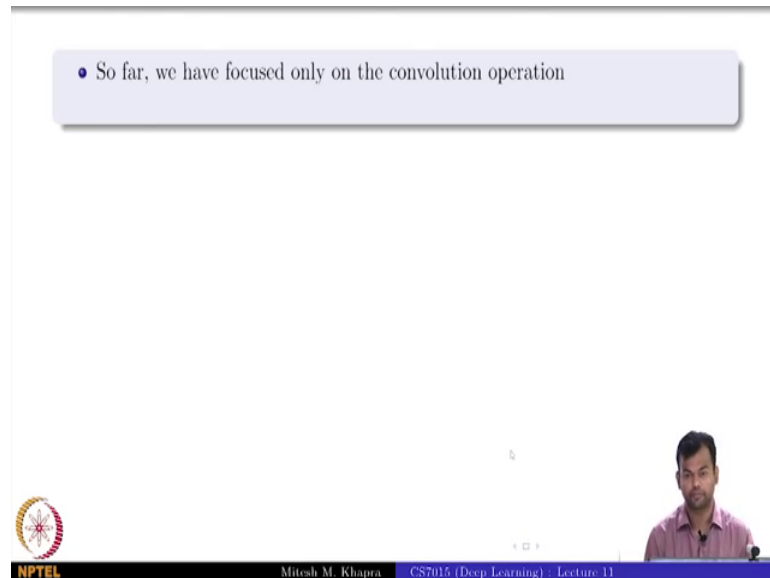


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 11
Convolutional Neural Networks (Contd.)

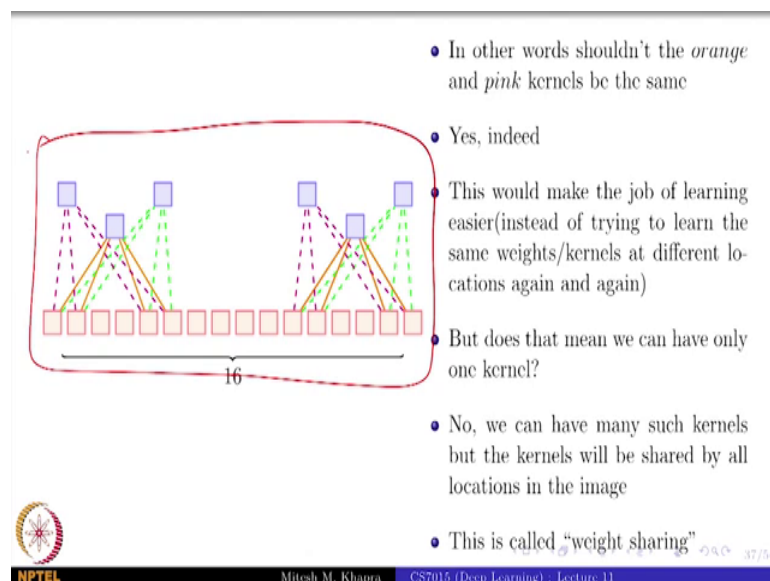
(Refer Slide Time: 00:11)



• So far, we have focused only on the convolution operation

The slide also features the NPTEL logo, the professor's name 'Mitesh M. Khapra', and the course title 'CS7015 (Deep Learning) : Lecture 11' at the bottom.

(Refer Slide Time: 00:17)



• In other words shouldn't the *orange* and *pink* kernels be the same

• Yes, indeed

• This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

• But does that mean we can have only one kernel?

• No, we can have many such kernels but the kernels will be shared by all locations in the image

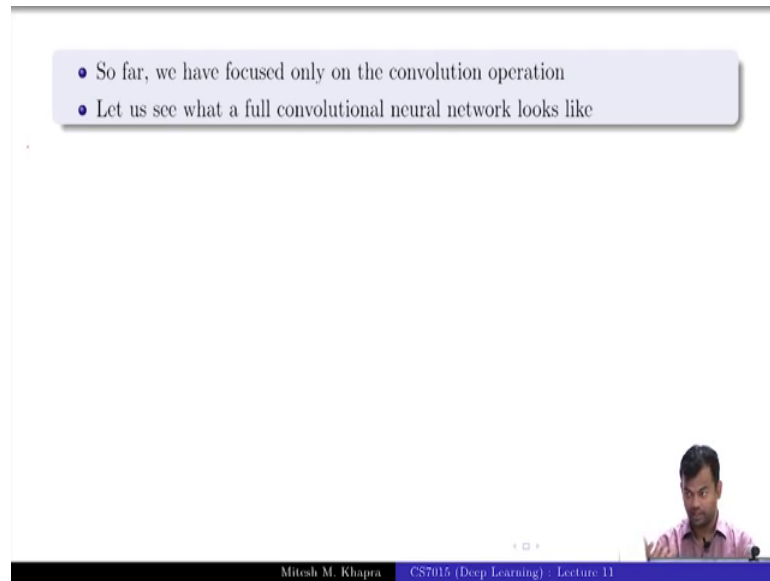
• This is called "weight sharing"

The slide also features the NPTEL logo, the professor's name 'Mitesh M. Khapra', and the course title 'CS7015 (Deep Learning) : Lecture 11' at the bottom.

So, we will start from where we left yesterday. So, this is what we had seen yesterday, we saw what's the difference between a convolutional neural network and a feed forward

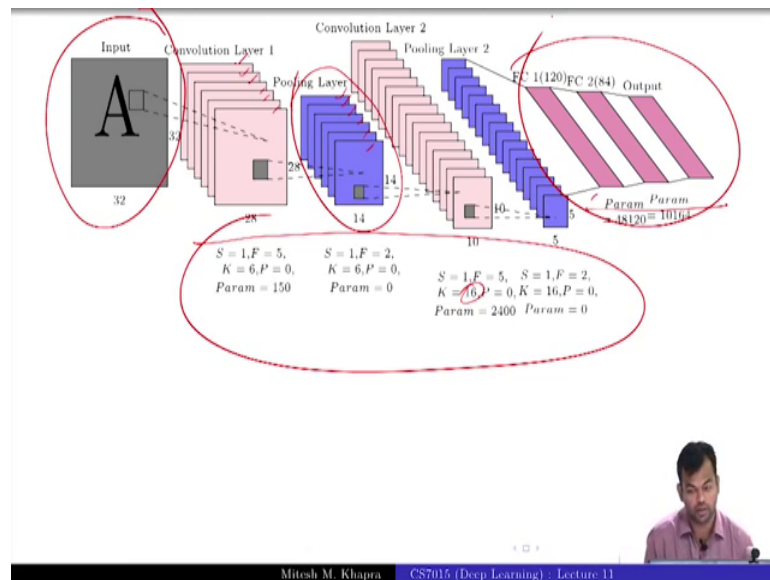
neural network and we focused on two main properties, one is sparse connectivity and the other was weight sharing. That is about it and then we saw that this representation of I mean, we saw this diagram about how to you could have multiple kernels and each kernel would apply across the entire image and the weights would be shared for that kernel.

(Refer Slide Time: 00:43)



So, far we have only focused on the convolution operation and even when you have seen the neural network or the convolutional neural network, we have only seen the convolution layers right. So, there is something more in a typical convolutional neural network and that is what I was about to start yesterday. So, we just continue from there.

(Refer Slide Time: 00:59)



So, this is what a full convolutional neural network looks like. So, ignore these things for now, all these parameters etcetera as a just ignore for them, I will just walk you through, what is important in this diagram right.

So, your input is an image and the tasks that you are dealing with here is digit recognition or handwritten digit recognition right and what you see here is that you have taken an input which is a 2 dimensional input and then the next layer you actually see 1, 2, 3, 4, 5, 6 outputs. So, what does that mean?

Student: (Refer Time: 01:33).

You have use 6 filters apply them, throughout the filter, throughout the image each filter, gave you 1 feature map. And so in this layer, you have 6 such feature maps. So, the original 2 dimensional input has now become 6 2 dimensional outputs, after that there is something known as a pooling layer, we will see what a pooling layer is in detail.

Now, what I want you to understand is let us assume that, what the pooling layer does is it does some kind of a shrinking, it takes the original output and shrinks it, how it shrinks it? We will see in a while, but let us see what happens after that. So now, you have 1, 2, 3, 4, 5, 6 as your input. So now, you have this is your input, this volume is now, your input I call it a volume, because it has depth height and width and now you are again

going to apply convolutions to that and what do you see, how many outputs do we have now? We have 16 outputs right. So, what does that mean?

Student: (Refer Time: 02:27).

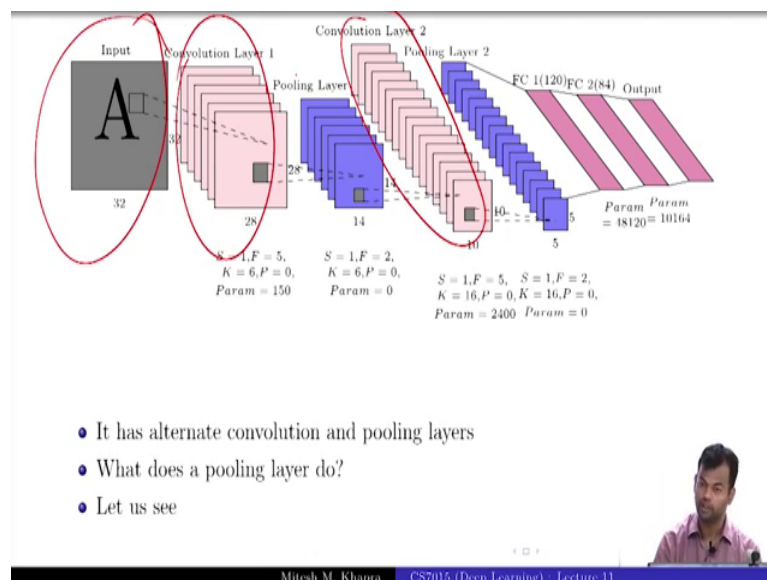
I took this 3 D input applied 16 3D filters on it each 3 D filter give me 1 feature map, one 2 D feature map, why one 2 D feature map?

Student: (Refer Time: 02:39).

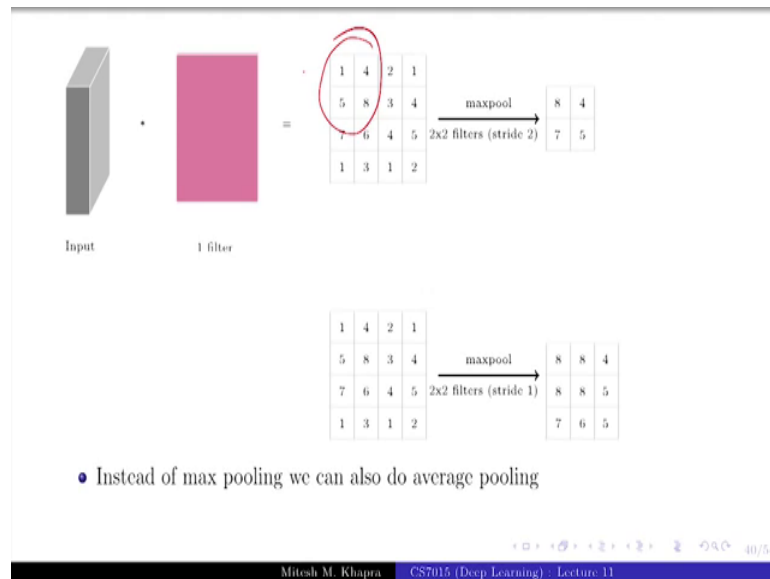
Because we are doing a 2 D convolution, we are taking a 3 D filter, but we are doing a 2 D convolution right and then after that again this becomes my input and then do a max pooling on top of that and then something else happens after that. So, there as which we will come to later.

So, right now I just want to say that there is this input, then you come out will come with some output after applying convolutions, now this becomes the input for your next stage, where you have done pooling. Now the output of pooling becomes the input for the next stage. So, you will take this as the input, apply some convolution on that get some output and continue in this way right and we will come back to what these are.

(Refer Slide Time: 03:19)



(Refer Slide Time: 03:23)



So, now let us dive deeper into what is pooling? What does the pooling layer actually do? So, here is your input again, it is a volume and now when I say input, you should not just think of the input as this remember that, all of these can be inputs right. So now, at every stage once, you have got an output for the next stage that becomes, input that is typically how it was even in the feed forward neural network right, once you compute a hidden representation that is the input to the next layer.

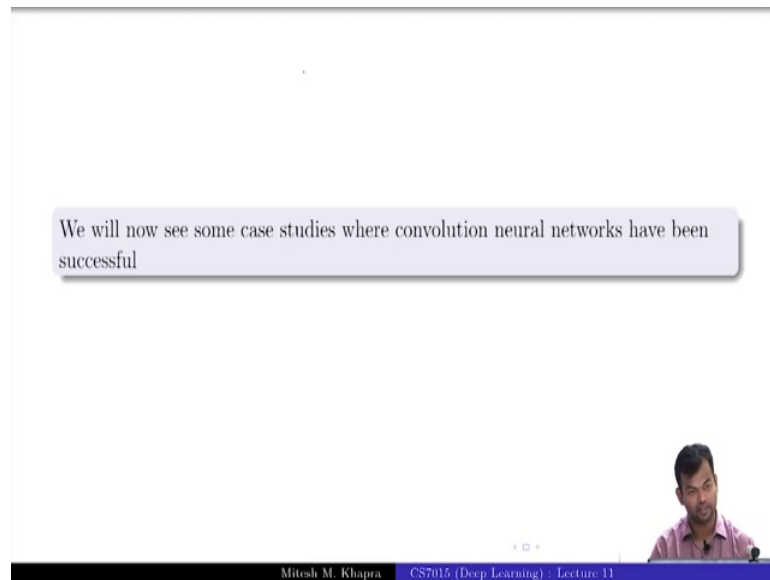
So, I have some input at one of these layers either, the input layer or any of the intermediate layers and I apply a filter on that and that filter gives me some 2 D output, it gives me 1 feature map and let us say, this is what the feature map looks like.

Now, what does the pooling operation do? So, I would apply 2 cross 2 pooling with a stride off 2. So, let us see what that will do? That means, I look at this 2 cross 2 region, I will pick up the max value from there that is why, this is max pooling. So, the max value is 8, I will just keep that then I am going to do a stride of 2; that means, I am not going to place it on this block, I am just going to shift to the next block. Again, I will take the max from there, which is 4 right and I continue this and I get this is the output. So, you see why the shrinkage happens because, I am taking a 2 cross 2 area and I am shrinking it by a picking up only one value from there right.

So, it actually kind of half the width and half the breadth so, total of 1 by 4 reduction is what you get, but you could also use a filter with stride off 1. So, this is what it would look like. So, you will place it here, take the max value then, the max value then, the max

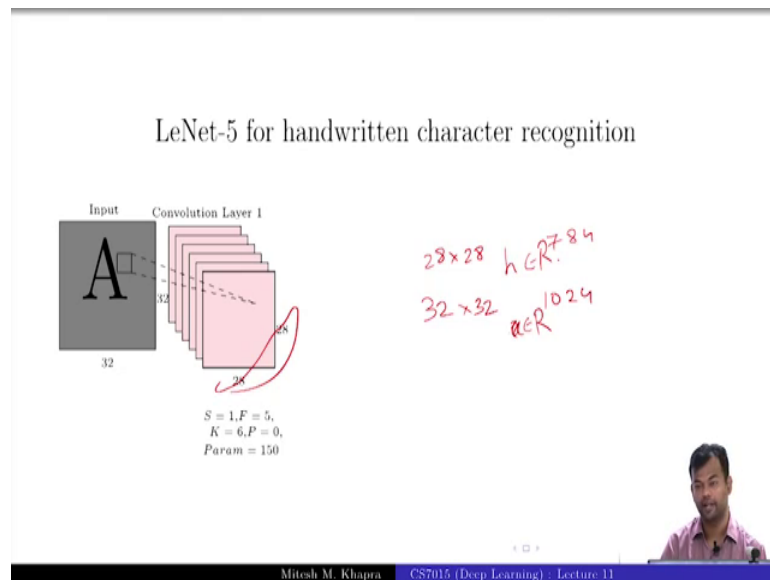
value and so on right. So, in that case you will get a lesser reduction and instead of max pooling, you could also do average pooling; that means, instead of taking the max of these 4, you could take the average of these 4, is it fine. So, is the pooling operation clear and how it results in the reduction of the size of your input.

(Refer Slide Time: 05:15)



So now, what we will do? Is now that we have some idea of what a full convolutional neural network looks like. So, it looks like alternating convolution and max pooling operations, we know what a convolution operation looks like, in particular we know that a 3 D filter applied to a 3 D input results in a 2 D output, because we are not applying the convolution along the depth, we just applying the convolution along the width and the height right. So, that is what we know so far.

(Refer Slide Time: 05:47)



And based on this knowledge, now we are going to see some success stories of convolutional neural network right.

So, we will start with the first one, which is LeNet-5. So, this was already the fifth version, this was around 97 or 98 or something and I had mentioned this, when we were doing the history lecture.

So, this is the input, now you have decided to apply 6 filters, you have said that the stride is going to be 1; that means, you are going to place at every location, the spatial extent is going to be 5 and the padding is going to be 0. Now, the question that I have for you is, how many parameters does this convolution layer have? What are the parameters here?

Student: The weights.

The.

Student: (Refer Time: 06:25).

The filters, the weights in the filters, how many filters do have?

Student: 6.

6, how many ways does each filter have?

Student: 25.

20.

Student: 5.

25 right, 5 cross 5 is a 25. So, the total number of parameters is 150. Now, I want you to appreciate something here. So, the input was actually 32 cross 32, which I believe 1024 and the output was 28 cross 28 right, that is what the output you got that is, I guess 784 yeah. So, in a feed forward neural network, if you had x belonging to \mathbb{R} raised to 1024 and your h belonging to \mathbb{R} raised to 784, how many parameters would you need? 1024 cross 784, how many parameters do you have here?

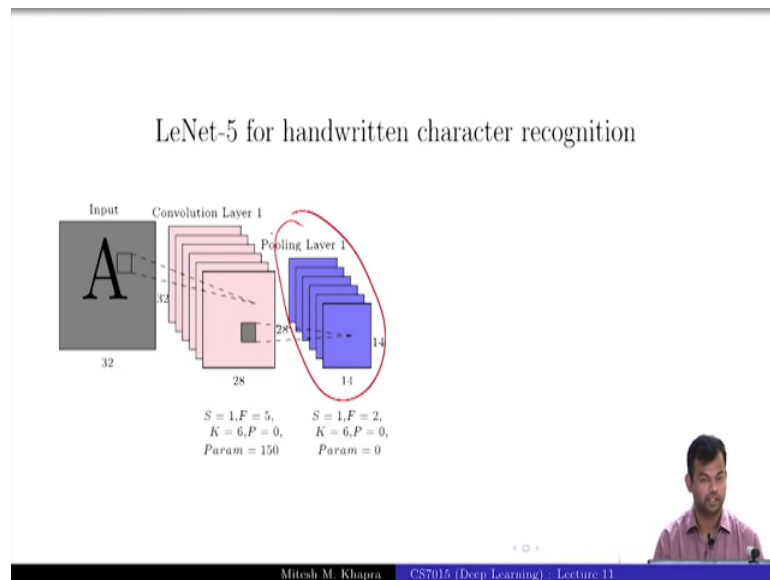
Student: 150.

150 much much smaller right, this is because of

Student: (Refer Time: 07:21).

Both sparse connectivity as well as weight sharing right, so now, you appreciate the difference between the two and that is one of the reasons that even, before this is for example, from 97, 98, 1997, 1998 right. So, even before the deep learning wave of 2006 or the revival after 2006 right, convolutional neural networks must still being trained for even deep networks 4 to 5 layers. Because they had much fewer parameters and that is why it was relatively easier to train them as compared to a very dense feed forward neural network.

(Refer Slide Time: 07:55)



So, I want you to appreciate that fact. Now, after this I have the pooling layer, when I am going to use a stride of 1 and F equal to 2; that means, I am going to pick up. So now, I am going to use a max pooling layer, where I am decided to use a stride of 1 and the max pooling will happen, in the region of 2 cross 2 and again K 6, because there are just 6 filters. So, max pooling happens on every feature map independently, it does not happen across the depth; that means, I am not going to pick the max along these 6 layers, I am going to pick the max along each of these feature maps. So, it is a per feature map operation.

And this since it is a 2 cross 2 filter, it will result in a size reduction and from 28 cross to 28, I will get to 14 cross 14, how many parameters is the max pooling layer have?

Student: (Refer Time: 08:42).

Wow good, there is no parameters in the max pooling layer, because you are not having any weight matrices, just taking the input and applying a simple max operation on that, there is no w transpose x or any kind of a transformation happening there. Now, this becomes your input what's the size of this volume?

Student: 32.

14 cross 14 cross.

Student: 6

6. So, all the filters that I am going to use from now on, what is the depth of those filters going to be?

Student: (Refer Time: 09:10).

What is the depth going to be?

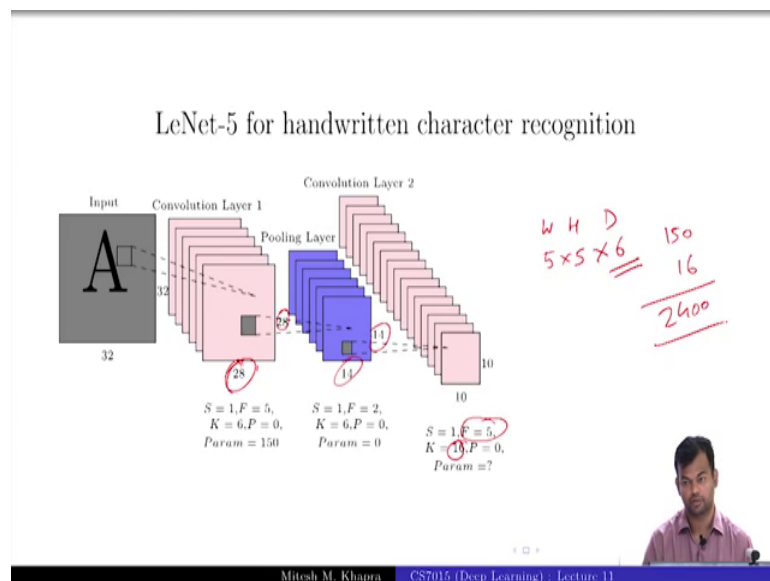
Student: 6.

I want to everyone to answer.

Student: 6.

6 right, because we are always going to assume that the depth is equal depth of the filter is equal to the depth of the input.

(Refer Slide Time: 09:19)



Now, here they decided to use 16 filters and by the way, you did hopefully notice that, this 28, how did you get it, from which formula?

Student: W_n minus (Refer Time: 09:34).

W_n minus F cross 2 P plus 1 right. So, that is the formula.

So, now we have a 14 cross 14 input and you have 16 filters. So, what is the depth of the output volume going to be?

Student: (Refer Time: 09:46).

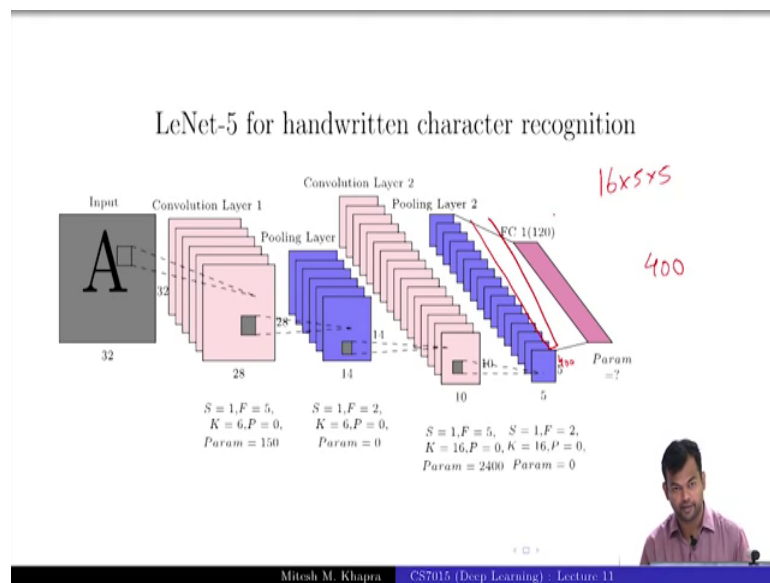
What is the depth of the output volume going to be?

Student: 16.

16 right and. So, you have 16 and you have a spatial extent of 5 cross 5, just a minute spatial extent of 5 cross 5, how many parameters does this layer have? I want everyone to say it.

Student: 400.

(Refer Slide Time: 10:11)



400 fine so, that is.

Student: (Refer Time: 10:10).

There are 16 of these each is 5 cross 5.

Student: (Refer Time: 10:13).

Into 6 into D good, then we made a mistake here also, no there the depth was 1 are you, do you get it how you got 2400 right, we forgot about the depth. So, each of these filters

is 5 cross 5 cross depth right, what is the depth? 6 the same as the input, right. So, each of these filters is 25 into 6, which is 150 into 16 yeah fine, is that ok? Did I confuse you or everyone back on track? Pooling layer, I should have been 2, because of that half reduction using.

Student: (Refer Time: 10:49).

Yeah maybe, can we just check this? I think it should be 2. There was a question.

Student: (Refer Time: 11:00).

Student: (Refer Time: 11:02).

Go from the pooling layer to the next layer from here to the next layer.

So now, what is the depth of your input volume? 6 and what is the width and height? 14 cross 14. So now, every filter that I going to apply at the next layer is going to have a depth of 6. So, I have decided to apply 16 such filters. So, what is the depth of this layer going to be? The depth of the output is equal to the number of filters. So, the depth is going to be 16 and all my filters are 5 cross 5, but what we forgot is that, when I say the filter is 5 cross 5, it is actually 5 cross 5 cross 6, because the depth is also there right. So, it is width into height into depth. So, that is this number of parameters in each of my filters right and that is 150 and I have 16 such filters. So, that gives me a total of 2400 parameters, is it fine?

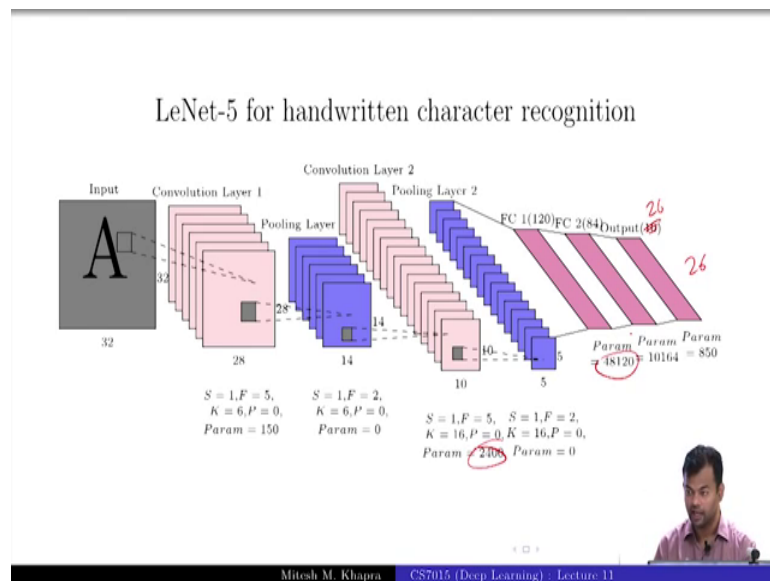
So now, we have a volume of size 16 cross 10 cross 10, now I am going to do max pooling on that maybe, again this should be 2, there was the same doubt you had fine. So, it will result in a reduction in the output and now what is the volume? What is the size of this volume? 5 cross 5 cross 16 and the parameters is 0, max pooling layer does not have any pooling.

Now, after this what we have is something known as the fully connected layer. So, now, as I said the size of this volume is 16 cross 5 cross 5. It is arranged in these feature maps, but I can always flatten it to get one single vector, do you get that? So, from these 16 feature maps each of 5 cross 5 size, I can flatten it out and get the single vector of size 400, do you get that. So, that is what, I do in the fully connected layer.

So now, I am going to flatten this treated as a single vector and then fully connect it to the next layer, what do I mean by fully connected? Dense connections, no more sparse connection. So now, we have a feed forward network from this point of view. So, you have 400 and that connects to a layer of size 120. So what are the number of parameters?

Student: 400 into 120.

(Refer Slide Time: 13:13)



400 into 120, plus we will have 120 biases. So, that is what this number is fine. So, this is one fully connected layer of size 120, after that I have another fully connected layer of size 84. So, the number of parameters would be 120 into 84 plus 84 and after that I have the output layer. So, the output layer this was 26 or what is the output?

Student: (Refer Time: 13:31).

But this is digit, this is alphabet recognition right. So, probably they have done the computation using 10, but it should have been using 26 as the output layer, because you want to predict one of the 26 alphabets. So, you can assume this is 26. So, it would be 84 cross 26 plus 26 right that is the size of the output layer right.

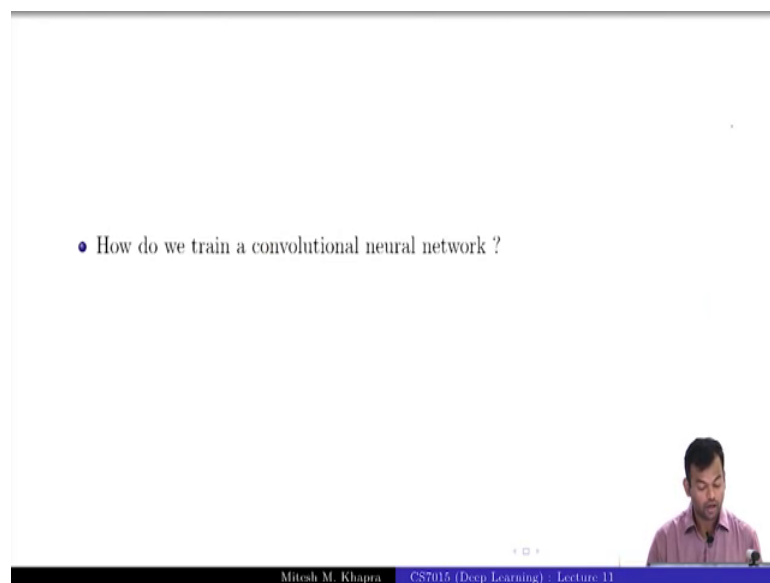
Now, do you observe something immediately is a something very striking immediately, in terms of the number of parameters?

Student: (Refer Time: 14:00).

The fully connected layers clearly dominated right here, we were dealing of order 2400 and max and here we just start with 48000 itself right. So, just keep this in mind that the fully connected layers have the largest number of parameters, that you have and we will try to come back to this and see if we can solve this problem.

So now, when you see a convolutional neural network, you should be able to reason about the following things. At each layer, what is the size of the input volume? What is the size of the output volume? What are the number of filters being used? And what are the number of parameters in that layer? Right, if you can reason these things and you have really understood, what is actually happened and unless you can reason these things, I do not see how you can efficiently code it up right. So, you should be able to know that, this is the size of the input, this is the size of the output and so on and I guess all of us are comfortable with others.

(Refer Slide Time: 14:51)



So now, how do we train a convolutional neural network? What is the answer? your nodding; that means, you do not know or you know it is too trivial to even ask this question, how will you train it?

Student: (Refer Time: 15:00).

But how do you back propagate through a convolution operation? That is a very nasty looking operation.

(Refer Slide Time: 15:07)

Input

b	c	d
e	f	g
h	i	j

Kernel

w	x
y	z

- We can thus train a convolution neural network using backpropagation by thinking of it as a feedforward neural network with sparse connections
- A CNN can be implemented as a feedforward neural network
- wherein only a few weights (in color) are active
- the rest of the weights (in gray) are zero

NPTEL Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

CNN can be implemented as a feed forward neural network with what being the difference?

Student: (Refer Time: 15:15).

Only some of these weights would be active, all the gray weights will not exist only the colored weights will exist right. Now can you back propagate to this network have you seen something similar before?

Student: Dropout.

Dropout right so, if you could do that you can do this also. So now, if you take this view of a convolutional neural network where so, this is just to give you an intuition or make you feel confident, that once you know the backpropagation algorithm, there is nothing much different from training a convolutional neural network operation.

So, everyone is fine with that, how many of you agree with that? That you can actually train using the same algorithm with some smart coding required to make sure that these weights are not active and so on. But in practice of course, you will not do this in practice, you will define the convolution operation, you will also define the derivative of the convolution operation and then use that right, because that would be much more efficient, this is very inefficient right because, you are assuming that there is a fully

connected network and then some things do not exist there. So, that is not that is the whole point, I mean you wanted to avoid such dense connections right.

But in principle you could have just used this and trained the convolutional neural network. In practice, you do not need to worry, because you just need to define your forward convolution operations and people like Google and all who release tensor flow, torch and all will do the hard work of doing the back propagation for you right. So, you never have to write back propagation in your life, apart from what you have already written in the assignment right that is why I make you go through that torture once.

So now afterwards whenever you use any of these platforms or pi torch or torch or tensor flow, the back propagation comes for free, you just need to write the forward progression; that means, you just need to write convolution operations and you do not need to worry about how the derivatives will be computed. But I what I want you to understand is that conceptually, it is the same, you can still use or in fact, you still use the same back propagation algorithm, to train a convolutional neural network also, everyone is fine with this? So now, we know how to trained a convolutional neural network also.