**NEURAL NETWORKS**

Neural networks is a machine learning model that makes decisions similar to the human brain. Their working mimics biological neurons, and arrive at conclusions according to that.

A neural network has several layers. The first layer is the input layer, the last layer is the output layer and in between these layers we have hidden layers. Each layer has a number of neurons. Each neuron has a certain number associated with it, it is called activation (between 0 to 1). We can consider each neuron as a function that takes input and gives out a number as output which acts as input for the next layer, that is activations from one layer fire neurons in the next layer. In each layer, each neuron is connected to all the neurons of the previous layer.

**Weights and biases—**

Each connection in the neurons have a specific weight associated to it. They determine what effect each neuron has to the next neuron. The weighted sum of the activations is passed to the next neuron. Now we may want the neurons to activate only after a certain threshold value of the activations. So we add a bias to the weighted sum. Bias tells us how the high the weighted sum needs to be in order to activate the neuron. Also, we want this activation value to be within the range of 0 to 1. Hence we use an “activation function” to squishify the weighted sum. We generally use sigmoid function for squishing the activation value between 0 to 1.

We can express this activation function using matrices.

We can arrange the weights as a matrix, where each row corresponds to the connections of a particular neuron to the previous layer. Multiplying this with the activation matrix, and adding to the bias matrix (each neuron has a specific bias of its own), and then passing it into the activation function, we get the activation for the next layer.

Just like this, each layer passes the activation to the next layer, and finally in the output layer, the neuron with the maximum activation is considered as output. This is what I understood form the video :)

**CONVOLUTIONS**

Convolution is one of the fundamental processes (like addition and multiplication) of combining 2 different arrays to form a 3rd array. To calculate convolution of 2 arrays, we flip one of the arrays and slide it all the way to the left such that just one element aligns. Then we multiply the elements that align, add them together and slide the lower array towards the right by one element. This gives us the convolved array.

**Where do convolutions show up?**

1)Adding two probability distributions—In the example of calculating the probability of generating 2 to 12 on the face of two dice, we use convolution to calculate the same.

2)Moving averages— In each iteration we take average of the data in the little window, overall, this process gives us a smoothed out version of the original data.

3)Image processing—We can blur out an image using convolution. A small matrix with sum of elements equal to 1, is multiplied with the corresponding pixel that it sits on top of. The overall effect is that the colour sort of averages out.. it bleeds into its neighbours which gives a blurry effect. This operation is done by moving the filter throughout the image, pixel by pixel. A more elegant way to bring about the blur is by using gaussian blur.

The smaller array is also called a kernel. Interestingly, by choosing different values inside the kernel, we can detect different features of the image, such as vertical and horizontal edges and also sharpen them.

**Time complexity and FFTs**

The method that we discussed till now, has a time complexity of the order of n^2, but FFTs speed up the process by following a different approach to convolution, and have a time complexity of the order of nlogn.

Interestingly the process of multiplying two polynomials is the same as extracting the coefficients of the two polynomials, form 2 arrays respectively and then find its convolution. With polynomials, we only need finitely many points to determine the coefficients. So we can find the coefficients directly by using sample points.

Further, we simplify this by using complex roots of unity(fast fourier transform). Whenever we have two different arrays of numbers, we first compute the fast-fourier transform of them, then multiply them pointwise, and then do an inverse fft. This gives us the convolution of the two arrays.

Also we see that multiplication is also a type of convolution, and we can perform large multiplications faster by using convolution rather than our traditional method.