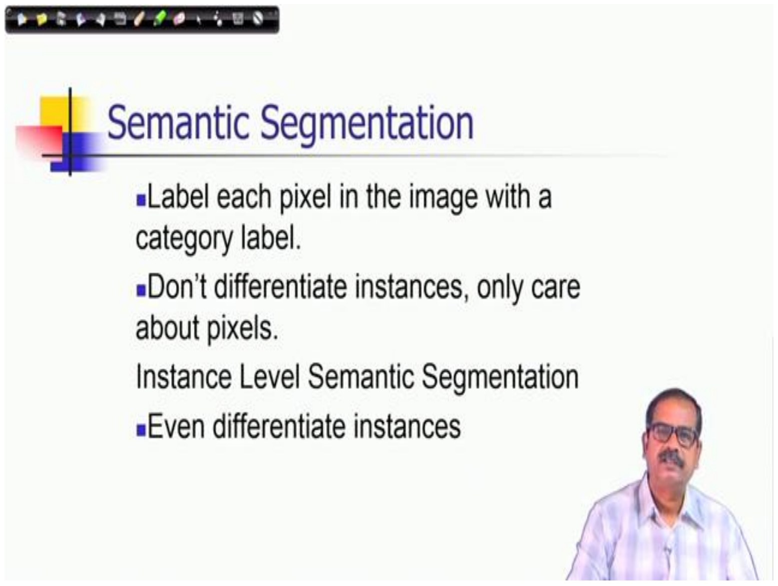
Computer Vision

Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 59
Deep Neural Architecture and Applications Part – V

We are discussing about different kind of operations or different type of tasks that we can perform using deep neural architecture. In the last two lectures we discussed about task of classification of objects and also task of classification as well as localization of objects. In this lecture we will discuss about the segmentation of images.

(Refer Slide Time: 00:41)



Semantic Segmentation

- Label each pixel in the image with a category label.
- Don't differentiate instances, only care about pixels.

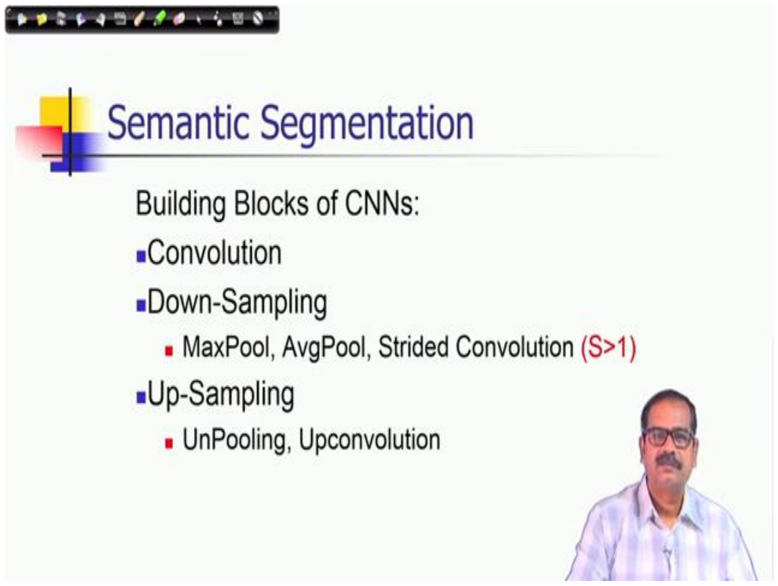
Instance Level Semantic Segmentation

- Even differentiate instances

So this kind of segmentation is called semantic segmentation, with respect to this particular architecture; because the task here is that we would like to level each pixel in the image with a category label. So, you can see it is also akin to classification, but instead of classifying the objects or a group of pixels, we are trying to classify each pixel. And with that also you are assuming the segmentation of images that we understand what is segmentation, that grouping a pixels of same classes or same categories.

And it does not different differentiate instances only care about pixels; that means, if there are say 3 human profiles and they are all connected profiles. So, it is sufficient if you denote the segment as all 3 connected as human profiles. But there could be instance level semantic segmentation also. So, even the task could be that you have to differentiate, even individual humans in that connected segment of human profiles.

(Refer Slide Time: 01:47)



Semantic Segmentation

Building Blocks of CNNs:

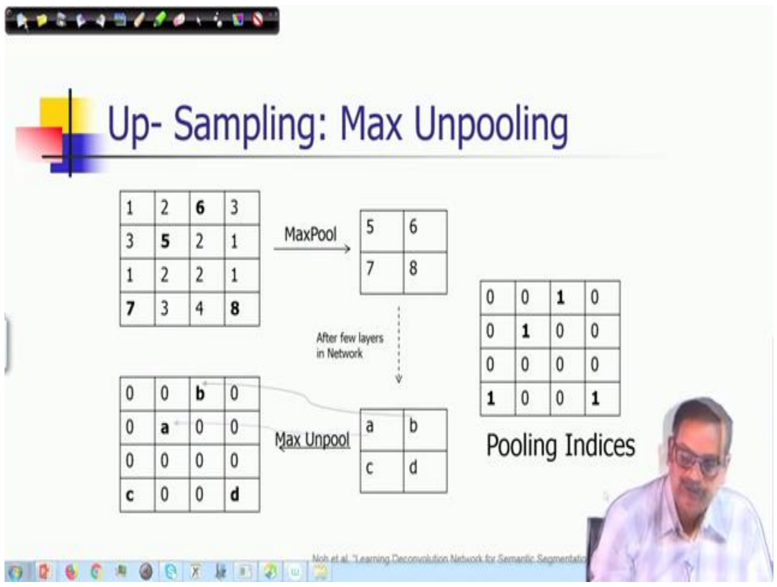
- Convolution
- Down-Sampling
  - MaxPool, AvgPool, Strided Convolution (S>1)
- Up-Sampling
  - UnPooling, Upconvolution

So, in a semantic segmentation, the building blocks uses convolutional neural network. And these are the certain things some of the computations what is being carried out standard convolution. Then it is a task of down sampling because it tries to learn the features with aggregations at different resolutions.

So, in down sampling, as we know that we can perform down sampling by this different kinds of processes; like you can do pooling, max pooling, average pooling even you can use strided convolutions, which means you leave some of the pixels and do not do convolution there. And in the output only you consider those pixels where you have convolved the image and that itself will down sample the image. And then there is a stage of up sampling because finally, in the segmentation the input and output both the sizes would be same; because every pixel of the input has to have some semantic level. So, this learned feature are again used and it is projected in that semantic space.

So, those are leveled those are learnt this projection is also learnt and that is up sampling task. So, there are operations like unpooling and upconvolutions I will explain those operations we understand the maxpool or pooling operations and the down sampling operations In this lecture we will discuss what is unpooling and up convolution.

(Refer Slide Time: 03:25)



Up- Sampling: Max Unpooling

1	2	**6**	3
3	**5**	2	1
1	2	2	1
7	3	4	**8**

MaxPool

5	6
7	8

After few layers in Network

a	b
c	d

Max Unpool

0	0	**b**	0
0	**a**	0	0
0	0	0	0
c	0	0	**d**

0	0	**1**	0
0	**1**	0	0
0	0	0	0
1	0	0	**1**

Pooling Indices

Noh et al. "Learning Deconvolution Network for Semantic Segmentation

So, let me explain this part first what is first the unpooling, max unpooling operation. Consider this particular image the example is shown here and its a 4×4 image and we would like to perform the maxpooling by 2×2 filters or its not exactly maxpooling what we are trying to do it is a kind of decimation. So, let me call no yes it is maxpooling and what we do while doing maxpooling, you also generate the index of the particular position.

See in this 2×2 block, the maximum value is here 5. So, output will have 5 from these 2×2 block and also its position in this 2×2 block has to be also stored and that information also you should be kept so that during unspooling, we will be using that information.

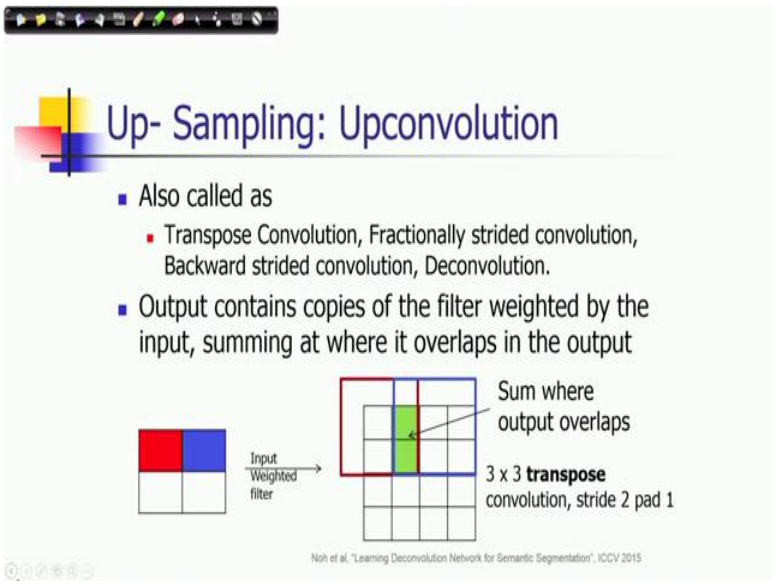
So, as you can see in these block 6 is the max in this block 7 is the max which these are shown in the bold font and 8 is the max. So, the maxpool operation will give you 5, 6, 7, 8 and also as it is shown here. And also the position can be stored in a binary mask; where the positions where the value from the value position from the position from here. The value is taken those position is indicated by 1 and all other positions are 0.

So, its a kind of indication indicator functions, we call this or a as pooling indices. So, this two things goes hand in hand. So, you are not only computing the maxpool operation and also you are computing the pooling indices. So, this pooling indices are used during the unpooling operations. So, how it is used? Let us see now the input to your unpooling operation is the values of maxpool operation; that means. Now this is your input and also it would be added by pooling.

So, if I consider max unpooling, then this is your input and also this is also your input because you will be using this indices to finally, generate the output. So, after a few layers of network, which means in the processing pipeline the unpooling operations can come later. So, you should tag the pooling indices to that layer. This is a output or this is the input to your maxpool through your un-pooling; because after a few network a few layers in network these values will change it is not the same value.

And suppose these values are shown here by these variables a, b, c, d. Then the unpooling operations max unpool will be like this. So, you can see that using pooling indices, we are putting the values to the corresponding locations we are positioning. So, a is positioned here from here and this is also guiding the position of a similarly position of b. So, this is the position of b; with respect to this block and you see that with respect to this block b is put here. So, in this way we are doing a max unpooling. So, this is how they are showing.

(Refer Slide Time: 07:15)



Up- Sampling: Upconvolution

- Also called as
  - Transpose Convolution, Fractionally strided convolution, Backward strided convolution, Deconvolution.
- Output contains copies of the filter weighted by the input, summing at where it overlaps in the output

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Now let us understand how we can perform up sampling and up convolution. You consider input to a block, input to the up sampling up block as I say 2×2 input here the colors are shown of the input. And there is a weighted filter from there. So, what we do that for each input we are multiplying this filter. So, so the idea is that output contains copies of this filter weighted by the input. And its summing at where it overlaps in output.

So, the let me explain it suppose your input here is given the value say $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and for each input we have a 3×3 weight filter; say 3×3 weight filter we will have say $\begin{bmatrix} x & y & z \\ u & v & w \\ p & q & r \end{bmatrix}$ So, what we will do and say v is a central place. So, far 1 will be generating the output as x, y, z, u, v, w, p, q, r then you consider 2; so the place of 2 if I consider the place of 2. So, so this would be the place of 2 and from there we will be using.

So, this should be I should say 2 x plus, 2 y plus, 2 z then v would be this is 2 u plus 2 v plus 2 w. So, this plus is this is not plus because this is the new these is the new position and still there was 0 earlier. And this is q plus 2 p r plus 2 q and 2 r so; this is how the 2 is used. Similarly for 3 we have to perform the same job; that means, if I consider the position of 3.

(position 2 become the following)

$$\begin{matrix} x & y+2x & z+2y & 2z \\ u & v+2x & w+2v & 2w \\ p & q+2r & r+2q & 2r \end{matrix}$$

Central position of 3 is here that is what you need to take a when you are considering position of 3; because you have to place the array and place this thing as the central positions and accordingly in each pixel locations you are doing it. Similarly when you place the central positions 3, then you will find that u will be 3 this will be 3 x. This will be added with 3 y this will be added with 3 z, this will be added with 3 u this will be added with 3 v, 3 w.

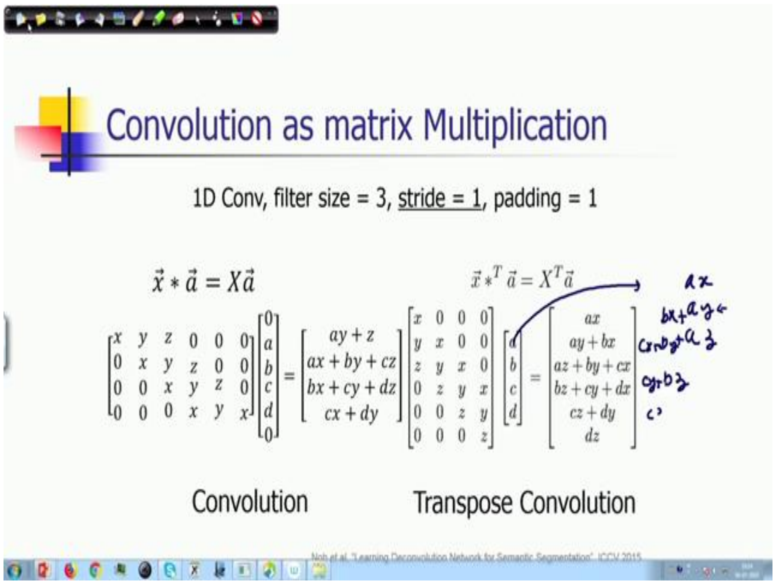
$$\begin{matrix} x & y+2x & z+2y & 2z \\ u+3x & v+2x+3y & w+2v+3y & 2w \\ p+3u & q+2r+3v & r+2q+3w & 2r \end{matrix}$$

And so, this is about 3 and also the lower row will come which I am not computing showing here. Similarly for 4 it should be this location. So, this would be 4 v once again this would be 4 x plus, 4 y plus, 4 z and this value will be 2 r it is 4 w. And this value would be here it is 4 u and also here the low bottom row I am not showing. So, in this way what you are doing you are you are coping the filter weighted by the input.

$$\begin{array}{cccc} x & y+2x & z+2y & 2z \\ u+3x & v+2x+3y+4x & w+2v+3y+4y & 2w+4z \\ p+3u & q+2r+3v+4u & r+2q+3w+4v & 2r+4w \end{array}$$

So, you are placing all those things in those locations and in the overlapping locations you are summing it up that is what is your output. I will do one exercise later on and you can find out how it is carried out there if you will understand fully this part. So, this kind of operation is also called us transpose convolution, fractionally strided convolution, backwards strided convolution and deconvolution these are there are different other names.

(Refer Slide Time: 11:58)



Convolution as matrix Multiplication

1D Conv, filter size = 3, stride = 1, padding = 1

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay+z \\ ax+by+cz \\ bx+cy+dz \\ cx+dy \end{bmatrix}$$

Convolution

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay+bx \\ az+by+cx \\ bz+cy+dx \\ cz+dy \\ dz \end{bmatrix}$$

ax
bx+ay
cx+by+az
cy+bz
c

Transpose Convolution

Noh et al. "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

The expression what I have given to you it can be also considered in a different form and that is very precise definitions from the mathematical form. So, if I consider 1 dimensional convolutions, then it would be more clear that what exactly you are doing we can show that why it is called transpose convolution. So, convolution operation in 1 dimension you can express very easily using matrix multiplication. And when you are doing 2 dimensionsthat you have to use if you have separable properties, then you can also perform it using matrix multiplications.

But let me first consider the 1 dimensional convolutions for the sake of convenience of our discussion. Consider a filter of size 3 and it is strided 1; that means that every pixel you are performing these computations and padding is 1. So, it is a the input is padded with 0s only at 1 pixel gap. So, and it is not pixel it is since these are samples these are 1 dimensional let me call it as sample. And then let us consider this convolution operation.

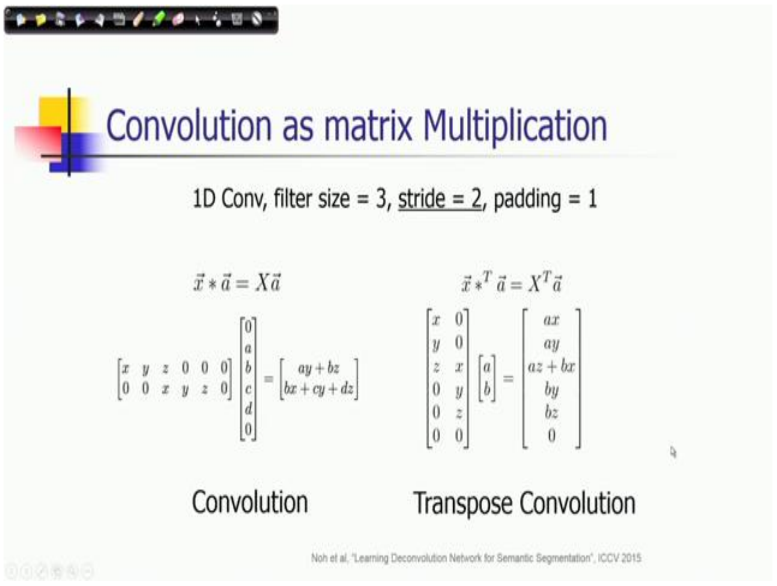
So, you can see that here the input is considered as a sequence of values $\begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix}$ So, this is the input and you are performing a convolution whose response is given by x, y, z. So, if I consider a input say a, b, c, d and we pad it with 0 and you know how convolution is perform. it is a marks x, y, z which needs to be multiplied with this and then the central value will be replaced.

So, if I consider these value it should be $ay + bz$. Similarly this marks will be shifted here x, y, z and the central value will be replaced by this b will be replaced by after convolution $ax + by + cz$. Exactly that is what is being shown here in this computation so, what you can see, we can construct a matrix where each row consider shifted response of the corresponding impulse response x, y, z it. And that matrix is multiplied with the input and you get the output of the convolutions.

So, convolution can we consider it as matrix multiplication. And we can show also this up convolution, which is nothing but what I explained that what you are doing. You are basically multiplying the filters using the input in the corresponding up sampled size up sampled array. And then add those values in the overlapping locations. So, that can be shown equivalently. So, this is what is convolution.

So, you just transpose this matrix, convolution matrix and multiply with the input a, b, c, d and you will get this up sampled value. Once again we can see that this pixel a should be replaced by as we know that it should be multiplied with ax then ay this is a central location then az. Similarly for b it would be bx, by, bz and you are adding them. For c it will be cx, cy, cz and you are going on doing these things. So, that is how you get all these values. And in fact, that operation is nothing but your transpose of convolution matrix and multiply that transpose of convolution matrix with the input a so, that is what you get it here.

(Refer Slide Time: 16:19)



Convolution as matrix Multiplication

1D Conv, filter size = 3, stride = 2, padding = 1

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Convolution

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

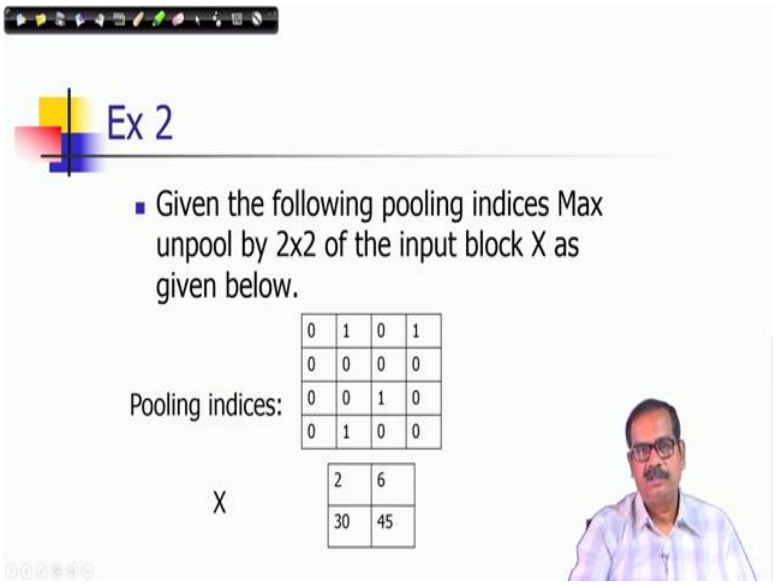
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Transpose Convolution

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Even for stride two also we can compute. Here in this case if it is stride to your computing at a then c and you can get the samples. So, you can get up sampling and upsampling with some kind of you know these upper stride two operations.

(Refer Slide Time: 16:37)



Ex 2

- Given the following pooling indices Max unpool by 2x2 of the input block X as given below.

Pooling indices:

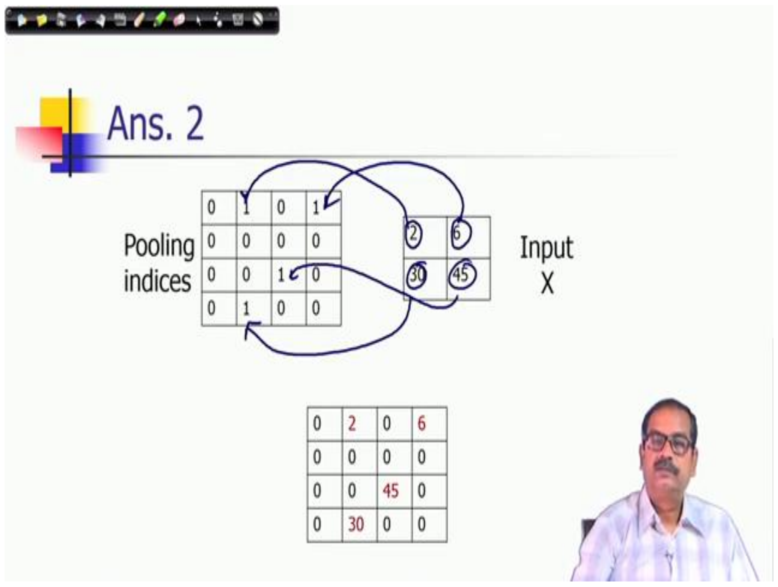
0	1	0	1
0	0	0	0
0	0	1	0
0	1	0	0

X

2	6
30	45

So, let me to explain these particular computations, let me solve one exercise. Suppose you have the following pooling indices and max un pool by 2×2 of the input block X the pooling index indices is shown here and also the input blocks is shown here. So this is your input and what you need to do? You need to unpool this particular indices. So, which is not written here, but you need to un-pool max you have to perform the max unpool or unpooling of these values X.

(Refer Slide Time: 17:19)



Ans. 2

Pooling indices

0	1	0	1
0	0	0	0
0	0	1	0
0	1	0	0

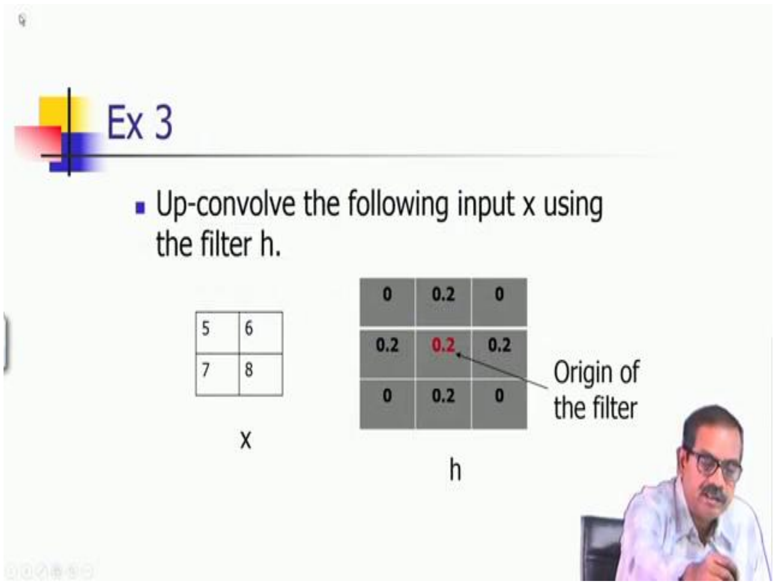
Input X

2	6
30	45

0	2	0	6
0	0	0	0
0	0	45	0
0	30	0	0

So we have already discussed how unpooling can be done. So, you have the pooling indices like this and you have the input X. And so, what you need to do you need to place the value 2 here, value 6 here, value 30 here and value 45 here, rest of the values should be 0. So, from 2×2 you are getting 4×4 image and these are the values you can get. So, this is how you get the unpooling of the input.

(Refer Slide Time: 17:57)



Ex 3

- Up-convolve the following input x using the filter h.

5	6
7	8

x

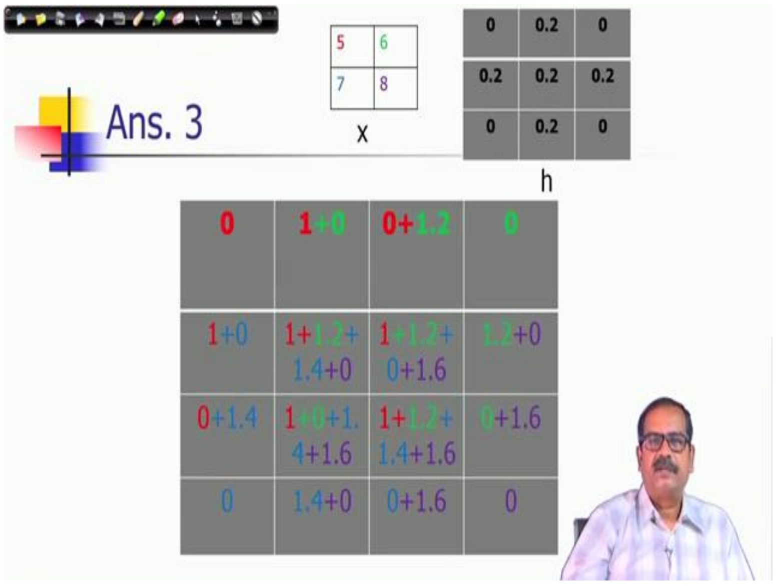
0	0.2	0
0.2	0.2	0.2
0	0.2	0

h

Origin of the filter

Let us consider another exercise that we need to up convolve the input x using a filter h. And as you can see origin of the filter is also given here. So, filter is a 3×3 filter and you have those input as 2×2 input values are shown for a toy example $\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$. So, you need to solve this problem. As I have mentioned that each input has to be multiplied with this filter in the up sampled block.

(Refer Slide Time: 18:34)



Ans. 3

5	6
7	8

x

0	0.2	0
0.2	0.2	0.2
0	0.2	0

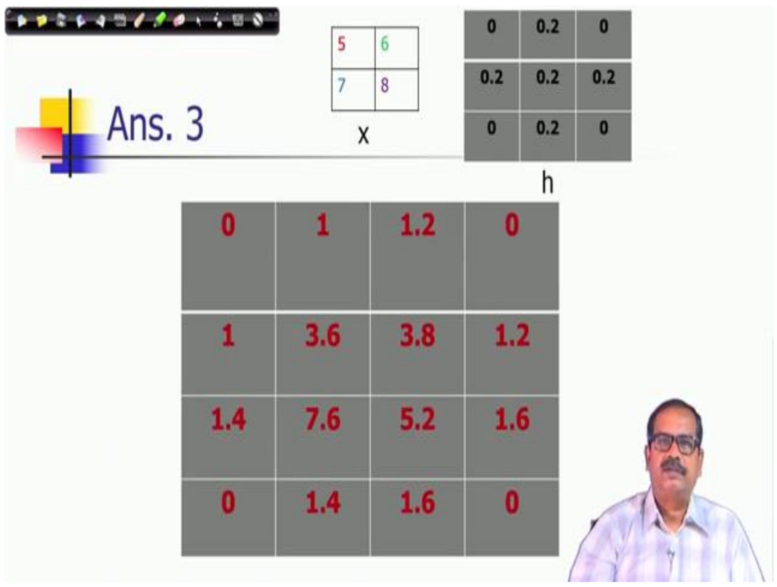
h

0	1+0	0+1.2	0
1+0	1+1.2+ 1.4+0	1+1.2+ 0+1.6	1.2+0
0+1.4	1+0+1. 4+1.6	1+1.2+ 1.4+1.6	0+1.6
0	1.4+0	0+1.6	0

So, we need to solve this problem. So, what I need to I should do here? As I consider that I have shown here with colors the pixel values because they are contributions to the up sampled values can be shown by color. So, for example, 5 we will be contributing a 3×3 block centering at that pixel with say 5 in (2 0) locations in the diagonal neighbor and like this. So, let me show you by the add itself. So, you can see all these contributions are shown here.

So, all the red values are contributed by a 5 all the green values green values or colors here they are contributed by 6 similarly blues from 7 and violet from 8. And at respective cells respective locations you are adding all of them and this is how you are doing up sampling and if you add them, then you get this value.

(Refer Slide Time: 19:40)



Ans. 3

x

5	6
7	8

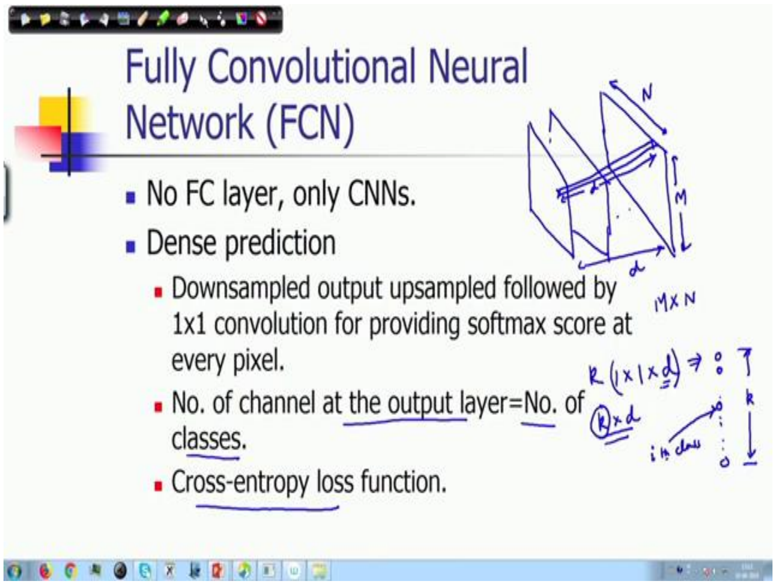
h

0	0.2	0
0.2	0.2	0.2
0	0.2	0

0	1	1.2	0
1	3.6	3.8	1.2
1.4	7.6	5.2	1.6
0	1.4	1.6	0

So, this is a output this is a up sampling value from 5, 6, 7, 8 you get this particular values.

(Refer Slide Time: 19:53)



Fully Convolutional Neural Network (FCN)

- No FC layer, only CNNs.
- Dense prediction
  - Downsampled output upsampled followed by 1x1 convolution for providing softmax score at every pixel.
  - No. of channel at the output layer=No. of classes.
  - Cross-entropy loss function.

Performing semantic segmentation, we use a bit different form of neural architecture and which is called fully convolutional neural network. So, in this case we do not have a fully connected layers as we use in traditional CNNs which is used for particularly classification of images. Here we use all the layers as if fully convolution as a convolutional neural network layers convolutional layers.

So, so finally, we have an output is also an image of certain number of channels. And usually the size of the output the number of pixels associated with that output is the same as the number of pixels of the input. And every pixel at every pixel we perform a prediction of the class to which the pixel belongs and that is why this kind of prediction is called dense prediction. And a belongingness to the class that would determine the semantics associated with that pixel and through which you can perform the segmentation.

Now, let us understand how this kind of classifications or this kind of prediction is carried out at pixel level. As we understand that in the process of computation input image is processed at several layers of convolution. And also there are downsampling operations either by max pooling or by strided convolutions or by pooling operations. So, at certain stage you will get the downsampled processed results processed arrays processed tensors. And those are to be again up sampled.

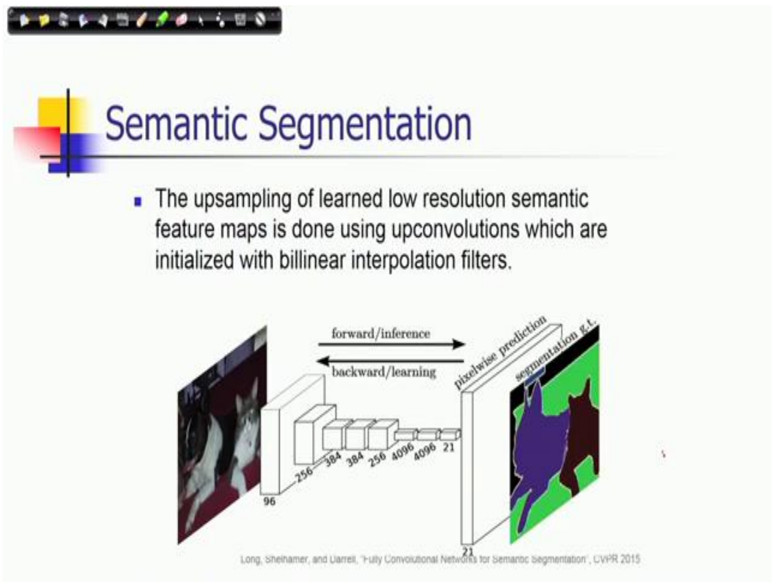
So, that it could be brought to the size of the input of the pool network. So, when we have the same size, but the number of channels at that layer at the layer just previous to the output previous to the prediction layer that number of channels may differ and let us consider an example. Suppose before dense prediction we have d number of channels let us consider there are d channels. And each of size say $M \times N$; that means, there are there are M rows and N columns; so, each of size $M \times N$ And here what we would like to do we would like to predict the level of a pixel. So, we perform a convolution with a filter if it is an 1 dimension filter of length d; effectively in the convolutional neural networks terminology we specify this filter as 1×1 filter and its length is d. So, this is one filter and that would give some response some value. And if I consider k of these filters for each pixel you will get k such response. So, there will be k such response. And each filter is associated with certain class. So, the say i th filter say associated with say i th class.

So, by using the softmax principle we can compute the probability of that pixel. Probability of a class that would be assigned to that pixel and it is the same principle that we can follow here. And these are the parameters so, these many number of parameters associated with each pixel that can be applied here I mean that needs to be learned. So, there should be $k \times d$ parameters that should be learned at this output layer.

So, as we mentioned here that number of channels at the output layer which is say k. That should be equal to number of classes that is a kind of restriction that we need to follow here because this is meant for this semantic labeling of the pixels. And for performing the optimization, here we use a loss function and since we are classifying each pixel cross entropy loss function is a natural choice in this case.

So, this is how we can use a fully convolutional neural network and that can be used for labeling every pixel of the at the output from this network. And we can perform semantic segmentation using this network. So, let us now see that how this network, what are the different kind of variations of this architecture?

(Refer Slide Time: 26:08)



Semantic Segmentation

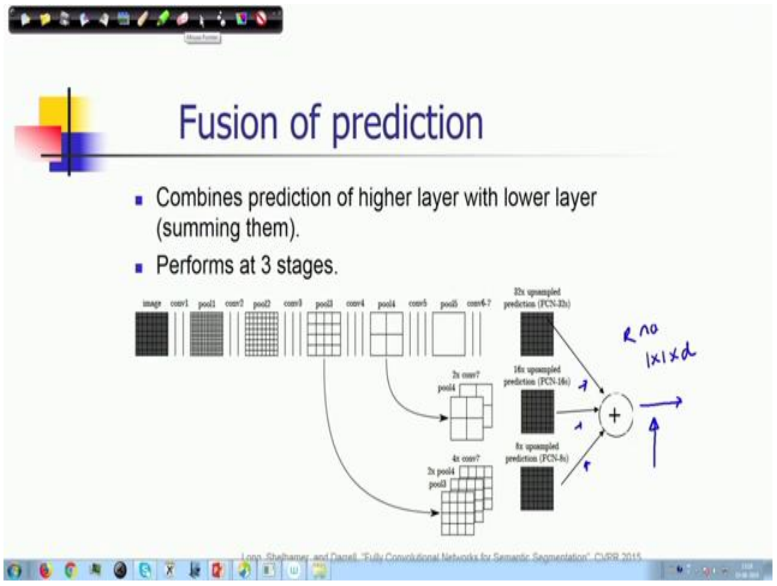
- The upsampling of learned low resolution semantic feature maps is done using upconvolutions which are initialized with billinear interpolation filters.

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

So, let us consider this particular work that is also performing semantic segmentation. And as you can see, what it is doing? It is in the process of computations it is performing the corresponding downsampling operations and also convolution operations. And also then after that it is also going through the up sampling operations at certain stages. And it has been brought to the same size of the input image and then it performs a pixel wise prediction as we have considered.

While doing up sampling, you it has considered that the initialization of the up sampled filter could be done by using billinear interpolation filters weights. That is a very standard interpolation filter whose weights are already predetermined because of the billinear operations there. So, after initialization you perform the optimization operations then those weights we will get updated. So, the next we will see what kind of operations so, this particular network is doing.

(Refer Slide Time: 27:37)



Fusion of prediction

- Combines prediction of higher layer with lower layer (summing them).
- Performs at 3 stages.

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation" CVPR 2015

So, it combines the what it does it not only takes a prediction from the last layer, it also considers predictions of some previous layers operations. Let us consider this particular example say while down sampling the images at different stages of conclusion layers, then it down samples say always it down samples by a factor of 2. So, you have a down sampled by factor of 2 here, say in this case by a factor of 4; that means, you required up sampling by factor of 2 in this case up sampling by factor of 4 to bring it to the size of the input. So, this is factor of 8 and this is factor of 16 and say this is factor of 32.

So when you are trying to bring to the same size of the input what we would like to do here, we would like to perform an up sampling of by factor of 32. And we call on this up sampled output before the classification operations that is 32 X up sampled a prediction or FCN 32 s. So, prediction can be done from here using as I mentioned by using the k number of 1×1 cross the number of channels.

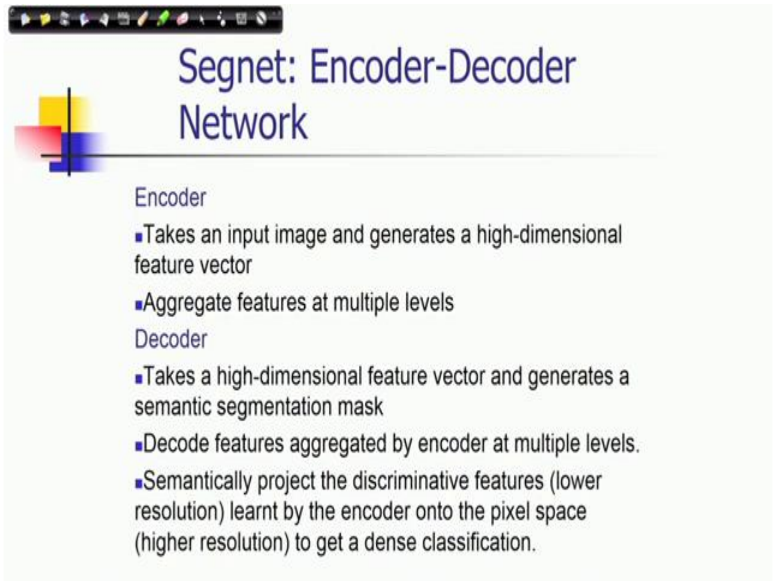
So, number of channels is not mentioned here suppose there are d number of channels at this layers at the final layer d number of channels. So, we have to predict using $1\times1\times d$ filters and for k semantic classes you have to use k number of such filters. So, we can perform prediction simply from this layer itself using this operations. But what it has been found that actually this performance could be improved.

If we use also the information from the down the lower layers for example, from here if I perform a $16\times$ up sampling and then which means from this layer it is $2\times$ conv7 that is what it means? But it is it means that you are doing $16\times$ up sampling. And then you again you also consider predicting using k number of filters at this stage. Similarly you are performing $4\times$ up sampling.

So, all those values all those values are what we are doing we are not predicting here. We are simply adding them or we are adding the corresponding values and then performing the prediction using $1\times1\times d$ filters k number of $1\times1\times d$ filters. So, if I consider this these operations it could be done in various ways. Probably in this work what it has been done, it has been applied independently for k times and then added them then the softmax softmax principle has been used.

So, you are learning these weights at all these stages and finally, the lost function is computed after fusing those values using soft max principle. And that would give variations and it has been found that that has improved the performance of this network for predict for performing semantic lebeling of the pixels. So, this is one kind of architecture, one kind of a strategy that has been followed. There are other architectures also, other variations also architectures are most similar.

(Refer Slide Time: 31:45)



Segnet: Encoder-Decoder Network

Encoder
- Takes an input image and generates a high-dimensional feature vector
- Aggregate features at multiple levels

Decoder
- Takes a high-dimensional feature vector and generates a semantic segmentation mask
- Decode features aggregated by encoder at multiple levels.
- Semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification.

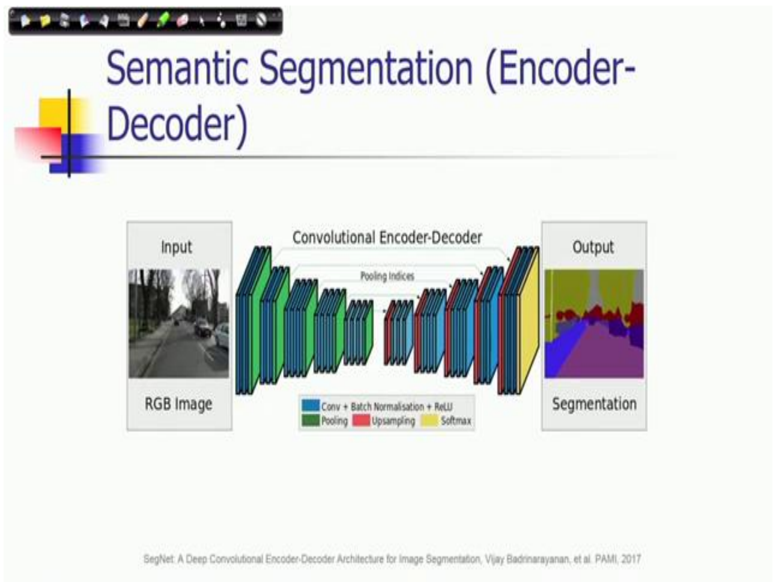
This architecture is called encoder decoder network and it is called segnet. So, what it does at the encoder level, it takes an input image and it generates a high dimensional feature vector. And also through this process it aggregates features at multiple levels.

Whereas, at the decoder it takes a high dimensional feature vector and generate a semantic segmentation mask.

So, in this process we will see instead of a the difference is that instead of up sampling using up sampling filters. In this network the un pooling operation has been used as we discussed earlier how un-pooling could be done; because the idea here is that by doing unpooling you are preserving the location. So, the high responses as we observe during the encoding stages.

So, those are those information are passed to the decoders and that helps in improving the performance that has been observed. So, so, the decoder it decode features a aggregated by encoded at multiple levels that is what its a reverse operation. And as you can see that it projects the discriminated features, which are being learned by the encoder to get it dense classification in the pixel space or in the original image.

(Refer Slide Time: 33:23)



Semantic Segmentation (Encoder-Decoder)

SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, Vijay Badrinarayanan, et al. PAMI, 2017

So you can see that what are the processing here. So, you have these are the encoder stages and at every in encoding stage; that means, these are the convolutional layer. And the there are actually 3 convolutional operations and then there is a pooling operation max pooling operations. So, after every max pooling naturally there is a down sampling operations so, after pooling operations.

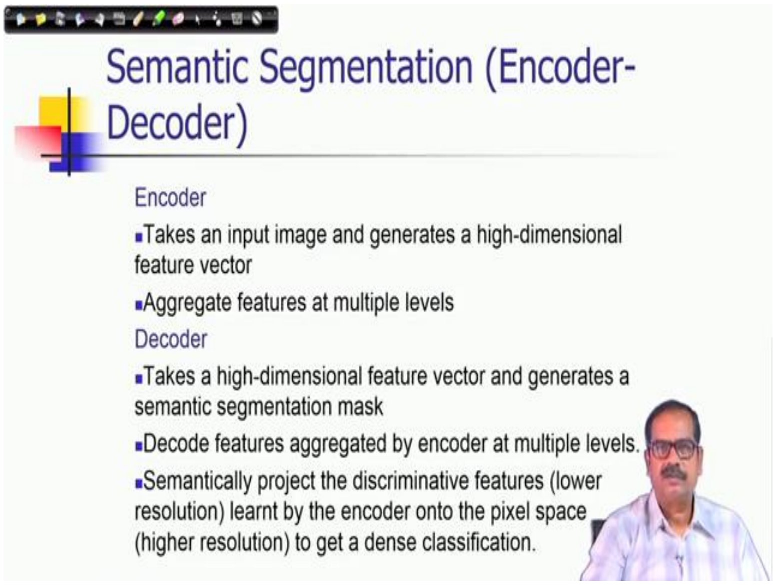
So, every pooling, the pooling indices are correspondingly passed to the corresponding decoding stage at the decoder stage. And they are in this case that is used for unpooling. So, finally, at the in the same way you get at the output at the final level which becomes the same size of the input. And you use the same 1×1 filters of that many number that many number of know levels that you would like to use for segmenting the image.

For example, in this case we could see there are around 1 2 3 4 5 6 say around 6 levels what are we shown here, but you know it may happen that there are many other levels, which are not reflected here because of the content of the image. So, we mentioned earlier that if there are k number of labels. So, at the output you should have $1 \times 1 \times$ filters of length the number of channels what you used in the output that that many number of filters should be there.

This is how from the decoder we generate the segmentation output. However, there could be various decoding strategies that, we would like to see. So, when we are performing semantic segmentation what we are doing . there are two stages; first is a learning stage or that is you are trying to represent it as a feature and learning the features. And the next you are doing the up convolutions and to project it to the semantic domain.

So, there is a stage as you can see the up sampling of learn low resolution semantic feature maps and that is done using up convolutions which are. So, during up convolutions you are using by linear interpolation filter weight initialization of weights and then pixel wise you are predicting. So, the whole thing is being trained in this way. So, your input output specification is there and you are training the network through downsampling and upsampling using this kind of structure.

(Refer Slide Time: 36:24)



Semantic Segmentation (Encoder-Decoder)

Encoder

- Takes an input image and generates a high-dimensional feature vector
- Aggregate features at multiple levels

Decoder

- Takes a high-dimensional feature vector and generates a semantic segmentation mask
- Decode features aggregated by encoder at multiple levels.
- Semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification.

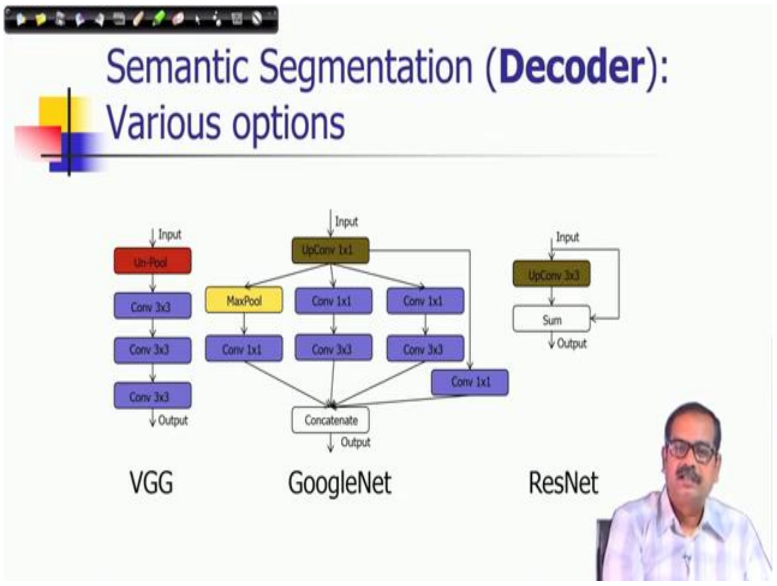
So, in the encoder, it takes an input image and generates a high dimensional feature vector. Encoder aggregates features at multiple levels and in the decoder it takes a high dimensional feature vector and generates a semantics segmentation mask. And decode features aggregated by encoder at multiple levels. Semantically it project the discriminative features and also that is learned by the encoder what I mentioned earlier to get a dense classification.

So, the basic thing here is that in this semantic segmentation, the encoder it takes the input image process in input image and it generates a high dimensional feature vectors. And it uses the features at multiple levels whereas, in decoder its input is that high dimensional feature, then it tries to project those features into the semantic domain. So, it tries to correlate those features with the labels and that is what it tries to get a dense classification out of this process.

So, this diagram shows that what could be the flow of computation. So all the convolutions layers and pooling layers are shown; so, in the left side you can see this is the encoder part and in the right side this is a decoder part. And you can see as I mentioned that pooling indices are to be provided for un-pooling operations in corresponding layer.

So, unpooling at this layer should be done using pooling indices of this one. So, that is why it is shown by these arrow. So, these are all pooling indices and this is a flow. So, it is actually you can see there is no fully connected layer in this particular architecture. So, we call it also fully convolutional neural network architecture.

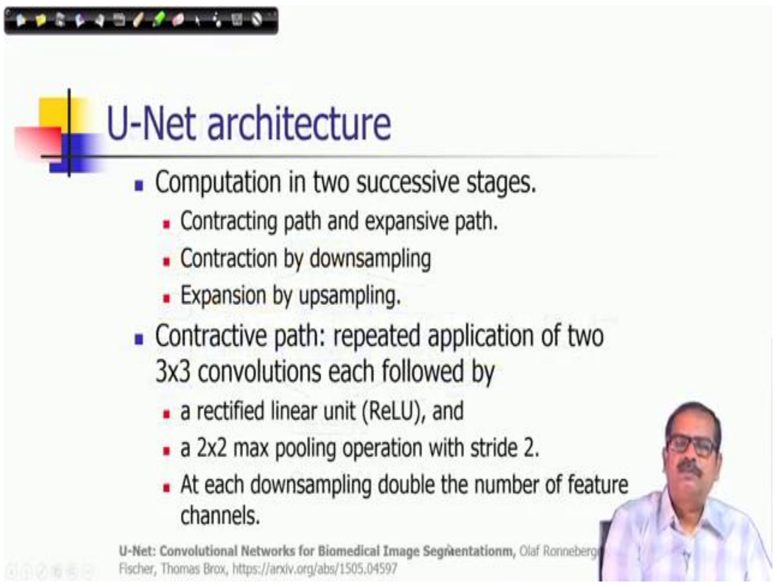
(Refer Slide Time: 38:23)


Semantic Segmentation (**Decoder**): Various options

For decoding there could be various options also, like you know you can use the vgg network which has been shown which uses a convolutional neural network. So use the unpooling operations and then perform convolutions stages of convolutions already you can use Google net where the inception models are used. Once again unpooling has to be there and then as you know that Google net it concatenates the output of different filters sizes but it keeps the output size from each filter the same by using 0 padding.

And you can use this kind of decoder. You can use also resonate a structure, where you know you will be only learning the residual errors. So, several such options are several such architectures are configurations are possible.

(Refer Slide Time: 39:25)



U-Net architecture

- Computation in two successive stages.
  - Contracting path and expansive path.
  - Contraction by downsampling
  - Expansion by upsampling.
- Contractive path: repeated application of two 3x3 convolutions each followed by
  - a rectified linear unit (ReLU), and
  - a 2x2 max pooling operation with stride 2.
  - At each downsampling double the number of feature channels.

U-Net: Convolutional Networks for Biomedical Image Segmentationm, Olaf Ronneberg Fischer, Thomas Brox, https://arxiv.org/abs/1505.04597

There is one particular architecture, which is a very popular in this with respect to this segmentation and that is called U-Net architecture and this is also used similar concepts. So, let me take a break here and then we will discuss particularly this architecture. And also we will show some of these results from this architecture in the next lecture.

Thank you very much for listening to this lecture.