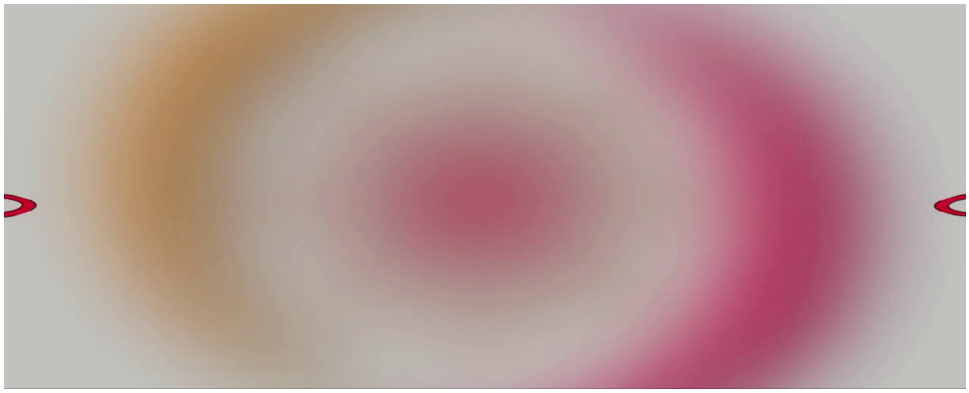
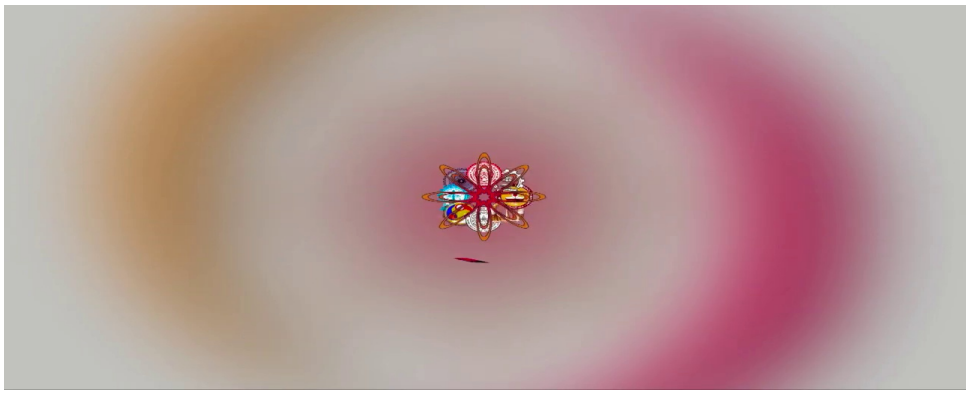
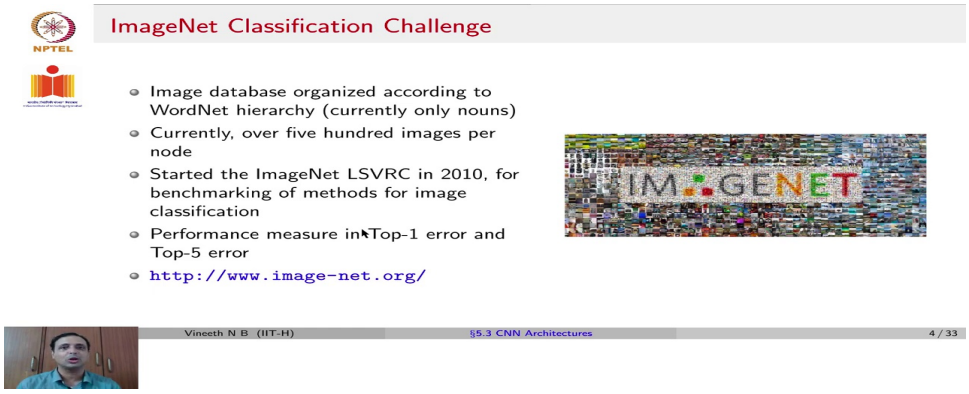
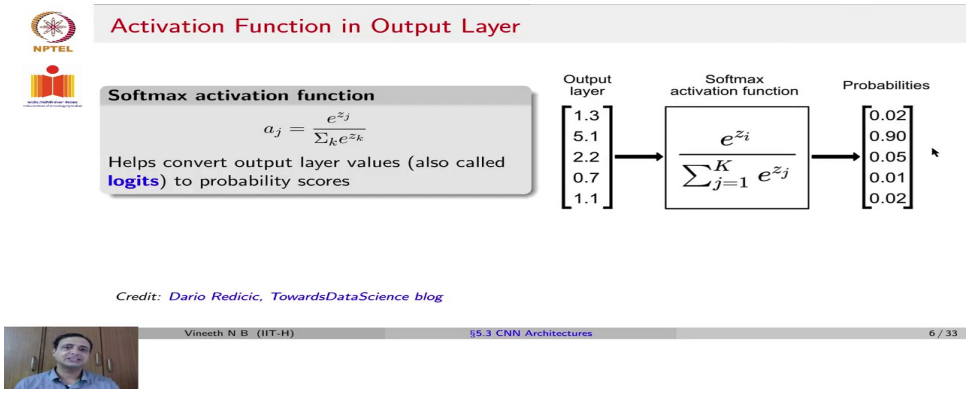




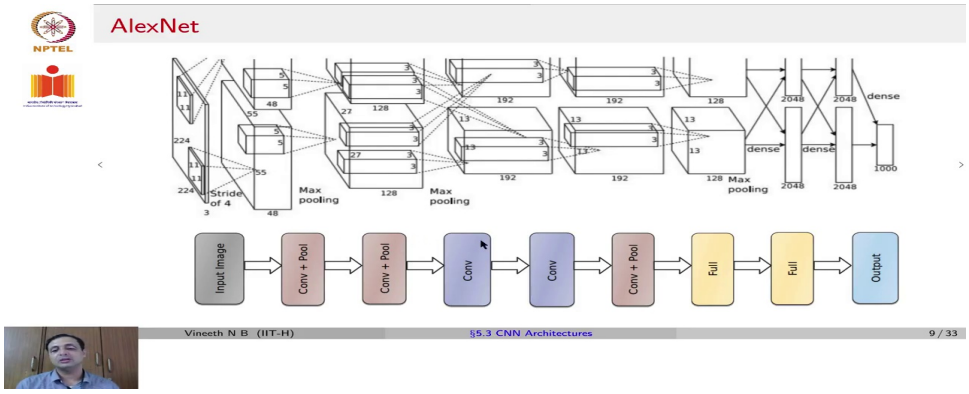


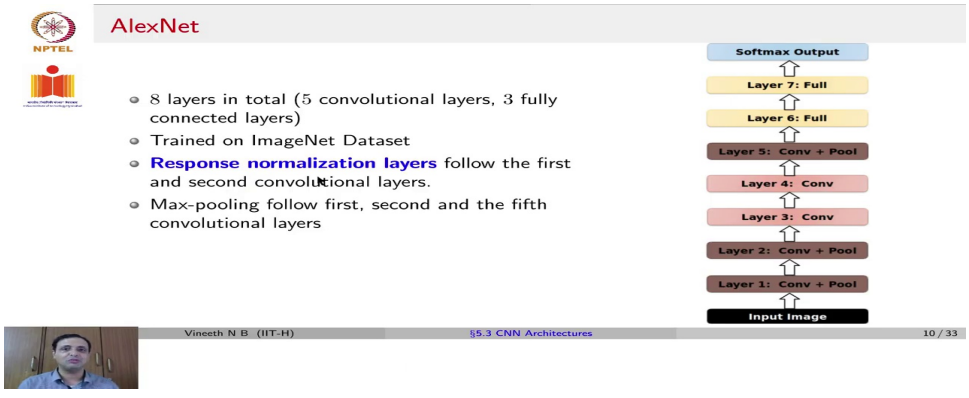
Before we move forward and discuss CNN architectures, let us take a step back and see, how neural networks in general can be used for the classification problem.When we discussed feed forward neural networks, we took the loss function as the mean square error, which was fine for that time.But, when you are working on a classification problem, mean square error may not be the right choice of a loss function.Mean square error works better for regression problems.In classification problems, the most common loss function is known as the cross-entropy loss function which measures the entropy between the probability distribution on the output labels in the ground truth, versus the probability distribution on the output labels as output by the neural network.So, the cross-entropy loss function is given by minus 1 by C, where C is the total number of classes, sum up yi log yi hat; where yi hat is the prediction for a particular class I of the neural network or the score of the last layer of the neural network for the ith class, and yi is the correct label for that particular class.Dou L by dou wj will be minus 1 by n, y by sigma of z, how did we get sigma of z? Sigma of z is the same as y hat; that is the prediction of the neural network.So, this would give you a y by sigma of z, minus second term here will give you the second term in the gradient here.Now, with these two, you can write your next term as, you can now say dou sigma by dou wj can be written as written as sigma dash of z into xj.You will find that this entire term here absorbing the negative sign, the numerator would become sigma of z minus y, and the sigma prime of z xj goes here.Now, sigma prime of z and sigma into 1 minus sigma are the same, because that we know is the derivative of the sigmoid function.That would get cancelled and you will be left with summation over x, xj into sigma of z minus y.What is interesting here is that this term here is very similar to the

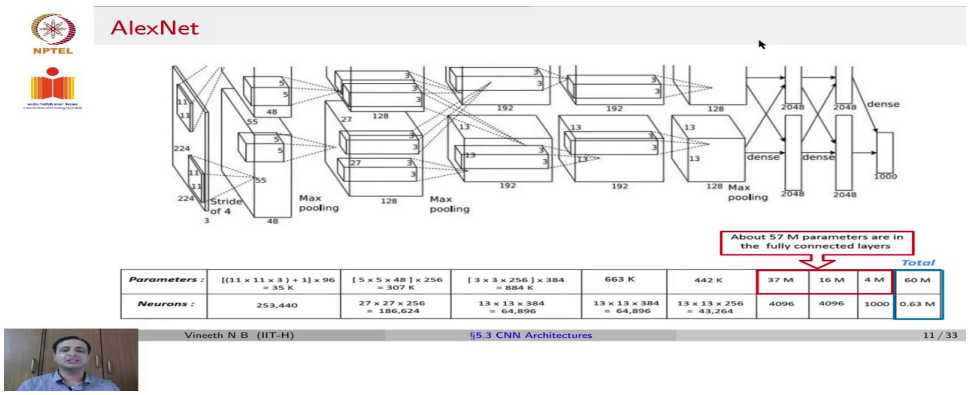
gradient of the mean square; remember sigma of z is y hat.



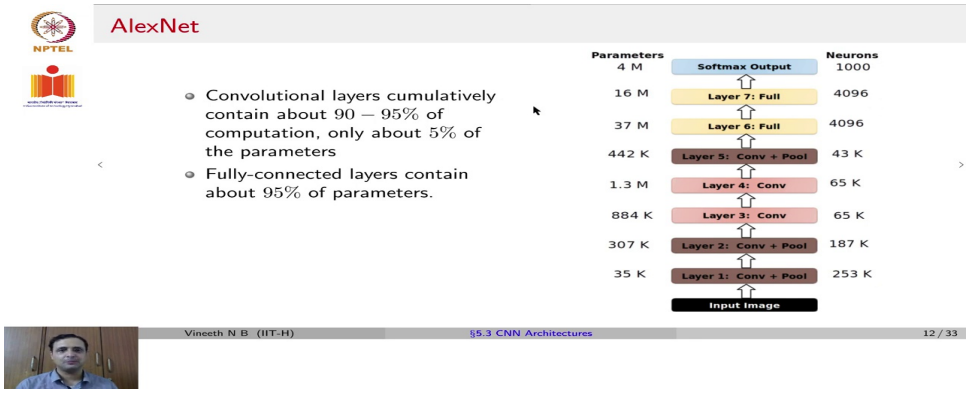
And you would repeat this for every neuron in that last layer and this sum would ad up to 1.In a way now we have converted whatever scores, the neural network outputs into a vector of probability values; where each is lying between 0 and 1, and they all add up to 1.This allows us to interpret the output of a neural network as a probability distribution over your output layer space.So, softmax is a very is a default activation function; that is used in the output layer of neural networks used for classification.A small notation or a nomenclature here the value is before the probability score are typically called the logits of a neural network.So, the neural network outputs logits, you apply a softmax activation function and convert it into probability scores.The first layer had Conv plus Pool, which you see the output together here.Then once again they had a Conv plus Pool, then only a Conv, then only a Conv, then again a Conv plus pool; then a fully connected layer, then a fully connected layer and then finally the output layer.

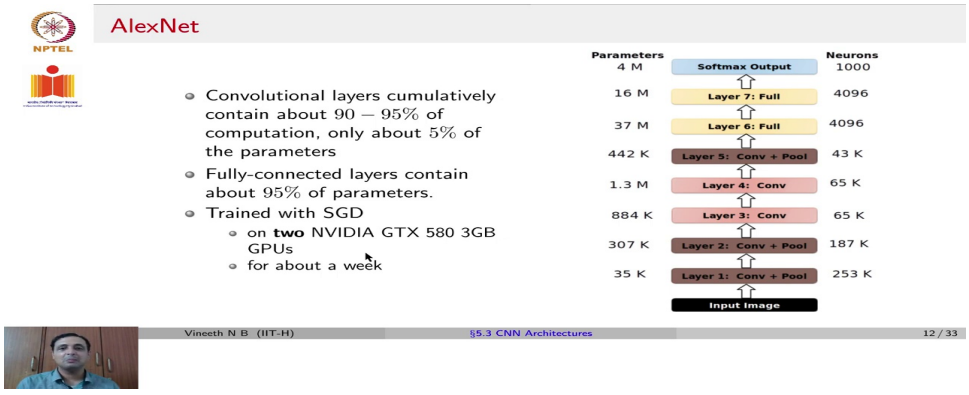


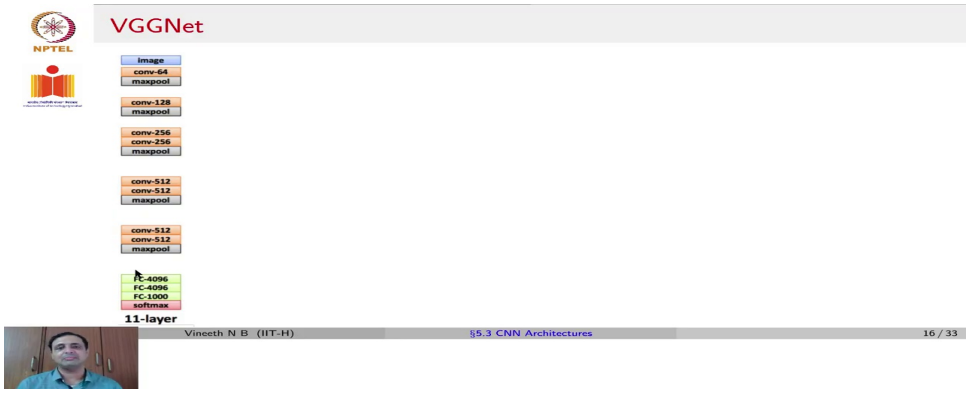




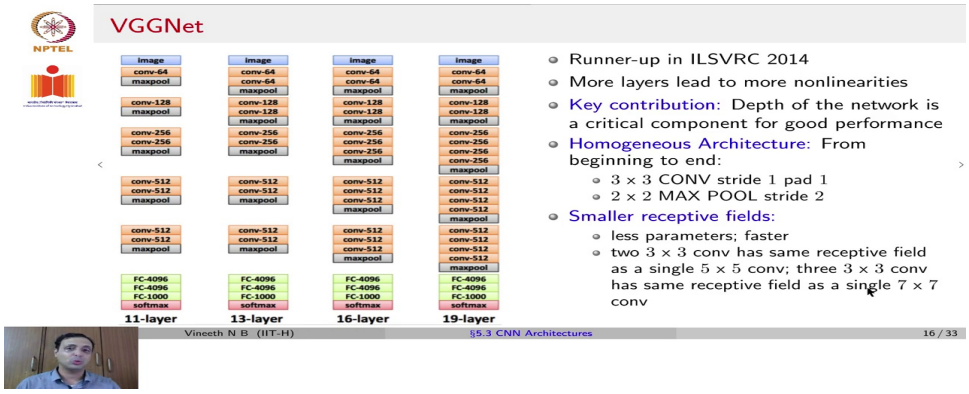
You see here the parameters in the first layer would be given by, there are 11 cross 11 cross 3 is the size of the filter.And because we do pooling, the size of the image becomes from 224 cross 224 to 55 cross 55 to 27 cross 27 to 13 cross 13.And because of the reducing size, the neurons reduce but the number of parameters keep increasing because the number of filters taken are significant.Initially it was 96, the next layer had 128 plus 128 you know filters or totally 256 filters, then 192 plus 192 that is about 384 filters and so on and so forth.You can see that as you go deeper, the number of parameters keeps increasing, the number of neurons decreases until you go to the fully connected layers, where the number of parameters simply takes off and goes to the order of millions.So, about of the total 60 million parameters about 57 million parameters are located only in the fully connected.







Their main idea was that a smaller receptive field means lesser parameters, which means if you have a 3 cross 3 filter; there are lesser parameters to learn.When you do one 3 cross 3 convolution, you get an output; if you do a 3 cross 3 convolution in the output of that first one.Rather, two 3 cross 3 convolutions have the same receptive field as a single 5 cross 5.Similarly, three 3 cross 3 convolutions have the same receptive field as a 7 cross 7 convolution and so on and so forth.



Rather, instead of trying to keep filter sizes like 11 cross 11 or 7 cross 7; they argued that they could always keep the filter size as 3 cross 3.Importantly, as we said it would have fewer parameters, if you took a single 7 cross 7 convolution; you would have 7 square C square parameters, 7 square is 49 C square.Whereas, if you had three 3 cross 3 convolutions arranged in sequence; you would have 3 into 3 square C square parameters, which would be 27 versus 49.Memory would simply be the number of neurons across all of your layers in the VGG versus number of neurons across all of the layers in your AlexNet.