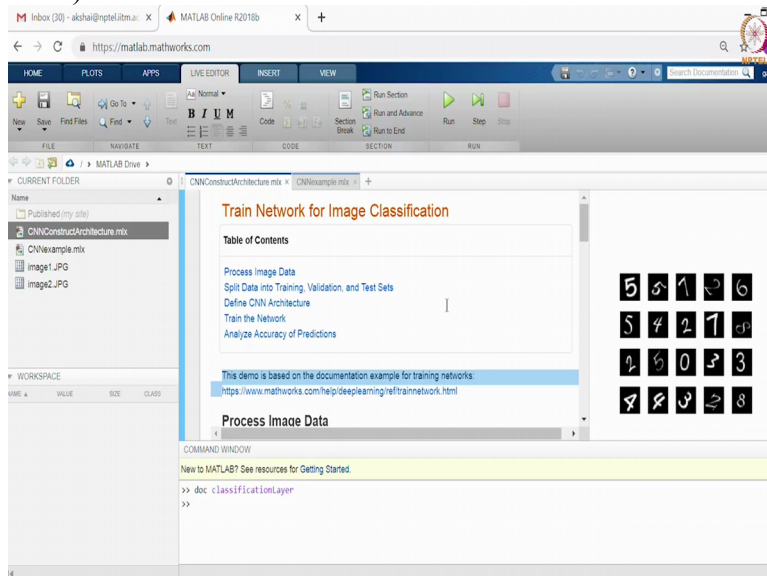


**Machine Learning for Engineering and Science Applications**  
**Professor Dr. Ganapathy Krishnamurthi**  
**Department of Engineering Design**  
**Indian Institute of Technology Madras**  
**Train Network for Image Classification**

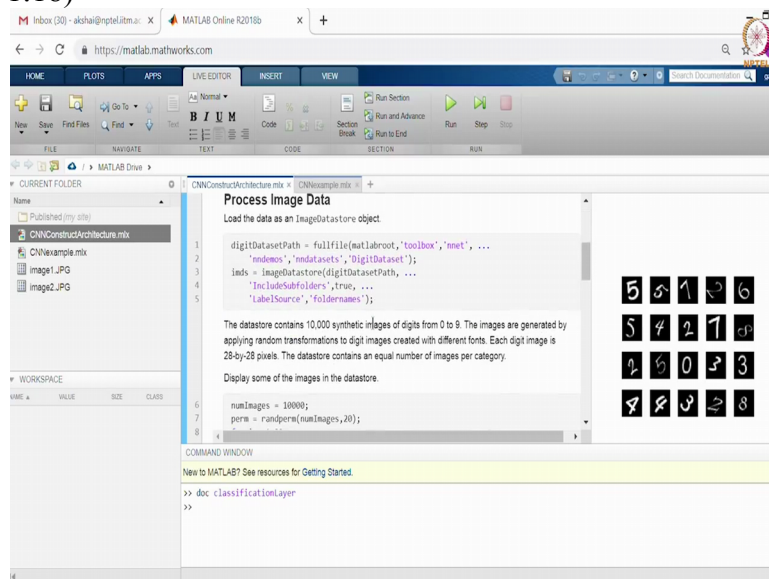
(Refer Slide Time: 0:18)



Hello and welcome back. So last week, we were looking at convolutional nodal networks, what are the operations in nodal networks and we also looked at some of the operations, the convolutions how they are done and some of the more common architectures so on and so forth. So in this week we will start off with how you will code such a thing like CNN in MATLAB right? So there this is based on this example script provided by MATLAB. I will just walk you through some of the piece of the code just to give you a head start and also show you in case you don't understand something, how to look it up and you know how to proceed.

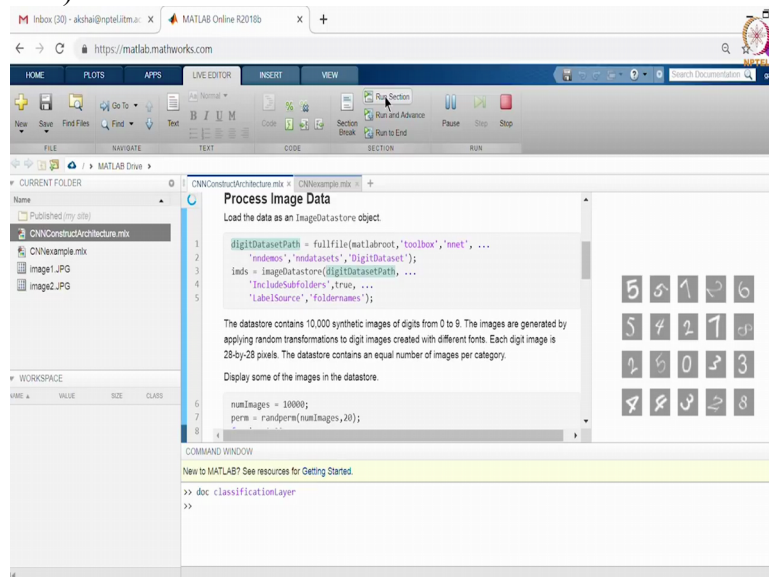
So it is fairly easy to walk through and create one for yourself. So this particular script that we are looking at, it tells you how to create a CNN architecture for Emnist digit classification okay.

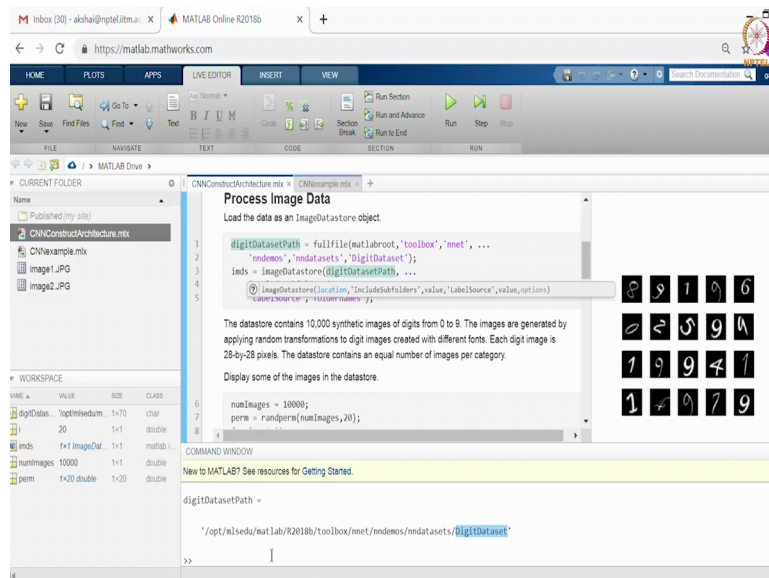
(Refer Slide Time 1:18)



So the script itself is available. I will upload the script if it is not. The link to the script or just the script itself so that you can look at it. So we will start off. So this is again a CNN for Emnist digit classification. So let us look at the data first.

(Refer Slide Time 1:37)





So we have to load the data. So MATLAB gives you something called the image Datastore object. Okay. You can create one, the object itself. So what it has basically is that it does not actually contain the data directly. So let us say you have like 10,000 images, so you can and you do not have the hardware to load all of them. So it is not practical to load 10,000 images, you may run out of memory very quickly if you do not have that kind of hardware. Support this image Datastore does is that it stores the links to the images.

So in the sense like it tells you where to find the images. It shows the image directory, the filenames of the images that we are using or the data files that you are using or some meta-information about the images. So in this case, size of the image, number of channels in the image, so on and so forth. So you can create a Datastore object. So this piece of the code here that I have highlighted right now, it tells you how to create the Datastore object. So the first line of the code digit dataset path and if you look at it, it is full file and we have all this MATLAB root and all these folder names basically.

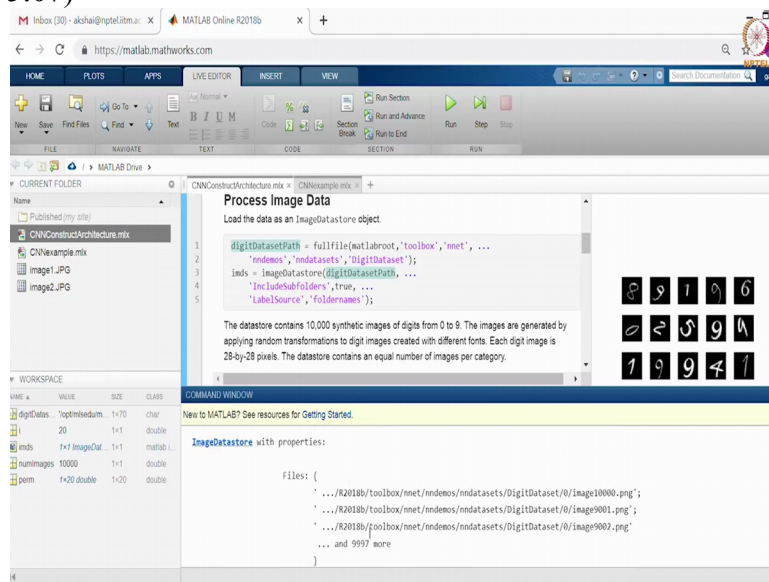
It creates a path, the pathname. That is all it does. So if you want to see what it is, so let me just, you can run section by section. So I will just run this one section. Right? So it is running on the cloud, I guess it is kind of slow. Right, so it is done. So if you look at the digit dataset path, okay I will just, if you want to know what task, you just type it down here. So that is it. It has basically the pathname to the directory containing the digits data, okay. So you can actually put it on your

file browser and see the data usage you want to. So that is why, that is what this command does. Okay. Full file, it creates the pathname. Right?

And the image Datastore function, okay, it creates this variable `imds` which has all the meta-information about the data, right? So what does it do? So you give it the path to that directory which has the data and you ask it to include subfolders, okay? So basically your organisation would be, you will have a directory where you have folder directly if you are using Linux where you have all the Datastores. Okay. So in this case let us say you have a directory called `digits` and inside you have subdirectories which are labelled 0, 1, 2, 3, 4 up to 9, right? So there are subdirectories. 0 which has all the images of digits 0.

Subdirectory 1 which has all the images of digits 1 so on and so forth. So it would ask you to look at subdirectories also, so include subfolders too. And label source is folder names. Okay. So imagine data store can also store the labels. So what does that mean? Label source folder names. So you have to name your folders so that it is actually a label for that image. So you know that this is supervised learning. So for every  $X$ , you need a label  $Y$ . So and where do we get that label information from? You can name your directories, subdirectories so that or the subfolders so that the name is reflective of the class of images stored in that subdirectory. So it take out the, it takes up the label data from the folder names themselves, okay. So every subfolder has to be named intelligently for you to do that. Okay? Then you can just, these are just objects that you can look at.

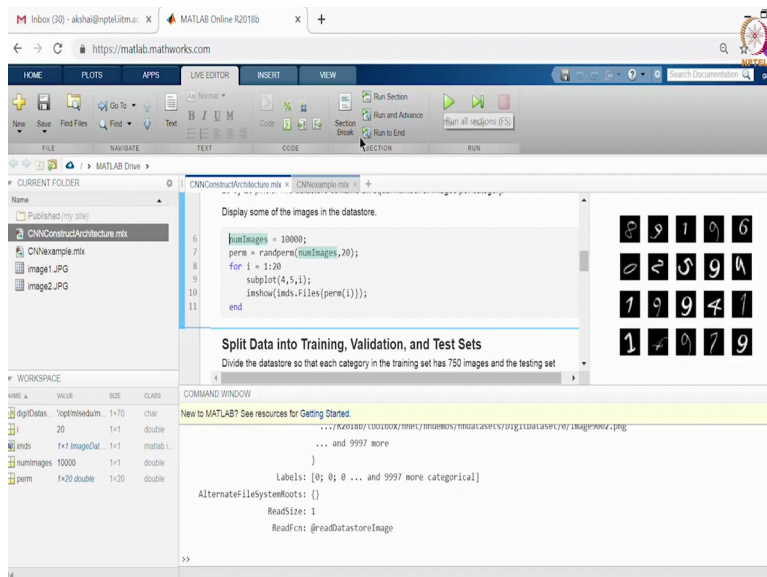
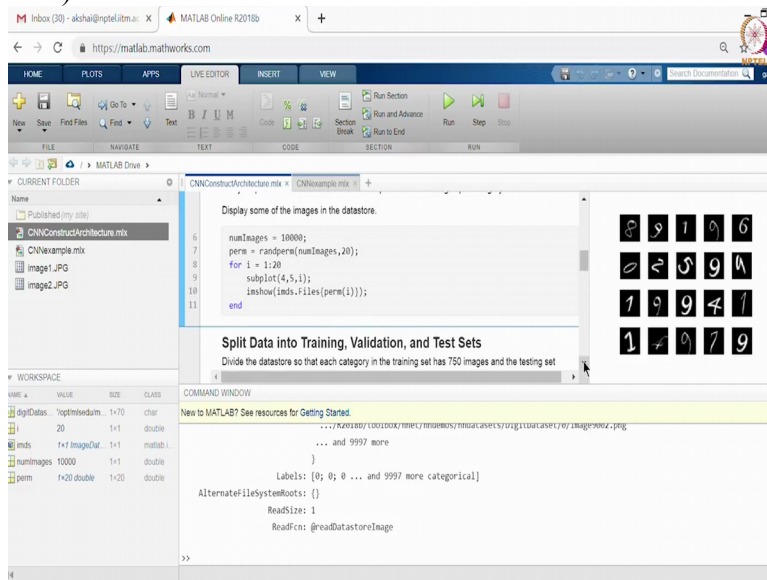
(Refer Slide Time 5:07)



So for instance, you can do `imds`. Okay. So it tells you what it has. Okay. There are all the labels for each one of them. Okay. And it has the filenames. So you see that the image Datastore has the following property. It has all the filenames. So it has the entire path, so look at this, there is a folder 0 and inside folder 0, are all these images. Okay? Each one of them is a image of the digit 0. And there will be a folder 1 so on and so forth. Similarly, it has a label for each one of those images. Okay. So there are about you know 10,000, there are 10,000 images, just 10,000 labels. Okay?

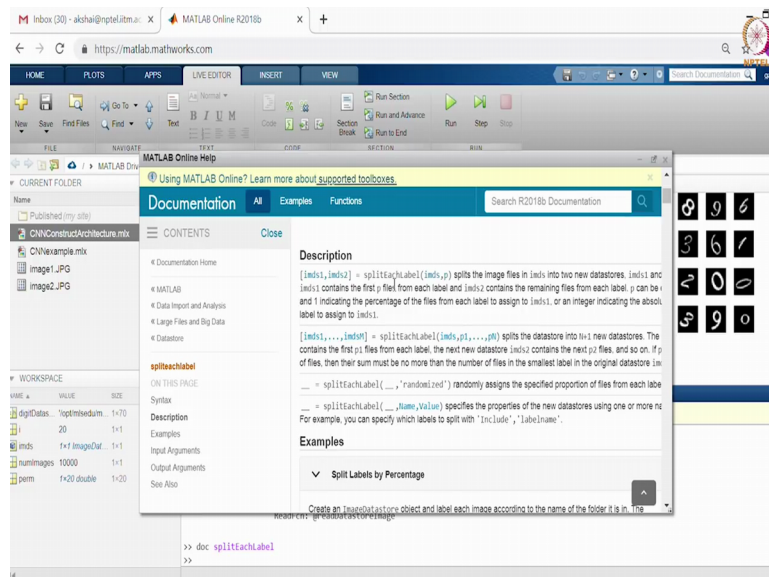
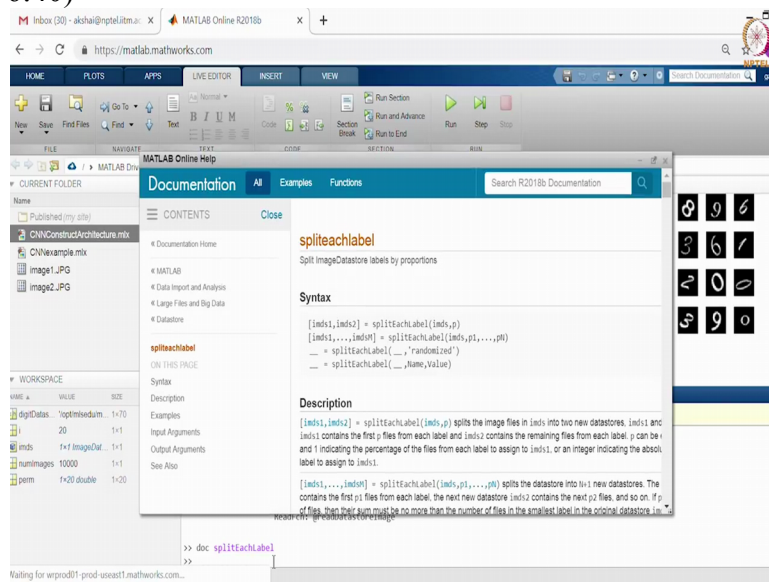
So 997 plus 9997 plus 3. So this is what these 2 instructions are completing, the commands are completing in the MATLAB environment.

(Refer Slide Time 5:59)



So you need the deploying toolbox for this. So you just want to look at the data, right? So you just want to make sure that you have the data. So the number of images is 10,000 and then you have a random permutation of 20 of them, okay? Choose a random permutation of 20 of them and for each one of them, you show you display the images. So that is a very simple enough, so make sure you run section. See how long it takes. So it just displays a different set of 20 images in a panel. So your subplot just creates 4 rows and 5 columns of image display panels, okay. These are small images because each of them are of size 28 by 28. So it like comes up very nicely in that panel. Okay.

(Refer Slide Time 6:40)



So the next task we have to do like we usually do is to split these data into training like here, validation and testing. Okay. So there is a function which does that, a method which does that. It splits each label and you have to give it as input the image Datasource. Okay. Because the Datasource itself has the label information in it. So for every image that you have that is a reference in a much Datasource, there is a corresponding label, it knows that. So we will say and we split it in proportion, right? So 0.7, 0.2, 0.1, you see if they add up to 1, so 70% of the data would be training, 20% of the data would be validation, 10% of the data would be testing, okay?

And randomise means it will just select 70% of the data at random. So there are 10,000 images, it will select 70,000 at random. If we leave this out, it will just choose the first 7000 images and the next 2000 images and the next 1000 images. So randomise is better because if you want to train, you would like to have a random sample of your data, okay. So if you want to know what these functions do, the easiest way to look them up so if you are going through this code and some of it appears opaque and you do not know what is going on, easiest way is to just do doc and you do split.

Sometimes if you use tab, it works but it does not in this case. So doc, if we do that, then MATLAB helps take you that particular function or method that you are using whatever. So what gives you different versions of it. Right? There are several ways you can use this. It is all listed here. I will not go through them but the one version we saw is this one which has randomised in it. okay. So you just you can use this to split your training. So this is important because whenever you are doing supervised training, you do need to have a training validation and testing split and this lets you do that.

And image Datastore is a very convenient way of doing it because you are actually not manipulating the images directly. Like you do not have them on memory, you just have the link to the images in the form of filenames, meta data about the images, their label information, so on and so forth and this lets you split, okay. So this is, this is how you see if you do not understand any of the functions in there, you can actually do just doc that function. In fact, but if you want to use doc, you have to know the exact function name or the method name or whatever that is that you are using or the construct, you have to know that exactly.

Otherwise you just try help, sometimes that works too. But in case you are going through a piece of code which they have provided, this is very easy, just to doc whatever and it will pop up the corresponding method, okay.



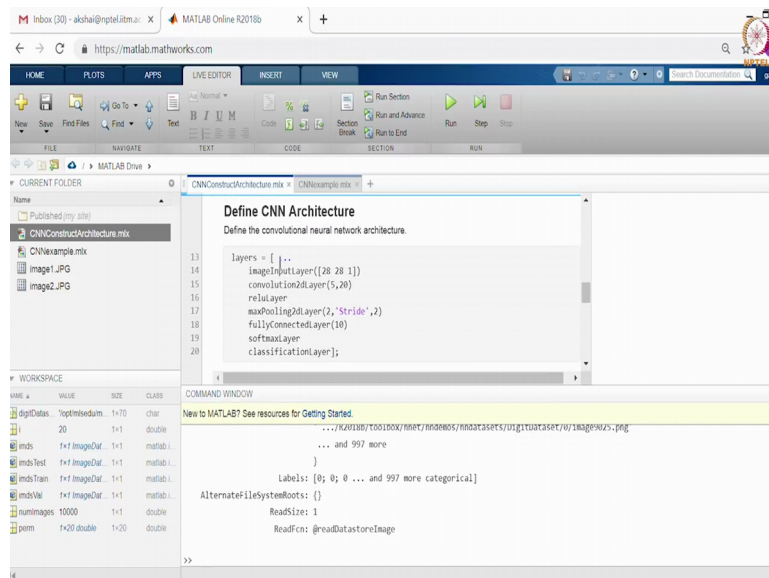
(Refer Slide Time 9:31)

The screenshot shows the MATLAB Online R2018b interface. The current folder contains 'CNNConstructArchitecture.mlx', 'CNNExample.mlx', 'image1.JPG', and 'image2.JPG'. The workspace lists variables: 'digitData' (1x70 char), 'inds' (20 1x1 double), 'indsTest' (1x1 ImageData 1x1 matlab.I), 'indsTrain' (1x1 ImageData 1x1 matlab.I), 'indsVal' (1x1 ImageData 1x1 matlab.I), 'numImages' (10000 1x1 double), and 'perm' (1x20 double). The command window shows the following code and error:

```
11 end
12 [indsTrain, indsVal, indsTest] = splitEachLabel(inds,7,2,1,'randomize');
splitEachLabel splits the image files in digitData into three new datastores, indsTrain,
indsVal, and indsTest.
>> indsTrain
Undefined function or variable 'indsTrain'.
>>
```

The screenshot shows the MATLAB Online R2018b interface after the code has been executed. The command window displays the following output:

```
New to MATLAB? See resources for Getting Started.
AlternateFilesystemRoots: {}
ReadSize: 1
ReadFcn: @readDatastoreImage
>> doc splitEachLabel
>> indsTrain
.../R2018b/Toolbox/nnnet/mnist/mnistDataSets/127123434347/0/1849/915.png
... and 997 more
Labels: 0; 0; 0 ... and 997 more categorical
AlternateFilesystemRoots: {}
ReadSize: 1
ReadFcn: @readDatastoreImage
>>
```



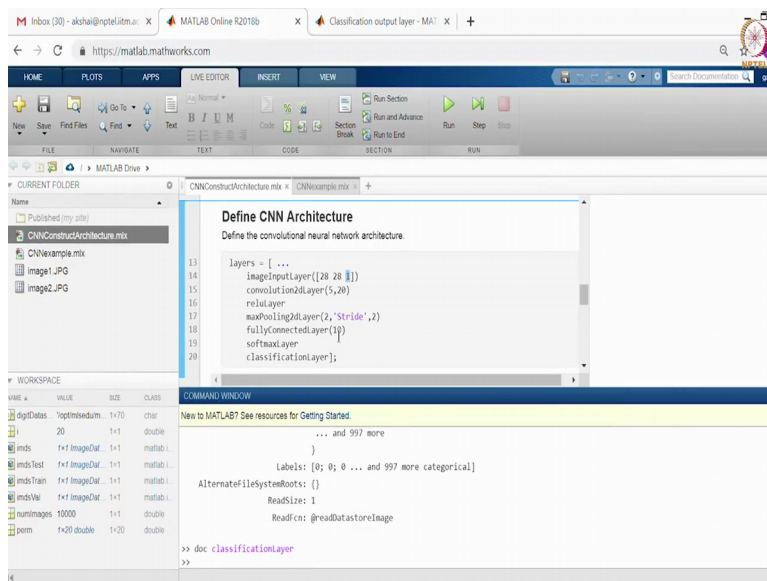
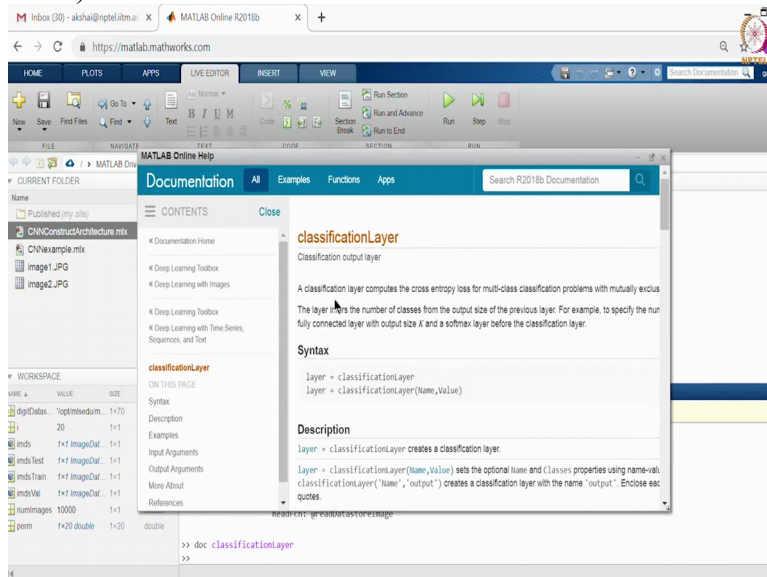
So that is what split each label does. So it splits the data into, so if you want, if you are curious as to what it has, so you can just come here and type this at the prompt `imdstrain`, sorry I have to run this before I do that. So I will just run this. So run section. So it is done. Okay. So it will tell you, so 6997 more plus 3. So that is like 7000 data points in the training. So if you look at the testing data, that is because that we have set 70% here. You can change this around and this convinces us that it works that way. And so for instance, test lets you do `imdstest`.

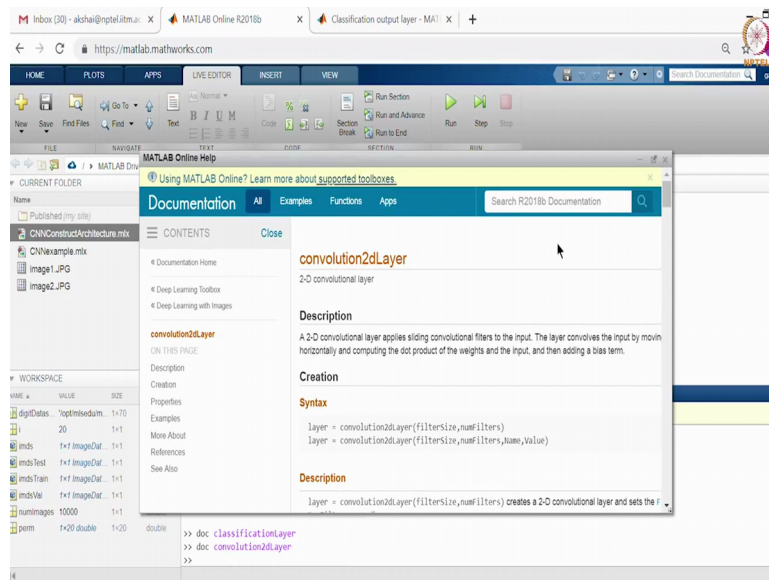
So it lets you see, there are 10% of 10,000 images, you have a 1000 images for your testing. Okay. So we have 3 new Datastores, `imds`, `train`, `validation` and `testing`. So then the next step is to define the CNN architecture. Again this is the important step. This is very nice because it actually does most of the layers that you so every layer is defined by the function and it has its own arguments or method and its own arguments. Okay. Then you just know what they are. So the image input layer is 28 by 28, right?

That is the size of the Emnist data and it has one channel, right? You can think of it that way. So image input layer. 2d convolutional layer, the kernel size of 5 by 5 20 sets filters, you have `reluLayer`, you know what `relu` is. And the max pooling layer, 2 by 2 max pooling with a side of 2. You have a fully connected layer, 10 neurons in a fully connected layer. And this feeds into a soft max layer followed by a classification layer. So what does the soft max layer? You know what it converts these raw numbers into properties, you can think of it that way.

And the classification layer actually calculates the cross entropy, binary cross entropy cross function. So if you do not know what these things do, that is again like I said, so for instance dark, let us do the classification layer, right? Doc classification layer.

(Refer Slide Time 11:51)



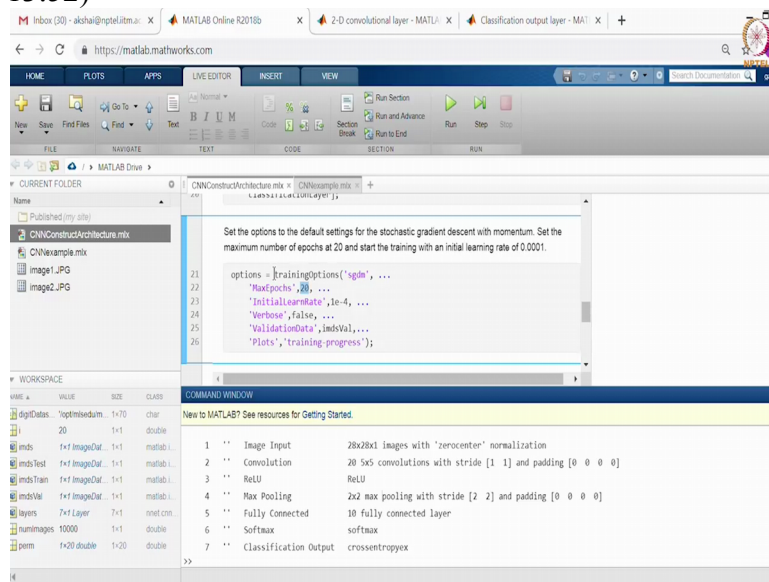


So if you do that it will show you what exactly it does. So it is a layer that computes the cross entropy loss for multiclass classification. So we are trying to do digit classification, 0 to 9, so there are 10 classes. And with mutually and if you, I will just, I am trying to make this full-screen because then I have to, it will come up, ya, for mutually exclusive classes, right. So then it actually infers the output size of the previous layer. So we saw that we have a, so if we go back here to the window, we have a fully connected layer of 10 which is then converted into a soft max.

So it knows that there are 10 soft max layers corresponding to the 10 classes and it calculates the binary cross entropy cross function. That is what this layer does. So if you want to do so let us say doc convolution2d layer, so this again tells you what that particular function does and what the inputs can be. So if you see that filter size, number of filters, right? So you can come back here, you see the filter size is 5, number of filters is 20. Right? So you can do the same thing for relu layer, max pooling 2d layer, so on and so forth. So if you run this particular section of the code, you can do run section, it will run it. So it is done.

So then you can look at the layers that you have created. So again it will just list you basically this lists the architecture, you have seen the architecture, right? Even input layer, convolution for the relu, so the relu is actually treated as a layer. But typically you treat them as 1 but in most of the programming environment, this is treated as separate layers and we have max pooling, fully connected layer, soft max, classification output, right? So this outputs the (())(13:46).

(Refer Slide Time 13:52)



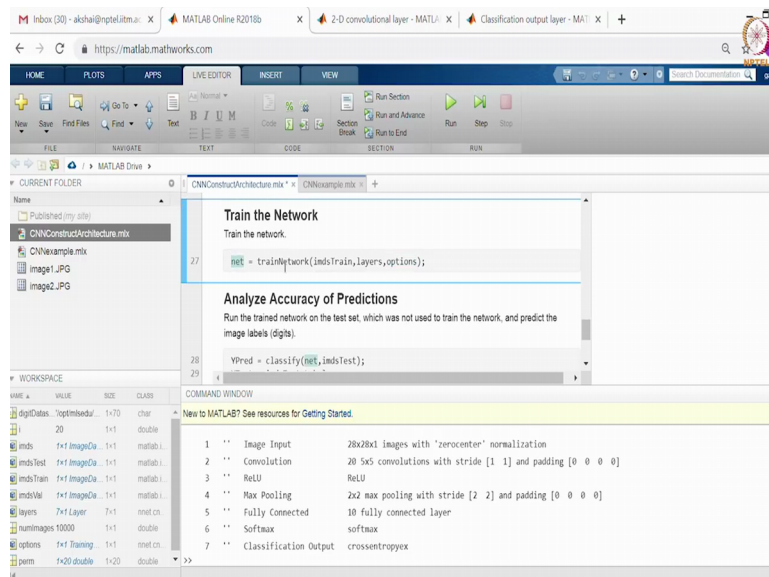
The screenshot shows the MATLAB Live Editor interface. The current folder contains files: CNNConstructArchitecture.mlx, CNNExample.mlx, image1.JPG, and image2.JPG. The workspace lists variables: digitDates, options, i, imds, imdsTest, imdsTrain, imdsVal, layers, numImages, perm. The code editor shows the following code:

```
Set the options to the default settings for the stochastic gradient descent with momentum. Set the maximum number of epochs at 20 and start the training with an initial learning rate of 0.0001.
```

```
21 options = trainingOptions('sgdm', ...  
22 'MaxEpochs',20, ...  
23 'InitialLearnRate',1e-4, ...  
24 'Verbose',false, ...  
25 'ValidationData',imdsVal,...  
26 'Plots','training-progress');
```

The command window shows a table of layers:

1	** Image Input	28x28x1 images with 'zerocenter' normalization	
2	** Convolution	20 5x5 convolutions with stride [1 1] and padding [0 0 0 0]	
3	** ReLU		
4	** Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]	
5	** Fully Connected	10 fully connected layer	
6	** Softmax	softmax	
7	** Classification Output	crossentropyex	



The screenshot shows the MATLAB Live Editor interface. The current folder contains files: CNNConstructArchitecture.mlx, CNNExample.mlx, image1.JPG, and image2.JPG. The workspace lists variables: digitDates, options, i, imds, imdsTest, imdsTrain, imdsVal, layers, numImages, perm, net, and Ypred. The code editor shows the following code:

```
Train the Network  
Train the network.
```

```
27 net = trainNetwork(imdsTrain, layers, options);
```

```
Analyze Accuracy of Predictions  
Run the trained network on the test set, which was not used to train the network, and predict the image labels (digits).
```

```
28 Ypred = classify(net, imdsTest);  
29
```

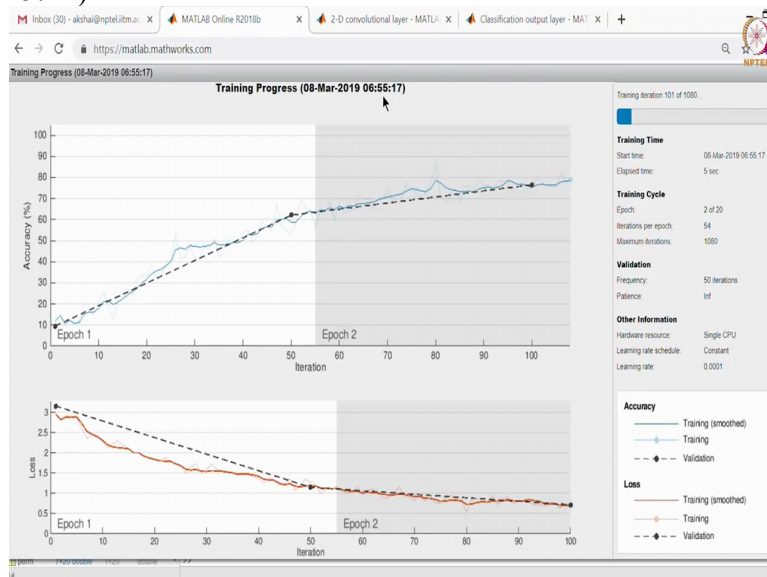
The command window shows the same table of layers as in the previous screenshot.

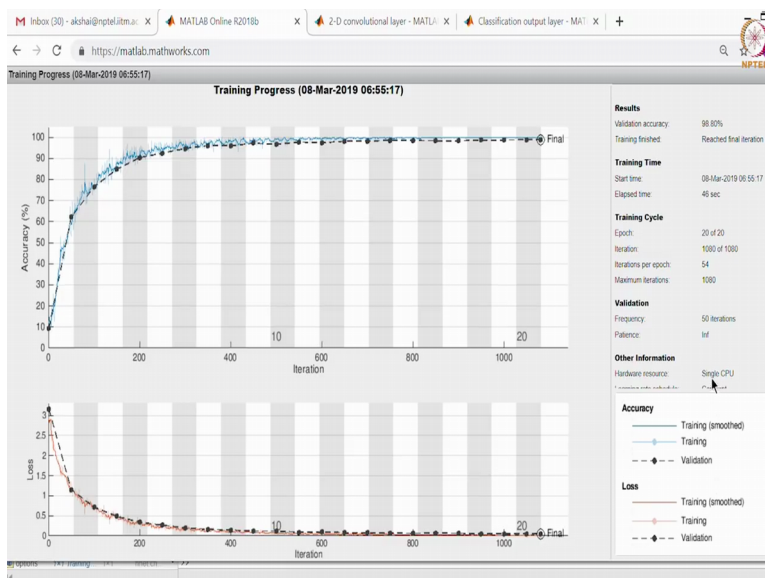
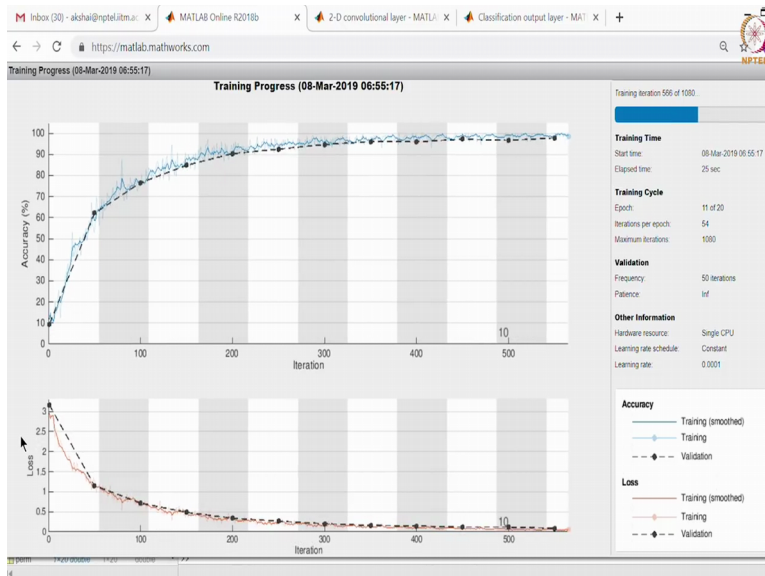
So now if we go down, so we have to set the parameters for training that network. So we have that network. So we have to train the network. Again this is passed on like in the form of again another variable called options. Okay. And you can call this function, training options, do it. So we are using stochastic variant descent, the maximum number of epochs, we want to run 20 epochs through, in fact 1 epoch is when you have used when you have done backprop using all your data. You have initial learning rate. Verbose is false meaning that it will not have too many outputs.

This validation data give it the data store, the data is validation data. Plots is the training progress plots will show up, okay. We will run it and we will see what this comes up like. Again if you want to explore about the different function, again do doc training options, it will show whwhat can be done. Okay. I will run it. I did not run this one. So this one is run section. So that works okay. So here and now we are all set now. We want to train the network so there is a function, train network.

Takes the training data as input, right? The layers, this is architecture and the options for your optimisation of data. So that is what it does. So we will train it, let us see how fast this runs. I am not sure. This is not exactly a fast computer, runs on the cloud. So we will see what pops up.

(Refer Slide Time 15:14)



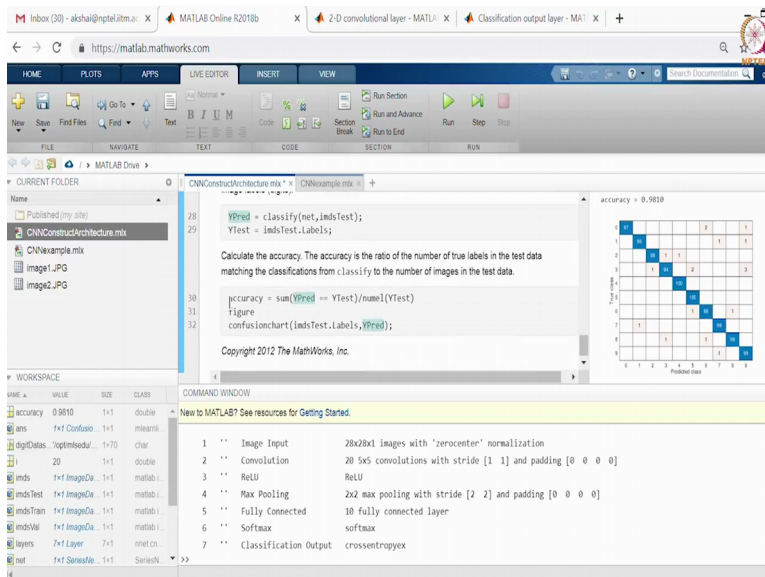
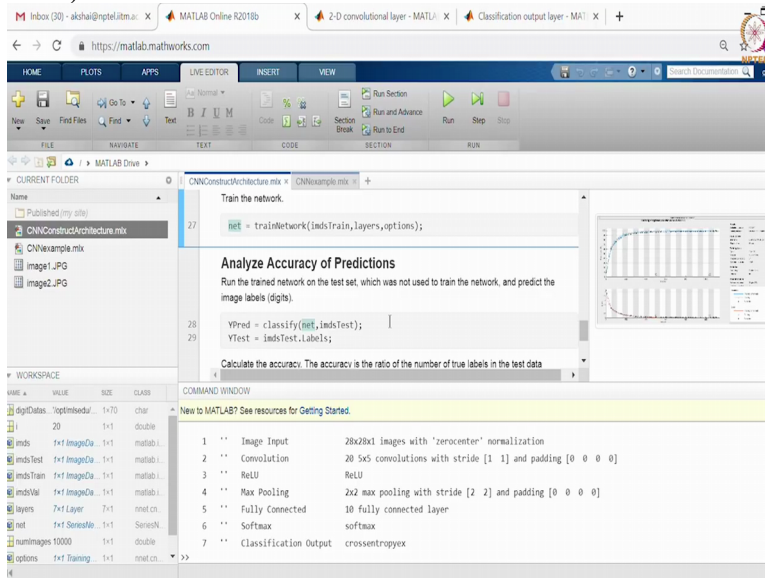


So this window will pop-up. Right? So you can see the number of iterations, it will stop at 1080, so at each epoch, the number of epochs it shows sub here right? epoch 6. So we are doing 20 epochs if you remember. This is the accuracy of classification versus number of epochs. This is the loss, binary cross entropy loss that the classification layer ( $\text{loss\_c}$ )(15:38). And we can see that 2 different curves, right? The last is the training smooth loss and there is a validation loss. You see that they are actually on top of each other.

Similarly here, the validation accuracy and the training accuracy, you can see that there too. So it is running quite fast, we should be done. The 20<sup>th</sup> epoch, that is the final, ya. So it is pretty good. So

we are almost close to 99, more than 99%, you can see what happens. It use the CPU okay, it works very well on the CPU. So that is fine. The hardware information comes up here. So that now if you are done, then we can close this window because this just pops up, just tells you how the training progresses, okay. So it is a very nice information that you can get.

(Refer Slide Time 16:30)



So once training is done, the network has the trained rates, net has the trained rates. So then now you want to see the performance on the testing data so then if you have a network, how do you do the forward pass if you have new data? Right? so that is this function classify, so run here. So it takes all the testing data, runs it through the network and gives you the output in the prediction,



this is the predicted output and the white test is basically, it tells you the labels for the data in the test data store, right? So this is the ground truth, this is your protection. Okay. So let us just to run this one. Section, this should be fairly straightforward. So it is done. Hopefully yes.

(Refer Slide Time 17:17)

The screenshot shows the MATLAB Live Editor interface. The main editor window contains the following code and text:

```

YPred = classify(net, indsTest);
YTest = indsTest.Labels;

Calculate the accuracy. The accuracy is the ratio of the number of true labels in the test data
matching the classifications from classify to the number of images in the test data.

accuracy = sum(YPred == YTest)/numel(YTest)
Figure
confusionchart(indsTest.Labels,YPred);
Copyright 2012 The MathWorks, Inc.

```

On the right side, there is a plot showing a confusion matrix and a bar chart. The accuracy is displayed as 0.9810.

The workspace table shows the following variables:

NAME	VALUE	SIZE	CLASS
accuracy	0.9810	1x1	double
ans	1x1 Confusion	1x1	miscarr
dayData	testImagesData	1x70	char
inds	20	1x1	double
indsTest	1x1 ImageData	1x1	matlab.i.
indsTrain	1x1 ImageData	1x1	matlab.i.
indsVal	1x1 ImageData	1x1	matlab.i.
layers	7x1 Layer	7x1	matlab.c.
net	1x1 SeriesN	1x1	SeriesN

The command window shows the following output:

```

New to MATLAB? See resources for Getting Started.

1 ** Image Input      28x28x1 images with 'zerocenter' normalization
2 ** Convolution     20 5x5 convolutions with stride [1 1] and padding [0 0 0 0]
3 ** ReLU            ReLU
4 ** Max Pooling     2x2 max pooling with stride [2 2] and padding [0 0 0 0]
5 ** Fully Connected 10 fully connected layer
6 ** Softmax         softmax
7 ** Classification Output crossentropy

```

The screenshot shows the MATLAB Live Editor interface with the title "Define CNN architecture". The main editor window contains the following code:

```

layers = [ ...
    imageInputLayer([28 28 1])
    convolution2dLayer(5,20)
    reluLayer
    maxpooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

```

The workspace table shows the following variables:

NAME	VALUE	SIZE	CLASS
accuracy	0.9810	1x1	double
ans	1x1 Confusion	1x1	miscarr
dayData	testImagesData	1x70	char
inds	20	1x1	double
indsTest	1x1 ImageData	1x1	matlab.i.
indsTrain	1x1 ImageData	1x1	matlab.i.
indsVal	1x1 ImageData	1x1	matlab.i.
layers	7x1 Layer	7x1	matlab.c.
net	1x1 SeriesN	1x1	SeriesN

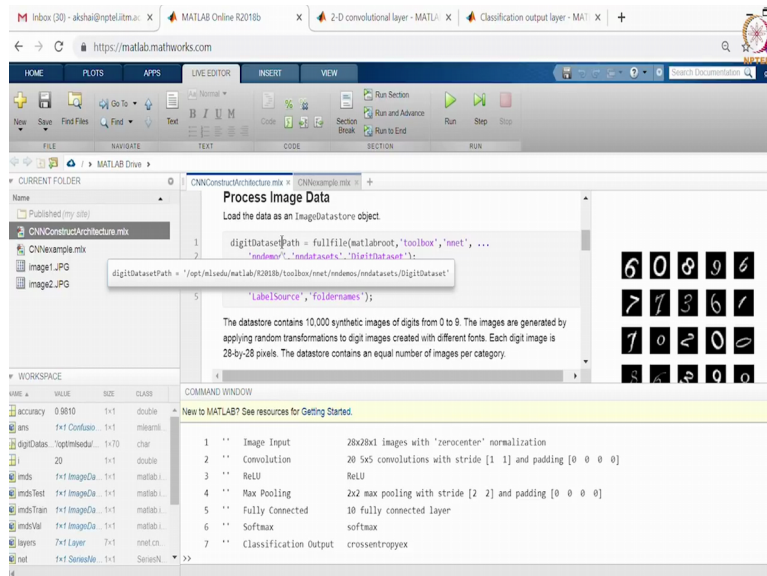
The command window shows the following output:

```

New to MATLAB? See resources for Getting Started.

1 ** Image Input      28x28x1 images with 'zerocenter' normalization
2 ** Convolution     20 5x5 convolutions with stride [1 1] and padding [0 0 0 0]
3 ** ReLU            ReLU
4 ** Max Pooling     2x2 max pooling with stride [2 2] and padding [0 0 0 0]
5 ** Fully Connected 10 fully connected layer
6 ** Softmax         softmax
7 ** Classification Output crossentropy

```



Then you can look at the accuracy. Okay. And you can look at the confusion matrix, both of them using these 3 lines of code. So we will just run that and just check whether this works out fine. Okay. So the accuracy is okay. The sum of Y predicted equal to equal to Y test. So this is the logical test. Okay. This is not the equal to sign. So it just considers all the predictions that are equal to your ground truth. So it says that you have one. So you add them up. So the number of Y predictions divided by total number of test data points, that will give you the accuracy.

The confusion chart, again is a function with the test label and predictions, it gives you the confusion matrix, okay. So this is the predicted class as opposed to the true class. And you know and if it is off, its main diagonal, then those are the false predictions okay. So that is what we, we are looking at in the confusion matrix. Typical confusion matrix, it will be nice if it is a diagonal. Typically it is not. So we have some errors in our predictions but it is a pretty good performance like that.

Okay. So this is just a very simple example of how we can do, we can create a CNN for, these are small images for Emnist classification. You can adopt the same strategy, of course each of the, the important piece of code, I mean all of them are important in order to make the network. So you just have to, in order to create the architecture you just create this layers variable which has every conceivable kind of layer possible, okay. So if you want to clear a deep network, then you just have to put in those commands there and MATLAB does the rest for you.

And you should also be able to configure your optimisation with the right algorithm, with the right choices and if you go wondering what the choices are, you just do doc training options. I will show you what are all available and you can choose the corresponding algorithm. We looked at some of the more commonly used optimisation algorithms, they are all enabled in MATLAB most of them should be, so it should be fine. Okay. So this is just a, like I said, simple example. So in the later weeks, there will be one application week, maybe we will look at a slightly more complicated architecture, they will look at what they are like.

We may try to code them in MATLAB but demonstration will be difficult because of the size of the data. Another point to note is that you look at how the data was organised, okay. So they have a root, a MATLAB, a main directory which has all the folders which has the data stored in subdirectories and each subdirectory is named according to its class and inside each subdirectory is the actual data, okay. So when you are trying to solve a new problem, it is up to you. So this is a canned problem, MATLAB has done all the work for you.

But if you have some new data and you are trying to let us say run a CNN or for that matter, any network through it, or any network on that data, to train the network using that data then it is your responsibility to create these folders. Okay. So you have to curate the data yourself, that is the essential part of you know coding (21:04). Make sure that the data is clean, nicely organised because once you have them organised in this fashion, then you can exploit the existing methods and functions that are in MATLAB right?

Nothing prevents you from writing your own data store, writing your own code to load one image or 100s of images in a time, at a time and training your network with it. I mean you can do that too but as it is very convenient if you use the MATLAB Datastore. And for that, you have to organise your data in this fashion, right? So make sure that your subfolders or subdirectories are named according to your classes that you have for supervised training and then you have and then make sure that all the subfolders are in one fixed place from which you all from which MATLAB can upload the data sets. Right?

And of course you have to curate them, separate the data and all that. So that is, that is a lot of hard work, most of the time, that takes up a lot of your time okay. So this is how we start off this week. We will be looking at transfer learning, again hyper parameter optimization, we will also look at

one application in how CNNs can be used for medical image analysis you know, brain tumour segmentation specifically. Okay. So thank you.