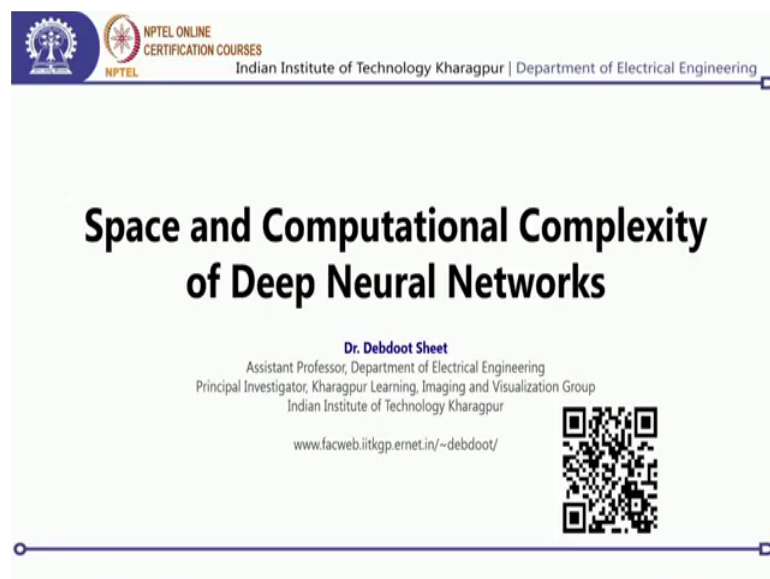


Deep Learning for Visual Computing
Prof. Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 41
Space and Computational Complexity in DNN

Welcome, so today in this lecture we are going to start discussing about some of these very important aspects of operating down on deep neural networks.

(Refer Slide Time: 00:19)



The slide features the NPTEL logo and text at the top: 'NPTEL ONLINE CERTIFICATION COURSES' and 'Indian Institute of Technology Kharagpur | Department of Electrical Engineering'. The main title is 'Space and Computational Complexity of Deep Neural Networks'. Below the title, it lists 'Dr. Debdoot Sheet' as Assistant Professor, Department of Electrical Engineering, and Principal Investigator of the Kharagpur Learning, Imaging and Visualization Group. A QR code is located on the right side, and a website URL 'www.facweb.iitkgp.ernet.in/~debdoot/' is at the bottom.

And like by this classes, which we have done already you have got a very preliminary idea on how to create down your neural networks and then how to go around with training, how to define your cost functions, and then certain kind of learning rule. So, and all of these aspects around learning rules, which you were doing down was either sample wise update or you had a whole epoch wise only ones update or a batch wise update. And you did not realize through these lab experiments, which we were doing down that there is definitely some important aspect about the time being consumed in each of them.

Now the question was which we had not yet touch down, which is for very practical problem for your data is really large. Now, what will you do in that case; and how does it go around and more of from another very important perspective was is there some possibility that you while you are designing the network and you calculate it out.

So, while in one of these earlier lectures, we had done a very preliminary one on calculating out the total model complexity, space complexity and the operational time complexity, for few of these very basic components. So, one of these components was fully connected layer and the other was convolutional layer. Now, this was while you are doing it on layer by layer bases over there we showed you a very simple example; and then while I was doing very deep neural networks over there, I was opening up another table to show you, what is the total number of parameters over there and what is the total number of operations which comes down.

Now the question is that while so these were networks which have been already designed by people and they are available out over there for you to go and try out. The question always comes down will, they be working on your computer on, which you are trying to do it do you need to get down some specialized hardware equipments or maybe get a cloud axis.

Now, quite earlier in the first few weeks of this lecture, when it was starting as well as on the forum; we did not receive a lot of queries on whether there is a requirement for you to get down cloud axis or not. And in the earlier classes what we were quite specifically saying is that; there is no specific necessity for a cloud axis, and infact there is not a specific necessity for getting you very specialized hardware.

Most of these examples which we had done was done in a way, which you can run down on your very simple laptop computers as well. Now; and it never even needed you to run on a g p u most of the times you could do it on a CPU itself; and except for very deep neural networks going down like GoogLeNet, and ResNet and DenseNet is where, you wake making use of some of these very specific g p us. Now, we would standing on top of these experiences which you have as of now.

So, one part was the theory part of it yes the signs over there works out, the math is perfectly tractable, but then you need to have an engineering aspect about that. So, which comes down to the point is that, while you would look down that your network converges in very lesser number of epochs over there. There is also a question like how much time is it going to take per epoch, and if you are given down two networks, which perform equivocal good, but then one of them takes a more number of epochs to converge, where as another one takes lesser number of epochs.

Now, in generally you cannot say that the one with lesser number of epochs is something I would prefer. Now, what would happen down is that; maybe the one which is taking lesser number of epochs per epoch time is much larger and as a result, what it is doing is; it is taking more duration. So, if it consumes one day of a time. So, 24 hours in order to just run 10 epochs versus, the other one which is taking just 2 and half hours to run 100 epochs and both of them are producing equivocal results. In that case, I would be preferring; the later one which takes in just 2 and a half hours to run down 10 epochs over there.

Now, how do you get to these calculations and these are very much machine dependant as well. So, whether you are; what kind of a CPU you are using; what kind of a g p u you are using; it is very much dependant on that. So, today I would be starting with this very basic introduction and going through the revisions once again on the compute complexity and the space complexity. So, that it gets quite into your thought process much earlier over there.

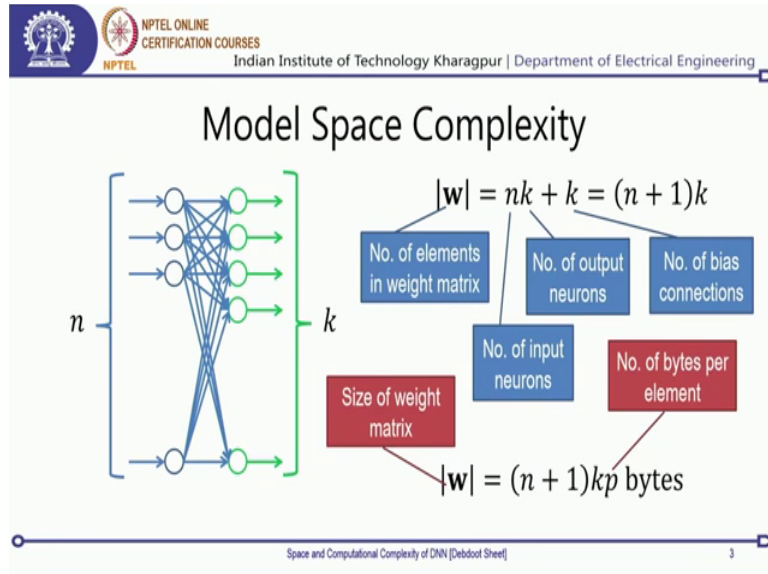
(Refer Slide Time: 04:25)



The slide is a presentation slide from NPTEL. At the top left, there are logos for NPTEL and the Indian Institute of Technology Kharagpur. The text at the top right reads "NPTEL ONLINE CERTIFICATION COURSES" and "Indian Institute of Technology Kharagpur | Department of Electrical Engineering". The main title of the slide is "Organization". Below the title, there is a bulleted list with three items: "Space complexity of model", "Space complexity during operation", and "Computational complexity". At the bottom of the slide, there is a small video inset of a man speaking and a footer that reads "Space and Computational Complexity of DNN [Debdoot Sheel]".

So, what we today do is; we look into the space complexity of model, then the operation complexity and then computational complexity and eventually like after this first bunch of revisions; we would be doing a few of these case studies for very standard neural networks, which we had done in the last week itself.

(Refer Slide Time: 04:42)



Now, coming down to the simplest point over there, what we had is; if we are connecting down n number of neurons to k number neurons in a fully connected layer. So, this was the first one with which had started down ok. Now, what you have is essentially that each of these output neurons over here has connections to each of these input neurons over here. So, that would mean that there are n number of such weights, which are connecting over there and for each of these k ok.

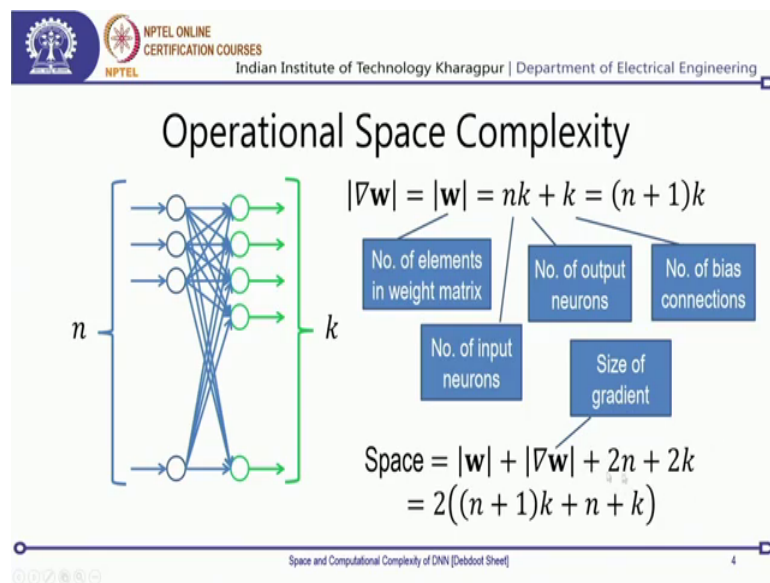
Now, so that essentially brings me that the size of this whole weight matrix over there is something like n into k . Now, each of these k neurons over here, which are neurons on the output also has a bias associated with it, which technically means that there are k number of biases over there and that would make it n k plus k is; what is the size of this weight matrix. So, this is what we had already done and that in a reduce form comes down to n plus 1 times k .

And now since each of these weights are certain elements over there, and this is a matrix technically an array, which resides on your ram over there; now for any kind of an array you; obviously, have a data type restrictions and then this data type is very much specific over there. Now, whether you are using a character type data, which is just 8 bits over there or you are 16 bits of data which is long hint; which is an 16 sixteen bit integer or you are using a long hint it depends on what compilers you are using or you are using a say floating point number on the i triple e floating point number system, which would make it as a typically a double precession floating point number or 64 bits or 8 bytes over there.

So, that is going to be another extra factor, which will consume what is the precession of your number system or p and how many bytes per number are you taking; and that is going to define; what is the total amount of memory it is going to consume while it resides on your ram or on your hard drive. So, in fact while you are downloading the model with pre trained weights; obviously, not the untrained ones. So, when you are downloading the model with pre trained weights, how much does it take? So, in a subsequent lecture, when we enter into transfer learning we would be getting into more of what is the model with a pre trained weight and what happens; when you try to use any of them we also have some interesting exercises lab sessions, actually which take care of all of this.

Now, if I have my weight matrix over here, which is now stored on my hard drive and then I take down p number of bytes per element to store; so this effectively makes the amount of space which I require to store it on my ram or on my hard drive as n plus 1 times of k p bytes over there.

(Refer Slide Time: 07:20)



Now, that brings us to the next point, which is operational space complexity. So, this is what was the space; which I had required to store all of this on to my ram, but now I am going to make use of it ok. So, there will one sample point of data coming to it, and then going through on the output over there and then independently how much of space do I take it over there. So, typically I will have an input x , which has a dimension of n ; which

comes over here. And then what it produces is an output over here which has a dimension of k and then intermediately it will be having certain activations as well.

So, what that comes down is that you also have this nabla of w or what is the gradient of this activation with respect to the network being created over there not the gradient of the activation of the activations with respect to the nonlinearity, but just the gradient, because the nonlinearity is after here over here and then that will take the same amount of space as in k .

Now, we are not looking into that part of it, we are just looking at for each of these samples going over here; what is the gradient which is getting computed. So, that is also typically the size of the matrix w itself, because this is $\frac{\partial}{\partial w}$ of the y activation over here. So, that is the total size of the matrix which comes down. Now, what that brings down is that; your nabla of w is another additional space, which it is going to compute; when you are doing a back propagation.

Now you need to keep one thing in mind, that whatever learning rule you are doing you are always going to back propagate it out. Now, based on what is your optimiser that is a very different point like, what is your update rule, which comes down, but for most of them you are going to; obviously, do a back propagation ok. Now for your simple vanilla gradient decent, which we are looking down as of now; so will need the same amount of space as in the amount of space acquired by these bytes and this is for just one sample. Now, when you are changing over to multiple samples this gradient over here is; obviously, also going to change.


So, that is something you need to keep in mind. So, that we will do that in a later on point of time when we look into actual compute scenario within one particular machine at a given point of time. Now, here is where I would be looking into it; so one part is that you have your weights of this matrix, and you also have your gradient of the activations computed out over there and both of them are of the same size ok. Now together when you look into the total space required for implementing this network. So, that is the some of these two bytes.

Now, on top of that you have your input coming down over there as well as the gradient with respect to your input being taken down. So, that is $2n$ number of amount of space which it would take and there are this k output; which goes down over here and then you

have nonlinearity or your back activation going down over there. So, your backward calculation over there is going to take down $2k$ amount of space. So, this together sums up as twice of n plus 1 times k plus n plus k is what you have as your total operational space complexity required for this operation ok.


Now, this was about how much of space will it consume, while staying on the ram. When you want to do one forward operation or you want to do batches of forward operation. Now, what changes down when you are trying to do a batch of forward operation is that this weight does not change until it is updated, but then Δw is something ∇w is something which will be updated; which will be unique for each of these samples, which is going down over there. And for that reason this is something which will also get multiplied by the batch size. So, subsequently your n and k these values will also be changing for each sample. So, dependent on how many samples constitute your batch is; where you are going to bring in that factor as well ok.

(Refer Slide Time: 11:00)

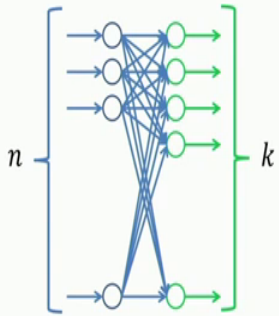


NPTEL ONLINE
CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering



Operational Compute Complexity



multiply = n

add = n


Ops. = $2nk$

- 100 \rightarrow (fc)10
- #Ops. = $2nk = 2,000$
- 4096 \rightarrow (fc)1000
- #Ops. = $2nk = 8,192,000$

No. of ops. per output neuron

Space and Computational Complexity of DNN [Debdoot Sheet]

5



Now next is looking into the operational compute operational compute complexity. Now, why we have looked into space on one side of it the next aspect of any of the algorithms is to look at how many operations is it going to perform in order to do this compute. Now, what I have typically is that I have k number of neurons over here and there are weights associated with k number of neurons there are n number of such weights.

Now, for each neuron I will be doing a multiplication of this input with the weight. So, there are this is an array of length n and this weight over here, which connects down to one neuron is also array of length n . So, now, it is basically a dot product between two arrays of length n . So, element to element, I am doing a product over there ok. Now, when I do element to element product between two arrays, which are of length n in that case then total number of multiplications, which I would be doing is n over there ok. Now, I need to add all these multiplied resultant element. So, I have n and n and then I add all of these together over there. Now, the moment I add all of these together over there. So, what I would typically get down is another array. And then I have n element long array over there.

Now, in order to add down all elements in a pair wise format on a 1 increment array I will have to do n minus 1 number of addition operations over there. Now keep in mind one thing this is not a 0 bias auto encoder, though we have not marked down explicitly the biases over there, but then for a simple standard fully connected network over here I will have down my biases as well. So, there would be for each neuron one extra bias, which is one extra addition which comes down; so that makes it n minus 1 times additions plus another extra addition which makes it up to n .

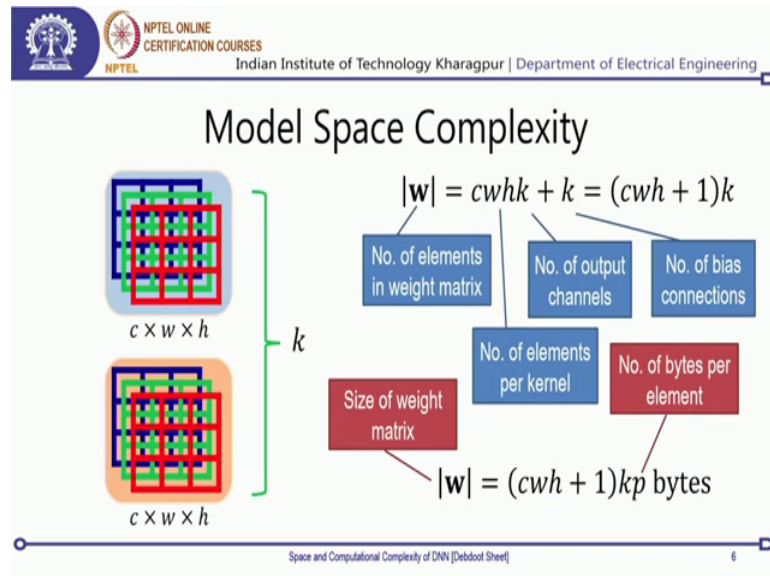
So, there are n number of additions now for our current generation of machines as we are discussed earlier multiplies, adds and everything happen pretty much at the same machine cycle over there. And you can call this as just one operation itself; whether it is a multiply or add we are not much worried about it is just an arithmetic operation which is going down.

So, that would mean that my total number of operations now over here while trying to do it comes down as twice n k. So, it is n plus n and which makes it $2n$, and then I do it for each of these neurons which is for k number of times ok. So, that makes it $2nk$ operations over here in order to implement it. Now for us simple example we had seen that, if I connect down 100 neurons to 10 neurons, then this whole things comes down to about 2000 operations over there. Now, if I connect 4096 to 1000 neurons, then this comes down almost like 8.2 million operations.

So, just from changing down from 100 neurons to 4096, which is say approximately 40 times more over here I really had about 400 times more of an operation going down. So,

that is where the change really comes into it. So, it is its there is no guarantee that how it work out. So, you need to really figure it out and it is just always dependent on the total number of operations. Now, this was about the operational complexity for a fully connected layer and we had seen that while the operations over here are still look as if they are much denser, but then the total number of operations is much lesser.

(Refer Slide Time: 14:17)



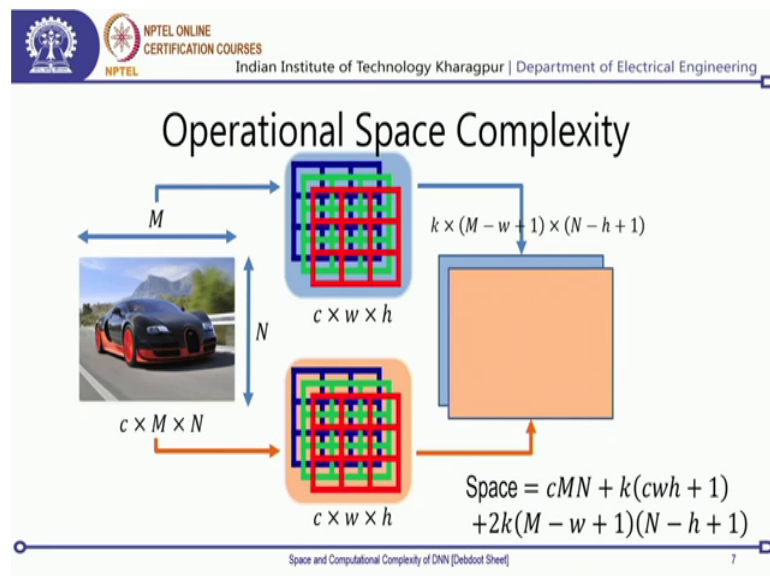
Now standing on that we; went over to the next one, which is to look into convolutional network over here. Now for convolutional network, if you look into the space complexity just for the model which is; if you are downloading the weights and storing it over there and how much of space would these models just be taking down over there. So, just for it you have if you have input number of channels, which is equal to c and the width and height are w and h , then your total space being taken down just for storing 1 kernel is equal to cwh ok.

Now, if I have k number of such kernels coming down, then there will be another extra factor, which comes into play which is k which is the total number of kernels for the output channels over there. Now associated with each channel there is also weight there is also a bias over there as in we had in a fully connected network. Now these biases would mean that there are extra k number of such additions, which come down over there and so, this brings it down to $cwh + 1$ times k . Now you can note a similarity between a fully connected layer and a convolutional one. So, in a fully connected layer

basically your k is the total number of output neurons your n is the total number of input neurons, which is equal to $c \times w \times h$ and that something which guides down just the weights over there.

So, this is what we have for a plain simple convolutional network and based on these operations; as if you can break it down always you can come down to where it goes down to ok . Now, in the same way if I am looking at the models space complexity then the total amount of space I would require to store it down; if I am using a number system with the precision of p is something equal to $c \times w \times h$ plus 1 times of $k \times p$ and that is plain and simple as one an analogy from what we had done for a fully connected network itself ok .

(Refer Slide Time: 16:01)



Now comes the next part, which is to look into the operational space complexity. Now here you need to keep something in mind that say you have an M cross n image and your convolving it over here with the c cross w cross h kernel over there.

Then you are going to get down M minus 1 M minus w plus 1 cross n minus h plus one sized resultant and there will be case number of such. So, where k corresponds to the total number of kernels, which you have for your convolutions being drowned down now that brings us that the space complexity is something of this part. So, your input over here will take c into M into n that is your input space over there now your kernels over here they would take down k times $c \times w \times h$ plus 1 amount of space good; now that is done.


Now your output which you get generated over here, that is what is $M - 1$ $M - 1$ $w + 1$ $N - w + 1$; $N - h + 1$ into k , that is the space which it takes down.

Now, I also have the gradient of the activations, which is ∇w which is calculated and that is the same size as that of this space. Now, this is where the whole aspect varies; and is different from a fully connected network. So, while in a fully connected network the advantage was that your output size was dependent on the total number of neurons over here, but then here in a convolution your output size does not depend on the total number of neurons, it is based out of this convolutional calculations that you get down your output size.

However, the major point is that the; gradient of these activations and the activations they are of the same size and they would be of the same size as this response map, which you find it out. So, there are it takes in twice the amount of memory, which is to be consumed over here ok. Now, that is what the operational space complexity, which goes down which we had done pretty much. Now, the fact which we had not done yet in the earlier one and I had left it out for you guys to actually solve it out and then as an interesting problem, but I would be covering it today to check out whether your solutions are matching down with our solutions, which we know directly from the analysis aspect over there.

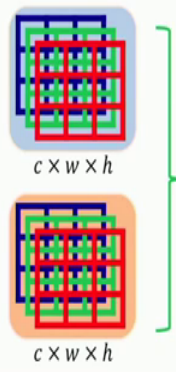
So, you could pretty much pull down any of your analogies, which you had learned down in your fully connected network to do these calculations for your convolutional networks as well ok.

(Refer Slide Time: 18:13)



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

Operational Compute Complexity




$c \times w \times h$
 $c \times w \times h$

k

- # multiply = $cwh(M - w + 1)(N - h + 1)$
- # add = $cwh(M - w + 1)(N - h + 1)$
- # Ops. = $2cwh(M - w + 1)(N - h + 1)k$
- (Input)(3,224,224)->(Conv)6c5w1s0p
- #Ops. = 43,560,000
- (Input)(3,224,224)->(Conv)16c3w1s0p
- #Ops. = 42,581,376

Space and Computational Complexity of DNN [Debdoot Sheel]



So, there comes down aspect of operational compute complexity for convolutional network. Now, here you would see that you have c cross w cross h number of kernels over there. And c cross w cross h is the size of the kernel and you have k number of such kernels available with you. Now, if you look into the total number of multiplications, which it does over there; then your total number of multiplications is something, which comes down of this form now it is it is not; so hard to understand over there ok. So, what you would do is essentially you would take your kernel and then you superpose on your image on the location, which superposes over there.

Now, the moment you are superposing over there; so underneath this kernel; so this kernel matches down with c cross w cross h number of elements present over there. So, and it is doing a dot product with each of them. So, technically it means that it does a dot product or does a point to point product over there for c into w into h number of times ok.

Now I am going to give it a stride and move it down. So, based on how much my strides is my resultant matrix size varies. So, the total number of times I would be moving over there is; what is M minus w plus 1 as of now, if I am not taking a particular stride if I am just doing it with a classical method, which is just with a stride of one. So, now, what I am doing is that for each location I am going to do these many number of multiplication $c w h$ and then I am going to move over so many times.

So, the total number of times I would be doing this dot product operation is M minus w plus 1 times of N minus h plus 1 great. Now, if I look into my additions as well. So, for each of these when it superposes over there I will have $c \times w \times h$ number of which is the size of that array over there and then I am going to add all the elements. So, it requires $c \times w \times h$ minus 1 number of additions in order to add it ok. There is also a bias associated with each of these kernels over there and that is what I am going to put down ok. So, technically that would mean that for each of these kernels over there I will have $c \times w \times h$ minus 1 plus 1 ok. And that I am going to repeat it over the whole image which comes down. Now, if I look into my total number of operations.

So, the per kernel it becomes twice of $c \times w \times h$ into M minus w plus 1 into N minus h plus 1 ok. And now I have k number of such kernels over there. So, this is what is what gets multiplied and I get down in the number k ok. Now, say for a very simple one, which is with a three channel input image of 224×224 and then I convolve it with a kernel of 5×5 and stride of one and then there are 6 output channels over there. Now, that would mean that my total number of operations which I am doing in this case is about 43 million operations, which I have 43 point million operations.

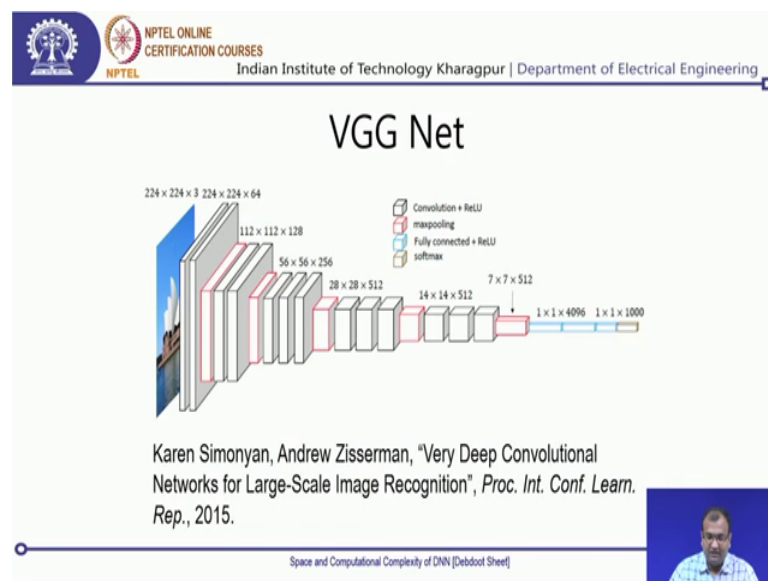
Now, if you look into it then the total number of neurons, which you actually had over here was much lesser, because the size of this convolution kernel is $5 \times 5 \times 3$ over here and then you have total 6 number of these. So, the total number of parameters, which you have is much lesser, but then total number of calculations you are doing is significantly larger; now this is where it really makes it interesting over here.

Now, whereas, on the other side of it, you would have observed that when we were doing these bench marking and learning rules and what happens across each of them fully connected layers sometimes do 10 to take a lot of time to converge as compared to an in fact like per epoch time for a fully connected layer is generally more than per epoch time. But, then you have realized that the number of operations for these convolutional layers is much larger than the number of operations you have for fully connected layer.

Now this is something I will keep on for the next lecture where; I would be going into handing on hardware bottle next and what are the challenges which are faced over there for each of them ok. Now, this is what comes down for a decent sized small network.

Now, if I have another network, which is say which connects down 3 cross 224 cross 224 cross onto 16 cross 16 channels over here and each of 3 cross 3. In that case you would see that the number of operations is slightly reduced, but that not much. So, it is it is about just 1 million operations which are been reduced, but then 1 million over a factor of 43 million is not a significant reduction as you know ok. So, this is about going down with computational complexity operational compute complexity for your convolutional methods which we had not done in the earlier on.

(Refer Slide Time: 22:40)



Now, where that brings us is if you look into a standard VGG net like architecture over here; you would find out the total number of parameters which are described out over there.

(Refer Slide Time: 22:44)

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

VGG Net – Parameter Space

Conv-Net Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
conv-3-64	conv-3-64 LRN	conv-3-64 conv-3-64	conv-3-64 conv-3-64	conv-3-64 conv-3-64	conv-3-64 conv-3-64
maxpool					
conv-3-128	conv-3-128	conv-3-128 conv-3-128	conv-3-128 conv-3-128	conv-3-128 conv-3-128	conv-3-128 conv-3-128
maxpool					
conv-3-256	conv-3-256	conv-3-256 conv-3-256	conv-3-256 conv-3-256	conv-3-256 conv-3-256	conv-3-256 conv-3-256
maxpool					
conv-3-512	conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512
maxpool					
conv-3-512	conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512	conv-3-512 conv-3-512
maxpool					
FC-4096	FC-4096	FC-4096	FC-4096	FC-4096	FC-4096
FC-1000	FC-1000	FC-1000	FC-1000	FC-1000	FC-1000
soft-max	soft-max	soft-max	soft-max	soft-max	soft-max

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Space and Computational Complexity of DNN [Debdoot Sheet]

And, I hope you have already done your bit of calculations though we had done it in the our practical sessions with VGG net, but then it was also left out to you that you tried down using your pen and paper method for each of them and then verify, because this is important from your learning perspective that you have actually learnt the whole data structure. The moment you are able to actually find out, what is the total space requirement over there and what is the size of the parameters you are learning. You know quite intricately; what is the structure of the data and how it is working out within the network over there ok.

(Refer Slide Time: 23:24)

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]

Now, for a GoogLeNet what we had seen is you had this kind of a whole table given out with GoogLeNet, which made out one simple aspect which is at the end of it you could get down, what is your total number of parameters which is being used in this particular layer as well as what is the total number of operations which comes down.


(Refer Slide Time: 23:26)



Parameter Space and FLOPs

type	patch size/ stride	output size	depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5	pool proj	params	ops
convolution	7x7/2	112x112x64	1							2.7K	34M
max pool	3x3/2	56x56x64	0								
convolution	3x3/1	56x56x192	2		64	192				112K	360M
max pool	3x3/2	28x28x192	0								
inception (3a)		28x28x256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28x28x480	2	128	128	192	32	96	64	380K	304M
max pool	3x3/2	14x14x480	0								
inception (4a)		14x14x512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14x14x512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14x14x512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14x14x528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14x14x832	2	256	160	320	32	128	128	840K	170M
max pool	3x3/2	7x7x832	0								
inception (5a)		7x7x832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7x7x1024	2	384	192	384	48	128	128	1380K	73M
avg pool	7x7/1	1x1x1024	0								
dropout (40%)		1x1x1024	0								
linear		1x1x10000	1							1000K	1M
softmax		1x1x10000	0								

Going Deeper with Convolutional Neural Networks [Deeboot Sheet]

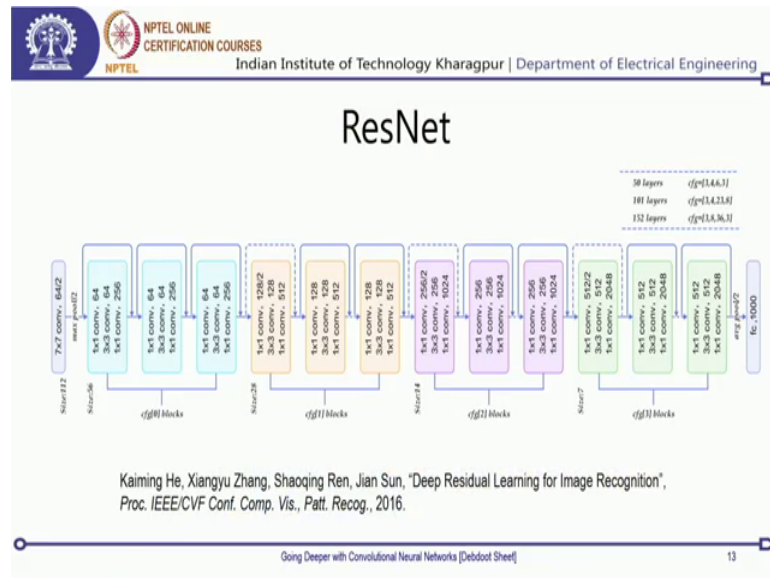


Now, you can see pretty much for a decent sized layer, where it has 159000 parameters you have about 128 million operations going down, where you have 380000 parameters you have about 403 million operations which are going down over there. So, this is what is (Refer Time: 24:00) an interesting; now it is not always necessary that, if your number of parameters is high that your operations will also be high or even higher, because if you look down typically like; if I compare this particular layer, which has 380000 parameters, verses this layer which has 437 parameters 437000 parameters.

Now the next layer which is inception 4b has more number of parameters than inception 3a. However, you can see that the total number of operations in inception 3b is 304 million whereas, in inception 4b it is just 88 million.

Now, there is no direct one to one correspondences, which I have meant by this one and this is something came into keep in your mind as well. So, it is not necessary that more number of parameters would incur a larger number of operations over there though there is another interesting aspect about the memory bottle necking, which might happen on hardware, but that we keep it down for the next lecture to come on.

(Refer Slide Time: 24:58)



And next comes down your residual networks.

(Refer Slide Time: 25:02)

Parameter Space and FLOPs

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112x112	7x7, 64, stride 2				
		3x3 max pool, stride 2				
conv2_x	56x56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28x28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14x14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7x7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1x1	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]

So, in your residual networks they have a very good table given out. So, if you have an 8 layer residual network versus the 34 layer versus 112 even 152 layers, then what is the total size for each of these layers over here and what is the total number of operations, which it is going to take down. And at any given point of time you can see that your residual networks have lesser number of compute demand as compared to GoogLeNet over there and or a VGG net.

(Refer Slide Time: 25:30)

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 similarly comes down to your we come down to the DenseNet.

(Refer Slide Time: 25:32)

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

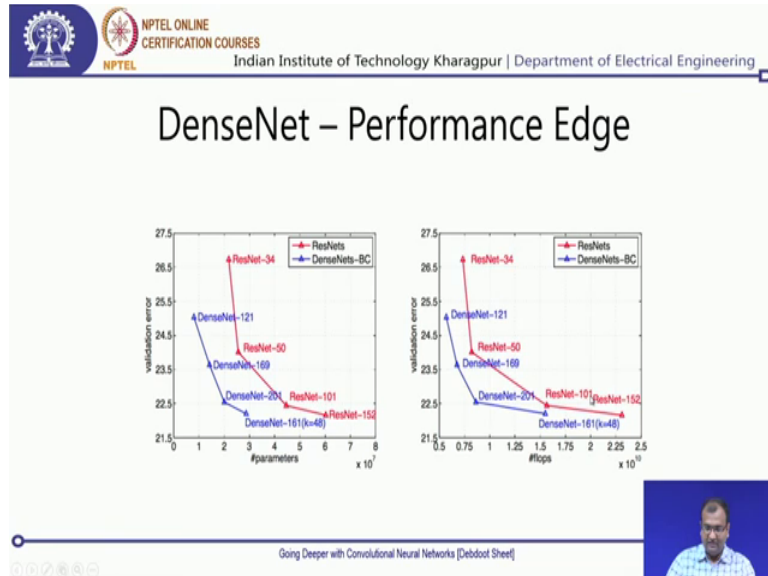
DenseNet Architecture

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	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6
Transition Layer (1)	28 × 28	1 × 1 conv 2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12
Transition Layer (2)	28 × 28	1 × 1 conv 2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	1 × 1 conv 3 × 3 conv × 24	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 48	1 × 1 conv 3 × 3 conv × 36
Transition Layer (3)	14 × 14	1 × 1 conv 2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	1 × 1 conv 3 × 3 conv × 16	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 24
Classification Layer	1 × 1	7 × 7 global average pool 1000D fully-connected, softmax			

Going Deeper with Convolutional Neural Networks [Debdoot Sheet]

And you can also have a similar kind of a table done for your DenseNet where whereas, the compute complexity is where they put down in terms of your graph.

(Refer Slide Time: 25:37)



In order to compare residual network with a densely connected residual network to find out that with lesser number of parameters, you can achieve the same validation error and in fact with lesser number of parameters you also have a lesser number of flops, which is directly associated over there. So, this is where we come to an end for this lecture and then one important take home message is that deep neural networks do not or deeper networks do not necessarily imply that there would be a higher model and space as well as compute complexity taking over there.

(Refer Slide Time: 26:03)

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

Take Home Messages

- Deeper networks do not necessarily imply higher model space and compute complexity

Space and Computational Complexity of DNN [Debbot Sheet]

So, with that we came to an end to this one. And just stay tune for the next lecture on which we are going to discuss more about the hardware aspects, and what happens

within when you are trying to run it down on your computer system, and what are your hardware specification on computer system which are going to restrict or allow you to have your flexibility, and how fast will it be working. Till then stay tune.

Thanks.