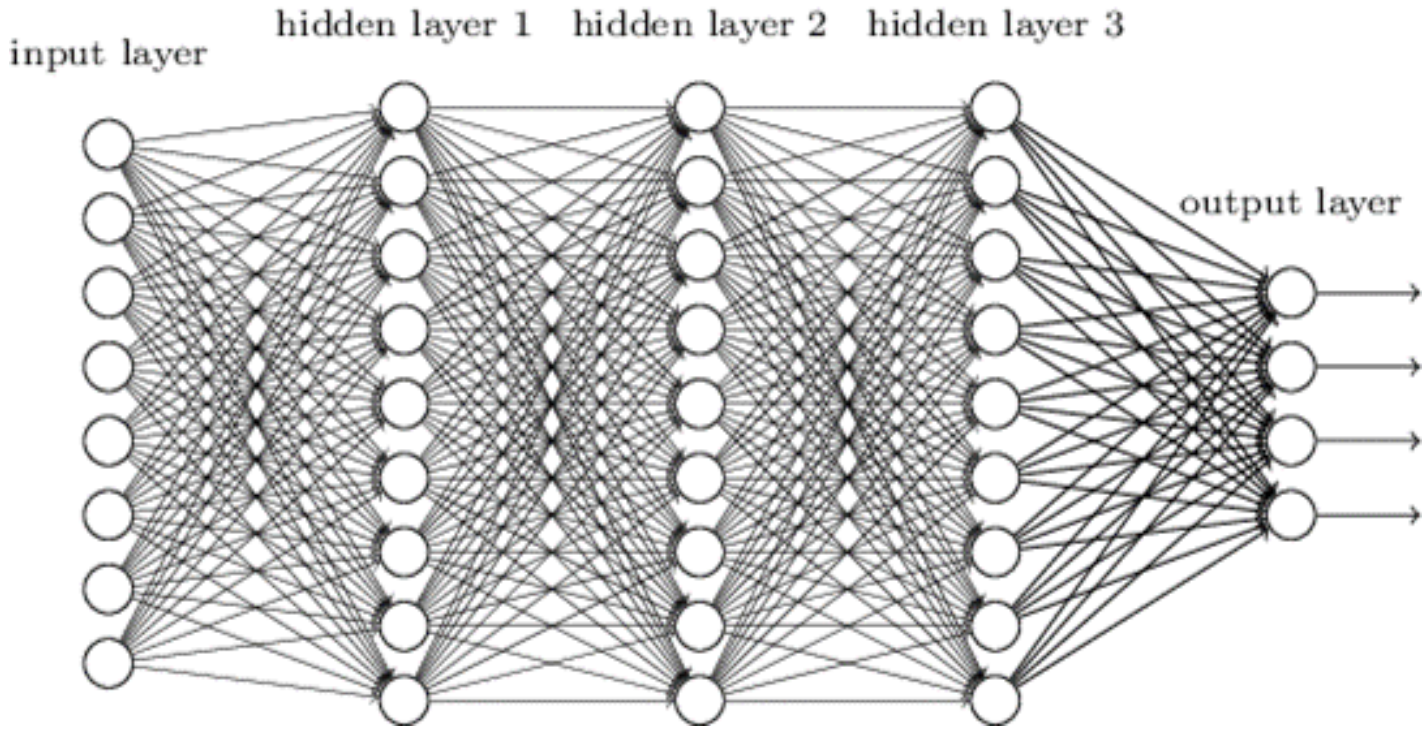
- Drivers jobs ???
- Real estate business ???

Applications



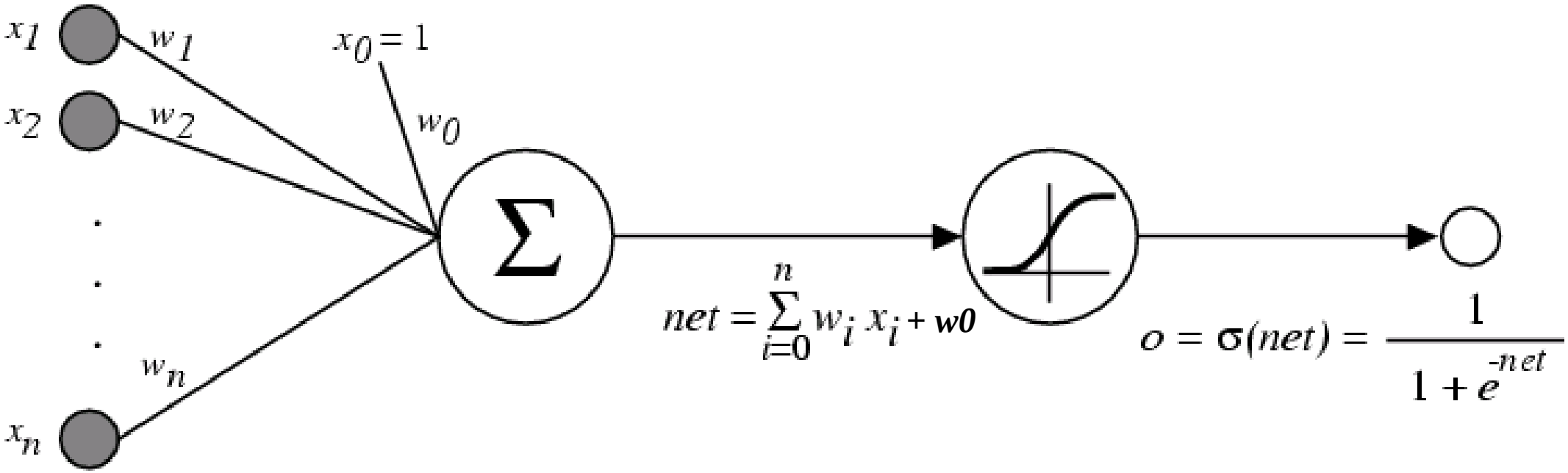
Architecture of Deep Neural Network

- Input layer
- Two or more hidden layers
- Output layer

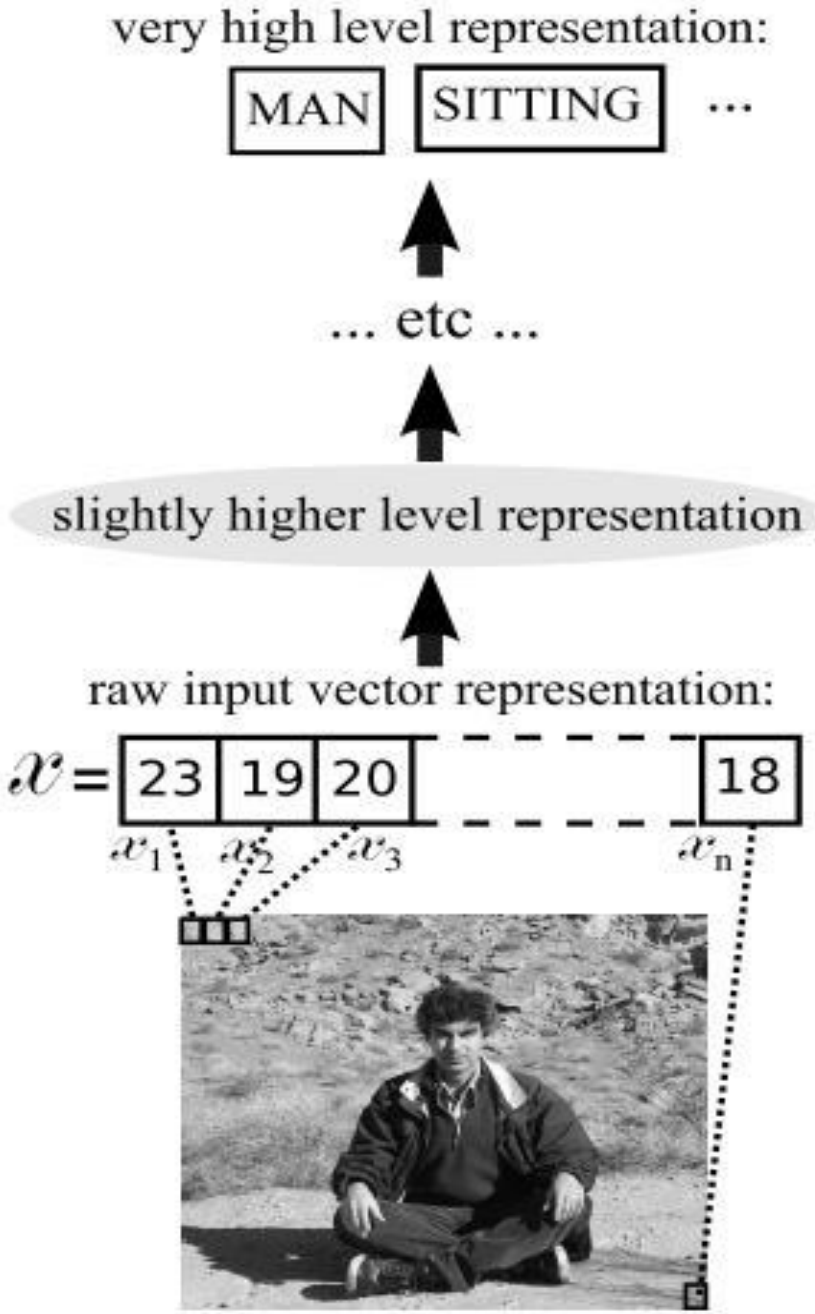


- Deep architectures are composed of multiple levels of non-linear operations - neural nets with many hidden layers.

Architecture of Deep Neural Network



Low level representation



Challenges in Deep Learning

Data availability: It requires **large amounts of data** to learn from. For using deep learning it's a big concern to gather as much data for training.

Computational Resources: For training the deep learning model, it is **computationally expensive** because it requires specialized hardware like **GPUs and TPUs**.

Time-consuming: While working on sequential data depending on the computational resource it can take **very large** even in days or months.

Interpretability: Deep learning models are complex, it works like a **black box,** it is very difficult to interpret the result.

Overfitting: when the model is **trained again and again**, it becomes too specialized for the training data, leading to overfitting and poor performance on new data.

Hyperparameters in Neural Networks

Learning rate: This hyperparameter controls the **step size** taken by the optimizer during each iteration of training. **Too small** a learning rate can result in **slow convergence**, while **too large** a learning rate can lead to **instability and divergence**.

Epochs: This hyperparameter represents the **number of times the entire training dataset is passed through the model** during training. Increasing the number of epochs can **improve the model's performance** but may lead to overfitting if not done carefully.

Number of layers: This hyperparameter determines the depth of the model, which can have a significant impact on its **complexity and learning ability**.

Hyperparameters in Neural Networks

Number of nodes per layer: This hyperparameter determines the **width of the model**, influencing its capacity to represent **complex relationships in the data**.

Architecture: This hyperparameter determines the overall structure of the neural network, including the number of layers, the number of neurons per layer, and the connections between layers. The optimal architecture depends on the complexity of the task and the size of the dataset

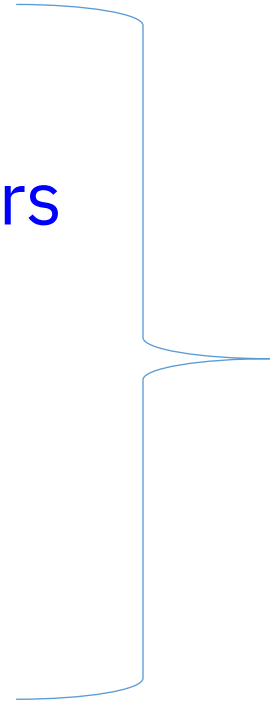
Activation function: This hyperparameter introduces non-linearity into the model, allowing it to learn complex decision boundaries. Common activation functions include sigmoid, tanh, and Rectified Linear Unit (ReLU).

Self study

- **Difference between Machine Learning and Deep Learning**
- **Advantages of Deep Learning**
- **Disadvantages of Deep Learning**
- **Hyperparameters in Support Vector Machine**
- **Hyperparameters in Bagging and Boosting models**

Types of deep learning models

- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Autoencoders and variational autoencoders
- Generative adversarial networks (GANs)
- Diffusion models
- Transformer models



Types of deep learning models

- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Autoencoders and variational autoencoders
- Generative adversarial networks (GANs)
- Diffusion models
- Transformer models

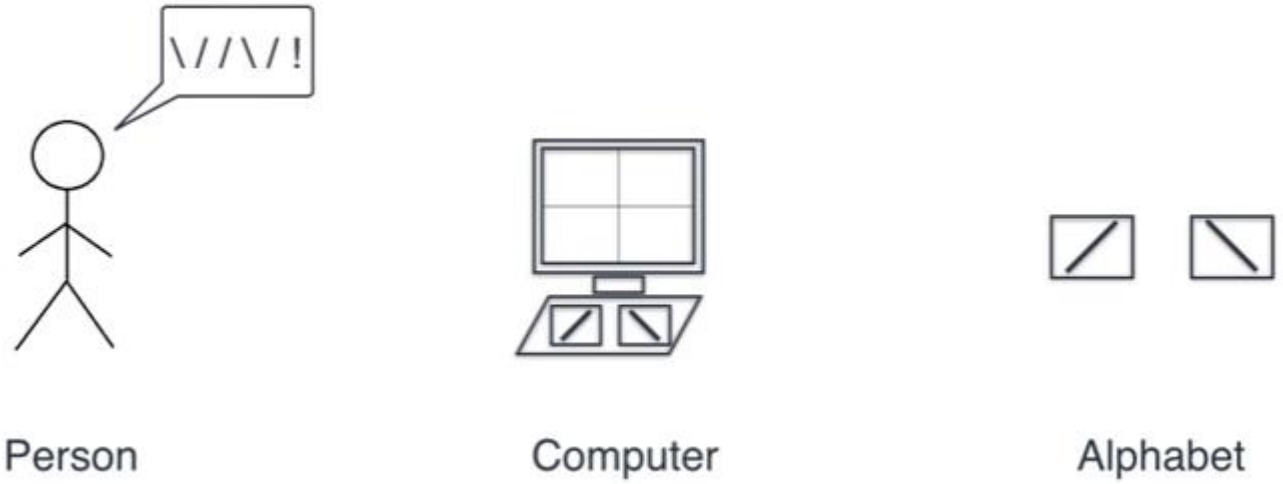
Convolutional Neural Networks

Convolutional Neural Networks

- Convolutional neural networks (CNNs) are widely used tools for deep learning.
- Suitable for applications such as
  - Images
  - Text
  - Signals, and
  - other continuous responses - as inputs

Convolutional Neural Networks

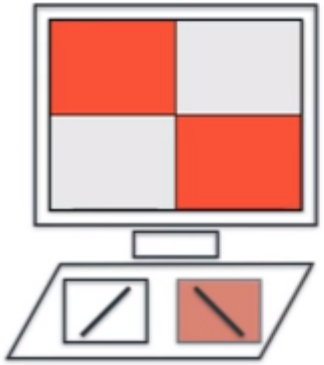
Consider a simple world scenario

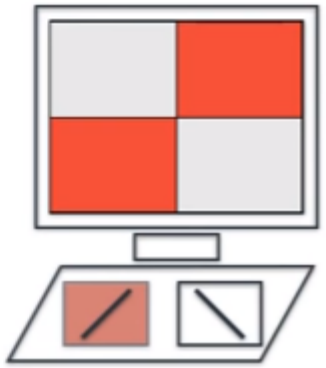


- Computer – 2 X 2 pixel resolution
- Keyboard with only two alphabets / and \

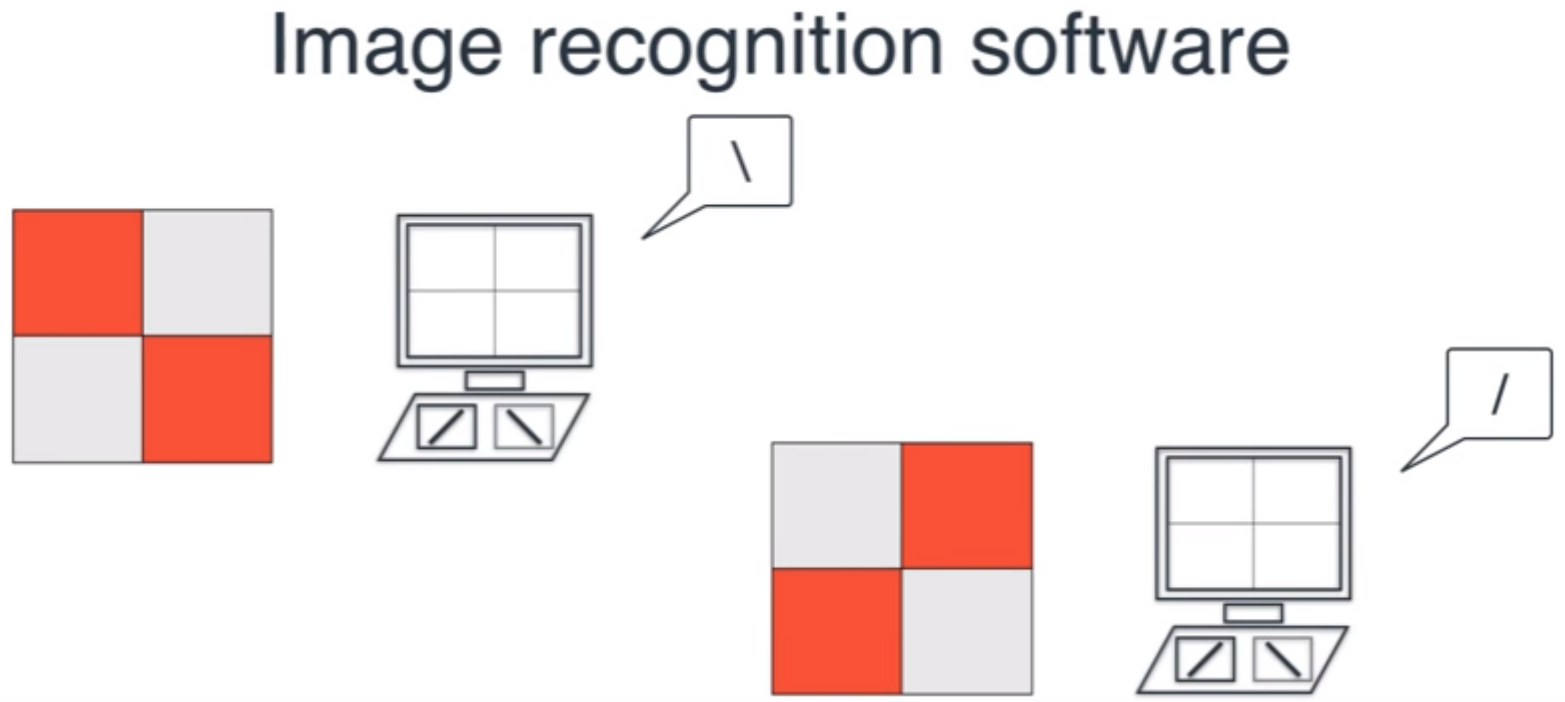
A simple image recognition system

- If you press the key in the right, you can see the image in the screen.



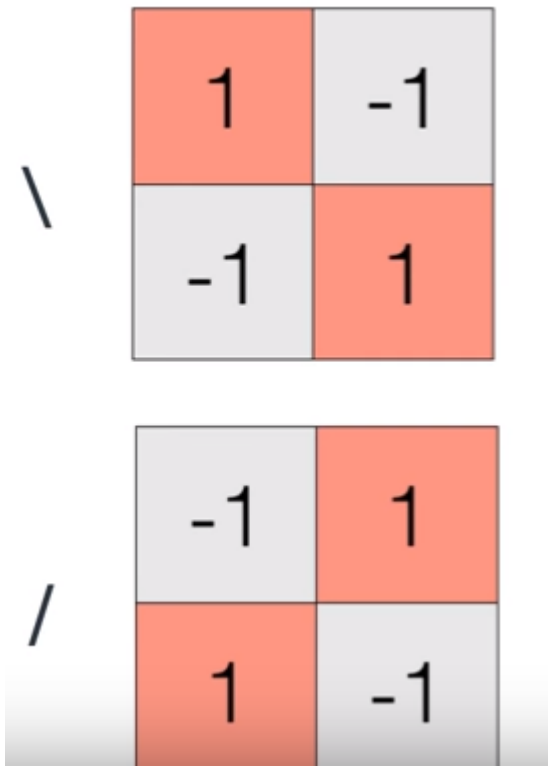


- If you press the key in the left, you can see the image in the screen.



- Computers see the pixels as numbers, say, red pixel as 1 and gray pixel as -1

Image

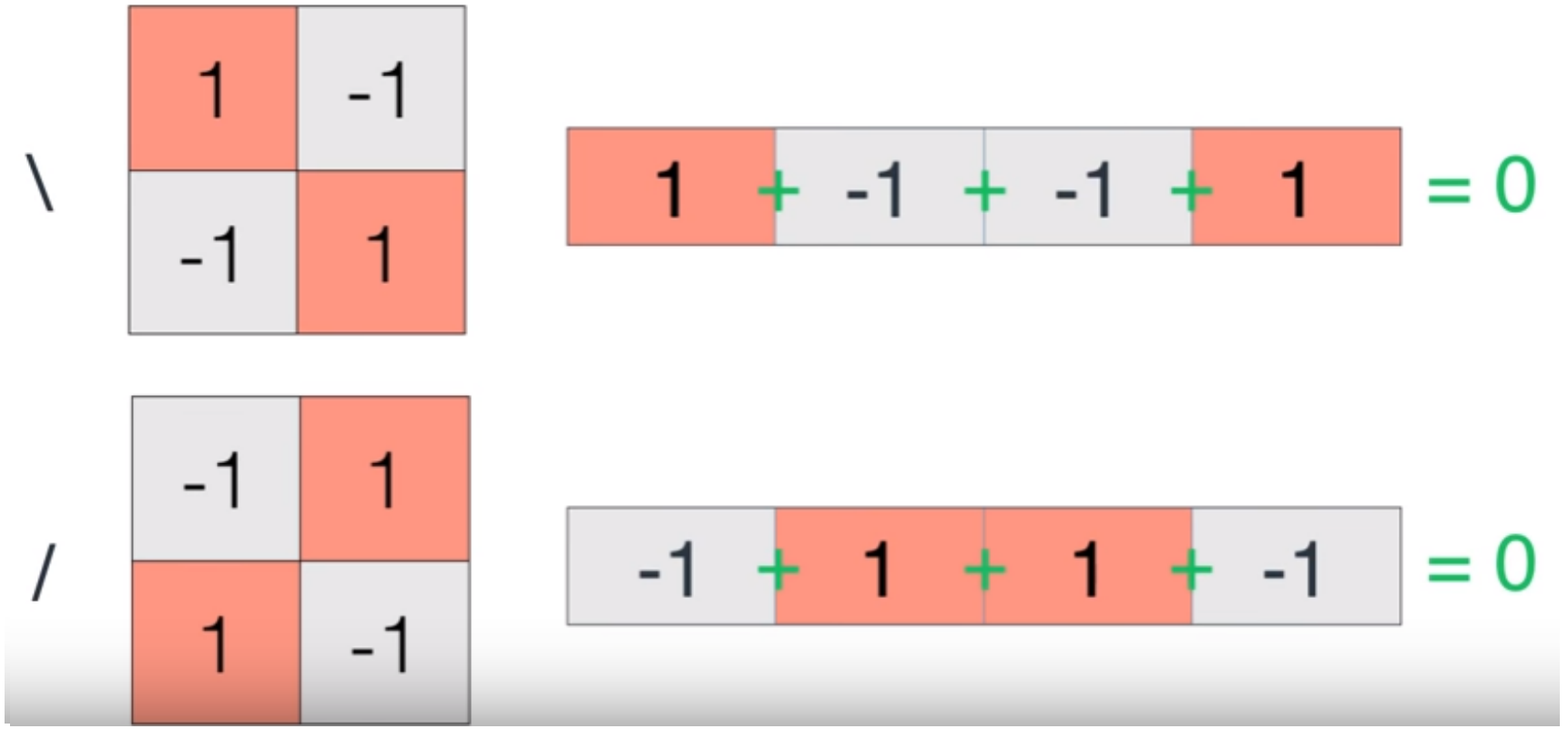


1	**-1**	**-1**	**1**

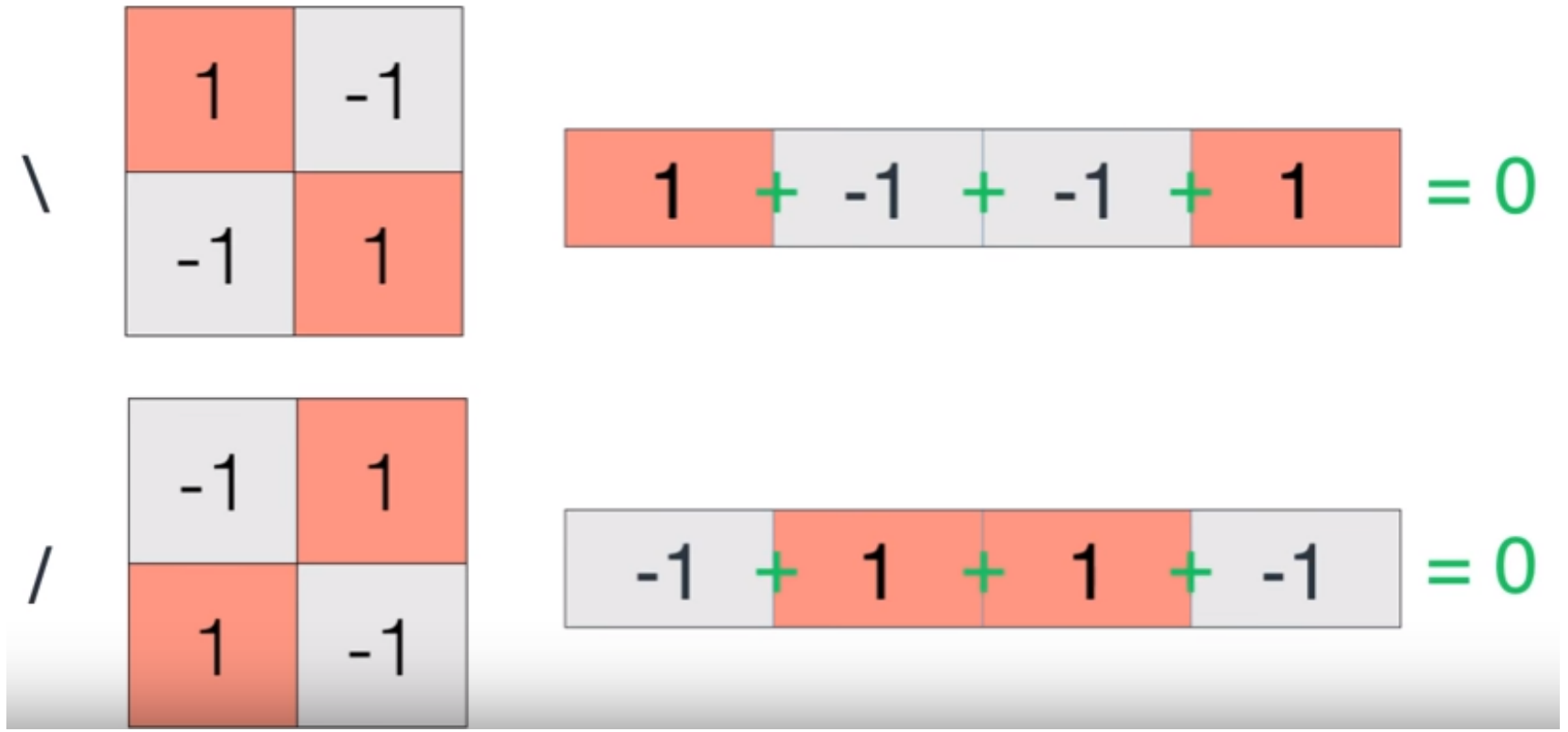
-1	**1**	**1**	**-1**

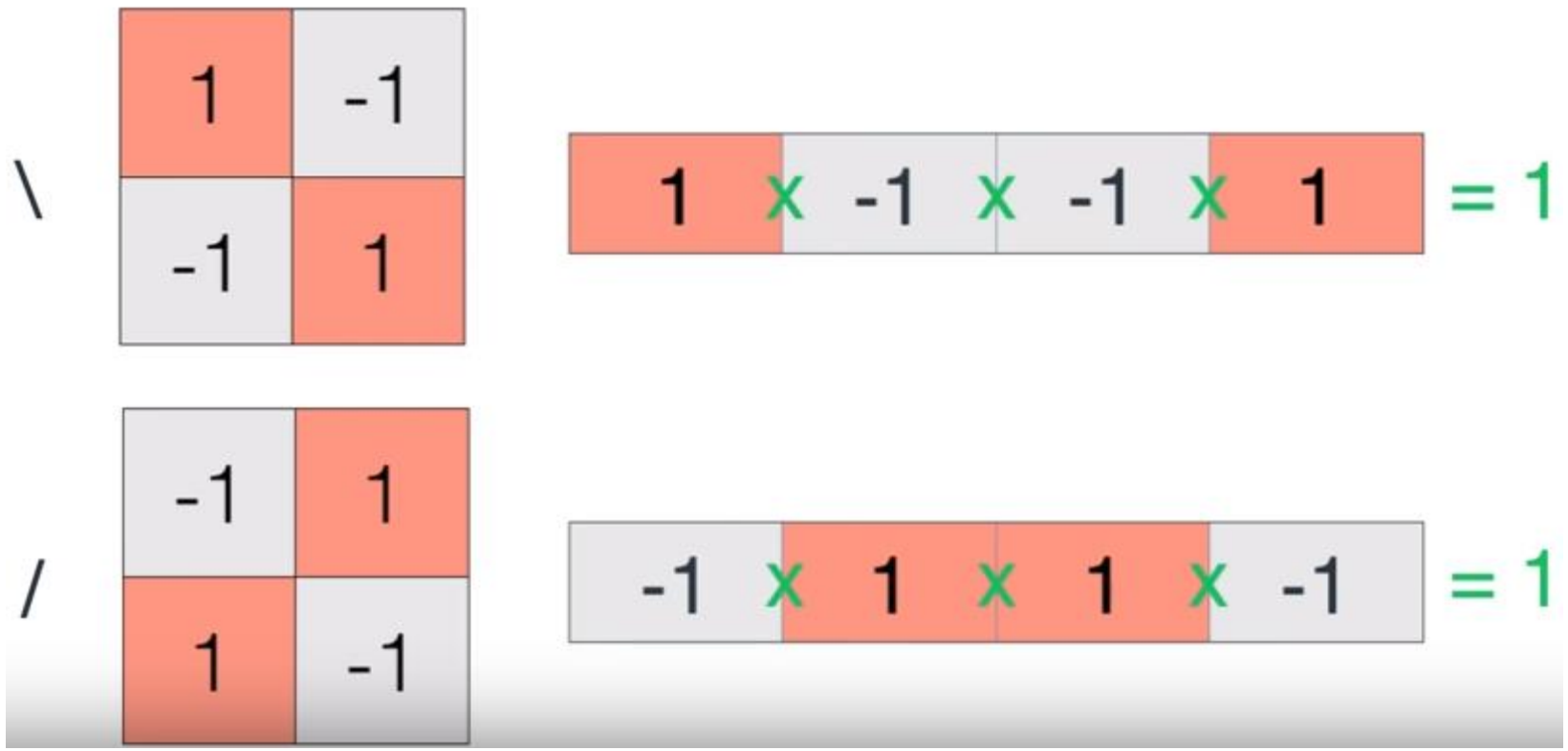
Use some mathematical operation to distinguish these two operations

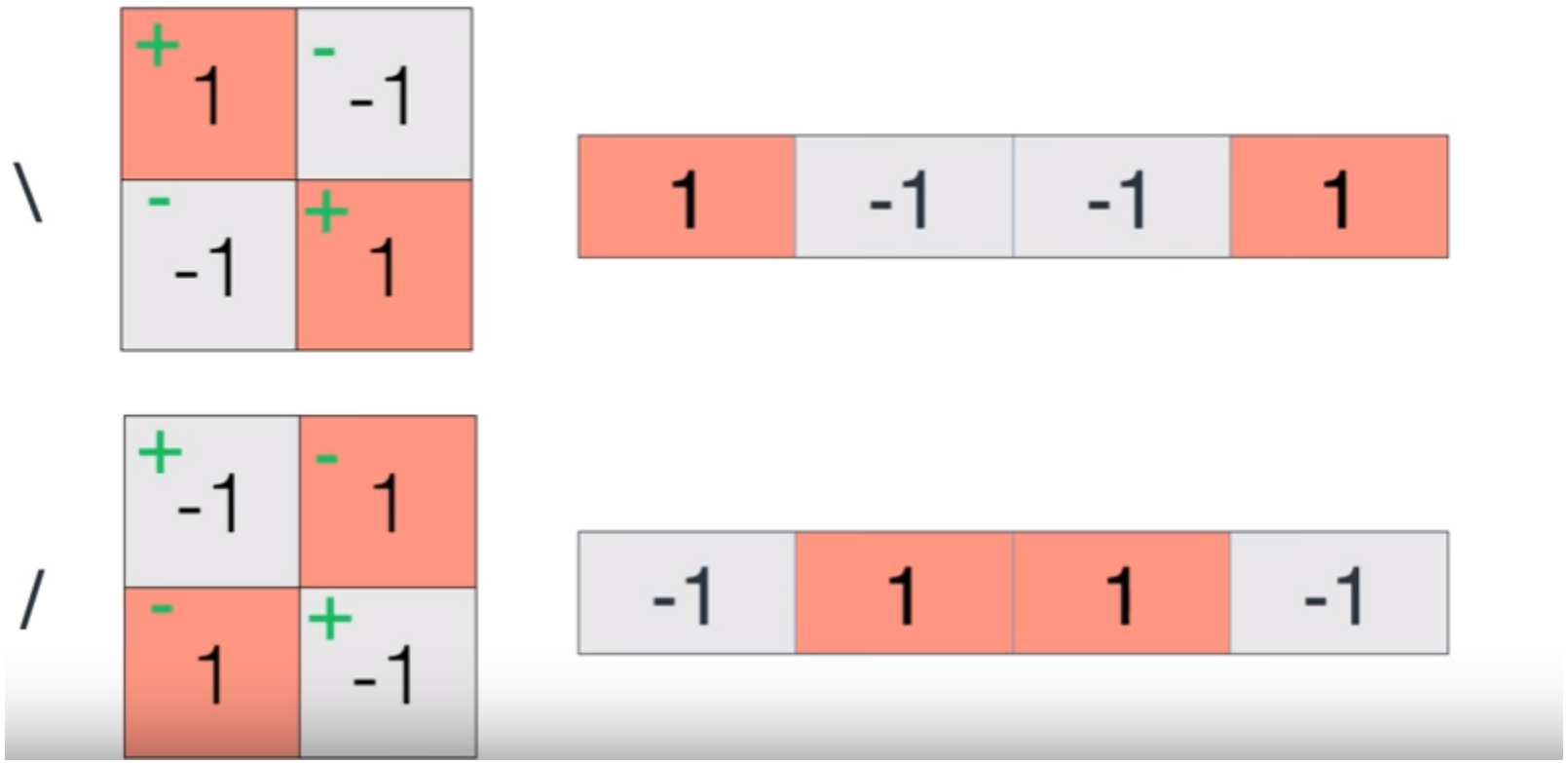
Computers see this pixels as a sequence of 1's and -1's.

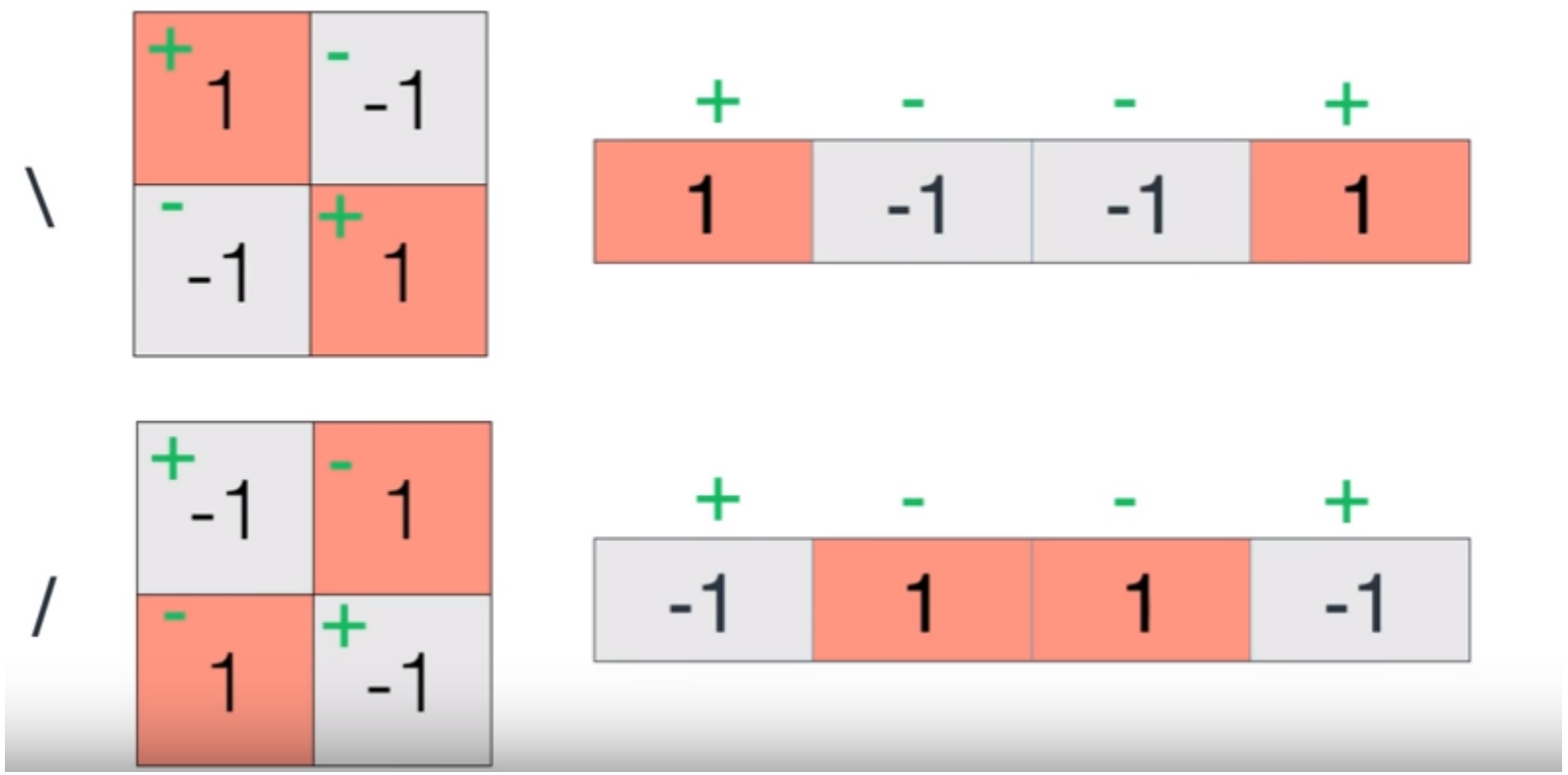


- Use some mathematical operation to distinguish these two operations


\
1
-1
-1
1
1 + -1 + -1 + 1 = 0
/
-1
1
1
-1
-1 + 1 + 1 + -1 = 0


\
1
-1
-1
1
1 x -1 x -1 x 1
= 1
/
-1
1
1
-1
-1 x 1 x 1 x -1
= 1


\
+ 1
- -1
- -1
+ 1
1 -1 -1 1
/
+ -1
- 1
- 1
+ -1
-1 1 1 -1


\
+ 1
- -1
- -1
+ 1
+ - - +
1 -1 -1 1
/
+ -1
- 1
- 1
+ -1
+ - - +
-1 1 1 -1

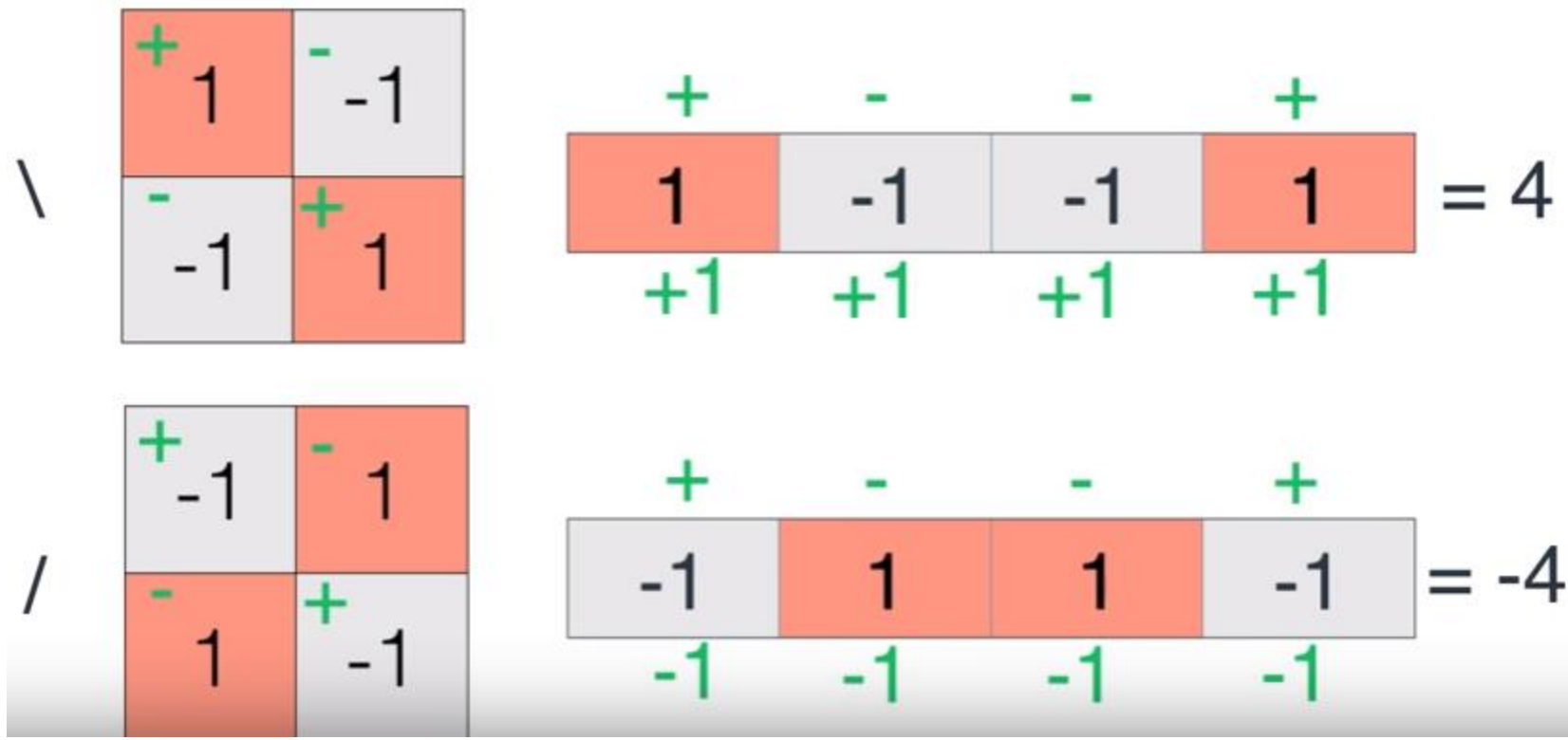
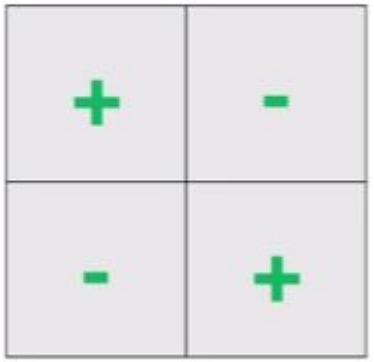

\
+ 1
- -1
- -1
+ 1
+ - - +
1 -1 -1 1
= 4
+1 +1 +1 +1
/
+ -1
- 1
- 1
+ -1
+ - - +
-1 1 1 -1
= -4
-1 -1 -1 -1

Image Recognition Classifier

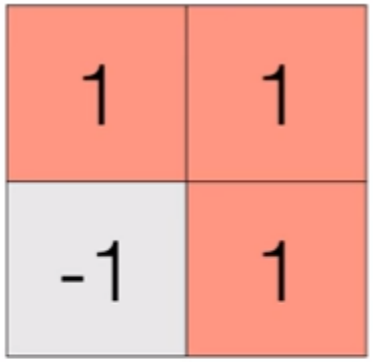


If positive, “\”

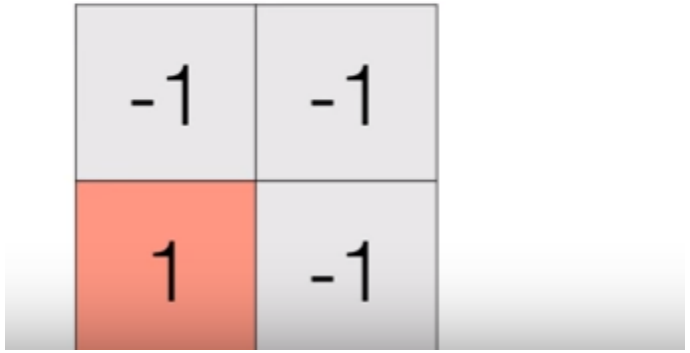
If negative, “/”

- This image recognition classifier tells
  - when you add these numbers, if you get positive - it is a backward slash
  - when you add these numbers, if you get negative - it is a forward slash

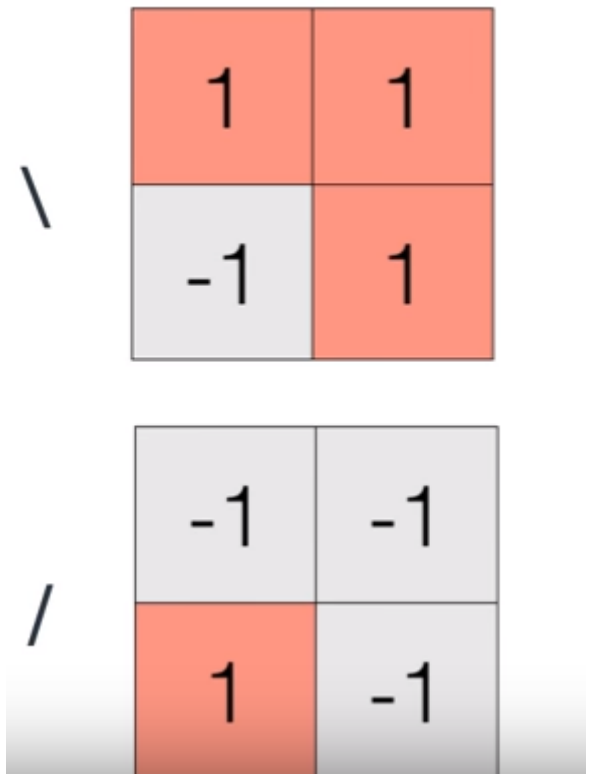
Poorly drawn backward slash

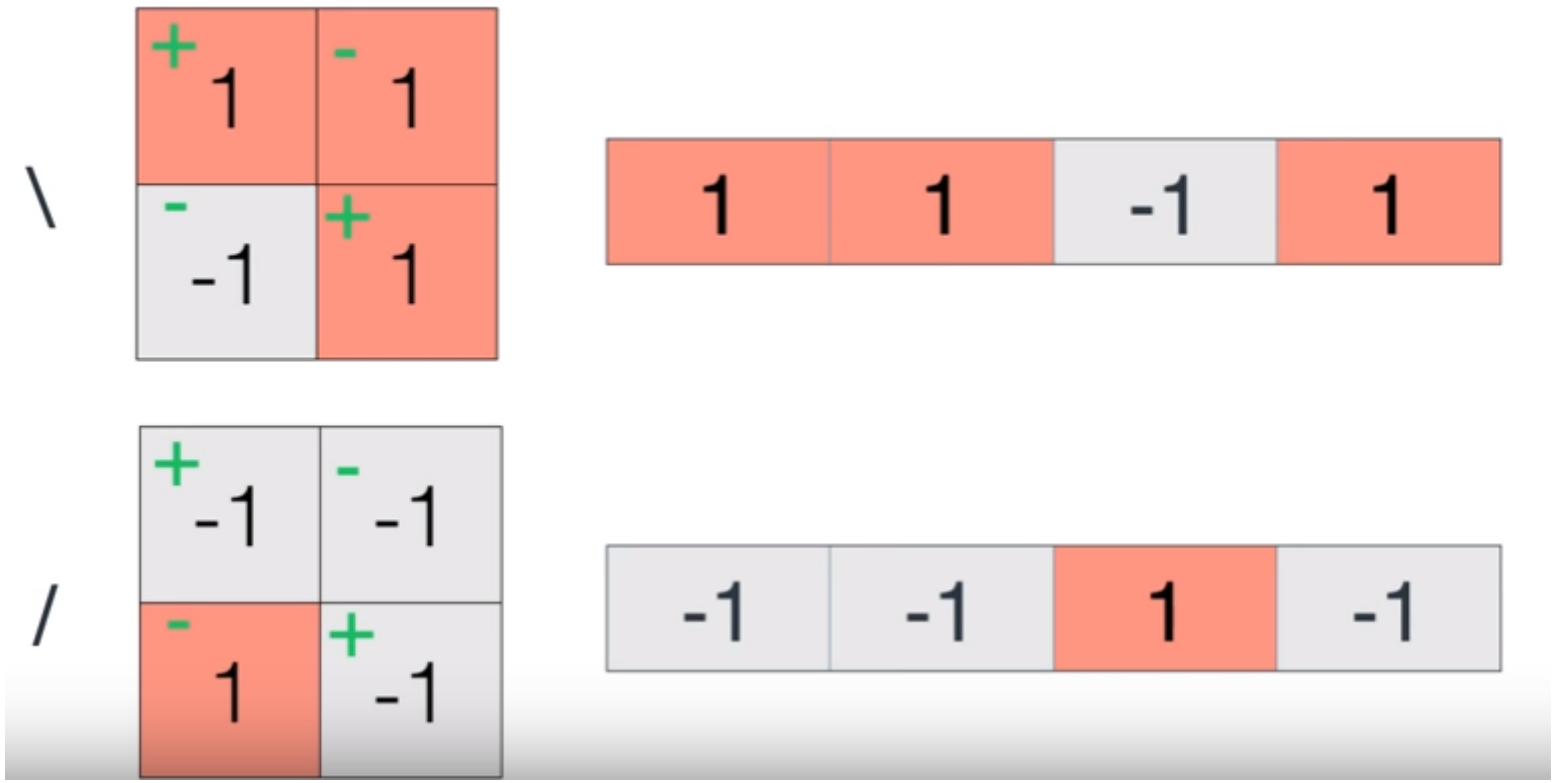


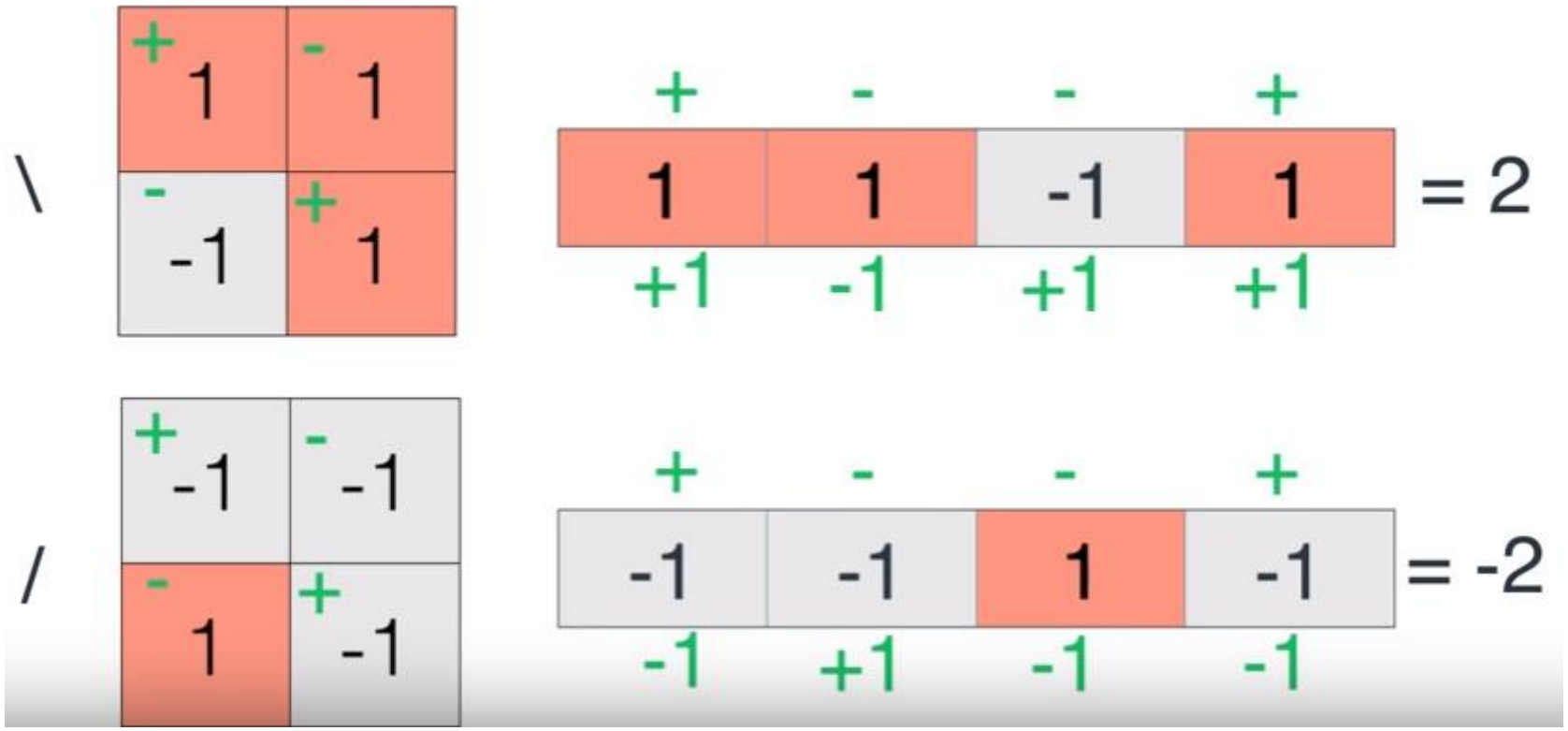
Incomplete forward slash (ran out of ink??)



if we use our existing classifier, what does the computer see?

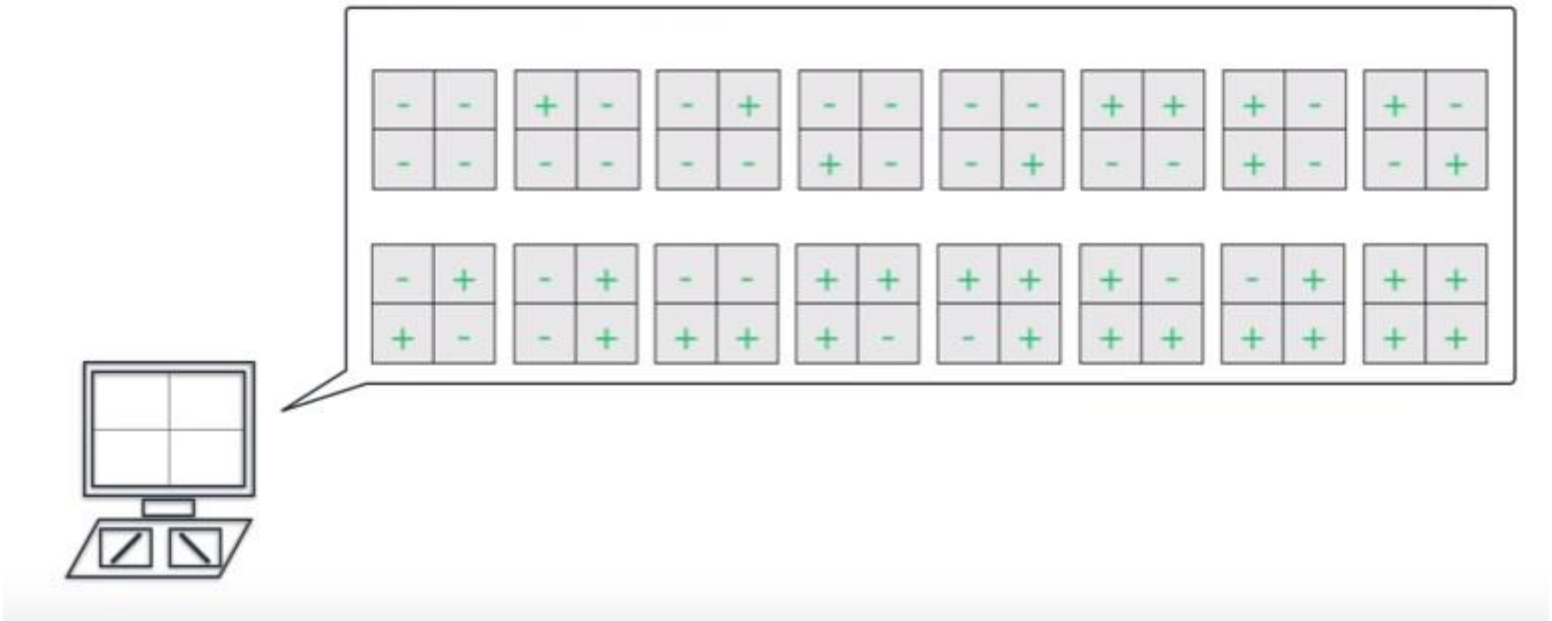

\
1
1
-1
1
/
-1
-1
1
-1


\
+ 1
- 1
- -1
+ 1
1
1
-1
1
/
+ -1
- -1
- 1
+ -1
-1
-1
1
-1



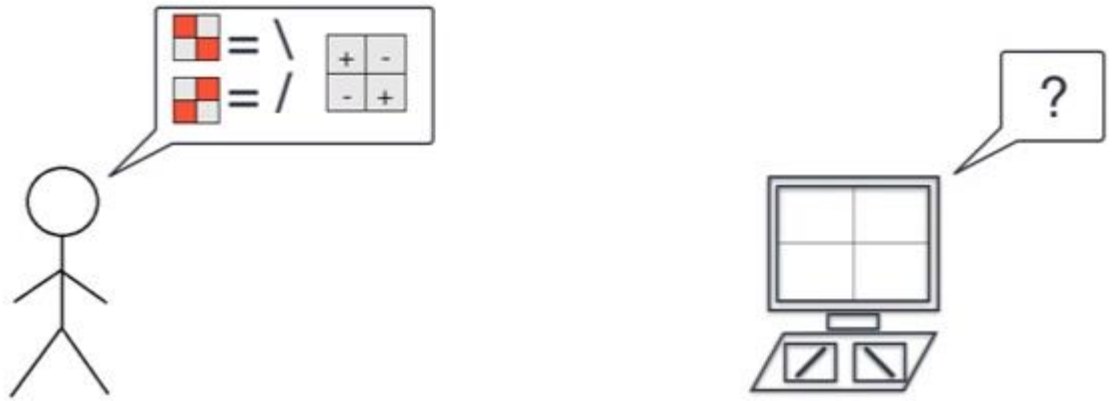
- It classifies as backward and forward slashes with less values.

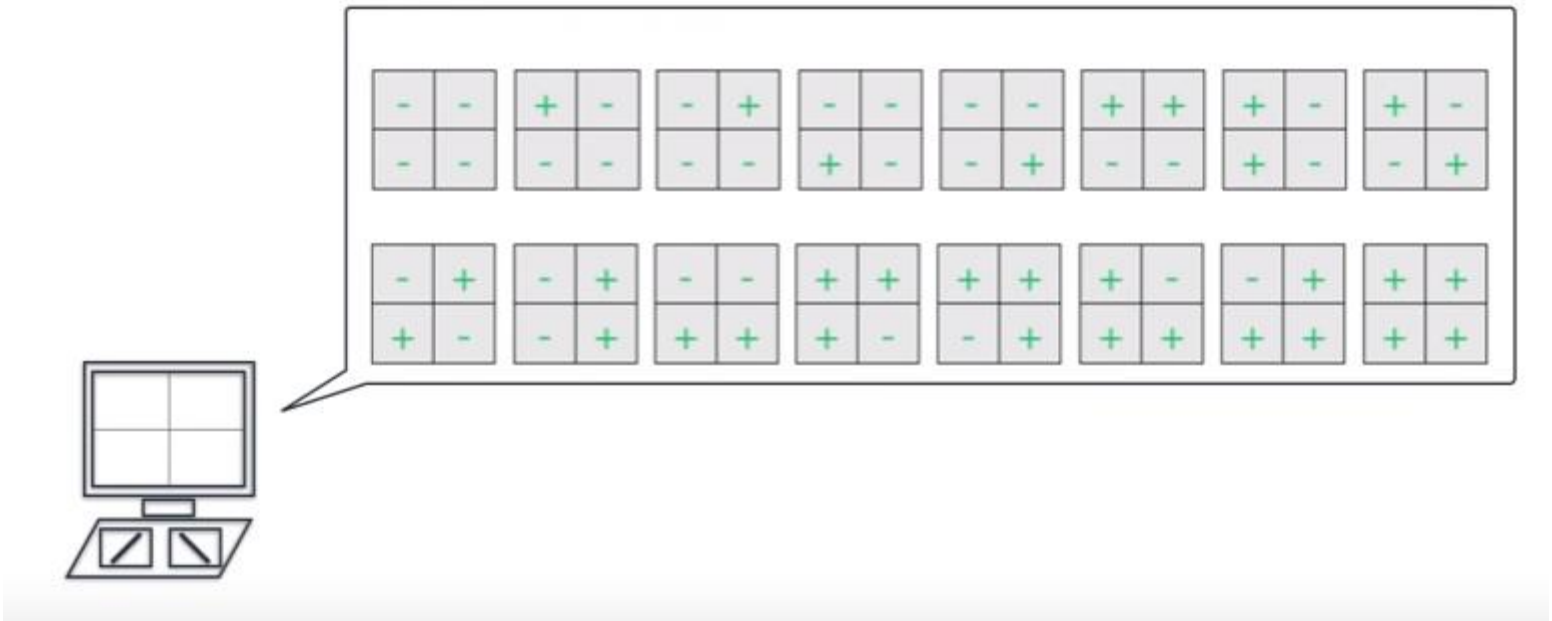
How will you find the classifier



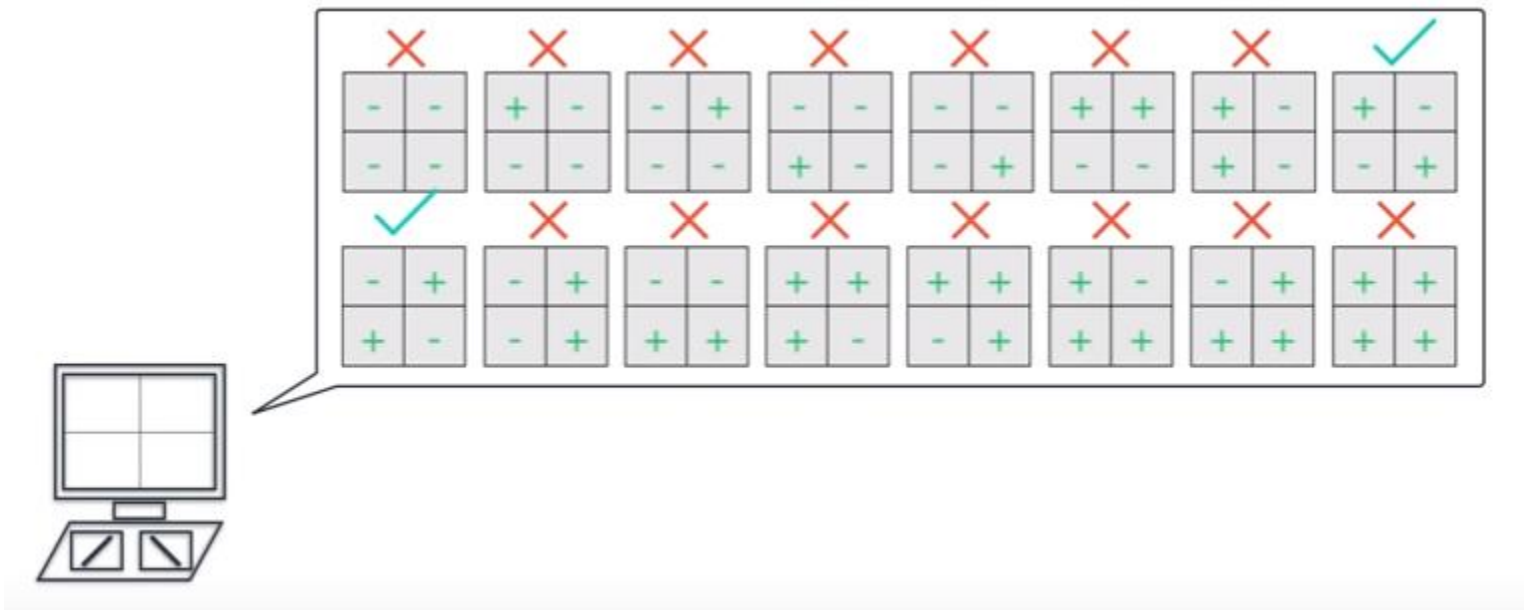
- It go through all the possibilities
- Checks all of them

We want a robust classifier. How can the computer do?





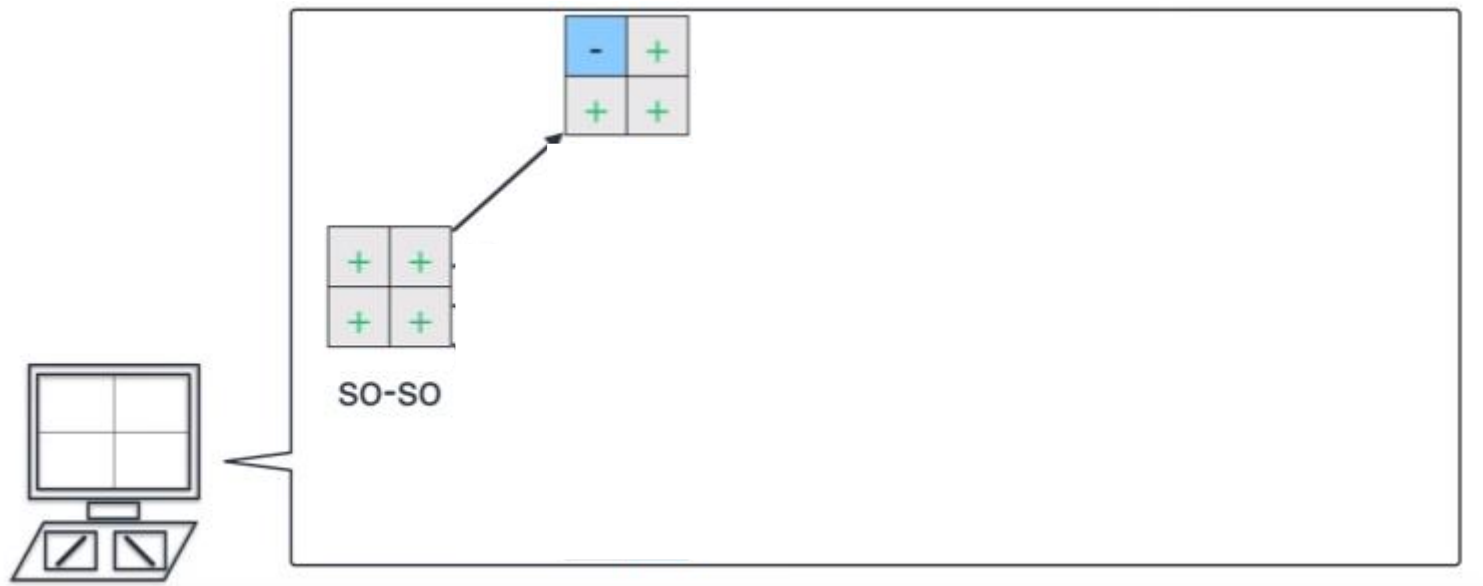
- It go through all the possibilities
- Checks all of them

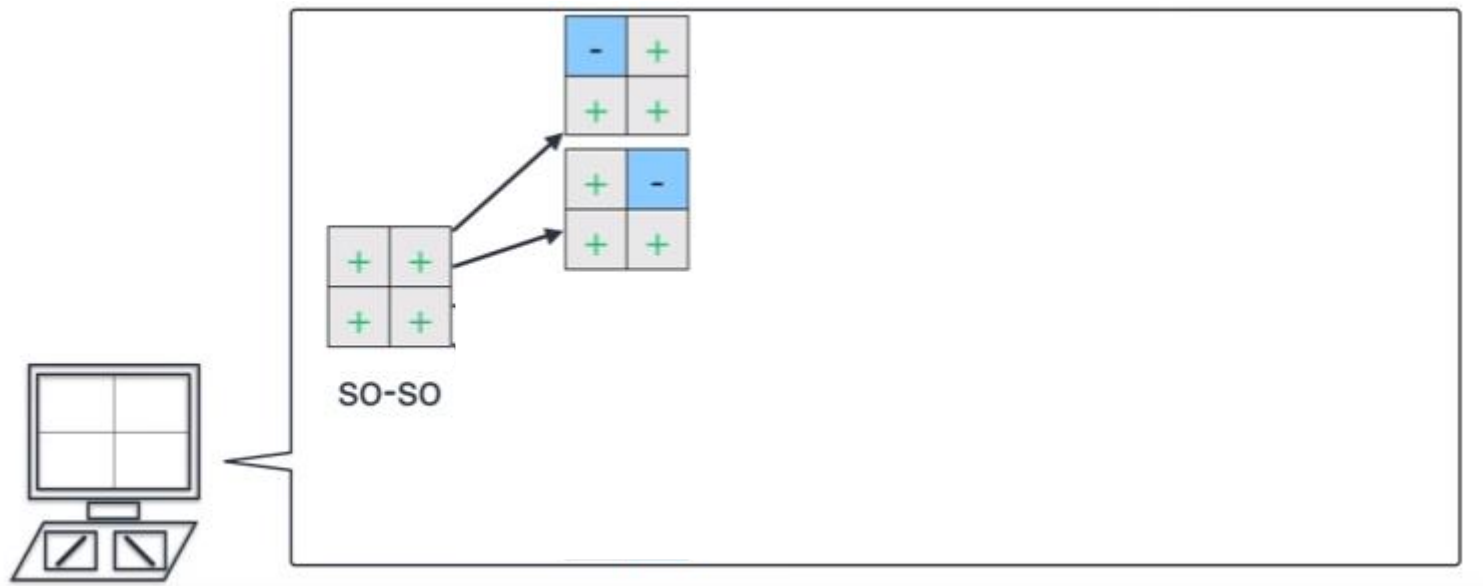


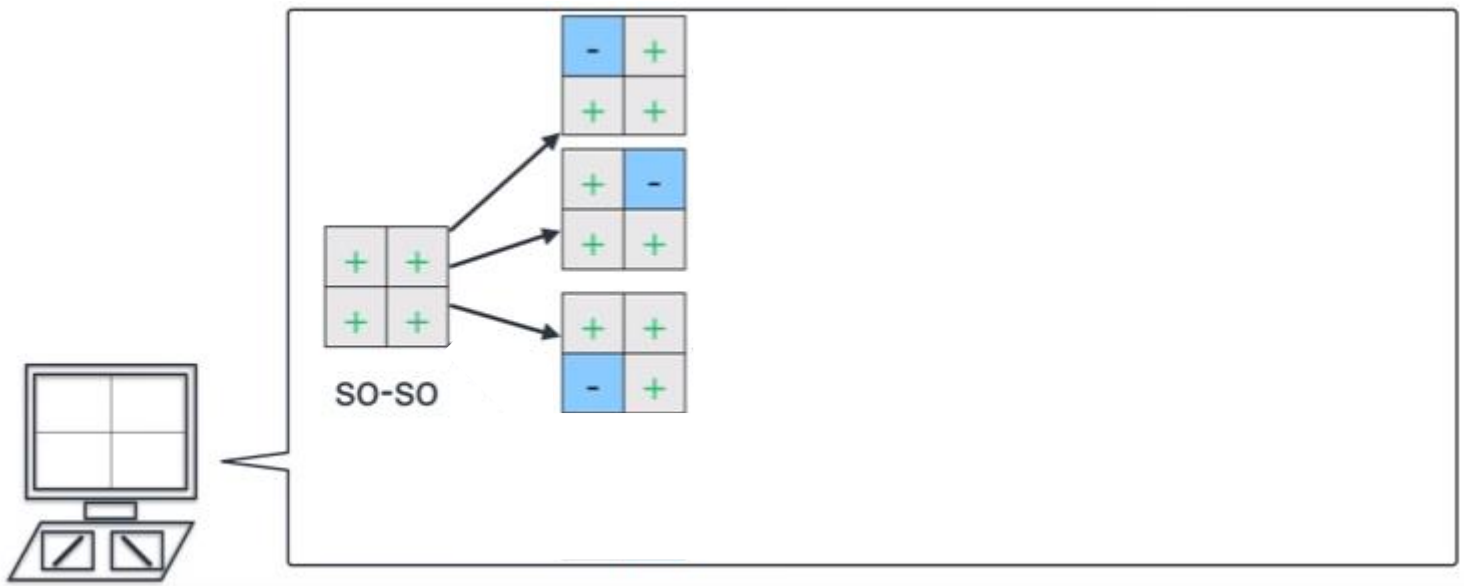
- Two classify very well
- Others classify poorly

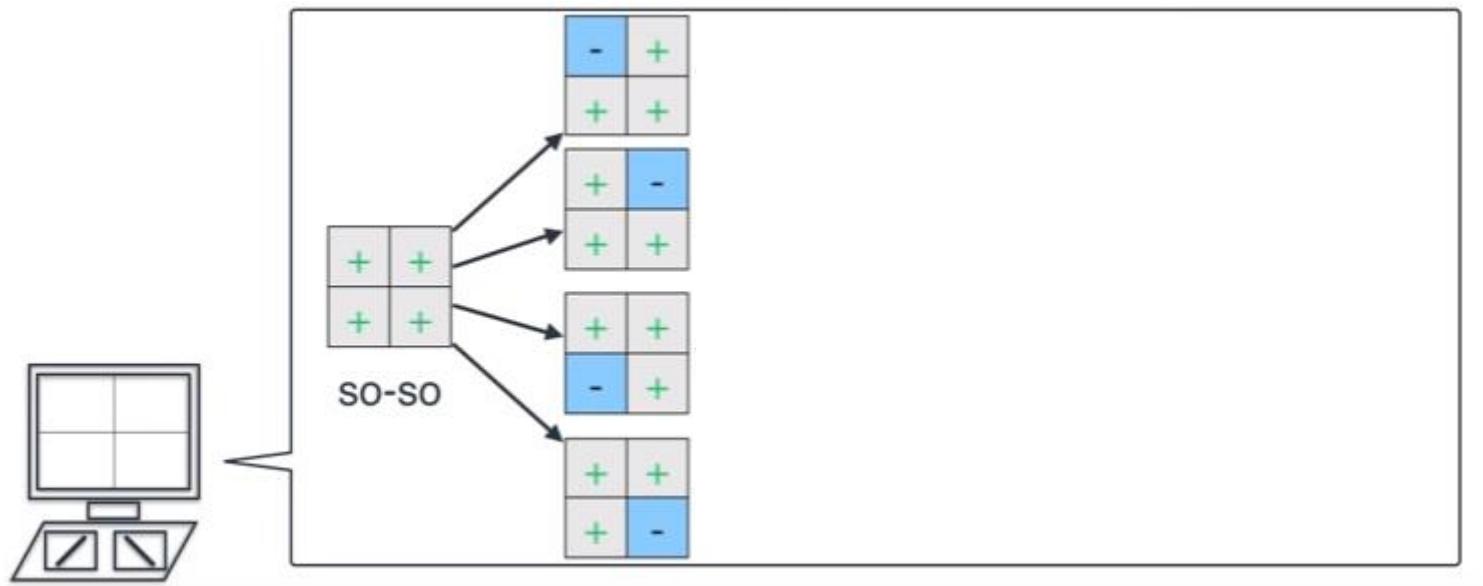
- We have screen of **2 x 2** pixel, keyboard with 2 letters - two images per classifier – 16 choices
- Imagine how many choices in real time – with so many pixels, colors and images – we have to find a smart way.

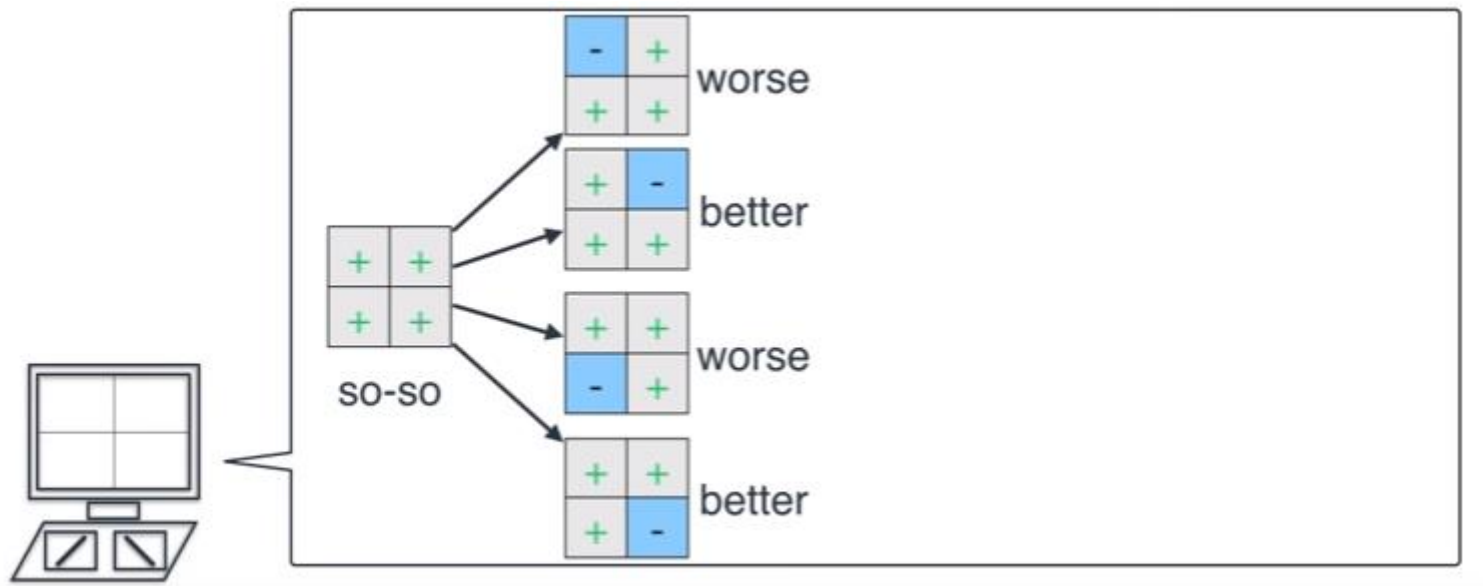
- Idea:
  - Start with so-so combination
  - Let us take - change separately each + into -

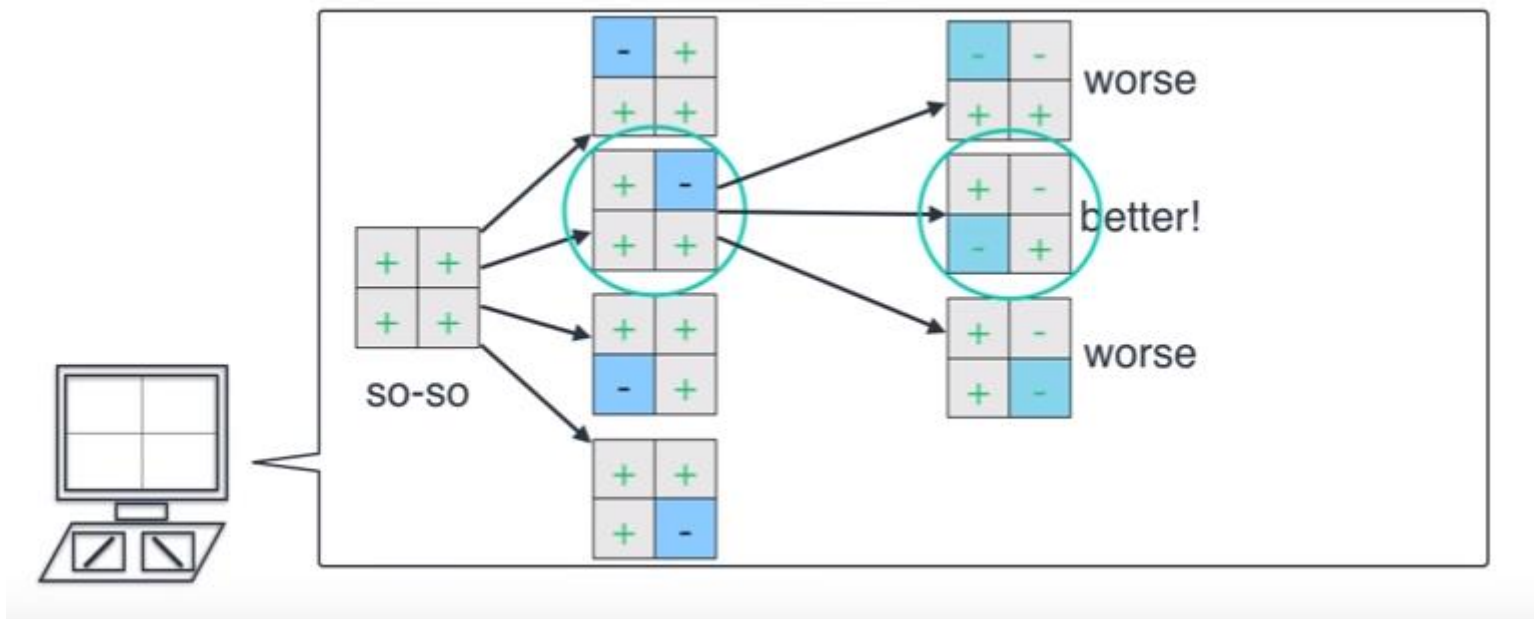

-
+
+
+
+
+
+
SO-SO

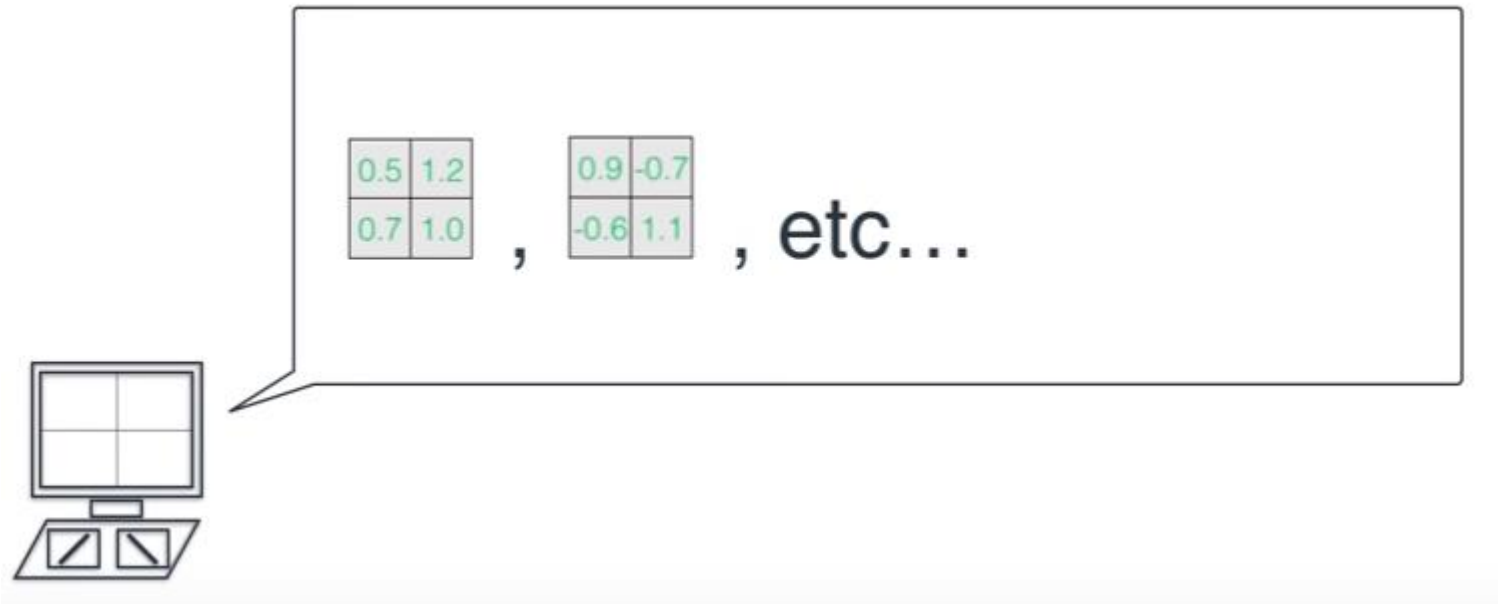

SO-SO


SO-SO


SO-SO

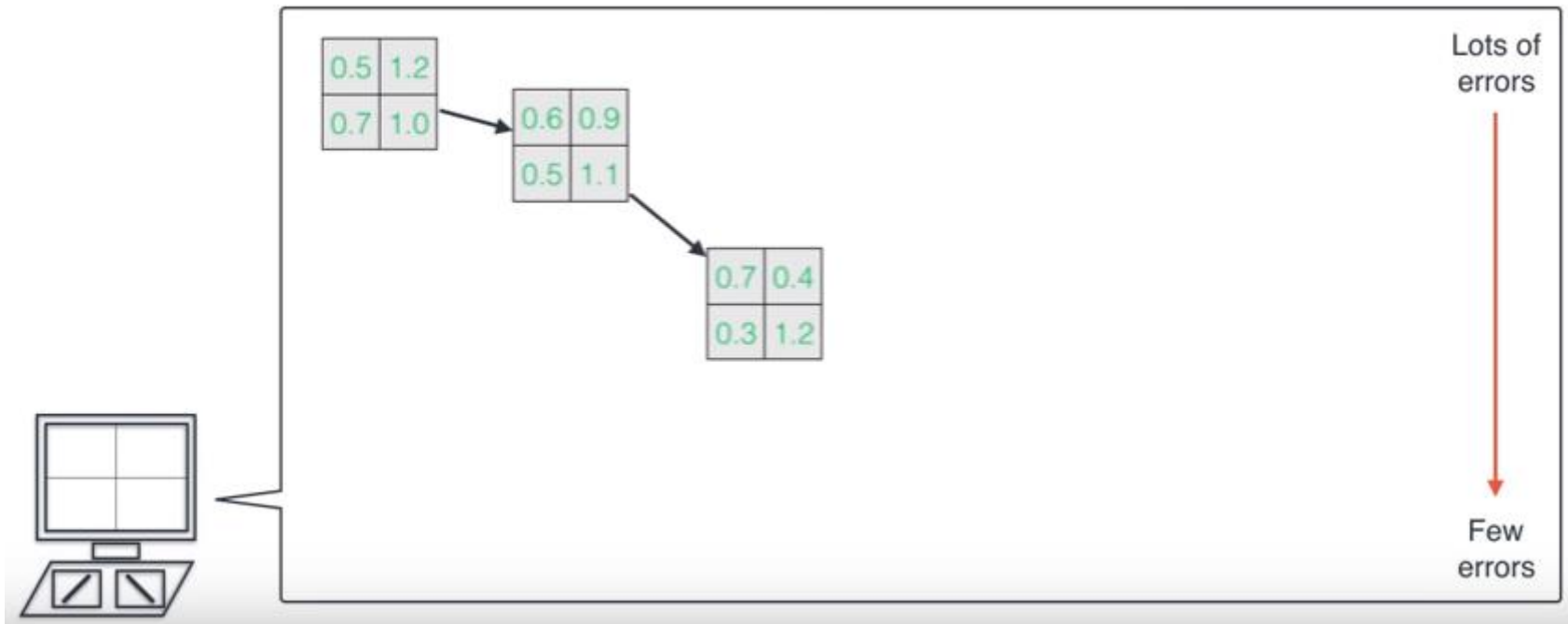

worse
better
worse
better
so-so

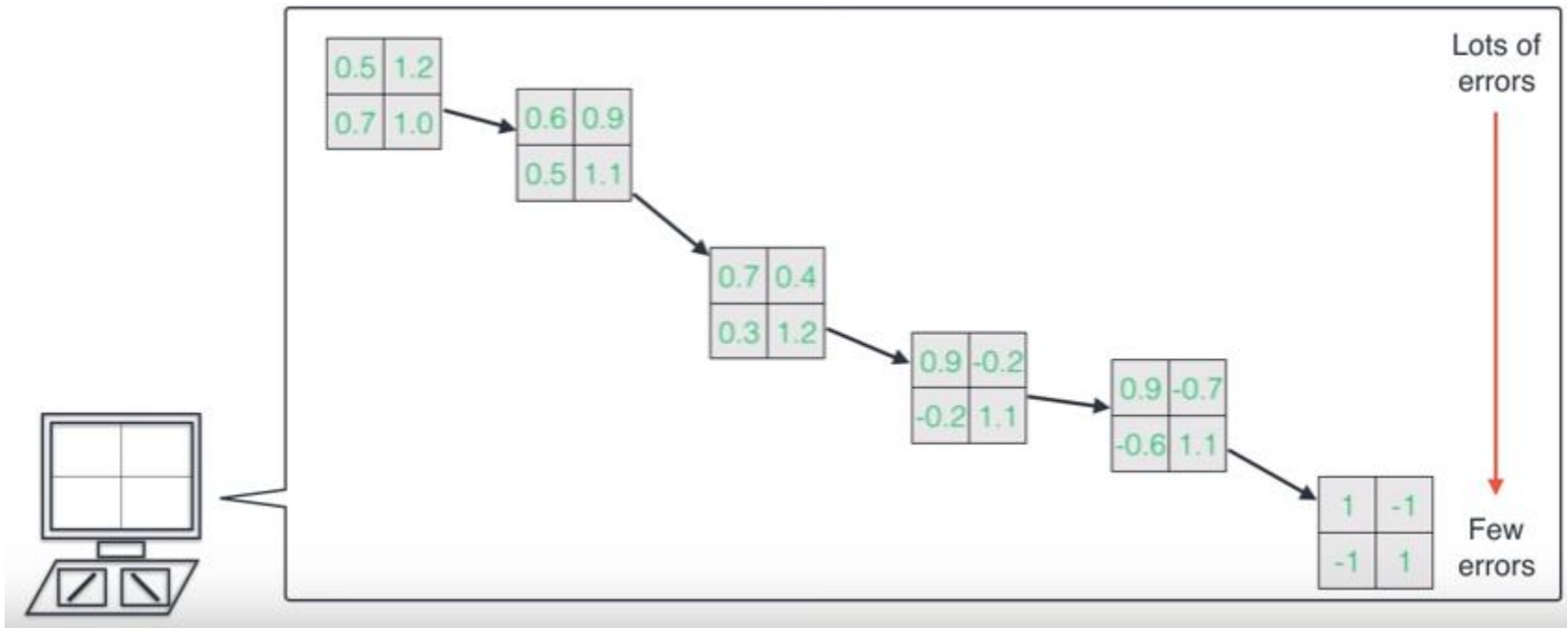

so-so
worse
better!
worse



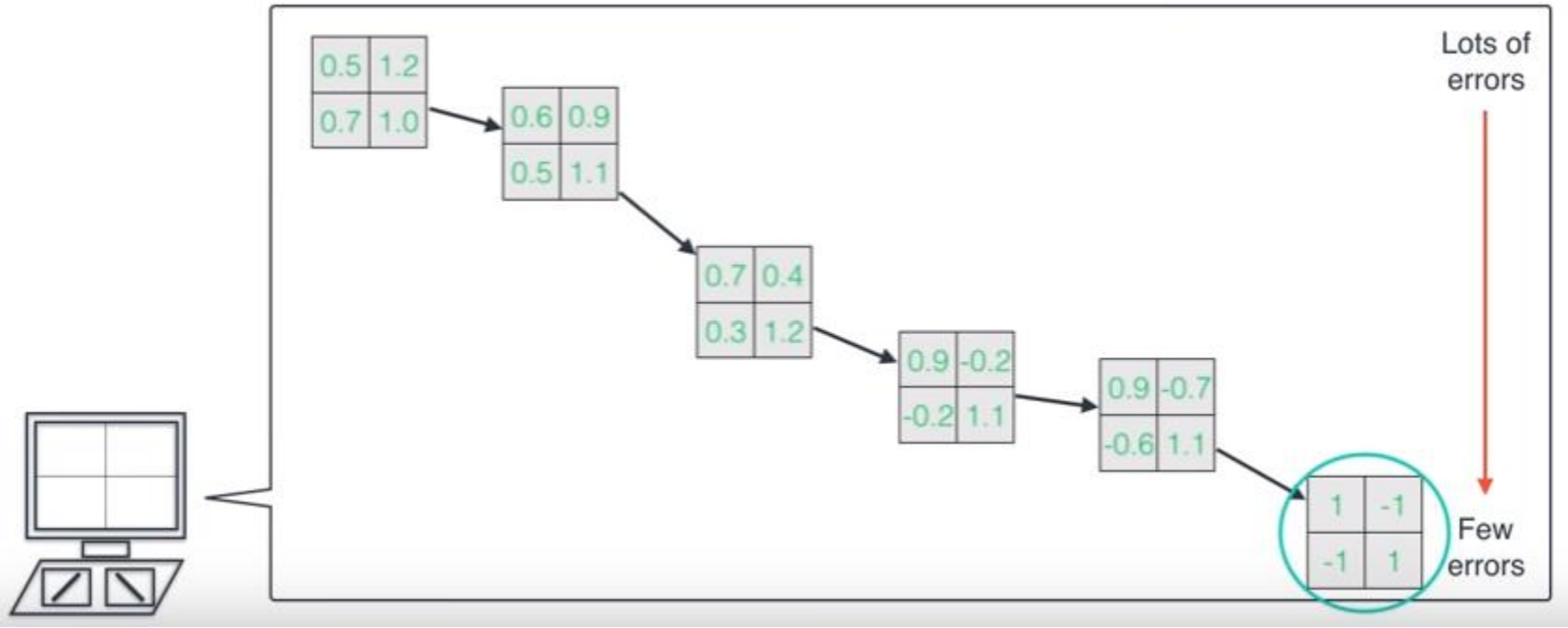
- But in reality, we may have continuous numbers
- Use derivatives of gradient descent – start with four random numbers
- Check how this classifier works

- If it make lots of errors, then change them slightly to see the less error
- Repeat this process until, you get few errors


0.5 1.2
0.7 1.0
0.6 0.9
0.5 1.1
0.7 0.4
0.3 1.2
Lots of errors
Few errors


0.5 1.2
0.7 1.0
0.6 0.9
0.5 1.1
0.7 0.4
0.3 1.2
0.9 -0.2
-0.2 1.1
0.9 -0.7
-0.6 1.1
1 -1
-1 1
Lots of errors
Few errors

Gradient Descent Approach



- Use error function – to find the error in each step
- Use gradient – to get direction to makes less error

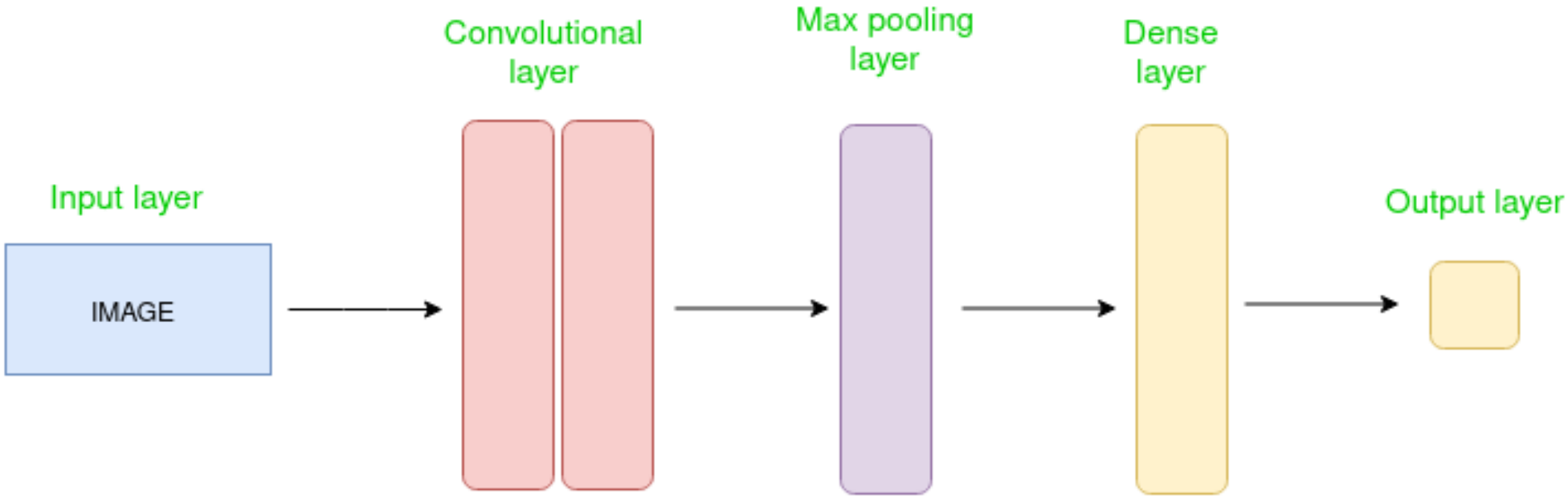
Convolution Neural Network (CNN)

- CNN is the version of artificial neural networks (ANN) which is predominantly used to **extract the feature from the grid-like matrix dataset**.

- Example: visual datasets like images or videos where data patterns play an extensive role.

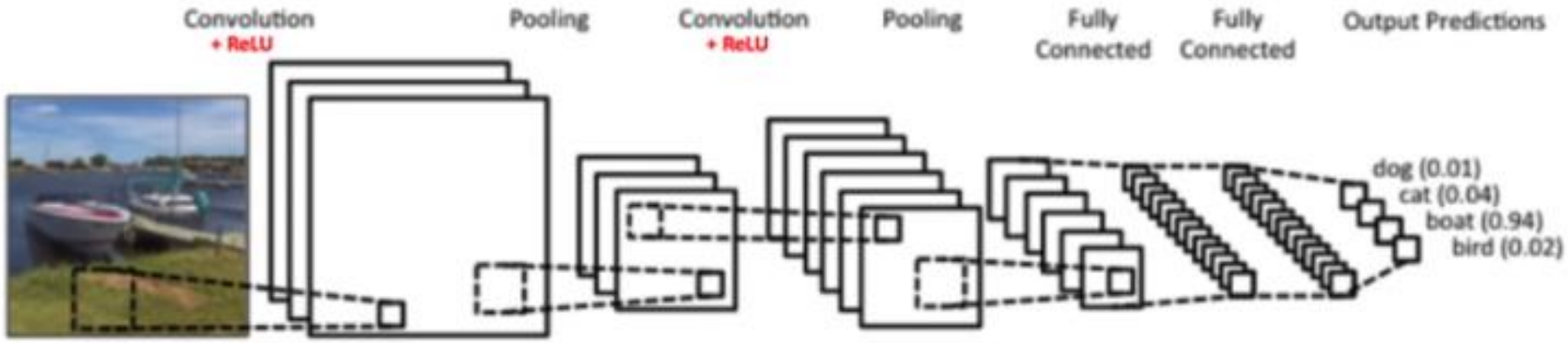
CNN architecture

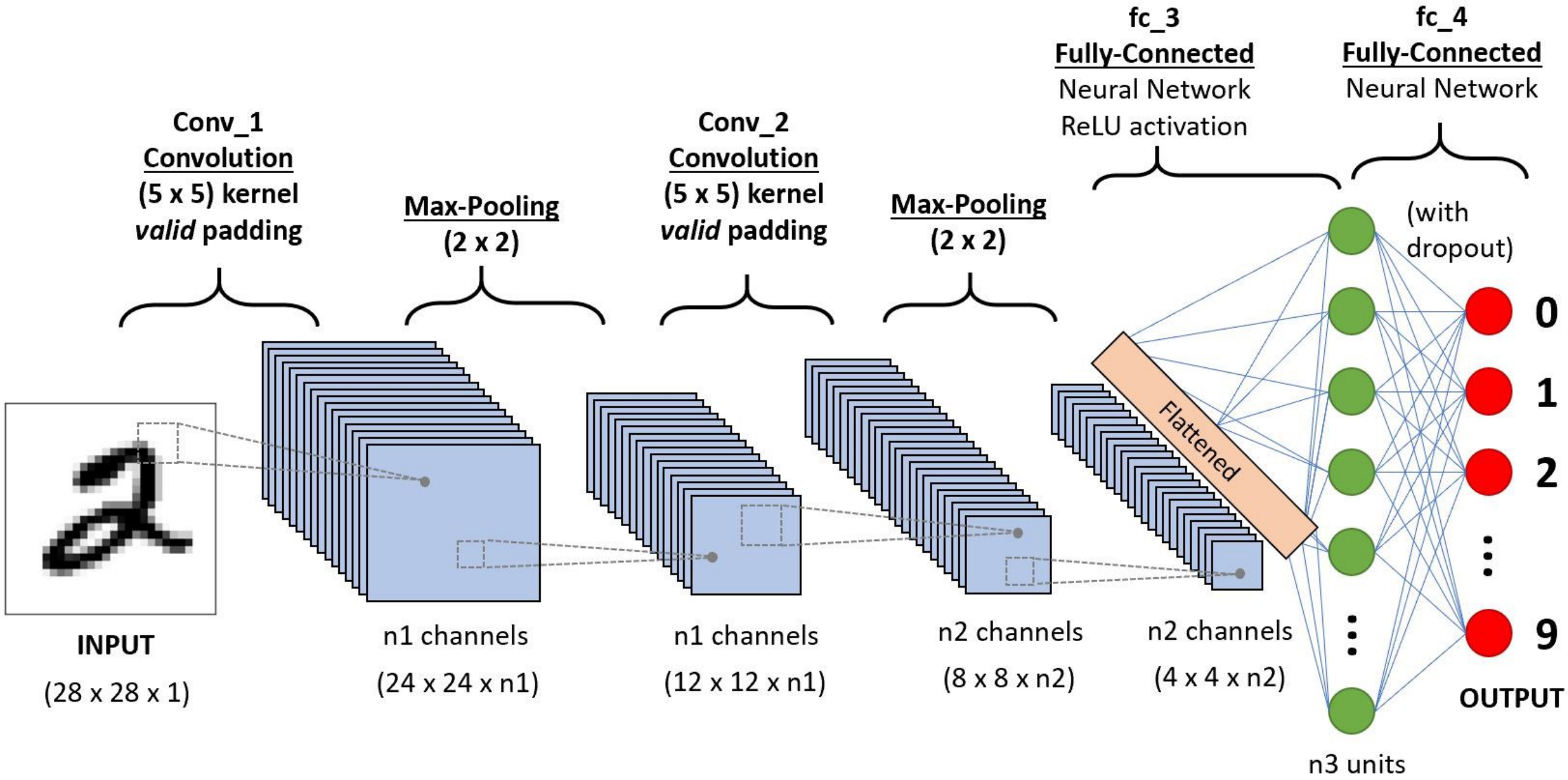
- CNN consists of multiple layers: the input layer, convolutional layer, pooling layer, and fully connected layers.



Components of a Convolutional Network

- The convolutional layer
- The Pooling layer [optional]
- Fully Connected Layer (output layer)




Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

CNN architecture

- Convolutional layer - applies filters to the input image to extract features
- Pooling layer - down-samples the image to reduce computation
- Fully connected layer - makes the final prediction.
- The network learns the optimal filters through backpropagation and gradient descent.

The Convolution Layer

- Input image of size 6*6.
- Filter/ weight matrix 3*3, which extracts certain features from the images

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429

This weight shall now **slides across the image** such that all the pixels are covered at least once, to give a convolved output

The Convolution Layer

- Input image of size 6*6.
- Weight matrix 3*3, which extracts certain features from the images

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

18x1+51x1+121x1+35x1+204x1 = 429

429

This weight shall now slides across the image such that all the pixels are covered at least once, to give a convolved output

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

The Convolution Layer

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

1	0	1
0	1	0
1	0	1

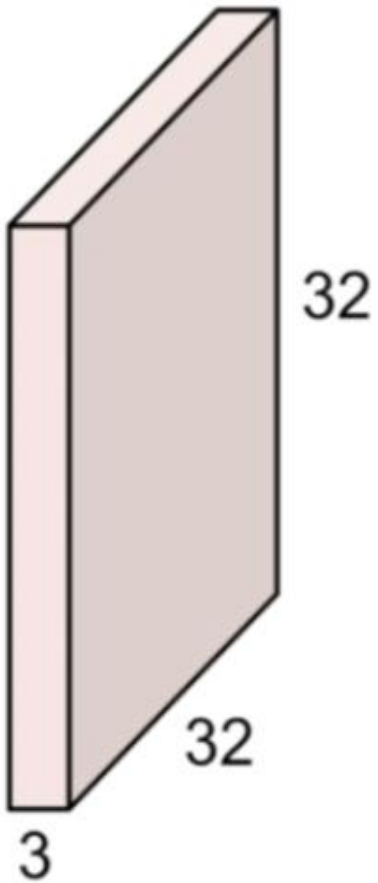
429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

Output size = [(N-F)/Stride] + bias = [(N-F)/1]+1

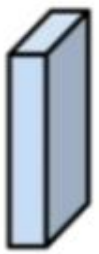
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

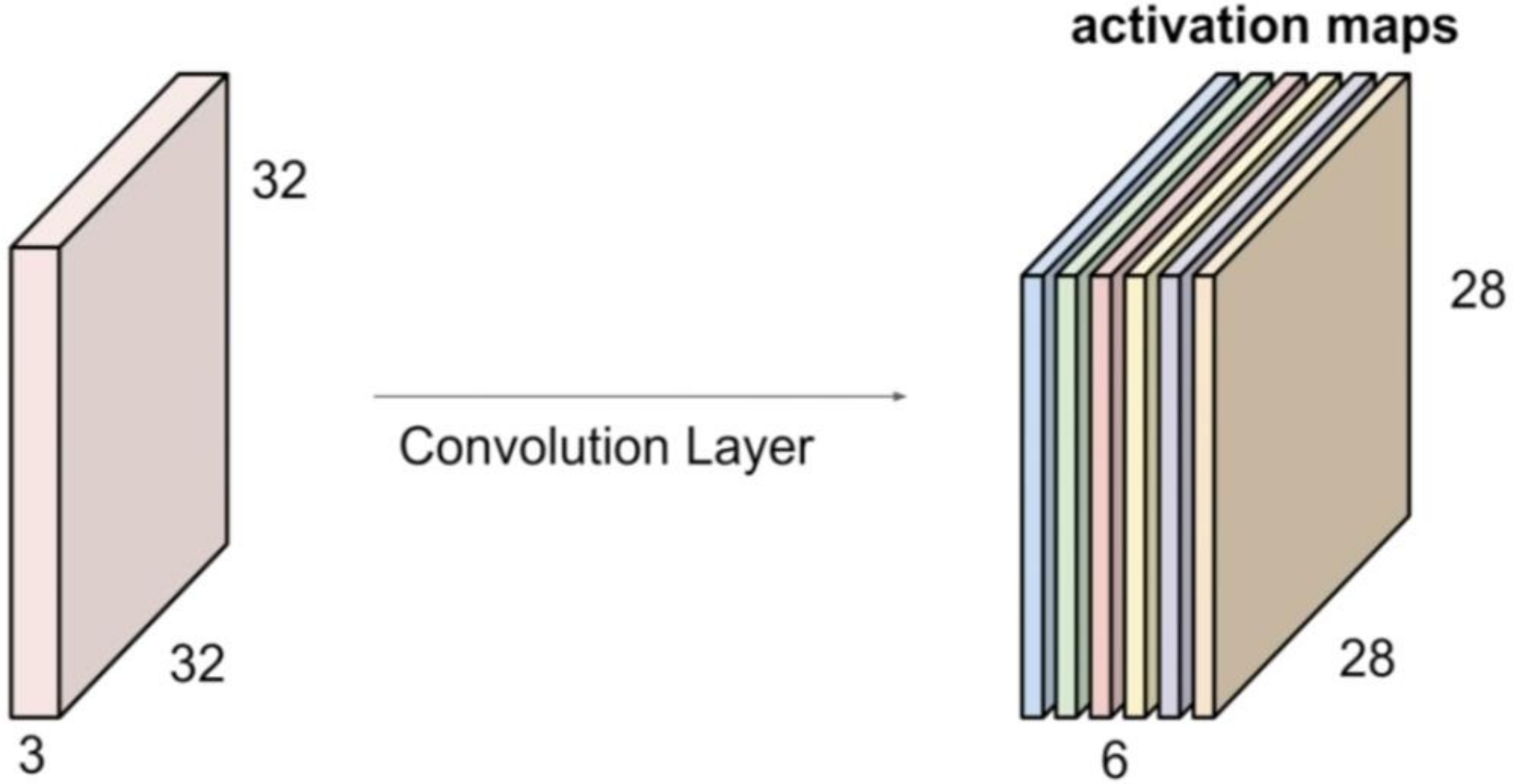


5x5x3 filter

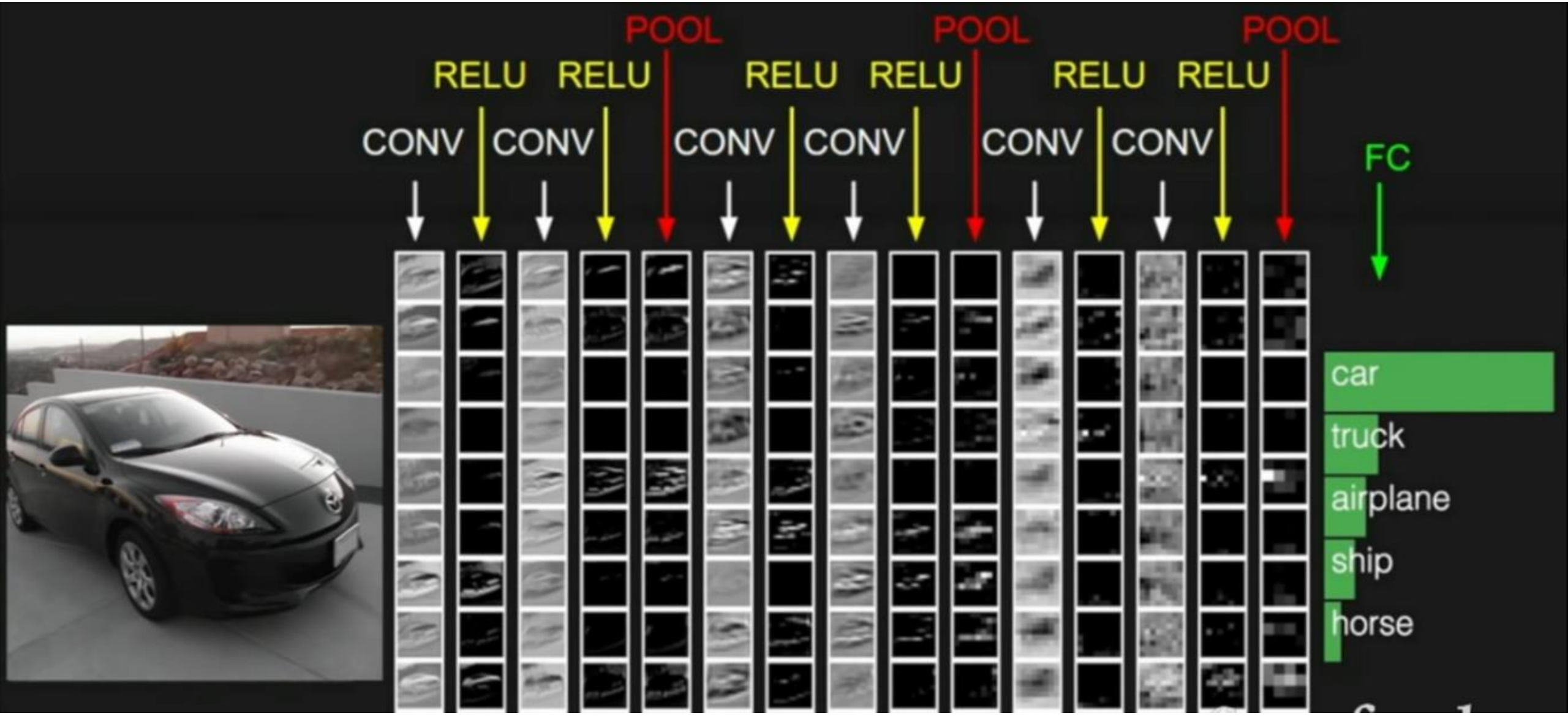


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

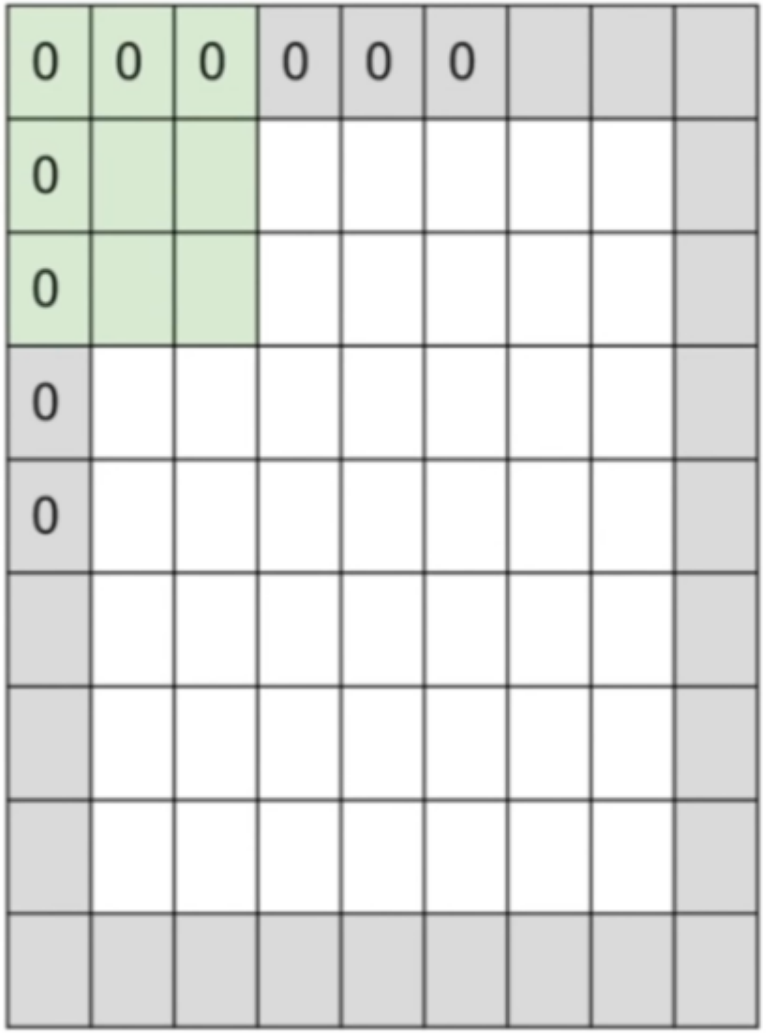
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6


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

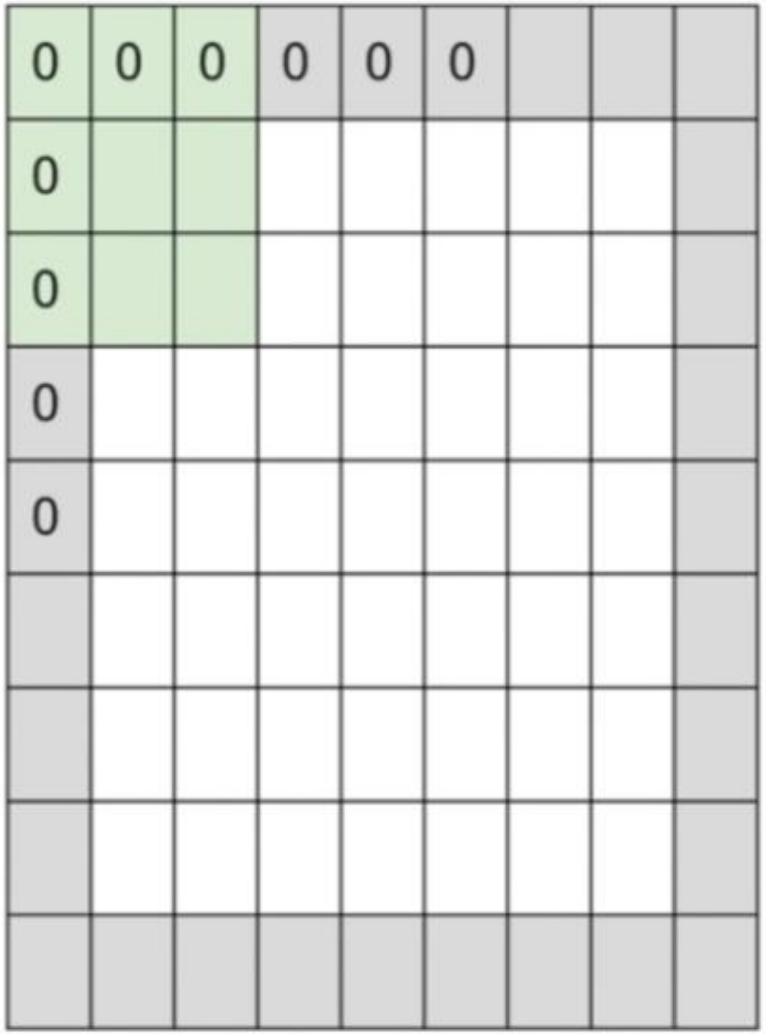
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

Output = (N – F +2P)/S + 1

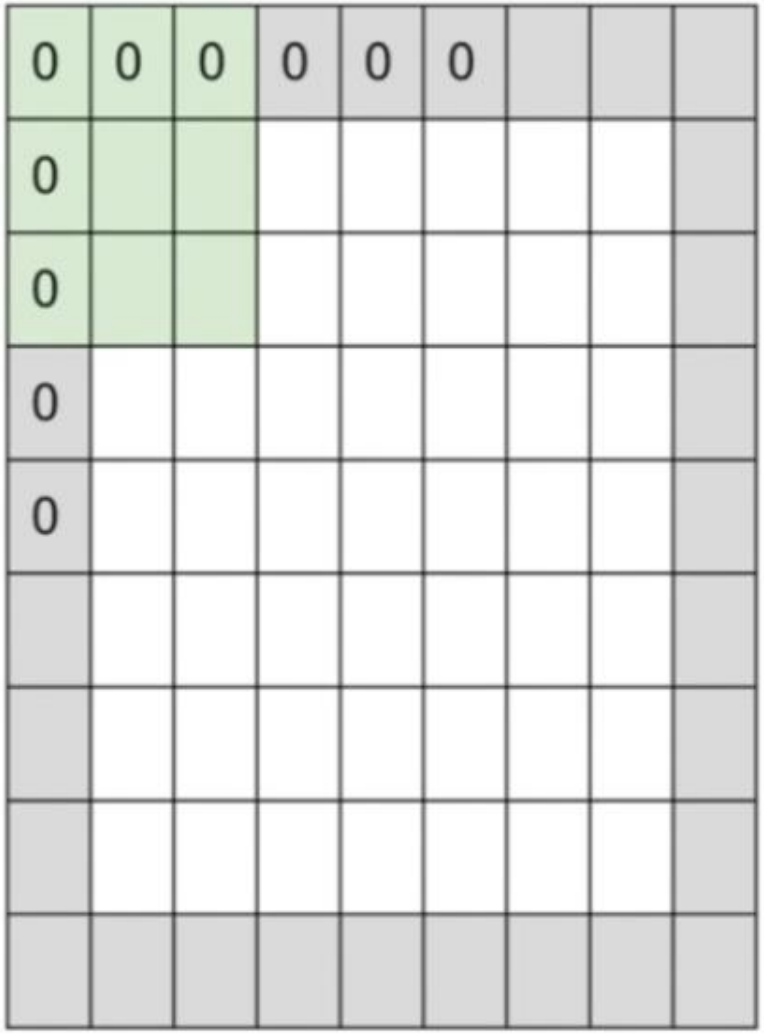
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border



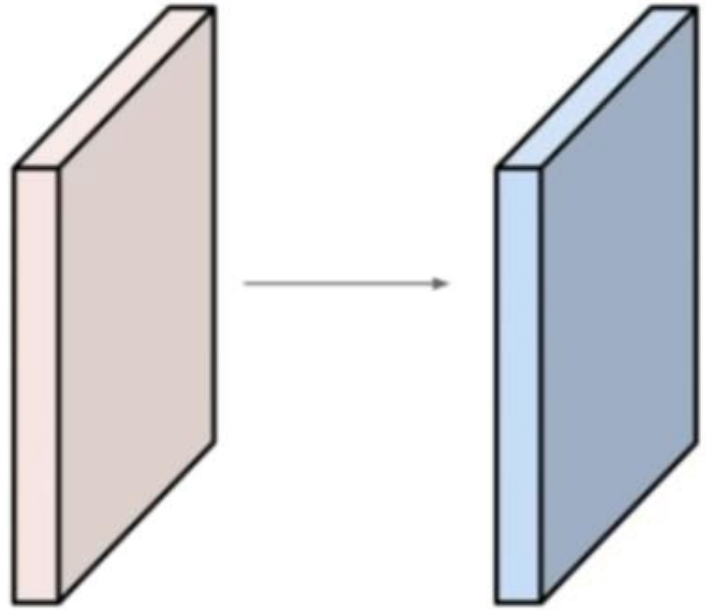
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

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

$$(7 - 3 + 2 \times 1)/ 1 + 1 = 7$$

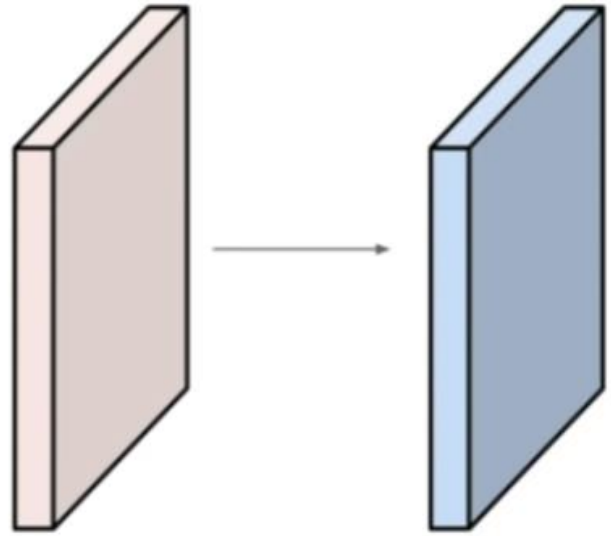
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Examples time:

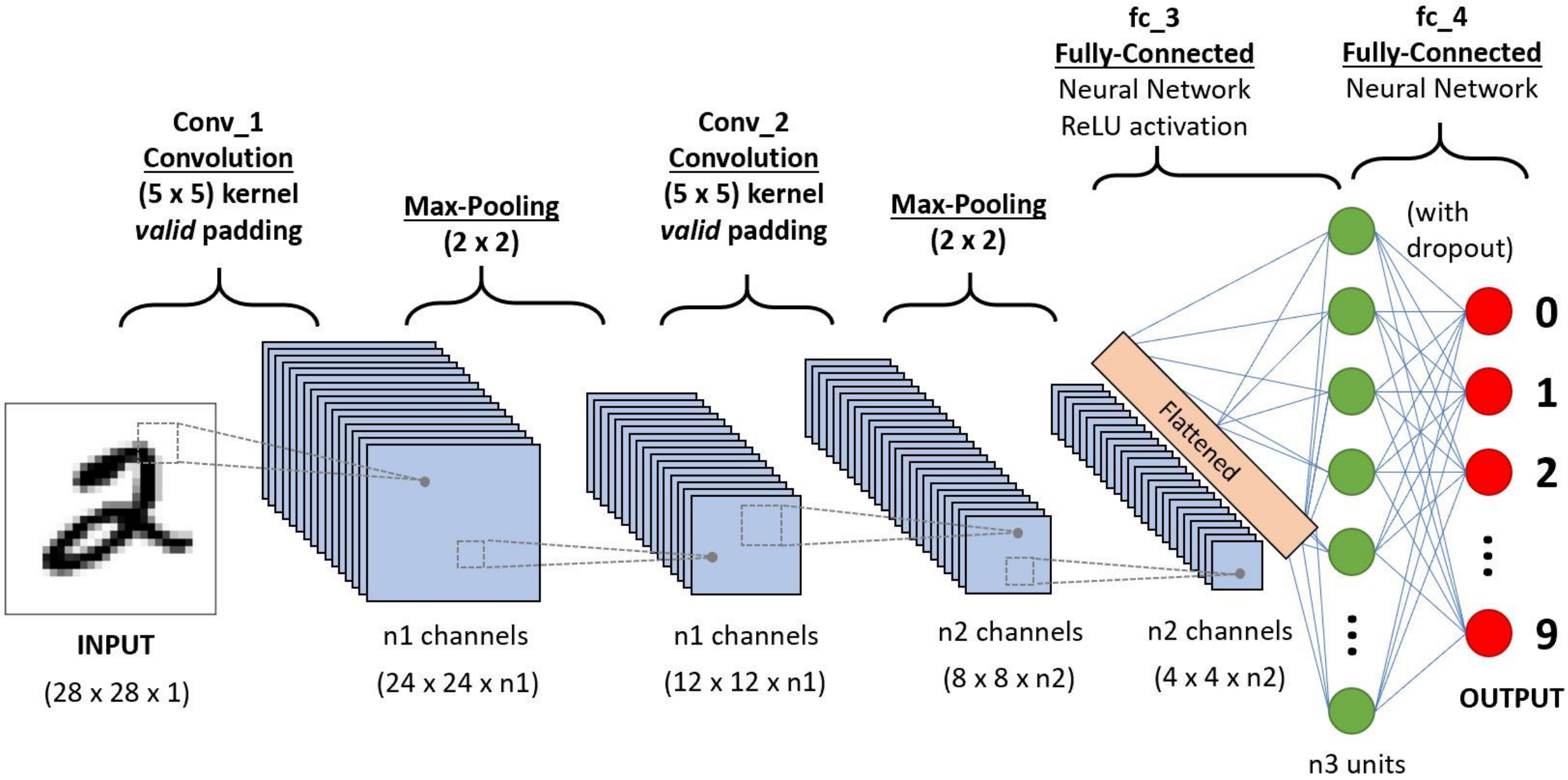


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

The Pooling Layer

- When the images are too large, to reduce the number of trainable parameters, pooling layer is used.
- Periodically introduce pooling layers between subsequent **convolution layers**.
- Pooling is done for the purpose of reducing the spatial size of the image.
- The most common form of pooling layer generally applied is the max pooling.


Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

The Convolution Layer output…

- The 6*6 image is now converted into a 4*4 image.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

Filter/weight

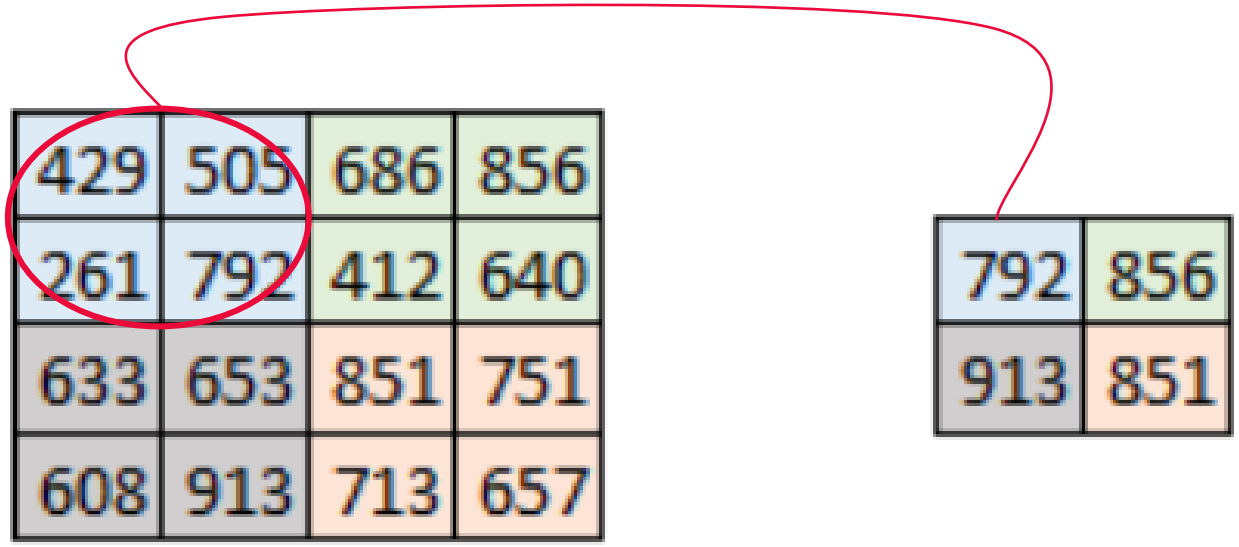
1	0	1
0	1	0
1	0	1

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

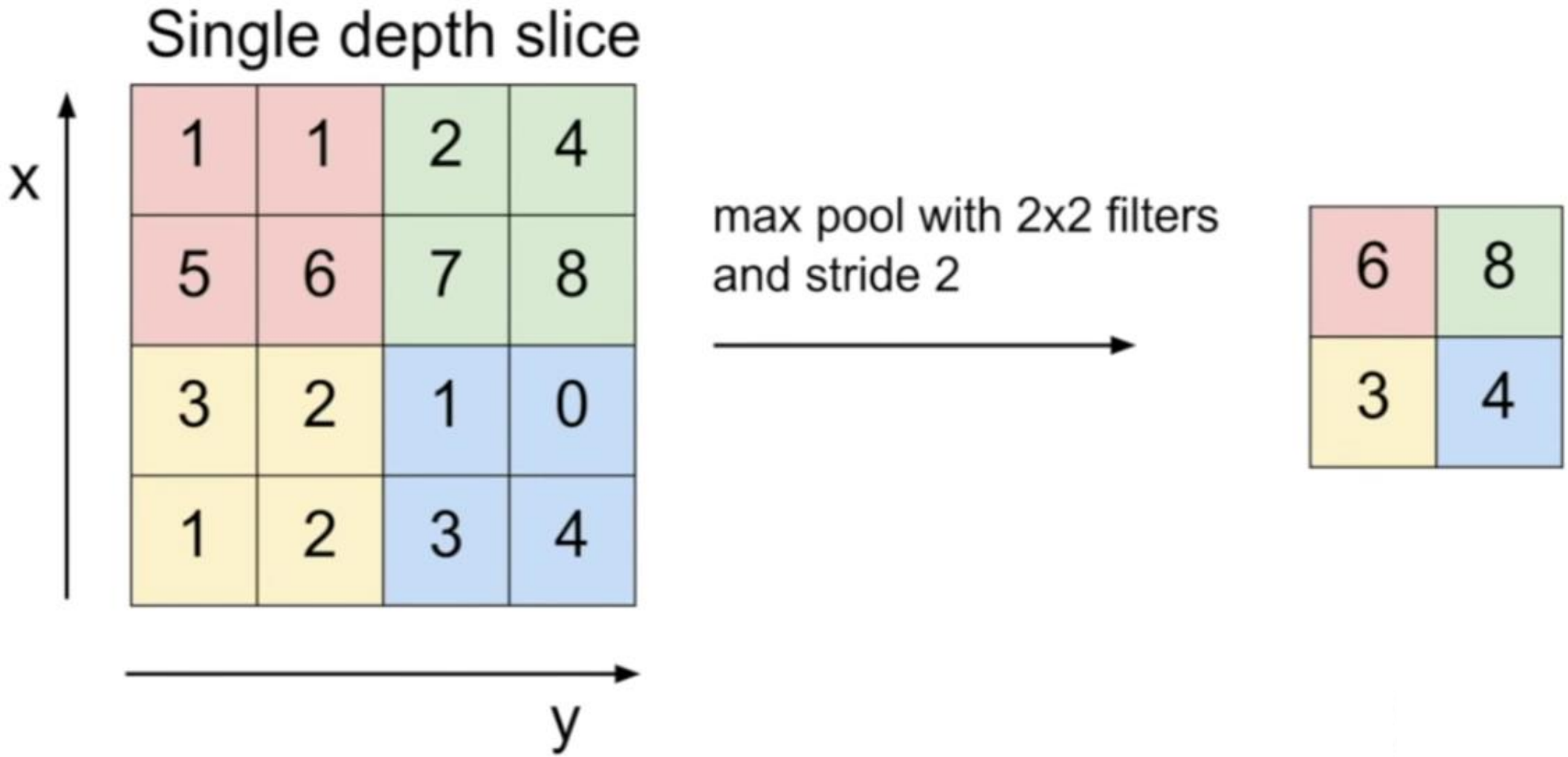
Output size = (N-F)/Stride + bias = (N-F)/1+1

Example

- We have stride as 2, while pooling size also as 2.
- The max operation is applied to each depth dimension of the convolved output.
- The 4*4 convolved output has become 2*2 after the max pooling operation.



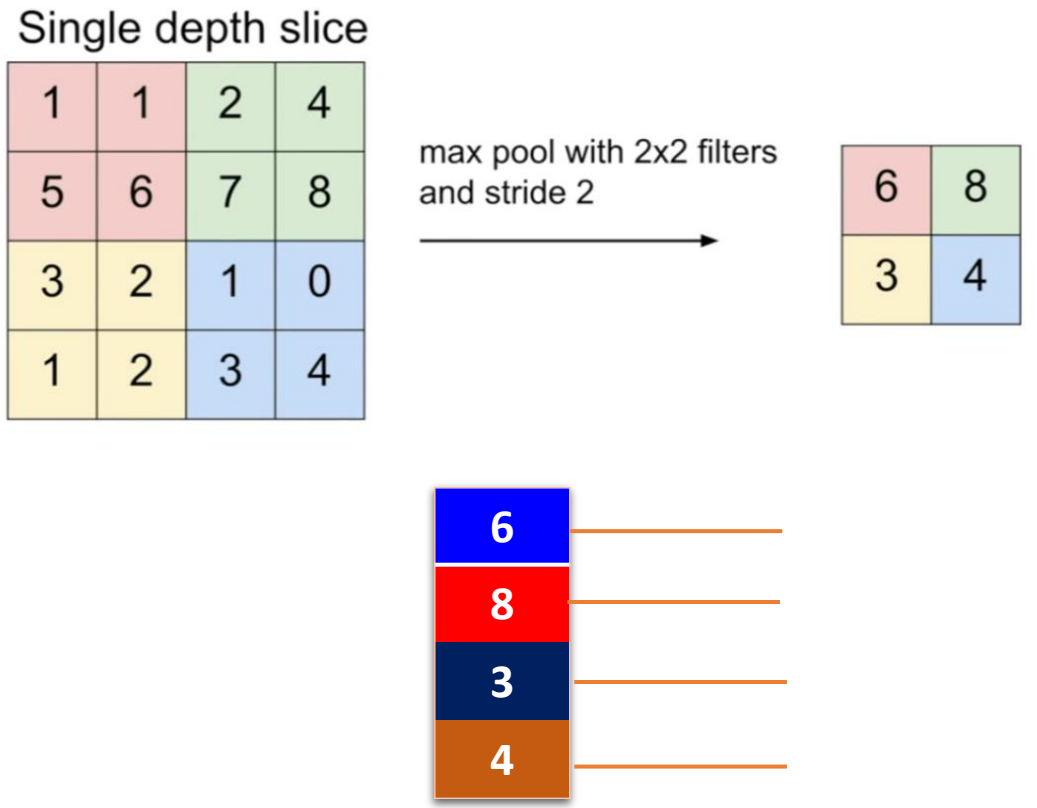
MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

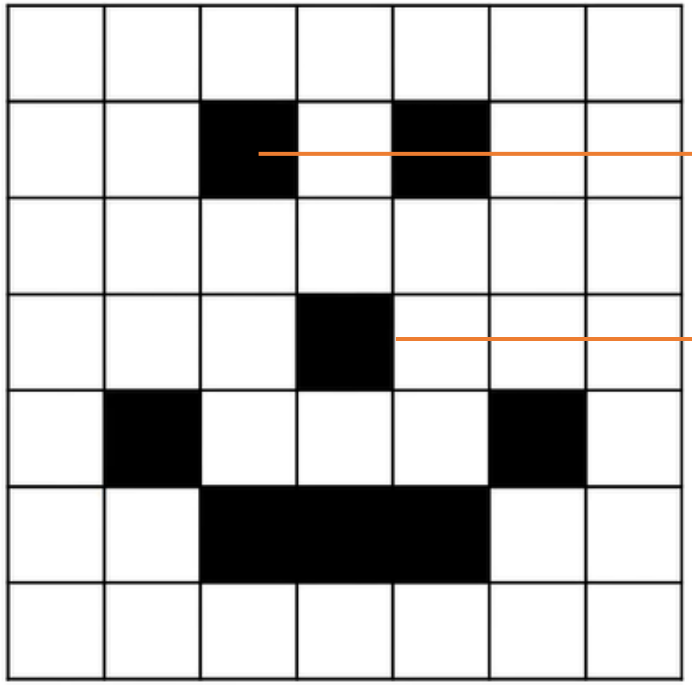
Fully Connected & Output layer

- After multiple layers of convolution and pooling, we need the output in the form of a **class**.
  - **convolution layers** would only be able to extract features
  - **pooling layers** reduce the number of parameters from the original images

- **Fully connected layer** to generate an output equal to the number of classes we need.



Simple Input Image - Happy Face



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

To simplify, let's take a simple black and white image with 0 and 1 pixels

Step 1 - Convolution

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

Input Image

7x7

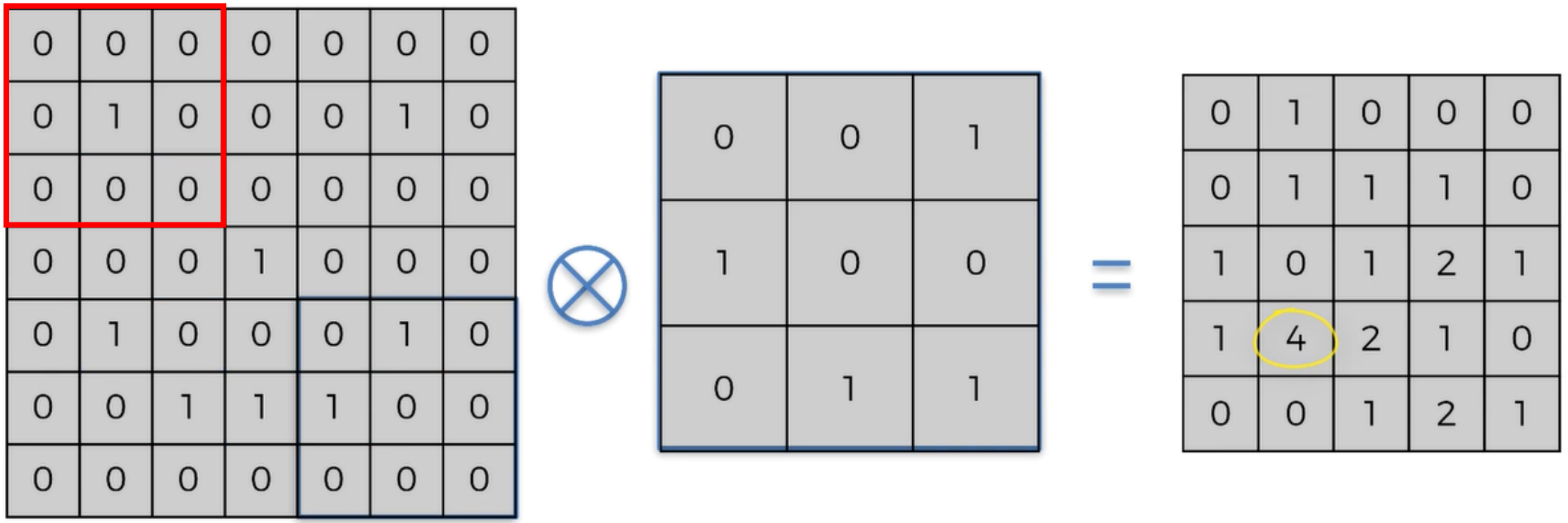
0	0	1
1	0	0
0	1	1

Filter

Feature Detector

3x3

Step 1 - Convolution



Input Image

7x7
Stried of 1

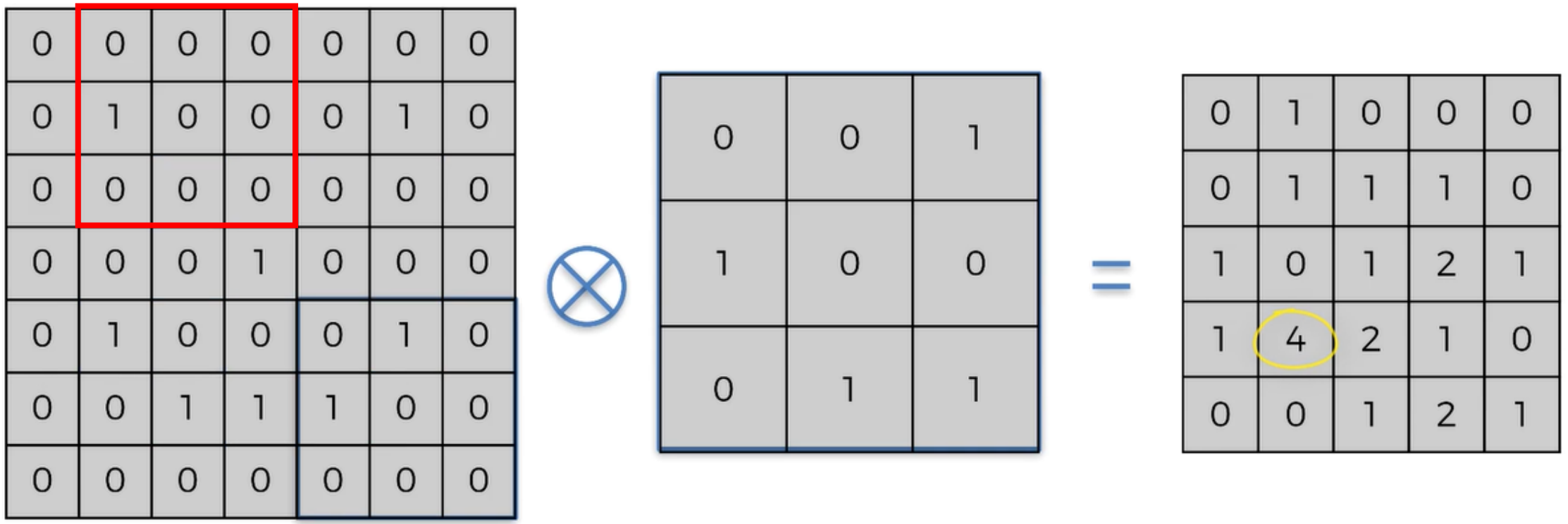
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



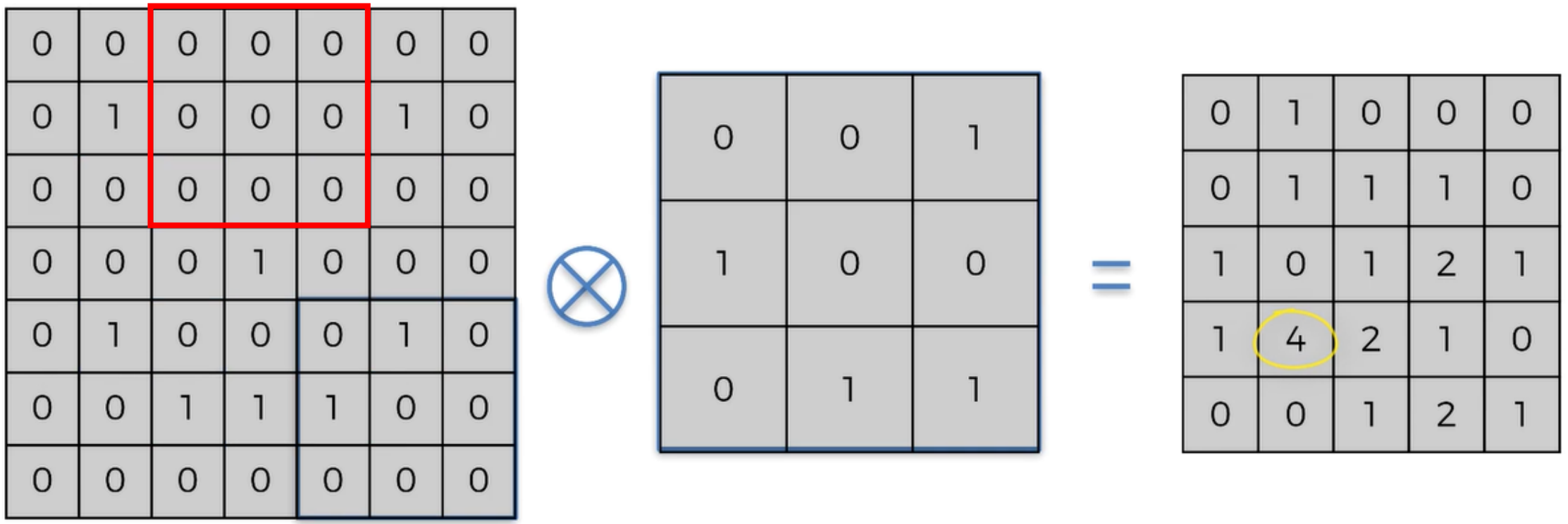
Input Image
7x7

Feature Detector
3x3

Feature Map
5x5

Stried of 1

Step 1 - Convolution



Input Image

7x7

Stried of 1

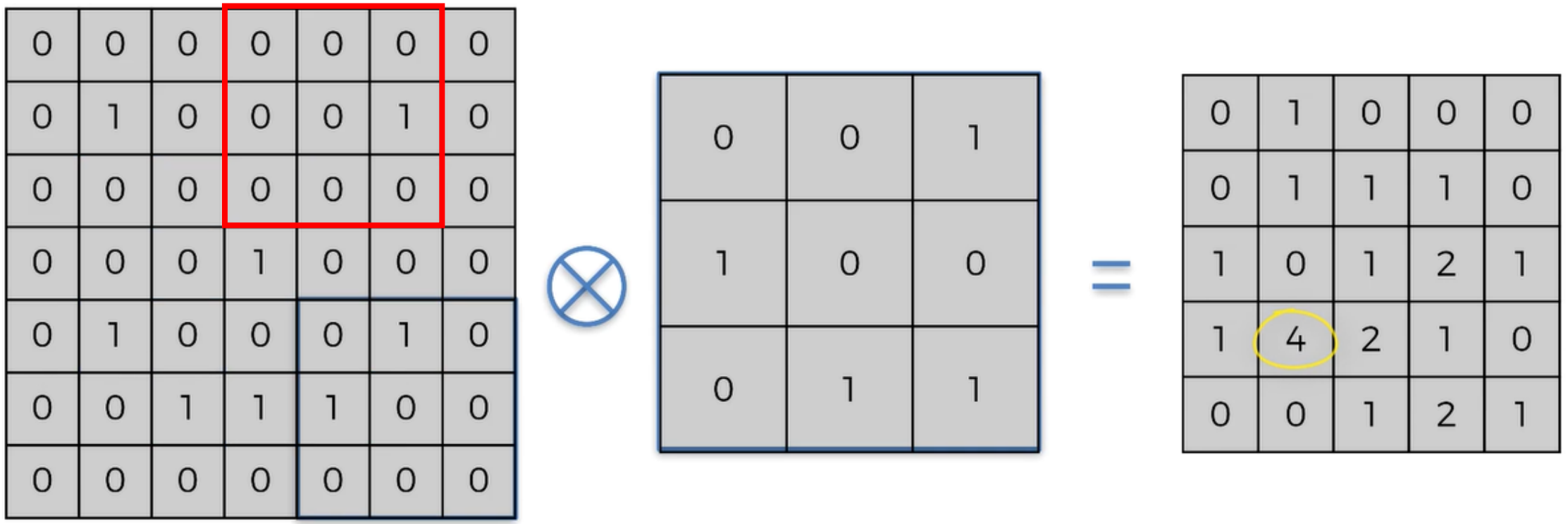
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

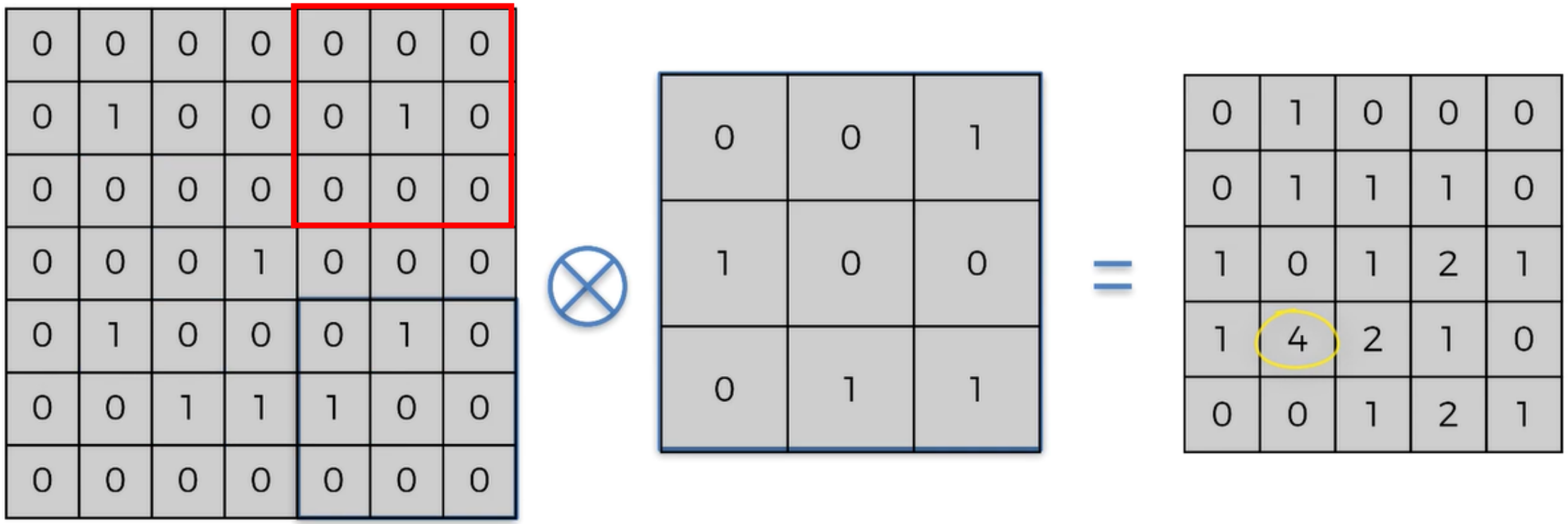
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

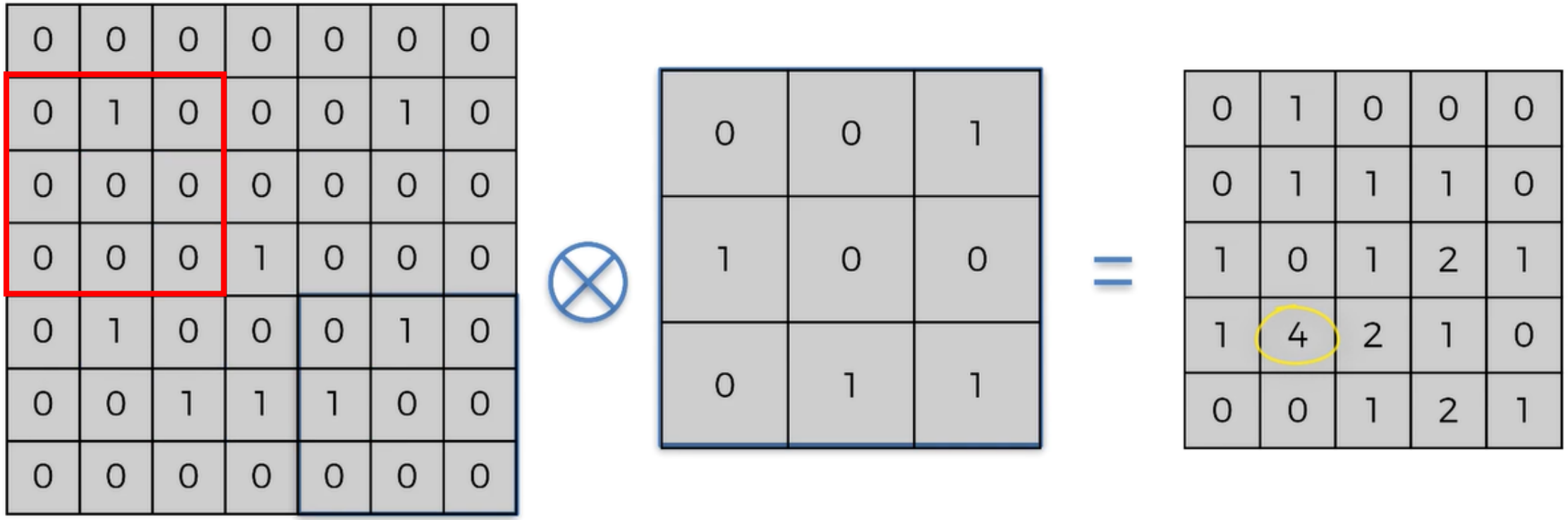
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

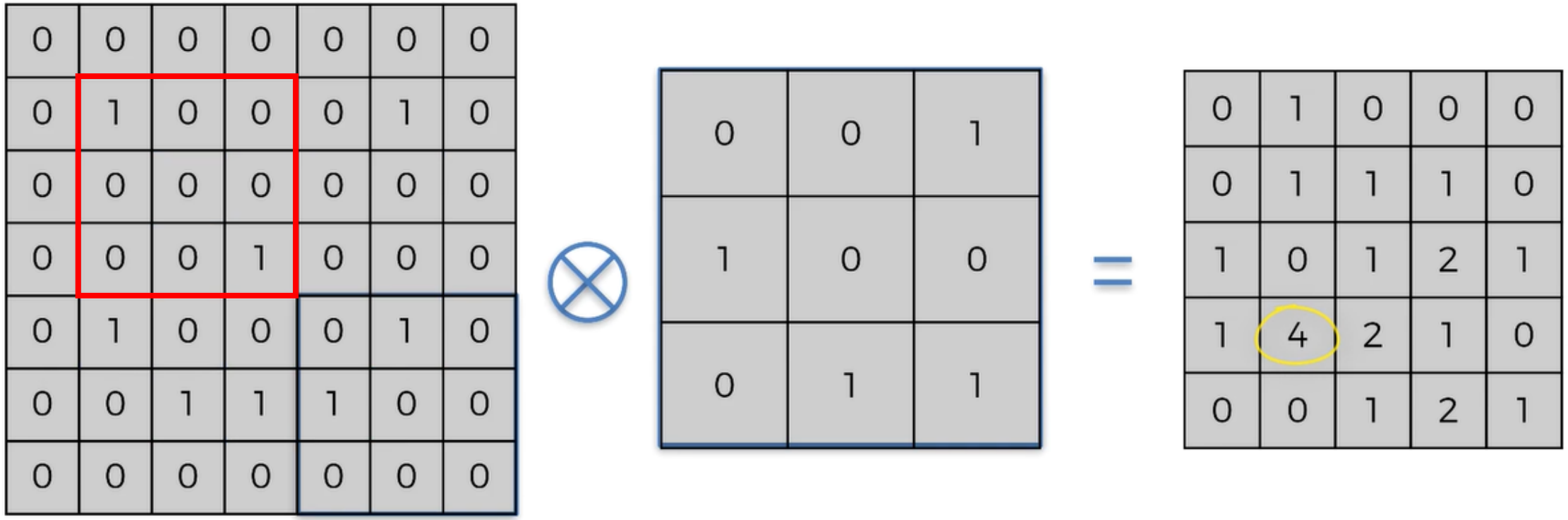
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

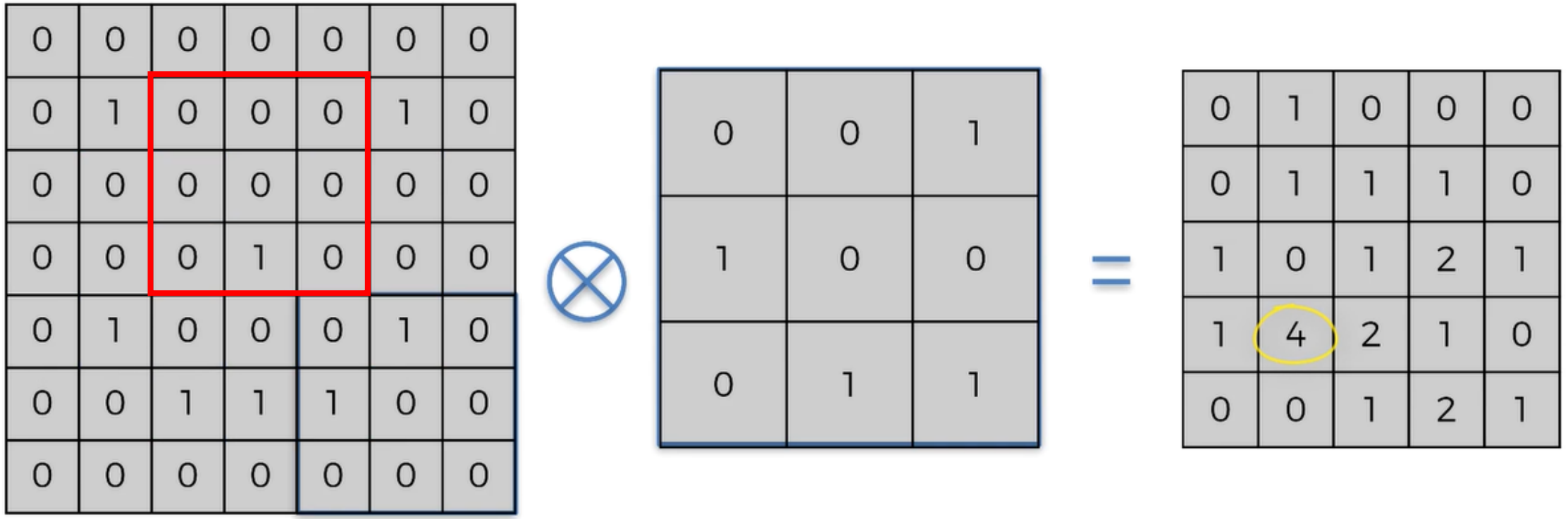
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

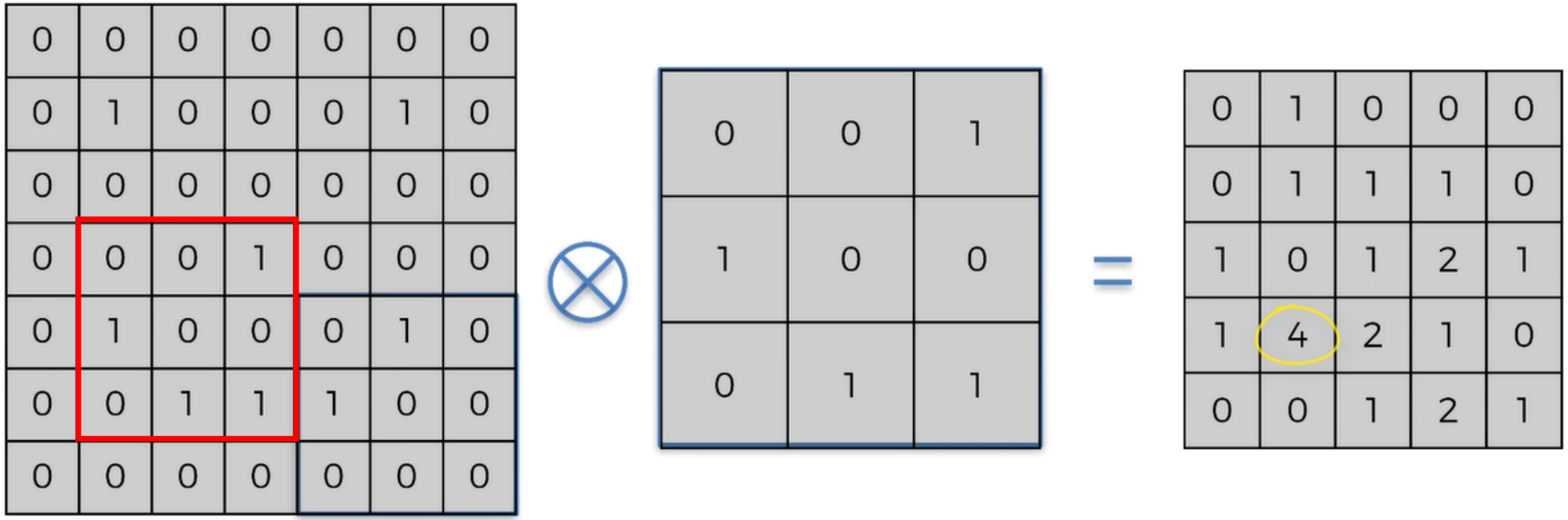
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

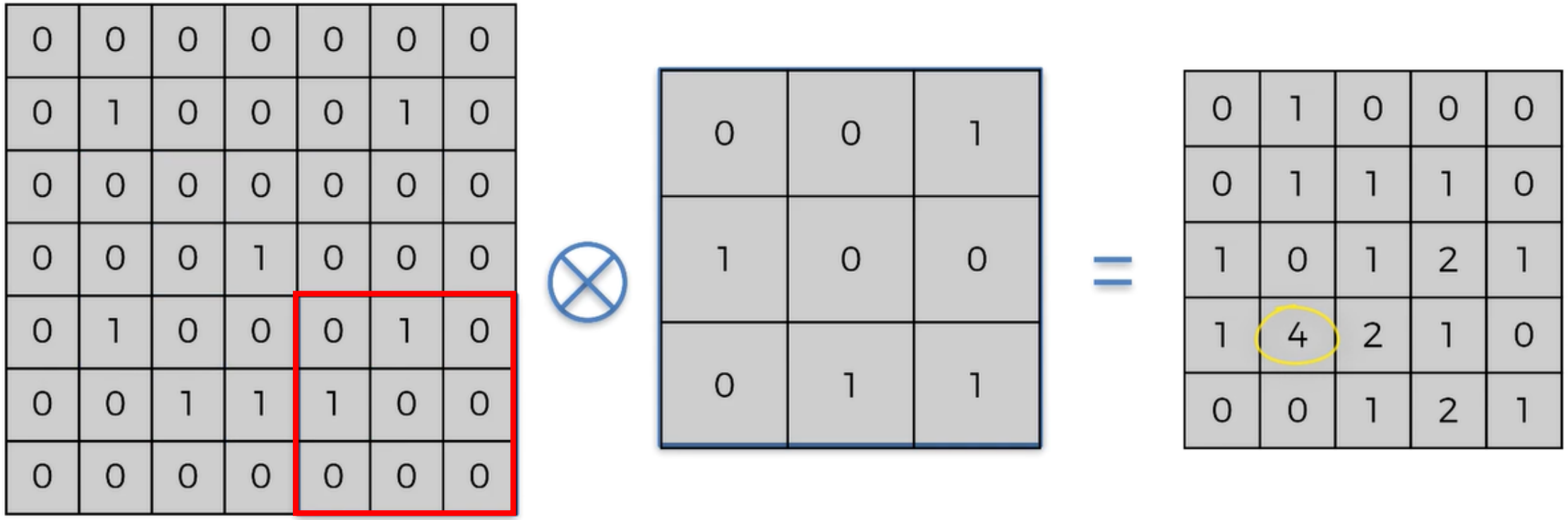
Feature Detector

3x3

Feature Map

5x5

Step 1 - Convolution



Input Image

7x7

Stried of 1

Feature Detector

3x3

Feature Map

5x5

Convolutional Neural Network

Initialising the CNN

classifier = Sequential()

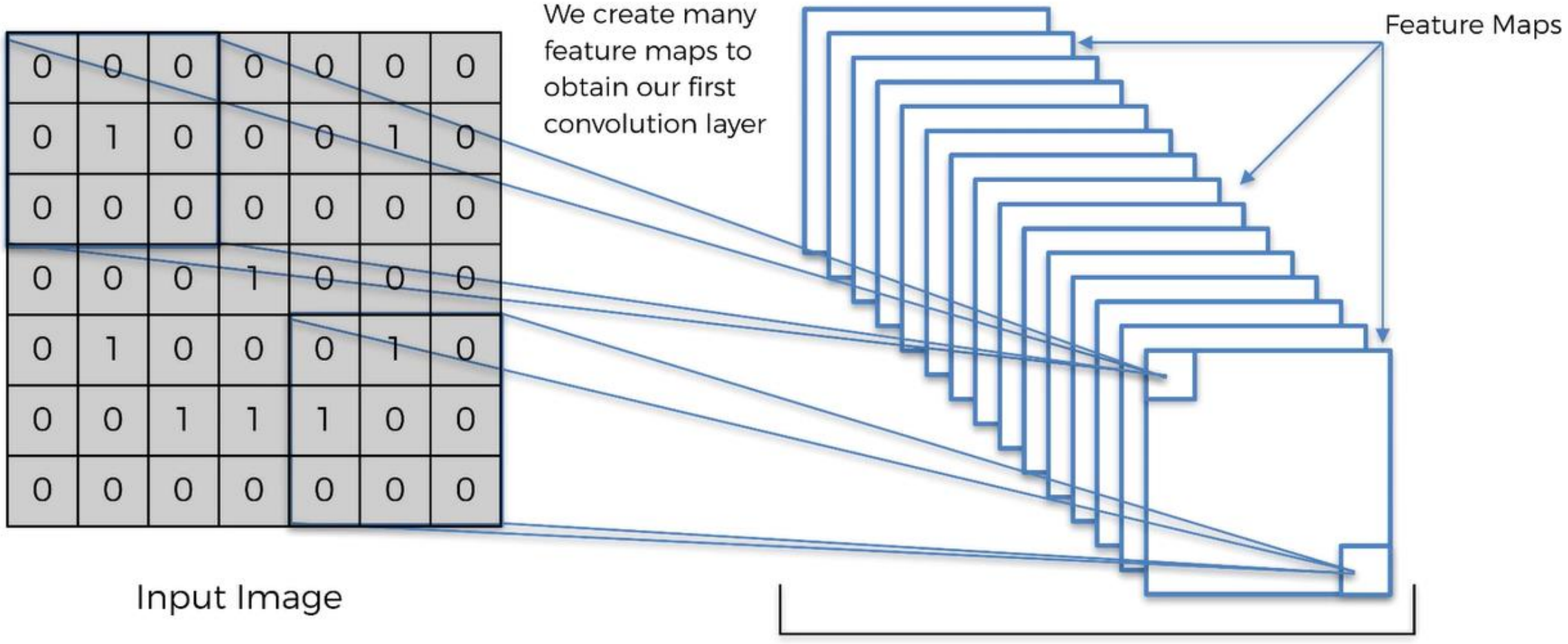
Step 1 - Convolution

32 filters of each 3x3 size

```
classifier.add(Convolution2D(32, 3, 3,
input_shape = (64, 64, 3), activation = 'relu'))
```

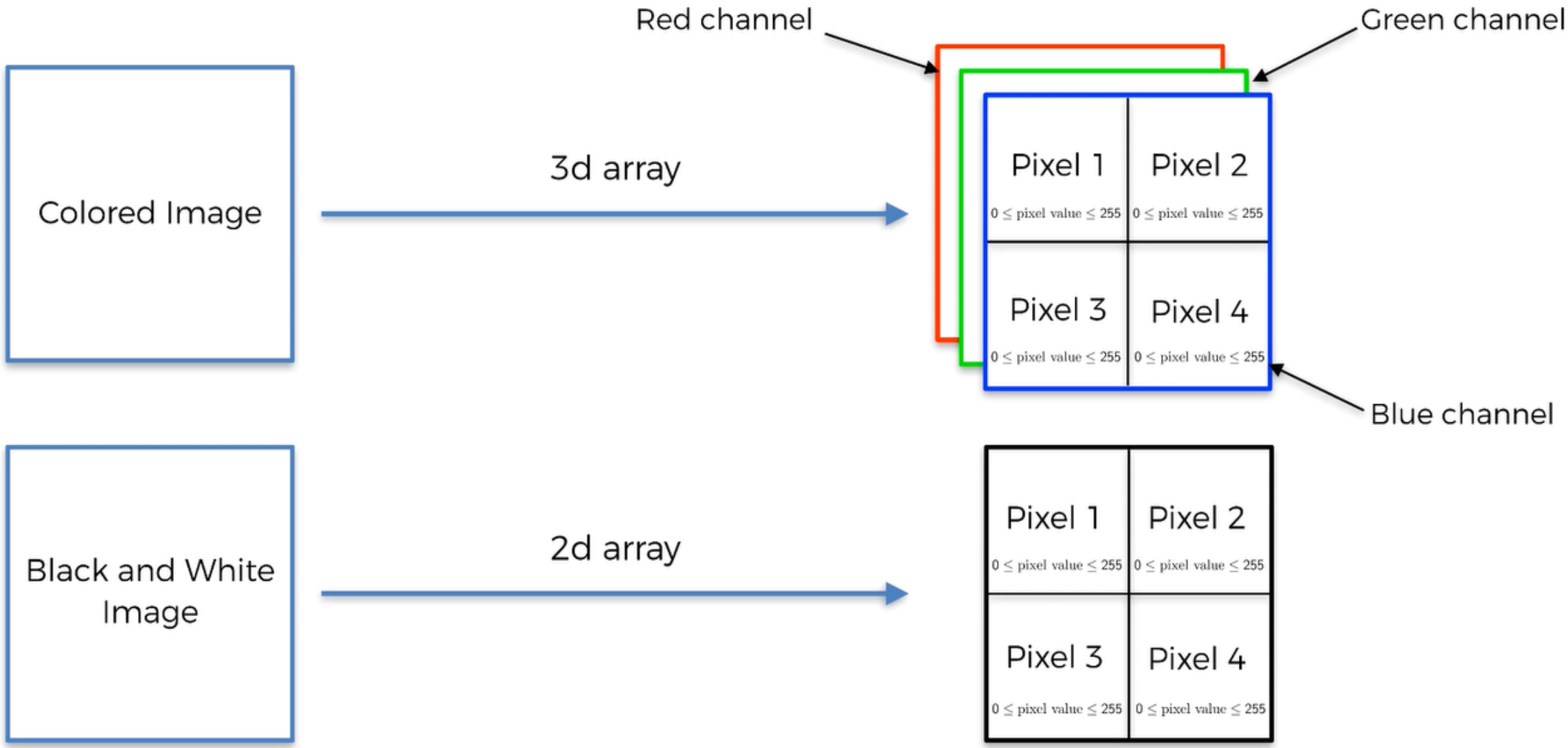
Input size of 64x64 RGB color image

Step 1 - Convolution

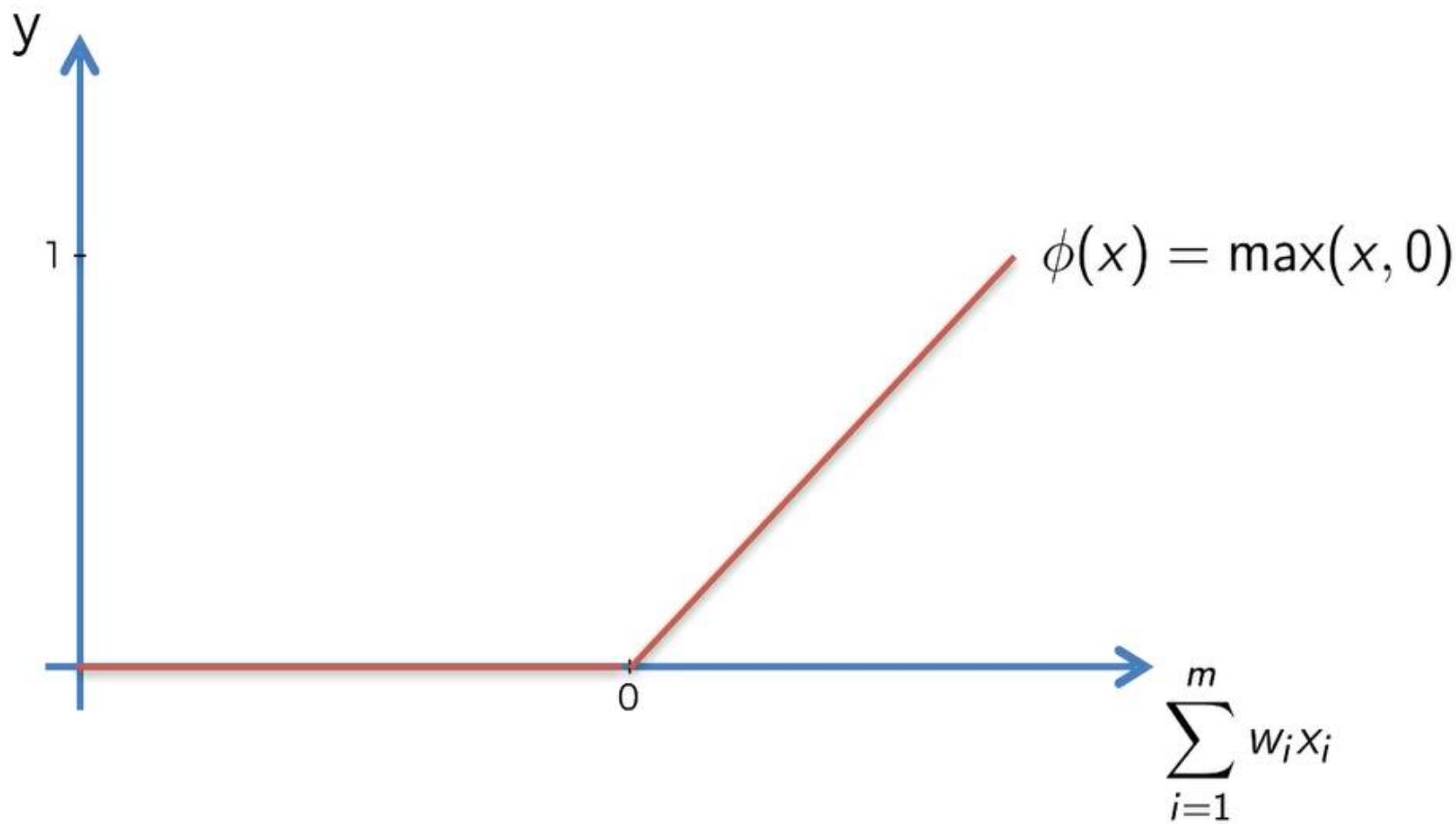



```
Image(64, 64, 3) filter(32, 3, 3);
size of the filtered image = (64-3+2*p=1)/s=1+b=1
= 64
```

Input Images are converted into arrays



Activation function



Convolutional Neural Network

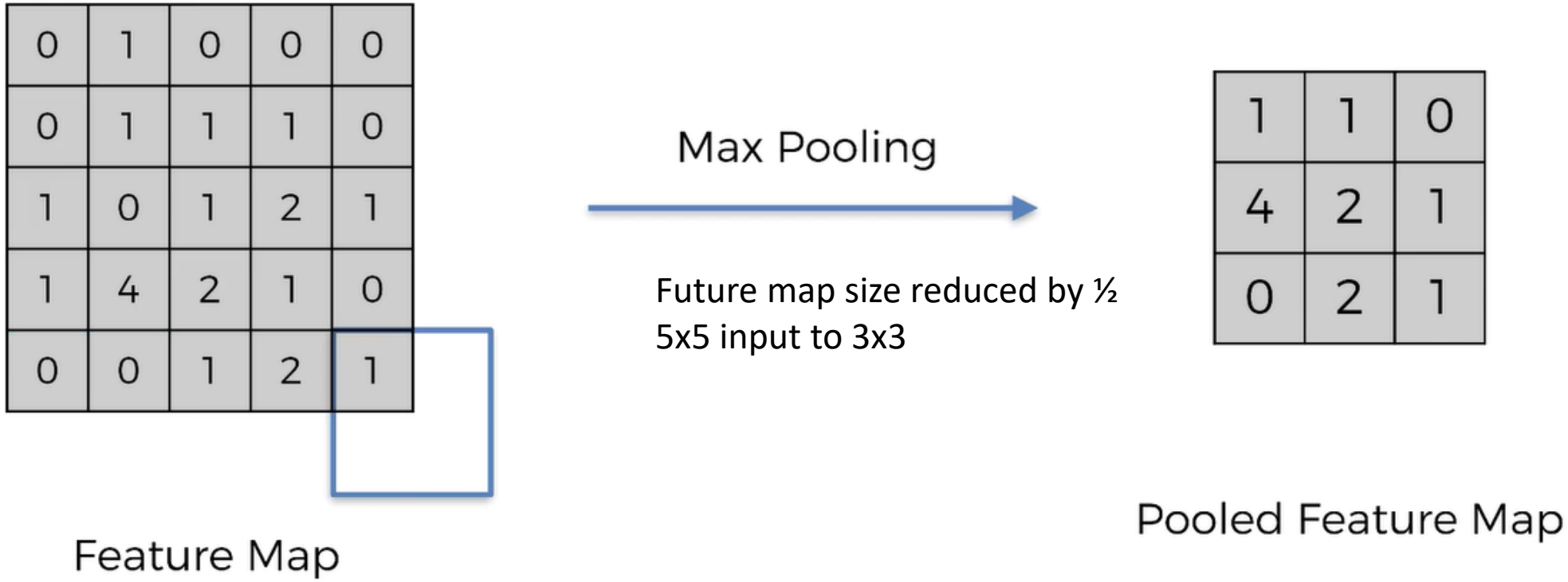
Initialising the CNN

classifier = Sequential()

Step 1 - Convolution

```
classifier.add(Convolution2D(32, 3, 3,
input_shape = (64, 64, 3), activation = 'relu'))
```

Step 2 - Max Pooling




```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

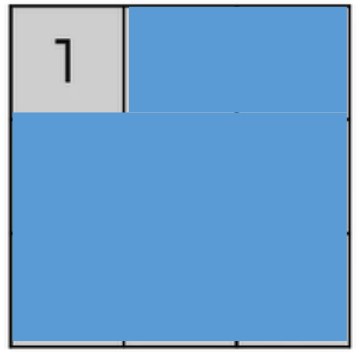
Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Max Pooling

Future map size reduced by ½
5x5 input to 3x3



Pooled Feature Map

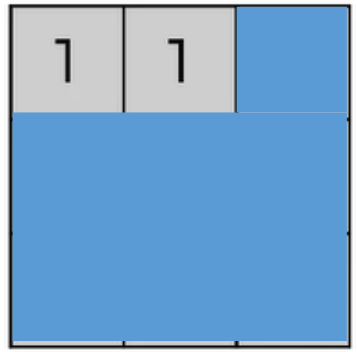
Step 2 - Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

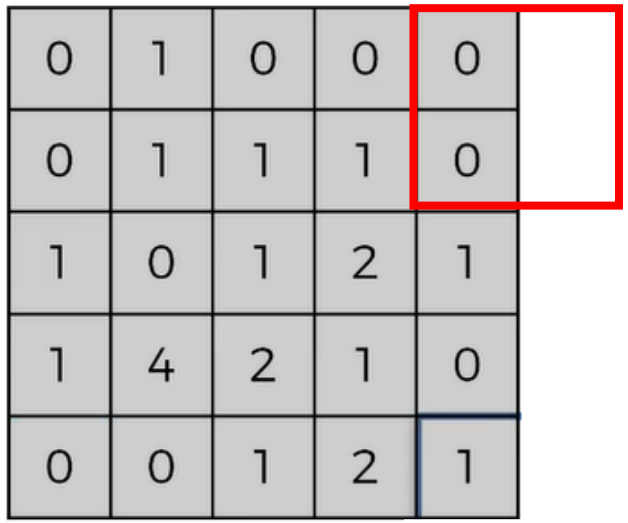
Max Pooling

Future map size reduced by ½
5x5 input to 3x3



Pooled Feature Map

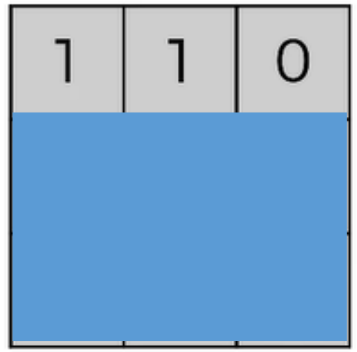
Step 2 - Max Pooling



Feature Map

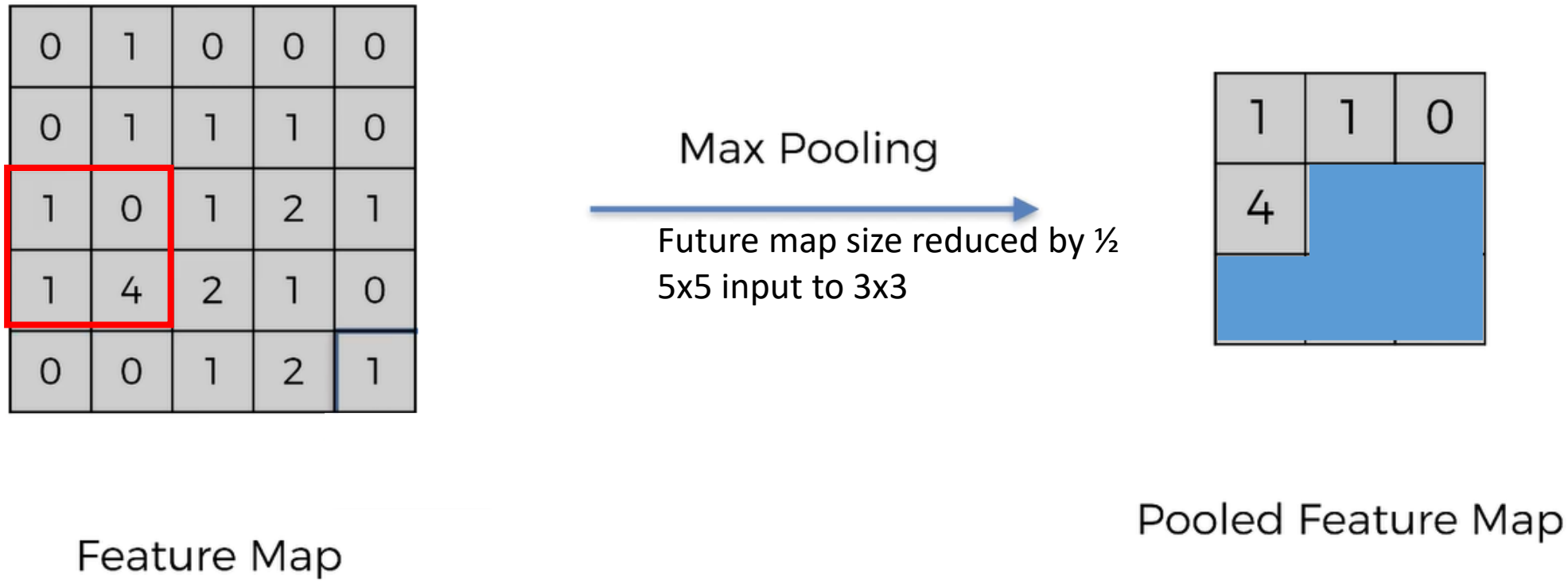
Max Pooling

Future map size reduced by ½
5x5 input to 3x3

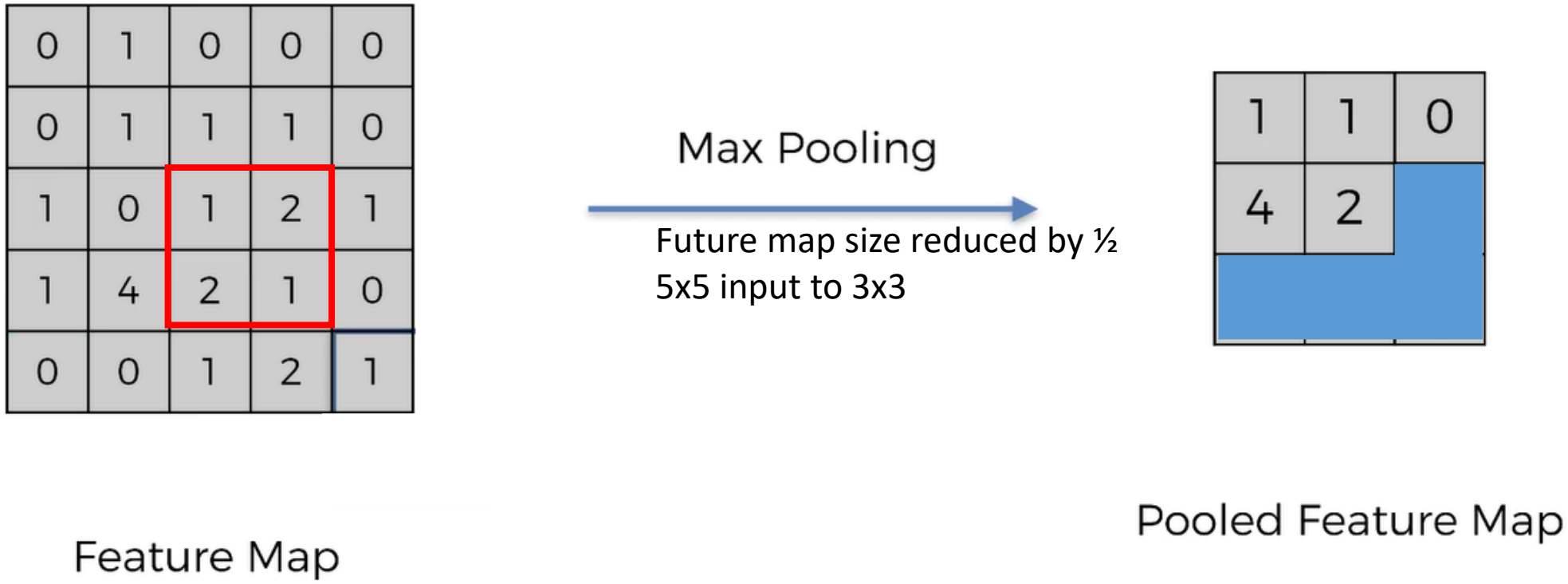


Pooled Feature Map

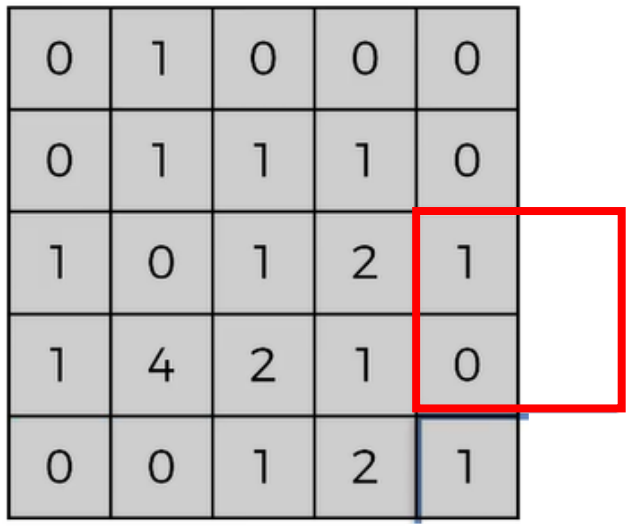
Step 2 - Max Pooling



Step 2 - Max Pooling



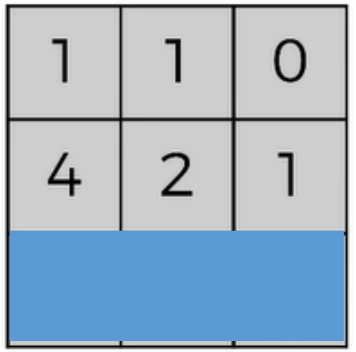
Step 2 - Max Pooling



Feature Map

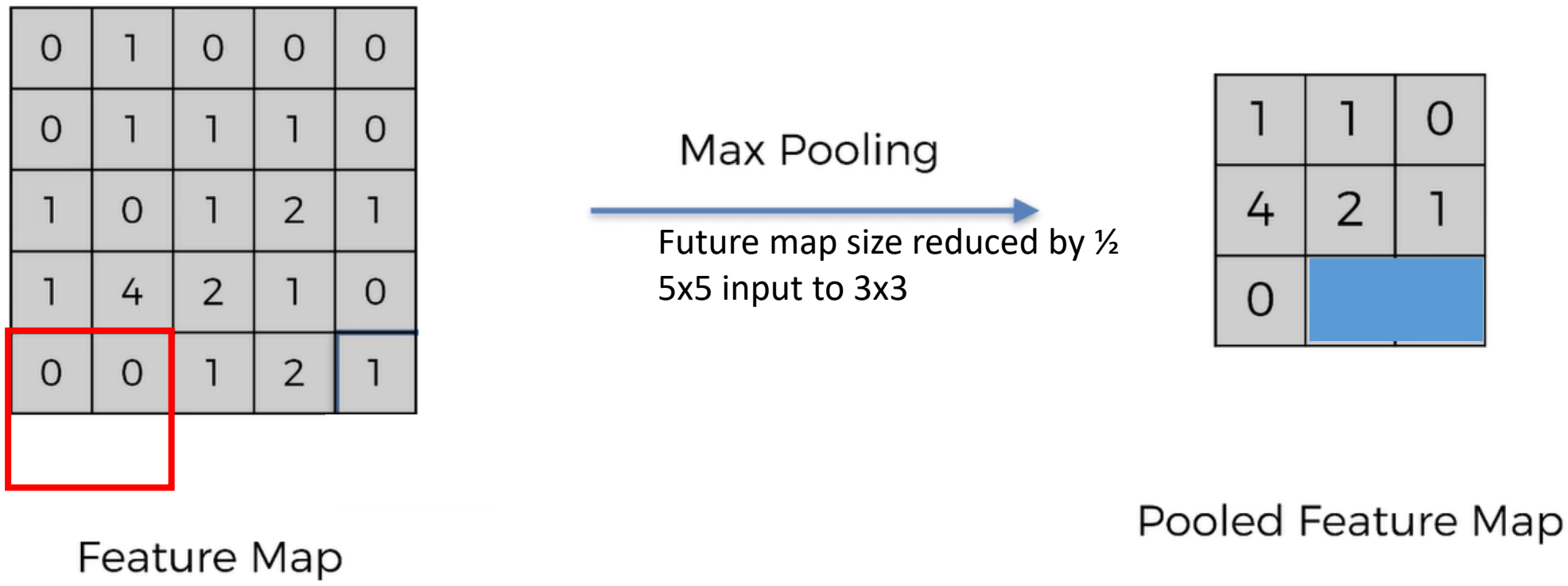
Max Pooling

Future map size reduced by ½
5x5 input to 3x3

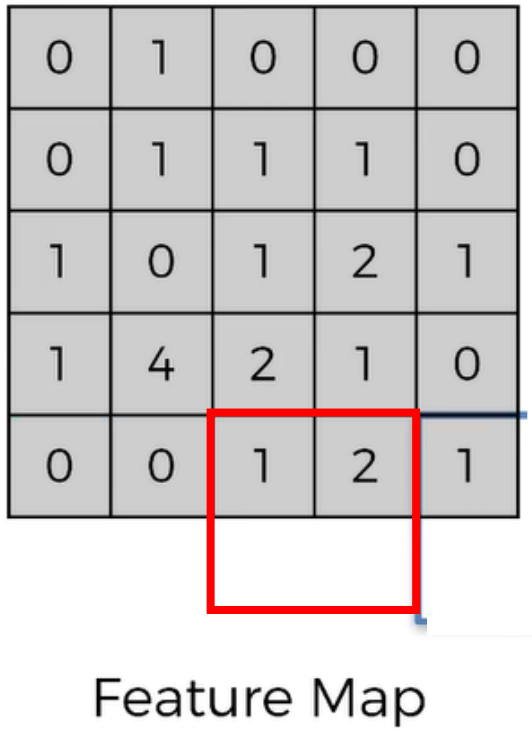


Pooled Feature Map

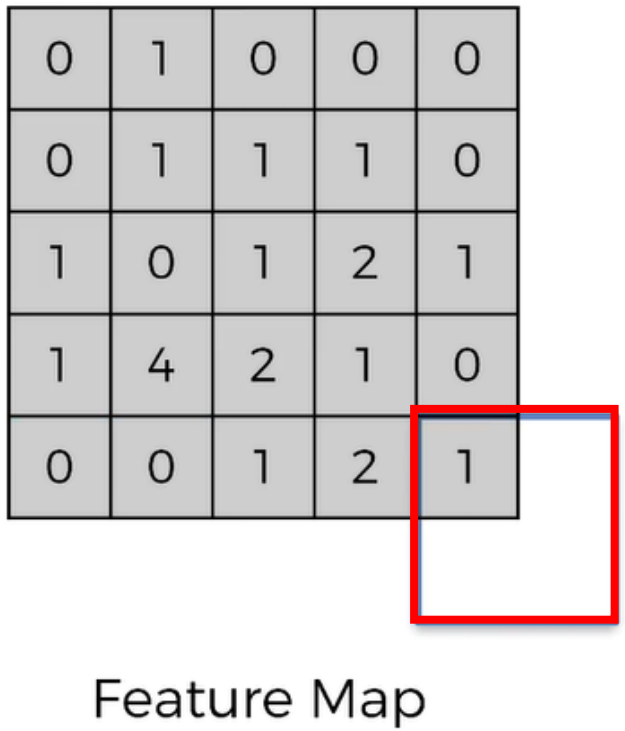
Step 2 - Max Pooling



Step 2 - Max Pooling



Step 2 - Max Pooling



Feature Map

Max Pooling

Future map size reduced by ½
5x5 input to 3x3

1	1	0
4	2	1
0	2	1

Pooled Feature Map

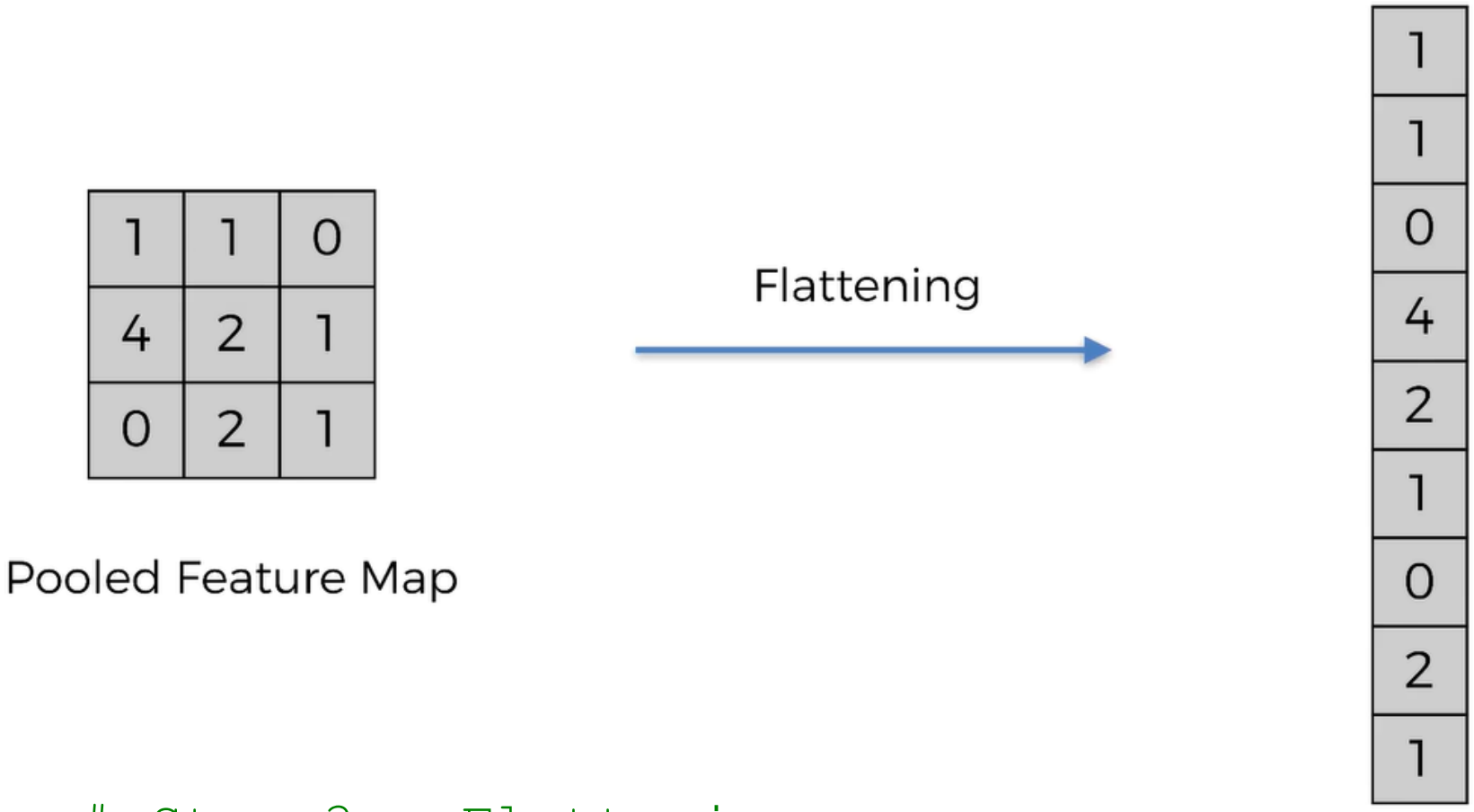
```
# Adding a second convolutional layer

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

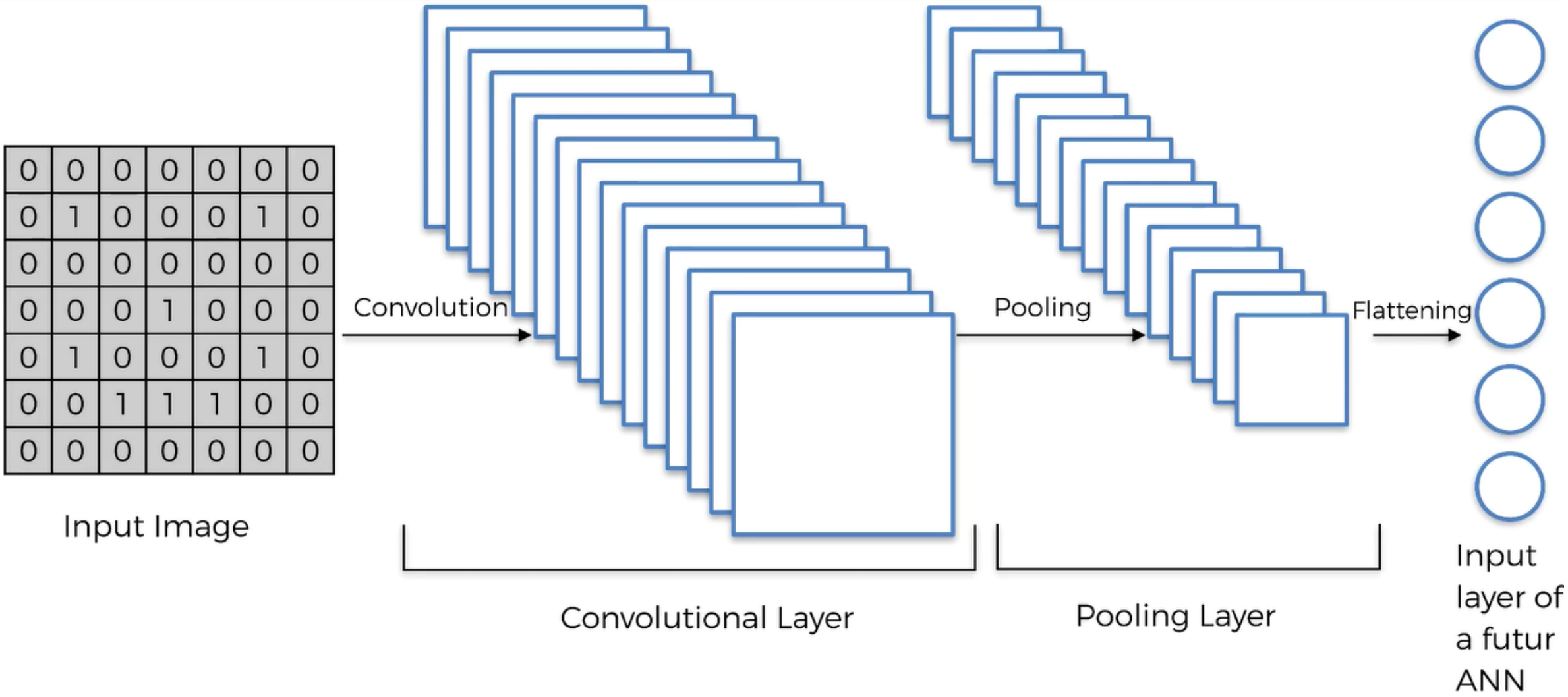
```
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation =
'relu'))
```

Step 3 - Flattening




```
# Step 3 - Flattening
classifier.add(Flatten())
```

Step 3 - Flattening



Convolutional Neural Network

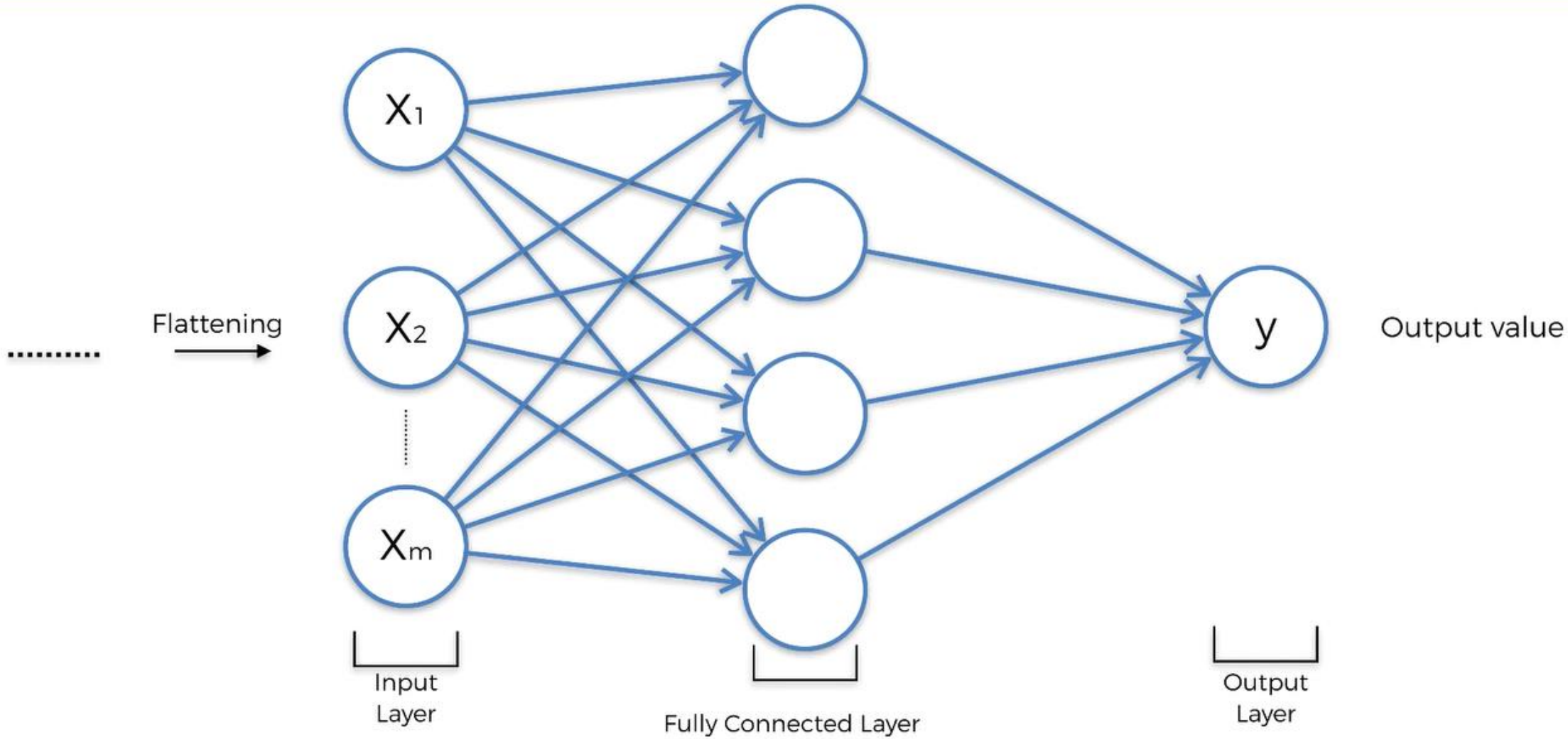
# Initialising the CNN
classifier = Sequential()

Step 1 - Convolution
```
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
```

Step 2 - Pooling
```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

# Step 3 - Flattening
classifier.add(Flatten())

Step 4 - Full Connection



Convolutional Neural Network

Step 4 - Full connection

```
classifier.add(Dense(units = 128, kernel_initializer =
'uniform', activation = 'relu'))
```

```
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

softmax

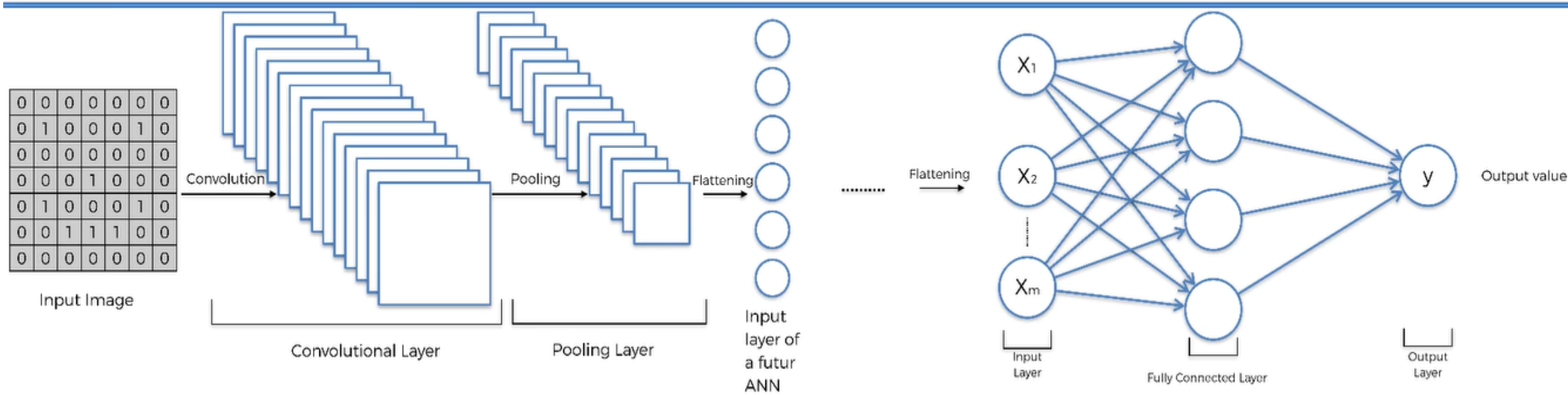
Compiling the CNN

```
classifier.compile(optimizer = 'adam', loss =
'binary_crossentropy', metrics = ['accuracy'])
```

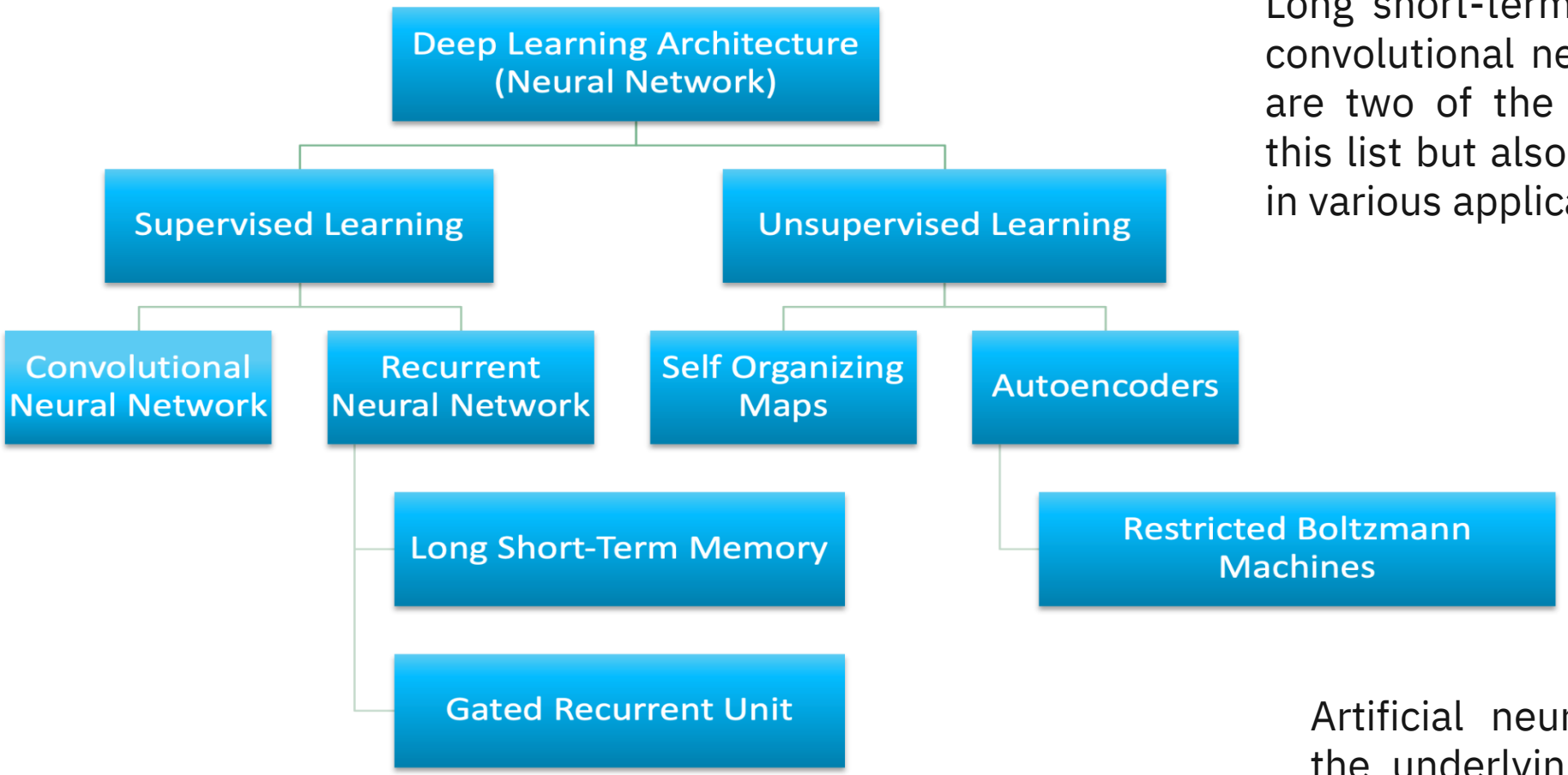
categorical_crossentropy

Step 3 - Flattening

Step 4 - Full Connection



Deep learning architectures



Long short-term memory (LSTM) and convolutional neural networks (CNNs) are two of the oldest approaches in this list but also two of the most used in various applications.

Artificial neural network (ANN) is the underlying architecture behind deep learning

Thanks