

Deep Learning for Visual Computing
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

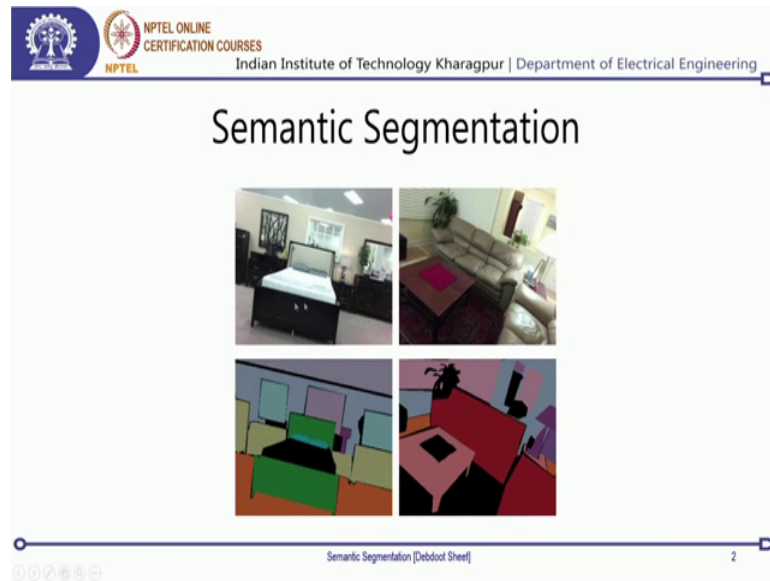
Lecture – 49
Semantic Segmentation with CNN

Welcome this next topic which we are studying today is called as semantic segmentation. So, in much simpler terms, a semantic segmentation is also in the community known down as a simple segmentation problem. And we are going to use on convolutional neural networks for doing it.

Now, where the difference comes in is whatever networks you have done till now they are all classification networks. So, given an image, what is the class of the object represented in image, they can be multi hot classifications which are point on all the different objects which are present in an image. And you can even do localization with region proposal network or with a activation map over there.

Now, here where it comes as a difference is that you try to solve each pixel to pixel labeling as a convolutional neural network problem over here. And what that would essentially give is that pixels can either be exclusively labeled on in exclusively labeled, but then the classification problem is no more for the image as a whole. The classification problem is now a pixel-to-pixel basis classification problem and that is what we are going to solve out. So, we will take so let us see what how these things look like. So, in a typical example, you would have an image coming down over here.

(Refer Slide Time: 01:31)



So, in typical example, you will have these image coming down over here, and you will have some sort of a map present ok. Now, what happens is that each of these pixels over there will be associated a certain class. Now, these classes can either be something which is undefined say the black ones over here are typically taken down as undefined; and the other ones are which belong to a particular object category. Now, there can be these mirrors, there can be glass panes, there can be this picture, or say the bed sheet then this wooden part over here and similarly.

Now, these kind of data sets what they do is it its manually segmented out each of these regions, and you have some sort of a watershed kind of an algorithm running down which makes it easier for manual segmentation basically. So, there is some sort of a manual initialization and then you let flood filling algorithm go down. But then the main problem for a cnn based system is that you will be given down this image and you will have to predict out something which looks like this kind of an image. However, this these colors are not pseudo colors, so they are just colors which are associated with one particular classification label associated with a given pixel over there.

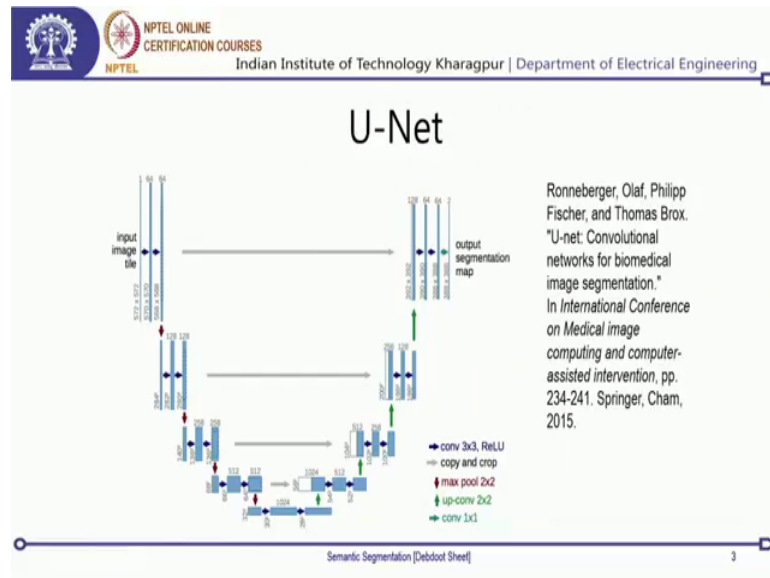
So, now, my problem boils down to that for a given pixel based on whatever is present in the surrounding of that pixel and everything which is present on the image I want to find out a class label, a class label for every given pixel. So, my output size x y dimension is going to be the same as the x y dimension of my input image itself, they are not going to

be different in any way. Now, that they have to be the same the point is how do we actually find out our errors and try to do it and how will this network be created. So, this has been an interesting problem within the community. Now, traditional computer vision community was something which was doing really high in this context; so a lot of these developments with respect to graph theory; then with respect to Markovian processes and random fields over there. Now, it has been just in the recent past that we have seen these kind of convolutional deep neural networks take a center stage in order to solve these kind of problems as well.

So, today we are going to discuss two very specific over there; one of them is called as U-net which was a really a winning solution and with a lot of effort put down into the engineering over there how to solve this as a problem. So, it was not just the architecture which was designed, but the training mechanism was also an intuitive part. The cost function used for training was also another interesting major contribution brought in down by the inventors of U net.

So, this was used for solving a semantic segmentation in biological images. So, you have images from microscope or you have images say from a transmission electron microscope or from the phase contrast microscopes. So, these images have really a very low contrast coming down of the object to be segregated from the background. Now, what these networks actually do is despite there being a low contrast they can very accurately find out what is the boundary around the object and which is the object and what pixels correspond to this particular object over there.

(Refer Slide Time: 04:37)



So, let us get into what this network looks like. Now, the whole reason why this was called as a U-net is because the architecture is something which resembles a U, a complete U over there ok. Now, it takes in an input image over here. Now, they call it as a tile. This was done more for simplification of the training process and the native model used to take in an image of the size of 572 cross 572. And these were all grayscale images, they were just one single channel that is what you see over here as a channel. So, it just says one channel over here.

Now, this was the whole reasoning for a 572 cross 572 is just to get down from this lower level reasoning because at the lower level they just wanted to have a 30 cross 30 matrix size over there. And then if you go down with this 30 cross 30 matrix size. And just keep on upscaling then you would see down why it comes down to this one ok. Now, you have typically seen the convolutional neural networks, where everything is arranged one by one in a sequence fashion and it keeps on going. But here you see that there is some sort of a pass then the output goes down then it passes and that is something which we uses over here.

Now, the color conventions of each of these the arrow marks are mentioned down over here clearly and that should be making it easier. So, what they do is you have an image of seven 572 cross 572, and then you run a convolution. This convolution is a 3 cross 3 convolution with a transfer function of ReLu, and since they are not having any kind of a

padding added over there. So, whenever you convolve with 572 cross 572 with a stride of one. So, no pooling or no multiple strides involved over there. Your resultant is 570 cross 570. You do the same kind of a convolution, you again get 568 cross 568.

Now, for the first convolution you have 64 kernels, 64 channels which will be giving out over there. So, there are 64 convolution kernels. And as a result you have 64 channels being created for the next bunch of convolutions as well you have a similar kind of a option going down ok. Now, from there when it comes down to when there are after two subsequent convolution layers, you have a max pooler. So, this max pooling is with the max pooling of two cross two. So, what essentially it would do is that the number of channels is still going to remain the same.

So, there are 64 channels, but the spatial span x and y span over there is going to reduce down by a one-fourth. So, each axis is going to be reduced down by half. So, 568 now gets mapped on to 284. So, this is a 284 cross 284 sized matrix which has a 64 channels over there ok.

Now, from there you can again do a convolution 3 cross 3 convolution and 228 channels. So, you get a hundred and with 128 convolution kernels over there. So, you get 128 channels. You do the same thing and again you come down over here. So, and each of these places, you since you do not have any sort of a strided sorry any sort of a padding done, but you are just doing it with the stride of one. So, a full convolutional scale over there is going to make you come down to a size of 280 cross 280. Similarly, you again do a max pooling and then come down and here your channels are increasing.

Now, you can see a distinct similarity with respect to VGG net, where what was happening is that whenever you are reducing by half on the x y size over there, you are basically doubling up the total number of channels over there at that particular layer and everything is happening in banks. So, if you follow this down so every time there is a max pooling and then there are two banks of convolution, then there is a max pooling, there is a bank of convolution then a max pooling then a bank of convolution then a max pooling and then it passes through ok.

Now, at the last part over here, you do not basically do some sort of a fully connected layer over there, but you are just doing this convolution and passing it up. Now, on the other side of it where it starts going up which is every time I am going to double up my

spatial size. So, here I had 28 cross 28, now I do an up sampling over there which is also called as an up convolution. They do it with a 2 cross 2, so it comes down to a 54 cross 54. Now, we had done in our earlier lectures on encoders and decoders, we had studied about how to do a up convolution over there. So, one was where you have a learnt up convolution; the other is where you can just do an interpolation and make it double the size

Now, here they had used just a simple interpolation kind of a technique. Now that you interpolate and go up and you have 54 cross sorry you have a 54 cross sorry 56 cross 56 channel output over there and that has so over sorry. So, what you do over here is that from 1024th channels this is something which is it is not clearly mentioned out over here. So, from 1024 channels you are supposed to come down to a 512 channels over here. Now, you are going to up convolves this 512 channels and that will contribute to 56 cross 56 cross 512 over here, 512 channels each of 56 cross 56 size. Now, these 512 channels which are of 64 cross 64 size are now mapped onto this extra channel over here.

Now, what is essentially done is you do a copy and crop arrangement, which means that on this 64 cross 64 if I want to chunk out of 56 cross 56 which basically means that four border pixels on all the sides have to be reduced out. So, 64 minus 56 is 8 that divided by 2, so it means 4, 4 pixels on all the sides. So, these border pixels are what chunks out over here. And now if you look carefully at this image, so you can actually download this paper and zoom in and look into this image. You would see a dotted line and this whole purpose of this dotted line was that just a chunk of it is going to be copied down over there.

Now, this is copied and appended over here. So, in essence you get down 512 channels which come from here, and 512 channels which come from here and appended. So, this has 1024 channels. Now, there are 512 kernels which will map these 1024 channels onto over here such that this comes down to a size of 54 cross 54. And then again you do a 3 cross 3 convolution using these blue arrows at all 3 cross 3 convolutions with ReLus you come down to a 52 cross 52.

Now, from there you are again going to go up now here what this whole thing does is when it does a 2 cross 2 convolution, it reduces them the total number of channels as well. So, from 512 channels, you are going to go up one to 256 channels over here. And

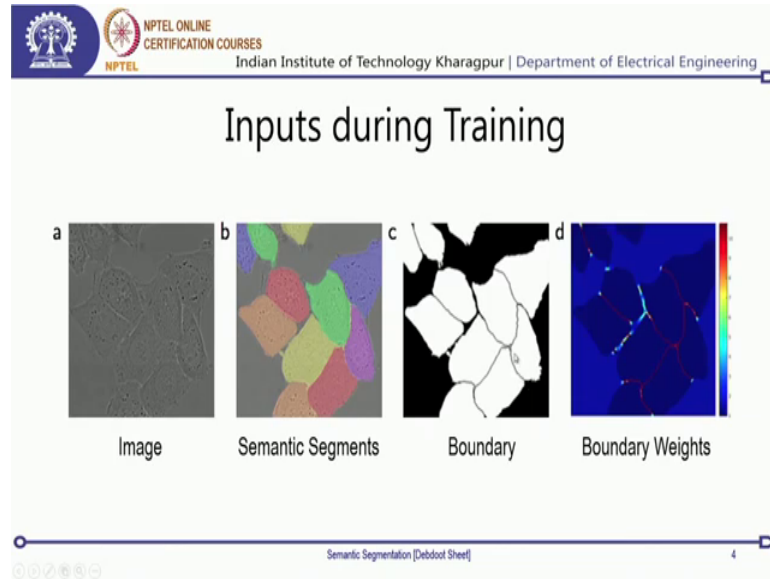
then you take down a chunk truncated part over here of 256 channels and merge it over here. Now, once you merge it out over here, you get 512 channels coming down. So, this truncation is going to ensure that whatever is the output which you pool from the earlier layer on this part of the network which can also be called as an encoder network is some sort of compliant with the size, so that you can actually append and keep this one going.

So, you keep on doing the same thing and repeat and come down to this upper layer. And on this upper layer, what you have is from this last convolution layer, you have everything appended over here, so that makes it 64 channels from here and then other 64 channels from here so 128 channels. Now, you do a convolution and get down 64 channels you do another convolution 64 channels. Now, from the 64 channels, you need to come down to 2-channel output over there. And this is for the foreground and the background that is just two classes over there.

Now, the best way of expressing two classes is as in say a classification problem, you had just two neurons. So, if you had an image, you would be and if you want to classify this image into two classes, you just have two neurons. And based on which neuron is firing up as high is what you say that this is the class present in the image. Now, here the same thing boils down to as I want to classify each single pixel. So, I need to have one neuron representing each single so and if there are two classes over there, then there should be two neurons which represent each pixel.

Now, if I have a semantic segmentation, where I have 10 classes to solve out for each pixel, then over here, I should be having down ten output channels such that each channel at a given pixel is what is representing a particular class. So, you have one class coming out for each of them so that is the whole architecture, and how it comes down. Now, essentially on having solved it out, so what we do is on your input side you are going to feed an image; on your output side, you get down these different channels coming out. Now, your data for training should also have all of those channel data present in the same way; otherwise it is not going to work in any way that is that is for any kind of a semantic segmentation problem. So, this is about U-net.

(Refer Slide Time: 13:29)



Now, cleverly what these people had done is so you have this image. So, you can look down how low contrast this is. Now, this is the ground truth which is available for semantic segments. So, you see each in a different color because that is so it is it is not something like there are different number of channels over there it is just two channels of foreground and a background. Now, the whole purpose of doing this one was just to enable you to count out how many objects are present over there. So, this was just an object counting.

So, each object is given down a different label and that can be implemented with a very simple algorithm called as connected component labeling which you have studied known in your image processing classes. Now, what you need to essentially have is a very good segregation of these boundaries between these cells and that is a very important aspect. So, what they did is once you have these objects present down over there, and if you can count out what each object is then you can do some sort of a morphological operation and actually chunk out these boundaries over there.

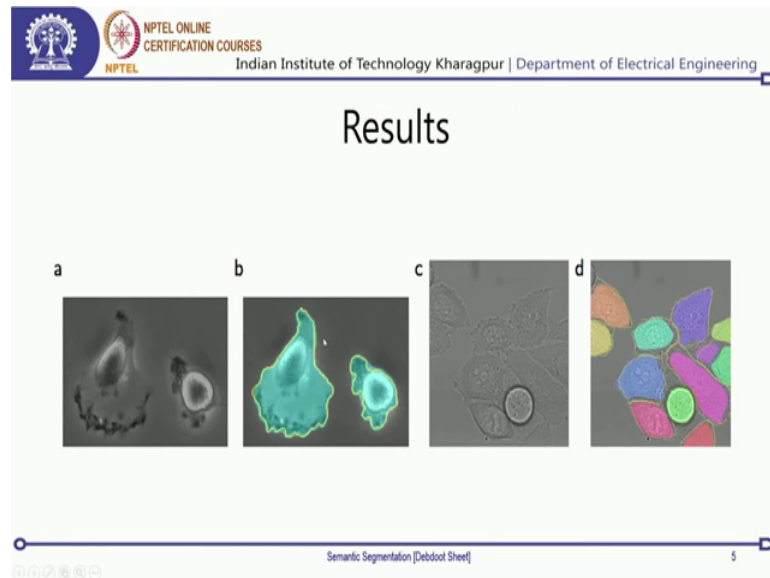
Now, this boundary map is very important to calculate something which is called as a boundary weights. Now, what they do over here is that you have your foreground and background which classification problem you are trying to solve. But these boundaries are very critical, and these boundaries between cells are what will be segregating one cell from the other and this boundary belongs to this background class itself. Now, you know

if we can associate certain weights distinct weights with these boundaries over there, then when I am classifying it out what it would do is that if there is an error associated with so you remember about the different cost functions which we had done.

So, we had these weighted cost functions for classification. And one important aspect of having a weighted cost function for classification was that so that if certain classes are underrepresented. So, over here you would see that majority of it is foreground some of it is background, and then these border regions or the boundaries between these neighboring cells is what is really underrepresented. So, if certain classes are very underrepresented then I would try to somehow gives the errors over there a much higher weight such that, when I am forcing these errors to come down to a higher weight I will essentially try to make lesser errors. If there is more penalty associated with one miss classification, I will try to reduce that miss classification. So, this is what will happen within a learning algorithm.

And for that purpose, we actually associate different weights over there. And these weights are on a pixel to pixel basis ok. Now, in the earlier class, in the earlier examples, where you are doing a classification of an whole image. So, you had a weight associated with a class now you have a weight associated with a pixel and where that pixel is located ok. Now, if you have a pixel which is located on the boundary, and if you are making an error then that costs you more, then if you are making an error somewhere in between over here. So, this kind of weighted scheme of finding out an error is something which really helped in the performance of U-net kind of an algorithm.



(Refer Slide Time: 16:25)



Now, finally, these are some of these results, which come out over there. So, if this is an input over that this is the output which it shows out. So, in the yellow you basically have the ground truth and the one in cyan which is a shadow out is what is the actual result of segmentation. So, here also you see the same thing in yellow, you have the ground truth of the boundaries over there or for each of these objects what is the ground truth of the borders. And then the ones in color are the segmentation outputs. So, output for each of these channels which comes out over there.

Now, this is one kind of a network a very famous network in the biomedical image segmentation community called as U-net.

(Refer Slide Time: 17:04)



NPTEL ONLINE
CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Results

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	0.9203	0.7756

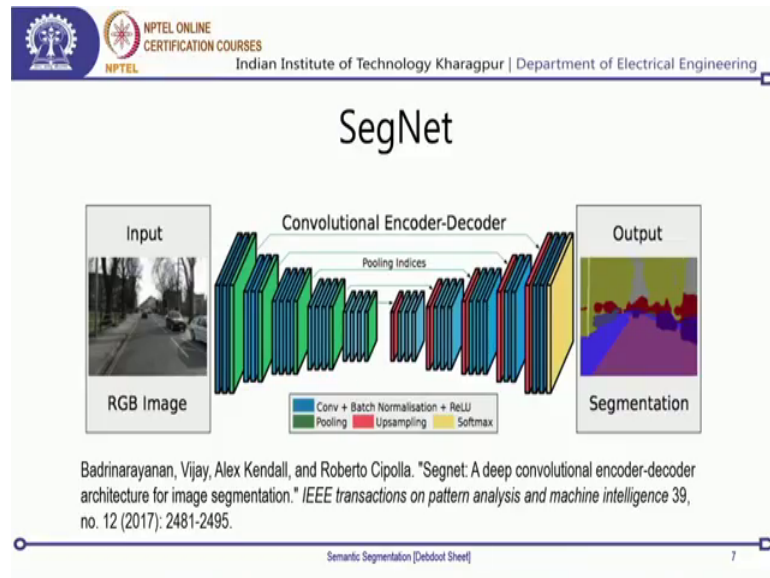
Semantic Segmentation [Debdoot Sheet] 6

Now, so these were and that when it was solving out in 2015 and on the cell segmentation challenge in (Refer Time: 17:11). Then this is where it actually stood down. So, this was like way ahead over here, and then this score over here is just a IOU or intersection over union. So, what they do is over here the ideas that you find out total number of pixels which are common between the ground truth and your segmentation result. And you divide this by the intersection or all the sum total of all the pixels which are common which are there either in the ground truth or in your segmentation results.

So, it is it is just $A \cap B$, where A can be your ground truth and B can be your segmented result divided by $A \cup B$ So, so it is a just a modulus count over there and um. So, this PhC was a phase contrast image, and DIC-HeLa was a images taken down from a different kind of a microscope called as a differential interference contrast microscopy over there.

Now, in either of these cases, you see that this particular network really stands ahead of a even its next competitor. And one of the reasons why this was working is a definitely there was this architecture which is working out great. The resolution level dependent transfer of certain weights is what was helping out, but more importantly was this weighted error function which played a significant role in how it really worked out ok.

(Refer Slide Time: 18:31)



Now, the next network which we study is from a actually last year it so the network was out there in the community for quite some time; it got published on archive in 2015 and then has been there in rounds. And in 2017 on i triple e transactions on pattern analysis and machine intelligence it came out as a full paper. Now, this kind of a network called a SegNet is a really a interesting one which has taken the community largely basically because of the speed up which it gets over here. It takes in certain aspects from a certain understanding of resnet, dense net or even U nets as well, but not all of them.

So, here this network is specifically designed for RGB images. So, you do not take a one channel output over the input over there, but nonetheless the output mechanism is the same. So, for each single class, you have one channel associated over there. So, if you are changing the number of classes, you will have to change the number of channels in the output as well. Now, what it does is you have some convolutions convolution layers over there. So, the ones in blue are convolution and batch normalization and relu. So, batch normalization is just for a kind training it down in a batch training method. So, otherwise it is on the structure side it is just a convolution and relu which is present on these blue layers; on the green layer is where your pooling comes down..

And if you look at it clearly this also looks very similar to a VGG kind of a structure. So, you have a bank of these convolutions then you do a max pooling then a bank of thicker convolution or fatter convolutions you do a max pooling and subsequently you go on.

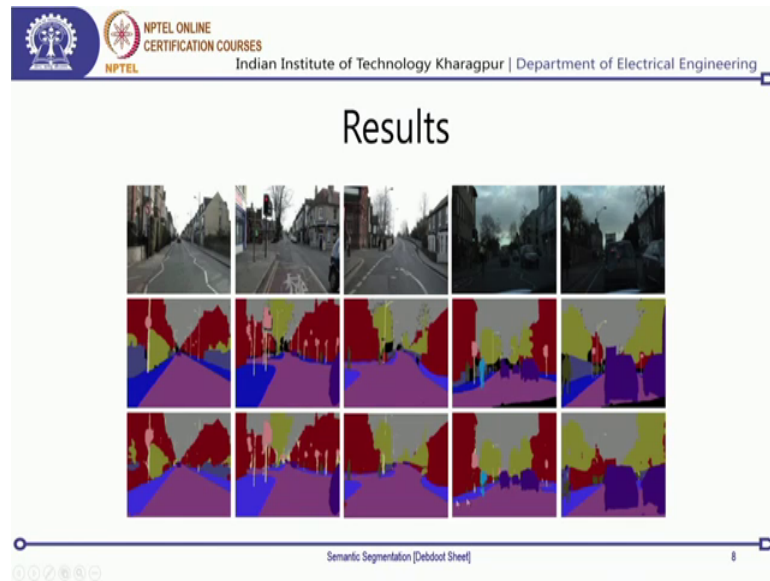
So, if you look at this part of it, it pretty much resembles a VGG 16s initial layers over there, but on the other side you have the decoder part over there. So, this decoder is something it, it rings a bell because it sounds as if like an encoder decoder yes it is an encoder decoder. The only thing is it is not a auto encoder in any way because you are not reconstructing the same image over here, but you are just finding out the semantic maps ok.

Now, when I am upscaling so what would happen is that what we were doing out in U-net was necessarily that we were just making it interpolated by double the factor. So, if I had a two cross two upscale, then I will interpolate on both the axis by double the size over there, but then here that might not work out. So, instead there is another mechanism which is called as a pulling index translation or in index passing and pulling. So, what that essentially does is if you look at a if you if you remember clearly from our pooling operations over there, and an unspooling, so you have a two cross two block say for our two cross two max pooling and then you would represent it at one region.

Now, if it is a max pooling operation then one of these pixels is what is represented as one pixel now if I backtrack and try to put it back over there then all the four pixels are the same maximum value. Now, this is not right because I am going to have patchy kind of efforts and lose down on the high frequency and the smooth components coming down over there.

Now, here essentially what we do with this pooling transfer is with this index transfer, you have this 2 cross 2 region where you are going to put it down, but then you know exactly from which location this was pooled down. So, you are going to replace it back into that region and all the other missing ones are what you can fill down with an interpolation any now. So, that is the difference which comes down over here.

(Refer Slide Time: 21:54)



But essentially that allows you to preserve a lot of your boundary properties. Now, if you look down into your results over there, so given this kind of an image, this is the semantic segmentation ground truth which is present down. The second row is the ground truth, and the third row over here is what is my predicted results. And you can see pretty sharp predictions coming down over there, and one of the reasons was that because of this un pooling this decided index passing un pooling which you could now do over here, you had very sharp boundaries which were retained out over there and instead of blurring out in any way. So, that was one very important aspect which comes down in this kind of a network ok.

(Refer Slide Time: 22:30)



Now, if you look into another result which was like one example which I had taken on in the first slide on a pixel by pixel segmentation problem, you still see the performance coming out quite good. Now, the downside is definitely that say black as a unknown class if it is not taken down in your training data then it will not come out. So, if your unknown classes are not marked as an unknown class separately here, then there is a high chance that because nonetheless it has to point out any one of these object classes as being over there.

So, if none of the above is not an object class and you do not have a separate layer defined for that zero labeled out class, then it is going to put on any pixel as one of these labels because it does not know exactly what is an unknown label over there. So, that was a only downside over here, but then the method and the performance has improved over here and the results are much better if you look at them now.

So, this is where we end now and on semantic segmentation we will be solving out on the tomorrows class in the lab session on trying to do something and this will be an interesting problem. We will not be using any of these data sets, but a very different data set and try to see if semantic segmentation can also solve that kind of a problem on these images has got. So, with that we come to an end for today's lecture.

Thank you.