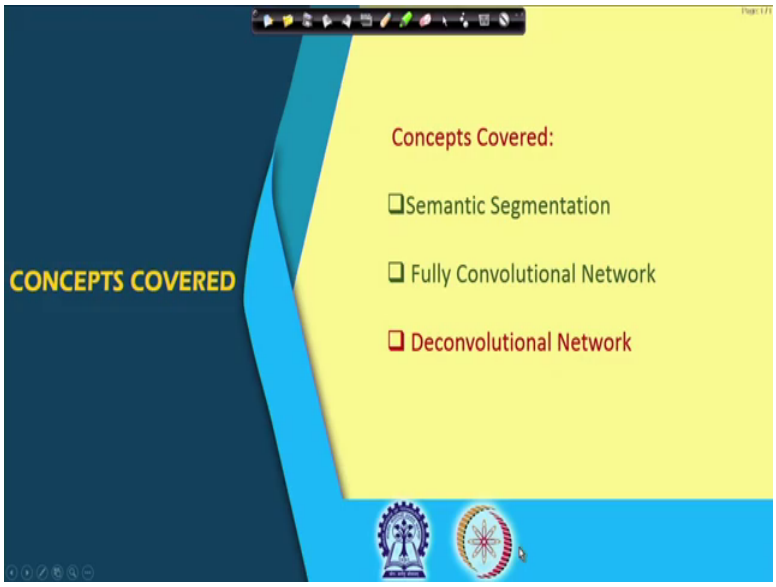
Deep Learning
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 54
Semantic Segmentation – II

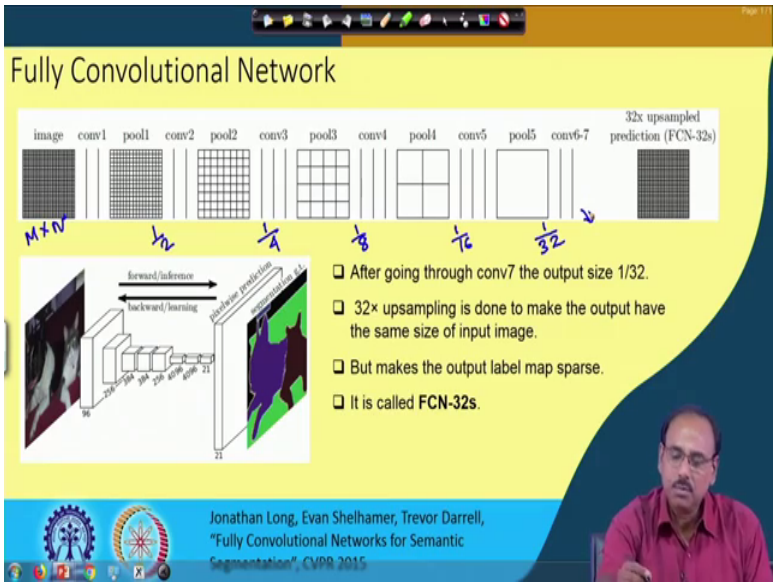
Hello, welcome back to the NPTEL online certification course on Deep Learning. So, we are discussing about the problem of Semantic Segmentation of images using the deep learning approach. In our previous class, we have talked about fully convolutional neural network used for semantic segmentation.

(Refer Slide Time: 00:54)



In today's lecture, we will talk about another kind of network which is deconvolutional neural network for the same semantic segmentation purpose. Now, before we go to the deconvolutional network let us quickly try to summarize what we had seen in our previous class, and we will also try to see what is the drawback of this fully convolutional neural network which the proposers of the deconvolutional neural network have tried to address.

(Refer Slide Time: 01:28)



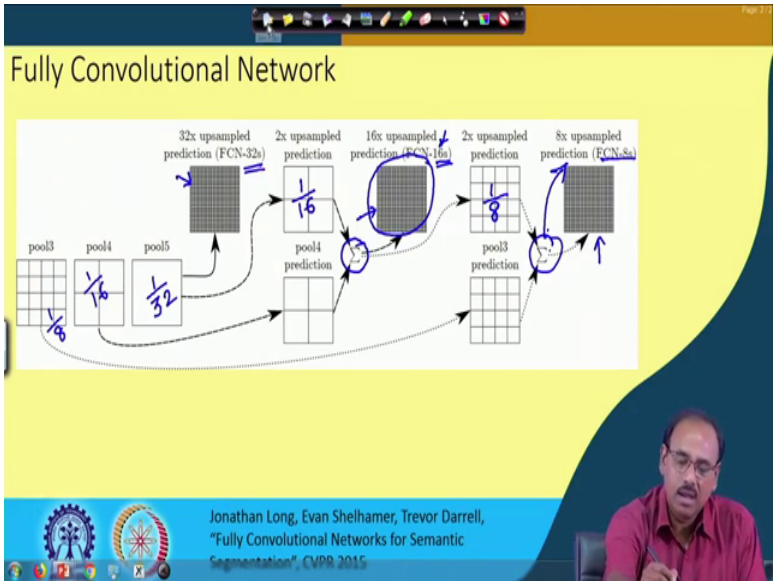
So, in a fully convolutional neural network as we have seen in our previous lecture, the fully convolutional neural network consists of seven convolution layers, and it also consists of five max pool layers. So, after the first pooling layer, if my input image is of size M by N, after the first pooling layer the input image or the size or the feature map becomes half of the original input size that is it becomes M by 2 by N by 2.

After the second pooling layer, the size of the feature map becomes one-fourth of the original image size. After convolution 4, it becomes, after pooling 3 the size of the feature map becomes one-eighth of the original image size. After pooling 4, it becomes 1 upon 16th of the original image size; after pooling 5, it becomes 1 upon 32 of the original image size.

And we have also seen that instead of considering the output layer of a convolutional neural network meant for classification purpose which is actually a fully connected layer. The proposers of this approach have considered that output layer as a convolutional layer. So, as a result, at the output of that convolutional layer what you get is a heat map of your original input image. And to convert that into a semantically segmented output, this output has to be blown up to the size of your original image input, that means, the feature map that you get at the output of pooling layer or at the output of convolutional layer 7 that has to be blown up by a factor of 32. So, that every pixel in that feature map

or in that label map can be classified belonging to one of the classes of objects which are present within your input image.

(Refer Slide Time: 03:57)



So, in order to do this, the proposers what they have suggested is that over here you directly take the output of the convolution layer 5, so here it was actually 1 upon 32 of the original image size. Here it was 1 upon 16 of the original image size, here it was 1 upon 8 of the original image size.

So, one of the approach is that you directly up sample the output of the convolution layer 7 which is the size of the output of pooling layer 5 by a factor of 32. So, this is 32 x upsampled; obviously, this is a sparse map. So, to densify it what you have to do is you have to pass it through the deconvolutional filters, where the parameters of the deconvolution filters are to be learned during training of the neural network using back propagation learning algorithm. But still as it is being done in a single step upsampling by a factor of 32 the output becomes very very coarse or the result becomes very very coarse.

The next approach that they have shown is that output of pooling layer 5, you upsampled by a factor of 2. So, here you are as you are up sampling by a factor of 2, this upsampled output now becomes 1 upon 16th of the original input inner size. And this upsampled version from the output of pooling layer 4, you add with the output of this upsampled

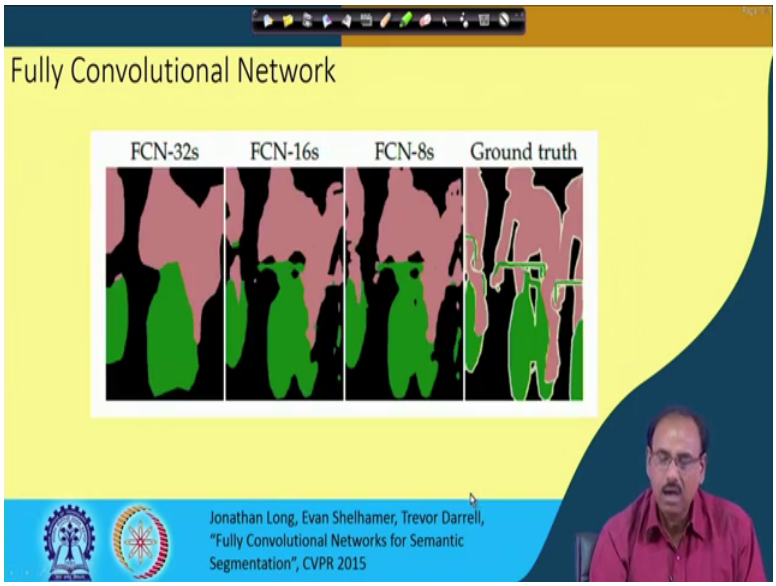
version from the output of pooling layer 5, you add with the output of the pooling layer 4.

So, here you see that what has been used is the concept of skip connection that we have discussed before about the skip connection network or residual network. So, the same concept has been used here that a number of connections or the number of layers have been skipped, so that the output of pooling layer 4 or the subsequent convolution layers that is directly added by passing the input intermediate layers to this up sampling layer. And this is also 1 by 16. So, these two are added together to give you and then it is this output is upsampled 16 times, so that this output is same as the size of this output is same as your original input image. And this is output is what is known as FCN 16.

For further refinement what the authors have suggested is that this output that you are getting over here which is sum of output of pooling layer 4, and the output of pooling layer 5 upsampled by a factor of 2, these two you upsampled by a factor of 2 again. So, here what you are getting is a feature map which is of size 1 by 8 of the original input image and that you add with the output of the pooling layer 3. Again you find that this is a concept of skip connection, these two are added together and this output and then finally, upsampled by a factor of 8.

So, it is quite obvious that the output or the semantic segmentation that you get using this FCN-8s will be finer than the semantic segmentation of FCN-16s. And of course, the semantic segmentation output of the FCN-32s will be the worst among all these three.

(Refer Slide Time: 07:58)



And that is what was also evident from the result that was reported in that particular paper. So, here you find that this is the ground truth. This is the output of FCN-32, this is the output of FCN-16 and this is the output of FCN-8. So, it is quite obvious that output of FCN-8 is very close to actual ground truth, whereas, output of FCN-13 is far away from the ground truth. And output of FCN 16 is somewhere in between.

So, this is what was deported in a paper by Jonathan Long at et. al in CVPR 2015. And they named it as Fully Convolutional Network, because they had interpreted the fully connected output layer as a convolutional layer and the inter walk then based on upsampling and deconvolution operations of the output of that final convolutional layer.

Subsequently, in the same year in ICCV 2015 another paper was published which was the concept of deconvolution network. So, now we will try to discuss about what is that deconvolution network and how it is used for semantic segmentation of input images.

(Refer Slide Time: 09:40)

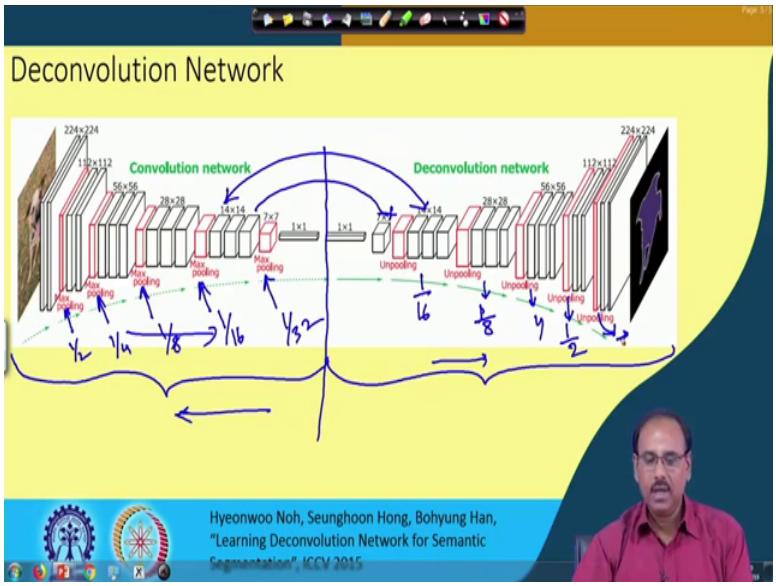
Now, one of the problem that was pointed out by the authors of this deconvolution layer which is an ICCV 2015 pair paper they had are good that in case of this FCN the receptive field of the network is fixed. And because this receptive field is fixed, if the receptive field is very high or the receptive field is very low compared to the size of the objects which are present within the input image, then it is possible that the semantic segmented output that you get many of the objects might be missed. If the receptive field is very large compared to the object size or the objects may be or the segments may be broken into sub segments, if it is otherwise.

So, that is what has been shown within this diagram, they demonstrated within this diagram that here we have a bus the semantically segmented output the ground truth output should be something like this, whereas, the output of the FCN network was shown of this form. Whereas, this is the output of in front of a store here from the ground truth shows something like this there are a number of persons, but the actual output which was produced by the FCN the fully connected network was something like this, where many of the persons have been missed. They could not be segmented by that FCN network. And this problem they have attributed to the fixed size of the receptive field in case of fully convolutional neural network.

So, what these authors have suggested is that instead of having or instead of trying to up sample the output of the convolutional layers in just one step or two steps like FCN 16,

FCN 8 or FCN 32. So, they have given three different steps. What these authors have suggested that the output can be upsampled gradually. The way it has been down sampled during the convolution operation; in the deconvolution operation, it can be up sample following the same number of steps.

(Refer Slide Time: 12:14)



So, as a result of that they have come up with a deconvolution network architecture which is something like this. So, you find that in this deconvolution network architecture we have this deconvolutional network part which structurally is the mirror image of the convolution network part. So, in case of deconvolution portion, the number of unpooling layers, the number of deconvolution layers is just is having a mirror symmetry of the number of unpooling layers and the number of convolution layers that you have in the convolutional part.

So, I can say that if I just simply split over here this part is just mirror image of this portion or the deconvolution network portion is mirror image of the convolutional network portion. And here every max pooling layer has its corresponding unpooling layer. Every convolutional layer has its corresponding or peer deconvolution layer.

So, the advantage of this network is that as while doing the convolution, the size of the feature maps are being gradually reduced. So, first it becomes 1 by 2 after the first max pooling operation. After second max pooling operation, maybe it becomes 1 by 4, then it

becomes 1 by 8, then it becomes 1 by 16, then it becomes 1 by 32. So, this is half, this is becoming 1 by 4, 1 by 8, 1 by 16 and 1 by 32.

So, just conversely when you go for unpooling, after first unpooling it becomes 1 by 16; after second unpooling it becomes 1 by 8. After third unpooling, it becomes 1 by 4. After this unpooling it becomes 1 by 2 and after this final unpooling, your input image size sorry. So, after final unpooling over here your feature map size or the label map size become same as the size of the original image.

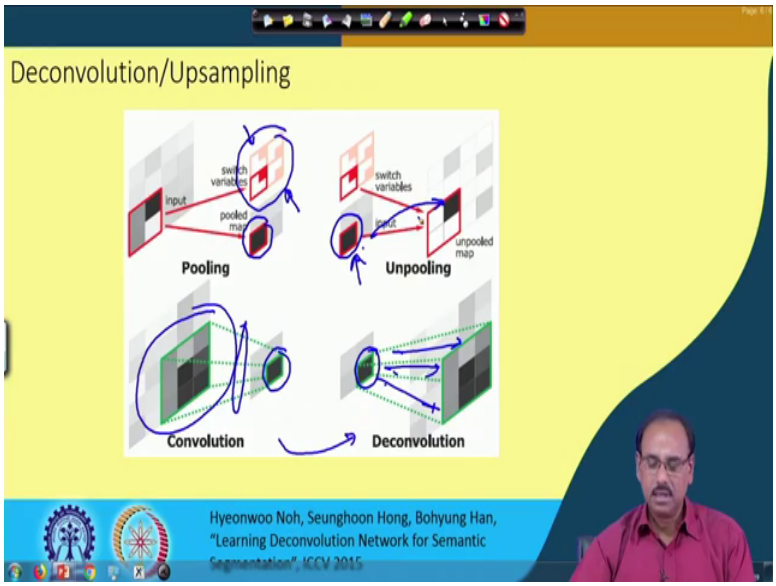
So, here it is being a gradual change or gradual up sampling or up scaling the problem which was first in convolutional neural network, fully convolutional neural network of having fixed receptive field size that is avoided using the deconvolutional neural network.

Now, in order to make your convolutional part and the deconvolutional part exactly matching in terms of max pooling and unpooling, the authors have introduced the concept of switch variable. The purpose of the switch variable is that say when you are going for max pooling operation within a max pool window, you are picking the activation which is having the maximum value, and that a location of that maximum activation may be anywhere within this max pool window. And in subsequent layers the location information of this maximum activation value is lost, it is not being detained anywhere.

Whereas, we want that when you are doing exactly the reverse operation during unpooling, the values which had to be replaced in the unpooled array those values should be placed in the same location and then you perform the deconvolution operation. So, that your max pooling operation, unpooling operation, the convolution operation, the deconvolution operation they become exactly matching.

So, in order to do that what we need is whenever you perform a max pooling operation, in addition to the max pooled value, we also need to retain the location from where that max pooled value was taken. And this location is retained is stored in a variable which is which they have called as a switch variable. So, for every max pool layer, I had to have a set of switch variables and these switch variables are used when you go for the unpooling operation.

(Refer Slide Time: 17:06)



So, the kind of variables is somewhere over here, here it shows that what you do in case of a pooling operation and a convolution operation. So, in case of convolution operation, you take the values from the receptive field or the activation values from the receptive field. You perform the convolution operation over this using the convolution kernel and you get the activation value.

In the deconvolution operation, it is exactly the reverse, you take an activation value of a particular layer and then spread it to its neighborhood to the corresponding receptive field using the coefficients or the parameters of the deconvolution kernel. In case of max pooling operation, the maximum value is pooled and in the switch variable you retain the location from where this value has been pooled.

So, when you go for unpooling, you get the value which has to be or a layer feature map that has to be unpooled or it has to be upsampled. But when you do this up sampling this value which is to be replaced we want that this value has to be replaced in the same location from which the maximum activation was taken. And for doing that you make use of this particular variable switch variable, which has stored the location from which this max pooled value was taken. And this information is used for replacing this value into the corresponding location when you go for unpooling in the upsampled array.

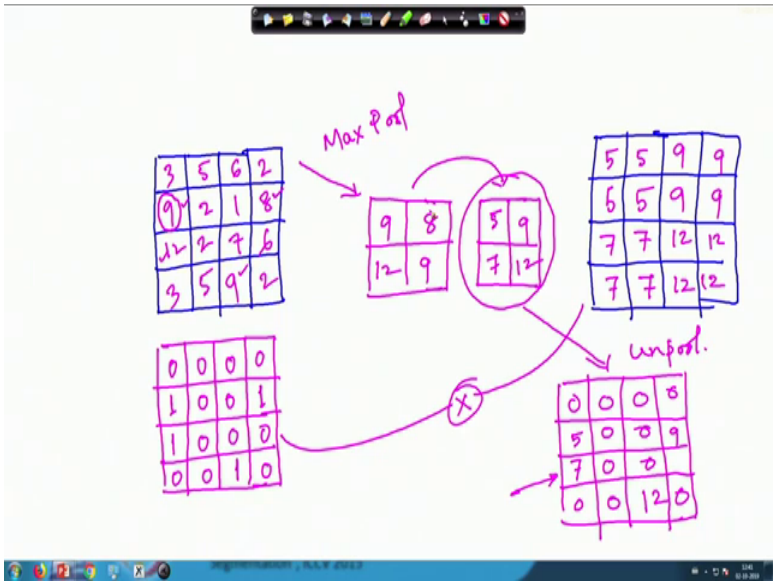
And then you perform the convolution over operation over this. And, the deconvolution that we have shown earlier that when you perform deconvolution over an upsampled

array using stride 1, your array of the map or array of the feature map or array of the label map in this particular case because when you are going for deconvolution let us call it label map. Because finally, what we want is in the segmented output, we want the level of every pixel; it is not the feature that every pixel has to be leveled to belong to a particular class. So, if it belongs to class 1, the level has to be 1; it belongs to class of objects 2, then the level as we 2. So, let us call this as label map instead of feature map.

So, this upsampled array when it is deconvoled with the deconvolution network; deconvolution kernel then the size of the label map increases. And as has been shown over here that the size is gradually increased, and finally what you get is your final label map size over here which will be same as the size of the input image.

And using this concept the authors have shown that, the quality of the output or the quality of the semantic segmentation that you obtain is much better than what you get in case of fully connected network. Now, before I come to the output, let me just elaborate on how this switch variable actually works, so that I can use the information present in the switch variable to replace the values in my upsampled label map.

(Refer Slide Time: 20:53)



So, for doing that let us assume that we have say an array of size 8 by 8 of feature maps or say let us have an array of size 4 by 4, 8 by 8 will be too large. And let me assume that the feature values in this 8 by 8 map are something like this say I have 3 9 5 2 6 2 1 8 say 12 anything 3 2 5 9 7 6 2. So, after doing this, if I assume that I perform a max pooling

function with a max pooling window of 2 and with size 2. In that case, you will find that within this max pooling window of 2 by 2, the maximum value is 9. So, my max pooled output over here will be something like this, here I get 9. Similarly, over here, it is 8; over here, it is 12 and in this max pooling window, it is 9.

Now, the locations from which this max pooled value was taken is, in this window it is this location, in this window it is this location, here it is this location and here it is this location. So, I can form a switch variable or a switch array, the switch array will be of this form. Again, this is a 4 by 4 array. I will make this value to be 1 from where the value has been pooled. I will also put this value to be 1, because from here the max value has been pooled, this value to be 1 and this value to be 1.

So, this indicates wherever I have value 1, those are the locations from which the maximum activation values have been pooled. And the remaining locations in these two arrays keep them equal to 0. And during unpooling operation, suppose after passing through a number of convolution layers and the deconvolution layers at the corresponding deconvolution side, my mapped are the label variables, the label values have been something like this 5 7 9 12 which is of this form.

And when I go for unpooling of this, again I have to make an 8 by 8 window sorry 4 by 4 window. So, the way I can make this 4 by 4 window is something like this. I know my pooling window size was 2 by 2 with size 2. So, this is my 4 by 4 window or the unpooled window, and in this unpooled window, I have to keep these values.

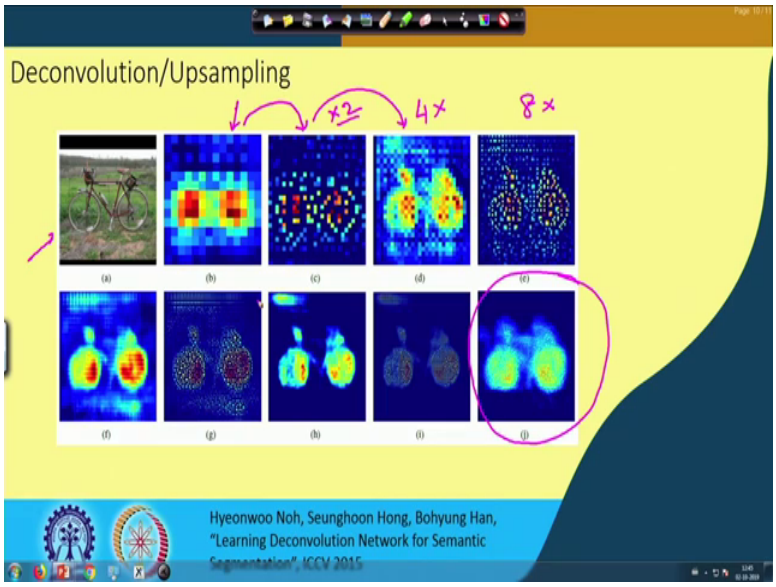
So, initially what I can do is because this value is 5, and I know this 5 has been pooled from this particular pooling window. So, I will keep all these values to be equal to 5. Similarly, here I will make all these values to be equal to 9. Here I will make all these values to be equal to 7, and here I will make all these values to be equal to 12. And then what I do is I multiply this output with the switch variable. And once you do this multiplication, you find that only at the locations where you have value equal to 1, those locations will be retained and all other locations where the value is 0, those will be made 0.

So, as a result here you will get a value of 5, here you will get a value of 9, here you will get a value of 7 and here you will get a value of 12. So, you find that these values has been placed in exactly the same locations from which the values are pooled. So, this is

you are pooling operation or max pool operation. And this is what is your unpool operation; output of unpooling and all the remaining values will be equal to 0.

So, now that I have this upsampled label maps, on this upsampled label maps you perform the deconvolution operation using the deconvolutional kernel. And you get a label map which is of larger size than this particular label map, and this can be done in every label in your deconvolution layers.

(Refer Slide Time: 26:27)



So, by performing this kind of operation, the authors have shown that using this deconvolution network. The nature of the feature map that you get or semantic segmentation output that you get is much better than the semantic segmentation output that you get from a fully convolutional network.

So, here again some result as reported in the same paper that has been shown. So, this is your original input image, you find that this input contains the bicycle. This is the feature map or the label map that you get the coarsest at the coarsest level. This is the feature map that you get after the next up sampling and the deconvolution operation. This is the feature map that you get the next up sampling and convolution operation.

So, whatever was the size over here, here the size is twice of this, here the size is 4 x of the size of this, here it is 8 x of this and so on. And this is your final label map that will be obtained. So, you will find that gradually as you move from one layer to another layer

in the deconvolution part, gradually the labels are being refined. And as a result the semantically segmented output that you get using this deconvolutional neural network is much better than the output that you get in case of fully convolutional neural network. But at the cost of what is it that everything is good or everything is advantageous, there is no disadvantage.

You find that in case of fully convolutional neural network, we did not have any switch variable, whereas when you go for this deconvolutional neural network you have the switch variables. And you have to have set of switch variables for every channel for every layer, that means, the memory requirement for deconvolutional neural network is much more than the memory requirement that you have in case of fully convolutional neural network. So, in order to gain something, you have to pay for it, so that is the pay that you make in terms of memory requirement when you use this deconvolutional neural network.

So, till now what you have discussed is that given an input image, I assume that I have our fully convolutional neural network or the deconvolution neural network, they are properly trained. And bypassing this input image through any of these networks at the output you get the semantic segmentation. But to get the semantic segmentation, the first operation is how do you identify a segment because the output does not give you the segment itself it keeps some real numbers.

In our next lecture, we will see that what is this real number that you get and from that real number you have to get you have to obtain the class level or the segment level. So, in your next class, we will talk about how this class level can be obtained. And the other very very important part is training of this neural network's whether it is fully convolutional neural network or the deconvolution neural network. In both the cases, the networks are to be trained properly before they can be used for semantic segmentation purpose.

So, we will talk about the training as well as generation of the class level of every pixel or classification of every pixel into one of the possible classes available classes we shall discuss that in our next lecture.

Thank you.