

Intro to Deep Learning

(without PyTorch 😭)

DECATHLON

UNIVERSITÉ
Concordia
UNIVERSITY



ERICSSON

Deep Learning

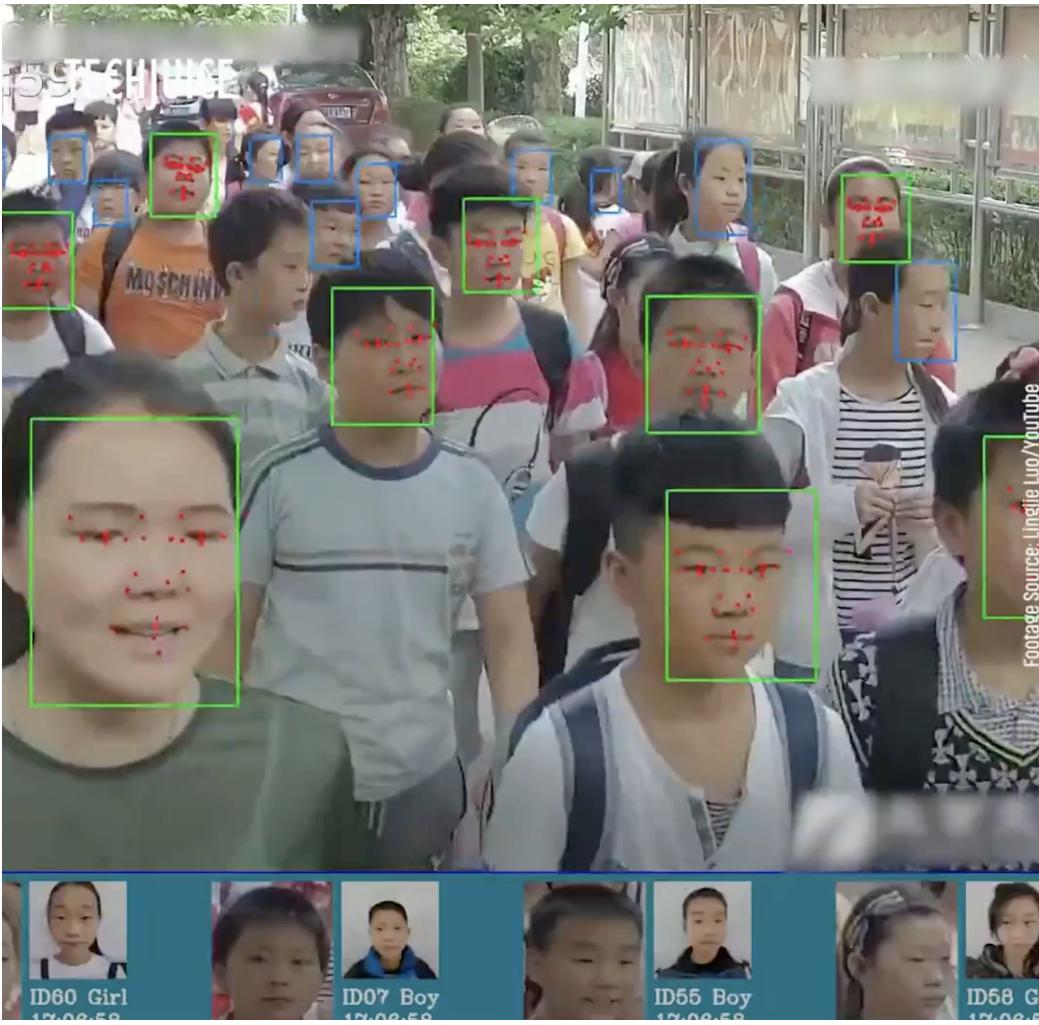
Deep learning (DL) is a *machine learning technique* that teaches computers to do what comes naturally to humans: **learn by example.** - MathWorks

[Deep] Neural Networks

The core of DL is built on **neural networks** which are developed as a *machine learning* approach to artificial intelligence (AI). - Wiki

Interview: <https://youtu.be/FTrQNmI3G0s>

Some real world applications





Vision-based tracking
Facebook Reality Labs and Oculus





A screenshot of a web browser window displaying the Google Translate interface at translate.google.ca. The browser has a standard OS X look with red, yellow, and green window controls. The address bar shows the URL. The main content area is titled "Translate". At the top, there are language selection boxes: "English", "Spanish", "French", "English - detected", and "Greek". Below these are two text input fields. The left field contains the English sentence "Today we will be studying machine learning?". The right field shows the Greek translation "Σήμερα θα σπουδάζουν μηχανική μάθηση;". Between the fields are icons for text alignment, font style, and a "Translate" button. At the bottom of the translation area, there's a "Suggest an edit" link. The footer of the page includes links for "Google Translate for Business", "Translator Toolkit", "Website Translator", and "Global Market Finder". A large black banner at the bottom left contains the text "Language translation". The footer also includes links for "About Google Translate", "Mobile", "Community", "Privacy & Terms", "Help", and "Send feedback".

Speech recognition



A screenshot of the Netflix homepage displayed in a web browser. The top navigation bar includes the Netflix logo, a 'Browse' dropdown, a 'Kids' link, a search bar, a bell icon for notifications, and a user profile icon. Below the header, there's a row of movie and TV show thumbnails. The first two are 'ANT-MAN' and 'LEGO BIONICLE: THE JOURNEY TO ONE'. Following these are several other thumbnails: 'LEGO DC COMICS SUPER HEROES: BATMAN: BE-LEAGUERED', 'YOUNG JUSTICE', 'INSIDE OUT', and parts of 'LEGO BATMAN 2' and 'LEGO STAR WARS'. A 'Trending Now' section follows, featuring 'DAREDEVIL', 'HOUSE OF CARDS', 'FULL HOUSE', 'THE AMAZING SPIDER-MAN', and 'SOME ASSEMBLY REQUIRED'. The bottom section is labeled 'Top Pic' and shows thumbnails for 'TEENAGE MUTANT NINJA TURTLES', 'Kingsman: The Secret Service', 'NIGHT AT THE MUSEUM: SECRET SECRETS', 'THE SPONGEBOB MOVIE: SPONGE OUT OF WATER', 'BATMAN: THE DARK KNIGHT RETURNS', and 'FANTASTIC FOUR'.

Recommender systems

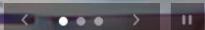


[Home](#) [Gallery](#) [FAQ](#) [Blog](#) [Contact us](#)

Art made by Artificial Intelligence

Your NFT comes with a free physical copy printed on museum-quality canvas shipped anywhere in the world.

OUR NFT'S



"Cubist Colors of Fall"
0.07 ETH 0.1 ETH

SALE



"de los Cuérragos"
0.1 ETH

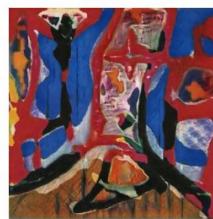


"Ali the Duchess of Norbury"
0.1 ETH



"Sligoville"
0.07 ETH 0.1 ETH

SALE



"Muhimbasa"
0.07 ETH 0.1 ETH

SALE



"Big Blue Boy"
0.1 ETH



"The Golden Giraffes"
0.1 ETH



"Wake up Daddy, it's a Starry
Safari"
0.1 ETH

Most of these technologies
(if not all!) are powered by
neural networks

<https://aiderm.herokuapp.com/>

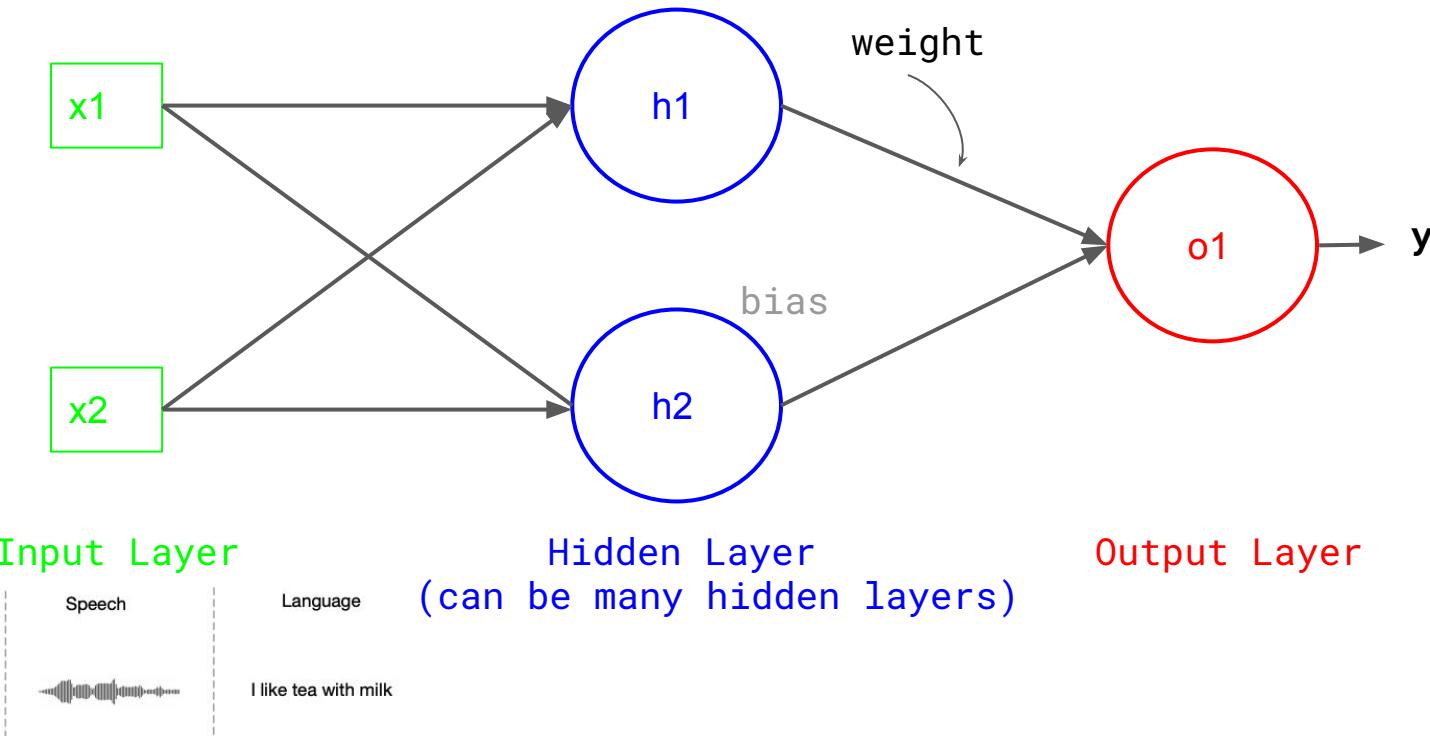
<https://hasibzunair.github.io/resm3dvton/>

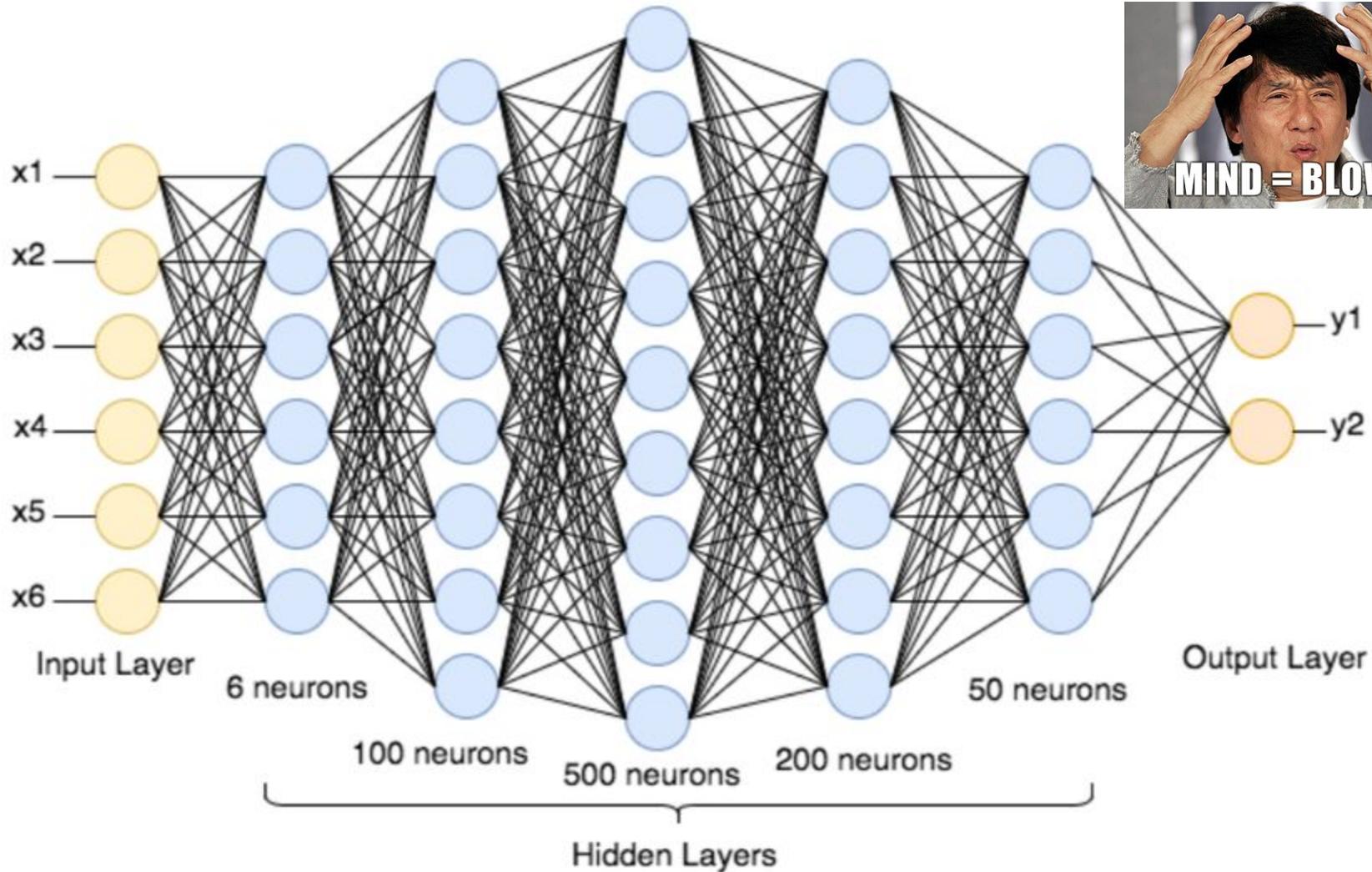
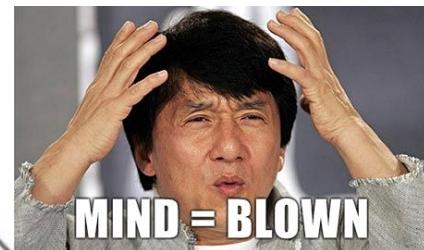
<https://github.com/hasibzunair/boss-detector>

How to make them?

Anatomy of neural networks

This is a “2-2-1” sigmoid net with logistic output

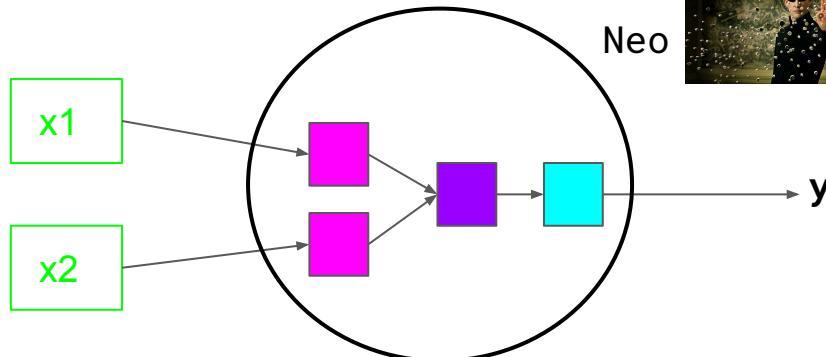




A Single Neuron

Takes **input**, does some math and produces an **output**.

Neo  can be **h1**, **h2** or **o1**

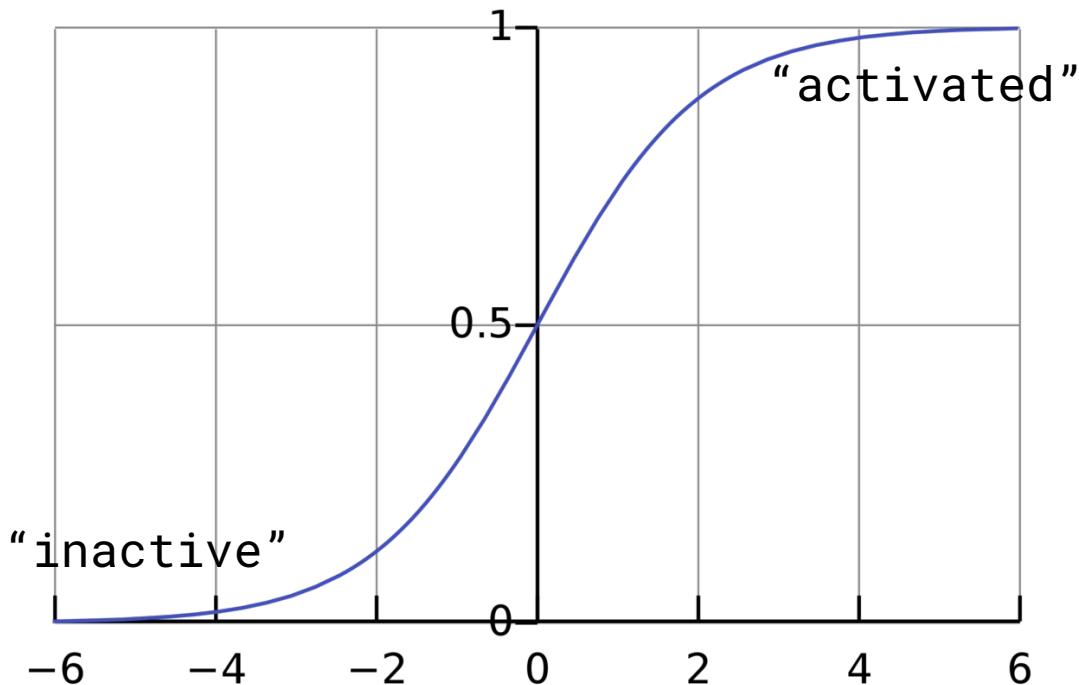


 “multiply”

 “add” bias

 “activate”

Activation Function (How is Neo a neuron?)



This is a sigmoid function.

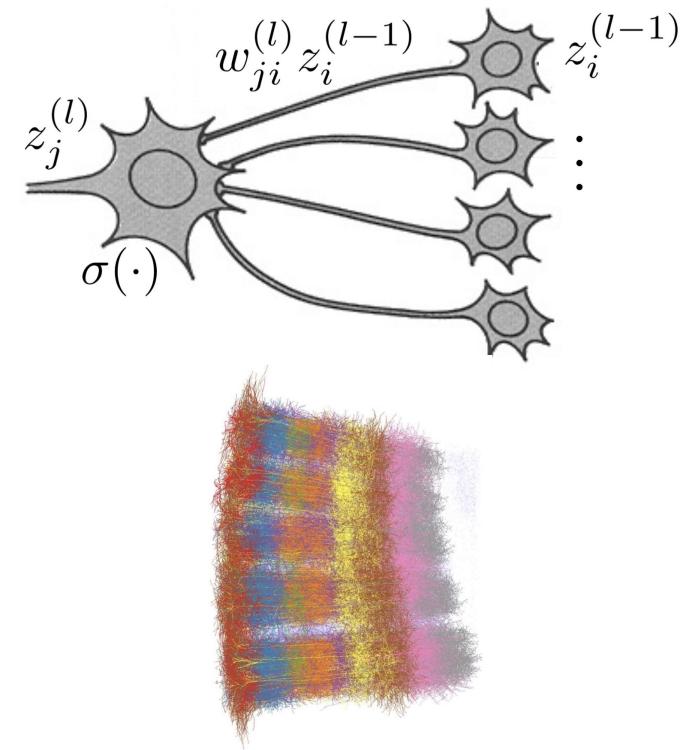


Image credit: Marcel Oberlaender

An example with one neuron (math)

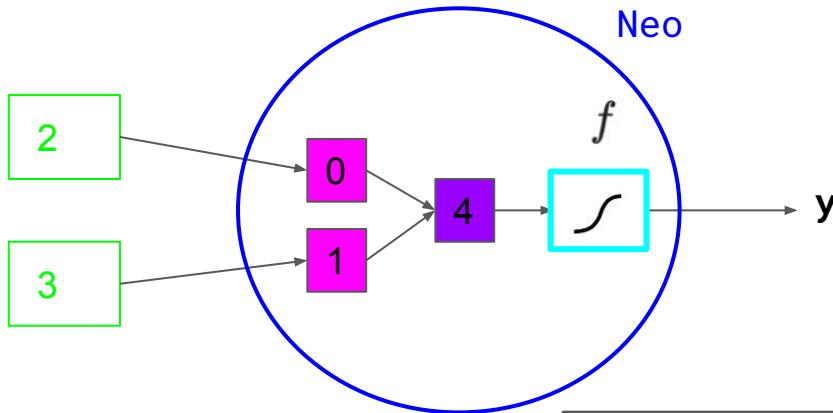
$$w = [0, 1] \quad b = 4 \quad x = [2, 3]$$

$$\begin{aligned} & \text{"multiply weights"} \qquad \qquad \qquad \text{"add bias"} \\ (w \cdot x) + b &= ((w_1 * x_1) + (w_2 * x_2)) + b \\ &= 0 * 2 + 1 * 3 + 4 \\ &= 7 \end{aligned}$$

$$y = f(w \cdot x + b) = f(7) = 0.999$$

This is known as the **forward pass** in a neural network.

An example with one neuron (visual)



$$(w \cdot x) + b = ((w_1 * x_1) + (w_2 * x_2)) + b$$

$$= 0 * 2 + 1 * 3 + 4$$

$$= 7$$

$$y = f(w \cdot x + b) = f(7) = 0.999$$

An example with one neuron (Code)

```
def sigmoid(x):
    """
    Sigmoid activation function.
    f(x) = 1 / (1 + e^(-x))
    """
    return 1 / (1 + np.exp(-x))

class BabyNeuron:
    """
    Implementation of one neuron.
    """

    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

    def forward(self, inputs):
        """
        Multiply inputs with "weights", then "activate".
        y = W * x + b
        """
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
        """
        activate!
        """
```

1

```
# Setup weights and bias (W and b)
weights = np.array([0, 1]) # w1 = 0, w2 = 1
bias = 4 # b = 4

# Create our baby neuron
babynet = BabyNeuron(weights, bias)
```

2

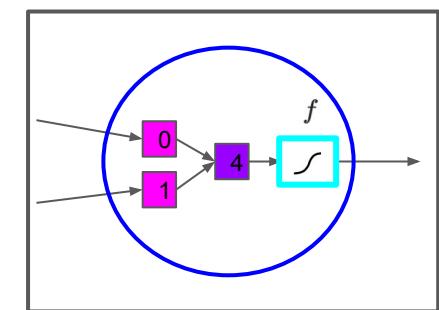
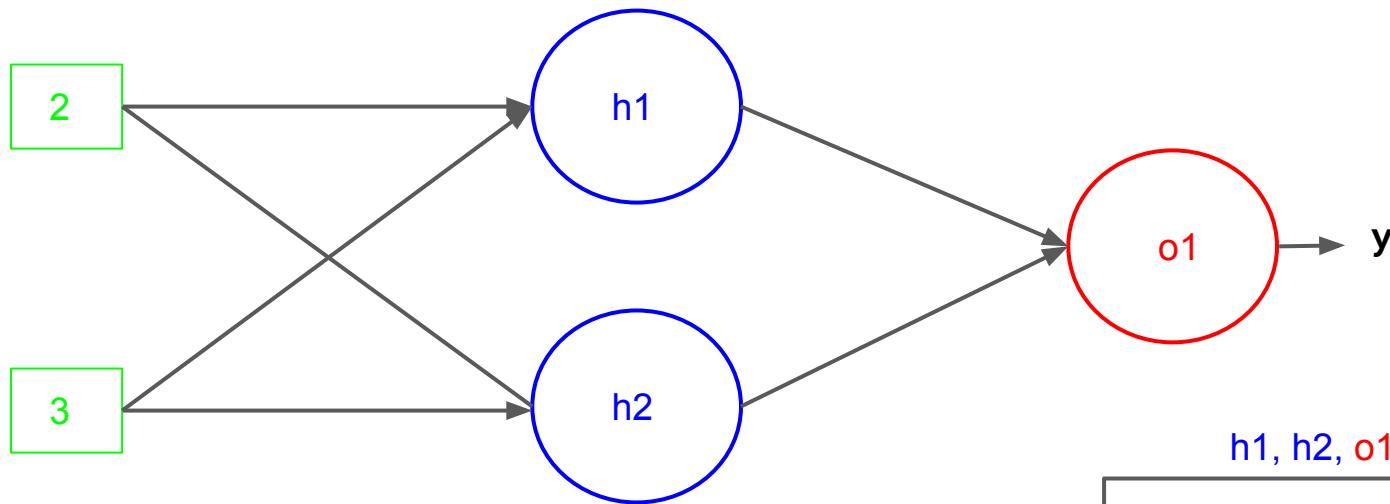
```
# Set inputs, x1 = 2 and x2 = 3
x = np.array([2, 3])

# Pass input x to babynet
print("Output: {:.5f}".format(babynet.forward(x)))
# Should get 0.99908
```

3

Output: 0.99909

A small neural net (visual)



A small neural net (math)

$$w = [0, 1] \quad b = 0 \quad x = [2, 3]$$

3 neurons (h1,h2,o1)
with same weight

$$\begin{aligned} h_1 &= h_2 = f(w \cdot x + b) \\ &= f((0 * 2) + (1 * 3) + 0) \\ &= f(3) \quad \text{"activate!"} \\ &= 0.9526 \end{aligned}$$

$$\begin{aligned} o_1 &= f(w \cdot [h_1, h_2] + b) \\ &= f((0 * h_1) + (1 * h_2) + 0) \\ &= f(0.9526) \\ &= 0.7216 \end{aligned}$$

A small neural net (code)

```
class BigBaby:  
    """  
    A "network" with:  
    - 2 inputs  
    - a hidden layer with 2 neurons (h1, h2)  
    - an output layer with 1 neuron (o1)  
  
    Each neuron (i.e. BabyNeuron) has the same weight and bias:  
    - w = [0, 1]  
    - b = 0  
    """  
  
    def __init__(self):  
        # We initialize using same weights and bias for each neuron  
        weights = np.array([0, 1])  
        bias = 0  
        # BabyNeuron class from previous section  
        self.h1 = BabyNeuron(weights, bias)  
        self.h2 = BabyNeuron(weights, bias)  
        self.o1 = BabyNeuron(weights, bias)  
  
    def forward(self, x):  
        out_h1 = self.h1.forward(x)  
        out_h2 = self.h2.forward(x)  
        # The inputs for o1 are the outputs from h1 and h2  
        out_o1 = self.o1.forward(np.array([out_h1, out_h2]))  
        return out_o1
```

1

```
# Create our BigBaby network  
network = BigBaby()  
  
# Set inputs, x1 = 2 and x2 = 3  
x = np.array([2, 3])  
  
# Pass input x to network and make predictions  
print("Output: {:.5f}".format(network.forward(x)))  
# Should get 0.72163
```

2

Output: 0.72163

For each neuron: “multiply weights” “add bias” “activate!”

Training a neural network

- Neural networks are trained by **gradient descent**
- Gradients are computed by a special algorithm called “backpropagation”
- “Backpropagation” is an *efficient algorithm to compute derivatives*

Stochastic Gradient Descent (SGD)

lower loss = better predictions

- Update **parameters** of neurons such that it **minimizes** a cost/error/loss/objective function.
- Loss is used to quantify how “good” network is
- Requires:
 - Current weights (and biases)
 - Loss
 - Gradient with respect to loss (backprop)

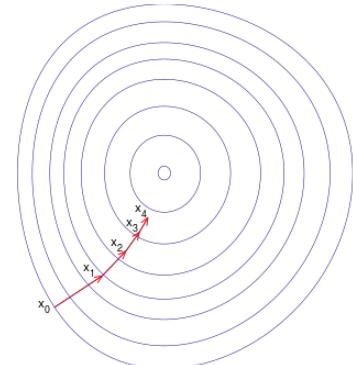


Image credit: Wiki

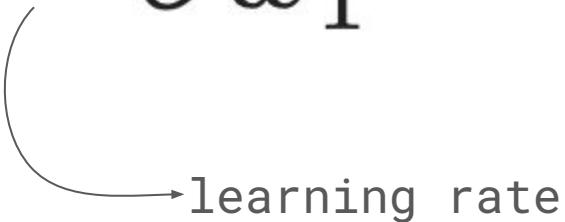
Stochastic Gradient Descent



Here, **elevation** is our loss.
We want to **minimize** it by going down!

Gradient Descent

Move towards negative of the gradient in order to “minimize” the loss

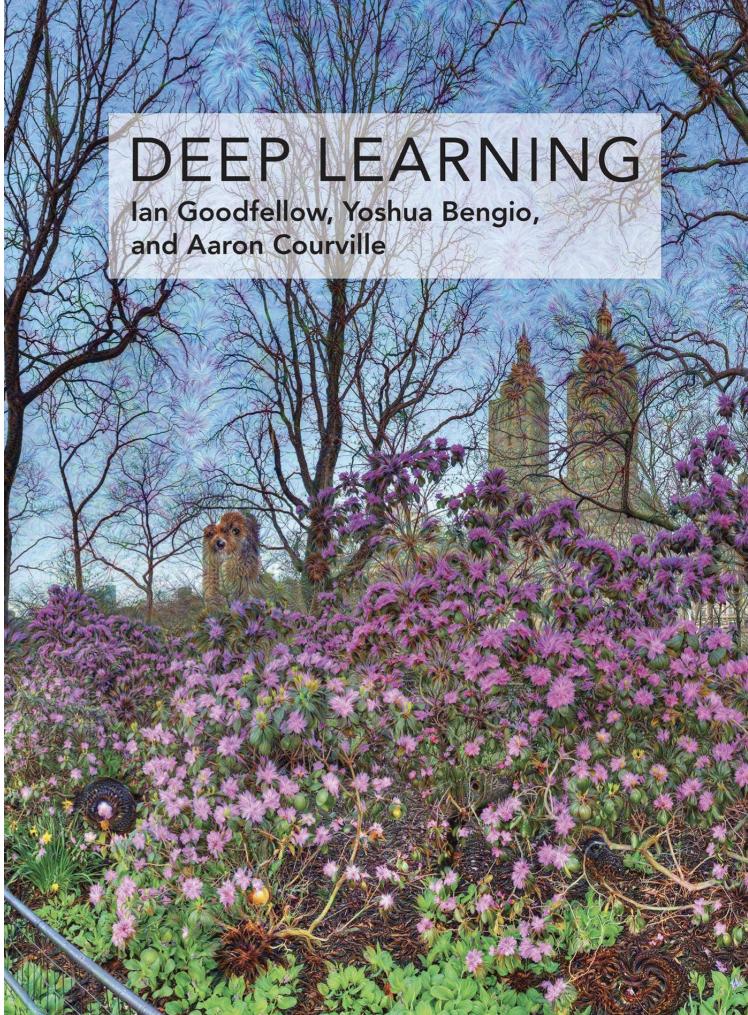
$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$


A curly brace is positioned under the term $\eta \frac{\partial L}{\partial w_1}$. An arrow originates from the bottom right corner of this brace and points to the word "learning rate" located at the bottom right of the slide.

How a neural network learns?

- Neural net makes a “guess” (**forward pass / make predictions**)
- Compute how good the **guess** is against the **known correct answer** (**loss**)
 - prediction**
 - ground truth**
- Update weights so that **guess** is **better** by
 - lower loss = better predictions**minimizing the loss (**gradient descent**)
- Do until certain number of steps or loss is minimal

Let's see an example!



DEEP LEARNING

Ian Goodfellow, Yoshua Bengio,
and Aaron Courville



WERE

FINISHED

