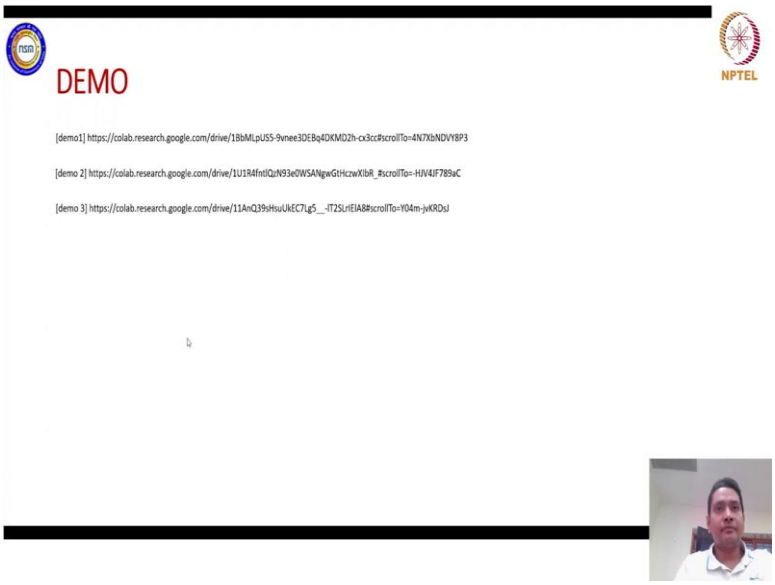
Applied Accelerated Artificial Intelligence
Dr. Satyajit Das
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

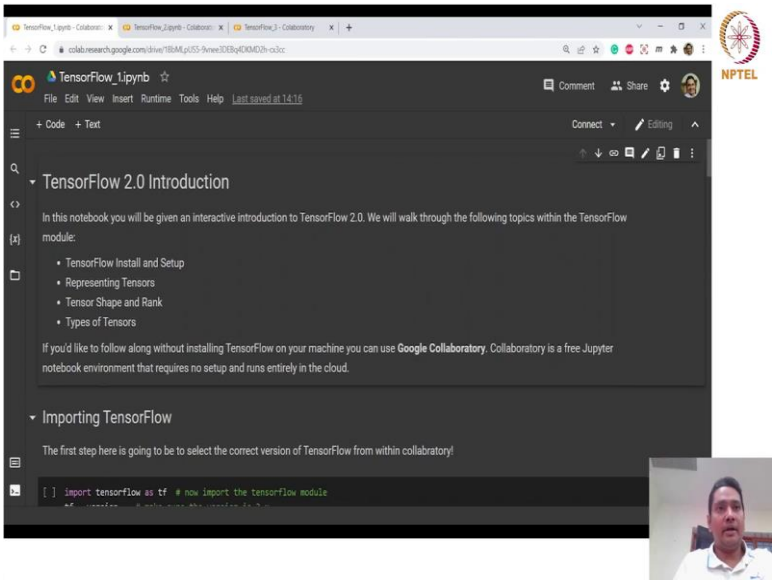
Lecture - 25
Introduction to TensorFlow Part - 2

(Refer Slide Time: 00:16)



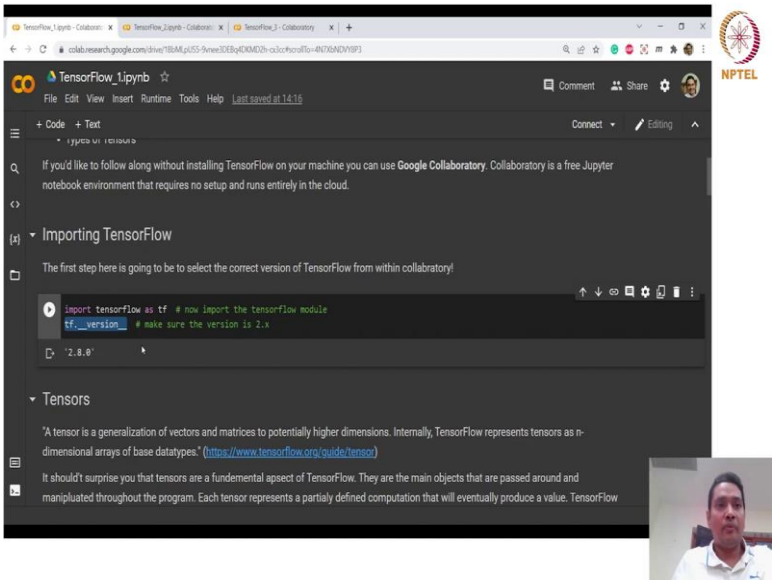
Now, we will go into the demo, where we will see some of the concepts or Tensor data ok. So, tensor data we have seen right.

(Refer Slide Time: 00:31)



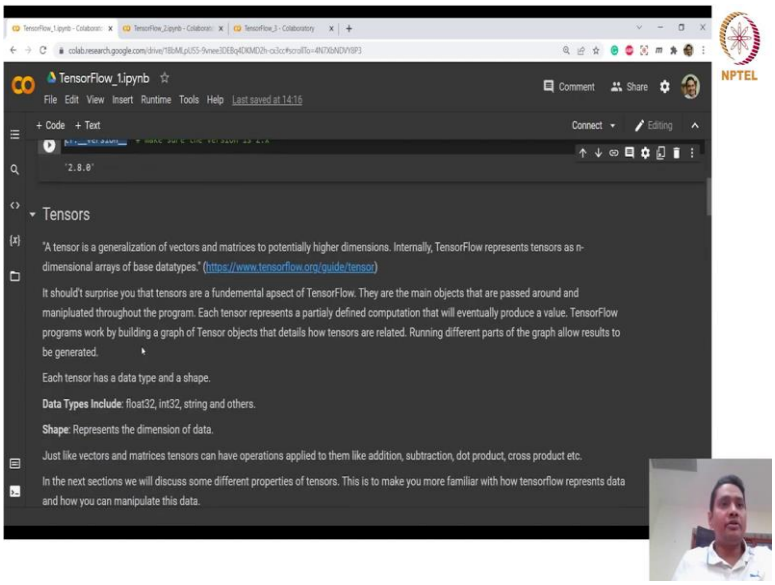
Now, how we can define the tensor data and how we can use different functions that is as similar to your numpy functions that is similar. So, you do use, so you will use colab I hope you are seeing the window for browser. So, here we are using colab to run this previous let us say a TensorFlow 1.

(Refer Slide Time: 00:58)

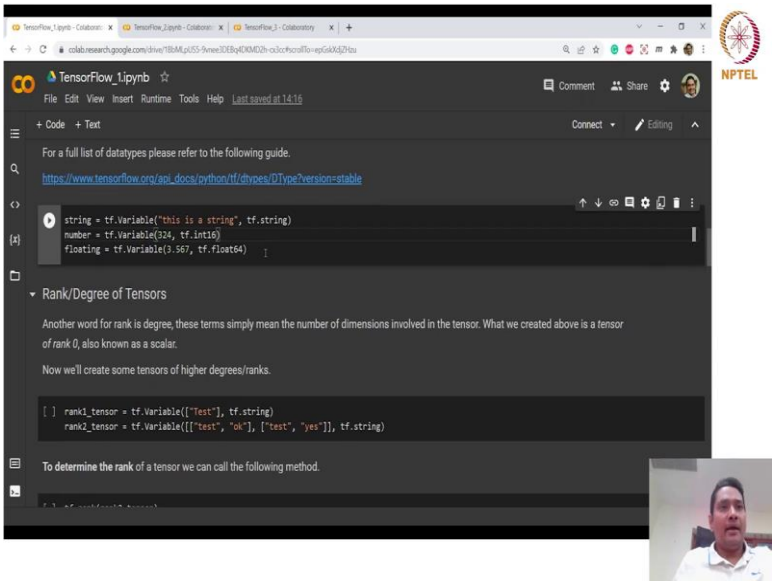


Now, to use tensorflow you just need to import. So, if you are using local machine, you need to install tensorflow and you need to check, which versions you are using. So, tensorflow version is very very important to confirm, so that you are not using tensorflow 1. And here in all the tutorials demo sessions we will use tensorflow 2.x any versions after that ok that is fine. So, if you are using your local machine, you should ensure that you are using tensorflow version greater than 2.4.

(Refer Slide Time: 01:39)



(Refer Slide Time: 01:43)

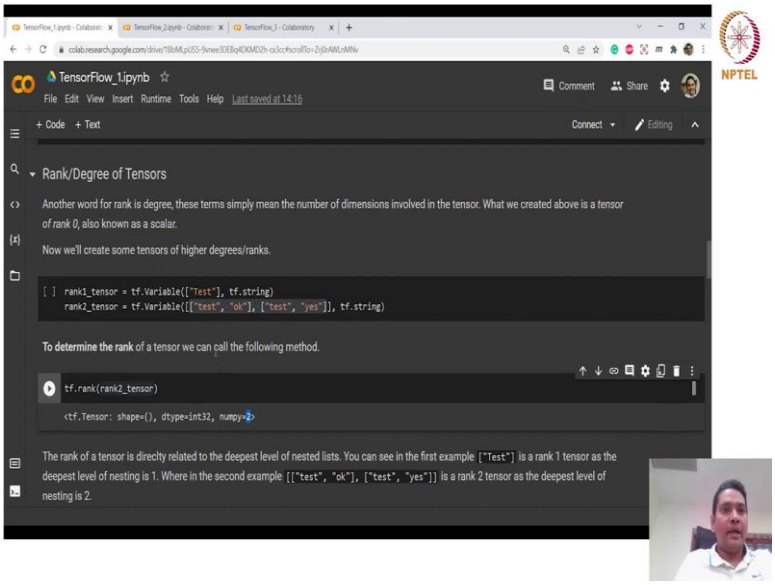


Now, tensors as I was mentioning, defining tensors is pretty easy. So, basically this is a string tensor, which we are defining as variables, right. Now, this is a single dimensional scalar with rank zero string ok and this number is again another rank zero tensor, which is again a variable floating variable. Now, we can see that tf.data type we can use ok. And several data types you use.

Now, these data types is very important in model training because in the previous class also we have seen that with mixed precision training, we can enhance or improve the performance manifold. And using different types of data and exploring your design model is very very important to see how you are improving your performance. Both

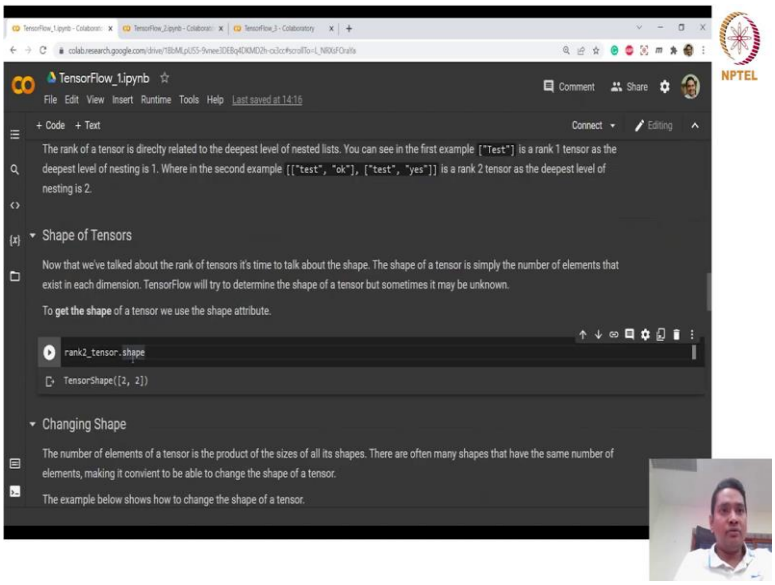
performance and how it affects your accuracy. Now, rank as I was mentioning, that if you want to see that rank let us say I want to see this tensorflow's rank. So, basically this is a single dimension and this is two dimensional right.

(Refer Slide Time: 03:00)



So, the rank of rank 2 tensor is basically numpy 2 ok. So, 2 rank. So, this is the 2 dimensional tensor. So, you have 2 rank and the shape of the tensor. So, rank2.shape.

(Refer Slide Time: 03:13)

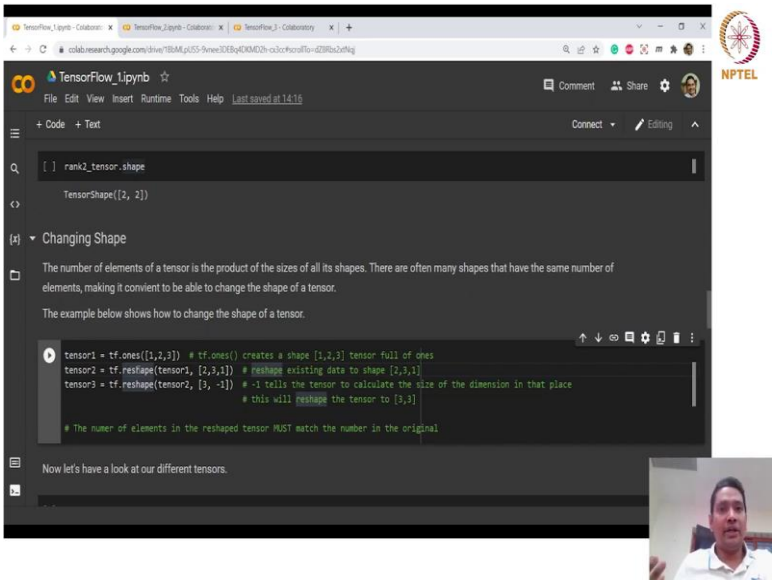


Now, 2 rank means, it has 2 dimension and in each dimension you have 2 elements. So, 2 by 2 is essentially your shape right. So, understanding shape of the tensors and rank of

the tensors these two are very important, because to compile the model that you have defined you will get different shape and ranks of the tensors. So, rank is essentially how many dimensions are there. So, let us say you if you are training the model using batch.

So, how many batches you are using in each layer that will come as rank ok. And the shape then how many elements in each dimension, so that is the shape that we will have.

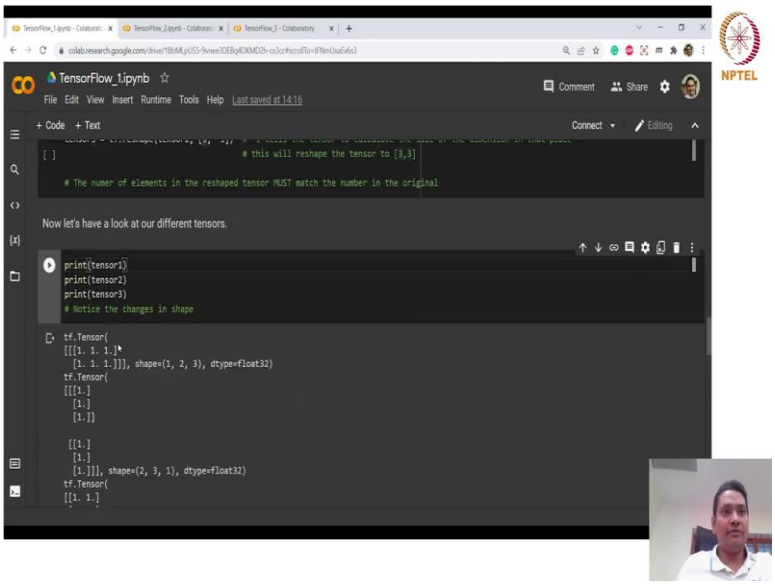
(Refer Slide Time: 04:01)



Now, change in the shape, we can use reshape method to actually change the shape. So, here we have this once created shape as 1, 2, 3 right. So, all the elements in this tensor is essentially ones. Now, I want to reshape this 1, 2, 3 to 2, 3, 1 ok. So, tensor reshape and tensor 3 is reshaping to 3, -1. So, -1 is telling you how many elements will be there automatically in this dimension.

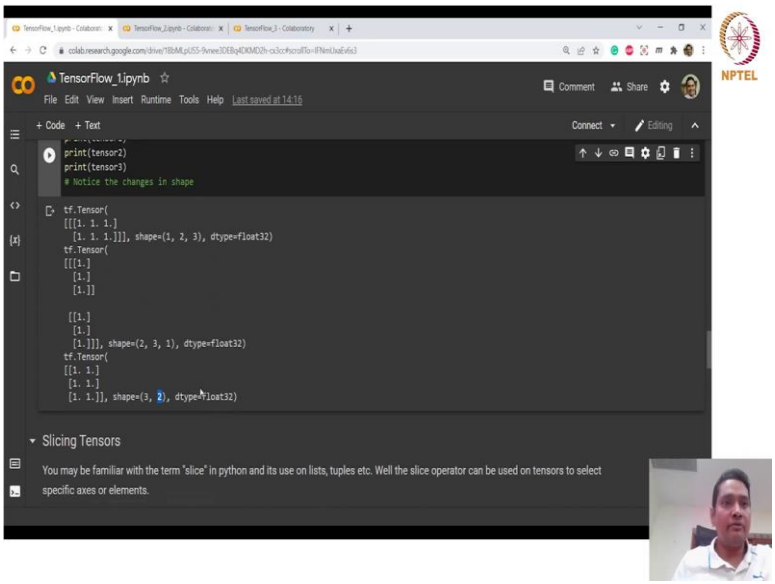
So, here in the first dimension we have 1, second dimension we have 2, third dimension we have 3, first 2, 3, 1 here we are saying first dimension we have three elements and in the second dimension we do not know infinite from the reshape ok. So, same -1 you have used before in pytorch.

(Refer Slide Time: 04:57)



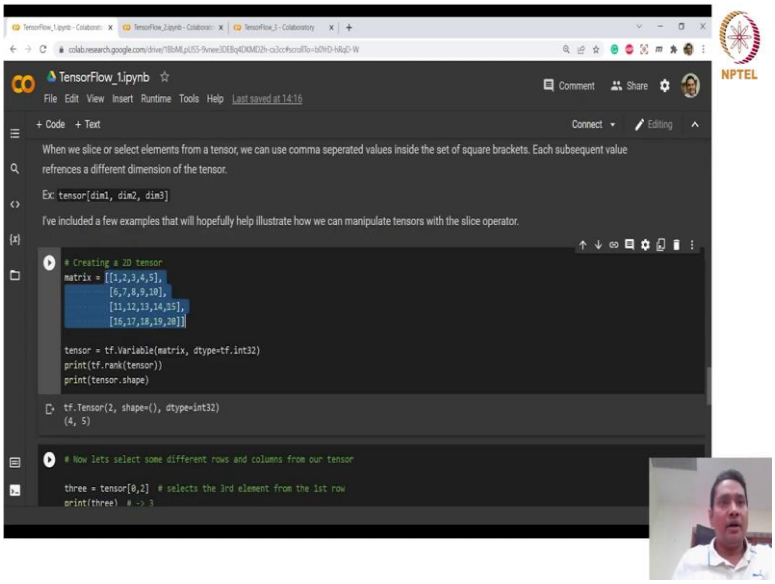
So, printing this the tensor 1, tensor 2, tensor 3 that is the different tensors that we have initialized. So, you we can see 1, 2, 3 shape right.

(Refer Slide Time: 05:03)



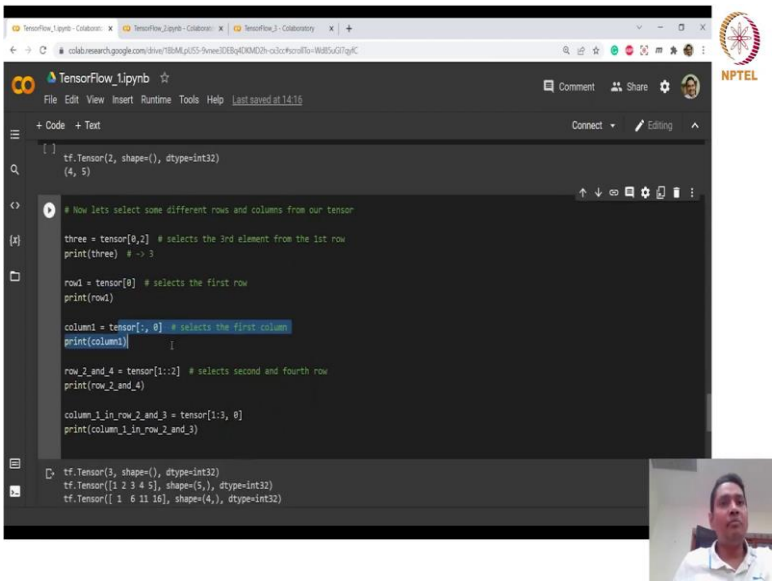
The ones one completely one element 1, 2, 3 tensor the second tensor was actually 2, 3, 1 right. So, 3 and 2. So, two total of elements and three elements in these two elements and total of sorry three elements and one actually element in each dimension right. So, one element in each dimension tensor. Now, in the third you can see that the shape already taken as 2, because we had -1. So, it will automatically compute first how many elements will be the second dimension. So, this is basically 3 rows and 2 columns matrix.

(Refer Slide Time: 05:58)



Slicing tensors: So, this is also important if you are using different data types in different shapes. So, basically here we have defined one 2D tensor.

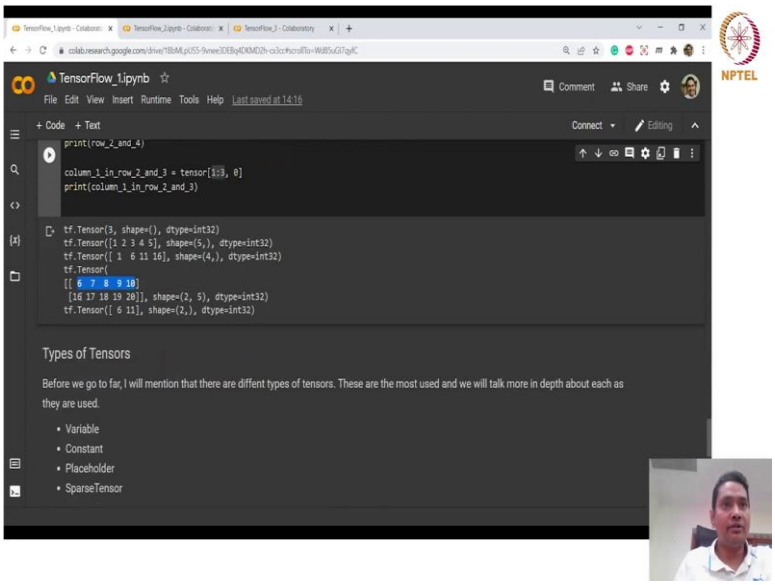
(Refer Slide Time: 06:09)



And then, we are actually selecting let us say third element from the first row selecting the first row completely selecting the first column selecting the second and fourth column and printing all the column values here. And then, we are selecting column in 1 in row 2 and 3. So, column 1 in row at 2 and 3, so basically column 1, from 2 and 3 ok.

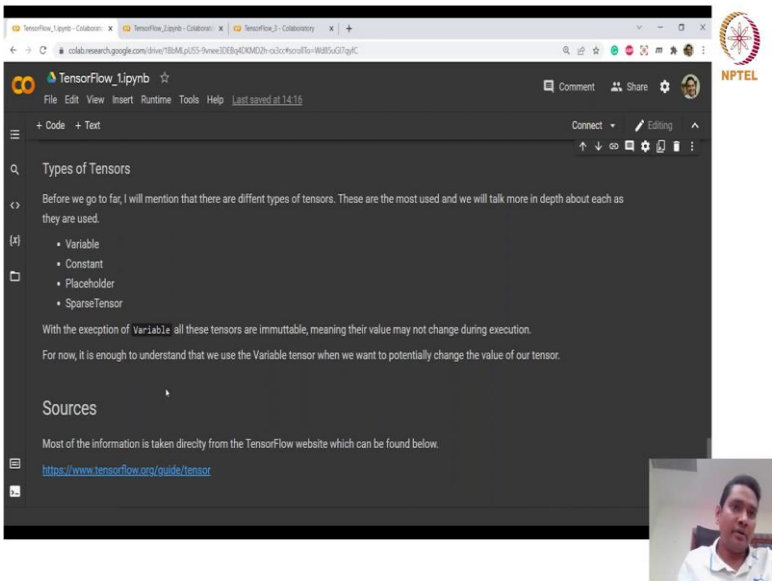
So, this is the column 1 value 0 and from 1 to 2. So, basically 0, 1, 2. So, basically 2 and 3 row you are accessing.

(Refer Slide Time: 06:47)



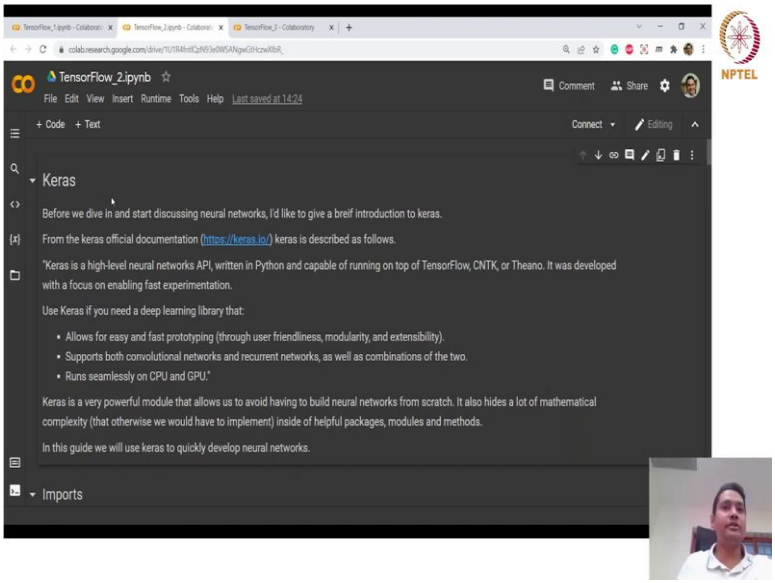
So, basically if you see the outputs, you will see the tensors whatever tensors you have got. So, first we are accessing the element from third element, which was 3 right. So, 3 and then you are accessing the first row then the first column, then the second and fourth row and then the first column in row 2 and row 3 right. So, this is the first column in row 2 and row 3.

(Refer Slide Time: 07:25)



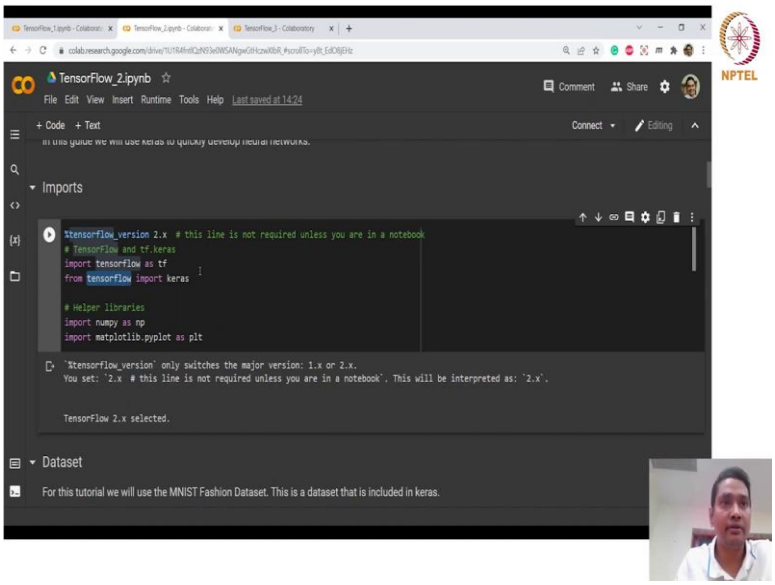
And you can see the shapes and different data types that are being used ok. Now different types of tensors are there variable constants sparse tensor and placeholder, placeholder is not recommended also in the 2.x versions. So, you can go to this link to see more of these usage of tensors. So, this is just few basics that I wanted to cover for the tensors because, these tensors we will use almost in all the layers that we want to define in the model ok.

(Refer Slide Time: 08:00)



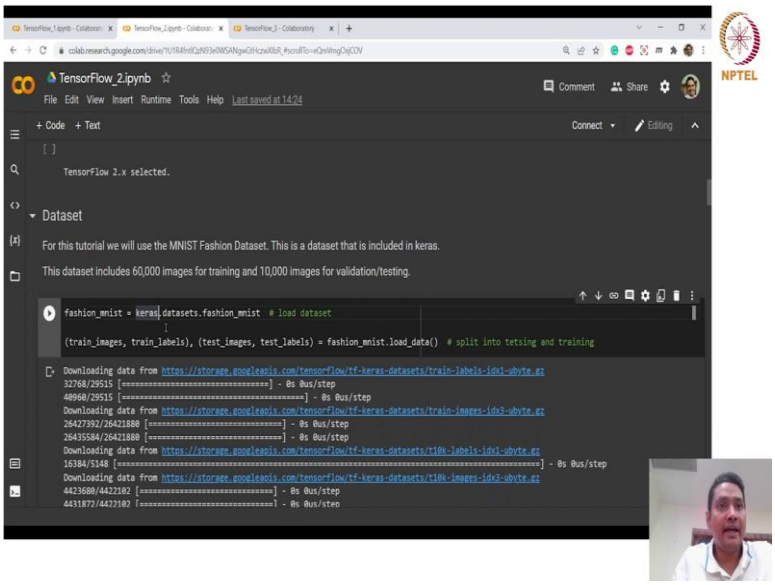
Now, coming to the model. So, what we have told that you can define your models by using sequential API, you can use functional API plus plus plus. So, basically customized callbacks your customized optimizers and so on and then subclass and in this tutorial, we will see the usage of Keras API with sequential paradigm right.

(Refer Slide Time: 08:29)



So, importing the necessary versions. So, now percentage tensorflow version 2.x this comment is necessary, if you have multiple versions available inside your system for tensorflow. And it will actually force to use the 2.x version level. We are just importing then the keras, the API functional the keras high level API from tensorflow, we are not using the keras IO library which is the native keras we are not using. We are using tensorflow keras. Helper libraries from numpy and matplotlib just to work with arrays and plotting.

(Refer Slide Time: 09:18)

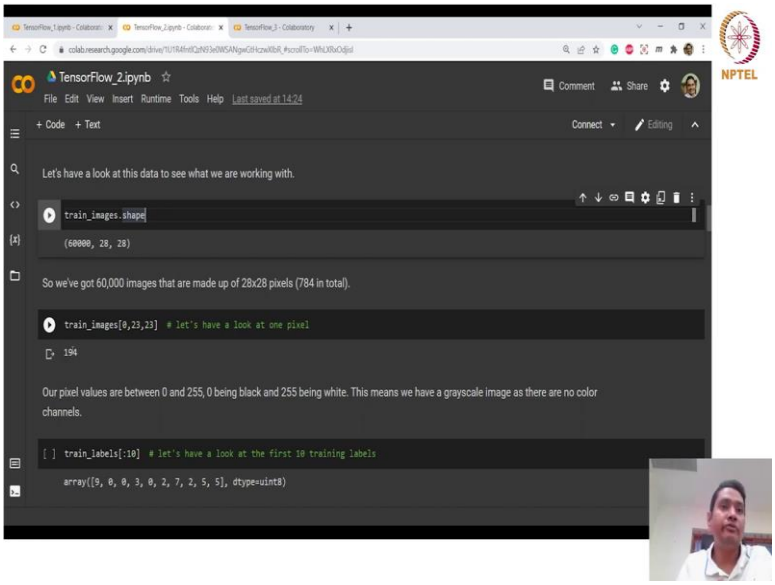


Now, in the dataset, we can see we are using keras dataset here ok. Not the tensorflow dataset. So, all these things you should keep in mind while you are programming or you

doing the model definition. Because most of the students or researchers I have seen that they just copy some code from somewhere else, but they do not know what kind of pipelines that are being used, because tensorflow has several options, several combinations, several compositions of different APIs.

So, if you are not aware of which API you are using you will not be able to accelerate your train. So, that is why here we are using keras just keeping in mind, ok; just because to be sure that we are just extracting unpacking the training images labels and test images labels from the fashion mnist dataset that dataset you have seen also previously.

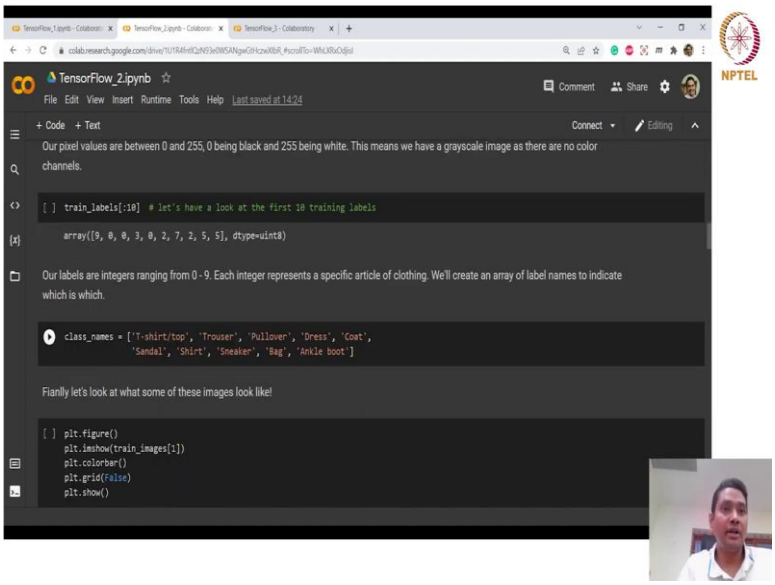
(Refer Slide Time: 10:18)



And we are using dot shape attribute to see or how many images that train images have right. So, after downloading, so basically we have downloaded everything. Now, you can see that 6000 that is the number of samples that is available 28 by 28. So, you can see the correspondence with the numpy ok. So, we have got 60000 images. So, train images just to see what is there in let us say 23 by 23 pixel of the first image ok; you can see 194.

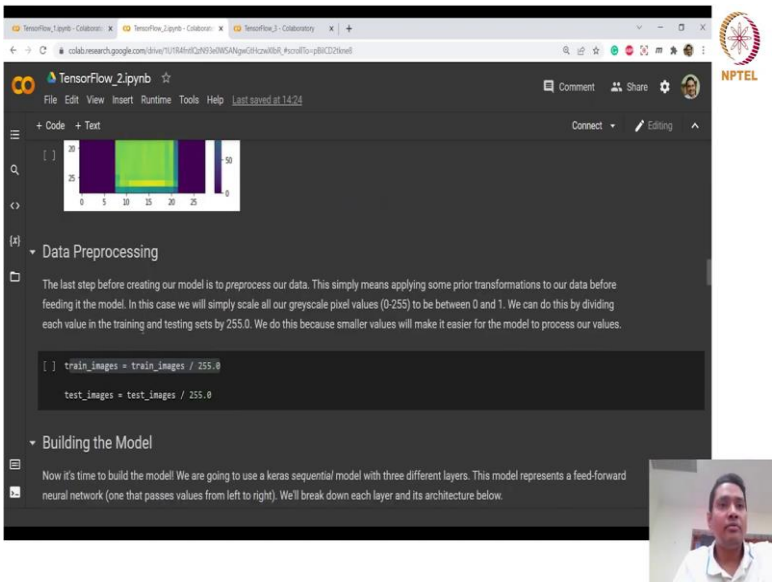
So, the values here actually varies from 0 to 255. So, basically what is to be done to be preprocessing, in the preprocessing part you just need to normalize it with 255.

(Refer Slide Time: 11:04)

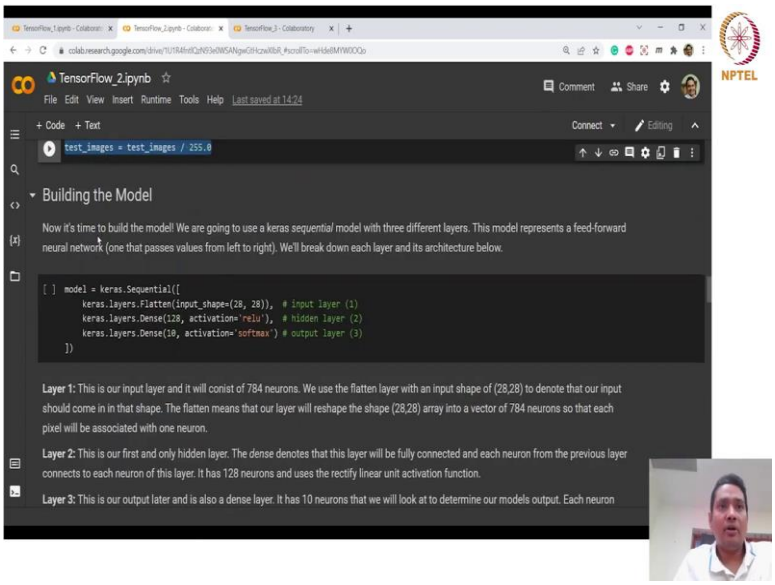


So, that is what we are doing here. The same thing you have seen in the data loader for the python pytorch for the training sets and test sets preprocessing of the data. Class names we are extracting and saying that these many classes will be there and these are the names of different classes, right. We have extracted the models.

(Refer Slide Time: 11:37)



(Refer Slide Time: 11:41)



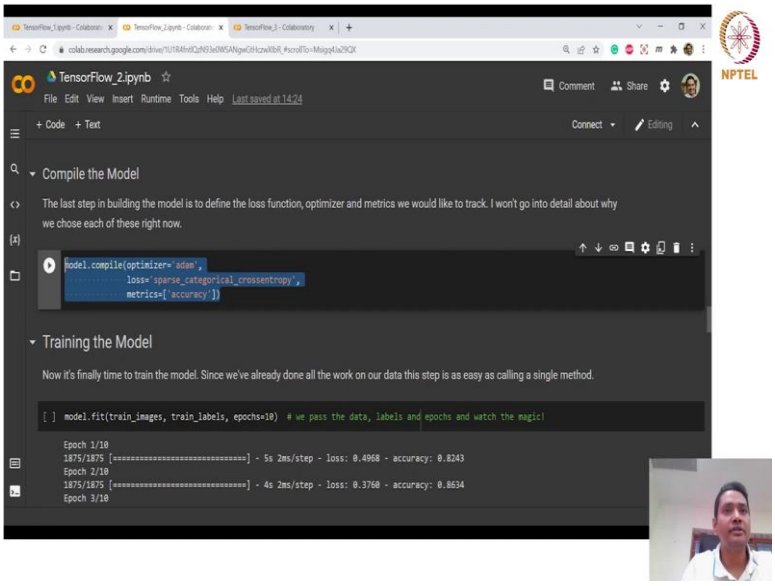
Now, data preprocessing part is essentially just normalizing ok. Now again, all these operations are happening inside your CPU, ok and we are using simple numpy arrays and numpy operations ok. So, simple map that is operated on the array ok. Building models: So, now, we will try to build the models, you can see that we are using keras dot sequential API. Sequential API for beginners and all the models model layers, what are defined for let us say you are targeting classification problem; you are targeting NLP problem; as if I mentioned.

So, there numerous numbers of layers that is already defined in keras. So, keras tensor flow you go to that website, for keras tensor flow and see how many layers that is already available in the sequential API. So, as layers, we are using flatten layer and dense layer. Flatten is essentially the helper function to flatten your input ok. So, this is exactly the input layer. So, in the pytorch also. So, every time you will just need to see the equivalence in the pytorch ok. So, you just visualize whatever we have seen so far and what we have seen in further classes.

So, in the dense layer, so this is the dense layer, basically this is the fully connected dense layer or linear layer that you have seen so far ok. So, again I am mentioning that you go to keras sequential API and see what are the layers have output tensor dimensions. So, 10 is the output number of neurons that I want from this dense layer because 10 classes are there and activation value again all the are predefined; activation softmax.

So, basically we are just defining this layers. So, dense layer so it will take whatever you are giving input from here and give output as 128 neurons with this activation. And then we will have the second dense layer, so two layer definition model that we are defining here. And you can see in the output we are having 10 classes here. So, this is just as simple of definition a model. And there is nothing else you need to do. This is very simple sequential API. Definition of layer 1, 2, 3 that you can see here, but what is most important is that model dot compile.

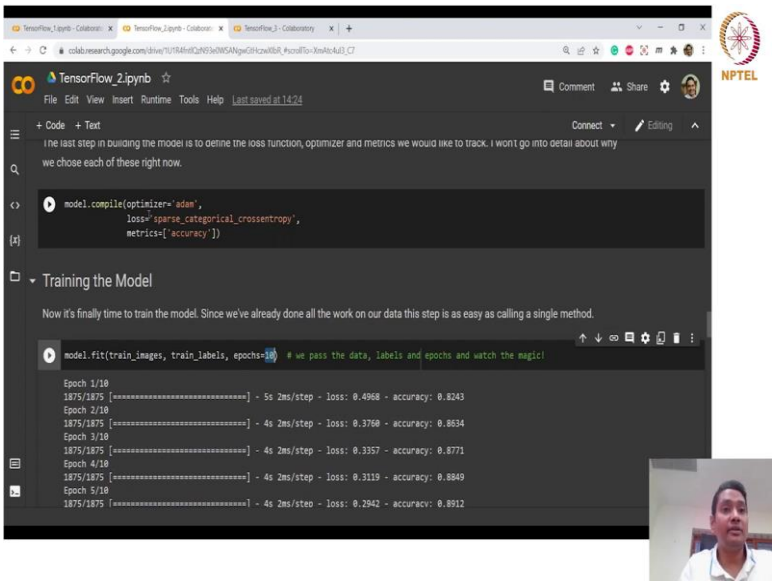
(Refer Slide Time: 14:16)



Now, this is the eager execution. So, when you are using this compile, then at this time itself you are actually compiling your model; without running the model, you are actually analyzing, what are the attributes for the layers that you have defined is that correct or not? All the parameters that you have used for defining your layers. So, I did not know. So, if you are working elsewhere, if you are not using compile, then you are not essentially taking the efficiency of eager execution.

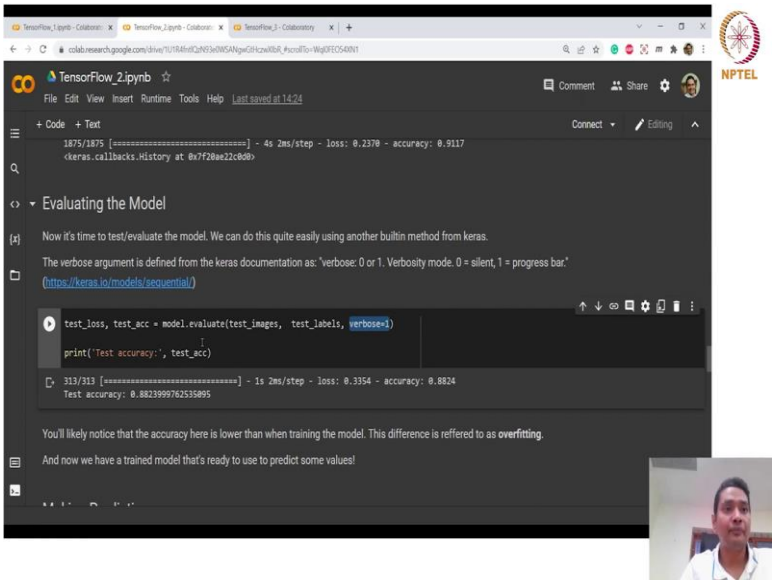
As long as you have defined it and run it, the graph is already defined and evaluated for compile. So, basically in the compile we have seen all the things that you have defined is correct or not and compile check compile time check you can do it.

(Refer Slide Time: 15:21)



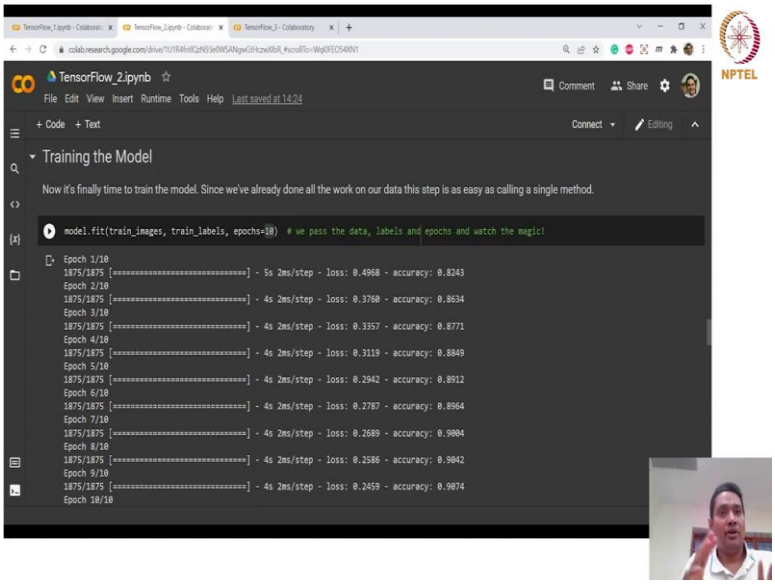
Then we if we just so everything is ok, there is no error you can fit it for the training images and training labels for these many number of epochs. So, you can see how easy it is to define one model training with so, defining layers with sequential API and training model with fit method that you have seen in the slides. So, on what training images train lables and epoch that is it. And it will train it, but before that you have already checked your model is correct or not right.

(Refer Slide Time: 16:01)



And evaluating the model is again it is easy with evaluate method ok. So, model.evaluate(). So, all the things we are doing here is sequential API and we are not using sub classing or functional API.

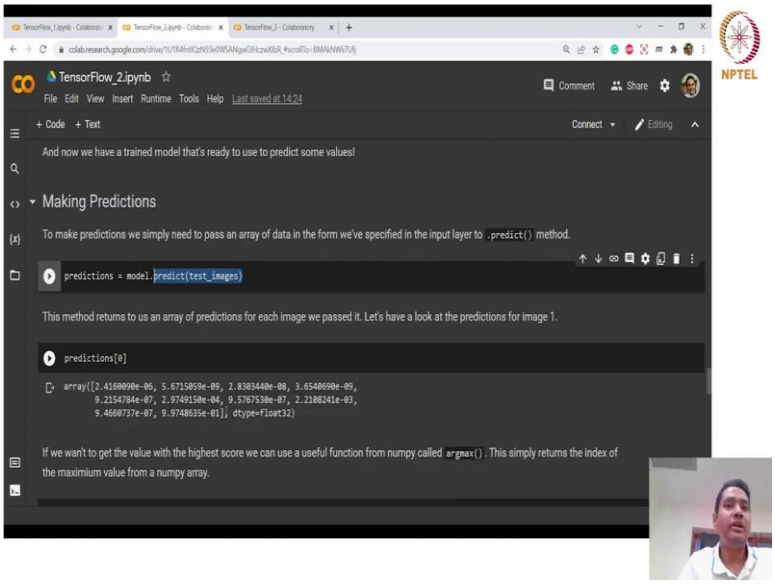
(Refer Slide Time: 16:14)



So, we will see sub classing and functional API with the different data distribution strategy that are available in tf dot data in the next class, but just get familiarized with the simple beginners APIs that are available with tf. So, we will go in advanced model definition in the coming classes. So, you should be aware of these simple models definition.

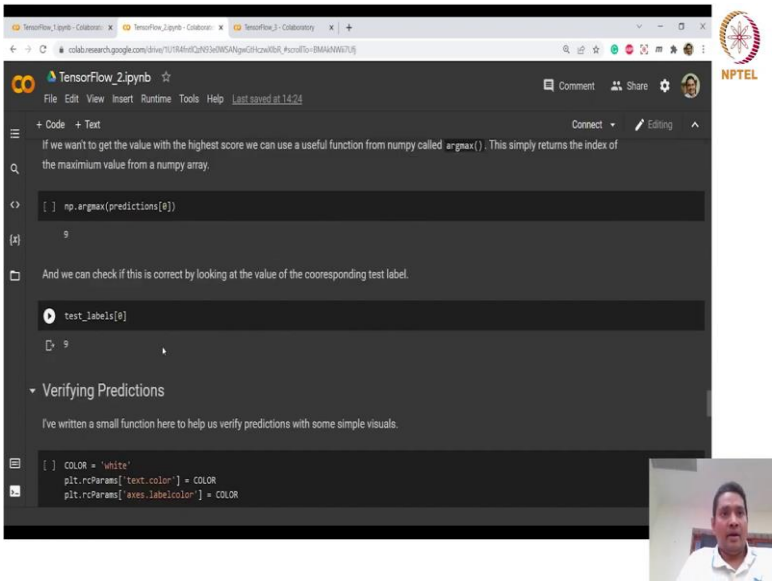
Now, printing the testing and accuracy. So, all this, so when you are actually calculating the evaluation you have getting the tuple for the loss and accuracy, I mean you can print it ok. So, this is all these are actually abstracted you do not need to define anything. So, basically if you want to verbose the output or not that also we can define.

(Refer Slide Time: 17:15)

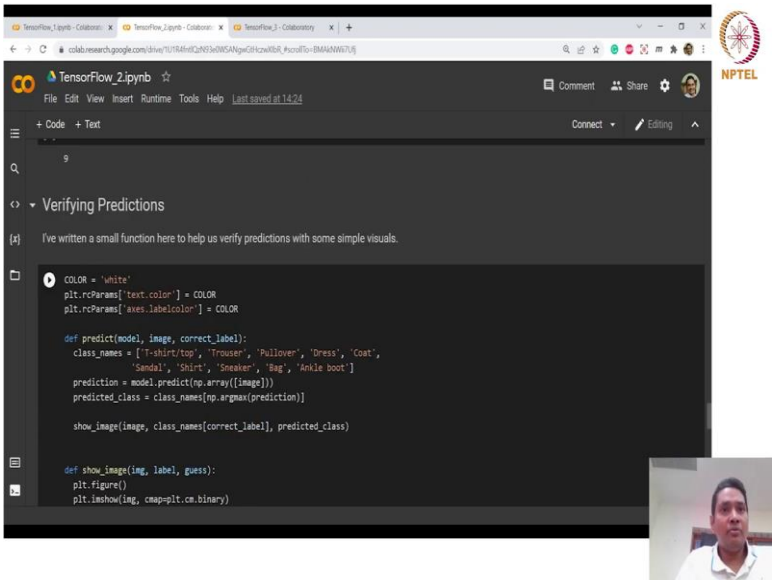


Now, we can we can actually test them. So, basically testing accuracy that we have evaluated, but to test you can call this predict method to get the predictions and you can use this predictions. So, basically which prediction index has the highest prediction probability that is it right.

(Refer Slide Time: 17:37)

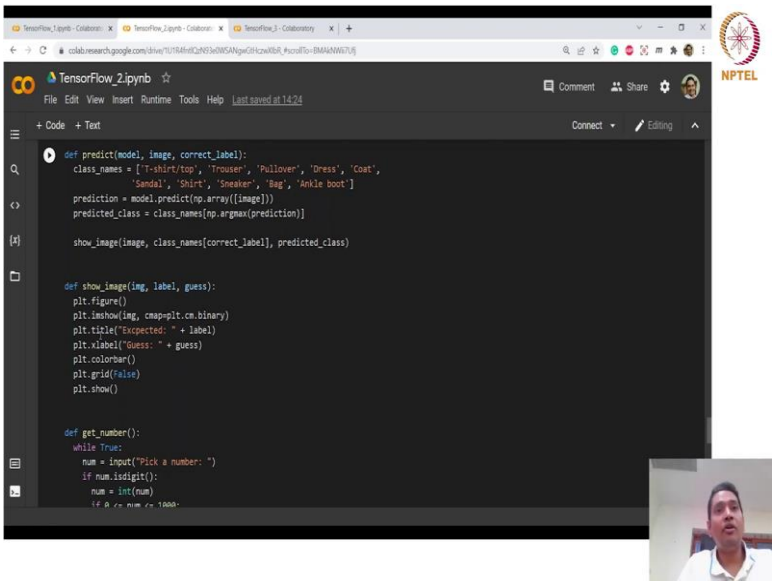


(Refer Slide Time: 17:43)

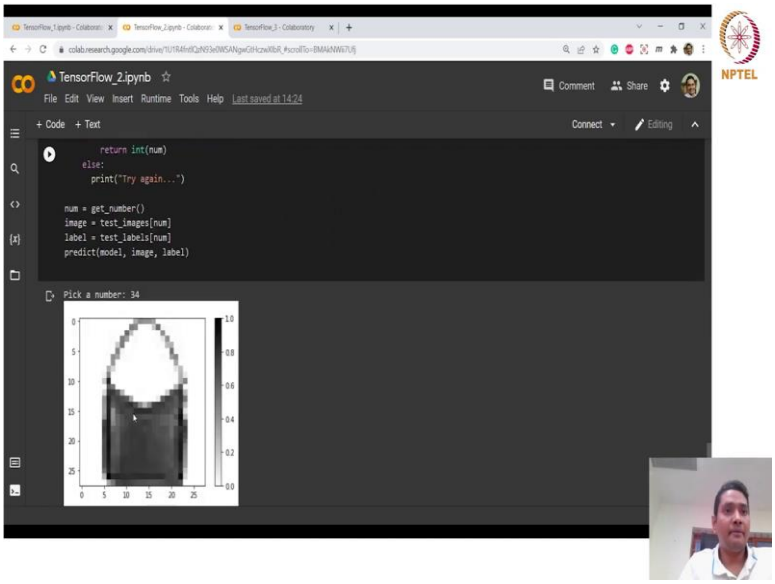


So, you can get it right; which is the argmax in right. So, now, we want to verify the predictions. So, basically whether your 9th level is actually correct or not.

(Refer Slide Time: 17:53)

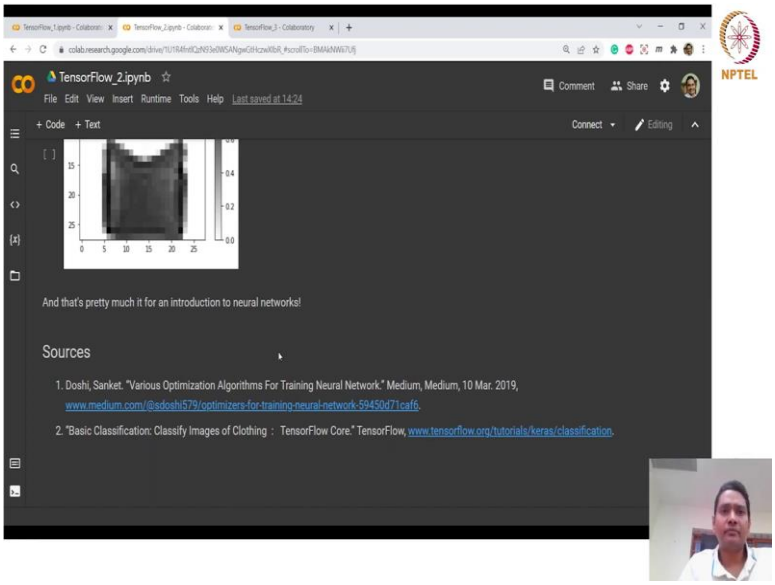


(Refer Slide Time: 17:57)



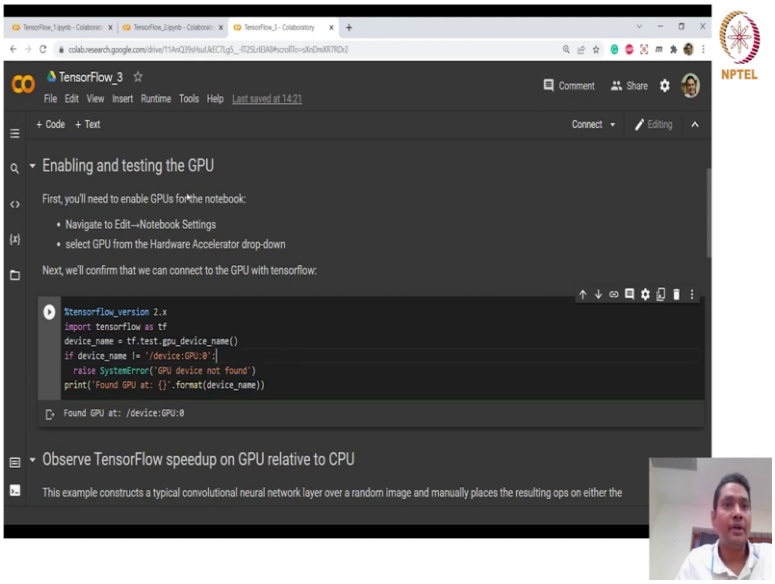
So, here is simple function, this is nothing to do with training or evaluating. So, this is just we are checking whether the 9th level is fine or not accessing the particular index ok indexed image ok.

(Refer Slide Time: 18:09)



And you can go to this links for detailed definitions of or different optimizations that you can use in the tensorflow and so on. You can also visit tensorflow sequential API to see what are the; what are the flexibilities you can have like how much you can go with the fit and predict and evaluate right.

(Refer Slide Time: 18:47)

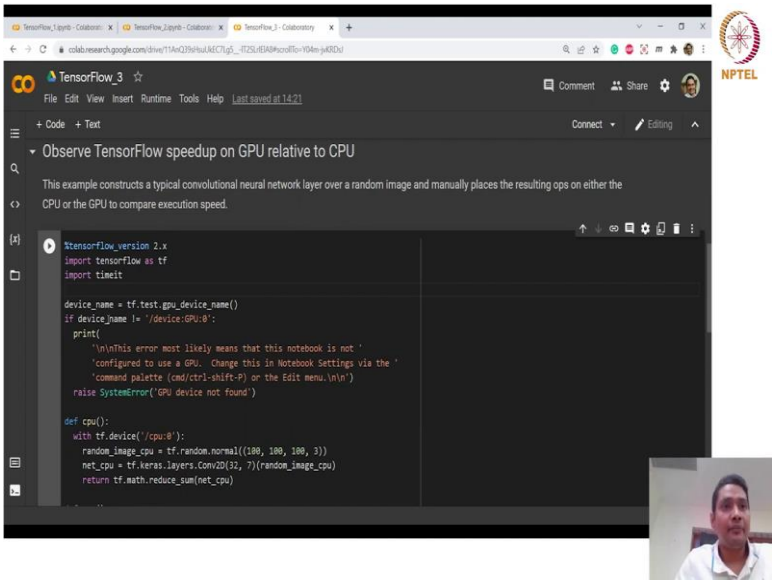


Now, we will see a bit about how we can actually compute in GPU. So, basically this is a bit different from defining the device for in pytorch and transforming the data into your GPU and model inside the GPU. Here are actually have much more flexibility in terms of accessing or offloading the tasks or pipeline of your training module in inside your GPU. So, that way the distributed training of tensorflow is much more efficient.

So, in this class, we will just see how to define the GPU and then how to define the program codes which will go to your GPU or TPU rather, depending on the device you want to access or CPU I do not know. So, here accessing to your importing your module and then device name is essentially tf.test.GPU device name. So, whether what kind of GPU is available GPU 0, if not found it will raise the error.

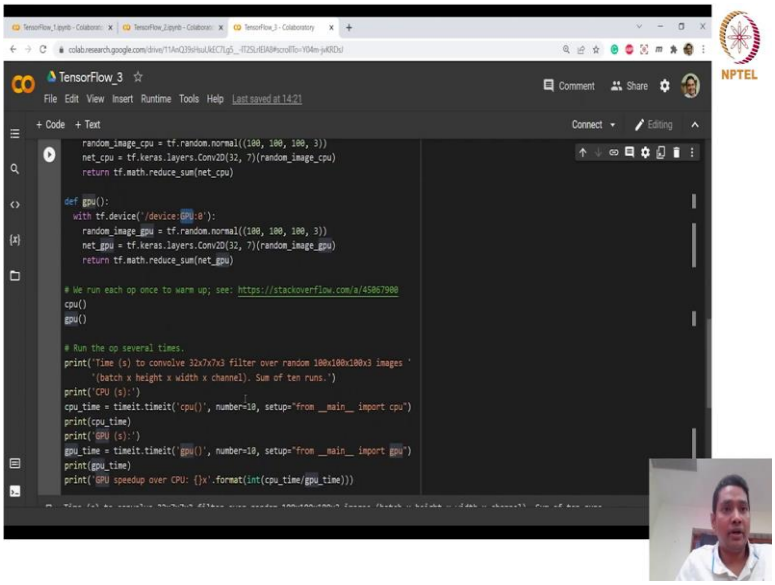
So, you just change the run time type to GPU if you are using colab right. And if you have already in your local machine if you have GPU then run could.

(Refer Slide Time: 20:00)



So, now in the tensorflow, how we will offload the task right? So, basically device name we have gathered if whether it is a GPU or it is a CPU or it is a so for the TPU we have not written the code here. So, we just see the CPU and GPU here TPU maybe we will see in the next class ok.

(Refer Slide Time: 20:16)



Now, for the CPU so basically def cpu this function I will run. So, basically with cpu.tf.device, if it is CPU then this segment of code I want to run. So, what segment of code I am running here, again you can see that layers.layers.Conv2d. So, this is again

defined in keras.layers and some input random image I am creating with 100 batches of 100 by 100 into 3 channels ok. So, this is the tensor dimension ok.

So, again the rank, the shape ok, so you just correlate to the idea of the tensors that that we defined concepts. So, basically this is giving the output of your convolution 2D of this random on this random image CPU ok. Again, this is what kind of layer that I will not tell you now, but this is just I am running one layer of one model just to see whether it is running in CPU and another layer we are defining here net GPU and that also we are running in inside our GPU ok.

So, we will call these two methods to run both in CPU and GPU the same dimensions of inputs we are running here and actually just to see what kind of achievement we are getting, we have call these two methods and printing the time taken by the two devices right. So, as you can see we are getting 67 times just to simulate one layer only yeah. So, but you can see right.

So, defining functions with devices like which device will run this piece of code yeah; that flexibility is something that we will exploit in distributed training yeah. So, basically we will distribute of course, there are several abstractions of the APIs you may not exactly define like these functions. But this is giving you one idea that yes we can customize our pipeline much more efficiently with what kind of device it will use.

Let us see how tensor processing unit available, GPU unit available, CPU available, multi core CPU in multiple nodes. Now this is where we will see that whether this piece of code you want to run in GPU or this piece of code you want to run in GPU right. And numerous examples will be there to explore this distribution strategy of course, as I was mentioning that is there are several abstractions of these abstracts of the strategies that you can use directly without defining that this piece of code will go to this piece of device.

But the strategies in underlying these are the techniques that will be used to actually distribute with different strategies that we will use ok. With this we will conclude today’s session.