LECTURE - 1

Workshop Neural Networks

SHAPEAI

GETTING STARTED WITH NEURAL NETWORKS

HISTORY

Neural Networks were inspired in the early 1940s by researchers tried to implement the same ideas of neuroscience to computers.

COMMONLY USED IN

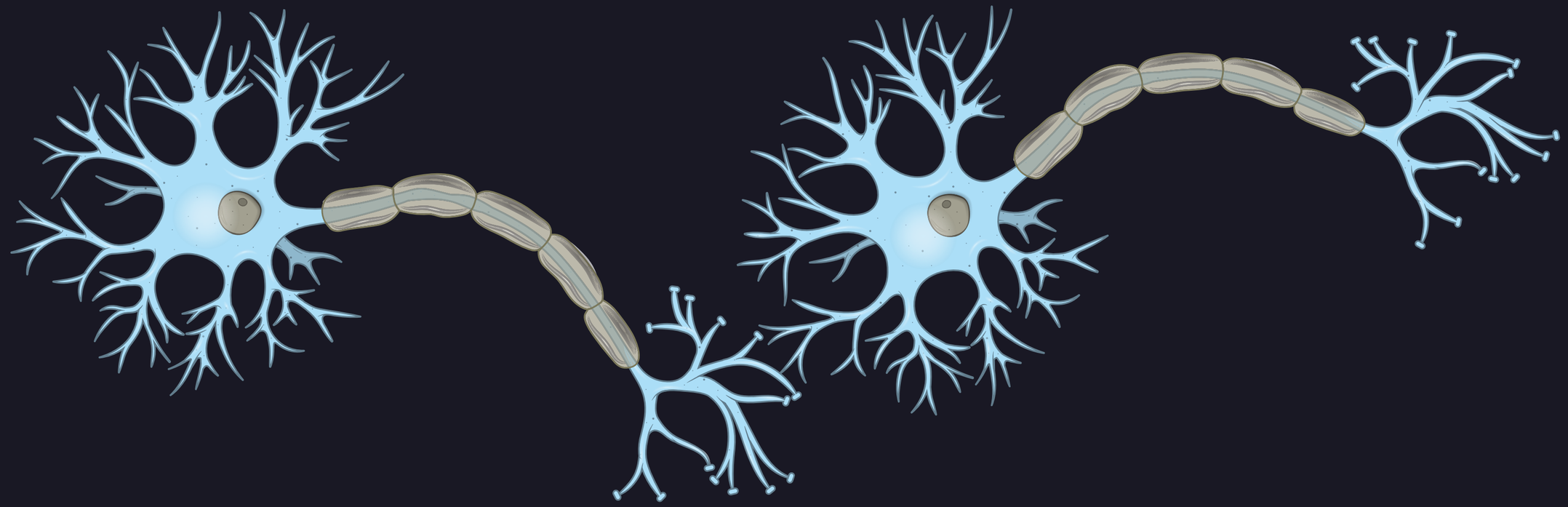
CNNs are commonly used in image search services, self-driving cars, recommendation system, fraud detection, and many more systems.

ALSO USED IN

They are also successful in many other tasks, such as voice recognition and natural language processing. There is just so much to neural networks.

03

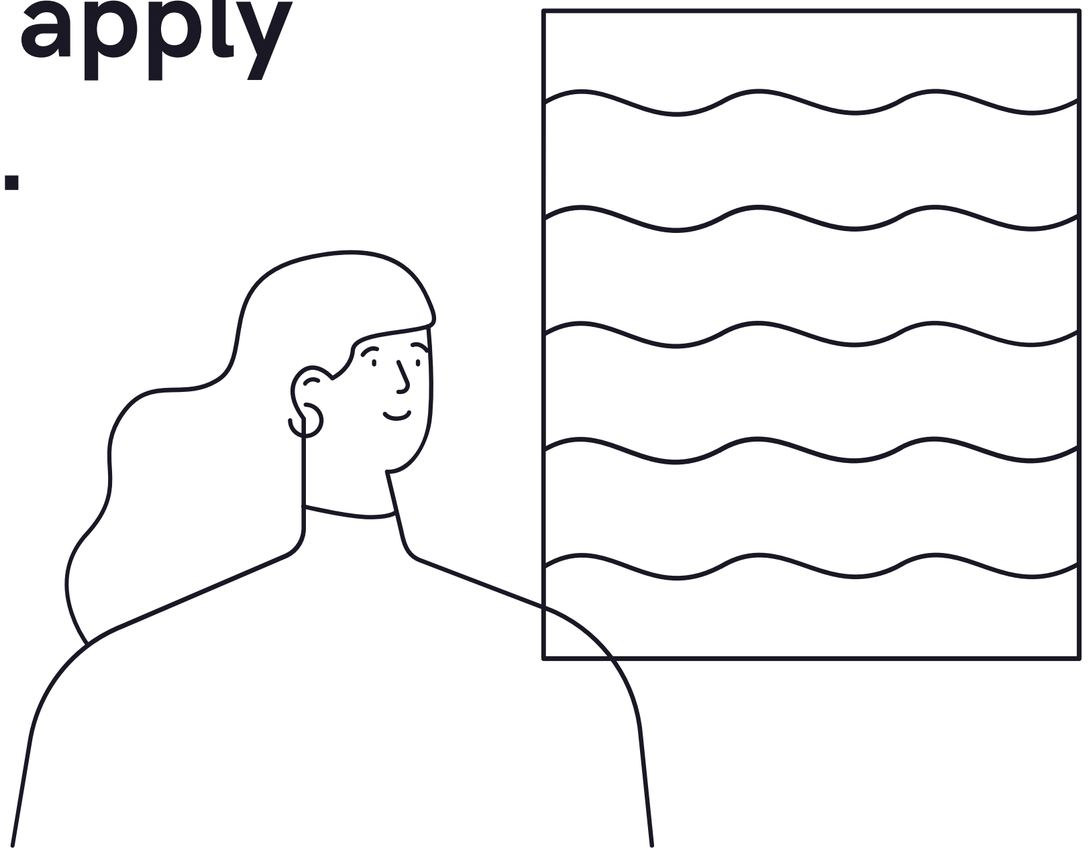
NEURAL NETWORKS

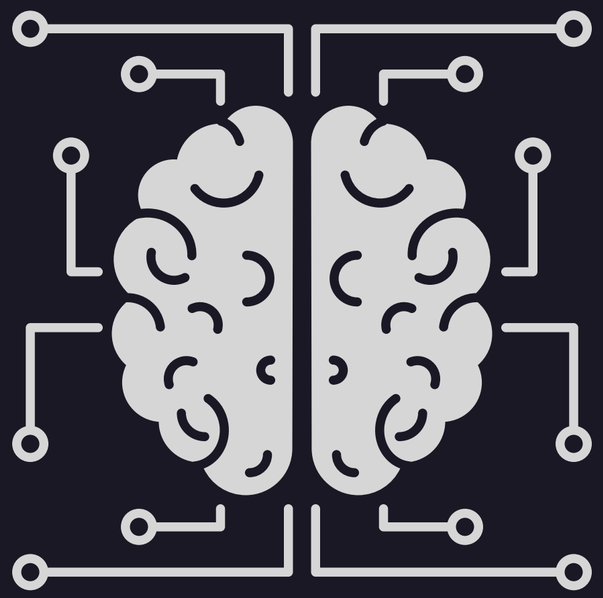


NEURAL NETWORKS

- Neurons are connected to and receive electrical signals from other neurons.
- Neurons process input signals and can be activated.

Then the question arised can we take in this biological idea of How humans learn? and apply that to machines as well.





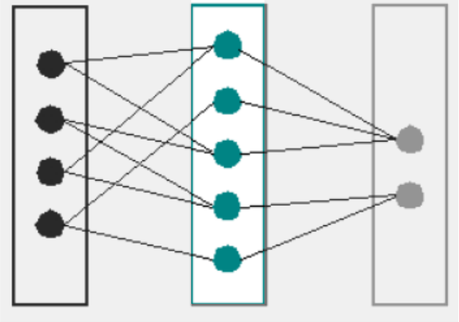
ARTIFICIAL NEURAL NETWORKS

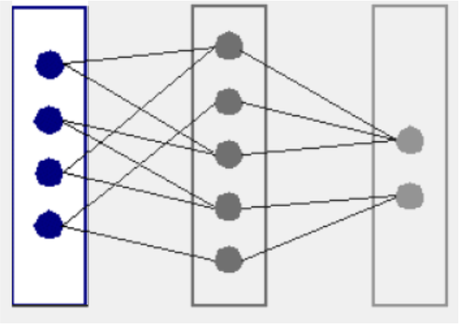
Mathematical Model for learning inspired by biological neural networks

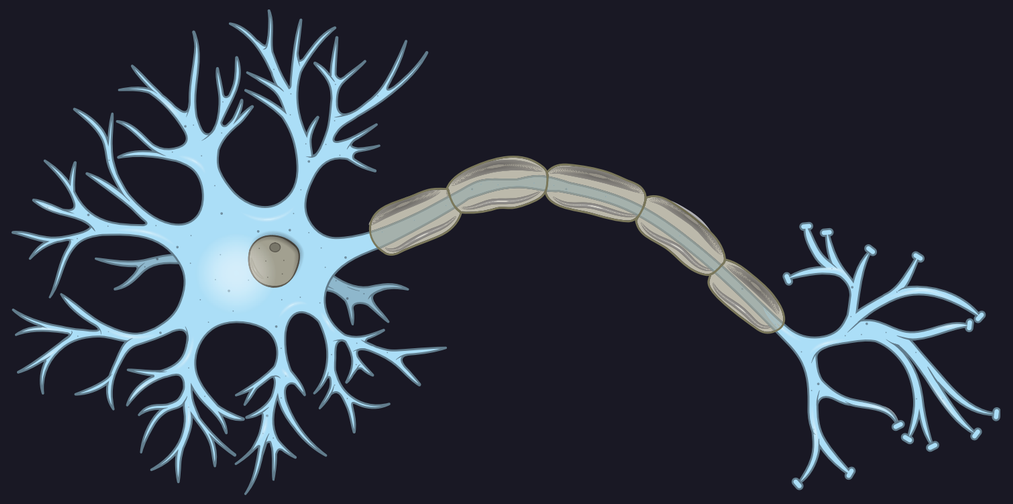
ARTIFICIAL NEURAL NETWORKS

- Model mathematical function from inputs to the outputs based on the structure and parameters of the network.

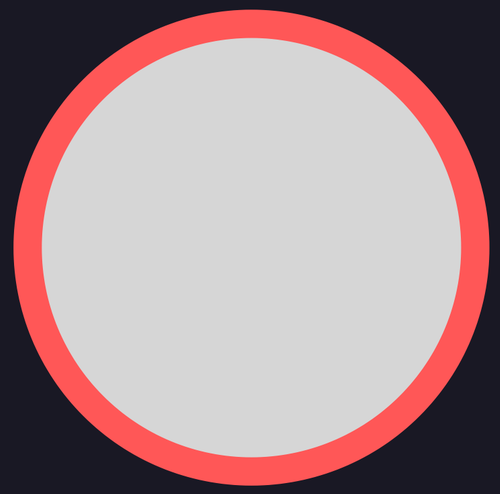
- Allows for learning the network's parameters based on data.



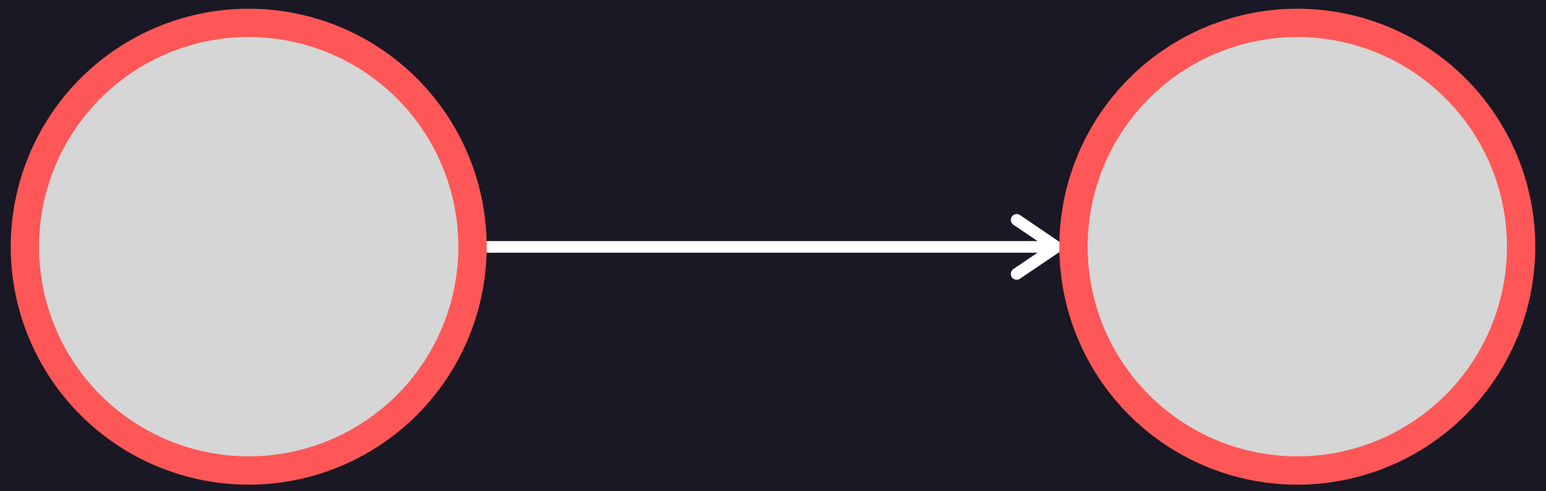




biological neurons



Neuron / Units



These Neurons can be
connected to one another

(x1, x2) ⟶

Using some input attributes predict death in titanic.

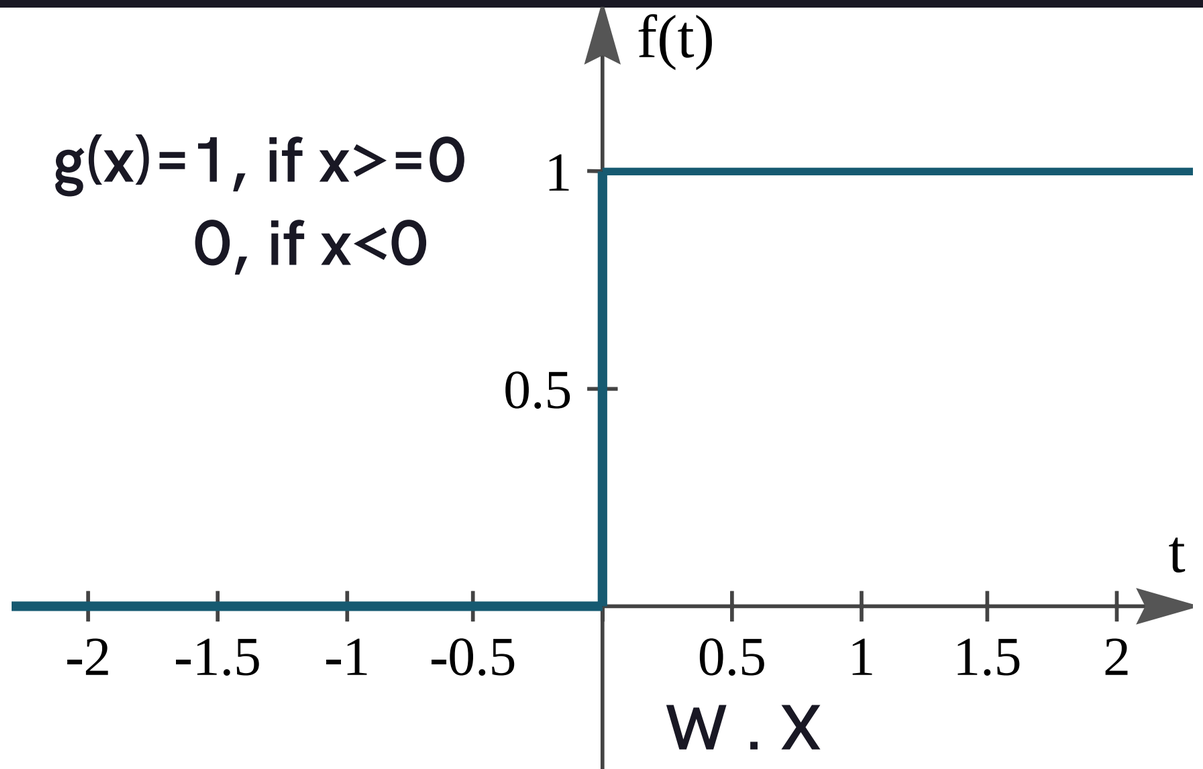
$$h(x1, x2) = w_0 + w_1 x_1 + w_2 x_2$$

In order to determine this hypothesis function we just need to determine what these wights should be.

ACTIVATION FUNCTION

Step Function

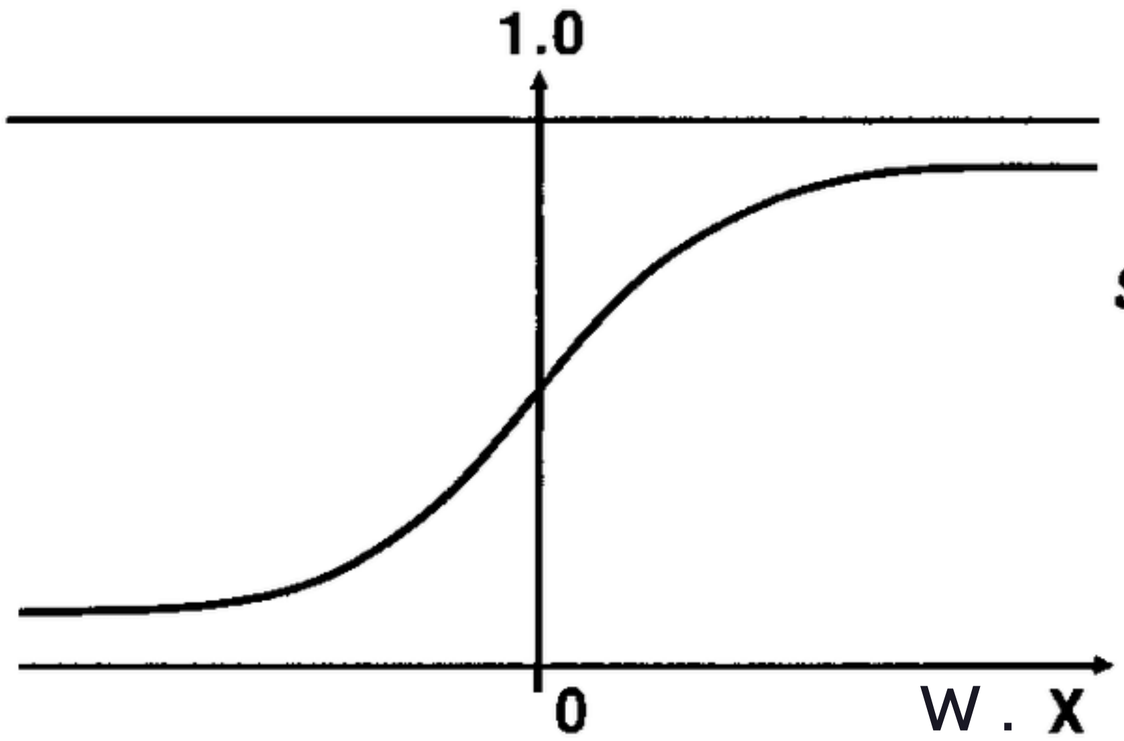
At the end of the day we need to do some classification ie. will a person live or die if they were on titanic. And hence we will have to define some type of threshold.



ACTIVATION FUNCTION

Sigmoid Function

There will some times were we don't want 0 or 1 we want somewhere in between like the probability of a persons death on titanic.



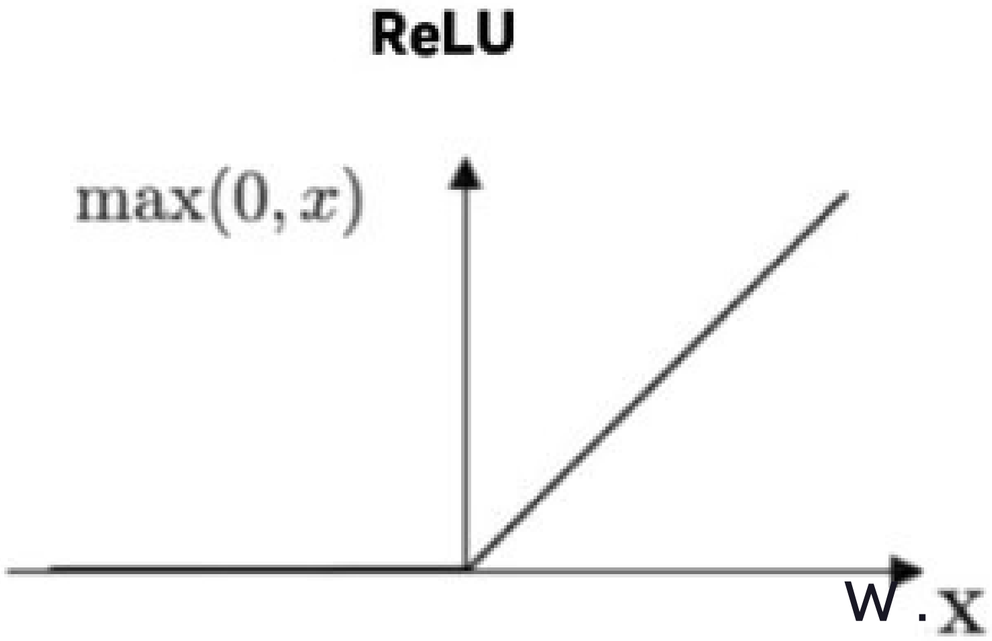
Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

ACTIVATION FUNCTION

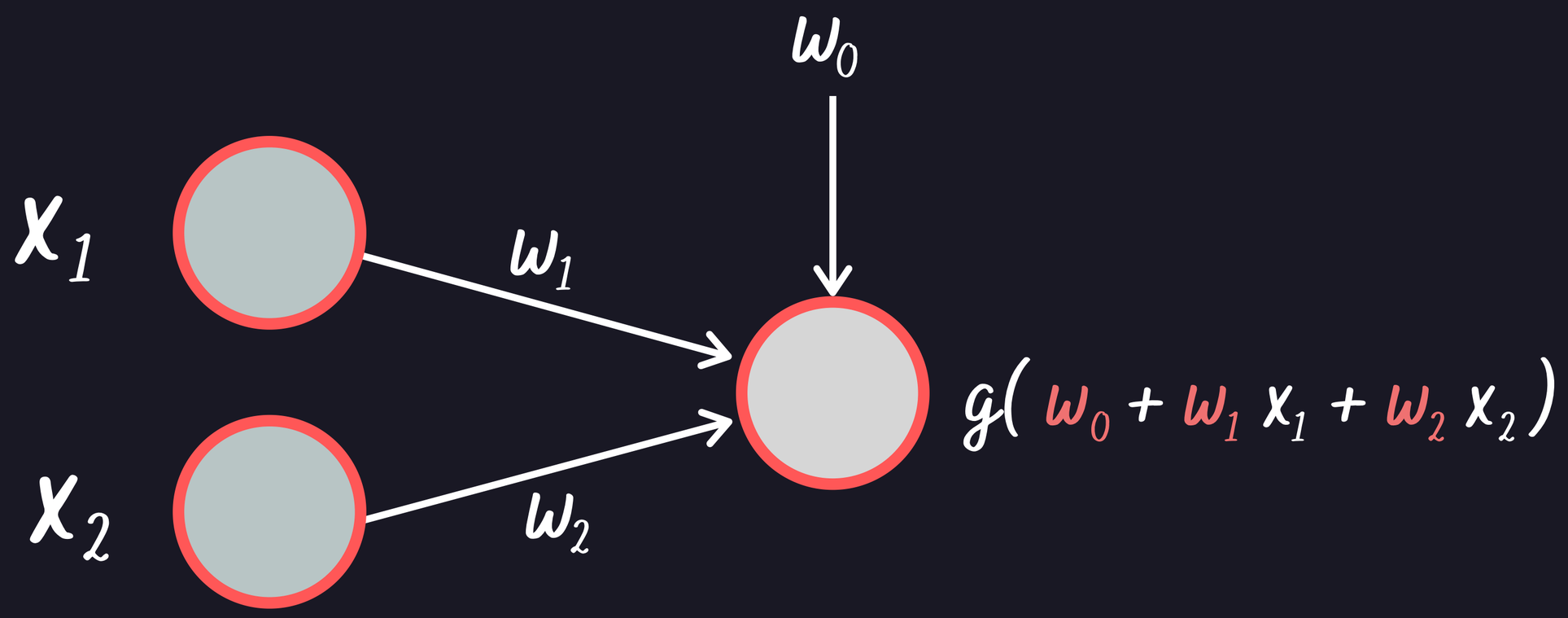
ReLU Function

This is an other activation function where in if w.x is positive then it remains unchanged while if it is negative then it becomes 0.



$$h(x1, x2) = g(\, w_0 + w_1\, x_1 + w_2\, x_2\,)$$

Activation function can be thought of as an other function g() that is applied to the result of all of this computation.



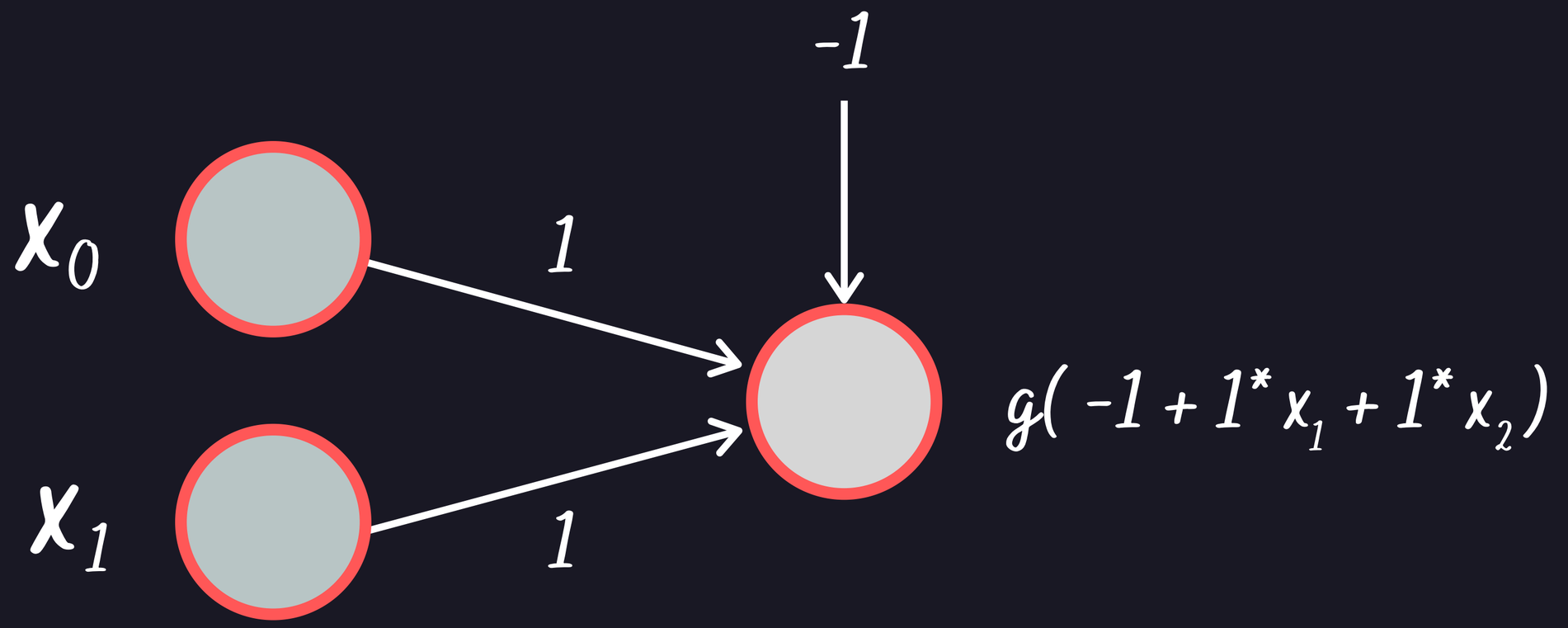
We will represent the above mathematical formula using a structure like this.

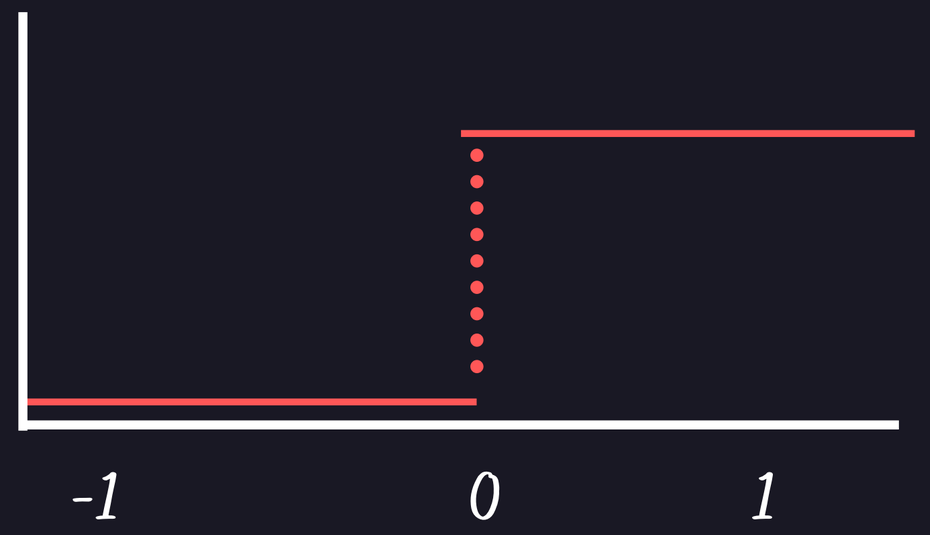
OR FUNCTION

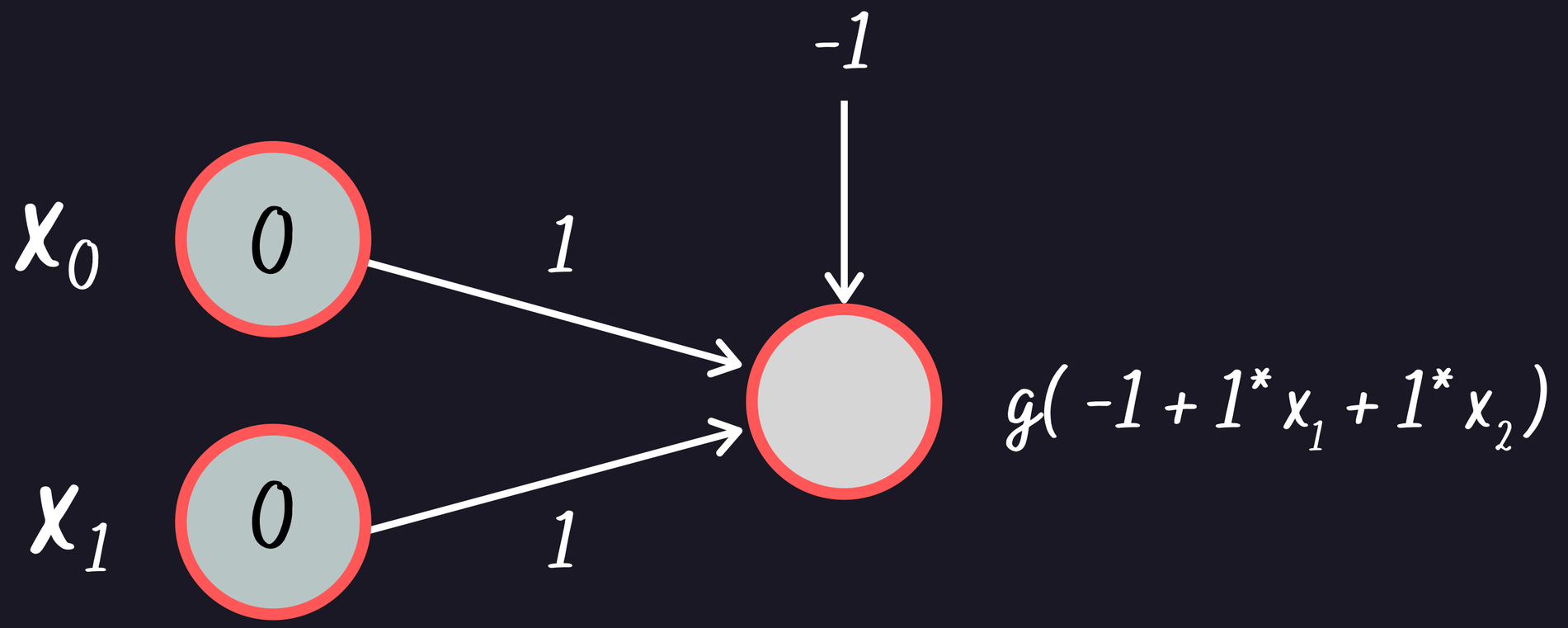
Taking a simple example of OR function to understand neural networks.

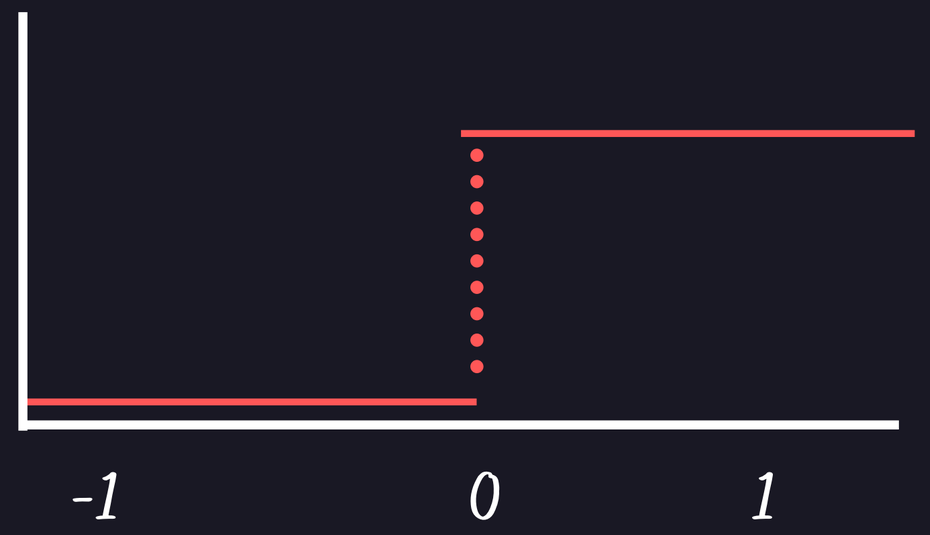
Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

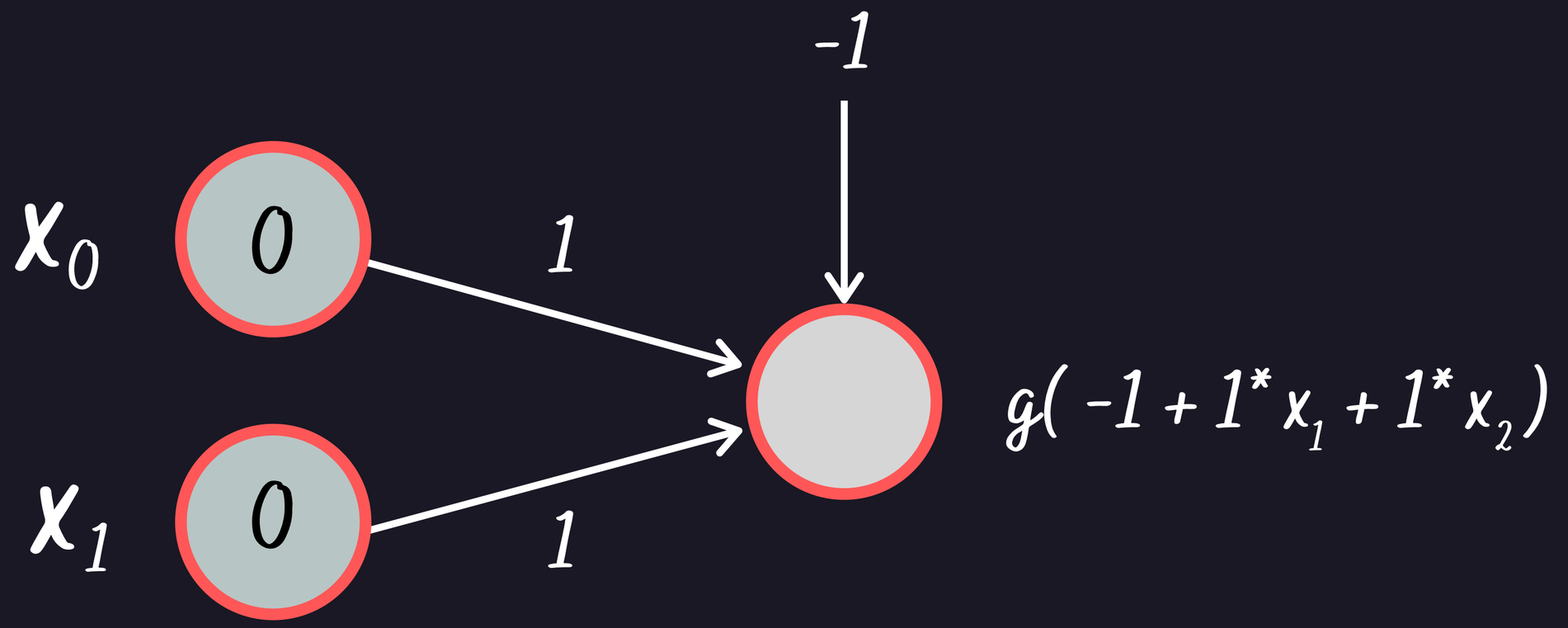
w

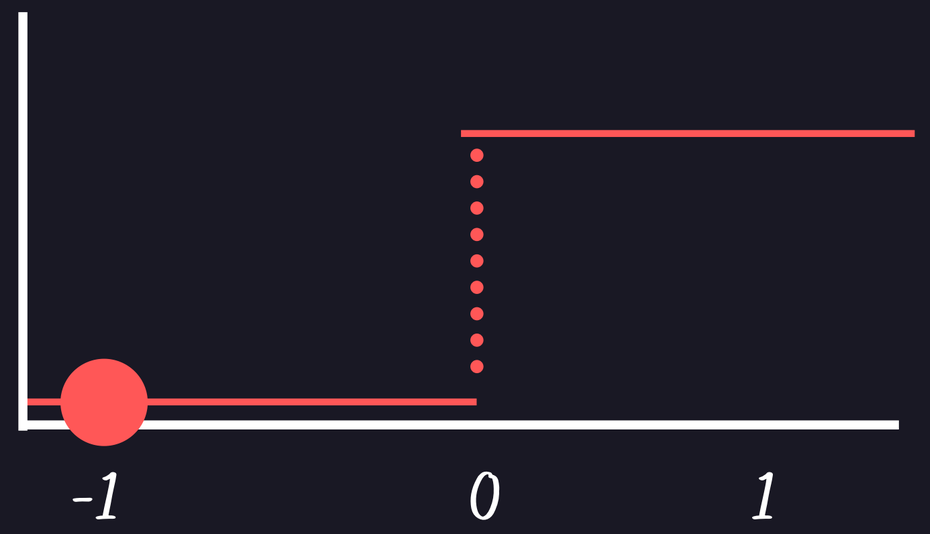

-1
x_0
1
x_1
1
$g(-1 + 1^* x_1 + 1^* x_2)$

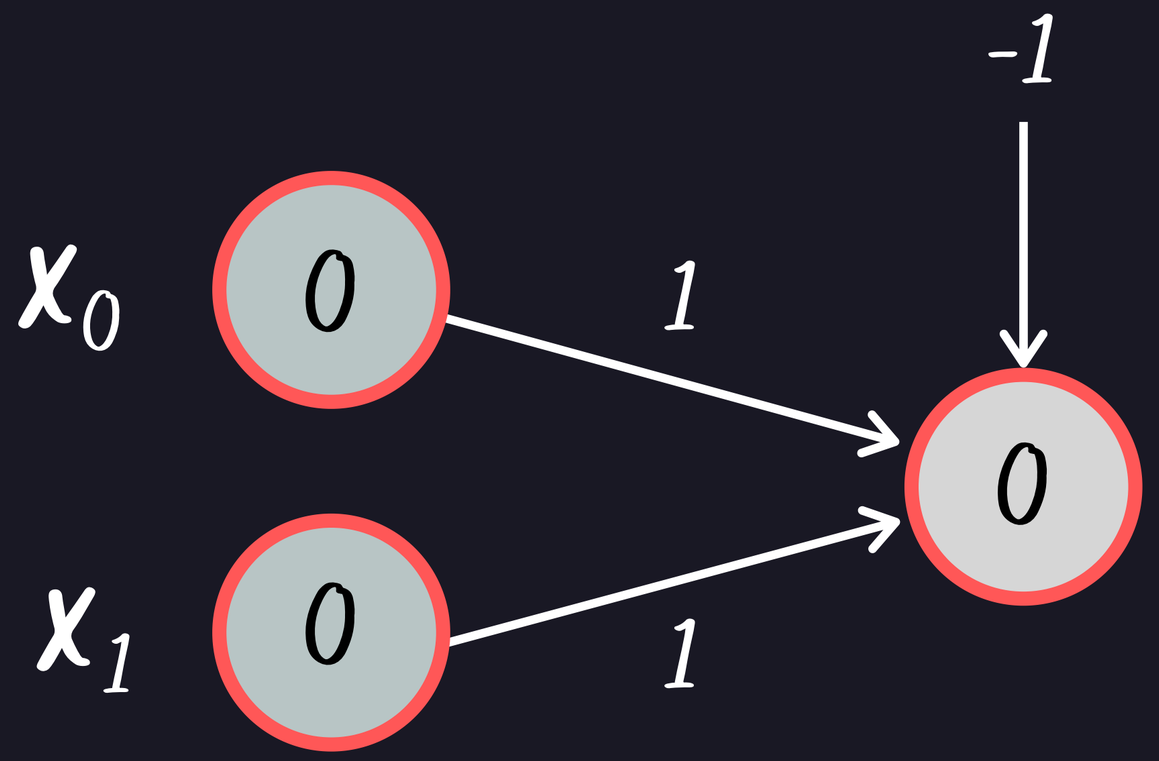

-1
0
1


-1
x_0
0
1
x_1
0
1
$g(-1 + 1^* x_1 + 1^* x_2)$

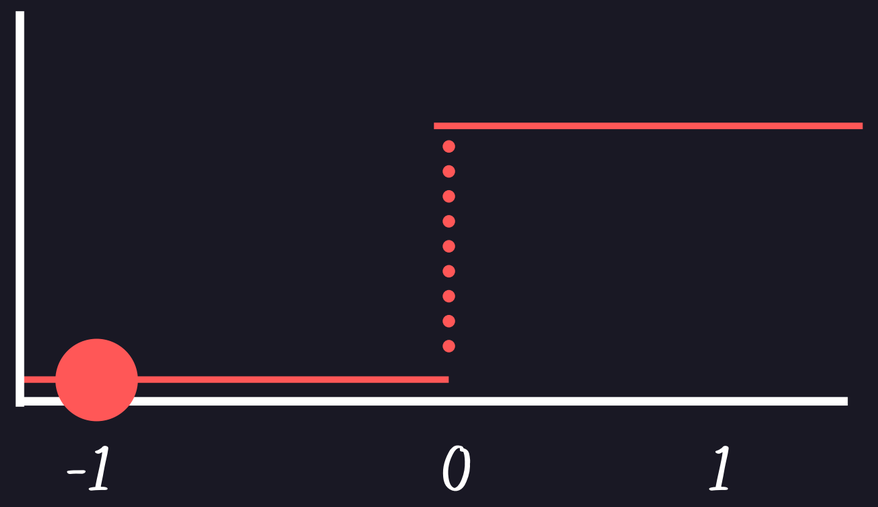

-1
0
1


-1
x_0
0
1
x_1
0
1
$g(-1 + 1^* x_1 + 1^* x_2)$

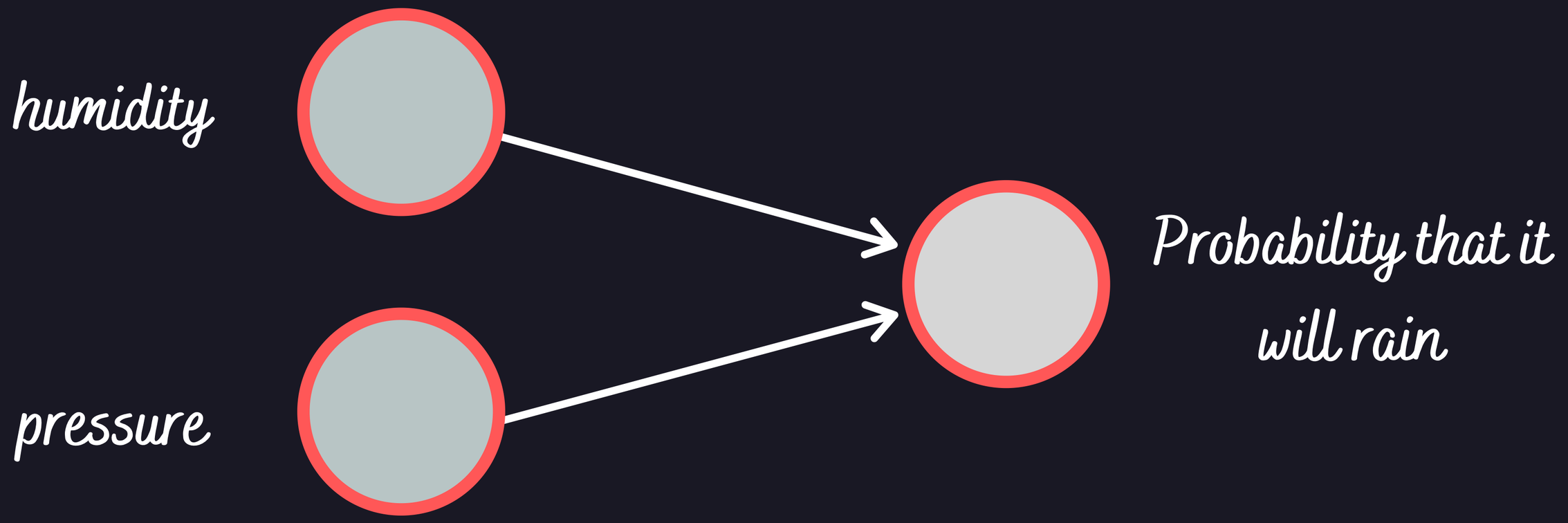

-1
0
1



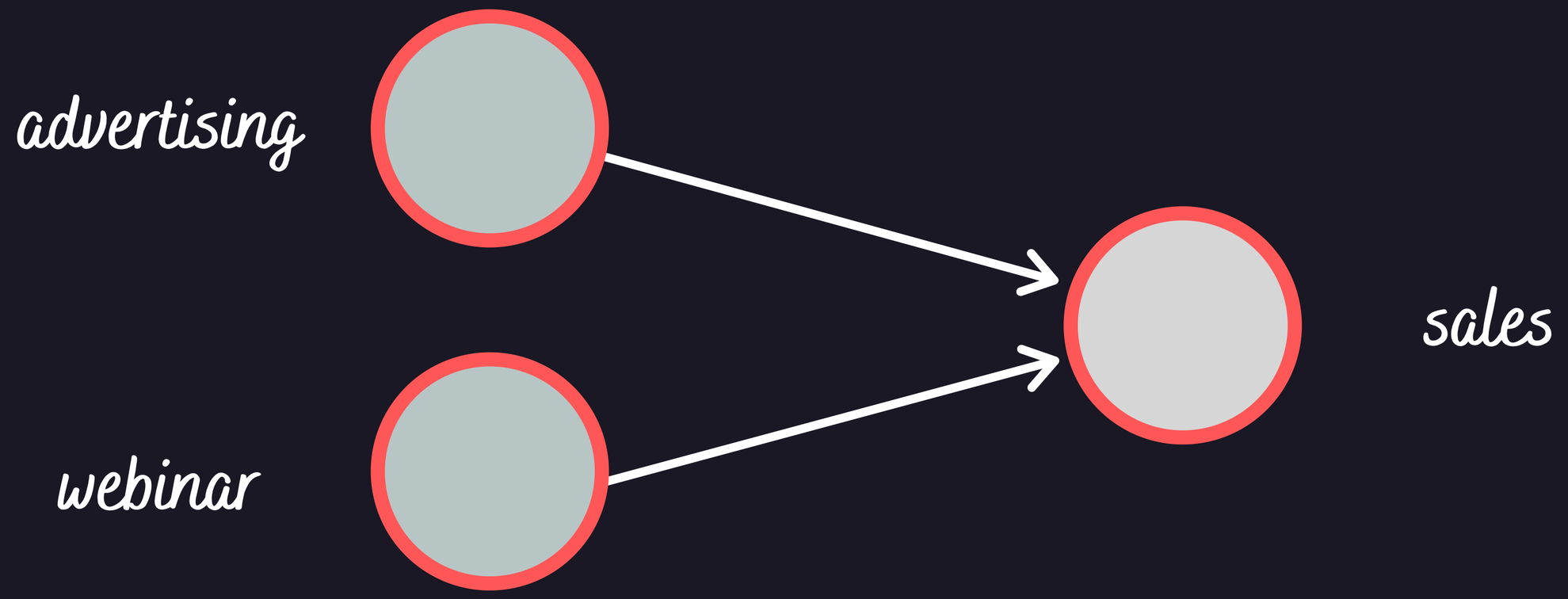
$$g(-1 + 1^* x_1 + 1^* x_2)$$



CLASSIFICATION



REGRESSION



COMPLEX NEURAL NETWORKS

DEEP NEURAL NETWORKS

FEED-FORWARD NEURAL NETWORKS

- Neural Networks that has connections in only one direction.
- Such that the inputs move from the input to the hidden layers and then to the output. They propagate only in one direction.

LOSS FUNCTION

Loss (x, y, w)

A Loss function quantifies how unhappy you would be if you used 'w' to make a prediction on x when the correct output is y.

TYPES OF LOSSES

$y-y'$

Residual loss

$(y-y')^2$

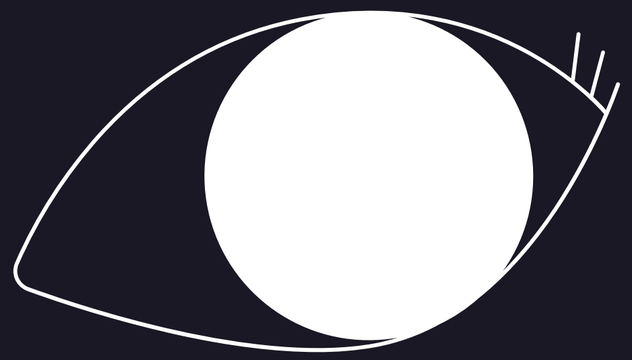
Squared Loss

$|y-y'|$

Absolute loss

W1 VS W2 PLOT

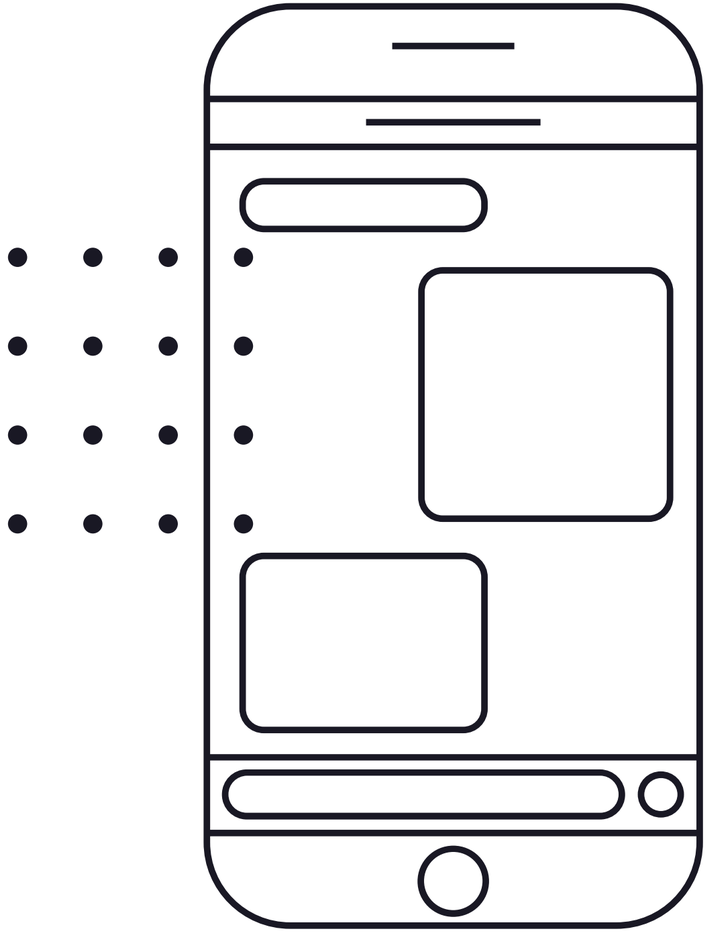
GRADIENT DESCENT

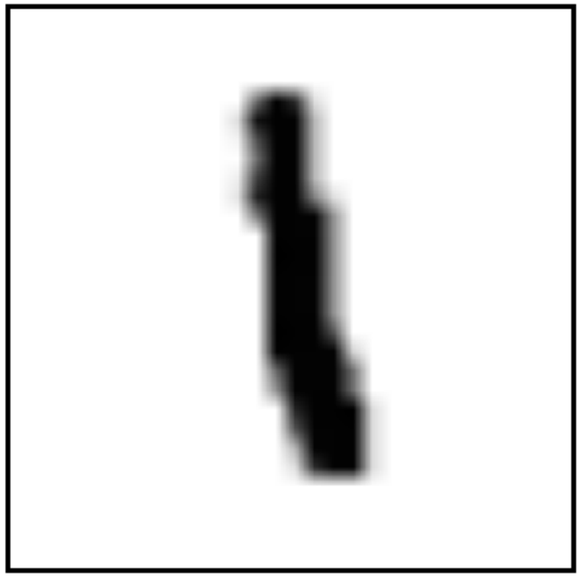


CONVOLUTIONAL NEURAL NETWORK (CNN)

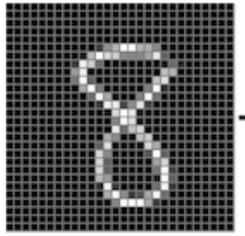
HOW A DIGITAL DEVICE SEES AN IMAGE

For a device an image is nothing but a 2D matrix of numbers. Where each entry represent the intensity of light for the following position or so called pixels.



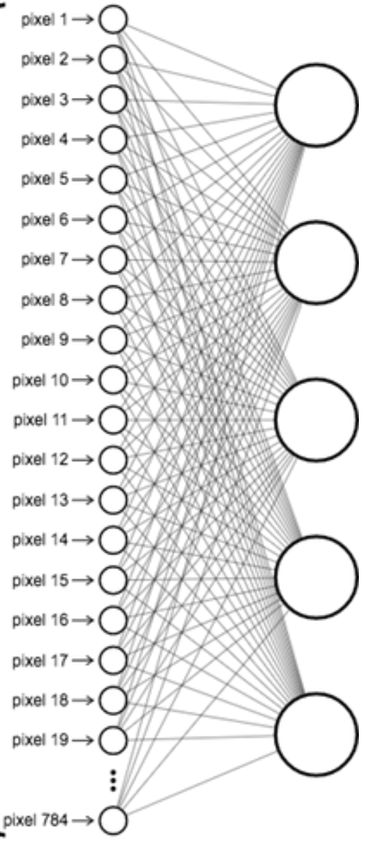


$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$



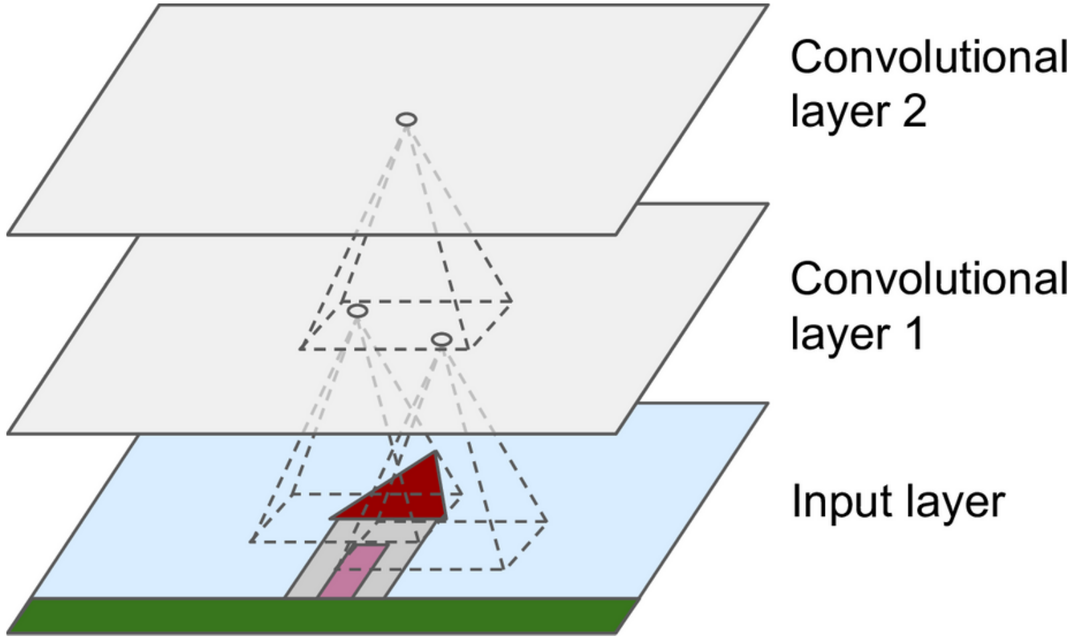
28 x 28
784 pixels

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	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	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	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	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	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	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	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	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	0	0	0	0	0	0	0	0	0	0	0	0	0

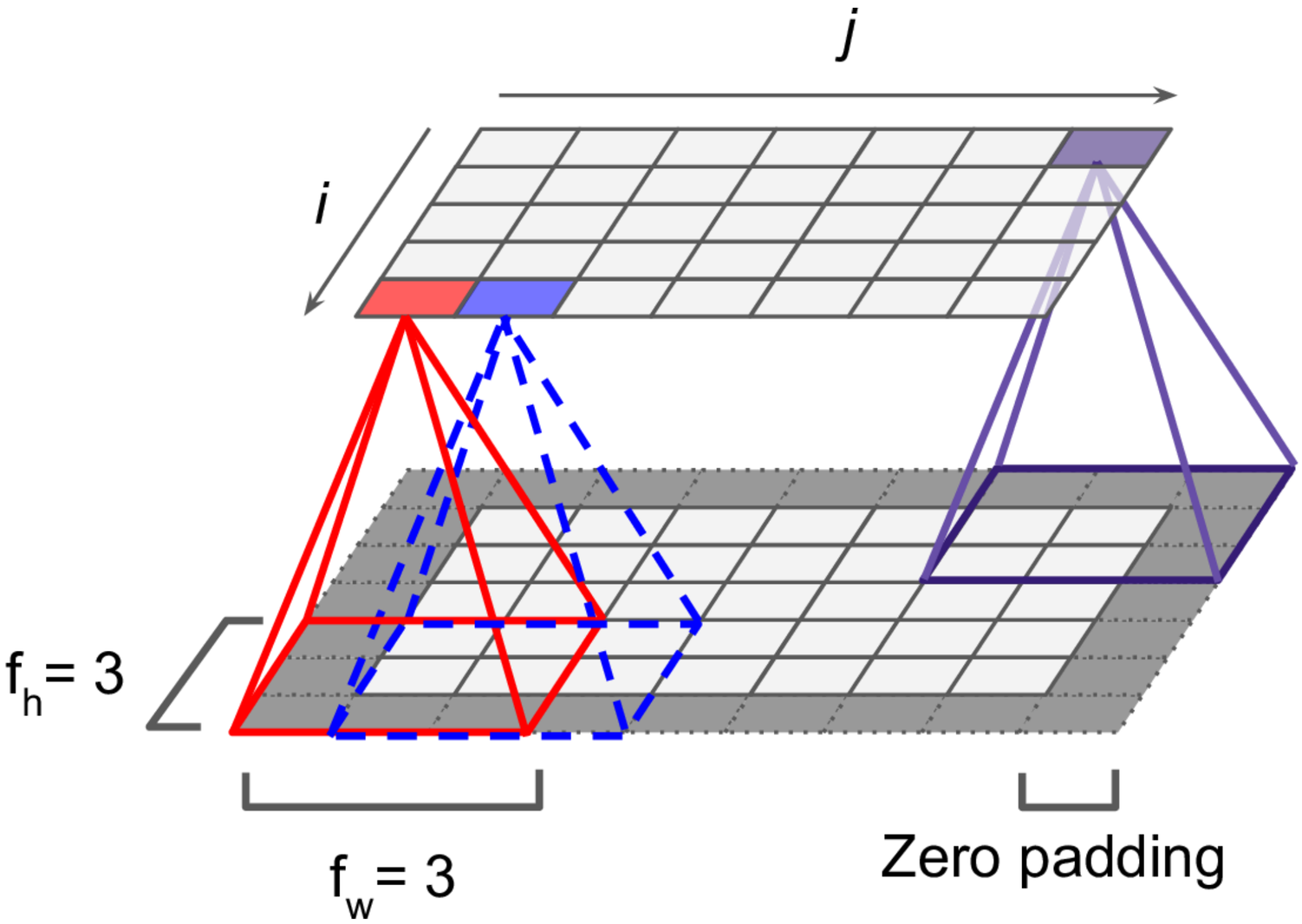


CONVOLUTIONAL LAYER

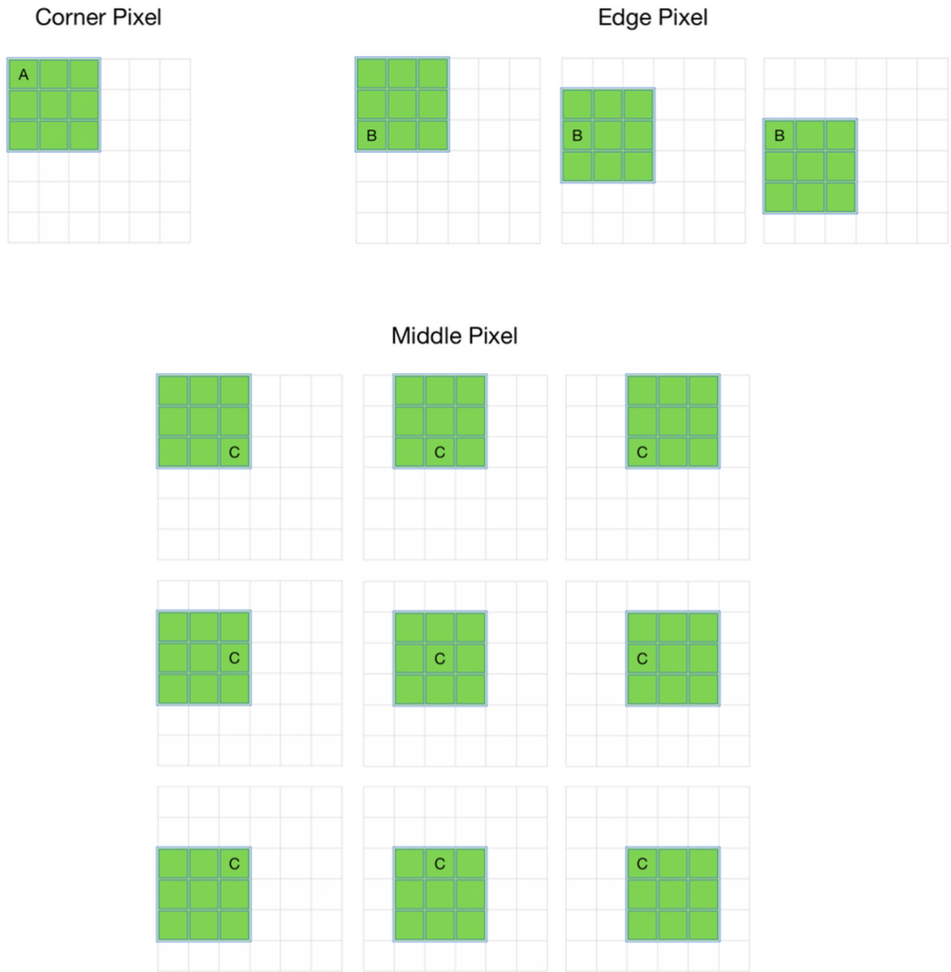
- Neurons in the first convolutional layer are not connected to every single pixel in the input image but only to pixels in their receptive fields.
- In turn, each neuron in the second convolutional layer is connected only to neurons located within a small rectangle in the first layer.
- This architecture allows the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer, and so on.
- This hierarchical structure is common in real-world images, which is one of the reasons why CNNs work so well for image recognition.

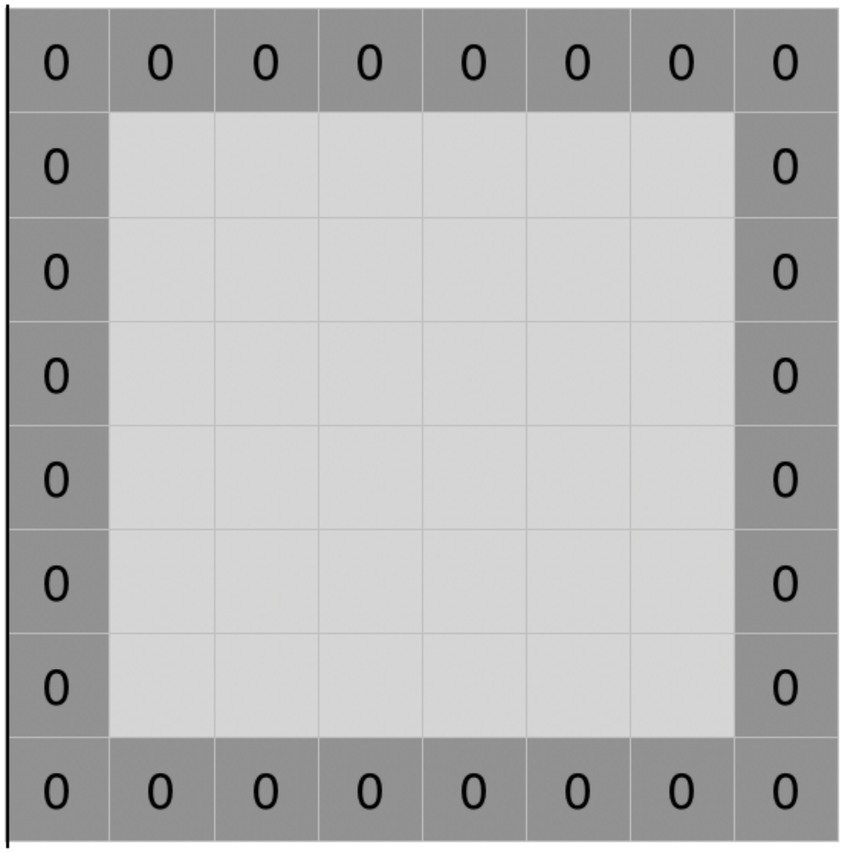


CONVOLUTIONAL LAYER



CONVOLUTIONAL LAYER

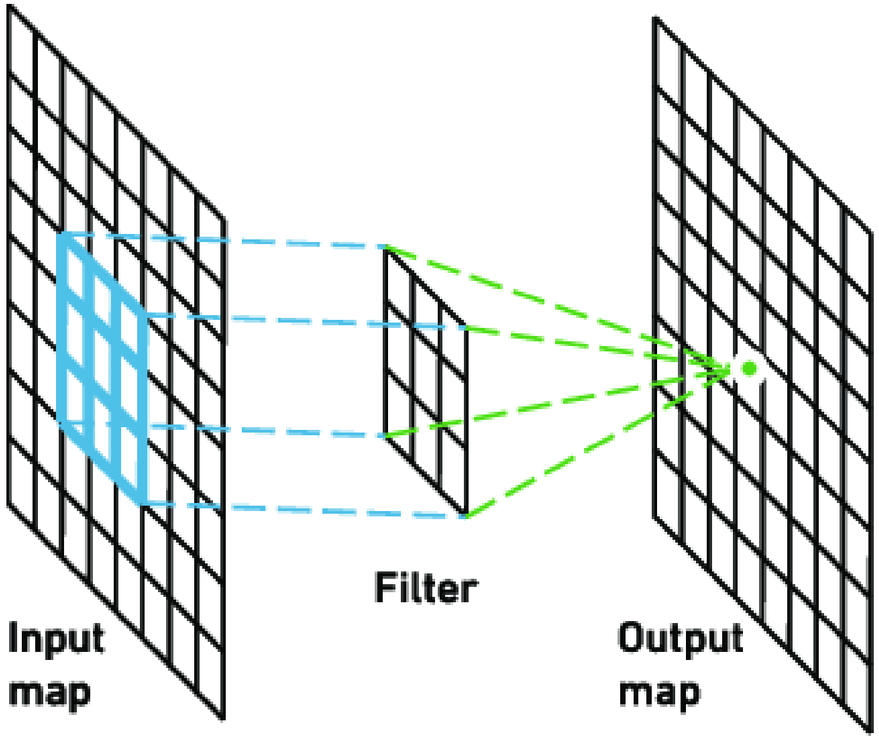




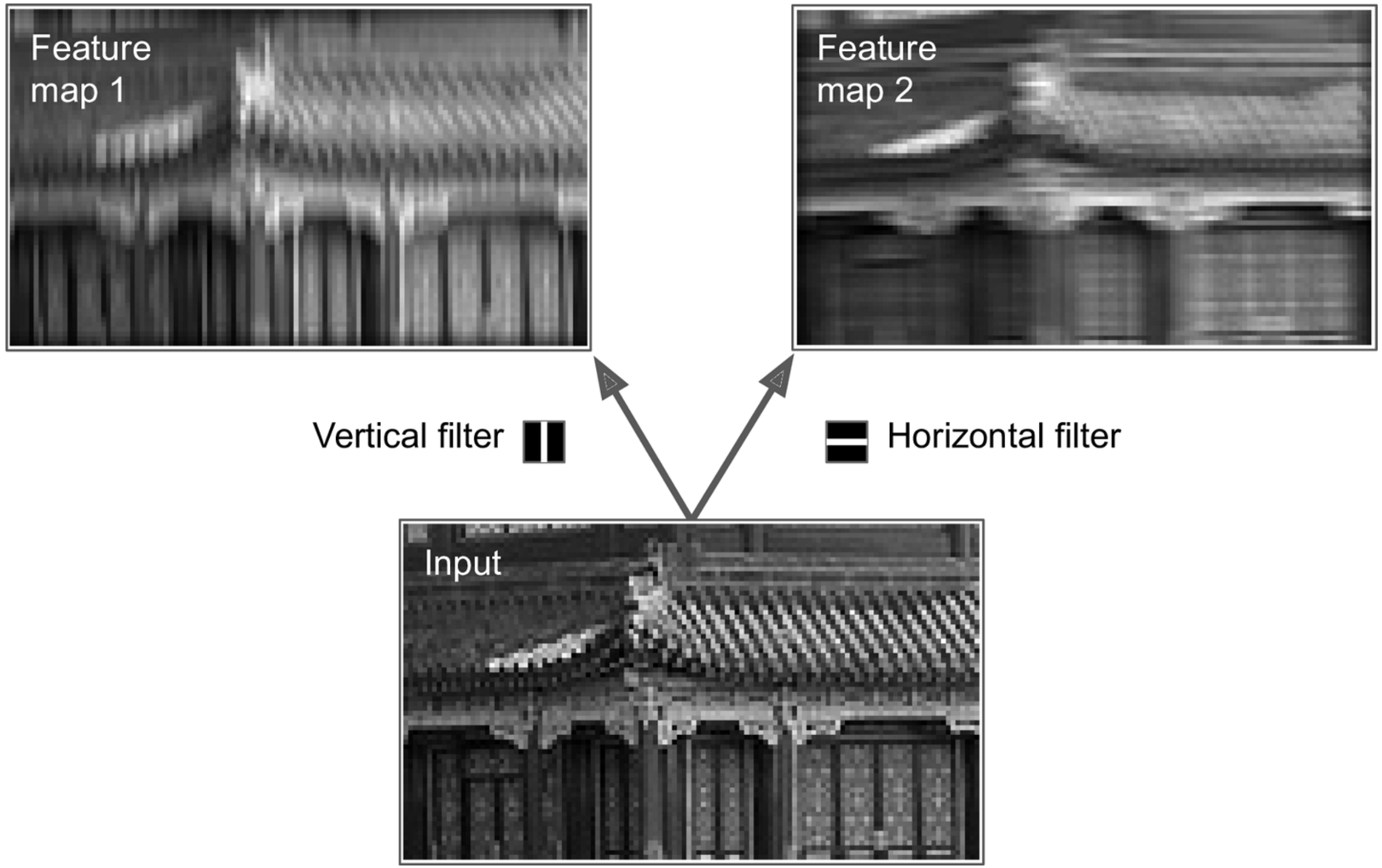
Zero-padding added to image

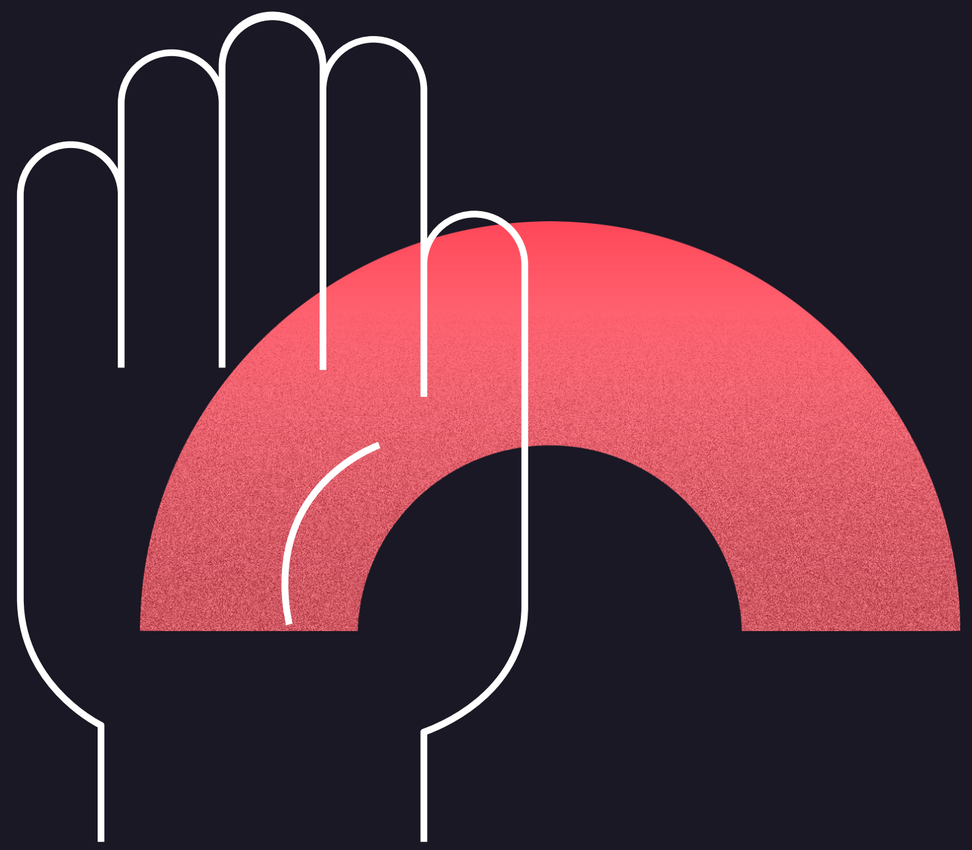
FILTER

A convolutional layer contains a set of filters whose parameters need to be learned. The height and weight of the filters are smaller than those of the input volume. Each filter is convolved with the input volume to compute an activation map made of neurons.



FILTER


Feature
map 1
Feature
map 2
Vertical filter
Horizontal filter
Input

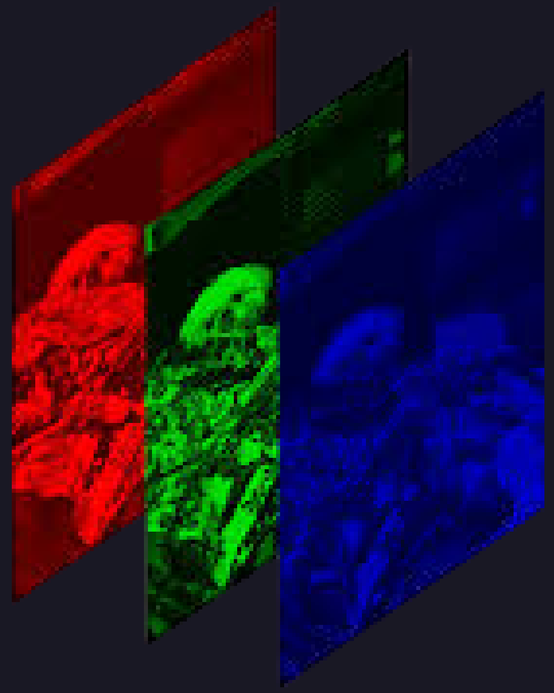


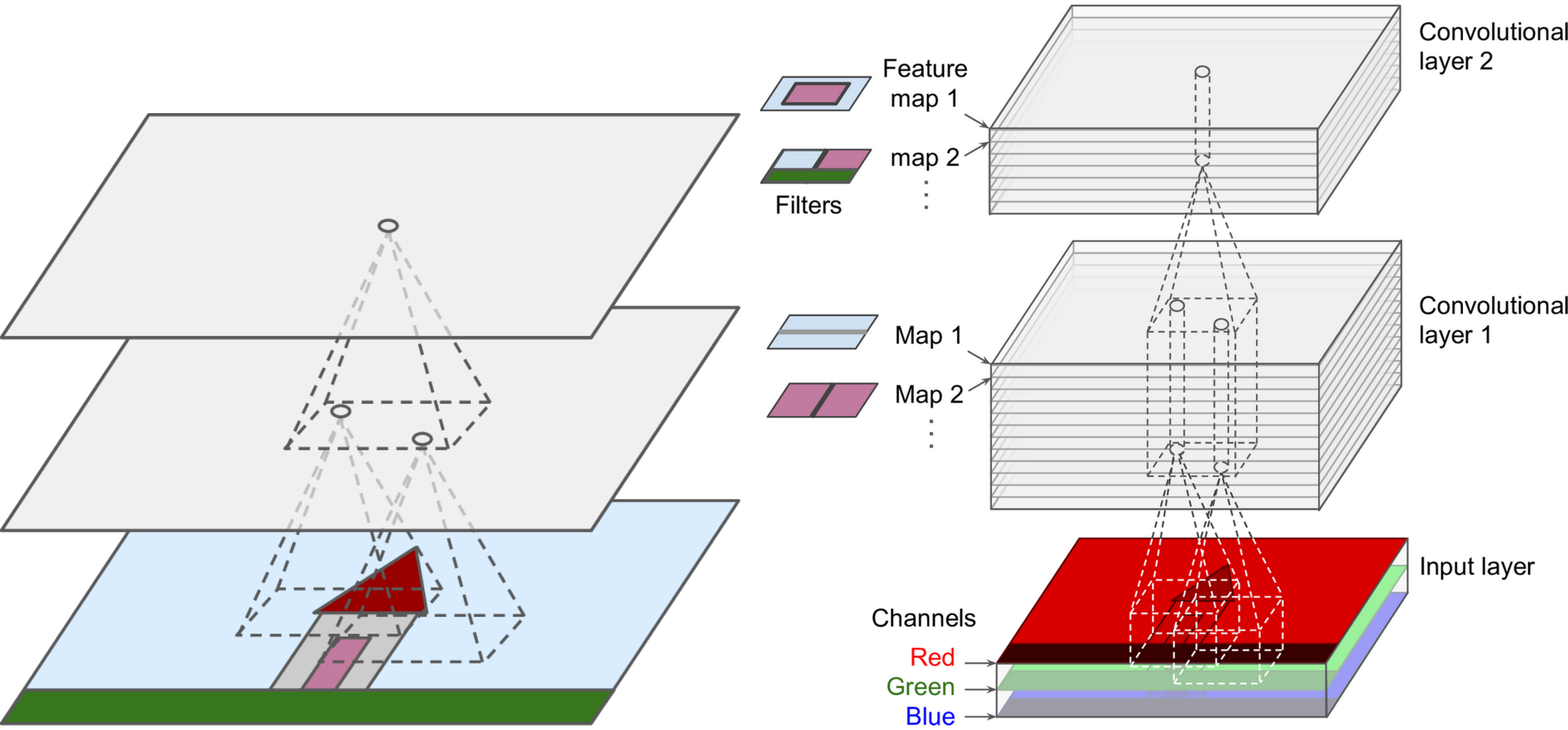
COLOURED IMAGE

Any colour can be represented as a combination of three colours:

- RED
- GREEN
- BLUE

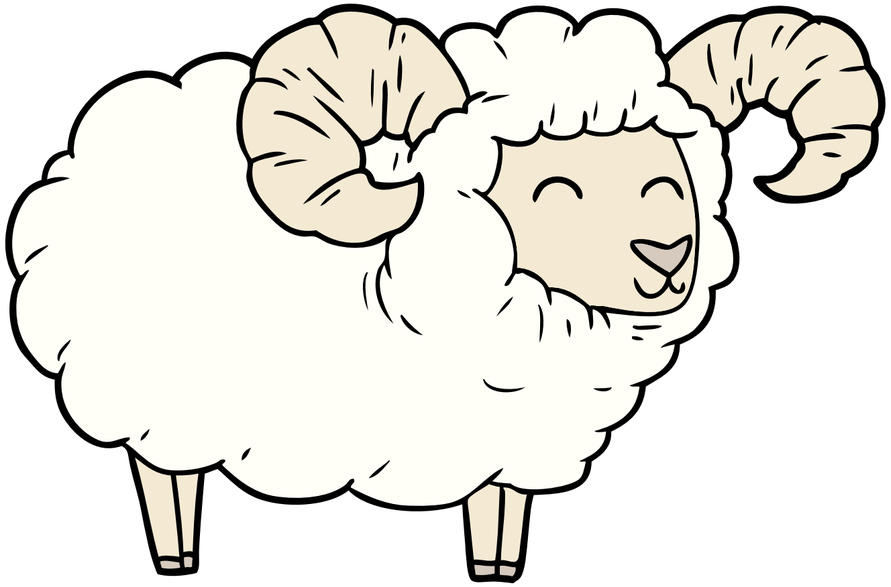
So a coloured image can be represented as a 3D array of numbers.




Feature
map 1
map 2
Filters
Convolutional
layer 2
Map 1
Map 2
Convolutional
layer 1
Input layer
Channels
Red
Green
Blue

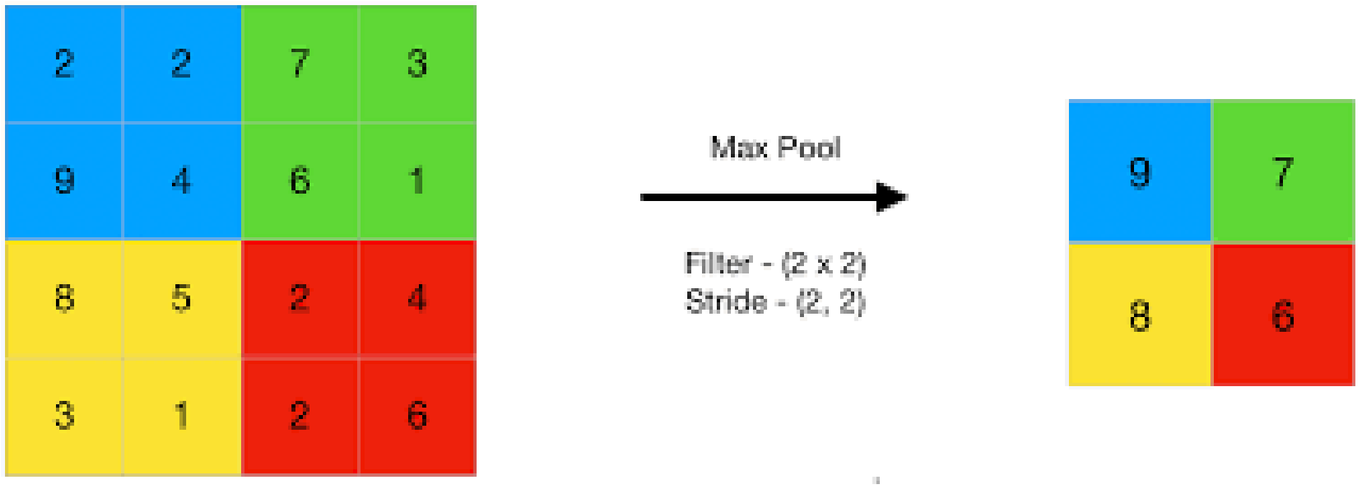
MEMORY REQUIREMENTS

Another problem with CNNs is that the convolutional layers require a huge amount of RAM.



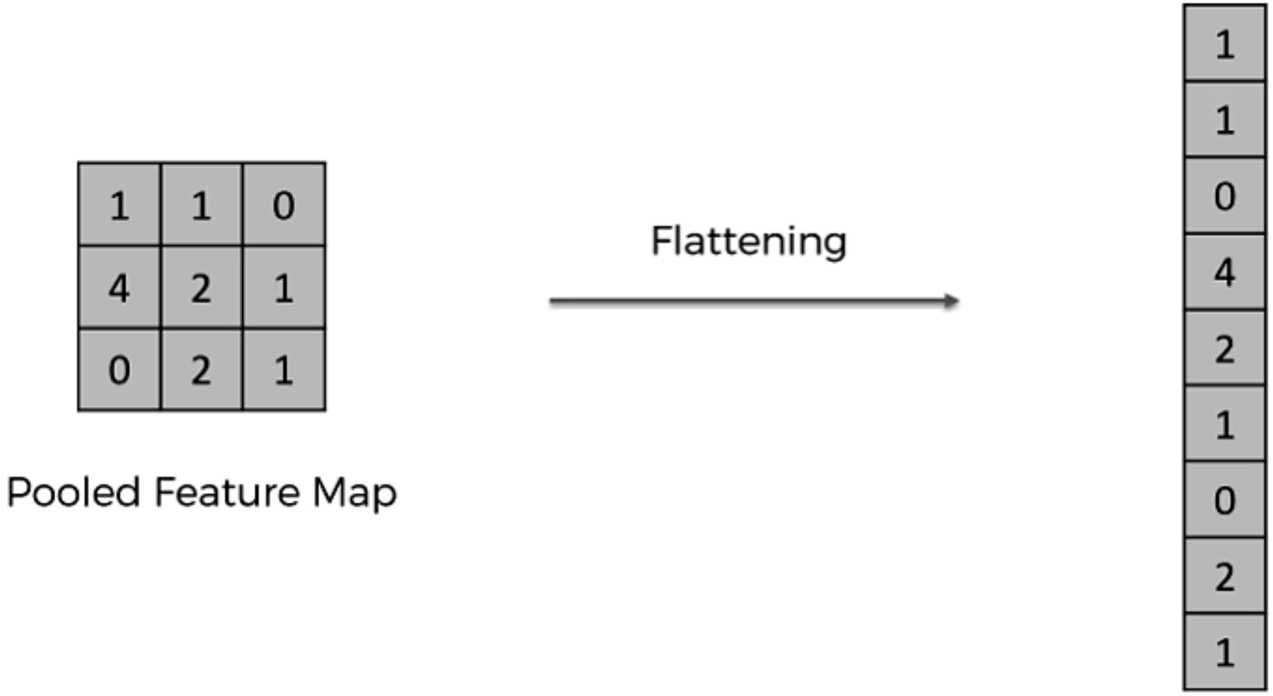
POOLING LAYERS

Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.



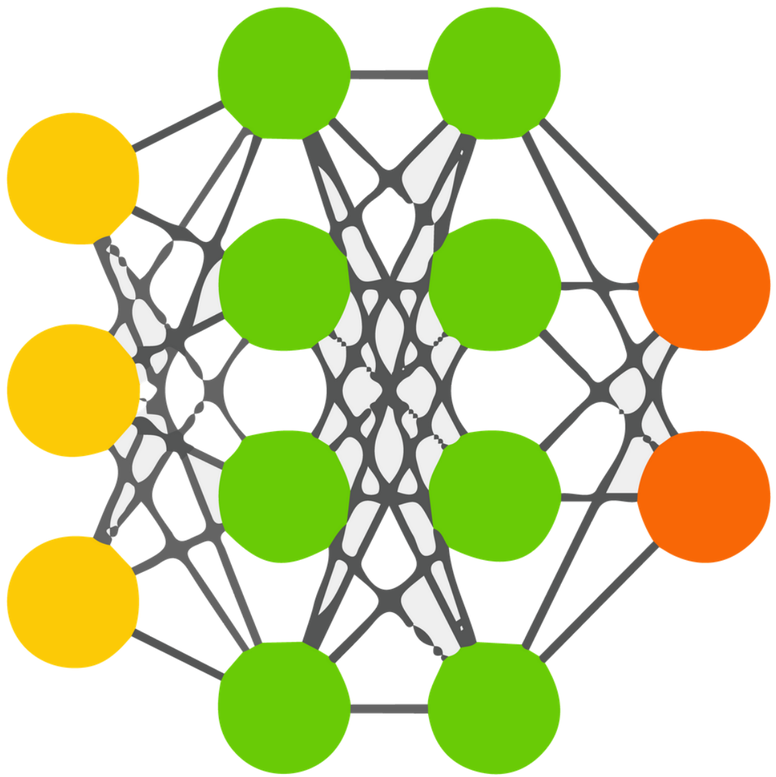
FLATTEN LAYERS

A flatten layer collapses the spatial dimensions of the input into the channel dimension

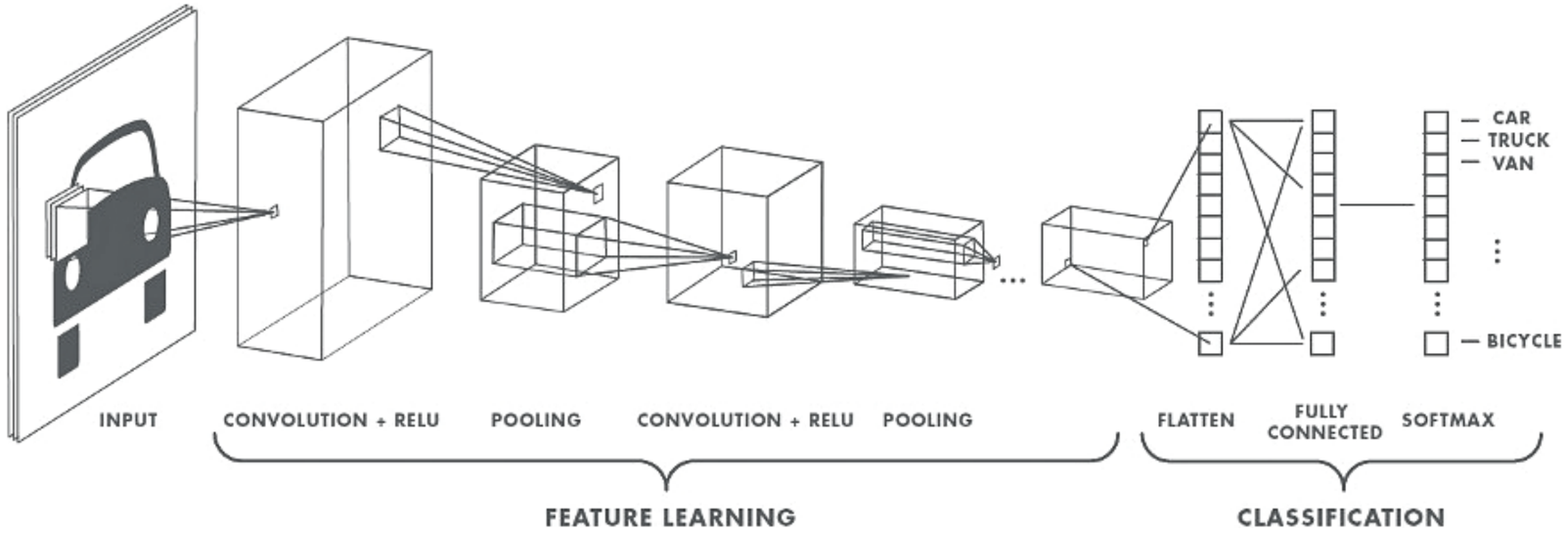


DENSE LAYERS

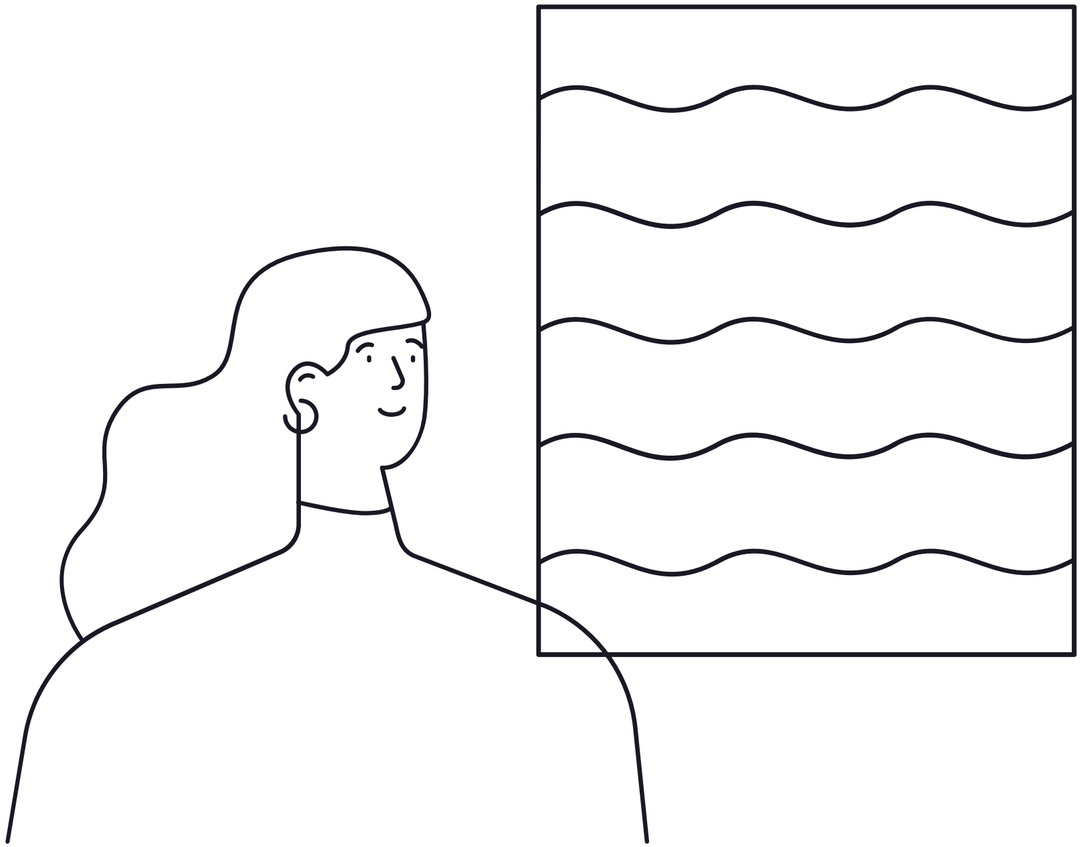
A dense layer is just a regular layer of neurons in a neural network. Each neuron recieves input from all the neurons in the previous layer, thus densely connected.



A CNN ARCHITECTURE



WHY CNNs AND NOT ANNs FOR IMAGE CLASSIFICATION



THANKS

SHAPEAI