Deep Learning for Visual Computing

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

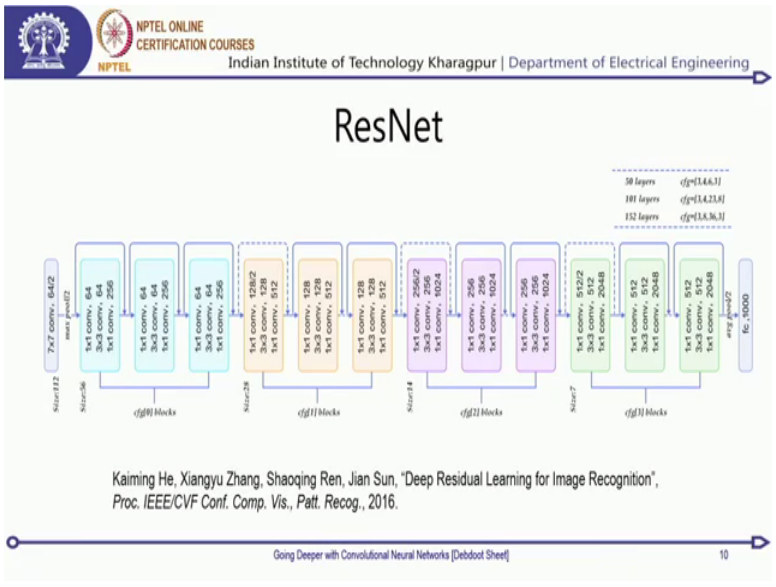
Lecture - 38
ResNet - Residual Connections within Very Deep Networks and DenseNet Densely Connected Networks

So, welcome. So, today we are going to discuss down with very and very deep convolutional neural network. So, in the last one like with VGG nets and then, with Google nets you have gone down from the version of the 16 layers, 19 layers and subsequently to 23 layers over there. Now, here is where we are speaking down about 100 of layers, ok?

Though we will start with very simple networks over there, but then, these are networks which are available for 100, 200 and 250 plus kind of layers and then so you can sort of imagine like between going from your VGG net to Google net, the amount of increase in operations which you had to face down though there was a slight decrease in the total parameter space, as well now here if you go down in hundreds of depth, then what will be the total number of parameters which you will have to learn as well as what will be the total compute complexity? Now, that is something to be really in curious about any Q C river work.

So, bringing back from what I had discussed in my earlier lecture on Google net is here, I would be covering down to specific one. So, Google net was the award winning C V P R paper in 2015. The next one was ResNet or the residual network which came down in 2016 and subsequent to that we have the densely connected networks dense nets in 2017. So, we are going to cover the ResNet and dense net over here.

(Refer Slide Time: 01:42)


NPTEL ONLINE CERTIFICATION COURSES

NPTEL

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

ResNet

50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 128/2 3x3 conv, 128 1x1 conv, 512
1x1 conv, 128 3x3 conv, 128 1x1 conv, 512
1x1 conv, 256/2 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 256 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 512/2 3x3 conv, 512 1x1 conv, 2048
1x1 conv, 512 3x3 conv, 512 1x1 conv, 2048
avg pool/2
fc, 1000
Size:112
Size:56
Size:28
Size:14
Size:7
cfg[0] blocks
cfg[1] blocks
cfg[2] blocks
cfg[3] blocks

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", *Proc. IEEE/CVF Conf. Comp. Vis., Patt. Recog.*, 2016.

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 10

Now residual network or ResNet as as its known down as. So, what it does essentially is that, it implements something which is called as a residual pipeline. Now, we have this really well written down diagram over here much more easy to recognize as compared to a Google net. Now, I never drew the diagram. So, you have to pardon me for that. It was a direct take away from the papers which we have been referring to and that is the reference which gave it. Now, here it is easier to look at.

It will come down to even deeper like what, what happens as we keep on going deeper and with them, now here what you essentially do is, there are two parts of it. You see this factor, which is called as a size, these basically X and Y, sizes. So, input whatever you give over here is the image net standard size; image which is 224 into 224 into 3. So, that is an RGB image of 2 to 4 cross 2 to 4 sized given as input over here. Now, we have convolution with 7 cross 7 kernels, there are 64 such kernels and the convolution is with the stride of 2 ok? So, this is what you get down over here. Now, since if you have padded convolutions and with the stride of 2.

So; obviously, the output size over here is 100 and 12 cross 100 and 12 ok? And the receptor field 7 cross 7 pretty big and this is what we are keeping it down pretty much standard from Google net. We are keeping it pretty much standard from Leanette itself and then, where you had the first major receptor field which was really wide at 7 cross 7 that goes down from the aspect that majority of these images only mentioned and for our simple images in day to day life, which we are using down.

This kind of a 7 cross 7 is what has a significant role and it can encapsulate most of the appearances of objects coming down over here, ok? Now, after this we have a max pooling by a factor of 2. Ok, now due to this max pooling, this is going to get reduced by another half. So, my 1 1 2 cross 1 1 2 sized image gets now reduced 1 – 2 5 56 cross 56, ok? It is no more an image; it is basically a volume whose spatial size is 56 cross 56 and the number of channels is 64.

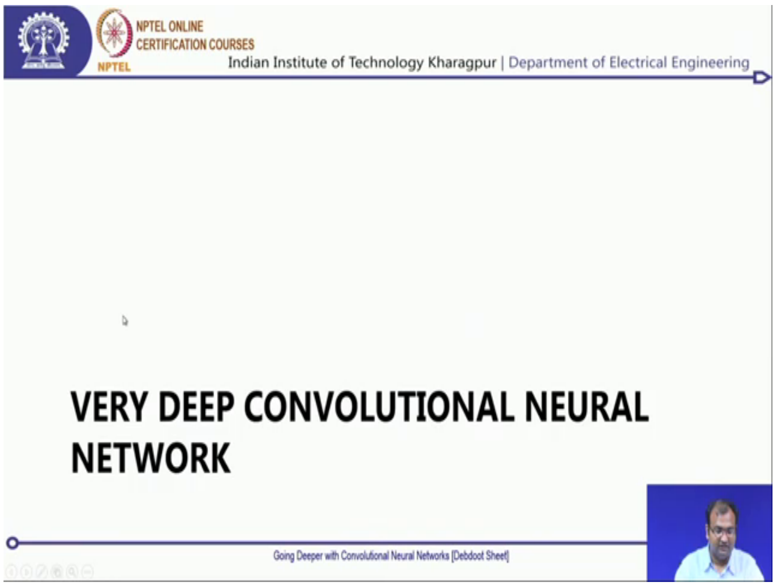
So, its 56 cross 56 cross 64 which is available over here as an input, over here. Now, what we do is, we do a, first is a 1 cross 1 convolution with 64 channels. So, essentially the output of this first layer which is, 1 cross 1 convolution and 64 set channels is going to lead me to a similar sized matrix. The output matrix, remember that these 1 cross 1 convolutions have actually ended up mixing all the properties along one particular pixel location across all the channels and that is the interesting fact which comes down over here.

Now, subsequent to that we have a 3 cross, 3 convolutions with 64 channels and then, again a 1 cross - 1 convolution with 256 channels. Now, this is the place where your numbers of channels actually explode out, ok? So, here we did local mixing across the different channels. In the next one it is where there is a spatial mixing spatial and across the channels mixing on 3 cross 3 and subsequent to that, I just have 1 cross 1, which is again for mixing along it, but then, mixing along one particular pixel location, but you are increasing the number of channels over here.

So, this is where your information is actually getting spread apart. Now, you see this solid line, which comes down over here and that is something which is called as a residual connection link, ok? So, what it essentially does is it takes it all the values which come down over here. So, whatever it was the values available over here and just adds it onto these connections over there. Ok now, addition is as in plain and straight forward addition over there. One thing, which you should ask me at this point of time I was yeah, but then, we are not asking on a real time, it is already recorded.

So here, a question which should pop up is that, see the spatial span is pretty simple. It is 56 cross 56, ok? And here, know where we are doing downs are playing or a striated convolution or a max pool pooling nothing. So, the output spatial size is still 56 was56, but the number of channels has definitely changed out. Now here, we had just 64 channels. Here, you have 256 channels that is 4 times that number of channels over here. So, what we essentially do is that, we just do a 1 cross - 1 convolution and then, up sample it; hope around over here.

(Refer Slide Time: 06:11)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

VERY DEEP CONVOLUTIONAL NEURAL NETWORK

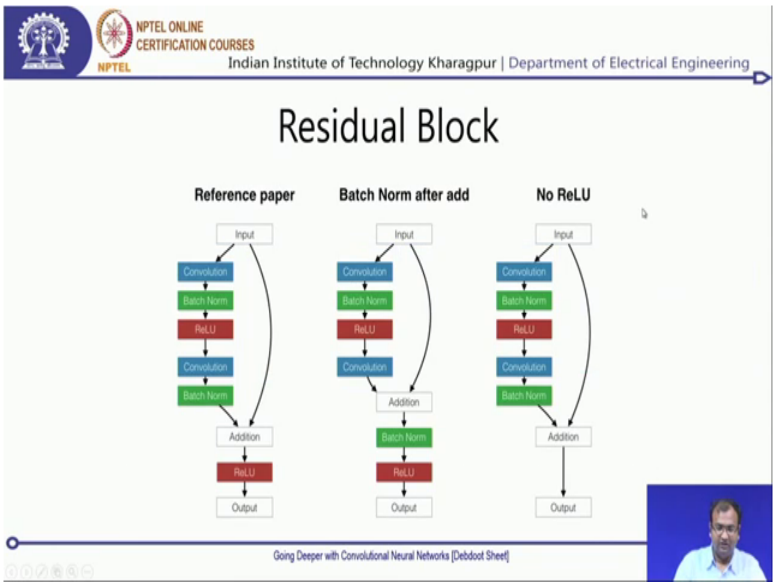
Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

Sorry. So, that you have and this, whatever comes down over here, is a tensor of size 56 cross 56 cross 256, ok? Now, from there is what goes down over here. Now, here again, you have 1 cross - 1 convolutions and getting down to 64 channels. So, your total number of channels over here, which was 256, that again gets reduced to 64, then 64, then again a similar one, but then, this pipeline over here had 256 channels. So, you can actually populate that back over here and the spatial size is still the same, because there are no stripes or there are no max pooling and anything? So, now, it happens like this and comes down over here. Now here, your output is 56 cross 56 cross 256.

Now, you do the next level of convolution over here, where you have 1 cross - 1 convolution with 128 channels and astride of 2. So; obviously, the size had decreased over here. So, the output over here is going to be 28 cross 28 cross 512, but then, here you had 56 cross 56 cross 256. So, you need some sort of a change over here. So, this is this dotted lines over here is what conveys that, over here you are doing, going to do another kind of a convolution with and then, change it down from 56 cross 56 cross 256 to 28 cross 28 cross 512 that is a learnable block as well over here. So, this is how it keeps on going down.

So, whenever you have dotted line, it means that there is some sort of a down sampling in the convolution as well and otherwise, it does not, ok? Now, there are these different kinds of blocks, these are called as CFG0, CFG1 and CFG2 and CFG3. Now, if you want to make 50 such layers, then what you would do is at layer number 3, 4, and 6 ah. So, so what you would do is, you would put down these blocks subsequently at these particular layers over there, this is what is the layer number, which corresponds to there and then you can get down this 50 layer, 101 layer or 152 layers. This is what it comes down.

(Refer Slide Time: 08:20).


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Residual Block

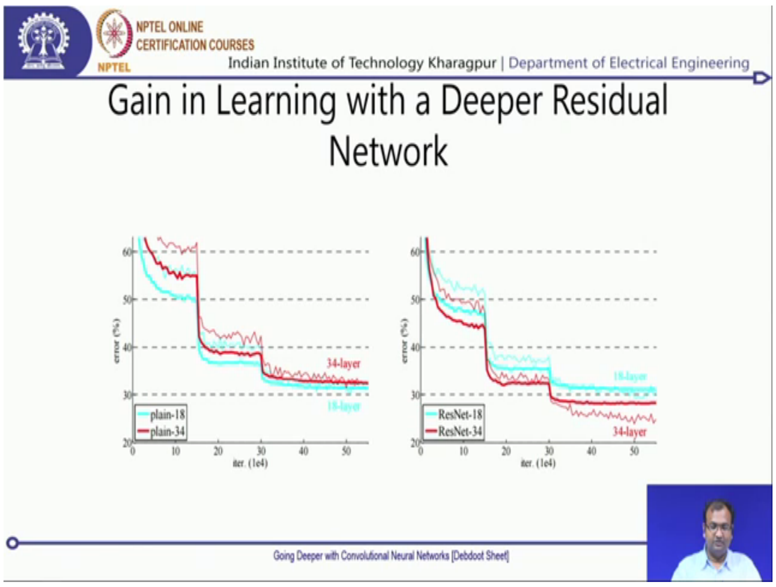
Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

Now on the block side over there, if we are trying to look directly into what happens with the residual, then this is what it looks like. So, the actual paper which implements it out, is something like this that you have your input coming down and this is that part of the residual, that was those pipelines which we were looking down. On the other side, is this convolution which is going down. Now technically we had convolution, then you could do batch normalization, then ReLU and then, a convolution or a batch normalization, you add everything and then, do a ReLU, ok? Now, there was also another option, which have been provided by people who are other than the authors.

Now, here what they did is, they instead of doing a batch normalization over here, they do a batch normalization after the addition and then, so, the addition and batch norm, this location suggest change. So, that technically means that not just this part of it is what is batch normalize, but something which comes from here as well as from this output, together you add and batch normalize and. In fact, that makes a more plausible sense because now your batches do not have a tendency to overshoot even if there is an over shoot coming from here, you would still be normalizing it out over that.

Now, the other one is what they did is, they had all of these batch norms and everything, but they removed out this last ReLU. So, there are just different ways of making it less computer intensive because it is really deep. You have seen that it can go down 152 layers as well.

(Refer Slide Time: 09:41).


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Gain in Learning with a Deeper Residual Network

Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

Now, getting back from there this was the first layer where an interesting thing was observed. So, if you are trying to create a deep network with just 18 layers versus if you are trying to create a deep network with 34 layers. Now, what would typically happen is and then, what they do over here is, these solid lines which you see, they are all training errors and the thin lines,, the bold thicker lines are poor training error. So, the bold one in cyan there is this kind of a blue like colour is what is for 18 layers. The 100 is for 32, 34layers.

Now, technically whatever we were saying till now is that, if you are going deeper and deeper, then your errors should be lower and lower, ok? For the same kind of a baseline. So, that would mean that if I am at a given learning rate and in condition, then this is with one learning rate. These are definitely with lower learning rates over then. That is how you get it down. Now, definitely over there, when I am saturating somewhere over here and then, changing the learning rate.

In order to come lower and lower at this point of time, my cyan curve should actually have been above the red curve or the red curve below, because this is an error. So, my error with a deeper network should be lower than my error with a shallower network, but that is not happening over here. It is very different. Now, this is what got people really intrigued into it, that what was the reason and now, for resonates, what they figured out is one of the major reasons why this was happening is that your input is not getting convict. You need a part of the input to also be converted.

Now, definitely there comes down another question is, if you have this kind of a very deep network, you had a 152 layers over there for Google net, which was just barely about 23 layers or something, you were already facing the problem of decaying and died down gradients. Now, how will you train this whole thing? Now, this is where there is a clue left town. You see this, these extra pipe lines over here. So, whatever is the gradient, this comes over here, it passes through this one. So, there will be like 1 and 2. So, they will say 10 power of 2 change which comes down by the time.

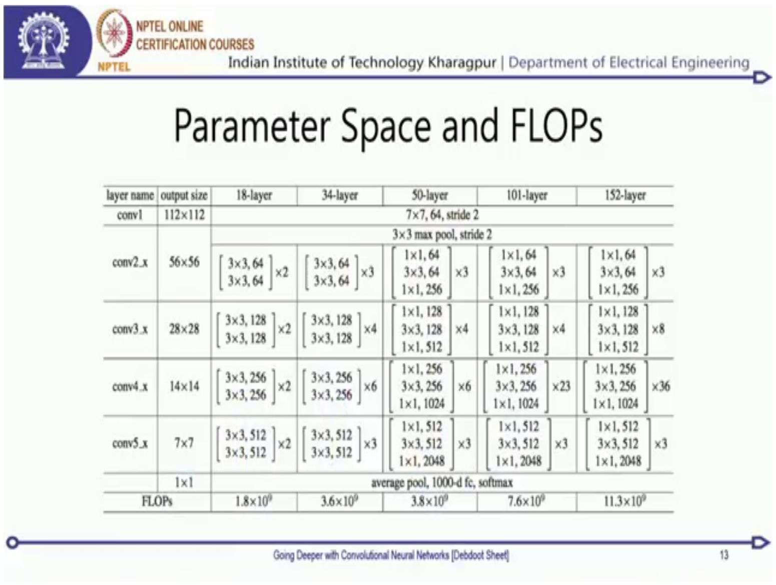
So, if over here it is unity gradient, here, it becomes 10 power of 2, but then, when you are passing it down through this, other pipeline that does not have any attenuation on it, because there are no gradient multiplications with activations over here. So, this is still unity gradient whereas, that gets added to 10 power of 2.Now, this is what is going to influence on everything. So, technically if we look over here, just by virtue of all of these kinds of pipes, which are existing over here, the exact value of gradient can actually be carried forward back till over here and that is interesting.

So, now, you created a network where not just your forward pass could put down the same component of input and carry it down till the end, but as well as your gradient from the end in its exact value, can actually be carried down for influencing this first node over there and; that means, that you do not have vanishing gradient problem over here. So, there were two birds in one single shot that you could hit down. The one you actually have a part of your input going down and merging to create your output, which is really great because they are not able to do that.

You had your white receptacles and everything over there, but a part of the input was never getting merged onto the output. The next, which you could do, was that your gradient as such from the output could be back propagated to your input, making it really easy to learn over there. Now, that is the great thing which happens with a resonate and in essence because of that purpose, you would see that residual networks are the first one which demonstrate, that a deeper network actually gives you a lower error as compared to a shallower network. So, ResNet34in its training is what is giving a lower error than the 18 layer residual network and you can also see that the validation error for the 34 layers goes even below the training error.

Now, that is curious because that is something which we have always desired to have, that whatever be the training error, that should have sort of be the upper bound on my error to be achieved in validation and that is what I see for residual networks coming down. Now, it is interesting that, whether is it only for image net kind class of datasets or do you have it for other class of datasets? That is something I leave it to you. So, we will be doing experiments only on image kind of, Cifar kind of datasets over here. To show it down on a small toy, example that you can take all of these back on your learning exercises and do it on even bigger data sets as well

(Refer Slide Time: 14:10).


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

Parameter Space and FLOPs

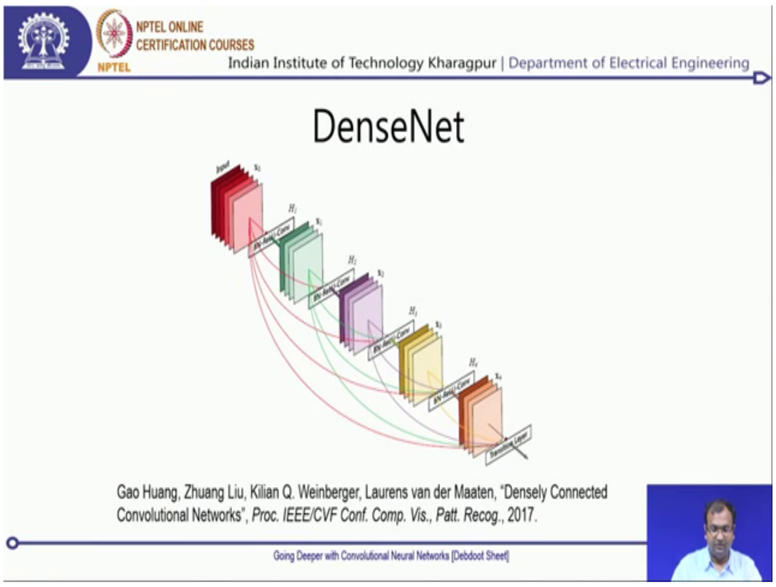
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 13

Now, from there comes down my parameters pieces and the total number of operations which comes down. Now, technically for a 18 layer, what you have is these kind of convolutions over there and then, you can; what you can do is pretty simple. You just need to multiply the parameters over here. These are the parameters which define down your kernels over there and then, just do it and you will find out what comes down as the total number of parameters which you will have to learn down in your model. We will be exploring more deeper into the next model when we go down in terms of versus the number of parameters over there and in terms of number of operations, you see that it takes down 1.8 into 10 power of 9.

So, this is 1.8 billion operations for a18 layer and by the time you are at 152 layers, it is 11.3 into 10 to the power of 9. So, 11.3 billion. Now, from 1.8 billion to 11.3 billion, that was almost like 8 times of a rise which goes down, but from 18 to 252 is the total number of layers, which you have traverse. So, you know how deep you have actually gone down over there. Now, that says sort of like scales or because 18 into 8 would give you a number, which is quite close to 152 maybe. So, that is an. interesting point of an observation over here

(Refer Slide Time: 15:30)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

DenseNet

Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten, "Densely Connected Convolutional Networks", *Proc. IEEE/CVF Conf. Comp. Vis., Patt. Recog.*, 2017.

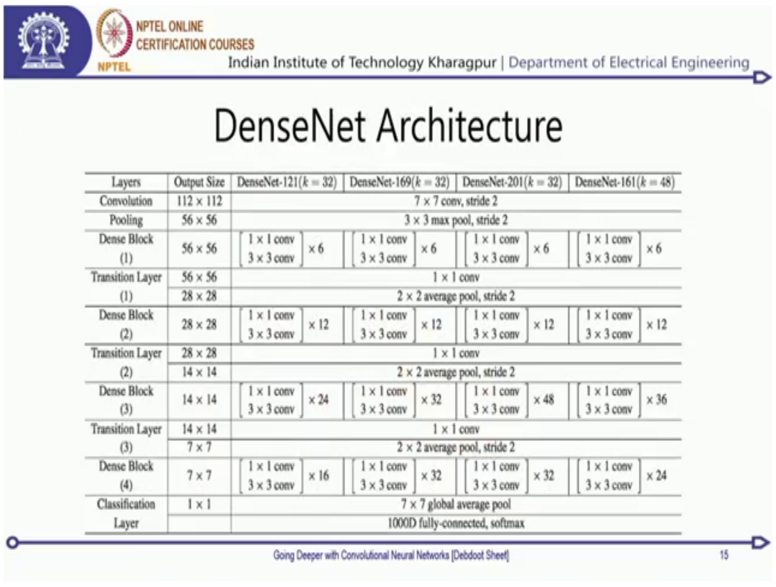
Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

Now, after that is what we have is what is called as a dense net. So, immediately when you had residual connections and everything coming out. So, people thought that while in residual connection what you do is, that the output from the first layer is what is connected to the second layer, but then you do not have a long order dependency, which means that, can this output over here also, influence and connect down to my next one? So, that would mean that, can I have output from any layer connecting?.

Now, to all the subsequent layers and influencing it out. So, that is what a dense net comes down. So, if you look clear carefully through it, what it does is, you have your first set of filter banks over here. You have your batch normalization ReLU and then, a standard convolution. Then, you feed it down to the next bunch of your filter banks. Now, this output is added to the output over here. Next, you have your bunch of filter banks and then, whatever comes down as a output over here, you would see that the output from the first filter bank is what is added, as well as the output from the second with the filter bank.

Now, each filter banks output has the same colour code as in the filter bank. So, the red ones are from the first, the green ones are from the second, the violet ones were from the third, the yellow ones are from the fourth and then, over here from the fifth, you have the orange ones and you see that everything is connected to everything. So, this is a very dense flow of information and then, back when you traverse from here, you would see that all of these get added down and there is; obviously, an original form of the gradient, which gets back over here and modifies all of this whereas, you also have this mapped out gradient slowly like the gradient, which is coming from here, as well as the gradient from here, as well as the gradient from here, which also comes down directly and I would sit down.

(Refer Slide Time: 17:25).



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

DenseNet Architecture

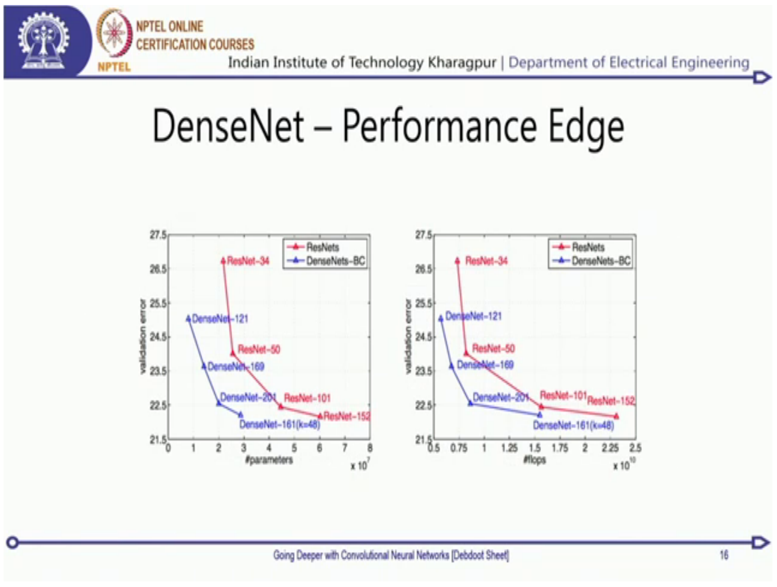
Layers	Output Size	DenseNet-121 ($k = 32$)	DenseNet-169 ($k = 32$)	DenseNet-201 ($k = 32$)	DenseNet-161 ($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 15

So, that means that I have in some way created a mechanism of pushing down gradients from all the other ways and getting it down. Information forward flow is also there, as well as while I am modifying, I am also getting down gradients, coming down from each of them and then, this is a very densely connected network which works out and that is the name for it. Now, where it goes down is, it is more complicated than the earlier ones over there.

So, you have your output flow sizes, block sizes and everything given down and then, you can sum them up totally. So, in the next labs, where we do this ResNet and dense net, we will be looking into the total number of parameters and then, doing their whole summations and finding out how it works out.

(Refer Slide Time: 17:47)



DenseNet – Performance Edge

Now, finally, is this where I would come down as a as a concluding node over here now what you see it is quite interesting. So, we should in the ResNet that if you have residual connections available over there then the deeper you go the lower is the error or the batteries accuracy which you would go down. So then, that would bring you to the point that yes, with residual connections, deeper networks are really good and I could train down deeper networks by not even having the vanishing gradient problem and then, not even trying to have these auxiliary classifier arms, which were otherwise present down in my Google net, in order to get out of this vanishing gradient problem.

In fact, that is also sold out the problem and then, not the most problem, but the interesting fact is that, how what is the order of parameters and then, how does it fare across in both of them? If you look down, say at ResNet, 152, the total number of parameters which you have, is somewhere around 60 60 million 60 into10 power of 6, ok? So, there is 6 into 10 power of 7 that would make it 60 million. A similar order network or even deeper is dense net 161.Now, dense net 161 is what positions itself somewhere over here, ok.

Now, the number of parameters it needs is much lesser than the number of parameters which will be needed down by ResNet, ok? What Is even more interesting is, you go down towards dense net 100, 201, which is actually a more deeper model because you have 201 such convolution layers, but then, the point is that your number of channels over there per layer has been reduced in dense net.101, what that leads to is that for a denser for a deeper network. You actually use lesser number of parameters and now, this model uses just 20 million parameters.

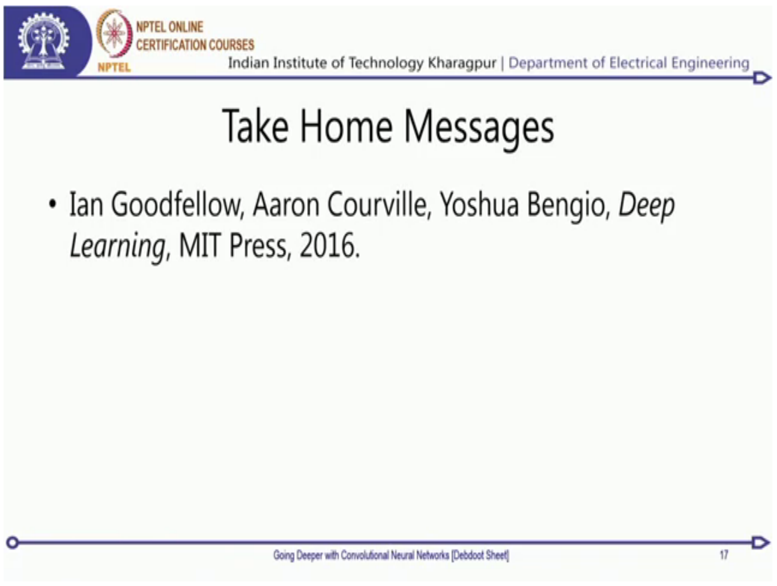
You go down to 152, which is just 50 orders less than that, but this order uses 3 times more number of parameters as compared to this one and if you look into the validation errors over here, you would see that dense net 161, which has the lowest validation error, achieved something with 330 million parameters, 30 million learnable parameters whereas, ResNet152 at the same kind of a validation error, would require 60 million parameters which is double that. Now, move the number of parameters; obviously, you need more number of samples to learn and you need larger number of epochs as well to learn.

So, at any point of time, it gives out that these kinds of dense connections, everything to the other one is where it makes it really interesting. Now, on the other side of it, we look into the total number of compute versus the error which comes down now for a given error vertical. So, somewhere over here, we are sticking down to the same one. So, you have your dense net 161, which would be taking down 1 point or 15 billion flops,15 billion floating point operations at the same error level, you have ResNet152 which would be taking down 22.5 billion operations.

Now, going away from 15 billion to 20.5 billion operations, there is almost 7.5 billion more operations which are needed. So, that is technically 50 percent surge over here. So, whatever is needed over here, we need 50 percent more on top of it. So, half of 50 percent of 15 billion is going to a certain half billion. You add that, when you come down to this position. Now, you see that there is a significant increase as you are going down with just residual network as compared to dense net and this is what the authors have also brought down over there.

You have a much more deeper network; you have lesser amount of compute. You have lesser amount of parameter space and this is a clever engineering which you will have to do. So, while that is really something which I would leave it out to you and while we go down in subsequent, more advanced topics and then, this being just a beginners over here, this was to give you an idea that deeper networks are not necessarily the most compute intensive or not necessarily the most, say parameter intensive networks, as well you can build up deeper networks which can converge much faster, take lesser number of operations to converge them, as well as they would even have smaller parameter space available to them. So, that is with what I would end down this one and on the take home message.

(Refer Slide Time: 22:29)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Take Home Messages

- Ian Goodfellow, Aaron Courville, Yoshua Bengio, *Deep Learning*, MIT Press, 2016.

Going Deeper with Convolutional Neural Networks [Debdoot Sheet] 17

My only point is that, you can go down through this particular book on deep learning and the other, because these networks are pretty new, so, they are not in the book, to all these networks are even beyond the date of publication of the book. So, I would suggest that you actually find them out and then, they are freely available on CBF, on the CV forum on this computer vision foundation online open access points and just go through them. They would be much more interesting.

So, in the subsequent two lectures, we will be having a hands on session on understanding dense net and ResNet and that is where we will do a pentacle calculation versus actual parameter calculation, as we have done for earlier Alex net and Google nets, Alex net and VGA nets as well. So, with that we come to an end on this one and.

Thanks.