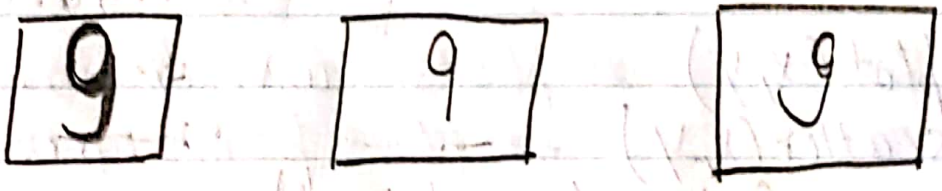
Deep Learning

Deep learning is a subset of machine Learning which in turn subset of Artificial intelligence.

A.I is a technique that enables a machine to mimic human behavior.

Machine Learning is a technique to achieve A.I algorithms trained with data and finally, deep learning is a type of machine Learning inspired by structure of human brain in terms of deep learning, this structure is called artificial neural network

Working of Neural Networks :)

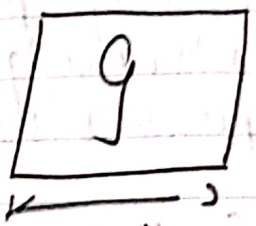
eg: 

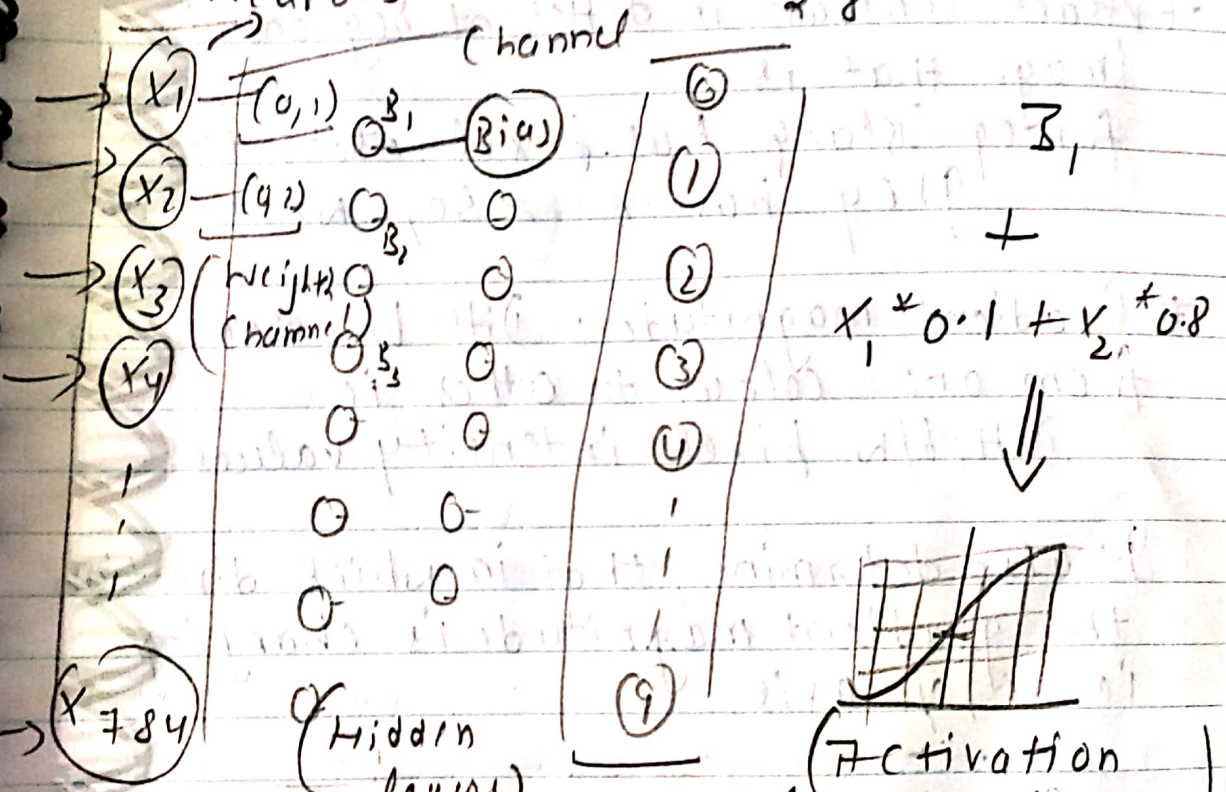
there are digits of same number but, they are not identical, human brain can easily recoginise it, but for computer had to recoginise thats where deep learning comes in,

There is a neural network trained to identify handwritten digits, each number

is present as an image of 28 x 28 pixels

$28 \times 28 = 784$ px





eg Customer, Support, Medical care, Self D.ri,Ca

HOG - Intuition

Histrogram of oriented Gradients is the one of popular feature extraction techiniques in the the field of computer vision.

- Extract Contrast in different regions of image that is
  for eg Black has pixel intensity '0'
  grey has " " so, White '00'

Gradient magnitude : Diff. b/w transition from one colour to other i.e.
Diff. b/w pixel intensity values.

It also determine, that in which direction the gradient magnitude is changing i.e. x, y axis.

Gradient Magnitude + Direction = feature Vector (f.v)

Using f.v, we can analyzes images and do further processing

HOG feature, Vector Calculation

Implementing HOG using tools like openCV is extremely simple, it just a few lines of codes since we have a predefined function called `hog` in the `skimage.feature` Library.

Process of Calculating the HOG :)

① eg Consider a image of size (say 180x280)

Step ① Preprocess that data (64x128)
We need to preprocess the image and bring down the width to height ratio 1:2, the image should preferably be 64x128. This is because we will be dividing image in to 8x8 and 16x16 patches to extract the features.

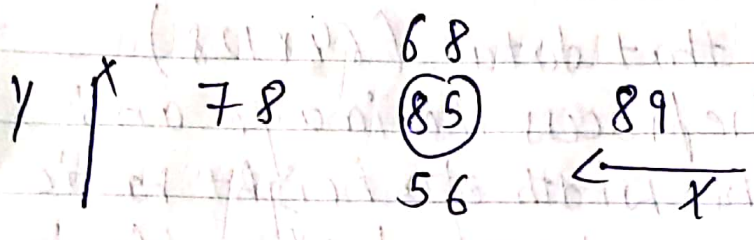
Step ② Calculating Gradients (direction x & y)

The next step is to calculate gradient for every pixel in the image.
Gradients are small change in x and y direction

for eg: We get a matrix for a patch.

152	68	125
78	(85)	89
214	56	200

in order to calculate gradient for x & y we will use the values only which are up & down, avoiding diagonal values therefore matrix looks like.

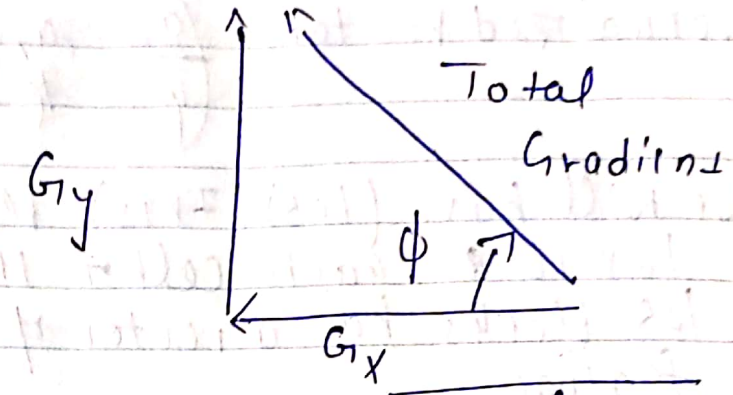


$$\begin{bmatrix} G_x = 89 - 78 = 11 \\ G_y = 68 - 56 = 8 \end{bmatrix}$$

This process will give us two new matrices, one storing gradient in x-direct & y-direction. [The magnitude of intensity]

Step (3) Calculating Magnitude & orientation.

Using the gradients, we calculate, mag & orientation.



Total Gradient Magnitude = $\sqrt{(G_x)^2 + (G_y)^2}$

eg. $\sqrt{(11)^2 + (8)^2} = 13.6$

Orientation, $\boxed{\tan(\phi) = \frac{G_y}{G_x}}$

Step ④ Calculate Histogram of Gradients in 8×8 cells (9×1)

We divide the image into 8×8 cell & generate histogram, get 9×1 matrix for each cell. once, we generated HOG for 8×8 patch the next step is normalize the Histogram.

Step ⑤ Normalizie gradient in 16×16 cell (36×1)

$V = [a1, a2, a3, \ldots a36]$ (36×1)

$V = \sqrt{(a1)^2 + (a2)^2 \ldots (a36)^2}$

Normalized vector = $\left(\frac{a_1}{k} \quad \frac{a_2}{k} \quad \cdots \quad \frac{a_{36}}{k}\right)$

We will have (105) (7x15) blocks of 16x16 , each cell of this 105 blocks has a vector of 36x1 os feature.

Total feature → 105x36x1 = 3780 features

---

importing required Libraries

```
from skimage.io
    import imread, imshow

from skimag.transform
    import resize

fromskimage.feature
    import hog

fromskimage.
    import exposure

import matplotlib.pyplot as plt
%matplotlib inline
```

Reading the image
```
img = imread('path.jpg')
imshow(img)
print(img.shape)
```

Resizing image ⟶ to (64x128)
```
resized_img = resize(img, (128,64))
imshow(resized_img)
print(resized_img.shape)
```

→ By using hog function from skimage.features directly, so it will calculate gradient, magnitude & orientation & return to feature matrix.

[if we set parameter 'visualize=True', it will return an image of the HOG]

Creating hog features
```
fd, hog_image = hog(resized_img, orientations=9
    , pixel_per_cell=(8, 8), cell_per_block=(2,2)
    visualize=True, multichannel=True)
```

Understanding of hyperparameters!)

- The orientations are number of buckets we want to create, since 9 want to have 9x1 matrix, so 9'll set it to 9.
- pixels_per_cell defines the size of the cell for which we create the histogram, We used 8x8 cell.
- cells_per_block → means size of block over which we normalize the histogram Here, we mention the cell per blocks and not no. of pixel, so instead of writing 16x16, we use 2x2 here.

The feature matrix from function is stored in variable (fd), image is stored in hog_image.