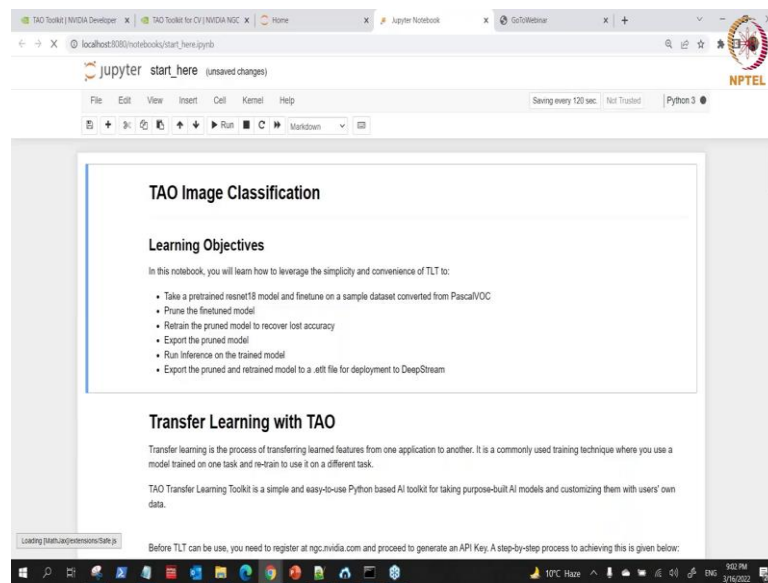


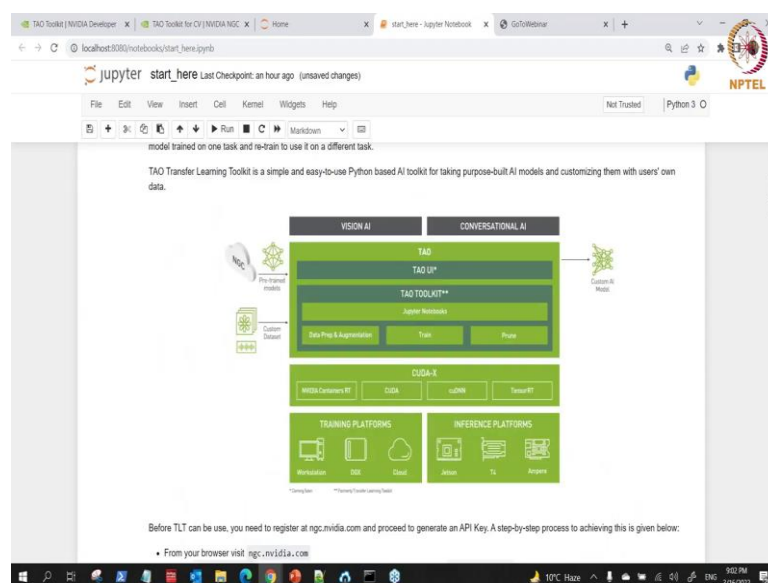
Applied Accelerated Artificial Intelligence
Dr. Tosin Adesuyi
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

End to End Accelerated Date Learning
Lecture - 33
Optimizing Deep Learning Training: Transfer Learning Part - 2

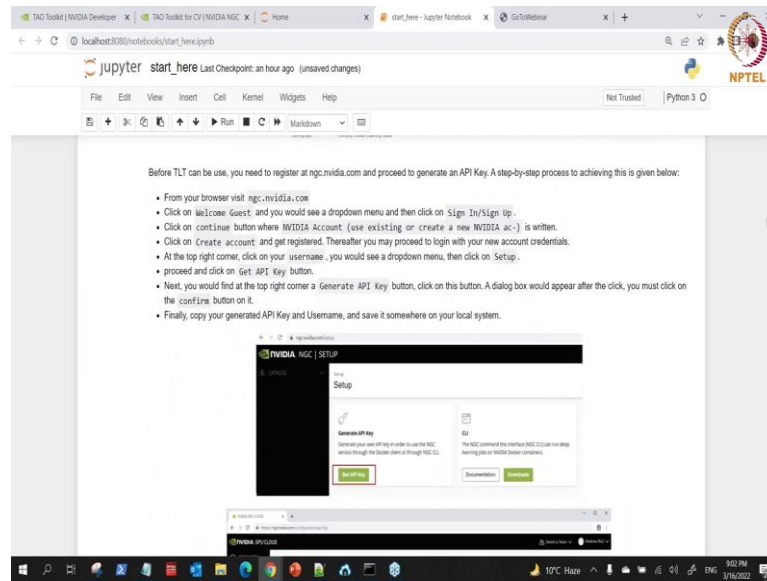
(Refer Slide Time: 00:14)



(Refer Slide Time: 00:19)

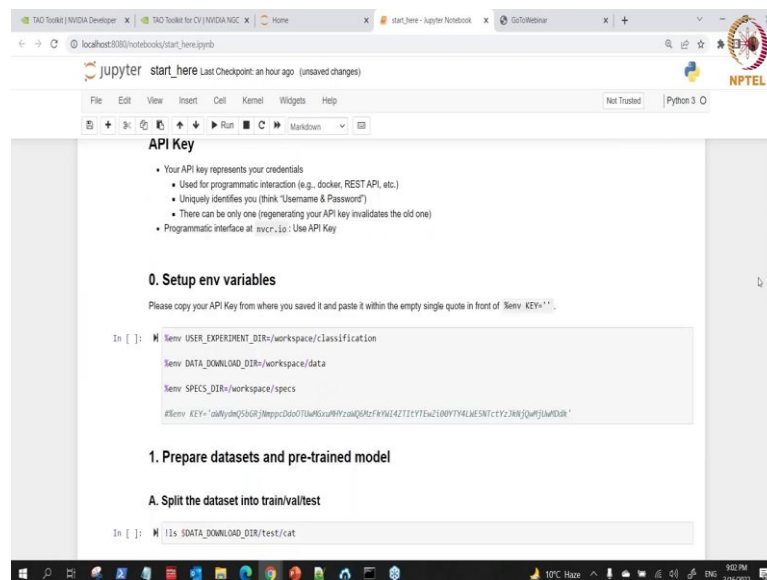


(Refer Slide Time: 00:21)



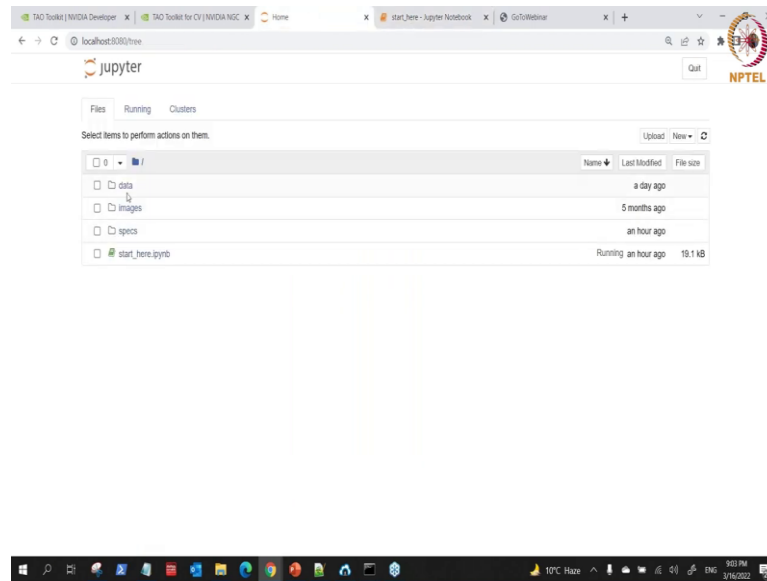
So, here we go. Our notebook is ready. So, next thing to do is I explain all what you can see here. So, let us just go quickly, so that we do not waste time. We go to run the notebook.

(Refer Slide Time: 00:23)



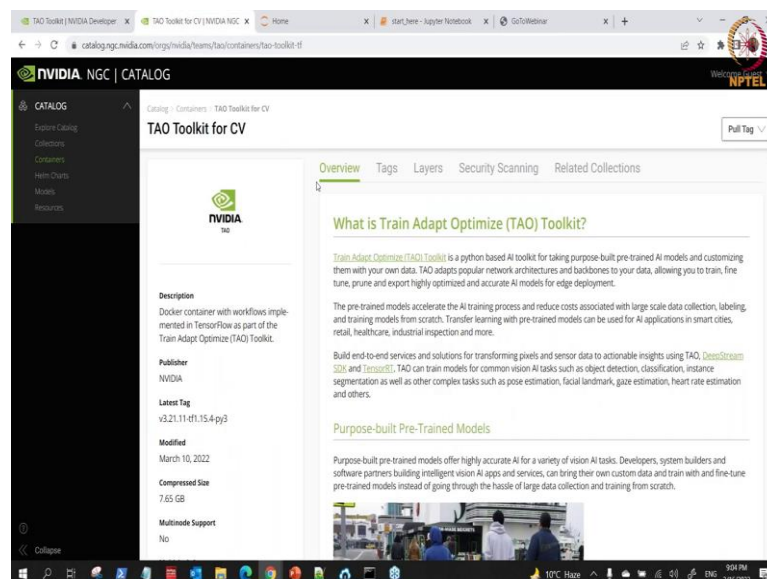
So, the first thing we have to do is to set our environment variable. We set the environment variable to USER_EXPERIMENT_DIR. So, it to be on the workspace classification. This one is by default. Then, the next one is what where we have our data.

(Refer Slide Time: 00:45)



So, we are setting this path, this is what we are setting this path to data. So, this is to data. Then, we set for our specification workspace to the specification file. This is for this specification file the path. So, why we said this is that we do not want along the line we need those path and we do not want to be writing lengthy path that way. So, it will just be represented by all these variables there.

(Refer Slide Time: 01:32)

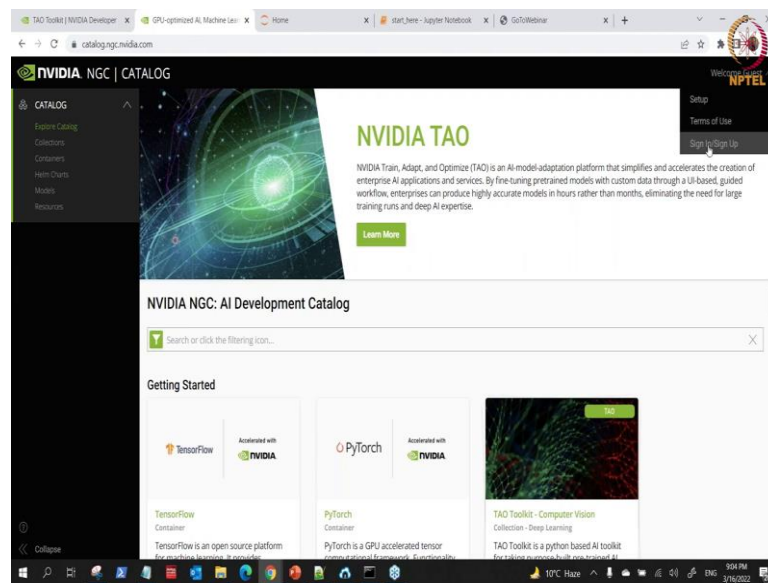


So, the next one is we need environment key. Now, the environment key, you need to have an environment your key for the TAO key. So, what you need to do API key, is

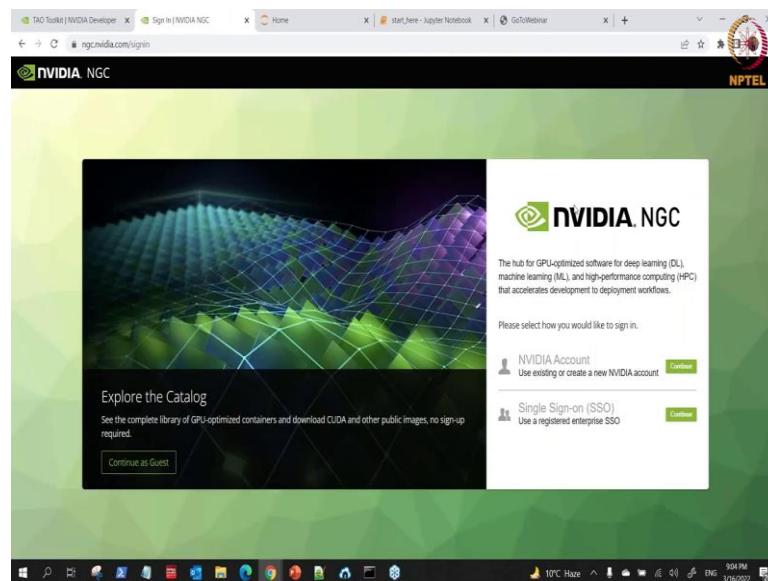
called API key rather; you need to register on the NVIDIA NGC forum. These are the steps you need to follow, but I will show you the step. Now, let us quickly run that.

So, when you get here you know you pull, when you pull your container you do not need to, you do not need any registration, but now you will want to get our API key. So, what we need to do is what you come here; ok.

(Refer Slide Time: 01:50)



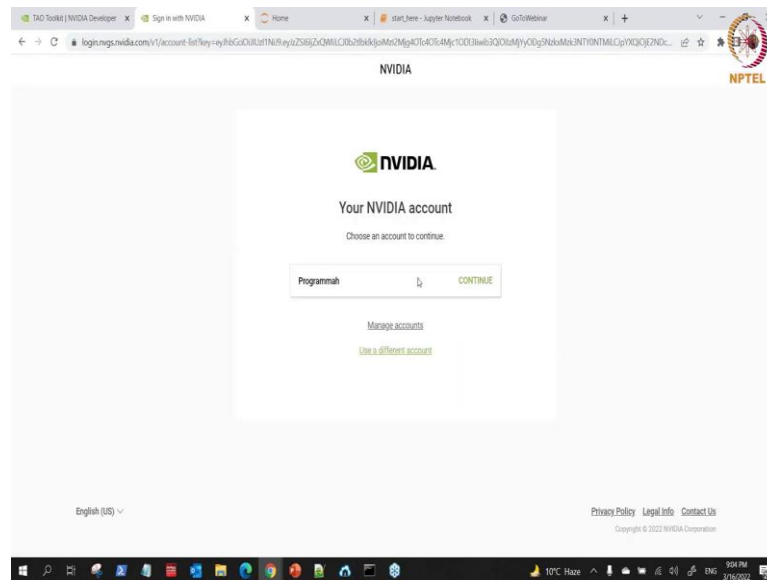
(Refer Slide Time: 01:57)



Let me go back to and you see here this is the first page.

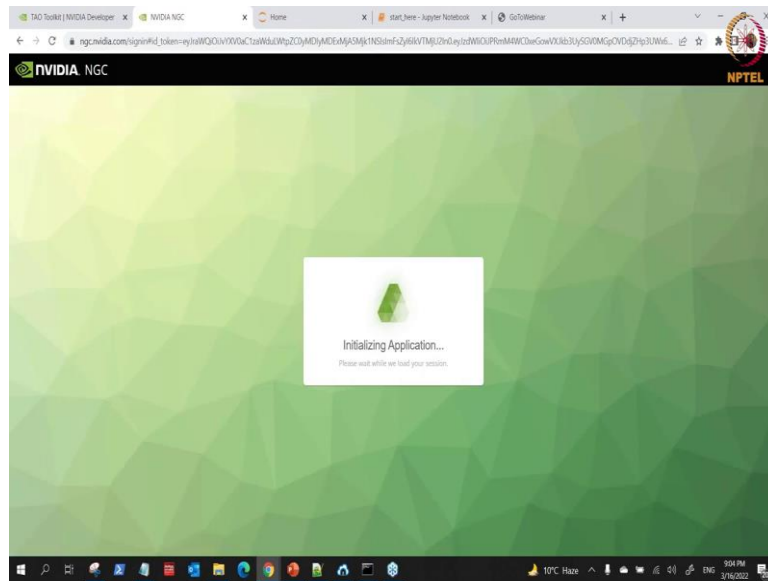
You see where you have guest here. So, what you need to do is what you click on sign in or sign up, sign in or sign up. So, when you have your sign in or sign up, when you get here, you click you make use of the NVIDIA account, use existing account continue, that is the one you click, first one.

(Refer Slide Time: 02:09)



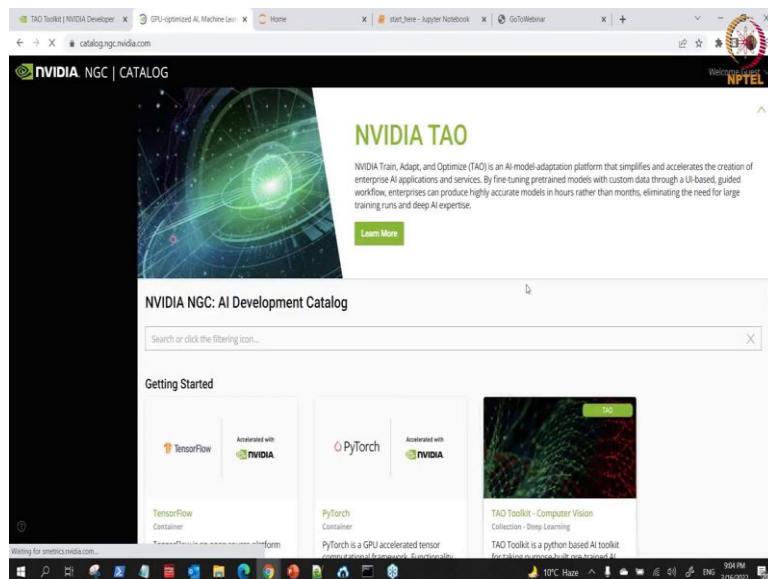
So, because I have an account, I can continue this way. For those that have not registered at all, you can use a click on different account. You will see where you sign up. So, after you sign up you can come back and then log in. So, once you log in like the way I am going to, it is going to allow me to log in here.

(Refer Slide Time: 02:29)



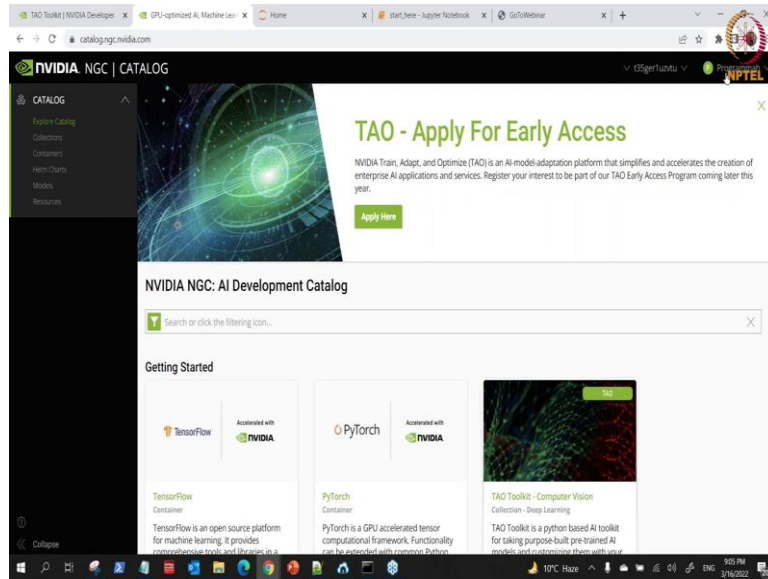
So, I am logging in now.

(Refer Slide Time: 02:33)



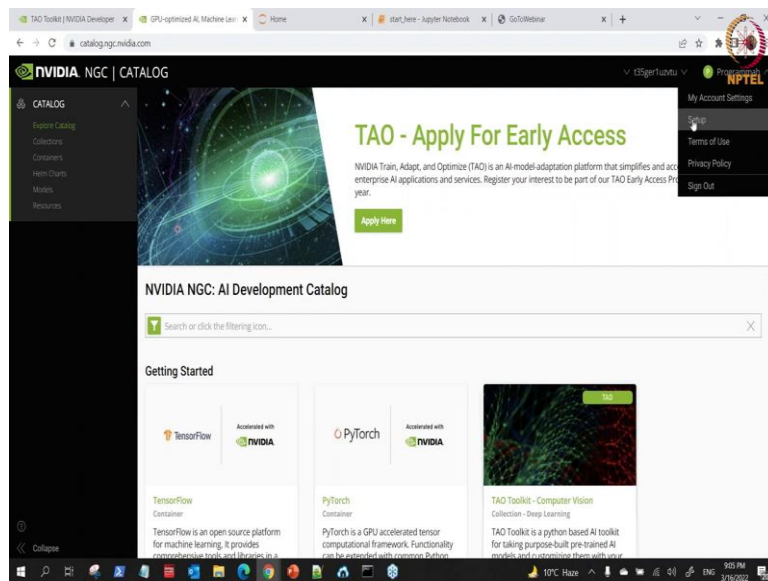
So, yes what I am looking for login then you see this place will change it will no longer be guest.

(Refer Slide Time: 02:34)

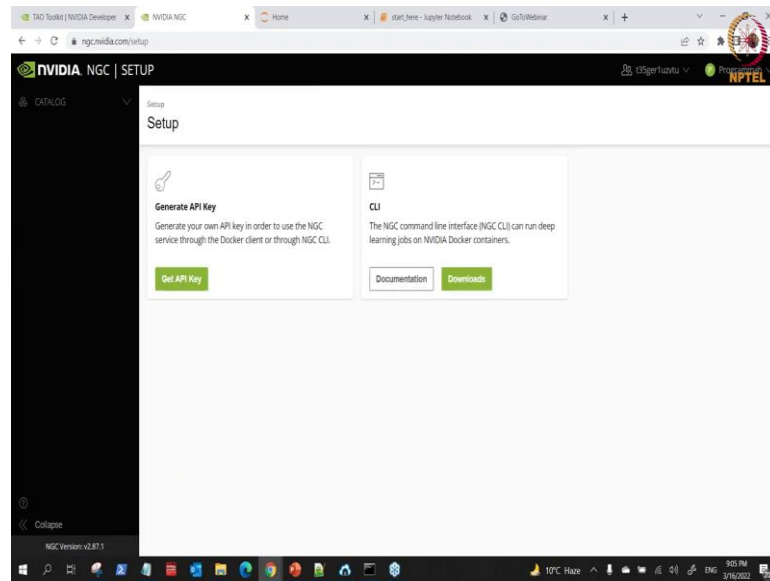


You see my, it has changed my name here programmer.

(Refer Slide Time: 02:43)

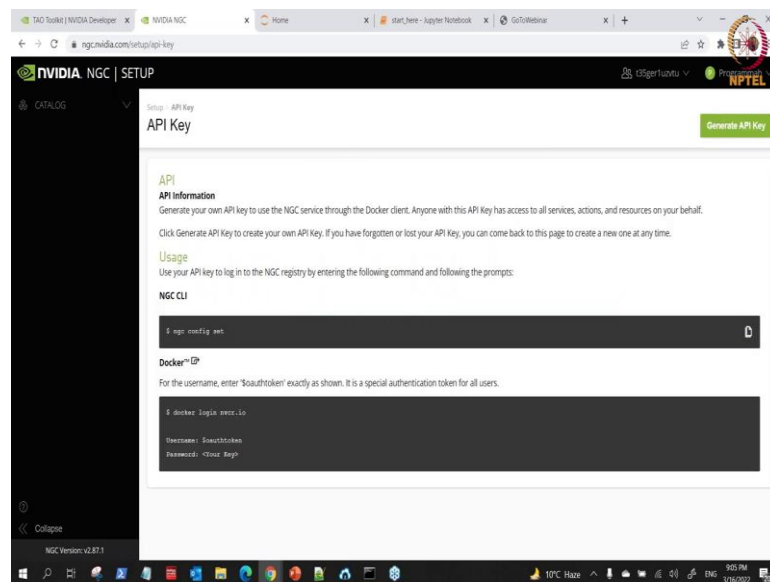


(Refer Slide Time: 02:48)



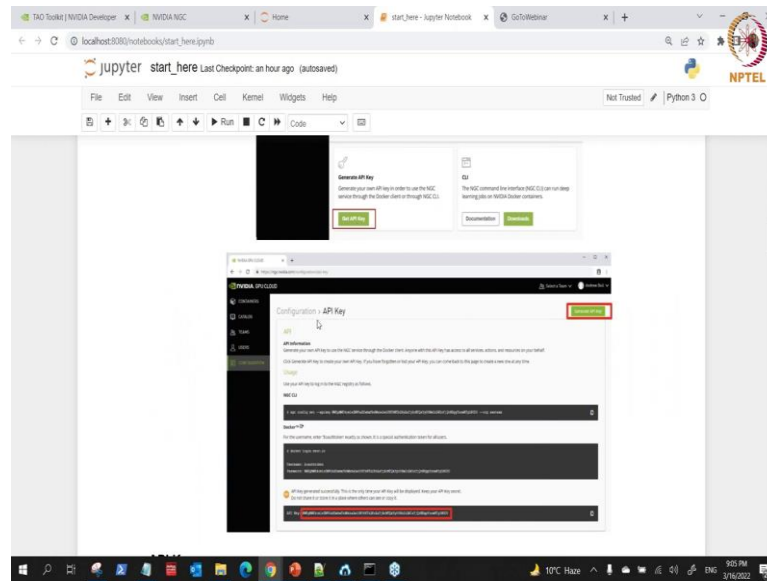
So, what you need to do next is to drop it down and look for click on word setup. So, you click on setup and it will load. So, you need an API key. So, you can click on the get API key.

(Refer Slide Time: 02:54)



So, when you click on get API key, the next phase to do is to work to generate, to click on generate. But because I have key, I do not need to click on generate again. So, we can come back here and say see.

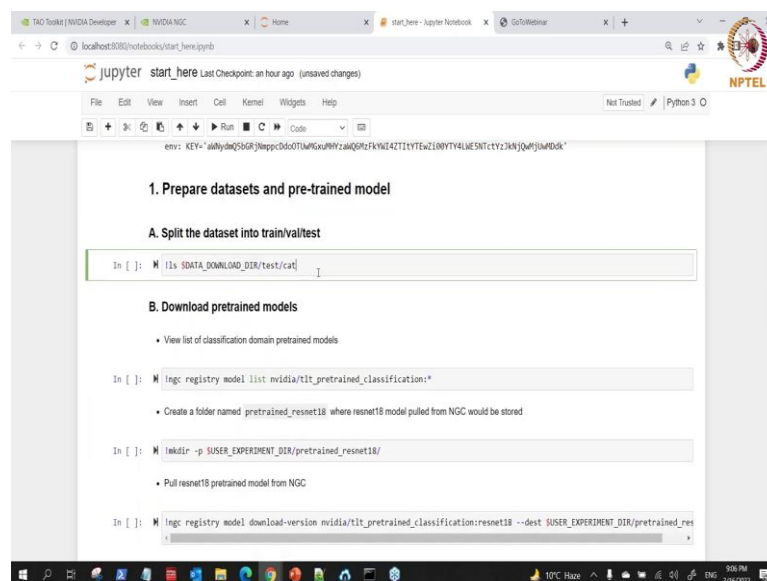
(Refer Slide Time: 03:08)



So, when you click on the API key, get API key, it will bring this interface. So, on this interface you click on generate, when it generates here, you will see the API key here. So, you copy it and save it somewhere. So, anytime you want to work with you using your NVIDIA TAO, you can now work paste the API key. Make use of it here. So, this is our API key which is being used.

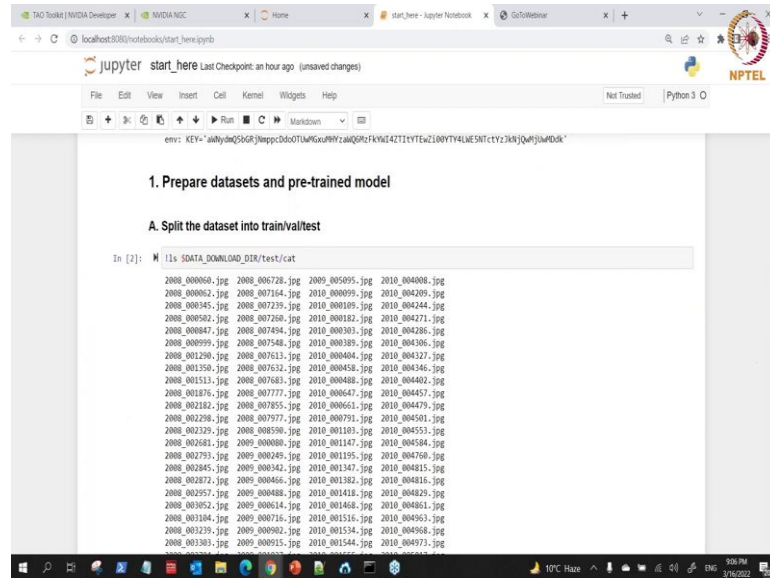
So, the first thing to do now is to run this cell. So, I am going to run this cell. So, everything is initialized this way, running this cell.

(Refer Slide Time: 03:41)



So, the next one is to let me see I want to know if my data are visible. So, I want to look at the test data to look at what just one class is cat.

(Refer Slide Time: 03:52)



```
env: KEY="aWlydnQ2S6RjNppcDdo0TlMGcUPRZakQ0rFKYNE4ZT1YtEwZ100YTY4LIESNTCYz3KjQmJSMjMEdk"
```

1. Prepare datasets and pre-trained model

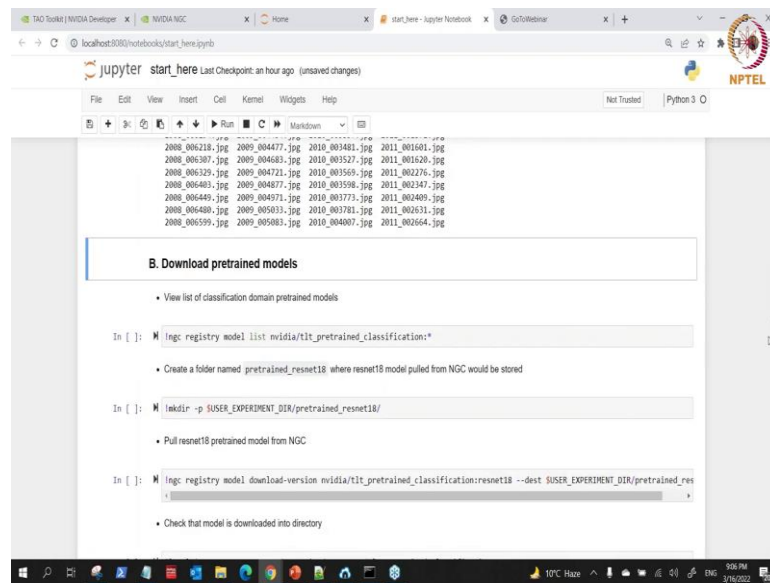
A. Split the dataset into train/val/test

```
In [2]: !ls $DATA_DOWNLOAD_DIR/test/cat
```

```
2008_000060.jpg 2008_006728.jpg 2009_005095.jpg 2010_004008.jpg
2008_000062.jpg 2008_007164.jpg 2010_000099.jpg 2010_004209.jpg
2008_000045.jpg 2008_007235.jpg 2010_000109.jpg 2010_004244.jpg
2008_000050.jpg 2008_007260.jpg 2010_000185.jpg 2010_004271.jpg
2008_000047.jpg 2008_007494.jpg 2010_000303.jpg 2010_004286.jpg
2008_000999.jpg 2008_007548.jpg 2010_000309.jpg 2010_004306.jpg
2008_001290.jpg 2008_007613.jpg 2010_000404.jpg 2010_004327.jpg
2008_001350.jpg 2008_007632.jpg 2010_000454.jpg 2010_004346.jpg
2008_001513.jpg 2008_007683.jpg 2010_000468.jpg 2010_004402.jpg
2008_001876.jpg 2008_007777.jpg 2010_000647.jpg 2010_004457.jpg
2008_002182.jpg 2008_007855.jpg 2010_000661.jpg 2010_004479.jpg
2008_002290.jpg 2008_007977.jpg 2010_000791.jpg 2010_004501.jpg
2008_002329.jpg 2008_008590.jpg 2010_001103.jpg 2010_004553.jpg
2008_002681.jpg 2009_000000.jpg 2010_001147.jpg 2010_004584.jpg
2008_002793.jpg 2009_000249.jpg 2010_001195.jpg 2010_004760.jpg
2008_002845.jpg 2009_000342.jpg 2010_001347.jpg 2010_004815.jpg
2008_003871.jpg 2009_000466.jpg 2010_001382.jpg 2010_004816.jpg
2008_002957.jpg 2009_000488.jpg 2010_001418.jpg 2010_004829.jpg
2008_003052.jpg 2009_000614.jpg 2010_001468.jpg 2010_004861.jpg
2008_003104.jpg 2009_000716.jpg 2010_001516.jpg 2010_004963.jpg
2008_003239.jpg 2009_000902.jpg 2010_001534.jpg 2010_004968.jpg
2008_003381.jpg 2009_000915.jpg 2010_001544.jpg 2010_004973.jpg
2008_003400.jpg 2009_000922.jpg 2010_001554.jpg 2010_004974.jpg
```

So, it can fetch that is, ok it is reachable, our data is reachable.

(Refer Slide Time: 03:57)



```
2008_006210.jpg 2009_004477.jpg 2010_003481.jpg 2011_001601.jpg
2008_006307.jpg 2009_004683.jpg 2010_003527.jpg 2011_001620.jpg
2008_006320.jpg 2009_004721.jpg 2010_003569.jpg 2011_002276.jpg
2008_006403.jpg 2009_004877.jpg 2010_003598.jpg 2011_002347.jpg
2008_006440.jpg 2009_004971.jpg 2010_003773.jpg 2011_002409.jpg
2008_006480.jpg 2009_005033.jpg 2010_003781.jpg 2011_002631.jpg
2008_006599.jpg 2009_005083.jpg 2010_004007.jpg 2011_002664.jpg
```

B. Download pretrained models

- View list of classification domain pretrained models

```
In [ ]: !ngc registry model list nvidia/tit_pretrained_classification:*
```

- Create a folder named pretrained_resnet18 where resnet18 model pulled from NGC would be stored

```
In [ ]: !mkdir -p $USER_EXPERIMENT_DIR/pretrained_resnet18/
```

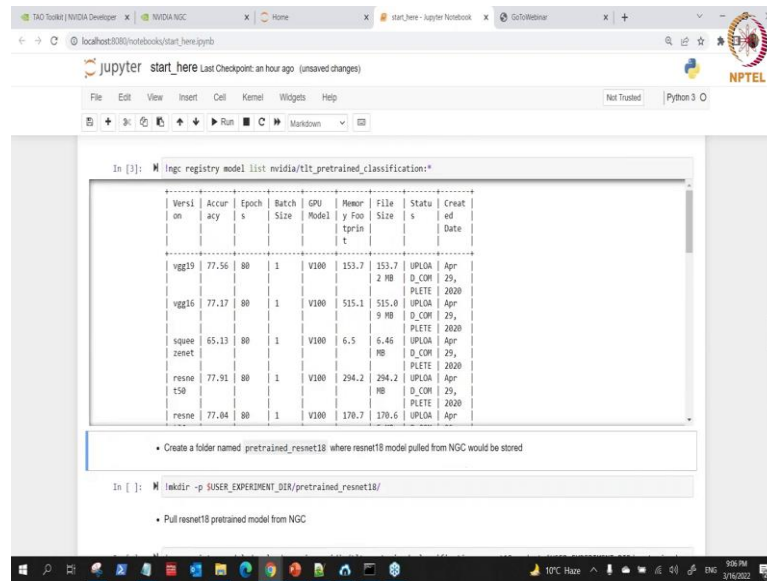
- Pull resnet18 pretrained model from NGC

```
In [ ]: !ngc registry model download-version nvidia/tit_pretrained_classification:resnet18 --dest $USER_EXPERIMENT_DIR/pretrained_resnet18/
```

- Check that model is downloaded into directory

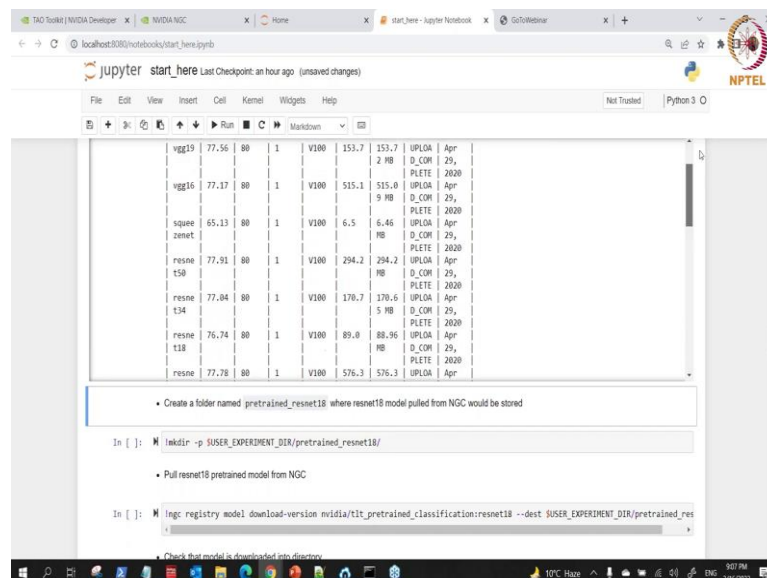
Then, the next thing to do is to see what are the available, what are the available pre-trained model for classification that exists here.

(Refer Slide Time: 04:25)

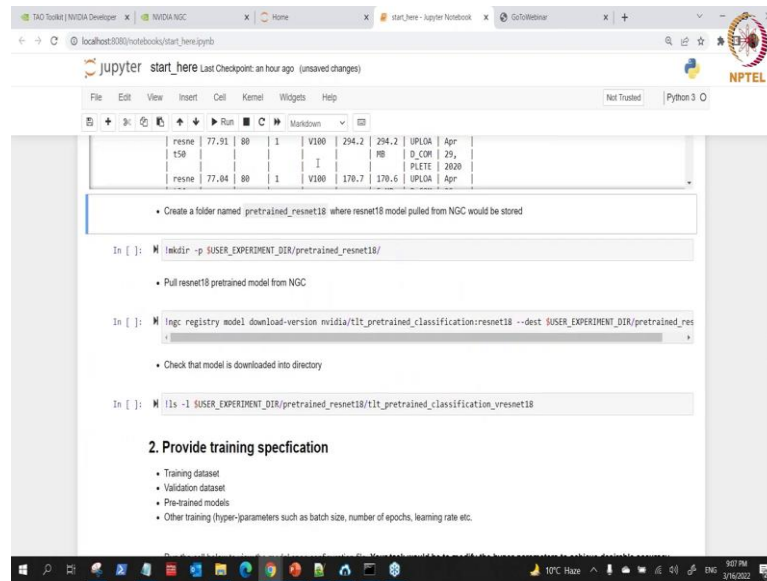


So, with this ngc registry model list, it will list them nvidia/tlt_pretrained_classification:*, that is all. If you show all it can show all to you. So, we will be able to see the list of all the, yes it has shown all of that. So, you can see versions vgg19, vgg16, their accuracy, the epoch you can see. So, there are many of them there resnet 50, 34, 18, 101 and all of them. So, we can make use of you can make use of any of them that are there. So, we can make use of them.

(Refer Slide Time: 04:44)



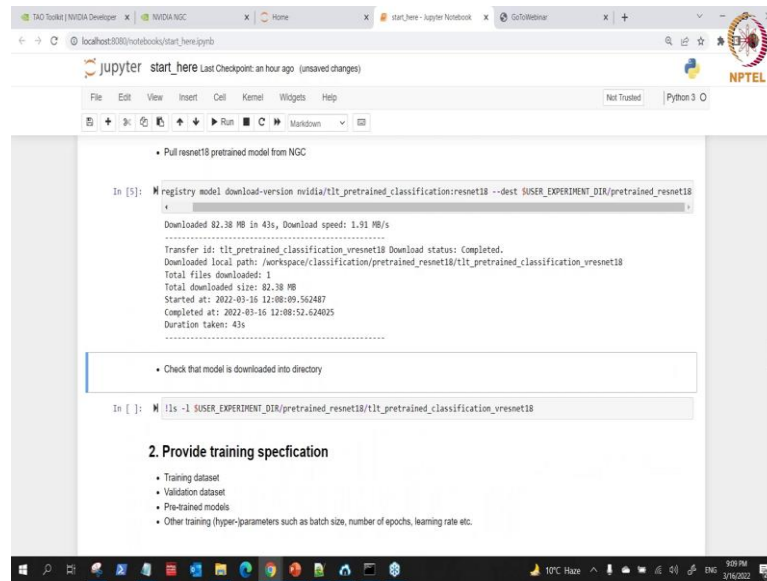
(Refer Slide Time: 04:52)



So, the next phase not to waste time. So, is to what; since we know the pretrained model the that exists we can now make a folder directory where we will save our downloaded pretrained model.

So, let us create a directly call pretrained resnet18 because that is what we want to use. So, I am running that. So, that is done. You can see when its gives a number it shows it has run. And then, so now, let us pull that what pretrained model into this directory, yeah. So, how do you put that? You have, do you when you write ngc registry the model, then download version. So, we have nvidia/tlt_pretrained_ classification. So, what kind of model? Resnet18.

(Refer Slide Time: 05:46)



```
• Pull resnet18 pretrained model from NGC

In [5]: !registry model download-version midia/tlt_pretrained_classification:resnet18 --dest $USER_EXPERIMENT_DIR/pretrained_resnet18

Downloaded 82.38 MB in 43s, Download speed: 1.91 MB/s
-----
Transfer id: tlt_pretrained_classification_vresnet18 Download status: Completed.
Downloaded local path: /workspace/classification/pretrained_resnet18/tlt_pretrained_classification_vresnet18
Total files downloaded: 1
Total downloaded size: 82.38 MB
Started at: 2022-03-16 12:08:09.562407
Completed at: 2022-03-16 12:08:52.624025
Duration taken: 43s
-----

• Check that model is downloaded into directory

In [ ]: !ls -l $USER_EXPERIMENT_DIR/pretrained_resnet18/tlt_pretrained_classification_vresnet18
```

2. Provide training specification

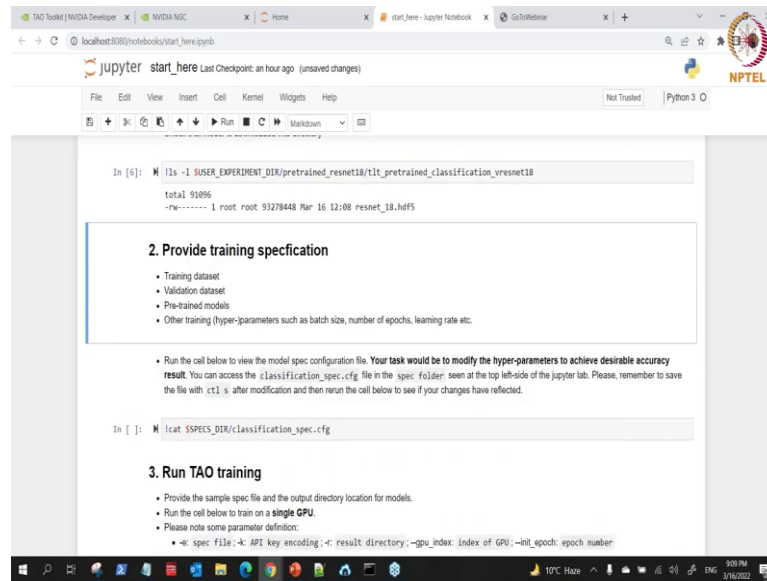
- Training dataset
- Validation dataset
- Pre-trained models
- Other training (hyper-)parameters such as batch size, number of epochs, learning rate etc.

The destination you should save it here because you know we created this folder this one we created it here. So, it is going to you should save it in this destination. So, I can run that now. So, it is going to run and we can see our model will be downloaded. So, the download has begun.

So, why that is loading? So, once that is completed, the next thing to do is to just verify and see whether it is the directory is the right directory. So, just confirm if it is a right directory because of the ssh it should have been much more faster than this I guess, should be faster than this. So, that have not done, ok, yeah. So, it is done. So, the model has been work has been unloaded.

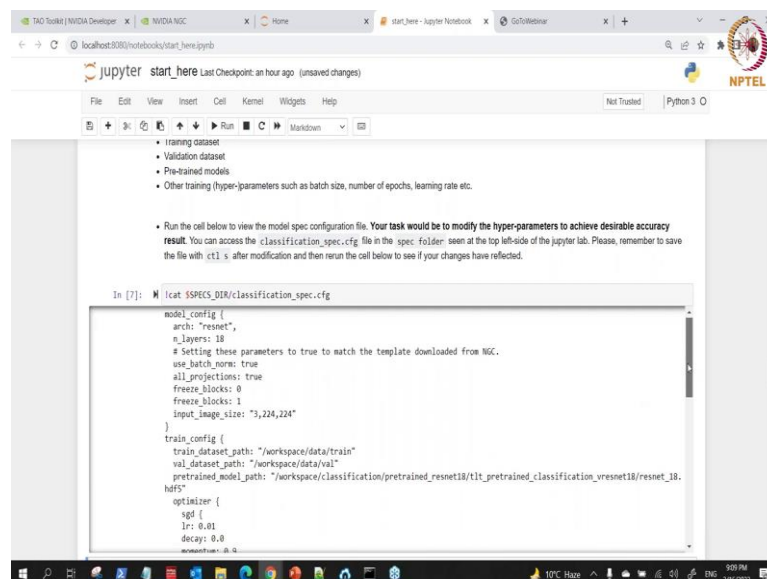
So, we can view the model here. So, we so, the model resnet_18.hdf5. So, that has been done.

(Refer Slide Time: 06:41)

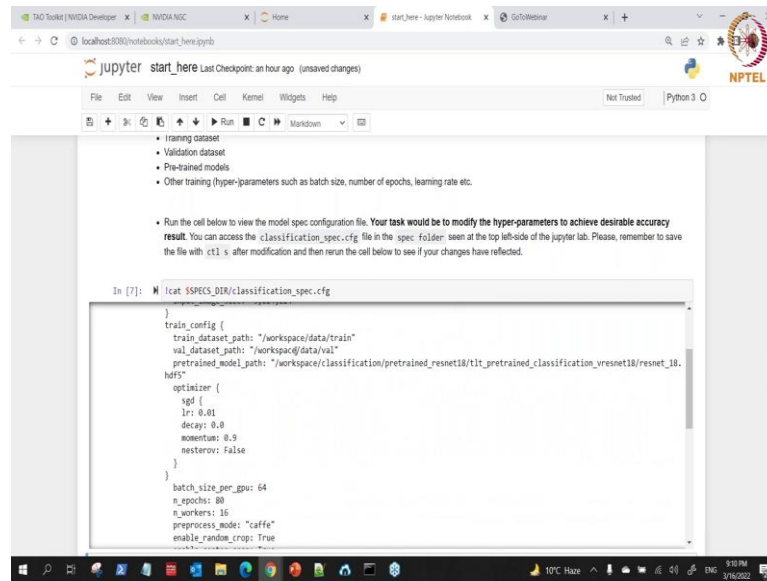


So, the next thing to do now we want to train. Since, we have our model, we have our data set, we want to train now. So, before we train like I have explained earlier on that, you need to configure your configuration that is specification file and this is the path to the configuration file, the specification file, this file, yeah. So, we want to see the one for training is the one we call the classification and that is called spec.cfg.

(Refer Slide Time: 07:14)



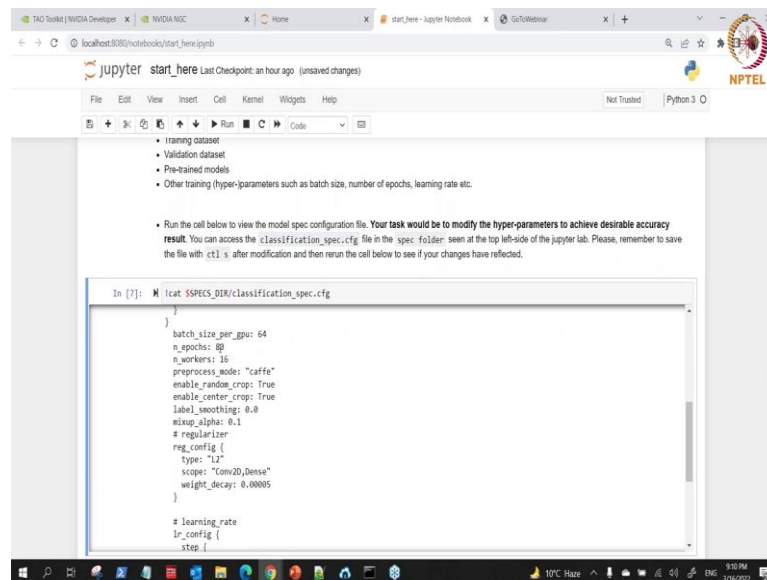
(Refer Slide Time: 07:36)



```
In [7]: !cat $SPECS_DIR/classification_spec.cfg
}
}
train_config {
  train_dataset_path: "/workspace/data/train"
  val_dataset_path: "/workspace/data/val"
  pretrained_model_path: "/workspace/classification/pretrained_resnet18/t1_pretrained_classification_vresnet18_hdf5"
  optimizer {
    sgd {
      lr: 0.01
      decay: 0.0
      momentum: 0.9
      nesterov: false
    }
  }
  batch_size_per_gpu: 64
  n_epochs: 80
  n_workers: 16
  preprocess_mode: "caffe"
  enable_random_crop: True
}
```

So, we can open that and see what is inside. So, if you look at it this is what I showed you before. The architecture is what resnet, the layers 18, our input is 3; 3 by 224 by 224. The path to our training, the path to our validation set, the path to our model to be trained all of that you can see here.

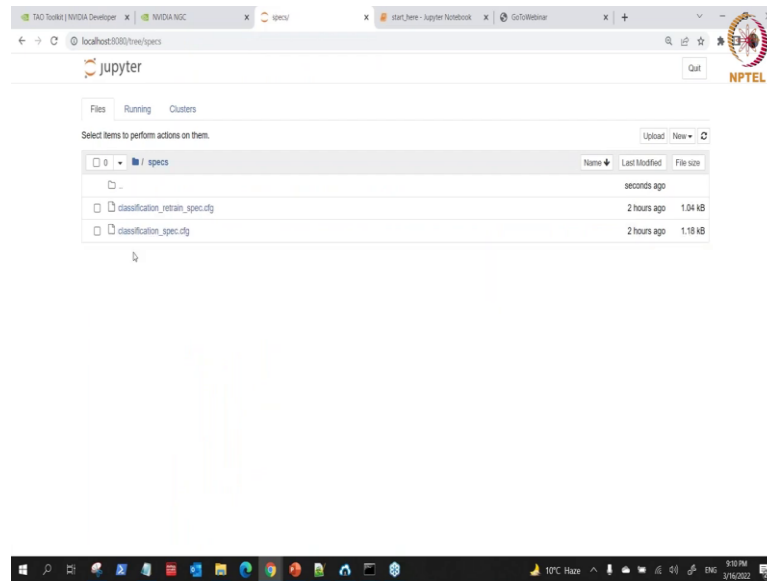
(Refer Slide Time: 07:51)



```
In [7]: !cat $SPECS_DIR/classification_spec.cfg
}
}
batch_size_per_gpu: 64
n_epochs: 80
n_workers: 16
preprocess_mode: "caffe"
enable_random_crop: True
enable_center_crop: True
label_smoothing: 0.0
mixup_alpha: 0.1
# regularizer
reg_config {
  type: "l2"
  scope: "Conv2D,Dense"
  weight_decay: 0.00005
}
# learning_rate
lr_config {
  step: 1
```

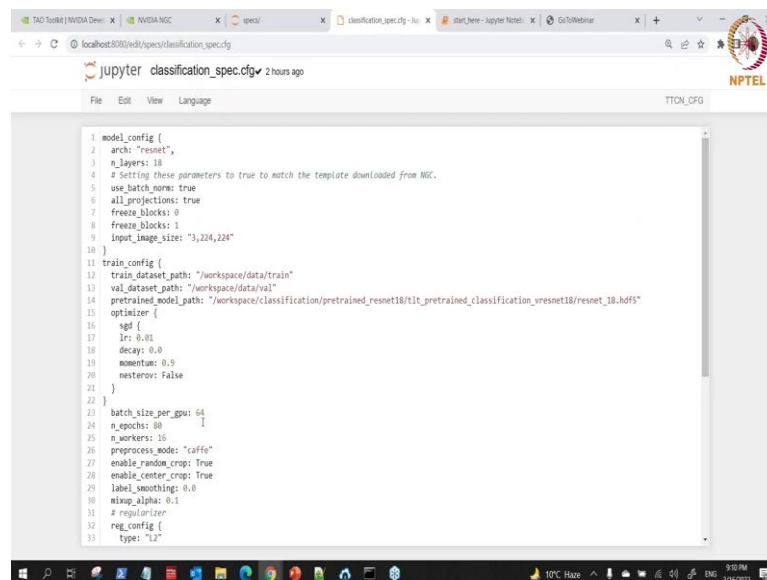
And then the number of epoch, now this is 80. We cannot use 80 now because this is for demo. So, otherwise we want to sleep here. So, we need to change this for this demo, yeah.

(Refer Slide Time: 07:56)



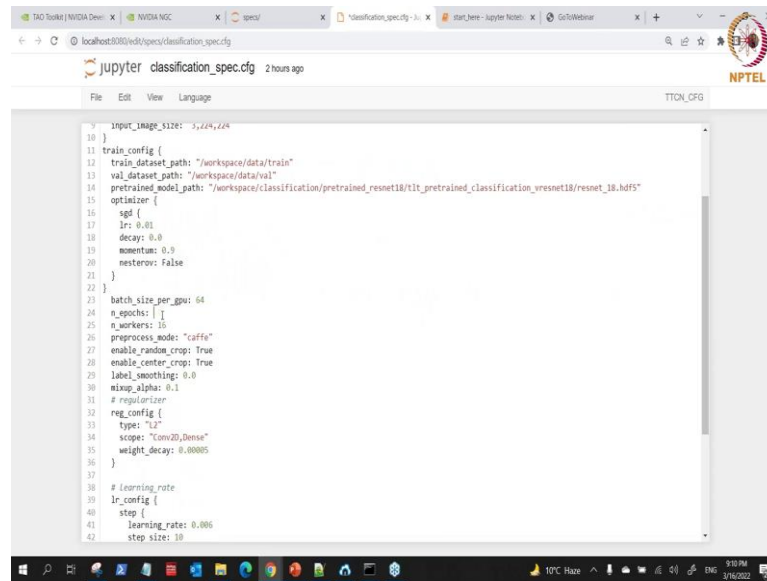
So, for this demo. So, you can come here, I can come here. So, if you are training on your workstation you do not need to change because that is your own time, but now we just have limited time.

(Refer Slide Time: 08:07)



So, what I will do is what I open this, this is the one for training, then I will look for where we have the 80 just to show you, just to show. So, we are less concerned about accuracy now because just use 4 epochs. So, that it can train faster.

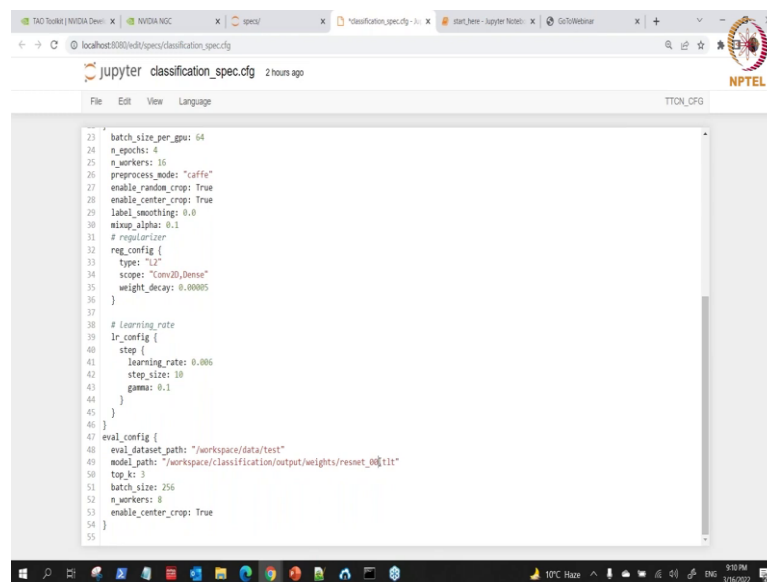
(Refer Slide Time: 08:13)



```
9 input_image_size: 224,224
10 }
11 train_config {
12   train_dataset_path: "/workspace/data/train"
13   val_dataset_path: "/workspace/data/val"
14   pretrained_model_path: "/workspace/classification/pretrained_resnet18/11t_pretrained_classification_resnet18_bdfs"
15   optimizer {
16     sgd {
17       lr: 0.01
18       decay: 0.0
19       momentum: 0.9
20       nesterov: False
21     }
22   }
23   batch_size_per_gpu: 64
24   n_epochs: [ ]
25   n_workers: 16
26   preprocess_mode: "caffe"
27   enable_random_crop: True
28   enable_center_crop: True
29   label_smoothing: 0.0
30   mixup_alpha: 0.1
31   # regularizer
32   reg_config {
33     type: "L2"
34     scope: "Conv2D,Dense"
35     weight_decay: 0.0005
36   }
37 }
38 # Learning rate
39 lr_config {
40   step {
41     learning_rate: 0.005
42     step_size: 10
43   }
44 }
45 }
46 }
47 eval_config {
48   eval_dataset_path: "/workspace/data/test"
49   model_path: "/workspace/classification/output/weights/resnet_0811t"
50   top_k: 3
51   batch_size: 256
52   n_workers: 8
53   enable_center_crop: True
54 }
55 }
```

And when you say, 4 we have to consider the evaluation also because this is for 80. So, you have to change, I have to change this also.

(Refer Slide Time: 08:28)



```
23 batch_size_per_gpu: 64
24 n_epochs: 4
25 n_workers: 16
26 preprocess_mode: "caffe"
27 enable_random_crop: True
28 enable_center_crop: True
29 label_smoothing: 0.0
30 mixup_alpha: 0.1
31 # regularizer
32 reg_config {
33   type: "L2"
34   scope: "Conv2D,Dense"
35   weight_decay: 0.0005
36 }
37 }
38 # Learning rate
39 lr_config {
40   step {
41     learning_rate: 0.005
42     step_size: 10
43     gamma: 0.1
44   }
45 }
46 }
47 eval_config {
48   eval_dataset_path: "/workspace/data/test"
49   model_path: "/workspace/classification/output/weights/resnet_0811t"
50   top_k: 3
51   batch_size: 256
52   n_workers: 8
53   enable_center_crop: True
54 }
55 }
```

Well, you do not need to bother about this. This is your training for me ideally you should train with 80 epoch or more because for training because for demo that is why I reduce these to just 4 epoch. So, I have saved that, then I close that, I get back here.

(Refer Slide Time: 08:57)

```
In [ ]: # !classification train -e $SPECS_DIR/classification_spec.cfg -r $USER_EXPERIMENT_DIR/output -k $KEY

# To run this training using multiple GPUs, please uncomment the cell below and update the --gpus parameter to the number of GPU's you wish to use.
# However, you are restricted to maximum of 2 GPUs per teams on the cluster.

In [*]: # !classification train -e $SPECS_DIR/classification_spec.cfg \
        -r $USER_EXPERIMENT_DIR/output \
        -k $KEY --gpus 2

Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.

# To resume from a checkpoint use --init_epoch along with your checkpoint configured in the spec file
# Please make sure that the model_path in the spec file is now updated to the .tlt file of the corresponding epoch you wish to resume from. You may
# choose from the files found under $USER_EXPERIMENT_DIR/output/weights folder.

In [ ]: # !classification train -e $SPECS_DIR/classification_spec.cfg \
        # -r $USER_EXPERIMENT_DIR/output \
        # -k $KEY --gpus 2 \
        # --init_epoch N

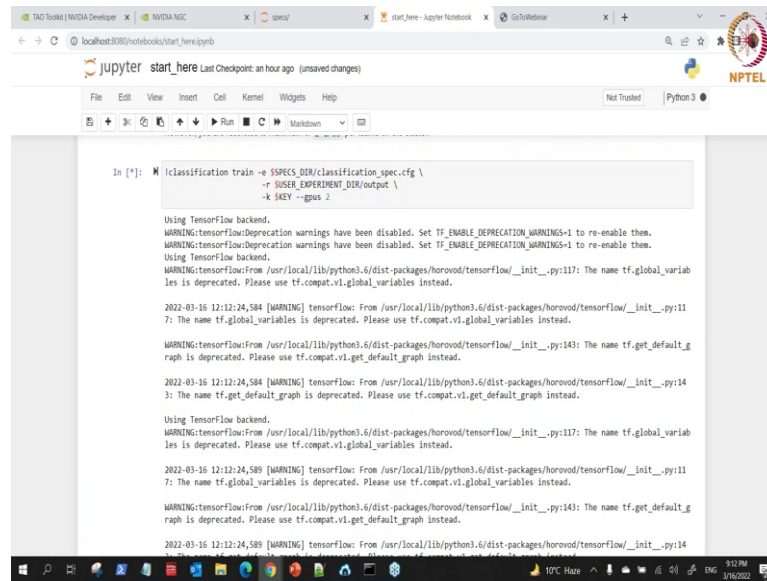
4. Evaluate trained models
```

So, let me relaunch this one again and see, ok it has been affected. So, we can go now. No train for just for epoch. The result might not be good, but we are not bothered by that one for now because we just want to show you demo.

So, here you can train if you want to train you have classification, then you this keyword train. So, these are flag here, here what is specify is that it is referring to the specification file which is just configured now. And then so, when it finished training where does it need to save it, this one refer to that it will save it in a folder called output. And then it needs an assets. So, this is the key assets. So, these are the flags to use to do that.

So, this in this is meant for just single GPU. So, because I have two GPUs, let me use two GPUs. This is meant for two GPUs here. So, I can just specify this as. So, if you want to specify number of GPU, this is how you do that --gpus and 2. If you have 4 GPUs on your workstation, it will be work 4. If you have 8, it will be 8. So, I have just to; so, I can use this now and run this.

(Refer Slide Time: 10:02)



```
In [*]: !classification_train -e $SPECES_DIR/classification_spec.cfg \
      -r $USER_EXPERIMENT_DIR/output \
      -k $KEY --gpus 2

Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
Using TensorFlow backend.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

2022-03-16 12:12:24,584 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

2022-03-16 12:12:24,584 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

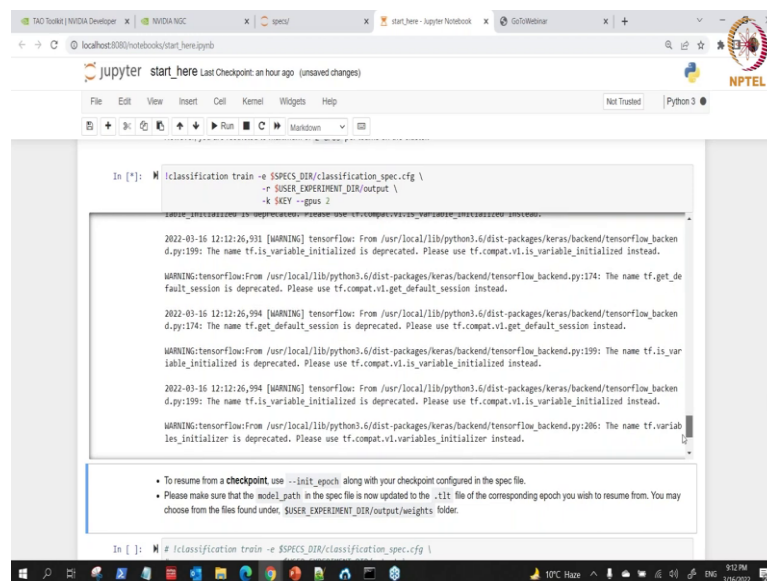
Using TensorFlow backend.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

2022-03-16 12:12:24,589 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

2022-03-16 12:12:24,589 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init_.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

(Refer Slide Time: 10:08)



```
In [*]: !classification_train -e $SPECES_DIR/classification_spec.cfg \
      -r $USER_EXPERIMENT_DIR/output \
      -k $KEY --gpus 2

2022-03-16 12:12:26,931 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend_d.py:199: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:174: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

2022-03-16 12:12:26,934 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend_d.py:174: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:199: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

2022-03-16 12:12:26,934 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend_d.py:199: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

• To resume from a checkpoint, use --init_epoch along with your checkpoint configured in the spec file.
• Please make sure that the model_path in the spec file is now updated to the .ckpt file of the corresponding epoch you wish to resume from. You may choose from the files found under $USER_EXPERIMENT_DIR/output/weights folder.

In [ ]: !classification_train -e $SPECES_DIR/classification_spec.cfg \
```

(Refer Slide Time: 10:20)

The screenshot shows a Jupyter Notebook interface with the following content:

```

les_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

2022-03-16 12:12:27,172 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

2022-03-16 12:12:27,240 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.
    
```

- To resume from a **checkpoint** use `--init_epoch` along with your checkpoint configured in the spec file.
- Please make sure that the `model_path` in the spec file is now updated to the `.tlt` file of the corresponding epoch you wish to resume from. You may choose from the files found under `$USER_EXPERIMENT_DIR/output/weights` folder.

```

In [ ]: # !classification train -e $SPECES_DIR/classification_spec.cfg \
#       -p $USER_EXPERIMENT_DIR/output \
#       -k $KEY --gpus 2 \
#       --init_epoch N
    
```

4. Evaluate trained models

In this step, we assume that the training is complete and the model from the final epoch (`reset_880.tlt`) is available. If you would like to run evaluation on an earlier model, please edit the spec file at `$SPECES_DIR/classification_spec.cfg` to point to the intended model.

```

In [ ]: # !classification evaluate -e $SPECES_DIR/classification_spec.cfg -k $KEY
    
```

(Refer Slide Time: 10:24)

The screenshot shows a Jupyter Notebook interface displaying a summary table of model layers:

block_1b_bn_shortcut (BatchNorm)	(None, 64, 56, 56)	256	block_1b_conv_shortcut[0][0]
add_2 (Add)	(None, 64, 56, 56)	0	block_1b_bn_2[0][0]
			block_1b_bn_shortcut[0][0]
block_1b_relu (Activation)	(None, 64, 56, 56)	0	add_2[0][0]
block_2a_conv_1 (Conv2D)	(None, 128, 28, 28)	73728	block_1b_relu[0][0]
block_2a_bn_1 (BatchNormalizati)	(None, 128, 28, 28)	512	block_2a_conv_1[0][0]
block_2a_relu_1 (Activation)	(None, 128, 28, 28)	0	block_2a_bn_1[0][0]
block_2a_conv_2 (Conv2D)	(None, 128, 28, 28)	147456	block_2a_relu_1[0][0]
block_2a_conv_shortcut (Conv2D)	(None, 128, 28, 28)	8192	block_1b_relu[0][0]
block_2a_bn_2 (BatchNormalizati)	(None, 128, 28, 28)	512	block_2a_conv_2[0][0]

- To resume from a **checkpoint** use `--init_epoch` along with your checkpoint configured in the spec file.
- Please make sure that the `model_path` in the spec file is now updated to the `.tlt` file of the corresponding epoch you wish to resume from. You may choose from the files found under `$USER_EXPERIMENT_DIR/output/weights` folder.

```

In [ ]: # !classification train -e $SPECES_DIR/classification_spec.cfg \
#       -p $USER_EXPERIMENT_DIR/output \
#       -k $KEY --gpus 2 \
#       --init_epoch N
    
```

4. Evaluate trained models

(Refer Slide Time: 10:27)

The screenshot shows a Jupyter Notebook interface with the following content:

block_2b_bn_shortcut (BatchNorm (None, 128, 28, 28))	512	block_2b_conv_shortcut[0][0]
add_4 (Add)	(None, 128, 28, 28)	0
		block_2b_bn_2[0][0]
		block_2b_bn_shortcut[0][0]
block_2b_relu (Activation)	(None, 128, 28, 28)	0
		add_4[0][0]
block_3a_conv_1 (Conv2D)	(None, 256, 14, 14)	294912
		block_2b_relu[0][0]
block_3a_bn_1 (BatchNormalizati	(None, 256, 14, 14)	1024
		block_3a_conv_1[0][0]
block_3a_relu_1 (Activation)	(None, 256, 14, 14)	0
		block_3a_bn_1[0][0]
block_3a_conv_2 (Conv2D)	(None, 256, 14, 14)	589824
		block_3a_relu_1[0][0]
block_3a_conv_shortcut (Conv2D)	(None, 256, 14, 14)	32768
		block_2b_relu[0][0]

Below the table, there is a code cell with the following text:

```
In [ ]: # !classification train -e $SPECES_DIR/classification_spec.cfg \  
# -r $USER_EXPERIMENT_DIR/output \  
# -k $KEY --gpus 2 \  
# --init_epoch N
```

At the bottom of the notebook, there is a section titled "4. Evaluate trained models" with the following instructions:

- To resume from a **checkpoint**, use `--init_epoch` along with your checkpoint configured in the spec file.
- Please make sure that the `model_path` in the spec file is now updated to the `.tlt` file of the corresponding epoch you wish to resume from. You may choose from the files found under `$USER_EXPERIMENT_DIR/output/weights` folder.

(Refer Slide Time: 10:30)

The screenshot shows a Jupyter Notebook interface with the following content:

```
2022-03-16 12:12:41.217 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.  
WARNING:tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.  
2022-03-16 12:12:41.327 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.  
WARNING:tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.  
2022-03-16 12:12:41.596 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.  
WARNING:tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.  
2022-03-16 12:12:41.710 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.
```

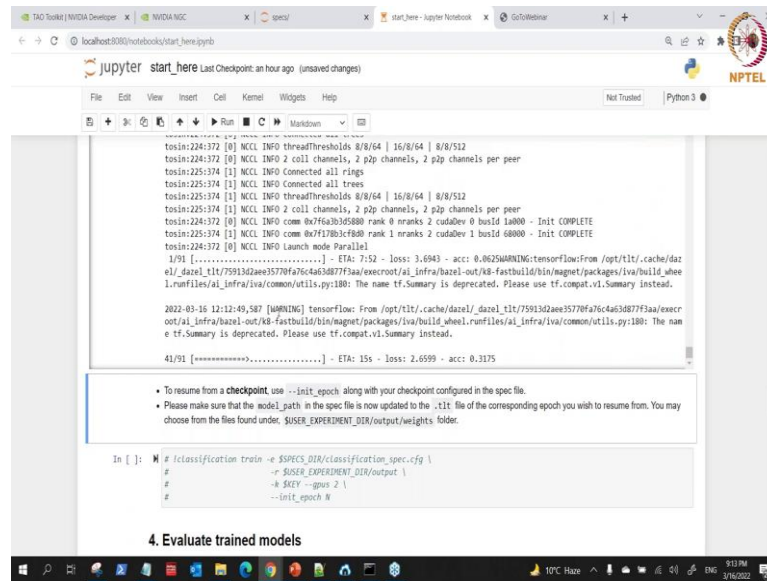
Below the warnings, there is a code cell with the following text:

```
In [ ]: # !classification train -e $SPECES_DIR/classification_spec.cfg \  
# -r $USER_EXPERIMENT_DIR/output \  
# -k $KEY --gpus 2 \  
# --init_epoch N
```

At the bottom of the notebook, there is a section titled "4. Evaluate trained models" with the following instructions:

- To resume from a **checkpoint**, use `--init_epoch` along with your checkpoint configured in the spec file.
- Please make sure that the `model_path` in the spec file is now updated to the `.tlt` file of the corresponding epoch you wish to resume from. You may choose from the files found under `$USER_EXPERIMENT_DIR/output/weights` folder.

(Refer Slide Time: 10:34)



```
tosin:224:372 [0] NCCL INFO threadThresholds 8/8/64 | 16/8/64 | 8/8/512
tosin:224:372 [0] NCCL INFO 2 coll channels, 2 p2p channels, 2 p2p channels per peer
tosin:225:374 [1] NCCL INFO Connected all rings
tosin:225:374 [1] NCCL INFO Connected all trees
tosin:225:374 [1] NCCL INFO threadThresholds 8/8/64 | 16/8/64 | 8/8/512
tosin:225:374 [1] NCCL INFO 2 coll channels, 2 p2p channels, 2 p2p channels per peer
tosin:224:372 [0] NCCL INFO comm 0x7f6a3b3d5680 rank 0 nRanks 2 cudaDev 0 busId 1a000 - Init COMPLETE
tosin:225:374 [1] NCCL INFO comm 0x7f178b3cf840 rank 1 nRanks 2 cudaDev 1 busId 68000 - Init COMPLETE
tosin:224:372 [0] NCCL INFO Launch mode Parallel
3/91 [.....] - ETA: 7:52 - loss: 3.6943 - acc: 0.0625WARNING:tensorflow:From /opt/tit/.cache/dazel/dazel_tit/75913d2aee35770fa76c4a63d877f3aa/executor_ai_infra/bazel-out/x86-fastbuild/bin/magnet/packages/iva/build_0ee1.runfiles/ai_infra/iva/common/util.py:180: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.
2022-09-18 12:12:49.587 [WARNING] tensorflow: From /opt/tit/.cache/dazel/dazel_tit/75913d2aee35770fa76c4a63d877f3aa/executor_ai_infra/bazel-out/x86-fastbuild/bin/magnet/packages/iva/build_0ee1.runfiles/ai_infra/iva/common/util.py:180: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.
41/91 [*****>.....] - ETA: 15s - loss: 2.6599 - acc: 0.3375
```

- To resume from a **checkpoint**, use `--init_epoch` along with your checkpoint configured in the spec file.
- Please make sure that the `model_path` in the spec file is now updated to the `.tit` file of the corresponding epoch you wish to resume from. You may choose from the files found under: `$USER_EXPERIMENT_DIR/output/weights` folder.

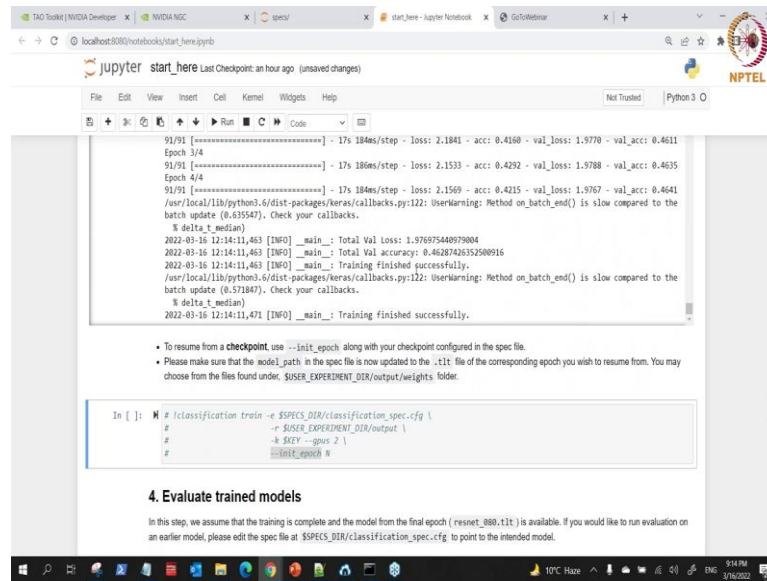
```
In [ ]: # !classification train -e $SPECS_DIR/classification_spec.cfg \
# -r $USER_EXPERIMENT_DIR/output \
# -k $KEY --gpus 2 \
# --init_epoch N
```

4. Evaluate trained models

So, it is running now, its running and see it is running. So, it is running. So, ok can see. So, this are the is loading the what the layers of the model here, exactly, loads them and then it will continue to the epoch. So, you can see here to start running the initial epoch. So, its run them epoch by epoch. So, we are on the second epoch now, so it do not take us more than 2 minutes to do to complete everything then.

So, the next what you can see here why that one is training. The next phase here is I have commented this one out, this one is meant for maybe your trained and you are unable to complete your training, you just save it somewhere. So, by the time you want to start your training again, instead of you starting from the beginning, you can continue from where it stops.

(Refer Slide Time: 11:36)



The screenshot shows a Jupyter Notebook interface with a terminal output window displaying training progress. The output shows training for 4 epochs, with the final epoch (Epoch 4/4) showing a loss of 2.1569 and an accuracy of 0.4215. Below the terminal output, there is a code cell with the following content:

```
In [ ]: # !classification_train -e $SPECES_DIR/classification_spec.cfg \
# -r $USER_EXPERIMENT_DIR/output \
# -h $KEY --gpus 2 \
# --init_epoch N
```

Below the code cell, there is a section titled "4. Evaluate trained models" with the following text:

In this step, we assume that the training is complete and the model from the final epoch (resnet_880.tlt) is available. If you would like to run evaluation on an earlier model, please edit the spec file at \$SPECES_DIR/classification_spec.cfg to point to the intended model.

So, you can continue from where it stop by word, specifying the word initial epoch. So, let us say we stop at 4 epoch now. If I decide to continue this training tomorrow I do not need to start from 1 epoch again, I will just continue from the where do I stop 4 epochs. But will not be needing that for now, for this demo.

So, we are, it is almost done here, it is almost done. So, you see it is how do you know it is almost done? When this is it the star. So, now, when it gives it the number everything is done that cell has completed. So, it has finished successfully. We can see this is the loss and this is the accuracy 0.4, but you know because we trained for 4 epochs. So, we are not expecting the accuracy to be that good so, but that is just for a demo, it is just for a demo. If you train with your 80 epoch the accuracy would be fantastic.

(Refer Slide Time: 12:21)

```
In [*]: !classification evaluate -e $SPECES_DIR/classification_spec.cfg -k $KEY

Using TensorFlow backend.
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
WARNING:tensorflow:From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:28: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

2022-03-16 12:15:08,081 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:28: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:30: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

2022-03-16 12:15:08,082 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:30: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.set_verbosity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

2022-03-16 12:15:08,350 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.set_verbosity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.
```

(Refer Slide Time: 12:47)

```
The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.set_verbosity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

2022-03-16 12:15:08,350 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.set_verbosity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

WARNING:tensorflow:From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.INFO is deprecated. Please use tf.compat.v1.logging.INFO instead.

2022-03-16 12:15:08,350 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2ae35770fa76c4a63d877f3aa/executor/ai_infra/bazel-out/k8-fastbuild/bin/magnet/packages/iva/build_wheel.runfiles/ai_infra/iva/makernet/scripts/evaluate.py:77: The name tf.logging.INFO is deprecated. Please use tf.compat.v1.logging.INFO instead.

2022-03-16 12:15:08,350 [INFO] _main_: Loading experiment spec at /workspace/specs/classification_spec.cfg.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

2022-03-16 12:15:09,227 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

2022-03-16 12:15:09,246 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:245: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

Then we need to evaluate what we have trained. So, how do we evaluate? You have what this to evaluate, then classification, then evaluate you have you specified the word these configuration file which is the specification file and the key also this is what we specified then.

Let me around that is going to do the evaluation. So, during the evaluation if we plot the confusion matrix and also give us the precision and also the recall and all that everything will be given so, it is almost done.

(Refer Slide Time: 13:11)

4. Evaluate trained models

In this step, we assume that the training is complete and the model from the final epoch (reset_880.tlt) is available. If you would like to run evaluation on an earlier model, please edit the spec file at \$SPECS_DIR/classification_spec.cfg to point to the intended model.

```
In [*]: !classification evaluate --SPECS_DIR/classification_spec.cfg -k SKEY
```

block_4b_relu (Activation)	(None, 512, 14, 14)	0	add_8[0][0]
avg_pool (AveragePooling2D)	(None, 512, 1, 1)	0	block_4b_relu[0][0]
flatten (Flatten)	(None, 512)	0	avg_pool[0][0]
predictions (Dense)	(None, 20)	10260	flatten[0][0]

 Total params: 11,552,724
 Trainable params: 11,376,820
 Non-trainable params: 176,704

 Found 3345 images belonging to 20 classes.
 2022-03-16 12:15:14,061 [INFO] _main_: Processing dataset (evaluation): /workspace/data/test
 Evaluation Loss: 1.9528126450254957
 Evaluation Top K accuracy: 0.7264573991209581
 Found 3345 images belonging to 20 classes.
 2022-03-16 12:15:22,301 [INFO] _main_: Calculating per-class P/R and confusion matrix. It may take a while...

5. Prune trained models

- Specify pre-trained model

(Refer Slide Time: 13:22)

```
In [*]: !classification prune --SPECS_DIR/classification_spec.cfg -k SKEY
```

```

0 0]
[ 1 36 2 0 0 0 0 0 0 0 3 0 0 2 0 2 56 0 0 0
1 0]
[ 12 0 67 12 0 0 0 1 5 1 1 0 17 1 0 32 0 2 0
0 2]
[ 4 0 0 43 1 0 8 0 0 0 0 1 1 0 1 40 0 0 0
3 0]
[ 2 2 1 0 10 0 5 2 17 0 2 1 0 1 85 0 0 0
1 13]
[ 1 0 0 3 0 14 52 0 0 0 0 0 0 0 0 11 0 0 0
4 0]
[ 15 2 1 6 0 3 107 0 4 1 1 2 2 10 71 0 0 0
6 2]
[ 0 0 1 0 1 0 2 159 9 1 0 21 0 0 20 0 0 1
0 1]
[ 1 2 2 1 2 0 1 6 83 0 1 7 0 0 88 4 0 4
1 21]
[ 5 0 5 1 0 0 0 1 0 2 0 15 10 0 22 0 0 0
0 0]
[ 0 0 1 0 4 0 0 2 51 0 4 1 0 0 38 1 0 0

```

5. Prune trained models

- Specify pre-trained model
- Equalization criterion
- Threshold for pruning
- Exclude prediction layer that you don't want pruned (e.g. predictions)

Usually, you just need to adjust --pth (threshold) for accuracy and model size trade off. Higher pth gives you smaller model (and thus higher inference speed) but worse accuracy. The threshold to use is depend on the dataset. A pth value 0.68 is just a starting point. If the retrain accuracy is good, you can increase this value to get smaller models. Otherwise, lower this value to get better accuracy.

(Refer Slide Time: 13:22)

The screenshot shows a Jupyter Notebook interface with the following content:

```

trainable params: 11,796,928
Non-trainable params: 176,704

Found 3345 images belonging to 20 classes.
2022-03-16 12:15:14,051 [INFO] _main_: Processing dataset (evaluation): /workspace/data/test
Evaluation Loss: 1.9928126450254957
Evaluation Top K accuracy: 0.7264573991209581
Found 3345 images belonging to 20 classes.
2022-03-16 12:15:22,381 [INFO] _main_: Calculating per-class P/R and confusion matrix. It may take a while...
Confusion Matrix
[[107  0  0  1  0  0  11  0  0  0  0  0  0  0  0  15  0  0  0
  0  0]
 [ 1 36  2  0  0  0  8  0  3  0  0  2  0  2  56  0  0  0
  1  0]
 [ 12  0  67 12  0  0  1  5  1  1  0 17  1  0 32  0  2  0
  0  2]
 [  4  0  0 43  1  0  8  0  0  0  1  1  0  1 40  0  0  0
  3  0]
 [  2  2  1  0 10  0  5  2 17  0  2  1  0  1 85  0  0  0
  0  0]]

```

5. Prune trained models

- Specify pre-trained model
- Equalization criterion
- Threshold for pruning
- Exclude prediction layer that you don't want pruned (e.g. predictions)

Usually, you just need to adjust `--pth` (threshold) for accuracy and model size trade off. Higher `pth` gives you smaller model (and thus higher inference speed) but worse accuracy. The threshold to use is depend on the dataset. A `pth` value 0.68 is just a starting point. If the retain accuracy is good, you can increase this value to get smaller models. Otherwise, lower this value to get better accuracy.

(Refer Slide Time: 13:26)

The screenshot shows a Jupyter Notebook interface with the following content:

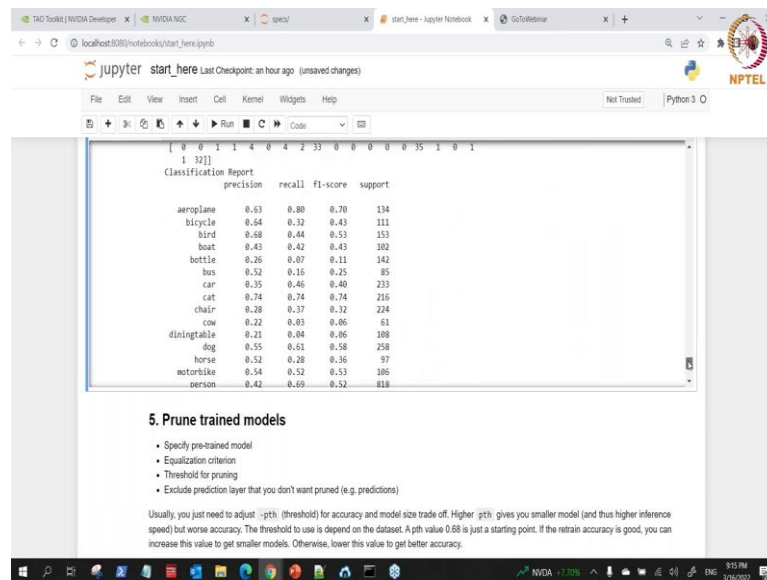
bottle	0.26	0.87	0.11	142
bus	0.52	0.16	0.25	85
car	0.35	0.46	0.40	233
cat	0.74	0.74	0.74	216
chair	0.28	0.37	0.32	224
cow	0.22	0.83	0.06	61
diningtable	0.21	0.04	0.06	108
dog	0.55	0.61	0.58	258
horse	0.52	0.28	0.36	97
motorbike	0.54	0.52	0.53	106
person	0.42	0.69	0.52	818
potteplant	0.89	0.01	0.02	106
sheep	0.80	0.00	0.00	65
sofa	0.23	0.83	0.05	102
train	0.63	0.57	0.60	109
tvmonitor	0.31	0.28	0.29	115
accuracy		0.46		3345
macro avg	0.41	0.34	0.35	3345
weighted avg	0.44	0.42	0.43	3345

5. Prune trained models

- Specify pre-trained model
- Equalization criterion
- Threshold for pruning
- Exclude prediction layer that you don't want pruned (e.g. predictions)

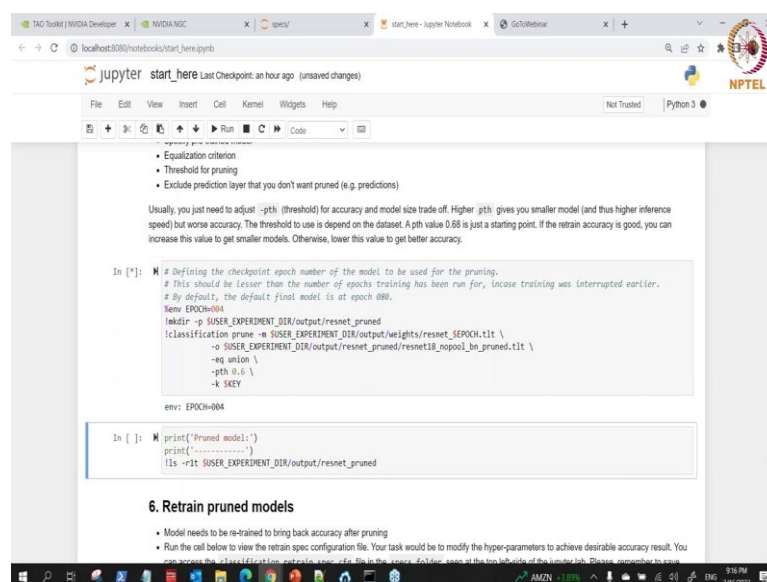
Usually, you just need to adjust `--pth` (threshold) for accuracy and model size trade off. Higher `pth` gives you smaller model (and thus higher inference speed) but worse accuracy. The threshold to use is depend on the dataset. A `pth` value 0.68 is just a starting point. If the retain accuracy is good, you can increase this value to get smaller models. Otherwise, lower this value to get better accuracy.

(Refer Slide Time: 13:28)



So, it is trying to what, calculate the precision. And we see all that and once it is done you see it will give it a number here. So, yeah, this is the see the confusion matrix, then it will then generate go to classification reports. You see the classification report here, precision, recall, f1 score, support and all that. So, I think it is done now, yeah. So, we can proceed for that.

(Refer Slide Time: 13:44)



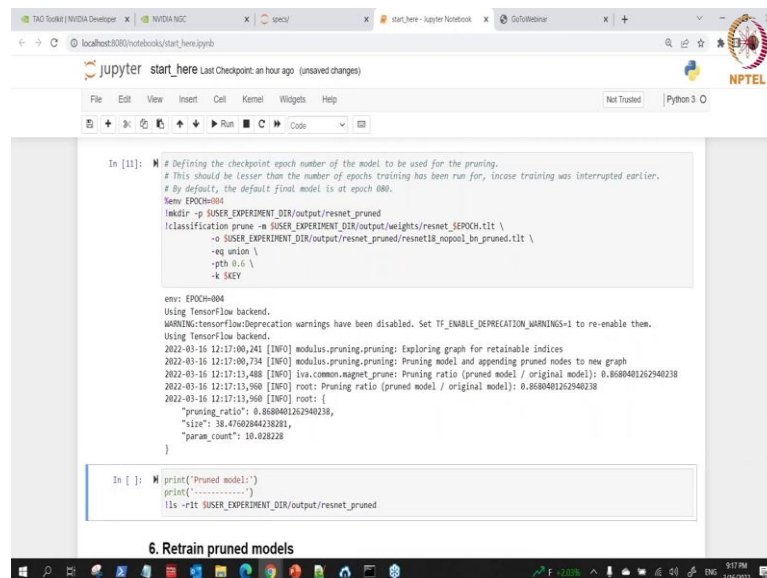
And then, so we can now what prune, yeah, this is where you prune. So, for you to prune we need to create a folder where we store our pruning, create a directory where we start

our pruning. This is the epoch the default epoch is 80. So, because I am using 4 epoch, I am going to swap these to 4. And then, so how do you prune? You have your classification and the keyword prune, yeah.

So, you supply where is the model that was trained. This is where the directory it is. So, where do you want to store when you prune? The prune you will store it here in this directory output. And what percentage do you want to use to prune? We just want to prune 60 percent, this means 60 percent.

If you specify 0.2, that means, 20 percent of the model will be prune. So, we can run these now and it is going to do the pruning.

(Refer Slide Time: 14:34)



```
In [11]: # Defining the checkpoint epoch number of the model to be used for the pruning.
# This should be Lesser than the number of epochs training has been run for, incase training was interrupted earlier.
# By default, the default final model is at epoch 800.
Newer EPOCH=804
mkdir -p $USER_EXPERIMENT_DIR/output/resnet_pruned
./classification_prune -n $USER_EXPERIMENT_DIR/output/weights/resnet_SEPOCH.tilt \
-o $USER_EXPERIMENT_DIR/output/resnet_pruned/resnet18_nopool_bn_pruned.tilt \
-eq union \
-ptb 0.6 \
-k SKEY

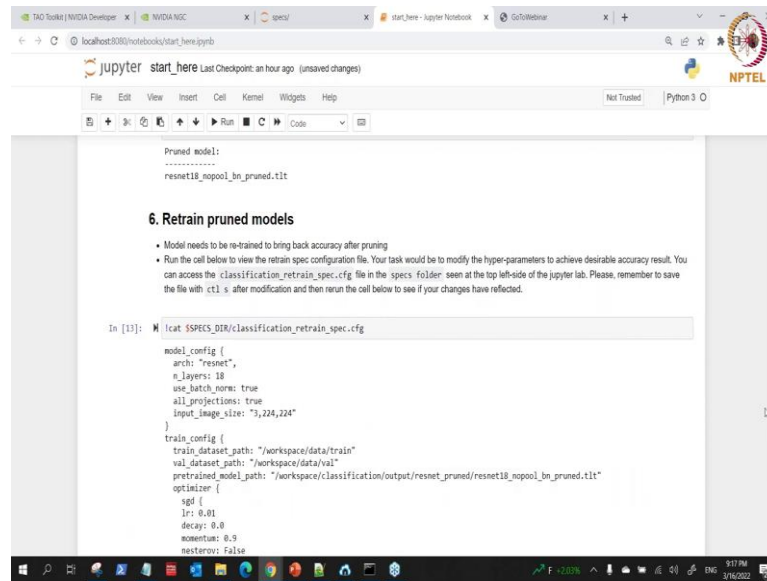
env: EPOCH=804
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
Using TensorFlow backend.
2022-03-16 12:17:00,241 [INFO] modulus.pruning.pruning: Exploring graph for retainable indices
2022-03-16 12:17:00,734 [INFO] modulus.pruning.pruning: Pruning model and appending pruned nodes to new graph
2022-03-16 12:17:13,488 [INFO] iya.common.magnet_prune: Pruning ratio (pruned model / original model): 0.8680481262940238
2022-03-16 12:17:13,968 [INFO] root: Pruning ratio (pruned model / original model): 0.8680481262940238
2022-03-16 12:17:13,968 [INFO] root: {
  "pruning_ratio": 0.8680481262940238,
  "size": 38.47602844238281,
  "param_count": 18.028228
}

In [ ]: print('Pruned model:')
print('-----')
ls -l $USER_EXPERIMENT_DIR/output/resnet_pruned
```

6. Retrain pruned models

So, why that is going on? So, after the pruning is completed then we can view the file here. We will be able to feel, ok that is done. If that is given it number 11, then we can now see the pruning, yes, ok. This is what it saves. This is the name of what it has saved.

(Refer Slide Time: 15:01)



The screenshot shows a Jupyter Notebook interface with a code cell containing a configuration file for retraining a model. The code is as follows:

```
Pruned model:
-----
resnet18_nopool_bn_pruned.t1t

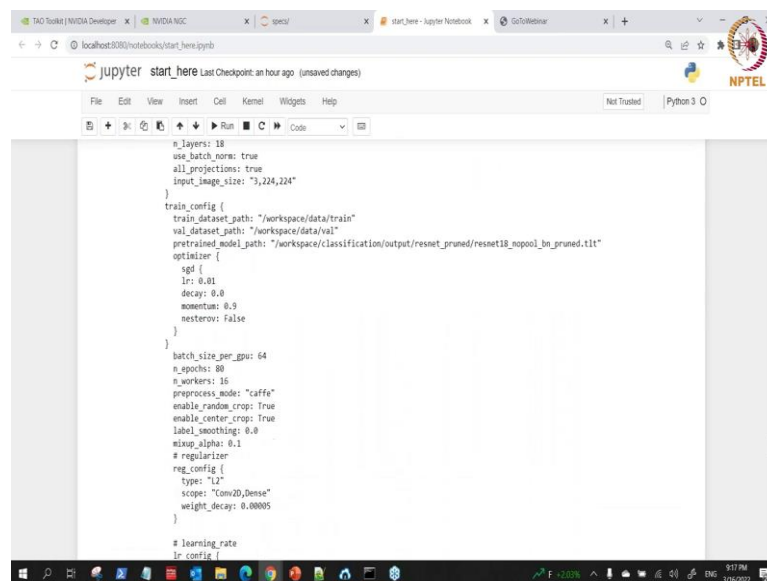
6. Retrain pruned models

• Model needs to be re-trained to bring back accuracy after pruning
• Run the cell below to view the retrain spec configuration file. Your task would be to modify the hyper-parameters to achieve desirable accuracy result. You can access the classification_retrain_spec.cfg file in the specs folder seen at the top left-side of the jupyter lab. Please, remember to save the file with .t1t after modification and then rerun the cell below to see if your changes have reflected.

In [13]: !cat $SPECES_DIR/classification_retrain_spec.cfg

mode_config {
  arch: "resnet",
  n_layers: 18
  use_batch_norm: true
  all_projections: true
  input_image_size: "3,224,224"
}
train_config {
  train_dataset_path: "/workspace/data/train"
  val_dataset_path: "/workspace/data/val"
  pretrained_model_path: "/workspace/classification/output/resnet_pruned/resnet18_nopool_bn_pruned.t1t"
  optimizer {
    sgd {
      lr: 0.01
      decay: 0.0
      momentum: 0.9
      nesterov: false
    }
  }
  batch_size_per_gpu: 64
  n_epochs: 80
  n_workers: 15
  preprocess_mode: "caffe"
  enable_random_crop: true
  enable_center_crop: true
  label_smoothing: 0.0
  dropout_alpha: 0.1
  # regularizer
  reg_config {
    type: "L2"
    scopes: "Conv2D,Dense"
    weight_decay: 0.00005
  }
  # learning_rate
  lr_config {
```

(Refer Slide Time: 15:26)

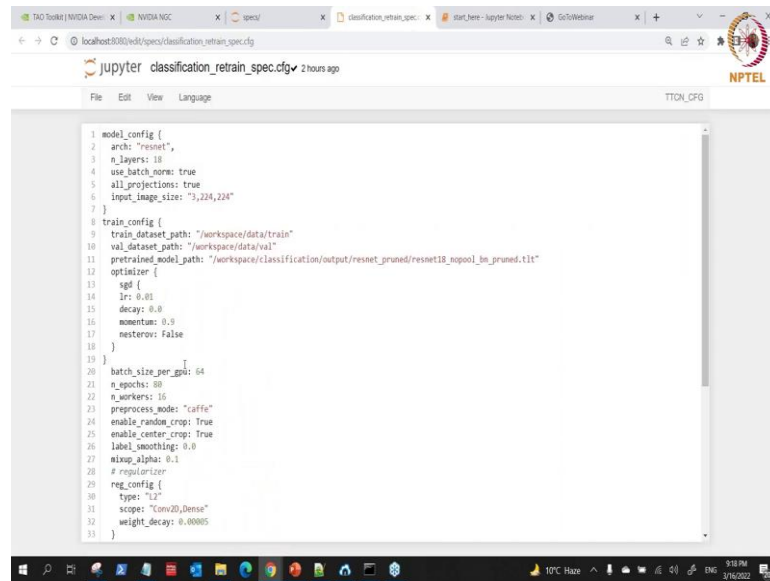


The screenshot shows a Jupyter Notebook interface with a code cell containing a configuration file for retraining a model. The code is as follows:

```
n_layers: 18
use_batch_norm: true
all_projections: true
input_image_size: "3,224,224"
}
train_config {
  train_dataset_path: "/workspace/data/train"
  val_dataset_path: "/workspace/data/val"
  pretrained_model_path: "/workspace/classification/output/resnet_pruned/resnet18_nopool_bn_pruned.t1t"
  optimizer {
    sgd {
      lr: 0.01
      decay: 0.0
      momentum: 0.9
      nesterov: false
    }
  }
  batch_size_per_gpu: 64
  n_epochs: 80
  n_workers: 15
  preprocess_mode: "caffe"
  enable_random_crop: true
  enable_center_crop: true
  label_smoothing: 0.0
  dropout_alpha: 0.1
  # regularizer
  reg_config {
    type: "L2"
    scopes: "Conv2D,Dense"
    weight_decay: 0.00005
  }
  # learning_rate
  lr_config {
```

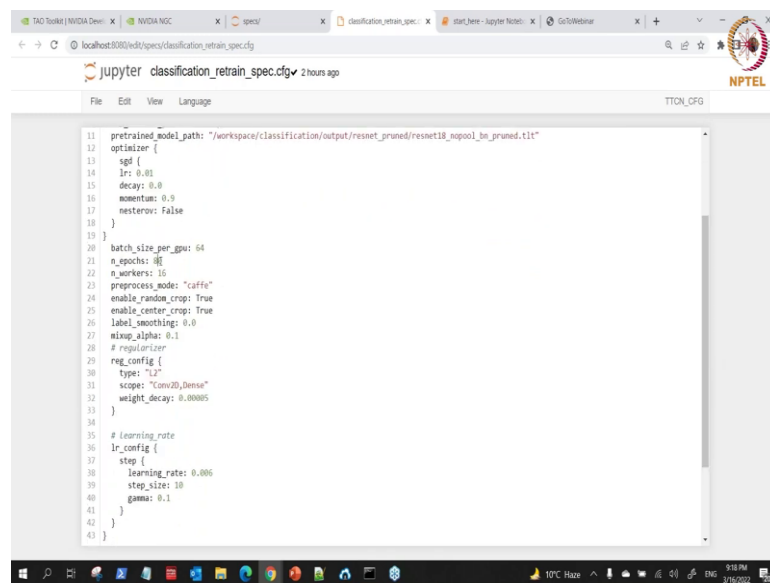
Then so, because we have pruned there is need for us to retrain again. So, when we want to retrain we have to use the word the specification file which is for work for retrained.

(Refer Slide Time: 15:35)



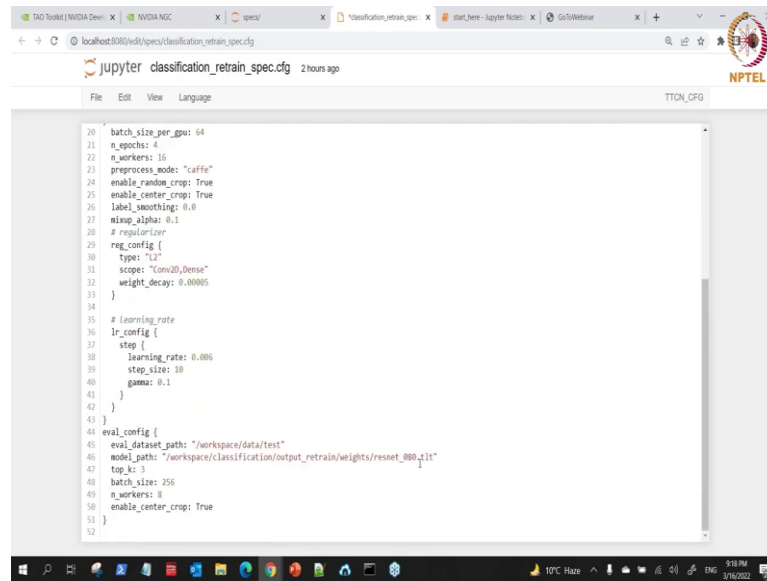
```
1 model_config {
2   arch: "resnet",
3   n_layers: 18
4   use_batch_norm: true
5   all_projections: true
6   input_image_size: "3,224,224"
7 }
8 train_config {
9   train_dataset_path: "/workspace/data/train"
10  val_dataset_path: "/workspace/data/val"
11  pretrained_model_path: "/workspace/classification/output/resnet_pruned/resnet18_nopool_bn_pruned.t1t"
12  optimizer {
13    sgd {
14      lr: 0.01
15      decay: 0.0
16      momentum: 0.9
17      nesterov: false
18    }
19  }
20  batch_size_per_gpu: 64
21  n_epochs: 10
22  n_workers: 16
23  preprocess_mode: "caffe"
24  enable_random_crop: true
25  enable_center_crop: true
26  label_smoothing: 0.0
27  mixup_alpha: 0.1
28  # regularizer
29  reg_config {
30    type: "L2"
31    scope: "Conv2D,Dense"
32    weight_decay: 0.00005
33  }
```

(Refer Slide Time: 15:41)



```
11 pretrained_model_path: "/workspace/classification/output/resnet_pruned/resnet18_nopool_bn_pruned.t1t"
12 optimizer {
13   sgd {
14     lr: 0.01
15     decay: 0.0
16     momentum: 0.9
17     nesterov: false
18   }
19 }
20 batch_size_per_gpu: 64
21 n_epochs: 10
22 n_workers: 16
23 preprocess_mode: "caffe"
24 enable_random_crop: true
25 enable_center_crop: true
26 label_smoothing: 0.0
27 mixup_alpha: 0.1
28 # regularizer
29 reg_config {
30   type: "L2"
31   scope: "Conv2D,Dense"
32   weight_decay: 0.00005
33 }
34
35 # learning_rate
36 lr_config {
37   step {
38     learning_rate: 0.006
39     step_size: 10
40     gamma: 0.1
41   }
42 }
43 }
```

(Refer Slide Time: 15:50)

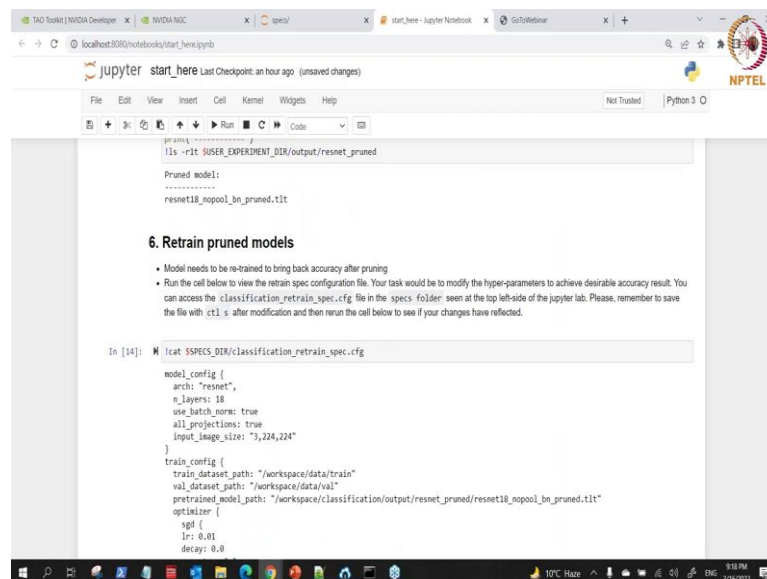


```
20 batch_size_per_gpu: 64
21 n_epochs: 4
22 n_workers: 16
23 preprocess_mode: "caffe"
24 enable_random_crop: True
25 enable_center_crop: True
26 label_smoothing: 0.0
27 mixup_alpha: 0.1
28 # regularizer
29 reg_config {
30   type: "L2"
31   scope: "Conv2D,Dense"
32   weight_decay: 0.00005
33 }
34
35 # learning rate
36 lr_config {
37   step {
38     learning_rate: 0.006
39     step_size: 10
40     gamma: 0.1
41   }
42 }
43
44 eval_config {
45   eval_dataset_path: "/workspace/data/test"
46   model_path: "/workspace/classification/output_retrain/weights/resnet_080.t1t"
47   top_k: 3
48   batch_size: 256
49   n_workers: 8
50   enable_center_crop: True
51 }
52
```

And then so, let us cut this and see what is inside. This is what is inside is the same thing as for the train, but the parts that are different is the retrain path and the evaluation path.

Also, here we cannot use 80 because we cannot wait for long. So, I would just I will come here again and look for, ok this is the retrain and change the number of epoch to change the number of epoch to 4. So, that we can just change with that and then the evaluation here, I would change that 1, ok as well. So, I can save, come to save, then close that, come back to here.

(Refer Slide Time: 16:05)



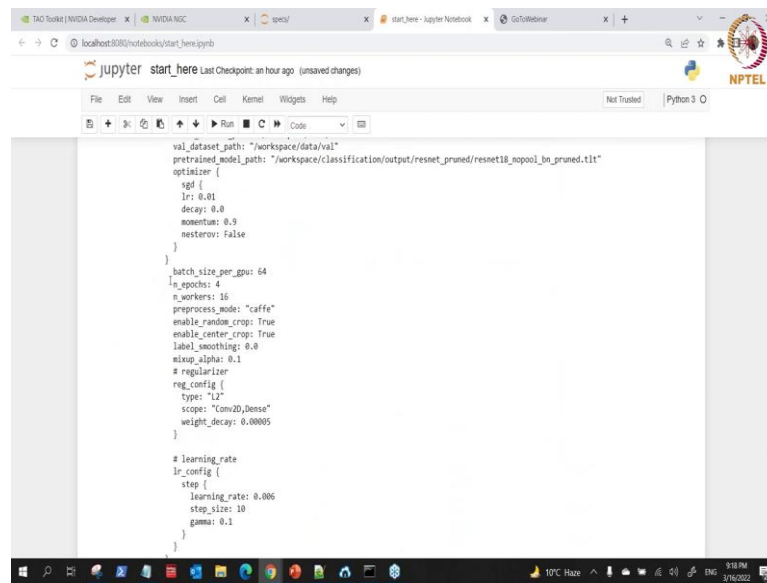
```
pruned_model:
-----
lls-vr1t: SUSER_EXPERIMENT_DIR/output/resnet_pruned

6. Retrain pruned models
• Model needs to be re-trained to bring back accuracy after pruning
• Run the cell below to view the retrain spec configuration file. Your task would be to modify the hyper-parameters to achieve desirable accuracy result. You can access the classification_retrain_spec.cfg file in the specs/ folder seen at the top left-side of the jupyter lab. Please, remember to save the file with ctrl s after modification and then rerun the cell below to see if your changes have reflected.

In [14]: !cat $SPECES_DIR/classification_retrain_spec.cfg

model_config {
  arch: "resnet",
  n_layers: 18
  use_batch_norm: true
  all_projections: true
  input_image_size: "3,224,224"
}
train_config {
  train_dataset_path: "/workspace/data/train"
  val_dataset_path: "/workspace/data/val"
  pretrained_model_path: "/workspace/classification/output/resnet_pruned/resnet18_nopool_bn_pruned.t1t"
  optimizer {
    sgd {
      lr: 0.01
      decay: 0.0
    }
  }
}
```

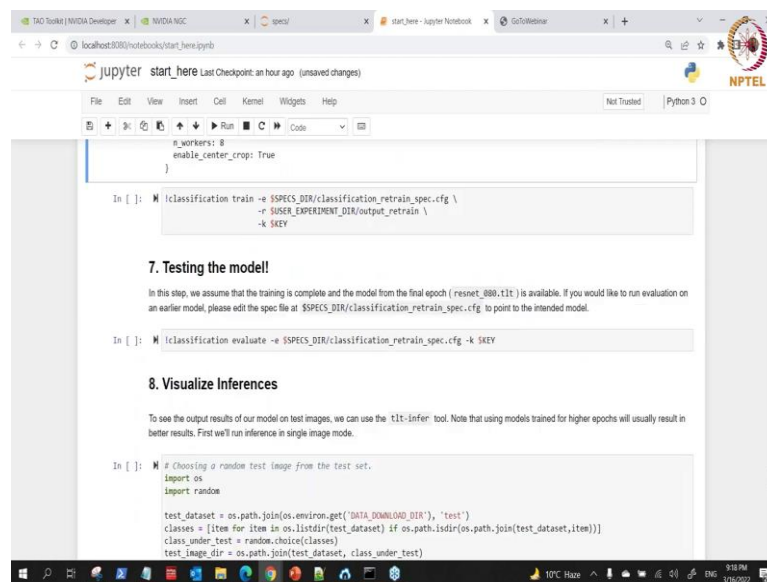
(Refer Slide Time: 16:08)



```
val_dataset_path: "/workspace/data/val"
pretrained_model_path: "/workspace/classification/output/reset_pruned/reset18_nopool_bn_pruned.tlt"
optimizer {
  sgd {
    lr: 0.01
    decay: 0.0
    momentum: 0.9
    nesterov: False
  }
}
batch_size_per_gpu: 64
n_epochs: 4
n_workers: 16
preprocess_mode: "caffe"
enable_random_crop: True
enable_center_crop: True
label_smoothing: 0.0
mixup_alpha: 0.1
# regularizer
rng_config {
  type: "L2"
  scope: "Conv2D,Dense"
  weight_decay: 0.00005
}
# learning_rate
lr_config {
  step {
    learning_rate: 0.005
    step_size: 10
    gamma: 0.1
  }
}
```

Then, I need to do is to let me check here whether that has been affected I think. So, yes it has been affected, here you can see, yeah.

(Refer Slide Time: 16:13)



```
n_workers: 8
enable_center_crop: True
}

In [ ]: !classification train -e $SPECES_DIR/classification_retrain_spec.cfg \
-r $USER_EXPERIMENT_DIR/output_retrain \
-k $KEY

7. Testing the model!
In this step, we assume that the training is complete and the model from the final epoch ( reset_880.tlt ) is available. If you would like to run evaluation on an earlier model, please edit the spec file at $SPECES_DIR/classification_retrain_spec.cfg to point to the intended model.

In [ ]: !classification evaluate -e $SPECES_DIR/classification_retrain_spec.cfg -k $KEY

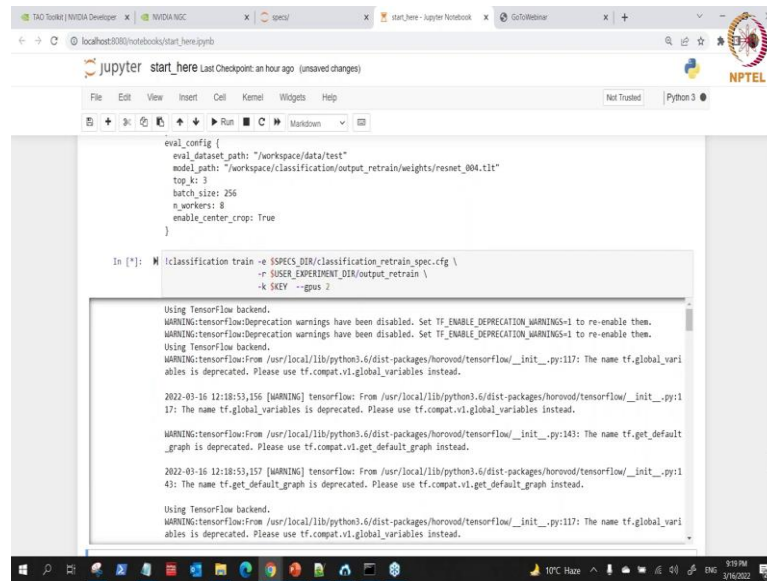
8. Visualize Inferences
To see the output results of our model on test images, we can use the 'tlt-infer' tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

In [ ]: # Choosing a random test image from the test set.
import os
import random

test_dataset = os.path.join(os.environ.get("DATA_DOWNLOAD_DIR"), "test")
classes = [item for item in os.listdir(test_dataset) if os.path.isdir(os.path.join(test_dataset,item))]
class_under_test = random.choice(classes)
test_image_dir = os.path.join(test_dataset, class_under_test)
```

Then, we can proceed further here to retrain. So, to retrain here I would like to retrain using two GPU as well gpus 2. So, that can be faster than do that we are almost done with this. So, you can see this making things effortless.

(Refer Slide Time: 16:35)

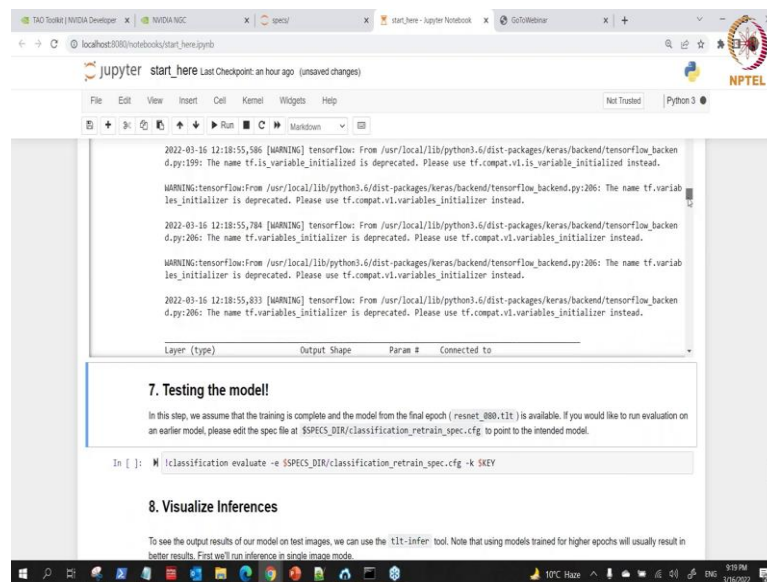


```
eval_config {
  eval_dataset_path: "/workspace/data/test"
  model_path: "/workspace/classification/output_retrain/weights/reset_004.tit"
  top_k: 3
  batch_size: 256
  num_workers: 8
  enable_center_crop: True
}
```

```
In [*]: !classification train -e $SPECS_DIR/classification_retrain_spec.cfg \
      -r $USER_EXPERIMENT_DIR/output_retrain \
      -k $KEY --gpus 2
```

I must say because we are using multi GPU here, if you have to write it from the scratch it is going to be a lot of effort. Now, we are not writing any special code, yeah. And we are training our model.

(Refer Slide Time: 16:55)



```
2022-03-16 12:18:53,156 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init__.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
2022-03-16 12:18:53,157 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init__.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
2022-03-16 12:18:53,157 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init__.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
2022-03-16 12:18:53,157 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init__.py:143: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
2022-03-16 12:18:53,157 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/horovod/tensorflow/_init__.py:117: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
```

Layer (type)	Output Shape	Param #	Connected to
--------------	--------------	---------	--------------

7. Testing the model!

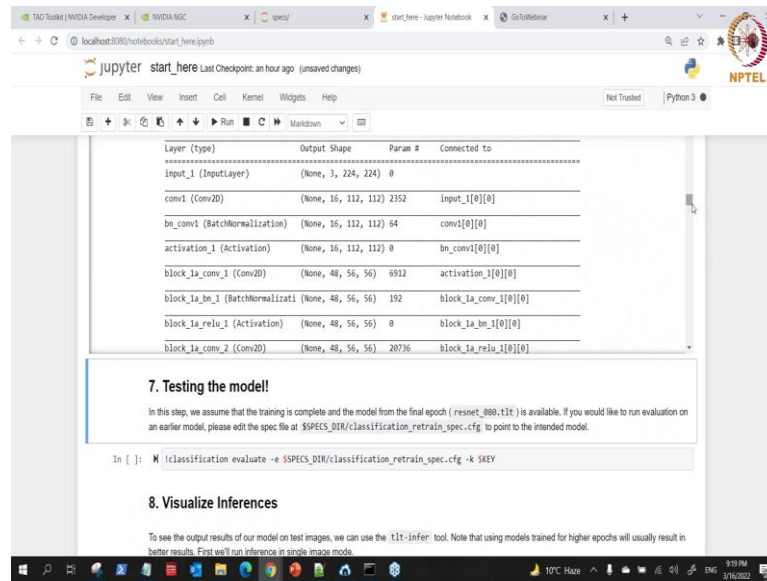
In this step, we assume that the training is complete and the model from the final epoch (reset_000.tit) is available. If you would like to run evaluation on an earlier model, please edit the spec file at \$SPECS_DIR/classification_retrain_spec.cfg to point to the intended model.

```
In [ ]: !classification evaluate -e $SPECS_DIR/classification_retrain_spec.cfg -k $KEY
```

8. Visualize Inferences

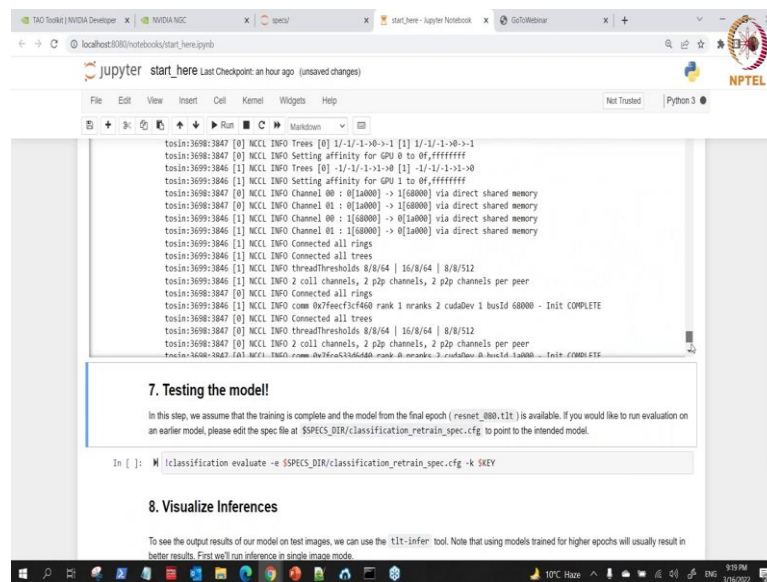
To see the output results of our model on test images, we can use the 'tit-infer' tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

(Refer Slide Time: 16:54)



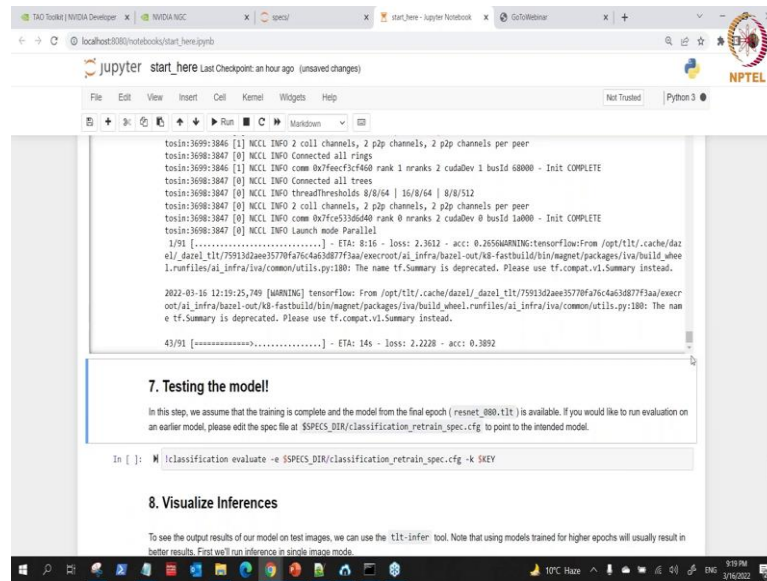
We are not writing any special code. What we only did was to just bring our raw data and do some configuration with the specification file. So, it is loading the model already there.

(Refer Slide Time: 17:05)



So, then it commence the training. So, you can see how it is been trained, yeah.

(Refer Slide Time: 17:11)



```
tosin:3699:3846 [1] NCCL INFO 2 coll channels, 2 p2p channels, 2 p2p channels per peer
tosin:3698:3847 [0] NCCL INFO Connected all rings
tosin:3699:3846 [1] NCCL INFO comm 0x7fecf3c460 rank 1 nrank 2 cudaDev 1 busId 68000 - Init COMPLETE
tosin:3698:3847 [0] NCCL INFO Connected all trees
tosin:3698:3847 [0] NCCL INFO threadThresholds: 0/0/64 | 16/0/64 | 0/0/512
tosin:3698:3847 [0] NCCL INFO 2 coll channels, 2 p2p channels, 2 p2p channels per peer
tosin:3698:3847 [0] NCCL INFO comm 0x7fec5336d40 rank 0 nrank 2 cudaDev 0 busId 1a000 - Init COMPLETE
tosin:3698:3847 [0] NCCL INFO Launch mode Parallel
1/91 [.....] - ETA: 8:16 - loss: 2.3612 - acc: 0.2650WARNING:tensorflow:From /opt/tlt/.cache/daz
ai/_dazel_tlt/75913d2aee35778fa764a53877f3aa/executor/ai_infra/dazel-out/K8-fastbuild/bin/magnet/packages/iva/build_whee
1.runfiles/ai_infra/iva/conda/utlis.py:180: The name tf.compat.v1.Summary is deprecated. Please use tf.compat.v1.Summary instead.
2022-03-16 12:19:25,749 [WARNING] tensorflow: From /opt/tlt/.cache/dazel/_dazel_tlt/75913d2aee35778fa764a53877f3aa/executor
oot/ai_infra/bazel-out/K8-fastbuild/bin/magnet/packages/iva/build_whee1.runfiles/ai_infra/iva/conda/utlis.py:180: The nam
e tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.
43/91 [=====] - ETA: 14s - loss: 2.2228 - acc: 0.3892
```

7. Testing the model!

In this step, we assume that the training is complete and the model from the final epoch (`reset_880.tlt`) is available. If you would like to run evaluation on an earlier model, please edit the spec file at `$$SPEC_DIR/classification_retrain_spec.cfg` to point to the intended model.

```
In [ ]: !classification evaluate -e $$SPEC_DIR/classification_retrain_spec.cfg -k SKEY
```

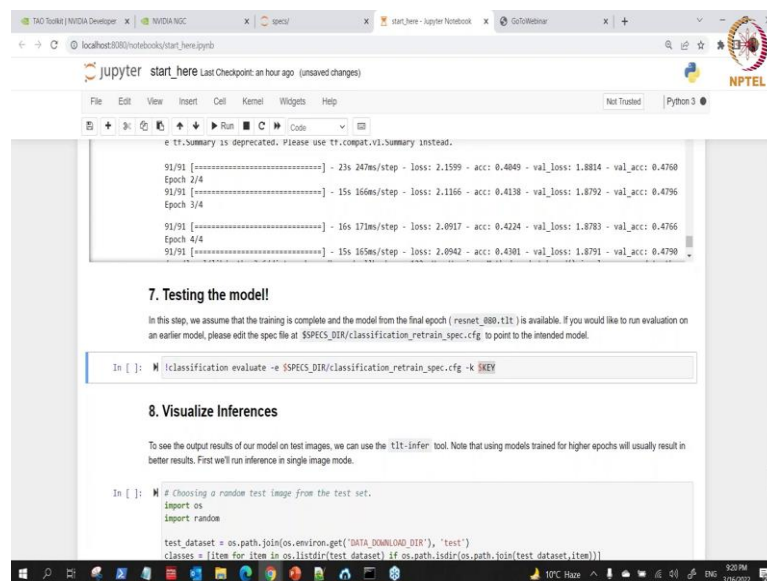
8. Visualize Inferences

To see the output results of our model on test images, we can use the `tlt-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

```
In [ ]: !# Choosing a random test image from the test set.
import os
import random

test_dataset = os.path.join(os.environ.get('DATA_DOWNLOAD_DIR'), 'test')
classes = [item for item in os.listdir(test_dataset) if os.path.isdir(os.path.join(test_dataset, item))]
```

(Refer Slide Time: 17:14)



```
e tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.
91/91 [=====] - 23s 247ms/step - loss: 2.1599 - acc: 0.4049 - val_loss: 1.8814 - val_acc: 0.4760
Epoch 2/4
91/91 [=====] - 15s 166ms/step - loss: 2.1166 - acc: 0.4138 - val_loss: 1.8792 - val_acc: 0.4796
Epoch 3/4
91/91 [=====] - 16s 171ms/step - loss: 2.0917 - acc: 0.4224 - val_loss: 1.8783 - val_acc: 0.4766
Epoch 4/4
91/91 [=====] - 15s 165ms/step - loss: 2.0942 - acc: 0.4301 - val_loss: 1.8791 - val_acc: 0.4790
```

7. Testing the model!

In this step, we assume that the training is complete and the model from the final epoch (`reset_880.tlt`) is available. If you would like to run evaluation on an earlier model, please edit the spec file at `$$SPEC_DIR/classification_retrain_spec.cfg` to point to the intended model.

```
In [ ]: !classification evaluate -e $$SPEC_DIR/classification_retrain_spec.cfg -k SKEY
```

8. Visualize Inferences

To see the output results of our model on test images, we can use the `tlt-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

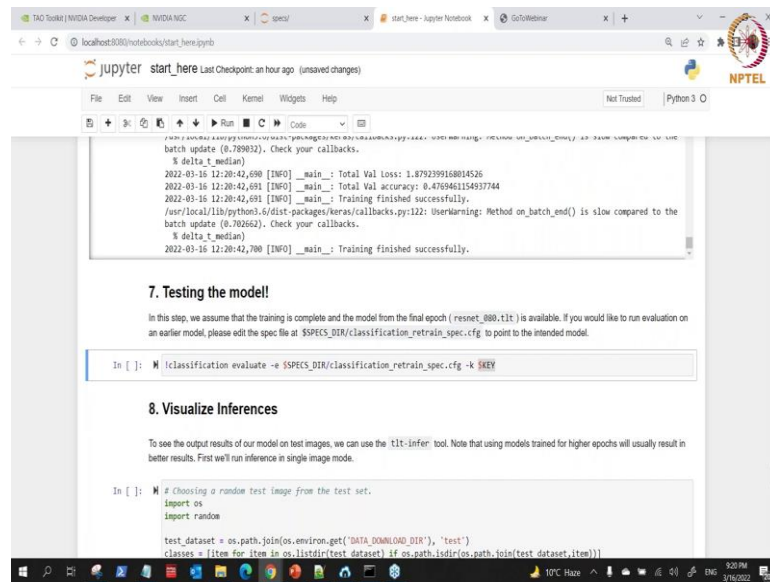
```
In [ ]: !# Choosing a random test image from the test set.
import os
import random

test_dataset = os.path.join(os.environ.get('DATA_DOWNLOAD_DIR'), 'test')
classes = [item for item in os.listdir(test_dataset) if os.path.isdir(os.path.join(test_dataset, item))]
```

So, once this is done the next thing to do is to proceed to testing. So, to test is also evaluation. So, you have just the keyword is just classification, then evaluate. So, with what specification file? So, with this specification file, but retrain specification file then you supply the key as well while second epoch, so in less than 2 minutes I am sure be done with that.

So, if we once it finishes, then we test then we can now do our visualized inference here. We can visualize our, do our inferencing. And once we have done with the inferencing, then we will be able to export our model and that is done for.

(Refer Slide Time: 18:24)



The screenshot shows a Jupyter Notebook interface with the following content:

```
batch update (0.789032). Check your callbacks.
  % delta_t_median)
2022-03-16 12:20:42,690 [INFO] _main_: Total Val Loss: 1.8792399160804526
2022-03-16 12:20:42,691 [INFO] _main_: Total Val accuracy: 0.4769461154637744
2022-03-16 12:20:42,691 [INFO] _main_: Training finished successfully.
/usr/local/lib/python3.6/dist-packages/keras/callbacks.py:122: UserWarning: Method on_batch_end() is slow compared to the
  % delta_t_median)
2022-03-16 12:20:42,700 [INFO] _main_: Training finished successfully.
```

7. Testing the model!

In this step, we assume that the training is complete and the model from the final epoch (`reset_880_tit`) is available. If you would like to run evaluation on an earlier model, please edit the spec file at `$$SPEC_DIR/classification_retrain_spec.cfg` to point to the intended model.

```
In [ ]: !classification_evaluate -e $$SPEC_DIR/classification_retrain_spec.cfg -k SKEY
```

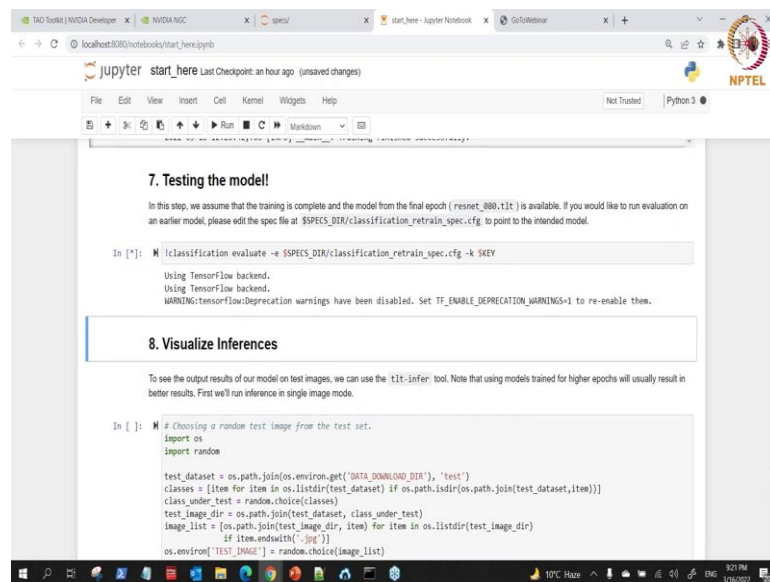
8. Visualize Inferences

To see the output results of our model on test images, we can use the `tit-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

```
In [ ]: # Choosing a random test image from the test set.
import os
import random

test_dataset = os.path.join(os.environ.get('DATA_DOWNLOAD_DIR'), 'test')
classes = [item for item in os.listdir(test_dataset) if os.path.isdir(os.path.join(test_dataset,item))]
```

(Refer Slide Time: 18:38)



The screenshot shows a Jupyter Notebook interface with the following content:

7. Testing the model!

In this step, we assume that the training is complete and the model from the final epoch (`reset_880_tit`) is available. If you would like to run evaluation on an earlier model, please edit the spec file at `$$SPEC_DIR/classification_retrain_spec.cfg` to point to the intended model.

```
In [*]: !classification_evaluate -e $$SPEC_DIR/classification_retrain_spec.cfg -k SKEY

Using TensorFlow backend.
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
```

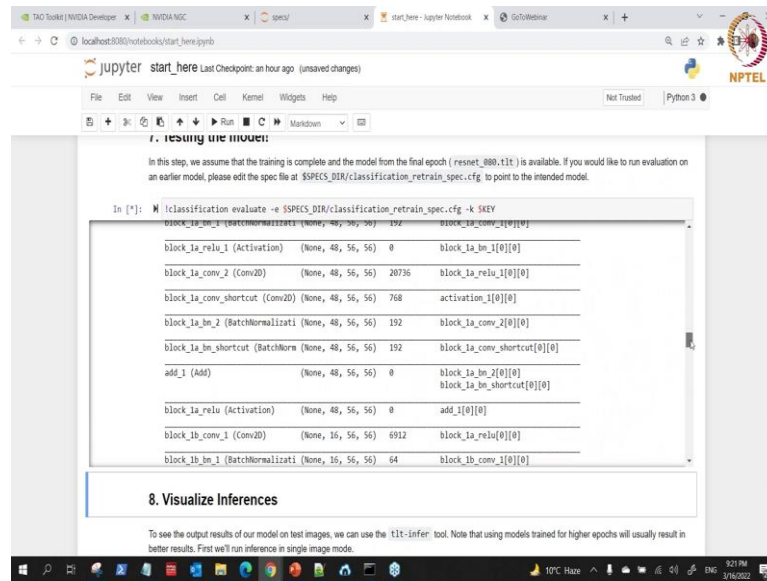
8. Visualize Inferences

To see the output results of our model on test images, we can use the `tit-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

```
In [ ]: # Choosing a random test image from the test set.
import os
import random

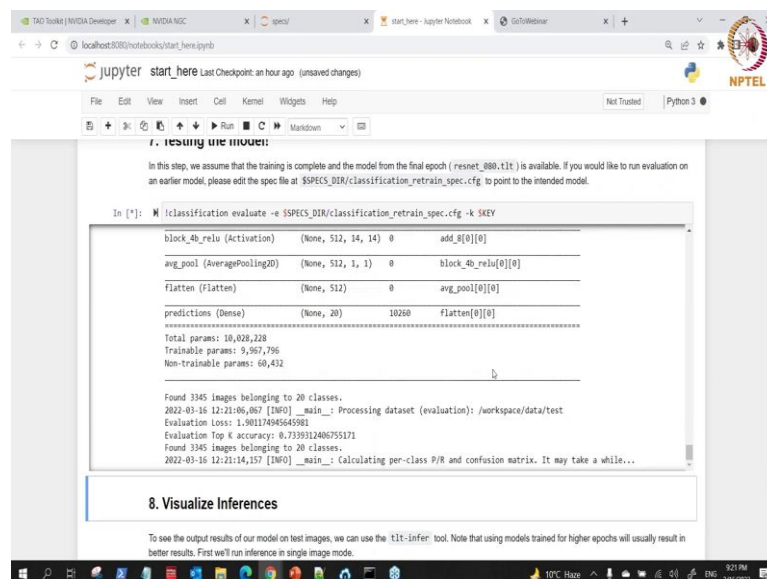
test_dataset = os.path.join(os.environ.get('DATA_DOWNLOAD_DIR'), 'test')
classes = [item for item in os.listdir(test_dataset) if os.path.isdir(os.path.join(test_dataset,item))]
class_under_test = random.choice(classes)
test_image_dir = os.path.join(test_dataset, class_under_test)
image_list = [os.path.join(test_image_dir, item) for item in os.listdir(test_image_dir)
               if item.endswith('.jpg')]
os.environ['TEST_IMAGE'] = random.choice(image_list)
```

(Refer Slide Time: 18:47)



So, successfully retrain, that has been retrained successfully; you can see it that has given that then. What do we need to do? We need to test that. So, what we do is what we test that. So, that we also go into testing. It is just the what we do even when we write it manually ourselves. You know when you train a model you have to test it with some data sets. Since we have a test data sets, so it is good to test with that.

(Refer Slide Time: 18:57)



(Refer Slide Time: 19:13)

```
In [16]: !classification_evaluate -e $SPECS_DIR/classification_retrain_spec.cfg -k SKEY
```

```
Classification Report
```

	precision	recall	f1-score	support
aeroplane	0.65	0.82	0.72	134
bicycle	0.51	0.41	0.45	111
bird	0.62	0.51	0.56	153
boat	0.44	0.47	0.45	102
bottle	0.56	0.10	0.17	142
bus	0.50	0.59	0.54	85
car	0.50	0.40	0.44	233
cat	0.72	0.71	0.72	216
chair	0.29	0.38	0.33	224
cow	0.29	0.03	0.06	61
diningtable	0.11	0.01	0.02	106
dog	0.57	0.57	0.57	258

8. Visualize Inferences

To see the output results of our model on test images, we can use the `!tlt-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

(Refer Slide Time: 19:15)

```
In [16]: !classification_evaluate -e $SPECS_DIR/classification_retrain_spec.cfg -k SKEY
```

```
Classification Report
```

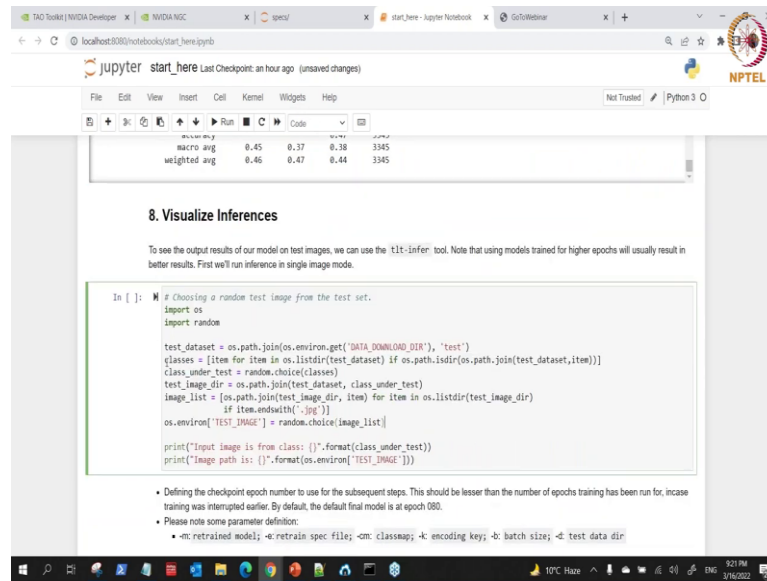
	precision	recall	f1-score	support
car	0.50	0.40	0.44	233
cat	0.72	0.71	0.72	216
chair	0.29	0.38	0.33	224
cow	0.29	0.03	0.06	61
diningtable	0.11	0.01	0.02	106
dog	0.57	0.57	0.57	258
horse	0.53	0.34	0.42	97
motorbike	0.55	0.47	0.51	106
person	0.41	0.70	0.52	818
pottedplant	0.11	0.01	0.02	106
sheep	0.31	0.15	0.21	65
sofa	0.32	0.06	0.10	102
train	0.60	0.48	0.53	109
tvmonitor	0.40	0.28	0.33	115
accuracy			0.47	3345
macro avg	0.45	0.37	0.38	3345
weighted avg	0.46	0.47	0.44	3345

8. Visualize Inferences

To see the output results of our model on test images, we can use the `!tlt-infer` tool. Note that using models trained for higher epochs will usually result in better results. First we'll run inference in single image mode.

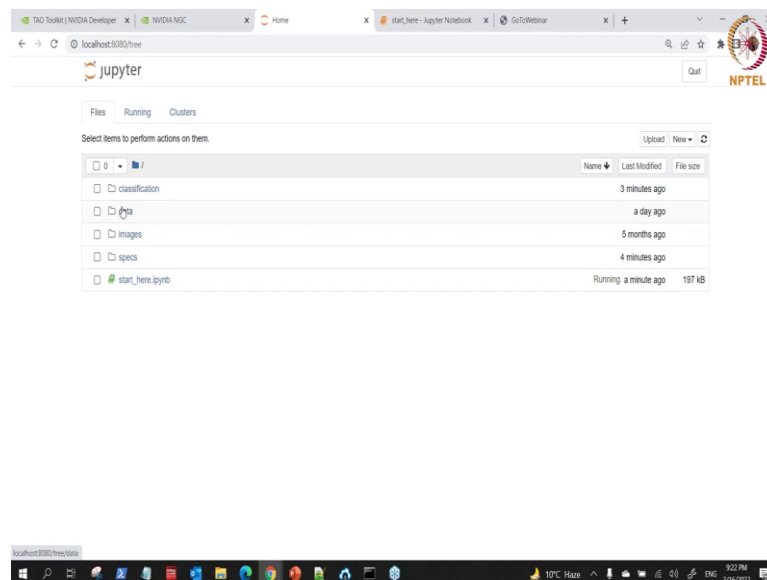
So, it is going to test and then once it the for the testing is just evaluation if we plot the confusion matrix and then gave us the classification report there.

(Refer Slide Time: 19:20)



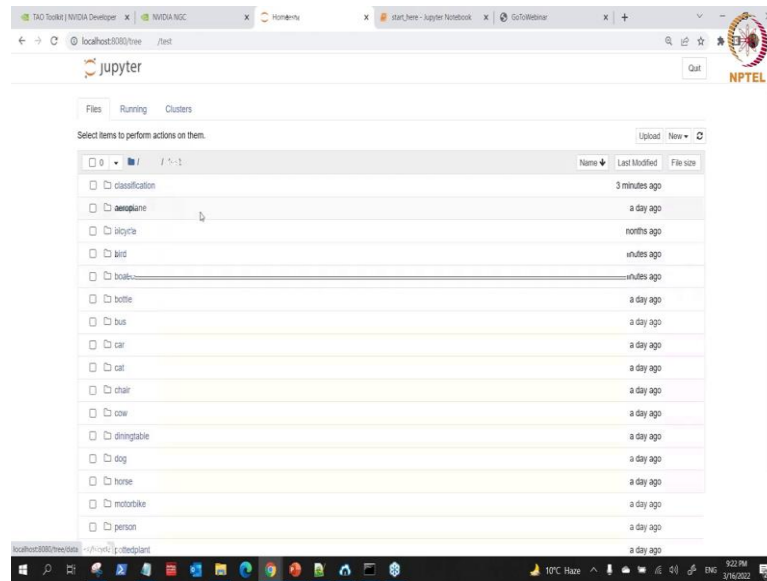
So, after that is done the next thing to do is what, ok let us start to inference. So, what we are trying to do here you can write this on your own, it is just that we want to select a particular image at random from the test folder.

(Refer Slide Time: 19:34)



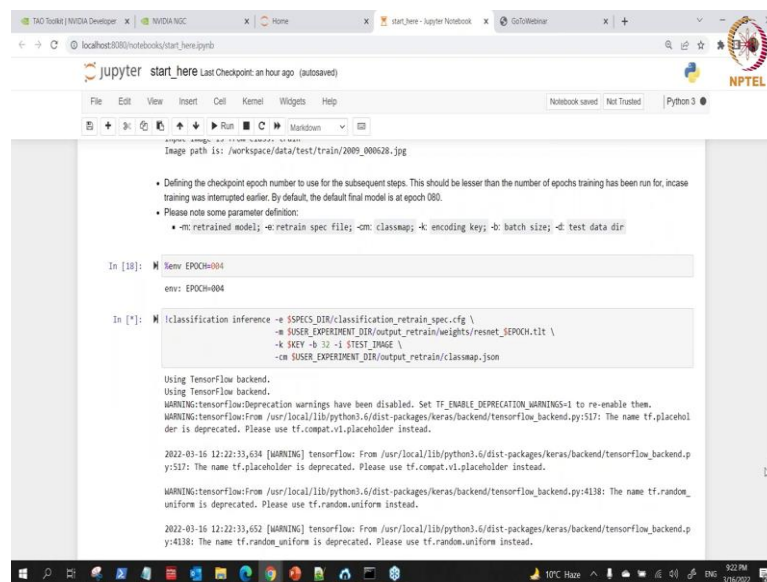
So, from the test folder here, let me come here, this is data and from the test folder here we just want to select the random, want to select a random data from there.

(Refer Slide Time: 19:36)



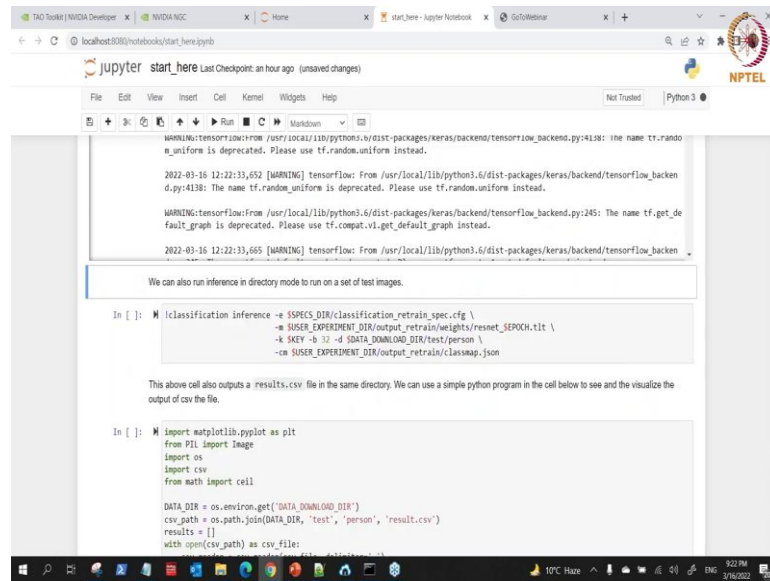
So, if I run this that is what he is doing there. So, you just speak the class, the class is trained and it has pick a particular image from there.

(Refer Slide Time: 19:44)



So, I am not expecting it to be able to identify these very because we trained with 4 epoch, mind you. But this is just a demo. It is just for you to know the flow. So, I do that, then here to run the inference here. So, even if the inference is not, the result is not correct with I am not bothered with that it say because we are just using it for demo here.

(Refer Slide Time: 20:17)



The screenshot shows a Jupyter Notebook window titled 'start_here' with a Python 3 kernel. The notebook contains several code cells. The first cell shows a command to run classification inference, which results in several TensorFlow deprecation warnings. The second cell contains a Python script that imports matplotlib.pyplot as plt, PIL Image, os, csv, and math, and sets up environment variables for data and result files. The third cell shows the execution of the inference command again, with more deprecation warnings.

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: the name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

2022-03-16 12:22:33,652 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:245: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

2022-03-16 12:22:33,665 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: the name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

We can also run inference in directory mode to run on a set of test images.

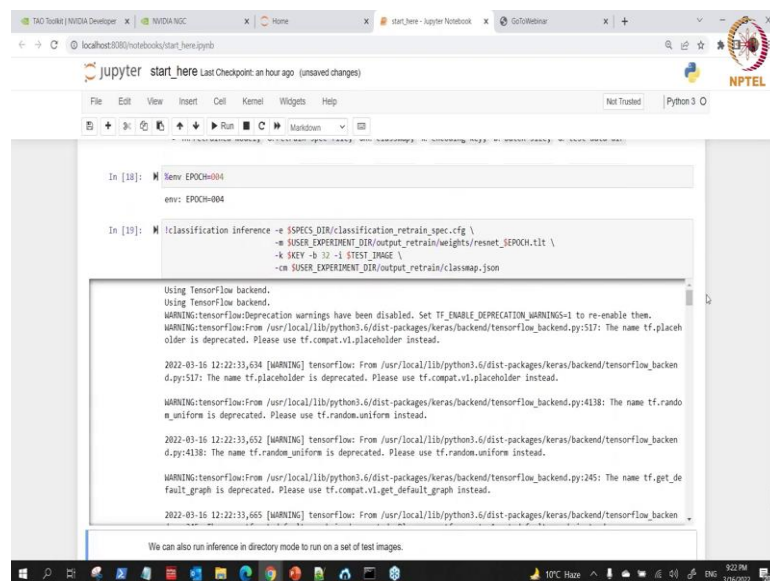
In [ ]: !classification_inference -e $SPECES_DIR/classification_retrain_spec.cfg \
      -m $USER_EXPERIMENT_DIR/output_retrain/weights/reset_SEPOCH.tilt \
      -k KEY -b 32 -d $DATA_DOWNLOAD_DIR/test/person \
      -c $USER_EXPERIMENT_DIR/output_retrain/classmap.json

This above cell also outputs a result.csv file in the same directory. We can use a simple python program in the cell below to see and the visualize the output of csv file.

In [ ]: import matplotlib.pyplot as plt
      from PIL import Image
      import os
      import csv
      from math import ceil

      DATA_DIR = os.environ.get('DATA_DOWNLOAD_DIR')
      csv_path = os.path.join(DATA_DIR, 'test', 'person', 'result.csv')
      results = []
      with open(csv_path) as csv_file:
```

(Refer Slide Time: 20:22)



The screenshot shows a Jupyter Notebook window titled 'start_here' with a Python 3 kernel. The notebook contains several code cells. The first cell shows a command to run classification inference, which results in several TensorFlow deprecation warnings. The second cell contains a Python script that imports matplotlib.pyplot as plt, PIL Image, os, csv, and math, and sets up environment variables for data and result files. The third cell shows the execution of the inference command again, with more deprecation warnings.

```
In [18]: !env EPOCH=004
env: EPOCH=004

In [19]: !classification_inference -e $SPECES_DIR/classification_retrain_spec.cfg \
      -m $USER_EXPERIMENT_DIR/output_retrain/weights/reset_SEPOCH.tilt \
      -k KEY -b 32 -d $DATA_DOWNLOAD_DIR/test/person \
      -c $USER_EXPERIMENT_DIR/output_retrain/classmap.json

Using TensorFlow backend.
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

2022-03-16 12:22:33,634 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: the name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

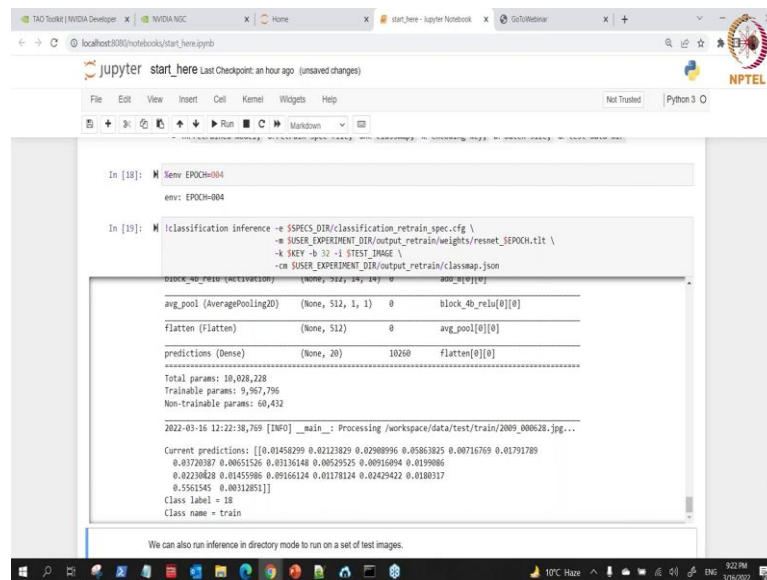
2022-03-16 12:22:33,652 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:245: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

2022-03-16 12:22:33,665 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: the name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

We can also run inference in directory mode to run on a set of test images.
```

(Refer Slide Time: 20:28)



```
In [18]: %env EPOCH=004
env: EPOCH=004

In [19]: %!classification inference -e $SPRCS_DIR/classification_retrain_spec.cfg \
-e $USER_EXPERIMENT_DIR/output_retrain/weights/resnet_SEPOCH.tlt \
-k SKEY -b 32 -i $TEST_IMAGE \
-cm $USER_EXPERIMENT_DIR/output_retrain/classmap.json

block_4b_relu (Activation) (None, 512, 1, 1) 0 block_4b_relu[0][0]
avg_pool (AveragePooling2D) (None, 512, 1, 1) 0 avg_pool[0][0]
flatten (Flatten) (None, 512) 0 avg_pool[0][0]
predictions (Dense) (None, 20) 10260 flatten[0][0]
-----
Total params: 10,028,228
Trainable params: 9,967,796
Non-trainable params: 60,432

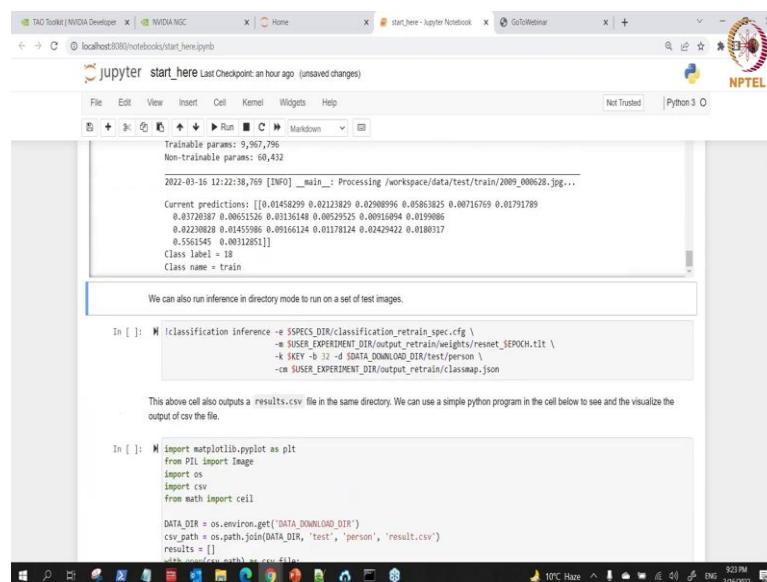
2022-03-16 12:22:38,769 [INFO] _main_: Processing /workspace/data/test/train/2009_000628.jpg...

Current predictions: [[0.01458299 0.02123829 0.02908996 0.05863825 0.00716769 0.01791789
0.03720387 0.00051526 0.01136148 0.00529525 0.00916094 0.0199086
0.02230828 0.01455986 0.09166124 0.01178124 0.02429422 0.0180317
0.5561545 0.00312851]]
Class Label = 18
Class name = train

We can also run inference in directory mode to run on a set of test images.
```

Then, we will be able to see the reported get it loads the model, oh luckily he is able to get it as train. I think it has. So, what did we selected is a train, ok. The class is trained. This is the image. It was able to identify it also as trained, the label, ok. We are lucky there.

(Refer Slide Time: 20:44)



```
Trainable params: 9,967,796
Non-trainable params: 60,432

2022-03-16 12:22:38,769 [INFO] _main_: Processing /workspace/data/test/train/2009_000628.jpg...

Current predictions: [[0.01458299 0.02123829 0.02908996 0.05863825 0.00716769 0.01791789
0.03720387 0.00051526 0.01136148 0.00529525 0.00916094 0.0199086
0.02230828 0.01455986 0.09166124 0.01178124 0.02429422 0.0180317
0.5561545 0.00312851]]
Class Label = 18
Class name = train

We can also run inference in directory mode to run on a set of test images.

In [ ]: %!classification inference -e $SPRCS_DIR/classification_retrain_spec.cfg \
-e $USER_EXPERIMENT_DIR/output_retrain/weights/resnet_SEPOCH.tlt \
-k SKEY -b 32 -d $DATA_DOWNLOAD_DIR/test/person \
-cm $USER_EXPERIMENT_DIR/output_retrain/classmap.json

This above cell also outputs a results.csv file in the same directory. We can use a simple python program in the cell below to see and the visualize the output of csv file.

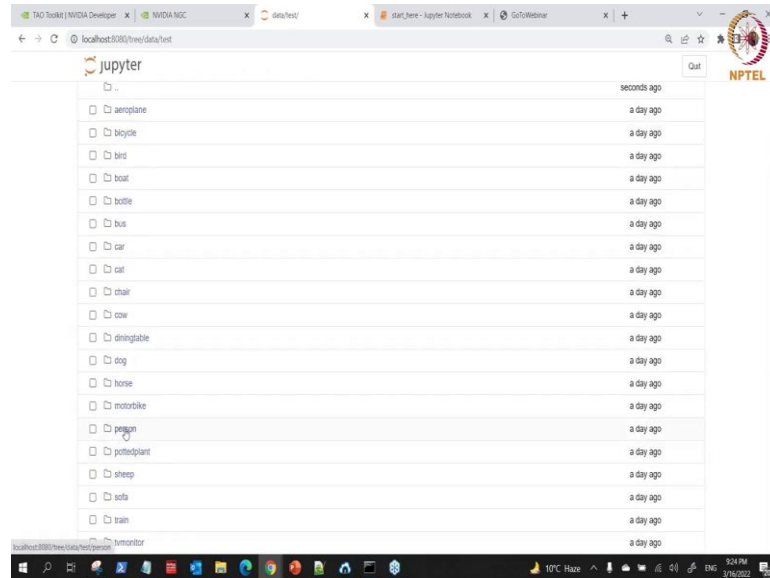
In [ ]: %import matplotlib.pyplot as plt
from PIL import Image
import os
import csv
from math import ceil

DATA_DIR = os.environ.get('DATA_DOWNLOAD_DIR')
csv_path = os.path.join(DATA_DIR, 'test', 'person', 'result.csv')
results = []
```

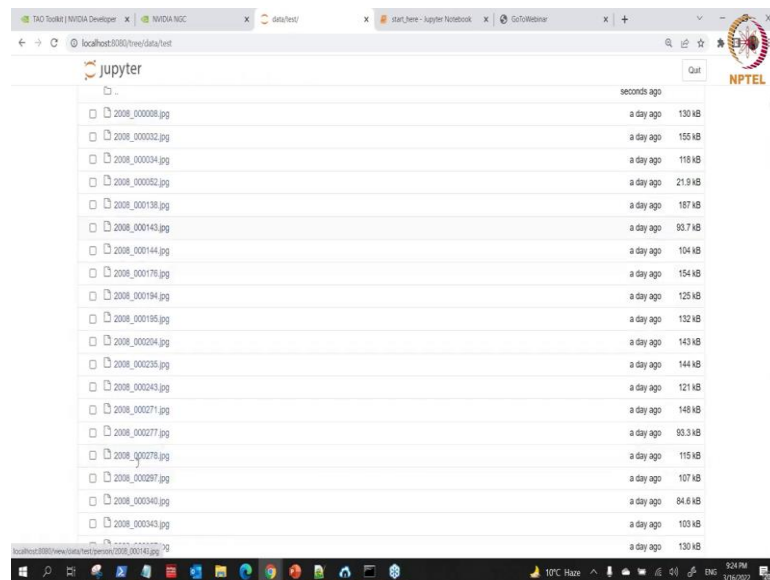
So, we are lucky there with 4 epoch is able to recognize that. Then also we can train we can also evaluate it inference on set of images. So, on set of images to inference that you can do you have classification and the keyword inference, then where which specification file to use retrain, where is the weights of the retrain model the paths, the

key here, the batch size start to. Then, data where do you load the class, the name, they have know which for that you want to use to evaluate or inference.

(Refer Slide Time: 21:42)

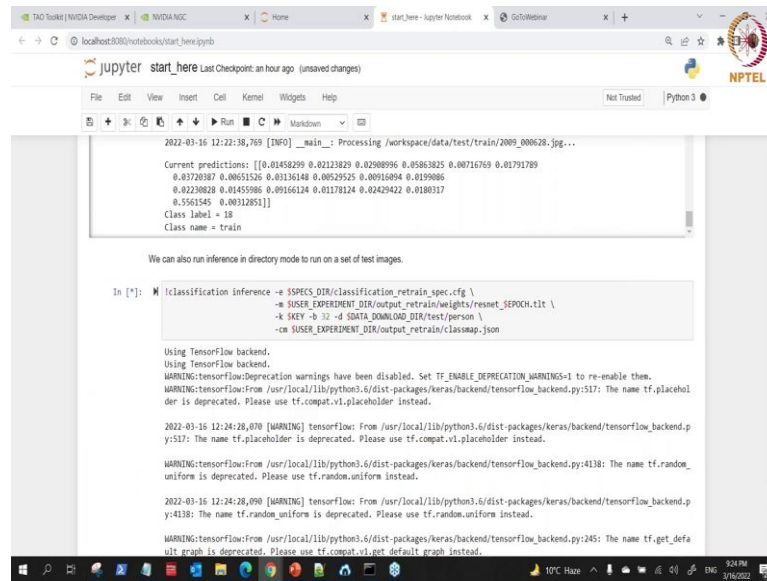


(Refer Slide Time: 21:44)



So, the folder the folder want to use the test 1 and we want to get a person from the test 1. So, from the test 1 here, you see if you come to data test, if we look for a person this is person. So, it will want to use it to test all these persons here. That is what we want to use it to want to test with that.

(Refer Slide Time: 21:58)



```
2022-03-16 12:22:38,769 [INFO] __main__: Processing /workspace/data/test/train/2009_000628.jpg...
Current predictions: [[0.01458299 0.02123829 0.02908996 0.05863825 0.00716769 0.01791789
0.03720287 0.00515126 0.01136148 0.00529525 0.00910994 0.019986
0.02238828 0.01455986 0.00166124 0.01178124 0.02429422 0.0180117
0.5561545 0.00312851]]
Class label = 18
Class name = train

We can also run inference in directory mode to run on a set of test images.

In [*]: !classification_inference -e $SPECES_DIR/classification_retrain_spec.cfg \
-e $USER_EXPERIMENT_DIR/output_retrain_weights/reset_SEPOCH.ttl \
-k SKEY -b 32 -d $DATA_DOWNLOAD_DIR/test/person \
-cm $USER_EXPERIMENT_DIR/output_retrain/classmap.json

Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

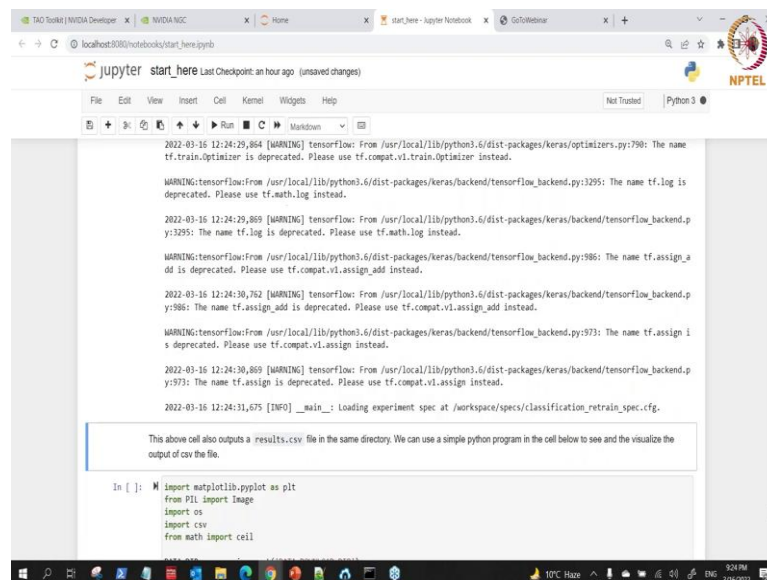
2022-03-16 12:24:28,070 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

2022-03-16 12:24:28,090 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:245: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

So, come back to data. So, that is what we are doing here. Then it is this one we seem we indicate the class name. So, it is we displayed graphics face and we can be able to see whether it is classifying well or not so, and that we go and then, ok. It is done.

(Refer Slide Time: 22:10)



```
2022-03-16 12:24:29,864 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3295: The name tf.log is deprecated. Please use tf.math.log instead.

2022-03-16 12:24:29,869 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3295: The name tf.log is deprecated. Please use tf.math.log instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:586: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

2022-03-16 12:24:30,762 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:586: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:573: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

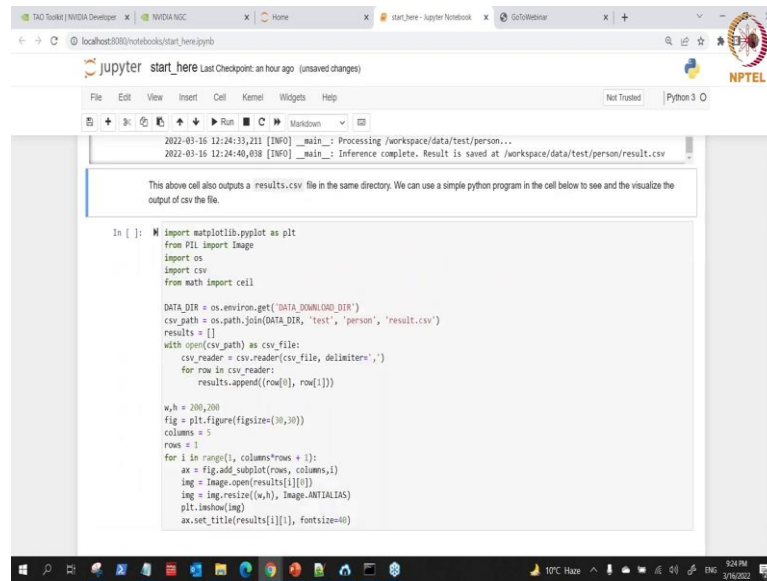
2022-03-16 12:24:30,869 [WARNING] tensorflow: From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:573: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

2022-03-16 12:24:31,075 [INFO] __main__: Loading experiment spec at /workspace/specs/classification_retrain_spec.cfg.

This above cell also outputs a results.csv file in the same directory. We can use a simple python program in the cell below to see and the visualize the output of csv file.

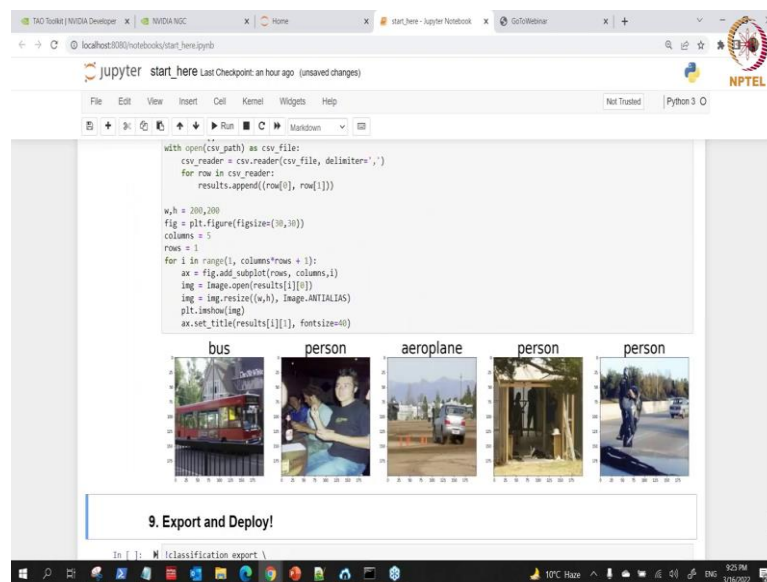
In [ ]: import matplotlib.pyplot as plt
from PIL import Image
import os
import csv
from math import ceil
```

(Refer Slide Time: 22:24)



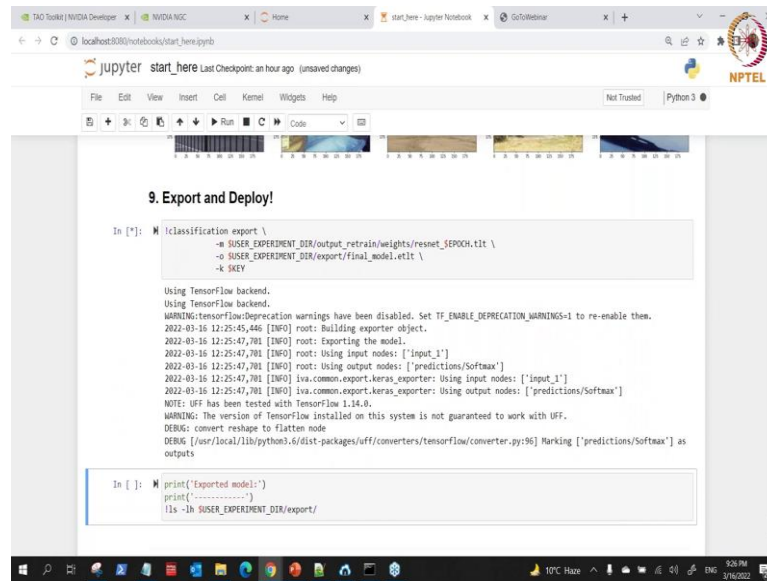
So, what we need to do is to what visualize the result. So, we can visualize the result with this, running this cell to visualize the result you can write yours, ok.

(Refer Slide Time: 22:33)



Let us see is able to classify that very well, ok. Get this is bus, this is person, this is not aeroplane, this is person and this is person because we train with 4 epoch. By time if we train with maybe 60, 80 epoch, everything will be the answer would be perfect there.

(Refer Slide Time: 22:55)



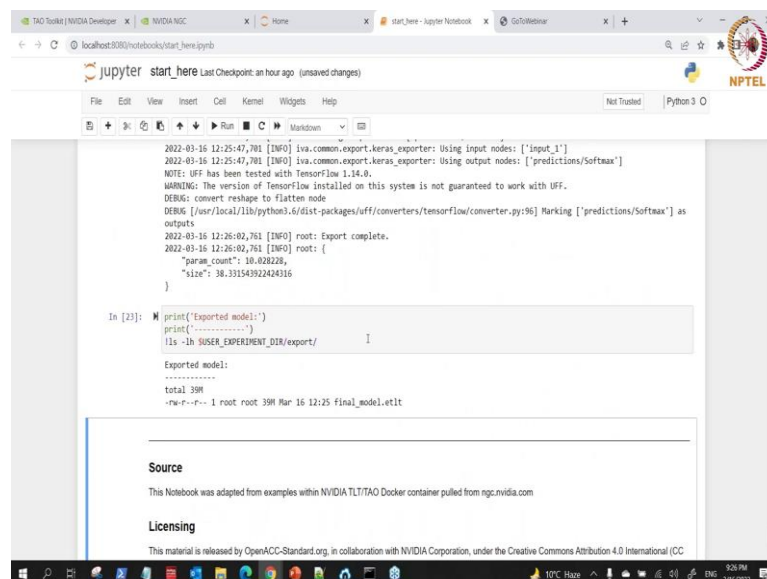
```
In [*]: !classification export \
        -o $USER_EXPERIMENT_DIR/output_retrain/weights/reset_epoch.tit \
        -e $USER_EXPERIMENT_DIR/export/final_model.tit \
        -k $KEY

Using TensorFlow backend.
Using TensorFlow backend.
WARNING:tensorflow:Deprecation warnings have been disabled. Set TF_ENABLE_DEPRECATION_WARNINGS=1 to re-enable them.
2022-03-16 12:25:45,446 [INFO] root: Building exporter object.
2022-03-16 12:25:47,781 [INFO] root: Exporting the model.
2022-03-16 12:25:47,781 [INFO] root: Using input nodes: ['input_1']
2022-03-16 12:25:47,781 [INFO] root: Using output nodes: ['predictions/Softmax']
2022-03-16 12:25:47,781 [INFO] iiva.common.export.keras_exporter: Using input nodes: ['input_1']
2022-03-16 12:25:47,781 [INFO] iiva.common.export.keras_exporter: Using output nodes: ['predictions/Softmax']
NOTE: UFF has been tested with TensorFlow 1.14.0.
WARNING: The version of TensorFlow installed on this system is not guaranteed to work with UFF.
DEBUG: convert reshape to flatten node
DEBUG: [/usr/local/lib/python3.6/dist-packages/uff/converters/tensorflow/convolver.py:96] Marking ['predictions/Softmax'] as
outputs

In [ ]: print("Exported model:")
        print(.....)
        !ls -lh $USER_EXPERIMENT_DIR/export/
```

So, then, so the next thing to do is to what export our model. So, how do you export your model? You have the flag a which is classification then the keyword export. So, export which model where is the path of the model, this is where the model save the path. And where do you want to export it, which folder? This is the path of the folder here. There is a folder that I will call export here. Then, you need the key also to do that.

(Refer Slide Time: 23:40)



```
2022-03-16 12:25:47,781 [INFO] iiva.common.export.keras_exporter: Using input nodes: ['input_1']
2022-03-16 12:25:47,781 [INFO] iiva.common.export.keras_exporter: Using output nodes: ['predictions/Softmax']
NOTE: UFF has been tested with TensorFlow 1.14.0.
WARNING: The version of TensorFlow installed on this system is not guaranteed to work with UFF.
DEBUG: convert reshape to flatten node
DEBUG: [/usr/local/lib/python3.6/dist-packages/uff/converters/tensorflow/convolver.py:96] Marking ['predictions/Softmax'] as
outputs
2022-03-16 12:26:02,781 [INFO] root: Export complete.
2022-03-16 12:26:02,781 [INFO] root: {
  "param_count": 10.028228,
  "size": 38.33154392424316
}

In [23]: print("Exported model:")
        print(.....)
        !ls -lh $USER_EXPERIMENT_DIR/export/

Exported model:
.....
total 39M
-rw-r--r-- 1 root root 39M Mar 16 12:25 final_model.tit

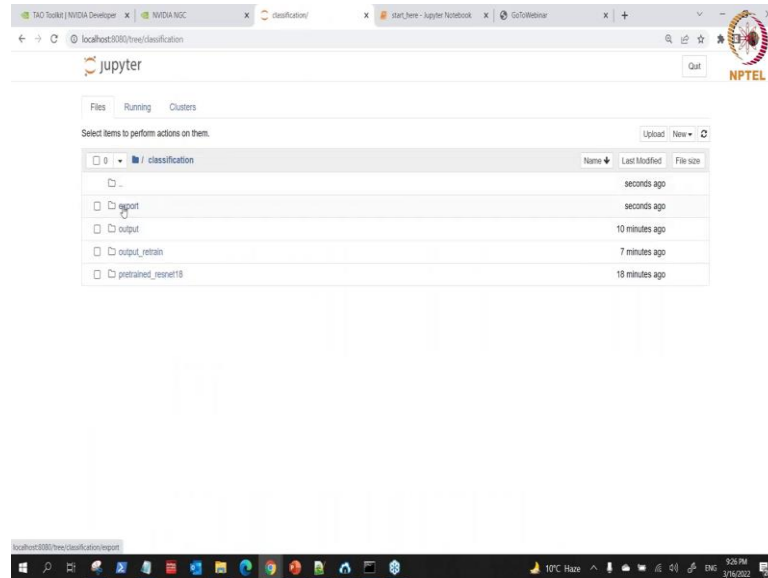
Source
This Notebook was adapted from examples within NVIDIA TLI/TAO Docker container pulled from nvc.nvidia.com

Licensing
This material is released by OpenACC-Standard.org, in collaboration with NVIDIA Corporation, under the Creative Commons Attribution 4.0 International (CC
```

So, we just run these and then we will have I am using on how. So, this way run. Then, after this run we just validate that whether it is actually saved in the path we specify shall

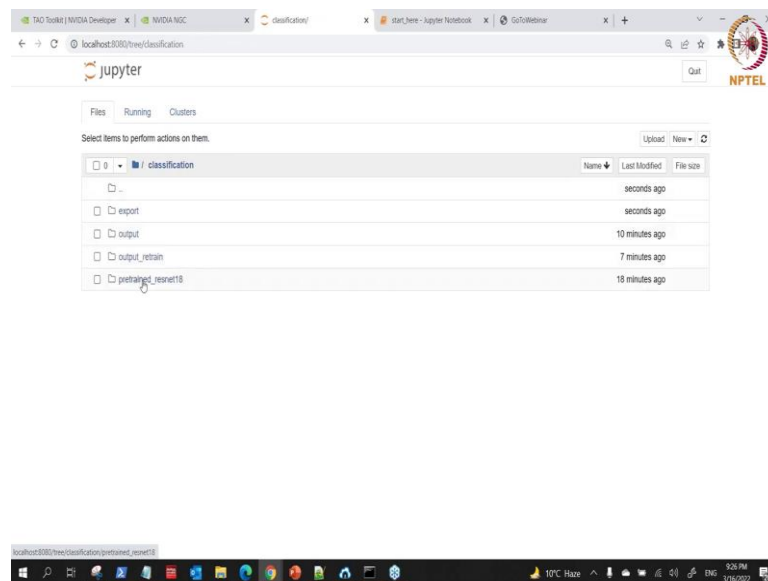
we do that, ok. I think that is done. Then let us check here we check that it is, ok. So, that is that about NVIDIA Tao tool kit to use it that, save it here.

(Refer Slide Time: 23:58)

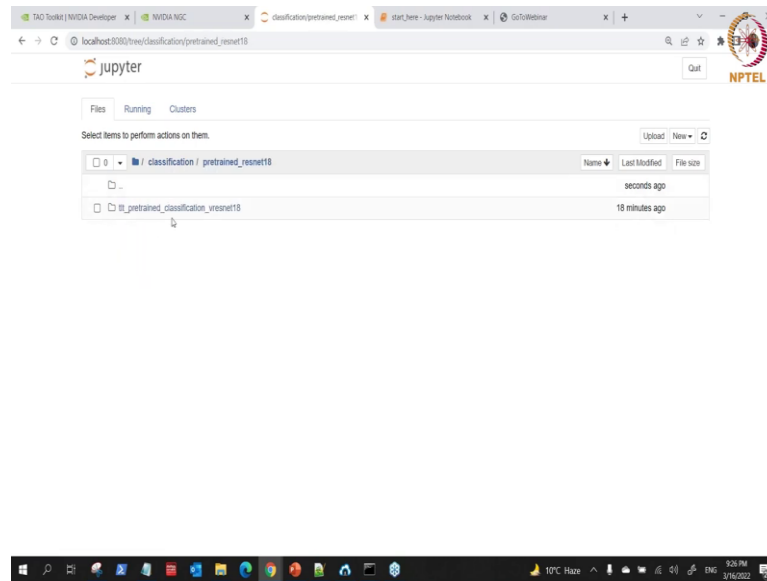


So, you will be able to check your folder, you see classification, you will be able to see export that this is the model there. I have been exported, this is the pretrained model, you can also see that there. So, all of that is done.

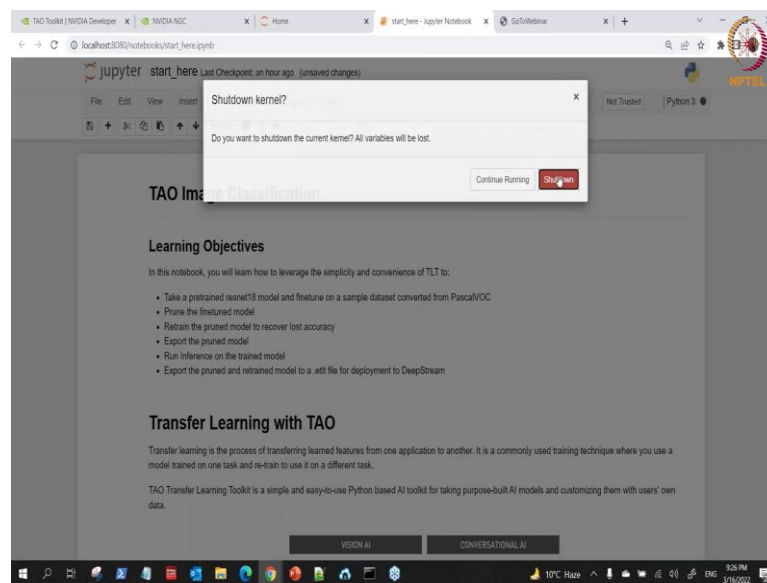
(Refer Slide Time: 24:03)



(Refer Slide Time: 24:04)

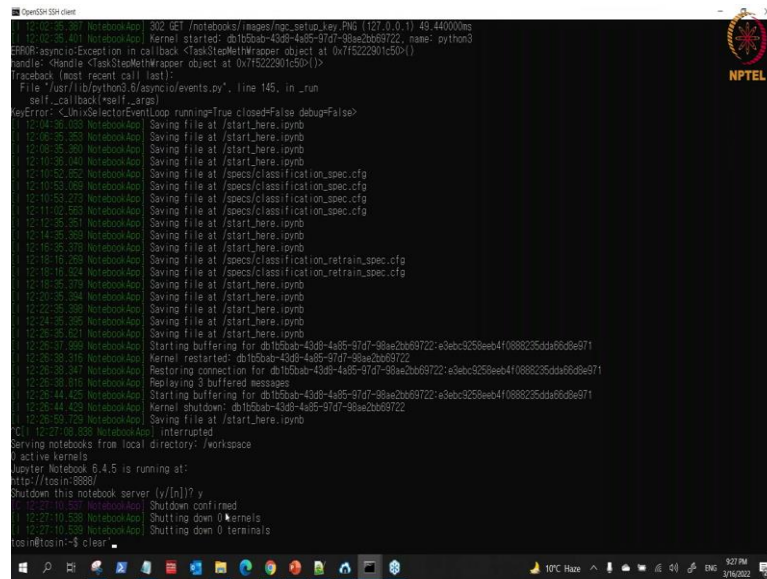


(Refer Slide Time: 24:23)



So, what I would do now is just to clear this, then turn down this kernel because we need to move folder. So, after shutting down this kernel, I can just close these, ok, alright, yes.

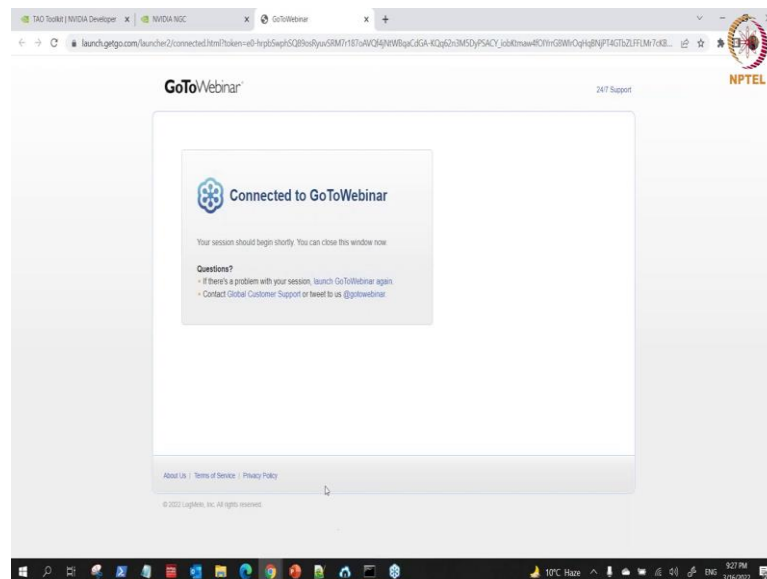
(Refer Slide Time: 24:44)



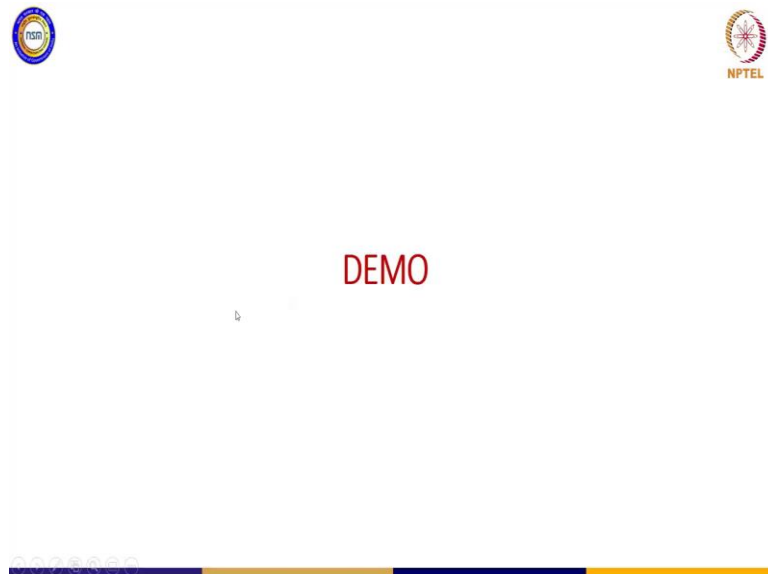
```
OpenSSH client
12:07:35.361 NotebookApp 302 GET /notebooks/pages/setup_kernel (127.0.0.1) 49.440000ms
12:07:35.401 NotebookApp Kernel started: db1b5bab-43d9-4a85-97d7-98ae2bb69722, name: python3
ERROR:asyncio:Exception in callback <TaskStepperWrapper object at 0x7f522291c509()
handle: <Handle <TaskStepperWrapper object at 0x7f522291c509()!>
[TaskError: (test recent call) (last)]
File ~/usr/lib/python3.8/asyncio/events.py, line 145, in _run
self._callback(*self._args)
KeyError: <UnixSelectorEventLoop running=True closed=False debug=False>
12:07:35.403 NotebookApp Saving file at /start_here.ipynb
12:07:35.523 NotebookApp Saving file at /start_here.ipynb
12:07:35.569 NotebookApp Saving file at /start_here.ipynb
12:10:53.040 NotebookApp Saving file at /specs/classification_spec.cfg
12:10:53.052 NotebookApp Saving file at /specs/classification_spec.cfg
12:10:53.068 NotebookApp Saving file at /specs/classification_spec.cfg
12:10:53.273 NotebookApp Saving file at /specs/classification_spec.cfg
12:11:02.559 NotebookApp Saving file at /specs/classification_spec.cfg
12:12:25.371 NotebookApp Saving file at /start_here.ipynb
12:14:25.363 NotebookApp Saving file at /start_here.ipynb
12:18:25.378 NotebookApp Saving file at /start_here.ipynb
12:18:15.259 NotebookApp Saving file at /specs/classification_retrain_spec.cfg
12:18:15.324 NotebookApp Saving file at /specs/classification_retrain_spec.cfg
12:18:25.370 NotebookApp Saving file at /start_here.ipynb
12:20:25.354 NotebookApp Saving file at /start_here.ipynb
12:22:25.359 NotebookApp Saving file at /start_here.ipynb
12:24:25.351 NotebookApp Saving file at /start_here.ipynb
12:26:27.359 NotebookApp Starting buffering for db1b5bab-43d9-4a85-97d7-98ae2bb69722: e9bc9259eeb410888235d456d8e971
12:28:33.318 NotebookApp Kernel restarted: db1b5bab-43d9-4a85-97d7-98ae2bb69722
12:29:38.347 NotebookApp Restoring connection for db1b5bab-43d9-4a85-97d7-98ae2bb69722: e9bc9259eeb410888235d456d8e971
12:29:38.318 NotebookApp Replaying 3 buffered messages
12:28:44.425 NotebookApp Starting buffering for db1b5bab-43d9-4a85-97d7-98ae2bb69722: e9bc9259eeb410888235d456d8e971
12:28:44.423 NotebookApp Kernel shutdown: db1b5bab-43d9-4a85-97d7-98ae2bb69722
12:30:59.729 NotebookApp Saving file at /start_here.ipynb
12:12:27.439 NotebookApp Interrupted
Saving notebooks from local directory: /workspace
0 active kernels
Jupyter Notebook 6.4.5 is running at:
http://localhost:8888/
Shutdown this notebook server (y/[n])? y
12:27:10.538 NotebookApp Shutdown confirmed
12:27:10.538 NotebookApp Shutting down 0 kernels
12:27:10.538 NotebookApp Shutting down 0 terminals
top@topin:~$ clear
```

And then come here, I just I am stopping this, ok. Have clear, ok. And I can close this.

(Refer Slide Time: 25:04)



(Refer Slide Time: 25:11)



So, we will proceed. So, we have run that demo, then let us look at a little time we have if we can move to YOLO5.

(Refer Slide Time: 25:16)



So, YOLO5 for computer vision, YOLO is a family of object detection. So, we use YOLO for object detection. It is highly based for computer vision. So and where do you get that? You can get that by visiting this website here. It is the original website. And then this is the Git Hub repo. So, you go to Git Hub, we clone, you can clone this repo.

So, now how do you do that? You need to clone following these steps. After you clone then you have you cd into the file the folder the directory, then you keep install some requirements. So, after you have done that, then you can now run your YOLO.

You can run because it can detect up to 80 objects. So, you just run python, detect.py and then source; source means the input where do you want to get your input from. If you want to supply input from a webcam, a single webcam you put it 0 here. So, it shows inputs will be coming and you will be seeing the computer vision the data will be what object detection running live.

If it is images you supply image here the path of the image rather than putting 0 here, you put the path of the path of the image, you put the path of the image there. If this video you can also what do detection on video. So, you put the path of the what video there then if you want to do object detection on set of images, you also put the path there. And you can do as well for YouTube and also for RTSP also. You can also run that there.

(Refer Slide Time: 27:03)

Custom data training with YOLOv5

- Prepare your dataset
- Annotate/label your dataset using Roboflow (<https://roboflow.com/?ref=ultralytics>). You may as well use Yolo_mark
- Prepare your data.yaml file
- Select weight based on the YOLOv5 model size preference (<https://github.com/ultralytics/yolov5/releases>)
 - YOLOv5n
 - YOLOv5s
 - YOLOv5m
 - YOLOv5l
 - YOLOv5x
- Train your custom data


```
python3 train.py --img 640 --data ./yolov5/dataset/data.yaml --cfg models/yolov5l.yaml --weights yolov5l.pt --batch 64 --epochs 100
```
- Run inference


```
python3 detect.py --source 0 --weights ./yolov5/runs/train/exp6/weights/best.pt --img 640
```

Source: <https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>

So, and now, so I have not done that. So, where does the transfer learning it comes in? It comes in when you want to use that particular model for your custom data. So, they call it custom data training with YOLO5. So, you bring your own data, not the data in the YOLO. So, YOLO5 was trained using coco dataset.

Now, you have your own data. So, how do you prepare your data set? You have your data in this format here. So, test, train, and validation in this set here. So, in each of them you will have folder in each of them which is called images. This is where your image will be. And you will have another one label, although this label will be empty at first.

So, the next thing to do is to what? Is to label these images so, how do you label or you annotate these images? So, you go to what we call the roboflow. This is the website here. So, in that website you upload this your data there and you do labelling and after then if you generate this label file for you. Or, you can also use what we call the YOLO mark, although you may need to fine tune some other things using YOLO mark because YOLO mark is specifically dedicated for YOLOv4.

However, the only thing you need to do is just the way it arranged the classes. So, if we generate the label. So, what is inside the label? This is what is inside the label, yeah. It has 5 features. Now, the first one is to give it the object class. So, if you have if you are classifying 5 objects. So, it would be, you would give them value of 0, 1, 2, 3, 4.

So, this is an object class. The second one is the X-center of the object in the image and this is the Y-center of that object, and the third one the third one is the Y-center. The fourth one is the width of the image, and the last one is the height of the image. So, if you see the second one here is 0, this is another class.

Then, the next thing to do is to prepare your data.yaml file. So, the data.yaml file is what you have here. So, you have to what you add these paths. So, based on your folder, the folder the YOLO folder you clone, YOLOv5, then you include data set path. This path is to this data. Then, your train, you have your train and validation. So, it will be train/images train/validation.

Once you use Robo, Robo will help you to generate these. So, the nc here means what? The number of classes. Let us say we are considering two classes and the name of the classes is just this. So, you have two classes, dog and person which is two classes. So, after you have done that. So, your data set will look this way. You have test, you have trained, you have validation, you have your data.yaml there.

So, the next thing to do is to select the weights you want for YOLO5 because YOLO5 has set of weights, different set of weight based on the model. So, you can have this is

for YOLO, if you want to consider nano YOLO, if you want to consider small YOLO, you want to consider medium size, large size or extra large size are there. And these are the data set, these are the weights that you can download. This how they look like.

So, after you have done that then you can train your custom model. So, how do you train your custom model you can train them using this python 3 because I am running on Bluetooth, that is why we have python 3 train.py and this img is talking about the dimension of your image. Then, data, that is where is the path to this data yaml. You specify that.

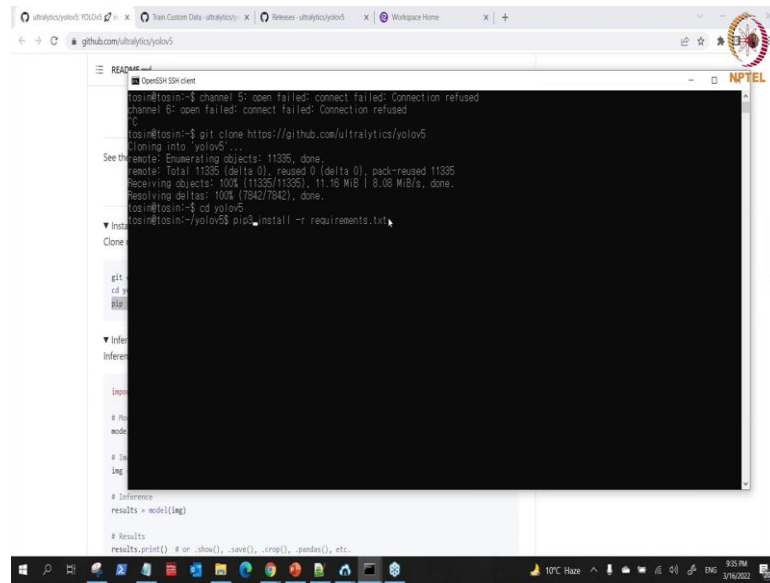
And then there is the word the configuration file for YOLO5. This is the path you specify it. Then, the weight you want to use based on any of these so if we select large, so the weight will be YOLOv5l.pt that is the weights. And then the batch size and then the number of epoch you want to.

So, after you have done that then you can also run; if after you have trained successfully you can run your inference using this python3 detect.py. Then your source, if this is an example for using camera webcam camera, so when I say source, its 0. So, that is webcam camera.

Then, the weight. So, the weight will not be the weight that is here again because this one is YOLOv5 weight. Now, you have trained your model your model the weight will be saved in this paths. So, it will save it as dot best. So, you specify that weight and the image size and then you are good to go.

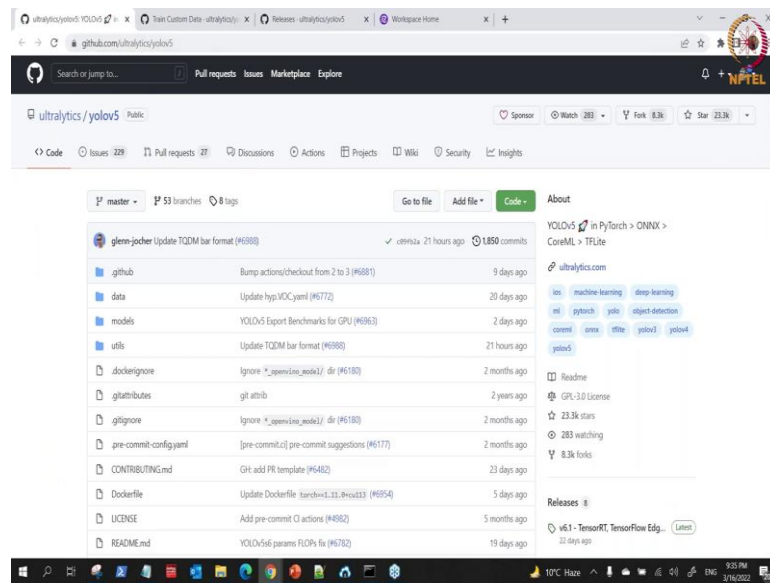
So, all of that you can see, you can be for custom image, you will be able to access them in this with this link. So, I will give a demo. But unfortunately, we may not be able to use webcam because I am doing necessary I have been trying it, but it is not working with this size.

(Refer Slide Time: 32:21)



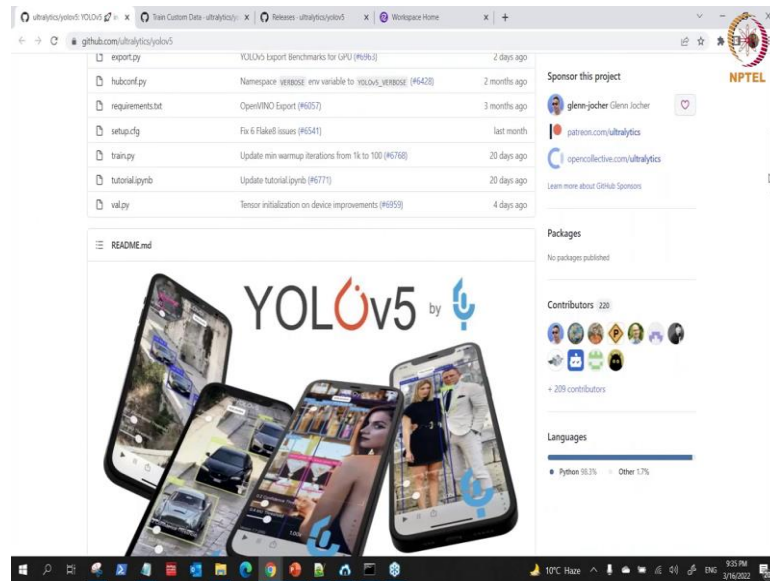
```
tosin@tosin:~$ channel 5: open failed: connect failed: Connection refused
channel 5: open failed: connect failed: Connection refused
C
tosin@tosin:~$ git clone https://github.com/ultralytics/yolov5
Cloning into yolov5...
See the status: Enumerating objects: 11335, done.
remote: Total 11335 (delta 0), reused 0 (delta 0), pack-reused 11335
Receiving objects: 100% (11335/11335), 11.16 MiB | 8.08 MiB/s, done.
Resolving deltas: 100% (7842/7842), done.
tosin@tosin:~$ cd yolov5
tosin@tosin:~/yolov5$ pip3 install -r requirements.txt
```

(Refer Slide Time: 32:36)

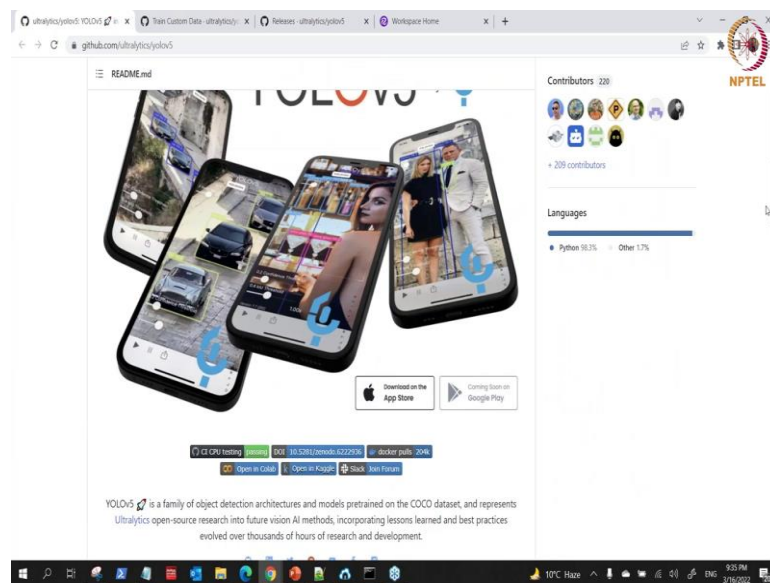


The screenshot shows the GitHub repository page for `ultralytics/yolov5`. The repository is public and has 238 watchers, 8.3k forks, and 23.3k stars. The commit history is visible, with the most recent commit by `glennjocher` titled "Update TQDM bar format (#6588)" committed 21 hours ago. The repository includes a README, LICENSE, CONTRIBUTING, and various utility files like `dockerignore`, `gitattributes`, and `gitignore`. The right sidebar shows the repository's license as GPL-3.0 and lists related topics like `machine-learning`, `deep-learning`, and `object-detection`.

(Refer Slide Time: 32:43)

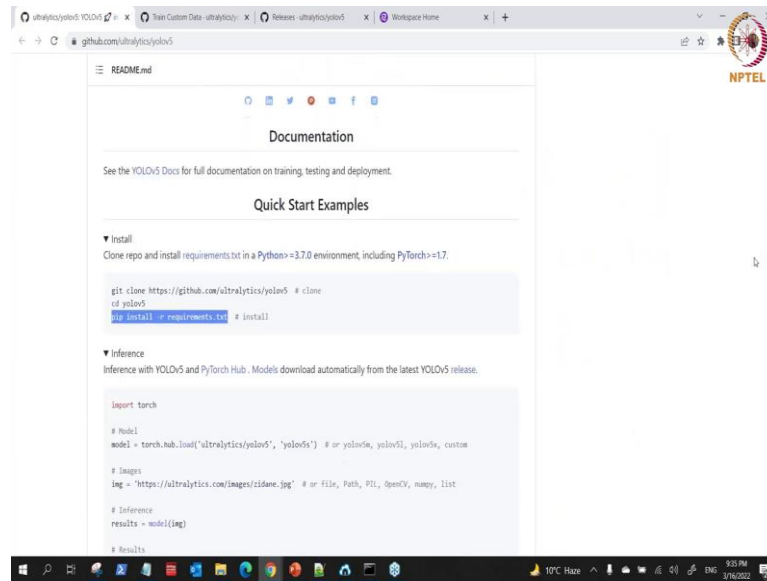


(Refer Slide Time: 32:44)



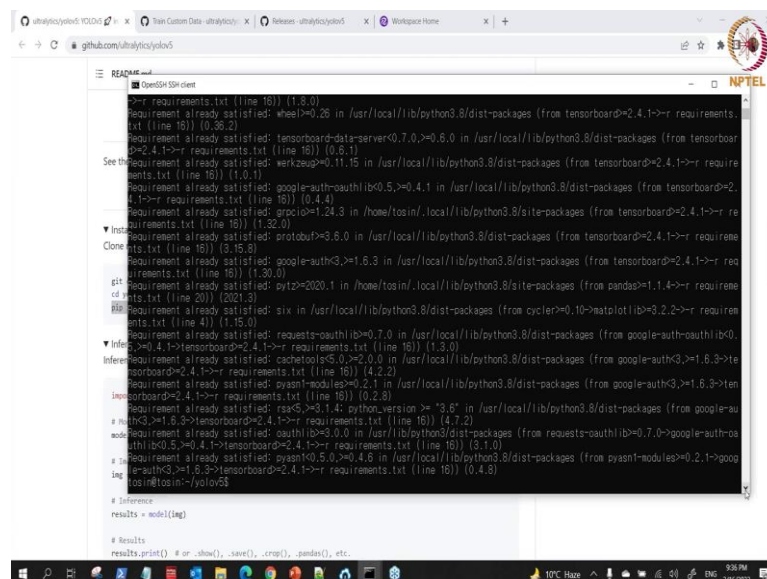
So, what we need to do now is to come here, ok. So, first thing to do is to download. So, let us go here, ok. So, this is the site for the GitHub for YOLOv5. So, you can see it for YOLOv5. They are requirement there the steps to follow.

(Refer Slide Time: 32:45)



So, we can come here and say, ok let us copy this. It is clone you just copy, it clone and you are here. So, I just cannot put us here. So, I just it will clone, hope there is more because I have run this before. So, let us see then cd yolov5, then. So, the next thing to do is to what pip install the requirements. I hope there will be no conflict because I have done this for; so, because I am on Ubuntu my pip will be pip 3.

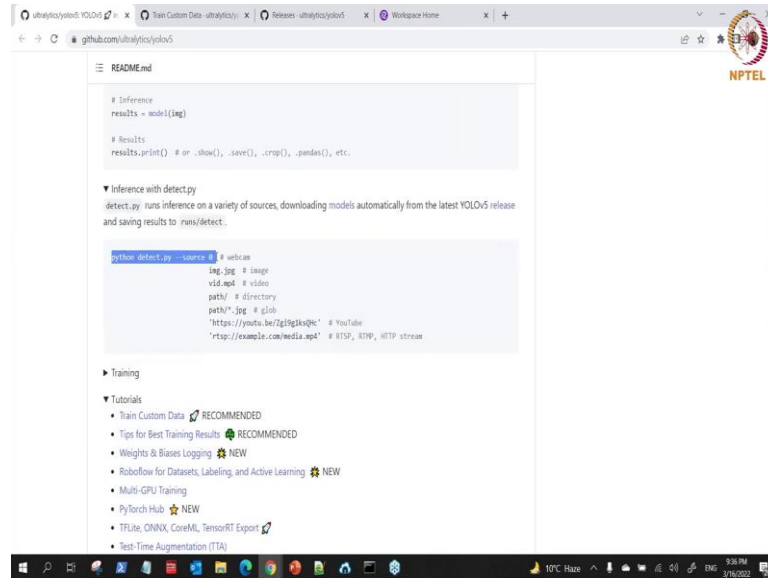
(Refer Slide Time: 33:24)



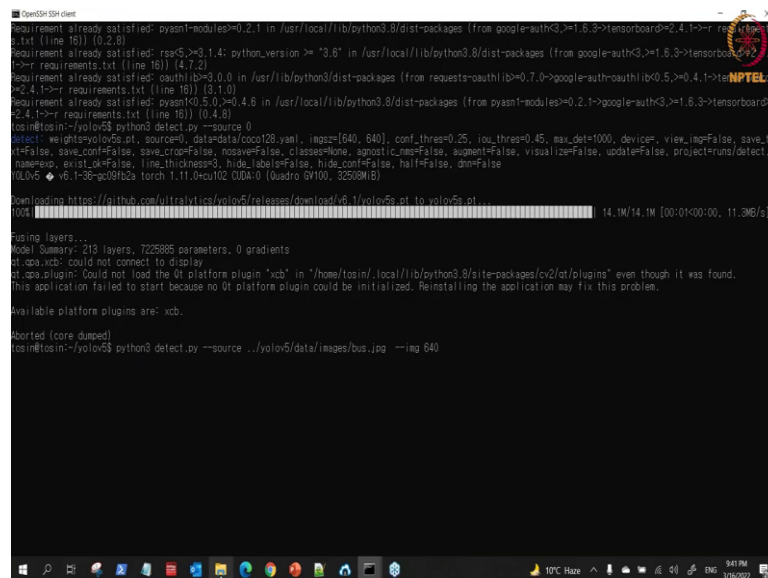
So, it we do that, ok since it has been stopped before with that. So, now, YOLOv5 is ready, not the custom one, but YOLOv5 is ready now. So, how do we run YOLOv5

now? So, if I want to use, I want to use what we call the webcam that will be an issue, but I will show you.

(Refer Slide Time: 33:54)

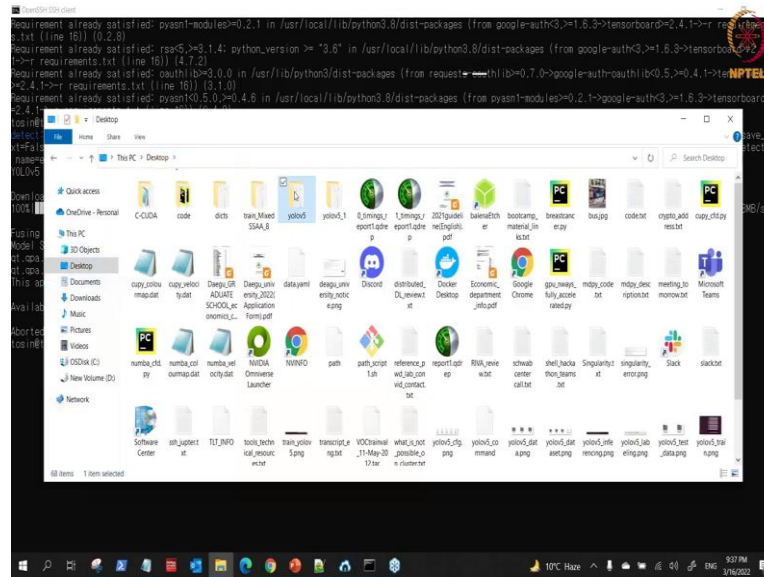


(Refer Slide Time: 34:02)



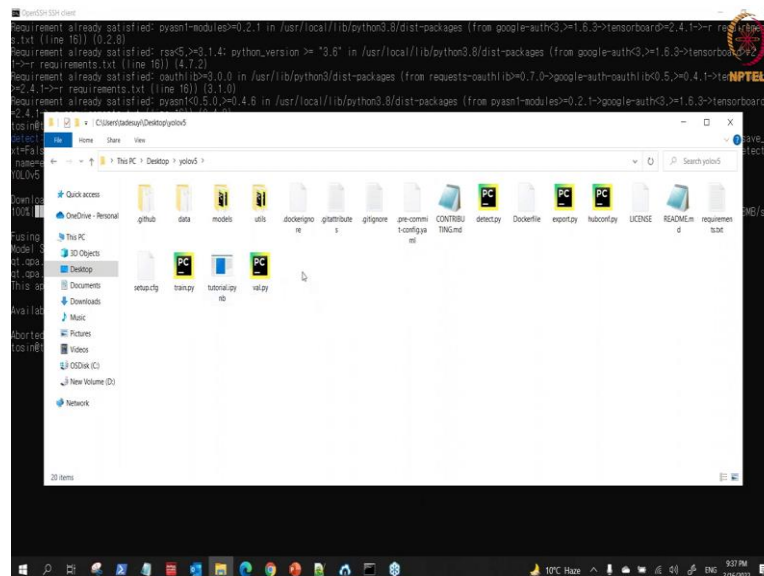
So, if I say python detect. So, this there will be it will complain of something, so it will be python 3 rather, ok. So, python 3 detect. I hope you can see my screen. So, that is, so this is python 3 detect, so, ok. So, it is running now. So, it is downloading the weights. So, it aborted because ideally there is we cannot use webcam.

(Refer Slide Time: 34:56)



So, what we need to do is let me just show you let me go to my folder here. So, once you download your YOLO, this is YOLO. So, once you download it this is how the folder looks like.

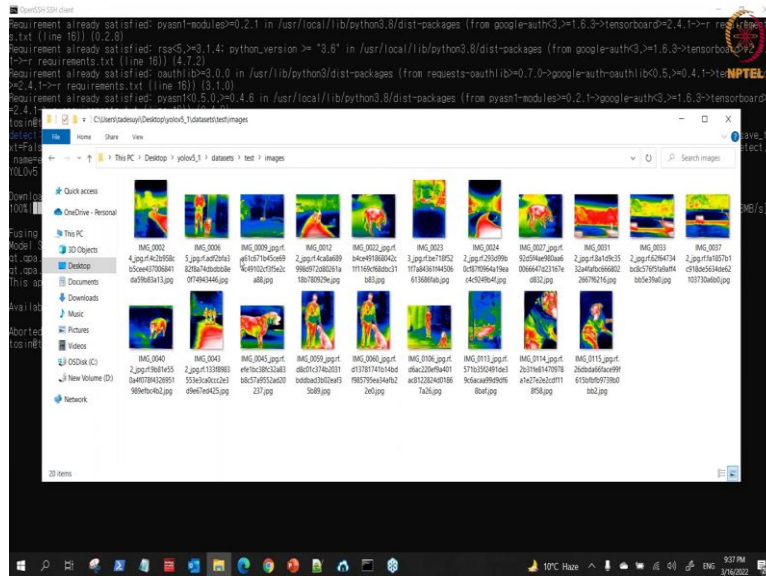
(Refer Slide Time: 34:58)



So, the folder looks like this, then what we need to do? In order to run that is if you go to, you need to include a new folder for your a folder for your dataset. So, the one I have done before is that I included the new folder, this is the one I have done before. So, I included this, this is the data set.

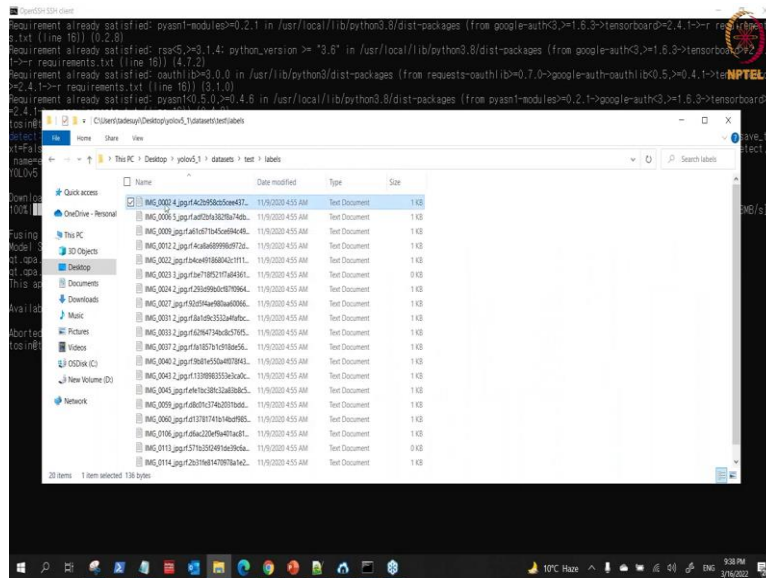
So, what happened here? This is the test, ok. Let me show you this is the test. So, you have its image and what; label here.

(Refer Slide Time: 35:28)

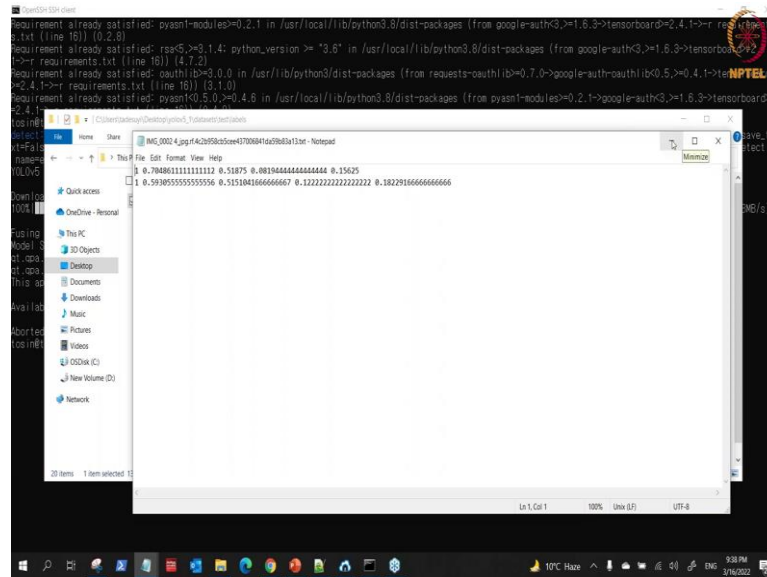


So, this is the image. I am just using this. This image is for image for dog and person.

(Refer Slide Time: 35:40)

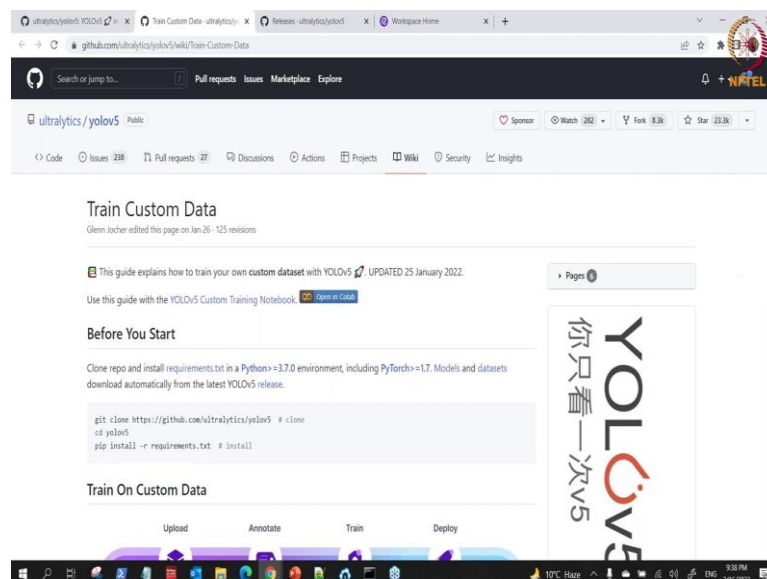


(Refer Slide Time: 35:44)



So, then the label is in this format. So, if you look at the label here, this is the format here. So, in this class which I have shown you before. But how do you get this format? So, for you to get this format what you need to do is to, what you need to do is to go to come here please; I want to waste the time. What you need to do is to go to Robo, here there is Robo here.

(Refer Slide Time: 36:12)



(Refer Slide Time: 36:16)

The screenshot shows the 'Train Custom Data' page on the YOLOv5 GitHub repository. It features a workflow diagram with four steps: Upload, Annotate, Train, and Deploy. Below the diagram, there is a section titled '1. Create Dataset' which explains that YOLOv5 models must be trained on labeled data. Two options are provided: using Roboflow to label and prepare data, or manually preparing the dataset. The page also includes a '2. Select a Model' section and a 'Download on the Google Play' button.

```
git clone https://github.com/ultralytics/yolov5 # clone
cd yolov5
pip install -r requirements.txt # install
```

Train On Custom Data

Upload Annotate Train Deploy

Creating a custom model to detect your objects is an iterative process of collecting and organizing images, labeling your objects of interest, training a model, deploying it into the wild to make predictions, and then using that deployed model to collect examples of edge cases to repeat and improve.

1. Create Dataset

YOLOv5 models must be trained on labelled data in order to learn classes of objects in that data. There are two options for creating your dataset before you start training:

- ▶ Use Roboflow to label, prepare, and host your custom data automatically in YOLO format. **NEW** (click to expand)
- ▶ Or manually prepare your dataset (click to expand)

2. Select a Model

(Refer Slide Time: 36:17)

The screenshot shows the '2. Select a Model' section of the YOLOv5 'Train Custom Data' page. It lists five pre-trained models: Nano (YOLOv5n), Small (YOLOv5s), Medium (YOLOv5m), Large (YOLOv5l), and X-Large (YOLOv5x). Each model is represented by a neural network icon and a table of metrics including file size, inference time, and mAP.

2. Select a Model

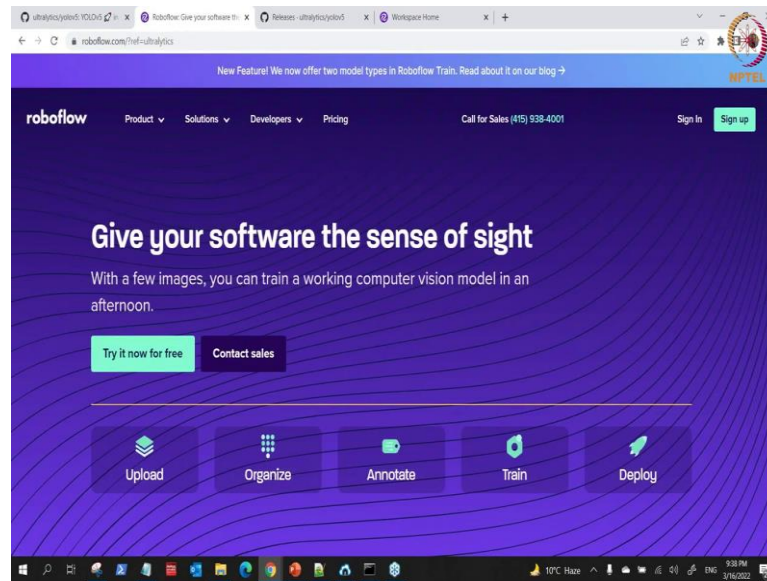
Select a pretrained model to start training from. Here we select YOLOv5s, the smallest and fastest model available. See our README table for a full comparison of all models.

Nano	Small	Medium	Large	X-Large
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB	14 MB	41 MB	69 MB	166 MB
6.3 ms	6.4 ms	8.2 ms	10.1 ms	12.1 ms
28.4 mAP	37.2 mAP	45.2 mAP	48.8 mAP	50.7 mAP

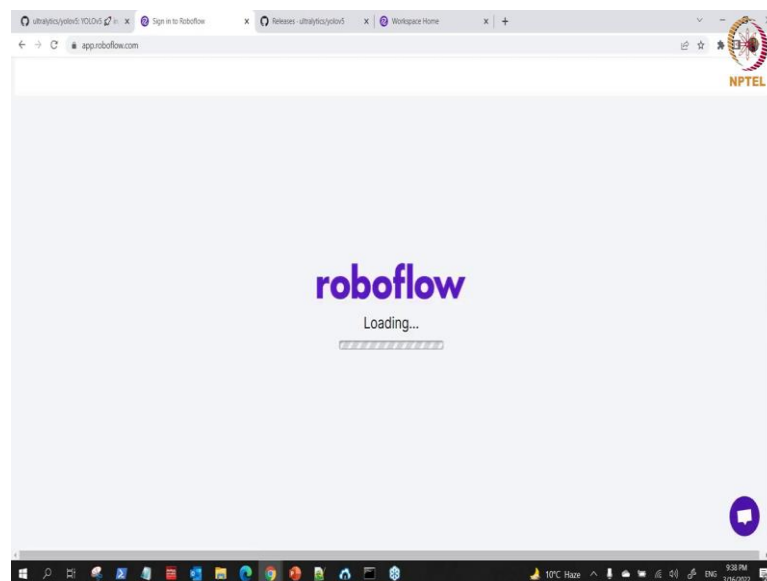
3. Train

Train a YOLOv5s model on COCO128 by specifying dataset, batch-size, image size and either pretrained (recommended), or randomly initialized (not recommended). Pretrained weights are auto-downloaded from the latest YOLOv5 release.

(Refer Slide Time: 36:21)

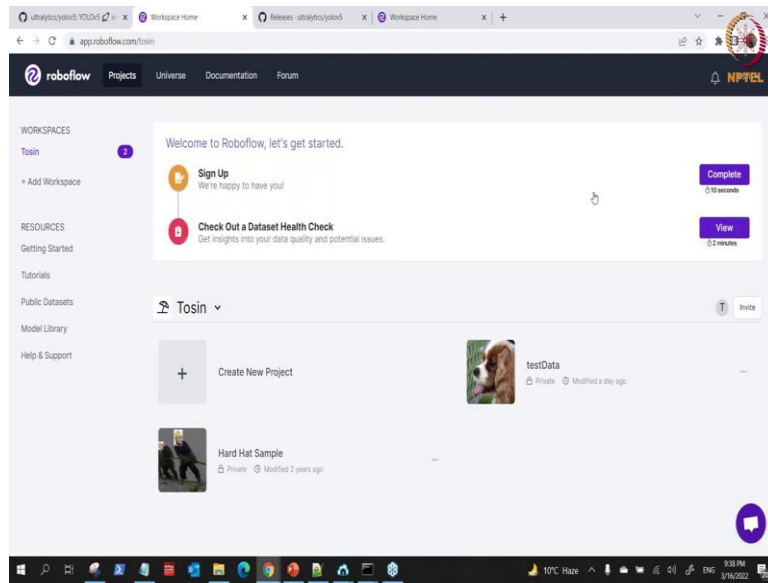


(Refer Slide Time: 36:26)

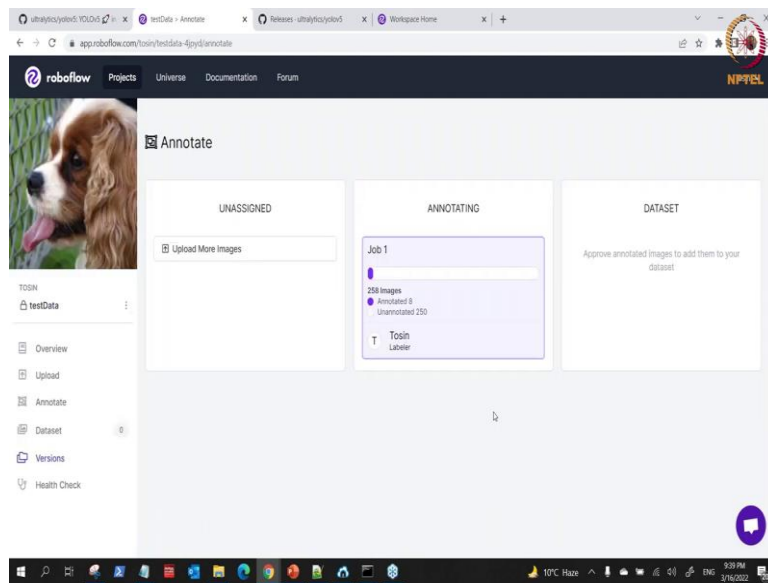


So, this is custom data training, if you want to do custom data training. So, when you get to custom data training then you go to Robo, this is Robo. Then, you need to sign in. I do not know if it allow me to sign in because I have been signing in before, yeah.

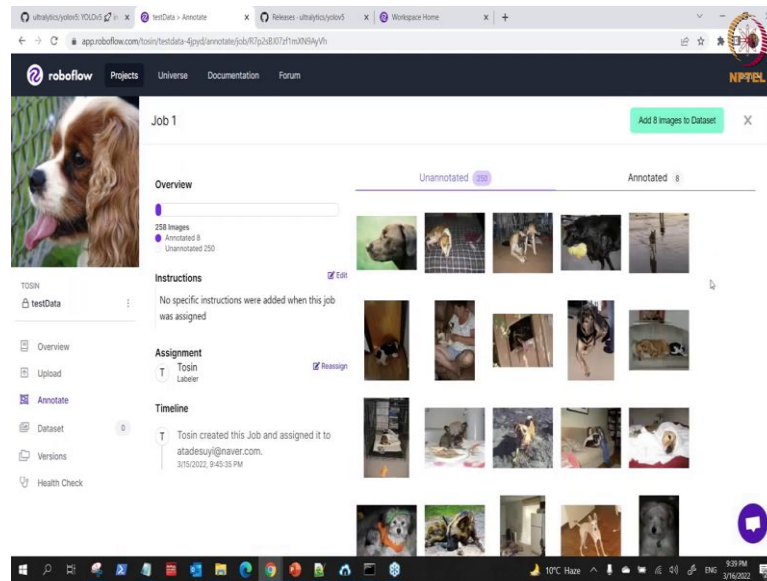
(Refer Slide Time: 36:29)



(Refer Slide Time: 36:41)

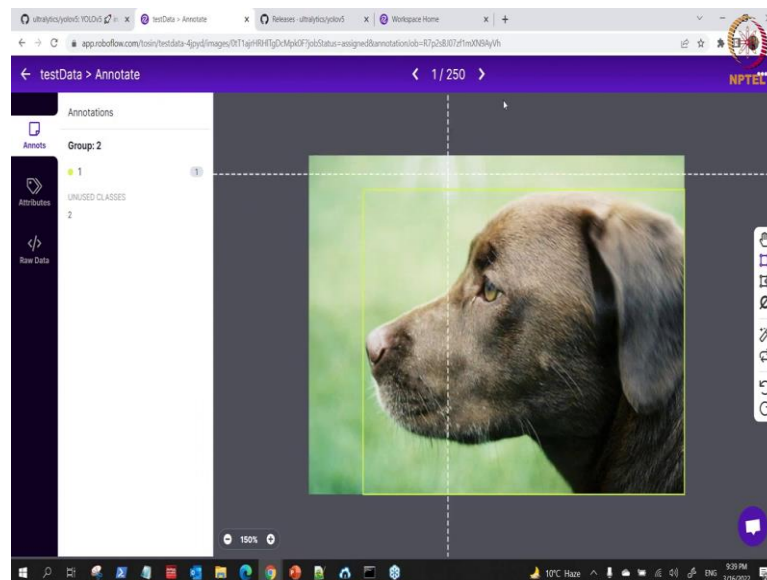


(Refer Slide Time: 36:47)



So, if you do not; if you are not signed up before you can sign up. So, once you sign up what happened is you upload, you create a new project and upload the data. So, when you upload the data for example, the one I have here then you go to annotation to annotate the data the. So, this is the data here.

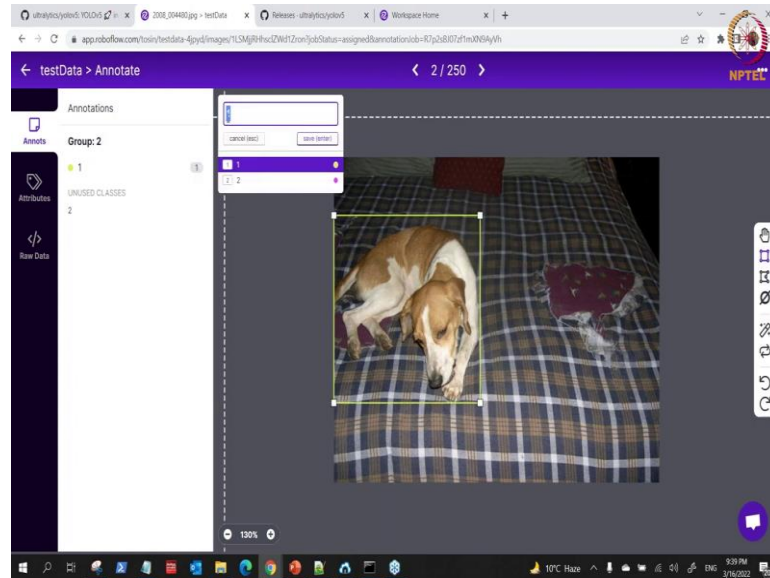
(Refer Slide Time: 36:50)



So, I can click on them you start annotating. You will have specified the number of classes because I am dealing with two classes here. So, how do you annotate this one?

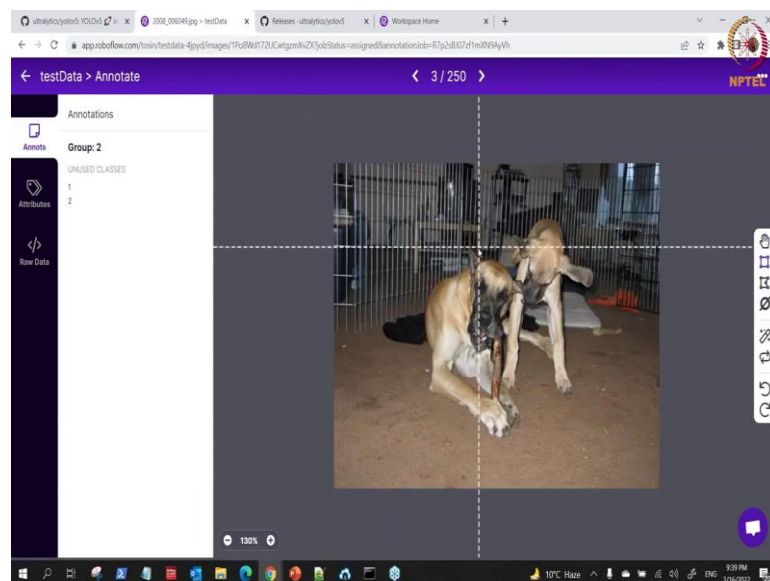
Once you are here, so you just click on the, you just have this. So, I can you draw the you draw the box, it is called box, you draw it there.

(Refer Slide Time: 37:16)

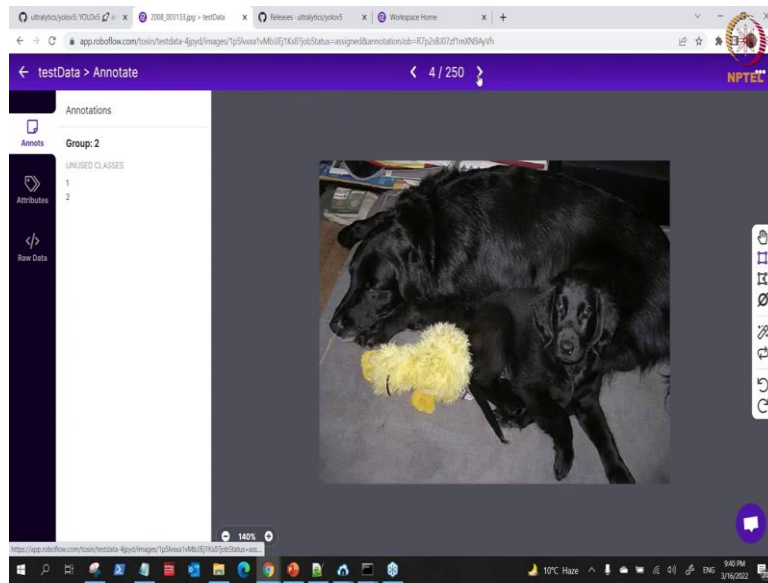


So, once you draw it there, so the class for dog is one you save that. Then you proceed again to the next one. So, you can draw the box on that one as well, draw it there, then you save the class is 1. By the time you get to another object which is not a dog, you continue doing the same way.

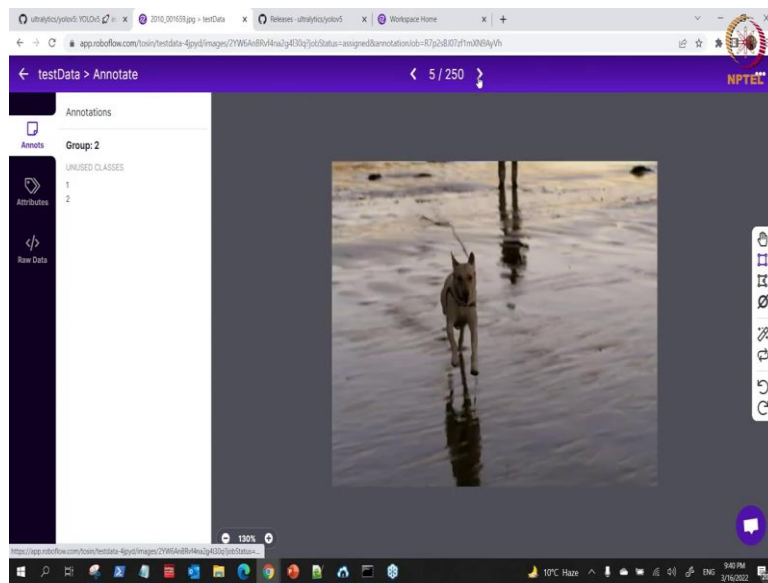
(Refer Slide Time: 37:30)



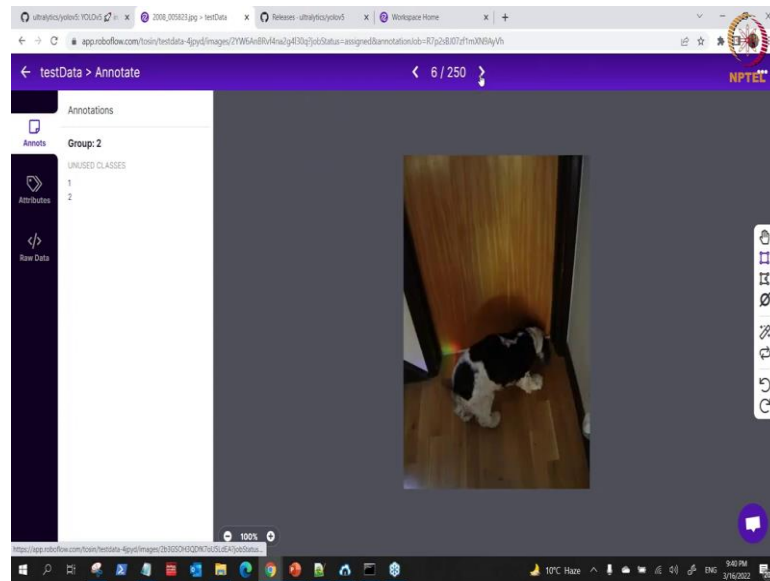
(Refer Slide Time: 37:33)



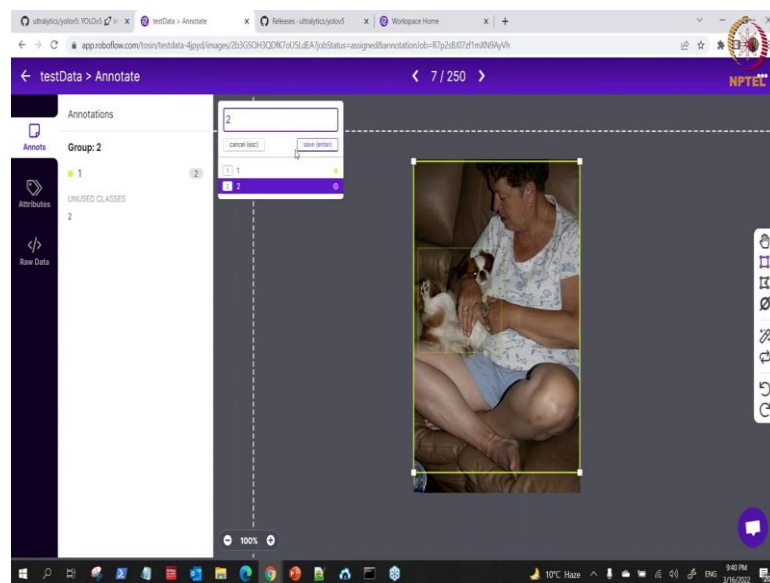
(Refer Slide Time: 37:33)



(Refer Slide Time: 37:35)



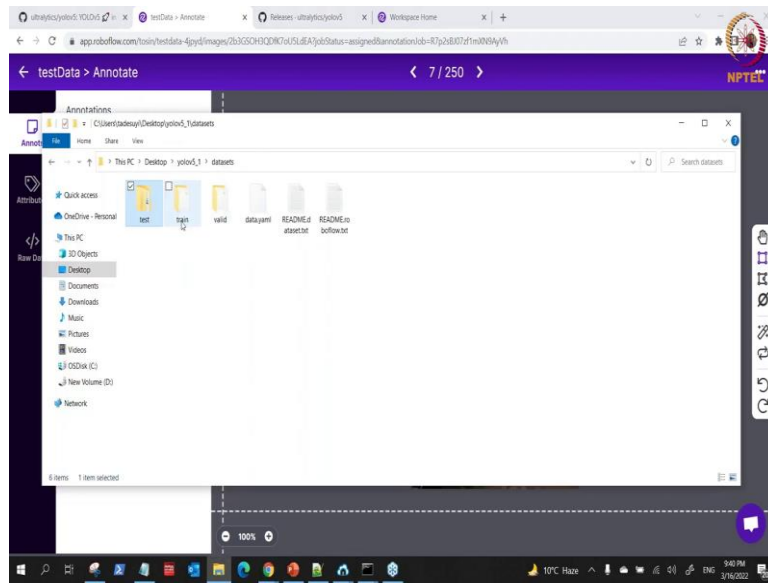
(Refer Slide Time: 37:36)



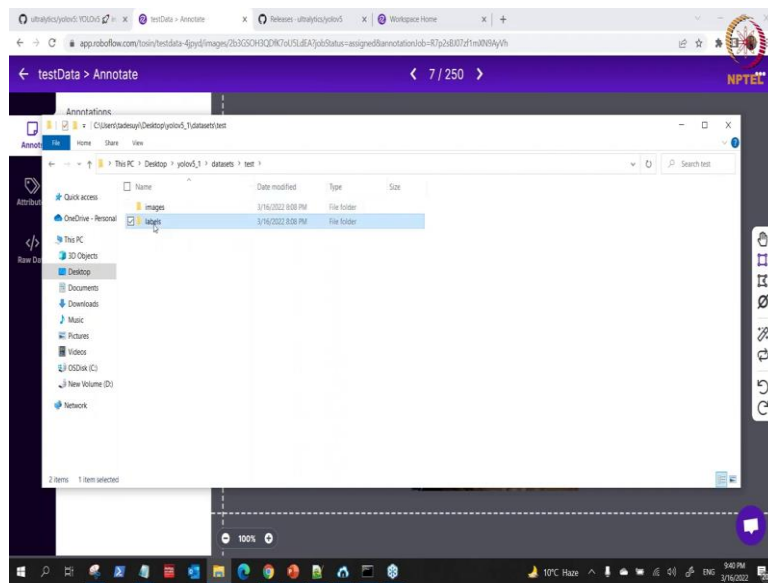
So, let me shift forward and see if I would say the one that include another one, ok. In this case, this is two class. So, what you need to do? I will draw the box for the dog this way the box, the box is can overlap.

Yes, I would say enter that is one for dog, then the class for person now. This is how I would do the one for person, yeah. So, a person which is what the class is 2 and then I can save that. So, that is how you do with Robo there.

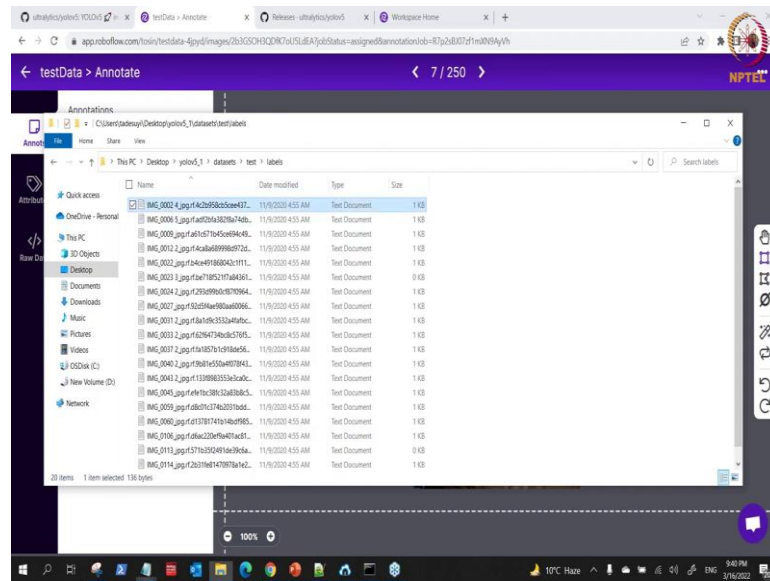
(Refer Slide Time: 38:05)



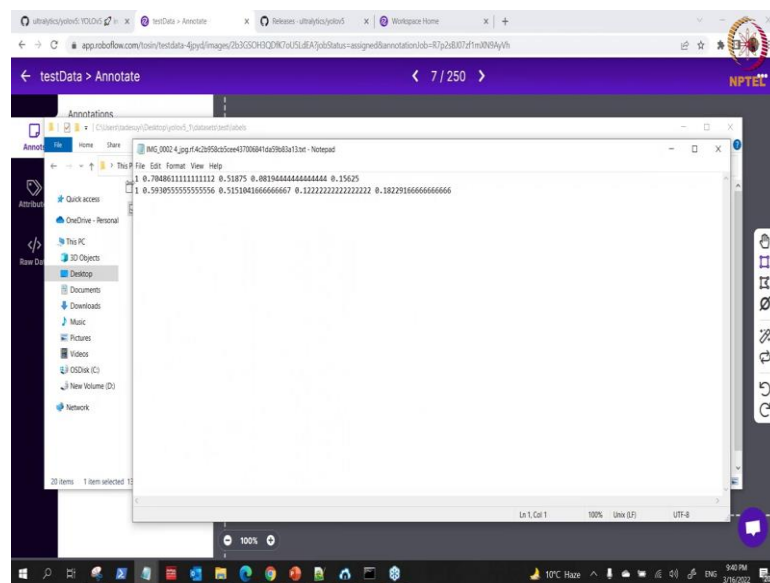
(Refer Slide Time: 38:11)



(Refer Slide Time: 38:14)



(Refer Slide Time: 38:15)

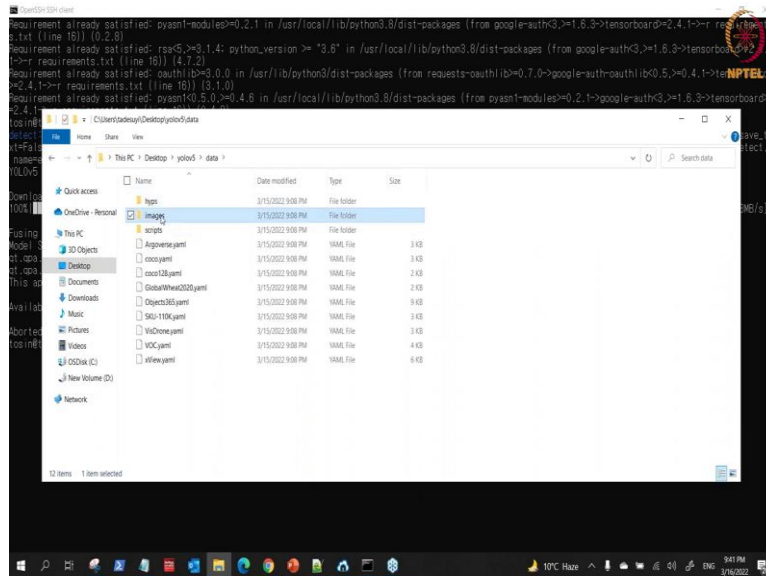


So, let us come back, yeah. I just show you how we came about this how we came about this label, that is how we came about this label. So, it is going to generate it in this way. So, once you draw the box this is the class, this is the for the X-center, Y-center, then the height, the width, and the height of the of the image.

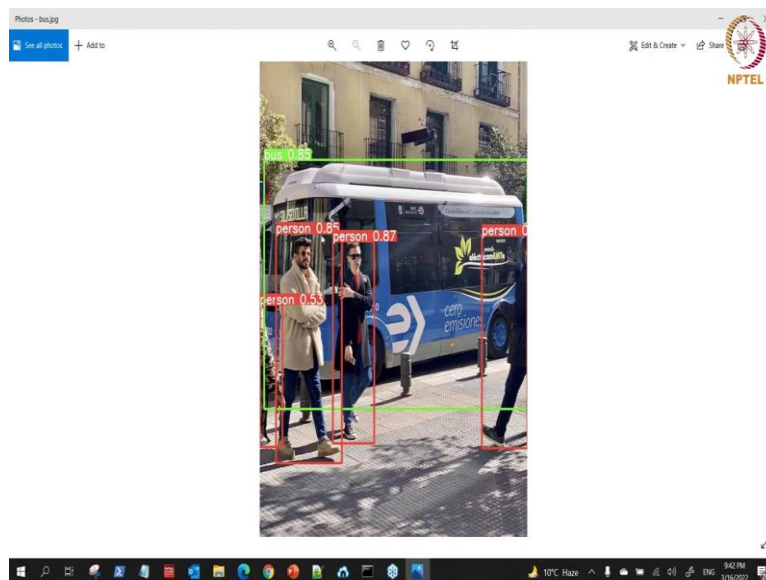
So, we are about to run them up, then before we run up let us just look at this. So, since that one is dumped, so what can we do? I can test with this image here. Let me test with

this. Just do that. We run. So, before I run that is there is an image here. So, in the YOLO which we opened. Where is the YOLO here? We downloaded.

(Refer Slide Time: 39:04)



(Refer Slide Time: 39:10)



So, if we go to data then images there is this image here. So, we wanted to identify this images. I can use webcam. We can use it to identify this image and you will see what will happen to that.

(Refer Slide Time: 39:26)

```

os.makedirs(save_dir, exist_ok=True)
save_conf=True, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=None, name_exp=False, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 v6.1.36-g09fbd2a torch 1.11.0-cu102 CUDA:0 (Quadro GV100, 32508MiB)
Downloading https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5.pt to yolov5.pt
100% |#####| 14.1M/14.1M [00:01:00:00, 11.3M/s]
Fusing layers...
Model Summary: 215 layers, 7225885 parameters, 0 gradients
qt.qpa.xcb: could not connect to display
qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "/home/taosin/.local/lib/python3.8/site-packages/cv2/qt/plugins" even though it was found.
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: xcb.

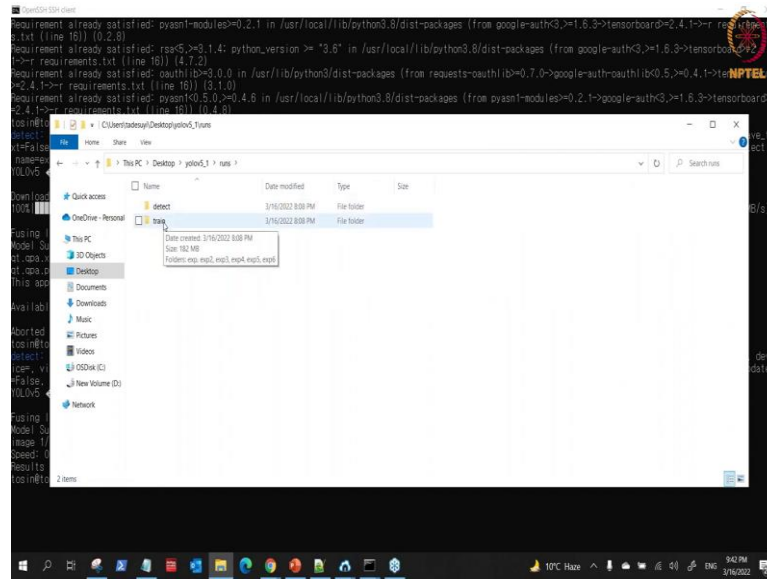
Aborted (core dumped)
taosin@taosin:~/yolov5$ python3 detect.py --source ../yolov5/data/images/bus.jpg --img 640
Model Summary: 215 layers, 7225885 parameters, 0 gradients
Image 1/1 /home/taosin/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, Done (0.004s)
Speed: 0.3ms preprocess, 4.5ms inference, 0.5ms i/o per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp2
taosin@taosin:~/yolov5$ python3 train.py --img 640 --data ../yolov5/datasets/data.yaml --cfg ../yolov5/models/yolov5l.yaml --weights yolov5l.pt --batch 84 --exp
srs 20
taosin@taosin:~/yolov5$ python3 train.py --img 640 --data ../yolov5/datasets/data.yaml --hyp data/hyps/hyp_scratch-low.yaml --epochs 20 --batch-size 64
--imgsz 640 --rect --resume=False --nosave=False --noval=False --noautoanchor=False --evolve=None --bucket=None --cache=None --image_weights=False --device=None --multi_scale=False --single_cls=False --optimizer=SGD --sync_bn=False --workers=8 --project=runs/train --name_exp=False --exist_ok=False --quad=False --cos_lr=False --label_smoothing=0.0 --patience=100 --freeze=0 --save_period=1 --local_rank=-1 --entity=None --upload_dataset=False --bbox_interval=1 --artifact_alias=latest
github up to date with https://github.com/ultralytics/yolov5 v6.1.36-g09fbd2a torch 1.11.0-cu102 CUDA:0 (Quadro GV100, 32508MiB)
hyperparameters: lr=0.01, lr_f=0.01, momentum=0.897, weight_decay=0.0006, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_loss=0.2, anchor_t=4.0, fl_gamma=0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, flip_lr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 v6.1.36 (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5l.pt to yolov5l.pt...
100% |#####| 70.5M/70.5M [00:08:00:01, 11.7M/s]

```

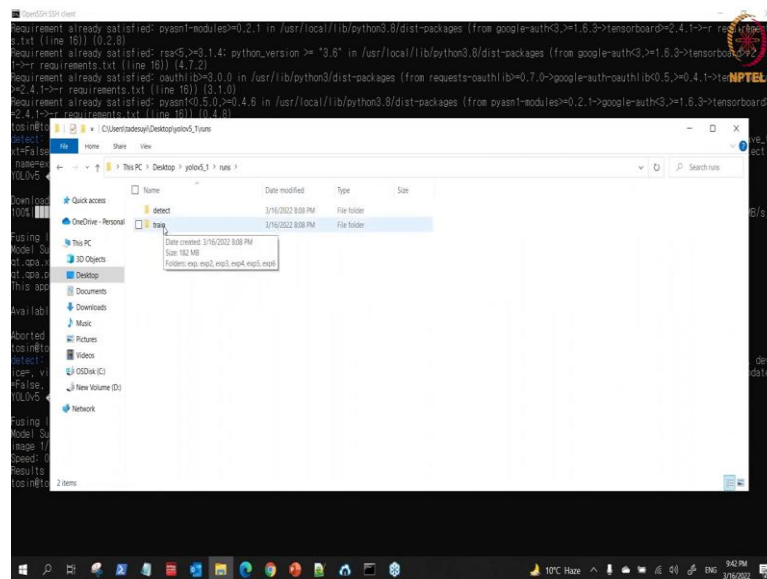
So, and so if I run this from the right track then if we run and save. So, we save it as save it. So, example of that. So, it is on my, it is on my workstation, then what I did is what a put image here somewhere, ok. That the image can be here, which is under the pitch, then I can see where we have bus here, ok, yeah.

So, you can see how it does the classification. It is able to recognize a bus, and this is a bus, this is a person, this is a person again, this also a person, is able to recognize that. Well, if you have a I know I am not using ssh, you can use your; you can use these directs and even with that if you have run that I think I run one before now which is under run dot train, is it train or detects well, yeah, yes.

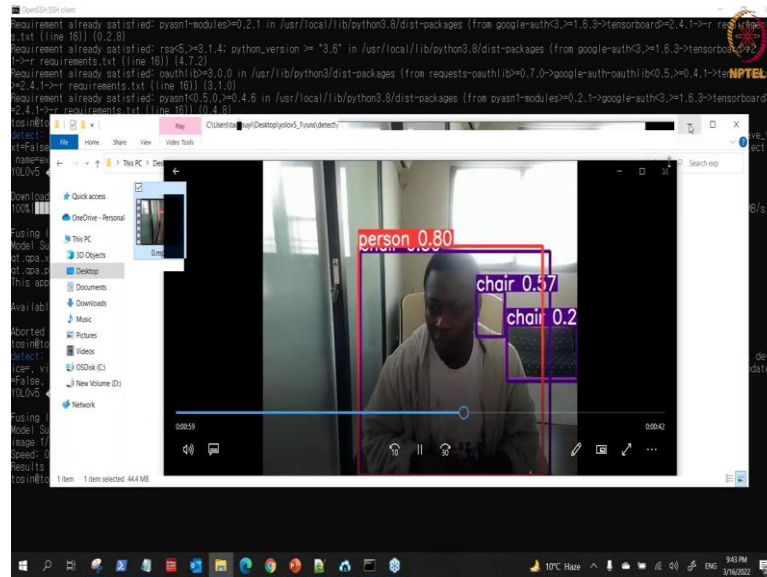
(Refer Slide Time: 40:27)



(Refer Slide Time: 40:30)



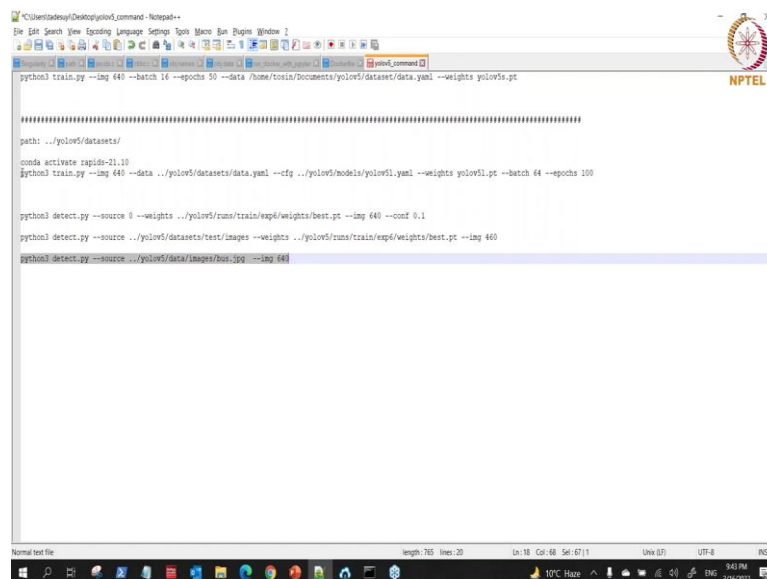
(Refer Slide Time: 40:37)



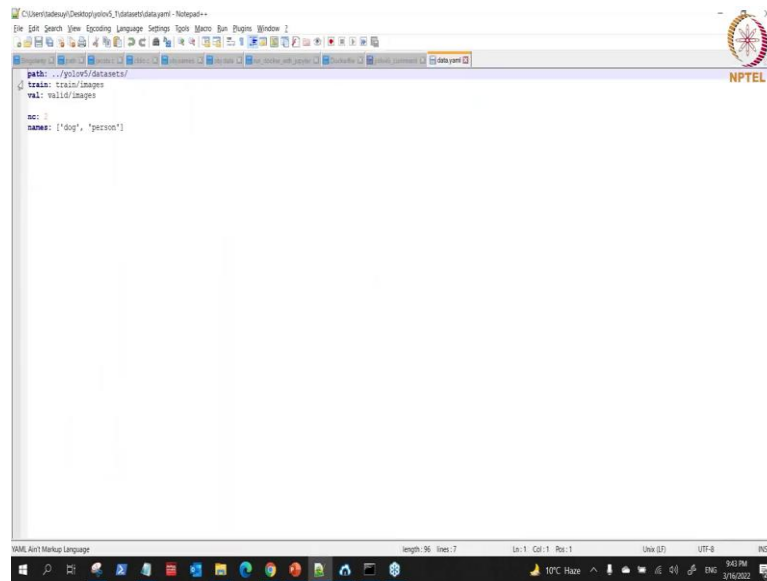
So, this is the way supposed to run you can see. This was a live one I did, not too long. It will be it will be recognizing this is a couch, this is a chair, that is how it will be running. We will be seeing all of that there.

So, now, let us just show let me just show you how do you train the custom. So, for the custom, what we need to do for the custom?

(Refer Slide Time: 41:02)



(Refer Slide Time: 41:30)



```
path: ../yolo5/datasets/
train: train/images
val: valid/images

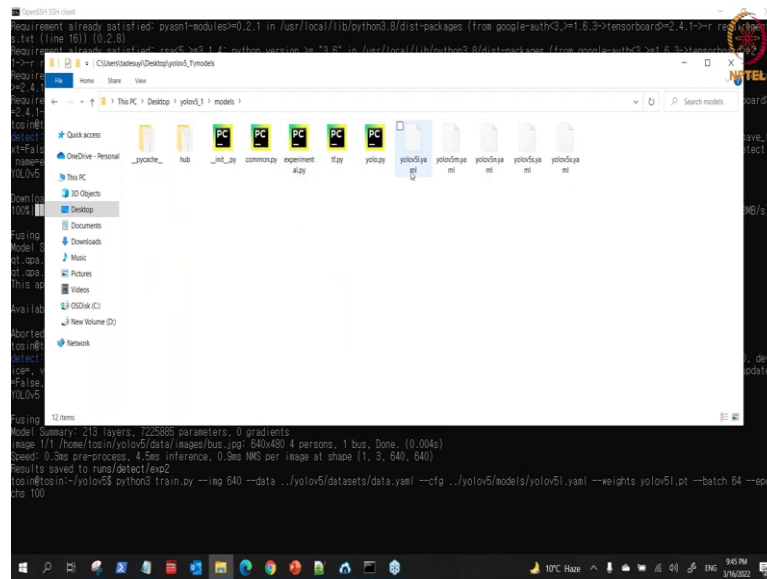
nc: 2
names: ['dog', 'person']
```

Image to train is to then it this, I will explain this to you. Since, we will set everything here into our YOLO; so into the YOLO what you need to do is this dataset is there. You have put your own data. This is your test, this is your train, this is your validation is there after you have download from Roboflow.

Then, this is the yaml file. Let me open this yaml file for you data.yaml file. So, you add this path there, this is YOLO.datasets, YOLO5 datasets. All this will have been there from Roboflow, will have given you all these train paths, validation path, this is number of classes, if detach your data maybe you are dealing with 10 classes what would be here will be 10. And we will list the names of all the what classes here. So, I am just dealing with two here, that is why I have this one here. So, this is how it looks like there.

So, to complete that one, so what I need to do here is, ok. So, I will need to run that here; I need to run that here then come in please working on this to put it there inside my model. Almost done with that, is copies, and bring it here, then just yeah, ok. So, here you have the train.py. This is the image size, this is the date, this is the path to my data which is datasets data.yaml. This is the configuration file. This configuration file is the one for YOLOv5 . The one for YOLOv5 is you can see from this path here model.

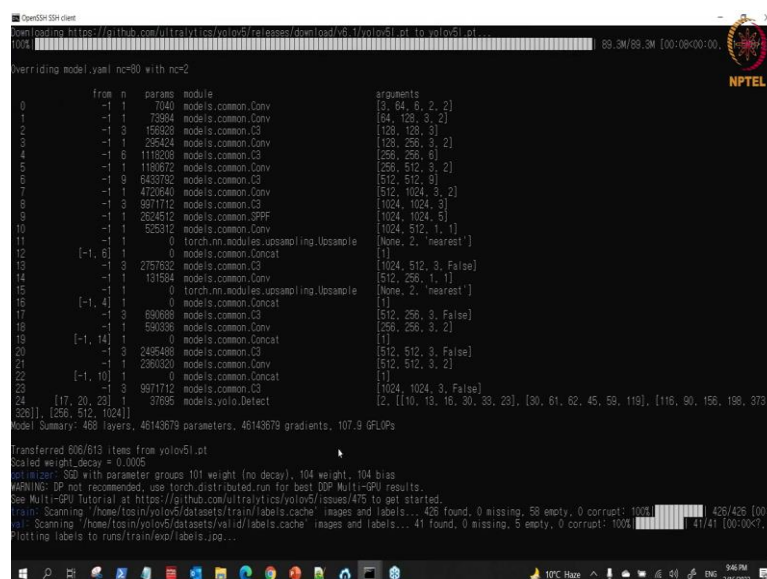
(Refer Slide Time: 43:26)



So, if I come back to model here, this is the path to the file here. Let me increase the view here, so that you can see the view. So, this is the folder here. This is YOLOv5.yaml. That is that for this then the weights. I am using this weights. It is going to download this with by itself. It does not exist 64 batch.

And the number of epoch is just a 100, but 100 would be too much for that. So, let me just say 20, and let us see what happened there.

(Refer Slide Time: 43:55)



So, this is how you would train. So, after you train you can also do inferencing. The same way see is downloading the weights itself or the large.

(Refer Slide Time: 44:02)

```
OpenSSH client
10 -1 1 525312 models.common.Conv [1024, 512, 1, 1]
11 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
12 [-1, 6] 1 0 models.common.Concat [1]
13 -1 3 2757832 models.common.C3 [1024, 512, 3, False]
14 -1 1 131584 models.common.Conv [512, 256, 1, 1]
15 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
16 [-1, 4] 1 0 models.common.Concat [1]
17 -1 3 890968 models.common.C3 [512, 256, 3, False]
18 -1 1 590336 models.common.Conv [256, 256, 3, 2]
19 [-1, 14] 1 0 models.common.Concat [1]
20 -1 3 2485488 models.common.C3 [512, 512, 3, False]
21 -1 1 2380320 models.common.Conv [512, 512, 3, 2]
22 [-1, 10] 1 0 models.common.Concat [1]
23 -1 3 9971712 models.common.C3 [1024, 1024, 3, False]
24 [[17, 20, 23] 1 37895 models.yolo.Detect [2, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 188, 378, 420]], [256, 512, 1024]]]
Model Summary: 466 layers, 46143679 parameters, 46143679 gradients, 107.9 GFLOPs

Transferred 606/613 items from yolo/v5.pt
scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 101 weight (no decay), 104 weight, 104 bias
WARNING: DP not recommended, use torch.distributed.run for best DP Multi-GPU results.
See Multi-GPU tutorial at https://github.com/ultralytics/yolov5/issues/475 to get started.
train: Scanning /home/tesni/yolov5/datasets/train/labels.cache images and labels... 426 found, 0 missing, 53 empty, 0 corrupt: 100% [████████████████████] 426/426 [00:00<
val: Scanning /home/tesni/yolov5/datasets/val/d/labels.cache images and labels... 41 found, 0 missing, 5 empty, 0 corrupt: 100% [████████████████████] 41/41 [00:00<
Plotting labels to runs/train/exp/labels.jpg...

AutoAnchor: 3.43 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✓
Image sizes 640 train, 640 val
Using 8 data loader workers
Logging results to runs/train/exp
Starting training for 20 epochs...

Epoch    gpu_mem  box      obj      cls  labels  img_size
0/19     18.2G   0.1099  0.03175  0.03103  114     640: 100% [████████████████████] 7/7 [00:00<00:00, 1.16s/it]
Class    Images  Labels  P      R      mAP@.5  mAP@.5:95  100% [████████████████████] 1/1 [00:00<00:00, 1.41it/s]
all      41      49      0.0178  0.109  0.01    0.00243

Epoch    gpu_mem  box      obj      cls  labels  img_size
1/19     19.7G   0.08404 0.02365  0.02674  101     640: 100% [████████████████████] 7/7 [00:03<00:00, 1.86it/s]
Class    Images  Labels  P      R      mAP@.5  mAP@.5:95  100% [████████████████████] 1/1 [00:00<00:00, 2.68it/s]
all      41      49      0.408  0.361  0.168  0.0666
```

So, it is running the configuration and this is how you train with your custom data and you start inferencing just the way others, that is the way we have done the initial one as well. So, you can inference with that.

So, it keeps running well. Keeps running, I know you can do the inference yourself as well by following the presentation that we.

Thank you for listening.