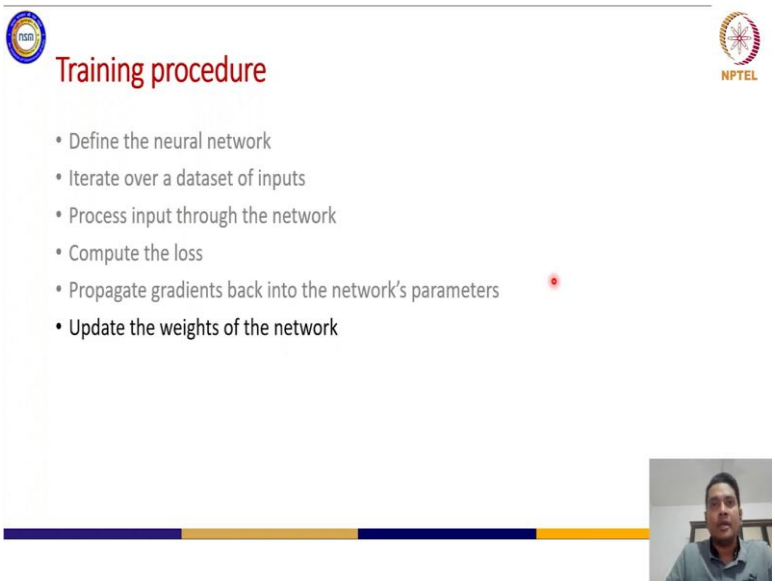
Applied Accelerated Artificial Intelligence
Dr. Satyajit Das
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 20
Introduction to PyTorch Part - 3

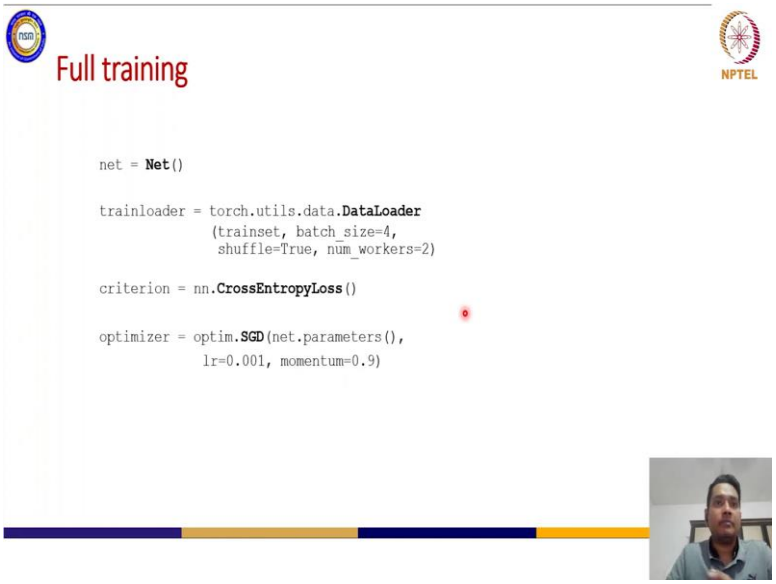
Good evening everybody. So, in the last session we started with the entire training pipeline for implementation in PyTorch for different neural networks. So, the pipeline is almost similar. So, if we just recap, if you can see here.

(Refer Slide Time: 00:38)



That first we have to define the neural network then, iterate over the data set of inputs and then process the inputs through the network, because you have to do the forward pass. And then after that, after the prediction output from the network you need to compute the loss and depending on that loss you need to update the gradients with for the parameters and then, you need to update the weights and again you do the forward pass.

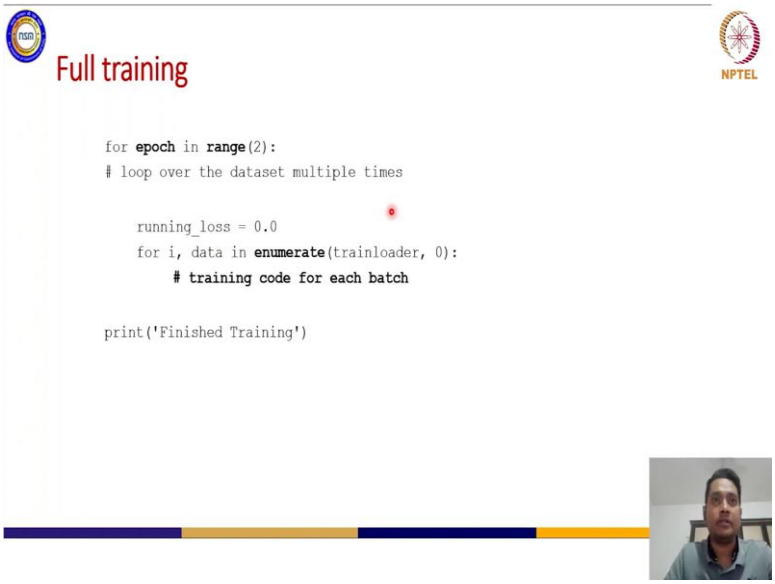
(Refer Slide Time: 01:21)



So, this is the entire pipeline that we talked about. And also we have seen how to implement that pipeline with PyTorch. So, first we have to define this network, then the data loaders for training and which different parameters that we have seen and then we have to define the optimizer. So the what kind of loss we are trying to compute in the end of the forward pass then, we need to compute the what kind of optimizer that we are trying to use for the network training.

So, several other optimizer are also there and with these certain parameters that you need to set the learning rate, momentum for stochastic gradient descent that you can see here.

(Refer Slide Time: 02:07)

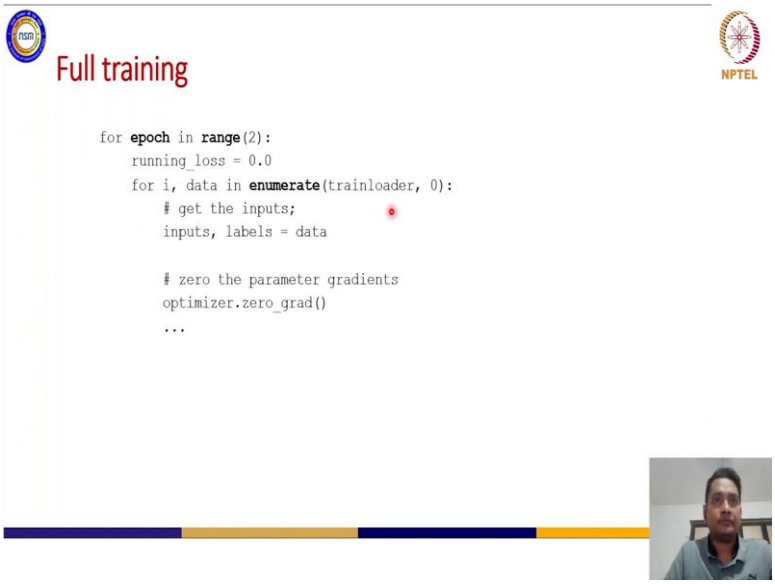


So, this is the step for defining your optimizer and loss function and once you have set up the training epoch so, basically where you will try to iterate through the entire data set of full training, ok. So now, you can take the data in batches. So, you can see that in this epoch, we are taking the data as batches from this training loader. So, a once entire training for the entire data is done which is the training data here, then we will go again for the second epoch and so on and so forth.

After the end of the epochs, all the epochs that you have defined, then you will just finish the training. So, this is essentially the loop for the training. So, defining the network and then you need to write the loop for the training. So, if you are iterating the training data set multiple times so that means, you are trying to improve the data performance you want to see the data set from several angle and that is why you will iterate through it several times.

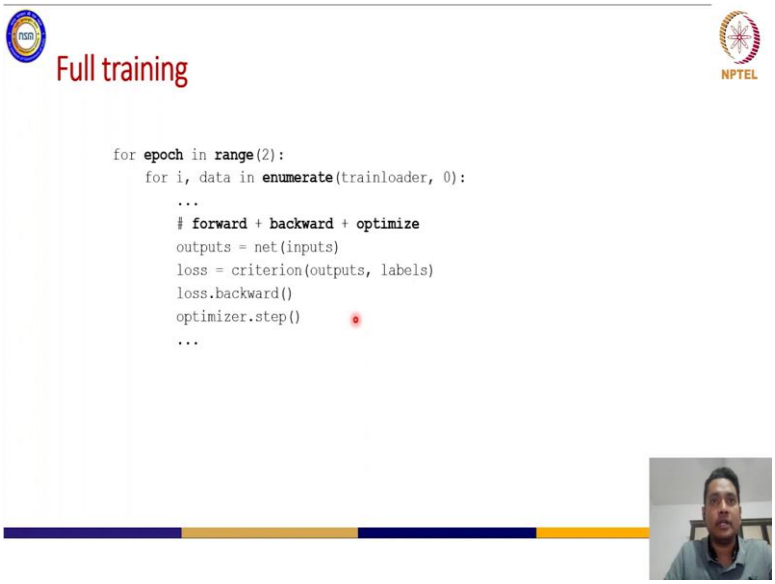
And you will see that the loss which will we will start with some random parameters. So, the loss will get updated and eventually we will get a minimized loss. So, that is the objective function.

(Refer Slide Time: 03:38)



So, after completing the loss and everything we need to optimize with the optimizer for the upgradation or computation of the gradient.

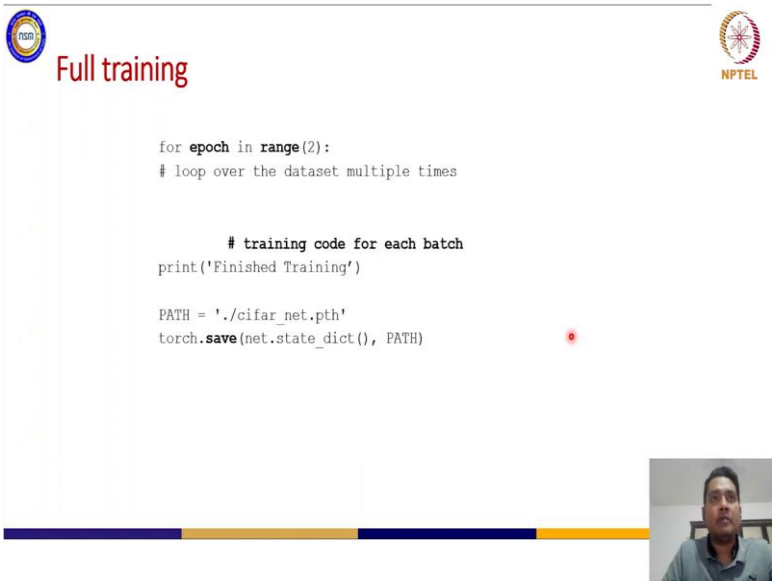
(Refer Slide Time: 03:52)



(Refer Slide Time: 03:56)

Upgradation of the parameters will happen in the next step optimizer pass dot step. So, this all this we have seen and we have also seen that you can write some statistics so, that you can see how the loss is getting improved or maybe how your accuracy is enhanced.

(Refer Slide Time: 04:12)



So the inter training pipeline we have talked about, but after the training what you will do is that you will try to save the parameters. So, for this you need to save in one path, defined by this path variable. After saving this parameter, so basically you are saving the network plus the weight parameters.

Because, when we load the network model, we load the inter parameter set that we have already stored and those parameters will be used for your inference or testing or validation. So, depending on what kind of problem you are targeting, you can if you want to validate your model with some validation data set you can do that using this model; you want to test some data set with this model that you can do also.

(Refer Slide Time: 05:10)

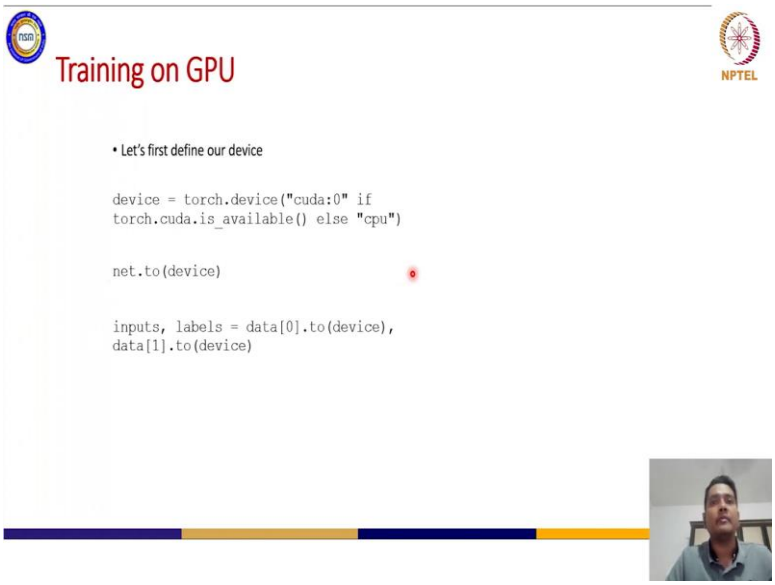
So, all this is the next step when you have already saved the more. So, for testing you need to load the test data which is from test loader that we have seen defining previously. And in the images and labels we will just load the images and labels. So, these labels are not mandatory because, for training you will compare the training output to these labels, but for testing you do not need test labels because, network will predict the labels. So, we are trying to target one classifier model, so that is why different labels will be there for different classes.

Now, defining the network so, initially initializing this network as net and then, you can actually load the model from this path where we have defined and saved it and then you will output. So, basically when you call this net with the images that you have taken here, you are doing one forward pass and you will get some predictions.

So, these predictions we are actually storing it in the predicted. And if you want to compute the prediction with the actual truth which is the labels then you can do that. You

can see how much whether the prediction is accurate or not comparing the labels that you have here.

(Refer Slide Time: 06:33)



So, this is very typical testing set environment very simple to implement in PyTorch. Now, we will see how to define our model which will run for GPU and CPU, ok so, that we will see. So, you need to define one device variable, where you want to keep whether you want to run your model inside the cpu or inside the GPU. So, if you have GPU available this code torch dot cuda dot is available will be true then, you will get the cuda code as let us say cuda 0 if you have one GPU and if you have multiple GPUs then it will take one list of ids.

So, basically 1, 2, 3, 4, 5, 6, 7, 8 depending on the how many number of GPUs you will have on your system. Or else so, if this returns false then you will take it as cpu because, then you would like to run your entire model inside the cpu. Now, as I have mentioned in the previous class that tensors are compatible for both GPUs and cpus. So, if you are not having GPUs then also you will be able to run your code inside or tensor codes inside the cpu as well. Now, once you have defined this device then you have to transfer your model to the device, because the model will run in the device.

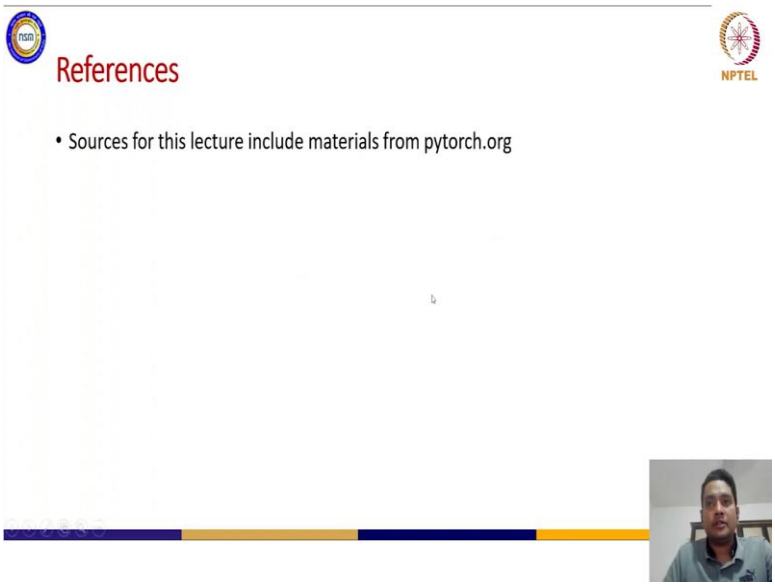
So, now we are talking about running in one GPU ok, because you can see cuda 0, 0 is the device id. So, cuda core which is the GPU one GPU which is available for this system. If multiple GPUs are available then also we are taking the first indexed GPU

which is cuda 0. And we are transferring the model into the device which is net.(to device).

So, anything you want to transfer your device, you can call this to function. So, I also want to transfer our data and labels to the GPU calls because, if the GPU is running the model; that means, the GPU will need the data and the labels to make the count for your loss and then it will actually update the parameters depending on the loss and computed gradients.

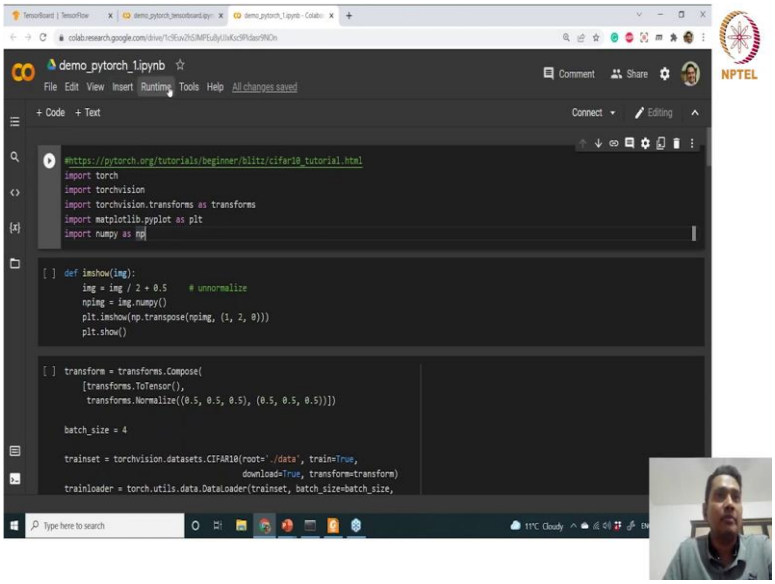
So, you need to transfer the data[0].(to device) and data[1].(to device), because here we have two sets of data 1 is for your input original data and for your labels. So, both we are sending to the device. So, this is how you can transfer your inter model and data to your device and it will run the training that you want to run in the GPU, right. So now, we will go directly to our code where we will see how to actually run.

(Refer Slide Time: 09:41)



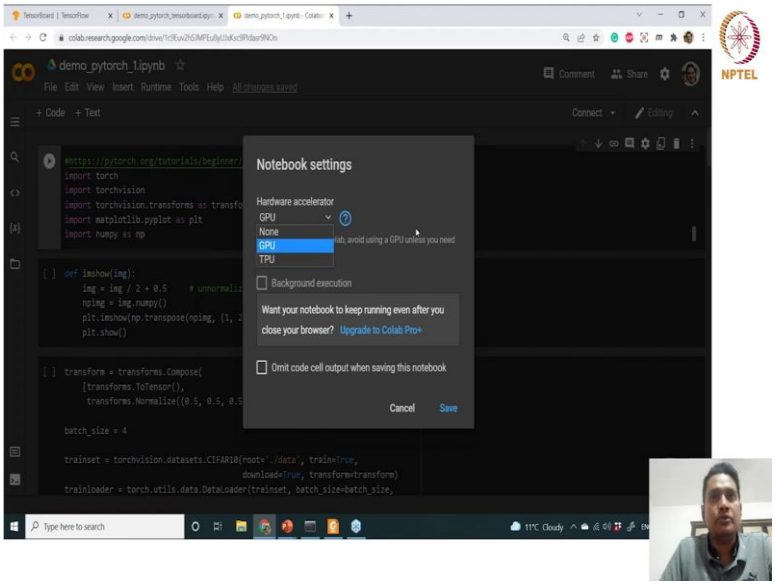
So, you can, for details of the materials that we are using here you can go to pytorch.org for the details of all these functionalities that we have discussed, ok. So, we will just now go towards the actual training where what is happening let us see.

(Refer Slide Time: 10:00)



So, we are running this training in colab. So, everybody who are attending this course you can actually take this piece of notebook file which is .ipynb and you can run that in any colab account, ok.

(Refer Slide Time: 10:37)

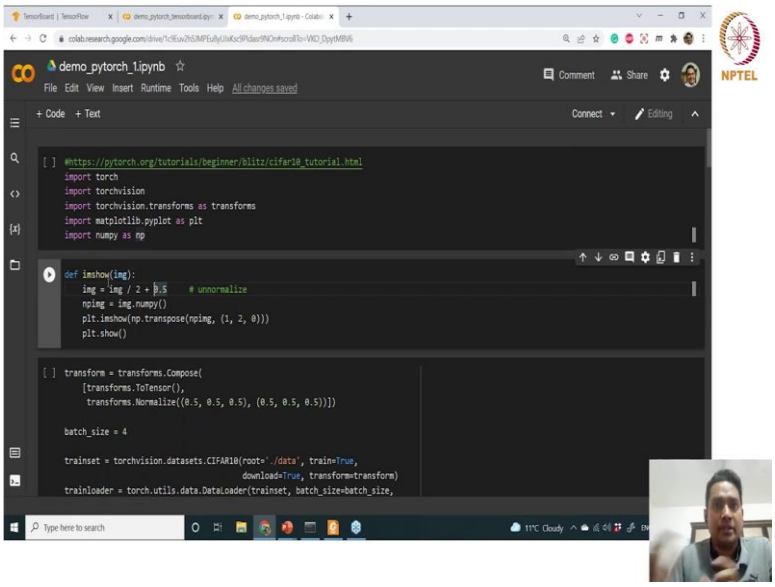


Now, if you want to enable the GPU access for your colab you can go to your run time and you can change the runtime type to GPU or TPU depending on what kind of architecture you are targeting. If you select none then, automatically only cpu will be

selected for running your training modules and if you select GPU your network will be done on GPU and if you select TPU it will run on TPU.

So, TPU is the tensor processing unit, ok. So, in this course we will also see how to accelerate our training with using TPU as well. So, today we will see how to do that with GPU for single GPU and multiple GPUs how to scale our training also we will see, but for your running you can see you can select either GPU or TPU. So, for now you can select GPU.

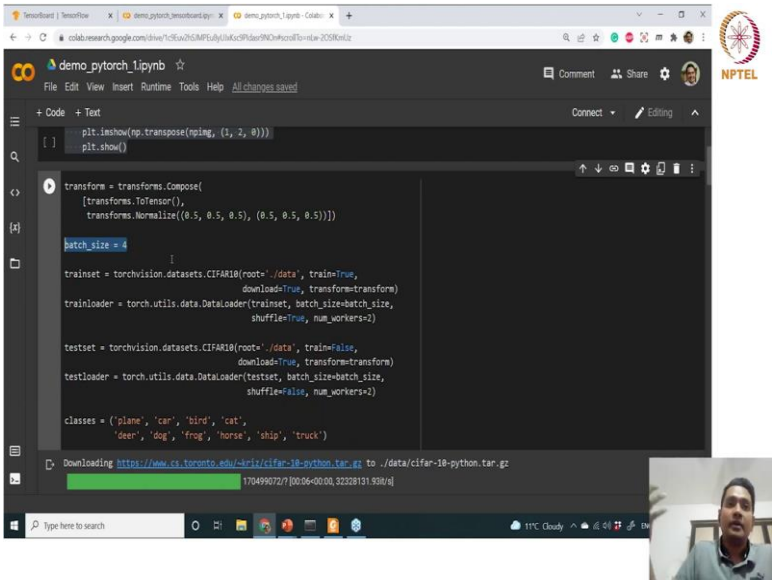
(Refer Slide Time: 11:29)



Now, you can see in this section of code we have imported the required libraries and packages torch vision as I was mentioning that once you are working with images and videos torch vision is very very handy and for different transformations. So, that for that we are importing those libraries. Matplotlib for matplotlib and numpy these are two stand up libraries for plotting and working with different data sets like array and so, right.

Now, we are defining one helper function for displaying the images into one corridor. So, basically this function will show some images. So that, whatever image that we will download from the data set that is available already inside the torch vision library that we will try to visualize with this function, ok.

(Refer Slide Time: 12:33)



Now, next as I was mentioning the a remember that training pipeline we have to define the transformers. So, basically what kind of transforms you will apply on the data. So, here we are normalizing the images with this mean and standard deviation. Now, we are defining the batch size as 4, now remember again for the training we are not giving individual images ok, we are giving the images as batches. So, your tensor dimension will have the batch size in the front, ok.

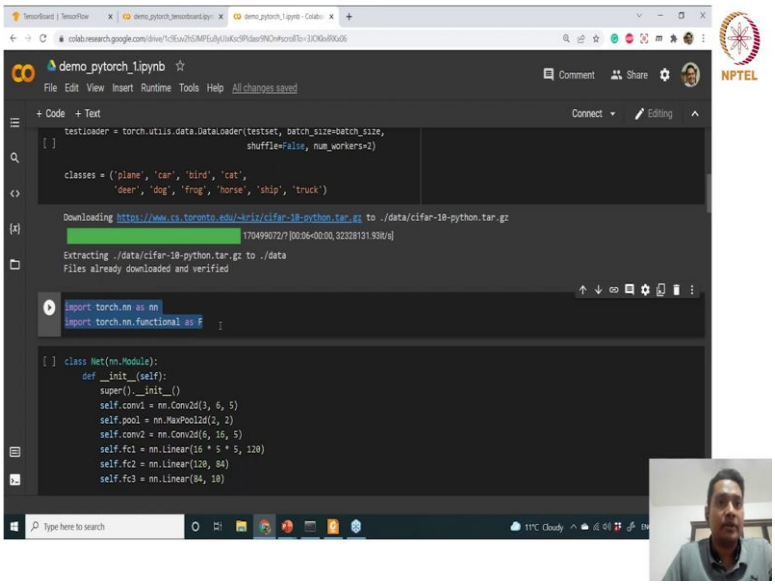
So, if you are interested in giving only one batch size like, one single image then, batch size will be one and if you want to increase the batch size and remember if you increase the batch size the memory required to transfer to your GPU that will also increase because that many number of images that you want to store also the gradients for those number of images; the intermediate transition values that you want to store and those many weights also you want to store.

So, any increase in the batch size will take more memory inside your GPU. You need to be aware of that and in this session we will see how to actually last day many of you asked this question like this is one hyper parameter, so how we will define this and what will be the value for this for a particular model, right. So, for that you can do some exploration. So, what kind of batch size is giving you better performance and better accuracy so, that you can explore.

Now, training set train loader, test set test loader, that we have defined as discussed in the lecture. Now, as we had seen that we are take taking CIFAR10 data set. So, CIFAR10 data set has 60000 images with 32 by 32 dimension. So, for 10 classes it has all the images, different images from different perspective, ok. With different resolution with different angle of capture and so on and so forth. So, that the training module that we want to train learns better features from these classes.

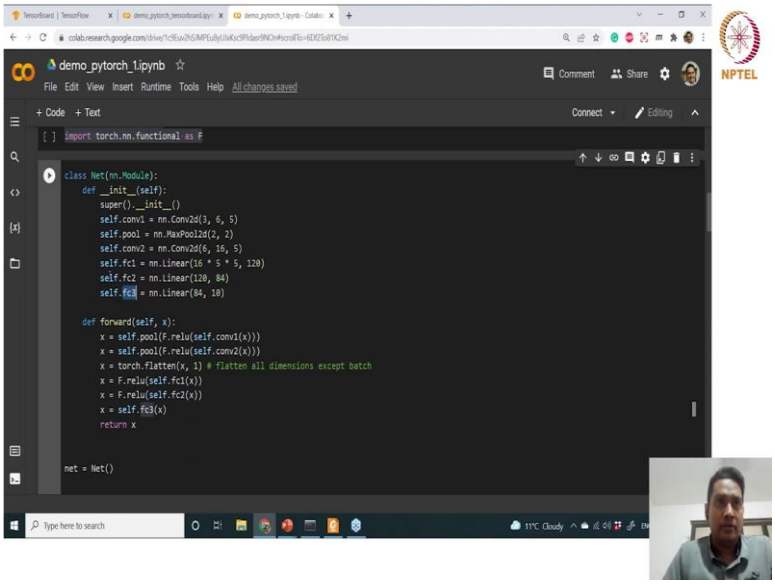
So, these 10 classes that you are seeing here that we will use to actually calculate the accuracy for each class, ok. If so, this is very optional to your training pipeline, but some kind of analysis that we want to do. So, if you want to apply some analysis into your training model that it.

(Refer Slide Time: 15:24)



So in this, then for the training we need that torch.nn which is we will be using as nn and then, we will need the functional api for the relu function and all these functions that are already available in inside this library so, that we will take help off.

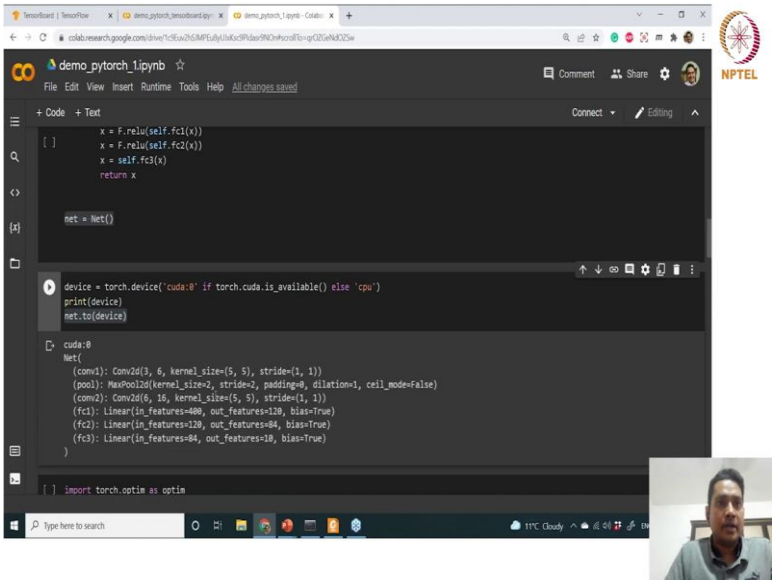
(Refer Slide Time: 15:44)



Now, we are defining the network here. So, basically in the initializer, so this class net as I was mentioning you can name it rename it to any name as your preference. And then, the definition of the networks so, we have first convolution layer then max pool layers so, this max pool layer if you want to define as a specialized with let us say 2 by 2 kernel. So, that we can define also convolution 2D so, this 2 or 2 convolution layers and 3 fully connected layers which are the linear layers that we as we say.

So, after the definition we have this a forward pass which will take the input. So, basically x is the input and it is return in the prediction as x which is the output of this fc 3, ok. So, this is just one pass ok forward pass. And so, initializing the network as net, right.

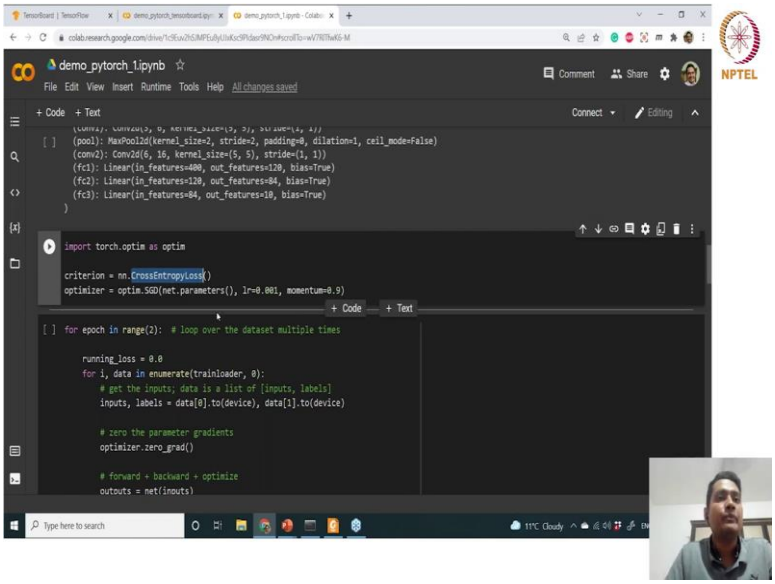
(Refer Slide Time: 16:53)



Now, we are transferring this module or net to the device ok. So which is basically cuda 0 here. So, if you are having activated run time as GPU it will take the GPU as the target device. So, you can print the target device as well. So, and see that we have a cuda 0 available and the network which is transferred to your device. What are the things that are transferred you can see the entire stages, ok.

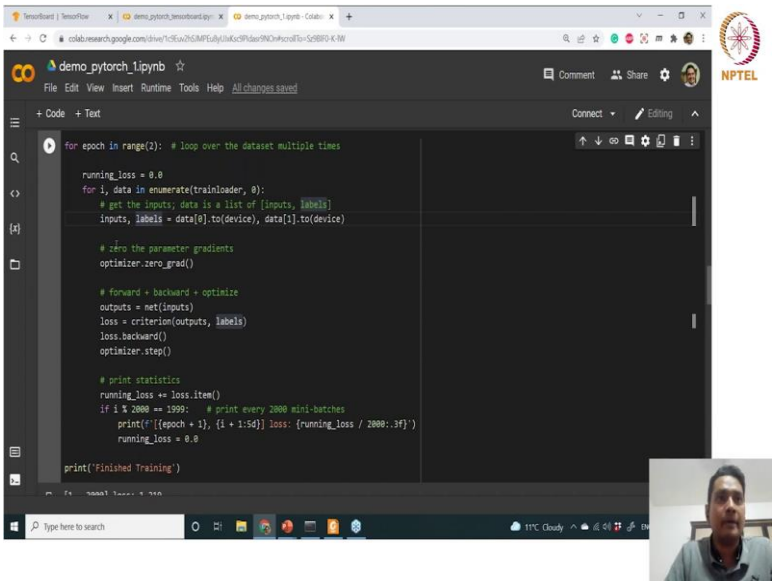
So, convolution layer 1, a pooling layer convolution layer 2, a fully connected layer fully connected 2 layer, fully connected 3 layer. Now, we have defined only one max pool 2D, but this will also be reused in and after this convolution layouts, ok. So, that is fine.

(Refer Slide Time: 17:50)



Now, in the next section we will see to define the optimizer. So, this is also we have seen. Now, what are the values for learning rate and momentum for your SGD? So, here we are using SGD depending on the model training what kind of model you are using and what kind of optimizer that you will use you can see several other optimizers are also there to compute the gradients from the losses, also several other losses are also ok like mean square loss cross entropy loss we are using here.

(Refer Slide Time: 18:25)

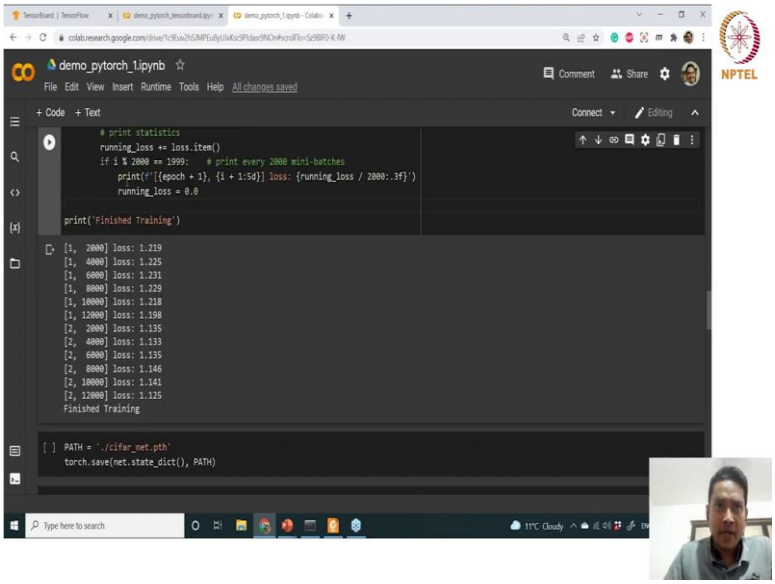


Now, this is your training loop, ok. So, basically once you have defined the network you can see that inside this epoch. So, we are running this entire epoch which is going through the entire data set twice ok. And then, the running loss we are actually be initializing to 0 and for i data which is from the training loader now training loader will have the entire data set and as batches we are actually transferring into the data inputs, ok.

So, inputs we will segregate the data in inputs images and labels here we are working with images as data. And then, we are initializing the 0 grad and this is the forward pass. Forward pass, the plus your loss which is useful in the backward pass. So, you can see that only one line of backward pass and step is the update.

So, this is where actually entire computation is happening and this loop will run for how many times how many data you have depending on how much batches batch size you are having access in each iteration. Now, we are having this statistics computed because we want to just print like how many loss we are improving.

(Refer Slide Time: 20:01)

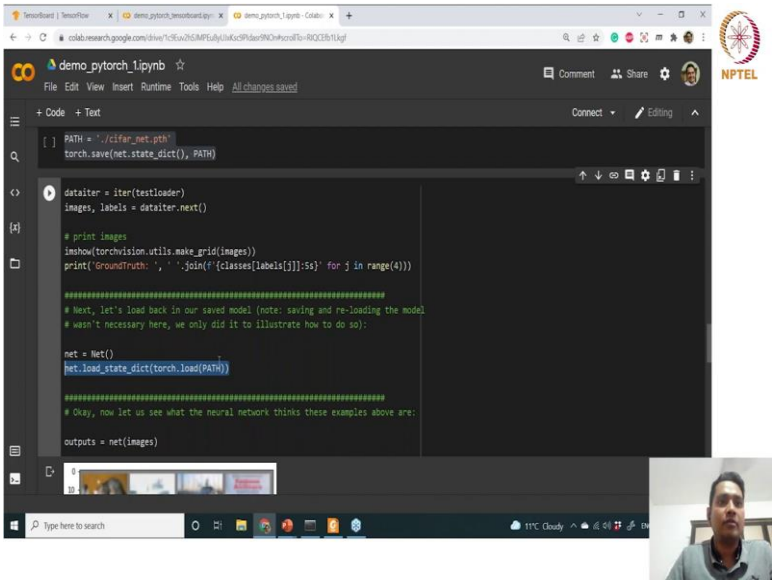


So, you can see that after so after so this is the first epoch and this is the second epoch as you can see, ok. So first, so you can see that we are giving print the epoch and the how many date data batch that we have processed. So, after every 20000 mini batches we are actually updating this. So, the loss has improved from 1.2 to 1.1 ok and we have just

finished the training. You can go on with several other numbers of epochs or increased number of epochs to improve the loss, ok.

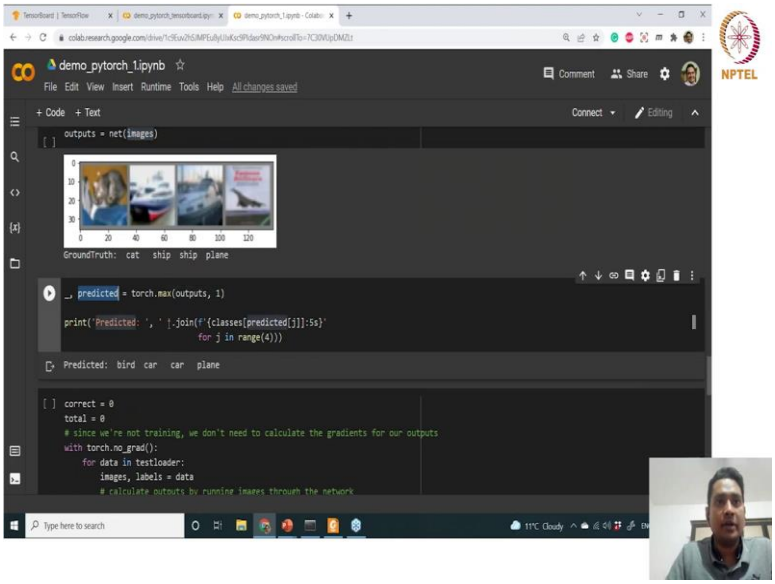
So, after the end of the training so, once you have computed the entire loop here so that means, you have finished the training process.

(Refer Slide Time: 20:58)



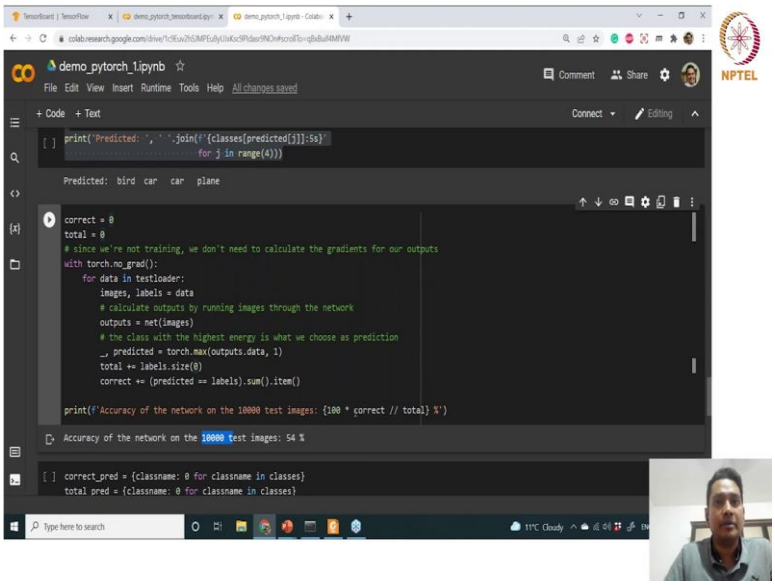
So after the training process, you will need to save the model and we will save here we are saving the model inside this. And then, we are just printing the images just to see what kind of images are there and then, we are loading that model again inside this net ok, because we want to test now, right. So, outputs test outputs is now images which are taken from the test loader, ok.

(Refer Slide Time: 21:27)



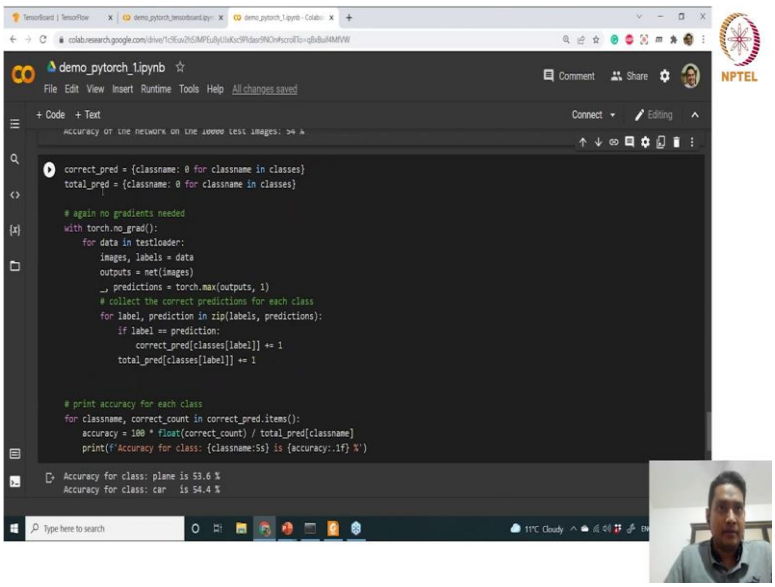
So now, we want to see what kind of predictions it did. So, outputs will give. So, ground truth is so, these are the images that we have loaded. So, basically cat ship and plane and then, the predicted value that we have predicted from this output, ok. Because, this output is essentially the predicted output from the network depending on these images that we have fit and this predicted value will have the predictions and the maximum predictions actually. So now, we are guessing what kind of class it has test or predicted.

(Refer Slide Time: 22:07)



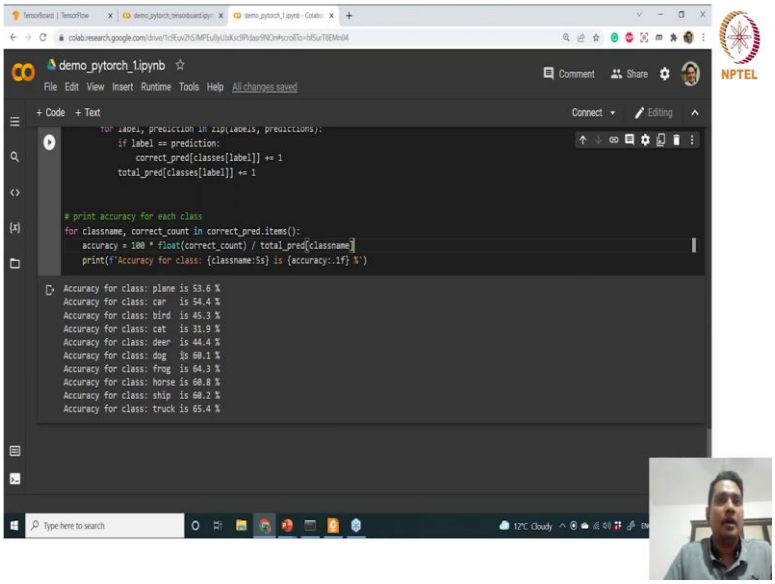
So, the last one only it has correctly predicted as well ok, because we have trained that model with very limited or low number of epochs, ok. So, if you increase the number of epochs and you can see whether your prediction is increasing or not, ok. That means, your prediction is getting better or not. Now, here we are just calculating some accuracy for each class ok. so, accuracy for the network for let us say 10000 test images what is the accuracy it is giving we are just computing that for percentage and we see that almost 50 percent images that that were guessed correctly or predicted correctly by this predicted output, ok.

(Refer Slide Time: 22:58)



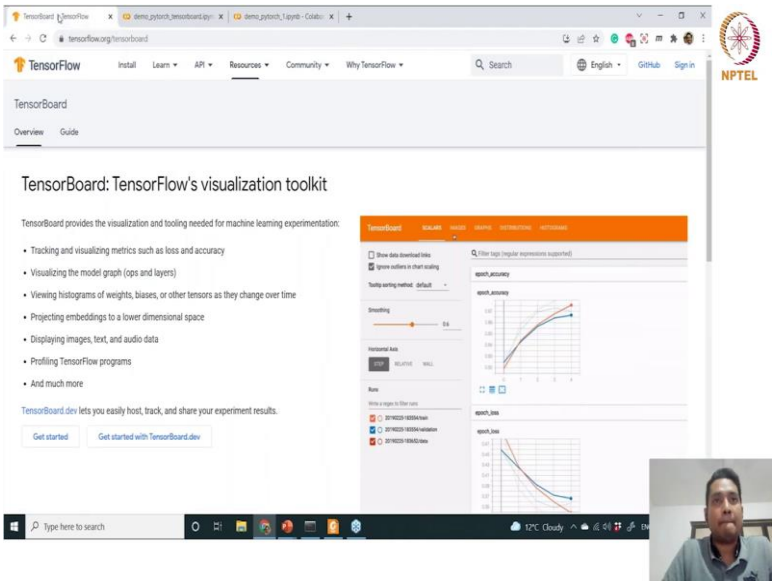
Now, so this is also optional just to see the statistics of each class prediction, ok. So, for each class ok so, we have defined this total number of classes as 10 ok and all these classes we have seen. So, print accuracy so we are calculating accuracy for each class and then printing the accuracy for each class.

(Refer Slide Time: 23:20)

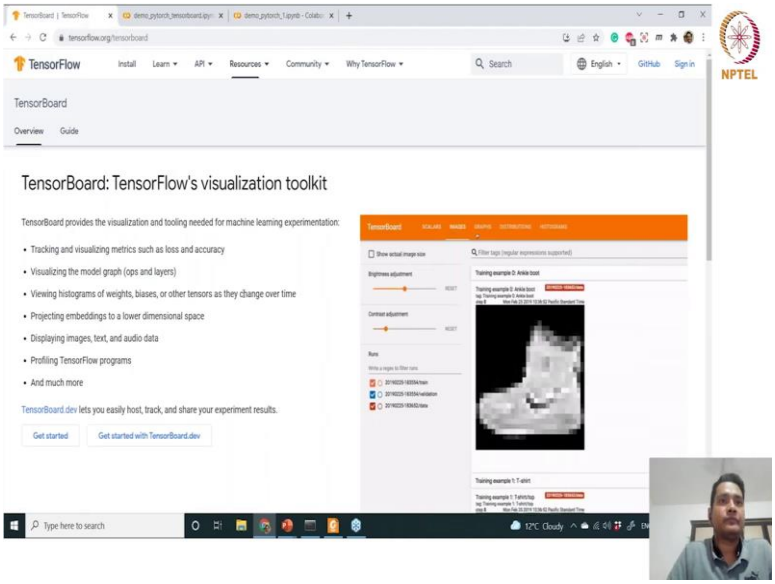


And you can see that almost 53.6 percent accuracy have got for plane class and so, this is class wise prediction, ok. So, this these are these are just some statistics that we want to evaluate after the training of the entire a network model, right. So, the next thing we will discuss is that how to profile our net training, ok. Because, this is very important when you are trying to tune for a particular data set and for a particular model that we want to training, ok. So, for that one handy tool is there which is the tensor board, ok.

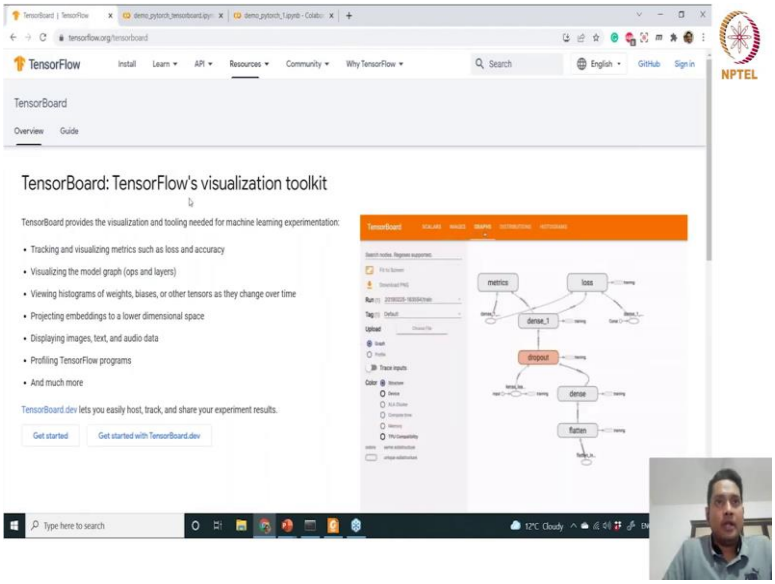
(Refer Slide Time: 24:02)



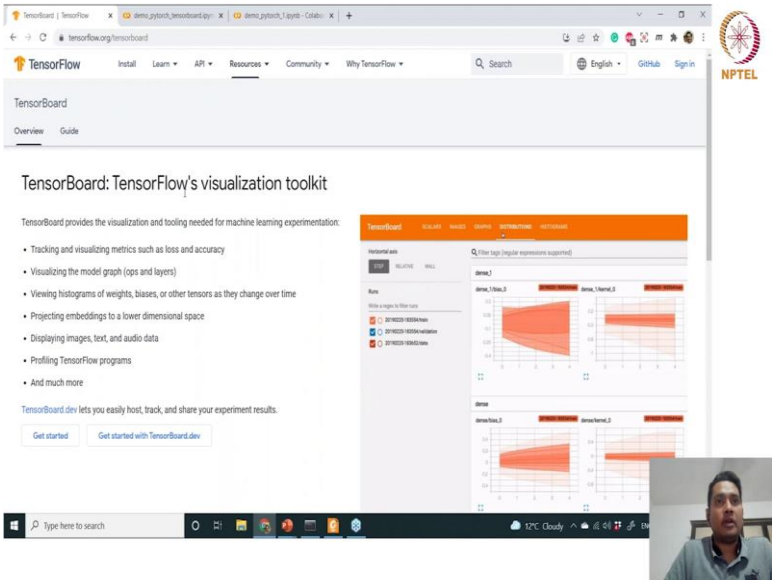
(Refer Slide Time: 24:03)



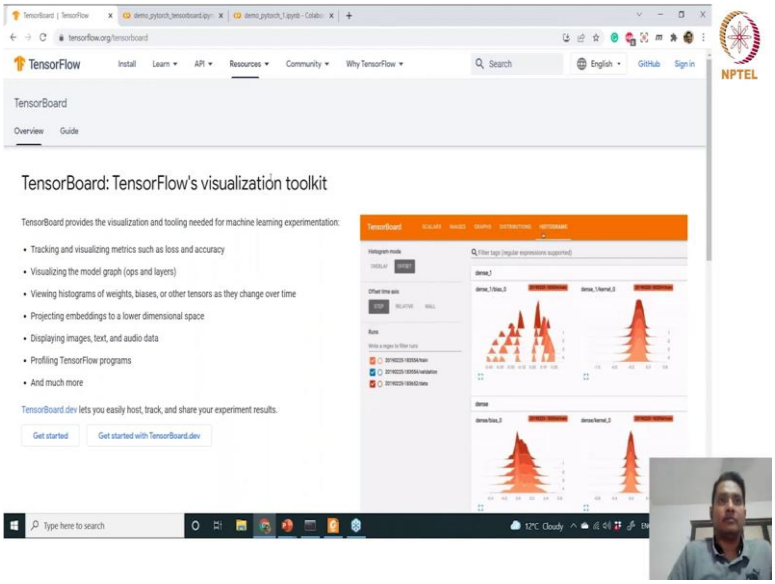
(Refer Slide Time: 24:04)



(Refer Slide Time: 24:06)



(Refer Slide Time: 24:08)



Now, tensor board is one visualization toolkit, it is from tensor flow actually, but you can use that using PyTorch as well ok. And it provides visualization and tool needed for machine learning experimentation. So, what kind of things we can see that you can see that here tracking and visualizing or metrics such as loss accuracy these are very very important parameter when you are actually training one network module, or any other machine learning model, ok. Visualizing model graph because, we have defined one a network model, right.

Now, when you are trying to optimize or let us say what kind of data types it is using how many parameters it is giving output. So, all these info you want to capture visually, if you have very very big training network then you can from the code to figure out what kind of output dimensions it is giving each layer and all these information getting is very hard. So, to visualize the entire model graph you can use this also viewing histograms of weights biases.

So, all these are actually parameters and your hyper parameters you can you can view in histograms. You can also project embeddings to a lower dimensional space, ok. So, this is also very handy feature you can display images text audio data. So, we are working with image data. So, we will display some image data to see what kind of images available inside this training image set that we have downloaded right, ok. And then, profiling tensor flow programs you can profile like what program module is using what kind of resources and all this evaluation you can do using this tool kit, ok.

So, today we will see some features of that and eventually we will quite we will do the other I mean go on learning some advanced topics. So, we will see more of kinds of more features, ok.