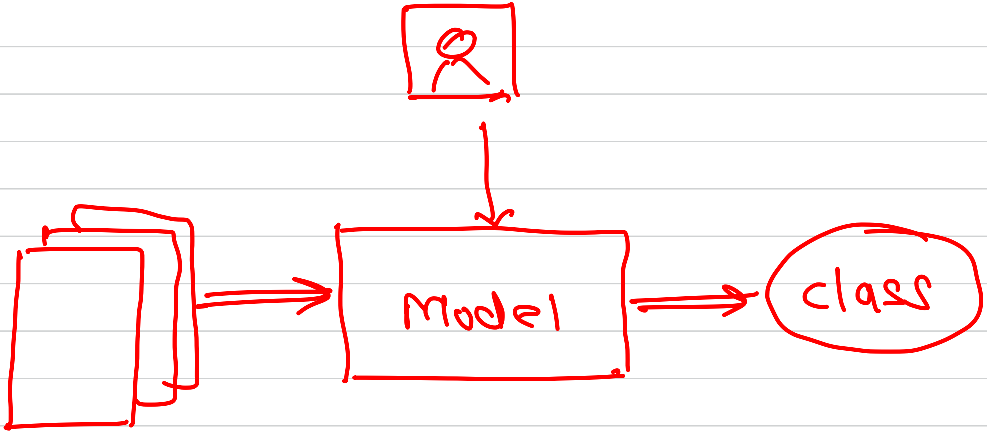
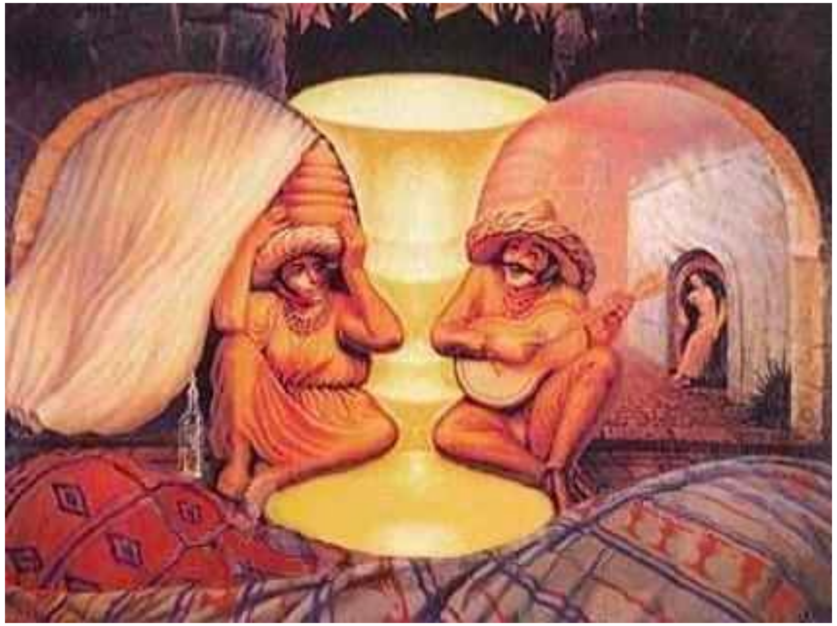
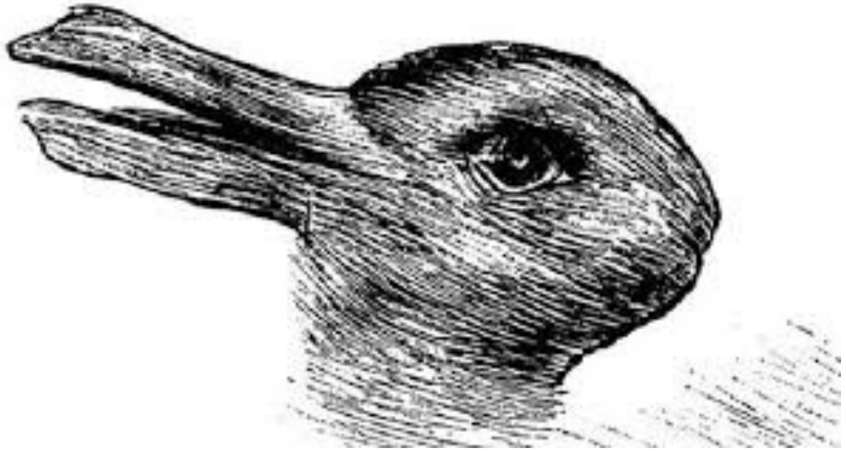
Machine Learning

CNN


Model
class

CNN

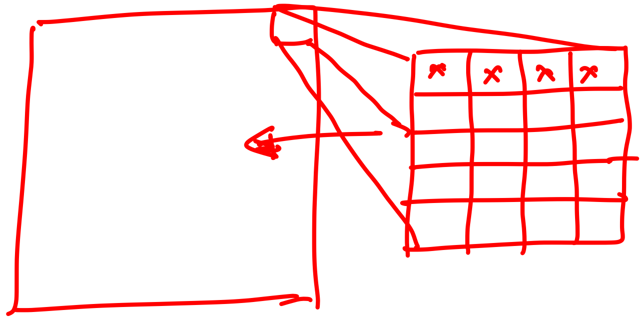




Image

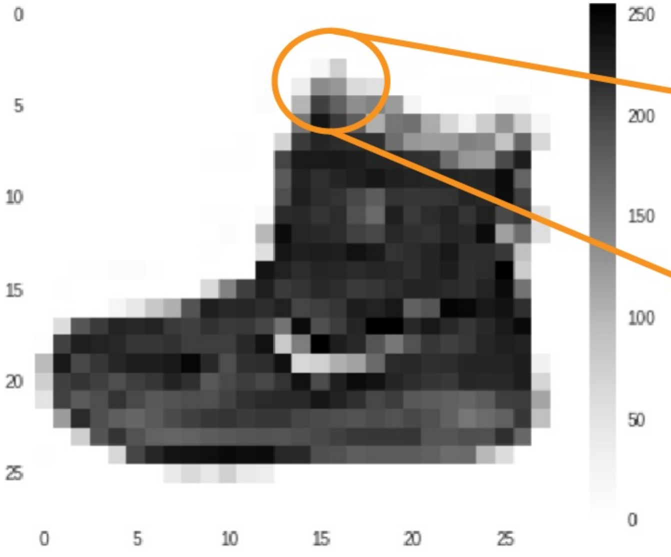
- An image is an artifact that depicts visual perception, such as a photograph or other two-dimensional picture, that resembles a subject—usually a physical object—and thus provides a depiction of it
- It can be seen as a two-dimensional (2D) view of a 3D world
- A digital image is a numeric representation of a 2D image
- It is a finite set of digital values, which are called pixels

pixel ⇒ picture + element



Convolution

- A convolution is simply a filter of weights that are used to multiply a pixel with its neighbours to get a new value for the pixel
- The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image



0	64	128
48	192	144
142	226	168

-1	0	-2
.5	4.5	-1.5
1.5	2	-3

Convolution

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

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

(kernel)

=

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

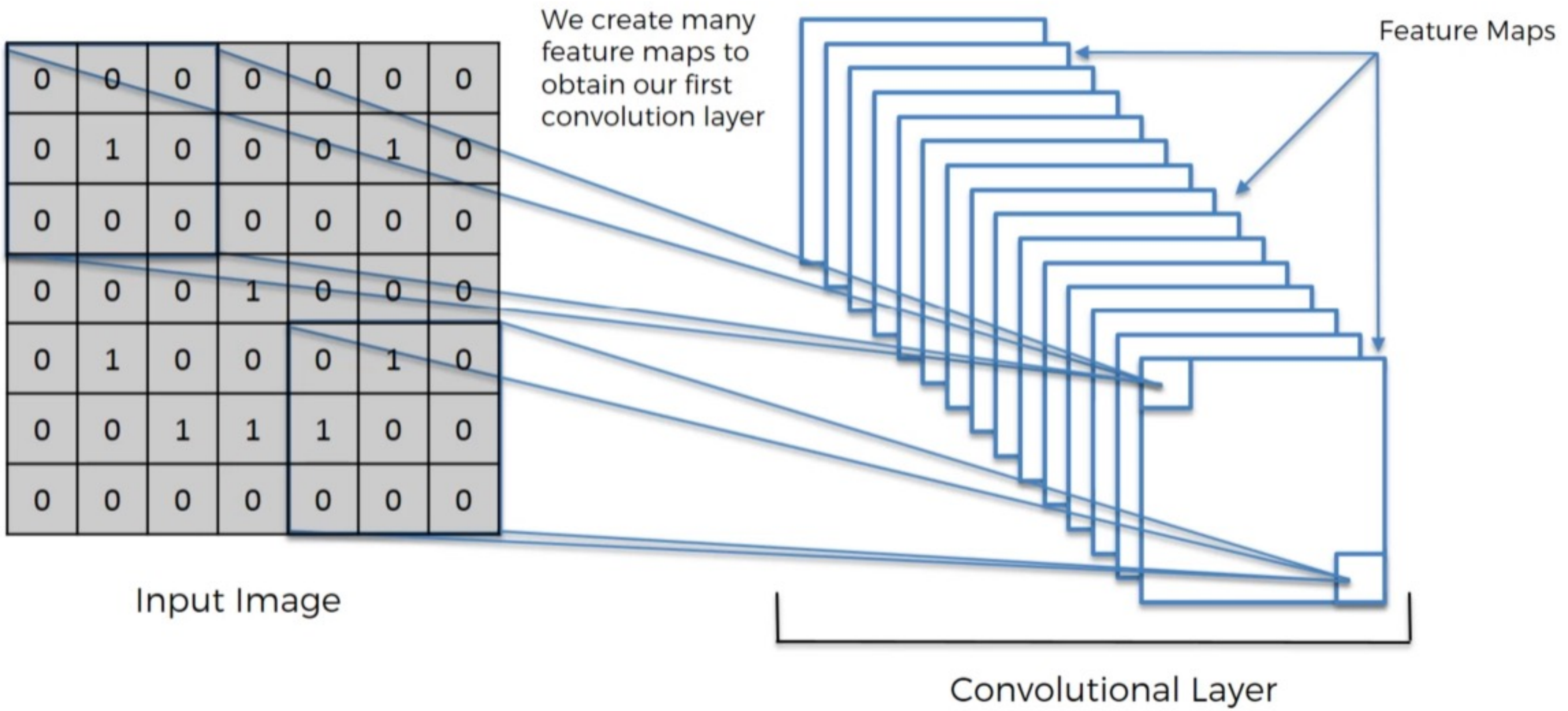
Convolutional Neural Network

- A Convolutional Neural Network (ConvNet / CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics
- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex
- Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field
- A collection of such fields overlap to cover the entire visual area

Convolutional Neural Network

- There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

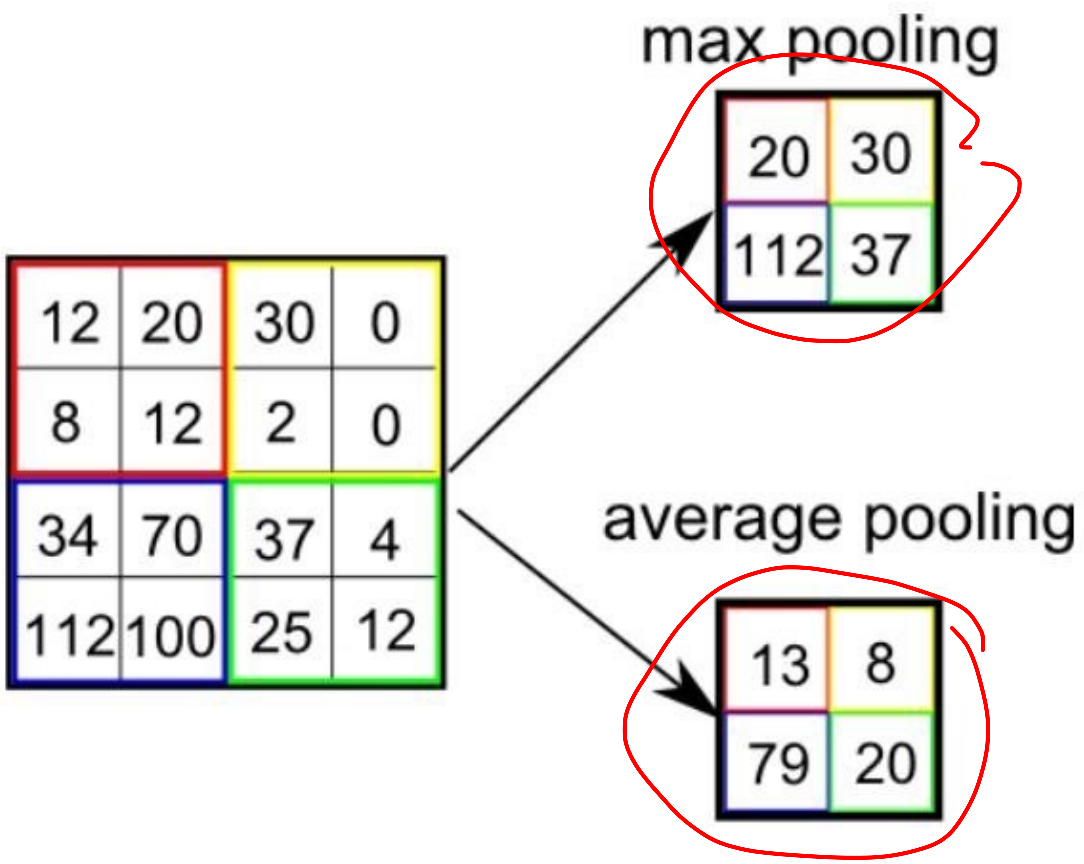
Convolution Layer



Max Pooling

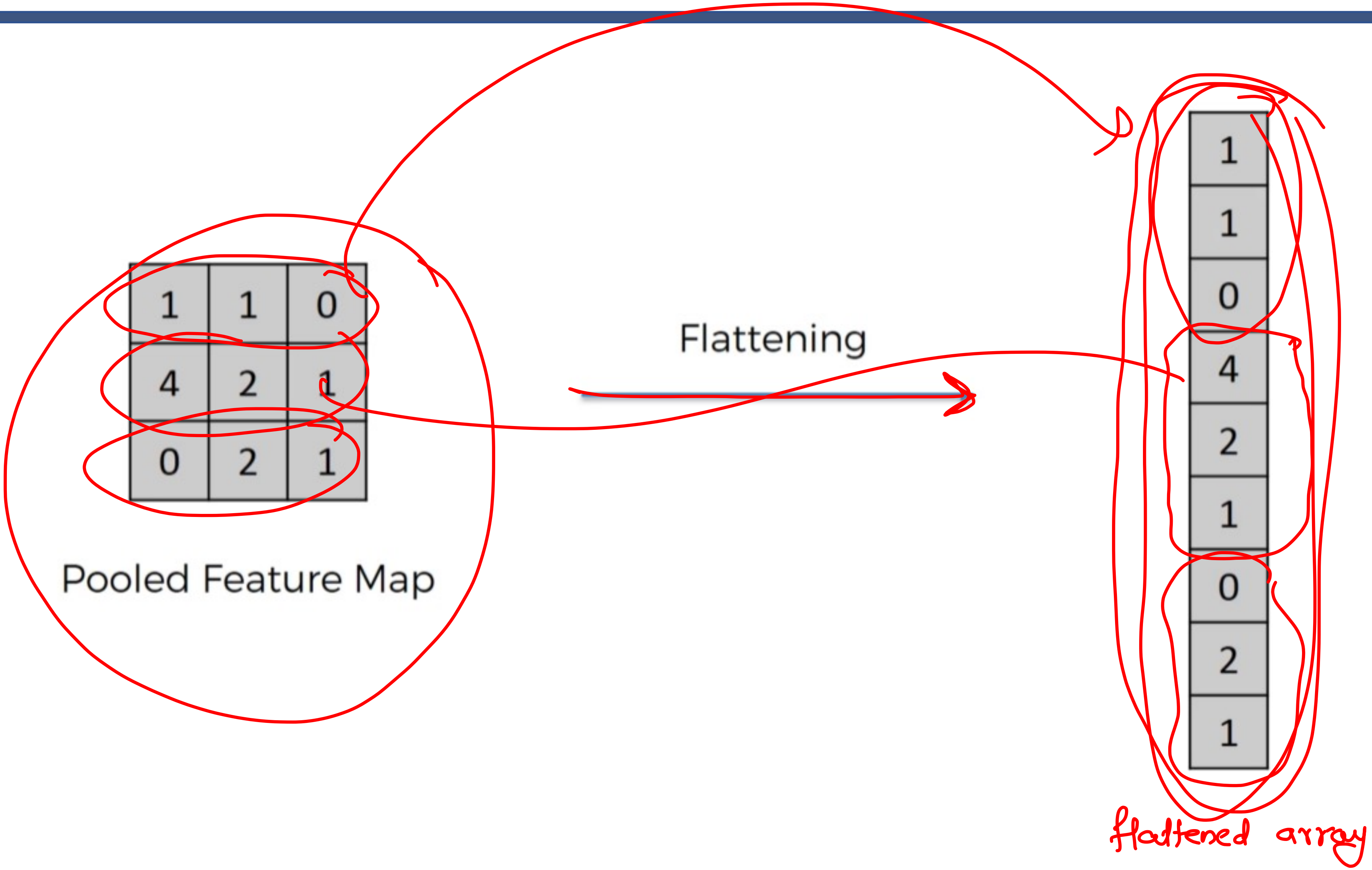
- Pooling layer is responsible for reducing the spatial size of the Convolved Feature
- This is to decrease the computational power required to process the data through dimensionality reduction
- Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model
- There are two types of Pooling
  - Max Pooling
    - Max Pooling returns the maximum value from the portion of the image covered by the Kernel
    - It also performs as a Noise Suppressant
    - It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction
  - Average Pooling
    - Average Pooling returns the average of all the values from the portion of the image covered by the Kernel
    - It performs dimensionality reduction as a noise suppressing mechanism

Max Pooling



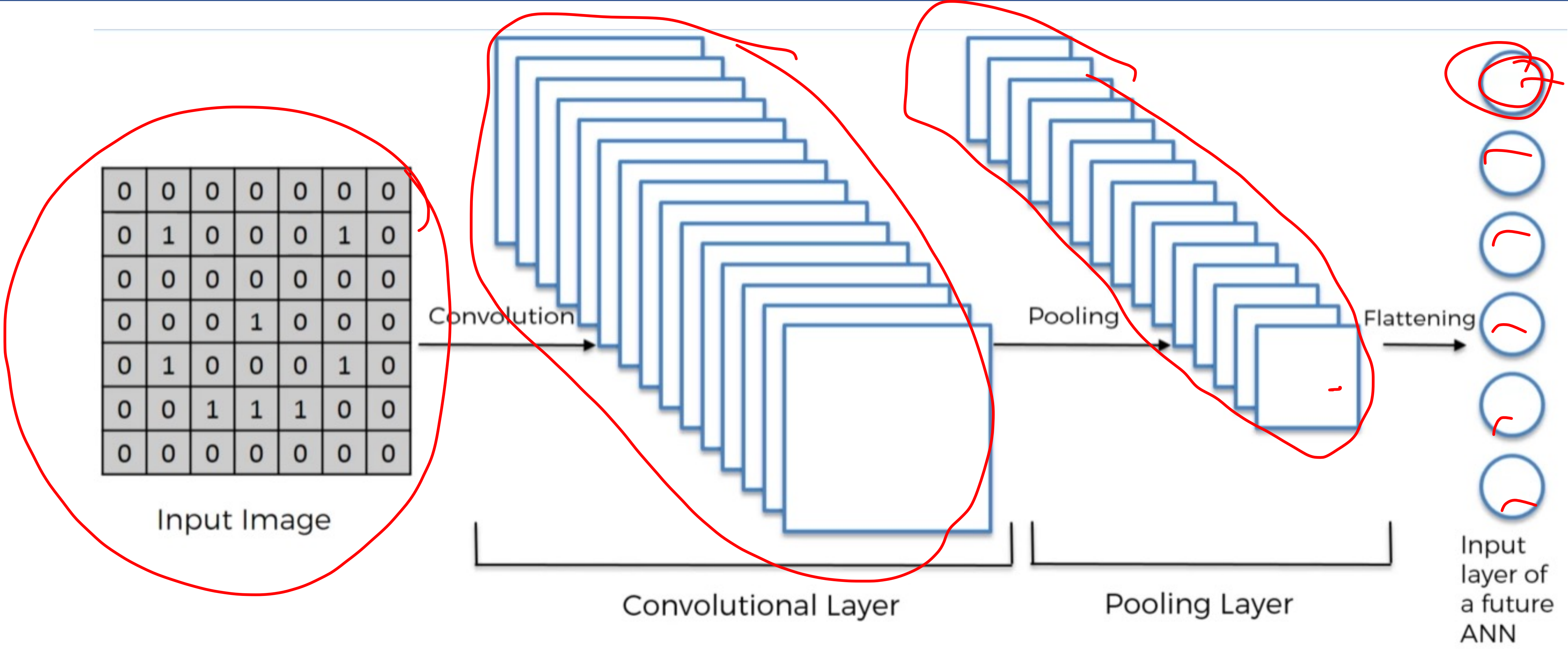
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
12
max pooling
20
30
112
37
average pooling
13
8
79
20

Flattening



1 1 0
4 2 1
0 2 1
Pooled Feature Map
Flattening
1
1
0
4
2
1
0
2
1
flattened array

Flattening



0 0 0 0 0 0 0
0 1 0 0 0 1 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
Convolution
Convolutional Layer
Pooling
Pooling Layer
Flattening
Input layer of a future ANN

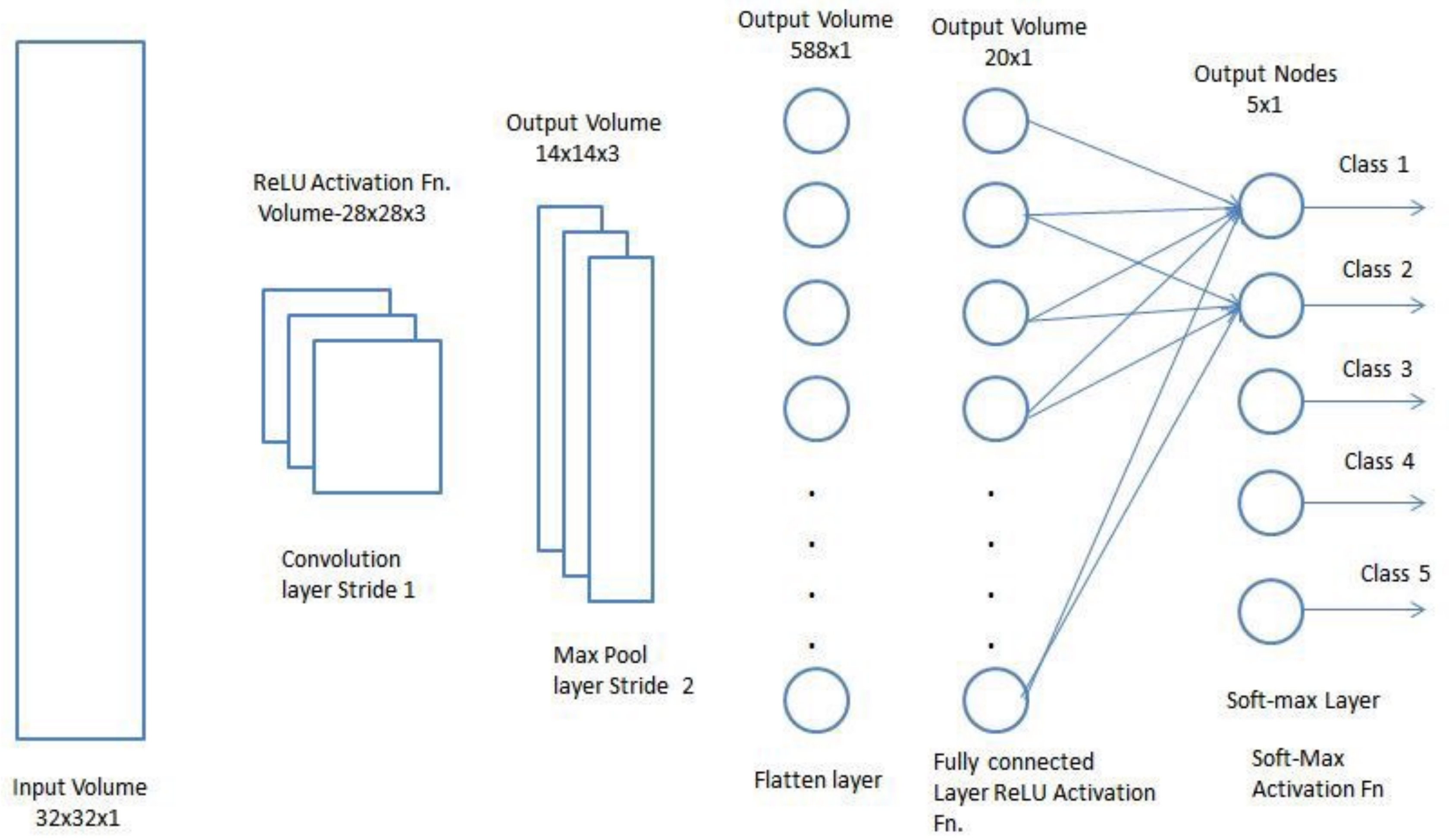
x1	x2	x3	y
chol	thal..	ecg..	target
			0
			1
			0
			1
			1

x1	x2	x3	x4	..	x10	y

human

y → human (1) horse (0)

Full Connection



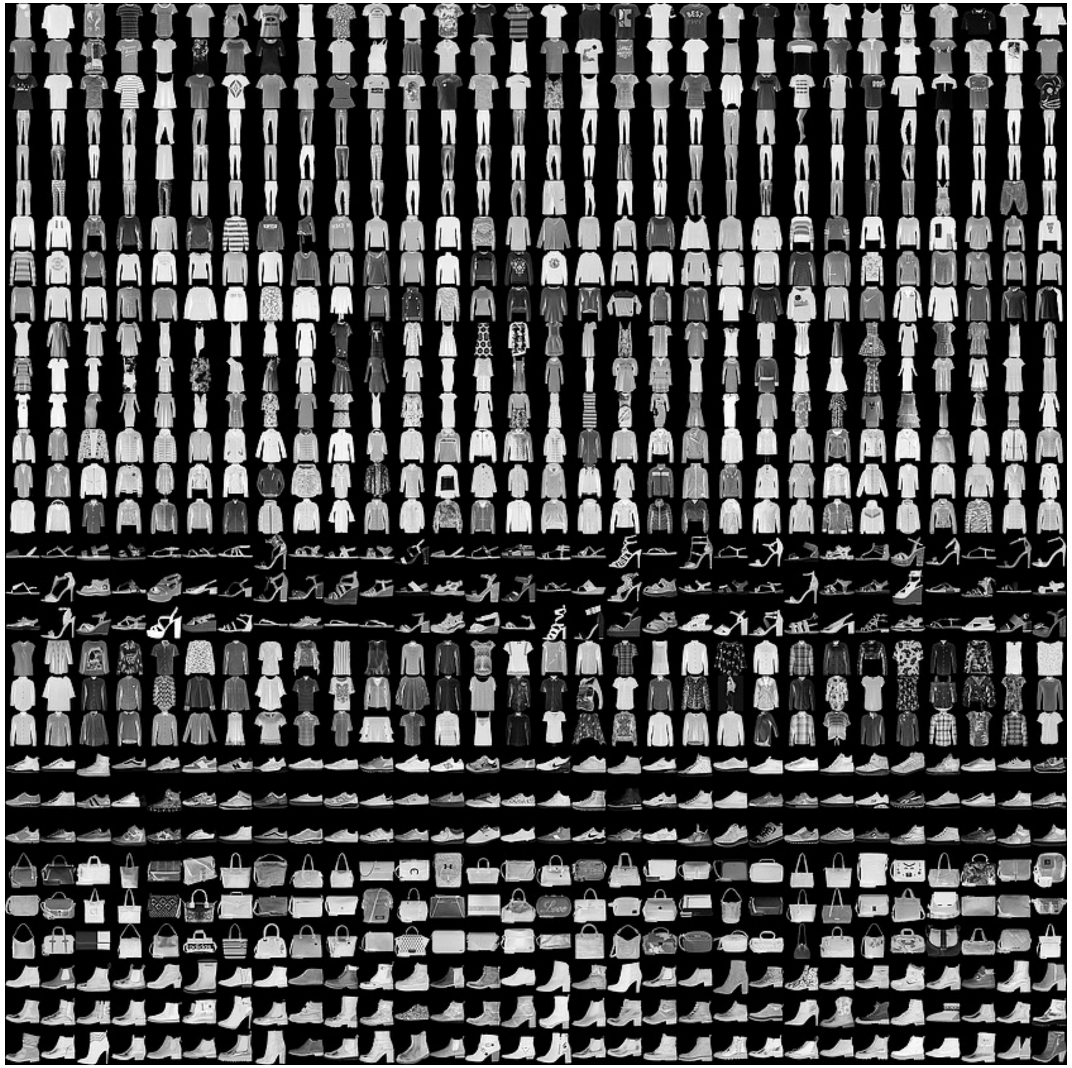
Using TensorFlow

Recognizing Clothing Items

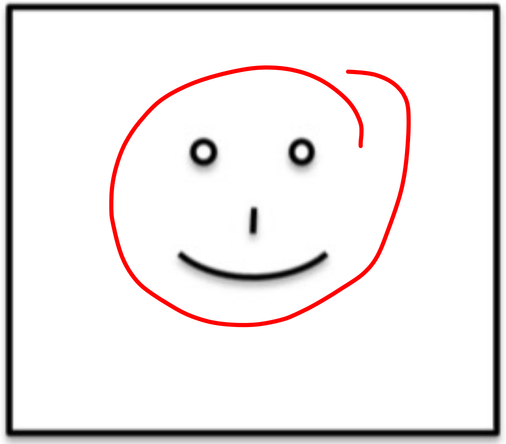
- There are a number of different clothing items here, and you can recognize them
- You understand what is a shirt, or a coat, or a dress
- But how would you explain this to somebody who has never seen clothing? How about a shoe? There are two shoes in this image, but how would you describe that to somebody?
- This is another area where the rules-based programming can fall down
- Sometimes it's just infeasible to describe something with rules

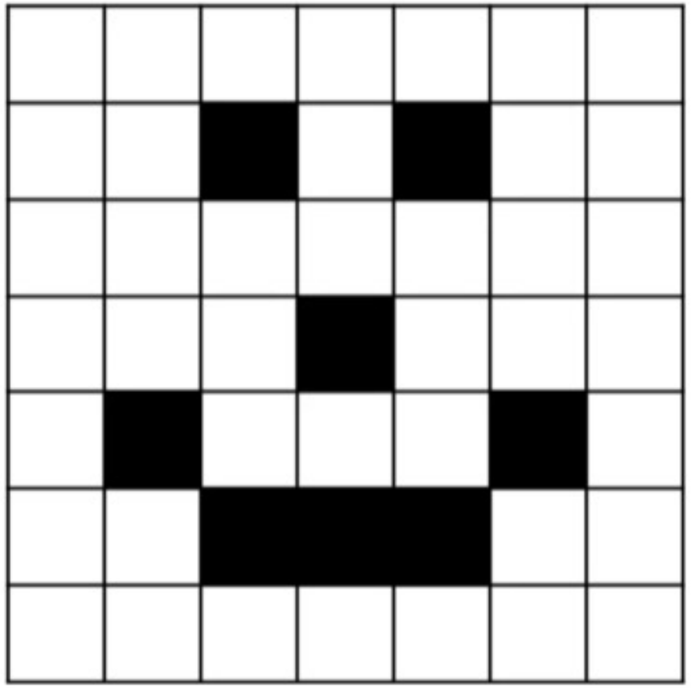
The Data: Fashion MNIST

- One of the foundational datasets for learning and benchmarking algorithms is the Modified National Institute of Standards and Technology (MNIST) database, by Yann LeCun, Corinna Cortes, and Christopher Burges
- This dataset is comprised of images of 70,000 handwritten digits from 0 to 9
- The images are 28 × 28 grayscale.
- Fashion MNIST is designed to be a drop-in replacement for MNIST that has the same number of records, the same image dimensions, and the same number of classes—so, instead of images of the digits 0 through 9, Fashion MNIST contains images of 10 different types of clothing



CNN





0	0	0	0	0	0	0
0	[illegible]	0	0	0	[illegible]	0
0	0	0	0	0	0	0
0	0	0	[illegible]	0	0	0
0	[illegible]	0	0	0	[illegible]	0
0	0	[illegible]	[illegible]	[illegible]	0	0
0	0	0	0	0	0	0

Designing the Neural Network

image size

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

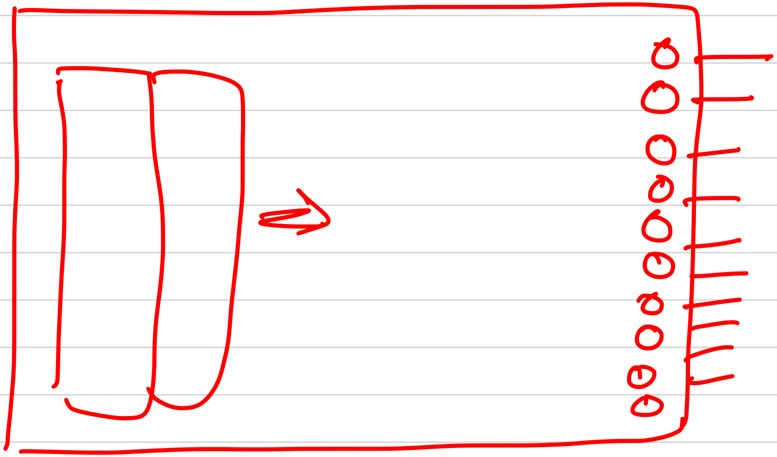
output

Designing the Neural Network

- The first, Flatten, isn't a layer of neurons, but an input layer specification
- Our inputs are 28 × 28 images, but we want them to be treated as a series of numeric values
- Flatten takes the image (a 2D array) and turns it into a line (a 1D array)

- Finally, there's another Dense layer, which is the output layer which has 10 neurons, because we have 10 classes
- Each of these neurons will end up with a probability that the input pixels match that class, so our job is to determine which one has the highest value

- The next one, Dense (hidden layers), is a layer of neurons, and we're specifying that we want 128 of them
- More neurons means it will run more slowly, as it has to learn more parameters
- It takes some experimentation over time to pick the right values. This process is typically called hyperparameter tuning



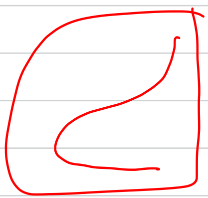
$y_1\ y_2\ y_3\ y.\ y_{.10}$

0.1
0.05
0.05
$\boxed{0.30}$
0.05
⋮

0
1

9

0 → 0
→ 0.1
⋮
9 — 0.8



binary

(0, 1)

↓

multi-nomial

(0, 1, 2, 3, 4)

↓

multi-class

(y, ... y_{10})

[. . .]