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

Lecture - 42
Assessing the Space and Computational Complexity of Very Deep CNNs

Welcome to today's lecture. So, today we are going to speak very much on some of these practical issues and practical aspects and, in fact I would be showing you a very detailed case study which we had run down on one particular model, one particular deep neural network on multiple kinds of hardware units over there to really come and, compare where the stand against each other ok.

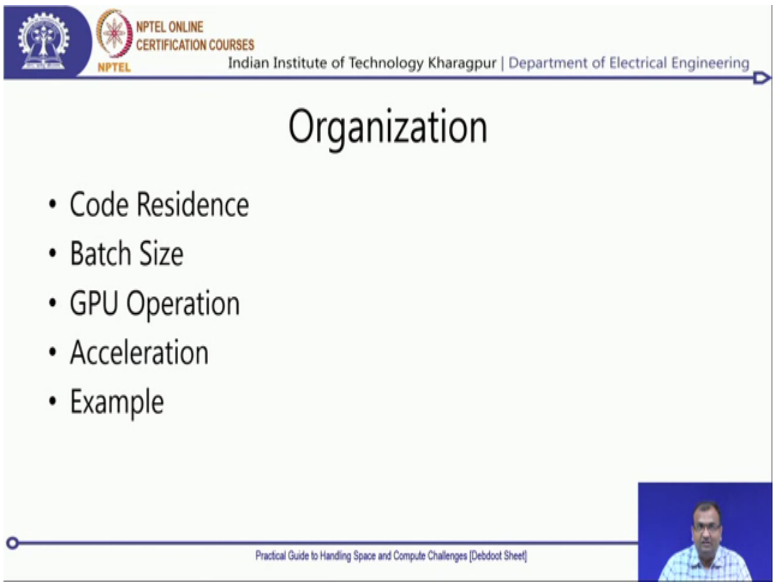
So, what I am going to do is there is going to be a bit of revision from some things, which you had learned way back in your undergrad possibly and, that that is about the computer organization and architecture. So, if they are current students who are in the second year or third year I am not quite sure. If you are doing these if you are doing this particular one this particular lecture series, then you might be studying those at this point of time as well.

Now, computer organization and architecture is quite an important subject, which I keep on telling people from the engineering aspect of whether you are machine learning algorithms, you are AI systems would actually be working out pretty fine or not. So, today what I am going to do is in the last lecture we had done a bit of theoretical analysis on what is the total space required, when just storing your model, what is the total space required when you are trying to do a forward pass and a backward over the model.

So, that that also has a dependency on the data size over there and, then you also discuss about what is the total number of operations which will happen, when you are trying to operate anything on the model. Now, today what we are going to do is on the aspect of that given that I have found out all of these criteria over there what is my total number of operations which is going to happen what is the total space.

I require in bytes to store it down, then what will happen on my side of the machine and, then can I keep on changing this number system precision over there to change something and, what kind of hard ways would allow me to have a variable precision number system and, which of them will not allow me to have that one. So, these are stuff which we are going to really look into while working out today.

(Refer Slide Time: 02:14)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Organization

- Code Residence
- Batch Size
- GPU Operation
- Acceleration
- Example

Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet]

So, today's organization is something of this one that I will initially start with something called as the code residence. So, code residence so, these are not concepts from machine learning as such, but these are engineering aspects about getting these AI systems working down.

So, code residence is like, if you have a piece of code with you then does it reside on your RAM, does it reside on the hard drive and does it reside on your CPU cache and, how do these operations have a very significant implication on how fast you are going to really work it out.

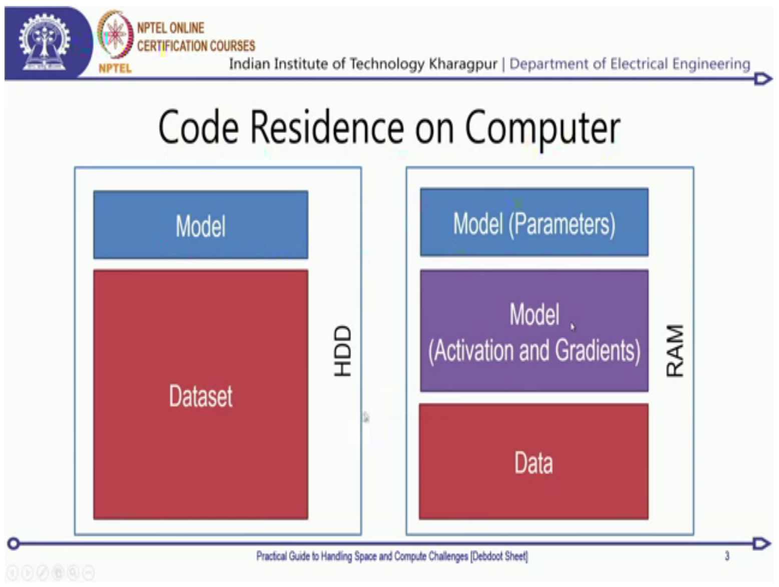
The next part is to come down to a very practical way of finding out your batch size. Now we have been doing all of our update equations, update rules and everything now, under an assumption of a batch size. So, we just assume that this batch size is what is working out good, but then while I was doing it you had realized that there was a different batch size I was using for different kind of compute mechanisms.

So, whether I was using a GoogLeNet, or a VGG Net, or a ResNet, or a DenseNet my batch size was always changing out over there and, what I did specify is that we try to pick up a batch size such that that it fits perfectly within the RAM it does not overflow as well as it does not under consume the GPU RAM on which I am working it out. Now however, I had not actually given an explanation at that point of time how to go on choosing around with this one and how, does it impact like whether your model is big or small how much is your did your operational space complexity and, how is it going to impact your batch size over there and, whether you can take down a decent sized batch or not.

So, this is where we are going to discuss it out exactly on how it impacts my batch sizes. Now then I have a GPU operation and what does come down on a GPU operation, which means that if I have something which I am working down traditionally on my CPU versus, if I want to work it out on a GPU, then transporting things from a CPU side onto a GPU side what is the total amount of change which is going on.

And what are the hardware units which are being invoked and whether there is some sort of bottleneck which might come down, because of certain features which are not supported, within my computer system, or within my motherboard itself ok, then I would come down to acceleration and how and what acceleration is it and how it works out and, then we come down to a very practical example for case study ok.

(Refer Slide Time: 04:33)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Code Residence on Computer

Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet] 3

So, now for the first part of it which is to understand code residence on my computer; ok. Now, typically what I have on my PC is a hard drive. So, when you are defining you are writing down some piece of code and, then keeping it down or you download a model.

So, what we were doing down initially is that you have your these header files which you had called it out. So, that was loading onto your RAM from the hard drive ok.

Next you define your network over there, or you downloaded your network over there, now when you download your network as well as when you are downloading your data you see that everything was getting stored on your hard drive over there and there was a some root slash path being defined, if your Linux user and if you are a windows user then you have some drive letter colon slash and then your path definition which comes down over there.

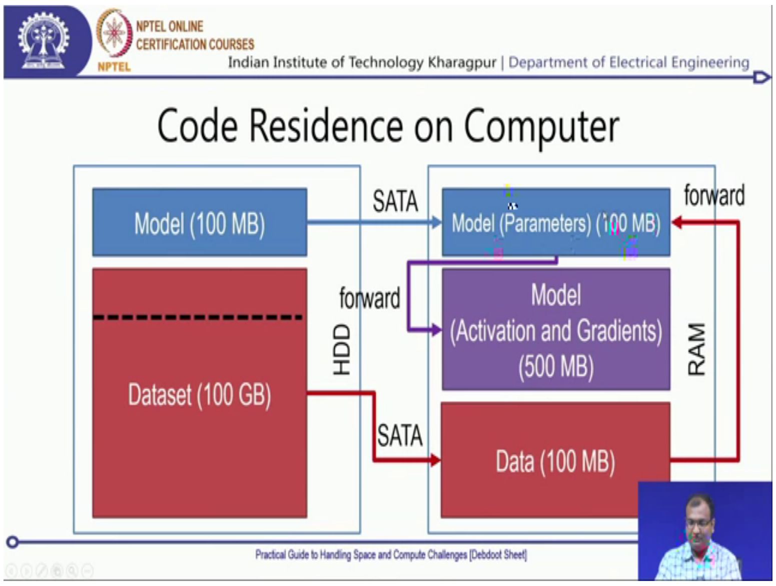
So, this is a part of the code which and then the data, which actually resides on your hard drive over there, now what for us would be that that I have my model and, I have my data set which together reside on my hard drive ok. Now, when I need to work it out on my RAM, then what I would be doing is, I will have my model which consists of my parameters over there, then I will have my model which will consist of my activation and gradients. Now, when we were looking into operational space complexity we figured out two things.

So, operational space complexity is typically greater than the model space complexity and, the reason it is greater is because you have the model space, which is the weights and everything taken down over there, as well as you have this activations and the gradient of these activations which are also stored down and, that is what I am putting down in this extra block over here, which is just activations and gradients ok.

Now, what you need to keep in mind is that you cannot load the whole dataset onto your RAM and typically we were doing it in batches, which was a chunk of the data which was getting loaded on to my RAM. And this activations and gradients are what correspond to this data which is loaded on to my RAM and, that is how it is it is going to perfectly work it out over there.

Now, this is typically an arrangement of how things would reside on your hard drive, where versus how things would reside on your RAM and the data which resides on your RAM is just a small part of the data set, which resides on your hard drive over there ok.

(Refer Slide Time: 06:51)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Code Residence on Computer
Model (100 MB)
Dataset (100 GB)
HDD
SATA
forward
SATA
Model (Parameters) (100 MB)
Model (Activation and Gradients) (500 MB)
Data (100 MB)
RAM
forward
Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet]

Now, let us look into very typical case scenario over their with certain numbers ok. Now, my hard drive I have my RAM both present with me, then say my model which is of size 100 MB ok. Now, when I am trying to load a model on to RAM, which is I define this command over there net is equal to torch dot load and then my path is given down over there. So, what it essentially does is that there is a connection via a serial ATA.

So, there is a SATA cable connection between your hard drive. So, it will be a thin colour thin blue coloured, or a black coloured cable you would see, which connects from your hard drive onto your motherboard and, then on the motherboard there is a separate driver unit for this one which is going to interface this protocol of SATA on to your RAM bus over there which connects it back on to the RAM and then, why IDMA transfer you can do all of this now.

This part over then if you if you look into your hard drive over there. So, your hard drives data transfer capacity is much lesser than your rams data transfer capacity significantly lesser I mean your RAM can transfer something in the range of gigabits per second whereas, your hard drive will be something in the order of megabits per second. So, this part is a slower part and, you would see that takes some time to load.

So, if you are copying down some model from one part to the another part of the RAM it would be significantly faster. So, you can have like just do a deep copy over there. So, deep copies are much faster over there because, they are just copying down on to your

RAM whereas, when you are doing a model copy then it is much slower because your hard drive is closed.

So, this is what will happen and this is this limitation is something which you need to keep in mind as well ok. Now, the next part is that you have your data set. So, typically your data set would be about 100 GB over there, some 100s of GB very practical data set.

So, we are not looking at say c (Refer Time: 08:51) 10 kind of data sets which are rather smaller 100 of MBs over there, but then if you look into the complete image net data set, or if you are in that subsequent ones when we will get into temporal deep learning. So, we will be understanding more about what happens down with see video datasets and, then they are really massive ok. Now, if I have a dataset which is about 100 GB over here, then I cannot load everything onto my RAM because, typically my CPU RAMs are.

So, most of your machines will be about 8 GB of RAM or 16 GB of RAM maybe 32 64 GB of RAM, I mean there will be few machines which will have a 100 GB of RAMs I mean that there is one back in our lab which has 128 GB of RAM addresses that is one of the major massive machines, where for a particular purpose we had to load the complete data set over there, but then most of our data sets are not 100 GB they are something which are in the order of multiples of 100 GB, or close to actually a terabyte of so in fact, we cannot also load down most of those data in one single shot over.

So, the idea why here is that you would be chunking a part of this data set over there and put down onto your RAM and, say that you are putting down some 100 MB of data at a point into my RAM. Now when you are writing your piece of codes over there, you had the data loader function and then you had something called as a batch size on the data loader and you had a number of workers being defined number of parallel channels per data loader.

So, what it was essentially doing is that every data loader just loads down a chunk of this data reads a particular part of the file on the hard drive and, then saves it out on to your RAM. And this connection is also via SATA because, that is the only way your hard drive connects down to your RAM and this is a slow process.

So, what we do by putting down parallel workers over there is that we instantiate a number of parallel threads of this process. So, is that while one data is getting used via this model and there is some processing going on, there are other channels which are coming over there and SATA as such is not just one single part.

So, it is not like you can access only one region over there, but you can access multiple points on the hard drive coming down together and, then there is a limit on this one. So, how many channels your hard drive supports and how many channels your interface of the SATA controller over here supports it is a trade off between that.

So, if your hard drive may support say 8 SATA channels over there, but your interface controller on the motherboard might support just 4, then you can transmit maximum 4, if you just write down 8 as a number of parallel workers on software you defining it within your hardware does not support that feature in any way, on the other side of it your motherboard may have 8 SATA channels, but your hard drive might have just 4 you are still limited by 4 because, defining more than that is not going to help you out in any way ok.

Now, what comes on subsequently after that is that once I have this loaded down. Next what I am going to do is I do a forward operation over there. Now, typically in my forward operation what comes down is that the data passes from this part of the RAM and, goes done as a feed forward through this model.

Now, for each sample point over there it is going to calculate an activation and, also associate a gradient vector a container for the gradient, you do not get the gradient calculate and until you do a backward operation. So, that is not of major concern as of now, but the space allocation is definitely over there ok. Now, you do a forward you pass it through the model for each single sample from your batch over here, which forwards through it, you will get down the activation and the gradients calculated and say that this is in the order of some 500 MB.

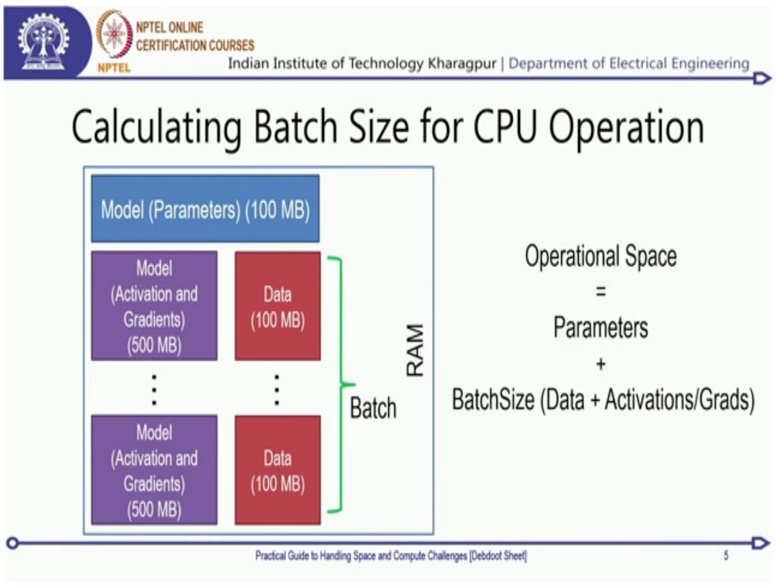
So, if I have 100 MB of input data going down it generates about 500 MB of this space coming down over here. Now, you can typically look down at this point of time on my RAM I am consuming about 700 MB of space and, this is just for a data of 100 MB and a model which is of 100 MB size. Now, if I change this 100 MB and go down to about say 1000 MB.

Now, based on that this space over here we change over from 500 MB to 5000 MB; So, that would mean that this is about 5 GB this is 1 GB. So, that makes it 6 GB plus another 100 MB over here. So, that makes it 6.1 GB of space which I consume on the RAM. Now my model my stationary model over there, we just has parameters that space is what is a constant fact, that does not get multiplied or anything over there and then this is what you need to keep in mind.

So, your model might be smaller in size and still you will not be able to give down a very large batch size because, this activation plus gradient is what is something which is going to consume significant part of your RAM over there and, that is a trade off which will have to calculate it out.

So, since we had already done pen paper calculations of how to do it for one layer at a time; Now, if you repeat it down for all the layers you will actually be able to get down, you're pay total space being consumed for your activation ingredients over there ok.

(Refer Slide Time: 13:43)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Calculating Batch Size for CPU Operation

Model (Parameters) (100 MB)
Model (Activation and Gradients) (500 MB)
Data (100 MB)
Model (Activation and Gradients) (500 MB)
Data (100 MB)
Batch
RAM

Operational Space = Parameters + BatchSize (Data + Activations/Grads)

Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet] 5

Now, say that how do we calculate out the batch size, now this is critical because this is where we were struck down in the earlier one as to how do we actually estimate out batch size and is there some empirical rule of doing it.

Now, what I said is that there is no straightforward empirical rule, it is something which is dependent on that you will have to actually find out what is the total space being taken down while the operation is going on and it pretty much depends on that. And now what we are going to look down is typically on to your CPU operation, which means that you do not have a GPU on your system your model is not type casted on to CUDA.

So, there is no acceleration that is pure CPU operations which are going on, maybe there is an acceleration on to your CPU which we will see in a later on point over there where we had some multicore CPUs available with us and, we used a very special precompiled library for python which supported multiple multithreading onto it and multiple pipelines, being created for acceleration as well.

So, that is what we will be doing on, but as of now let us consider, it just as plain simple thread one single thread CPU operation going on. So, now I have my RAM available to me ok, now what I would do on my RAM is that I have some 100 MB of parameters which is available so, this was the earlier case which we had looked on. So, my model space for parameters is about 100 MB ok.

Now, what I do is I load a batch of data from my hard drive over there and, say that I have this each sample is about 100 MB and then I load a complete batch over there ok. So, there will be some batch size into 100 MB is the total data size which comes down over here ok. Now, for each data point what would happen is that I would get down 1 model activation. So, for each single sample I get down an activation and ingredient. And similarly it keeps on going and then at the end of the batch I have this one. So, this is for my forward and then my backward and then I have this one.

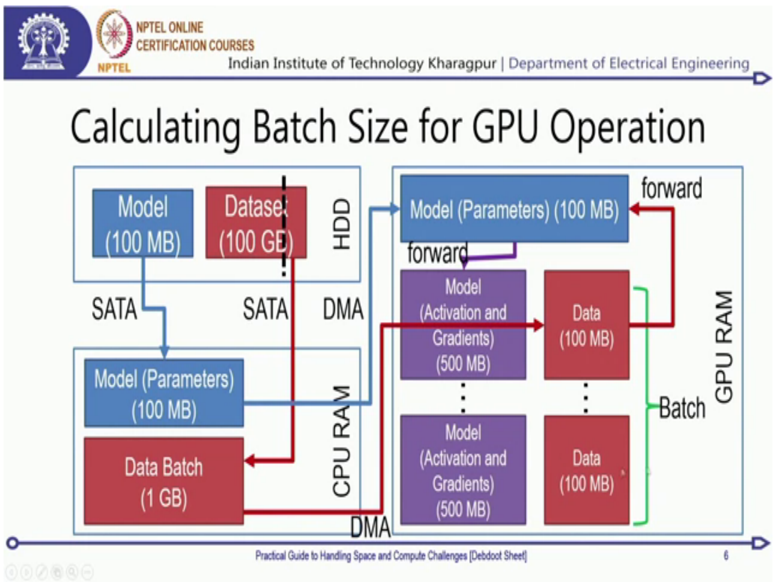
So, now what that makes me come down is that my operational space as I would need is a total space consumed by my parameters plus, batch size times my data size plus my activations and gradient size over there. So, for me it is something like this that my batch size my data size is 100 MB my activation and gradients is 500 MB. So, this makes it 600 MB. And now say that I have a batch size of 10. So, it is 600 MB into 10 which makes it 6 GB over here and, I have 100 MB of model space over here.

So, my total operational space requirement is 6.1 GB as of now. Now, typically if your CPU system has about 8 GB of RAM over then, you would see that your system operations and your operating system itself is going to consume roughly about say 1 to 1 and a half GB of space over there. So, you are left down with merely 6.5 something like 6.5 to 7 GB of space and, now if we run this one with a batch size of 10, then I am already consuming 6.1 GB and possibly that is the maximum I could consume and, typically I do not suggest that go down more than that.

So, at any point of time any single program which you are running, that is a cardinal rule we do not load it more than 80 percent of the space on the RAM, if that is the case then what happens is that it needs to initiate a paging operation over there. So, it will start writing down to the page files and that will even make it slower and slower for you yes, you need to leave down some space for the system operations to keep on going down over there.

So, in this kind of a scenario a batch size of 10 for a system for a CPU system which has 8 GB of RAM is something, which is ideal and this is a practical way in which you can calculate it out.

(Refer Slide Time: 17:26)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Calculating Batch Size for GPU Operation
Model (100 MB)
Dataset (100 GB)
HDD
SATA
SATA
DMA
Model (Parameters) (100 MB)
Data Batch (1 GB)
CPU RAM
DMA
Model (Parameters) (100 MB)
forward
forward
Model (Activation and Gradients) (500 MB)
Data (100 MB)
Model (Activation and Gradients) (500 MB)
Data (100 MB)
Batch
GPU RAM
Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet]
6

Now, on the next side is let us look into the batch size for a GPU operation over there. Now say that I have my hard drive, on which I have my model which is of 100 MB and, then I have my data set which is of 100 GB over there ok.

Now, typically what I would do is that first I would copy down this model on to my CPU RAM. Now, there are two aspects which you need to keep in mind. So, your GPU is a separate card which sits on top of your PCI slot. So, it is a separate PCI card which sits on the PCI slot and it is connected just by these PCI buses and PCI directly does not interface onto your RAM and, the CPU and under say the processing units within a GPU, they do not directly access the RAM present on your CPU.

So, your CPU might have 64 GB of RAM that does not mean that your GPU is able to do it. So, your GPU might have it is own 2 GB of RAM, or say 8 GB of RAM or 11 GIGS of RAM typical numbers. Now, you have GPU with about 16 GB of RAM to even massive ones like 24 GB of RAM it depends on what is the use and what is the make you are doing the more you go on that RAM side over there, the memory management and the cost associated with those high density and heavy speed rams is what increases the system cost as well.

Now, typically when you have your operation going down over there, the first part is where you copy down your model, which is you do a torch dot load for the model and; that means, that you have copied it down to your CPU it still does not reside on the GPU this is what you need to keep in mind ok. Next what you would do is that you cannot copy down the whole data set because it is about 100 GB over there. So, what you can do is you can copy one batch of the data. So, maybe 1 1 GB is a batch of the data which you have copied down on to your CPU RAM.

Now, if your CPU has larger size memory over there, now I have my the advantages over here is that in the earlier case, when we were doing it on the CPU system over there, you had this additional space of gradients which were stored down on to your CPU RAM itself, but then here my compute is not running on the CPU my compute is running on the GPU.

So, my activations and gradients are something which store which get stored on the GPU and not. So, on the CPU now on the other side of it, I have my GPU RAM available to me now when I do this GPU RAM call over here what comes down as this model is actually stored on to my GPU and under typical conditions, if I am using the same number system precision.

So, if my model was stored as a floating point number system on my hard drive and, it is imported as a floating point number system of 64 built floating one number system on my CPU RAM it will straighten the same amount of space it would think and, then when I convert it as a dot CUDA. So, I do a net dot cuda which is my typecasting. Now that is typecasting operator is which copies all of this on to my GPU memory over there. So, it will also be taking the same amount of space and this copy is via a DMA transfer which is much faster than a SATA transfer and, for that reason you will see that while it takes longer to load a model from hard drive, it would take much lesser time in order to just do a typecasting conversion.

Now, from your earlier experiences of data structure which you had then engineering, you might consider that typecasting over there is just changing the resolution of an array, but over here this typecasting operator is which move something physically from one part of the RAM to another system device which has a different RAM of it is own ok.

And the model is to reside on the GPU RAM, if it resides on the CPU RAM it does not work because, the GPU processors do not have direct interfaces and memory connections on to your CPU RAM. It just can work if the data is present, or the operators are present only on the GPU RAM over there ok. Now, from there next what I do is I have my data loader. Now if within my data loader what I was doing is if GPU is available, then I always used to do a typecasting of the data, which means that there was a say data dot cuda, or input dot cuda input dot cuda open and close brackets.

Now, what that did essentially was at this data, which I had on my hard drive over here from my hard drive, which I had chunked by other data loader and kept on the CPU RAM, I am converting all of that onto my GPU RAM space. So, I am just typecasting it and that initiates a DMA transfer via which it does and, because it is a DMA transfer and on the motherboard itself.

So, that is much faster than a SATA load over there and as what takes down. So, this is how it is divided on to my batch over there. So, I have 1 GB of data which basically is a batch size of 10 and there 100 MB of poor sample space which it requires over there. Now when I do the forward operation over there, what it does is that for each data point over there, it does a forward and then it creates a activation ingredients over there.

Now, these activation and gradients are what is stored on my GPU RAM, they are not present on my CPU RAM in any way and that is where it really saves. So, now if you can actually try doing this one take the same model run it only on the purely on the CPU do not typecast anywhere on the GPU. So, wherever you have a or what you can do is you can just set down on top of it like where it does a check for the GPU if GPU thing.

So, over that you can just put down that if GPU is equal to false, GPU availability is equal to false.

So that means, that even if the system has DP will not pass through the GPU root over there. Now then do a forward propagation on that and see. So, you would see that everything is what is changing on your CPU RAM; you can open up your GPU monitor and CPU RAM monitors and check out on this one. On the other side of it if you are passing it via the GPU would see that the models starts exploding and the space consumption increases on to your GPU side of it. Now, this is the difference which comes down, when you are operating on a CPU versus you are operating on a GPU and then how to work it out.

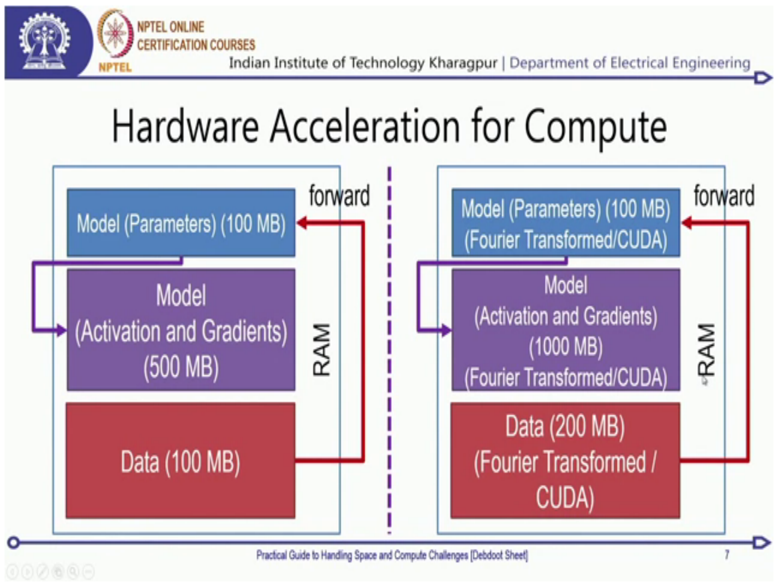
So, now here for this particular model, if I have a batch size of 10, then what comes down is that my poor sample of data I have about 600 MB consumed and so, over the batch size of 10 that would be 6 GB and I have a 100 MB. So, this is a 6.10 GB, now if have a GPU it just has barely 2 GB of memory this will not be working down over there because, you need more RAM than merely 2 GB over that.

Now, this kind of a mechanism of using a batch size of 10 is what is going to consume 6.1 GB. So, typically if I have a GPU of about say 8 GB it works out good. Now, say that I do not have a GPU of 8 GB I have a GPU just with 4 GB of space available. Now, let us change this back size and make it to say 5. So, if this one is 5, then the total space taken by the data and model activations is going to be 3 GB of space and plus this 100 MB of model parameter. So, that is 3.1 GB of space.

Now, 3.1 GB of space is something which is still within the 80 percent rule. So, 80 percent of 4 GB is about 3.2 GB. So, I am within that rule. So, with the back size of 5 I can run it down on a very plain simple GPU which has what 4 GB of RAM. So, that is say one of these commercially available models is in n media GTX 1 0 5 0 TI with 4 GB of RAM which can work it down on this kind of a simple example with just a batch size of 5.

Now, say you move over to a GTX 1 0 8 0 TI which has about 11 GB of RAM you can definitely increase batch size to about 12 or 15 and pretty much do it. So, that is how we actually go into a practical aspect of calculating out batch sizes on CPUs versus GPUs.

(Refer Slide Time: 24:56)


NPTEL ONLINE CERTIFICATION COURSES
Indian Institute of Technology Kharagpur | Department of Electrical Engineering
Hardware Acceleration for Compute
Model (Parameters) (100 MB)
forward
Model (Activation and Gradients) (500 MB)
RAM
Data (100 MB)
Model (Parameters) (100 MB) (Fourier Transformed/CUDA)
forward
Model (Activation and Gradients) (1000 MB) (Fourier Transformed/CUDA)
RAM
Data (200 MB) (Fourier Transformed / CUDA)
Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet]
7

Now, the next part which comes down is your hardware acceleration for compute now, typically when we are doing it on the CPU side over there, then I have my model and the parameters of the model which resides on the RAM and, I also have my data present on my RAM. Now, what I do is I do a forward and, then I get down my activations and the gradients coming it out coming out over there and, we had done a plain simple calculation into what is the total number of floating point operations and everything which it requires.

Now, if you would look into it if you are doing a convolution in the straightforward way, then there are more number of multiplications and the strides and everything which comes down. So, the total number of operations is significantly high, on the other side of it from our very classical image processing knowledge, we already know that convolutions can be represented in the Fourier domain as just dot products.

So, what that would mean is that if I have my kernel, I take a Fourier transform of my convolution kernel and my image, I take a Fourier transform of my image as well, then this Fourier transform things can just be done as a dot product. And that is much lesser amount of time a dot product is just point to point element to element products between these, now that will be taking much significantly lesser amount of time as compared to trying to do a direct compute on my with the cardinal form over there.

Now, typically what happens when you are not using accelerators, which is you can have things on the CUDA, but say you do not have special library called as a cuDNN available over there or say for your Intel processors and, there is something called as an MKL-DNN. So, if you are not using these MKL-DNN and libraries or cuDNN libraries, then you are still trying to do it with the classical way except for the fact that since you have a heavy parallelism in the data and, you have more number of cores available. So, you are going to push it down onto multiple number of parallel cores.

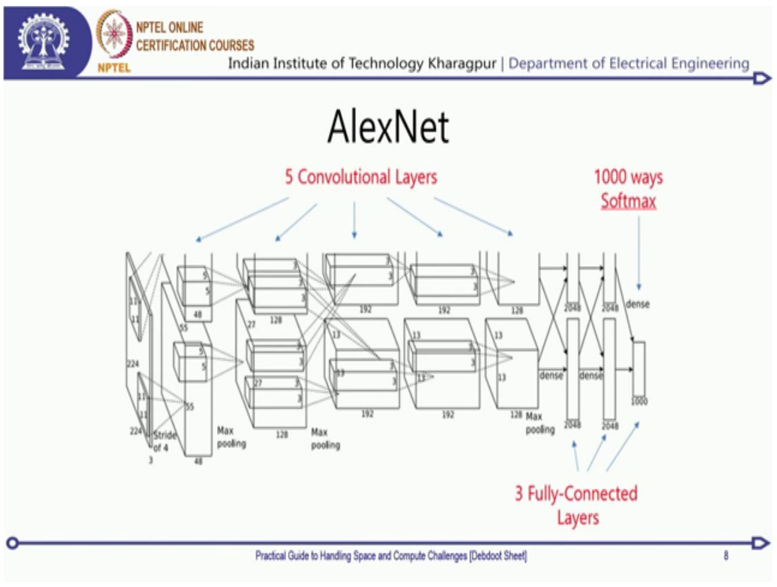
On the other side of it you are going to reform each of these computations. So, each convolution instead of in the spatial domain, this get is replaced as convolutions in Fourier domain, convolutions in Fourier domain and just you take and Fourier transform of the kernel and, Fourier transform of the image and you will do a dot product, which is much faster in terms of compute.

So, that is what happens; however, because you are making these Fourier transformed over there. So, Fourier transform data is going to take more amount of space. So, typically it is it is about double the space which it consumes over there and, for your activations as well as for your data.

So, this is where it goes. So, the downside is you will be requiring more amount of space, the upside over there is definitely that you would require less amount of time in which you can operate it out. So, this is typically what happens when you are trying to do it on accelerated form over there.

Now, this RAM does not necessarily mean that it is only CPU RAM; if you are doing it on the GPU then this is the GPU RAM where it works out. So, these are just a block level representation of what happens with each of them and when using CUDA versus not using CUDA and (Refer Time: 27:52) library versus not then what are the changes which comes down over here.

(Refer Slide Time: 27:57)


NPTEL ONLINE CERTIFICATION COURSES

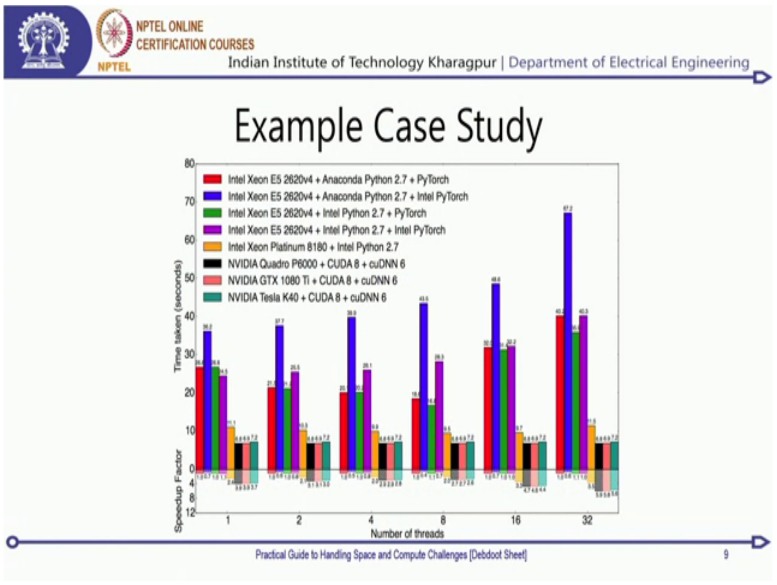
Indian Institute of Technology Kharagpur | Department of Electrical Engineering

AlexNet

Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet] 8

Now, what we would do is we have a very simple benchmark run down on a very simple algorithm on AlexNet and so, all of you are quite well aware of AlexNet.

(Refer Slide Time: 29:09)


NPTEL ONLINE CERTIFICATION COURSES

Indian Institute of Technology Kharagpur | Department of Electrical Engineering

Example Case Study

Practical Guide to Handling Space and Compute Challenges [Debdoot Sheet] 9

So, what we are going to do is just look into some of these example case studies over here ok. Now, you can see a lot of these bar graphs and stuff. So, let me explain it out. Now, if you look at this part which is from this 0 on to 80 which is the upper part over here.

So, what we have done is we have just plotted down what is the amount of time taken per epoch to execute. So, you do a forward and then do a backward over that and then within one epoch, how much of time is it going to take to execute it out ok.

Now each of these bars over here is for a different kind of a hardware configuration and a software configuration. So, we have 4 of these experiments which are run down on Intel Xeon E 5 2 6 2 0 V 4 which is just an 8 core processor and, the machine which we are using that has 2 of these processors.

So, in total you basically get down 16 cores available with you and, then what we do is we either use anaconda python and python or we use Intel python and PyTorch, or Intel PyTorch with python and these are the different combination which comes down over here ok.

The next one which we have done is on an experimental system, which is called as a Intel Xeon platinum 8 1 8 0, this is a newer version of Xeons which I would come down which have more number of cores. So, 8 1 8 0 typically has 28 course available to us and this particular configuration which we were using that had two of these 28 core machines and which work on a pure double precision floating point ok. Next we also tried against 3 different GPU.

So, one of them is Quadro P 6000 which has 24 GB of on board RAM, there is a GTX TI 1 0 8 0 TI which has about roughly 11 GB of on board RAM and the NVIDIA Tesla K 40 which has about 12 GB of RAM on it. Now, these are from different generations actually these two are pretty much from the same generation being Pascal's this is from the Kepler the intermediate Maxell, Maxwell is what we do not have over here for the comparison as such ok.

Now, if you look into it on the other X axis over here is what defines the number of threads, which is the total number of parallel processes which you can call down. Now typically you would see that if we keep on increasing this number of threads, then at a certain point of time the speed actually increases the total time taken goes down whereas, again it keeps on increasing.

Now, this is one of these machines which we were comparing it down, the one on yellow now that is something which is going to take down lesser amount of time as it keeps on going. Now, you would see that this has about 28 cores. So, at 16 courses we are at 16 threads is where it shows you the least somewhere over here as well. So, E 10 16 is typically where we were seeing down least, but then we have not yet looked into the total number of DMA channels available, it is not just a function of code which comes down. On the other side of it you have these massive GPUs as well which have almost a similar 1, where as saying that the Kepler is the older version whereas, the Quadro P 6000 is the newest version over there.

So, that is obviously, going to take lesser time and, what does not change is across the number of threads this does not change because, GPUs do not work in any way dependent on the CPU threads over that, they do with their own acceleration on cuda N cuDNN. So, on the other side we have plotted down the speed of factors and it going down. So, this was a typical example to show you that how different aspects do play a significant role and, for each of them based on how we are doing a software pipelining and in what order it is doing a DMA call and everything this is where it changes.

Now, this was just a comparative study of say on the CPU how can you use the best resources and bring down the total time consumed to the smallest possible, versus using a GPU and can at some point of time your CPU because, if you look into this Intel Xeon platinum. So, you would see that they are almost comparable to that of a GPU and this is an interesting fact because, they are the ones which come down close to a GPU and you might not even consider using a GPU at any point of time.

So, that is what I have till at the end of this one, in the next lecture we will start with certain other practical aspects, which are called as domain adaptation can I use trained model apprehend model available, for some other purpose to do another job over there and we keep on continuing on to more advanced topics. So, still then;

Thanks and stay tuned.