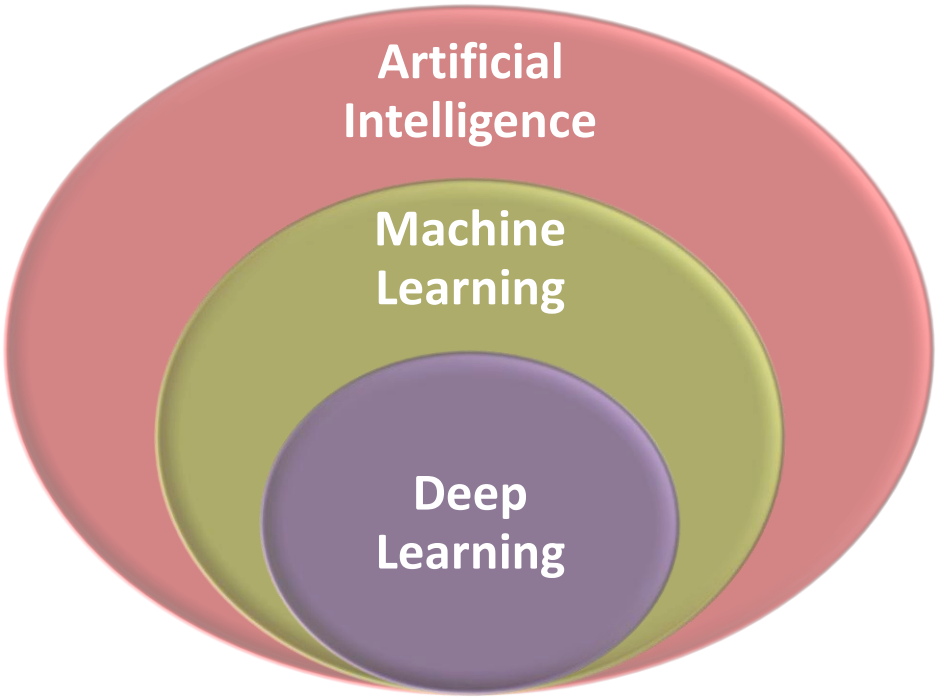
Deep Learning

Deep Learning

- Deep learning is an immensely rich subfield of machine learning, with powerful applications ranging from machine perception to natural language processing, all the way up to creative AI.



Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence

- A concise definition of **AI** would be:

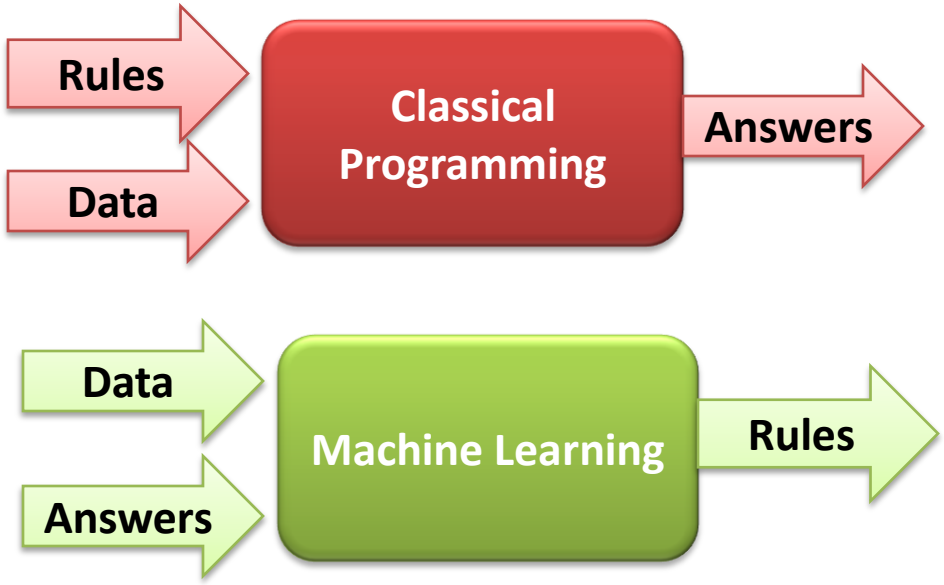
“ the effort to automate intellectual tasks normally performed by humans. “

- AI is a very general field which encompasses machine learning and deep learning, but also includes many more approaches that do not involve any learning.

Machine Learning

- With Machine Learning, humans would input data as well as the answers expected from the data, and out would come the rules.
- These rules could then be applied to new data to produce original answers.

A machine learning system is "trained" rather than explicitly programmed. It is presented with many "examples" relevant to a task, and it finds statistical structure in these examples which eventually allows the system to come up with rules for automating the task.



To do machine learning, we need three things:

- Input data points.
- Examples of the expected output.
- A way to measure if the algorithm is doing a good job, to measure the distance between its current output and its expected output.

  This is used as a feedback signal to adjust the way the algorithm works.

  This adjustment step is what we call "learning".

- **Machine learning is, technically:**

  Searching for useful representations of some input data, within a pre-defined space of possibilities, using guidance from some feedback signal. This simple idea allows for solving a remarkably broad range of intellectual tasks, from speech recognition to autonomous car driving.

Deep Learning

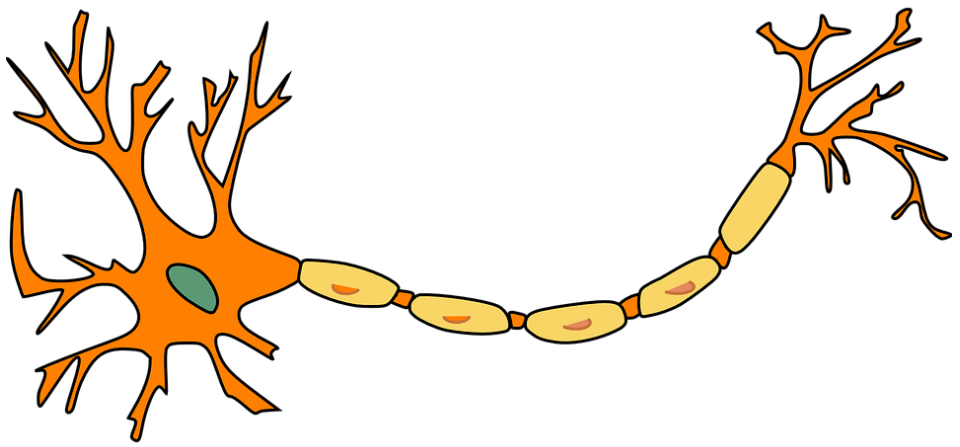
- Deep learning is a specific subfield of machine learning, a new take on learning representations from data which puts an emphasis on learning successive "layers" of increasingly meaningful representations.
- The "deep" in "deep learning" simply stands for this idea of successive layers of representations—how many layers contribute to a model of the data is called the "depth" of the model.

Deep Learning

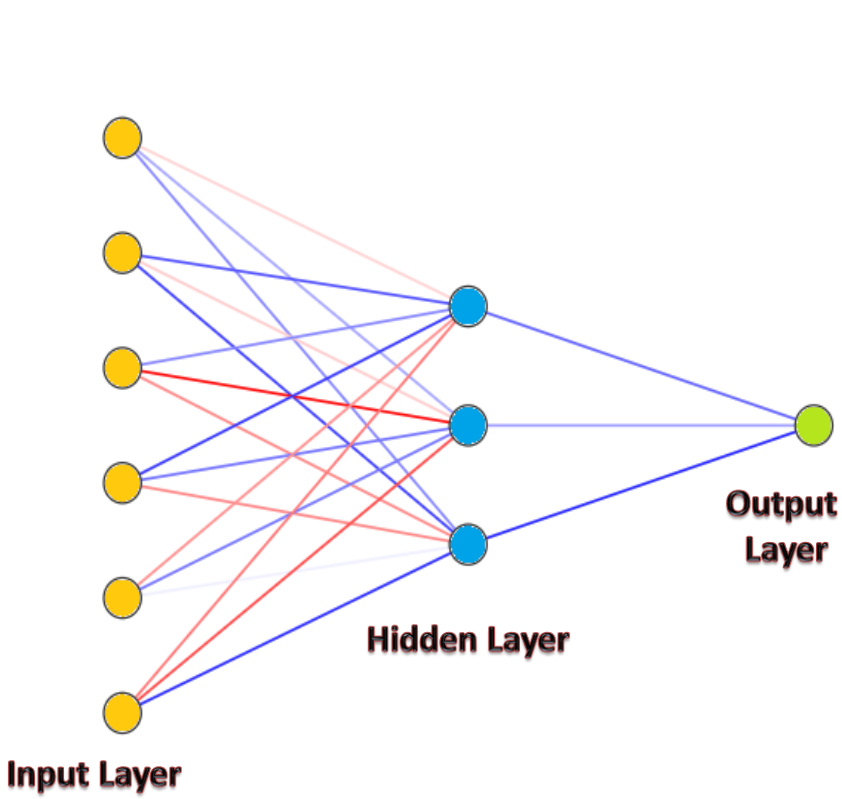
- Modern deep learning often involves tens or even hundreds of successive layers of representation and they are all learned automatically from exposure to training data.
- Other approaches to machine learning tend to focus on learning only one or two layers of representation of the data, sometimes called **"shallow learning".**

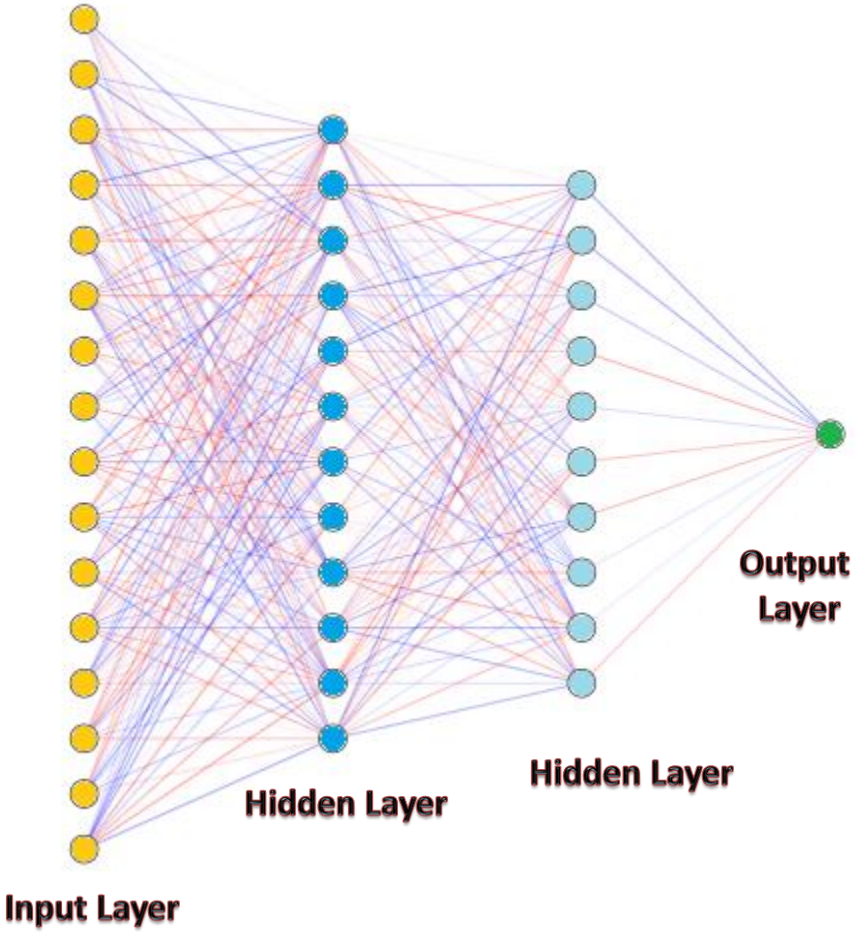
Deep Learning

- In deep learning, these layered representations are learned via models called "neural networks", structured in literal layers stacked one after the other.
- The term "neural network" is a reference to neurobiology.



Deep Neural Networks

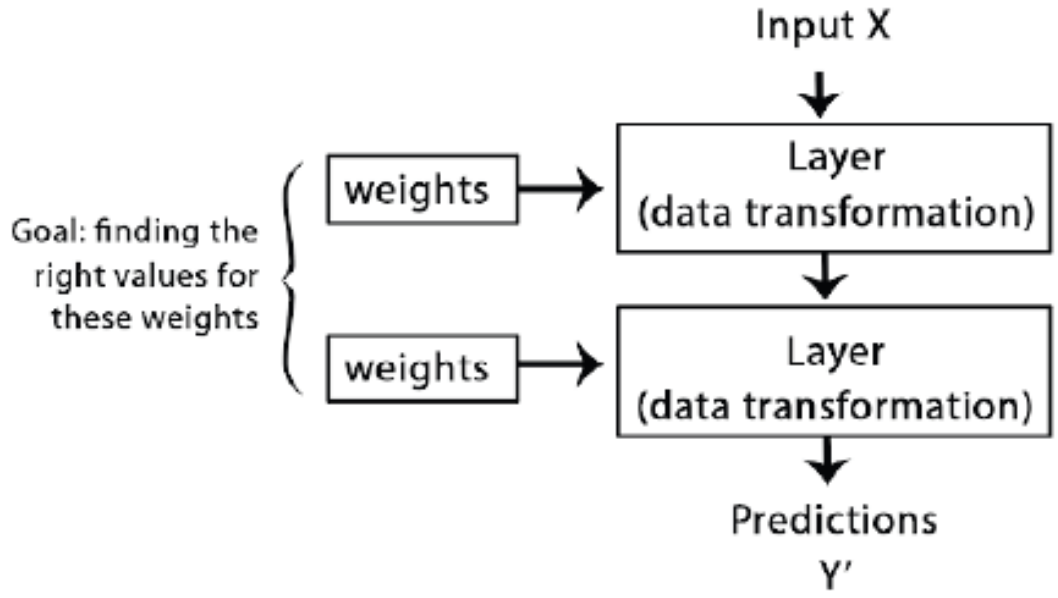




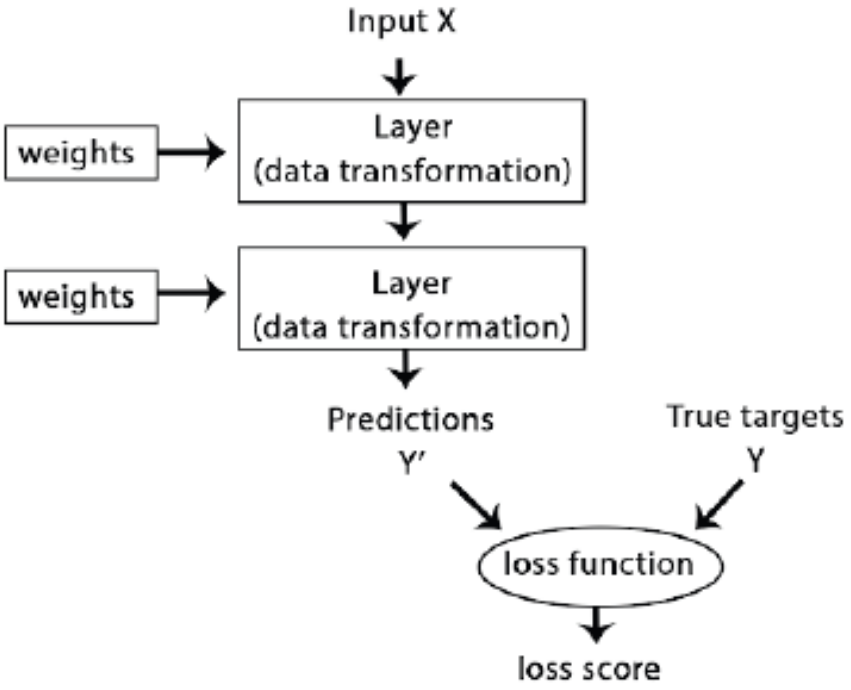
Understanding how deep learning works

- Deep neural networks do input-to-target mapping via a deep sequence of simple data transformations called "layers", and that these data transformations are learned by exposure to examples.

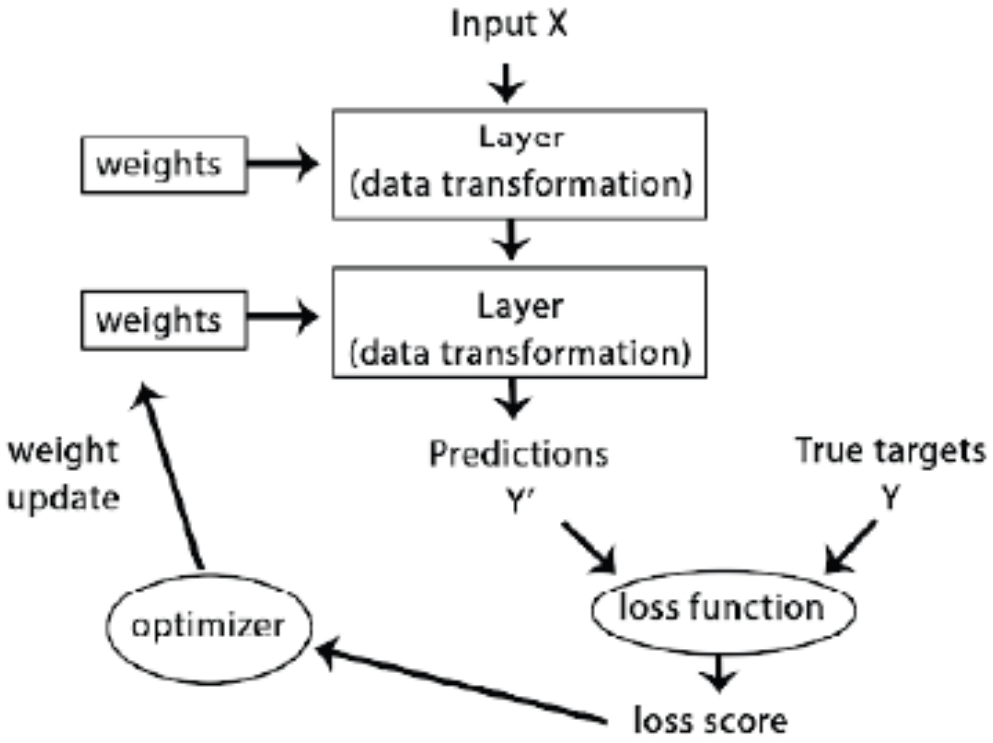
- The specification of what a layer does to its input data is stored in the layer's **"weights"**, which in essence are a bunch of numbers.
- Weights are also sometimes called the **"parameters"** of a layer.
- **"learning"** will mean finding a set of values for the weights of all layers in a network, such that the network will correctly map your example inputs to their associated targets.



- To control the output of a neural network, you need to be able to measure how far this output is from what you expected.
- This is the job of the **"loss function"** of the network, also called **"objective function".**
- The loss function takes the predictions of the network and the true target , and computes a distance score, capturing how well the network has done on this specific example.



- The fundamental trick in deep learning is to use the loss score as a feedback signal to adjust the value of the weights by a little bit, in a direction that would lower the loss score for the current example.
- This adjustment is the job of the "optimizer", which implements what is called the **"backpropagation"** algorithm, the central algorithm in deep learning.



What deep learning has achieved so far

- Deep learning has achieved the following breakthroughs, all in historically difficult areas of machine learning:

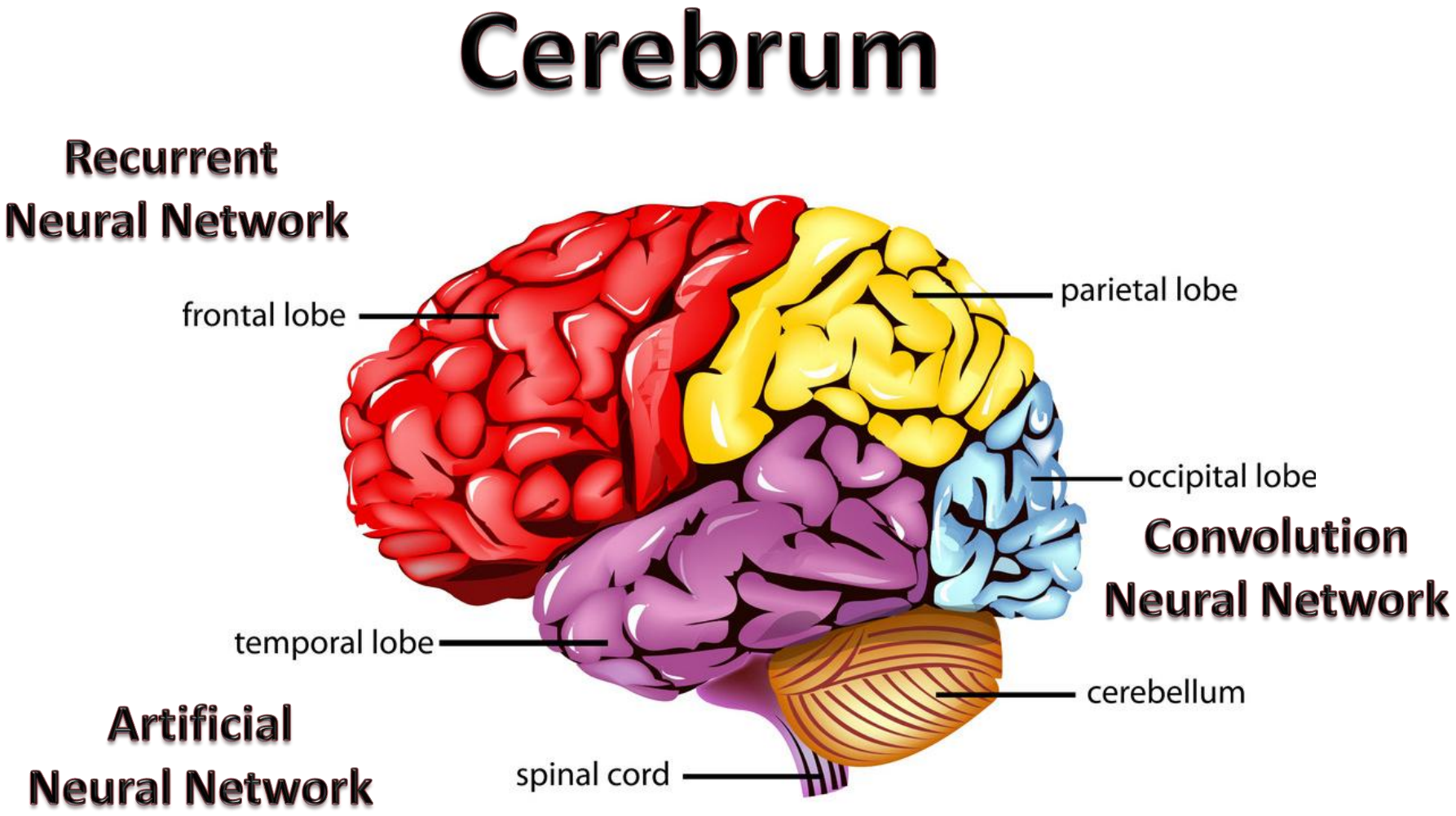
✓ *Near-human level image classification.*
✓ *Near-human level speech recognition.*
✓ *Near-human level handwriting transcription.*
✓ *Improved text-to-speech conversion.*
✓ *Digital assistants such as Amazon Alexa.*
✓ *Improved ad targeting, as used by Google, Bing.*
✓ *Improved search results on the web.*

What makes deep learning different

- Deep learning took off so quickly is primarily that it offered better performance on many problems.
- Deep learning is also making problem-solving much easier, because it completely automates what used to be the most crucial step in a machine learning workflow:

"feature engineering".

Human Brain


Cerebrum
Recurrent
Neural Network
frontal lobe
parietal lobe
occipital lobe
Convolution
Neural Network
temporal lobe
cerebellum
Artificial
Neural Network
spinal cord

Input Layer

- This layer accepts input features.
- It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.

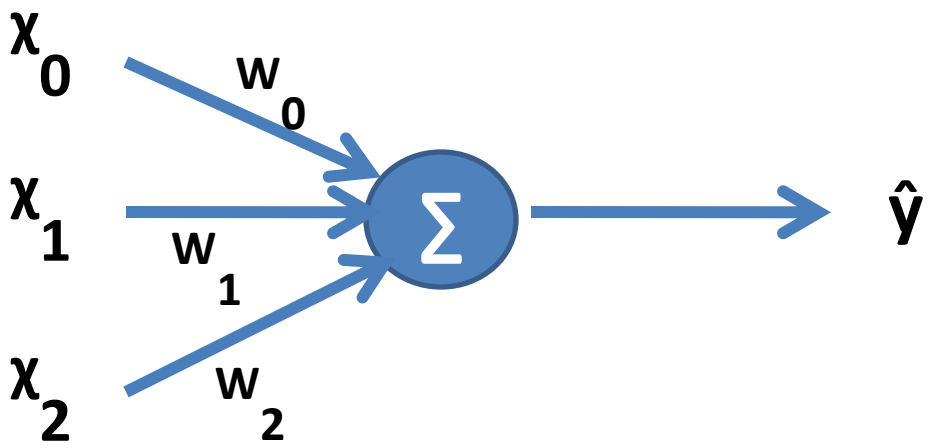
Hidden Layer

- Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by any neural network.
- Hidden layer performs all sort of computation on the features entered through the input layer and transfer the result to the output layer.

Output Layer

- This layer returns the final output computed by the network to the application.

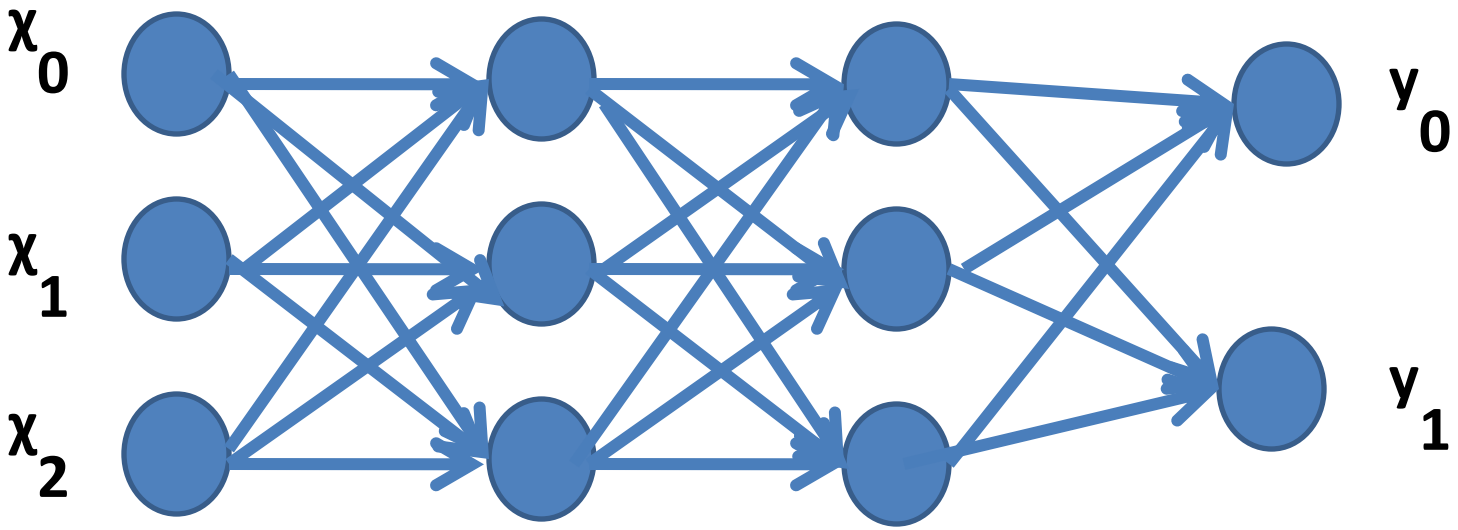
Neural Network with Single Neuron



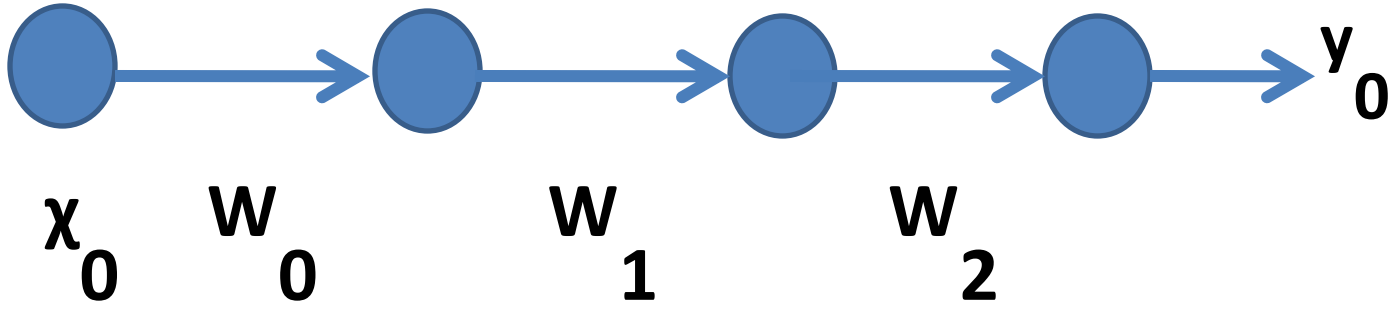
$$\hat{y} = x_0 w_0 + x_1 w_1 + x_2 w_2 = \sum_i x_i w_i$$

$$\begin{pmatrix} w_0 & w_1 & w_2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

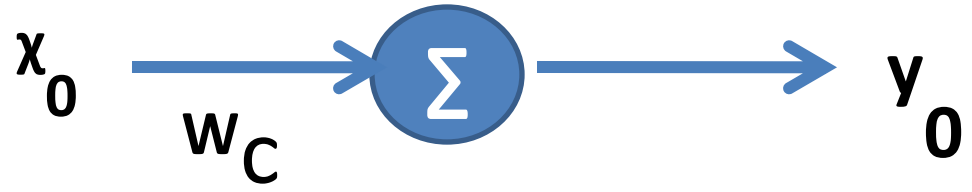
Muti-layer Perceptron



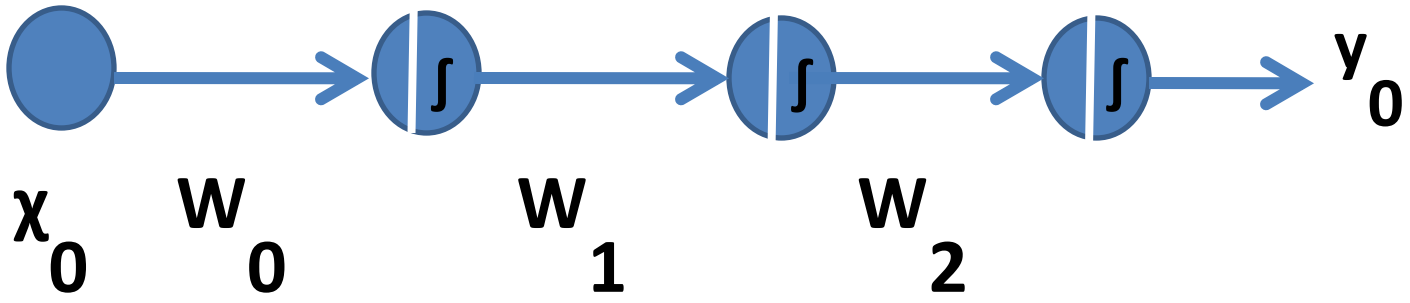
Linear Function



$$y_0 = x_0 w_0 w_1 w_2 = x_0 w_C$$

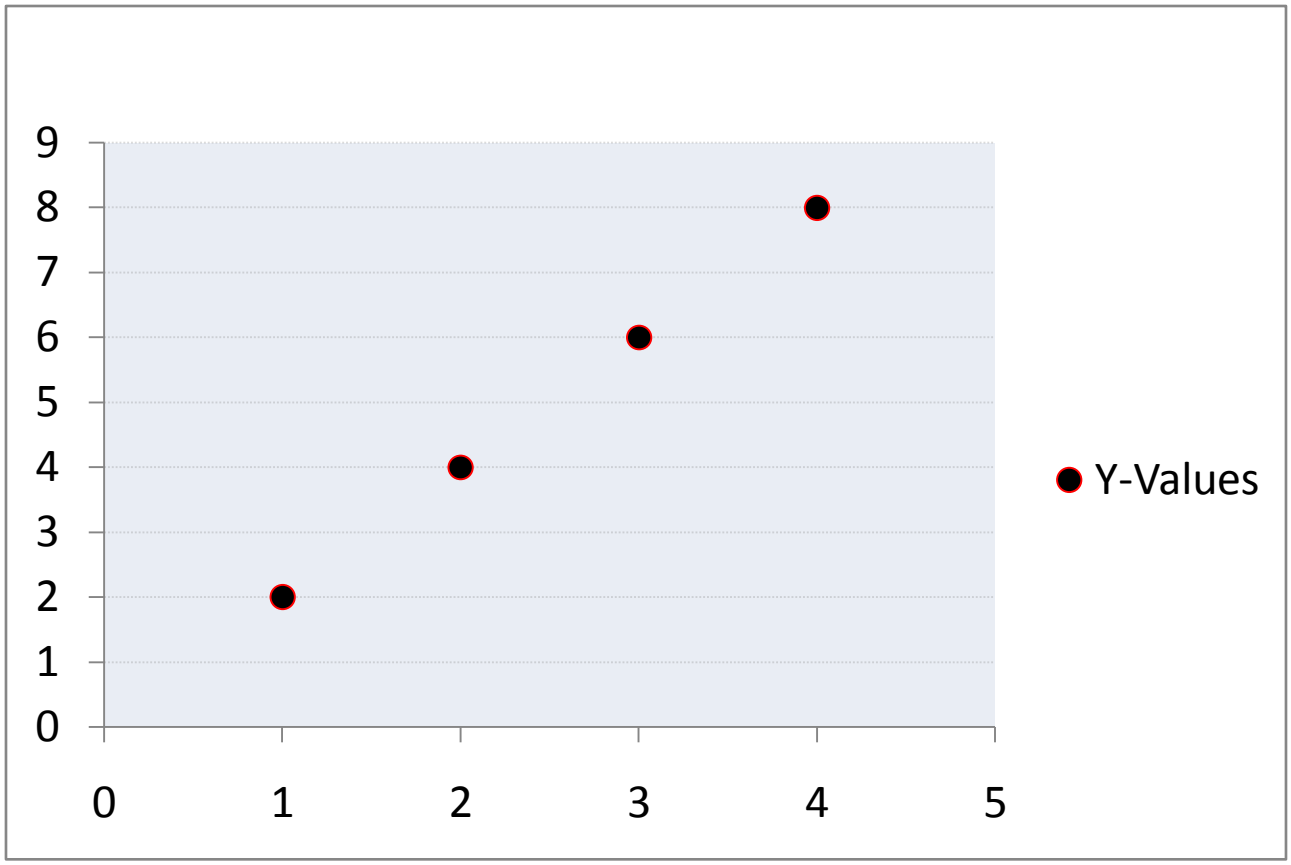


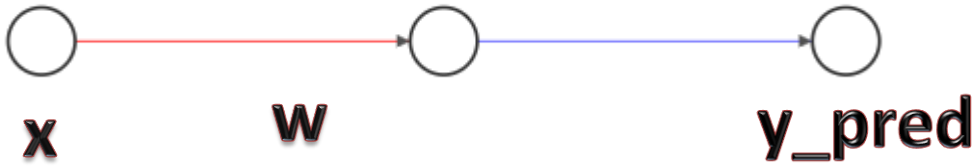
Non-Linear Function



$$y_0 = \sigma(\sigma(\sigma(x_0 W_0) W_1) W_2)$$

Simple Example





y_pred = xw

E = $(\hat{y}-y)^2 = (xw-y)^2$ *sum of squared errors*

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}} = 2x(xw-y)$$ Derivative

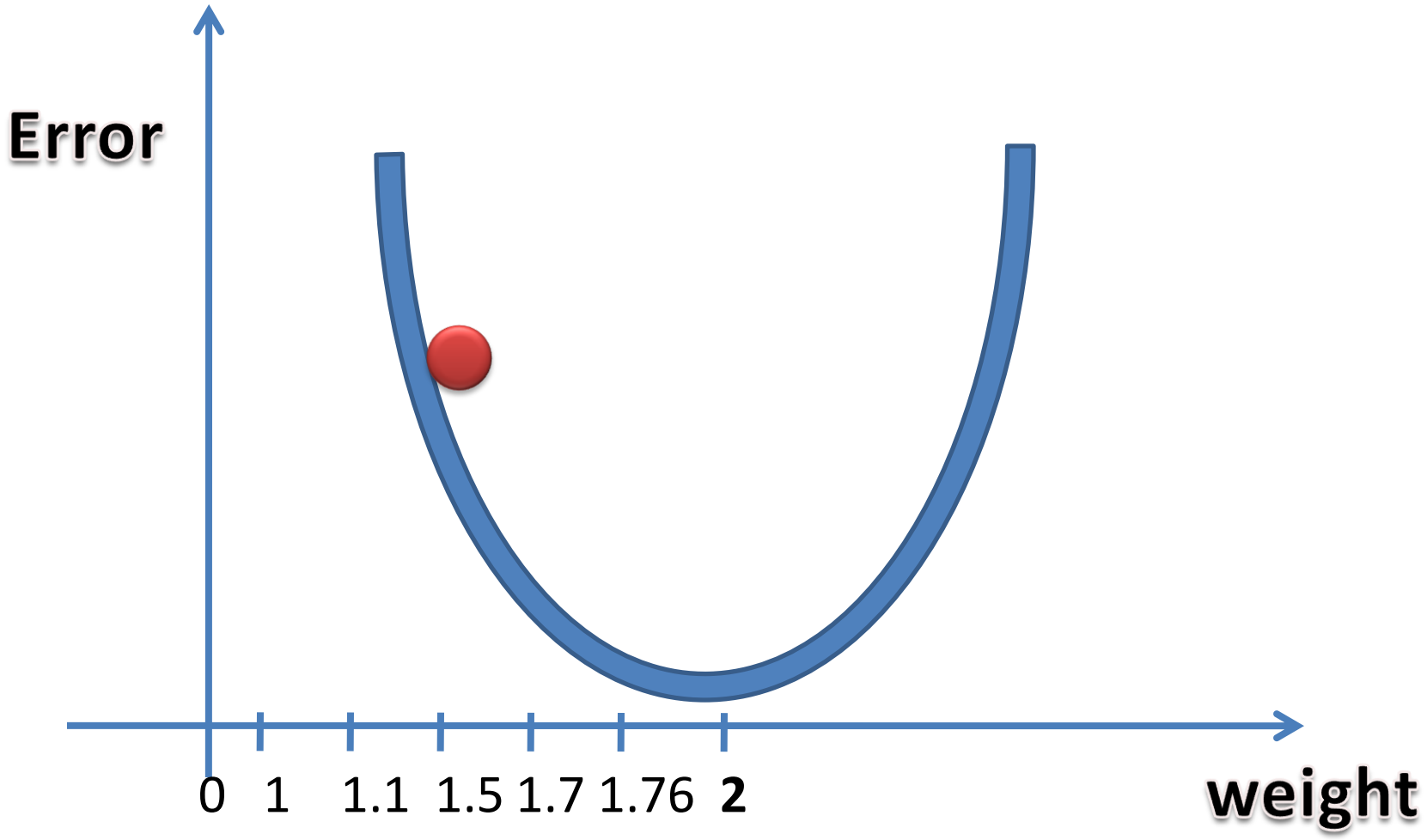
$$\mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}} = \mathbf{w} - \alpha\, \mathbf{2x(xw-y)}$$ Update Rule

- Random weight initialization w=0.5
- learning rating $\alpha = 0.1$

- $w_{new} = w - 0.1 * 2x(xw-y)$
- (x,y)
- (2,4) w=0.5 ← 0.5 - 0.1*2*2(2*0.5-4) = 1.7
- (1,2) w =1.7 ← 1.7-0.1*2*1(1*1.7-2) = 1.76
- (3,6) w= 1.76 ← 1.76-0.1*2*3(3*1.76-6)=2.192
- w ~ 2

Oversimplified Gradient Descent:

- Calculate slope at current position
- If slope is negative, move right
- If slope is positive, move left
- (Repeat until slope == 0)



Convolution Neural Network

- A convolutional neural network

  (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

Convolution Neural Network

- Facebook uses neural nets for their automatic tagging algorithms
- Google for their photo search
- Amazon for their product recommendations
- Pinterest for their home feed personalization
- Instagram for their search infrastructure.

Image classification

- Most popular, use case of these networks is for image processing.

How Humans identify images??

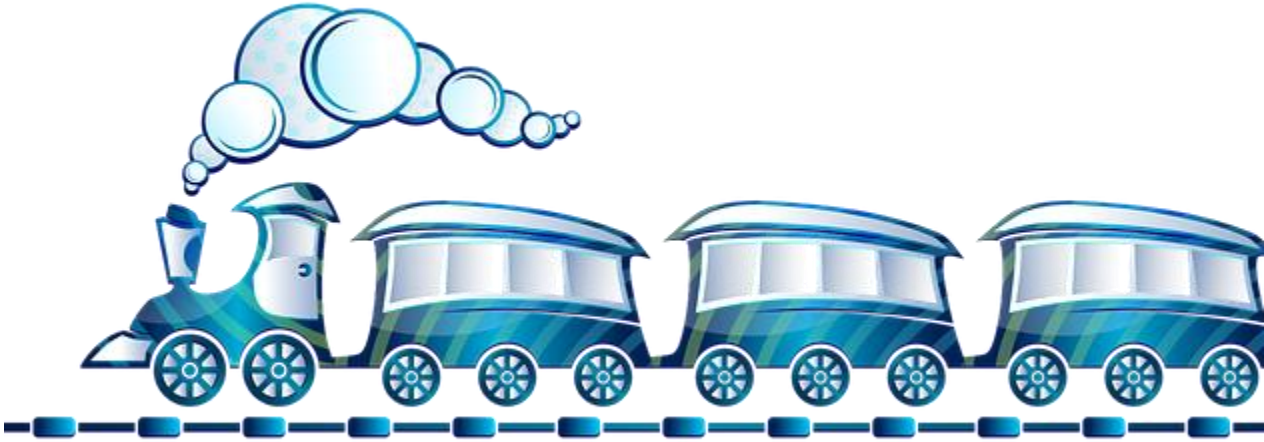
- For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults.
- Most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing.

What's that?

A Train

What's this?

A Train

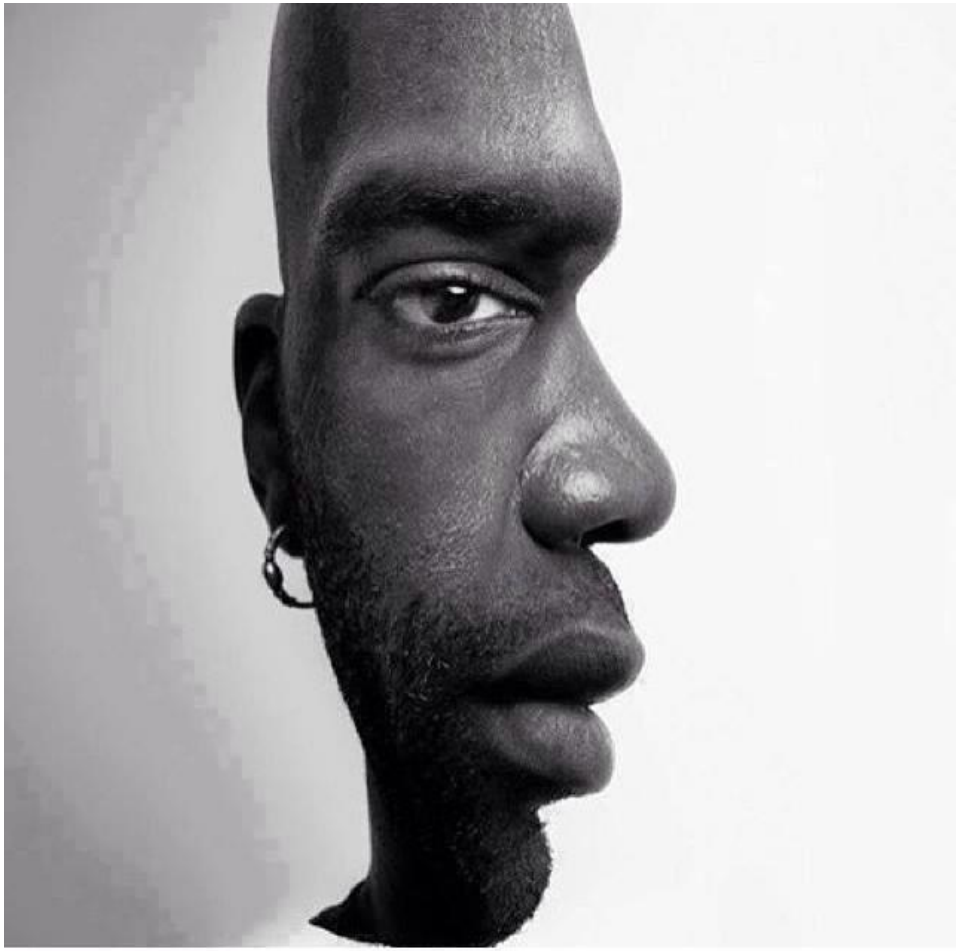


Teach Machine

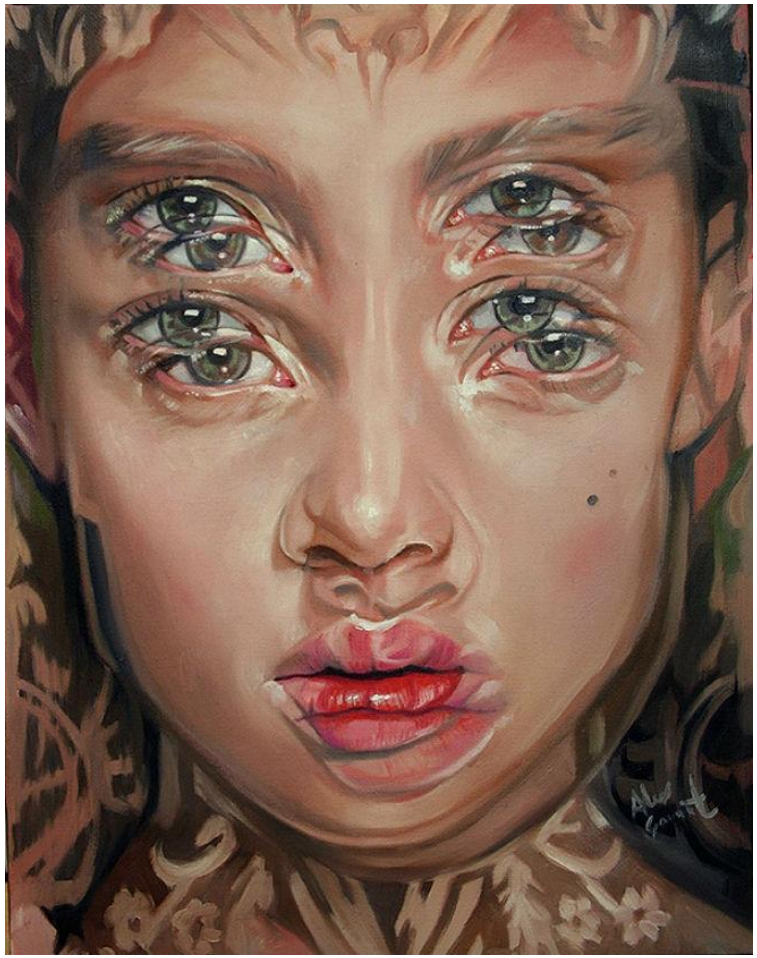
Human skills of being able to :

- Quickly recognize patterns
- Generalize from prior knowledge

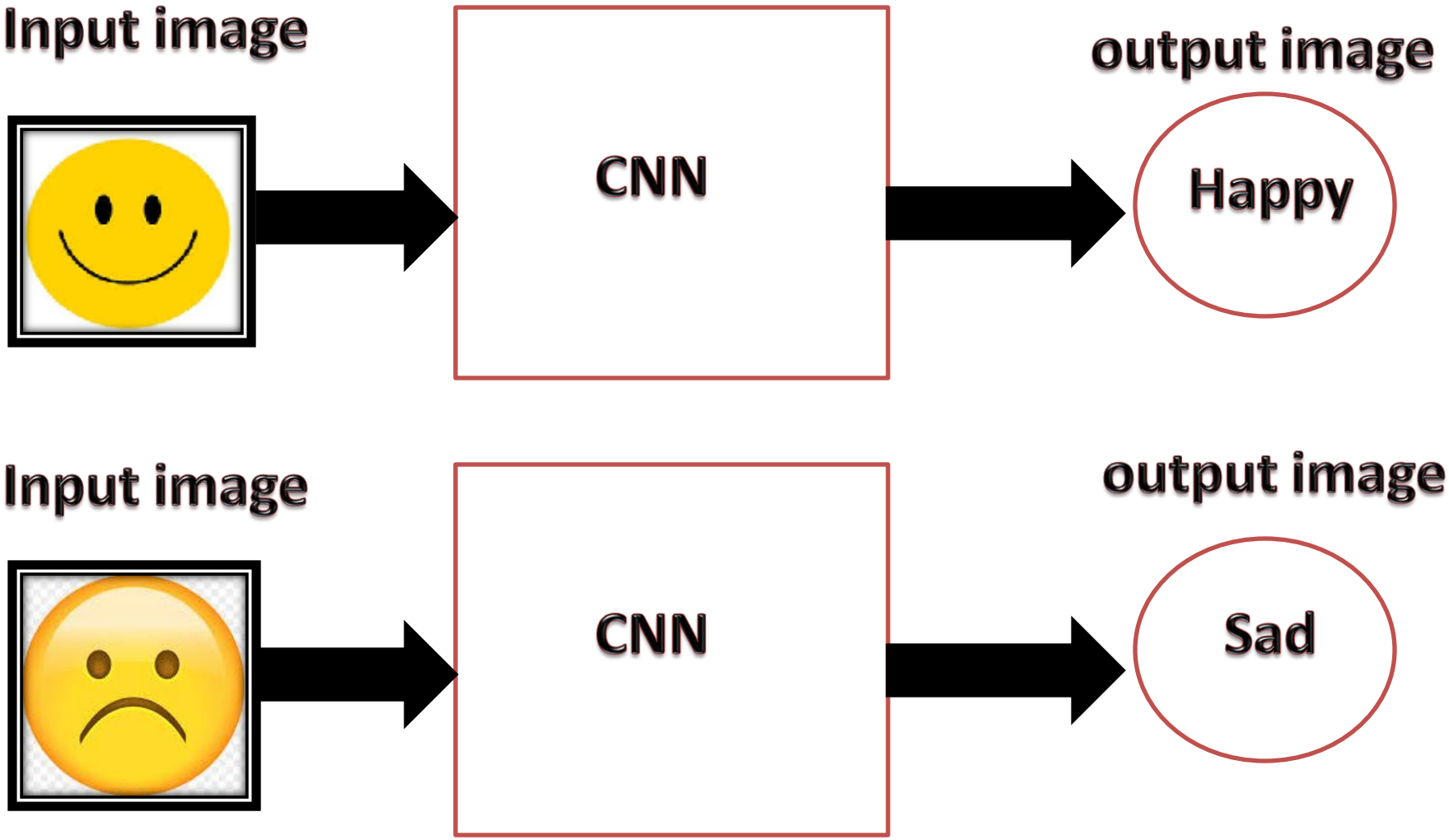
illusions



illusions



Convolution Neural Network



How Does Computer See an image??

When a computer sees an image ,it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values).

What we see

What computers see

How Does Computer See an image??

What we see

What computers see

These numbers, when we perform image classification, are the only inputs available to the computer.

The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class

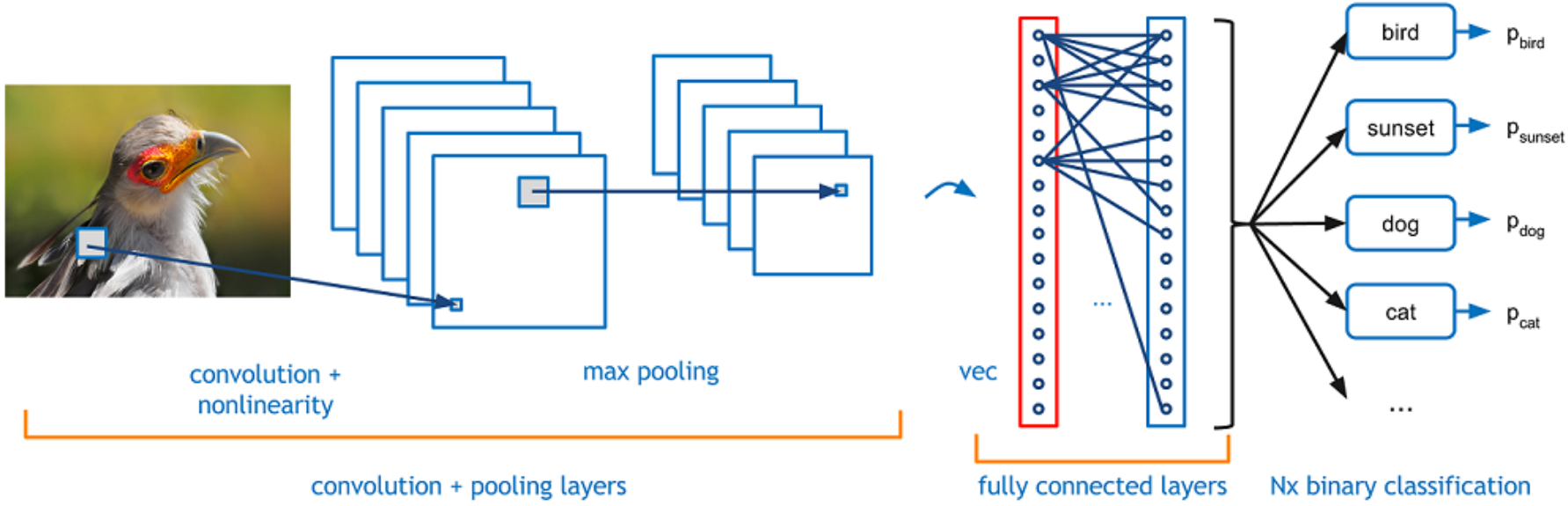
(80% for flower, 15% for sky etc).

What We Want the Computer to Do

- we want the computer to do is to be able to differentiate between all the images it's given and figure out the **unique features** that make a dog a dog , that make a cat a cat or that make a flower a flower .
- The computer is able to perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of **convolutional layers**.

Convolution Neural Network

CNNs take the image, pass it through a series of convolutional, nonlinear, pooling (downsampling), and fully connected layers, and get an output.



How computer sees an image

-1	-1	-1	-1	-1	-1	-1	-1	0.9	-0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	0.3	1	0.3	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-0	1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	0.8	1	0.6	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	0.5	1	0.8	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	0.1	1	0.9	-0	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-0	1	1	-0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	0.9	1	0.3	-1	-1	-1	-1	0.5	1	0.9	0.1	-1	-1
-1	-1	0.3	1	0.9	-1	-1	-1	0.1	1	1	1	1	1	-1	-1
-1	-1	0.8	1	0.3	-1	-1	0.4	1	0.7	-0	-0	1	1	-1	-1
-1	-1	1	1	0.1	-1	0.1	1	0.3	-1	-1	-0	1	0.6	-1	-1
-1	-1	1	1	0.8	0.3	1	0.7	-1	-1	-1	0.5	1	0	-1	-1
-1	-1	0.8	1	1	1	1	0.5	0.2	0.8	0.8	1	0.9	-1	-1	-1
-1	-1	-0	0.8	1	1	1	1	1	1	1	1	0.1	-1	-1	-1
-1	-1	-1	-0	0.8	1	1	1	1	1	1	0.2	-1	-1	-1	-1
-1	-1	-1	-1	-1	-0	0.3	0.8	1	0.5	-0	-1	-1	-1	-1	-1

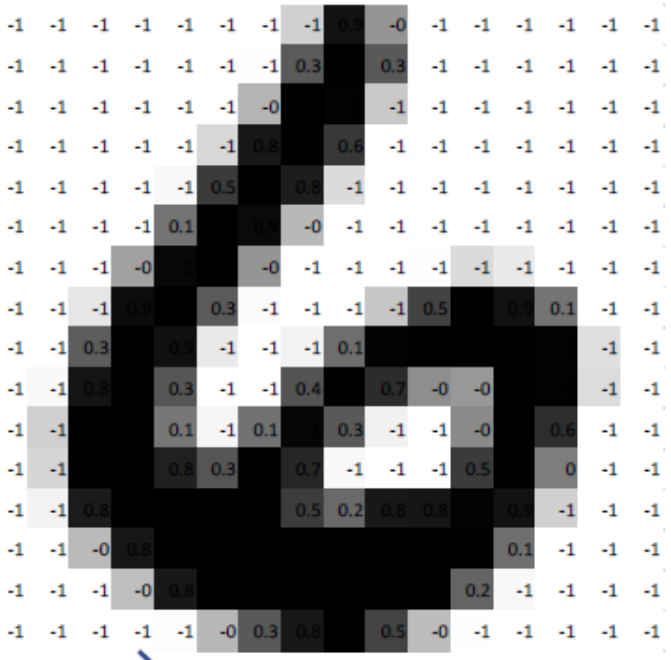
Humans see this

Computer sees this

How computer sees an image

-1	-1	-1	-1	-1	-1	-1	-1	0.9	-0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	0.3	1	0.3	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-0	1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	0.8	1	0.6	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	0.5	1	0.8	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	0.1	1	0.9	-0	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-0	1	1	-0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	0.9	1	0.3	-1	-1	-1	-1	0.5	1	0.9	0.1	-1	-1
-1	-1	0.3	1	0.9	-1	-1	-1	0.1	1	1	1	1	1	-1	-1
-1	-1	0.8	1	0.3	-1	-1	0.4	1	0.7	-0	-0	1	1	-1	-1
-1	-1	1	1	0.1	-1	0.1	1	0.3	-1	-1	-0	1	0.6	-1	-1
-1	-1	1	1	0.8	0.3	1	0.7	-1	-1	-1	0.5	1	0	-1	-1
-1	-1	0.8	1	1	1	1	0.5	0.2	0.8	0.8	1	0.9	-1	-1	-1
-1	-1	-0	0.8	1	1	1	1	1	1	1	1	0.1	-1	-1	-1
-1	-1	-1	-0	0.8	1	1	1	1	1	1	0.2	-1	-1	-1	-1
-1	-1	-1	-1	-1	-0	0.3	0.8	1	0.5	-0	-1	-1	-1	-1	-1

Computer sees this



Same matrix, highlight the cells based on cell value

How computer sees an image

Human Vision

Computer Vision

Kernel Matrix examples

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur

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

Kernel to make image sharper

-1	-1	-1
-1	9	-1
-1	-1	-1

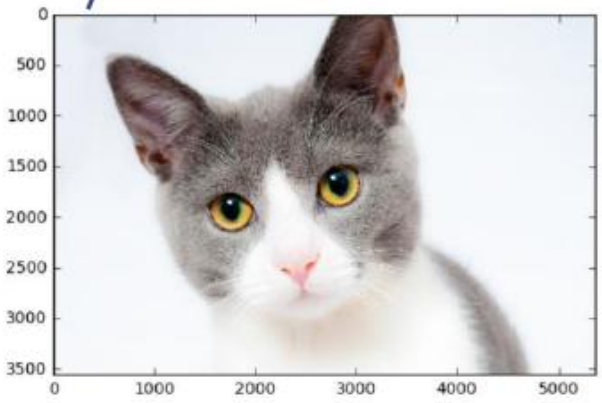
Intensified sharper image

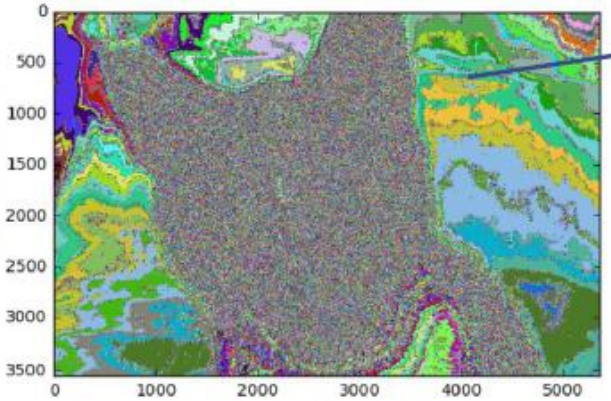
Gaussian Blur

Sharpened image

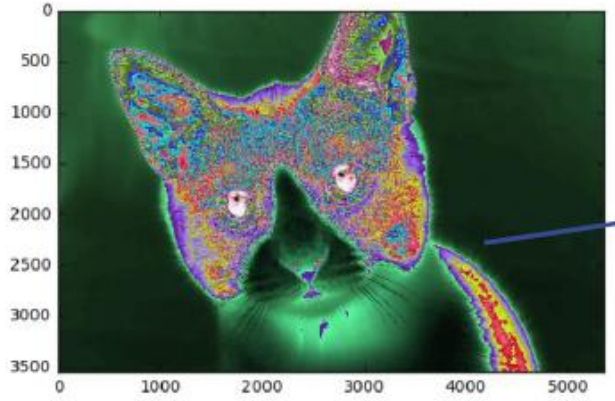
Convoluted features

Different filters capture different features





A filter to capture overall shape and edges



A filter to ignore white spaces on image

The depth in convolution layer

- Every filter gives us a resultant matrix (activation map)

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

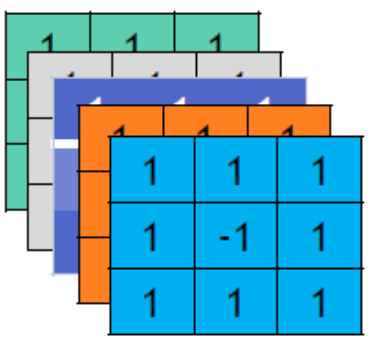
1	1	1
1	1	1
1	1	1

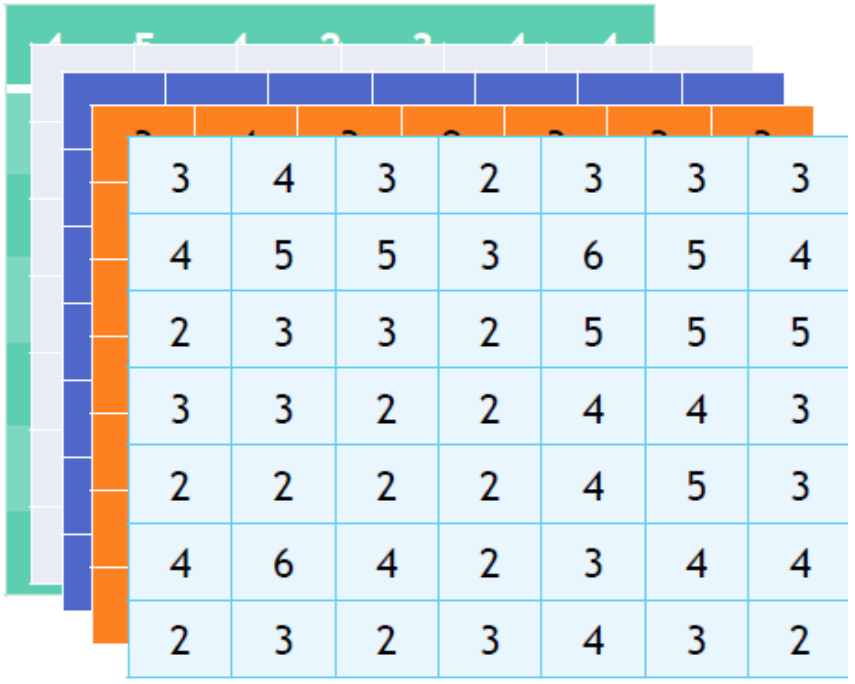
4	5	4	2	3	4	4
5	6	5	4	6	6	5
3	3	3	3	6	6	5
3	3	2	2	4	5	4
3	3	2	2	4	5	4
5	6	4	3	4	5	4
3	4	3	3	4	4	3

The depth in convolution layer

- If we apply 10 different filters then we will get 10 resultant matrices. The depth is 10

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





Max Pooling

Pooling layer

- Pooling helps in further downsizing the data
- Pooling is used to avoid overfitting and increase the robustness
- Reduces lot of computation time

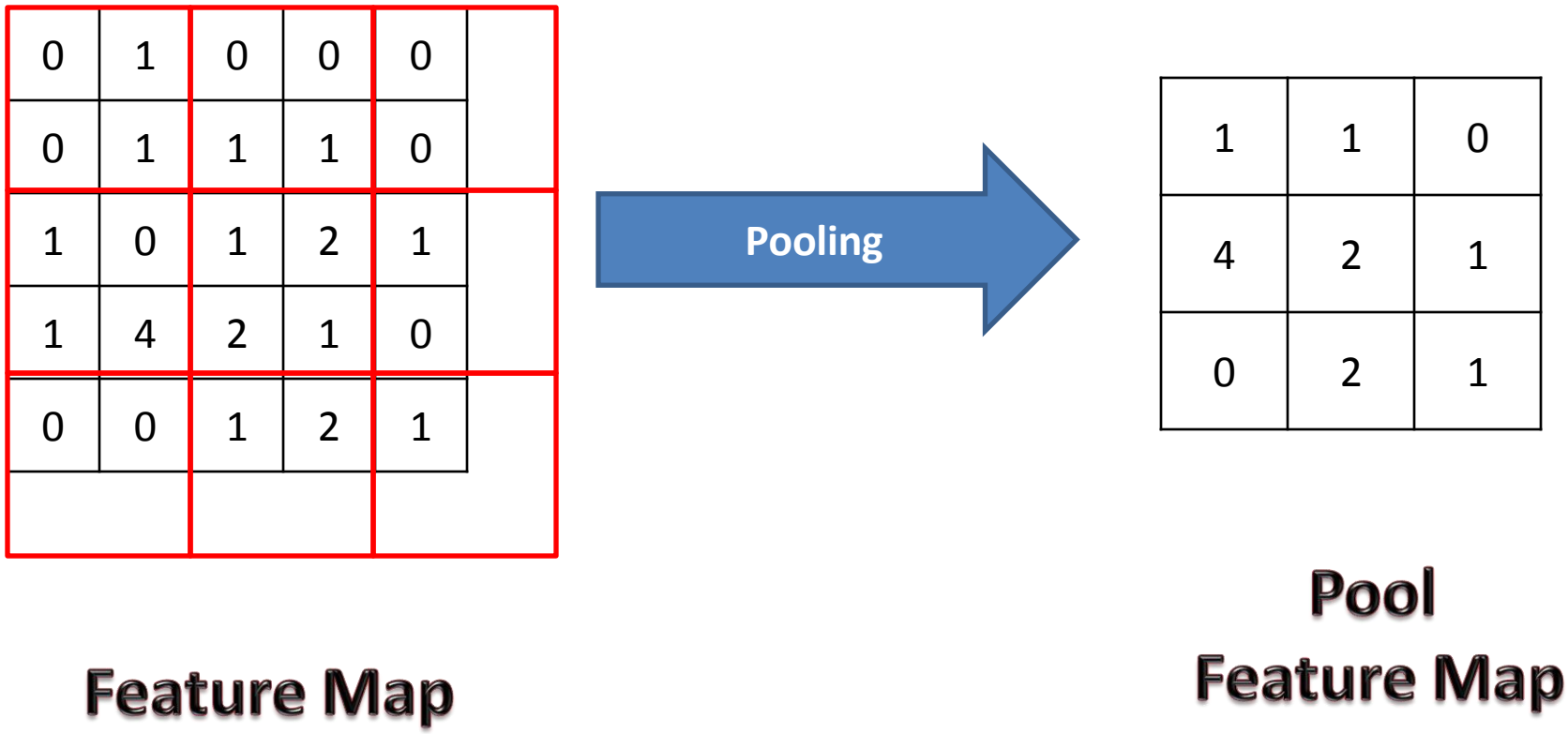
What happens in pooling layer

- Down sampling of the data.
- Since we preserved lot of information in each convolution layer, we can comfortably down sample it, in this pooling layer
- Yes, we do loose some information, but we will not loose the overall integrity of the data
- It is still good enough for a classification model
- Generally we try max pooling with 2X2 filter with stride 2

How many parameters in pooling layer

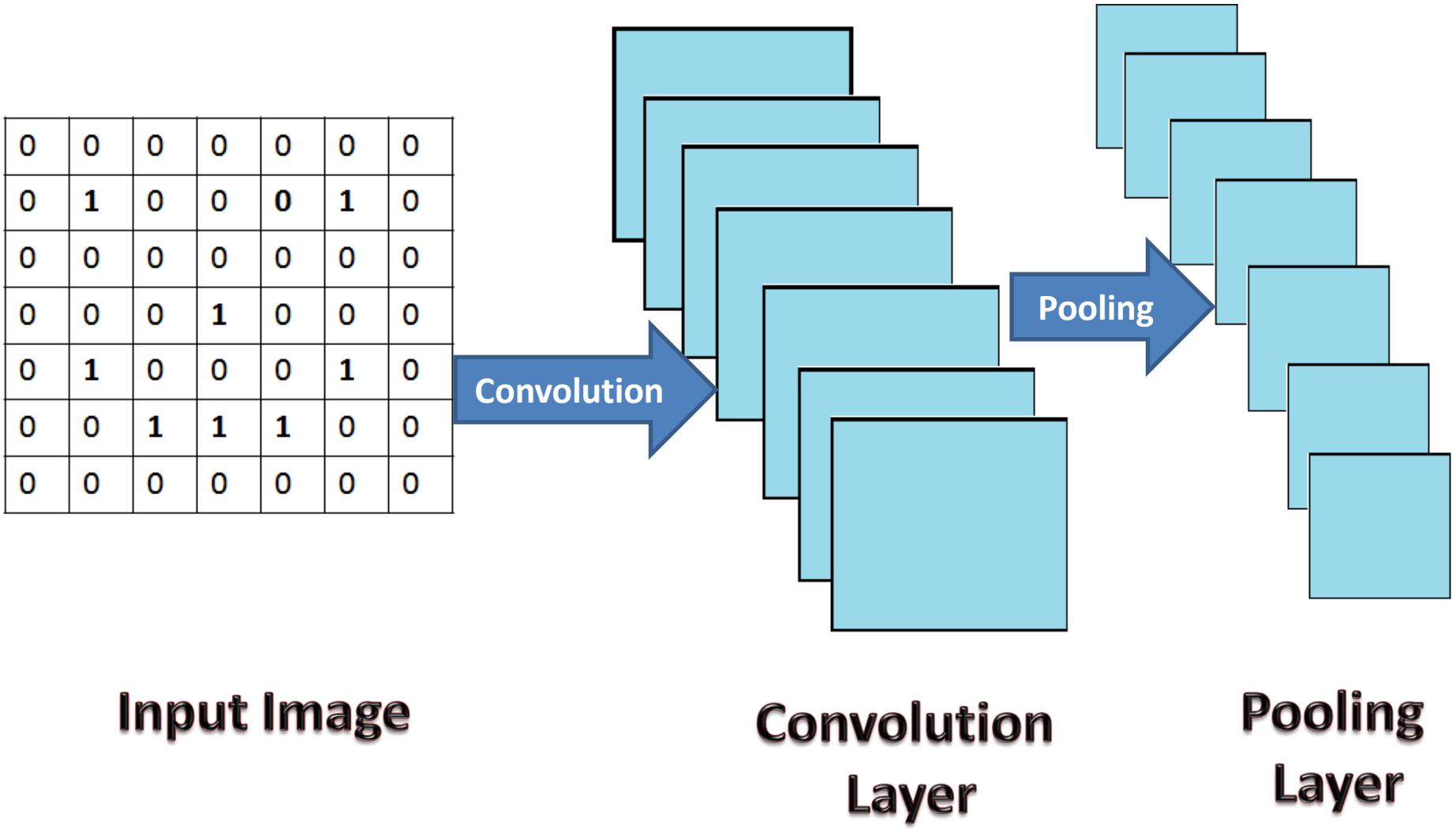
- No parameters
- We just perform down sampling, that's it. No further activation and no further parameters
- Number of parameters in pooling layer=0

Max Pooling



Max pooling is usually done with 2x2 windows and stride 2, so as to downsample the feature maps

Max Pooling


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
Convolution
Pooling
Input Image
Convolution Layer
Pooling Layer

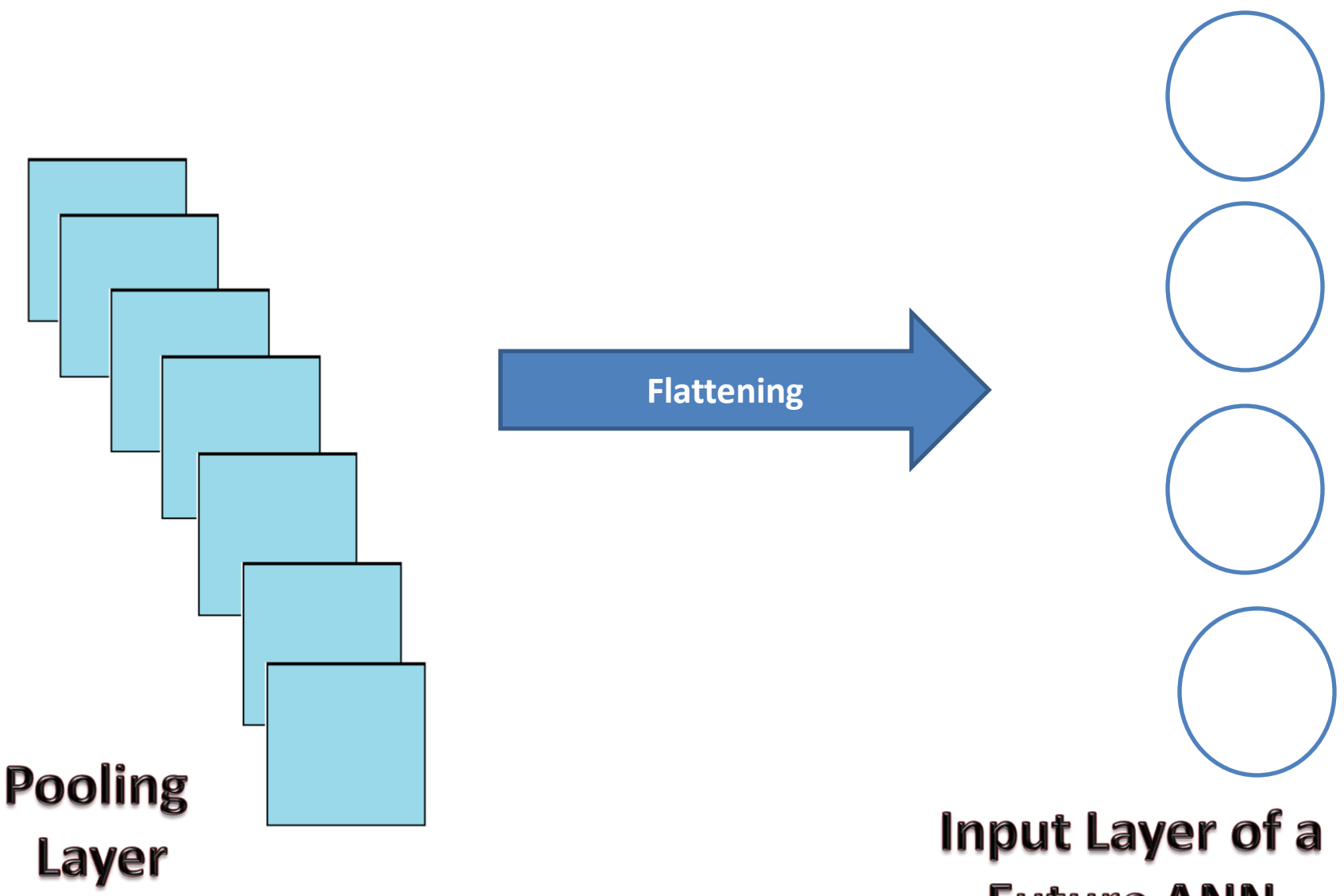
Flattening

1	1	0
4	2	1
0	2	1

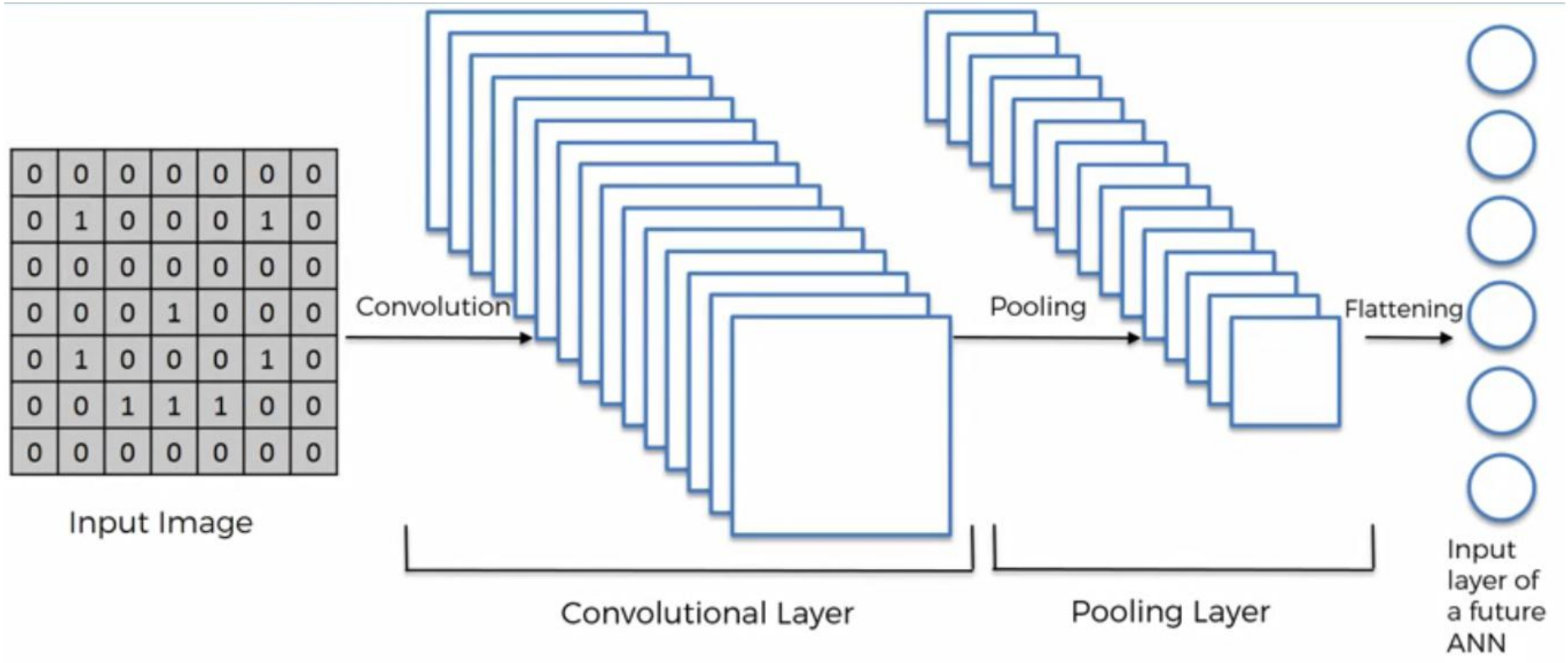
Pool
Feature Map

1
1
0
4
2
1
0
2
1

Flattening



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
Pooling
Flattening
Convolutional Layer
Pooling Layer
Input layer of a future ANN

Full Connection