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

Lecture – 26
Convolutional Neural Network Building Blocks

Hello and welcome to this like really interesting module. So, till now we have done down your introduction to visual computing and, then used very simple linear neurons and, then multi layer perceptrons and from there we went on to feature discovery kind of mechanism and the first versions of what is called as deep neural networks and, learning down from autoencoders to stack totter encoders and, then spots and denoising properties and using these kind of autoencoders to initialize a multi layer perceptron for your image to decision, or end to end pipeline for deep neural network based visual computing problem.

And most of these problems which we have a solving in terms of our theoretical lectures and, your lab exercises were all based on taking an images and trying to get to a decision, now keeping in mind that we were solving down practically things which were in the fully connected region, or multi layer perceptrons.

Today we are going to get you introduced into another newer kind of network which is called as a Convolutional Neural Network. So, this convolutional neural networks or CNNs are the ones which have really gained a big popularity into the community and, if you go down most people would actually be associating a deep neural network, or anything to do with a deep learning with the CNN. And if it is for images then people would just be doing it for CNNs, which is the very common acronym for convolutional neural network, but then what comes down is that if this term convolution comes to you, then typically you would be in terms of images you would be thinking of something like a sobel kernel private kernel and, then these or say even a wavelet over there. So, these are the ones which convolved around with the image and, then produce some sort of an output and then that is how this convolution is defined.

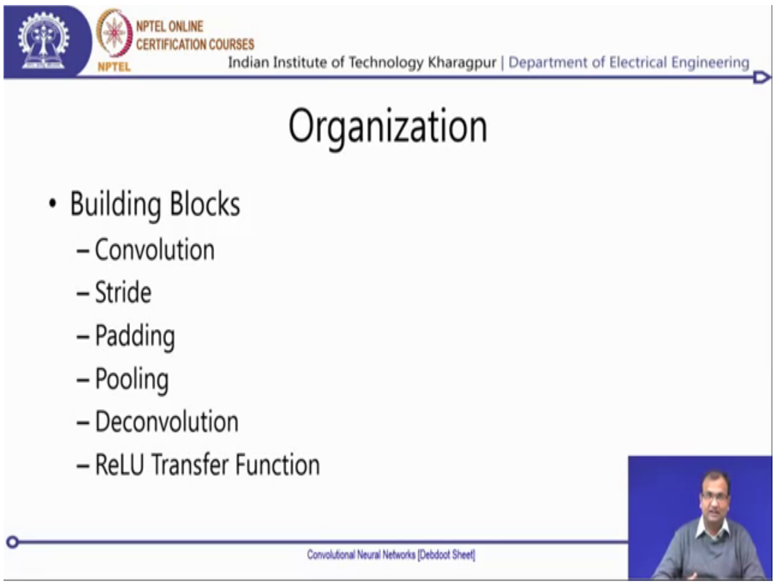
Now, it does raise a question in to our minds as to why do you think this is called as a convolutional neural network. So, do our neurons conform as well and yes that is that is answer. So, the they are like the CNNs or convolutional neural networks they are a big family of a networks which not necessarily just have only convolutions over that, but they are associated with some other common sort of functions and so, there is one function which is called as pooling, which is actually used to reduce down the size of the response which comes from your convolution and, that is quite a equivocal to on the classical computer vision side of it which is called as Laplacian pyramid decomposition.

So, if if you have your image and then you are always sub sampling it by a factor of 2 and then decomposing the size so, bringing it half of the size. So, you start or say technically one fourth of the size. So, you stand with a 200 cross 200 image in the next level of the pyramid you get 100 cross 100 in the next level of the pyramid, you get a 10 50 cross 50 and then in the next level at 25 cross 25. So, this is basically where your sub sampling is happening.

Now, there are multiple ways of doing this sub sampling one of them may be by trying to do interpolation and solve it out, the other may be by trying to do some sort of a pooling operator to get this one out. Now we would be making use and then studying all of them and in fact, there is also something called as the inverse of that and which is called this clever name fanciful name around called as deconvolution. So, we will be starting not with just one simple understanding of deconvolution though, there is and the future when we would be going down towards convolutional networks for semantic segmentation, we would be coming down into one very specific lecture which is devoted to understanding deconvolutions and the whole mathematical perspective from a signal processing point of view.

So, let us get started with this one, what I have with for CNNs today is just the basics of building blocks and I would be explaining you through what these building blocks are.

(Refer Slide Time: 04:12)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Convolutional Neural Networks [Debdoot Sheet]

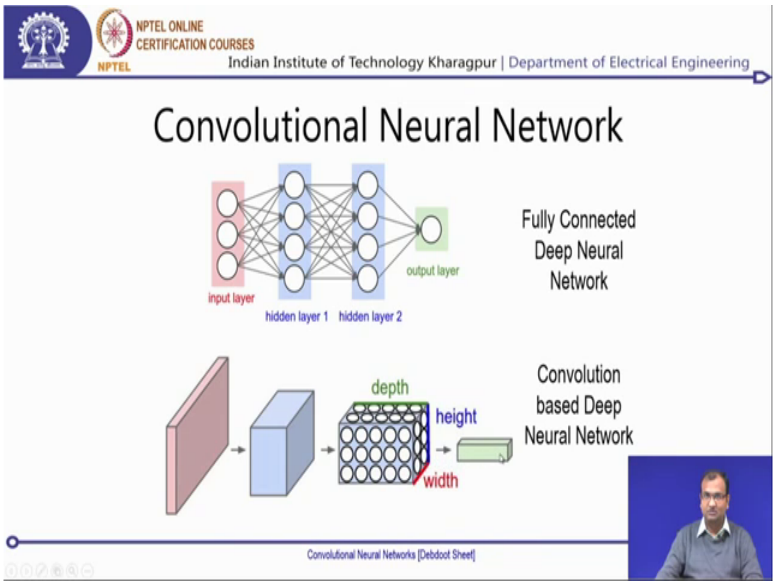
Organization

- Building Blocks
  - Convolution
  - Stride
  - Padding
  - Pooling
  - Deconvolution
  - ReLU Transfer Function

So, as the thing is organized it goes something like this, that within these building blocks we will be studying about convolution as an operator understanding what is called as a stride or how the movements happen in x and y direction. So, stride is basically if you are taking a stroll, then what is the length of your feet distance. So, after how many so, after whatever physical length is your feet falling down and, then that is what comes on in convolutions also as a stride because, you will be moving down kernel over an image as you are doing.

Then comes the concepts of padding pooling and deconvolution and ReLU as a transfer function. So, you remember that in one of our earlier lab classes, where we were drawing multi layer perceptrons, we had introduced this term called as a ReLU which I also said was a non-linearity in it is own sort, but more than these fully connected neural networks and multi layer perceptrons they are more implemented within CNNs as such.

(Refer Slide Time: 05:13)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolutional Neural Network

Fully Connected Deep Neural Network

Convolution based Deep Neural Network

Convolutional Neural Networks [Debdoot Sheet]

So, without reading much let us look into what a convolutional neural network would typically be doing. So, say that what happens within a fully connected neural network is something of this sort. So, you have your input layer over here with the number of neurons, which connect down to your inputs then each of these. So, these can be images as we had studied known with auto encoder initialize multi layer perceptron for classification of M nest for classification of fashion M nest kind of images. So, from there it goes down to hidden layer which has a lot many more number of neurons over there from there it can (Refer Time: 05:47) to another hidden layer and then finally, you have an output. So, this output can just be one class label for a classification problem and makes it much simpler.

Now, for a convolutional neural network this is how it get is more of represented in textbook parlance. So, what happens is you have an image and this arrow over here is some sort of a convolutional operator. So, these connections which you had in the earlier case for a fully connected network was from each neuron to every other neuron here, you do not have strictly that kind of a connection. So, these weights over there are some sort of shared weights.

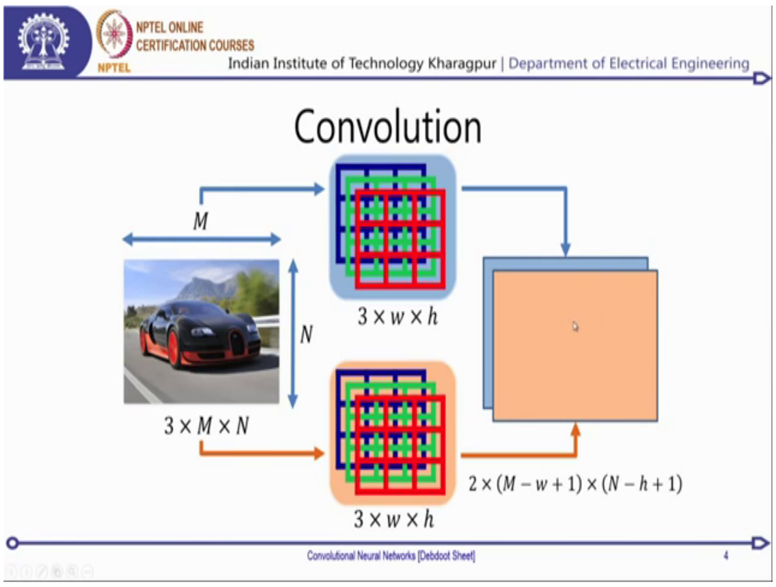
So, there will be a set of blocks over here, which will have certain weights which collect and feed it down to a one block over here, you can slide it down you get down the next block and that is how it keeps on going throughout. And then there is another concept of depth of here. So, initially if you are taking down just 1 grayscale image. So, that has a depth of 1, if you are taking down say a color image, then the number of channels over there is basically the depth present over there. So, for a color image this depth is going to be 3 ok.

Now, I can have any depth corresponding to any number of channels over there. So, what happens with the CNN is typically that we call something called as a number of channels, or a number of kernels which we are having and that will be defining the total depth over here and, typically what we do is we try to convolve and, then sub sample.

So, that the spatial resolution the x and y size over there keeps on decreasing. And after some time you would land down on our x y size equal to 1, which technically means that everything gets represented into one of these pixels, but then in order to preserve all of these information over there since you know that you cannot compress a whole image into 1 single pixel in any way and that that is a huge amount of loss which it incurs.

So, what we do in that case is that you keep on increasing along the depth and, then. So, now whatever was spread in space, now got spread across in this step and then by virtue of convolution which is a space invariant operator. So, if anywhere you place it down as the object is present it will give you the same kind of a response. So, that would help you in actually getting down your best responses coming down over here.

(Refer Slide Time: 07:56)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolution

Convolutional Neural Networks [Debdoot Sheet]

So, let us start with the convolution which is the first block and, then see what happens down. So, if you remember that say you have an image like this given down, which has a certain width and a height and this is a color image. So, the size of this image becomes 3 cross M cross N. So, number of columns over here or the width of the image is equal to M, the number of rows or the height of the image is equal to N. So, the total number of pixels present in this particular image is 3 cross N crossing ok.

Now with this say I have 1 particular filter over here, which is of this particular form. So, this is a say a 3 cross 3 filter over there. Now I can have another filter which is another 3 cross 3 and another 1 which is like this. So, typically say we take all these 3 together. So, this does for me a volume over there as well. So, you have studied convolution in terms of say for images or for signals and, there is where you had say for signals you just had a 1 d array where the signal was represented in case of an image you had a 2 d array where the signal was represented.

And then your kernel was also 2 d, but here the image which we have is a 3 d matrix it is no more just or 2 d, but over there. Now that really creates an issue. So, what we start by doing is we will be doing a simple spatial convolution, which is just move along this x and y direction, but since we have 3 channels which correspond to the color over here, we will be having a similar kind of a 3 d matrix over there.

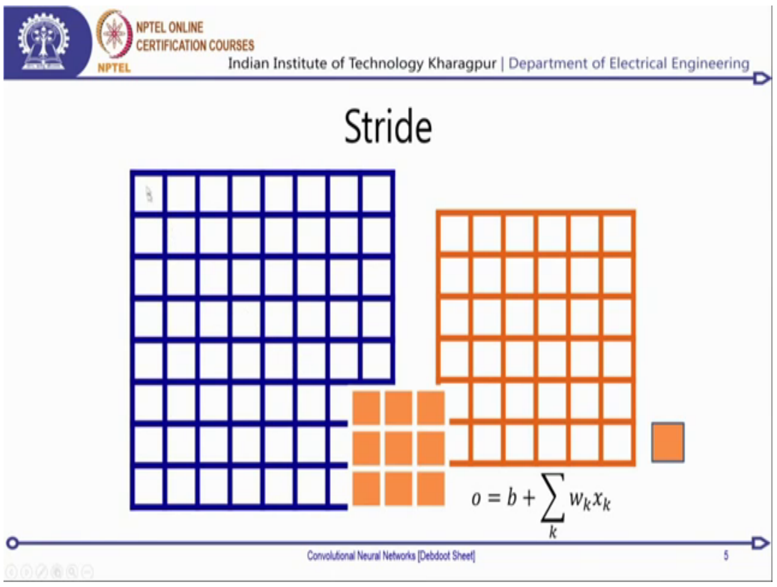
Now, keeping one thing in mind that the number of channels over here, is equal to the number of channels over here as well so, if this is 3 this has to be 3 and with that if I do a convolution with a kernel of say 3 cross w cross h, then I am going to get done one output over here. Now similarly if I take another kernel and then convolve with this 1, I get down another output over here ok. Now the size of this output so, you see that I had 2 such kernels with which I was convolved. So, the size of my output will necessarily have 2 output channels and that is why this 2 comes in.

Now, comes the width and height concept over there, now if you recall from your understanding of convolutions and, you know there is something which is called as the full width convolution. So, what happens in case of a full width convolution is that say you take this whole block over here, which has a width of w and a height of h ok. So, if I am starting from this particular location over here. So, I cannot start by so, my most valid will be when this particular pixel location over here, or the top left corner location actually matches with my top left corner of the image.

And similarly the last point when I can go is the right bottom corner matches down with, my right bottom corner right bottom corner of the kernel matches down with, right bottom corner of the image. And then the total number of convolutions which I can take or the number of pixels which I can get as an output of this convolution by striding it along will basically be M minus w plus 1, where M is the width of the image, w the width of the kernel and so, that is how it is related. So, this is an empirical understanding which we all have from our very basic classes of convolutions.

Now, what comes down is that since this is this has 3 number of channels and., that is it is moving down only in the x y direction. So, each of them is going to map down and come down over here, and since I have 2 different convolution kernels. So, I will be getting down 2 different outputs and that is what I put down on the channel over here and that gives rise to 2 different channels for me. So, I guess this is a very intuitive and simple explanation.

(Refer Slide Time: 11:49)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Convolutional Neural Networks [Debdoot Sheet]
5

Stride

$$o = b + \sum_k w_k x_k$$

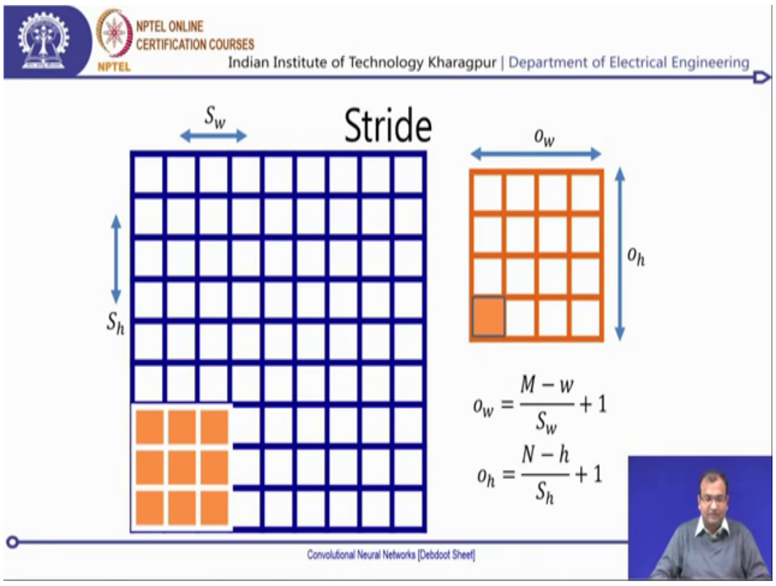
Now from there, we will enter into much more complicated thing which is called as straight ok.

Now, for stride what happens is something of this sort. So, initially for my convolution what I do is I have this 3 cross 3 I place it on my image and, then so, the point is that I just do. So, if w k is the weight of this particular location in the k-th location on the kernel and x k is the value of the pixel at the k-th location. So, so somewhere over here or just the pixel underneath that 1 is what is called as x k, then my output over there which is what comes down of the convolution over here, is defined something of this form. So, it is a just our dot product of w k x k and summation with the bias and, this bias is going to be independent of where I am sliding this convolutional window over there.

Now, this is for my first. Now what I can do is, I can move it by 1 pixel I get down the next location, I move it by another pixel, I get done by next location and then and so, on and so forth I keep on going and doing this one. Now as I keep on repeating and come down to the next location. So, this sliding movement over there keeps on continuing and I can fill up my whole space ok. And accordingly I would be getting down my output. Now over here if you look into it so, I had a image which was of the size of 1 2 3 4 5 6 7 8 and on this side also, I have 1 2 3 4 5 6 7 8.

So, there are 8 number of elements present over here, now over here what do I get 1 2 3 4 5 6 ok. So, if you clearly load note done from our earlier explanation so, that was supposed to be 8 minus 3 plus 1. So, 8 minus 3 is 5 plus 1 is 6. So, I get down my 6 over here and similarly I get down 6. So, my convolution of a 8 cross 8 matrix with a thick crusty kernel is definitely going to give me 6 for 6 kernel, but then the point is to understand down that here, we were moving down by 1 pixel at a time in order to do that and that is what is called as a movement or of convolution with the stride of 1.

(Refer Slide Time: 14:10)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Stride

$$o_w = \frac{M-w}{S_w} + 1$$

$$o_h = \frac{N-h}{S_h} + 1$$

Convolutional Neural Networks [Debdoot Sheet]

Now, if I would like to change that and go and make it something called as a stride of 2 then what happens. So, I start with my first location and do it, but now when I move to the next location instead of moving by 1 pixel I consider moving down by 2 pixels and, that is going to give me my next location. So, there is a bit of offset over here, this should actually be coming down at this location ok.

So, and similarly I keep on and then this amount of movement which happened over there is what is called as my stride S w. So, if I keep on going down to the next location and next location and accordingly I would be getting not so, I keep on repeating this over all of them and, then I am going to get down my output. So, the output is called calculated in the same way, but what comes interesting over here is that if my stride in the along the x direction, or along the width is called as S w and my stride along the height, or along the vertical direction is called as S h and my input image has the size of M and my kernel has the size of w which is the width over there, then my output over there is what is related according to this one. So, that becomes M minus w by S w plus 1.

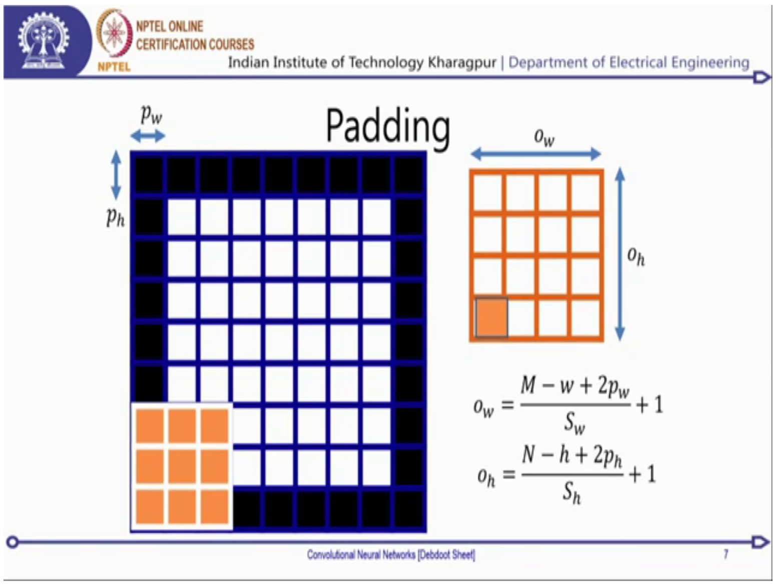
So, let us let us just do an empirical verification. So, I have 1 2 3 4 5 6 7 8 9 ok. So, I have a 9 cross 9 image 1 ah. So, 1 2 3 4 5 6 7 8 9 right now that I have a 9 cross 9 image and I have 3 cross 3 kernel. So, what comes down over here is 9 minus 3 and I took a 9 minus 3 becomes 6 and I take a stride of 2. So, 6 divided by 2 is 3 and a plus 1. So, that makes it 4. So, technically if I have this sort of a relationship over here that I take image which is of size M and a kernel which has a width of w, and then stride with S w then my output is guided down by this relationship.

So, this is the width of the output similarly if I have want to look into the height of the output, then I will have to look into the horizontal side as well and, the for no reason you have some mandate that your S w and S h need to be equal or your M and N need to be equal in any way, they can be in equal and in all sort of ways and in fact, your kernels also not necessarily need to be square shaped. So, your w and h can also be different from each other and that would make that this will be valid for a some sort of a rectangular kernel as well and, that is that is perfectly valid. So, that there are no issues with that coming up in any way.

Now, this was about getting you introduced on to if even if we are doing convolution, you do not need to always necessarily go by a stride of 1, or one pixel at a time and then keep on doing this slider motion, you can actually do hops and jumps and by doing hops and jumps you are actually going to reduce down this number comes over here. So, if I was doing with a S w of 1 and a S h of one this would have made 9 minus 3 which is 6 plus 1 which is 7.

So, my output with the stride of 1 would have given me a 7 0 7 matrix instead of a 4 plus 4 matrix. So, that 7 cross 7 matrix is actually a much larger sized and that is not a sub sized image. So, if you look into it that I had a 9 rows 9 image and, then the response is coming down as a 4 cross 4. So, this necessarily means I did some sort of a convolution and, then I also reduced the size of the image. So, this is going in the same way in your spatial decomposition of images, it is its it is practically the same way in which we are running down over here as well.

(Refer Slide Time: 17:50)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
p_w
p_h
o_w
o_h
Convolutional Neural Networks [Debdoot Sheet]
7

Padding

$$o_w = \frac{M - w + 2p_w}{S_w} + 1$$

$$o_h = \frac{N - h + 2p_h}{S_h} + 1$$

The next option is what is called as padding. So, till now what we had looked at that you can only operate from the valid ranges and that necessitates that you are not going to have an equal sized image at any point of time coming out because, you will never be able to place down your kernel over here. So, even if you do a stride of 1 and you have the smaller size of the kernel your output over there is going to be reduced in some way, except for that if you have a kernel of 1 cross 1, but then that would just be meaning that you are doing a pixel y is operations across all the channels you are not considering neighboring pixels in any way.

So, for that we have one clever option which is called as padding, now padding can be of different types and the most common way is what is called as 0 padding. So, the idea is that if I have a kernel to be placed down over it here, and to do it then these areas. So, 1 2 3 4 5 these 5 pixels which basically do not get any constituent coming down what will happen down to them. So, it is quite clear over here that if we try to pan down with a length of p w and p h, then that should be able to solve some sort of a problem. So, it means that I just put down extra number of 0s around over there ok. So, I can put p w number of 0s along the rows. So, I can create that any number of columns and ph number of rows can be padded as 0s and that will be the number of rows added down along each of these columns ok.

Now, with that this is what comes down my output and, then I can keep on doing my stride rate movements and, then accordingly I get my outputs coming down and, then I keep on repeating this across and my final output is formed. Now over here if my width of the output is called as o w and the rest of whatever we were taking that remains constant, then the output is related something of this what that o w is equal to M minus w plus 2 p divided by S w plus 1.

Now if you clearly recall that in our earlier cases it was M minus w by S w plus 1, now the only thing which we changed is we added a few 0s and how we were doing is that whatever is the p w value that much gets added on the left as well as on the right on both the sides of the image that would mean that my width of the image, now changes effectively from M to M plus 2 w. So, M plus 2 w becomes my width of the image and w becomes my width of the kernel as it has remained always.

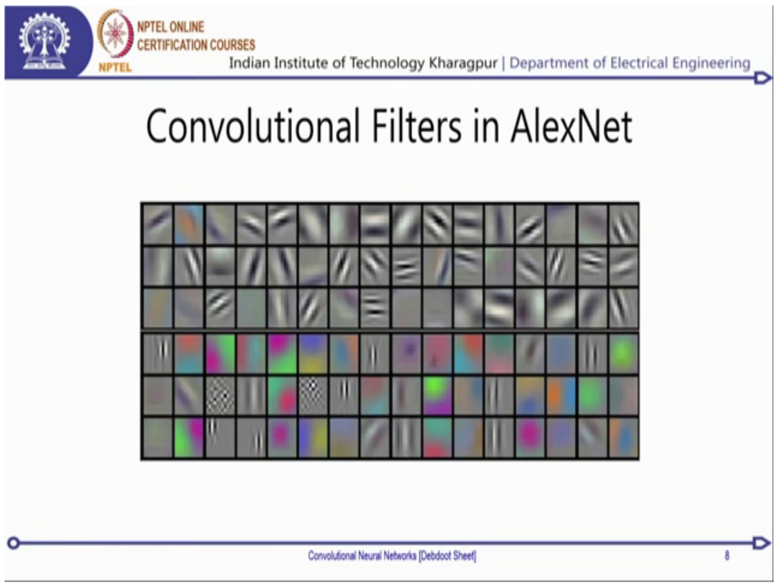
So, my equations over there so from my earlier slide on doing a studied convolution where I had M minus w by S w; so, my M changes to M plus 2 w and that is the only change which comes down over here and, now you can clearly see that I can get down a full length convolution taking down contributions even along these edges and I get an output. Now I can I can pose this interesting question to you and take a few minutes to look at it. So, under certain condition of combinations of say my only variable over here is w S w and p w. So, under certain combinations of say w S w and p w can you get o w is equal to M. So, let us let us do a simple exercise.

So, let us say that my w is 3 and I put down my p w as 1 and S w as 1. So, what comes out over here let us look into it. So, this becomes M minus 3 plus 2 ok. So, and then here on the output side of it this also becomes 1 ok. So, it becomes M minus 3 plus 2. So, that becomes an M minus 1 divided by 1 is M minus 1 plus 1 gives me and M. So, if I have a stride of 1 my width of my kernel S 3 and a padding of 1, I would actually end up getting an output which has the same x and y size as that of my input over here and that has a huge implication.

So, now, what you can do is you can have convolutional filters say noise reduction filters, or even h detection filters which you can directly design, you do not need to subsample I mean your whole image whatever is the size of the input, the same is what comes down a size of the output over there this has huge potential in something which is called a semantic segmentation, or pixel by pixel simple segmentation method which is used in much more advanced techniques. So, we have semantic segmentation lectures where well be exploring this kind of a network over there, which are called as fully convolutional and same sized network. So, it is more of like a piping network which keeps on going down.

So, similarly with your height you have the same sort of convergence and response which will be seeing down as well.

(Refer Slide Time: 22:54)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolutional Filters in AlexNet

Convolutional Neural Networks [Debdoot Sheet] 8

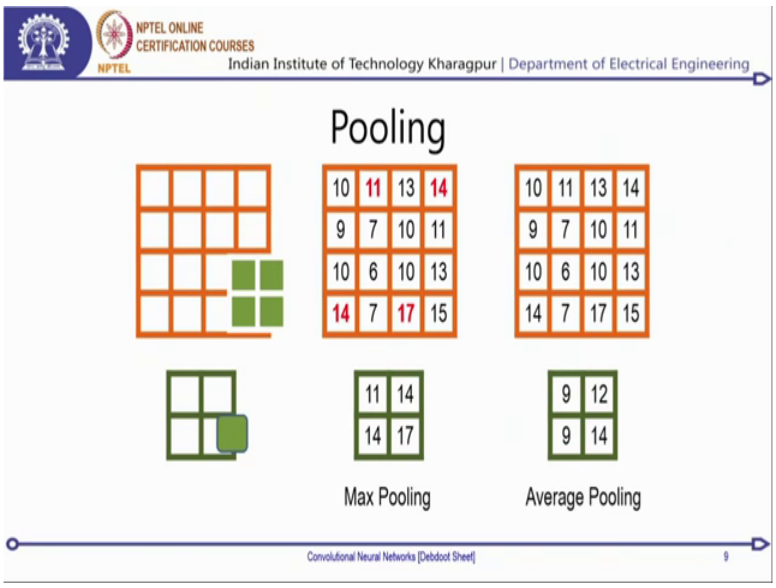
Now, given all of those so one of the famous filters is what is called as AlexNet, or this is Alexander Crips Keyes network for image net challenge prediction and, then we are trying to visualize down the kernels. So, you remember that we were looking into kernels of autoencoders, or what is the weight? So, these weight matrices if we just arrange them in some sort of a rectangular fashion and try to look into them, then how do they look like.

So, similarly if we look into these kernels over there for my convolutions, then what do this convolution kernels look like and this is what they end up appearing. So, you have very intuitive tools of visualizing them and as we do in the lab classes we will be coming down across and, then providing you with those insights into how to visualize them as well. So, you can pretty much visualize and see.

Now, you see some of them are in colors and that is for the reason that if your input is 3 channel inputs, then your weights will also be. So, your kernels if you remember from the first slides, they were also 3 channel kernels over there. Now if you visualize a 3 channel kernel over there you will get down your RGB all of them coming down together and that is what renders these has some colors.

So, these replicate sort of Gabor filters and in different angular orientations a few over here, as well and then these ones are the ones which have some sort of a color be variants filter coming down including these ones and these ones. So, these are the different filters which it gets learnt on and subsequently down the line with more advanced lectures coming down, we would also be explaining in details into what these filters are and how what they signify in terms of what has been learnt.

(Refer Slide Time: 24:32)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Pooling
10 11 13 14
9 7 10 11
10 6 10 13
14 7 17 15
10 11 13 14
9 7 10 11
10 6 10 13
14 7 17 15
11 14
14 17
9 12
9 14
Max Pooling
Average Pooling
Convolutional Neural Networks [Debdoot Sheet]
9

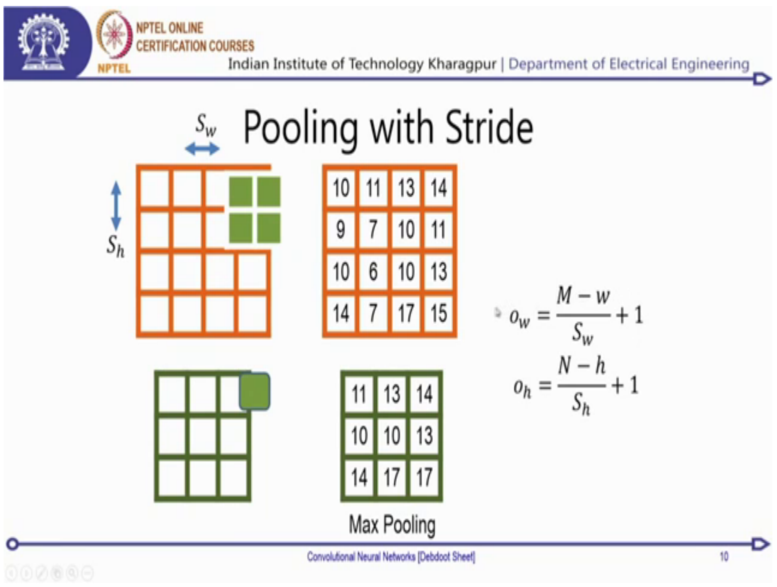
So, this was about convolutions the next part is what is called as pooling and pooling is basically a way of reducing dying down the size of an image. So, for pulling the idea is that say whatever is the response of your convolution in one particular convolution with one particular kernel, in terms of an output in a channel and let us look into that particular form. So, say I have these responses given down over here, now my concept is that I would look into a small neighborhood over here and, then try to cool it completely in terms of 1 single value. So, this is about aggregating all of them into 1 single value. So, this aggregation can be in different ways. So, you can take an average of all of these

4 values and replace it over here, you can take the maximum out of all of these 4 values you place it over here, you can take the minimum out of these 4 values and replace it over here, you can take a median out of these 4 values and replace it over here as well.

So, now in the same way as we were doing down with convolutions these 2 can be strided and accordingly you will get down your output. Now let us look into a very simple form of representation of doing this say, this is the response from one of my outputs over there and, if I have a pooling which is implemented as a max pooling, which means that out of this window over here whatever is the maximum value that is the 1 I will be doing.

So, my max pooling operator is going to return me a value of this sort, if I have something called as an average pooling on this, then average pooling is going to give me a value which looks something of this 1. So, these are 2 different ways of reducing and what I essentially did is that I had a forecast 4 sized output and, I got it into a 2 plus 2 which is reduced the size of the output by 1 4th of a factor and these can be used in what I told is constructing your Laplacian pyramid kind of an architecture for multi resolution decomposition akin to what we had in our classical ways of computer vision.

(Refer Slide Time: 26:28)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Pooling with Stride

10	11	13	14
9	7	10	11
10	6	10	13
14	7	17	15

11	13	14
10	10	13
14	17	17

Max Pooling

$$o_w = \frac{M-w}{S_w} + 1$$

$$o_h = \frac{N-h}{S_h} + 1$$

Convolutional Neural Networks [Debdoot Sheet] 10

Now, pooling can also be implemented with the stride because what we were doing over there was we essentially we were striding by a factor of 2. So, you pooled a 2 cross 2 point, then you went down to the next 2 cross 2 location, but you do not have any commonality or sharing of locations between these 2 poolings which we are doing, but I can choose to do some sort of a stride of one and pull down as well. And that is going to give me interesting locations coming down. So, your pooling as well does have something called as a stride and, now say that I have a stride of 1 as I was doing over here and try to pull then, let us look into what the output would be. So, I I give you say a few couple of minutes to just try to solve down the first 2 points over there. So, say this location and this location and, then quickly tell me what happens down with the max pooling.

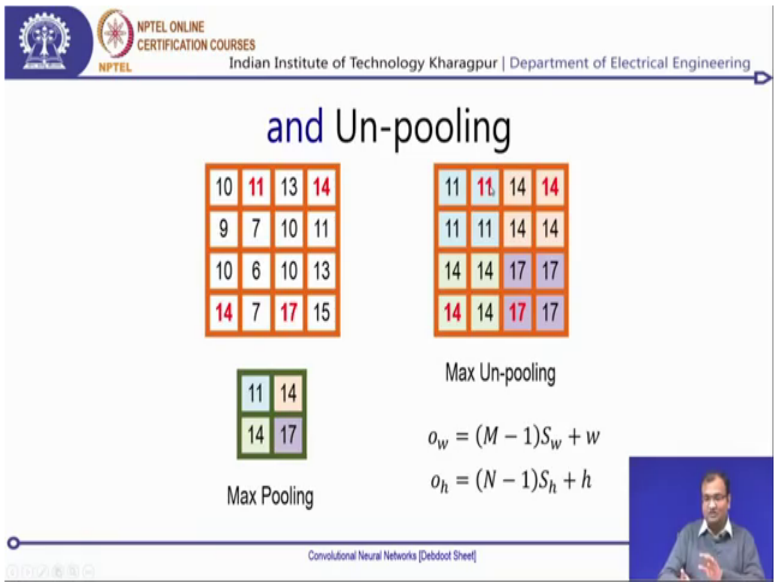
You see in this location it is 10 11 7 9. So, your output is going to be 11 which is the maximum value, in this location it becomes 11 13 10 7 and this is the maximum value eleven which comes down, then the next stride is over here and the maximum value is 14 and that is exactly what comes down over here. Now do you recall some sort of a similarity with something you had studied in image processing let us see.

So, if you recall down binary morphologies, then from binary morphologies you did go on to something called as a grayscale morphological processing. So, in your mathematical morphology call operators on grayscale, you had this dilation operation which was equal to taking a maximum value out of that region and that is exactly what your max pooling is as well doing. So, so is not that quite interesting how all of these learning mechanism, if you are using over here, they do have a 1 to 1 correlation to what we have been using as very interesting fundamental building blocks within our classical ways of visual computing problems and, that brings us to the interesting realization over here that this is not something new which just suddenly came up.

The only point is we are putting down all concepts which we knew about in classical vision into 1 single pipeline which can itself adapt and learn to use them and when to use them. So, that is where comes down the interest about this like real, such in use of deep learning for visual computing purposes. So, looking into the output side over there so, this also has a similar kind of a form. So, if your width of the pooling operator over that is w and your stride is S w, then you can relate it pretty much to your output in the same way as you were doing it with a convolution, except for the fact that in your convolution you had weights of the kernel to be learnt.

So, there was the tunable parameter and they that contributes to the parameters of your model, in case of pooling you do not have any learnable parameter as such and this does not contribute any way to the parameters update, this is just a feed forward pass which happens, it will just be doing down a fixed kind of an operation which you are given it is it is not going to learn anything from the data. So, during back propagation there are no updates, which happens down to stride in anyway which happens to the pooling in any way.

(Refer Slide Time: 29:36)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

and Un-pooling

10	11	13	14
9	7	10	11
10	6	10	13
14	7	17	15

11	14
14	17

Max Pooling

11	11	14	14
11	11	14	14
14	14	17	17
14	14	17	17

Max Un-pooling

$$o_w = (M-1)S_w + w$$

$$o_h = (N-1)S_h + h$$

Convolutional Neural Networks [Debdoot Sheet]

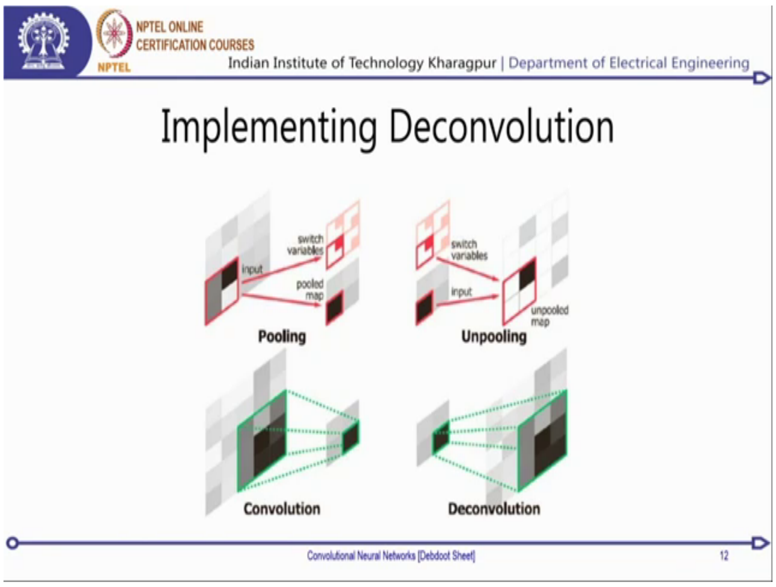
Now, once you have done pooling you need to understand that there should be an opposite side of it which is say I have a smaller size image and, I want to create to a bigger size image. So, I should be having something which is opposite of that or undoing it and, that is what is called as the un pooling. So, within un pooling the concept is something of this sort that if this is my input which has been pooled and using a max pooling as this my output.

Now when I try to do a max un pooling what that would do is that it would go and substitute these nearest neighbors into all of these blocks. So, this 1 which has to be done un pool version of 2 cross 2 then this is going to substitute over here, this is going to substitute over here and, here also in max un pooling you can do pretty much just writing as well. So, if your here we have strided down by 2, if you are doing it with the stride of

1, then what will happen is that you will have this average out movement and then you sum them up.

So, over here you will have a contribution from the first element and the second element. So, since they will be shared elements. So, over there you are going to take an average over all the values which comes down and have a much smoother transition. So, with this kind of a stride you see that there are some sort of a pixelated kind of an effect. So, all values are 14 14 all values are seventeen all values are 11, it is quite different because here it was much of a smoother region which comes down, but nonetheless I mean this is a computationally much more faster way to solve it out you can; obviously, use any kind of an interpolation as well to replicate a un pooling behavior also.

(Refer Slide Time: 31:05)



NPTEL ONLINE CERTIFICATION COURSES

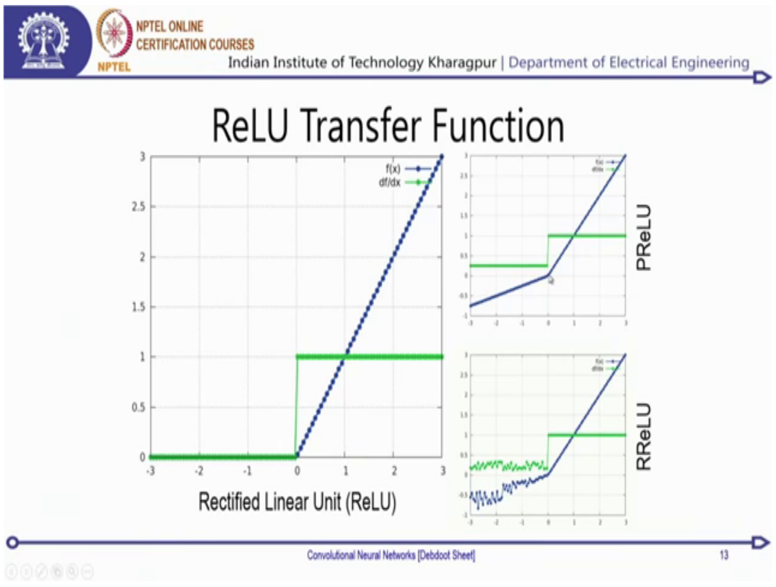
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Implementing Deconvolution

Convolutional Neural Networks [Debdoot Sheet] 12

Now, that replication of a un pooling behavior in terms of your interpolations is what is also available into something called as a deconvolution block. So, a deconvolution block is something which is supposed to be opposite of a convolution block and does it, but then it does have a much more intriguing theory because, there are certain number of convolutions which operate into a sub pixel shift region and, the sub pixel shift is what introduces this kind of a continuity coming. So, later on we have one lecture which is dedicated to deconvolution very specifically, where we will be reading about more of details into deconvolution.

(Refer Slide Time: 31:45)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

ReLU Transfer Function

Convolutional Neural Networks [Debdoot Sheet] 13

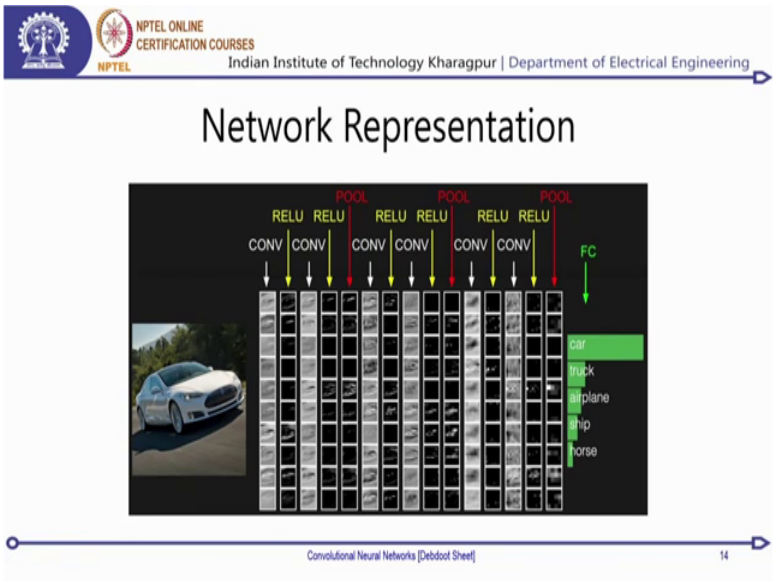
Now, these were about resizing of your operators, the next part comes down your rectified linear unit which we had referred down in an earlier class, but nevertheless it is always good to do. So, here since we are dealing more of with images which have a range of 0 to 1, or values greater than 0 always so, we will try to put down all responses in the same range and that brings us, to have a linear function rather than sigmoid kind of non-linearity.

So, here this kind of a function which is called as a rectified linear unit necessarily forces all values which are negative to go down as 0 and all values which are positive to go down as 1 and you would see that although there is a discontinuity at this point, but it it is still differentiable and the derivative exists of this sort. So, if it is 0 or lesser than 0 then the derivative is 0 if it is greater than the 1 that the derivative is unity which makes even calculations much more easier because, now your derivative of the non-linearity, which is one of the things to be computed, you know that for all values of for all positive values this derivative itself is 1.

So, the derivative is unit that there is not much of an issue to solve 1 over there, if the input is 0 the derivative is 0 if the input is 1 the derivative is 1. So, the computationally it makes it much more tractable when you are trying to do a back propagation via your derivative. So, this is an advantage of having and is a very commonly used transfer function within your convolutional neural networks.

Another variant is what is called as the probabilistic ReLU and the randomized ReLU. So, these will have some sort of a slope in your negative part as well, but the derivative again still stays down almost very close to 0 or 0 here is where you put down some sort of a randomization. So, that it does not have a local lock in and, then accordingly the derivative also varies slightly.

(Refer Slide Time: 33:33)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

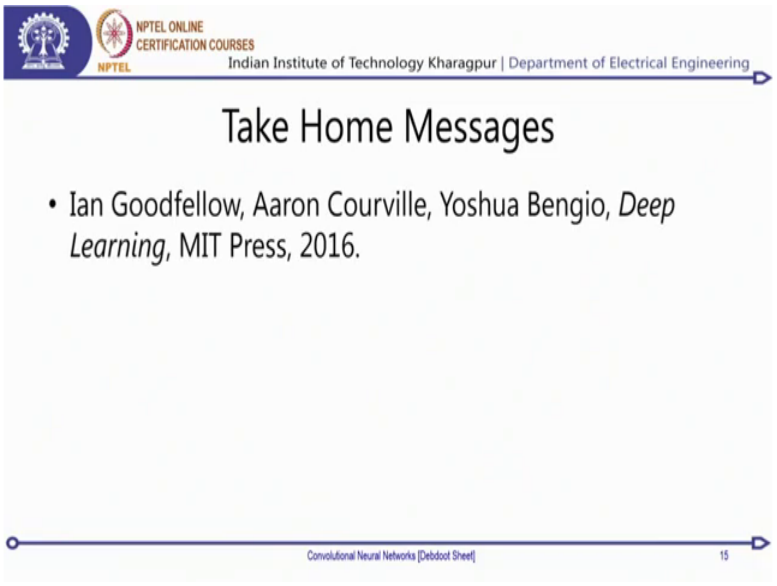
Network Representation

So, from there let us look into if we try to unroll a complete network and what happens this is what is called as an AlexNet and where it does. So, we will be in the next week we will be studying more detail about this network how this network looks like, but this was just about you understanding how these kernels and what are the features which learn stuff.

So, the response so, these are basically not the kernels, but the response which comes out of convolution of this image with the kernels in the first layer and, then these are the responses which come down. Once you do a ReLU this is what comes out. So, all the negative values get down suppressed to 0 and only positive values are forward passed, you do another convolution and they said what comes out and then eventually like as you keep on going, you would see that it loses down on the visual abstracts and starts becoming more of pixelated and based on like which neuron comes down over here after pooling and everything is what is going to give down the output from the fully connected layer.

So, the classification network well be doing in the next lecture where we learn about clay net which is one of the very basic ones for the handwritten digits classification and we will get gradually into how this information flow along the depth keeps on happening.

(Refer Slide Time: 34:41)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Take Home Messages

- Ian Goodfellow, Aaron Courville, Yoshua Bengio, *Deep Learning*, MIT Press, 2016.

Convolutional Neural Networks [Debdoot Sheet] 15

So, with that I come to an end and for more details you are definitely suggested to go through this book on deep learning by Goodfellow, Courville and Benjio which is as such one of the primary tech primary texts which we are falling down for this lecture series as well. So, with that we come to a closure of this lecture and thanks and stay tuned for the subsequent ones.