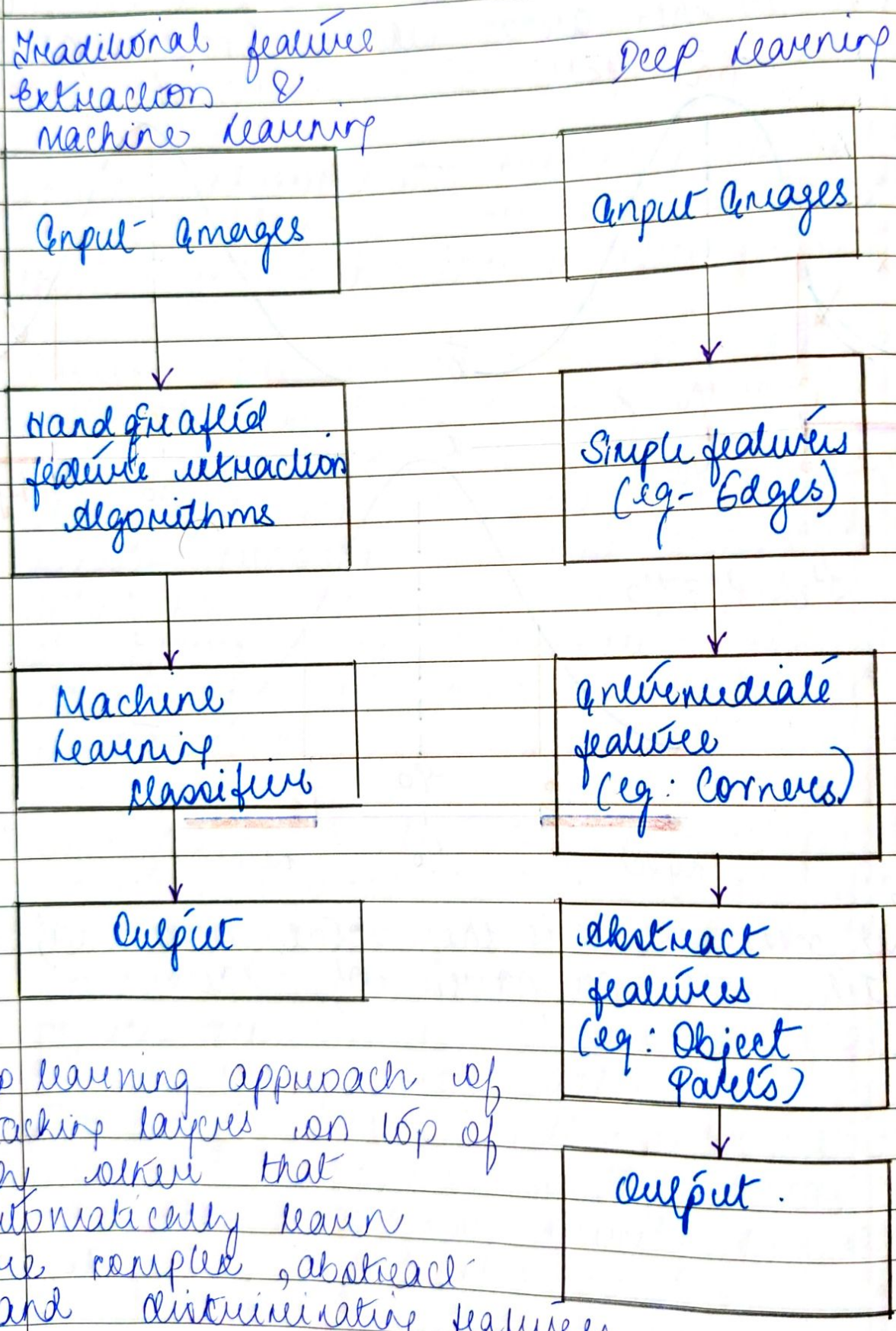
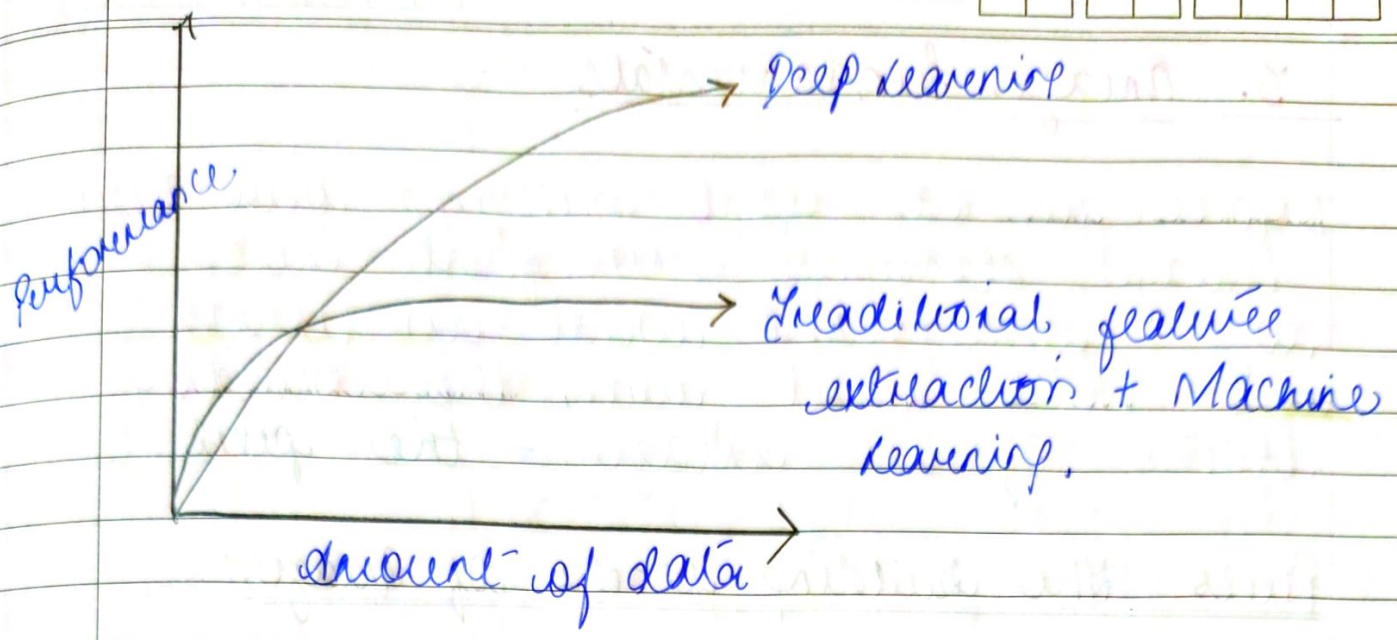
DATE

What is Deep Learning



→ Deep learning approach of stacking layers on top of each other that automatically learn more complex, abstract and discriminating features

DATE



→ As the amount of data available to deep learning algorithm increases, accuracy does as well, substantially outperforming traditional feature extraction + Machine learning approaches.

- Deep learning and especially convoluted neural networks instead of hand-defining a set of rules and algorithms to extract features from an image, these features are instead automatically learned from the training process.

- Using deep learning, we try to understand the problem in terms of a hierarchy of concepts. Each concept builds on top of the others. Concepts in the lowest layers level layers of the network encode some basic representations of the problem. This hierarchical learning allows us to completely remove the hand-designed feature extraction process and as end-to-end encoders.

classmate PAGE

DATE

3. Image fundamentals

Before we can start building our own image classifier, we first need to understand what an image is? We'll start with the building blocks of an image - the pixel.

Pixels: The building blocks of images:

Pixels are the raw building blocks of an image. Every image consists of a set of pixels. There is no finer granularity than the pixel.

Normally, a pixel is considered the "color" or the "intensity" of light that appears in a given place in our image. If we think of an image as a grid, each square contains a single pixel.

- Example: A image has a resolution of 1000 x 750, meaning that it is 1000 pixels wide and 750 pixels tall. We can conceptualize an image as a (Multidimensional) matrix. In this case, our matrix has 1000 columns (Width) and 750 (Row) (the height). Overall, there are 1000 x 750 = 750,000

classmate

PAGE

DATE

total pixels in our image.

- Most pixels are represented in two ways
  1. Grayscale / Single channel
  2. Color.

Greyscale

- On the grayscale image, each pixel is a scalar value between 0 and 255, where 0 corresponds to "black" & 255 being "white".
- Values between 0 & 255 are varying shades of grey, where values closer to 0 are darker and values closer to 255 are lighter.

Color

- Color pixels, however are represented in the RGB color space.

- Pixels in the RGB color space are no longer a scalar value like a grayscale / single channel image - instead the pixels are represented by a list of three values - one value for the Red component, one for green & one for Blue.

- To define a color in the RGB color model, all we need to do is define the amount of red, Green & Blue contained in a single pixel.

PAGE

classmate

DATE

- Each Red, green and Blue channel can have values defined in the range [0, 255] for a total of 256 shades: where 0 indicates no representation and 255 demonstrates full representation.
- Given that the pixel value only needs to be in the range [0, 255], we usually use 8-Bit unsigned integers to represent the intensity.

Forming an image from channels

An RGB image is represented by three values, one for each of the Red, green and Blue components, respectively. We can conceptualize an RGB image as consisting of:

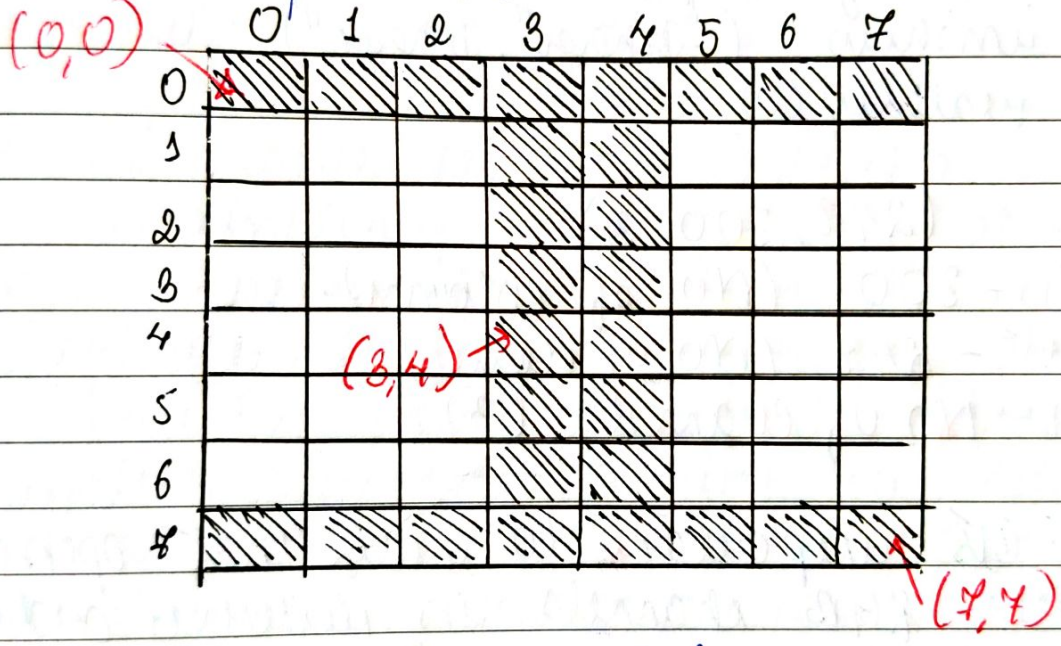
→ Three independent matrices of width W, height H, one for each of the RGB components.

→ We can combine these three matrices to obtain a multi-dimensional array with shape: W x H x D, where D is the depth or no of channels (for RGB color space, D = 3).

DATE

3.2 The Image Co-ordinate System

An image is represented as a grid of pixels. To make this point more clear, imagine one grid as a piece of graph paper. Using this graph paper, the origin point (0,0) corresponds to the upper-left corner of the image. As we move down and to the right, both the x and y values ~~are~~ increases.



The letter 'T' placed on a piece of graph paper. Pixels are accessed by their (x,y) co-ordinate, where we go 'x' columns to the right and y rows down;

DATE

Images as numpy arrays:

Images processing libraries such as OpenCV and scikit-image represent RGB images as multidimensional numpy arrays with shape (height, width, depth).

```
import cv2
image = cv2.imread("xamp.png")
print(image.shape)
cv2.imshow("Image", image)
cv2.waitKey(0)
```

Display: (248, 300, 3)
Width - 300 (No of columns).
Height - 248 (No of rows)
Depth - No of channels (3).

Note: It's important to note that opencv stores RGB channels in reverse order while we normally think in terms of Red, green, and Blue, Opencv actually stores the pixel values as Blue, green & Red order

Scaling and Aspect Ratio

Scaling or simply resizing is the process of increasing or decreasing the size of an image in terms of

DATE

width and height. When resizing an image, it's important to keep in mind the **aspect ratio**, which is the ratio of the width to the height of the image. Ignoring the aspect ratio can lead to images that look compressed and distorted.

To prevent the distortion, we simply scale the width and height of an image by equal amounts when resizing an image.

⇒ for some tasks - Maintaining or not maintaining the aspect ratio depends.
⇒ for some datasets you can simply ignore the aspect ratio and squash, distort, and compress your images prior to feeding them through your network.
- On other datasets, it's advantageous to preprocess them first by resizing along the shorter dimension and then cropping the center.