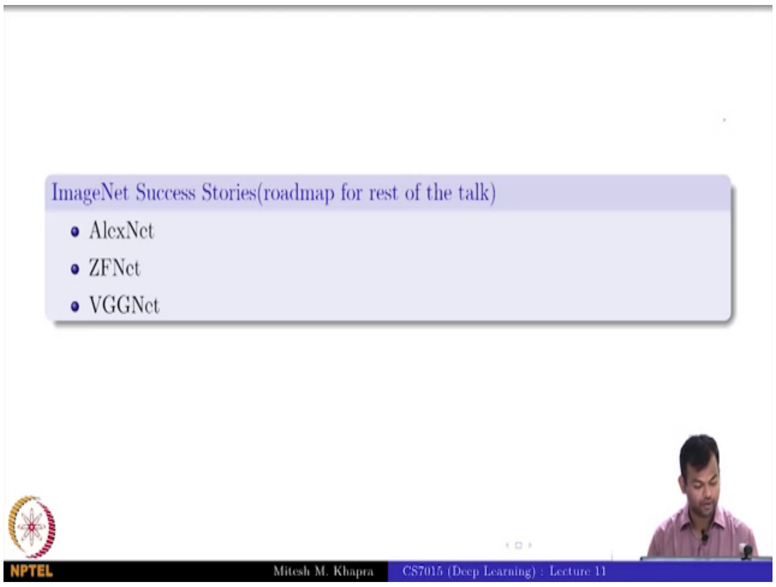
Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 11
CNNs (success stories on ImageNet)

So, now, we will go to the next module, we will talk about some success stories on ImageNet right. So, this is the challenge which actually made convolutional neural networks very famous back in 2012.

(Refer Slide Time: 00:23)

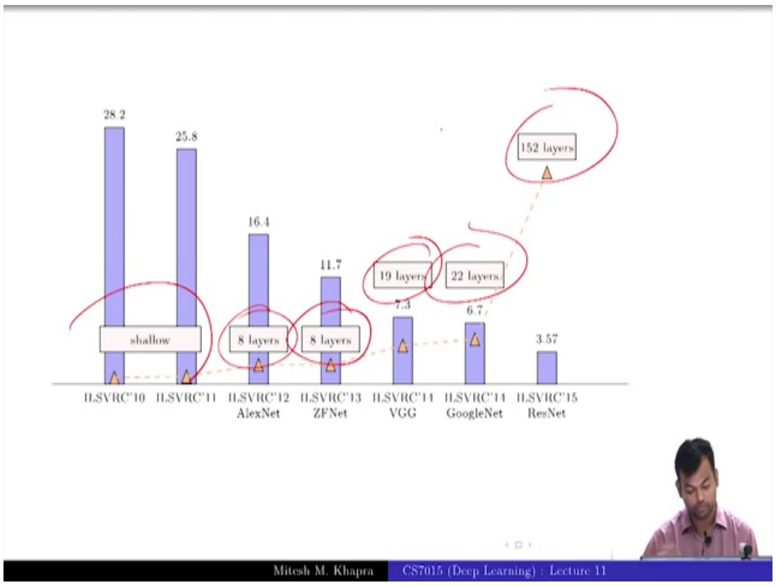


ImageNet Success Stories(roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet

So, they are going to look at some algorithms in fact, 2 more hopefully today.

(Refer Slide Time: 00:29)



So, this is the story right. So, there is this challenge or competition called ImageNet Large Scale Visual Recognition Competition right that is what ILSVRC stands for. And this was a data set created which had 1000 categories, it actually has 10000 categories, but in the competition we use only thousand of those categories yes.

So, 1000 plus 1000 I think, the roughly the data set size is 1 million. And so, that is what was used for training a classifier. And I am talking about 2010, the pre deep era right I mean so, of course, deep era networks existed at that time, but the participants and these challenges and that time were relying on the classical machine learning approaches. So, what was that approach, take the image.

Student: (Refer Time: 01:16) feature.

Extract features which features?

Student: (Refer Time: 01:20).

Predominantly.

Student: Sift and hawk.

Sift and hawk features were the predominant features at that time and then you train a classifier on top of that. And then, you use things like on symbols or better handcrafted features and things like that certain more tricks on top of that. So, that, with that on this data in 2010 the error was 28.2 percent; that means, if I give you a test set of 1000 images you will make 282 errors on that right, that is what this means. Then, in 11 there was still some progress, this was again pre deep era and there was this error came down to 25.8. And then, in 2012 there was this AlexNet, which was a deep convolutional neural network applied to the task of image classification and it gave a dramatic reduction right from 25.8 to 16.4.

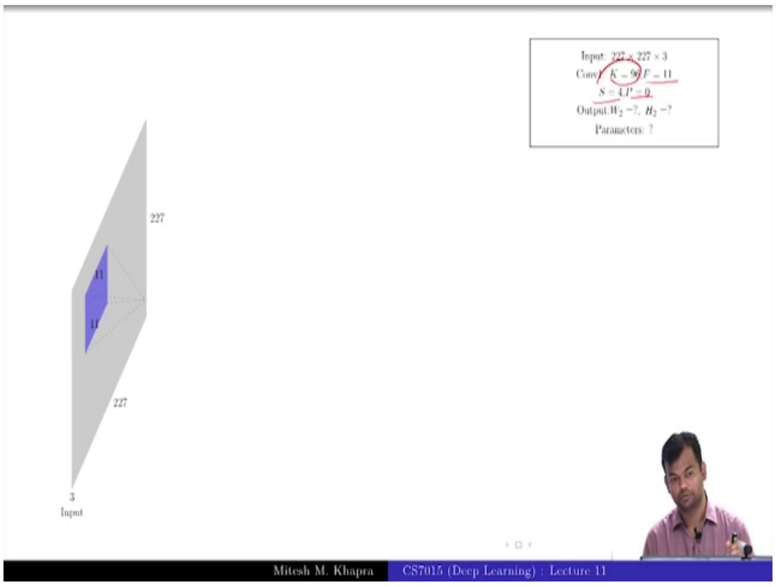
And was I think absolute; in absolute terms 8 to 9 8 to 9 percentage better than the best system in that competition that year, so that was in 2012. In 2013, there was further improvement on a different architecture for doing this and that give a further error reduction of 11.7. Then in 2014 there was VGG net. So, these are all 3 that we are going to see today, which give a further error reduction of 7.3. Then, Google decided to join the party and they make it 6.7 and as I have said before then afterwards Microsoft got crazy and they brought it out in 3.57 and this is when we started making claims that a convolutional neural network has become better at this task than humans right.

Because, if you show these 1000 images to a human, even a human is bound to make a 3.5 percent more than 3.5 percent error; that means, because some of these images would be blurred. So, I would not be very sure whether, this is a bulldog or a different type of dog or something like that right. So, I even a human cannot really recognize it correctly and that is the whole hype around, how convolutional neural networks have beating human level performance on this particular task right.

And let us see so, this was all the shallow pre deep era. The first architecture was 8 layers and I think this was called a varied no this probably not this yeah. The second architecture was also 8 layers then, we had 19 then, 22 and then 152 right. So, that is how the progress has happened. So, these are all the architectures that we are going to look at today or at least some of them today and the rest maybe tomorrow.

So, we will start with AlexNet and I am going to tell you the exact architecture of AlexNet, what was refused, what did it actually use.

(Refer Slide Time: 04:02)


Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$

$S = 4, P = 0$

Output: $W_2 = ?,\ H_2 = ?$

Parameters: ?

So, the input was an RGB image. So, it had a depth of 3 and it was 227 crossed to 27 that is what the data set input was, all the images in the data set were to 27 cross to 27 cross 3. So, the first thing that they did was they decided to use 96 filters, can you read that anyways I will say it outright.

So, they resided to use 96 filters with a spatial extent of 11 cross 11, a size of 4, and padding of 0. So, the moment you see size stride of 4, what do you know is going to happen, there is going to be some shrinkage, roughly by how much one-fourth right. So, now, can you compute these 3 things, what was W 2, what was H 2 and what was the number of parameters in this layer. We will do it for a few of these layers and then I will just rush through that. So, what is W 2 going to be, you have already done this computation right, the exercise that we did was exactly this computation. So, there was 55 cross 55 and what is the depth going to be, I want everyone to say that.

Student: (Refer Time: 05:15).

And what is the number of parameters.

Student: (Refer Time: 05:20).

96 into.

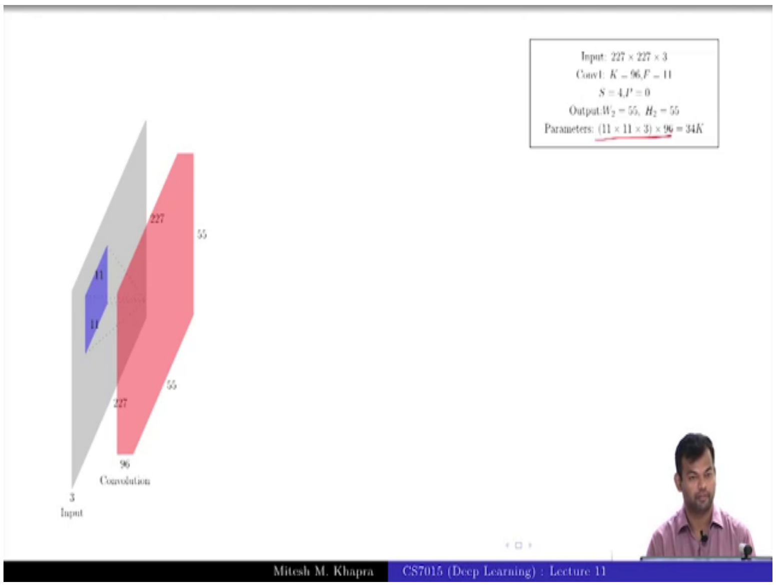
Student: 11.

11 into.

Student: 11.

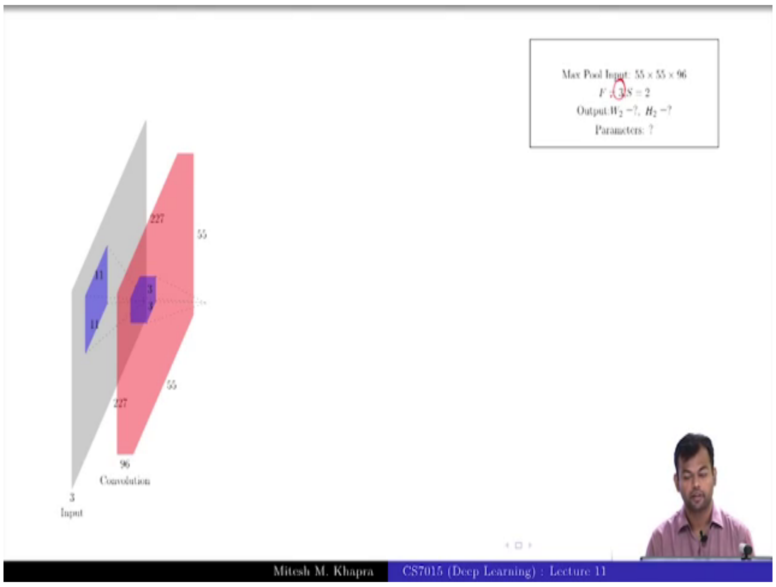
11 into 3, do not forget the depth, the depth is 3 here.

(Refer Slide Time: 05:31)


Input: $227 \times 227 \times 3$
Conv1: $K = 96, F = 11$
$S = 4, P = 0$
Output: $W_2 = 55,\ H_2 = 55$
Parameters: $\underline{(11 \times 11 \times 3) \times 96} = 34K$

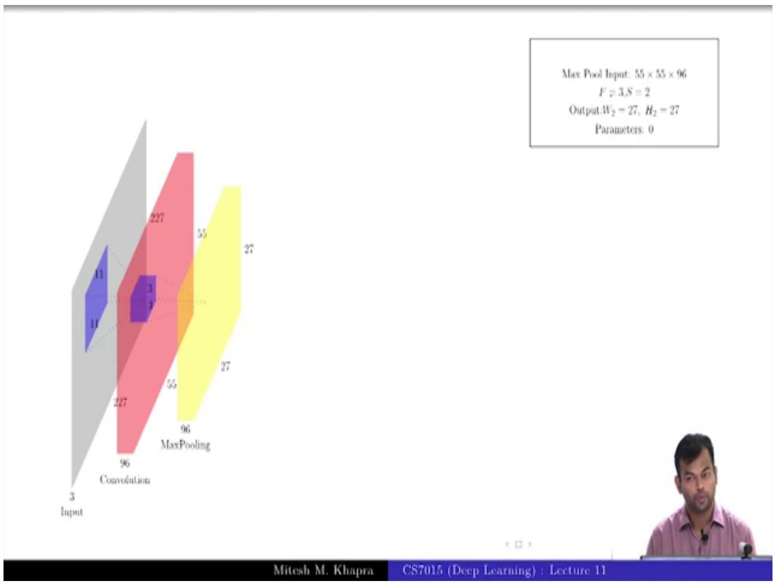
So, that is the number of parameters that they had in this layer, 11 into 11 into 3 into 96. Now, what is the next layer going to be, a max pooling layer.

(Refer Slide Time: 05:42)


Max Pool Input: $55 \times 55 \times 96$
$F = \bigcirc 3, S = 2$
Output: $W_2 = ?,\ H_2 = ?$
Parameters: ?

So, they had a max pooling layer, where they used a 3 cross 3 max pooling; that means, you are going to pick up max from a 3 cross 3 grid. And the stride was 2; that means we are going to get half the output. And now can you tell me, what W 2 H 2 would be roughly? Half of 55, 55 right so, 27 27.

(Refer Slide Time: 06:02)



Max Pool Input: $55 \times 55 \times 96$

$F = 3, S = 2$

Output: $W_2 = 27$, $H_2 = 27$

Parameters: 0

And what is the number of parameter is going to be? Do not be lazy, everyone be say it.

Student: 0.

0 right so, that is the max pooling layer.

Now, what is the size of your input volume at this point?

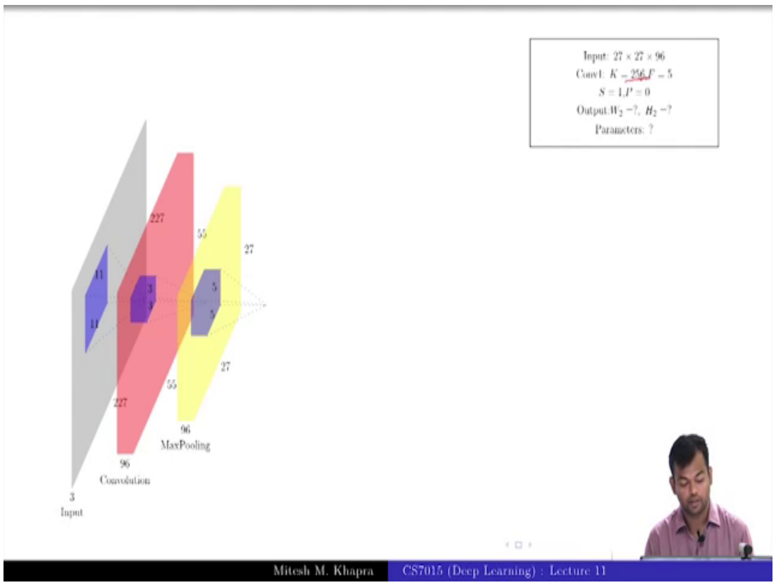
Student: 27.

27 cross 27 cross.

Student: 96.

96 as opposed to the original input which was 227 cross 227 cross 3, so as you keep progressing, your width and height is decreasing, but your depth is increasing because, you are using more and more filters to capture more and more patterns in the images.

(Refer Slide Time: 06:37)


Input: $27 \times 27 \times 96$

Conv1: $K = 256, F = 5$

$S = 1, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: ?

Now, so you have 27 cross 27 cross 96 then, they decided to use 256 filters, each of size 5 cross 5 with a stride of 1 and padding of 0, is it right? So, how many parameters do you have now?

Student: (Refer Time: 06:53).

256 into.

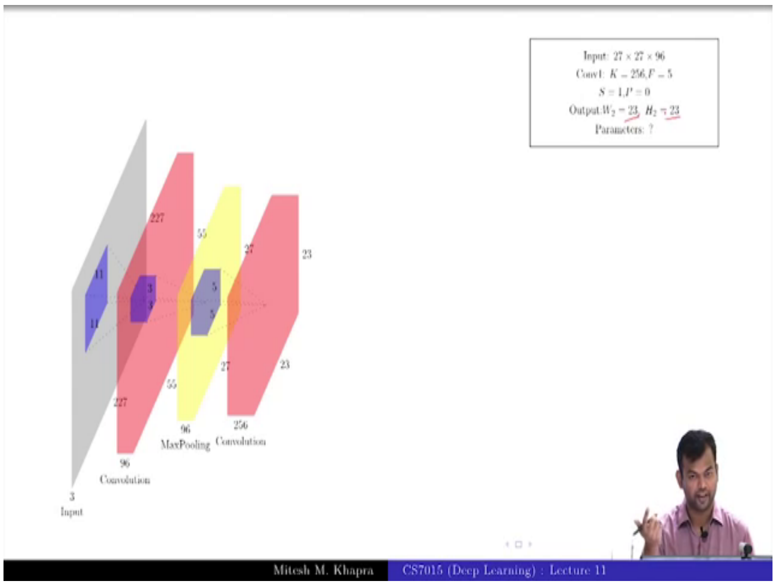
Student: 5 into 5.

5 into 5 into.

Student: 96.

96.

(Refer Slide Time: 07:00)


Input: $27 \times 27 \times 96$

Conv1: $K = 256, F = 5$

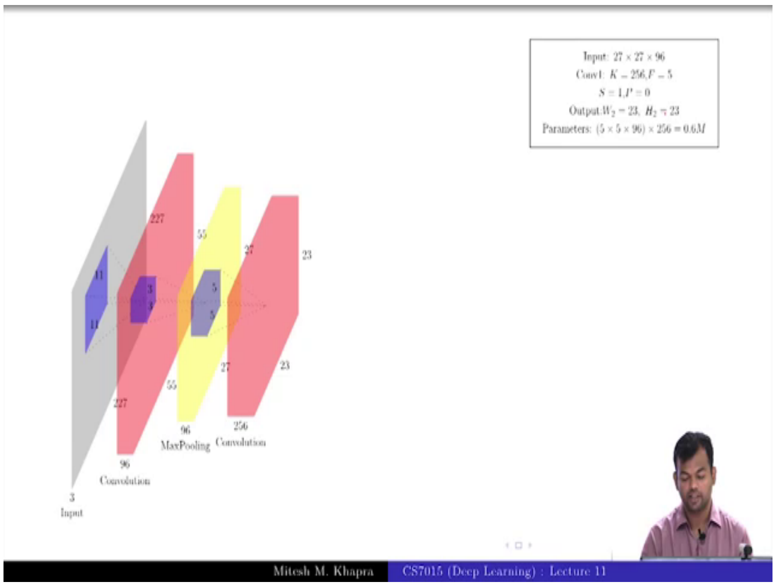
$S = 1, P = 0$

Output: $W_2 = 23$, $H_2 = 23$

Parameters: ?

So, that is the number of parameters that you will have and the size since would decrease only by 1 right because, you have a stride of it will decrease by 2 because, a filter size is 5 and you have a stride of 5.

(Refer Slide Time: 07:12)


Input: $27 \times 27 \times 96$

Conv1: $K = 256, F = 5$

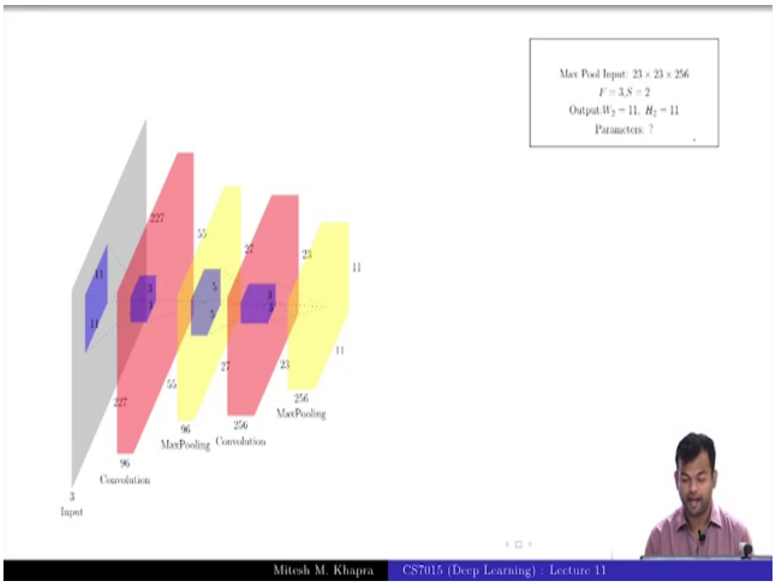
$S = 1, P = 0$

Output: $W_2 = 23$, $H_2 = 23$

Parameters: $(5 \times 5 \times 96) \times 256 = 0.6M$

So, these are the number of parameters, we had 0.6 million parameters in this layer. What is the next layer going to be? Pooling.

(Refer Slide Time: 07:20)



Max Pool Input: $23 \times 23 \times 256$

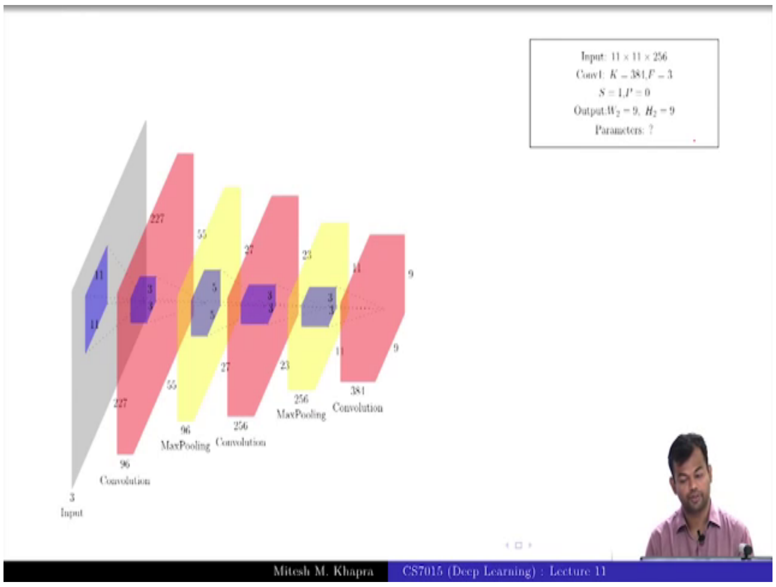
$F = 3, S = 2$

Output: $W_2 = 11$, $H_2 = 11$

Parameters: ?

So, you do a max pooling, again you do a 3 cross 3, you do a stride of 2, so your width in height is going to decrease, the depth does not change. Remember in max pooling, the depth does not change because the max pooling operation is for feature map, it is not across the depth fine. Then use a 3 cross 3 filter and 384 of those.

(Refer Slide Time: 07:40)



Input: $11 \times 11 \times 256$

Conv1: $K = 384, F = 3$

$S = 1, P = 0$

Output: $W_2 = 9$, $H_2 = 9$

Parameters: ?

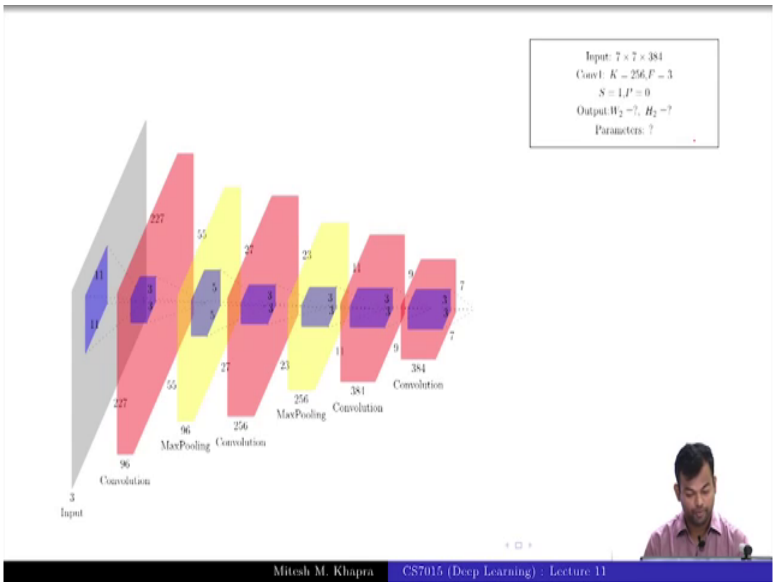
So, how many parameters would you have?

Student: (Refer Time: 07:43).

384 into 3 cross 3 into 256, the depth.

So, now you guys get it. So, I will not bore you anymore.

(Refer Slide Time: 07:51)



Input: $7 \times 7 \times 384$

Conv1: $K = 256, F = 3$

$S = 1, P = 0$

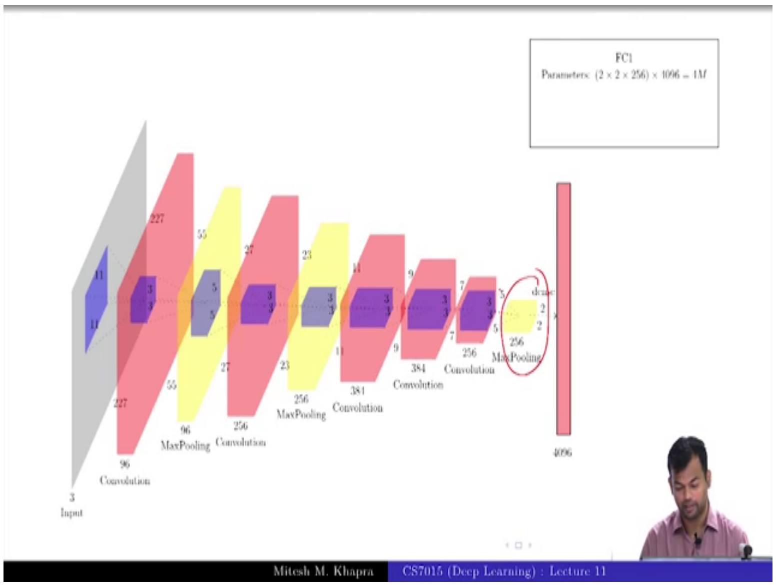
Output: $W_2 = ?$, $H_2 = ?$

Parameters: ?

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

And then you have a convolution operation again, which is a 384 convolutions each of size 3 cross 3 and so many parameters, followed by a convolution operation again, followed by a max pooling operation, then, followed by a fully connected layer.

(Refer Slide Time: 08:05)



FC1

Parameters: $(2 \times 2 \times 256) \times 4096 = 4M$

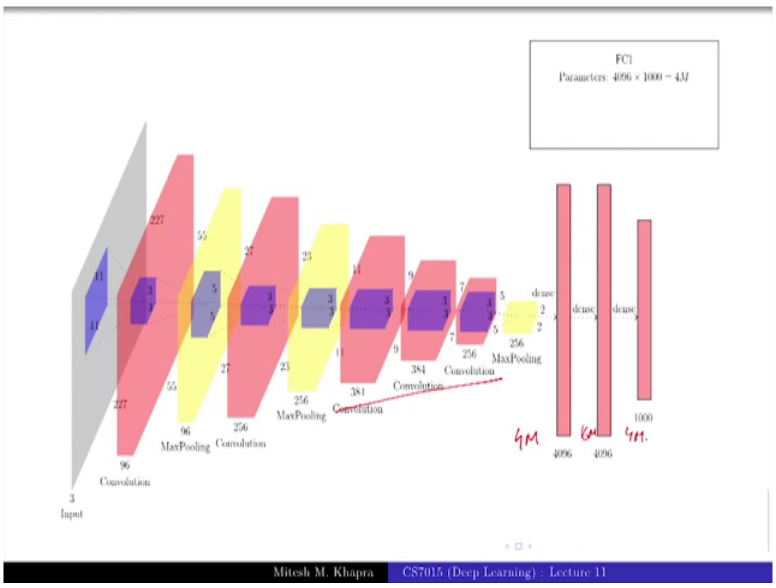
Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

So, what would I do to this 256 cross 2 cross 2, I will fatten it. So, I will get what dimensional output.

Student: 1024.

1024, 256 into 2 into 2, so this 1024 dimensional vector I am going to fully connect it to a 4096 dimensional vector. How many parameters? 4 million right, 4 into 10 is to 6 right so, roughly 4 million.

(Refer Slide Time: 08:30)


FC1

Parameters: $4096 \times 1000 - 4M$

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

Then you have another 4 million, another 4096 vector fully connected, how many parameters?

Student: 16 million.

16 million then, you have the 1000 classes that you are interested in right so again fully connected. So, you get the full architecture. Anyone has any questions? No one wants to know why this particular configuration among all the possible configurations, why not 10 layers, why not first 8 cross 8 filters, why not 9 cross 9 filters unfortunately, no one knows. [laughing]

Student: (Refer Time: 08:59)

So, this I mean see this what this is what would happen right. Now we get into something known as hyper parameter tuning right. So, what are the hyper parameters in this network, the kernel size is and the number of filters right. So, you would have tried a lot of these things, evaluated on the validation set, seen which one gives the best accuracy and then chosen right. So, that is probably what would have happened, but there is not enough insight into how this particular architecture came up.

Apart from some things right that 3 curves 3 neighborhood sounds reasonable. Initially, when you have the full image, you use larger filter sizes because, you want to capture a lot of things there but, once the image has shrunken you use smaller filter sizes. So, those are some rational decisions which look reasonable but, why these 3 convolutional filter layers back to back instead of convolution max pooling convolution max pooling and so on right.

So, the some of those things are not clear so, just in case you are wondering, do not wonder, this is just the architecture, this is known as modestly named as AlexNet. So, that is [laughing] yeah and so I said that this has 8 layers, but you clearly see more than 8 layers here. So, why did I say that has 8 layers, which are the layers we are not counting?

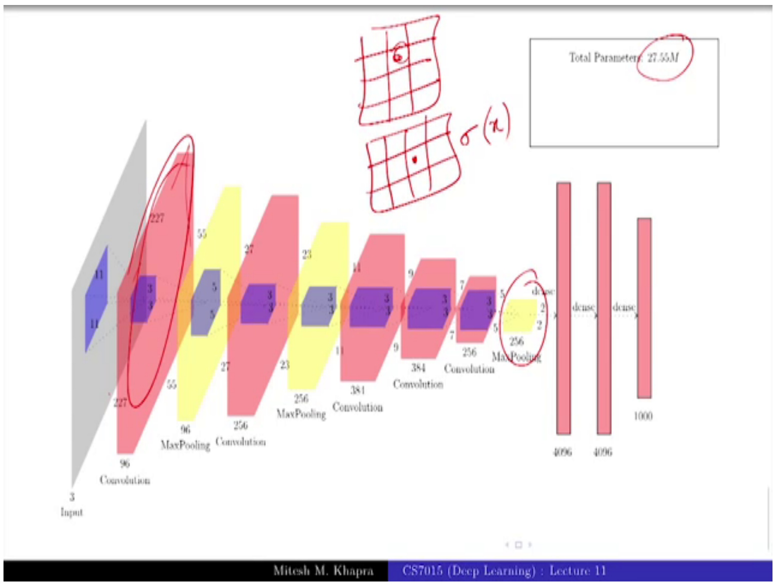
Student: (Refer Time: 10:10).

Why?

Student: (Refer Time: 10:12).

Because, they have no parameter right, so when you count the number of layers you only count those layers which have parameters. So, you have 5 convolutions and 3 fully connected layers.

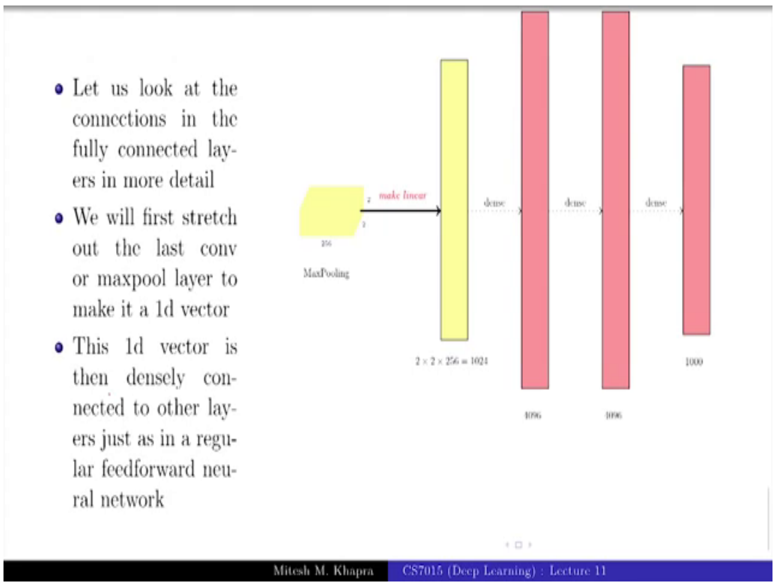
(Refer Slide Time: 10:21)



Then, so the total number of parameters in this network is 27.55 million parameters and at this point I will and; obviously, you notice that most of these parameters were there in the fully connected layer. So, you had 4 million here then, 16 million here and then again 4 million here right.

So, roughly 24 million of the 27 million parameters were there in the fully connected layer, you see that skew in the number of parameters.

(Refer Slide Time: 10:50)



- Let us look at the connections in the fully connected layers in more detail
- We will first stretch out the last conv or maxpool layer to make it a 1d vector
- This 1d vector is then densely connected to other layers just as in a regular feedforward neural network

make linear
MaxPooling
dense
dense
dense
$2 \times 2 \times 256 = 1024$
4096
4096
1000
Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11

And I will just look at the fully connected layer again. So, the last max pooling layer actually gave you a 256 cross 2 cross 2 output, you just flatten it to get a 1024 dimensional vector and then you connected fully to the 4096 vector right. So, that is what I mean by a fully connected layer. Why do you move max fully?

So, the reason for that is basically, to shrink the size of the image right because, after that if, if you keep working with this size right then, the number of parameters is going to really blow up, a by using a larger stripe yeah both of them are feasible right. So, now, see from here remember that we had the original image sizes 227 cross 227 and by the end we were just left with 2 cross 2.

And then adding a fully connected layer on that makes sense right. If I had not done this shrinkage throughout either by increasing the stride of the convolution layer or by doing max pooling right then, you would have left with something of the order of 200 cross 200 here and then, you have to do a fully connected on top of that is just infeasible right. It just throws away all the hard work that you have done by doing weight sharing and sparse connectivity right. So, that is not feasible.

There are also papers with say, which I think it is titled fully convolutional neural network which does not have any max pooling layers and they show that that also works fine. In fact, when we see VGG in net, we will see that it has back to back convolution layers and very few max fully layer right. So, these are all things which people have trained.

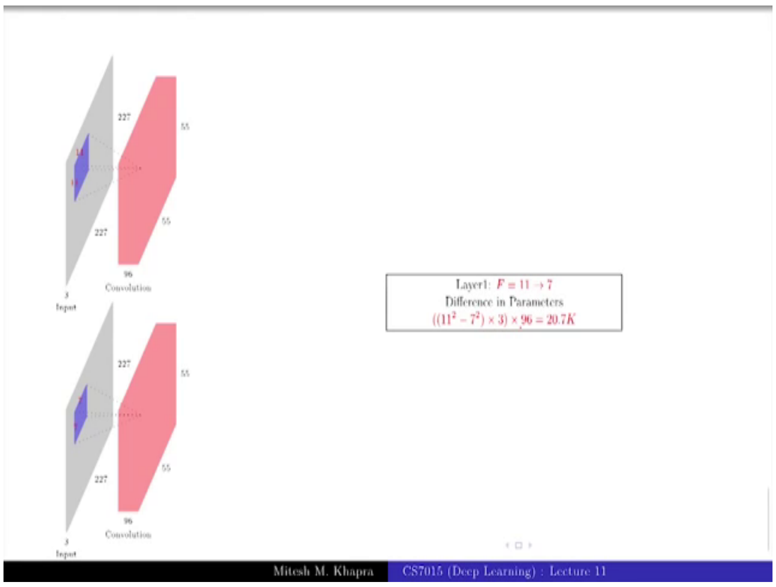
Not so many years, 2 years the challenge came out in 2010 and in 2012 this was used right. So, it is like not really a large gap right. And if, you read the original paper they had to do a lot of tricks to actually make this work, it was not as simple as I am showing it. Of course, now with all the stability which comes from these platforms, tensor flow pytorch, you can probably just go and implement this as it is and you should be able to reproduce the results but six years back that was not the case there was a lot of hard work involved in getting this too work. And they this was also the paper, which introduced the reLU non-linearity in the context of convolutional neural networks right. So, they had to change from sigmoid or tan edge to reLU.

A lot of these small small things which they had done And at that time it is also not possible with the existing hardware to train this on the given GPUs that you had at that time. So, they had to do some splitting across GPUs and so on. So, it was not as simple as it is today with all the hardware, as well as API developments or platform developments around this right. So, probably that is why it took 2 years to yeah sure.

So, each of these things, so after you do the convolution operation, you pass it through the reLU non-linearity. So, what does that mean is that, the convolution operation gave you a feature map, every entry here was just a weighted average of the neighbors right, you take this entry or rather you take this feature map and create a new feature map where every entry here is the sigmoid of every entry here. Do you get that or not sorry sigmoid, some non-linearity and they use the reLU has the non-linearity. So, you do get everyone, gets this. So, all the convolution layers are followed by a reLU non-linearity layer.

So, you get this volume, pass it through the reLU and get a new volume, but, I have just shown that as a single operation. It is before pulling so this was the fully connected layer. So, now, we look at the next architecture which is ZFNet.

(Refer Slide Time: 14:05)



Layer1: $F = 11 \rightarrow 7$
Difference in Parameters
$((11^2 - 7^2) \times 3) \times 96 = 20.7K$

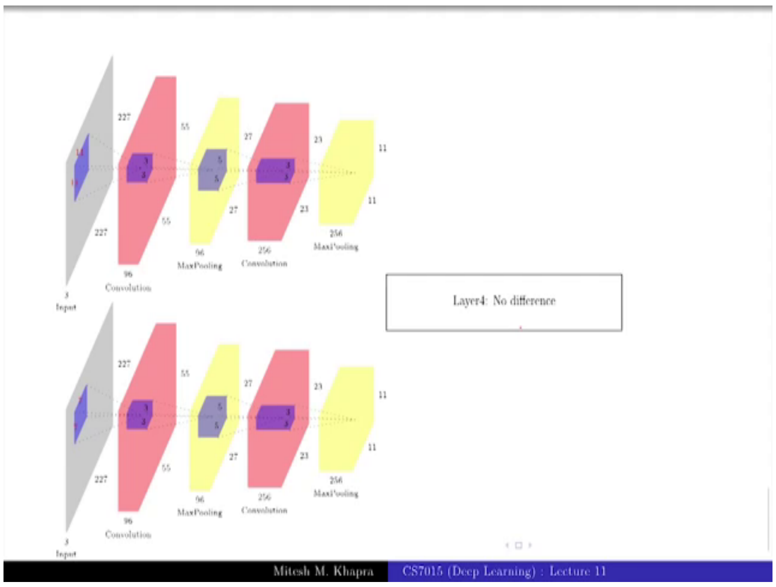
Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 11

Now, I am going to compare ZFNet with AlexNet. So, on the top you will see AlexNet, on the bottom you will see ZFNet ok. So, again the input was the same, 227 cross 227 cross 3. Now instead of 11 cross 11 filters ZFNet decided to use 7 cross 7 filters and their rationale was that you do not need such large neighborhoods, you do not need as small as 3 cross 3, but probably you need at least as much as 7 cross 7, you do not need 11 cross

11. So, that is the first change that they did and that would also result in some parameter pruning right.

Because, the number of parameters now would be 7 cross 7 into 3. So, the difference in the number of parameters at this layer for ZFNet which is at the bottom and AlexNet which is at the top, would be this. How many of you get this? So, that is in the difference in the number of parameters. So, now, the output volume still remains the same, it is 55 cross 55 crossed 96.

(Refer Slide Time: 15:03)

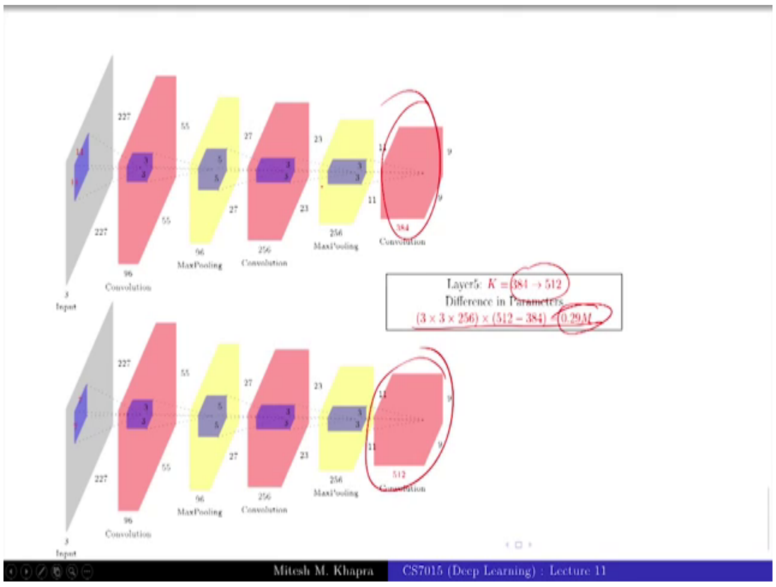


Layer4: No difference

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

Then again they had the same max pooling operation, this layer there was no difference between ZFNet and AlexNet and then after that you had again layer 3 which was exactly the same as AlexNet.

(Refer Slide Time: 15:20)



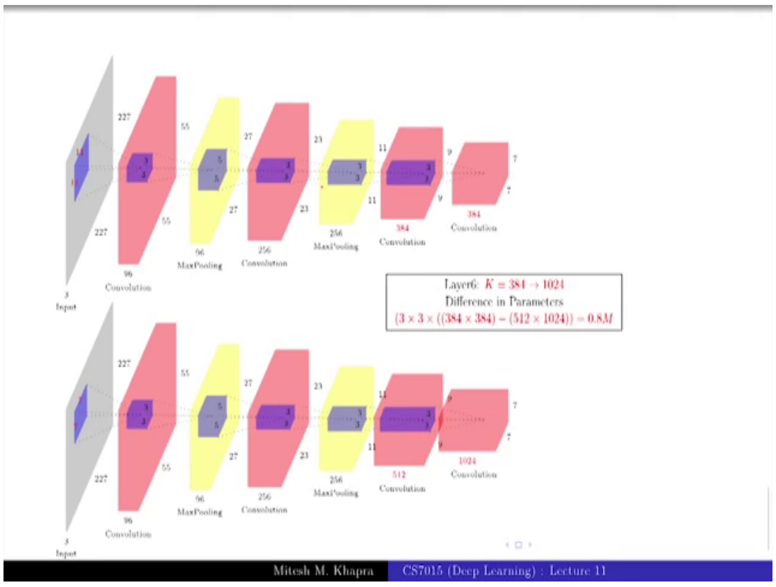
Layer5: $K = 384 \rightarrow 512$
Difference in Parameters
$(3 \times 3 \times 256) \times (512 - 384) = 0.29M$

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

Then layer 4 again the same as ZFNet, afterwards layer 5 instead of 384 filters, they decided to use 5 12 filters, the rest of the thing remains the same; that means, the size or the spatial extent of the filter remains the same. So, that again results in some difference in the parameters. So, that is the number of parameters that got added in ZFNet as opposed to Alex net.

And of course, the sorry, the bottom one is a ZFNet yeah that is correct sorry So, in zf net you had 512 filters as opposed to 384 filters in Alex Net, is it fine.

(Refer Slide Time: 15:52)

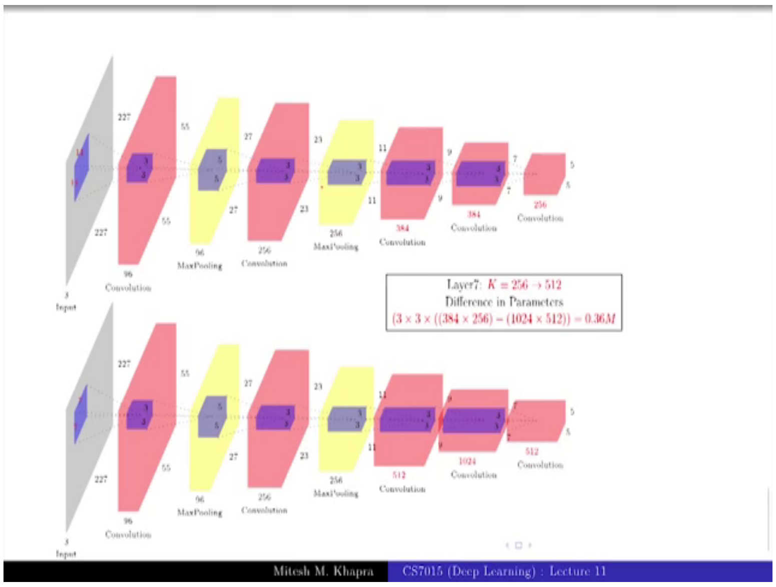


Layer6: $K = 384 \rightarrow 1024$
Difference in Parameters
$(3 \times 3 \times ((384 \times 384) - (512 \times 1024)) = 0.8M$

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

And then, the next layer again instead of 384 filters, they had 1024 filters.

(Refer Slide Time: 15:57)



Layer7: $K = 256 \rightarrow 512$

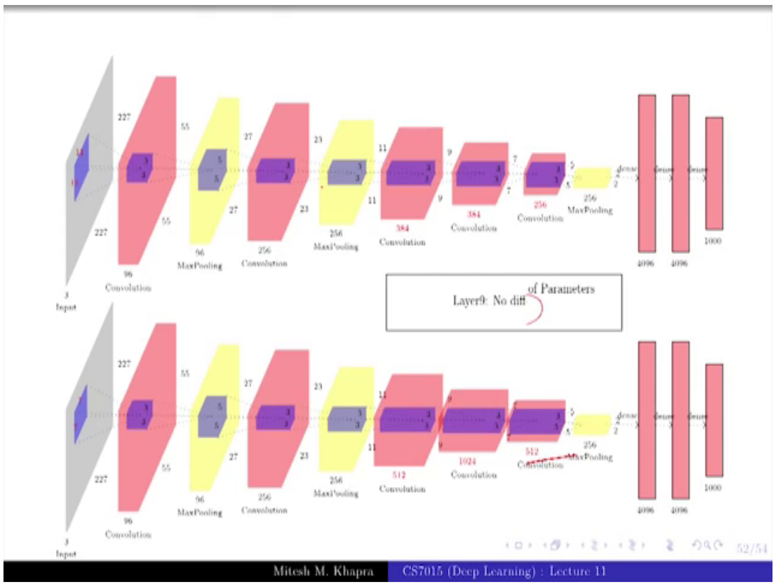
Difference in Parameters

$(3 \times 3 \times ((384 \times 256) - (1024 \times 512)) = 0.36M$

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

Then, again instead of 256, they had 512 filters and then, a max pooling layer then the same dense fully connected layers.

(Refer Slide Time: 16:02)

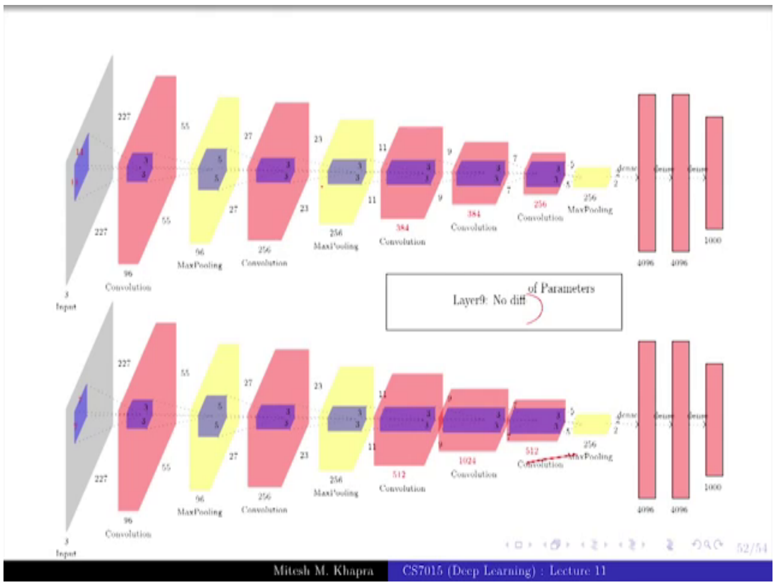


Layer9: No diff of Parameters

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

So, everyone gets this, this is the difference between the two architectures and this led to that difference in the error of around 3 to 4 percent is that we have seen earlier.

(Refer Slide Time: 16:15)

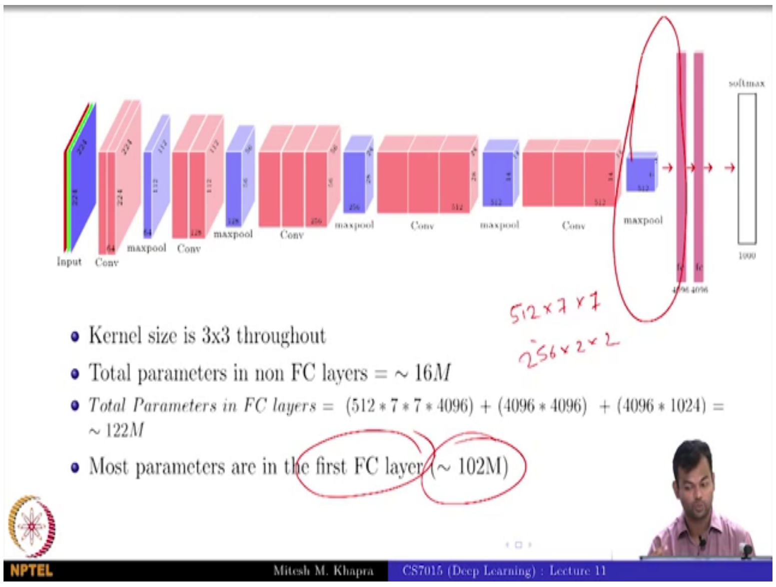


Layer9: No diff of Parameters

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

So, difference the total number of parameters was 1.45 million and of course, ZFNet had more parameters because, is that it has these more filters in the deeper layers ok. So, we go to the last point which is may be more in that VGG net.

(Refer Slide Time: 16:26)



- Kernel size is 3x3 throughout
- Total parameters in non FC layers = $\sim 16M$
- *Total Parameters in FC layers* = $(512 * 7 * 7 * 4096) + (4096 * 4096) + (4096 * 1024) = \sim 122M$
- Most parameters are in the first FC layer ($\sim$ 102M)

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

So, again in the case of VGG Net, the input was so I just want to I will not see it (Refer Time: 16:41) in. So, the input was again the same, it was RGB cross 227 cross 227.

So, this is what the VGG architecture looks like. They have so in VGG network throughout ok, wait. So, how many layers this ZFNet have, 8. So, you only count the pink boxes because, the those has ones which have two parameters. Now, VGG net has slightly more number of layers but, in all the convolution layers they use 3 cross 3 filters right, from the beginning they use 3 cross 3 filters So, you have the first convolution layer then, another convolution layer, another convolution, another max pooling layer followed by 2 convolution layers, then a max pooling layer, followed by 3 convolution layers, max pooling just keep adding box is writing just because, you can and then, you have the fully connected layers.

So, again there is not much intuition for why 16, in fact, later on someone came if this is the VGG 16 architecture because, it says 16 layers. Later on some of someone came up with the VGG 19 architecture, which has 19 layers right. So, a lot of this is data center even right, so you try your best to get the best possible accuracy on the ImageNet data and that is the architecture you came up with right.

But as long as how many of few feel comfortable with what is happening right and I mean, when I say comfortable, I mean that, you really understand the gory details of what is happening at each layer in terms of input volumes, output volumes, number of parameters, how are you going to train this network end to end. Can you see how are you going to train this. So, you will get some loss here that is going to propagate all the way back to the first layer right and this propagation is going to happen over some sparse connections that fine. Now, this is one very important point that I have skipped and which none of you is questioning. Is everything that is happening here differentiable?

Student: (Refer Time: 18:38).

What happens to max pooling? Is max pooling or differentiable operation? So, I am going to ask you this, how are you just note this down, how are you going to back propagate to the max pooling layer because, you need to see whether the max pooling layer is actually a differentiable layer or not . So, here I just some statistics about VGG net; everyone is writing that down [laughing] this perhaps, means I will not ask it

The kernel size is 3 cross 3 throughout; the total number of parameters in non fully connected layers is 16 million. The total number of parameters in fully connected layers is 122 million. So, you see that this fully connected layer is really a problem it like really hogs all the lime that it has the maximum number of parameters there right. And so and the most number of parameters are there in the first fully connected layer because, you have this 512 cross 7 cross 7, you remember then Alex Net and ZFNet, the last layer was 256 cross 2 cross 2, which has definitely more manageable than this layer which has grown almost 8 times in size, but not even 8 actually 4 into 4 into 2 right, 16 32 times in size right.

So, that is really blown up the number of parameters in the first fully connected layer. So, you just imagine the I mean, you have such a deep layer and then you realize that all the main number of parameters are there in this one particular layer, everything else is much much fewer parameters or orders of parameters less number of parameter is less then, this one fully connected layer