# Deep Learning for Visual Computing

**Prof. Debdoot Sheet**
**Department of Electrical Engineering**
**Indian Institute of Technology, Kharagpur**

## Lecture - 36
## GoogLeNet – Going Very Deep with Convolutions

So, welcome and while we have been going down to a quite few weeks and the different topics which we have done. And obviously you guys have got introduced on to deep convolutional networks, but then how deep is one of the predominant questions which we were asking us quite in the early week so over there and how deep can we keep on going down. And when do you stop when or like other some ways to go even deeper beyond it.

So, today's topic is more of what is in the community in fact the first paper which came out was going down as going deeper with convolutional Google networks and that was from Google research. And we are going to start with that one, but then the point when this particular network came down from Google was a quite interesting turn of events where happing around that point of time.

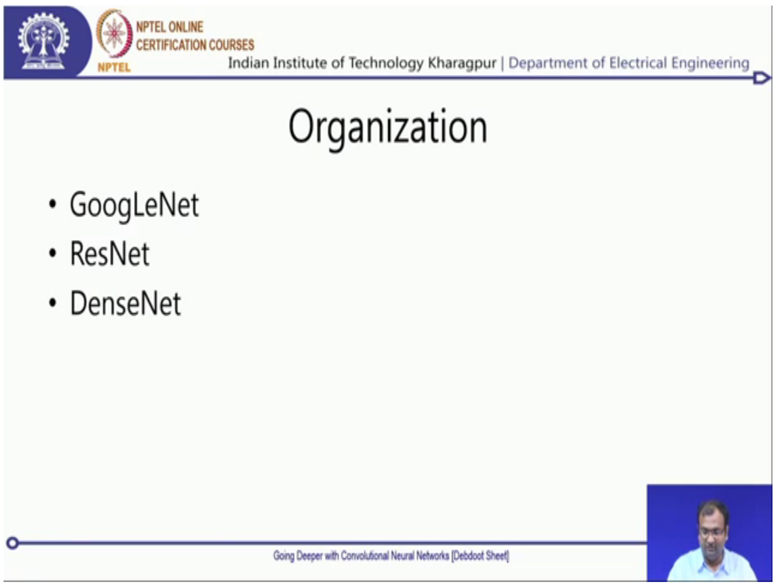
And one of the interesting events was that VGG NET was at that point of time quite ruling down. So, in the last lecture, we have covered down on VGG NETs, and then you had your labs and we also did some sort of revision over the compute complexity at the space required in these kind of deep networks, which included Alex net. Then it started more of the (Refer Time: 01:21) and then you went out over to Alex net and then on to VGG NET.

Now, today we are going to just do a brief revision of what we have done with the VGG NET to give you an idea of like where is the major challenge which you would be facing not in terms of how to handle down the data or how big and how complex is your model of what will be the total number of operations which takes place which as of now we have not done in much of detail. But these are kind of networks where we would be doing even on those.

Now, based on that if we are going deeper obviously on one side effect is your compute seems to increase but then is it actually increasing the first question, which we are going to look down through this lecture and the subsequent lecture as well. And the next question is that what do you gain by going deeper or is there a challenge which you would imminently face while going deeper and is there way out.

So, GoogLeNet came across three different versions there was version one, version two, version three. And what we are going to study today is actually v 3. So, v 3 or the third version of it is also what is called as inception v 3 because of particular module which is called as inception module. And these are the particular ones which have actually shown down that you can have like really deep networks, and with the clever idea of how to get down your gradients working out really good you can actually do a back propagation and that would stabilize the whole network much easily. So, let us get into that.

(Refer Slide Time: 02:49)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## Organization

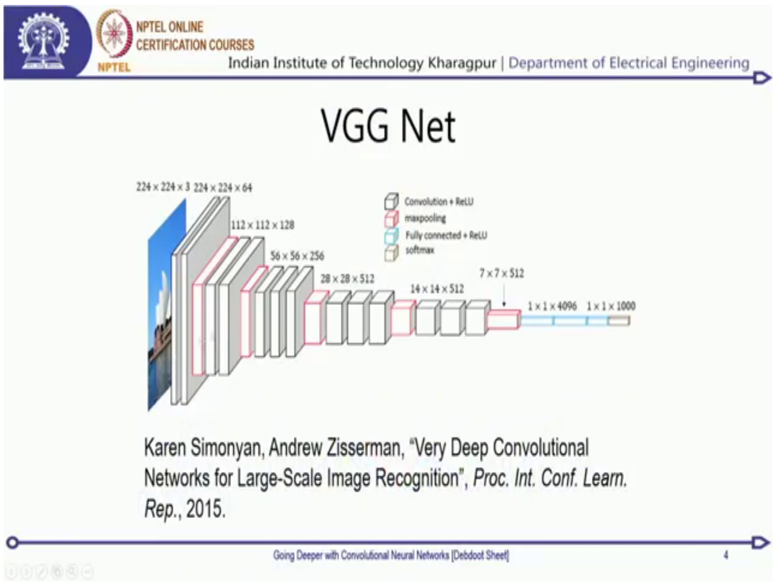
- GoogLeNet
- ResNet
- DenseNet

Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

The organization for this lecture and there subsequent to that one so in between we will in the next lec lab in the next lecture, we will be having a hands on session. On the lab model for GoogLeNet and then subsequent to that I would be covering two more one of them is called as ResNet and the other is called as DenseNet. And these if you go through a generation of this one then what we are speaking down is quite recent these are just last half a decade.

In fact, DenseNet is 2017 itself the paper was published in 2017 CVPR. ResNet is from 2016 CVPR. So, you do get like where we are currently standing now. So, let us start with going deeper with convolutional neural networks and this is the title of the paper which we are going to follow down and that is also called as GoogLeNet inception v 3 for common notions.

(Refer Slide Time: 03:35)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## VGG Net

Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *Proc. Int. Conf. Learn. Rep.*, 2015.

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 4

Now, let us let us have one way basic revision over VGG. Now, one thing which we are looking down when we are doing at VGG NET was that we were not taking down more of like uniform kernels and the point was that what you had is you had subsequent layers of convolution of with 3 cross 3. And one of the major reasons of doing multiple bankings, so somewhere in the first time we had just 2 convolutions then you down sample, then you have 2 convolutions then non sample and then you have three subsequent convolution. And one of the reasons why you had these kind of stacking down going down is that, so that you have a much wider receptor field over they are taken down.

So, instead of having one 5 cross 5 specially spend out convolution, we can think of looking into the same kind of an area by taking down multiple 3 cross 3 convolution. So, you can have two subsequent layers of 3 cross 3 convolutions and that would be able to discover features quite similar to what you would be discovering in a 5 cross 5 convolution.

However, the down side is that what you are essentially doing is over here if you have a 3 cross 3 convolution, and then a 3 cross 3 convolution over here, the features derived that the second layer are what are all dependent on the features derived over here. And they will all get modified, there is no way that the features over here can be carried forward subsequently over there. We do we do not have that option of doing it in a VGG NET.

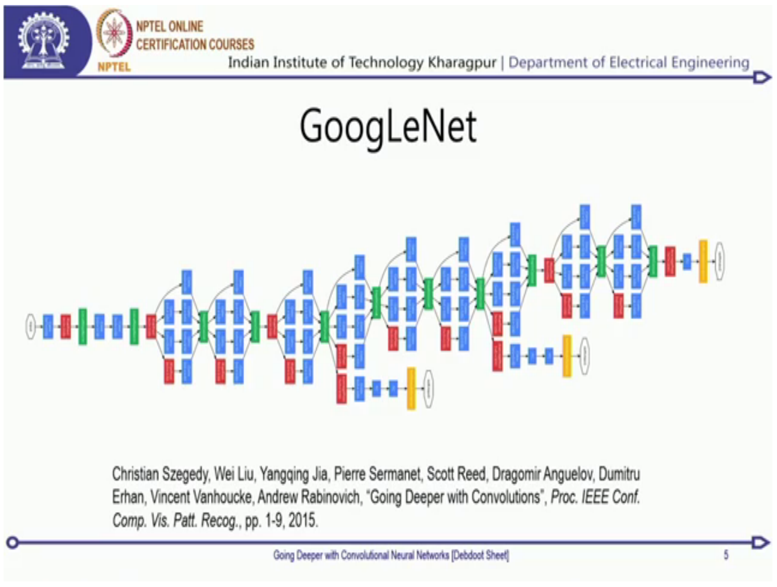
So, if you have subsequent convolutional layers, so it means that the output of the previous layer is definitely going to get modified. And you do not have any way of carrying down, so that essentially would mean that if I want this image in its row format also being carried down over here down till so may be at the last point of it. It just the intensity of the image is what it remains as one single pixel over here. I do not have mechanism of taking it down.

But then from your current understanding of traditional methods of learning ways computer vision is where you have seen that they can be very simple features, simple features as in what is the intensity of the image even that can be used as a classification matrix. What is the average intensity, whatever is the minimum intensity, what is the maximum intensity, then you can have some of these features like just take down in a block wise level what is the average intensity over there.

So, in fact, low past filtered versions of the image can also be used as some sort of features at the final classification. But here in these kind of direct convolutional networks, you do not have such way. And then all the features which you are going to describe they are within the same receptor field in terms of like your spatial span is always three cross three.

There is nothing like I can have certain features in 3 cross 3; I can have another group of features which are exited signify 5 cross 5, another group of features in 7 cross 7, and another group of features in 11 cross 11 that is not present over there. However, from our understanding of classical learning ways vision we do know there are feature descriptors which we use which actually are of a different receptor field. Now, this is where inception v 3 comes to play.

(Refer Slide Time: 06:33)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## GoogLeNet

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions", *Proc. IEEE Conf. Comp. Vis. Patt. Recog.*, pp. 1-9, 2015.

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 5

So, what we have is a model which looks quite like this that looks really scary spider web to a lot of people. Now, not to get you guys confuse, so this was 2015. So, what we are going to do now is to the best architecture from 2015, next the best architecture from 2016 next the best architecture from 2017. So, three award winning architectures in just one weeks lecture is what we give you a touch. So, it looks really hard, but we need to get started with that.

So, where it starts is over here. You see this point is my start point and you can follow these arrows ok. It is not hard to get these whole diagram because what you can do is you can go just Google down for this particular paper it is it is in it is on the proceeding of the CVPR.

So, you have on open access dot cum c b foundation computer vision foundation is where you would be finding all of these directly available, no need to like it is not necessary that you need to have a subscription over (Refer Time: 07:32) or any of those things in order to get access to get this paper. It is it is on computer vision foundation and CBF makes it as a point to just open source knowledge and give it out for free. So, you can get this one.

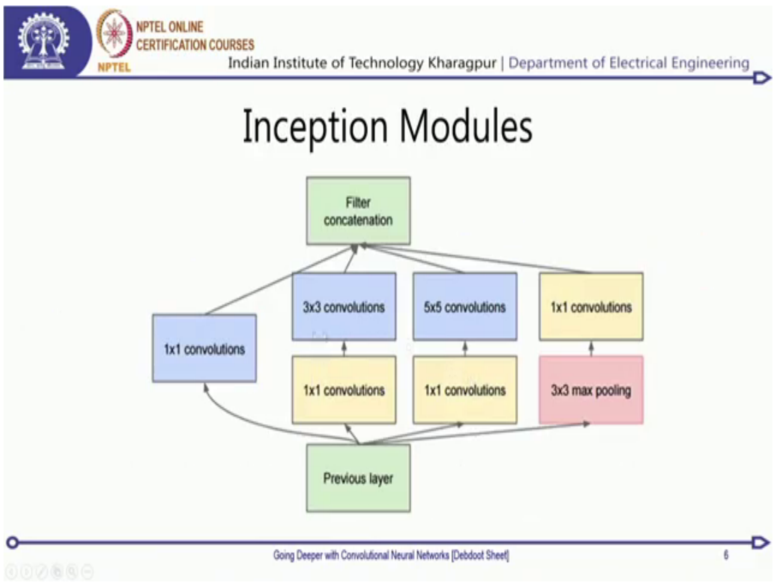
And what I would suggest is while you are watching out the video the do definitely make a point to really zoom into it and go through it ok. So, let us do some interesting points over there. Now, it is really hard to do. So, now you get your input over here and then what you have is some sort of convolution. This convolutions are seven cross seven convolutions. Then you have a max pooling over here and after that you have a local response normalization. So, LRN is something which you have seen down in your VGG for your local response normalization. And the idea of LRN was more of like if I have a window some sort of moving window along my number of channels over there for a given pixel.

And the moving the response is somewhat bound in a way that for any given pixel within that range over there if I sum up all the values and that is what is normalize one to one. So, this gives me bound a very tight bound and also makes all my responses in the range of 0 to 1, so that the cumulative summation over there in the moving window is always one. So, that is what was there in LRN.

Now, ones your local response normalization comes you again have convolutions over there the two layers of convolutions basically. So, one is a 1 cross 1 convolution, then you have another 3 cross 3 convolution, and then LRN local response normalization. You go through a max pooling and then there is a split which happens over here. Now, this is interesting because. So, in the subsequent slide, I will be showing you where this goes down. So, this is interesting from a point because what you would see is that each of these splits are which are of a different convo different receptor field. So, the kernels and no more just of fix 3 cross 3 and this is what I was telling you.

So, if I want a way of actually translating all my input information onto this place I do not have any other way, but then now I have this different varying kernel sizes on my receptor field at the same depth. So, it is not naturally that I will have to casket through depths in order to get down a wider receptor field, but then I can get wider receptor field as well as narrow receptor field in the same depth itself and that is where this keeps on going.

(Refer Slide Time: 09:41)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## Inception Modules

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 6

So, let us get into this slide which would make it much more intuitive for you. So, this is an input which comes down from the previous layer. Now, if you go down here this block is the block which you have over here ok. So, the output from max pooling is what goes down as a input from the previous layer good. So, I have my input coming down over here. Now, this input is send down through these two 1 cross 1 convolutions ok, it is directly sent out through a 3 cross 3 max pooling ok.

Now, subsequent to this 1 cross 1 convolutions, you have a 3 cross 3 convolution. And on this 1 cross 1 convolution you have a 5 cross 5 convolution. On the other side, you also put it down through a 1 cross 1 convolution and then after this subsequent max pooling you have 1 cross 1 convolution and everything goes down and does a concatenation. In a sense that whatever are the number of channels which comes out over here everything is just contracted and fixed up onto the total number of channels.

So, now technically speaking, so if I have some n number of channels which comes over here on from the previous layer say this is the 16 channels which comes over here. Now, when I do a 1 cross 1 convolution and feed it over here, I will have sixteen set channels over there. Now, essentially what that does is a cross my channels across any pixel on all my channels it is going to run a convolution, and find out across the channels for a given pixel what is the total aggregate response coming down and multiple of such aggregate response.

So, now I can have 16 inputs over here, and 16 such 1 cross 1 convolution kernels. So, this will give me 16 outputs ok. Now, from here if I do a 16 1 cross 1 convolutions that is also something of the same, but then now here I introduce this 3 cross 3 convolutions which are going to mix up the receptor fields over there. Now, technically you could say that well we could have pretty much put down only this one over here yes that is possible the only point is that you will have these extra buses coming down like from here you would be connecting down onto here, then from here you would be connecting down to here.

And the other point which will happen quite for sure is that what we are learning essentially in these 1 cross 1 convolutions is something are needed in terms of kernels of 1 cross 1 in order to learn down effective features in 5 cross 5. And so what that necessarily means is this 1 cross 1 convolution block is learning something which is very different from this 1 cross 1 convolution block and that is also learning something which is very different from this 1 cross 1 convolution block. Because this is one arm of discovery of features which works in tandem together; this is another arm which works in tandem together and that is why we do not have a sheared connections between this 1cross 1 convolution blocks ok.

Now, the next one is you push it through a 3 cross 3 max pooling. And now once you have it push through a 3 cross 3 max pooling, you go down to a 1 cross 1 convolution block again over here and then you concave it. So, if there are 16 channels from each of them, then you know that what connecting 4 such 16 channels you are going to get a 64 channel output from here and that is typically what a inception module does.

So, let us get back over here and then you would see what comes down. So, you have your inputs coming down and this is your inception module which goes down. Your depth concatenation now the output of this depth concatenate is what is again fed down to your inception module, which is quite in the similar case. Now, once this is done then you have the first level of max pooling. Now, this is ones where you have a size reduction by 2, this is again where you have a size reduction by 2. This is the next part where you have a size reduction by 2. So, nowhere in between you have reduction in the spatial size over there.

This is quite similar if you look down to the philosophy of a VGG NET. So, you had two blocks of convolutions and then you had a max pooling layer to reduce the size over there; instead of two blocks of convolution what we are doing is we are putting down parallel pipes of multiple convolutions which have a different receptor field. So, this is where GoogLeNet gets inspired from VGG itself.

Instead of putting down a battery of convolutions over here, so everything is of the same receptor field in this battery. Here we just have a parallel multiple pipe lines and some of them are three cross three, some of them are 5 cross 5, and some of them are 1 cross 1 this is what you changed down in your inception module. Other than that it looks quite similar to VGG NET to say in that form, you have your max pooling then you have your convolutions, then you have you do concatenation then you again do max poo this inception module over here.

Interestingly, you see another tap coming out over here. So, there is a tap output which comes down over here then you do some sort of average pooling instead of a max pooling after this you have a 1 cross 1 convolution, then you are fully connected layers and there is activation. So, this is sigmoid activation function and then you have your soft max output (Refer Time: 14:32).

Now, this node over here looks as if like a decision node, something pretty much similar at the end of it. Now, that has q d s point of what where it helps it out ok. So, let us let us keep this arm as of now this is also called as an auxiliary arm. So, this is what will come down in the training and is quite interesting ok. So, as of now let us let us take down over here. So, you got your second so this was your first inception block, then you have second inception block, then you have third inception block and you have forth inception block then you have fifth inception block.

Now, over here when you are again coming down you again have another auxiliary arm. Now, finally, when it comes down over here you get another two inception blocks, then you have max pooling, then a fully connected layer, and then this decision layer. Now, typically for if we are comparing it down with any of our other network. So, say for VGG what happens you have an input coming down over here, and then it goes all the way out and then here where your output comes down. So, and this is the 1 cross 1 cross 1000 or just a thousand plus a classification problem for your image net ok.

Now, here what you are doing is you put an input you will get an output over here, you will get an output over here, you will also get an output over here. And all of these are thousand cross one outputs which are waiting down here. Now, the question is if I can tap an output over here, why do I wait, and then go down all the depth and come over here.

Now, that is an interesting point, because that the whole reason for doing all of this is this is an output which looks at much lower depth of features ok. There is an output which goes more deeper a bit more depth and from our assumption that the deeper you go the better you learn, this should be able to give us some sort of better learning and better approximation.

Similarly, should this should also be giving because this is even deeper. Now, keep in mind one thing if that is the case then we get down three outputs, we are getting down three errors as well. If we because we are going to compare with each of them the ground flow. Now, when there is a back propagation which happens, so these layers over here all of these learn able layers which are there in blue. These are the ones which are learning it down your green block which is the depth concave that is no learn able parameter, your red blocks which are more of max pooling or average pooling, there are also no learn able parameter.

So, from here and at this soft max activation over here the yellow one does not have any learnable parameter the only learnable parameters are present down within your blue blocks over here. Now, from here when its back propagating till all the way up to here is where the gradient from this final decision is coming down. However, here you would see that this is this extra error and the gradient, which is also back flowing into it.

So, that would mean that over here we are going to get a part of this gradient from the terminal node and then also the part of this gradient from the intermediate auxiliary node or what is on the second auxiliary node as well. Now, at any given point of time since you know from your back propagation rules that it keeps on getting constantly multiplied by the activations of each of these layers; And the gradient of the operations in each of these layers.

Now, since all of these values activations are also limited down in a zero to one range. So, subsequently if you keep on multiplying numbers which are range restricted in 0 to 1, you would see that the value keeps on decreasing. Now, over here the challenge would be that at a point of time after traversing so much of depth the value. So, let us look into this. This is one layer; this is second layer; third, fourth, fifth, sixth, seventh. So, if everything is at over here if my error is at 0.1 10 power of minus 1. So, by the time it traverses seven size depth and everywhere if it is limited by a factor of 10 power of minus 1 so that means, your activations are in 10 power of minus 1.

So, by the time it comes down over here, it is already in 10 power of minus 7. Now, your weights are in the range of 0 to 1 and your gradient which is coming down is in 10 power minus 7. For that to modify this one, the dynamic range is really low; obviously, you can do the other way round which is let there be a different learning rate for each of these layers that that should be a plausible solution I mean I can sit down and sit down different learning rates over there. Instead of that this definitely a very tricky and complicated solution because you do not know what is the dynamic range at the start of it I mean anything can have a dynamic range.

Now, instead of that if you look into this kind of error, so this is where I get down something in the range of 10 power of minus 7, but this gradient which comes down this as 1, 2, 3, so this is already in 10 power of minus 3 over here. Now, 10 of minus 3 at any point of time is 10,000 times more powerful than the power of minus 7 and that would bring down much better improvements into the performance and that is the reason that is the whole job this one place down.

However, keep in mind that we are looking down only at the gradient of the error and not at the absolute error. So, while absolute error over here will be higher than the absolute error at the terminal node. So, at any given point of time my intermediate auxiliary node is going to give me a higher error rate than the terminal node.

Still what I would get down is even if it has higher value, but if its saturates and saddles out over there then the gradient is going to become 0 despite a high error the gradient becoming 0 means that the update over there is also 0. So, my weights are not changing in any way. So, it means technically that if nothing has to be updated which means there is no change in gradient, despite having a higher error, I will still be able to modify and on or not modify so that is not an issue which comes down.

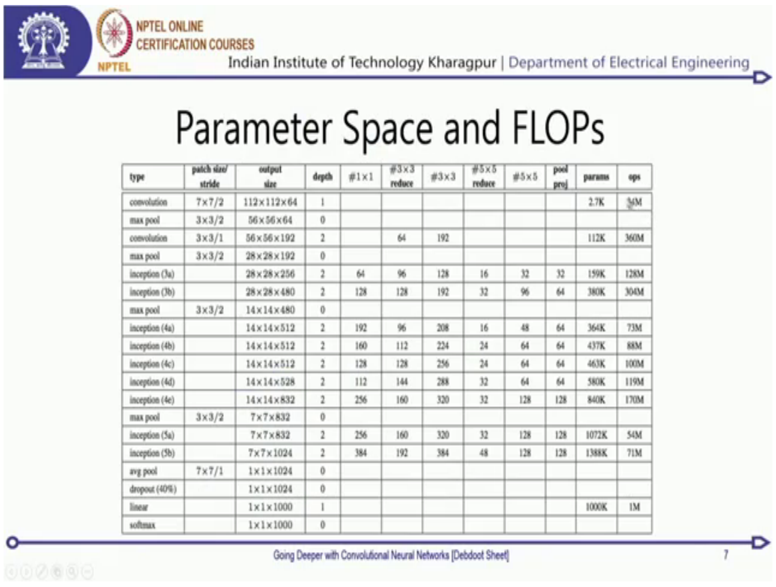
Now as we keep on going down over here, here also we will face down same problem which is actually called on a as a vanishing gradient issue. Now, this auxiliary arm actually plays a role. Now, where these guys had cleverly played a significant impact was while designing this kind of a deep network, you need to keep on tracking across each of these layers what is the order of change which is coming down in terms of my weights.

Now, wherever you see a thing which is your gradient is just going down, so these kind of a auxiliary decision node over there will actually help you boost your gradients. So, this is one of the reasons why this kind of deeper networks are really doing out good and in fact GoogLeNets one major achievement is having these kind of auxiliary arms.

Now, what one thing which you need to keep in your mind is that whether you are doing a inferencing GoogLeNet versus you want to use a GoogLeNet for training there are two different models which get downloaded from you models on your pie torch, whenever you are using torch vision like. And there you need to keep something in mind that if you just do a inferencing GoogLeNet fully trained GoogLeNet which is used only for inferencing if that is being used you will not be getting down these two auxiliary arms you will have to recode it back over there. So, please keep in mind that you do not do a false wrong down over there.

So, while we get down to the coding in the next lecture we will be discussing more on details in it. And this is the whole aspect. So, there are two aspects over here what you need to keep in mind, one is that there are multiple kinds of receptor fields of different size reducing different sizes of kernels. And along with that you also have these auxiliary arms which do a gradient injection into the error over there, and this helps you in training down this is the kind of very deep networks ok.

(Refer Slide Time: 21:46)


NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## Parameter Space and FLOPs

| type | patch size/ stride | output size | depth | #1×1 | #3×3 reduce | #3×3 | #5×5 reduce | #5×5 | pool proj | params | ops |
|---|---|---|---|---|---|---|---|---|---|---|---|
| convolution | 7×7/2 | 112×112×64 | 1 | | | | | | | 2.7K | 34M |
| max pool | 3×3/2 | 56×56×64 | 0 | | | | | | | | |
| convolution | 3×3/1 | 56×56×192 | 2 | | 64 | 192 | | | | 112K | 360M |
| max pool | 3×3/2 | 28×28×192 | 0 | | | | | | | | |
| inception (3a) | | 28×28×256 | 2 | 64 | 96 | 128 | 16 | 32 | 32 | 159K | 128M |
| inception (3b) | | 28×28×480 | 2 | 128 | 128 | 192 | 32 | 96 | 64 | 380K | 304M |
| max pool | 3×3/2 | 14×14×480 | 0 | | | | | | | | |
| inception (4a) | | 14×14×512 | 2 | 192 | 96 | 208 | 16 | 48 | 64 | 364K | 73M |
| inception (4b) | | 14×14×512 | 2 | 160 | 112 | 224 | 24 | 64 | 64 | 437K | 88M |
| inception (4c) | | 14×14×512 | 2 | 128 | 128 | 256 | 24 | 64 | 64 | 463K | 100M |
| inception (4d) | | 14×14×528 | 2 | 112 | 144 | 288 | 32 | 64 | 64 | 580K | 119M |
| inception (4e) | | 14×14×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 840K | 170M |
| max pool | 3×3/2 | 7×7×832 | 0 | | | | | | | | |
| inception (5a) | | 7×7×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 1072K | 54M |
| inception (5b) | | 7×7×1024 | 2 | 384 | 192 | 384 | 48 | 128 | 128 | 1388K | 71M |
| avg pool | 7×7/1 | 1×1×1024 | 0 | | | | | | | | |
| dropout (40%) | | 1×1×1024 | 0 | | | | | | | | |
| linear | | 1×1×1000 | 1 | | | | | | | 1000K | 1M |
| softmax | | 1×1×1000 | 0 | | | | | | | | |

Going Deeper with Convolutional Neural Networks [Debdoot Sheet]
7

So, let us get down into some of these intricacy in terms of the parameters space and what is the total number of operations which gets down over there ok. So, if you look down on the first part of it the first convolution is a 7 cross 7 convolution with the stride of 2. So, if you get down get an input over here of 224 cross 224 cross 3 or, or 3 cross 224 cross 224 whatever convention you would like to do.

So, the spatial size is basically 224 pixels into 224 pixels and its 3 channels over there. Now, the first one which has pack size of or pack size or kernel size of 7 cross 7 and a stride of 2 and it also has a padding over there keep in mind. So, these padding is basically 3 cross 3 by 3 padding, otherwise you will not be getting down 112 cross 112 cross 64 over here ok.

Now, ones this comes out and then so this convolution is there what is at a learnable depth of 1 ok. Now, from there you get down so the total number of parameters which comes down over here is 2.7 thousand parameters. And the total number of operations is the number of multiplies, adds and accumulates which you would do over there in terms of flooring and numbers and everything is what comes down to 34 million operations over here. So, this is to look into the compute side of it.

Next you have a max pooling level over there. So, max pooling of a 3 cross 3 and a stride of 2 is what produces this output from the given input. Next you again have a convolution over there; this convolution is a 3 cross 3 convolution and with a stride of one. So, what this is going to do is a 3 cross 3 convolution with a stride of 1 and a padding of 1 and 1 obviously, over there this is what produces this output and you have 192 such channels coming down over there. So, these are the depth of 2 ok.

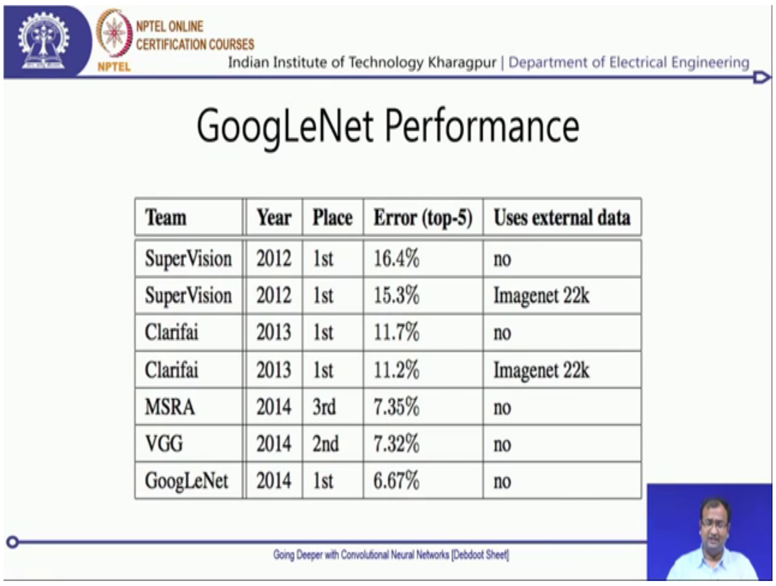
Now, over here in total the total number of parameters which you learn down; So, learnable parameters like; So, this is what we had discussed in the earlier one when looking deeper into deep networks and try to understand the compute complexity. So, the learnable parameters are just what recite within your weights over there and your max pooling or non-linear transformation operations they do not have any kind of learnable parameter. So, here we have 112000 learnable parameters and total number of operations it would do is 360 million ok. So, similarly you keep down keep on going for each of these inception blocks.

So, the next one is what is called as 3a and 3b. So, 3a and 3b are subsequently like this. So, this is at my first level, this is at my second level. Then I have three is this is my third level this is a, this is b ok. Then after this max pooling I get down my level four. So, this becomes 4a, 4b, 4c, 4d, 4e ok, so that is essentially what is what is going down over here. So, I have my 4a, 4b, 4c, 4d, 4e. And for each of them you have your kernel sizes which are given down as well as your output sizes also coming down. And similarly you keep on doing the same thing for your next layers.

And this now remember one thing like we do not give down explicit kernel sizes over here because your inception block is what is defined over here. So, there is no need because the these are pretty much fixed kernel sizes over there ok. Now, what is coming down is definitely what is the size of the output, and you can pretty much see this size of outputs over here.

So, this can be used for you to actually map down, what is the total number of channels in each of these kernels coming down over here and that combines together comes down to 364000 parameters and 73 billion operations going down. So, now finally, this is the total space. So, now, if you would like to look at the model complexity, you can actually sum up this parameter space and get down your size of the model which would be working it out ok.

(Refer Slide Time: 25:35)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

## GoogLeNet Performance

| Team | Year | Place | Error (top-5) | Uses external data |
|---|---|---|---|---|
| SuperVision | 2012 | 1st | 16.4% | no |
| SuperVision | 2012 | 1st | 15.3% | Imagenet 22k |
| Clarifai | 2013 | 1st | 11.7% | no |
| Clarifai | 2013 | 1st | 11.2% | Imagenet 22k |
| MSRA | 2014 | 3rd | 7.35% | no |
| VGG | 2014 | 2nd | 7.32% | no |
| GoogLeNet | 2014 | 1st | 6.67% | no |

Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

Now, let us get into what it performs like. Now, performance wise GoogLeNet in it, so it was published in 2015 on the conference, but then it did win the image net challenge in 2014 on the first place. And the top 5 percent error, where top 5 error is where out of 1000 categories which you have to predict if your answer matches down in the top 5 predictions over there you say it a true match.

Now, with that top 5 one the error percentage is the lowest till that point of time, the presiding one was obviously, VGG NET and its came down to 6.67 percent. So, this about training to go very deep with these because you can look down at GoogLeNet that is much deeper more number of convolutions is compared to the VGG. And obviously, it has now there is one thing interesting if you sum this one up, you would find that the total number of parameters are actually lesser than a VGG, whereas the number of operations is more than the VGG that is on the other side of it.

Now, that that is quite interesting that the total number of parameters goes down, but then this also reassures as back to the fact that the depth of the network actually does not have any role any direct role to the number of parameters. It is a complex function of the kernel sizes and the number of kernels used for channel in the depth and other than that it would not be coming out. So, that is where we come to an end with going deeper with convolutions. And stay tuned for the next lecture where we enter into much more very deep networks called as the ResNet and DenseNets.

Thank you.