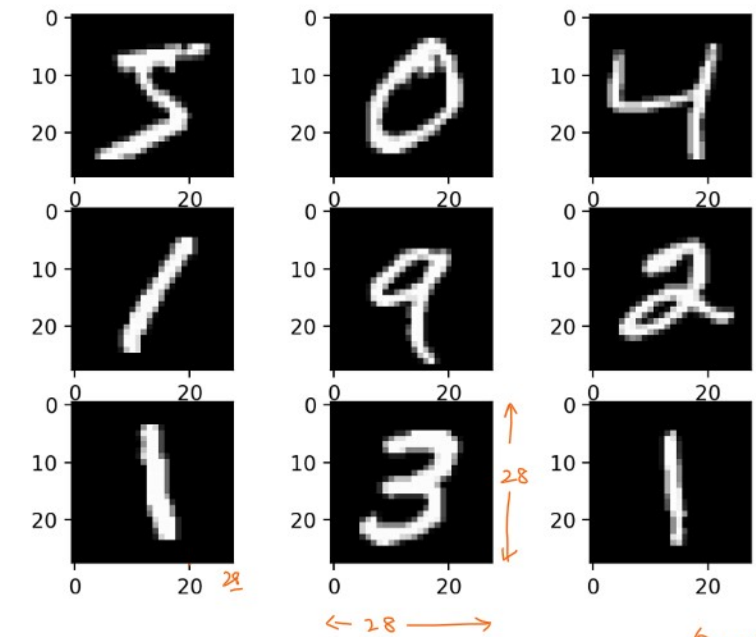
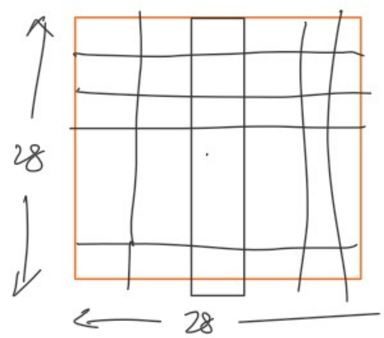
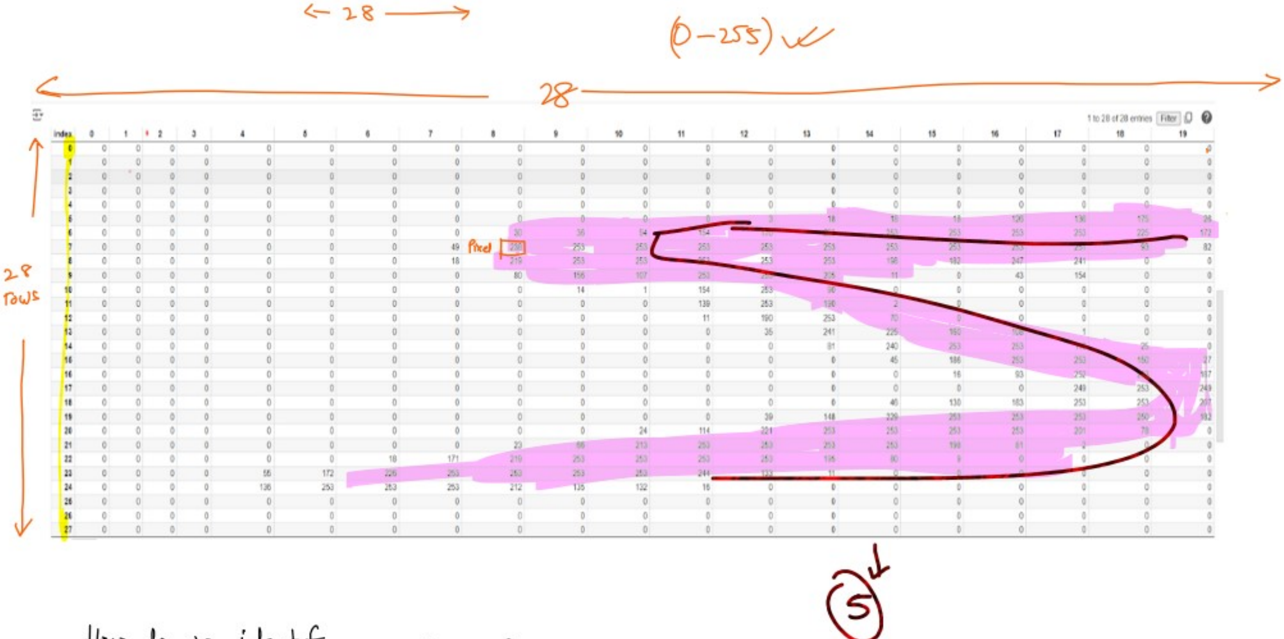
# Weights & biases

27 July 2024 20:10



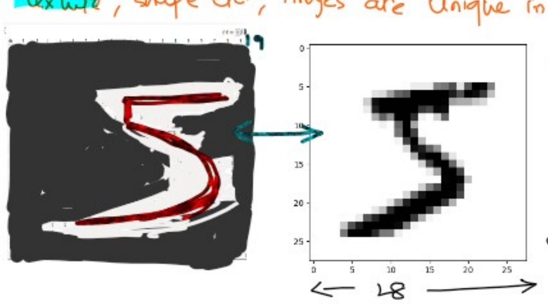




How do we identify any image ?

- depending on the edges, patterns texture, shape etc., images are Unique in nature

9 9 →



**Deep Neural Network**

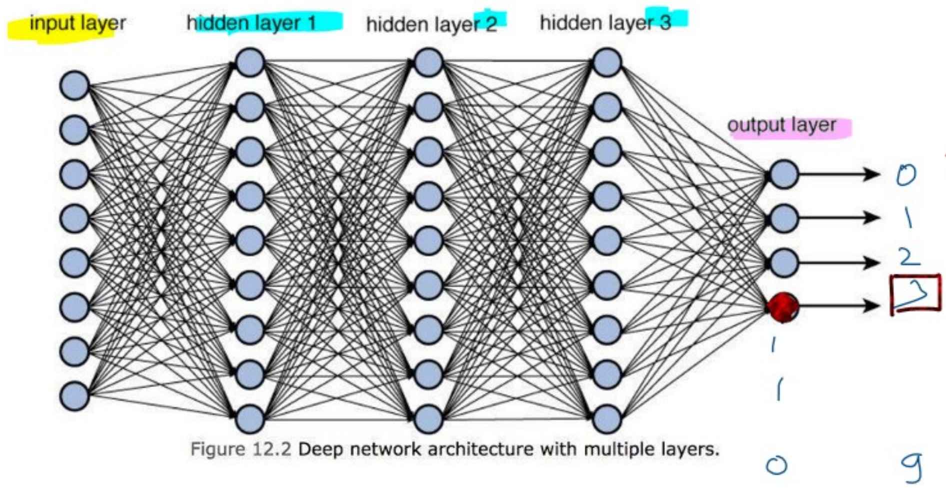
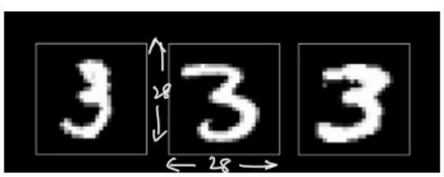



Figure 12.2 Deep network architecture with multiple layers.

Multi-class problem.

Input Layer:

To input the data features (Input variables); in our case, pixel values of an image



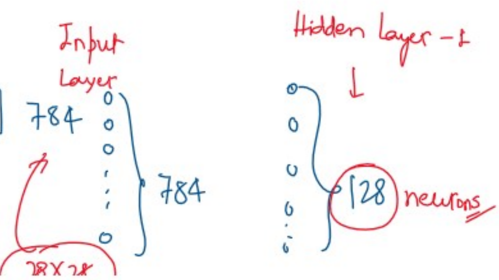

28*28=**784** → No. of input features (neurons) connecting to hidden layer.

MNSIT

| flatten_1 | input: | (None, 28, 28) |
|---|---|---|
| Flatten | output: | (None, 784) |

| dense_3 | input: | (None, 784) |
|---|---|---|
| Dense | output: | (None, 128) |

784



784

128 neurons

28X28

28 columns

28 rows

2D

28X28

Flatten → 1D

np.flatten()

pixel

dpi

dot per inch

4 by 3

What do you mean by pixel ?

- It means 'picture element' which is the smallest unit of a digital image or display.

Display resolution
Adjust the resolution to fit your connected display

1920 × 1080 (Recommended)

Pixels per inch → ~~ppi~~ → Higher the ppi ⇒ more details and sharpness in image.

1920 rows by 1080 columns

1080

1920

= 1920*1080=2,073,600 ⇒ 2 MP.

12MP → $12 \times 10^6$

$\frac{4 \times 10^3}{4000}$

3000
$3 \times 10^3$

12 MP

3000 pixels

4000 pixels

Let us assume

12 Million pixels.

300 PPI

300 pixel per inch

12 MP

b=13.33 inch

10"

← 6" →

← a →

10 inch

$\frac{4000}{300}$ = 4000/300=13.3333

inch

a) Kelan - 10x 13.33

b)

3000 pixels

4000 pixels

2D

b

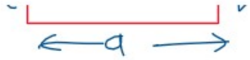
a

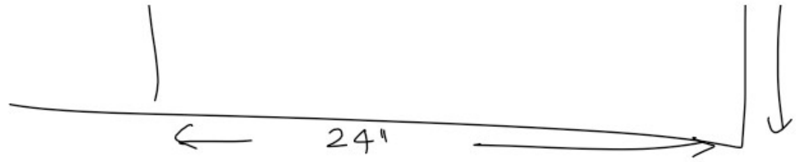
300 PPI

300

Landscape

20"




'stretched'

## Hidden Layer(s)

- It is a black box
  - we'll open the black box. ??

as agreed

Caution: going to be overwhelming

- intermediate layer between input and output layers.
- these layers (hidden) perform computations and extract the features from the input data

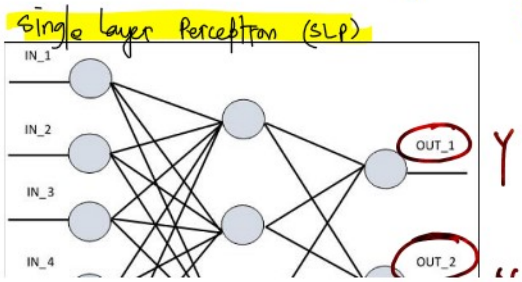
- 'deep' learning term refers to the network with multiple hidden layers.

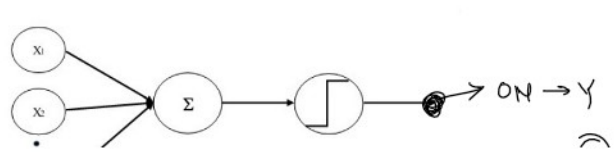
## Output Layer

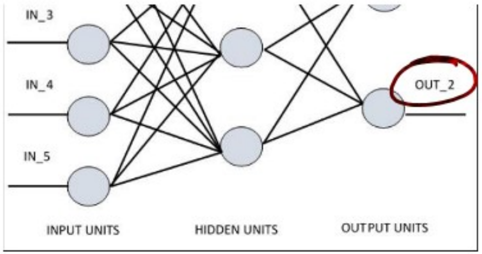
- It is the final layer that produces the o/p of the network.
  - one neuron in the output layer for binary classification

'sigmoid'

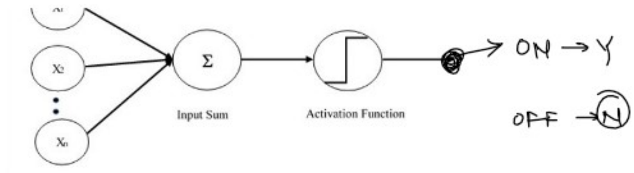
Single Layer Perceptron (SLP)



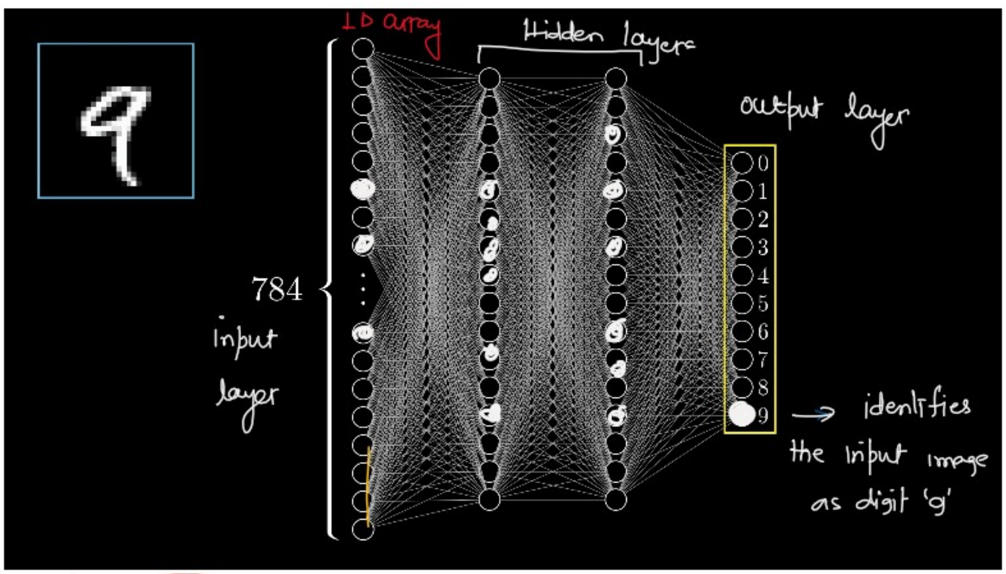



N




- multiple neurons for multi-class classification
  - softmax

Warren Mc culloh and Walter Pitts (1943) — proposed the first mathematical model for neural network.




0 to 255

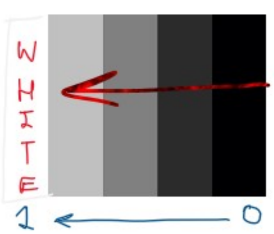
↓

Normalization

0 ⟷ 1

0 → Black

1 → White




0–255 range in pixels

Total 256 values

most digital images → JPEG, PNG

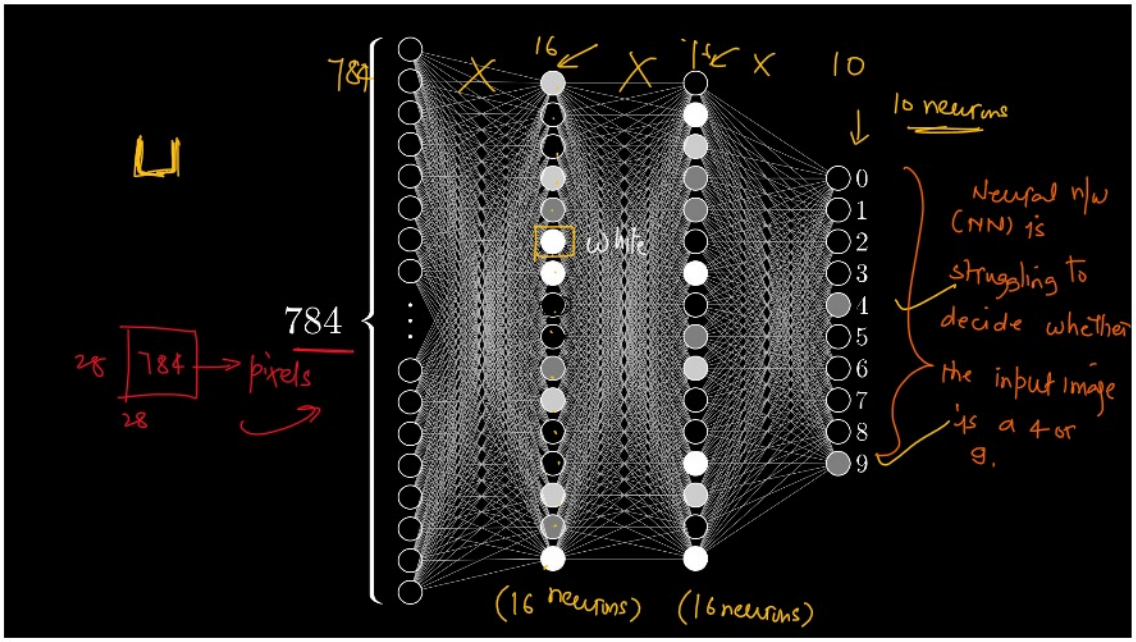
0: represents black

255: represents white

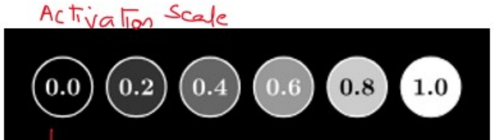
256 shades of black to white

127 → represents medium grey

0–255 $\xrightarrow{\text{normalization}}$ 0–1



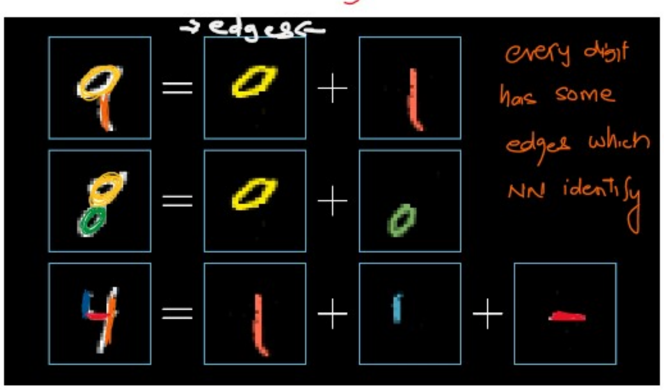

Activation Scale



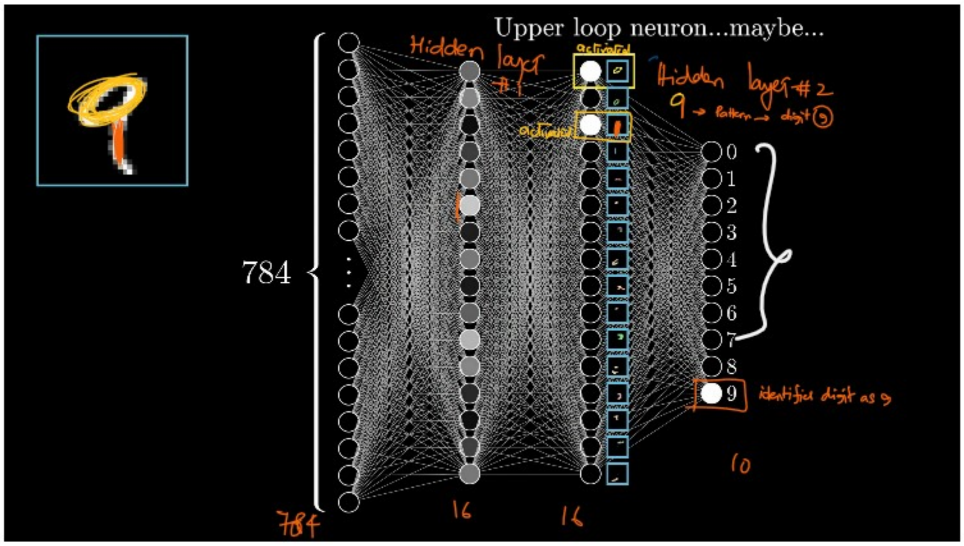

black → white

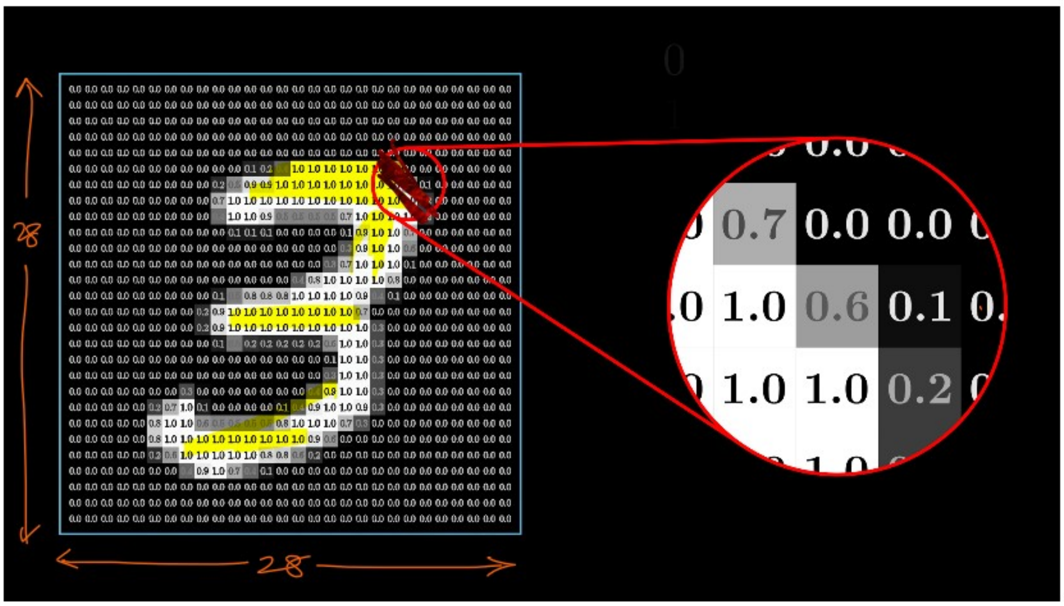
## Intuitive Understanding (going one layer deep)

How does the training work?




edges → pattern → digit



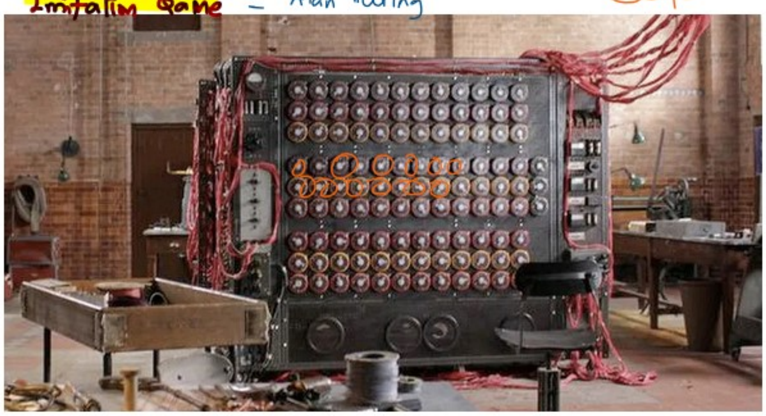


Patterns

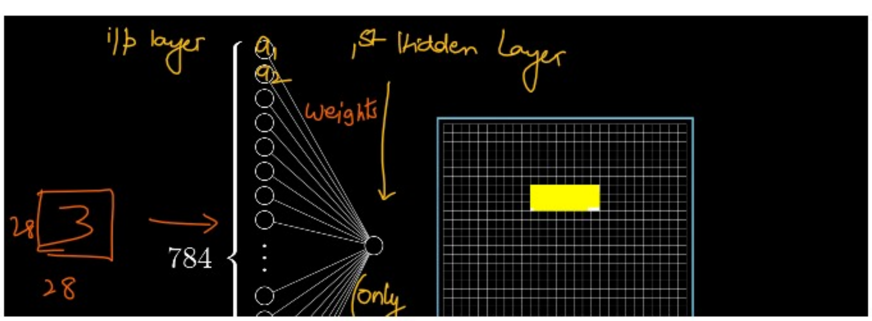
Detecting edges and forming patterns to help us with image - recognition tasks


Source:- Kevin Pleck

'Imitation Game' - Alan Turing ©aupc



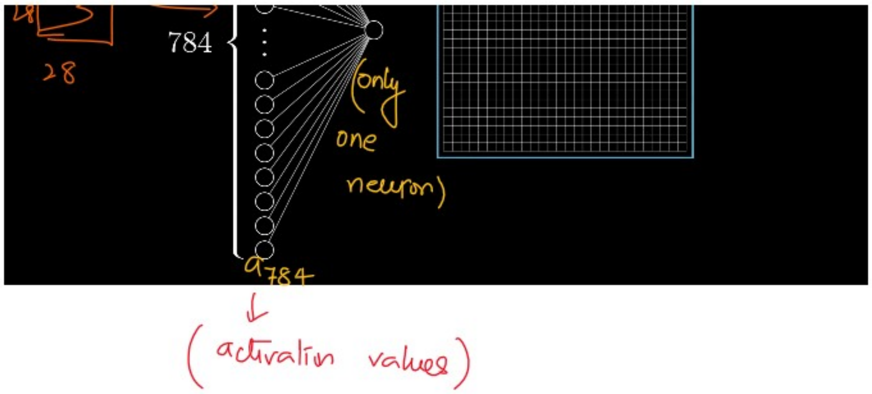
## How information passes between layers




$$\beta_0 + \overbrace{\beta_1 x_1 + \beta_2 x_2 + \cdots \beta_{784}\, x_{784}}^{784 \text{ variables}}$$

bias

weights

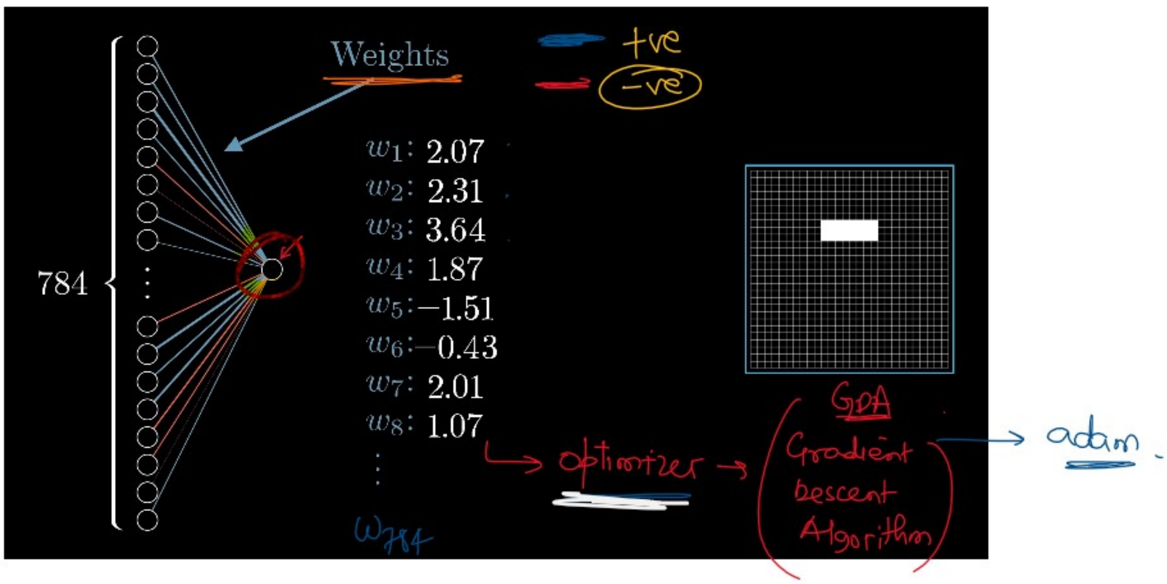


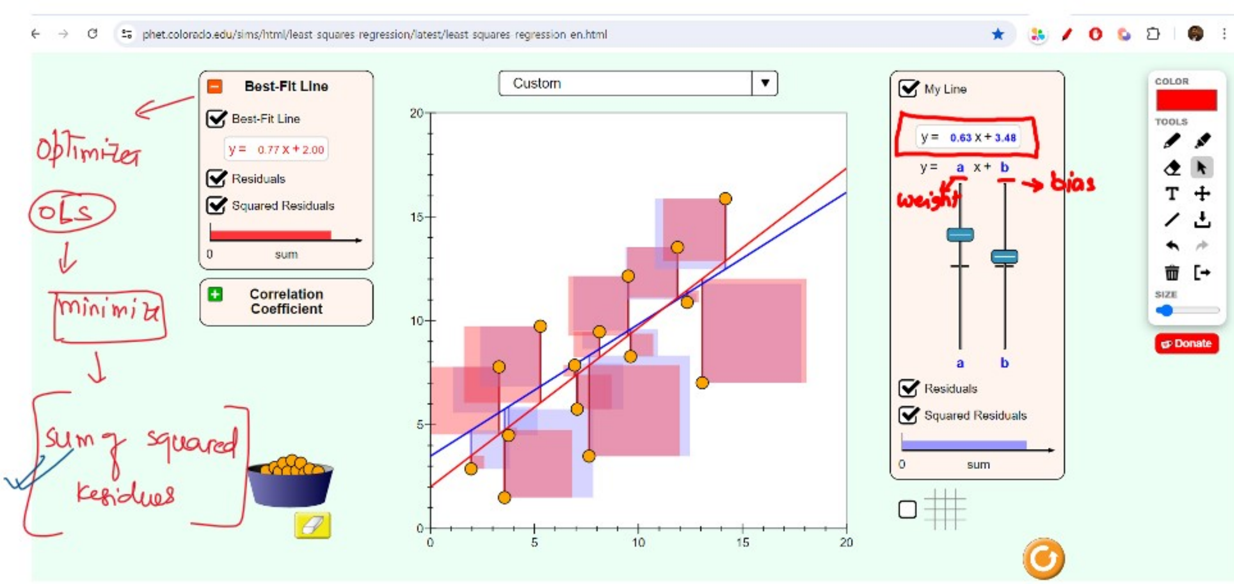

weights

$y = \boxed{20} + 100x_1 + 10x_2$

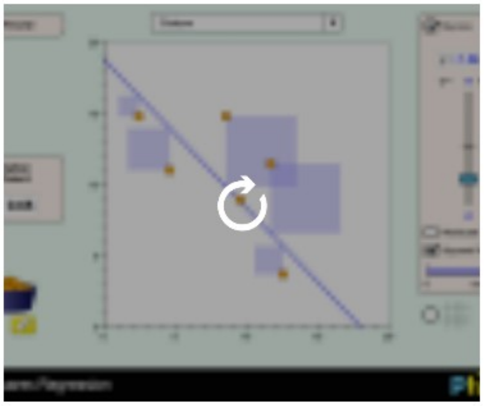
constant

Price of tomato

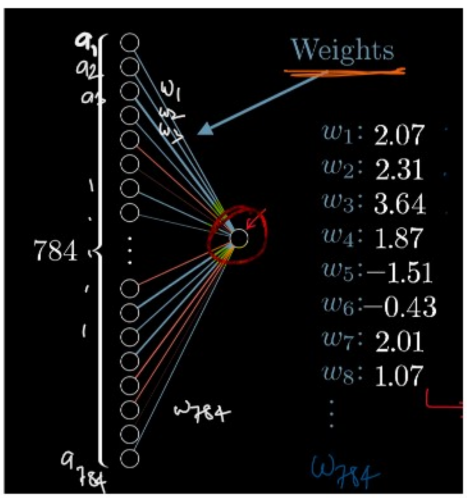



[Least-Squares Regression]



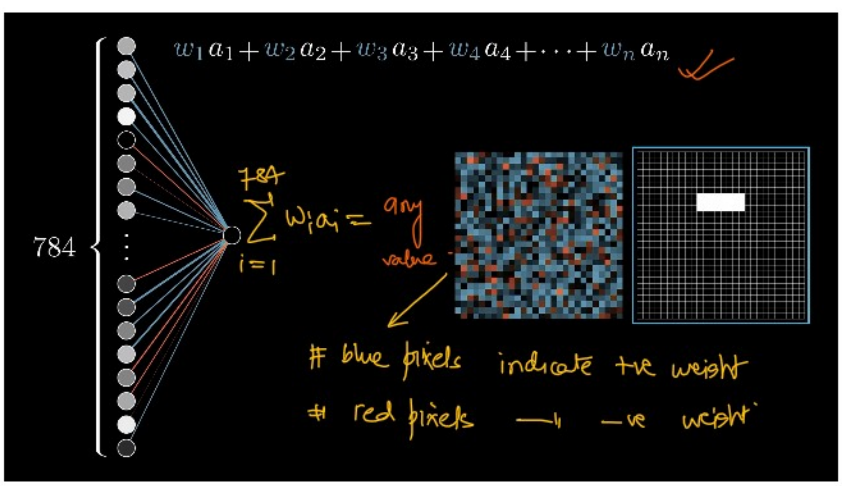
To actually compute the value of this 1st hidden layer neuron, take all the activations from (784) neurons in the input layer and computed their weighted sum




$\underbrace{w_1 a_1 + w_2 a_2 + \cdots + w_{784} a_{784}}$

(sum product of activation values and the associated weights)

$\sum_{i=1}^{784} w_i a_i$ ✓




$\sum^{784} w_i a_i = 9.2$ (+ve value) → white (lighter color)

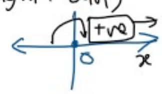
→ neuron

0.7 → A

0.5–0.7 → B

$\sum_{i=1}^{784} w_i a_i = 9.2$ (+ve value) → white (lighter color)

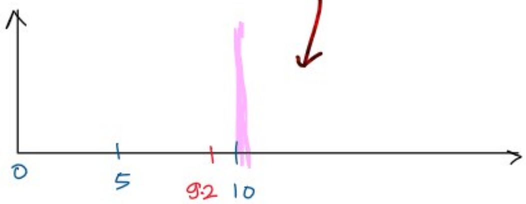
0 +




→ neuron lights up

threshold = 0.5

0.7 → A
0.5 – 0.7 → B
0.3 – 0.5 → C
< 0.3 → D

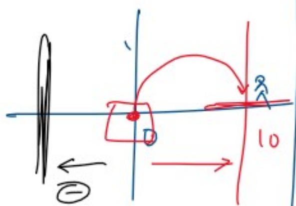



$\underbrace{w_1a_1 + w_2a_2 + \dots}\, + w_na_n + b$

$\left(\sum_{i=1}^{n} w_i a_i\right) + b$

$b = -10$

$[\underbrace{9.2}_{\sum_{i=1}^{784} w_i a_i} - 10] = -0.8$ → won't get lighten up




Here, we dont want the neuron to light up when the weighted sum is greater than '0'

What if we want the neuron to light up when the weighted sum is greater than 10

⇓

This is exactly the reason we need bias.

Bias helps us to shift the activation function in order to enable the model to fit the data more accurately

(slight pinch)

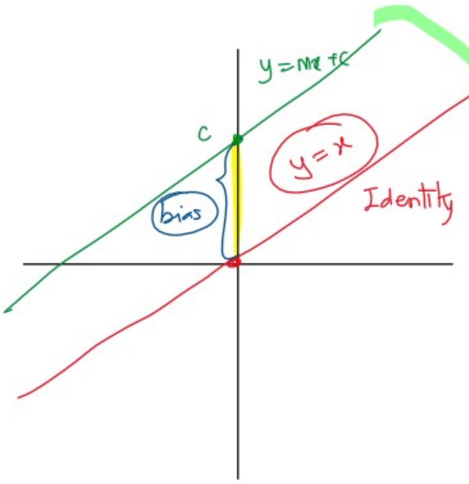
↓

Do not move the arm away in immediate reaction to the pinch

'certain threshold for the pain'

↓

bias




Finance

credit score | CIBIL score

'Loan approval status'

credit score

last 3 months

0 transactions

$x_1$ $x_2$ $x_3$

Y

bias ⇔ threshold

'loan gets rejected in reality'

'Boom light up'

0 — 10

[loan approved Yes]

(classification)

0.001 → In loan approval use case,

Neuron value

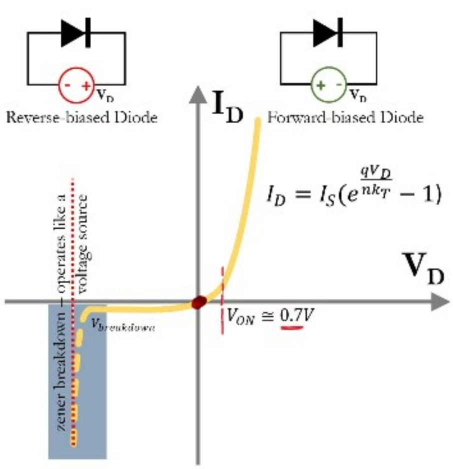
$f(x_1, x_2, \ldots x_{784})$

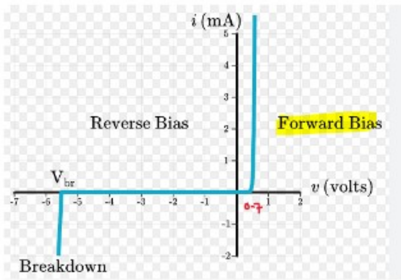
Model → Amount

(regression)

Loan requirement ⟹ 50 lakhs

~~Approve~~



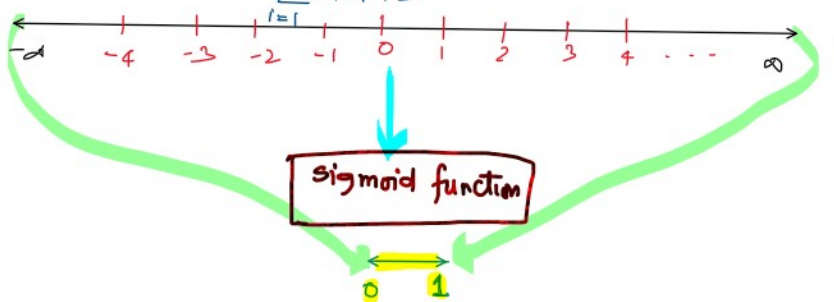


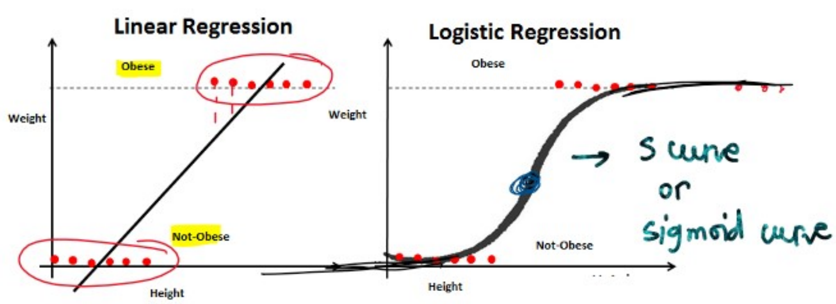

# Activation Function

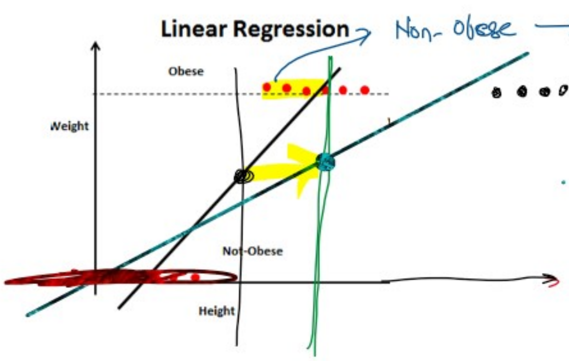
An activation function in a neural network is a mathematical function applied to the output of neuron before passing it to the next layer.

## Sigmoid Squishification

$$\sum_{i=1}^{784} w_i a_i + b$$



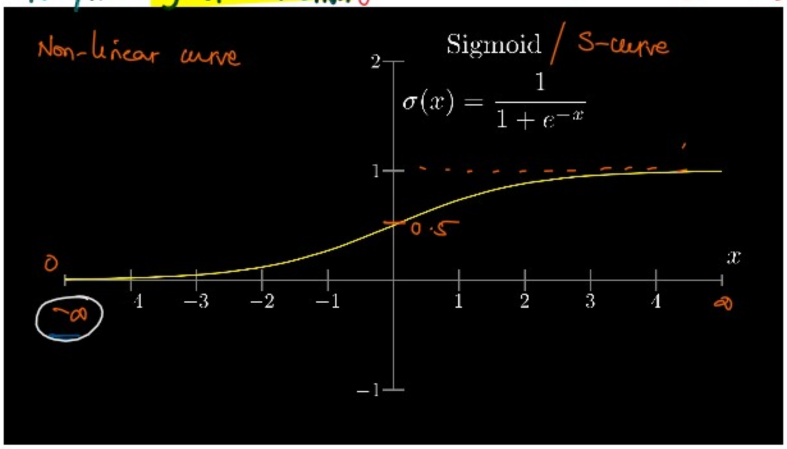




→ that's where linear regression fails.

In general, sigmoid function is used for binary classification.

## Graph- Sigmoid function




$\sigma(x)$:

$\sigma(\infty) \rightarrow 1$

$\sigma(-\infty) \rightarrow 0$

$\sigma(-1000) = ?$

A) −1000

B) −1

C) 0

D) 1

$$\lim_{x \to -1000} \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(-1000)}} = \frac{1}{1+e^{1000}} = \frac{1}{\infty} \rightarrow 0$$

$e^{100} = 2.68 \times 10^{43}$

### Applying sigmoid function to the weighted sum

$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \cdots w_{784}a_{784})$

greater than 8'

9.2

$$\sigma(9.2) = \left(\frac{1}{1+e^{-9.2}}\right) = \frac{1}{1+0.000101} = \frac{1}{1.000101} =$$ 1/1.000101=0.999899 $\triangleq$ 1 → light up the neuron.


### Applying sigmoid function to the weighted sum with bias (−10)

$$\sigma(9.2-10) = \sigma(-0.8) = \frac{1}{1+e^{-(-0.8)}} = \frac{1}{1+e^{0.8}} = \frac{1}{1+2.225} =$$ 1/(1+2.225)=0.3101

Refer this later → very good link
https://deeplizard.com/resource/pavq7noze2