**COMPUTER VISION**

In the recent years, the field of computer vision has found extensive applications. It is used widely in healthcare, agriculture, self-driving cars and many more areas. So, let’s get an overview of how computers can actually “recognise” all the elements in an image.

*How does a computer perceive an image?*

Unlike us humans, computers can’t directly “see” the different components of an image. For example, if given a picture of a dog, we can easily identify the picture, but a computer can’t. What it can see is the different colour intensities of the image. This is what’s called a digital image. Each pixel has a certain intensity, ranging from 0 (co colour) to 255(full colour).

Many images have different colour channels, like the most common RGB. We can split these three colour channels of the image, and look at the different intensities, using python libraries like PIL or OpenCV. OpenCV is more focused on computer vision, while PIL is used for simple image manipulation.

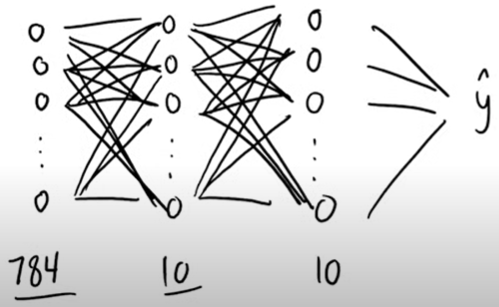
*How does a computer identify the elements in an image?*

Let us take the famous example of cats and dogs. We are given an image and we need to classify it as a cat or a dog. Let us assume we have found out certain “qualities” of the image (the bigger picture of how everything goes on in the back-end will be discussed in a later section).

Then, we compare it with the qualities of the known samples by calculating the Euclidean distance between the points. This algorithm is known as k Nearest Neighbours (kNN). If we set k = 3, and at least 2 of the nearest neighbours are dogs, then the test image is of a dog as well. This technique isn’t much accurate. So, we use something called neural networks to help with classification.

*What is a neural network?*

A neural network is a mathematical model, where several neurons (having some numerical value) are put in a layer, and such layers are connected with adjacent layers via some parameters called weights and biases. Again, taking an example of digit recognition using the MNIST database, we can design a simple neural network as shown.



The input layer has 784 neurons (since the images in the MNIST database are modelled as 28 x 28 pixels), each having some value, depending on the colour intensity (in this case, it’s a grayscale image). The hidden layer has 10 neurons, where each neuron is the linear combination of all the 784 neurons in the previous layer, along with some bias. So, there are 7840 weights and 10 biases.

The relation between the output layer and the hidden layer is also similar. But, before moving on to the output layer, the neurons in the hidden layer are subjected to an activation function such as ReLU or sigmoid. If we hadn’t done this, this would keep the linearity and ultimately, result in the ouput layer being a linear combination of the neurons in the input layer. So, it would have been meaningless to add a hidden layer.

Lastly, the output layer is subjected to the softmax function, since we are expecting the output to be any of the 10 classes from 0 to 9 (multinomial logistic regression). The neural network is trained by using backpropagation and gradient descent.

*Convolutional Neural Networks (CNN)*

If we take the cats and dogs example again, it is much more difficult to classify them by applying a simple multilayer perceptron like we did in the previous example. This is because there are different breeds of the animals, there might be interfering background elements and many other anomalies. This is why we need a much more complex model, like a CNN. The function of a CNN is similar to MLPs, but instead of having simple weights and biases between layers, we use convolution.

This helps us identify different patterns in the image, going from simple ones like edges and corners to more sophisticated eyes, ears and even entire faces. Convolution is done by using different filter matrices and operating them on the image to point out different features. The deeper we go into our CNN layers, the more patterns it would be able to recognise.

Object detection is also an important feature in computer vision. In a given image, a computer can identify all the elements, and put a bounding box around it. This can be done using various algorithms like HOG, RCNN, Haar Cascade, YOLO etc.