



## **#5 Neural Networks**

---

**AI Training**

# Neural Networks

# Neural Networks (overview)

Many tasks that involve intelligence, pattern recognition, and object detection are *extremely difficult to automate*, yet *seem to be performed easily and naturally* by animals and young children.

Have you ever wondered how your brain recognizes images?

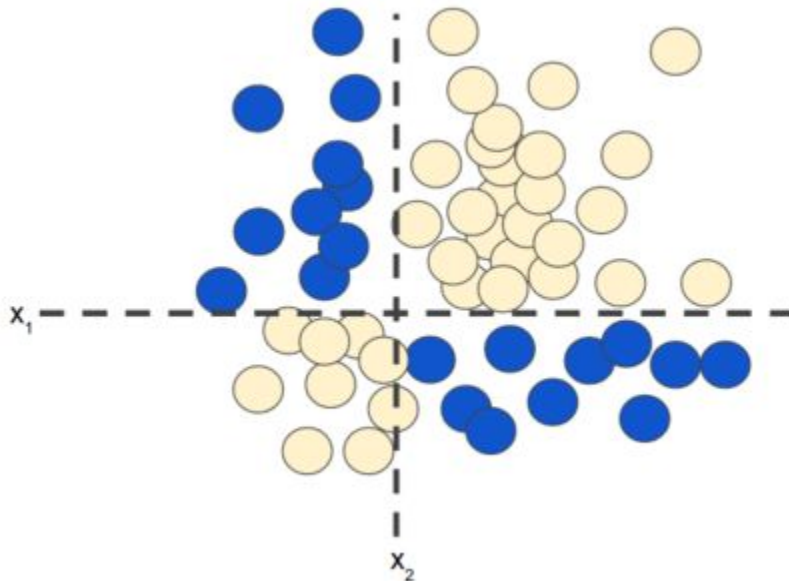
- No matter what or how the image looks, the brain can tell that this is an image of a cat and not a dog.
- The brain relates to the best possible pattern and concludes the result.

# **Why Neural Networks?**

# Neural Networks: Why?

If you recall, the following classification problem is **nonlinear**:

"**Nonlinear**" means that you can't accurately predict a label with a model of the form  $y = w_0 + w_1x_1 + w_2x_2$ . In other words, the "decision surface" is not a line.



# Neural Networks: Why?

Now consider the following data set:



The data set shown in the figure can't be solved with a linear model.

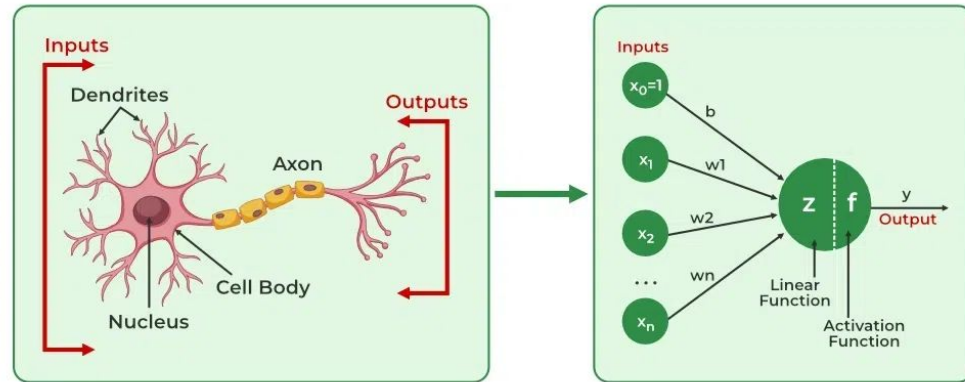
# How Neural Networks work

# Neural Networks

A neuron firing is a binary operation — the neuron either fires or it doesn't fire.

- Each neuron receives electrochemical inputs from other neurons at their dendrites.
- If these electrical inputs are sufficiently powerful to activate the neuron, then the activated neuron transmits the signal along its axon, passing it along to the dendrites of other neurons.
- These attached neurons may also fire, thus continuing the process of passing the message along.

Simply put, a neuron will only fire if the total signal received at the soma exceeds a given threshold.



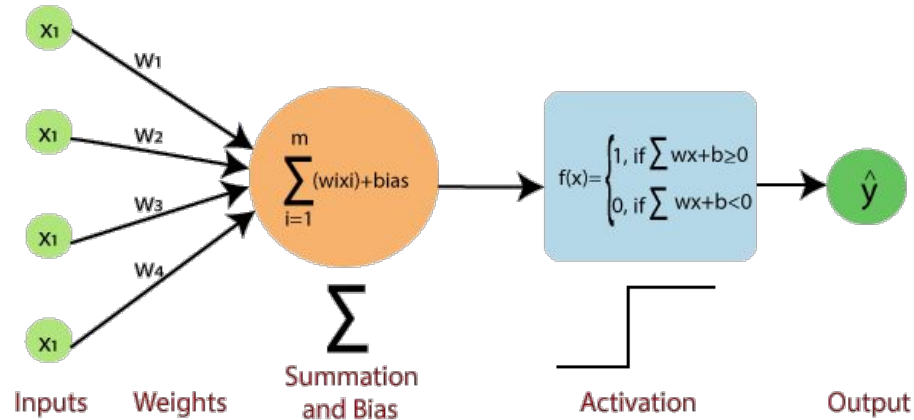


# Neural Networks

Neural Networks are complex systems with artificial neurons.

Artificial neurons or perceptron consist of:

- Input
- Weight
- Bias
- Activation Function
- Output



# Neural Networks: Topics

- Deep Learning
- Activation functions
- Gradient Descent
- Backpropagation
- Regularization
- PyTorch & Other Tools!

# Deep Learning

# Deep Learning

## Deep Learning



What society thinks I do



What my friends think I do



What other computer scientists think I do



What mathematicians think I do

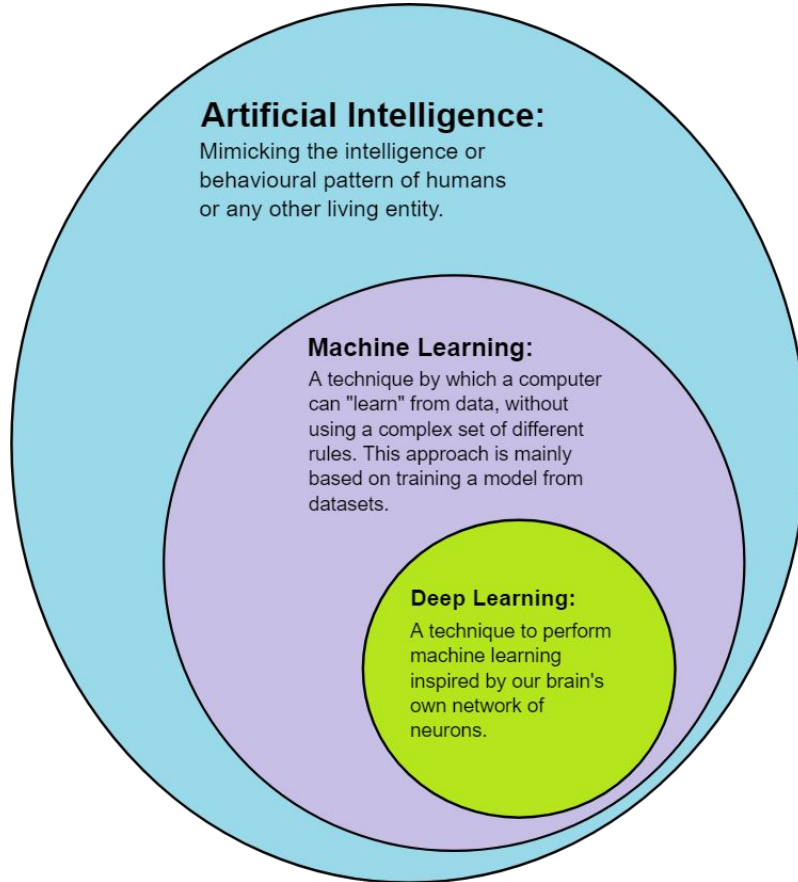


What I think I do

```
import keras  
Using TensorFlow backend.
```

What I actually do

# Machine Learning -> Deep Learning

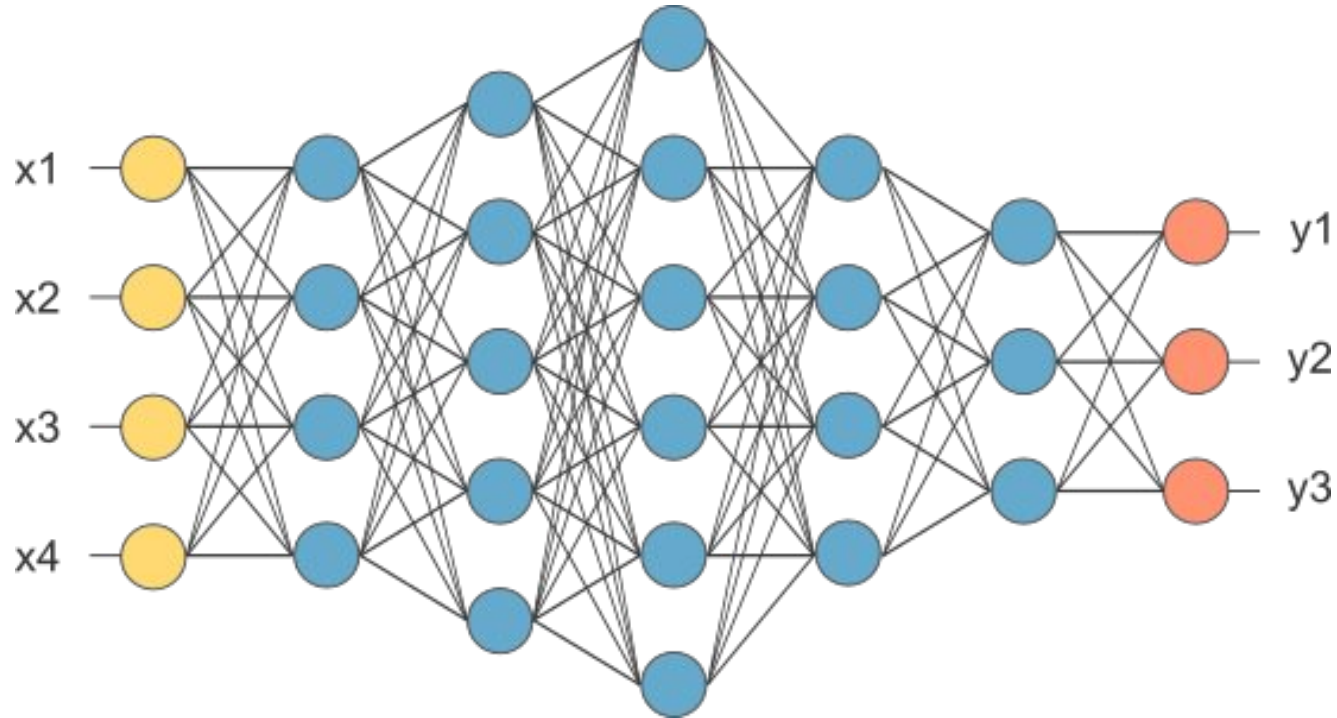


What you learn in Machine Learning  
**DOES** matter for Deep Learning!

# Deep Learning: The process



# Deep Learning: Neural Network (NN)



# Why is a Neural Network useful?

---



Our first goal for these neural networks, or models, is to achieve human-level accuracy. Until you get to that level, you always know you can do better.

Ivan Gomez • Data Scientist and Consultant • Zencos

---



# Deep Learning: Some Applications

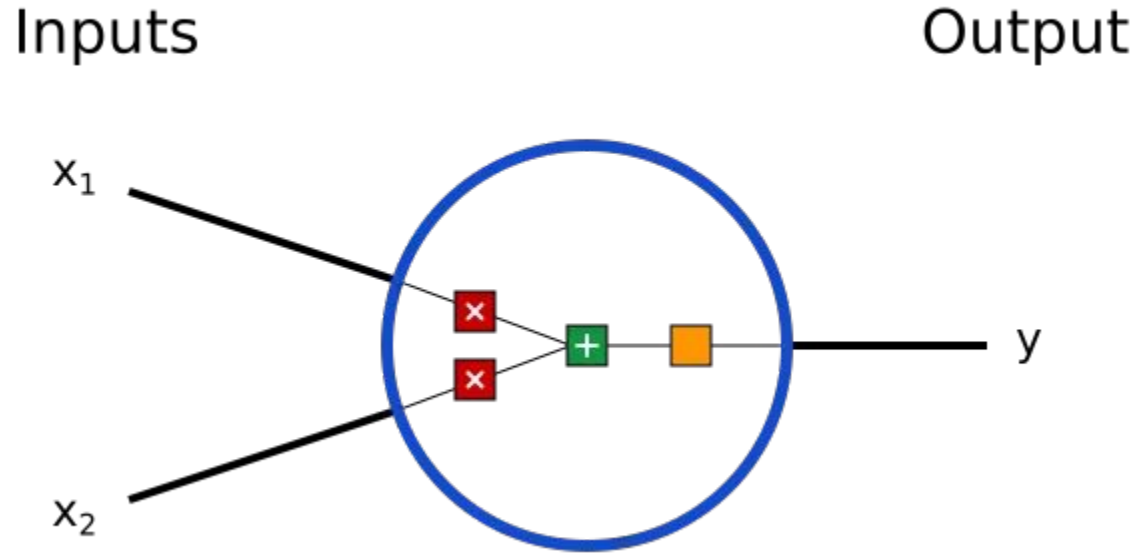
- Classification: do you have cancer or not?
- Speech recognition
- Natural Language Processing (NLP)
- Text recognition
- Text generation
- Image Recognition: Is It a Kitten or a Tiger?
- Particular example: Optical Character Recognition (OCR)
- Automatic translation
- Time series analysis
- Musical composition
- Video prediction

...

**What are the different components of a NN?**

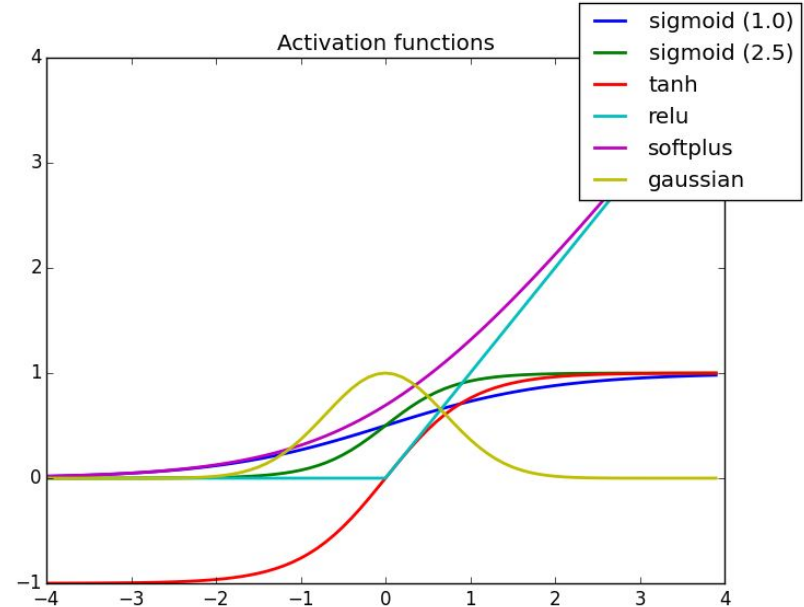
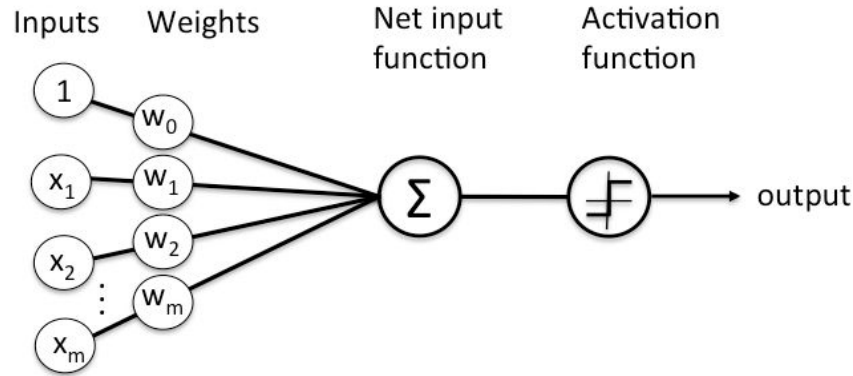
# Deep Learning: Perceptrons // Neuron

A single layer neural network



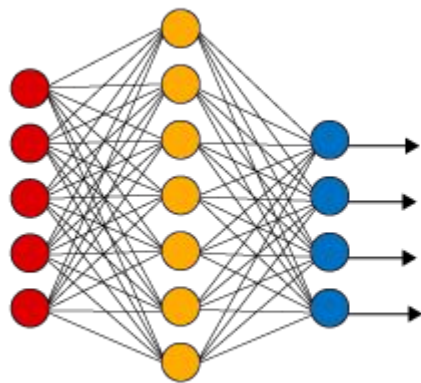
# Deep Learning: Perceptrons // Neuron

Number within each neuron: **Activation**

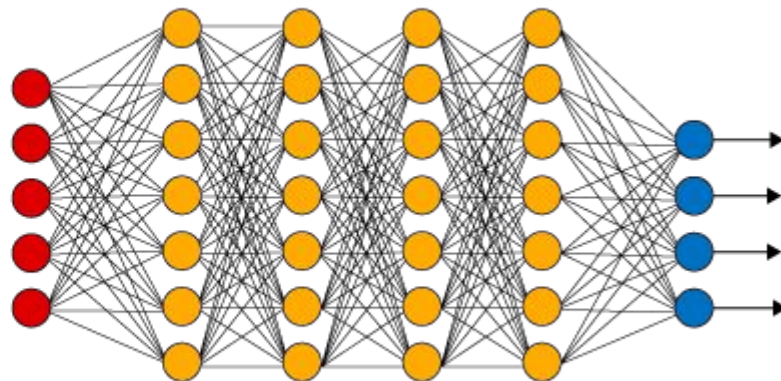


# Deep Learning

Simple Neural Network



Multilayer perceptron  
Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer



Neuron (activation  
function)



Weight

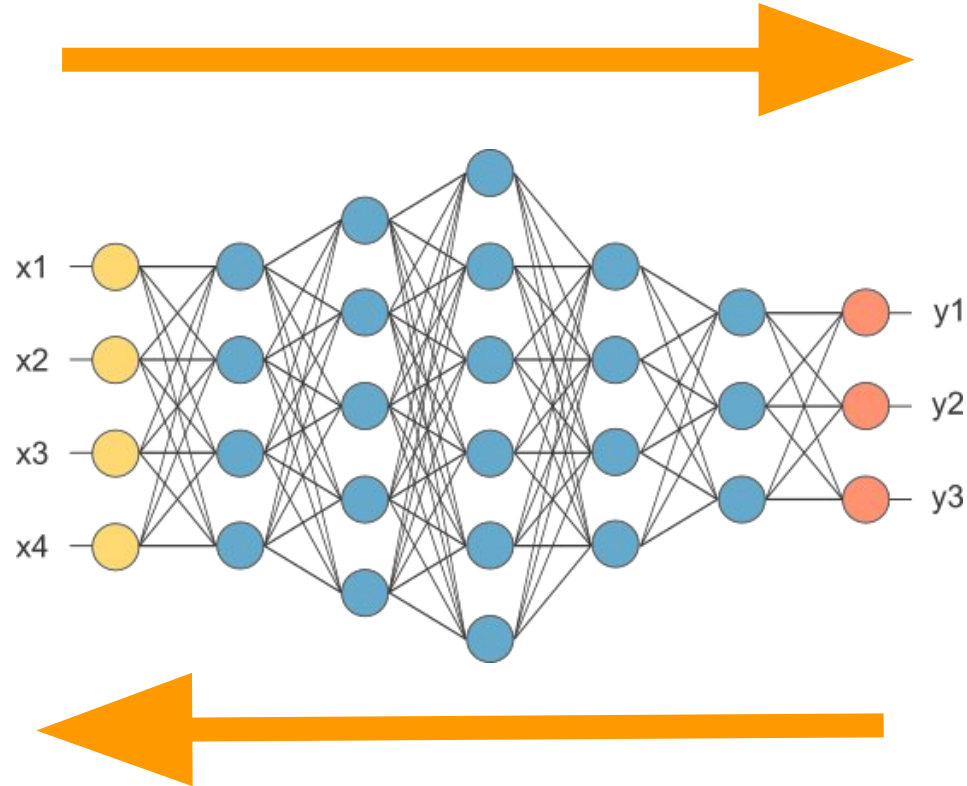
+ Bias

$$(y = ax + b)$$

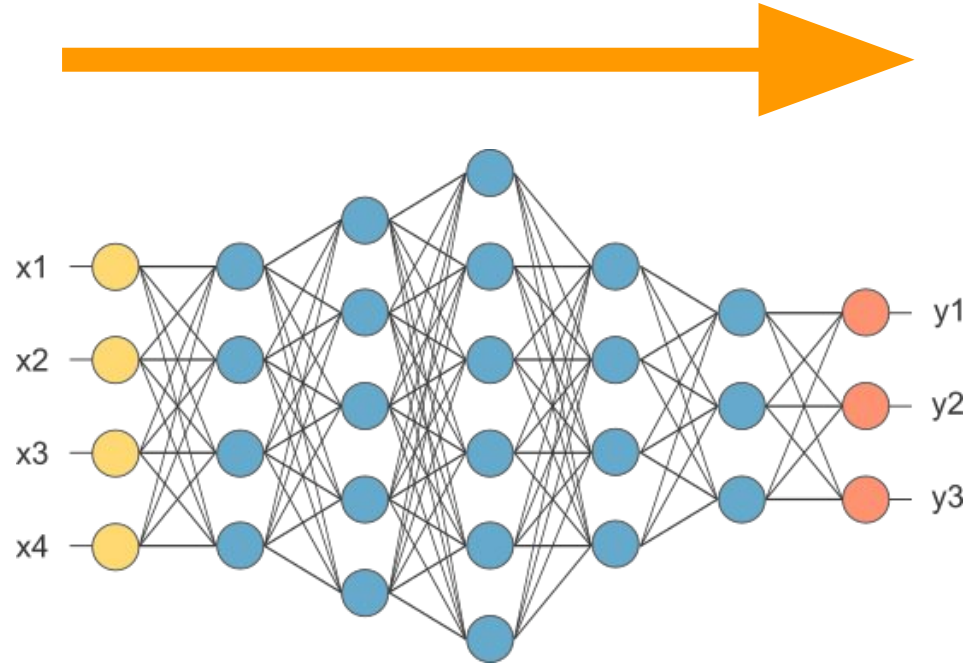
**How does it work?**

A solid orange horizontal bar spanning the width of the slide, positioned below the text.

# Deep Learning: Forward and Backward Propagation

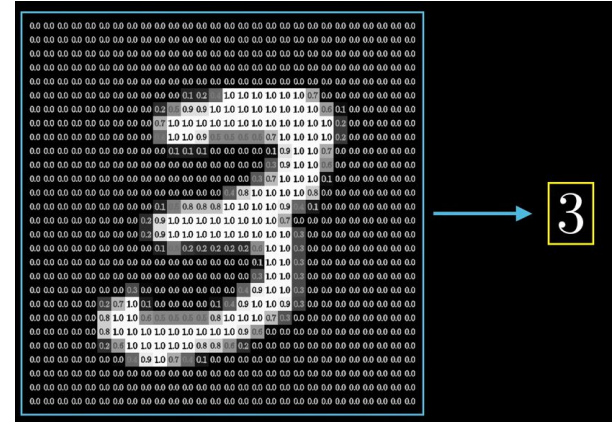
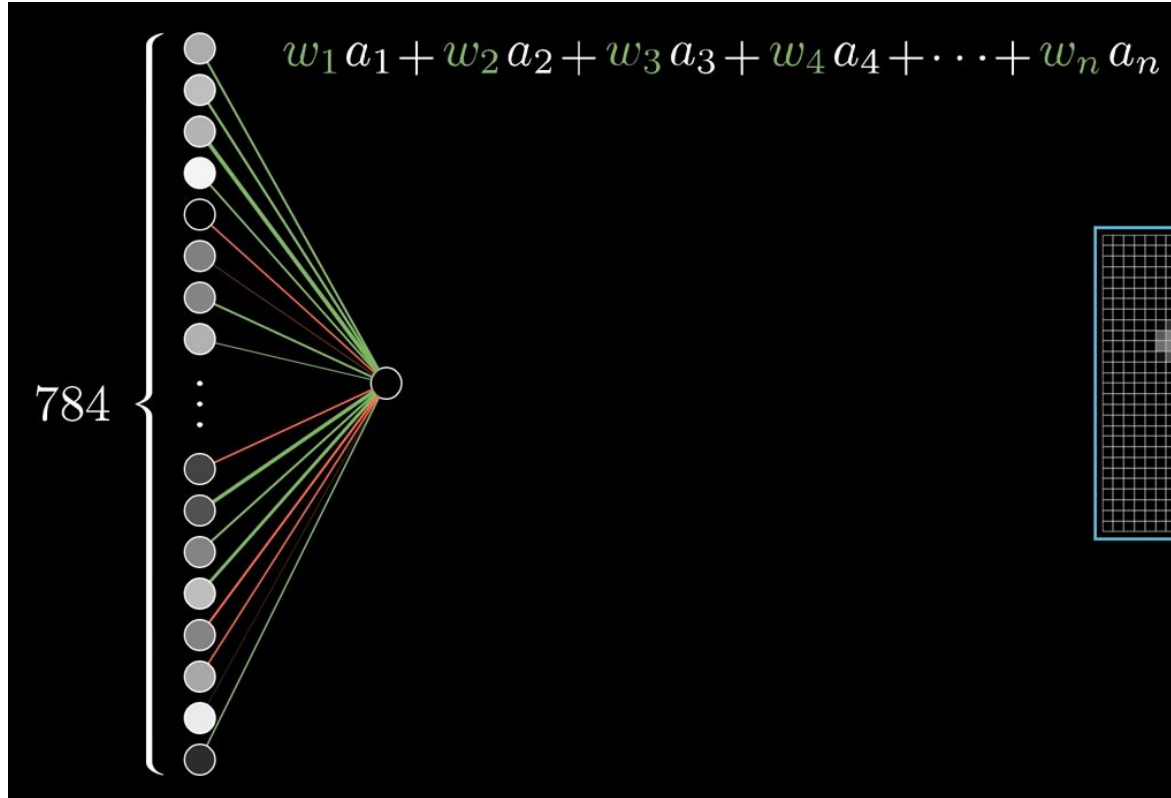


# Deep Learning: Forward Propagation (I)



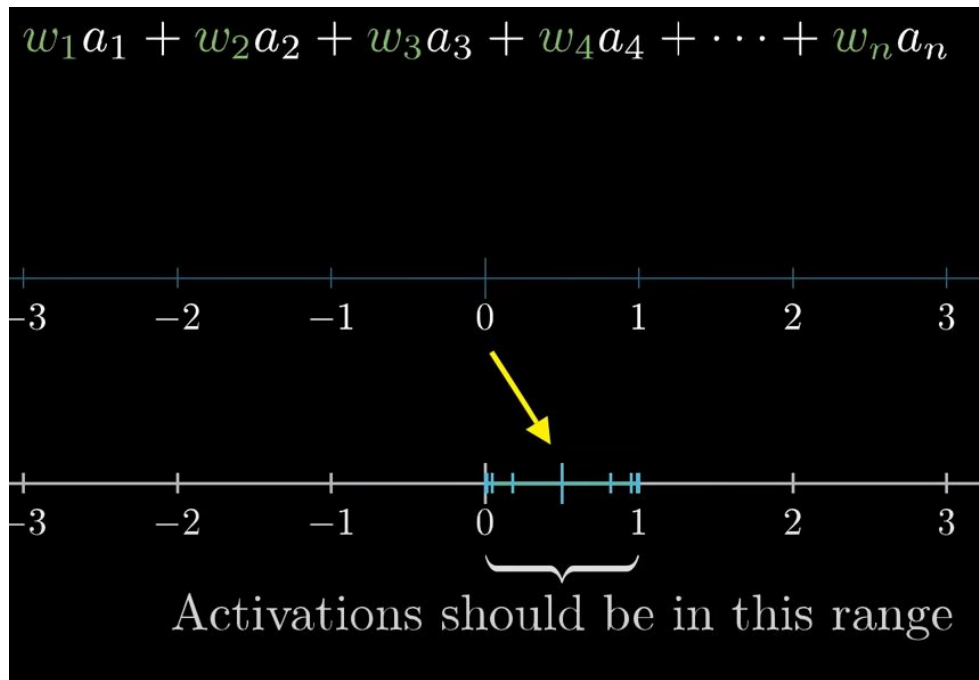


# Deep Learning: Forward Propagation (II)

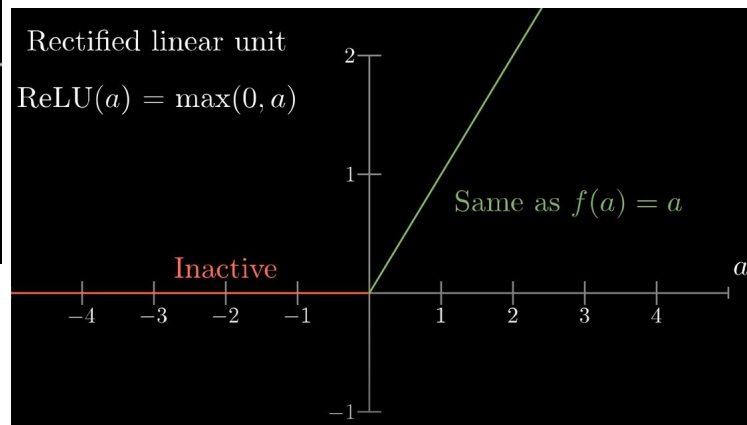
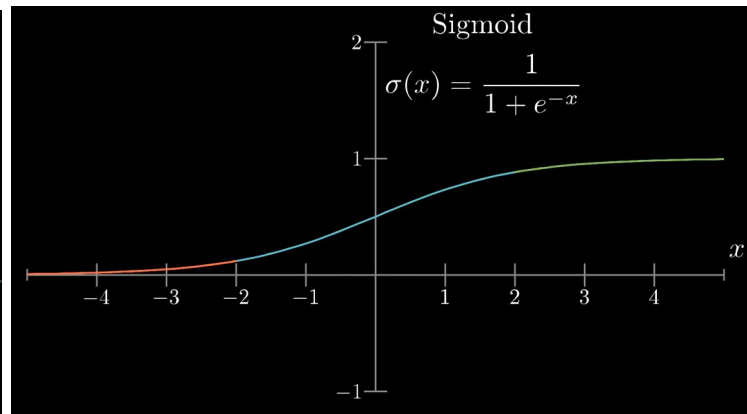


Source: 3Blue1Brown

# Deep Learning: Forward Propagation (III)



Source: 3Blue1Brown



# Deep Learning: Forward Propagation (IV)

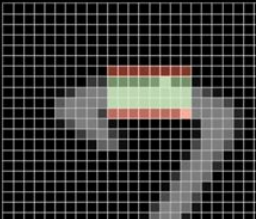
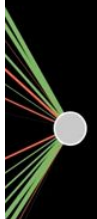
Sigmoid

How positive is this?

$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \dots + w_na_n \boxed{-10})$

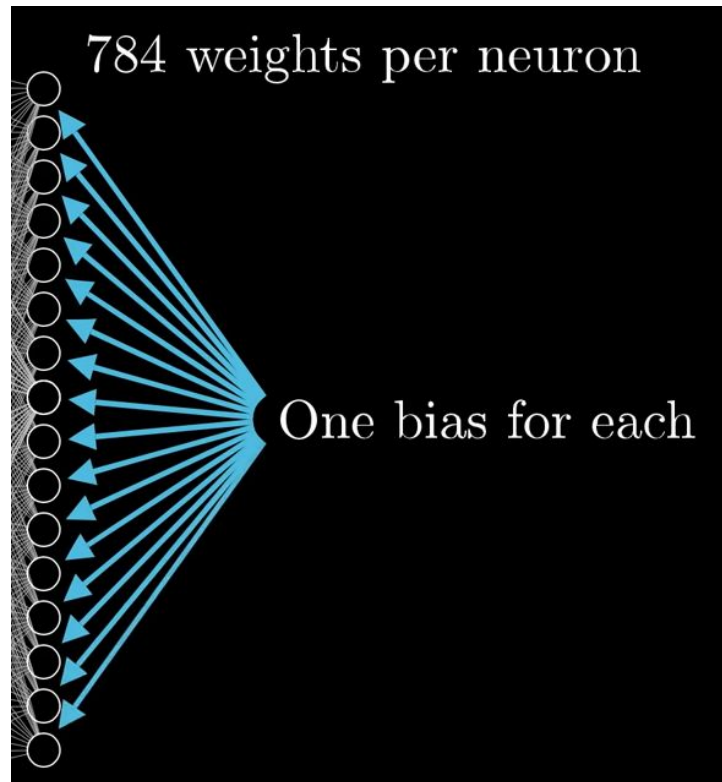
“bias”

Only activate meaningfully  
when **weighted sum**  $> 10$



Analogy:  
( $y = ax + b$ )

Source: 3Blue1Brown



# Deep Learning: Forward Propagation (V)

$$784 \times 16 + 16 \times 16 + 16 \times 10$$

weights

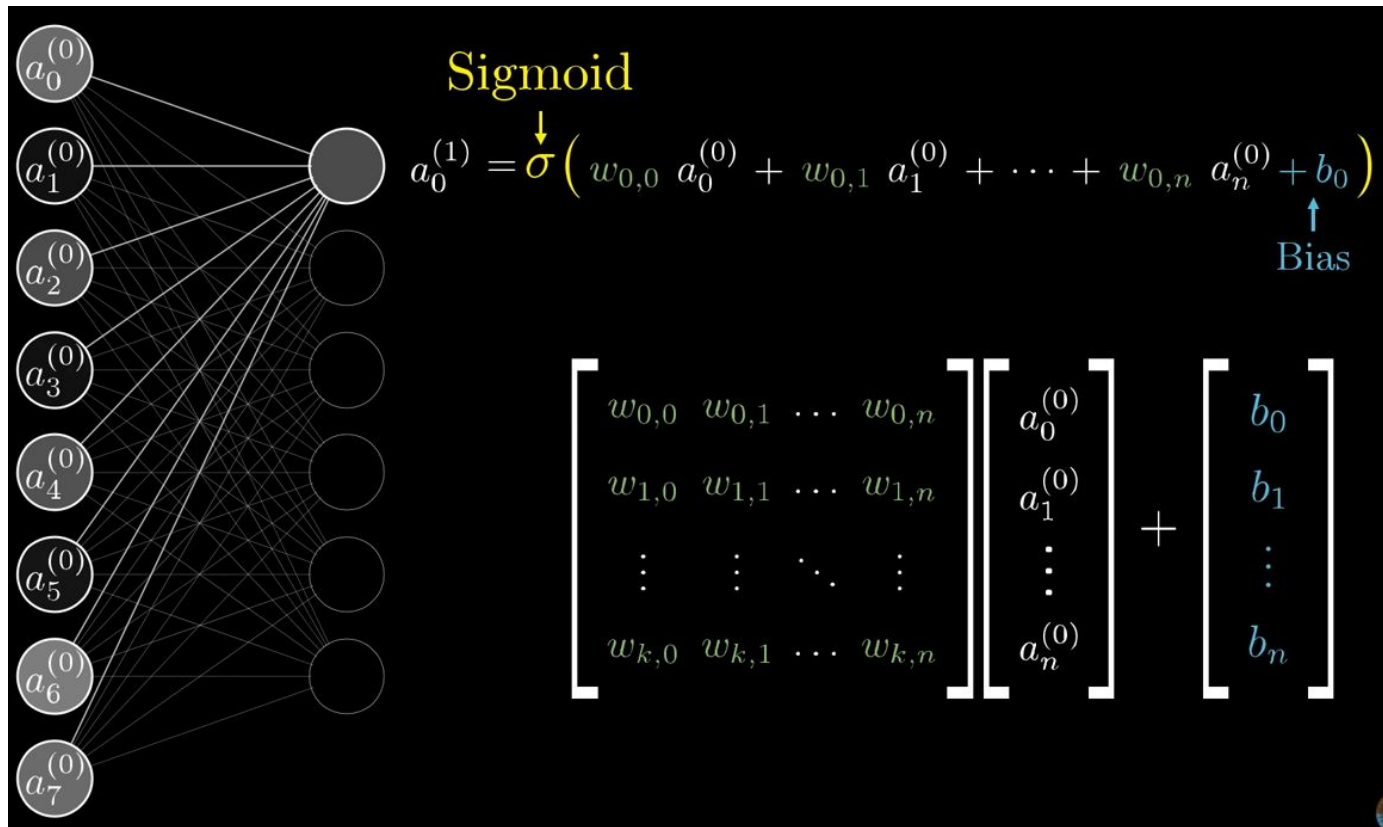
$$16 + 16 + 10$$

biases

13,002

Learning  $\rightarrow$  Finding the right  
weights and biases

# Deep Learning: Forward Propagation (VI)



# Deep Learning: Forward Propagation (VII)

$$\mathbf{a}^{(1)} = \sigma(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b})$$

$$\sigma \left( \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

Optimized  
calculation  
(Vectorization)

Source: 3Blue1Brown

# Neural networks: Nodes and hidden layers

*Forward Propagation*

<https://developers.google.com/machine-learning/crash-course/neural-networks/nodes-hidden-layers>

# Activation Function

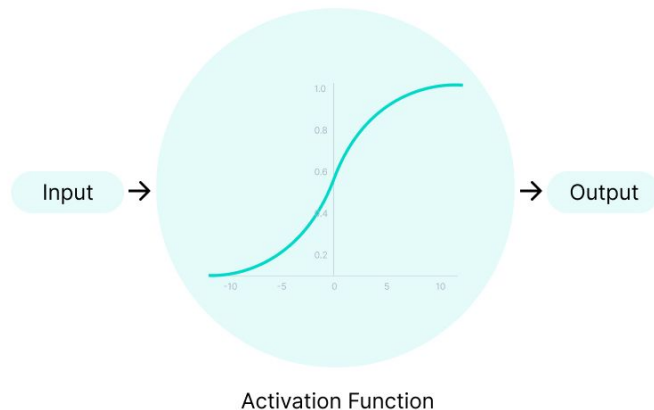


# Activation Function

An **Activation Function** decides whether a neuron should be activated or not.

This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

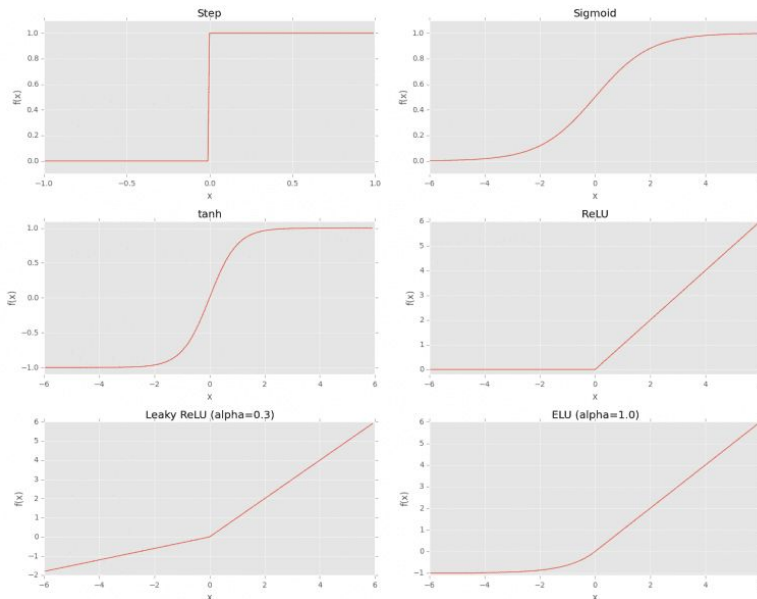
The purpose of an activation function is to add **non-linearity** to the neural network.



# Common Activation Function

Three mathematical functions that are commonly used as activation functions are

- **Sigmoid:** function transforms input to produce an output value between 0 and 1
- **Tanh:** (short for "hyperbolic tangent") function transforms input to produce an output value between -1 and 1
- **ReLU:** (short for "rectified linear unit") produces  $\max(0, x)$



# Neural networks: Activation functions

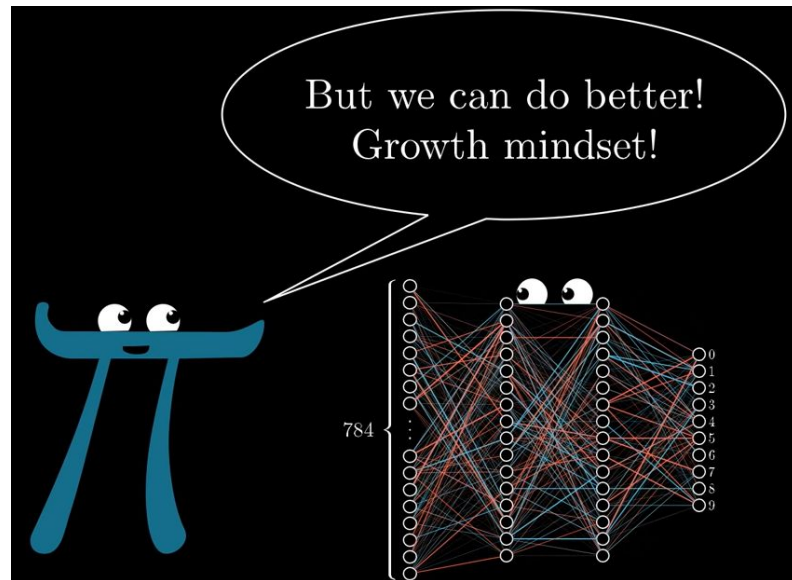
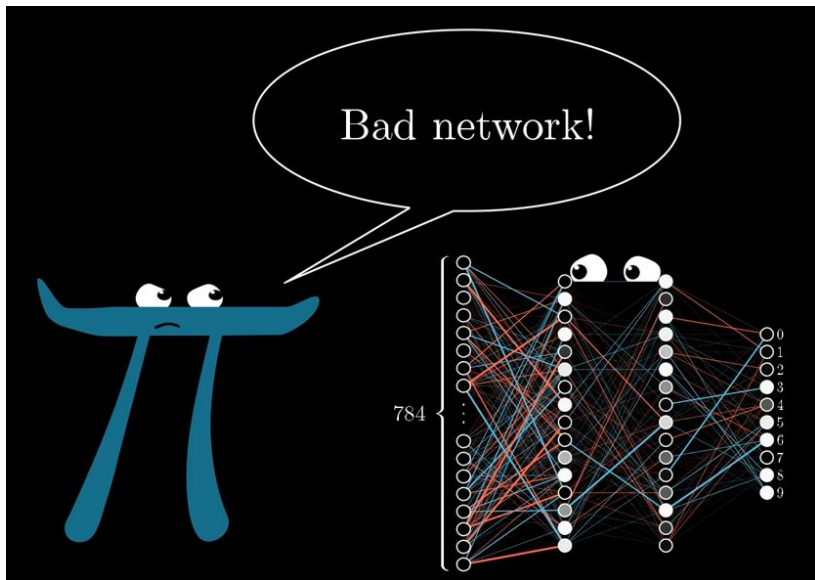
*Forward Propagation*

<https://developers.google.com/machine-learning/crash-course/neural-networks/activation-functions>

**How do we know if the NN will be good or bad?**

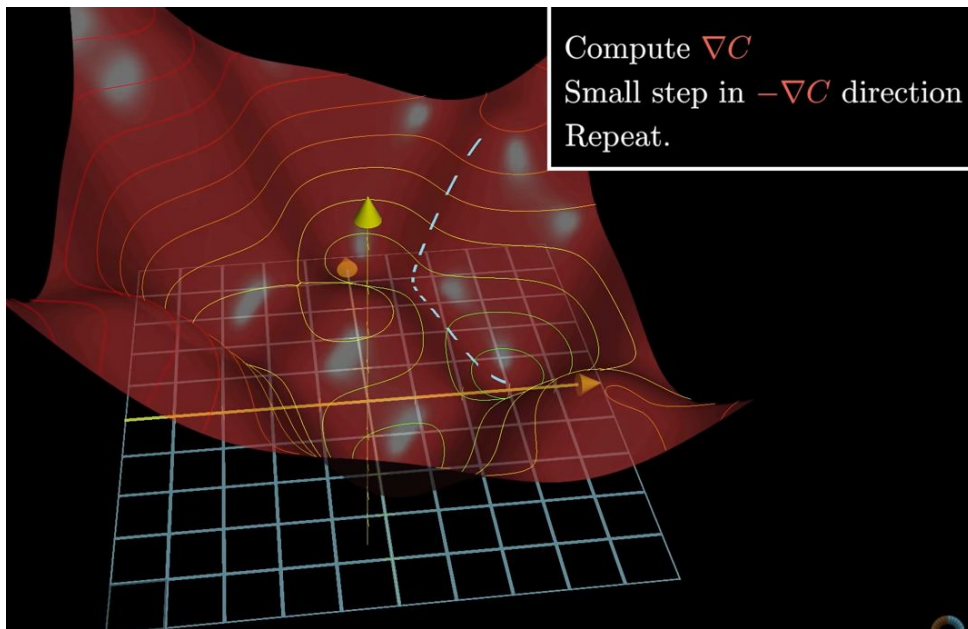
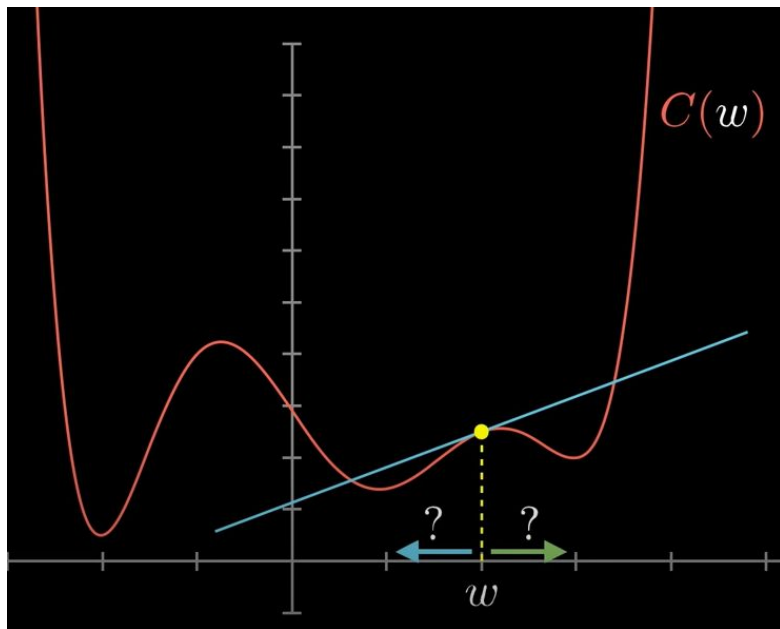
# Gradient Descent

# Optimizer: Gradient Descent (I)



Source: 3Blue1Brown

# Gradient Descent (II)



**NOTE:** You were already using this to optimize various models (Reg. Linear, Reg. Logistics, SVM)

Gradient = “Slope” (derivate) in multiple directions

Source: 3Blue1Brown

# Gradient Descent (III)

$$-\nabla C(\vec{W}) = \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix}$$

$w_0$	should increase somewhat
$w_1$	should increase a little
$w_2$	should decrease a lot
$w_{13,000}$	should increase a lot
$w_{13,001}$	should decrease somewhat
$w_{13,002}$	should increase a little

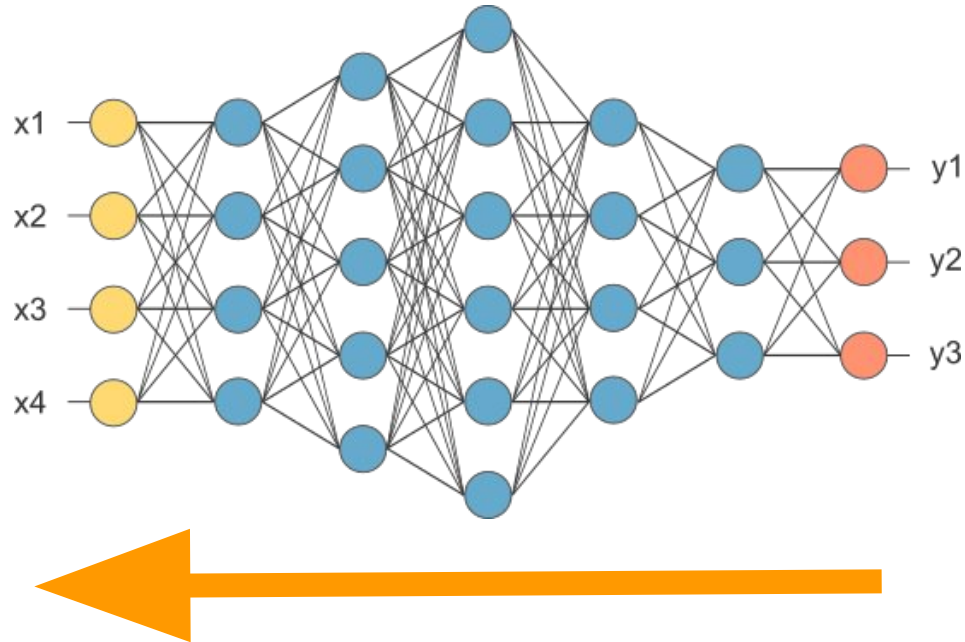


**How do we know if the NN will be good or bad?**

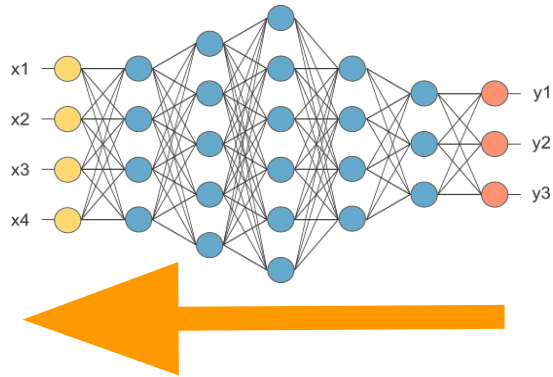
# Backpropagation

# BackPropagation (I)

Through the backpropagation process, **weights and biases are adjusted to optimize** (in this case, minimize) the **cost function** (for example, cross-entropy) of the neural network.

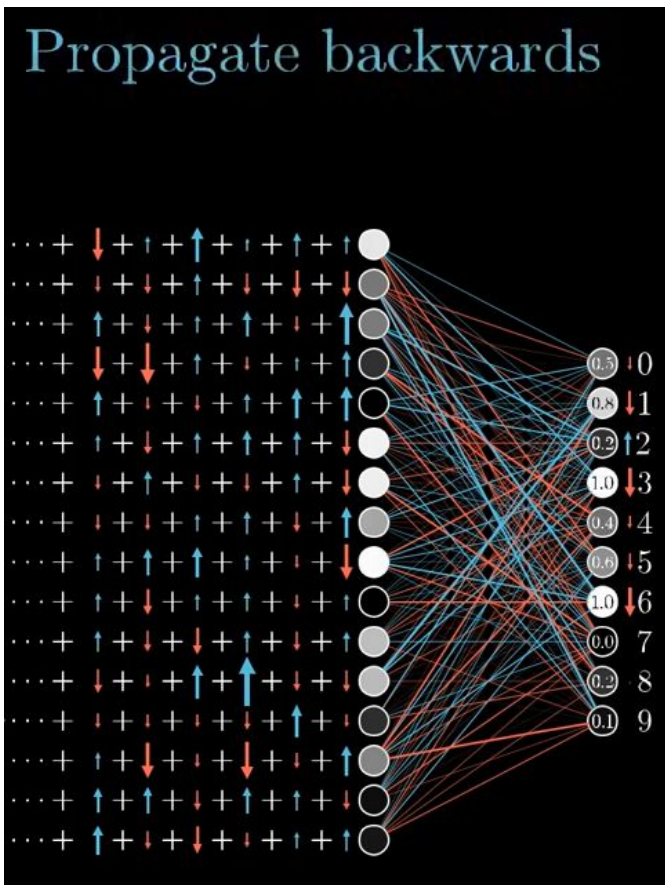


# BackPropagation (II)



$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(2)} (W^{(2)})^T \frac{\partial a^{(2)}}{\partial W^{(1)}}$$

# BackPropagation (II) - Cont'd



## ALGORITHM ("AS TO WALK AT HOME"):

1. Each neuron calculates the variation that affects it, either in a "positive" way (that is, the prediction is what is sought in that neuron) or "negative" (it is an "undesired" prediction, or in which the output does not must be activated).
2. Recursively, the same calculation is done on each layer of neurons, backtracking the effect of each neuron on the final output and updating the weights.
3. Once the input layer is reached, the next observation is taken.
4. Return to Step 1 until your training set has been fully used for training.

**HOW? WITH OPTIMIZERS (GD, SGD, etc.)**

# BackPropagation (III)

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

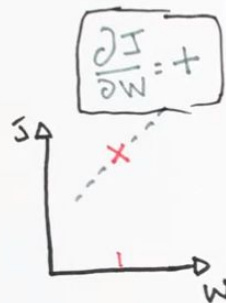
$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2} (y - \hat{y})^2 \quad (5)$$

$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)})W^{(2)}))^2$$

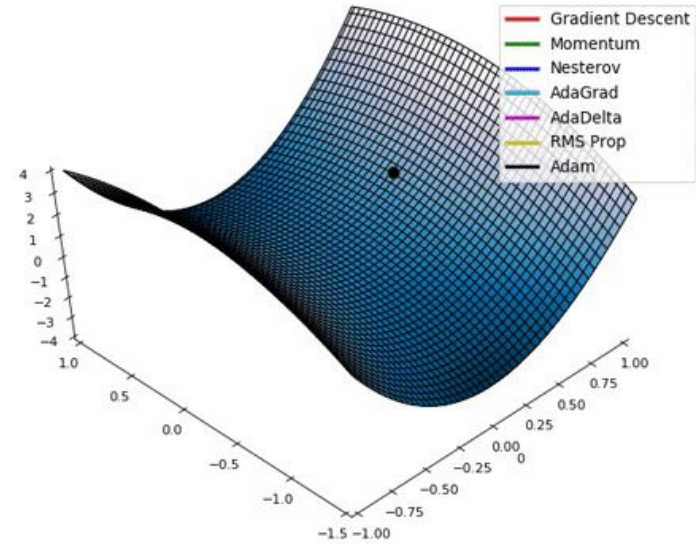
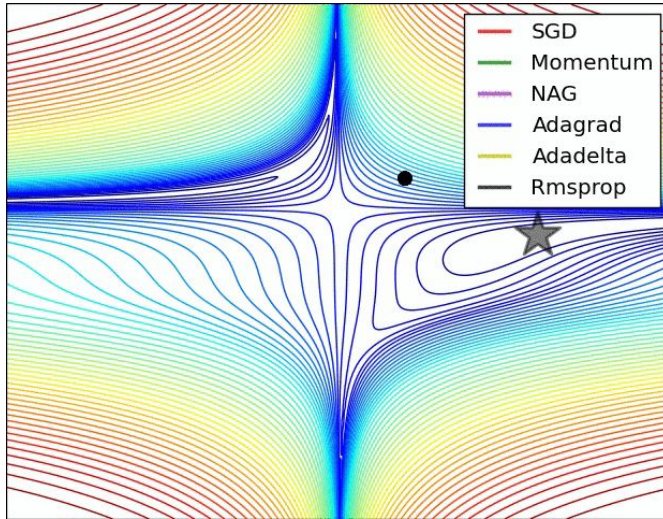
↑ HOW DOES THIS CHANGE  
IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$



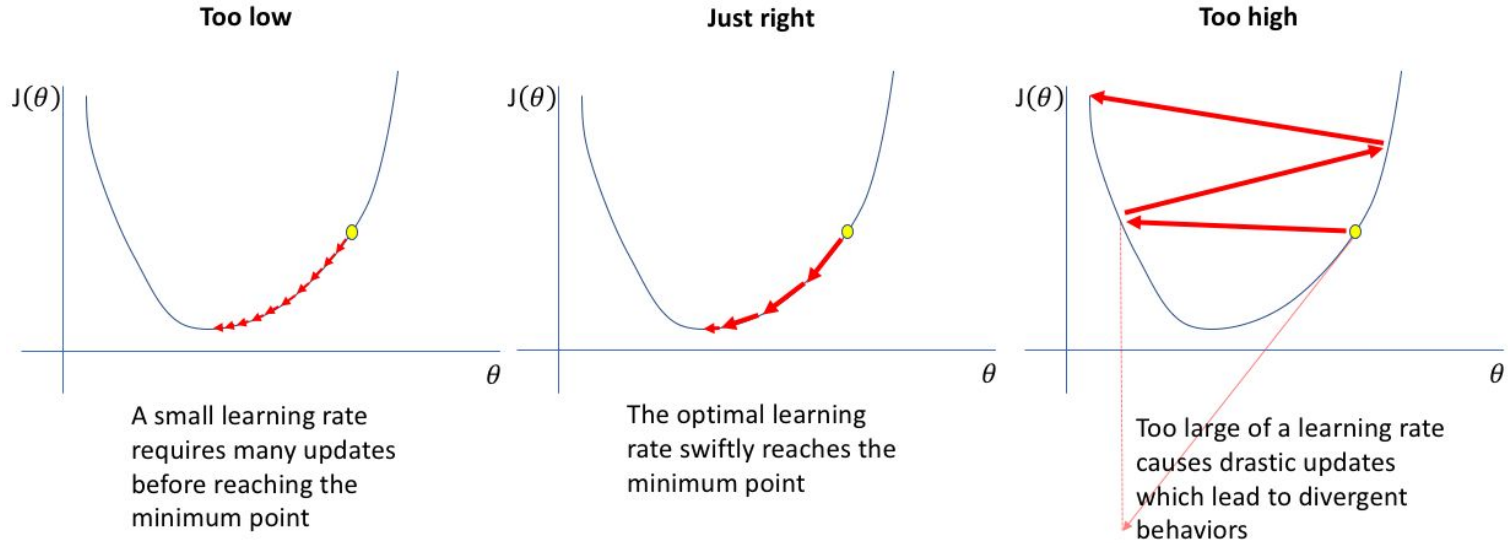
Source: Welch Labs

# Optimizers



Gradient Descent problem: PC dies... :- ( → **SGD (or Adam, SGD “improved” with moving averages)**

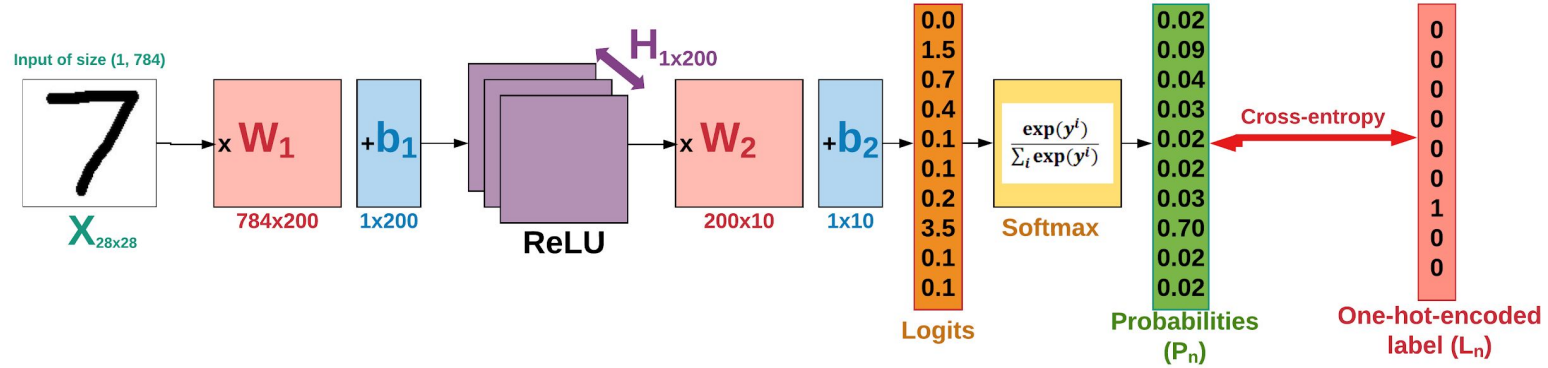
# Learning Rates



Understand the Impact of Learning Rate on Neural Network Performance



# Pipeline & Argmax



**CROSS-ENTROPY:** cost function used to evaluate the probabilities obtained. It is used in classification problems.

**So if this works like ML, can you overfit an NN?**

*INDEED!*

# Regularization

# Regularization: Methods

## 1) Dropout

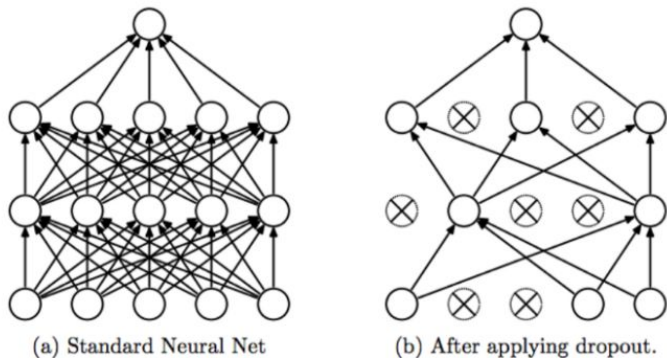
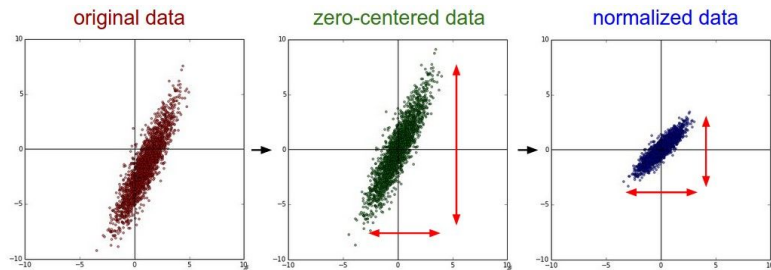


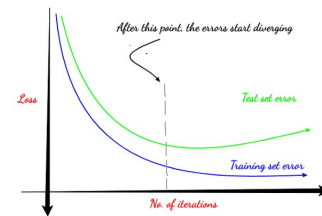
Image source: Google

## 2) Batch Normalization



3) Data Augmentation (p. ej. flip images)

4) Early Stopping (no “overtraining”)



5) Regularization L1 (subtract effect to gradient)

6) Regularization L2 (add component to loss to smooth small weight updates)

**What are the tools you can use to create a NN?**

# Deep Learning Libraries

 Keras **vs.**  PyTorch



theano

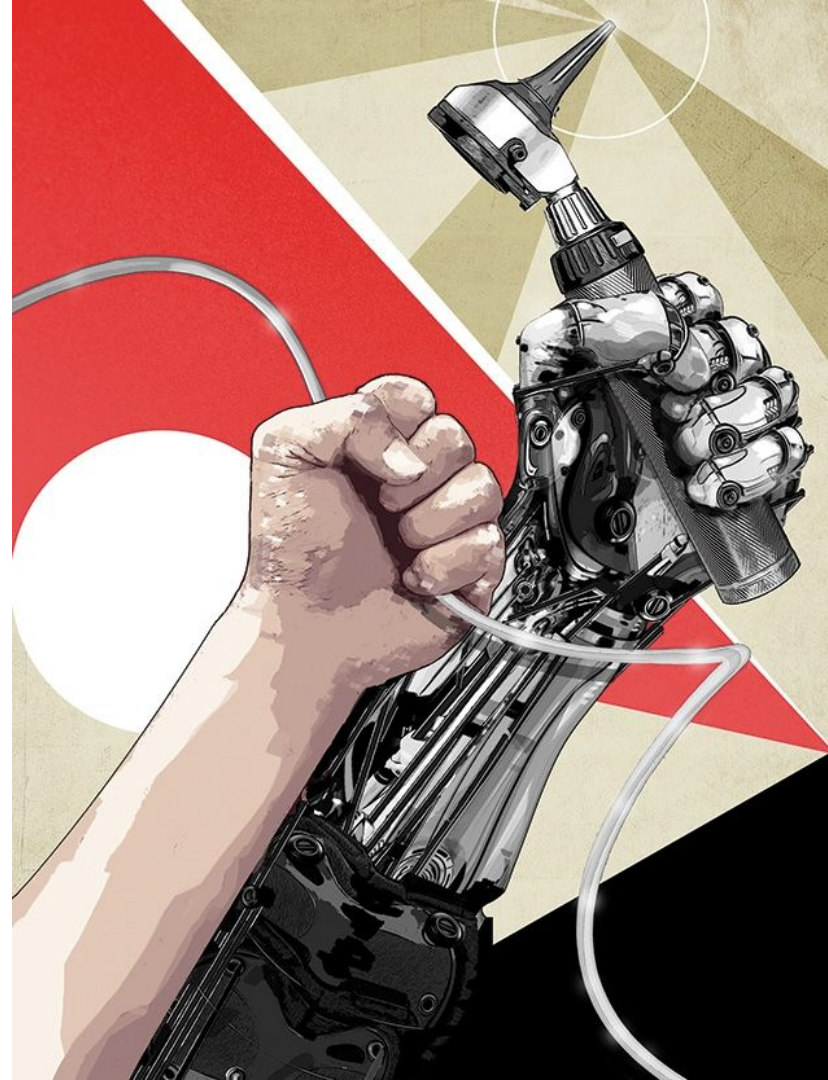


(aka CNTK)

[Keras vs Pytorch for Deep Learning](#)

# Practice - NNs

<https://developers.google.com/machine-learning/crash-course/neural-networks/interactive-exercises>



**Any  
questions?**



An abstract graphic featuring a network of interconnected nodes and lines. The nodes are represented by small circles in various shades of orange and brown, some of which are highlighted with a white outline. The lines are thin and light-colored, creating a complex web-like structure that fills the background. The overall aesthetic is modern and technological.

# THANKS