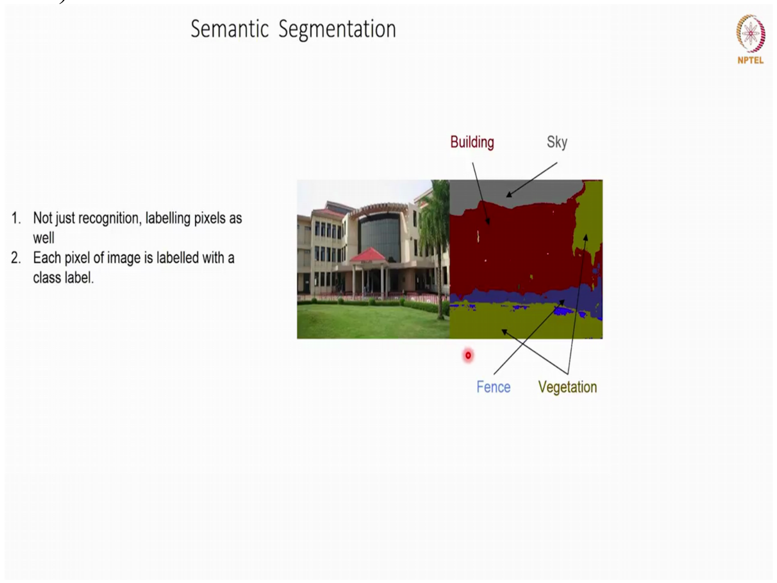
Machine Learning for Engineering and Science Applications
Professor Dr. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology Madras
Semantic Segmentation

Hello and welcome back. In the series of lectures, we will look at semantic segmentation and fully convolutional networks that are used for accomplishing this task.

(Refer Slide Time 0:25)


Semantic Segmentation

1. Not just recognition, labelling pixels as well
2. Each pixel of image is labelled with a class label.

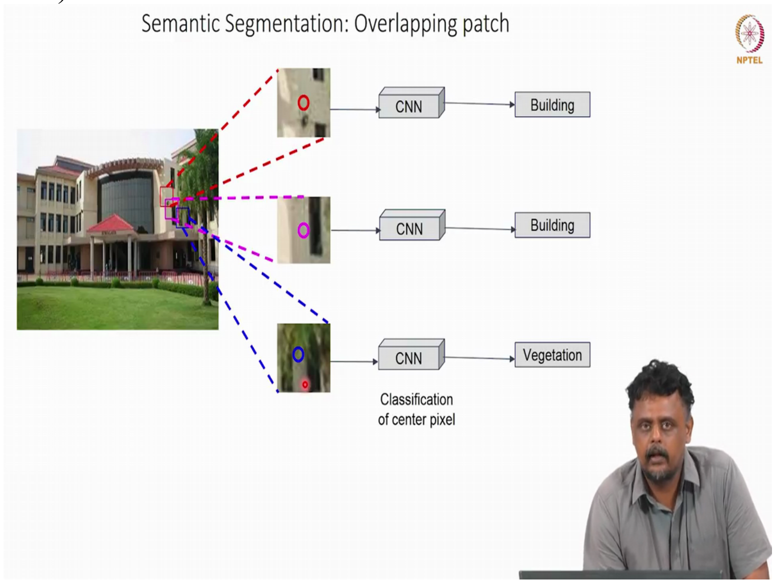
So let us consider this image here. It is an image of a building. So if you have a typical look if you have if you consider the MUL networks that we have looked at so far especially those that (())(0:37) challenge and if a building is one of the classes, what the network will typically do is to just identify this image as a building or for that matter, a lawn let us say a skyline for distance. However in many tasks, what is required is actually a pixel wise labelling of the scene.

So if you see on the right here, here is the result of a semantic segmentation network shown here. So it has classified each of the pixels into a corresponding category. So for instance, this region here, it is a building. This region all the pixels in this region have been classified as a building, the skyline is appropriately classified as skies and we have the vegetation here and here and the fencing is also been classified accurately. Of course, there are some errors where it mistakes the shops for the fencing also. But that is to be expected in some networks.

So this is the typical task that we would like to accomplish in semantic segmentation. That is basically classify each pixel as belonging to a particular category. Then of course there are many nuances here. So for instance, if you look there is just one building. Let us say there are a series of buildings separated by some space and in some applications, it is actually desirable to label each of those buildings and also recognise them as new individual buildings. So it is called instant segmentation, a slightly more fine grained task than what we are going to look at right now.

The applications we are going to look at, the networks are going to look at, are only segment objects of a particular type. So if we say, if there are multiple buildings, they will all be designated as buildings. If there are multiple objects of the same type, they will just be designated as objects okay of the same object. So for instance, just to reiterate, we want to classify every pixel as belonging to a particular object and that should be the output of the network. Okay.

(Refer Slide Time 2:31)


Semantic Segmentation: Overlapping patch
NPTEL
CNN
Building
CNN
Building
CNN
Vegetation
Classification of center pixel

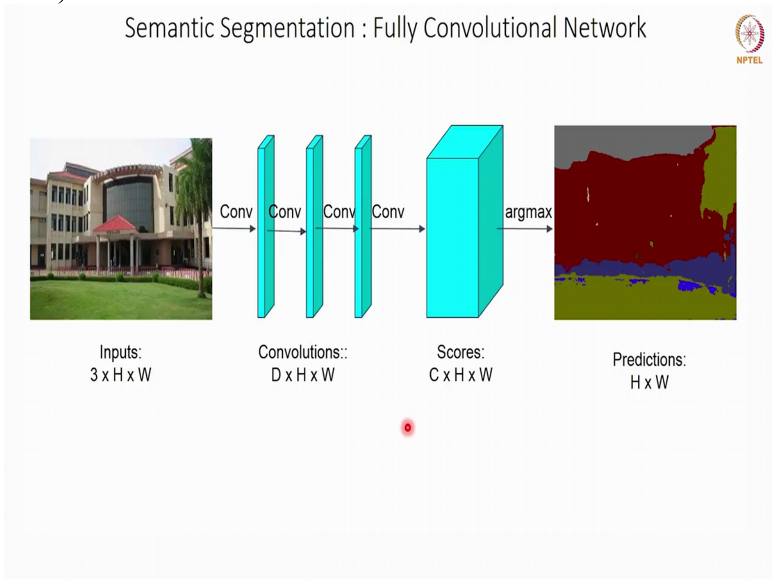
So how would you accomplish this with let us say some of the networks that we have seen so far? (())(2:36) we have looked at so far in these lectures. So first let us take this input, same scene of a building (())(2:44). So what we typically do is take patches of certain sizes, let us say 64 by 64 patches and give them as input to the convolutional network to classify it as belonging to one of N classes. So for instance, we might train a nodal network to identify sky, vegetation, buildings and objects, general objects or person and we train the network as such.

So for instance, the image may challenge at 1000 category. So we can use a similar network, train the network similarly to identify anyhow let us say 4 or 5 categories depending on the kind of scenes you are looking at. So for instance, images of parking lots, we just want to classify drive ways and cars and maybe signs. Okay. So the idea is to take patches from the image, pass it through the network, let us say a typical CNN and output the particular class, the prob will tell you the particular class, or the score for the particular class. So that way we will accumulate.

So this score is for the central pixel of that patch. So that is what is shown here. So that, so we would slide the patch, centre the patch around every pixel in image and classify that particular pixel as belonging to a particular category. So as you can see, this can lead to a lot of complexed (())(4:07) because typical images will be of the order of 200, 200 to 500, 500 to million pixels. So we will have to do 1 million for, pass it through the network in order to label every one of the pixels. Of course, you can subsample the image and do it but that will lead to errors in the semantic segmentation.

So the issues as stated earlier, it is very inefficient in terms of competition. And there is no feature setting among the overlapping patches, in the sense that the network only looks at that particular patch and it does not pay attention to what is around that. So ideally to make it more effective, you end up taking larger and larger patches, it is of course between that, you into doing more competition and the results will take a longer time to be output.

(Refer Slide Time 4:57)

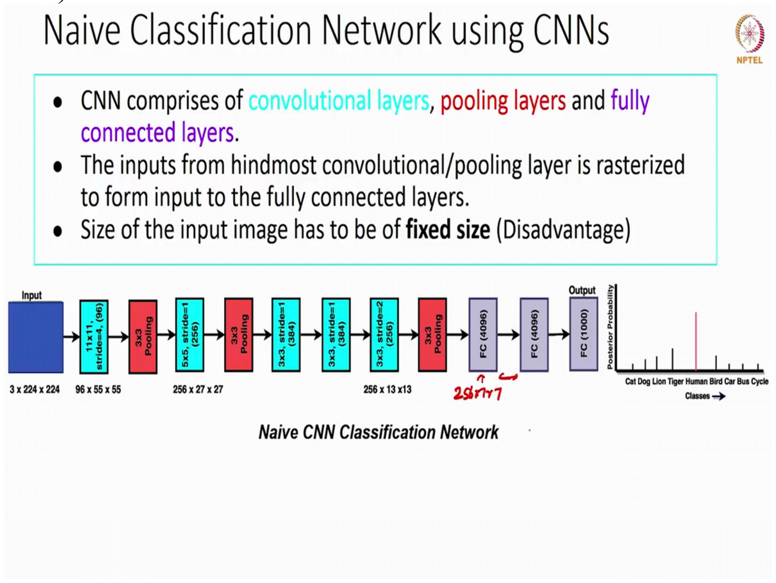


Semantic Segmentation : Fully Convolutional Network

So what would be ideally what? So that is why we are now slowly moving into this fully convolutional, we received word that is in a few moments. What will be ideally what? We want an input image, let us say of size 6 cross W with 3 channels, this is a typical RGB image and we would like to send the image through a series of convolutions and have an output layer which has as many feature maps as there are objects that we want to detect and each of those feature maps correspond to a score which gives the probability that a particular pixel in the map corresponds to a particular object.

So in the end, we take the R max across the feature map and assign labels to every pixel in the input image. This is typically what we would like to accomplish because this is more, this will be more efficient. And another advantage would be since if you make it convolutional, like we see here, we have left out Max pooling layers and all that and as it is convolutional, we can give inputs of any size. Okay that is another advantage. So we do not have to be arrested to a fixed size image. So if you look at the method we explode in the previous lights, it is basically we have a CNN which takes as input the particular size patch and then it classifies the centre pixel. Here, there will be no such restriction, that is typically what we would want and this is the aim for the fully convolutional network.

(Refer Slide Time 6:27)



Naive Classification Network using CNNs

- CNN comprises of convolutional layers, pooling layers and fully connected layers.
- The inputs from hindmost convolutional/pooling layer is rasterized to form input to the fully connected layers.
- Size of the input image has to be of **fixed size** (Disadvantage)

Naive CNN Classification Network

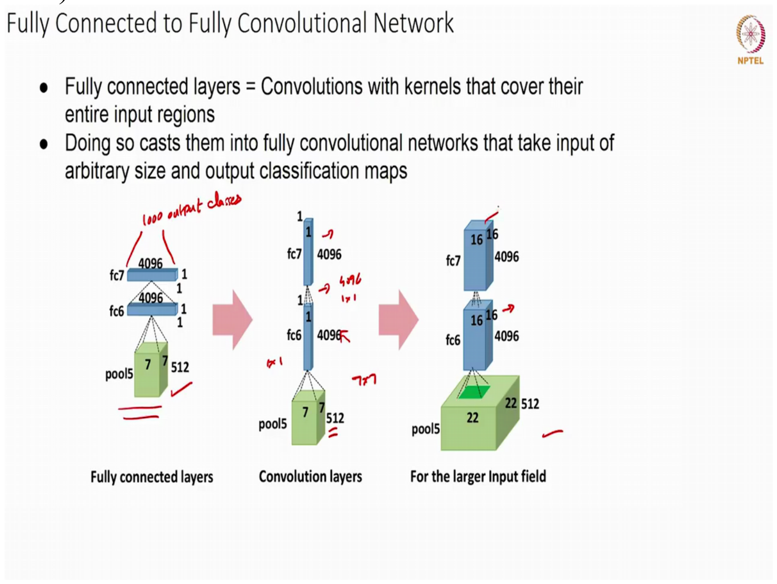
Now if we look at the, so let us go just a little bit more detailed as to how these things work, okay. So what we want is we have an input image, a typical classification network is what we want to use for semantic segmentation. So we would take patches from the input image and feed it to the CNN. Again this is Alex slide shown here. Alternative convolutional max pooling areas. However what we do see here at this layer here, the 3 cross 3 pooling layer followed by a fully connected layer, AFC is fully connected layer. Here what happens is here after Max pooling, its size is size would be 256 cross 7 cross 7.

That will be the size of the volume after Max pooling. Now this is fully connected to the 4096 neuron. So in order to do that, we have to rationalise it. So 256 cross 7 cross 7 is a rationalised into one vector and it is fully connected to the next layer which has 4096 outputs. So this is what prevents us from having an arbitrary sized input. Okay. Because if we have let us say a slightly larger input than this will be difficult to do because the number of weights here are fixed. Right?

So how many ways are there? So we calculate 256 cross 7 cross 7 times 4096. So this weight matrix, is the size of the weight matrix, number of elements in the weight matrix. Right? Now if we have let say a larger image, 224 cross let us say 400 cross 400, then what happens here is that this feature map size would increase, okay this number would be much larger. The size of the feature map will increase. Number of channels would be the same, size of the feature map should increase but then it is not possible to propagate beyond this point because the number of weights are fixed here. Okay.

So this is a restriction that comes in with whatever networks we have looked at so far is that size of the input has to be fixed. So we cannot use arbitrary input sizes. So we, in order to have a very flexible semantic segmentation framework, we would like to have a network which does not have this limitation.

(Refer Slide Time 8:52)



Fully Connected to Fully Convolutional Network

- Fully connected layers = Convolutions with kernels that cover their entire input regions
- Doing so casts them into fully convolutional networks that take input of arbitrary size and output classification maps

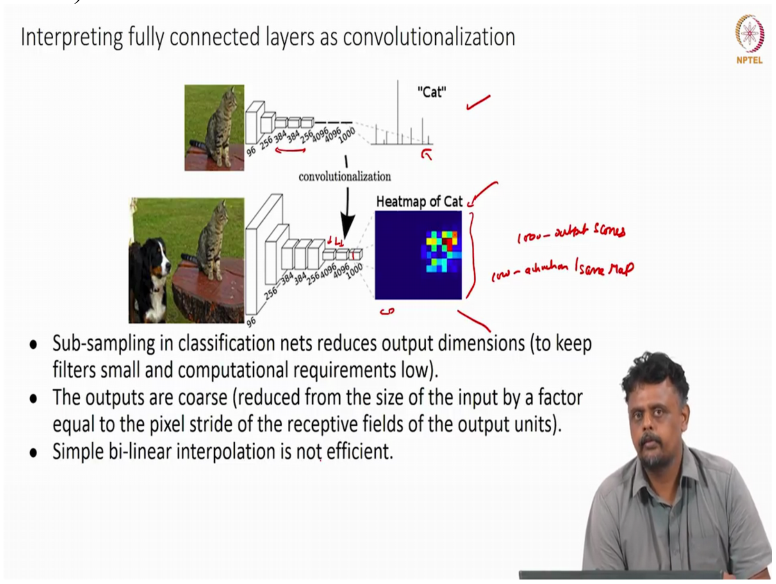
So what we do is to convert these fully connected layers into what are called fully convolutional layers? So how do we go about doing that? So let us just take this typical block of input coming into a fully connected network, right? So this 7 cross 7 feature maps about 512 of them and we rationalise this and fully connected to 4096 neurons which are in turn are again connected to 4096 neurons. Then we have let us say this was like Alex net we will have this will be connected to 1000 output classes. Soft Max, 1000 (())(9:42) soft Max.

So how do we convert this into a fully convolutional network? So we would not accomplish that, we would just define filter convolutes that covers the entire input region. Right? So for instance, we can define 512 filters or excuse me, so in order to convert this particular layer into a fully convolutional layer, what we would do is define 7 cross 7 filters, of course the depth of the filter is 512. We will define 4096 of them. Okay. So this will lead to 4096 one cross one feature maps. And then in turn, we would again define 4096 1 cross 1 filters whose depth is again 4096 and we will get for those 96 outputs. Okay. So this is how one would convert a fully connected network to a fully convolutional.

That is, to just before the fully connected layers, you define filters which should cover the entire input region. And then the fully connected regions themselves we would convert them into 1 cross 1 convolutional layers. So now the network is fully convolutional. So what advantage does it produce? So let us say for this particular network, we have an input that is coming in, which has 22 cross, this particular layer gets a larger input let us say because your input size is different, 22 cross 22 cross and 512 channels.

So you can see that since we have defined 7 cross 7 filters, we get a 16 cross 16 output. Again the number of channels is fixed for over 96. And again by doing 1 cross 1 convolutional, you get 16 cross 16 output feature maps, okay. So what that is accomplished now we have to see how we can use this for semantic segmentation. So what do these feature maps represent and in fact do they represent anything at all and how we would get to the point where we can semantic segmentation?

(Refer Slide Time 12:10)



Interpreting fully connected layers as convolutionalization

- Sub-sampling in classification nets reduces output dimensions (to keep filters small and computational requirements low).
- The outputs are coarse (reduced from the size of the input by a factor equal to the pixel stride of the receptive fields of the output units).
- Simple bi-linear interpolation is not efficient.

So let us see how ’s see the difference between image recognition and after fully convolutional, fully convolutionalisation, what does the output mean? Okay. So for a image recognition network shown right here, you would have as input an image and in this case what you are shown is a sequence of convolutional layers and these are the fully connected ones right here.

4096, 4096 and 1000 output classes. So these are the fully connected layer. So you would get 1000 output probabilities of scores and you would assign the label corresponding to the highest probability.

So in this case, the highest probability corresponds to cat, so you will call it a cat. Now let us say we have an image, in this case we have now a fully convolutionalised nodal network like we saw earlier. So instead of so going from the 256 to the fully connected layer, we define filters which spans the area of the input volume. Okay. So then once you have done that, then as we, we have a slightly larger image as shown here as input. Then we would get, again for 4096 feature maps of a given size and each of these we can again do a convolutional layer to give rise to 1000 feature maps, 1000 output maps.

And each and if we look at one particular map there, so let us so these are feature maps, these 2 feature maps, 4096 feature maps of particular size depending on your convolutional (())(13:56) and we have this 1000. So these 1000 output scores. Now previously in the conventional CNN, these were just 1000 what, single vector of size 1000 but in this case, there will be 1000 what you call activation maps or score maps. Each of these 1000 outputs is itself a, like a feature map or a map which contains the probabilities assigned to every pixel.

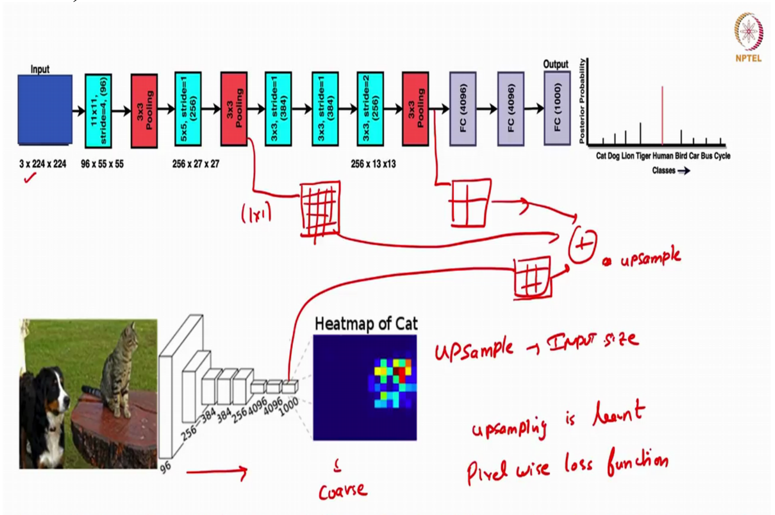
So if we take a slice of this 1000th and look at it, this is what is shown here, it will be like in the image wherein the values should the probability of every pixel being a cat. Okay. So this is how you, so we are converting from an image recognition, fully connected image recognition network to a fully convolutional network. The output scores themselves are like images with the pixel values of those images being the probability of that particular class.

So the Nth map will correspond to cat and if you pull that out and look at it and inspect the values, it will show you that each of these would correspond to, the probability of that pixel correspond being a cat okay. So now we have, for a input image we have a slightly, so if you look, this image is smaller than the input image. So we have a slightly coarser subsampled version of your probability scores. Okay. So this is the problem that we will address.

So we know that because of the subsequence layers of, sequence of layers of subsampling present in a typical network, the output score map will not be the same size as the input image.

So it will be a subsampled version. So the probabilities that are produced are also coarse. So of course the most logical thing to do would be to up sample this by interpolation. So you can up sample this to get the same size as the input with the every pixel showing the probability of the class.

(Refer Slide Time 16:18)



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.

So how do we actually accomplish that, this subsampling? That is the question, right? So we have once again we have this typically image recognition network and by turning the fully connected layers into convolutional layers, we opt in 1000 score maps corresponding to let us say, they are dealing with image like data, so it is okay to let us consider 1000 category. So 1000 score maps, but then these maps are coarse. What do I mean by course is that the size of the it is much smaller than the size of the input image, right?

So what we want to do ideally then is to up sample this to get to input size. So the simplest way as I mentioned earlier is to do by inner interpolation. You do not have to learn it. In fact you can have a (())(17:18) by interpolation. But that is not very efficient because in the process of downsampling throughout this network, you have lost some information. So one of the earliest papers, one of the earliest papers to do fully convolutional networks for semantic segmentation, what they did was to pick out feature maps from the intermediate layers and add that to the output.

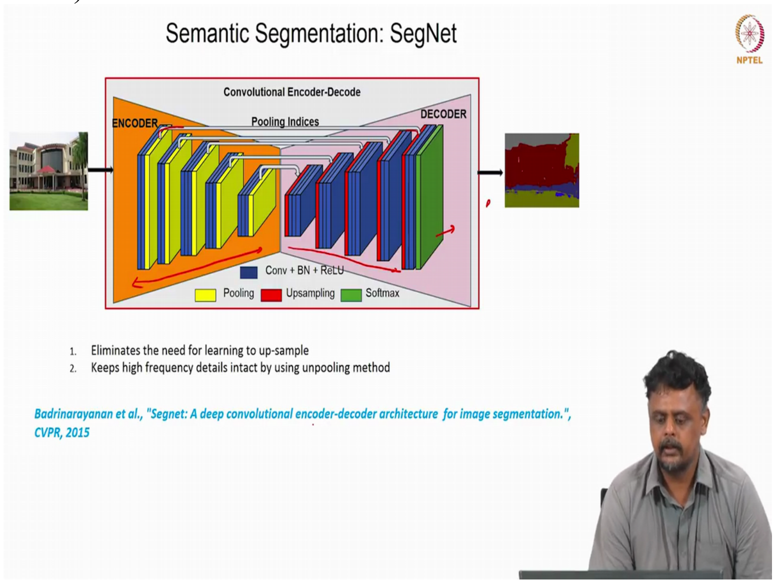
So how is that accomplished? So for instance let us consider the feature maps from this 3 by 3, okay. So they are of a particular resolution and workout exact numbers on your own but just for the sake of illustration. We can also pull out feature maps from this particular layer, they are of a particular resolution. Right? So, and then we have the output from here which again is of a particular resolution, let us say. Right? So the idea is to up sample each of them.

The idea is to opt sample each of them and added, okay. So what that accomplished is that the resolution have lost by going deep into network, (())(19:08). So how does this? So but then we will have multiple feature maps in the output of a 3 cross 3 pooling operation. So the way to do that is you would do 1 cross 1 doing1 cross 1 convolution to get one feature map and then you would do bilinear into up sample all of them and add. This up sampling can be learnt. Okay.

So this is what we saw as transpose convolutions in earlier lectures, that is what it enables. Means by up sampling feature maps from the earlier layers following a max pooling and adding it to the output layer again we still have to, this the output feature map, the 1000 output score maps are actually quite coarse, we still with the laptop sample them a bit in order to match the resolutions of the feature maps from the earlier layers. But we add them and produce one score and we do a pixel wise cos function.

So your loss function would be summation over the pixels in output score maps and summation over the number of elements in, number of samples in a mini batch and then of course back propagation is the same as earlier. Since we are using transverse convolution, again the it is just like the operation is inversed but still the back propagation of the algorithm will work seamlessly.

(Refer Slide Time 20:49)



Semantic Segmentation: SegNet

1. Eliminates the need for learning to up-sample
2. Keeps high frequency details intact by using unpooling method

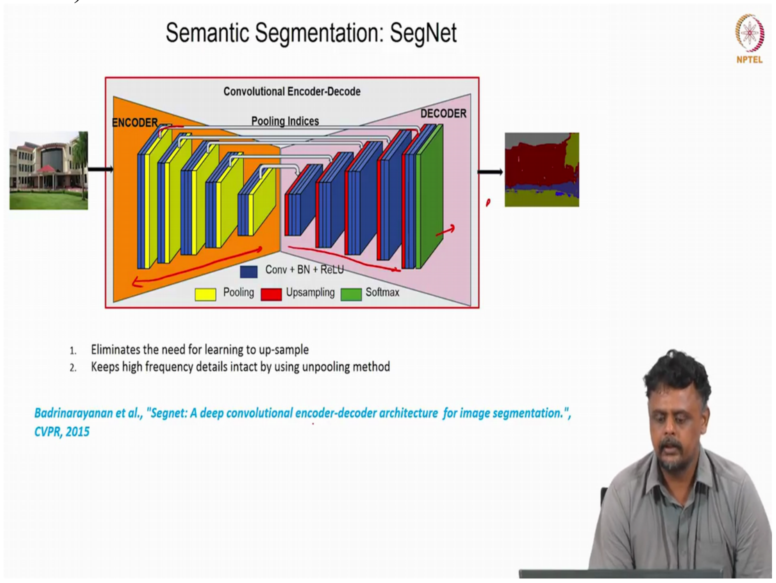
Badrinarayanan et al., "Segnet: A deep convolutional encoder-decoder architecture for image segmentation.", CVPR, 2015

So was one of the earlier techniques that was used but following that, other technique that was proposed and it is now very popular, it is very similar to that what we saw earlier, by pooling feature map from the earlier layers which is again we can call them skip connections or shortcut connections, okay and that too seem to work. However, in the recent past, the most common architecture is the encoder decoder network. So how this works is basically we have an encoder, series of convolutions and max pooling layers and after we need just specific size of feature map, we would once again do a decoder using again a series of convolutions and max unpooling XS scroll.

If we do that you get a feature map of the output scores would be the same size as the input. Okay. So the idea behind the encoding layer is to extract the most relevant features from your input and the decoding uses that to produce a pixel wise labelling of your input. So this encoder decoder structure is what is typically used for semantic segmentation. The advantage here is that as you can see, there are these cross, we look at what these are but we have these connections which take input from the earlier layers in the network and then pass them on to the corresponding layers in the decoder.

So you can see that there is a symmetry between the encoder and decoder. We will have similar sequence of convolution and max pooling in the decoder as well as the encoder side. You can say that the decoder side is the transpose of the encoder. Okay. So here, in this particular network, this uses a technique called max unpooling which eliminates the need for up sampling. Okay, need for learning to up sample. So you saw in the previous slides, the technique will look at was it has to learn to up sample the final output score, score maps as well as to up sample the feature maps of the earlier layers in network so as to match the input.So there there is some learning required, here they get over it by doing something called max pooling.

(Refer Slide Time 23:24)



Semantic Segmentation: SegNet

1. Eliminates the need for learning to up-sample
2. Keeps high frequency details intact by using unpooling method

Badrinarayanan et al., "Segnet: A deep convolutional encoder-decoder architecture for image segmentation.", CVPR, 2015

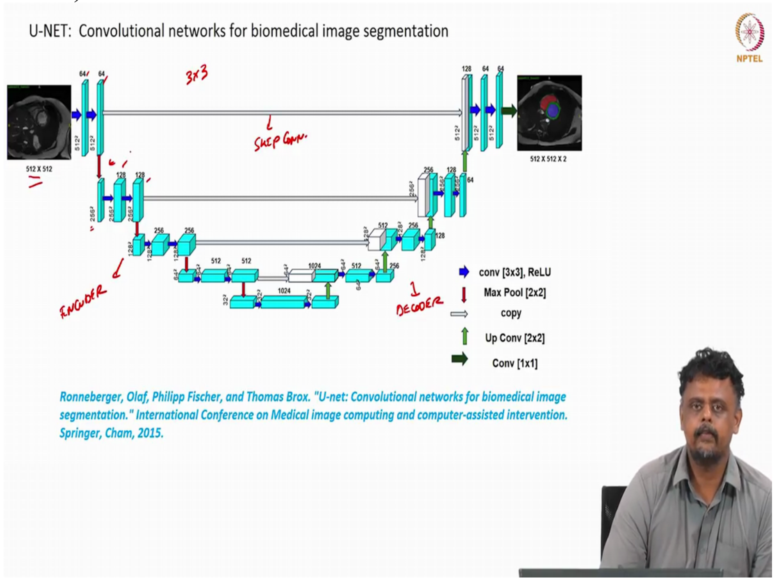
So how does max unpooling work? So here let us say this is a feature map. This is just a drawing example. Size 4 by 4. So if we do a 2 by 2 Max pooling of size 2, 2 cross 2 then in every block here, the red block here gives you 9, the green block here gives you 8, heritage 15 and 10. So those are recorded and this is your pooled feature map. However while doing that, we also keep track of the indices. So if you restore it so for instance 9 would be if you 1, 1 means what this is 1, 1, 0, 0 in each of the sub blocks. Right?

So for instance, here this is 1, 1 in this 2 by 2 matrix. This is 0, 0 in this 2 by 2 matrix. And 10 is 0, 1 in this 2 by 2 matrix. And 1, and if we say 15, 15 is 1, 1 in this 2 by 2 matrix. Okay. So if we keep track of this index, now we come to the unpooling layer. So where is the unpooling layer? Unpooling layer is on the decoder side where the feature maps are of the same size. When you come to the unpooling layer side, then you would make this larger feature map array and assign the values to the corresponding locations. Okay.

So for instance, you will have a sequence of convolutions on the decoder side. Following the convolution, you will have a max unpooling. So you will take the output of the convolution, so is the output of the convolution and assign it to a larger array, to an array with corresponding to the indices shown here. Okay. So by keeping track of the Max pooling indexes, you should be able to construct this matrix by assigning the values. Here just for the sake of illustration, I have just taken images, taken the values from a 4 cross 4 and assigned it to another 4 cross 4, we have to do something similar but just by but we have to keep track of the indices here, okay.

So now once the max unpooling is done, in this case by assigning the values to the larger matrix as indicated, we can then proceed to do convolution. The convolutions are done as in a usual conventional nodal network and of course the network learns to fill in the activations appropriately..

(Refer Slide Time 26:18)



U-NET: Convolutional networks for biomedical image segmentation

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

And other architecture which has again found wide applications in biomedical image segmentation tasks is called the U network. So this network is again very similar, so you have in this case the encoder and a decoder side here. And we have this skip connections between the encoder and decoder. This particular network again does transpose convolutions, so it does learn the up sampling. But it does go deeper than most networks and because of the skip connections, the generally the outputs are of a much higher quality.

So let us start with lets a typical input of size 512 by 512, here I have shown an image of cardiac MRI slice, size 512 by 512. So you would produce 64 feature maps, a succession of 2 convolutions produce 64 feature maps with padded convolution so that the size is retained. Here throughout this network, similar to VGG, we do 3 cross 3 convolutions okay. So the red indicates max pooling. So your 5512 cross 512 will become 256 cross 256, number of channels we mean to say.

Once again you will do a sequence of 3 cross 3 convolutions to produce again similar to VGG you have two 128 feature maps. The initial layers, you have 64, we have 128. And then followed by Max pooling, okay. And then 3 cross 3, again we will do 3 cross 3 convolutions to provide 256 feature maps. This has a max pooling layer there which gives you 64 size feature map. And then once again to convolution to produce 512 feature maps, to successive convolutions and then followed by a max pooling layer.

So you can go across this network, I have just shown you on one side, on the encoder size. So on the decoder side, again you would do, you would exactly duplicate this operation but then instead of convolution, you will do instead of the max pooling, you will do the up sampling. So here, it is max pooling on this side, on the encoder side you will have you will do up sampling. Again the convolutions are again the same as 3 cross 3 convolutions and you would systematically reduce the number of feature maps as you go towards the output side.

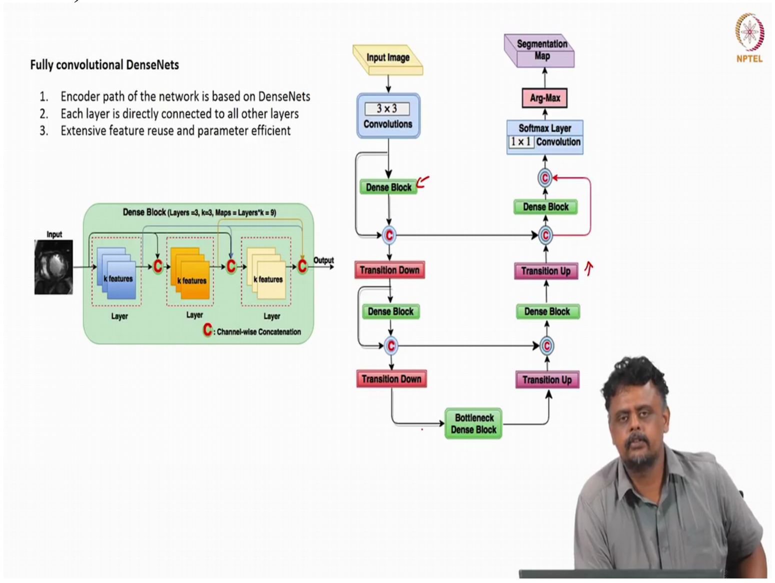
Now in between, you have, you would copy in between for instance as shown here, these grey lines show that you would take these feature maps from the encoder side and contact made them to the decoder side. Again here, there are multiple options. You can concatenate, here it is shown as concatenation or you can do a resonate like processing where you can just add them. That is also possible. And if the number of feature maps begins to difficult to handle, then you can just do a 1 cross 1 convolution, reduce them and add or concatenate to the encoder side.

So as you go up the encoder, one of the things I have noticed is that you would take feature maps from the decoder side at the same resolution and add it to the, sorry you take feature maps at the same resolution from the encoder side and add it to the decoder side. This is very similar as to what we saw with the fully convolutional networks, the first fully convolutional network that we saw is to we take networks from feature maps from earlier in the earlier layers and add them to the output. So this is done more systematically here.

So again at every layer in the decoder side, you would go and find a corresponding feature map of the same size and of the same resolution and add it, this case you can do a resonate like a residual connection or you can just concatenate, 1 cross 1 convolutions can also be done to sync the size of the feature maps so that you do not have a feature map explosion on the decoder side. And finally, you would produce an output with a requisite number of classes.

So you would do convolutions so that the number of channels for instance if you are looking at 5 classes, the output will have 5 channels of output with each channel corresponding to a particular category and the pixel values in that channel representing the probability of that particular class at that pixel value. So this is the very often use network for biomedical segmentation task. We will look at an application that this can be used, later on in the next few weeks.

(Refer Slide Time 31:06)



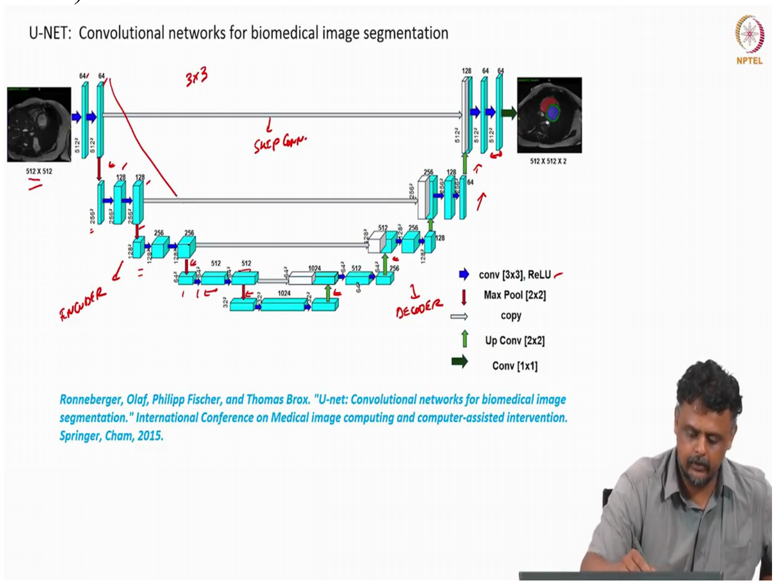
Fully convolutional DenseNets

1. Encoder path of the network is based on DenseNets
2. Each layer is directly connected to all other layers
3. Extensive feature reuse and parameter efficient

Of course we have also seen dense nets. So what we saw, we can do similar processing with dense nets. So instead of having these VGG convolution blocks, all you have to do is replace the convolution blocks with a dense block and instead of Max pooling, we have the transition blocks we saw, as we had seen earlier when we were discussing dense net. We used the transition down blocks to bring down the resolution on the encoder size. Similarly, we have transition up blocks on the decoder size to improve the resolution.

So this has the same thing like we do deconvolutions or transpose convolutions as they are called. Deconvolution is apparently a wrong nomenclature because deconvolution is well-defined (())(31:50). So transpose convolutions are a more appropriate terminology to be used. So the only difference between this network and the one we saw earlier in U-NET is that the VGG type convolution blocks are replaced by dense blocks. So we can, so this particular architecture which is basically the encoder and decoder, the idea is we can use any of the network topologies we saw, architects that we saw in the which were used for the imaginative channels. And but of course the only thing we have to do, on the decoder side we have to formulate the appropriate channels or the architecture.

(Refer Slide Time 32:33)



U-NET: Convolutional networks for biomedical image segmentation

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

So for instance, this is to just to illustrate again, if you go back to the U-NET architecture, you see that this is basically this a this is nothing but your, this particular whole thing is nothing but your VGG type block. Okay. So we have 64, 264, root 128 here and then we have two 256 and then 512 and so on so forth. Of course VGG had three 512s blocks. But we can have a VGG or Alex net on the encoder side and then we will just have to flip that on the decoder side. Of course, it is no guarantee that all of the network techniques should work. This particular architecture actually works. We that for sure, they are being published and applied to many image processing tasks.

In the next series of lectures we will consider transfer learning. Again, we will see how we can use this pretrained the networks to and apply them for related tasks and how that will in the case there is not enough data, so and also followed by that we will also look at hyper parameter optimisation.