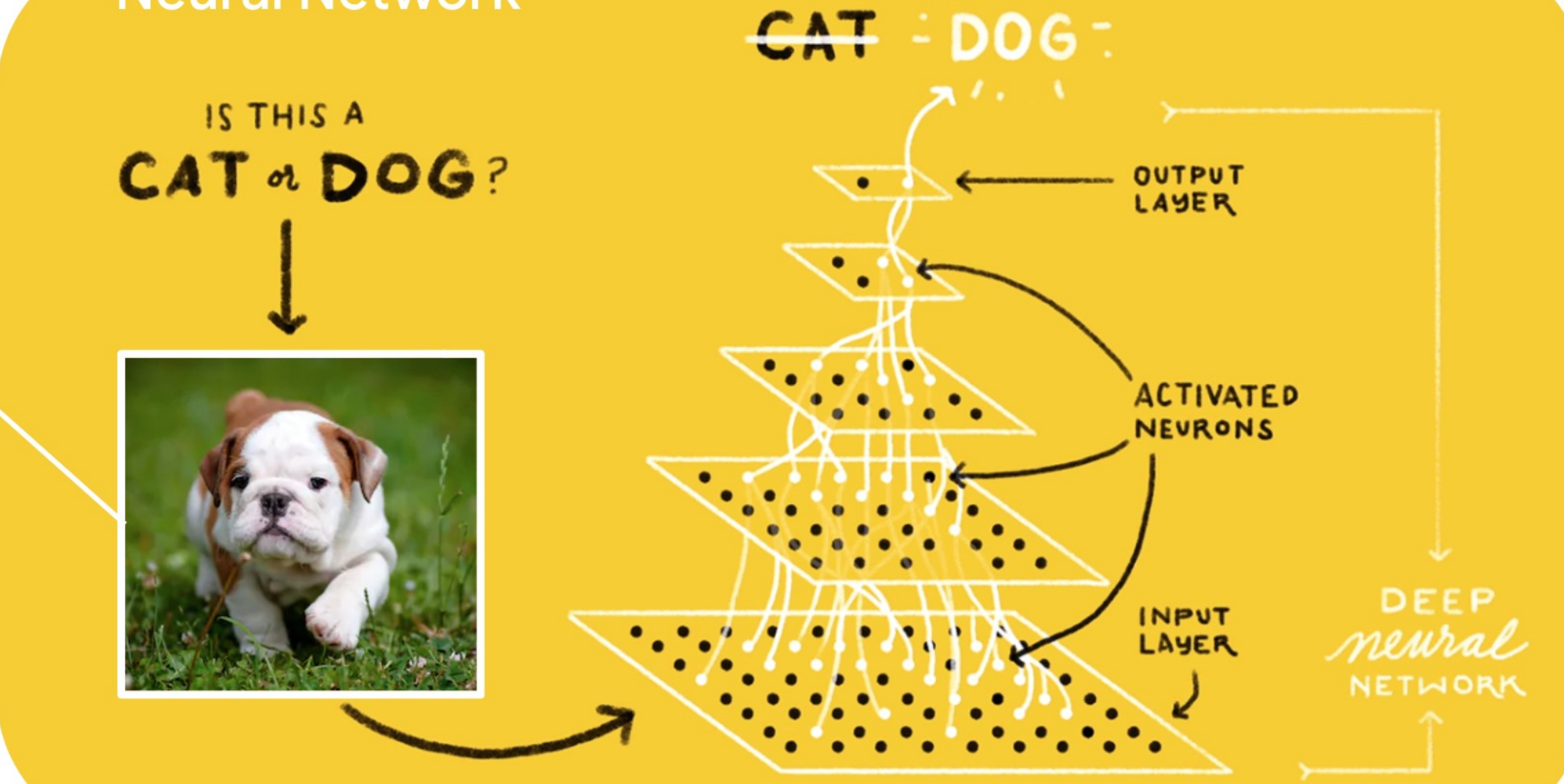
AIcademy

Deep Neural Networks and Convolutional Neural Networks

Neural Network



From google's tensorflow Manual

Training with 1000’s of images

Inputs:  

We show the network inputs with desired outputs and adjust weights w.

Outputs: Dog Cat

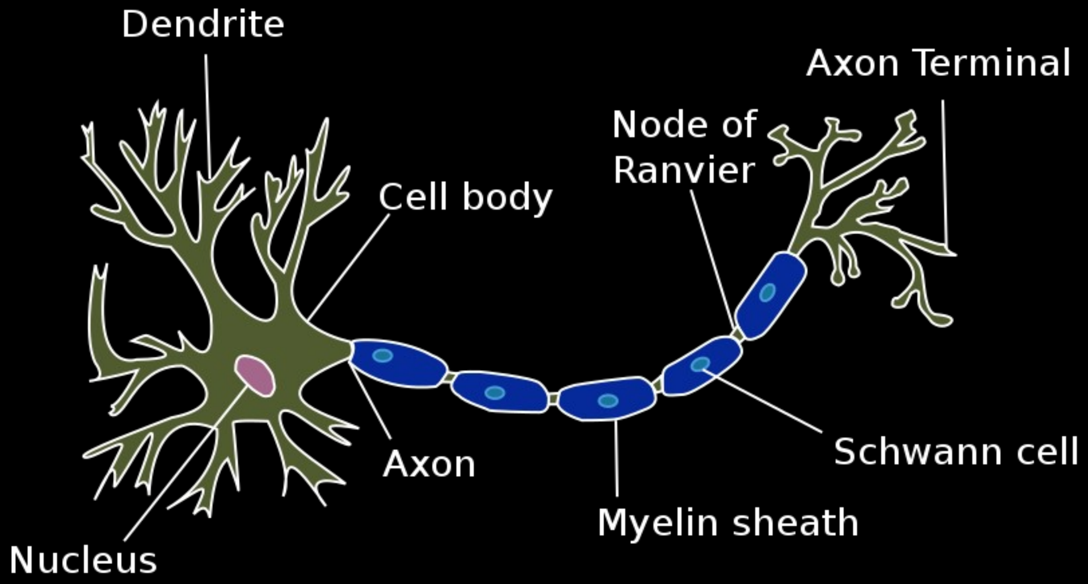
Builds: network(Inputs) → Outputs

Neural Network

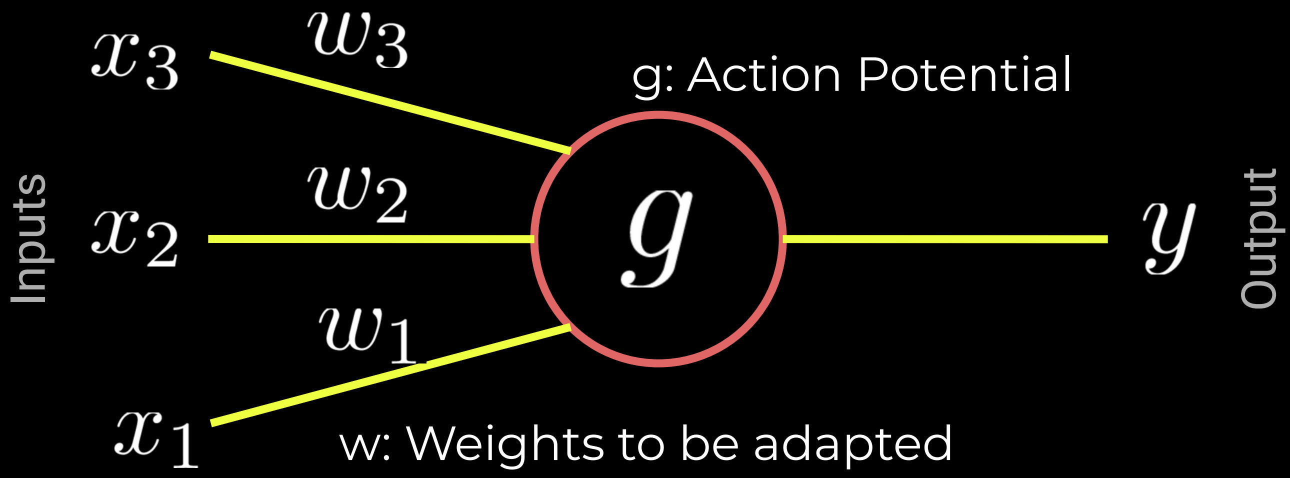
Machine Learning mechanism inspired by the brain.

A real neuron:

Connections and gating potentials are ought to encode data in the Brain



Artificial Neuron



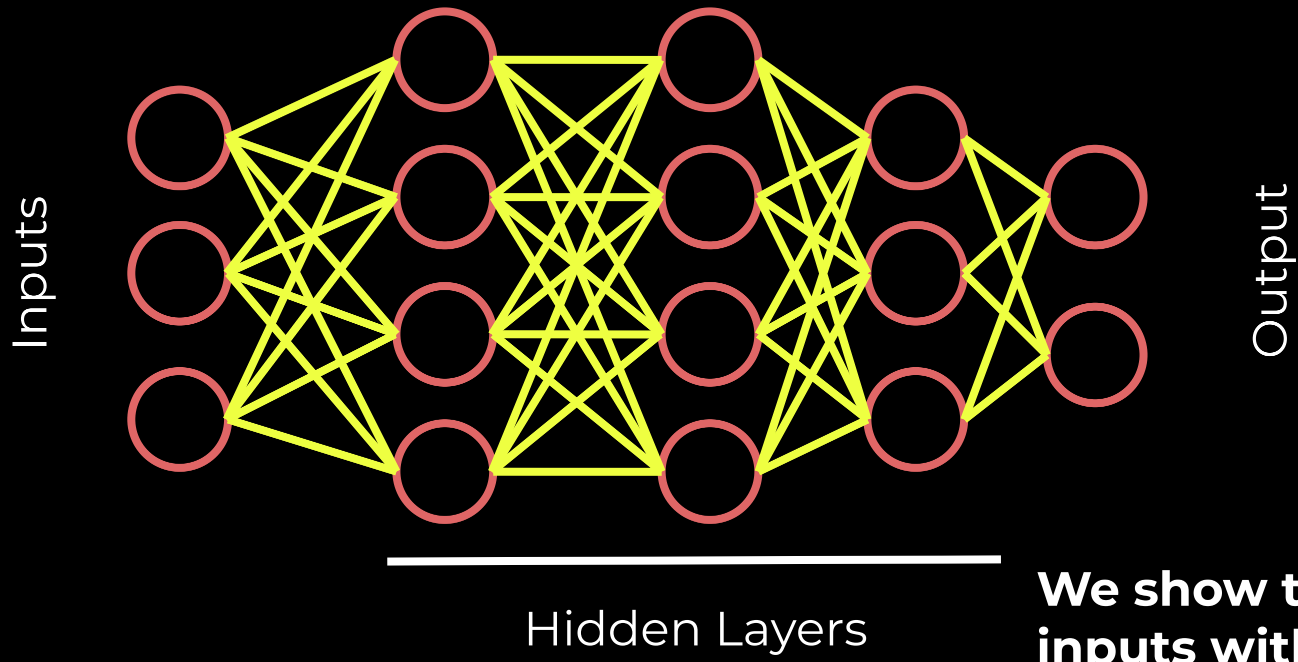
$$y = g\left(\sum_i x_i w_i\right)$$

w: weights
x: inputs

Artificial Brain / Deep Learning

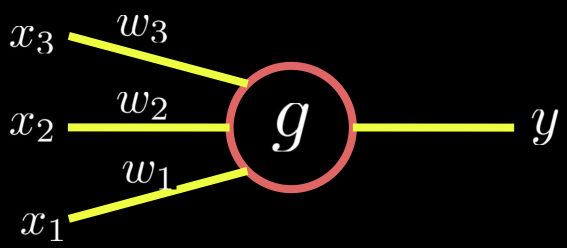
Each red node fires according to the sum of the inputs and the action potential

On each yellow edge there is a weight that we have to tune



We show the network inputs with desired outputs and adjust weights w

Options for g



$$g(z) = \max(0, z)$$

Rectified Linear Unit (ReLU)

$$g(z) = \frac{e^{x} - e^{-z}}{e^{x} + e^{-z}}$$

Hyperbolic Tangent

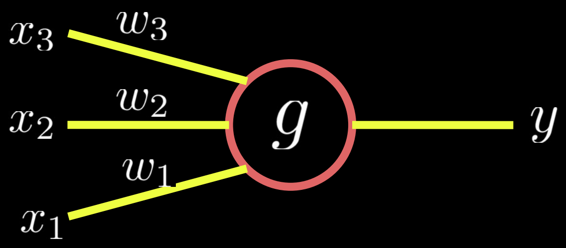
$$g(z) = \frac{1}{1 + e^{-z}}$$

Logistic Function

And many many more....

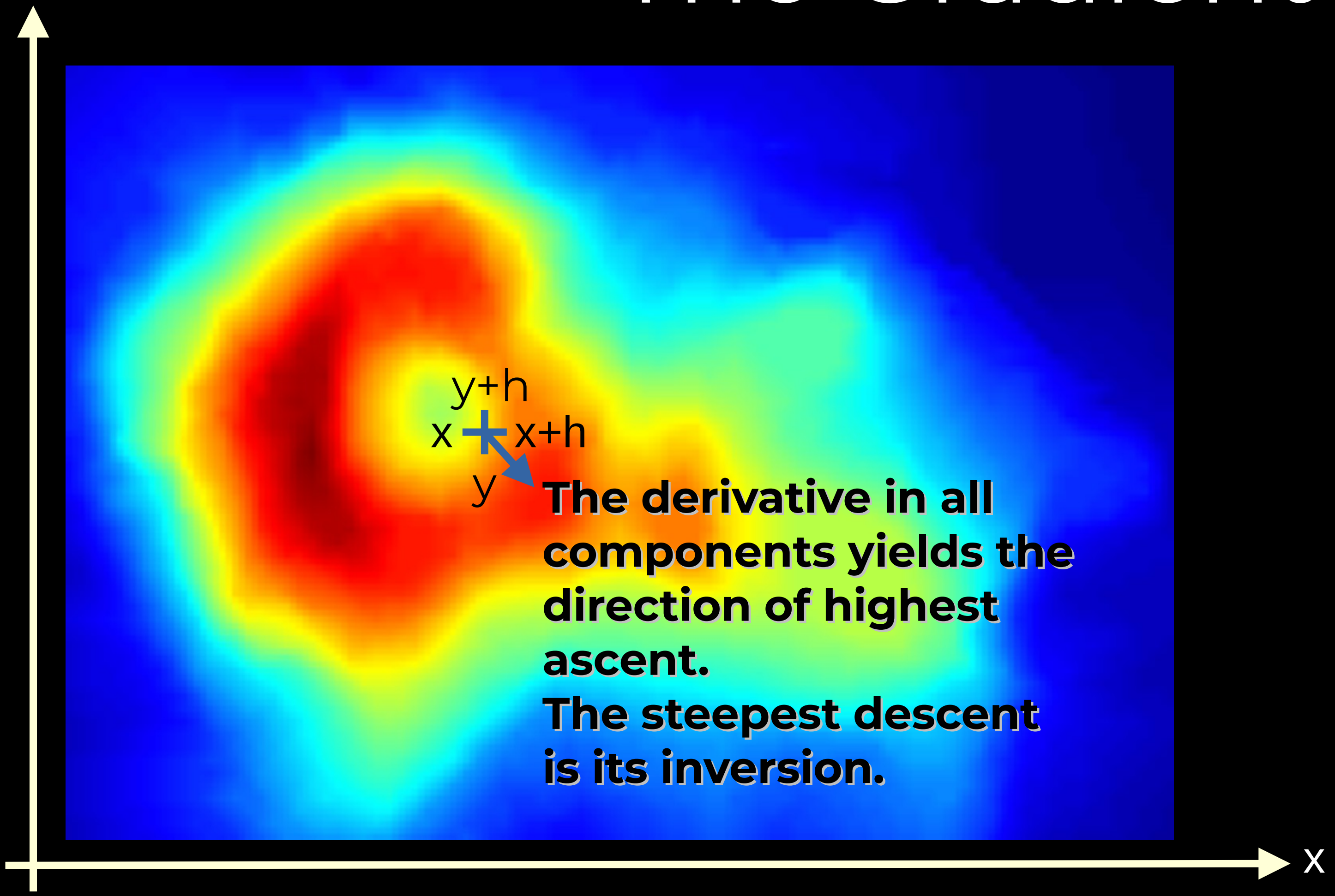
How to adjust the weights ?

- Using the Gradient Descent Algorithm.
- Implemented as “Back Propagation”.



- Loss function:
- L(w) = distance [outputs – desired outputs]
- We want to **minimize L, which depends on the weights, by varying the weights** for each image shown.

The Gradient

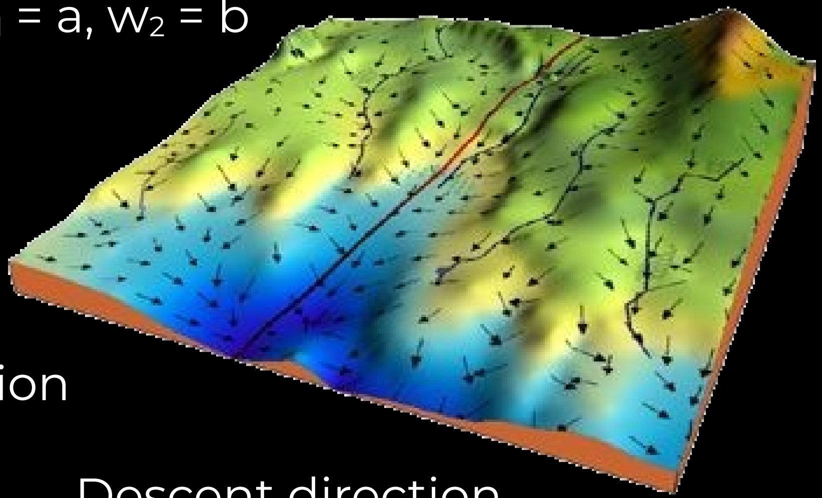


The derivative in all components yields the direction of highest ascent.
The steepest descent is its inversion.

Gradient Descent

$$\nabla_w L(w) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix}$$

Gradient of the Loss function according to model parameters: i.e. $w_1 = a$, $w_2 = b$



Gradient results
in the direction
of largest, local,
ascent;
descent by inversion

Vector Equation
all model parameters
updated simultaneously

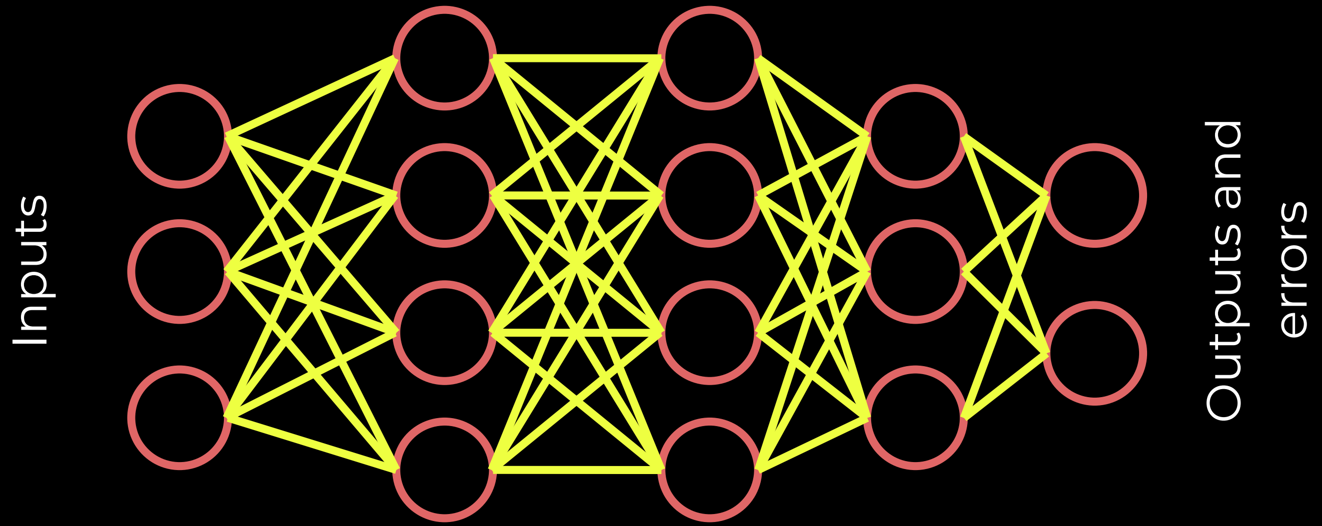
How far to move
(Learning rate)

Descent direction
with steepest slope in general
normalized

$$w_{s+1} = w_s - \gamma \nabla_w L(w_s)$$

From the composition follows back propagation

The error in each stage of the network is a composition of the errors in the previous stage.



Due to the chain rule and the composition we can “backpropagate” the errors by multiplication of derivatives at each stage of the network

Composition of the Network

$$f(w_i) = g_1(g_2(g_3(g_4(\dots g_n(w_i)\dots)))))$$

$$\frac{\partial f(w_i)}{\partial w_i} = \prod_{j=1}^{n} \frac{\partial g_j}{\partial w_i}$$

$$L(w) = d(f_{w_i}, y)$$

$$\frac{\partial L(w_i)}{\partial w_i} = \frac{\partial L}{\partial w_i} \prod_{j=1}^{n} \frac{\partial g_j}{\partial w_i}$$

The network output is a composition of the different layers

The derivative is the product of the derivatives of the different
layers.

The Loss function is describing how far we miss the expected value y

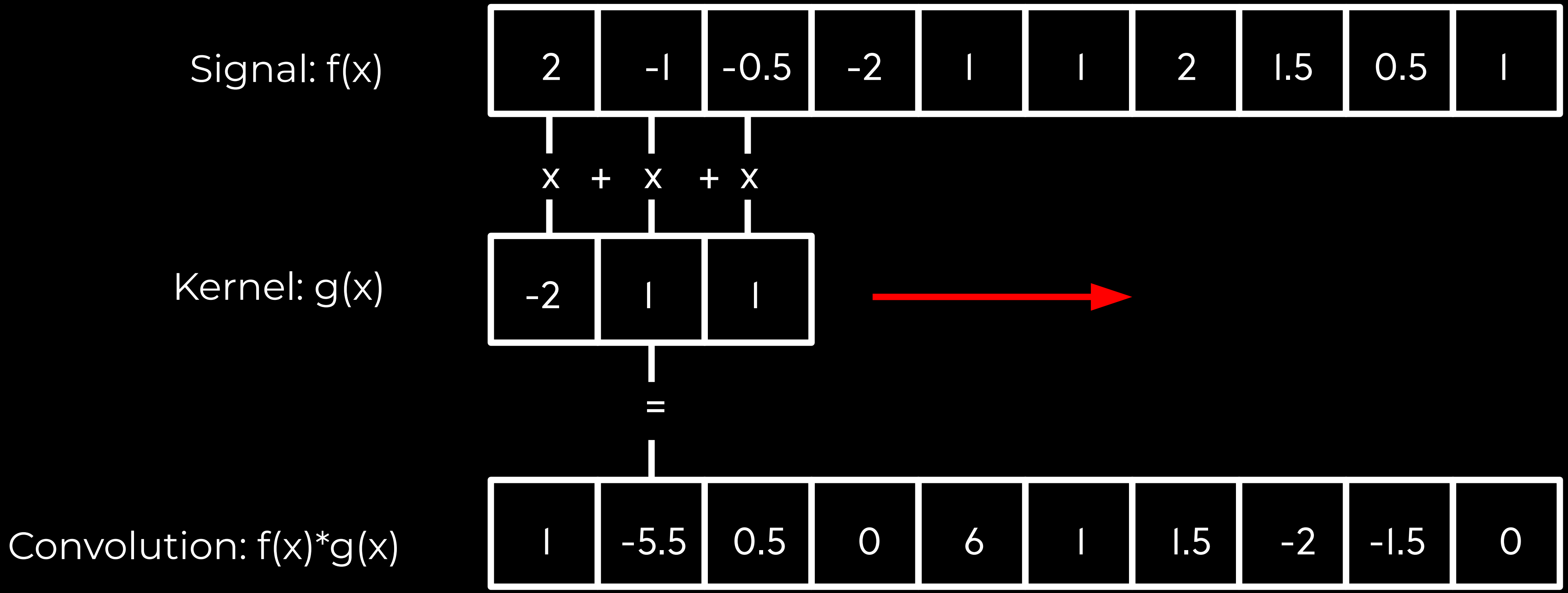
Convolutional Neural Networks

Make a Computer See

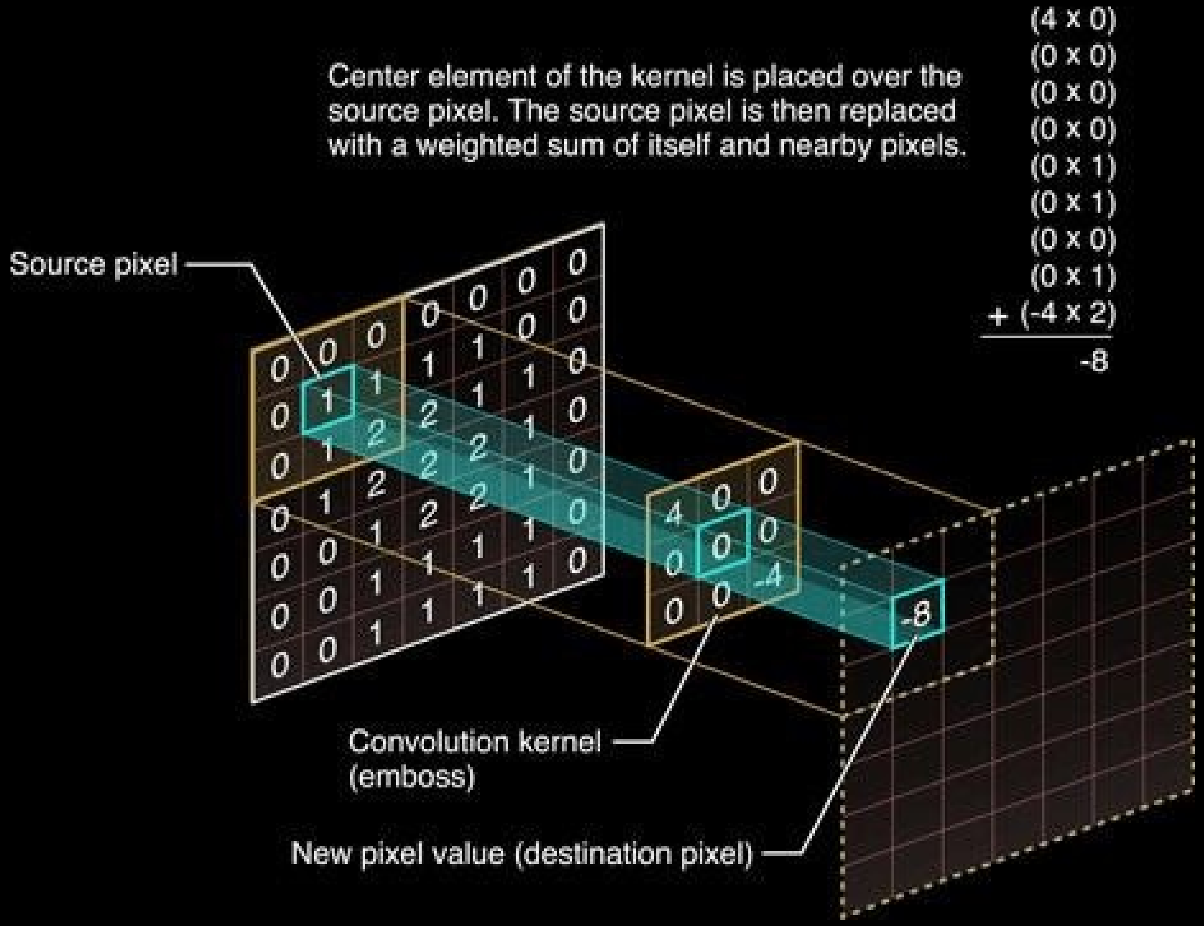
Networks and Convolution

- Convolutional networks were the key to highly efficient image classification / compression / generation using neural networks
- They are supposed to be closely mimicking the visual cortex.

What is a Convolution



In 2D and for Images



Application: Image Filter

Kernel: g(i,j)

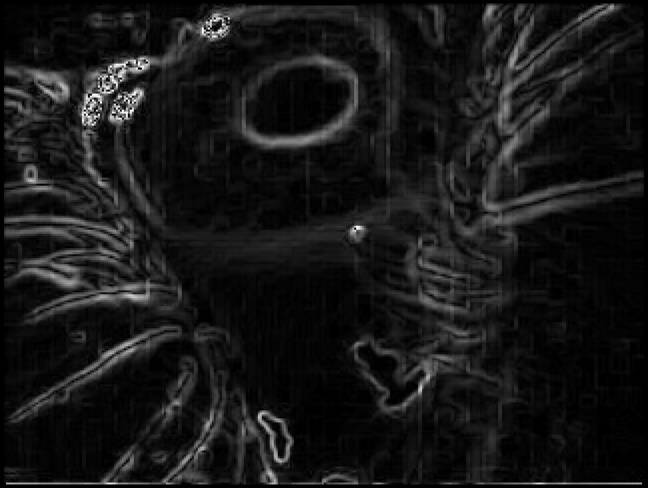
-1	-2	-1
0	0	0
1	2	1

Kernel: h(i,j)

-1	0	1
-2	0	2
-1	0	1

$$I[i,j] = \sqrt{(f*g)^2[i,j] + (f*h)^2[i,j]}$$

Sobel Edge Detector



Application: Neural Network

Input: Image: f(x)

2	-1	-0.5	-2	1	1	2	1.5	0.5	1

Kernel: g(x)

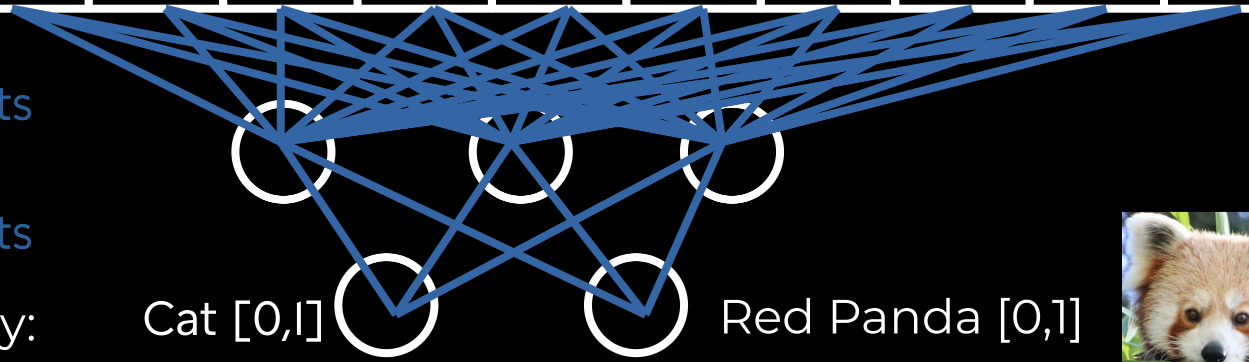
1	0	-1

Train Kernel

Convolution: f(x)*g(x)

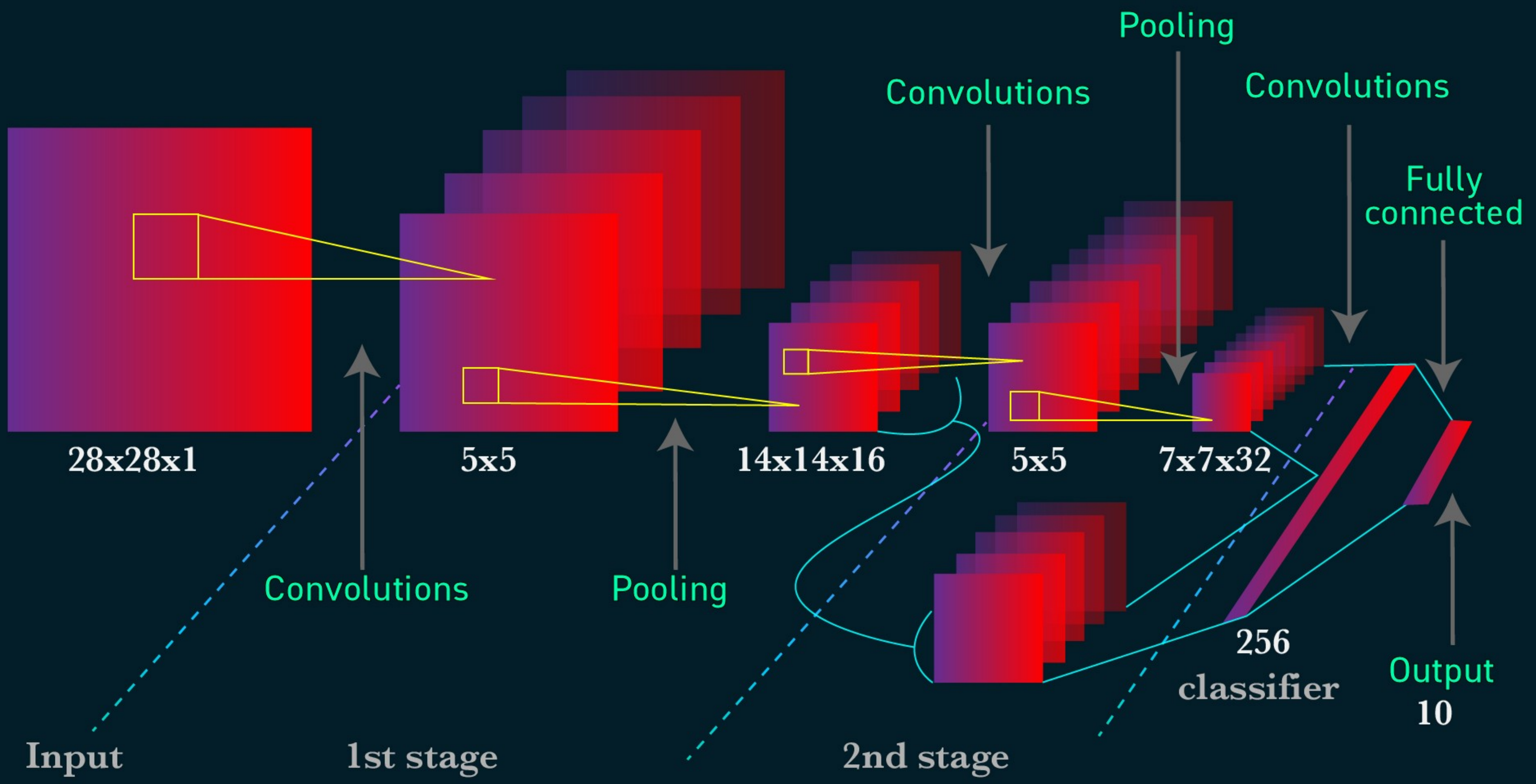
1	-1.5	1	-1.5	-3	-1	-0.5	1.5	0.5	0.5

Train Weights



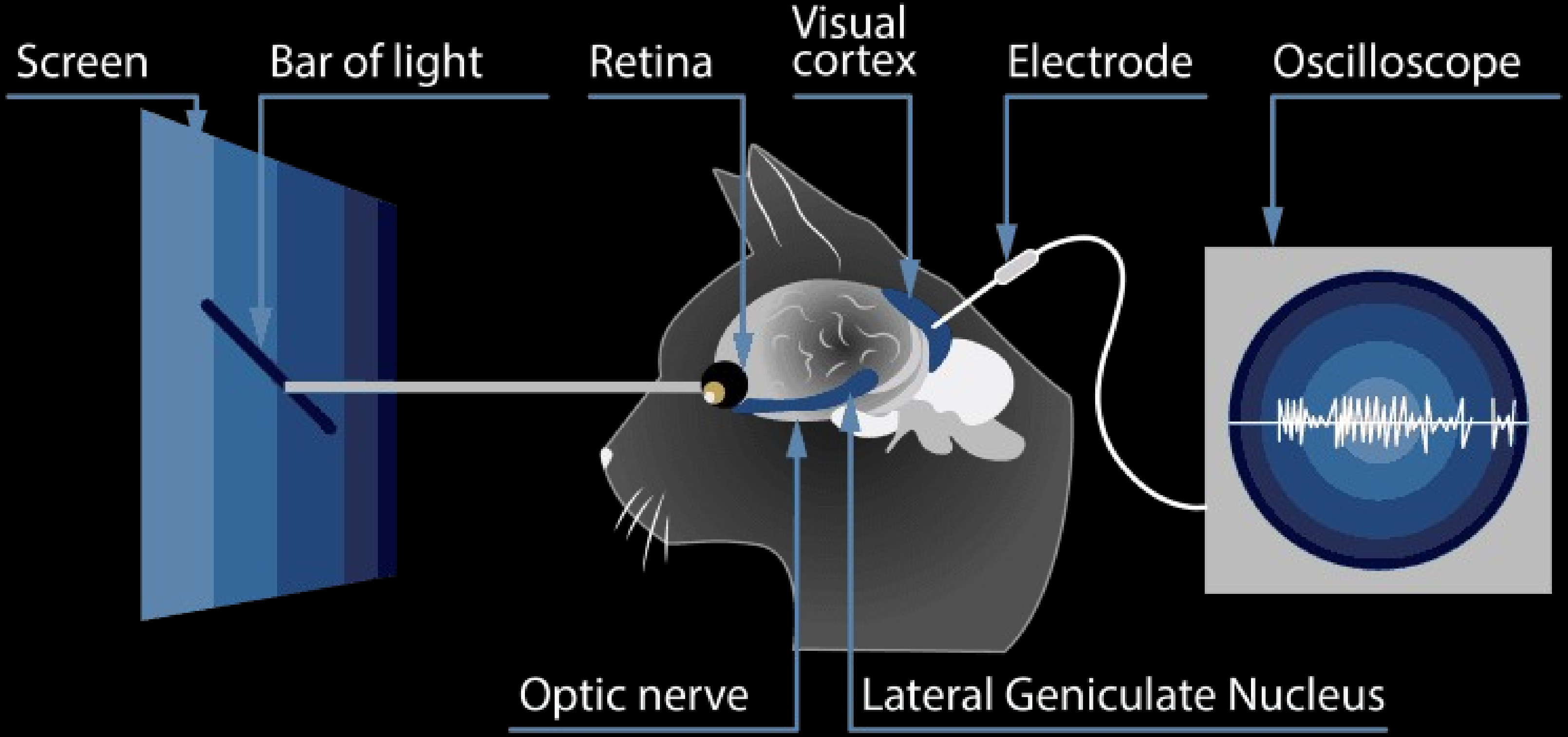
Train Weights

Output: Probability: Cat [0,1] Red Panda [0,1]



Pooling
Convolutions
Convolutions
Fully connected
28x28x1
5x5
14x14x16
5x5
7x7x32
Convolutions
Pooling
256
classifier
Output
10
Input
1st stage
2nd stage

The Famous Cat Experiment



Neurophysiologists David Hubel and Torsten Wiesel showed that different patterns make different neurons in the cat brain fire [in 1964].
Patterns hint for convolution like mechanisms in the visual cortex of the cat.


Nobel Prize 1981

Practical

- Let us build a Neural Network that detects hand written digits.
- MINST Dataset 60000 28x28 pixel images of handwritten digits from 0 to 9.

We will do this together

- With Tensorflow
- In google colab
- In python