Deep Learning

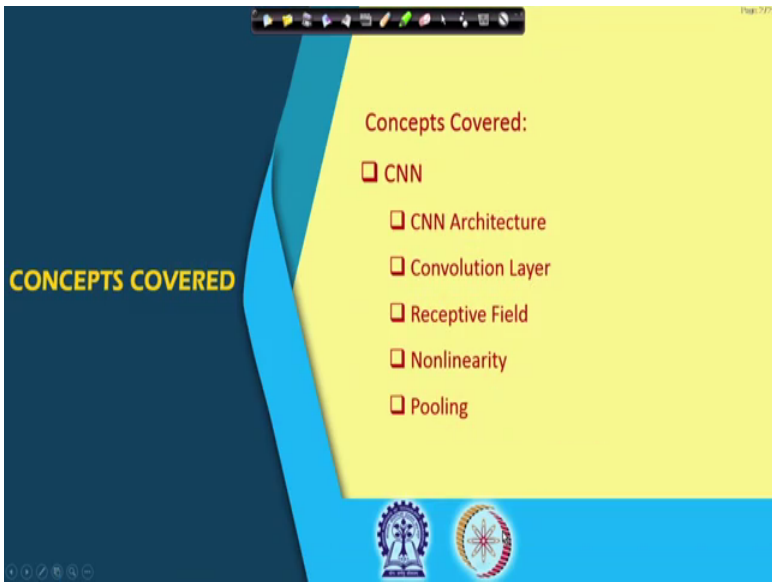
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
CNN Architecture

Hello, welcome to the NPTEL online certification course on Deep Learning. In our previous two lectures, we have discussed about the two operations, one of them was convolution operation, the other one was a closely related operation which was correlation or cross correlation. So, we have said that when you have been given a linear time invariant system or a linear space invariant system which is characterized by its impulse response, when given any input signal to such a system the response of the system or output of the system is actually convolution of the input signal with the impulse response of that LTI or LSI system.

Against that closely related operation which is cross correlation is something like if you are given two signals say one of them is f t, the other one is g t, you find out the cross correlation between f t and g t and this cross correlation value will be very high if the signals are very close to each other. That means, to find out the similarity or dissimilarity between two given signals you employ the correlation or cross correlation whereas, to find out the response of a system to a given input you use the convolution of the input signal with the impulse response of the LTI or LSI system.

(Refer Slide Time: 02:11)



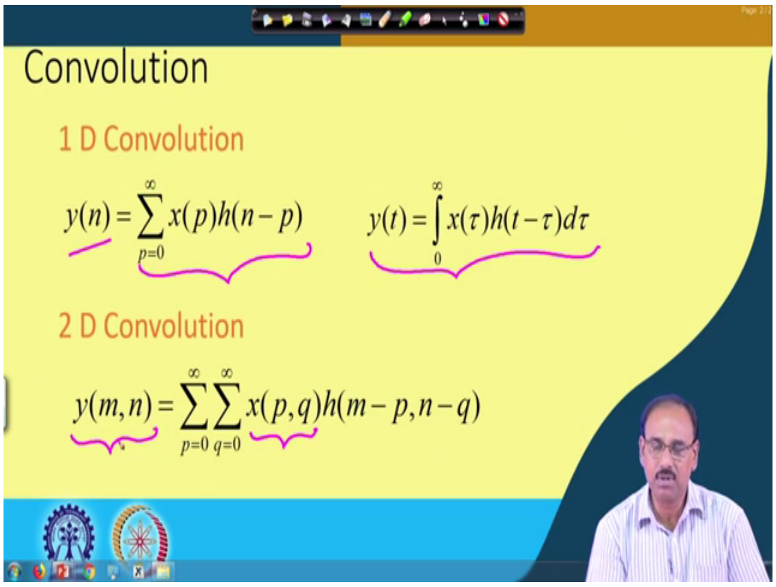
CONCEPTS COVERED

Concepts Covered:

- ❑ CNN
  - ❑ CNN Architecture
  - ❑ Convolution Layer
  - ❑ Receptive Field
  - ❑ Nonlinearity
  - ❑ Pooling

So, given this now we will talk about in today's lecture the architecture of convolutional neural networks. So, while doing this we will talk about what is a convolution layer. We will also give an idea of what is known as receptive field, then the non-linearity which is used in convolutional neural network and we also talk about the other operation which is used in convolutional neural network which is the pooling operation. So, let us see each of this one by one.

(Refer Slide Time: 02:46)



Convolution

1 D Convolution

$$\underline{y(n)} = \underbrace{\sum_{p=0}^{\infty} x(p)h(n-p)}$$ $$y(t) = \underbrace{\int_{0}^{\infty} x(\tau)h(t-\tau)d\tau}$$

2 D Convolution

$$\underbrace{y(m,n)} = \sum_{p=0}^{\infty}\sum_{q=0}^{\infty} \underbrace{x(p,q)}h(m-p,n-q)$$

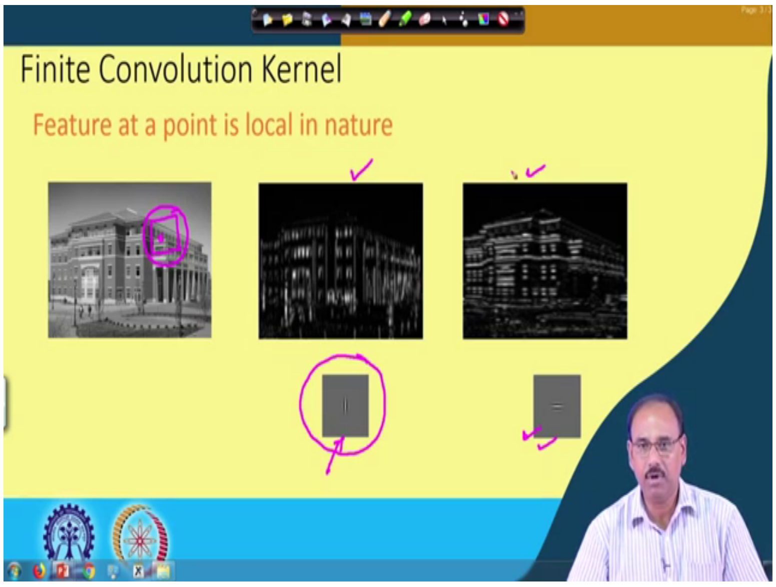
So, as we have seen before that in case of one-dimensional convolution, if you are given an input signal x in the form of a sequence of samples at regular intervals of time and if h is the impulse response of the input signal, in that case the convolution output or the output of the system for the given input say x is given by y, where y n is given by this particular expression which is the convolution of the input signal x with the impulse response of the system.

In the same manner, in the continuous domain we can represent the same convolution operation as this expression where y t is equal to x tau, h t minus tau d tau you take the integral over tau varying from 0 to infinity. So, this is what gives you the output signal at every time instant t and similarly, in the discrete case the nth sample of the response or nth sample of the output signal is given by x p times h n minus p take the summation from n equal to 0 to infinity. So, this is what you get for one-dimensional signal.

But in our case which we will talk about, yes, mostly on image classification techniques or computer vision techniques the signal is a two-dimensional signal or images. So, in that case, given a linear type space invariant system characterized by the impulse response say h m n, the given an input signal x the output signal or the response of this linear space invariant system is given by y, where every sample or every pixel in the output image y m n is given by the convolution equation which is x p q, h m minus p, n minus q, take the summation over p is equal to 0 to infinity and q is equal to 0 to infinity. So, this is how you get the output of the system when the input is an image x, output is the image y, where the system impulse response is given by h.

So, given this now let us try to see that how we can use these expressions or this convolution operation in our convolutional neural network architecture. So, now, onwards whenever I talk about convolution or the convolution operation, the convolution is purely characterized by the impulse response h and the operation as we said that both in case of cross correlation or correlation and in case of convolution they are similar, ok.

(Refer Slide Time: 06:11)



Finite Convolution Kernel

Feature at a point is local in nature

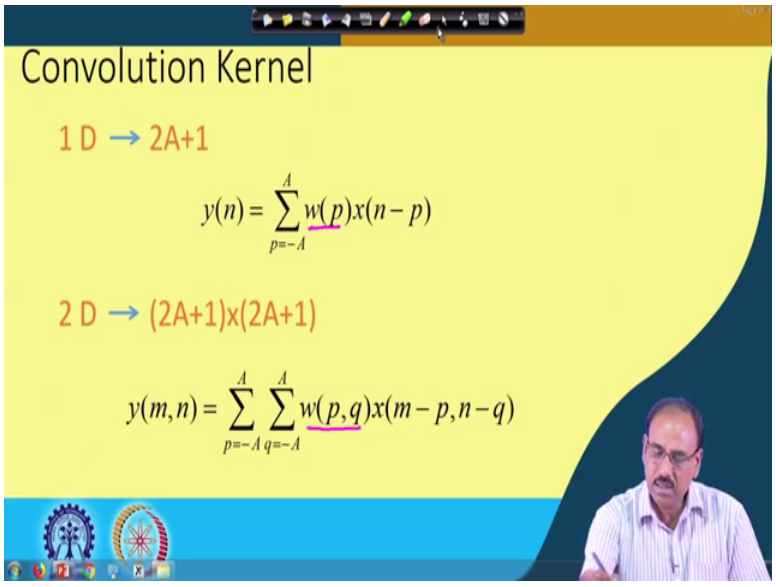
So, given this, you find that the purpose of this convolution operation is to extract certain features within the given image. Say for example, I am interested to find out whether an edge, whether it is an horizontal edge or vertical edge or a diagonal edge occurs at a particular location in the image or not. Say for example, I am interested to find out whether I have a horizontal edge at this particular point. So, in order to find out whether you have an horizontal edge at this particular point, it is not necessary that you have to look over the entire image. Rather, if I take a small neighborhood around this point and process the image within this small neighborhood then I can find out whether there is an edge whether it is horizontal or vertical present in this location or not.

So, as a result the convolution kernel that you have to use that convolution kernel has to look into only this particular neighborhood, it does not have to look over the entire image. So, given this, it is possible that in the previous slide what had shown is the summation from 0 to infinity, it is not necessary that I have to have convolution kernel which is a infinitely extended, rather I can have a finite convolution kernel, so as given over here.

So, if I have this finite convolution kernel, this kernel when slighted over the entire image and at every position on the image you do the convolution operation, this gives you a response to that particular region of the image to this particular system given by this impulse response.

And the result you see that if you convolve this image with this convolution kernel this is what is the output response that you get, that we said earlier as a feature map. Similarly, if you convolved the same image with another finite impulse response as over here you get this as the output, which highlights all the horizontal edges which are present in the image. So, this clearly shows that when I go for convolution operation the convolution kernel need not be an infinite lisp and kernel, but I can have a finite convolution kernel. So, using this finite convolution kernel now I can write my convolution expression in this form.

(Refer Slide Time: 08:56)



Convolution Kernel

1 D → 2A+1

$$y(n) = \sum_{p=-A}^{A} \underline{w(p)}x(n-p)$$

2 D → (2A+1)x(2A+1)

$$y(m,n) = \sum_{p=-A}^{A} \sum_{q=-A}^{A} \underline{w(p,q)}x(m-p,n-q)$$

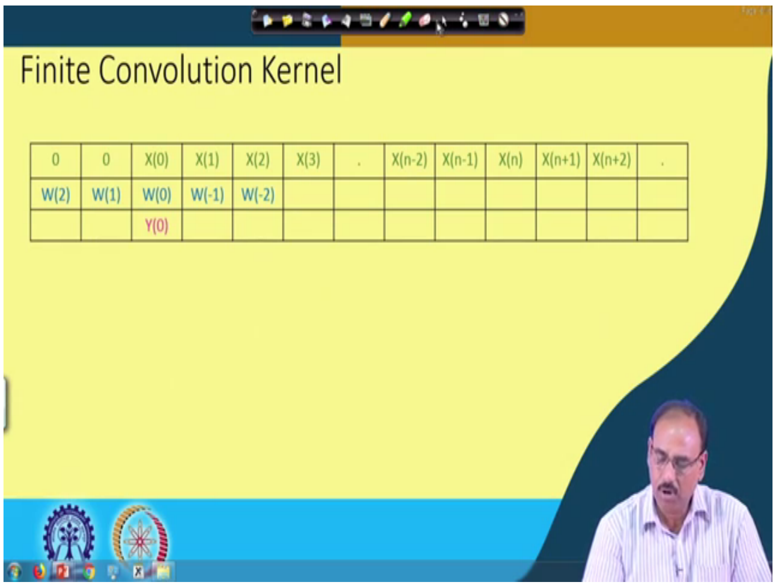
So, in one-dimensional case I assume that my span of the convolution kernel is given by 2A plus 1, where A is some integer maybe 1, 2, 3 and so on. So, value of A defines that what is the span or size of the convolution kernel. Similarly, in case of two-dimensional signal a convolution kernel can be of size say 2A plus 1 by 2A plus 1, again for different values of A. So, the values of A will determine what is the size of the convolution kernel and given this finite convolution kernel, now I can rewrite the same convolution equations that we had written earlier in a slightly different form, and you will find that both these forms are actually identical.

So, now I will write in one-dimensional case the outputs sample y n is equal to w p x n minus p, where now p varies from minus A to A. So, you find that if value of A is equal to 1, then p will have values minus 1, 0 and 1 which says that the convolution kernel is of size 3. Similarly, in two-dimensional case I will write the same convolution equation in the form say y m n, where y m n is the image sample at location m n in the output image.

So, here the convolution operation simply becomes y m n is equal to w p q x m minus p, n minus q where both p and q varies from minus 1 to minus A to plus A. So, if value of A is equal to 1, so this means both p and q will vary from minus 1 to 1, 0 inclusive that means, my convolution kernels will be of size 3 by 3 and you notice one more thing that in the earlier case the impulse response we had represented by h, but now I am writing the same impulse response as w. The reason being it is the same impulse response coefficients or the kernel coefficients will be used as weights when in a node in the convolution layer you accumulate the weighted sum of the inputs from the previous layer.

So, these coefficients in the convolution kernel are nothing, but the weights. So, instead of writing this as h, I am writing this as w indicating that the same kernel coefficients will be used as weights in input weights of a node in the convolution layer.

(Refer Slide Time: 11:59)



Finite Convolution Kernel

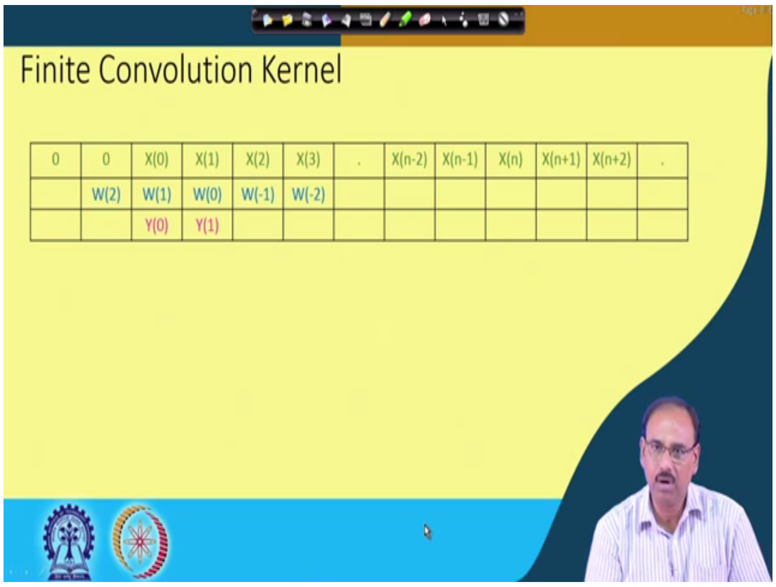
0	0	X(0)	X(1)	X(2)	X(3)	.	X(n-2)	X(n-1)	X(n)	X(n+1)	X(n+2)	.
W(2)	W(1)	W(0)	W(-1)	W(-2)								
		Y(0)										

So, given this now, I have the basics of a convolution network. So, just for an illustration let us see how this convolution will work in a one-dimension. So, I am using a convolution kernel of say W minus 1, W minus 2, W 0, W 1 and W 2. So, this is a kernel one-dimensional convolution kernel of size 3.

So, the first operation that we have seen earlier is we have to flip the kernel, then shift it to the location where I want to find out the output response. My input signal sequence is X 0, X 1, X 2, x 3 and so on. So, when I want to find out Y 0 the output response at time instant 0, then this flipped convolution kernel is centered at on the sample X 0 and then you go for point by point multiplication or sample by sample multiplication of the sample values with the corresponding kernel coefficients and add all of them together. So, this addition of this summation of the weighted input samples gives you the output sample value which is Y 0 at time instant 0.

And here you notice one more thing that we have added some additional elements in the input sequence which we have kept as values 0 and this is what is known as padding. So, you go for padding with extra elements with value 0 in order to make sure that your number of samples output samples will be same as the number of input samples. We will also see in case of two-dimension that we also use padding for two-dimensional cases in case of images.

(Refer Slide Time: 13:56)

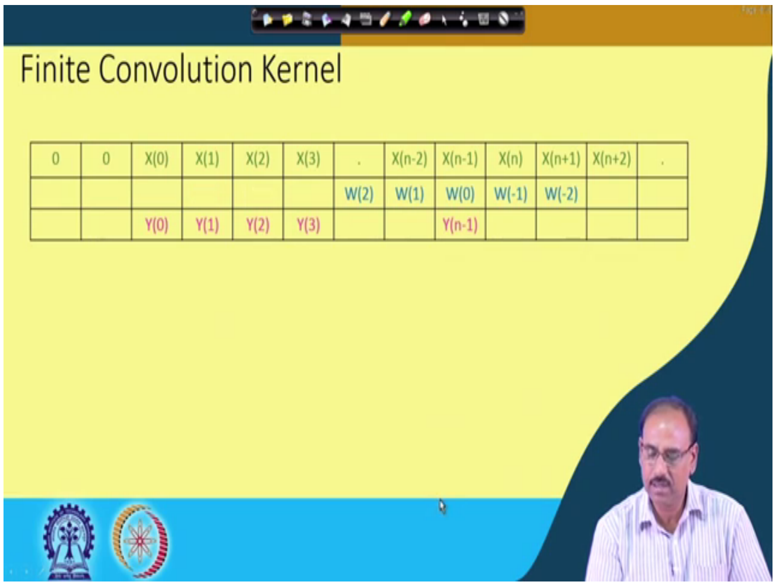


Finite Convolution Kernel

0	0	X(0)	X(1)	X(2)	X(3)	.	X(n-2)	X(n-1)	X(n)	X(n+1)	X(n+2)	.
	W(2)	W(1)	W(0)	W(-1)	W(-2)							
		Y(0)	Y(1)									

So, given this, so this is what we get for n equal to 0 at Y 0. Similarly, two for Y 1 you shift the kernel by one location perform the same operation.

(Refer Slide Time: 14:02)

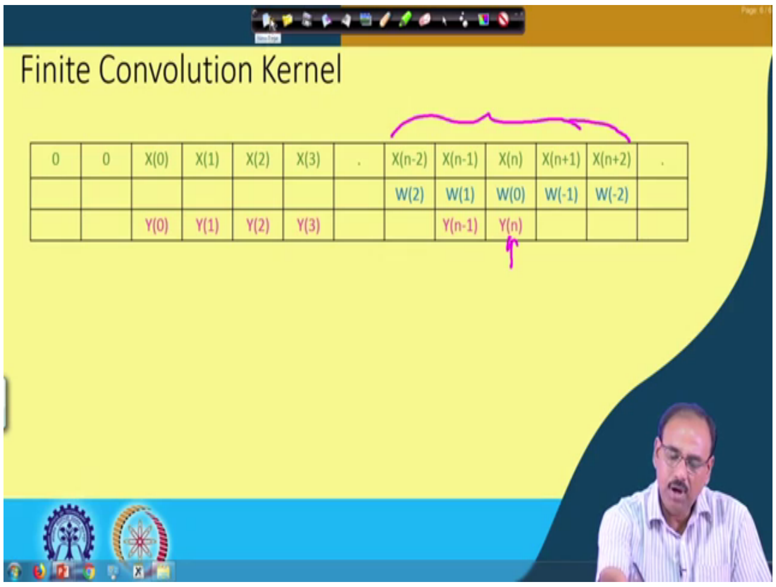


Finite Convolution Kernel

0	0	X(0)	X(1)	X(2)	X(3)	.	X(n-2)	X(n-1)	X(n)	X(n+1)	X(n+2)	.
						W(2)	W(1)	W(0)	W(-1)	W(-2)		
		Y(0)	Y(1)	Y(2)	Y(3)			Y(n-1)				

So, you get Y 1, you get Y 2, you get y 3, you get Y n minus 1. So, for to get Y n minus 1, again you find that your kernel is centered on sample X n minus 1.

(Refer Slide Time: 14:18)



Finite Convolution Kernel

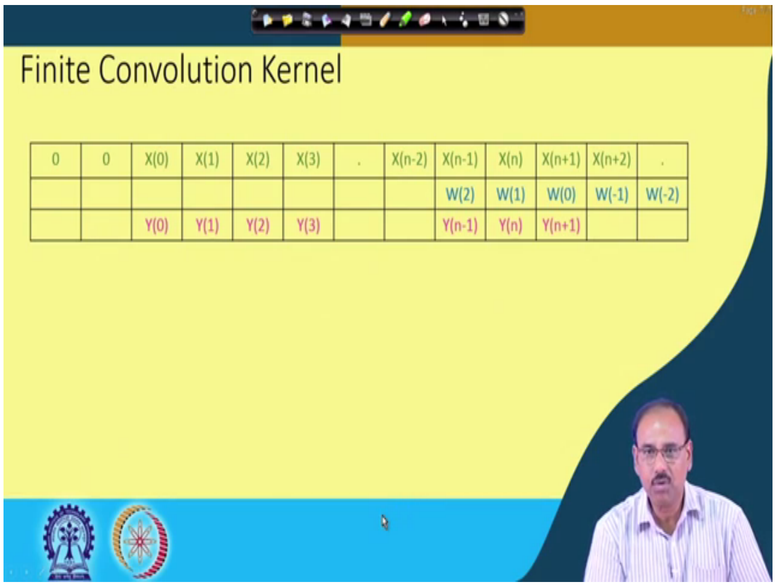
0	0	X(0)	X(1)	X(2)	X(3)	.	X(n-2)	X(n-1)	X(n)	X(n+1)	X(n+2)	.
							W(2)	W(1)	W(0)	W(-1)	W(-2)	
		Y(0)	Y(1)	Y(2)	Y(3)			Y(n-1)	Y(n)			

Similarly, Y n the kernel is centered on sample X n and that gives you output Y n. So, here you notice one more thing that when a convolution operator is convolving over the input sequence it is actually looking at a smaller number of sample values. It is not looking at the entire number of and number of samples and using the smaller number of sample values it is computing what is the output response.

So, in this case when I want to find out what is this X n we are actually looking at the sample values from X n minus 1, n minus 2 to X n plus 2 as the size of my convolution kernel is 5 and this is what is known as your receptive field that is the range of the input samples over which the convolution takes place. This is what is known as the receptive field.

We will also see that in case of images this receptive field will actually be a rectangular region. So, we can continue in the same manner to find out Y n plus 1, Y n plus 2 and so on.

(Refer Slide Time: 15:34)

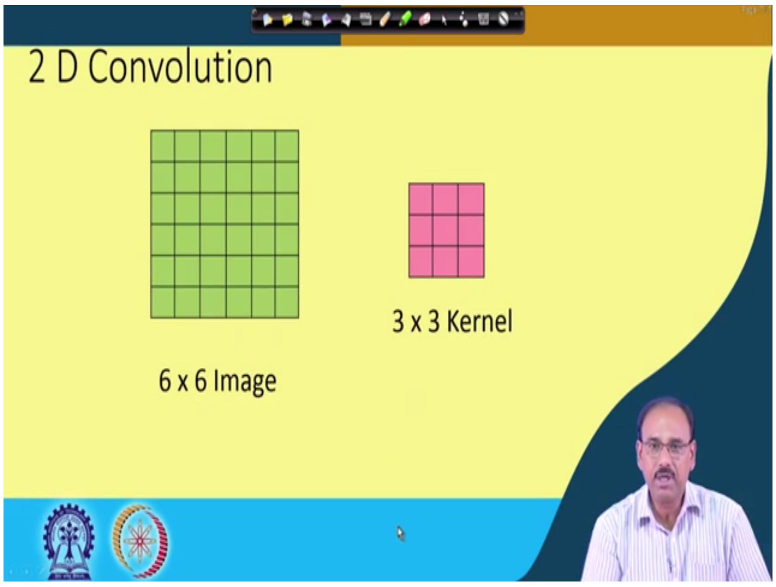


Finite Convolution Kernel

0	0	X(0)	X(1)	X(2)	X(3)	.	X(n-2)	X(n-1)	X(n)	X(n+1)	X(n+2)	.
								W(2)	W(1)	W(0)	W(-1)	W(-2)
		Y(0)	Y(1)	Y(2)	Y(3)			Y(n-1)	Y(n)	Y(n+1)		

So, this is your illustration for one-dimensional case.

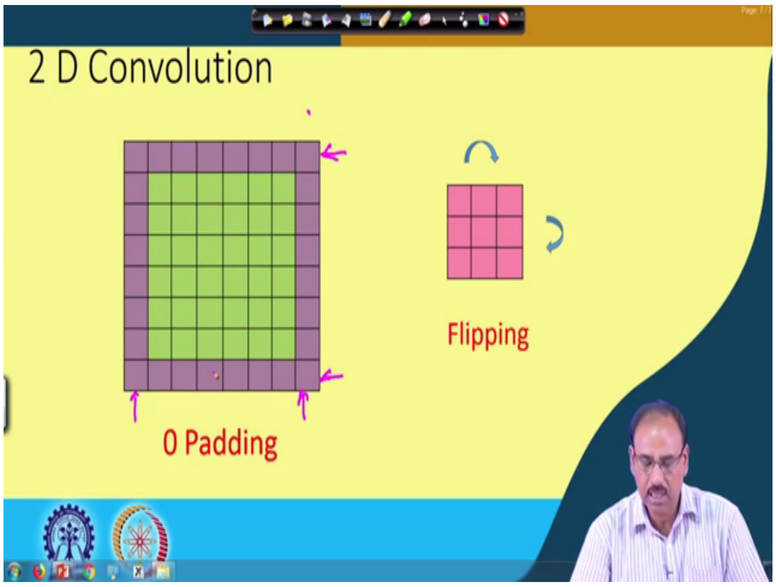
(Refer Slide Time: 15:41)



2 D Convolution

Similarly, in case of two-dimension or in case of images let us assume that I have an image of size 6 by 6 and I have a kernel of size 3 by 3.

(Refer Slide Time: 15:53)

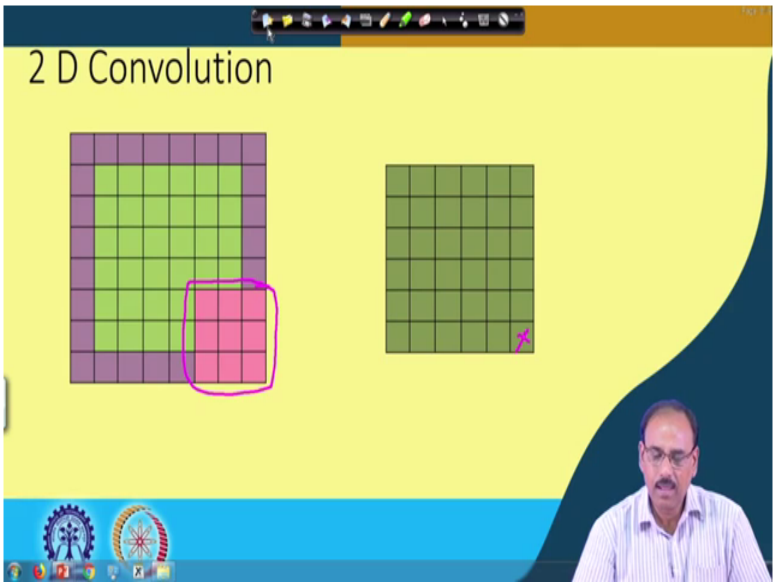


2 D Convolution

So, in order to perform the convolution the first operation that I have to do is because I have kernel 3 by 3. So, I had to go for padding with extra rows and columns all of them will have element values equal to 0. So, here you find that these are the additional rows and the additional columns which are padded, where all the values, values of all the elements in this additional rows and additional columns will be 0s. So, this is what is known as 0 padding.

Same, similarly, for the kernel also we have to flip the kernel both vertically and horizontally, around the vertical direction, around the horizontal direction. So, after fitting the flipping this two-dimensional kernel now we have to go for computation of the convolution.

(Refer Slide Time: 16:44)



So, here you find that after flipping you are placing your kernel on the pixel location 0, 0, then you are multiplying every kernel coefficient with the corresponding pixel in the input image and you sum them together that gives you the output response or the convolved image at location 0, 0 in the output image. So, this is what you are going over here.

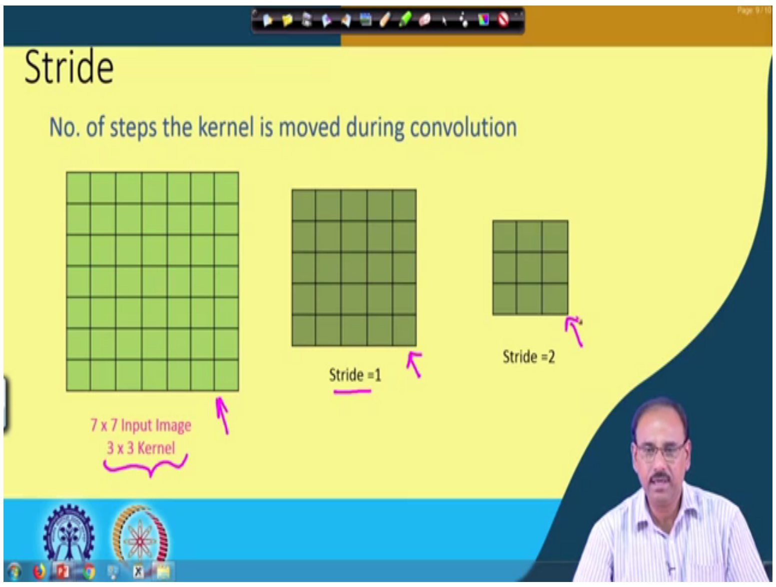
So, I am placing the kernel centered at location 0, 0 in the input image, doing multiplication of all the pixels with the corresponding coefficient the kernel coefficient sum them together and that gives you the value y 0 0 at location 0 in the output image which we are calling as feature map and the way you perform this convolution is that after convolving at this location, next what you have to do is you have to shift the kernel in the horizontal direction.

So, here we shifted by one location perform the same operation and you get the output of the next convolved value in your feature map and you continue like this, you shift the kernel again compute get the next value near feature map. Continue, and the same manner, same manner, same manner, now you come to the next row compute the same value and so on.

While doing so, as you scan or slide over the entire image computed in the convolution for at each of the locations ultimately you get your the feature map, the feature map is complete. So, I get the feature map as given on the right hand side over here. So, this is the convolution operation and in each of this case you find that when I come I compute the convolved convoluted value at any of these locations, I am actually considering the number of pixels or the inputs over a smaller region in the input image and this is what we said as your receptive field.

So, once this convolution is complete, now let us see what is the architecture of a convolutional neural network.

(Refer Slide Time: 19:23)



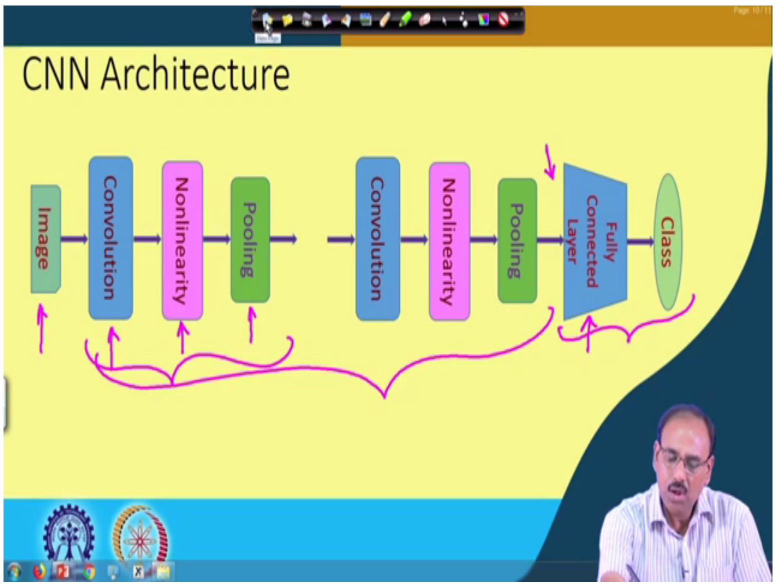
Stride

No. of steps the kernel is moved during convolution

And one more thing that when I talk about convolution you specify one more parameter which is known as stride. The stride simply says that when you shift the kernel or slide the kernel over the input image by how many locations the kernel has to be shifted. So, in the previous computations illustrations that I have shown the kernel was shifted by one pixel both in the horizontal direction as well as in the vertical direction. The stride specified in the previous case in the illustration that I have shown was equal to 1.

Similarly, if my shift is by two pixels in the horizontal direction as well as in vertical direction the stride is actually 2. So, if you increase the stride that effectively reduces the size of the feature map. So, here you find that in this case what we have shown that for an given input image of size 7 by 7 and with 3 by 3 kernel, if I do the convolution with stride equal to 1, my output image will be of size 5 by 5 whereas, if I perform the same convolution with stride equal to two then my output or the feature map will be of size 3 by 3. So, if you increase the stride, larger the stride is smaller the feature map that you get.

(Refer Slide Time: 20:53)



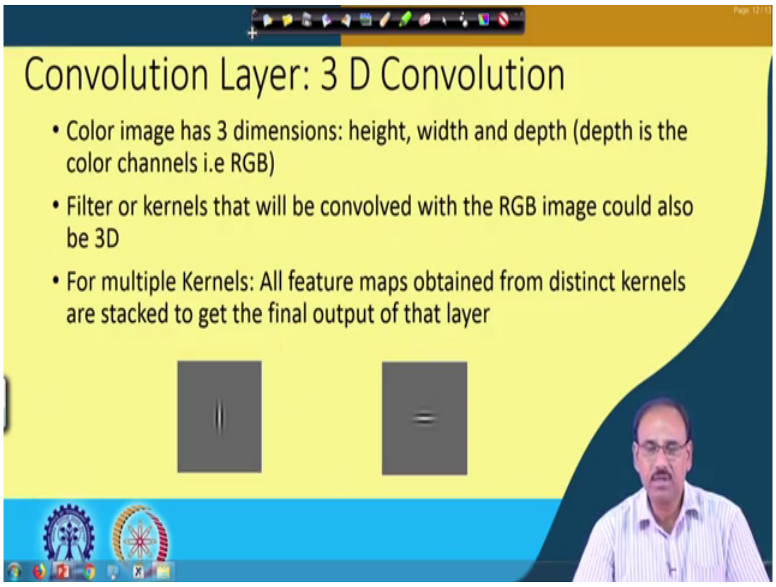
So, given this a typical architecture or CNN or convolution neural network will have the following layers. The convolution layer the convolutional neural network has a convolutional layer, a convolution layer which performs a convolution operation, then output of the convolution passes through the non-linearity. We have seen earlier the power of non-linearity that it gives a non-linear mapping of the input signal in such a way that after the non-linear mapping it becomes linearly separable. So, that is the power of non-linear mapping that we can have and along with non-linearity it also performs an operation pooling.

So, when you talk about convolution layer, the convolution layer actually performs all these 3 operations one after another. You have the convolution layer followed by non-linearity, then you have pooling layer and in a typical CNN architecture, these triplets that is convolution non-linearity and pooling they are repeated they are stacked one after another and how many such triplets will be stacked together determines what is the depth of a network, ok. Then after this is done as given over here it is followed by finally, one or more fully connected layer.

So, as we have seen that as we have a layer of nonlinearities over here it is expected that output that you get after this convolutions non-linearity and pooling operations the features after all these operations will become linearly separable or expected to be linear say linearly separable and then, I have a fully connected layer which classifies the input to one of the known categories.

So, this part of the network this fully connected network that gives you the classification. So, for an given input image you have all these operations in between finally, you can classify the input image to one of the given classes or one of the known categories. So, this is a typical architecture of a convolutional neural network.

(Refer Slide Time: 23:28)



Convolution Layer: 3 D Convolution

- Color image has 3 dimensions: height, width and depth (depth is the color channels i.e RGB)
- Filter or kernels that will be convolved with the RGB image could also be 3D
- For multiple Kernels: All feature maps obtained from distinct kernels are stacked to get the final output of that layer

Now, coming to this convolution once more, you find that in most of the cases the input image that you get is a color image. That means, till now what we have discussed is that image being a two-dimensional image and defined in say X Y domain, in a color image because the colour image has got 3 different planes red plane, green plane and blue plane. So, my input image is a actually three-dimensional image.

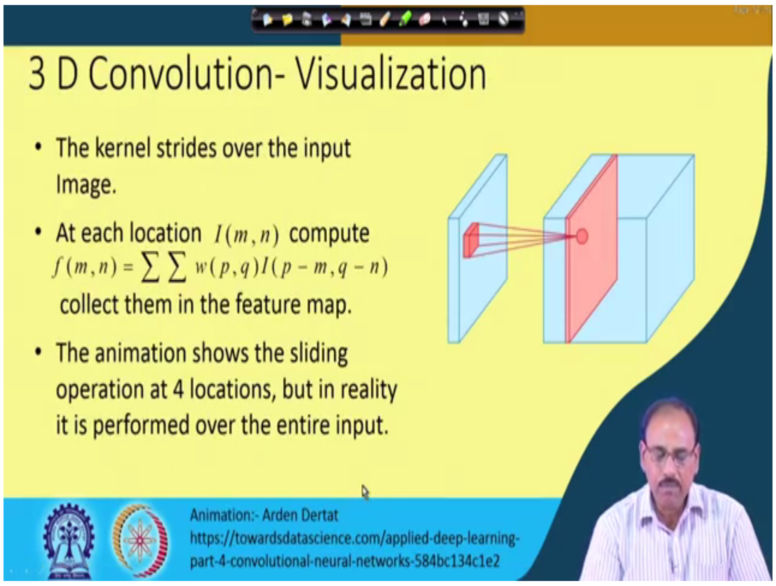
So, once I have a three-dimensional image then my convolution also has to be in three-dimension that means, the kernel the convolution kernel that you have to define is also a three-dimensional convolution kernel and if I want to determine multiple feature maps of course, that depends upon what is the kind of application that we will have, that how many feature maps I will want to generate depends upon the complexity of your input images, then for every feature map I will have a separate convolution kernel.

So, if I have say n number of convolution kernels then after convolving the input image I will have n number of feature maps. So, for every convolution kernel I will have one feature map and all these feature maps are stacked together which gives you the output of the convolutional neural network. So, as we have seen over here you find that this is one kernel and this is another kernel, with the help of these kernels we have seen what is the nature of the output in our previous slides.

So, this particular kernel gives you all the vertical edge information, and this kernel gives you all the horizontal edge information and both vertical information and horizontal information they are actually features of your input image.

So, I get one feature map with this, I get another feature map with this, so when you get the output you have stacks of these two feature maps. So, you find that the number of kernels that you will use you have so many number of feature maps and all these feature maps stacked together gives you the final output.

(Refer Slide Time: 25:46)



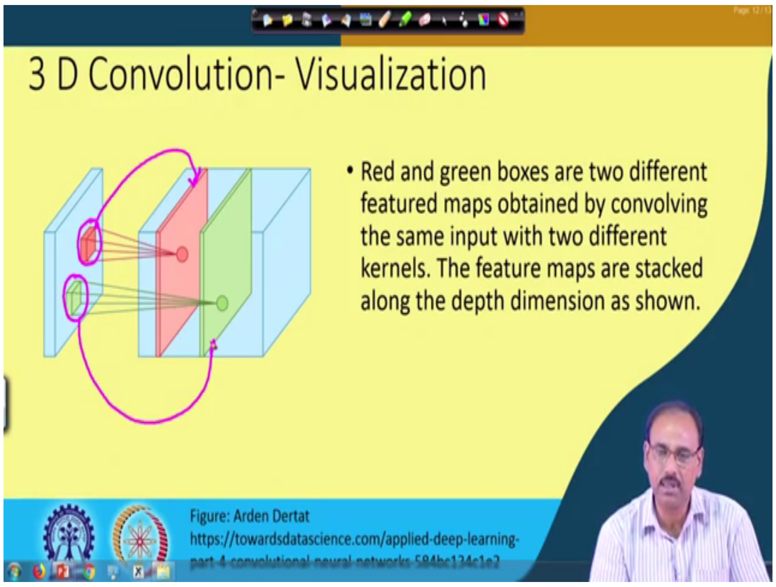
3 D Convolution- Visualization

- The kernel strides over the input Image.
- At each location $I(m,n)$ compute $f(m,n)=\sum\sum w(p,q)I(p-m,q-n)$ collect them in the feature map.
- The animation shows the sliding operation at 4 locations, but in reality it is performed over the entire input.

Animation:- Arden Dertat
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

So, I can visualize a convolution operation something like this. So, you find that at each location. So, this particular animation shows you that the kernel actually slides over the entire image and how it will slide that depends upon what is the stride that you have specified, whether the stride is 1 or stride is 2, and for every location of the kernel where the kernel is placed it computes the convolution and that gives you one sample at your output feature map. So, in this particular case it in this animation, it shows only 2 location, only 4 locations, but actually the sliding is done over the entire image.

(Refer Slide Time: 26:40)



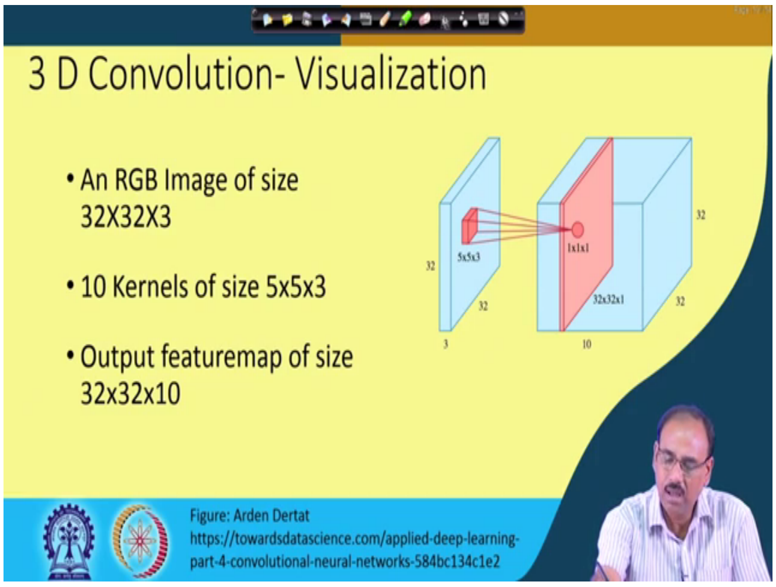
3 D Convolution- Visualization

- Red and green boxes are two different featured maps obtained by convolving the same input with two different kernels. The feature maps are stacked along the depth dimension as shown.

Figure: Arden Dertat
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

In the next operation, once I have this convolution output and as we have said that if you have multiple number of kernels you have multiple feature maps. So, here again what we have shown is, so I have two different kernels one kernel is shown by red and the other kernel is shown by green. So, this red kernel gives a one feature map which is shown over here and the green kernel shows another gives you another feature map which is shown over here. So, all these feature maps are stacked together to give you the final output.

(Refer Slide Time: 27:11)



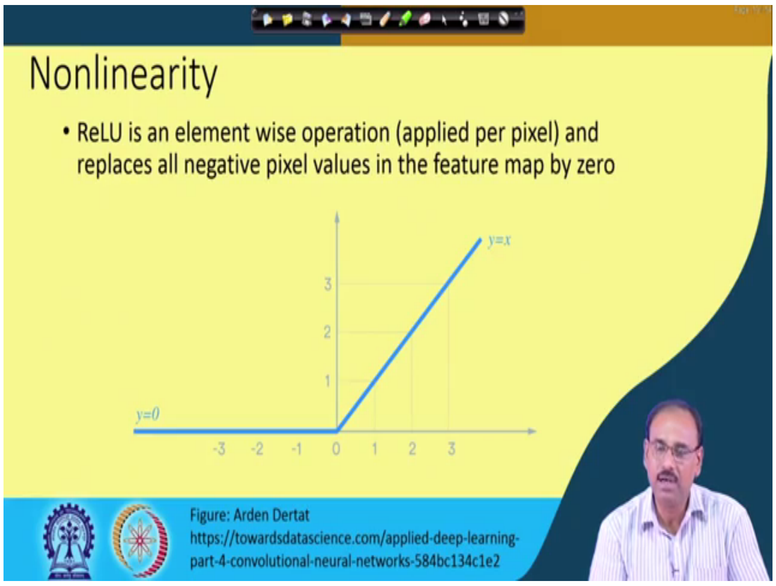
3 D Convolution- Visualization

- An RGB Image of size 32X32X3
- 10 Kernels of size 5x5x3
- Output featuremap of size 32x32x10

Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

And just an example suppose I have an input image of size say 32 by 32 by 3 and I have say 10 kernels each of size 5 by 5 by 3, and then my output feature map will be of size 32 by 32 by 10 because I have got 10 different number of kernels and I get, I assume that the input image was sufficiently padded so that your output feature map size is same as the input image and this is just for visualization.

(Refer Slide Time: 27:52)

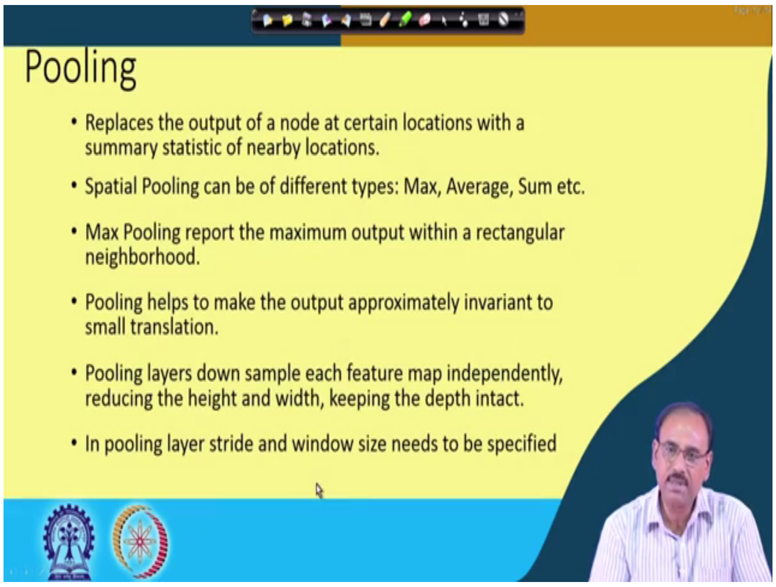

Nonlinearity

- ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero

Figure: Arden Dertat
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

So, given this visualization non-linearity function that you use after the convolution operation is usually a ReLU function. Though other nonlinearities are also possible that in case of, but in case of convolutional neural network it is ReLU which is more popular.

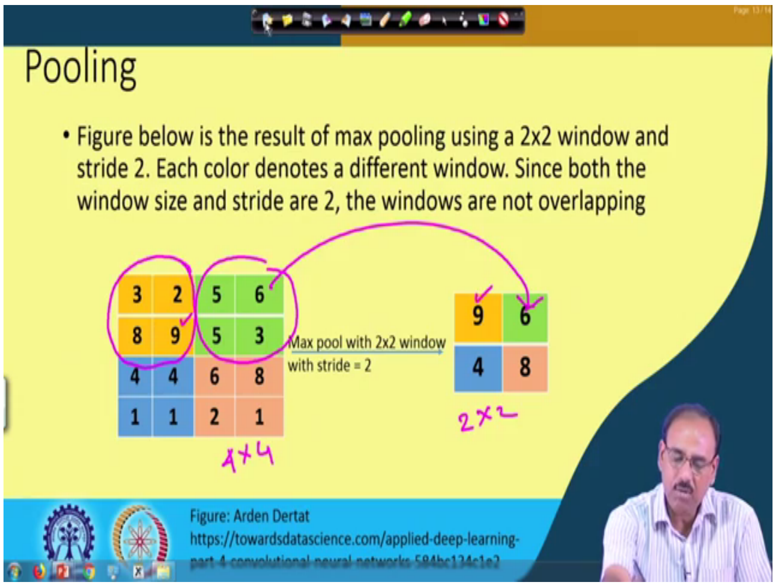
(Refer Slide Time: 28:11)


Pooling

- Replaces the output of a node at certain locations with a summary statistic of nearby locations.
- Spatial Pooling can be of different types: Max, Average, Sum etc.
- Max Pooling report the maximum output within a rectangular neighborhood.
- Pooling helps to make the output approximately invariant to small translation.
- Pooling layers down sample each feature map independently, reducing the height and width, keeping the depth intact.
- In pooling layer stride and window size needs to be specified

And after non-linearity, what we have said is that you have got another layer which is pooling layer. So, this pooling performs two tasks, one is it reduces the size of the feature map and another operation that the pooling does is at every location in the output the pooling operation replaces the output at that location with a summary statistic of nearby locations and while doing so the dimension of the feature map is reduced and the other effect of the pooling is that it makes your feature map to some extent invariant to translation.

(Refer Slide Time: 28:59)



Pooling

- Figure below is the result of max pooling using a 2x2 window and stride 2. Each color denotes a different window. Since both the window size and stride are 2, the windows are not overlapping

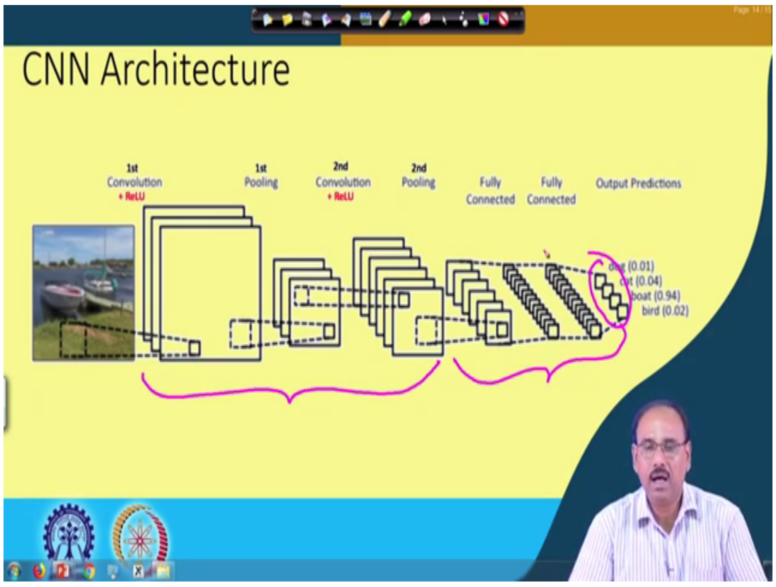
Figure: Arden Dertat
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

So, given these two again as an example, if my input feature map is as shown over here in the left hand side after performing a pooling operation, I can have different types of pooling, I can have max pooling, I can have average pooling and so on. So, here it shows a simple pooling operation which is max pooling, so on the left hand side we have a feature map of size 4 by 4 and this max pooling over an window of 2 by 2 what it does is it looks the feature values in a window of size 2 by 2, and then whichever is the maximum feature value that is put in the output.

So, as seen in this diagram that with this max pooling and with stride equal to 2 when you come to this 2 by 2 window the maximum value is 9, so that 9 is outputted as the output feature value. Similarly, over here the maximum value is 6, so 6 is outputted. And at the same time you find that the size of the feature map it reduces from 4 by 4 to size 2 by 2.

So, these are the different operations that we will have in a convolutional neural network. We will have convolution, we have non-linearity and follow it by a pooling operation.

(Refer Slide Time: 30:23)



So, given this a typical architecture of a CNN a convolutional neural network is something like this. You have an input image which is followed by a number of convolution, non-linearity and pooling operations and at the end after all these convolution non-linearity and pooling operations we will have one or more fully connected layers which are actually the classifiers, ok. And at the output layer we will have the number of nodes which is same as the number of classes that you have and this is the typical architecture of a convolutional neural network.

So, in today's lecture what we have talked about is how you perform convolution, what is the receptive field and the different operations of the different layers of operations in a convolution neural network. So, I will stop here today. In our next lecture, we will take up some of the popular convolutional neural networks which has been reported in literature.

Thank you.