Deep Learning for Visual Computing

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

Lecture - 35
Revisiting AlexNet and VGGNet for Computational Complexity

Welcome. So, today, we would be speaking on, computational complexity for deep neural networks. So, one of the networks which, we have done till the last one is what is called as VGG net or which is from one of the early versions of, deep neural network available.

Now, one of the parts which we had discussed till now, as with any of the networks was to look into the feed forward behavior and then on the back propagation behavior and all of these behaviors were just trying to look down into the linear algebra perspective and, whatever comes down, subsequently over there.

So, while we were looking down into how networks can be represented in terms of matrix operations as well as subsequent like, if these matrix operations can now be termed as some operations, which are available on tensors, from the perspective that all of these representations in terms of my input space my weights of, the connections on the network as well as my output, which comes down and represented as tensors then for back propagation.

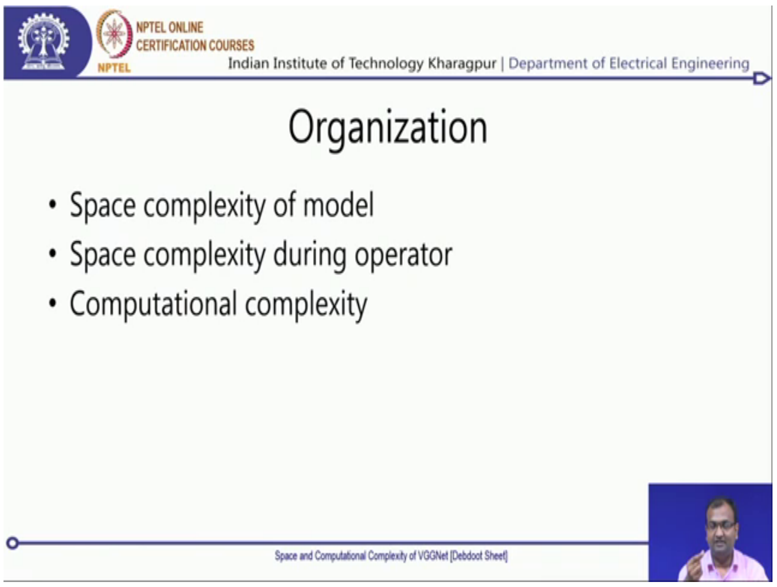
On the other side what I have is the derivative computed out, over these responses as falling down my chain rule. Now, what interestingly comes out is that, we throughout this whole process, we have not yet analyzed out, how much of space does it take to store all of these intermediate matrices and the inputs, outputs and, all of these intermediate activations or calculations, which happen down by a product on these matrices and on top of that, how much of space does it take to, in fact, stone down all of these gradients as well, because when you expand out of the chain rule, what you find out is that, these gradients are not just with respect to a, your error over there.

So, there is a part of the error, which is taken down as a partial derivative with respect to the, activation generated at the last note. And then you have the partial derivative of the activation generated at each note, taken down with respect to the non-linearity over them, and the linear summations output with respect to, the weights over there. So, this is how it goes down.

So, you need to have each of these stored and then goes on and on top of it another major part is it, you are seeing certain networks while we were training it down. They could very easily converge, they took less time in terms of CPU time, not in terms of epochs whereas, a few of them were taking much more time to come down to a convergence and this was; obviously, an issue as to what and how these are going to work out.

Now, today what we are going to discuss is actually to look into how much of bites or does it take to store all of these matrices? What will be the total memory, which is being consumed over here? What will the total number of elements as well present over there and, and not just in terms of these weights over there, but throughout the whole process as well as how many number of computes go down and that has a very significant role to play as, to what kind of a hardware you are using.

(Refer Slide Time: 02:54)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Organization

- Space complexity of model
- Space complexity during operator
- Computational complexity

Space and Computational Complexity of VGGNet [Debdoot Sheet]

So, without waiting, for much this is how the whole thing is organized down. So, first we are going to look into the space complexity of a model then, look down into space complexity during, the operation or, for that operator as it keeps on going on and then the computational complexity as these operators are operating out on a given sort of an input. So, this is typically the three different aspects, over which we divide and, to keep it simple, because VGG net kind of a network is quite big.

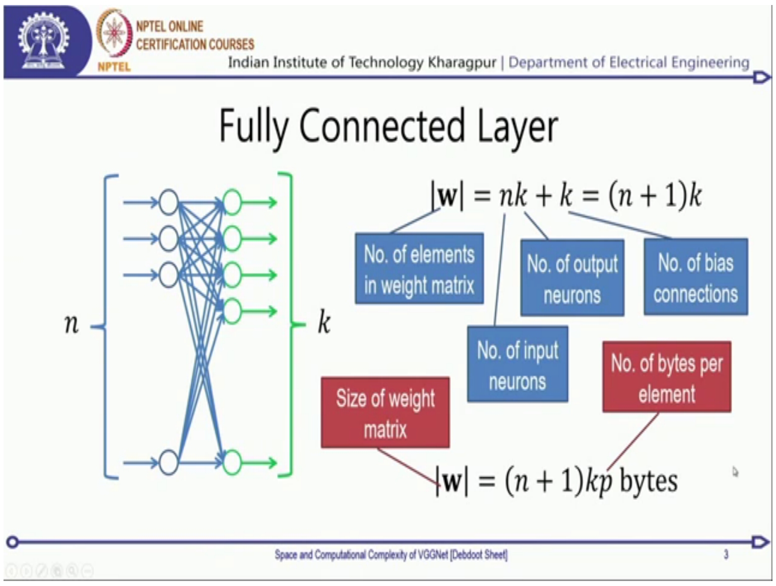
So, what I would be doing is I would give you explanations layer by layer, intuitively, we will start down with the first and the easiest of the layers, which is a fully connected layer and then we would enter into the convolution layers and then I will be giving you an example of how it keeps on going. So, this would then extend on to our VGG network, where you get to understand, like if you are calculating it out per layer basis, then you sum it up and then you get down the results over there.

So, some of these are in experiments, which you have done. So, you have already finished off VGG net, where we look down into the weights per layer and then we also look down into what is the total number of parameters. Now, this is a detailed explanation of how you calculate out exactly, the total number of parameters over there on a pen and paper method, even before you write down your code and try to get down, your parameters directly from within your code, form over there.

And this is quite crucial, because you would be seeing down that we just make small tweaks around the network over there. We change the number of channels change the. So, I like one of the examples, which I would be using is, reducing the kernel sizes; So, instead of 5 plus 5 kernels.

I would now, be using 3 cross 3 kernels, but then I would increase the channel width over there and the question is like, if you are changing this one then, does it impact the overall space consumed over there. So, these are kind of examples, which we will be looking through ok.

(Refer Slide Time: 04:39)


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

Fully Connected Layer

$$|\mathbf{w}| = nk + k = (n+1)k$$

No. of elements in weight matrix
No. of output neurons
No. of bias connections
No. of input neurons

$$|\mathbf{w}| = (n+1)kp \text{ bytes}$$

Size of weight matrix; No. of bytes per element

Space and Computational Complexity of VGGNet [Debdoot Sheet] 3

So, let us look into the first one over here. So, this is about n number of neurons connected down to k number of neurons and this is a, this is with a bias. So, this is not a 0 bias network as such. So, there are biases, but then I choose not to represent any of these biases, because they, they are inherently taken down that for each of these output neurons over here, you will have one-one bias connected.

So, now, if you look into the weight matrix over here, what you would see is that each of these n neurons is connected to the, k number of neurons over there or on the other side of it, if I look into one of these neurons. So, one of my output neurons over here is, what will have all of these ends coming down in terms of, multiplied through weights. So, there are n weights associated with this neuron and there is also a bias. So, there is n plus 1 number of weights which are associated with this neuron ok.

There are n plus 1 weights, which are associated with this neuron n plus 1 for this neuron so on and so forth. It gives, goes on till here. So, in total you would see that there are n plus 1 number of weights associated with k number of neurons. So, this matrix over there is of the size of n plus 1 cross k ok. So, that is, what I get down over here? So, you will have n number of neurons and k number of output neurons. This is n into k, which is the total number of connections over there and you have k number of biases, which come down over there and this whole thing can now, be represented in terms of n plus 1 into k right.

So, that is what we had done now. Now, this is for a fully, connected layer. Now, this what it essentially gives is that, what is the total number of elements which make up this weight matrix over there. Now, it does not give me exactly what is the size in terms of bytes. So, when you load a memory, load, load a model onto your memory. So, you have seen that from our, torch vision, we could actually download models on pi torch and then some of these models would take lesser amount of time to download some of the models, take longer amount of time to roll down.

So, if I was trying to download a VGG net, it takes longer. if I am trying to download an Alux net, though we did not, actually download an Alux net and, and in the subsequent ones, when you will be going on into much deeper networks like Google net, dense net, (Refer Time: 06:46) net, you would see that we will be downloading all of these models.

Now, for each model it shows you a size and this is the calculation, which we are going to do. Now, in terms of finding out what will be the size calculated out. So, now, say that, you have p number of bytes per element represented over here; So, then this whole thing becomes down as n plus 1 times of k into p number of bytes ok. Now, the question is what will leaks bytes actually represent ok.

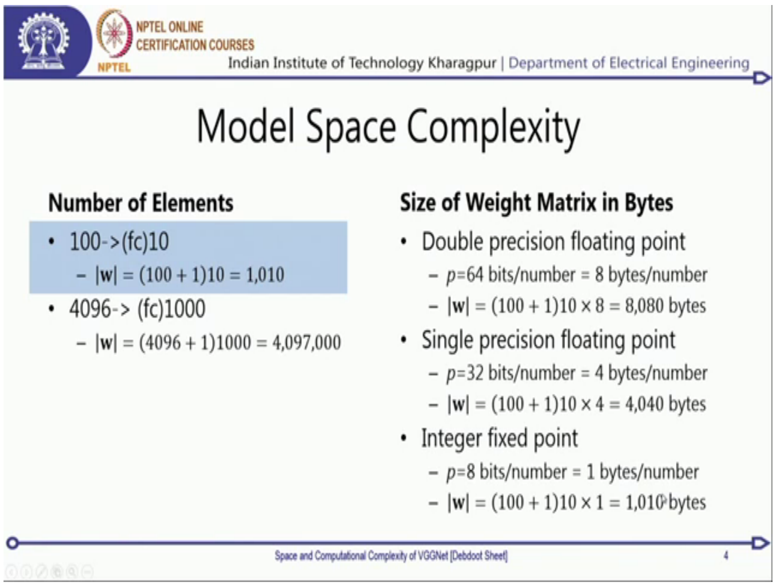
Now, that is right simple. So, most of our operations are, what are carried down in double precision floating point numbers. Now, per definition, double precision floating point numbers are what are 4 bytes or sorry, 8 bytes or 64 bits in size. Now, the moment you have 64 bits or 8 bytes over there. So, for representing one number, you are going to consume 8 bytes. So, if there is one number it is 8 byte long. Now, instead of that if I try to represent it in terms of single precision floating point number system, which is 32 bits or 4 bytes.

Then, the same value can now, be represented in terms of, half the number of bits; however, you need to remember one thing that since these are represented in terms of your, I triple e floating point number system. So, there is a part of it, which is a mantissa and that is an exponent and then you combine this together to counter. Now, the moment you reduce your byte size over there, the total number of I, sorry, the number of bits available in order to represent a number, you would see that there is a significant reduction in the precision of these numbers or to subsequent numbers. What will be the difference between them?

Now, if I go down and, and make it even further small. So, I am just going to represent it in terms of an integer ok. Now, in terms of an integer you have, just 8 bits present down over there and just a character type integer or 8 bit integer over here ok. So, now the difference between two numbers, so if my lowest number is 0 and my highest number is 1. So, I can have 256 levels, in which I can divide it out over here. So, that is 256 is, is. So, 1 by 256 is basically the precision of this number, which I can generate, by using 8 bits ok.

On the other side, if I reduce it even further, go down to just 4 bits or a nibble over there. So, this becomes 1 by 2 to the power of 4. So, that is 1 by 16. Now, that is even further or less. So, this is where it plays down, the precision has it's own significant impact as well into it, but nonetheless for us as of today, it makes one of the major impacts in terms of how much space is it going to consume.

(Refer Slide Time: 09:18)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Model Space Complexity

Number of Elements

- 100->(fc)10
  - $|w| = (100 + 1)10 = 1{,}010$
- 4096-> (fc)1000
  - $|w| = (4096 + 1)1000 = 4{,}097{,}000$

Size of Weight Matrix in Bytes

- Double precision floating point
  - p=64 bits/number = 8 bytes/number
  - $|w| = (100 + 1)10 \times 8 = 8{,}080$ bytes
- Single precision floating point
  - p=32 bits/number = 4 bytes/number
  - $|w| = (100 + 1)10 \times 4 = 4{,}040$ bytes
- Integer fixed point
  - p=8 bits/number = 1 bytes/number
  - $|w| = (100 + 1)10 \times 1 = 1{,}010$ bytes

Space and Computational Complexity of VGGNet [Debdoot Sheet] 4

So, let us take a very simple example over here, in order to find out my model complexity. Now, say that I start down with a simple example, where just 100 neurons are connected down to 10 neurons for me ok. So, f c is a fully connected layer. This is the kind of a convention, which we have already discussed in the earlier lectures. So, if I am connecting down 100 neurons to 10 neurons and this is some intermediate layer over there.

These 100 neurons might be directly coming from source or they might be coming down from any of the previous layers over there ok. Now, for my definition, the total number of elements over there; So, this is cardinality of the weight matrix that is why I just put down a mod symbol over here, that is equal to 100 plus 1; This side plus 1 into 10. So, that is 1000 and 10 number of, elements present down in this matrix ok. Now, say for any of our networks like VGG net ok, if I am looking down at the final classification layer.

So, the final classification layer is where 4096 neurons are connected down to 1000 neurons ok. Now, in that case this whole thing becomes down something like 4 million 97000 neurons over there. So, this is the total number of neurons present down in, in total number of elements, which would be representing my weight matrix over there that is plain and simple.

Now, comes the question that if I am choosing down my, precision of the system. So, let us take only one of these examples, where just 100 neurons are connected down to 10 neurons. It is a very straightforward example. Now, we will be looking down into by changing precision, what will be the total size in bytes, which this weight matrix is going to, occupy ok.

Now, if I do something like that I have a double precision number system. So, when I was defining my net I had defined everything in terms of double. So, now, my, my weights in the network, my input to the network, my output from the network everything is in a double precision number system.

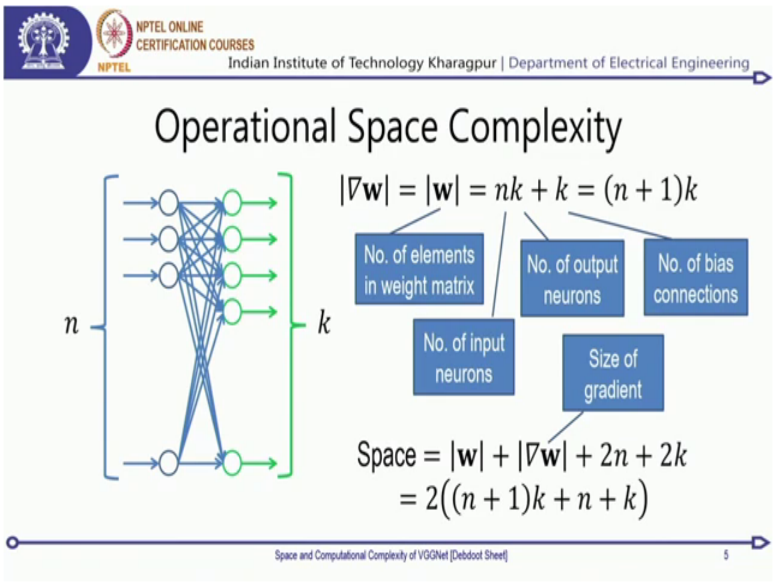
Now, over there what I have effectively is, we had this p on the precision, which is 864 bits per number or 8 bytes per number and then this model had 1 0 1 0 of 1000 and 10, such numbers present then in order to represent my weight matrix over there. Now, it is 8 bytes per number over there. So, in total it comes down at 8080 bytes ok. So, this is what goes down with a double precision floating point number system. Now, say that I change it and come down to a single precision floating point number system, which requires just 4 bytes per number. In that case, this comes down to 4040 and 40 bytes in total, which is the size of the weight matrix ok.

Now, if I go even further down and come down to an integer, fixed point number system; So, an integer number system is no more floating my number system. It becomes fixed point number system and typically, we would just be going down as a non fractional or, or just integer type numbers over there. Now, in that case, this whole thing becomes down, comes down to just 1000 and 10 bytes. Now, if you look into it clearly, by changing from using an integer type system to a floating point number system. I am consuming 8 times more of memory over there.

In fact, my model will also be bulky and it will take more number of bytes. So, the number of bytes downloaded is also larger. So, here while a this, this just. This simple layer is going to take you 8 kb in the earlier case, with just an integer point number system. This is going to take you 1 kb.

Now, you are going to take 8 times more, 8 times of more download time in order to even download the model. In fact, when you are loading down, something from your hard drive onto your, RAM space from us to as a store model being loaded down for further compute, you will be taking down 8 times more time over there as well. So, this is where the change comes in.

(Refer Slide Time: 12:54)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
n
k
No. of elements in weight matrix
No. of input neurons
No. of output neurons
No. of bias connections
Size of gradient
Space and Computational Complexity of VGGNet [Debdoot Sheet]
5

Operational Space Complexity

$$|\nabla \mathbf{w}| = |\mathbf{w}| = nk + k = (n+1)k$$

$$\text{Space} = |\mathbf{w}| + |\nabla \mathbf{w}| + 2n + 2k$$
$$= 2\big((n+1)k + n + k\big)$$

Now, comes the next part, which is called as the operational space complexity, for this network. Now, we take the same example of a fully connected network n number of neurons going down to k number of neurons and this is, network with bias itself. Now, what would come down is that, while you are operating. You have another part of it, which is called as Nabla of w.

So, Nabla of w is basically, the, the derivative of this weights present over there or the derivative of the network itself now the derivative of the network which you come down that is the same size as that of the weight matrix of the network because there is a derivative associated. So, it is basically del del w of y this output over that if this is my this output over here is y and then after this you have your non-linearity.

So, you do not have the non-linearity being seen over here they are not taking down the non-linearity into consideration because that is not a learnable parameter as such yes there is a derivative, but then that is not a learnable parameter and when you are writing down your code till now you have seen that first you just define your connections and then you have your non-linearities within a forward pass being defined over there. So, we choose the same way for doing over here as well.

So, now in, in the same logic, when you are coming down over here; So, if what you get down is if this output over here is some y and this is an x. So, here you are going to compute down for your back propagation del, del, del y, del w over there and that is what is also represented as Nabla w and Nabla net. Now, that has the same size as that of the weight matrix over here and then, you just stick down to n plus 1, times k ok. Now, what that would essentially mean is, if I am going to store this network and operate on top of it.

So, when I downloaded the network, I had just had to download these weights, but then when I want to operate. So, I will push one input over here; get these, intermediate calculations done and an output. So, it means that the memory space consumed by this input the memory space consumed by this output as well as these gradients, which are getting calculated during that provision is also something, which I need to keep in mind and that is where this whole thing comes to play.

So, I have my weight space, the total space consumed by my weight, then the total space being consumed by my, gradient as well over there ok. Now, what comes down interestingly is that all these outputs, which I am producing over here. So, I need, to store the that is k, but then you might come down to an idea, why is this? 2 times number of k and you need to keep one thing in mind that, mapping down these outputs, over there you have also a non-linearity coming down.

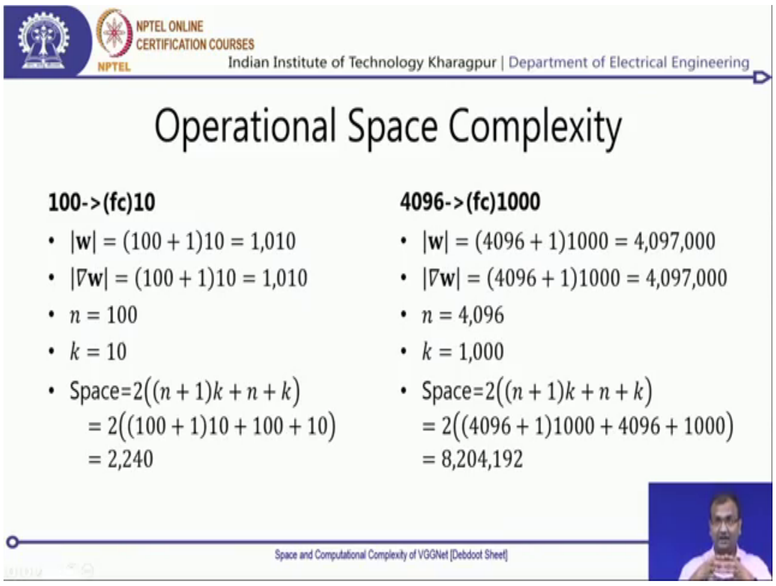
So, that is obviously, not just this, but, but the gradient of the non-linearity, which is also taken into consideration. So, for that I am just getting down 2 k. Now, n is which is corresponding to this input, which comes down over here, but then I also write down 2 n. The reason for getting this 2 n over there is, these are defined in terms of variables.

Now, in, in your pi torch program over there, you are using a package called as autocrat and the whole purpose was to do gradient computation and one of the ways. It was doing is that, it associates certain part of the variable for the actual path value of the variable and then certain part of it is, what is called as a derivative or the back part of the variable. Now, for each of these variables, it is going to do these two parts assignments. So, if there is n number of, bytes associated or n number of elements associated with the forward pass, it will also have a number of them associated with the reverse.

Now, this is not exactly and acquired in all places, but then since, you are building it down modularly with a pointer referencing going down. So, this is inherently an extra space, which it would consume and you need to keep in mind. So, in total this comes down to something like n plus 1 times k which is the size of this weight matrix plus n plus k and just make twice of that.

So, this is a simple cardinal rule, which can give you a rough estimate of the space complexity as well nonetheless. Some non few of these models also employ a compression of spaces over there and, because of this employment of a compression of space. So, what helps it out is that you can actually store it down into much lesser amount of space and, and then keep on continuing. So, that is, that is a good part, which works out as well ok.

(Refer Slide Time: 17:03)



NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Operational Space Complexity

100->(fc)10

- $|\mathbf{w}| = (100+1)10 = 1{,}010$
- $|\nabla\mathbf{w}| = (100+1)10 = 1{,}010$
- $n = 100$
- $k = 10$
- Space=$2\big((n+1)k+n+k\big)$
  $= 2\big((100+1)10+100+10\big)$
  $= 2{,}240$

4096->(fc)1000

- $|\mathbf{w}| = (4096+1)1000 = 4{,}097{,}000$
- $|\nabla\mathbf{w}| = (4096+1)1000 = 4{,}097{,}000$
- $n = 4{,}096$
- $k = 1{,}000$
- Space=$2\big((n+1)k+n+k\big)$
  $= 2\big((4096+1)1000+4096+1000\big)$
  $= 8{,}204{,}192$

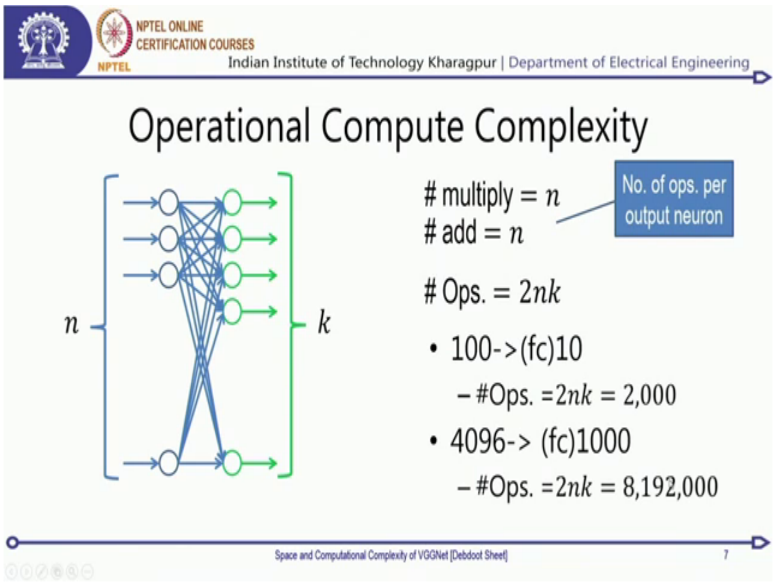
Space and Computational Complexity of VGGNet [Debdoot Sheet]

So, now, let us, get down into a very simple calculation. So, if I have 100 neurons, which are connected down to 10 neurons, then this is what the whole network would be looking down. So, you have 100 neurons connected down to 10 neurons. So, you are, w is something which is 1000 and 10 and then your Nabla w is also 1000 and 10. So, in total this comes down to a space of 2240 elements present over there.

Now, let us change this one and come down to connect down 4096 neurons onto 10000 neurons. Now, you would see that the moment I changed this space and then make more number of neurons, something which was in the range of 2 point, 2000 in some way 2240, suddenly goes up to 8 million.

So, this is a big realization. So, if you are changing down the number of neurons, it does increase in a significant way and this was a basic motivation, which led to the development of convolutional connections as such in order to reduce all of these problems.

(Refer Slide Time: 18:04)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
n
k
No. of ops. per output neuron
Space and Computational Complexity of VGGNet [Debdoot Sheet]
7

Operational Compute Complexity

$\#$ multiply $= n$
$\#$ add $= n$

$\#$ Ops. $= 2nk$

- 100->(fc)10
  - #Ops. $=2nk = 2{,}000$
- 4096-> (fc)1000
  - #Ops. $=2nk = 8{,}192{,}000$

Now, on the other side of it, let us look into the operational compute complexity, which is to look into, what is the total number of floating point operations number of multiplications, number of additions, which would be doing down. So, now, over here, what would come down is that you have, n number of neurons and then for each of these output over here. So, if I am considering only the first neuron over here.

Now, I will have n number of weights, which are multiplied by n number of inputs over there. So, in total there are n number of such multiplications, which will happen. So, this is just weight associated for one of these output neurons, which each of these input neurons. So, that is, that is the total number of multiplications, which will happen.

Now, once I multiply it out, what I need to do over here is I have to add. So, the result of this multiplication is still an array of size m. Now, if I want to add an array add, all the elements in an array of size n then I will have to perform n minus 1 number of additions over there ok, plain and simple; So, basically if I, do this dot product over there. So, there are n number of multiplies and n plus and n minus 1 number of additions, which will take place. Now, you need to keep in mind that there is also a bias associated with each of them and there is a plus 1 extra addition, which comes down for that bias.

So, in total n minus 1 number of additions for, this dot product over there and another addition for the extra bias, which comes into it and that makes it n number of additions as such. So, there are 2 n number of operations, which I am going to do over here, per neuron ok. Now, I have k number of such neurons with which for and for each of them there is a 2 n number of operations. So, in total, the total number of operations over here goes down as 2 n k. Now, you might come back and say that multiplies and not always vector implemented.

So, they are not one machine cycle operations, but sometimes, they are recursive adds or different ways of doing it. Now, for most of our modern computers multiplies are itself a block, level operation. So, there is a separate, hardware unit dedicated over there within your CPU, which can do one machine cycle multiply. The same way it can do one machine cycle add except for the vectorization is a significant part. So, it might just be able to multiply two numbers, but in the same machine cycle. It can add down 8 numbers, 8 integer type numbers whereas, it will be able to just multiply 2 integer.

So, this is, this is; obviously, a difference, which comes down in terms of checking the number of, operators, which it would require, but nonetheless this is a very simple, way of finding out. What is the total number of compute complexity in terms of my operator. So, over here I would require 2 n k number of operations going down ok. Now, let us take this example of connecting down 100 neurons to just 10 neurons, in that case my total number of operations, which comes down is just 2000 ok.

Now, say that I have a machine, which is running down and it can do, something like 200000 operations per second. So, what that would mean is if I have 2000 operations to take place over here, for, for this kind of a neuron, which 100 neurons to 10 neurons and my machine can actually do 200000 of operations over there. So, it technically means that I can operate on 100 samples per second. So, this is where it comes down.

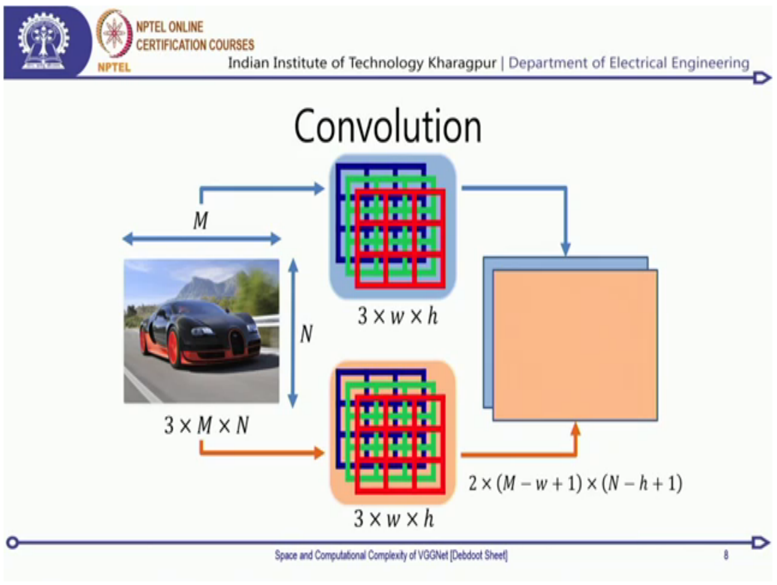
So, my machine will have the , if my machine has a capacity of 200000 operations per second or simply 200 kilo operations per second and over here, I would need just 2 kilo operation, operations to do. So, that is operations per second and here, it is 2 kilo operations, then I can do actually 100, such operations per second. So, that is 100, such neural computes per second. So, that is, where we come down to the fact of, if you know exactly what is the number of flops or floating point operations or number of ops even, when that is also valid.

We just computed in terms of floating point operations, because certain machines have, more time consumption for floating point as compared to just fixed point operations over there. So, if you know exactly, how many floating button operations are there and how many floating point operations your machine can do. So, you know per second how many; such data points can be processed out easily ok. Now, if we have 1096 neurons connected down to 1000 neurons, then the total number of operations comes, up to something like 8 million hundred and 92000.

Now, this is a significant number. So, if I still have my 2 kilo operations per second, thing over there and I want to process this one over there; So, that would mean that, sorry if, if I have my system, which has 200 kilo operations per second, but here I have, almost, 81 kilo operations to be taking place, 82 kilo operations and my system is what has, 200 kilo operations per second. So, that would mean that almost 40 times more. So, this would, this itself would be taking down about 40 seconds, in order to compute it up. So, this is where it changes.

So, if you have a bulkier model more number of neurons yes, you are going down, wider deeper everything is great, but then you are also going to take a lot of time. So, this is what you need to keep in mind, based on what is your target. So, whether you are using a simple CPU or your going on a GPU 1 and finally, for your deployments, maybe like for training, you have bigger resources available, but while you are influencing or putting it down to an actual, work, you will have to consider these significantly over them ok. So, this is what goes down with my fully connected neurons.

(Refer Slide Time: 23:35)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolution

Space and Computational Complexity of VGGNet [Debdoot Sheet]

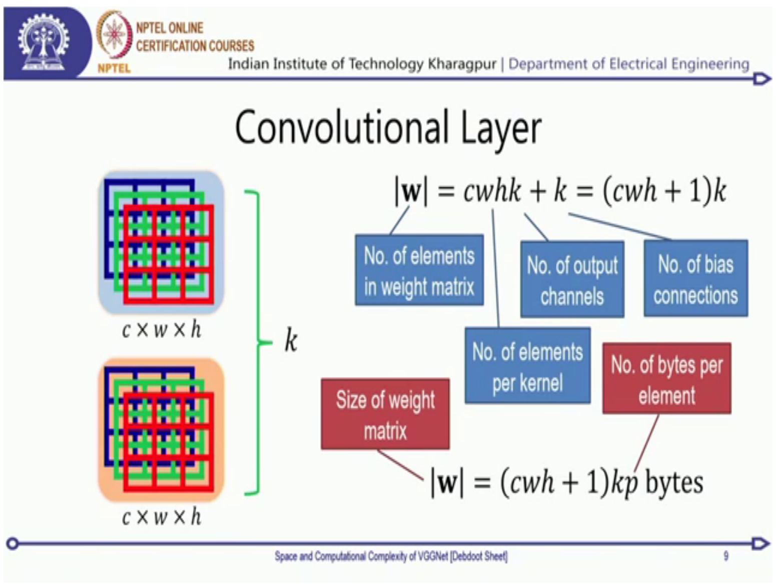
8

The next, which comes down is actually my convolutional neurons and these are quite interesting. So, what typically happens; let us, do a very brief revision of this convolution 1. So, our idea was that if I have a color image over there, which has 3 cross, m cross, n and 3 is the number of channels.

So, in the later on examples, I would replace this 3, we see. So, that you can take down the same kind of a logic for subsequently down the line for any kind of any depth of c n and over there and then, my total number of channels in my convolutions will be equal to the total number of channels in my input over there and then this becomes one of my kernels.

Now, if I convolve over, there I get one of these outputs over there. I can take another kernel, which also has the same number of channels and same width and height and I get done my next output over there and the size of this output is. So, the total number of kernels, which I have is 2. I can have k number of kernels, in that case this becomes as cray k cross m minus w plus 1 cross n minus h plus 1.

(Refer Slide Time: 24:34)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Convolutional Layer

$|\mathbf{w}| = cwhk + k = (cwh + 1)k$

No. of elements in weight matrix

No. of output channels

No. of bias connections

No. of elements per kernel

No. of bytes per element

Size of weight matrix

$|\mathbf{w}| = (cwh + 1)kp$ bytes

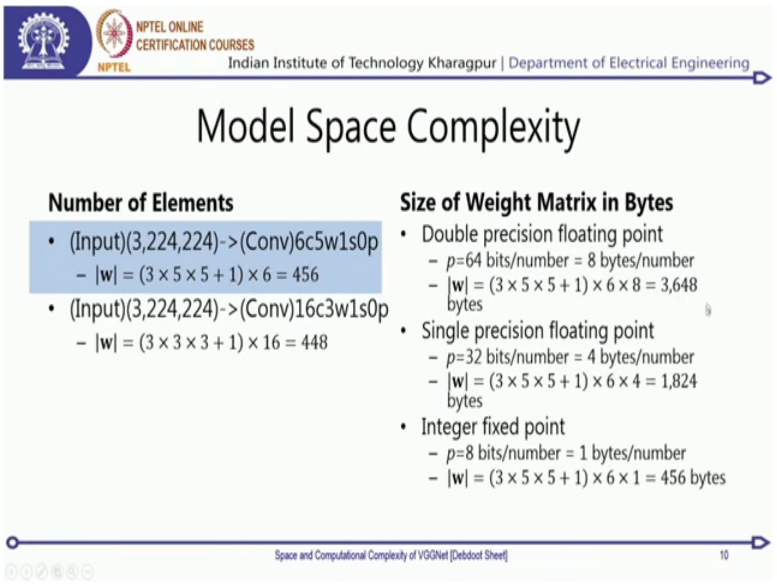
Space and Computational Complexity of VGGNet [Debdoot Sheet] 9

Now, let us look into the convolution layer, a bit more in depth. So, if this has c cross w across h number of elements present over there ok, then the cardinality is pretty simple. It is c cross w cross h. Now, it, it goes out pretty easy and without much of an issue, but then, what would come down is that, if I have k number of such kernels present over there then I also need to consider one part of it that, there will be k number of, output channels. So, there is c into w into h into k total number of elements present down in all of these kernels.

Now, along with each of these kernels over there, there is also a bias connection present over there and this bias connection is which is very unique to each of these channels. So, there are k number of bias connections each associated with k number of channels. So, that makes this one as c w h plus 1 into k. Now, if you get back into your fully connected neural network and try to look into it. So, you remember that, if I had n number of neurons over here, in the input and k number of neurons on the output then the c w h is basically, equal to n over there. So, it becomes n plus 1 times k.

So, that is that is pretty straightforward over there now, in the same logic, if we get down into understanding that if I have p number of bytes, per number being represented, then my total cardinality or the total number of elements over that comes down as c w h plus 1 times k p bytes and this is the total memory being consumed by one of these convolutional layers.

(Refer Slide Time: 26:06)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Model Space Complexity

Number of Elements

- (Input)(3,224,224)->(Conv)6c5w1s0p
  - $|w| = (3 \times 5 \times 5 + 1) \times 6 = 456$
- (Input)(3,224,224)->(Conv)16c3w1s0p
  - $|w| = (3 \times 3 \times 3 + 1) \times 16 = 448$

Size of Weight Matrix in Bytes

- Double precision floating point
  - p=64 bits/number = 8 bytes/number
  - $|w| = (3 \times 5 \times 5 + 1) \times 6 \times 8 = 3{,}648$ bytes
- Single precision floating point
  - p=32 bits/number = 4 bytes/number
  - $|w| = (3 \times 5 \times 5 + 1) \times 6 \times 4 = 1{,}824$ bytes
- Integer fixed point
  - p=8 bits/number = 1 bytes/number
  - $|w| = (3 \times 5 \times 5 + 1) \times 6 \times 1 = 456$ bytes

Space and Computational Complexity of VGGNet [Debdoot Sheet] 10

Now, let us get into the model complexity estimation. So, I will take down, these two very standard examples. So, one of these examples is where I have a 2 2 4 plus 2 2 4 plus 3 channel input, connected down and this goes into a convolution with 6 convolutional kernels, 6 convolutional channels each of 5 cross 5 with a side of 1 and a product of 0. So, this is what I had described in the earlier case. Now, if I go down by my same logic, which is number of input channels, which is c that is equal to three w is the width of the kernel, which comes down from here, h is the height of the kernel, which also comes down from here.

So, these are isotropic kernels which I am taking. So, this becomes 3 into 5 into 5 plus 1 into 6, which is the total number of channels or the total number of kernels, which are present, for my convolution. So, that makes this one come down to 400 and 56. Now, on the other side of it, if I change this one, I make a small, change over here and use just 3 cross 3 sized kernels over there; now, if I make this change and then I increase my total number of.

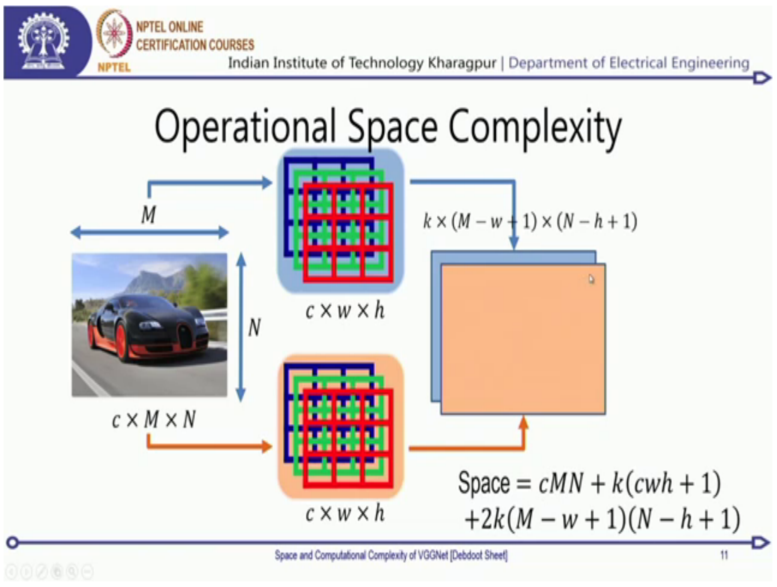
So, what I did is I reduced it from 5 cross 5 to 3 cross 3 and then I increase my total number of channels to 16. Now, in that case you would see that the total size of this weight comes down to 448.

So, you see that total number of elements is almost of the same size. It does not change much, but then, we will have some interesting twists coming down later on. Now, let us look into the precision of this number system and how it will impact it. Now, that is pretty simple. In the earlier case we had seen with a double precision number system that you have 8 bytes per number and accordingly, going down by this logic.

So, we will have 3.62 0.65 kilo bytes coming down, if I use a double precision number system, if I change that and come down to a single precision number system. I am going to make use of 1.8 kilo bytes, for this total weights being stored down over there. If I change down and go down to something called as an integer position, our characters region which is one byte per number.

So, that comes down to 456 or 0.5 roughly, 0.45 bytes, 0.45 kilo bytes over there. Now, this is definitely a big change, which comes down and one thing, but you can, note down from here is that even, if you have more number of neurons and, and larger descriptive connections coming down, you still, your total, size of the weight matrix is much smaller that is not large over there ok, but then we also need to understand the operational space complexity over there ok.

(Refer Slide Time: 28:34)


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

Operational Space Complexity

M
N
$c \times M \times N$
$c \times w \times h$
$c \times w \times h$
$k \times (M - w + 1) \times (N - h + 1)$

$$\text{Space} = cMN + k(cwh + 1) + 2k(M - w + 1)(N - h + 1)$$

Space and Computational Complexity of VGGNet [Debdoot Sheet]
11

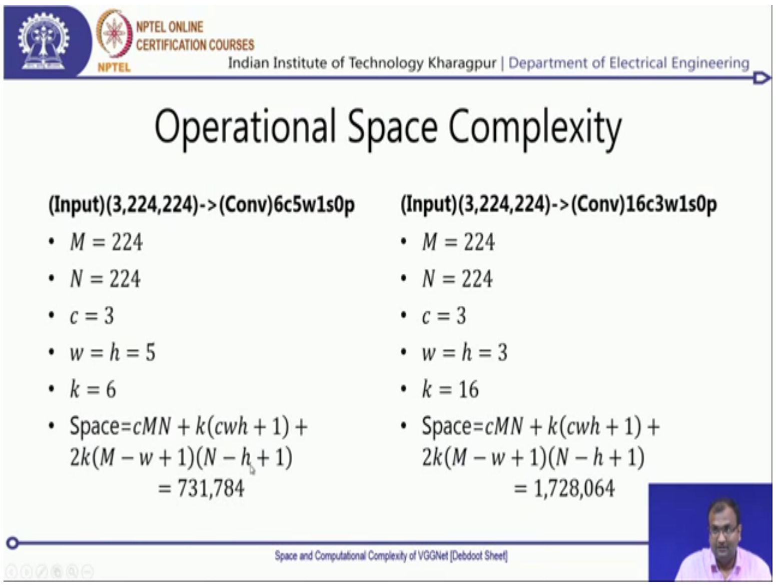
Now, for the operational space, complexity you would need some space to store this part you would need some space to store this part. This part as well as the derivatives, which are getting computed over there; Now, what that comes down is that c cross m cross m c into m into n. There is the total, space required to store. This 1 then c w h plus 1 times of k is the total weight, is a total space required to store down; These weights over here ok. Now, what happens over here is that quite contrary to the fully connected network, in the fully connected network.

What you had is, you had total k number of outputs being generated and you had with each of them the gradient being stored down over there ok. Now, here what you do is you do not have these gradients, stored down in terms of any moors linearized neurons, but in terms of the neurons, which is existing on this space. Now, that is the feedback, the backward or the Nabla part of it, which gets generated.

So, that technically means, you are going to anyways require just k into m minus 1 m minus w plus 1 into n minus h plus 1 number of space, but then, because you have to store down your backward or the gradient operator over there, you are going to require twice of that.

So, this is where the change comes in to, in terms of operational space complexity, for a convolutional neural network, and that brings us to the point that, if we look down with just one simple example, the first example, which we had taken down. Now, your total space taken down to operate it, out is significantly large.

(Refer Slide Time: 29:55)


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

Operational Space Complexity

(Input)(3,224,224)->(Conv)6c5w1s0p

- $M = 224$
- $N = 224$
- $c = 3$
- $w = h = 5$
- $k = 6$
- Space$=cMN + k(cwh + 1) + 2k(M - w + 1)(N - h + 1)$ $= 731{,}784$

(Input)(3,224,224)->(Conv)16c3w1s0p

- $M = 224$
- $N = 224$
- $c = 3$
- $w = h = 3$
- $k = 16$
- Space$=cMN + k(cwh + 1) + 2k(M - w + 1)(N - h + 1)$ $= 1{,}728{,}064$

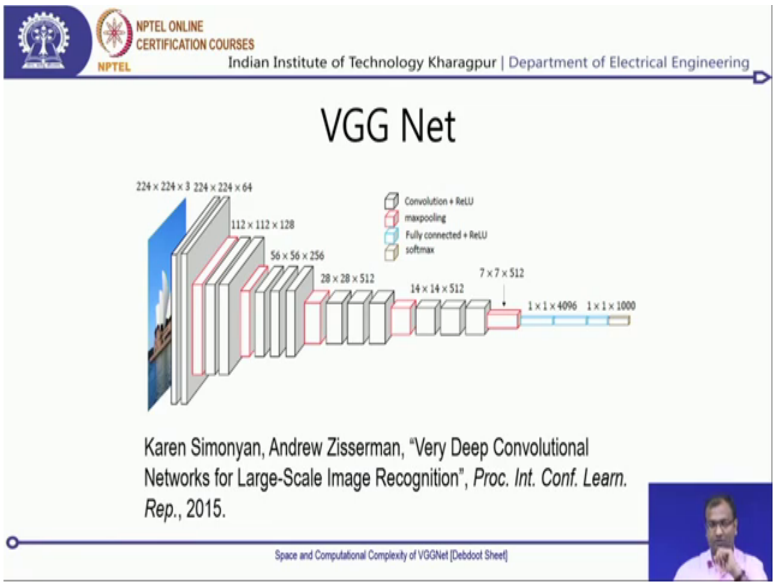
Space and Computational Complexity of VGGNet [Debdoot Sheet]

Now, while your, total number of bytes in order to store the weight was less, but then this total space or the total number of elements, you would be storing down, while you are operating on this, one both forward and backward is much larger and that is almost like 731000, 732000 hundred and approximately. Now, if we change it and go down to the other model, which is 16 channel and the 3 was the 3 cross 3 was the size of these kernels now you look down that typically between moving from this kernel on to this kernel we technically did not change much of the weights.

In fact, we reduce, but slightly we had just liked to reduce the total number of elements; however, if you look into the total operational space you see that this model consumes almost one million more parameters than this model over here that is something significant change which you need to keep in mind.

So, this is what would happen now. So, if you are just rapidly changing I mean do not try to do that please keep in mind as to what is the space being taken down and whether it would be working or not. So, now, you will get an understanding if when as we go into deeper and deeper models as to where the change will happen ok.

(Refer Slide Time: 31:15)


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.

Space and Computational Complexity of VGGNet [Debdoot Sheet]

So, now comes down my VGG net over there, know at this point, I would leave down 1 1 part to all of you to ponder up, one picture will come down at the later point of time for convolutions. I have not yet done on the computational complexity, for the model and the reason that I have not done the computational complexity for the model is that I am going to do it a bit later on.

Once we are done with even deeper networks. So, that is where I will come down into what is the total compute requirement, what is the space requirement and how we are going to play around with that 1 ok.

So, nonetheless, so this was the simple VGG net and as we had done it over there; You had seen that for each of these models, I had mentioned down that the total number of parameters, but then one thing we did not, quite clearly clarify over, there is how you come to this number of parameters.

Now, number of parameters is equal to just the size of the weight matrix. So, you need to compute, what is the size of the weight matrix for each of these layers and sum it up and then you would be coming up to that. So, for this one I leave it up to you, to do an exercise.

In fact, we have some interesting problems and assignments and assessments as well for each of these newer networks, which we are going to come down on, we would be looking into parameter space detailed completely unrolling out the model and then coming up with what is the total parameter requirement over there.

So, you, you get to know even more of details over there and this is the way of how we compute exactly and come up over there. So, as you, if , if you are going through the get up codes over, then you will find out the little calculation also given down, of how we are doing it. So, with that we come to an end and finally, if you want to get more details over here. So, just refer down to the textbook that is one of the best sources to actually get on.

So, stay tuned, till we get back later.