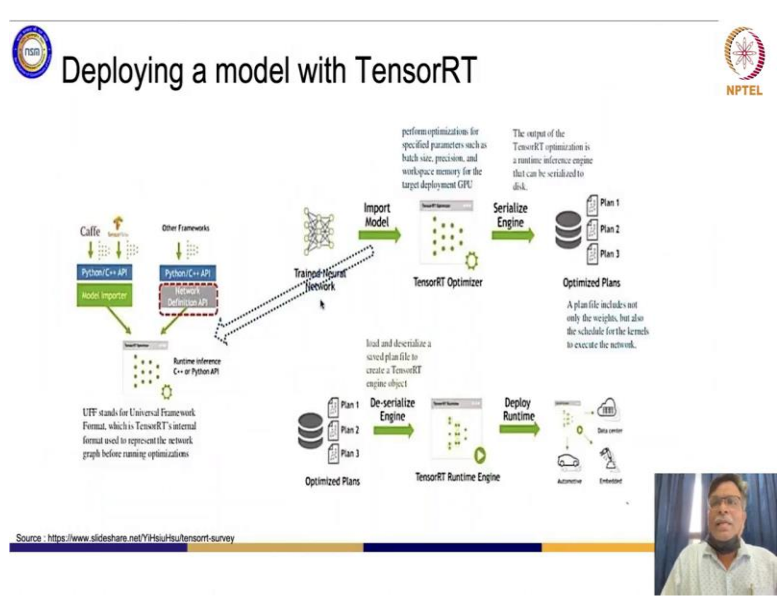
Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Fundamentals of Accelerating Deployments
Lecture - 42
Accelerating neural network inference in PyTorch and TensorFlow Part 1

Good evening everyone. Today I will try to actually do some demos on Accelerating deployments ok. And as I showed you yesterday some videos maybe today we will show you the demonstration by again connecting a webcam there is some issue with the team viewer, but we will anyway show you some more videos today in case that does not do materialize right.

Because it is getting struck the team viewer video is getting stuck. So, no issues for that but, let us start with certain things wherein we will try to show you the maximum demo stuff ok.

(Refer Slide Time: 01:00)



So, yeah so, yesterday actually we ended up at this point right that how do you actually deploy a model with TensorRT ok. And we showed you this particular concept we showed you a demo and we actually discussed this yesterday that we have either a TensorFlow or a caffe or any other framework which basically is written using python ok.

And then you have a model importer which actually imports ok using a TensorRT ok right. So, the idea is you basically come up with a model which is a universal framework format or at present you can use ONNX as well right. So, I have got two options of actually converting your actual model which you have trained it on a.

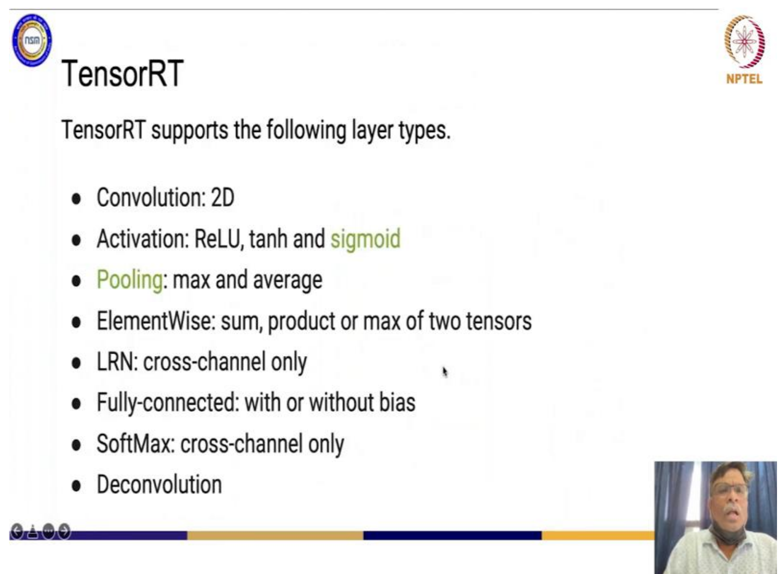
Students: Server.

Server right and then convert it into something which runs on a inferencing edge device right. So, you can use UFF or you can use ONNX. So, we can show we will show you ONNX also and we have done it using UFF also right. So, let us see how we can show you both of this right. So, the basic idea what we discussed yesterday was you have a trained neural network model you import that model and then try to use a TensorRT optimizer right for trying to optimize it. So, that it runs on your edge device right.

So, for that reason you actually have this process which is internally done right. You have to serialize it then you generate one or three optimized plans and then you get a plan file right with actually schedules ok for the kernel to execute whatever it is to be executed.

So, you take this optimized plan ok and then as shown you basically create a Tensor RT engine object and then you run it using a Tensor RT runtime engine and that is how you basically try to deploy on the edge device. So, basically you are trying to deploy the runtime. So, this is the overall model which we discussed yesterday right.

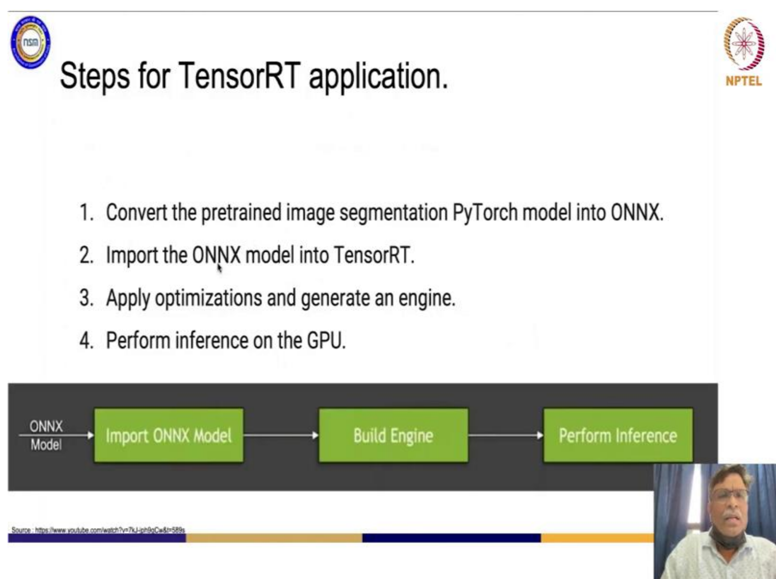
(Refer Slide Time: 03:51)



So, now today we will try to understand that when we are trying to work with TensorRT, TensorRT supports the following layer types. What does it mean? It means if you are using convolution layers if you are using activation layers which involve ReLU, tanh, sigmoid and you have this pooling max and average stuff. Elementwise sum, product or max of two tensors ok and then LRN and then fully connected with or without bias SoftMax, deconvolution.

All of these layers could be converted from TensorFlow or PyTorch to a Tensor RT survey right this is what it means.

(Refer Slide Time: 04:40)

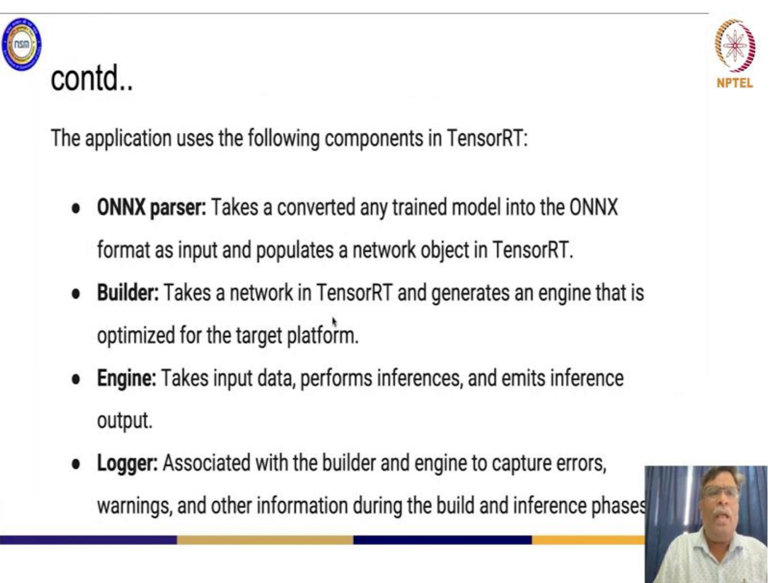


And now that effectively means that whenever you are trying to convert ok any model for inferencing the steps would be like something like this you actually convert for example, if you are trying to do a image segmentation application deployment. So, you convert the pretrained image that segmentation PyTorch model or for that matter the tensor flow model into ONNX or ok or UFF anything.

And then you import the ONNX model into Tensor RT model ok this conversion needs to be done and then you basically apply several more optimization generate an engine. So, technically speaking once your model gets converted into a Tensor RT runtime ok you can actually do the inferencing on that particular edge device right. So, we have written it on the GPU. So, you can do it on GPU you can do it on edge device you can do it on Jetson any of these right. So, this is how basically it is done ok.

So, there are various ways of doing it one of the ways what we are trying to tell you is this ok you can actually see in literature and various other places there are various ways of doing it one of the ways is you either convert your TensorFlow program into a TensorRT program through UFF or ok through ONNX. So, this is the basic idea ok.

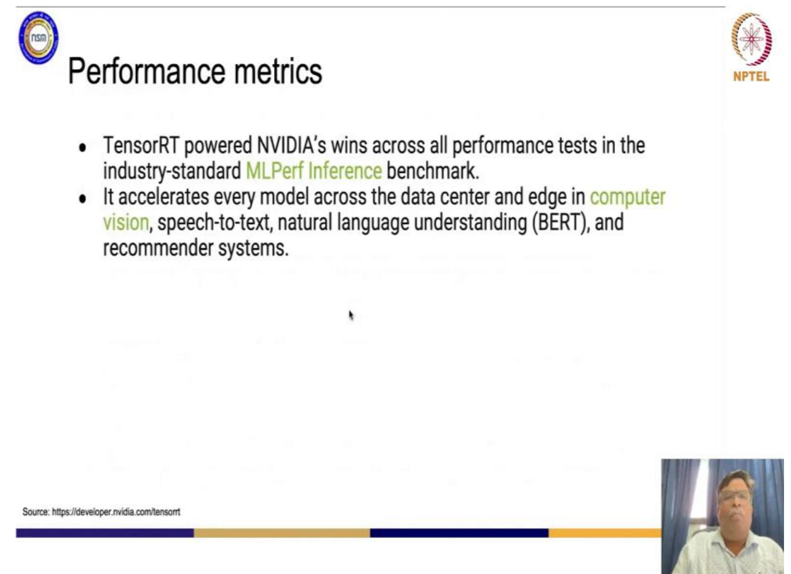
(Refer Slide Time: 06:27)



So, you have got various components in TensorRT you have a ONNX parser which basically converts ok or you have a UFF parser you basically take any trained model into the ONNX format as input and you get a network object in tensor RT. This is possible ok and then you take a network in Tensor RT and you generate an engine that is optimized for the target platform.

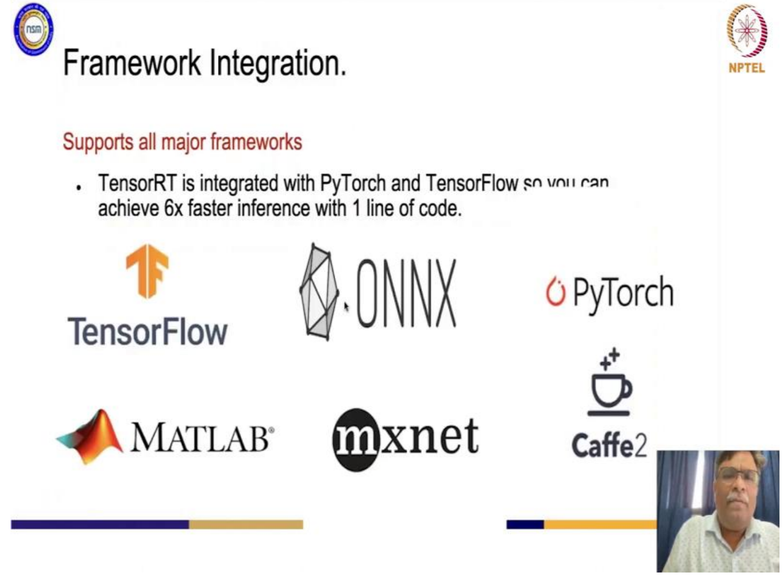
So, this target platform can be anything. So, ultimately you basically take a network in TensorRT and generate an engine similarly your engine takes an input data perform inferences and then emits the inferencing output and logger basically tries to give you warnings capture errors and how you run and everything right.

(Refer Slide Time: 07:24)



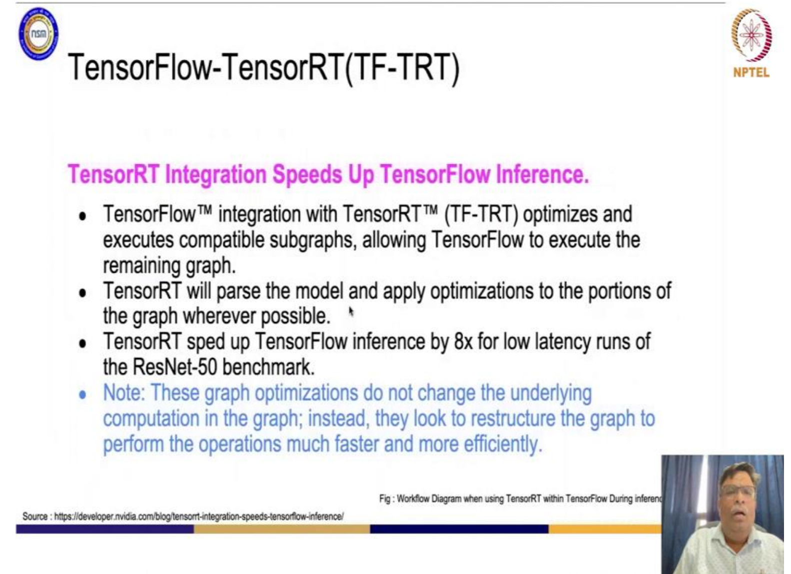
So, this basically is various component of a TensorRT. So, TensorRT powered NVIDIA edge devices actually if you see their benchmarking performances they are quite good they are very very good. So, you can accelerate every model across the data center and edge in computer vision applications speech to text NLP for example, BERT and various recommended systems right. So, all of this would actually be done ok on the edge devices using TensorRT optimizations right.

(Refer Slide Time: 08:04)



So, it supports all major frameworks TensorRT is integrated with PyTorch TensorFlow and you can achieve 6x faster inferencing with 1 line of code. So, this is the basic idea you can actually integrate everything ok.

(Refer Slide Time: 08:26)



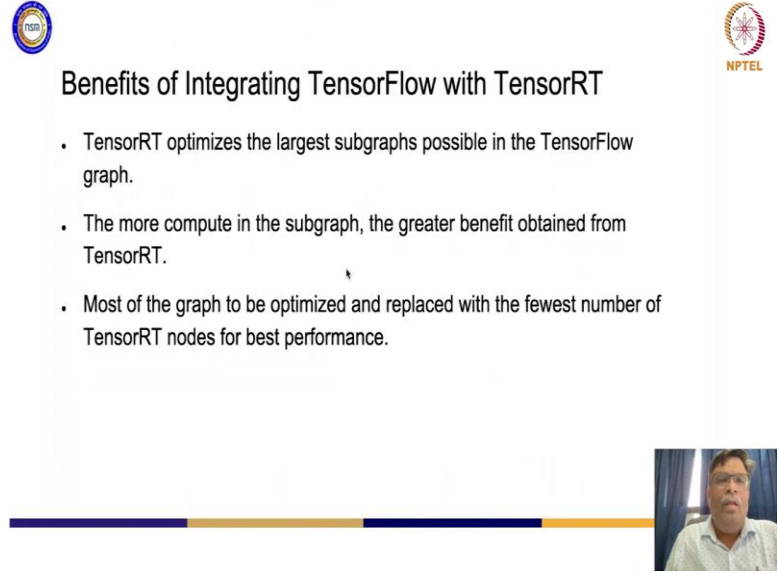
So, TensorRT then from TensorFlow: So, now, the idea is let us say we have a TensorFlow program and this TensorFlow program needs to be converted into a TensorRT program or a TRT program.

So, TF-TRT is what we generally call. So, TensorFlow integration with TensorRT ok; so, when you do this you basically are trying to work with sub graphs ok and then trying to understand how tensor flow ok graph can be converted into an optimized TensorRT graph. So, the effective idea here is your TensorRT is going to generate a model a parse model and then it applied optimizations to the portions of the graph wherever possible. So, we will show you various graphs right in the next few seconds or minutes right.

So, we will show you how basically a saved model a TensorFlow graph is converted into a optimized graph right that also you can visualize. So, technically speaking TensorFlow RT speeds up your inferencing by 8x for a ResNet 50 benchmarks right. So, there are certain things wherein you can see about graph optimizations ok do not change the underlying computations in the graph. So, that is very very important see your computations will not change ok, but the graph will be optimized.

So, basically you are restructuring the graph to perform the operations much faster and more efficient right efficiently right. So, this is the basic test software of it of why do you do actually this and how is that it is going to speed up ok that is also what you are going to actually see, ok.

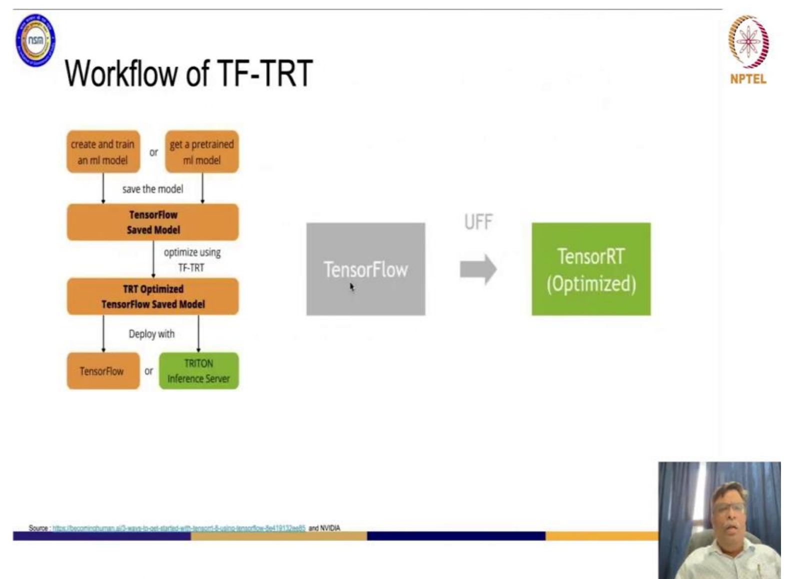
(Refer Slide Time: 10:31)



So, now when you integrate the TensorFlow with Tensor RT ok. So, Tensor RT basically optimizes the largest sub graph possible in the TensorFlow graph and if there is more compute in the sub graph; obviously, the TensorFlow optimizes it more ok. So, more computations are there in your model right more optimization you can expect out of TensorRT.

So, most of the graphs which are to be optimized and replaced ok with the fewest number of tensor node RT; so, you generate a TensorRT node actually. So, the idea is you can optimize and replace ok with certain specific less number of nodes as compared to your actual nodes in the graph right. So, that is what actually you are going to be doing right.

(Refer Slide Time: 11:36)

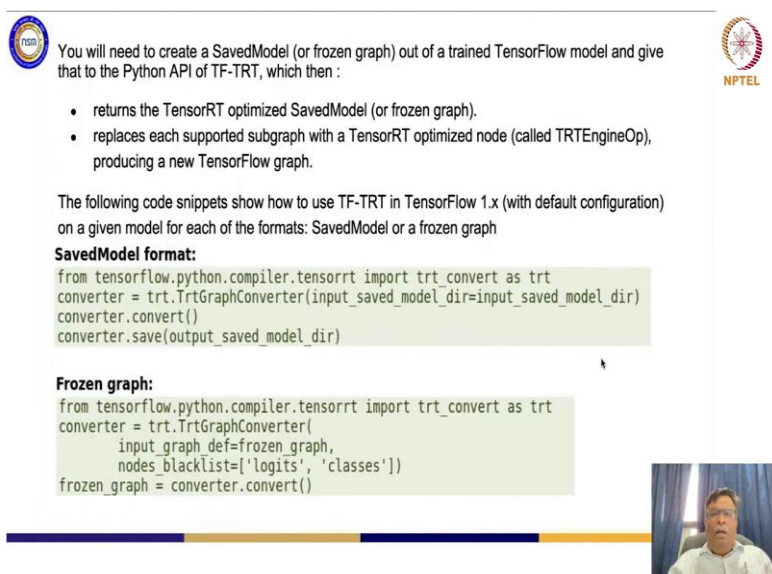


Now, this is how it actually happens you create and train an ML model or you get a pre trained ML model both are possible right you save that node; obviously, you are training your model and then you are saving your model once you save that model you get a TensorFlow save the model there is something which is called as TensorFlow frozen model ok. Now this TensorFlow frozen model cannot be edited or it cannot be changed that is what it means in model trainer right you can actually train a save model again and again.

But once you freeze that model right it has got all the weights parameters everything frozen right. So, you cannot actually use it for training again that is what it means and that is what you are going to optimize ok. So, you are going to optimize that ok saved model from the frozen model and then you deploy with either a train triton server or you do it with TensorFlow.

So, now, this is what I told you that your actual thing of understanding at the first level of complexity is this that you convert your TensorFlow program into a TensorRT optimized programs right by converting it intermediately to a UFF or ONNX right. So, open neural network exchange format ok. So, something like that. So, this is what it means that you are converting your TensorFlow program to a TensorFlow RT, TensorRT program right ok.

(Refer Slide Time: 13:22)

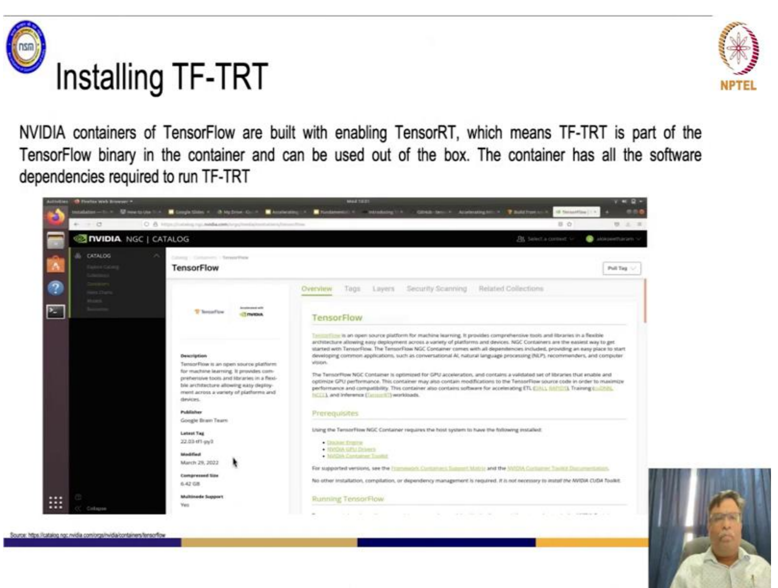


So, the idea is that you create a SavedModel or a frozen graph. Out of a trained TensorFlow model and give that to the Python API of TF-TRT which is going to return TensorRT optimized model; replaces each supported sub graph with a TensorRT optimized node or a TRTEngine operation producing a new TensorFlow graph.

So, this is how you actually convert ok I use the SavedModel format and this is how you get a frozen graph format. So, if you see here ok. This basic a thing TensorRT import trt converter as trt and then you are using a converter you are just trying to actually save a saved model ok.

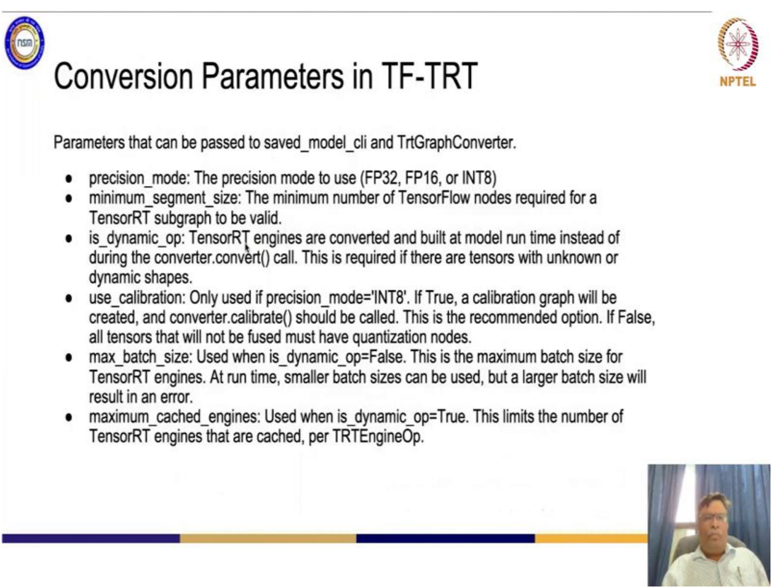
And then you actually get a frozen graph by using a converter convert ok and then you get a frozen graph ok. So, this is how basically you would be using it in your program ok this is just the idea of how you are basically using various converters which are available ok. For converting your various models into various formats like a SavedModel format or frozen graph format ok.

(Refer Slide Time: 14:59)



So, one of the things when you want to work with TF-TRT is that you need to install TF-TRT you have NVIDIA containers as we told you in our previous sessions as to how do you pull NGC docker container. So, you pull a NGC docker container and you actually use ok for TensorRT enabled TensorFlow thing right. So, TF-TRT container ok and this basically is supposed to be pulled or installed ok from a TF-TRT container ok from the NGC cloud ok.

(Refer Slide Time: 15:45)



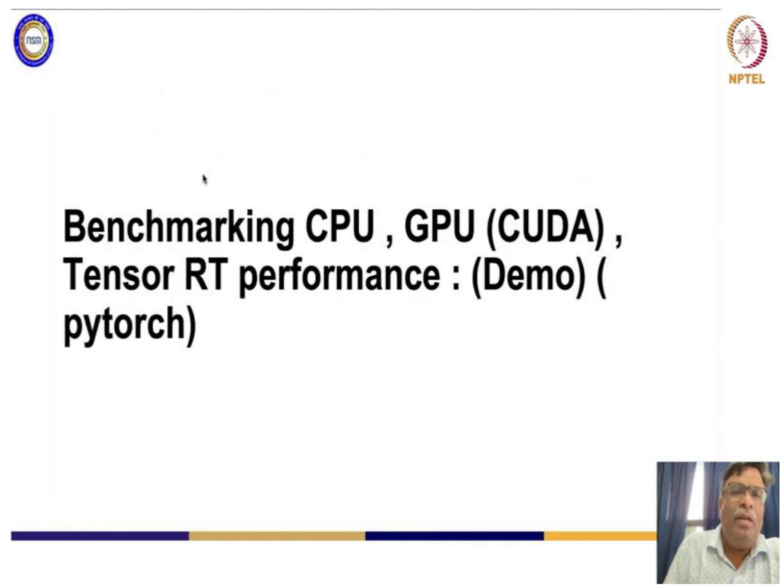
So, what are the various parameters when you try to actually use this ok? So, the parameters which can be passed ok so, this is a command line interface option saved model CLI to TRT graph converter you can work with any precision mode ok. So, the precision mode which you can use is FP32, FP16 or INT8 ok.

What is the minimum segment size? So, that basically means the minimum number of TensorFlow nodes required for a Tensor RT sub graph to be valid ok. This when you go on practicing and running your programs you will try to understand ok all of these particular parameters.

But just to have the completeness of trying to link ok. How these parameters are actually connected we are just trying to give you a very very brief idea to this ok. And then you can use calibration because calibration is needed yesterday I told you that when you are converting from FP32 to FP16 ok, it is done. But when you convert something from FP32 to FP16 to something like INT8 right there is a calibration required right. So, if the precision mode is INT8 the calibration is needed. So, effectively a calibration graph will be created ok.

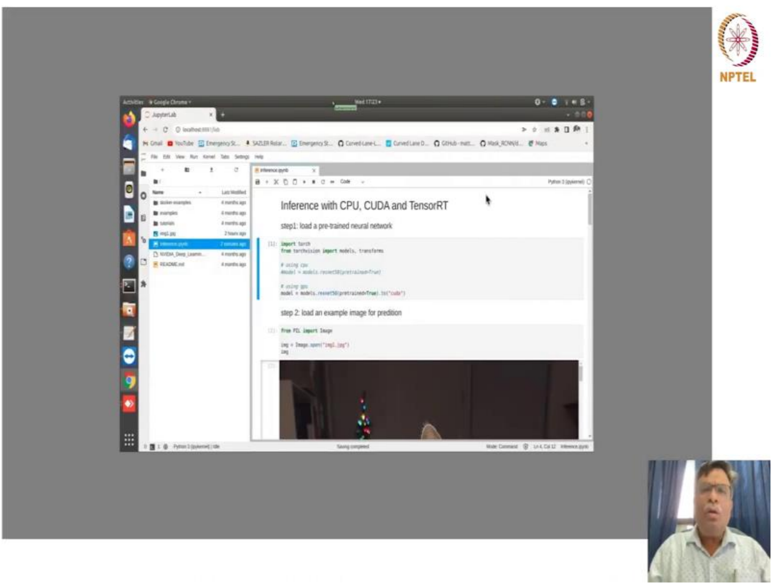
So, this is what it means ok and then you have the maximum batch size and the maximum cached engine thing right. So, the limit of the number of RT engines which can be cached per TRT engine operation because it depends on the memory of your edge device right based on that how much you can cache is going to actually make your system slower and it can get stuck or something of that sort might happen. So, that is why you are going to use some of these parameters ok.

(Refer Slide Time: 17:56)



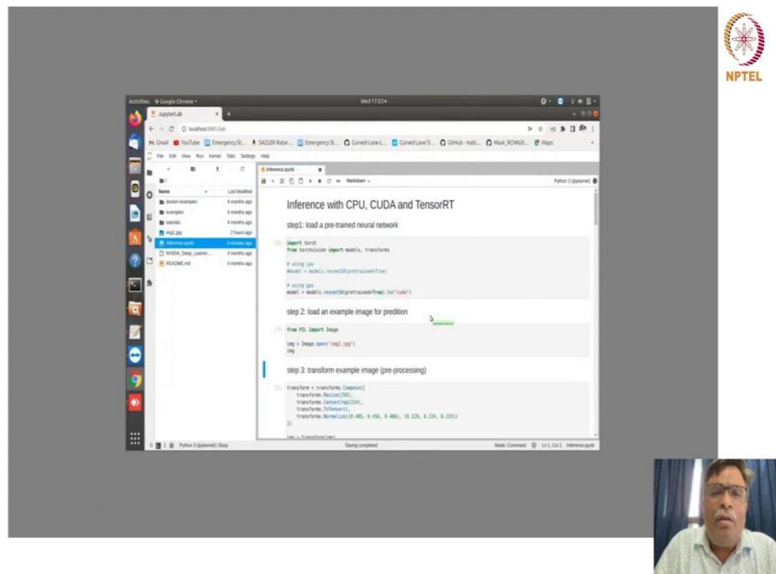
So, let us try to do some benchmarking ok.

(Refer Slide Time: 17:59)



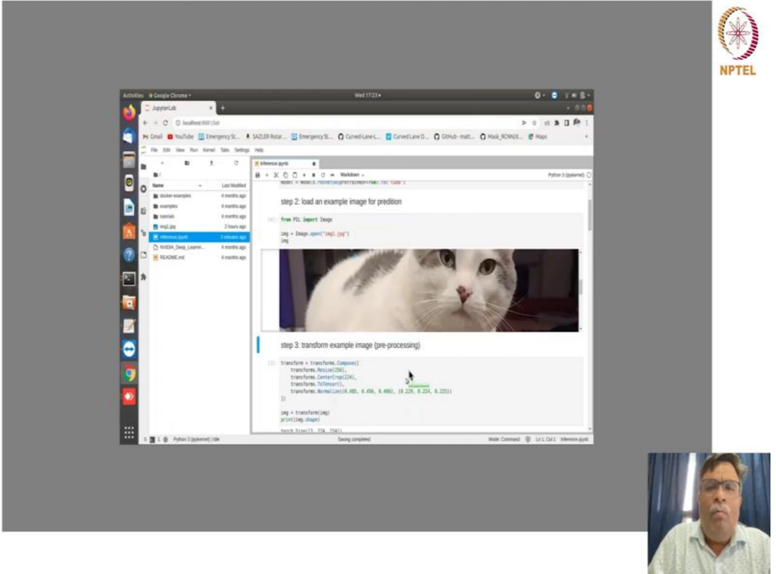
So, let us try to understand the inferencing portion on just GPU, CUDA and TensorRT. So, this particular thing is a very very simple example of trying to import ok a pre trained neural network model. So, we are just trying to load an example image ok.

(Refer Slide Time: 18:26)



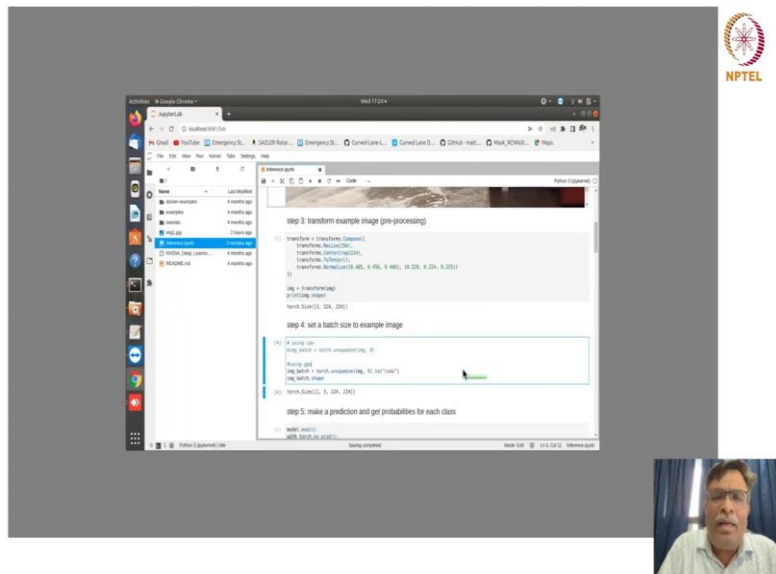
For the prediction which we are able to see ok.

(Refer Slide Time: 18:32)



Now, what we are trying to show here is this is a image ok and then there is some transform which is happening. So, we are trying to pre process this image ok.

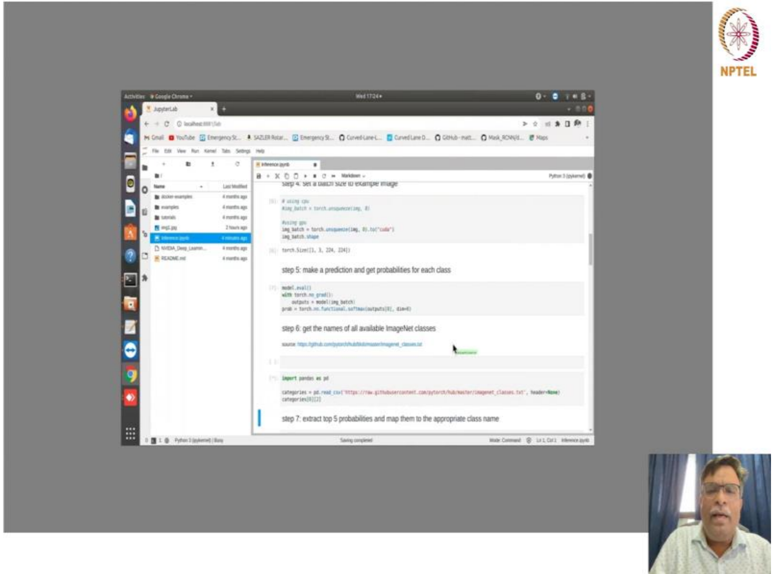
(Refer Slide Time: 18:49)



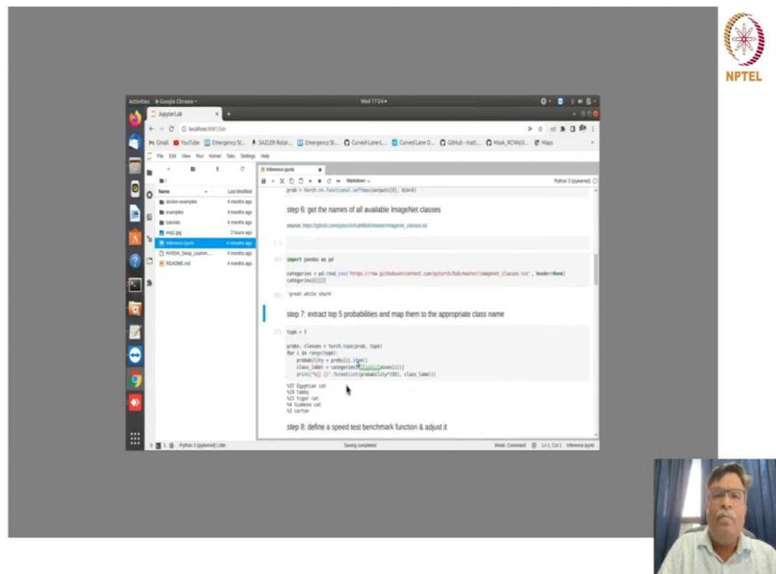
And then we are trying to actually set a benchmark size to this example image ok. So, you can do it using CPU you can do it using GPU and you can do it using TensorFlow RT ok. So, we will get the prediction and probability for each class right. So, this is what actually is a very very simple thing of where you are going to actually link it and link it with the image net classes right that is what you are trying to do ok.

So, yeah. So, we will run this and we will extract the five top probabilities.

(Refer Slide Time: 19:22)

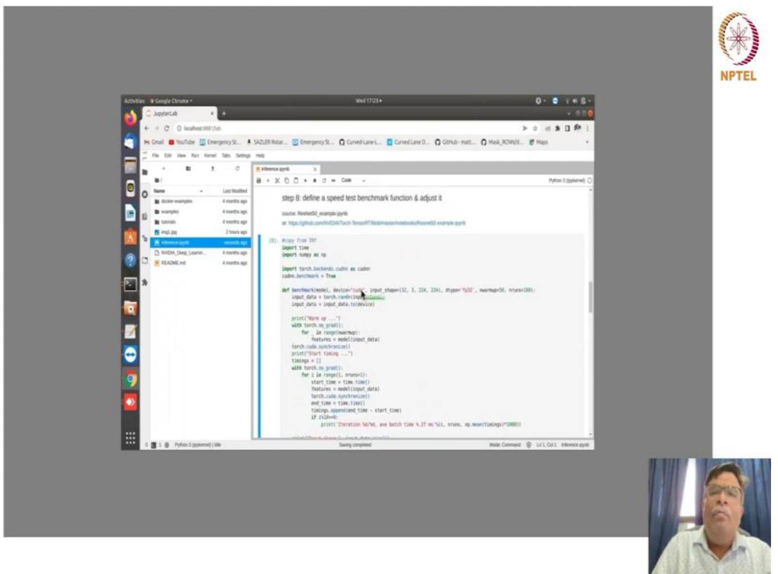


(Refer Slide Time: 19:26)



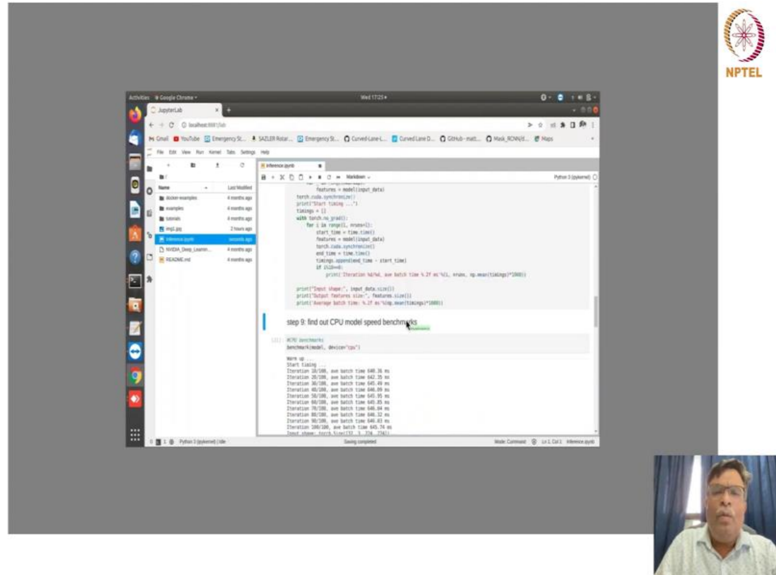
This is for that particular thing. So, it has detected that it is basically a Egyptian cat its tabby, tiger cat semi stat and carton right; so, with these many prediction percentages right.

(Refer Slide Time: 19:47)



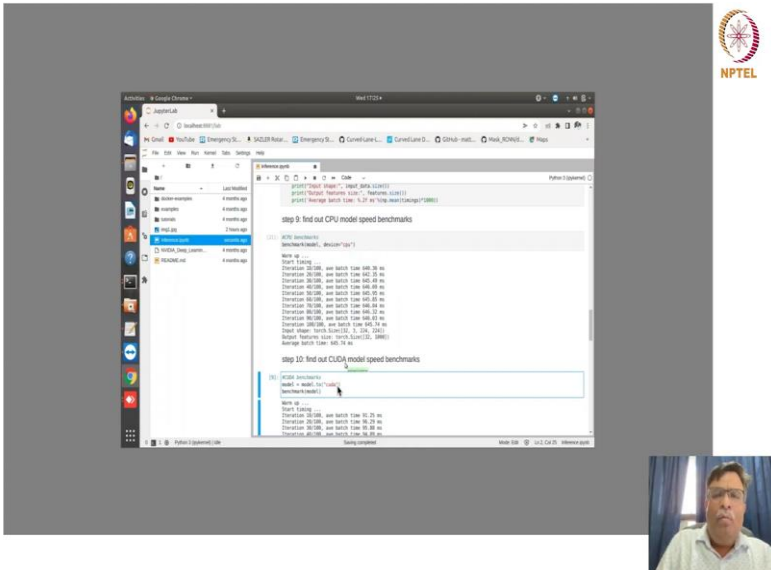
So, now we will try to understand the speed test with the ResNet 50 ok example. So, what we are trying to do is this is basically a TRT program ok and then we are trying to we are trying to just show you ok that how basically for each of this ok.

(Refer Slide Time: 20:05)

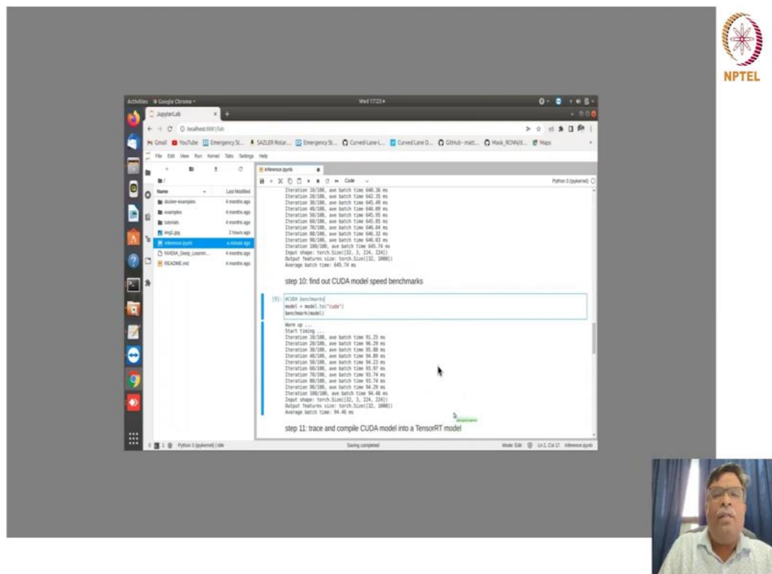


How much iterations it is taking how much time it is taking with conditions of whether the device in the CPU whether the device is using CUDA and what when it basically tries to develop a TensorRT model?

(Refer Slide Time: 20:23)



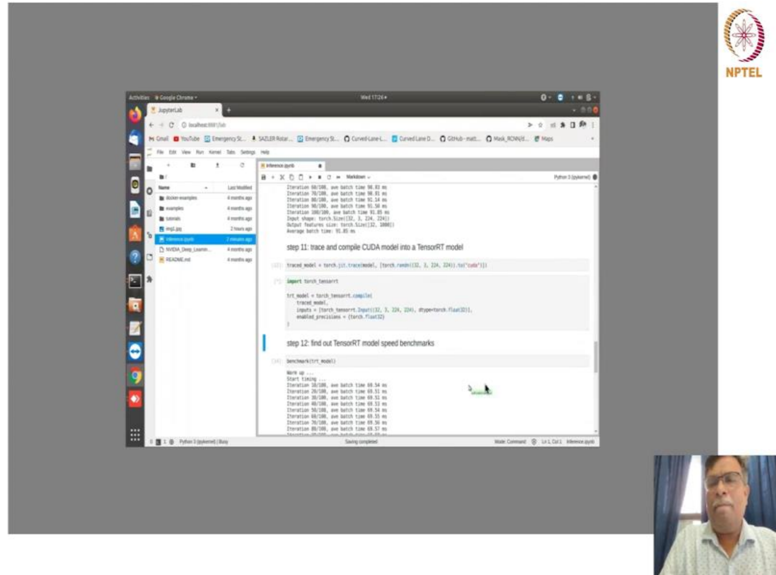
(Refer Slide Time: 20:30)



So, this is just a very very preliminary benchmarking thing which will help you to appreciate the optimize stuff right. So, that is what we are trying to do now. So, here so, if you run this ok which we have just now run it basically gives you 645.74 milliseconds right the average batch time and we are running it for CUDA here, which is executing which is telling you that it is giving you 91.85 milliseconds.

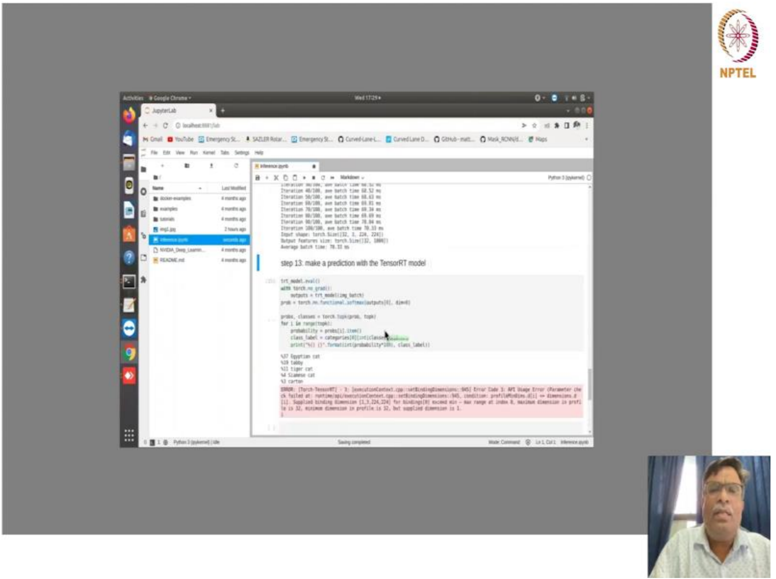
So, the idea here is you are trying to run the same model on a CPU which gives you 645.45 milliseconds 7 sorry 654 644.74 milliseconds. Whereas on CUDA ok the same model gives you 91.85 milliseconds for actually training right now TensorRT model if you see take a little bit of time yeah.

(Refer Slide Time: 21:31)



So, you see this it takes about 70.33 milliseconds right.

(Refer Slide Time: 21:47)

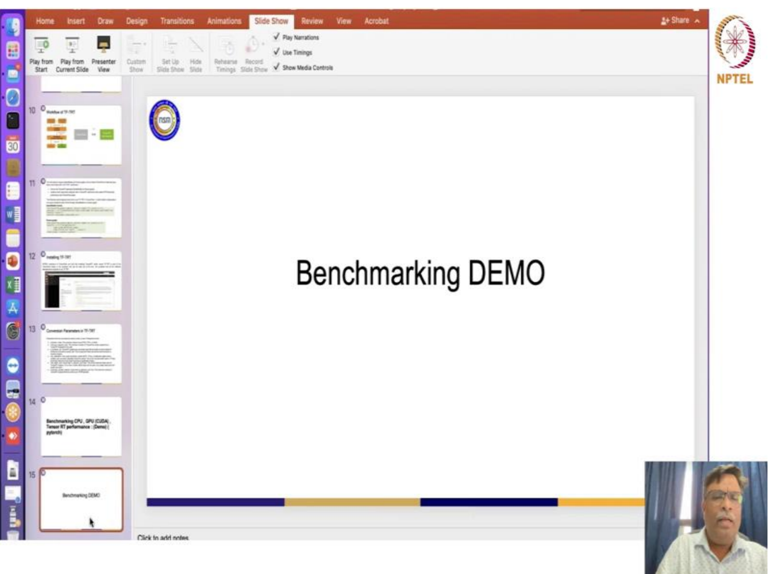


So, the idea is from 91.85 milliseconds to 70.33 milliseconds right. So, this is basically like a very very good what to say reduction in time optimize time ok because this is a very very simple example nothing we have done too much ok. And if you see the prediction right it gives you almost the same prediction right. 57 percent prediction gap almost the same everything is same right.

So, if you see here its 37 19 11 4 and 3. The same thing we get because we yesterday we are talking of even if it is reduced prediction sorry reduced precision right to get almost the same thing right there is nothing much of a difference ok right.

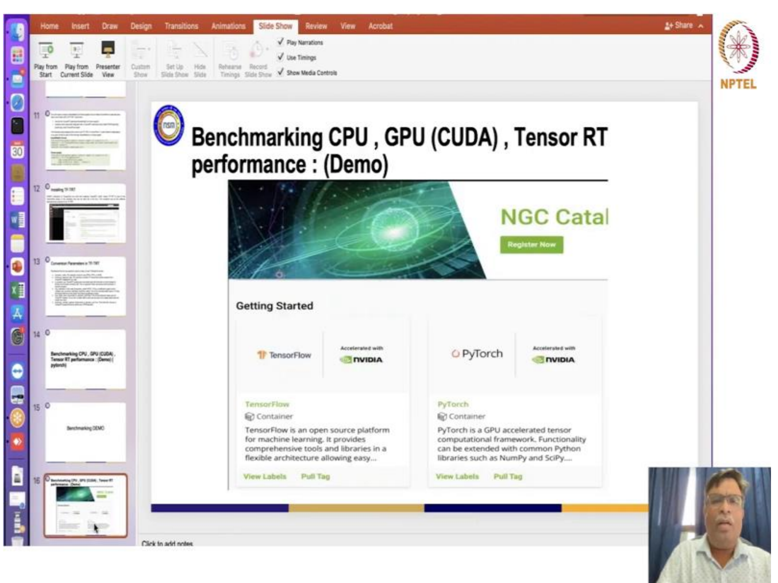
So, this is what we thought we will show you first. And then now so, now, once we have done this.

(Refer Slide Time: 22:47)

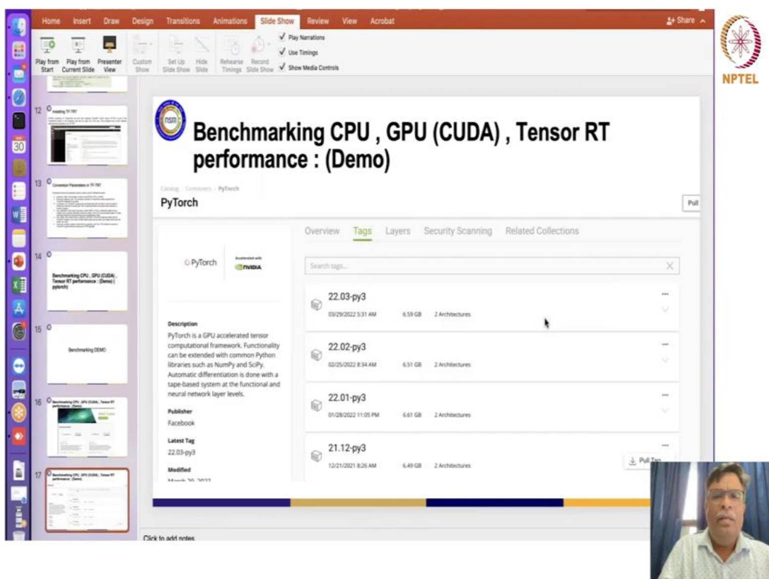


The benchmarking demo let us try to understand right step by step of how we have done it actually.

(Refer Slide Time: 22:49)

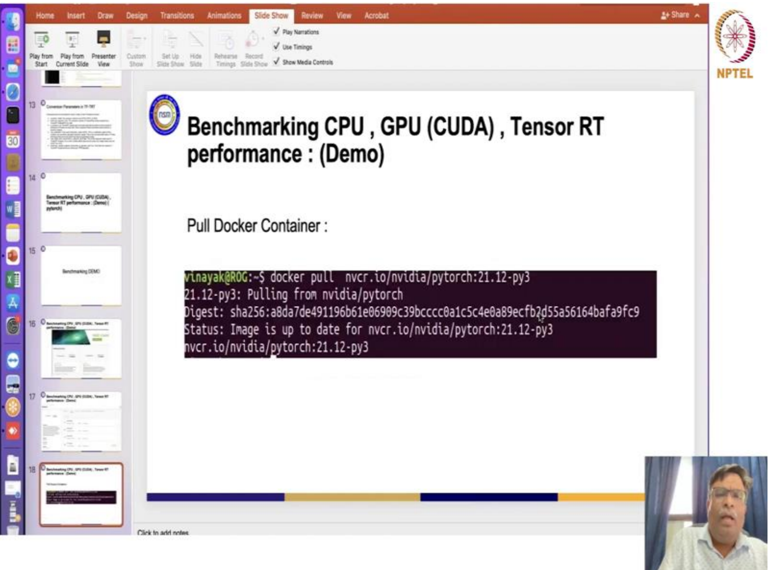


(Refer Slide Time: 22:52)



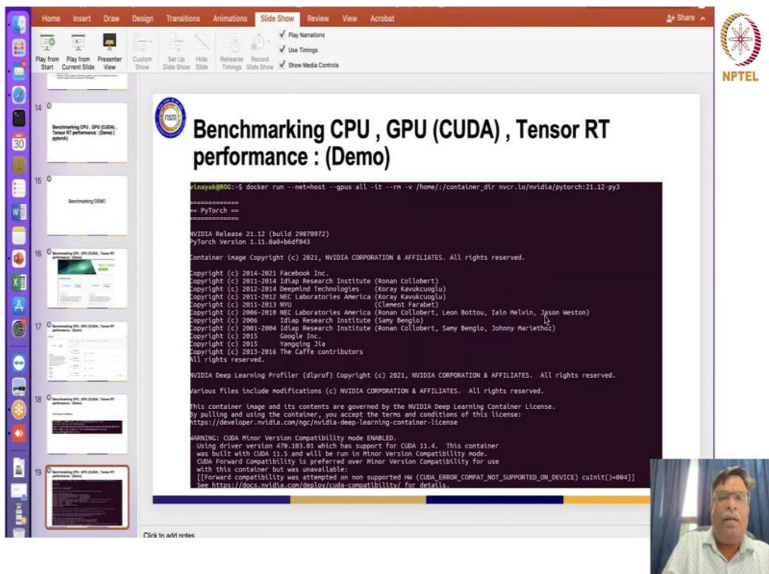
So, what we do is we basically as I told you we are supposed to be pulling the docker containers ok from the NGC site right. So, we are we have tried to download the PyTorch this thing ok.

(Refer Slide Time: 23:14)



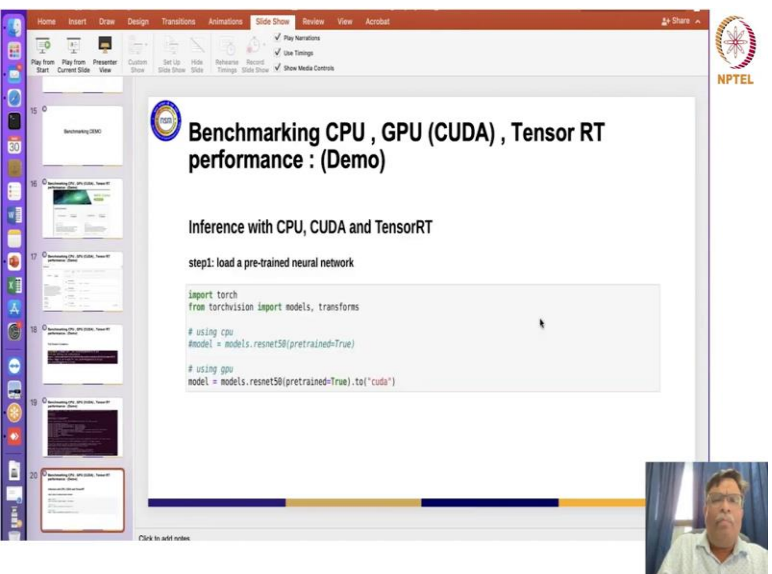
And then we basically pulled the docker container. So, we are actually telling you step by step of how we did it right all of this is executed. So, whatever we try to do till now we have actually tried to put it in the slide so, that tomorrow when you want to do it ok, you can actually see it automatically and do it step by step right.

(Refer Slide Time: 23:35)



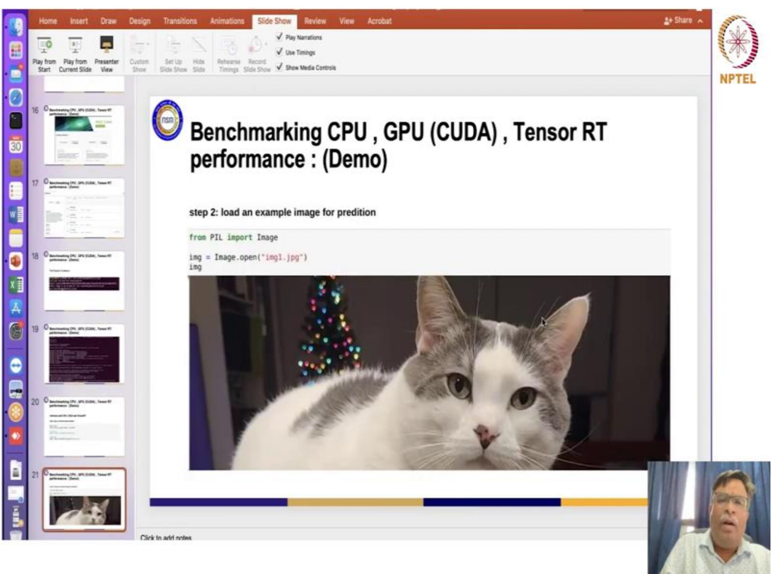
So, this is how when you run it you basically try to run that program ok.

(Refer Slide Time: 23:41)



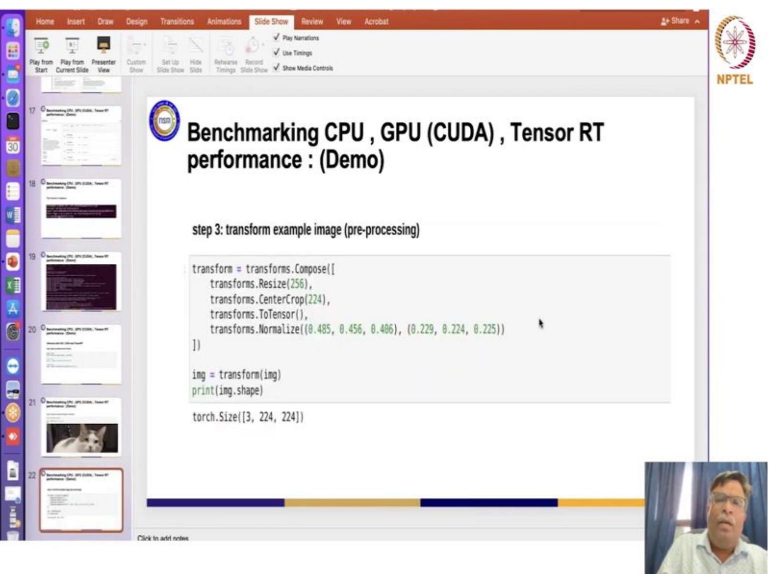
Then you know the pre trained network.

(Refer Slide Time: 23:44)



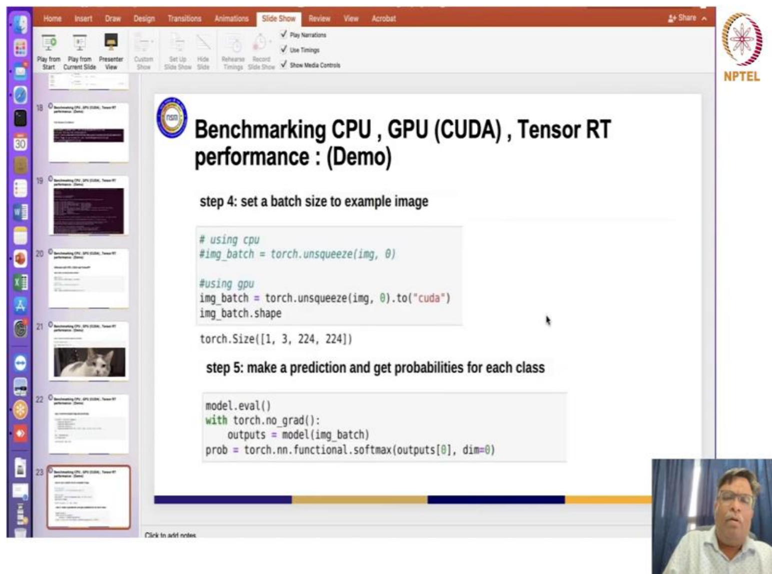
You get the image for production.

(Refer Slide Time: 23:46)



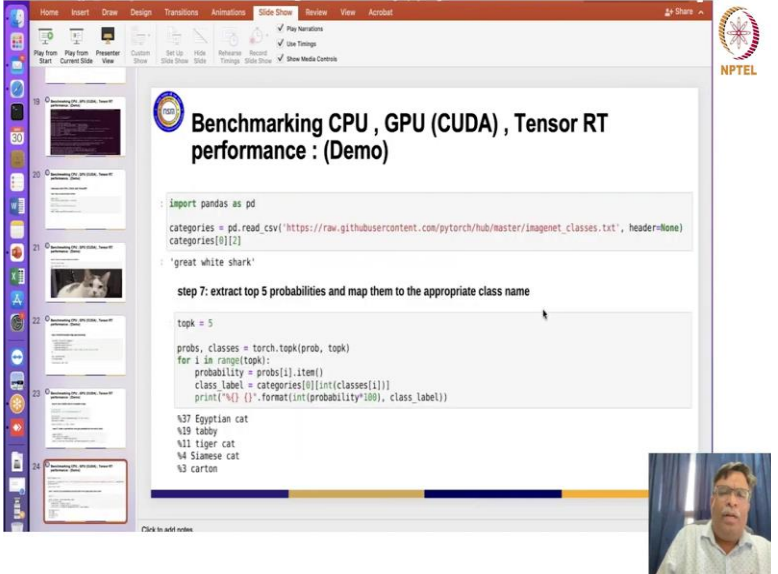
Then this is the transformed image.

(Refer Slide Time: 23:49)

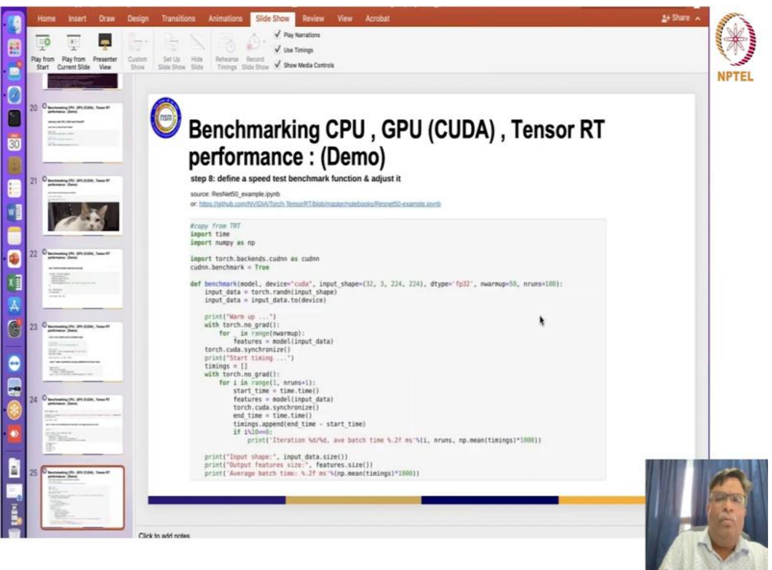


Then you see the batch size probability prediction for CPU ok.

(Refer Slide Time: 23:53)

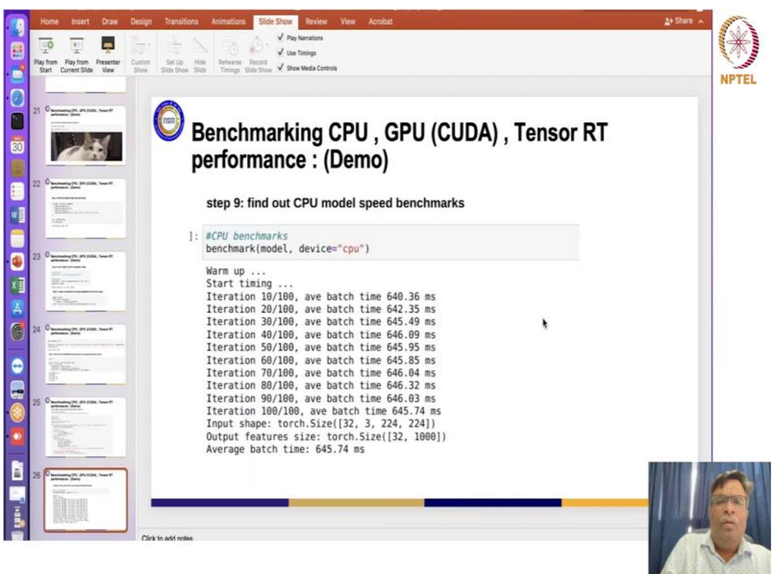


(Refer Slide Time: 23:54)



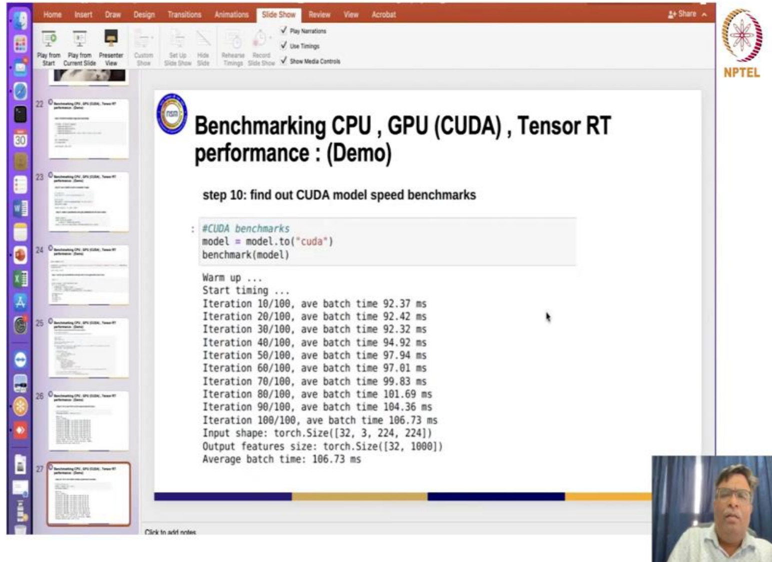
And then you do it for GPU.

(Refer Slide Time: 23:59)

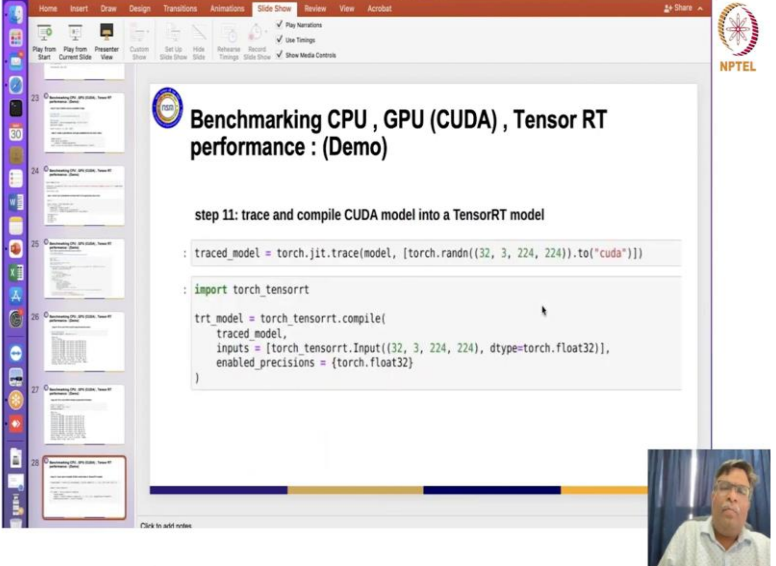


The CPU benchmarking, the GPU benchmarking.

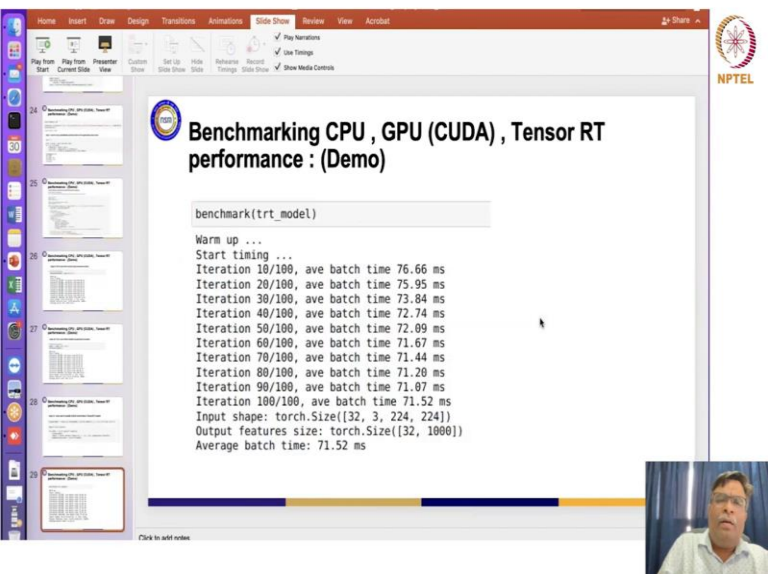
(Refer Slide Time: 24:01)



(Refer Slide Time: 24:03)

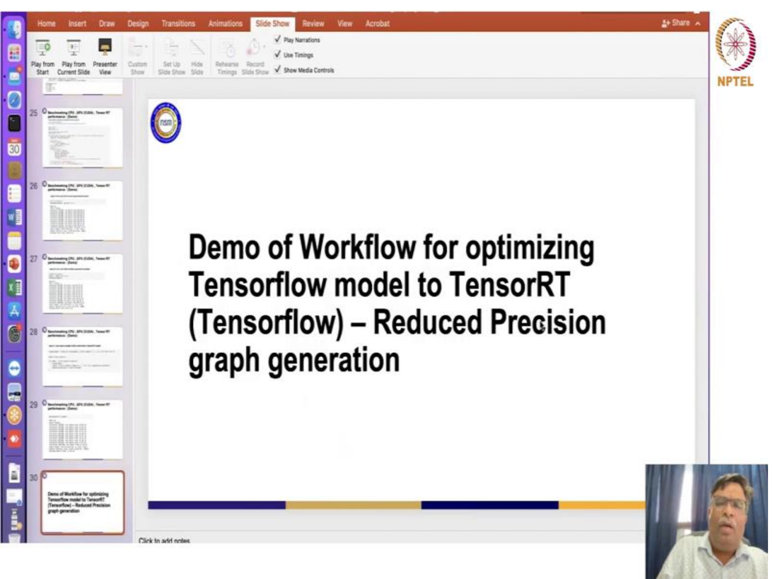


(Refer Slide Time: 24:04)



And then TRT model benchmarking ok.

(Refer Slide Time: 24:07)



Something like this.