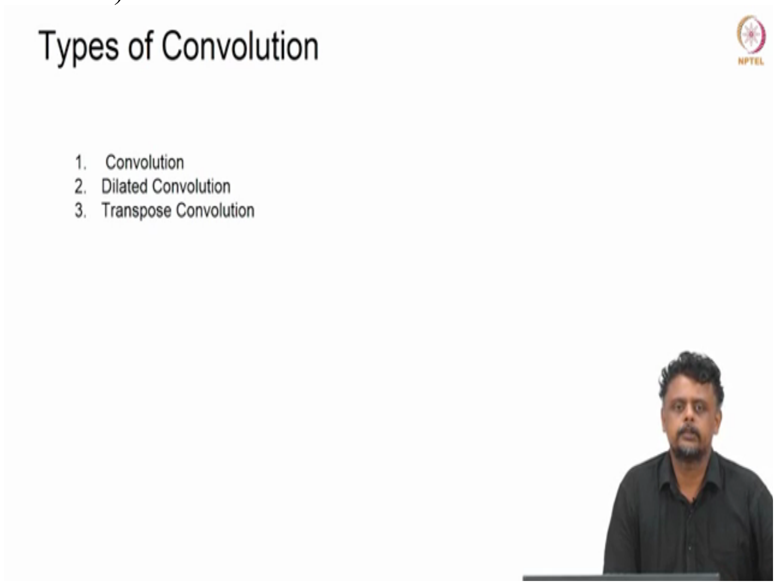
Machine Learning for Engineering and Science Applications

Professor Dr. Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology, Madras

Lecture 50

Types of Convolution

(Refer Slide Time: 00:13)

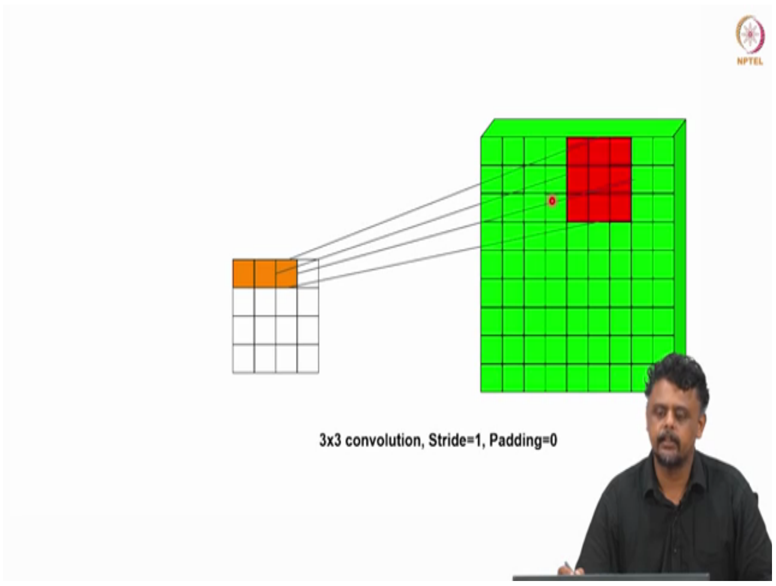


Types of Convolution

1. Convolution
2. Dilated Convolution
3. Transpose Convolution

In this video we will look at the different types of convolution that are typically done in a convolutional neural network and especially focus on the dilated and transpose convolutions which are often used in deep networks and also in networks that have the encoder decoder type of architecture.

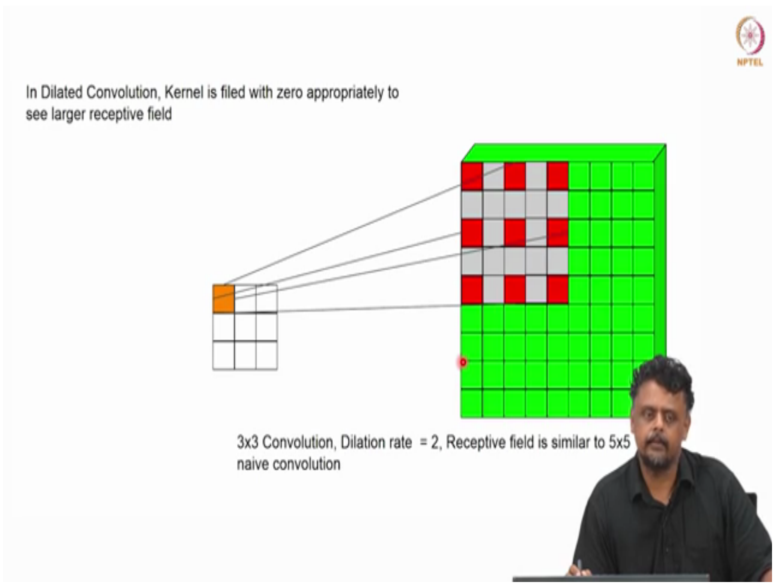
(Refer Slide Time: 00:32)



So just to recap is what we call naive convolutions, so this is the typical convolution operation that is done on a deep neural network. So we have a given kernel size, you have a given kernel size K. stride use in the convolution and the padding and given as input feature map of size I w and I s I h width and height, the output feature maps width and height are calculated by this formula, so it is the size of the feature map minus the size of the kernel plus 2 times the padding divided by the stride plus one that sees width, so the typical same formula applies to the height also.

So this is the operation that we typically see this is involves superimposing the filtered kernel which shown in red here over the green input volume and then striding it across the input volume. So just to illustrate so if you go back that is the first element calculated here by superimposing the 3 by 3 volume and then once again we do the same by moving the filter kernel by one side of in this case the side is actually 2 and though it is written as 1 here, it is actually 2 slide of 2 and we repeat this calculation throughout the cross section of the feature map.

(Refer Slide Time: 01:51)


In Dilated Convolution, Kernel is filed with zero appropriately to see larger receptive field

3x3 Convolution, Dilation rate = 2, Receptive field is similar to 5x5 naive convolution

Now dilated convolutions again or another that convolution that are used to increase the receptive field of the convolution given the same figure given a smaller filter size, so it is a cheap way of keep in the computational sense way of getting a larger receptive field using a smaller feature kernel filter kernel. So for instance in this image we have this input feature map in green and the filter kernel is of size 5 by 5 and as we saw earlier we just move the filter kernel across the feature maps in order for us to obtain the output feature map shown here, ok.

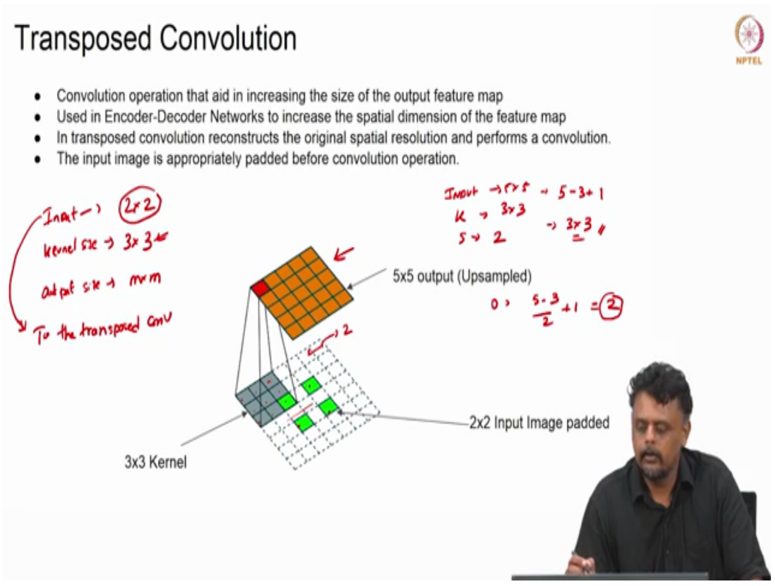
Now the idea is to use a 3 by 3 convolution with the dilation factor that can visualize the same area as a 5 by 5 convolution while the number of parameters remain the same that is basically you have 9 parameters instead of 25 parameters used in a 5 cross 5 convolutional kernel however by having a dilation in there we can you we can get the same receptive field size.

So how do we accomplish that? So what is shown here in this image is that there is this these red elements correspond to the elements of a 3 cross 3 feature map and by you know incorporating appropriately rows and columns of zeros, we are making the size of the feature of the filter kernel 5 cross 5 and then proceed to do convolution as before, say so this is very advantageous in the sense that you will add with little computational expense you are able to get the same receptive field size.

If we recall as we seen earlier in order to get a receptive field size of 7 cross 7, it is the same as during 3, 3 cross 3 convolutions in succession so that will be that is an increased number of operations while in this case we get directly get a 5 cross 5 receptive field just by adding a appropriately adding zero zeros and zero columns in the rows and columns of your filter kernel.

The number of rows and number of zeros zero rows and zero columns that you add as they first referred to as the dilation factor ok. So the just to recap the idea is to inflate the size of your kernel by inserting rows and columns from zeros, so that you can get a slightly larger receptive field.

(Refer Slide Time: 04:15)



Transposed Convolution

- Convolution operation that aid in increasing the size of the output feature map
- Used in Encoder-Decoder Networks to increase the spatial dimension of the feature map
- In transposed convolution reconstructs the original spatial resolution and performs a convolution.
- The input image is appropriately padded before convolution operation.

The next topic is the transposed convolution, here the idea behind transpose convolution is to aid in increasing the size of the output feature map and these are typically used in encoder decoder networks especially on the decoder size so as to increase the size of the feature map as you go towards the output ok. The idea is to regain the original spatial resolution not to regain the actual feature map itself but just to have a just to regain the original resolution that is the size of the feature map and you have to appropriately pad with zeros and also insert zeros and rows zeros in rows and columns of your input feature map in order to obtain the appropriate sized output, ok.

The good way of thinking about it is to see that it is how do the idea behind the transpose convolution is we interpret the input feature map let us say the input feature map is of size 2 cross 2 ok and you also been given a kernel size ok let us say this is a 3 cross 3 ok, now we want to get a certain sized output ok it is given let us say this is some m cross m we do not care ok, so what we have to do is we have to understand this just as that you assume that the input that you are given to the transpose convolution this is the input to the transpose convolution is an output of as a direct conversion ok.

So we have to find out what feature map size when operated upon with the kernel size of 3 cross 3 ok given the padding and slide or given we will give this output size ok and dilated conclusions we try to regain that particular we try to get the that particular input resolution which is basically trying to reverse that operation, here once again to reiterate we do not seek to get the input and in fact reconstruct the input feature up but just to regain the resolution.

The example given here, so you would you want to reconstruct this size right or you want to reconstruct this resolution 5 cross 5 is your target output and your input is actually a 2 cross 2 feature map given by the screen squares, so it is actually a 2 cross 2 input your filter kernel is of size 3 cross 3 given by this grey ok and you seek to obtain a 5 cross 5 output ok, the stride parameters are given so if you think about it the idea is what filters what input feature size when convolved with a feature map with the filter kernel of size 3 by 3 would give rise to a 2 cross 2 (out) output right, so that is what we would we would have to figure out.

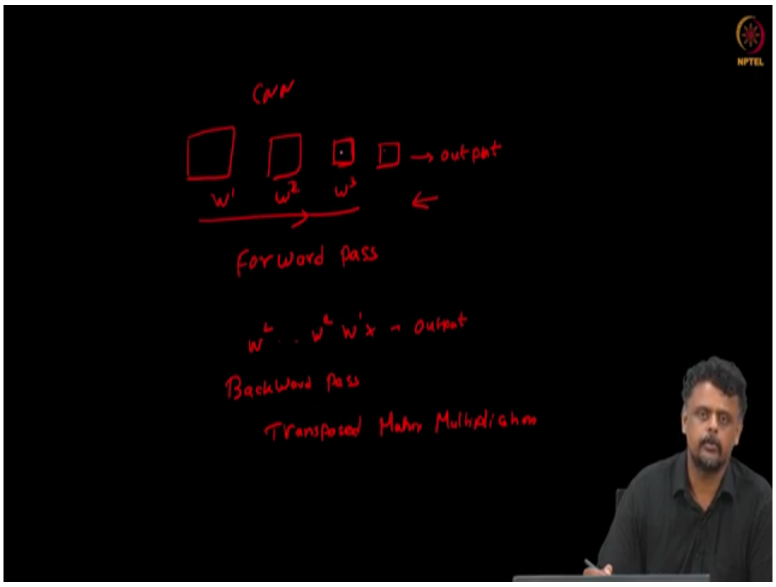
So they turns out that this it is 5 cross 5 and the idea is we in order to regain this 5 cross 5 size we have to actually have these zeros zero padding on the outside so zero padding of 2, and we also need to have this zero columns and zero rows inserted into the input feature map of size 2 cross 2 and then proceed with the convolutions as before ok. So for instance let us just understanding this better let us say your input size is 5 cross 5 ok and your kernel size is 3 cross 3 if you do a naive convolution then you would get the output size would be 5 minus 3 plus 1 which is 3 cross 3 ok, so which is not what we want.

So if you want to get a 2 cross 2 output what do we have to do? So let us say we have a stride of we add a stride of 2 in which case your output size would be 2, 2 cross 2 ok, so if we have a stride of 2 no zero padding and we have an input feature map of size 3 5 cross 5 and you and your filter kernel is of size 3 cross 3 then you decide of 2 your output feature map would be of size 2 cross 2, ok.

So now we want to reverse this operation, so since we have a side of 2 we insert 1 or s minus 1, we insert 1 row of zeroes and one column of zeros into the feature map of size 2 cross 2 right and there was no zero padding here so which means that we act so the everything is reversed so that to understand this so since there was no zero of padding here we have to do a zero padding when we do the transpose convolution if you think of it that way.

So we have a padding of 2 right, so in order to get the appropriate size which mapped on your stride this one as before so and you stride is again 1, so if you do the convolution as like a naive convolution or your usual convolution with this particular input size then you will end up with you 5 cross 5 feature map ok, so this transpose convolution is typically is the one that is often used in encoder decoder networks or in any situation where you have to up sample your feature maps.

(Refer Slide Time: 11:03)



CNN
output
w^1
w^2
w^3
Forward pass
$w^L \cdots w^2 w^1 x$ - output
Backward pass
Transposed Matrix Multiplication

Another way of looking at it is that let us say you have a network right and bunch of feature maps, this is CNN let us say typically where a bunch of feature maps ok, this is your forward pass through the network ok, so you get an your output right so we know that to get from the input output it is basically a sequence of matrix multiplication, so if the weights in every layer is this w2 w3 so on just to be slack with the notation here so the output will be a sequence of multi matrix multiplication right w 1 X and then you would have so on and so forth right, this case there are L layers then you have W L X right, so this is your output so during forward pass.

Now during the backward pass or back prop your gradients are propagated by transposed matrix multiplication right, so error so the error you by you back propagate in this direction so as you notice if your back (propatin) propagating the error from the smaller size feature map to a larger size feature map which means here you are actually doing an operation which is same as the transposed convolution ok, that is the way of understanding transpose convolutions.