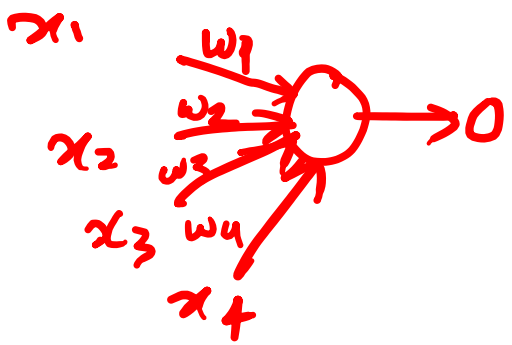
DEEP LEARNING

Trainer : Dr. Darshan Ingle

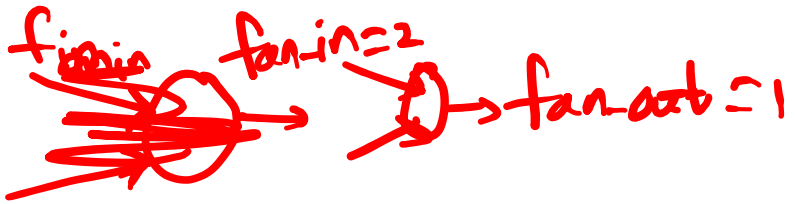
Weight Initialization

Imp Pointers:

1. Weight should be small.
2. Weights should not be same.
3. Weights should hv a good Variance.



Weight Initialization



① Uniform Distribution

$$W_{ij} \sim \text{Uniform}\left[\underbrace{\frac{-1}{\sqrt{fan\_in}}}_{a}, \underbrace{\frac{1}{\sqrt{fan\_in}}}_{b}\right]$$

Weight Initialization

② Xavier / Glorot (Sigmoid)

ⓐ Xavier Normal:

$$W_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{(fan\_in + fan\_out)}}$$

ⓑ Xavier Uniform:

$$W_{ij} \sim U\left[\frac{-\sqrt{6}}{\sqrt{fan_{in} + fan_{out}}}, \frac{\sqrt{6}}{\sqrt{fan_{in} + fan_{out}}}\right]$$

Weight Initialization

(3) He init (Relu)

(a) He Uniform:

$$W_{ij} \sim U\left[-\sqrt{\frac{6}{fan_{in}}}, \sqrt{\frac{6}{fan_{in}}}\right]$$

(b) He Normal:

$$W_{ij} \sim N(0, \sigma)$$

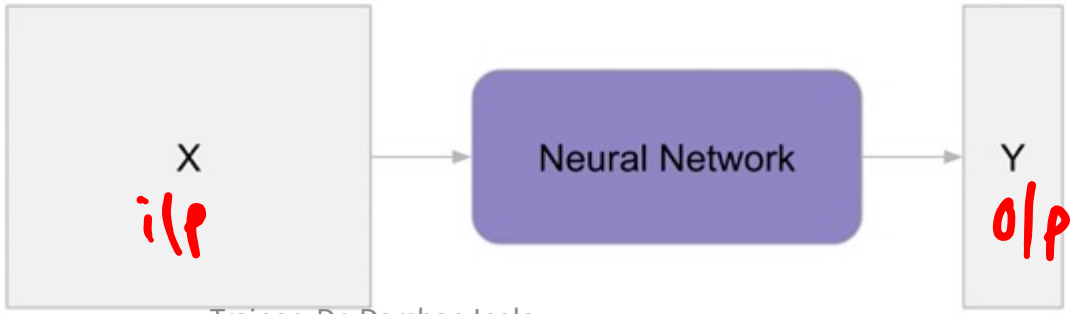
$$\sigma = \sqrt{\frac{2}{fan_{in}}}$$

How to represent Images?

NN really shine on Unstructured data i.e. images, sound, text.

Why is this a challenge?

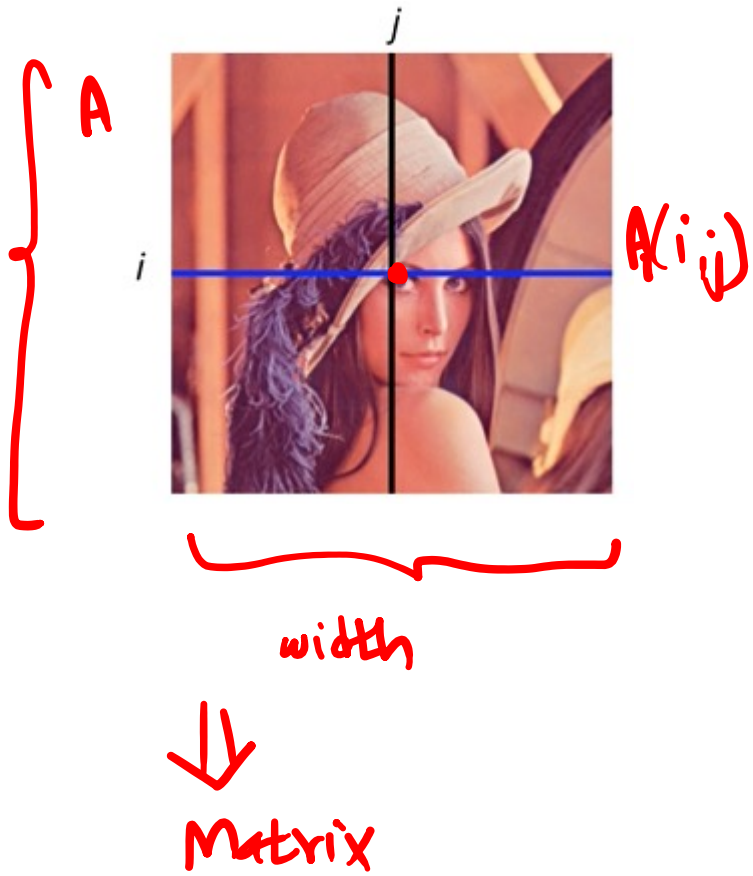
- Previously we discussed the famous rules:
  - ML is nothing but a Geometry Problem
  - All data is the same
- Ex:
  - y = Pass/Fail , x=(# hours studying, # hours video games)
  - y = Salary, x = (years of experience, degree type)
- The question now is if your input data X is an image.
- How does that fit into this picture.



How are the images stored?

- Consider the famous Lenna/Lena image

height



A(i,j) stores color at exact position (i,j)

Then
Q.) How do we store colors?

Matrix

How do we store colors?

Primary Colors: Red, Blue, Yellow.

Any other color can be made by combing primary colors.

Red + Blue = Purple

Blue + Yellow = Green

Red + Yellow = Orange

All primary colors together = White.

RGB (Red Green Blue)

Additive color mixing with red, green and blue additive primary colors.

How do we store colors?

Color is not just a no., it is 3 numbers

- how much red
- how much green
- how much blue

By mixing R, G, B in different proportions, we get different colors.

Additive color mixing with red, green and blue additive primary colors.

How are images stored?

- We need 3 dimensions: Height, Width, Color
- Color dimension always has size 3, corresponding to RGB channels.

A(i,j,k) stores the value of i^{th} row, j^{th} col, k^{th} color.

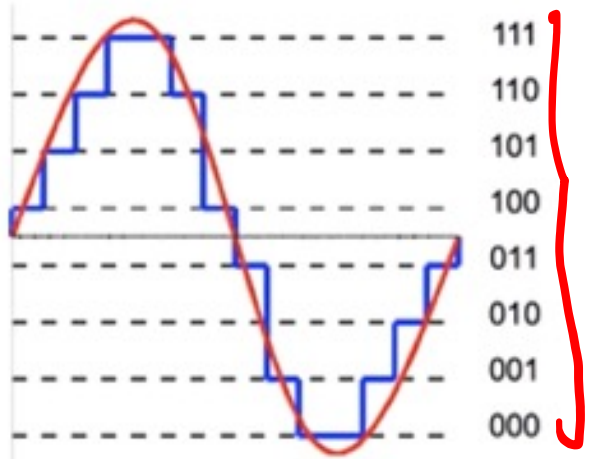
where k=red/green/blue.

Quantization

- Physically, color is light (measured by light intensity)
- It is a continuous value.
- ∴ There is an infinite no. of possible values.
- Unfortunately, computers don't hv infinite precision.
- More precision, requires the image to take up more space.
- Scientists hv figured out that 8bits (1 byte) is good enough.

$2^8 = 256$ possible values (0, 1, 2, ..., 255)

R.G.B = $2^8 \cdot 2^8 \cdot 2^8 = 16.8$ million possible colors.



How much space does a 500x500 image take up?

Soln: $500 \times 500 \times 3 \times 8 = 6$ million bits i.e. 750,000 bytes

i.e. 732 KB which is quite large for 500x500 image.

∴ We hv JPEG compression that allows us to compress images.

Grayscale images

- Images that do not have colors can be simplified
- We can call them “grayscale” because each pixel value can only be black, white or some shade of gray
- Black = 0, White = 255
- Only requires a 2D array(height, width)

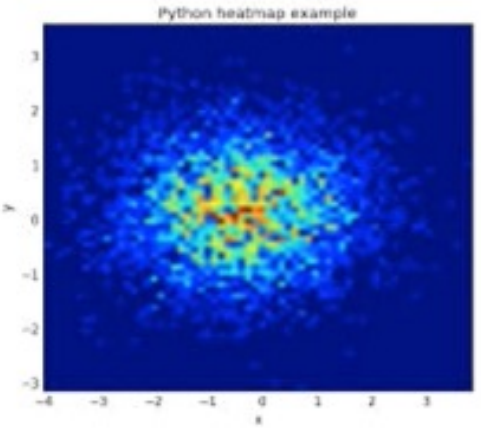
Plotting Grayscale images in matplotlib

Blue = cold (min value)
Red = hot (max value)

plt.imshow(arr_img) → Heatmap

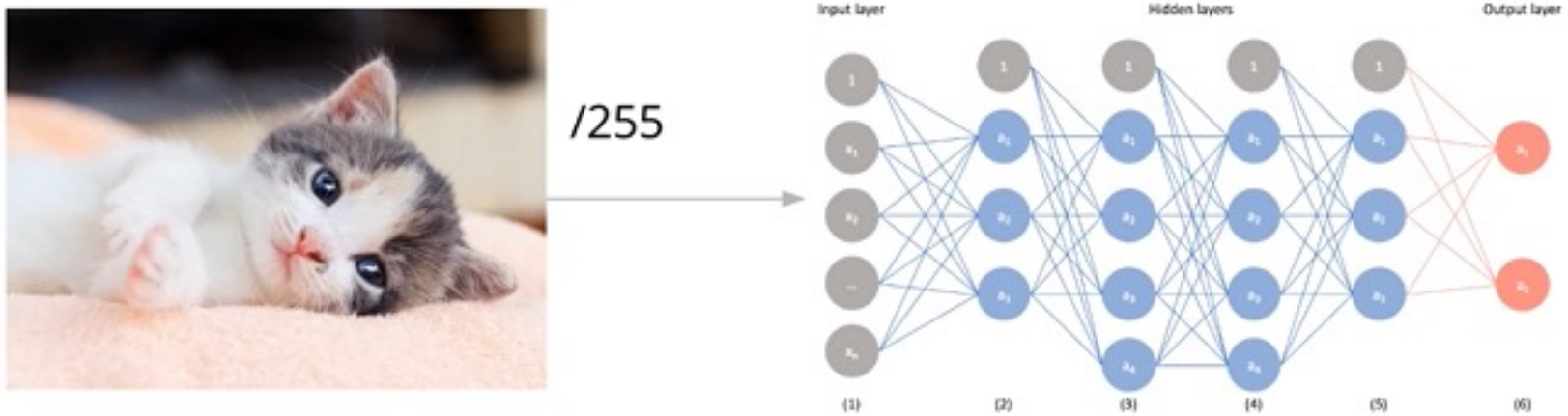
∴ we must use:

plt.imshow(arr_img, cmap='gray')



Images as input to Neural Networks

- Neural Network don’t like values on a large scale (0 to 255)
- It is more conventional to scale them up between 0 to 1
- These are not centered around 0 – there are always exceptions in Deep Learning!
- This is actually a convenient representation because they can (in certain scenarios) be interpreted as probabilities



Another exception

VGG – Visual Geometry Group.

VGG – won the ImageNet contest.

VGG – does not scale i/p data, but it does subtract the mean across each color channel. So the images are centered around 0, but the range is still 256.

Code:

tf.keras.applications.vgg16.preprocess_input

Images as input to Neural Networks

NN: expects an input X of shape N×D where N = #samples / rows / observations
D = #features / cols

Single image: H×W×C (height, width, color)

Q. A single image does not hv 'D' features.

A full dataset of images – shud hv the shape: N× H×W×C
60000 × 32×32×3

/255

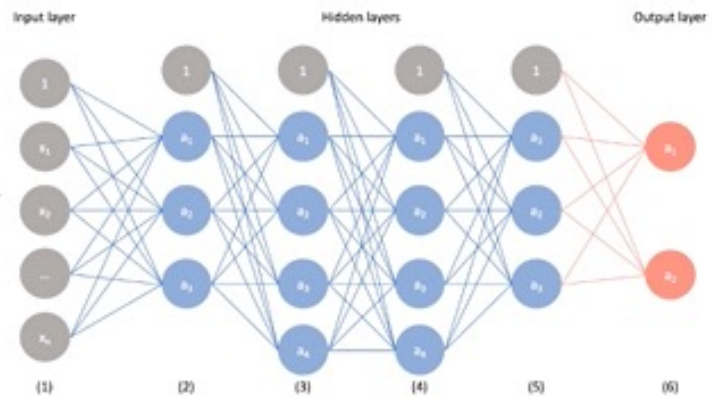


Image to Feature Vector

Lets consider Black & White img.

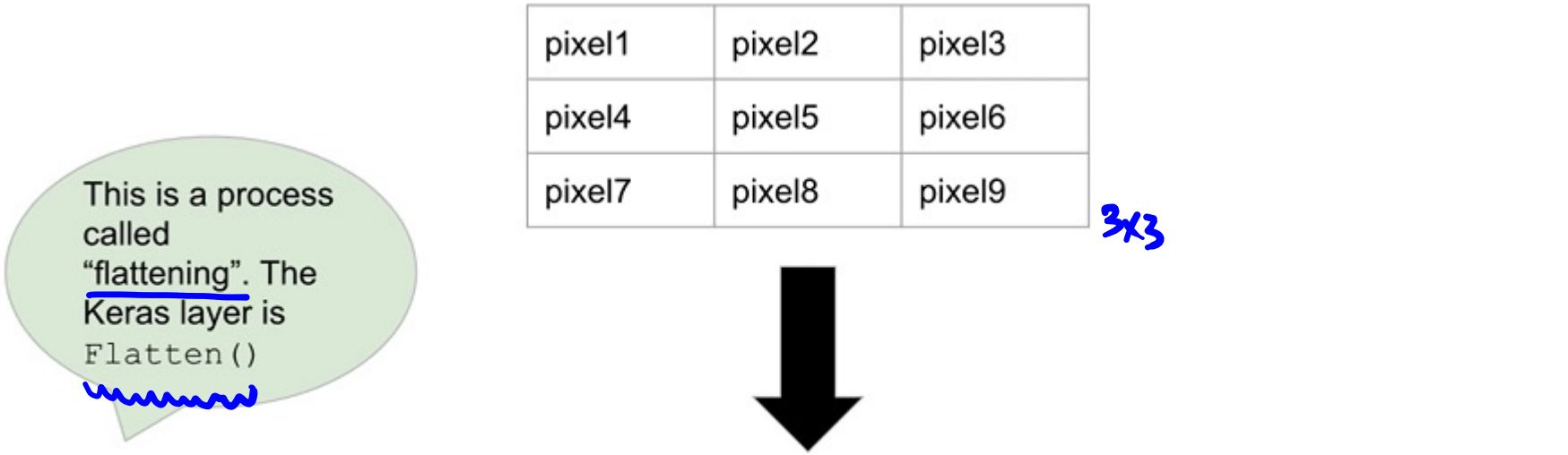
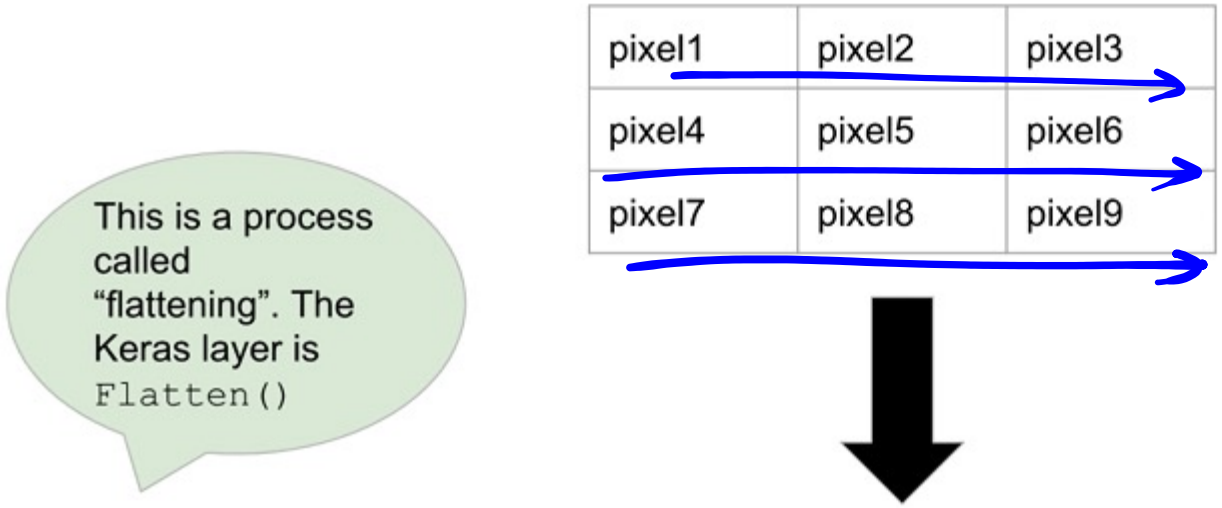


Image to Feature Vector

L → R, T → B



(1 × 9)

pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9

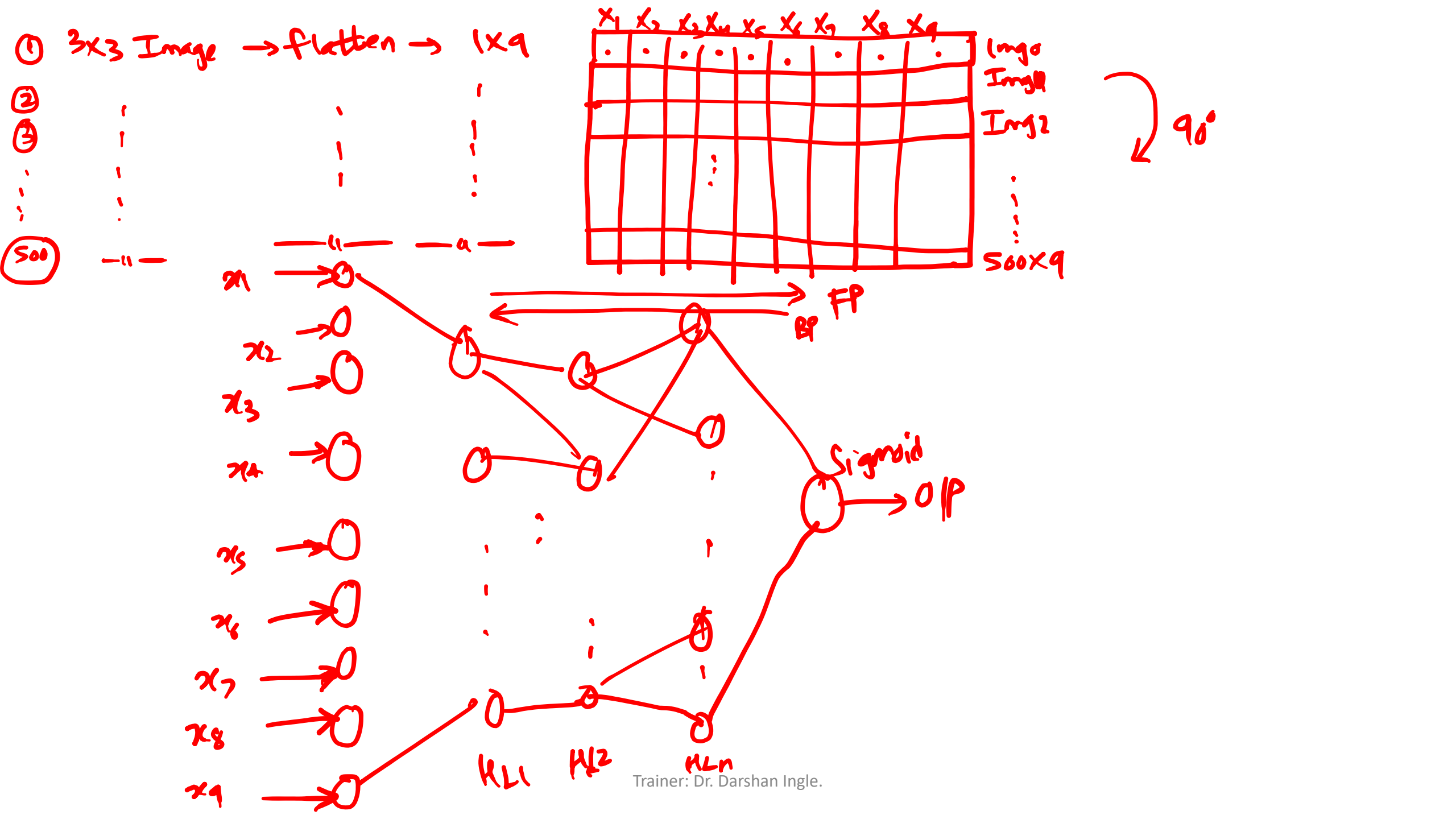
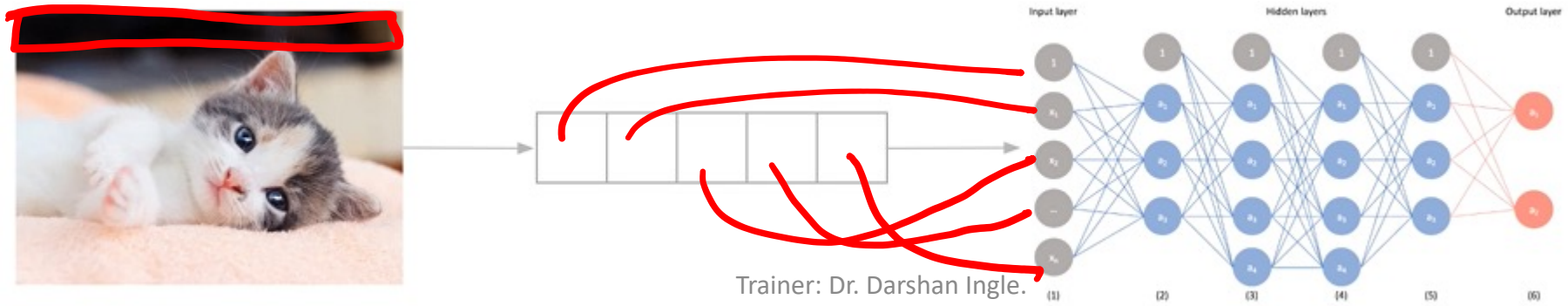

3x3 Image → flatten → 1x9
500
Img0
Img1
Img2
500x9
90°
FP
BP
Sigmoid
O/P
HL1
HL2
HLn

Image to Feature Vector

All data is the same.



ANN for Image Classification

ANN Code Preparation

1. Load in data: MNIST dataset. 10 digits (0,1,2,3,4,5,6,7,8,9)
   - already included in TF.
2. Build the model: Sequential Dense layers ending with multiclass logistic regression.
3. Train the model!
4. Evaluate the model
5. Make predictions

Model-Agnostic

ANN for Image Classification

- Load in the data

tf.keras.datasets

(x_train, y_train), (x_test, y_test) = mnist.load_data()

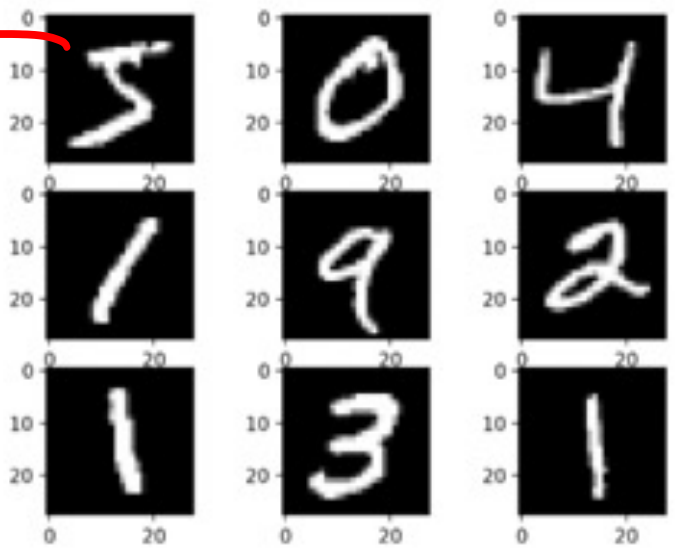
x_train.shape = N x 28 x 28

y_train.shape = N

x_test.shape = N_{test} x 28 x 28

y_test.shape = N_{test}

Hand written digits



each image is (28x28) = 784 pixels

Grayscale (pixels values are 0...255)

↓

we scale them to (0-1)

↓

flatten them using tf.keras.

Refer NB “3 ANN_MNIST Image Classification.ipynb »