

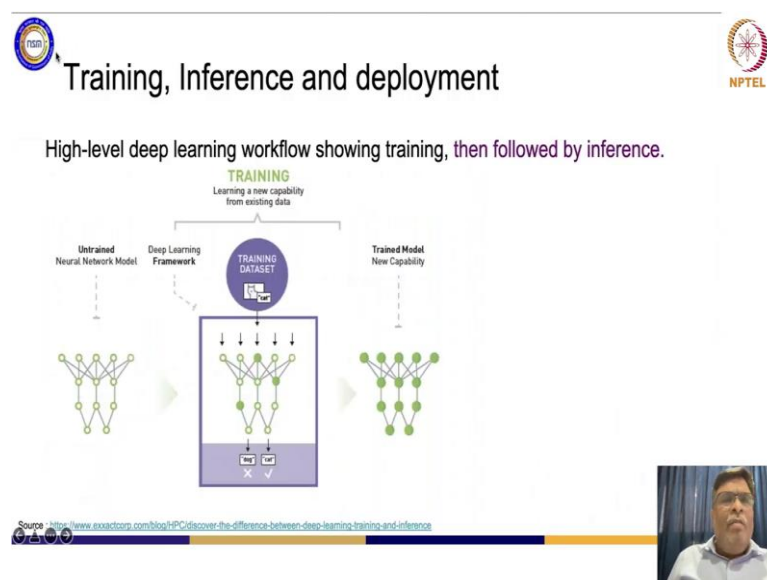
Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 40
Fundamentals of Accelerating Deployments Part 1

Yeah. So, today we will talk about Fundamentals of Accelerating the Deployments. Now, what actually these deployments mean and how is that it is different from whatever we have done till now ok. So, let us try to understand this and we will show you some videos of the demos which we have created. And, maybe tomorrow we will try to do some hands on them and show you as to how basically it is to be implemented.

The idea is to make you comfortable today with the flow so, that you people can actually sit, understand it once. And, then tomorrow when we do something, you would be able to actually correlate it with the total flow of what we have told you today right. So, with this intention we have split this lecture into two separate modules right. So, fundamentals of accelerated deployment, we will start with the basic idea of training, inference and deployment ok.

(Refer Slide Time: 01:17)




So, the idea here is that till now ok, we have been doing this. You have been actually doing a deep learning workflow which basically does the training right. And, after the training is the most important thing of inferencing right. Inferencing basically is your

trained model needs to be put to the actual prediction use right at the side of where your application should actually run.


So, when you have been doing this training till now, what effectively have done is there is a untrained neural network model which you have to you use that neural network model. You use any deep learning framework for that matter then you have your training data set. It can be cat dogs or it can be based on your problem which you are trying to solve right. It can be a classification problem, it can be a object detection problem, it can be any problem for that matter ok.

You have a training data set, then you all of this to develop a trained model with a new capability right. Now, this particular model which you have developed now which has been trained on this data set is capable of doing the classification task for you right. Now, what are you supposed to be doing is you have to actually use this particular developed model ok for inferencing or for using it through actually decide and do some prediction ok, based on what you have the trained model for.

(Refer Slide Time: 03:22)





Inference and Deployment



- Training is only half the story
- After a model is trained, we still need to actually use it to make predictions and/or decisions. This is usually called inference.
- Deployed ML inference system – Where do we deploy (Cloud or Edge devices)
- how do we measure the performance of a deployed ML inference system?

source: <https://www.cs.cornell.edu/courses/cs4787/2019sp/notes/lecture24.pdf>



So, this means that when you talk of inference and then deployment, training is only half the story of what you have done till now. You have been training it, you have done distributed training, you have done training with various parameters, you have automatic tuning, you have done everything. So, training is only half the story. After we have, after

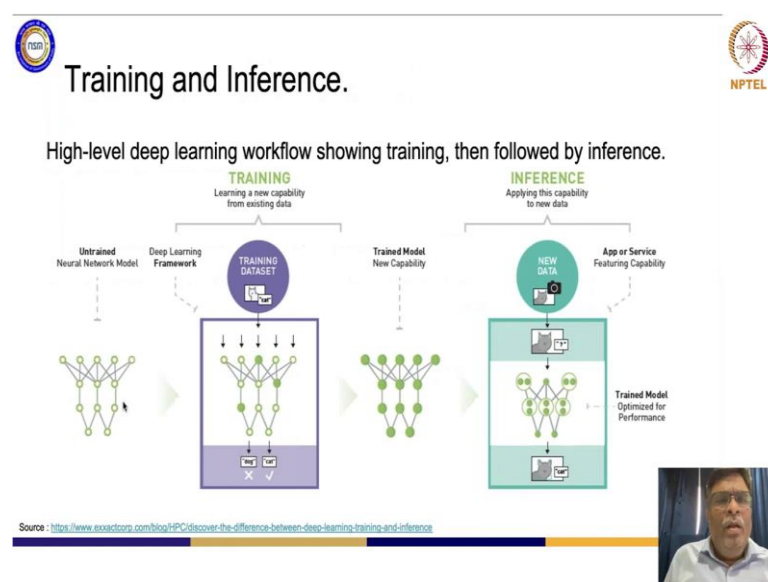
a model is trained we still need to actually use it to make prediction and or decisions right. This is what is called as inferencing or inference.

Now, once you deploy this particular model ok for achieving inferencing that is when you call it as a deployed ML inference system. Now, generally where do you actually deploy them? You deploy them either on the cloud or on the edge device right and I hope you people are very very clear about what does it mean. So, when you deploy it on the cloud, your inferencing portion or execution of the inference engine is actually run on the cloud or it can be done on the edge device right.

So, we would be showing you a video of doing it on an edge device right. And, once you are able to deploy it, actually how do you measure the performance of this deployed ML inference system, what do I mean by this right? So, let us try to understand, actually you would have trained a model, it is giving you some accuracy of let us say 95 percent.

But, when you deploy it at a place which is supposed to do the prediction for you ok on some device; it can be a cloud or it can be edge. How is that the performance of your trained model ok is happening on that particular inferencing system right. How does my model which is trained behave on that particular inference mode inference system ok?

(Refer Slide Time: 05:47)



So, so this basically means that once you have trained this model, it has to be put ok on the device or the edge device or wherever it is supposed to be run and where it will be

giving you ok the decisions. So, here you are applying this capability to new data. So, this trained model is there, you have put it on an edge device or you have put it on a cloud. This becomes your inferencing system.


And, once this becomes your inferencing system ok, there has to be app or a service which you are developing which will use this trained model which needs to be optimized. Please understand it needs to be optimized for running efficiently on this edge device ok. Now, why do you want to optimize it? The reason is this trained model was trained with parameters on a let us say a high end GPU or a data center GPU. You can do the inferencing there also.

But, will it be of any use to us? No, we want the inferencing to be done ok at the actual site or actual place right. Now, that actual place or a site would be either a cloud or a edge device. Now, this edge device which you talk about are not as computationally strong ok or as computationally strong as these data center devices are ok. These devices have very less amount of memory right, less amount of compute power and they have to use very less amount of power as well.


So, how do you actually now convert this trained model which was actually trained using this type of a huge system which; obviously, will also be very big in size or huge in size, but it has to be efficiently running on this small system ok and give you the correct prediction. So, this is where this talk is going to concentrate on ok. So, I hope you have understood the context of what actually is going to be discussed today.

So, you have trained, you do have to do inferencing. This inferencing has to be done on a device. It can be either on a cloud, it can be either on an edge device which has got certain constraints of memory, power, size everything. But, you should get the same performance ok of what you have already actually attained ok when you are doing your validation testing and some testing on your actual model, when you are doing it on the data center GPU or on the DGX right. So, this is the whole context.

(Refer Slide Time: 09:20)




Performance of the Deployed System



- Accuracy. How accurate is the model on the queries coming in?
- Latency. How long do we need to wait between when we make a query and when we get a prediction from the system?
- Throughput. If we have a large batch of queries we want a prediction for, how many predictions can we get per second out of the system?
- Energy/power. How much energy is consumed to make one prediction? How much power is used by the deployed ML system in general?
- Model size. How many bytes are needed to store and/or transmit the learned model?
- Memory use. How much memory is used by the deployed ML system?
- Cost. How much money does my deployment cost?

When deploying an ML system, there are trade-offs among ALL of these metrics!

source: <https://www.cs.cornell.edu/courses/cs4787/2019sp/notes/lecture24.pdf>



Now, what are the basic things which we need to keep in mind ok; when we try to work with? Understanding the performance of the deployed system; when you say performance of the deployed system, it means that you should be talking of all of these things which have been written on this slide.

What we will start with is accuracy. Now, when we say accuracy, you have developed a model. Now, you are supposed to be actually optimizing it further in size ok in the way in which it stores the information. There is so many things which you need to actually do which we will see and still it should be as accurate ok, as you would have done it during your training, validation and testing phase.

So, the accuracy thing is as to how accurate is the model on the queries which are coming in. Then, how long do we need to wait between when we make a query and when we get a prediction from the system. So, latency is very very important. You are deploying that system ok in real time. So, the latency is of utmost importance. Of course, accuracy is also of utmost importance, but latency will tell us as to how much time we will have to wait before the next query is answered right.

And, the third point is the throughput, in a sense that if we have a large batch of queries which we want to predict; how much predictions can we get per second out of the system? So, try to understand the context in which we are trying to actually equate the

throughput of the trained model as well as the throughput of this particular device on which you are porting this model ok which is optimized model though ok.


Now, how much energy per power is consumed to make one prediction? This basically means that how much power is used by the deployed system in general. Now, broadly these performance measures which we have discussed will depend on the model size. Now, you have x amount of model size which you have seen it in previous classes, now how many bytes are needed to store and to transmit the learned model? You have a learnt model now right.

Now, this model has to be shifted to a smaller size device. Now, what should be the size of that model now ok? How do you do it and how much memory is used on this deployed ML system? Because, it will not have too much memory right as compared to your DGX or as compared to your system which has a GPU connected to it or whatever right. And, once everything is done, you basically have to understand the cost as to how much money does my deployment cost?


How many edge devices you are going to deploy and how fast you are going to get the predictions done right and how much cost is going to be incurred for actually deploying ok these devices? So, the basic idea here is that when deploying this type of a system is to be done, there are tradeoffs among all of these matrices.

So, all of this you will have to keep in mind when you basically try to deploy a system ok. So, I hope this is clear as to why are we concentrating too much on this aspect of just trying to do inferencing on a system which needs to be actually deployed or that particular model needs to be deployed on a smaller system or something like that ok.

(Refer Slide Time: 14:10)



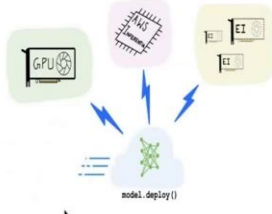
Model Deployment



- Integration of the model into an existing production environment which can take in an input and return an output that can be used in making practical business decisions.


Why it is important ?

- Effective deployment is needed to provide proper prediction to other software systems.
- To maximize the value of the Deep Learning model by reliably extracting the predictions.



model_deploy()

source: <https://towardsdatascience.com/a-complete-guide-to-ai-accelerators-for-deep-learning-inference-gpus-aws-inferentia-and-amazon-7a5d5804ef1c>



So, now let us try to go into what a model deployment means. We have thought of what are the parameters which we will need to keep in mind when you are trying to actually deploy a model. So, now, let us try to understand this model deployment. So, this is your model which needs to be deployed. You can deploy it anywhere right. You can deploy it on a GPU, you can deploy it on a cloud, you can deploy it on some small devices with GPU right.

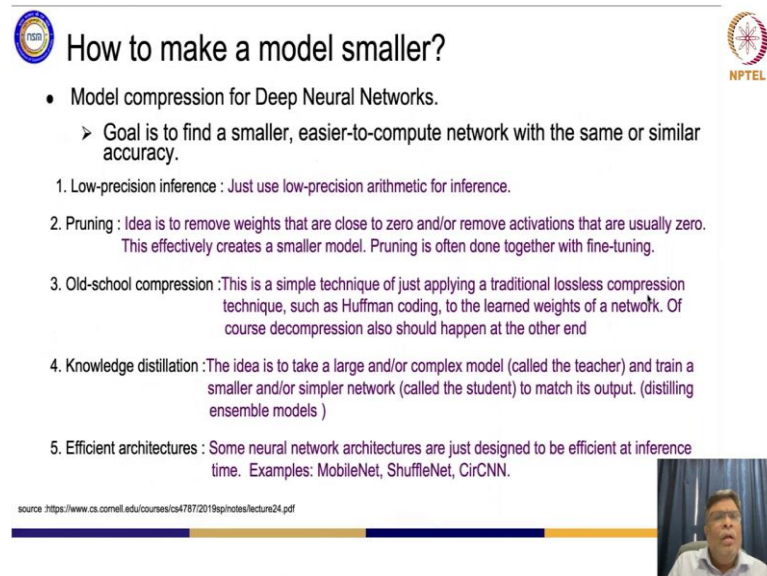
So, model deployment basically talks of integration of the model into an existing production environment which can take in an input and return an output that can be used in making practical business decisions. So, this is what you mean by a model deployment in technical terms. So, the idea is to integrate the model which you already have into an existing production environment which is going to take an input and return an output for making practical business decisions ok at the real point of decision making.

Now, why it is very very important is that effective deployment is needed to provide proper prediction to other software systems. So, you would be using this ok as input to other decision making softwares may be. So, that is where the effectiveness of this deployment is of much importance.

And, we also need to maximize the value of the deep learning model by reliably extracting the predictions. This means we should get maximum value out of the deep learning models which we have developed and we should also be able to extract good

reliable predictions out of them. So, this is the whole gist of trying to have model deployment on various different types of devices.

(Refer Slide Time: 16:50)



The slide is titled "How to make a model smaller?" and features a list of five techniques for model compression. It includes logos for CSO and NPTEL in the top corners. A source URL is provided at the bottom left, and a small video inset of a speaker is at the bottom right.

- Model compression for Deep Neural Networks.
 - Goal is to find a smaller, easier-to-compute network with the same or similar accuracy.
- 1. Low-precision inference : Just use low-precision arithmetic for inference.
- 2. Pruning : Idea is to remove weights that are close to zero and/or remove activations that are usually zero. This effectively creates a smaller model. Pruning is often done together with fine-tuning.
- 3. Old-school compression : This is a simple technique of just applying a traditional lossless compression technique, such as Huffman coding, to the learned weights of a network. Of course decompression also should happen at the other end
- 4. Knowledge distillation : The idea is to take a large and/or complex model (called the teacher) and train a smaller and/or simpler network (called the student) to match its output. (distilling ensemble models)
- 5. Efficient architectures : Some neural network architectures are just designed to be efficient at inference time. Examples: MobileNet, ShuffleNet, CirCNN.

source : <https://www.cs.cornell.edu/courses/cs4787/2019sp/notes/lecture24.pdf>

Now, very very important thing in this lecture is to understand how do we make a model smaller. In the sense, you would have trained a model, it would have got lot of layers, you would have got lot of weights, you would have done forward propagation, backward propagation. All of this is condensed ok into some information which depicts a specific model ok.

Now, what we are talking of is trying to actually reduce the model size, but we want to reduce the model size, but we want to keep the prediction rate the same. This ultimately means that we are trying to do some model compression ok. So, you are trying to compress the deep neural network model. Now, what will we achieve by doing this ok? Now, the goal is to find a smaller, easier to compute network with the same or similar accuracy.

Please understand this, that whatever deep neural network model we have trained, we have to now arrive at a smaller, easier to compute network. Because, you are trying to actually do it at the edge, which is not as compute intensive as your existing data center GPU was. So, you are supposed to find a smaller model which should be easier to compute and the condition is that you should actually have the same or the similar accuracy for doing that task for which your actual model was trained.

So, there are 5 different approaches of doing it of arriving at a smaller model from actually a trained bigger model which you already got. The first one and we will concentrate more on this in today's as well as the next class. So, when you talk of low precision inferencing or we would like to use low precision arithmetic for the inferencing part. We will discuss what does it mean, but the idea is to use int8 instead of FP32 in a general sense of training.

So, we would generally be using FP32 for actually trying to train our model for obtaining better accuracy, better precision and all of that. Once that is all done, how do we actually convert it into a low position arithmetic for achieving the same type of prediction ok during inferencing is what tells us of how do we actually do low precision inferencing. The other idea is pruning which basically means you remove the weights that are close to zero and or you remove the activations that are usually zero.

So, when you do this, this effectively creates a smaller model. Now, in some of the cases generally, you would have seen that when you are fine tuning your training model itself, people go in for pruning. So, the effective idea is there also if you reduce your model during training phase that is a good idea right. There also of course, you have memory that does not mean you should waste your memory right. So, you anyways are trying to optimize your model at that point itself.

So, people do pruning at the training phase itself to improve upon your model. Now, the third thing is there is something which people have been talking of, we know from decades that we have this compression method right. And, people also have been using lossless compression techniques as Huffman coding to actually code your learned weights of the network; so, that it reduces the model memory size. Of course, you need to have the decompression also happening at the other end.

So, the idea here is to reduce the memory footprint of your model. One of the other ideas is knowledge distillation which basically means that you take a large of a complex model which is called as a teacher and using that you train a smaller, a simpler network which is called as a student to match its output. So, this basically is related to something like using ensembles or distilling the ensembles models right.

Also, the 5th point is efficient architectures. When you say efficient architectures, this is not actually going to make a model smaller in a sense that directly it is not related, that

you are going to make a trained model smaller. But, there are certain architectures which are designed to be very very effective at the inference time. So, these particular models were developed to work on situations which are to be run on edge devices or on mobiles or on any other systems wherein the inferencing is to be done.

And, these architectures ok are very very efficient at inference time. So, you can use these architectures, train them and then use them effectively directly for the edge devices or the inferencing devices right. So, this is how you actually try to make a model smaller. Any questions till now or any doubts till now?