How to represent Images?

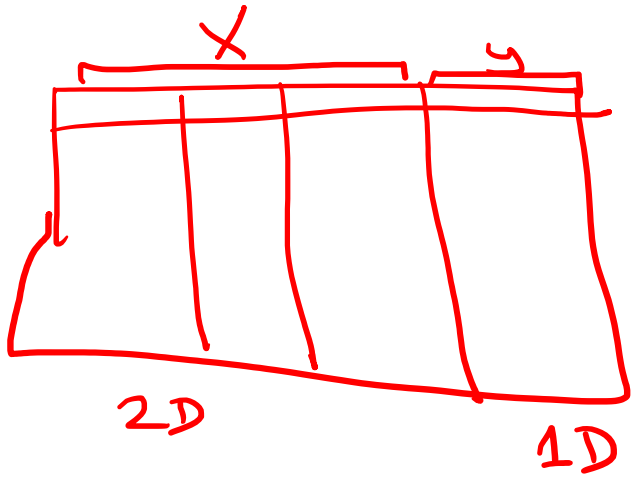
Discuss data

Data → NN (I know it)

Image → NN (I dnt know it)

NN — Shine on Unstructured data (image, sound, textetc)

ML



lr.fit(X, y)

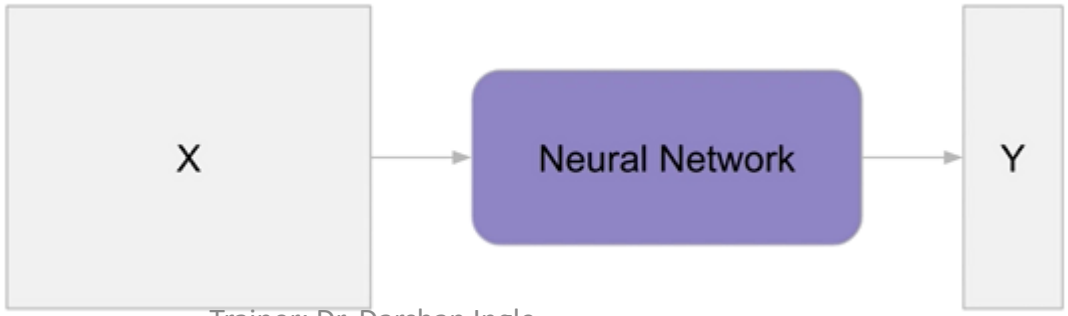
Geometry

All the data is the SAME

- kmeans
- KNN
- RF & DT
- LR
- Log.R.

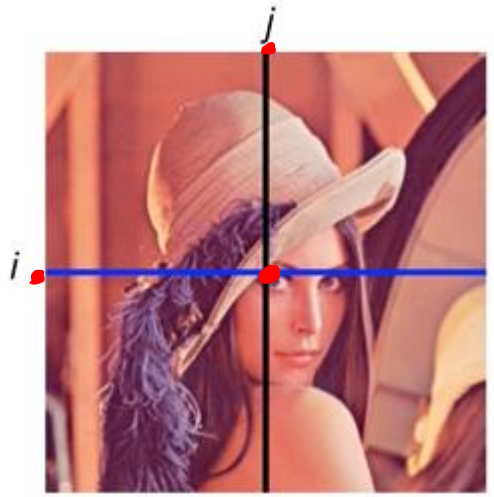
Why is this a challenge?

- 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



2 property: height & width

Image --> can be stored in the form of a MATRIX

A(i,j) = store the color at that position i.e. i^{th} row & j^{th} col.

$A = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$

How do we store colors?

Primary Colors → Red – Blue – Yellow Green

Any other color can be made using Primary Colors

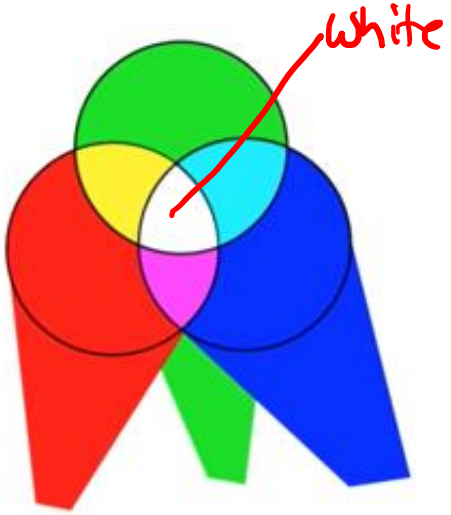
Red + Blue = Purple

Blue + Yellow = Green

Red + Yellow = Orange

Red · Green · Blue

R · G · B



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

R+G+B

How do we store colors?

So a color is not just a number, it is 3 numbers.

Color = how much of red + how much of Green + how much of Blue

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

How are images stored?

3D

3 - Dimensions: Height, Width, COLOR

Color dimension has size = 3 (RGB)

A(i, j, k)

where i = row no

j = col. no

k = red, green, blue

Quantization

Color is light, measured by light intensity. (continuous value)
↓
infinite no. of possible values
↓
COMPUTER do not hv infinite precision

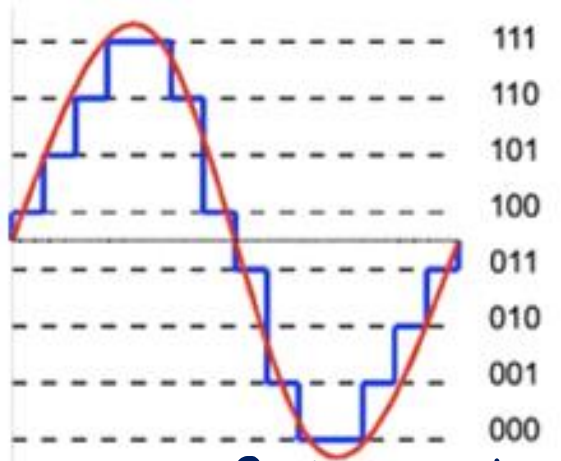
More precision require the image to take up larger space.

Researchers : 8 bits (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 500×500 image take up?

Soln: $500 \times 500 \times 3 \times 8 = 6$ million bits = 7,50,000 bytes ie. 732kB is quite large for a 500×500 image.

∴ We go for Image Compression (JPG)

Grayscale images

Images which do not have color are called

- 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)

0-255

Each pixel can be only black/white/some shade of grey

White = 255

Black = 0

0 – 255 – Some Shade of Grey.

500x500 x3 colors

500x500 (2D)

Grayscale = color 0-255

Plotting Grayscale images in matplotlib

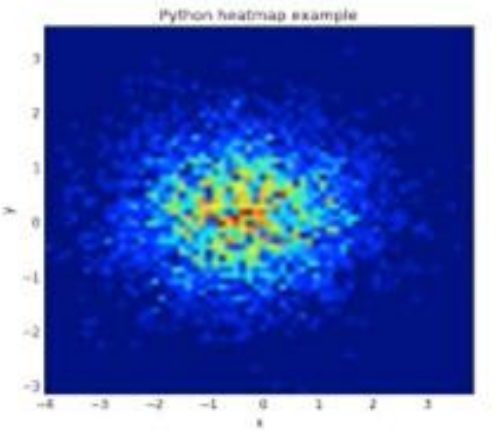
Blue = cold (min)
Red = Hot (max)

plt.imshow(array2D)

Heat Map (not True colors)

For Grayscale images,

plt.imshow(array2D, cmap = 'gray')



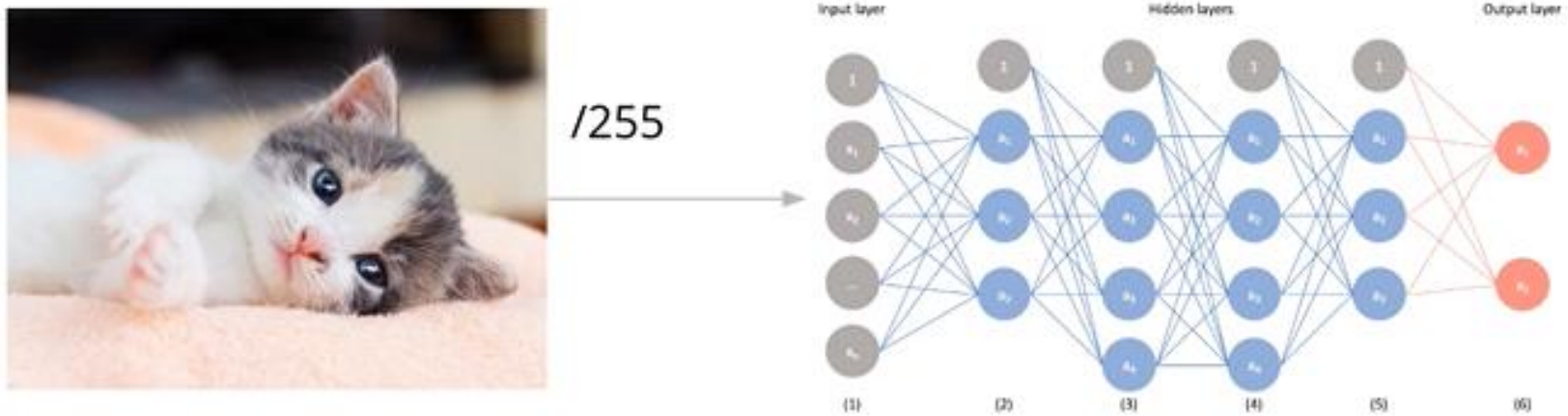
Images as input to Neural Networks

$2^8 \cdot 2^8 \cdot 2^8$

NN do not like values on a large scale (0 to 255)

Scale them such that they are centered around zero.

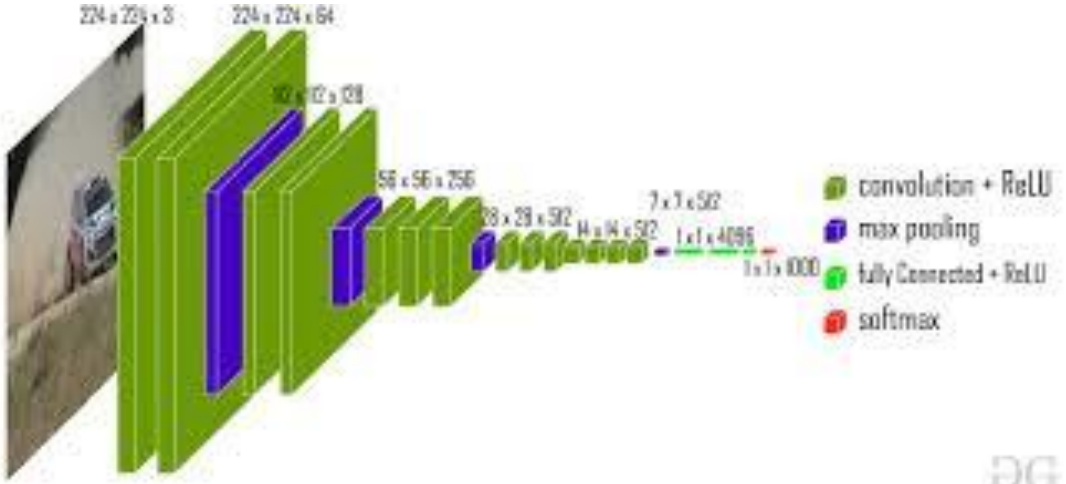
- Neural Network don't like values on a large scale (0 to 255)
- It is more conventional to scale them up between 0 to 1
- These are not centered around 0 – there are always exceptions in Deep Learning!
- This is actually a convenient representation because they can (in certain scenarios) be interpreted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



Another exception

Computer Vision → VGG

won ImageNet Contest



Subtract mean across color channel X −X.mean()

Images as input to Neural Networks

N = #rows / samples #D = features / cols.

NN expect X to be a 2D shape Matrix: N x D

Single image: H x W x C (it doesn't have 'D' features).

Dataset: 1000 images colored (N)

$N \times H \times W \times C$

1000 height width color

/255

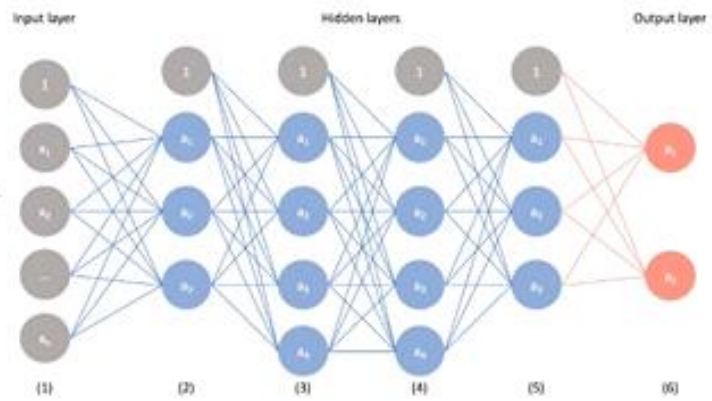


Image to Feature Vector

One Img. of a Dog (Grayscale Image)

pixel1	pixel2	pixel3
pixel4	pixel5	pixel6
pixel7	pixel8	pixel9

2D

3x3

This is a process called "flattening". The Keras layer is `Flatten()`

Flattening

pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9

1D

Image to Feature Vector

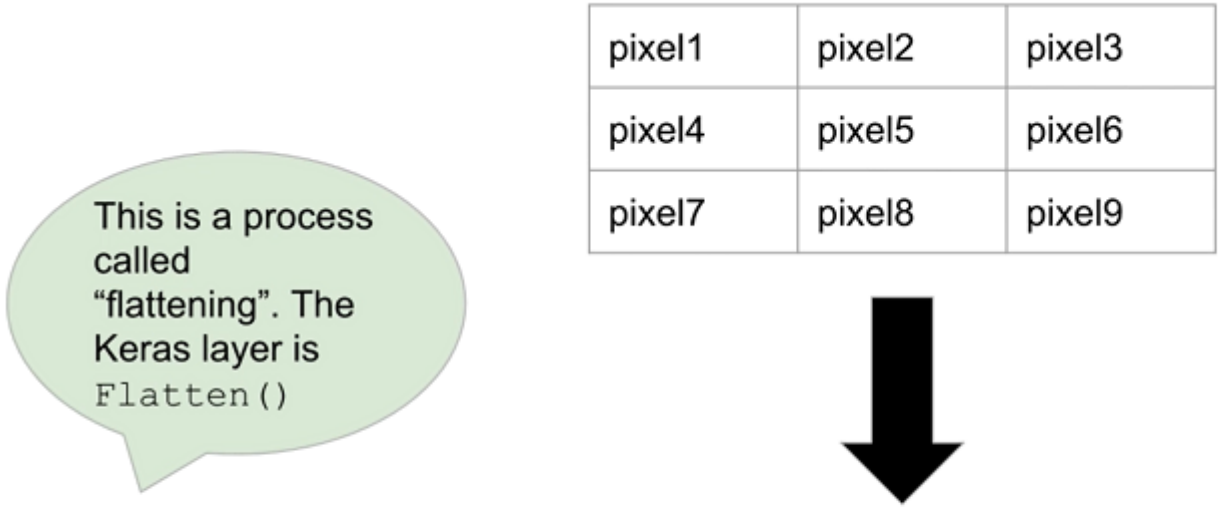
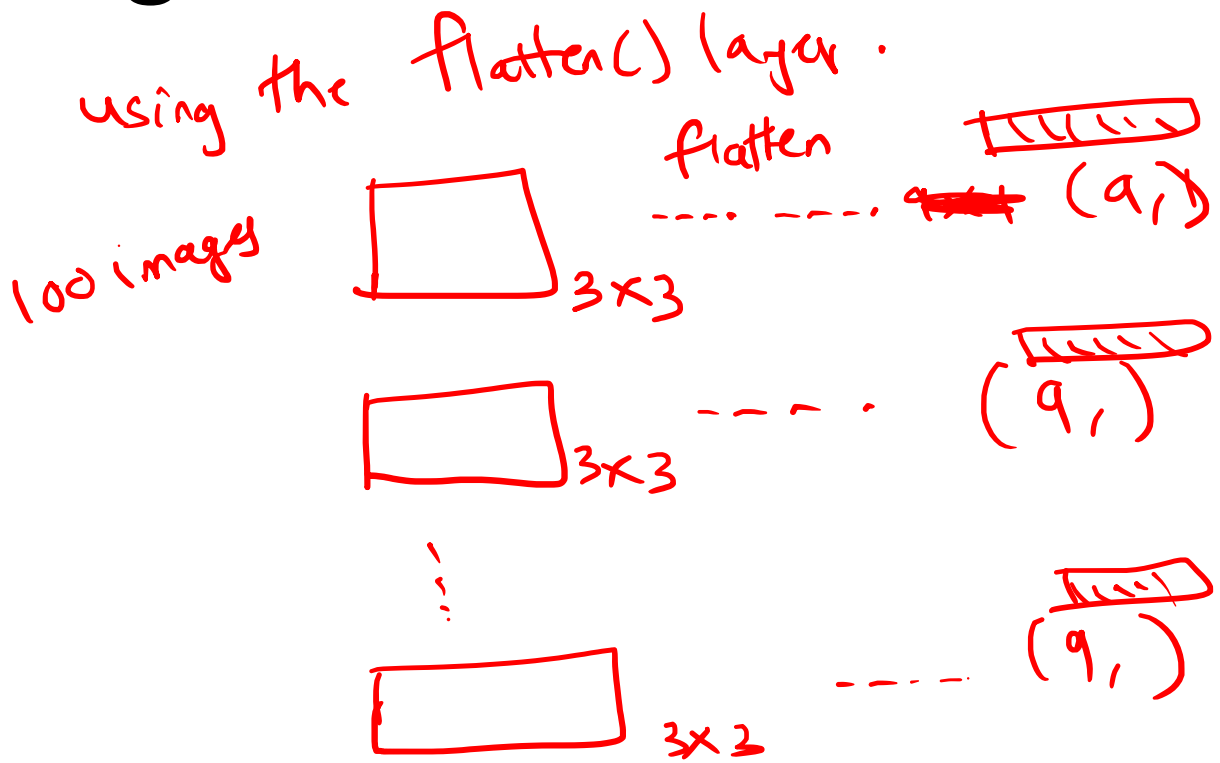
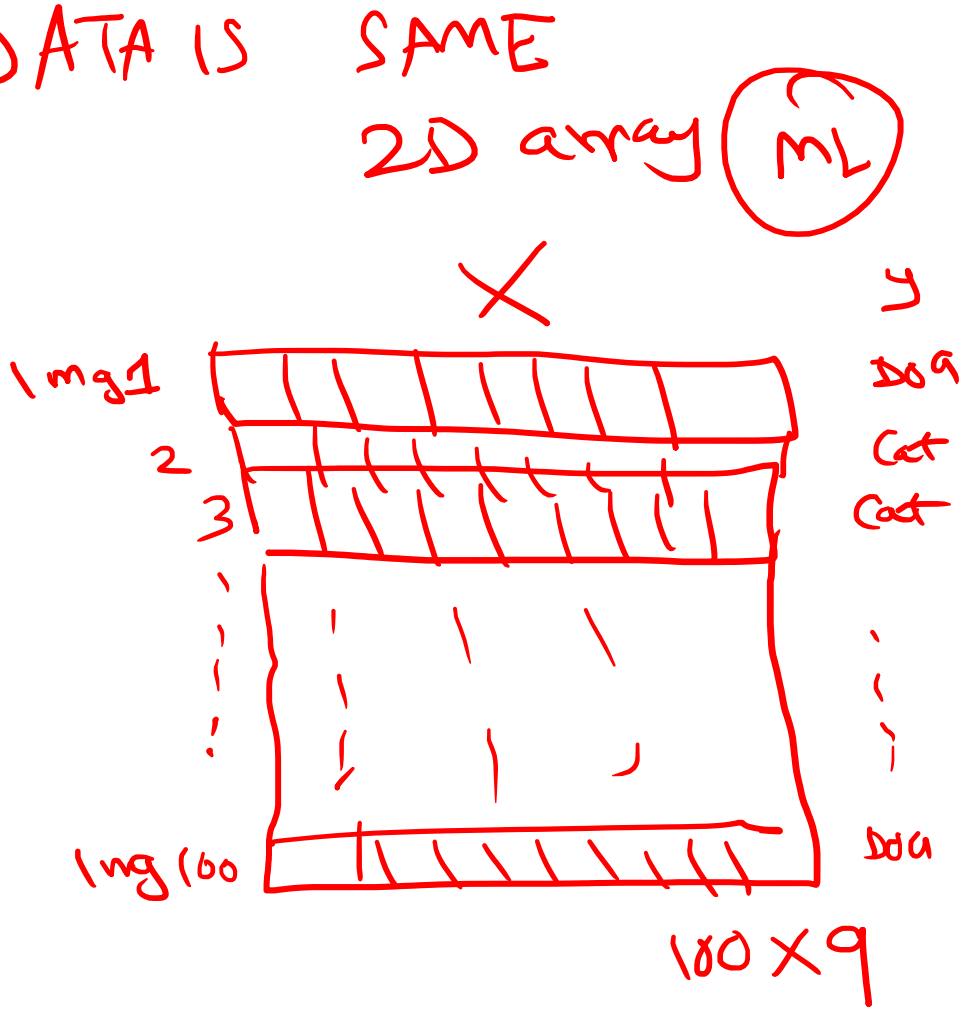
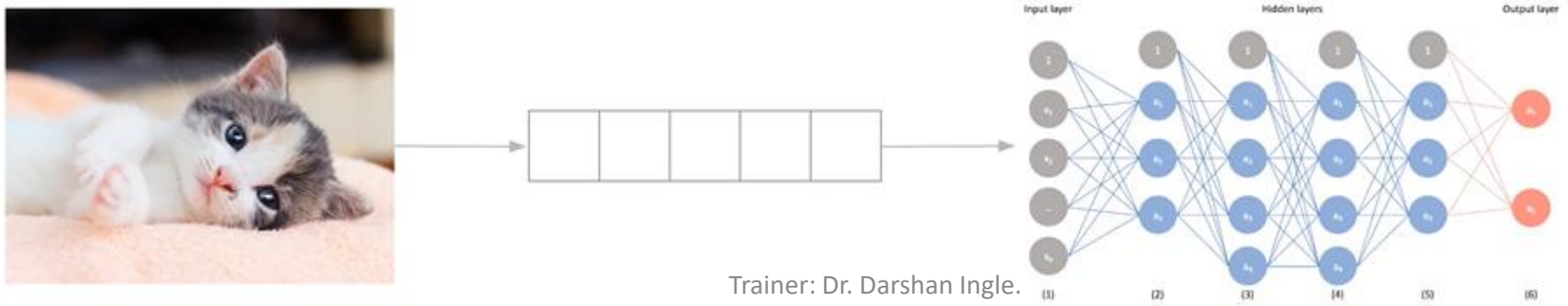


Image to Feature Vector







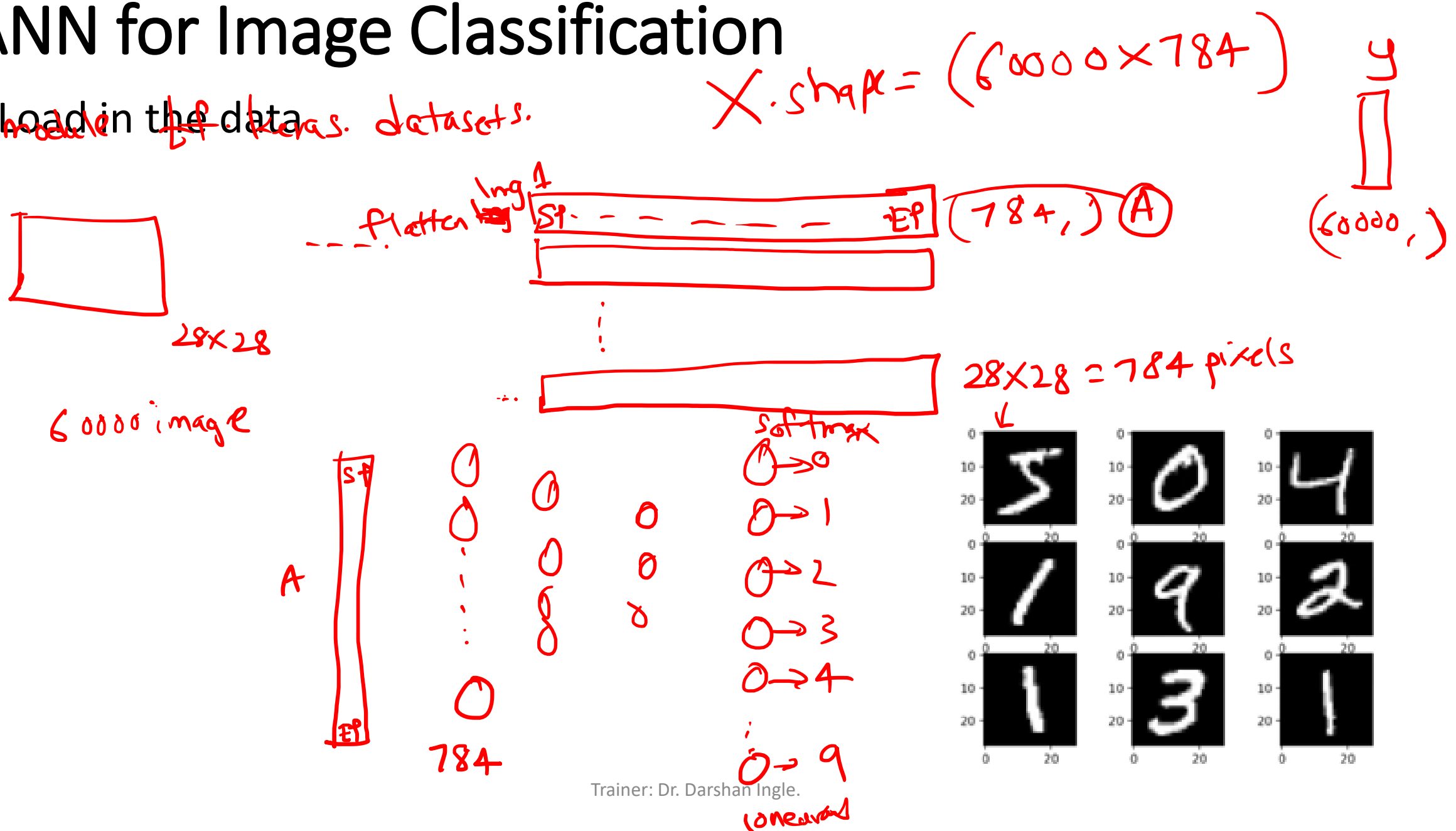
ANN for Image Classification (MNIST Dataset)

ANN Code Preparation

1. Load the data
   MNIST (images of Handwritten nos are stored) (0-9) 10 digits
   included in TF.
2. Build the Model
3. Train the model
4. Evaluate the model
5. Make predictions

ANN for Image Classification

- Load in the data



Refer NB “3 ANN_MNIST Image Classification.ipynb »