FLOWER RECOGNITION

Project by Team 11

Problem Statement

We had a large dataset of flowers of different variety and colour and we wanted to sort them with high accuracy while also decreasing the time and energy spent in the sorting process.

Objective

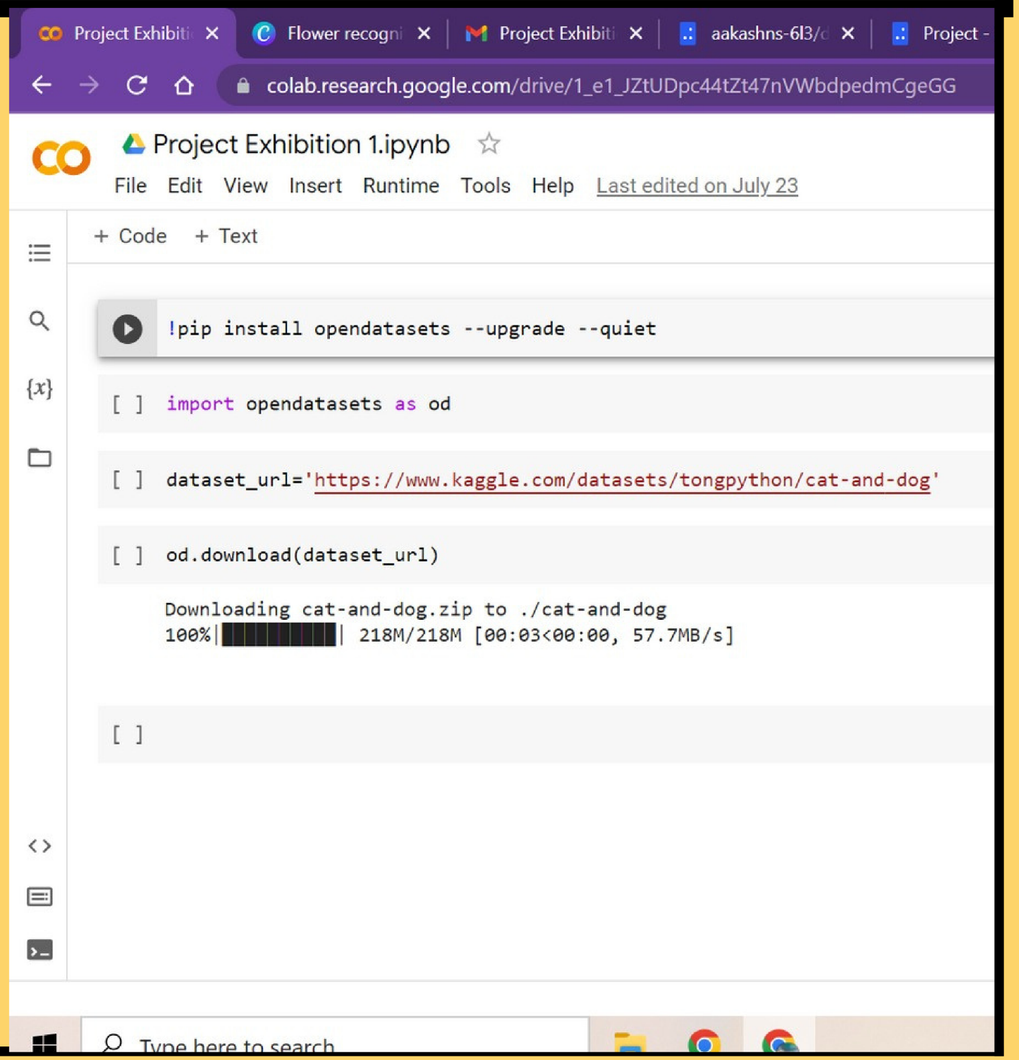
Our objective was to create a neural network (An application of machine learning) to automatically classify flowers.

Hardware Requirement

While projects using ML require a powerful dedicated Graphics Processing Unit (GPU),we leveraged the power of cloud servers to make this task easier for us.Hence any thin and light laptop is enough to execute the code.

Software Requirement

We used Google Collab to write and run our code since it provides access to free Tensor Processing Units (TPU) and Graphical Processing Units (GPU) and this greatly helped improve the efficiency of our project.



Architecture Diagram and Process Flow

The code is divided into the following important parts:

1.Downloading the dataset

2.Importing the dataset into pytorch

3.Configuring GPU and training utilities

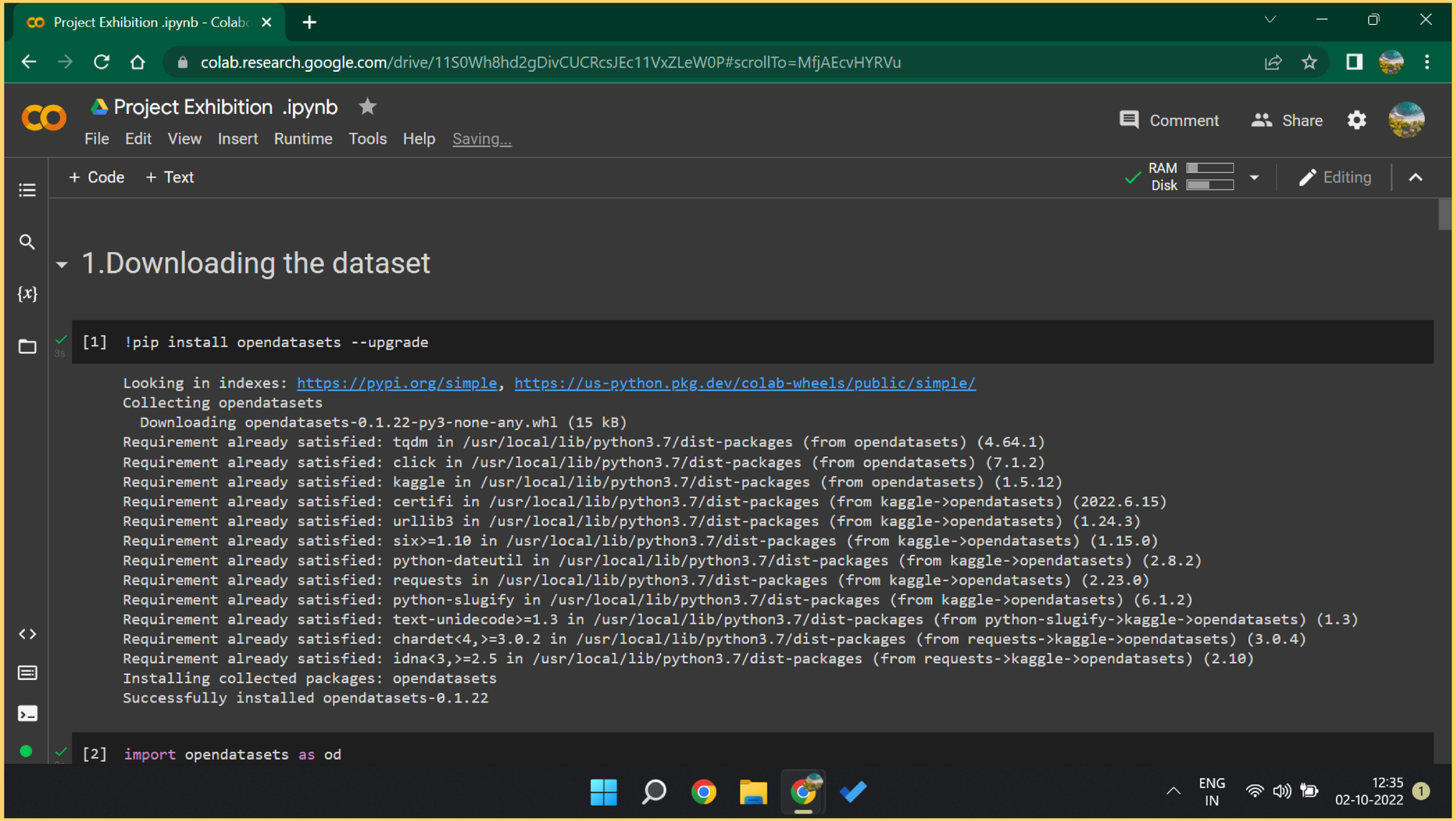
4.Creating the model

5.Training the model

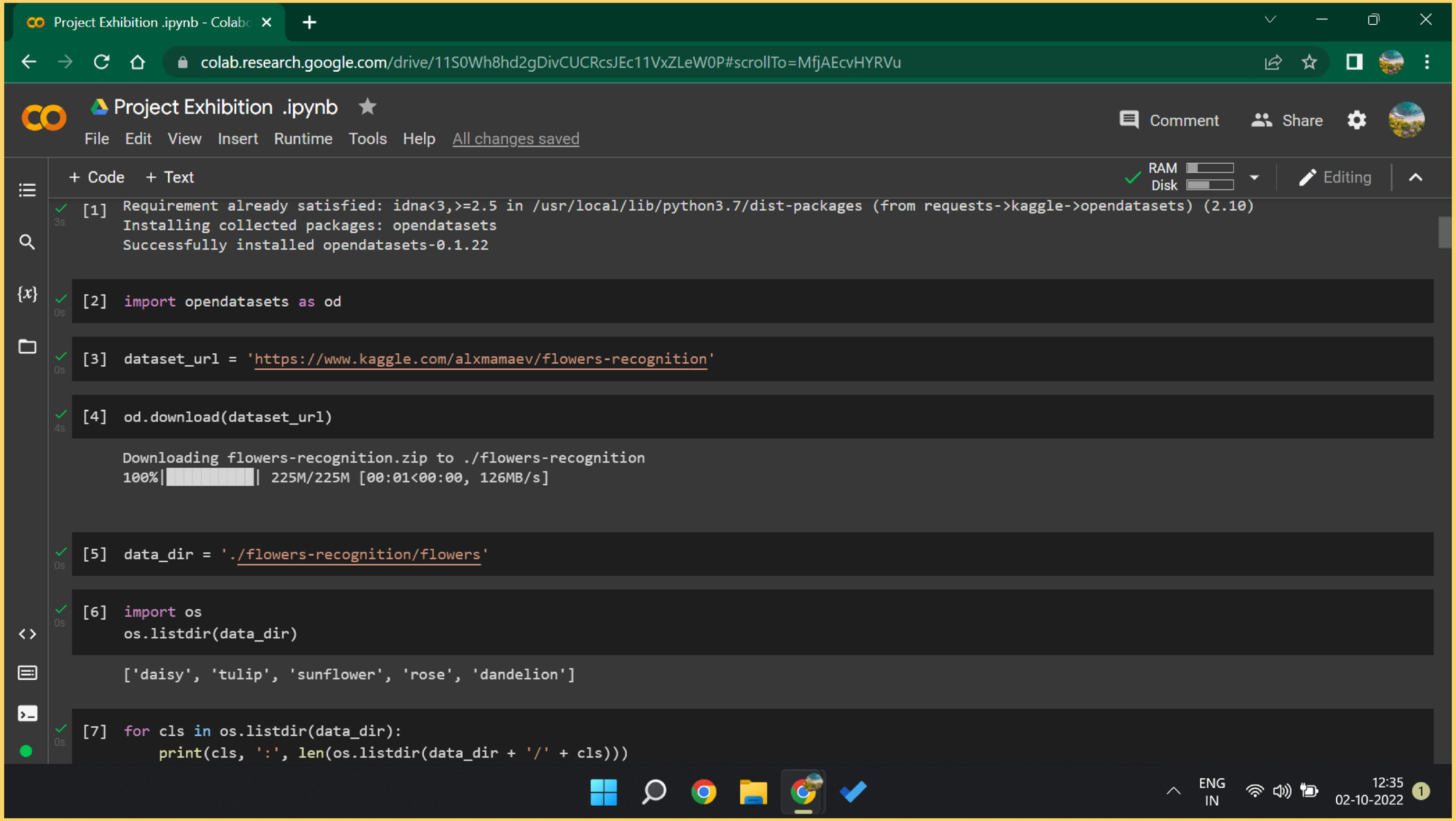
6.Testing the model

IMPORTANT POINTS AND PLACES WHERE THE CODE COULD BE OPTIMIZED FOR THE NEXT PROJECT EXHIBITION HAVE BEEN WRITTEN IN THE FORM OF COMMENTS..

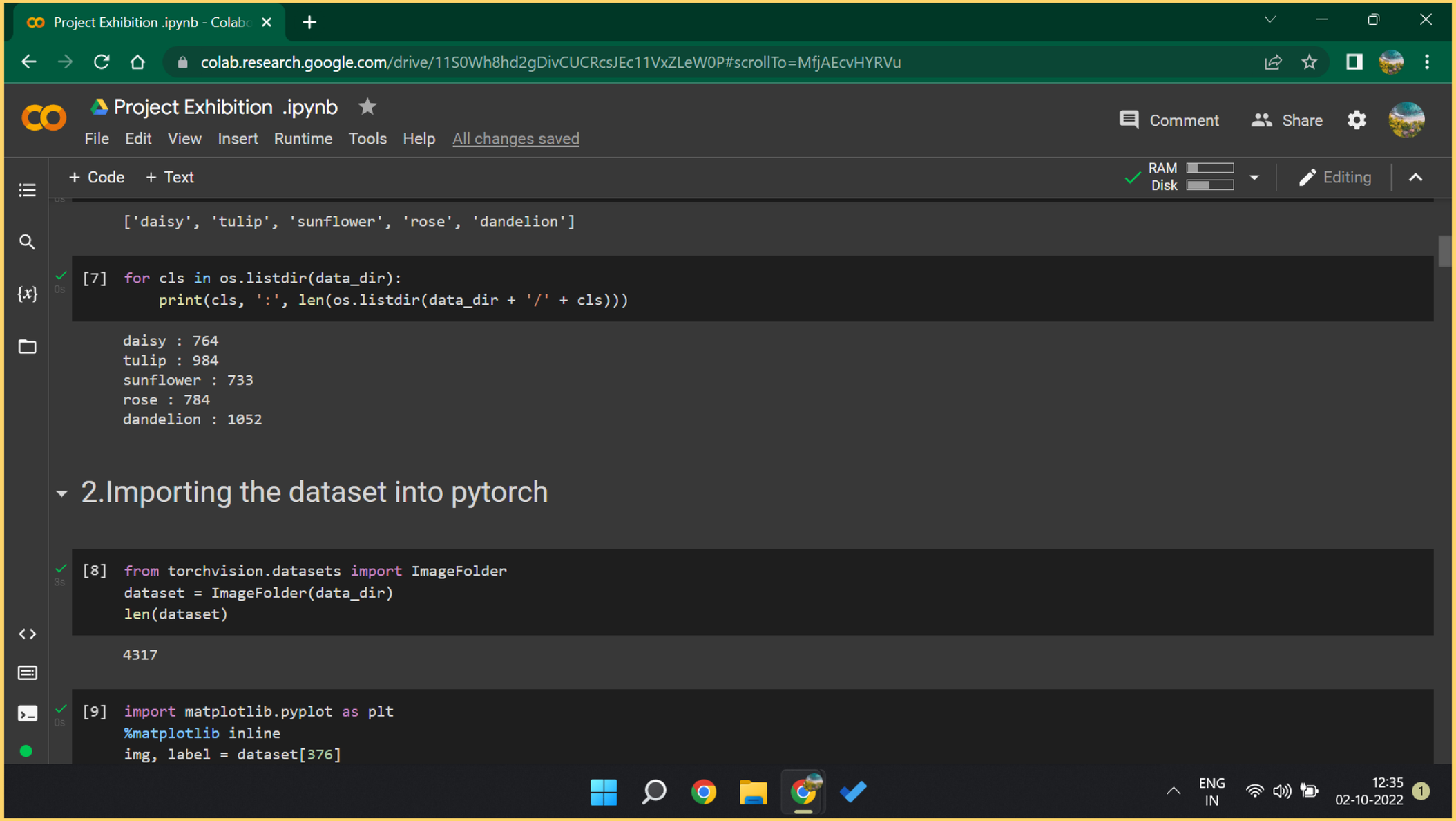
DOWNLOADING THE DATASET



DOWNLOADING THE DATASET



DOWNLOADING THE DATASET



EXPLANATION

In this part we basically downloaded a dataset from kaggle on which we would train our neural network on.

We used the opendataset library to download the dataset directly to the backend servers and after that we used the os library to find the number of flowers of each type.

This pretty much sums up our downloading the dataset part...

IMPORTING THE DATASET INTO PYTORCH

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=MfjAEcvHYRVu

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Editing

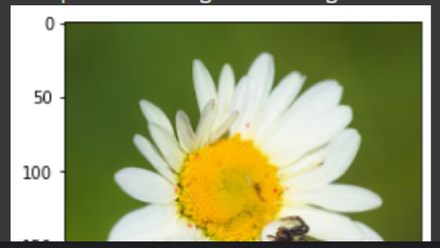
2.Importing the dataset into pytorch

```
[8] from torchvision.datasets import ImageFolder
    dataset = ImageFolder(data_dir)
    len(dataset)
```

```
4317
```

```
[9] import matplotlib.pyplot as plt
    %matplotlib inline
    img, label = dataset[376]
    plt.imshow(img)
```

```
<matplotlib.figure.AxesImage at 0x7f8d633cc4d0>
```



ENG IN 12:35 02-10-2022

IMPORTING THE DATASET INTO PYTORCH

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=MfjAEcvHYRVu

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

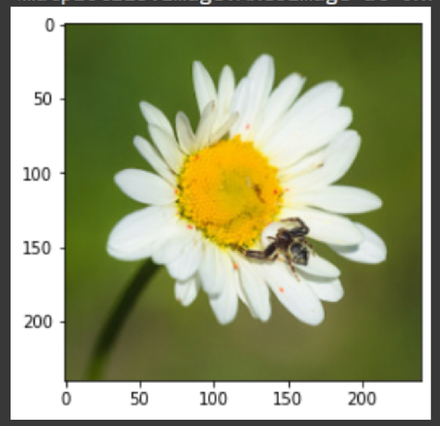
+ Code + Text

RAM Disk

Editing

```
[9] plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f8d633cc4d0>
```




```
[10] a = dataset[756]  #1:Find the image with the least size in the database and resize accordingly---will help get a more accurate result.
     b = dataset[0]
     c = dataset[358]
     print (a,b,c)
```

```
(<PIL.Image.Image image mode=RGB size=240x240 at 0x7F8D63541D10>, 0) (<PIL.Image.Image image mode=RGB size=320x263 at 0x7F8D63541B50>, 0) (<PIL.Image.I
```

ENG IN

12:35 02-10-2022

IMPORTING THE DATASET INTO PYTORCH

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=MfjAEcvHYRVu

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

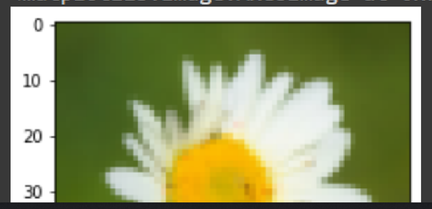
```
[10] a = dataset[756]  #1:Find the image with the least size in the database and resize accordingly---will help get a more accurate result.
     b = dataset[0]
     c = dataset[358]
     print (a,b,c)
```

```
(<PIL.Image.Image image mode=RGB size=240x240 at 0x7F8D63541D10>, 0) (<PIL.Image.Image image mode=RGB size=320x263 at 0x7F8D63541B50>, 0) (<PIL.Image.I
```

```
[11] import torchvision.transforms as tt
     dataset = ImageFolder(data_dir, tt.Compose([tt.Resize((64,64)),
                                                 tt.RandomCrop((64,64)),
                                                 tt.ToTensor()]))
```

```
[12] img, label = dataset[376]
     plt.imshow(img.permute((1, 2, 0)))
```

```
<matplotlib.image.AxesImage at 0x7f8d632ee890>
```



ENG IN 12:35 02-10-2022

IMPORTING THE DATASET INTO PYTORCH

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=MfjAEcvHYRVu

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

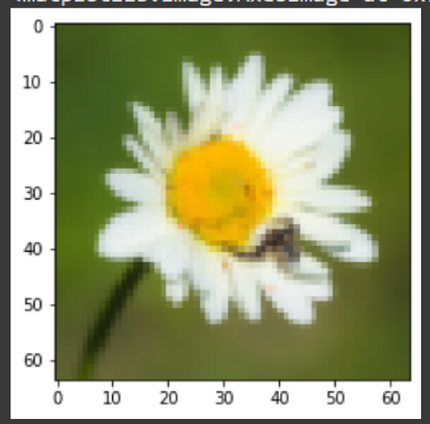
Comment Share

+ Code + Text

RAM Disk Editing

```
[12] img, label = dataset[376]
     plt.imshow(img.permute((1, 2, 0)))
```

```
<matplotlib.image.AxesImage at 0x7f8d632ee890>
```

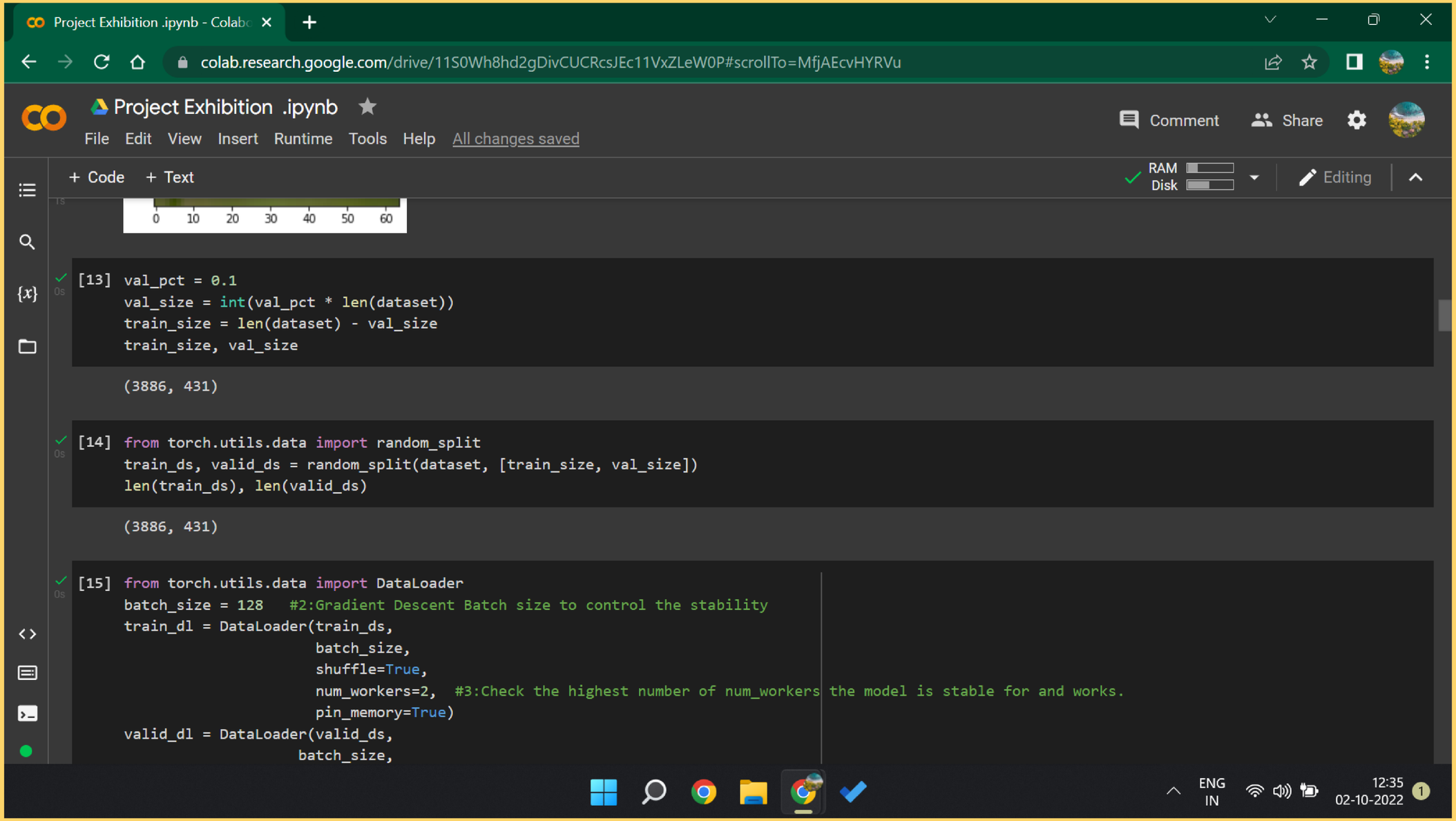



```
[13] val_pct = 0.1
     val_size = int(val_pct * len(dataset))
     train_size = len(dataset) - val_size
     train_size, val_size
```

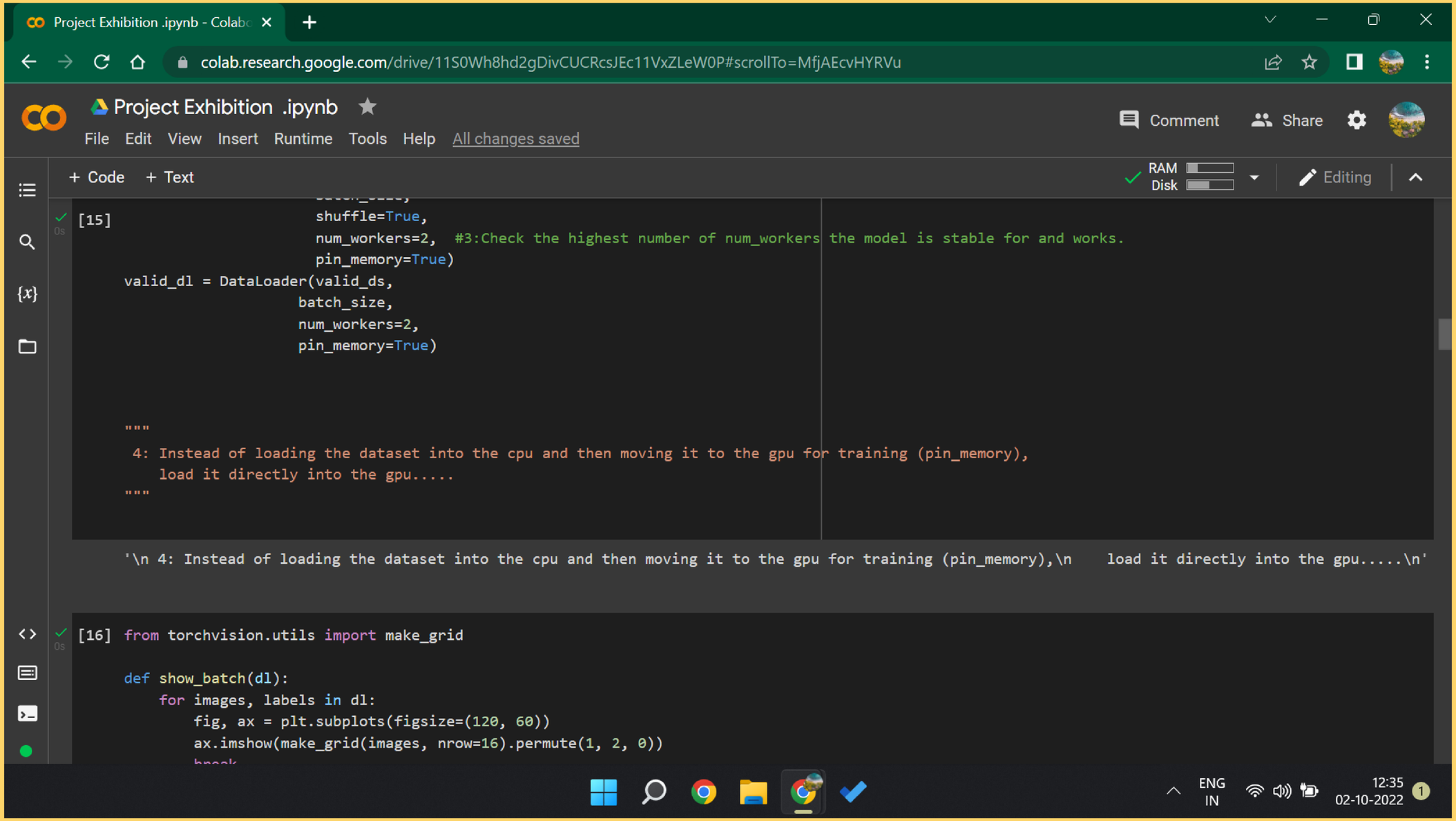
```
(3886, 431)
```

ENG IN 12:35 02-10-2022

IMPORTING THE DATASET INTO PYTORCH



IMPORTING THE DATASET INTO PYTORCH



IMPORTING THE DATASET INTO PYTORCH

Project Exhibition .ipynb - Colabo

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=MfjAEcvHYRVu

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Editing

```
[15] '\n 4: Instead of loading the dataset into the cpu and then moving it to the gpu for training (pin_memory),\n    load it directly into the gpu.....\n'
```

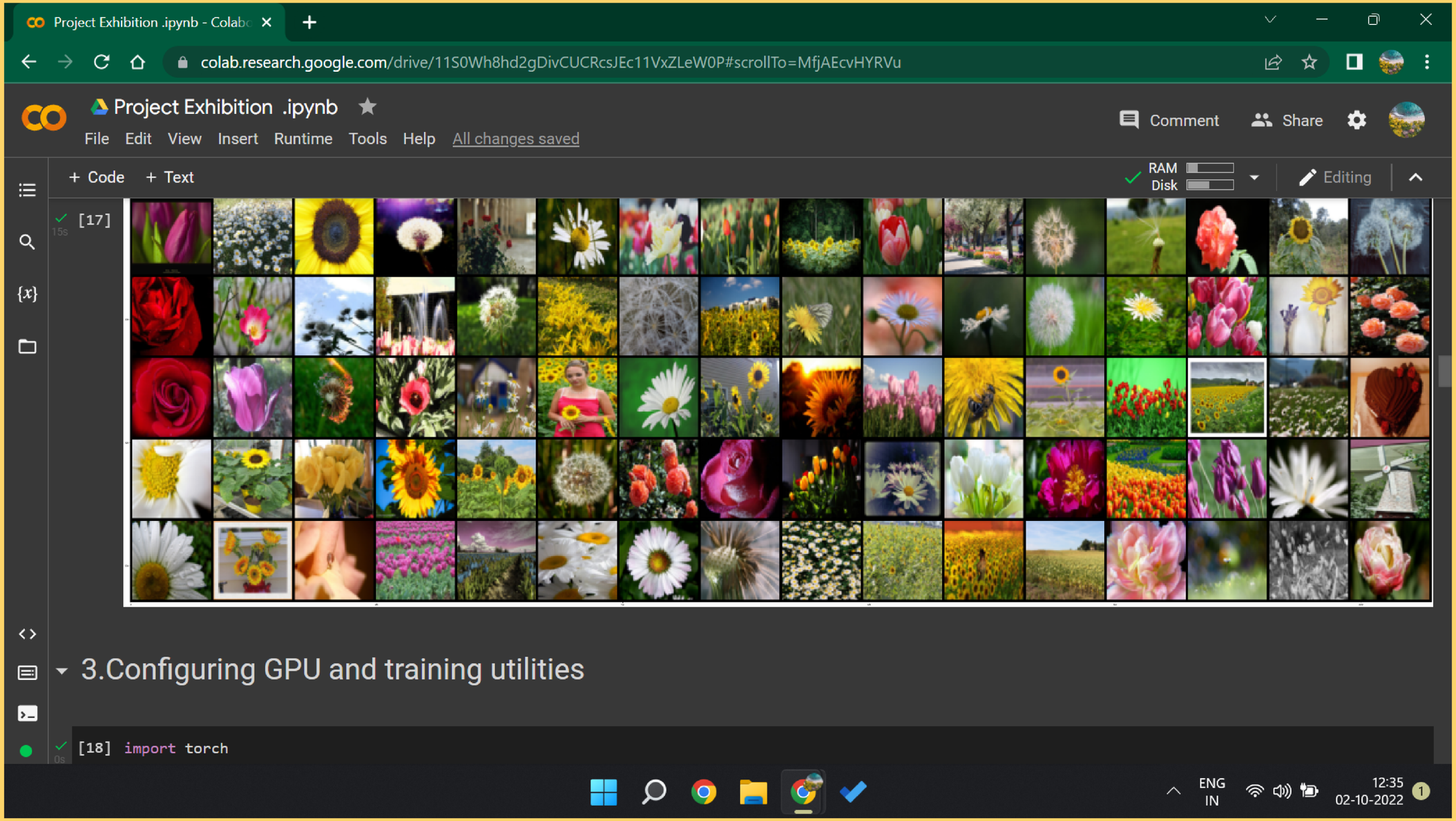
```
[16] from torchvision.utils import make_grid

     def show_batch(dl):
         for images, labels in dl:
             fig, ax = plt.subplots(figsize=(120, 60))
             ax.imshow(make_grid(images, nrow=16).permute(1, 2, 0))
             break
```

```
[17] show_batch(train_dl)
```

ENG IN 12:35 02-10-2022

IMPORTING THE DATASET INTO PYTORCH



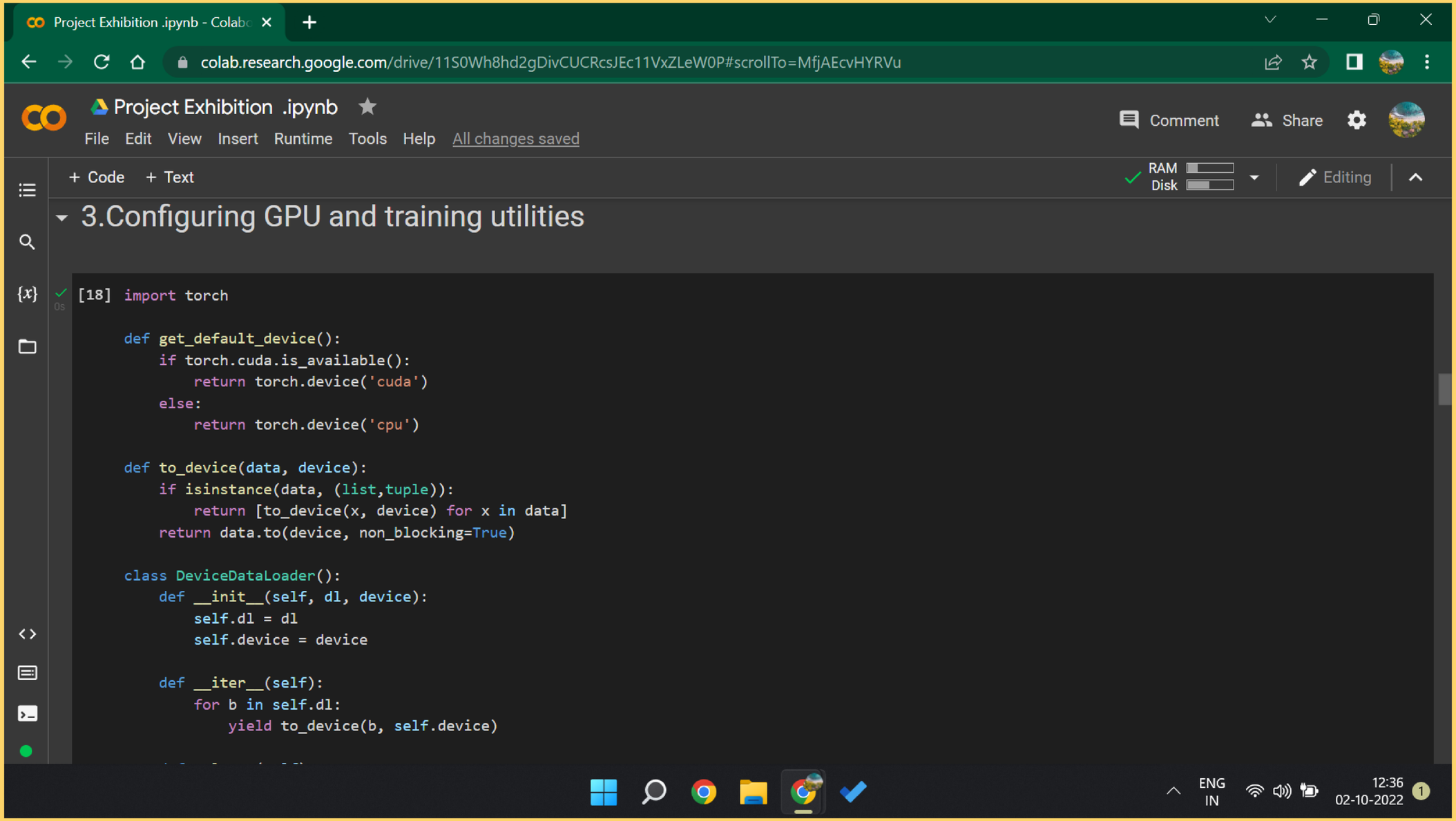
EXPLANATION

In this section of code, we found out the details and size of images in the database using the pytorch library and we found out that they were not consistent and hence we then normalized our dataset.

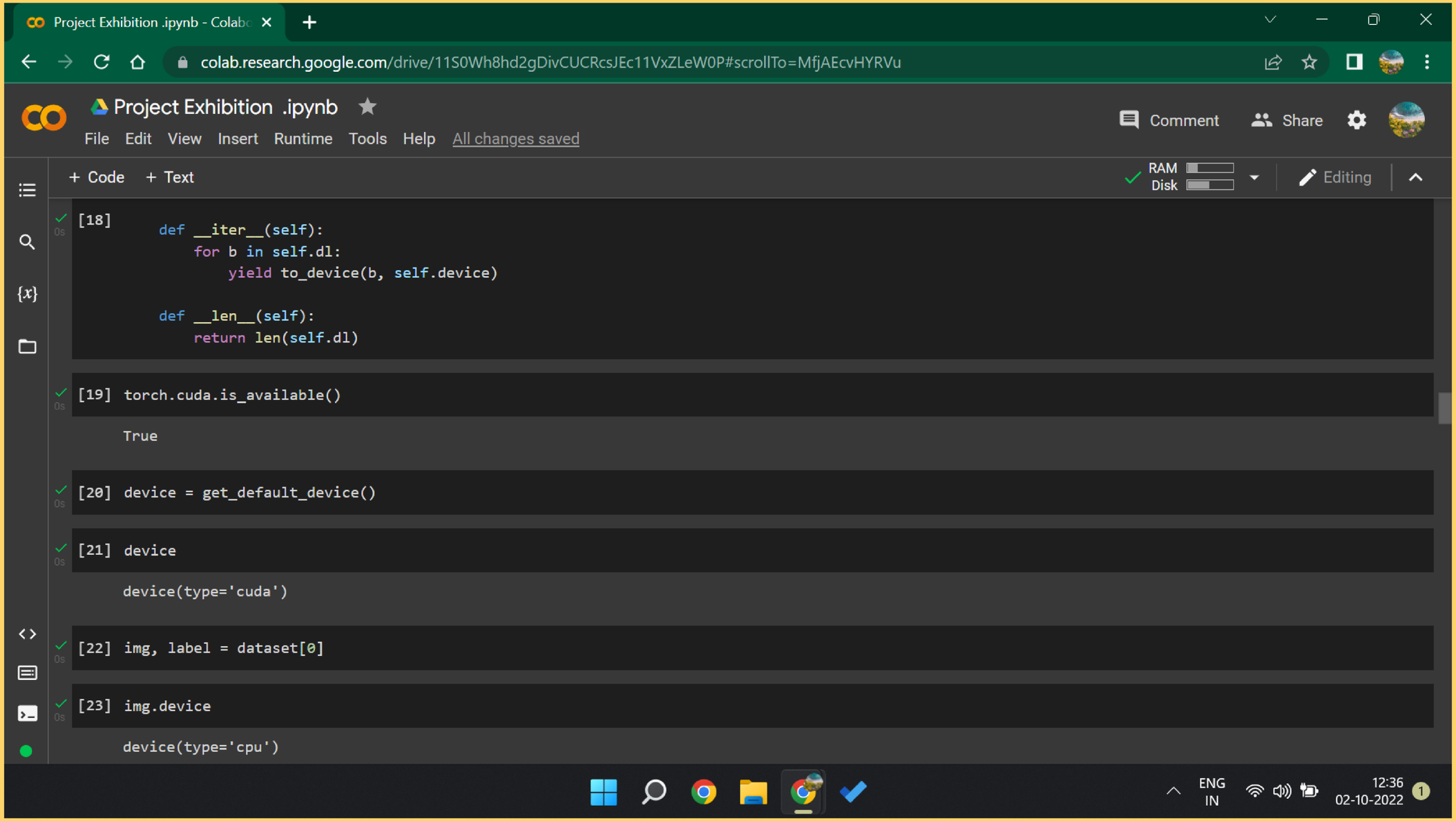
After this, we divided the entire dataset into two parts; one for training and one for validation and created dataloaders for the purpose of dividing the images into smaller bathes for the purpose of training and validation.

All of this helps immensely in increasing the accuracy of the neural network.

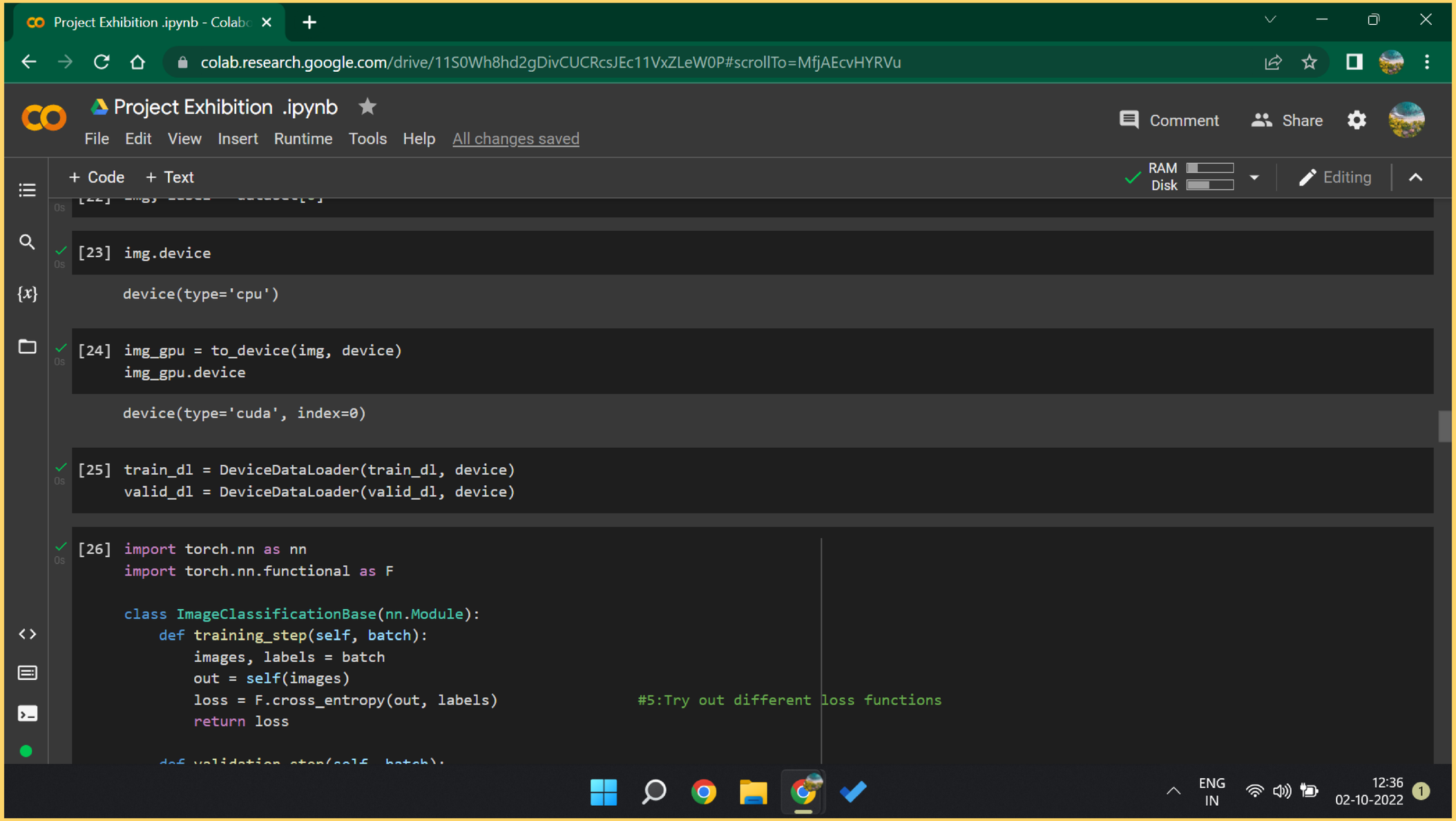
CONFIGURING GPU AND TRAINING UTILITIES



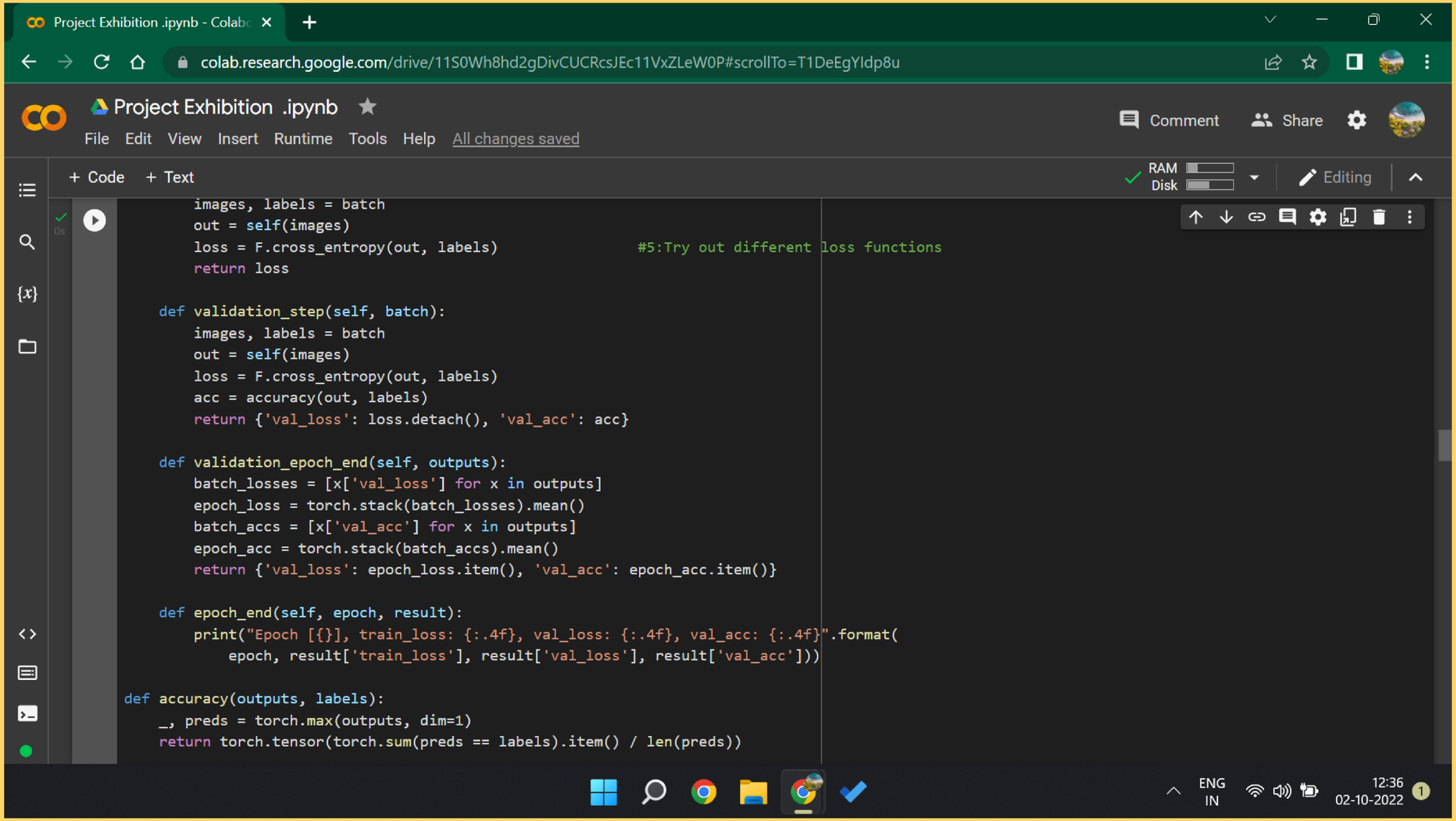
CONFIGURING GPU AND TRAINING UTILITIES



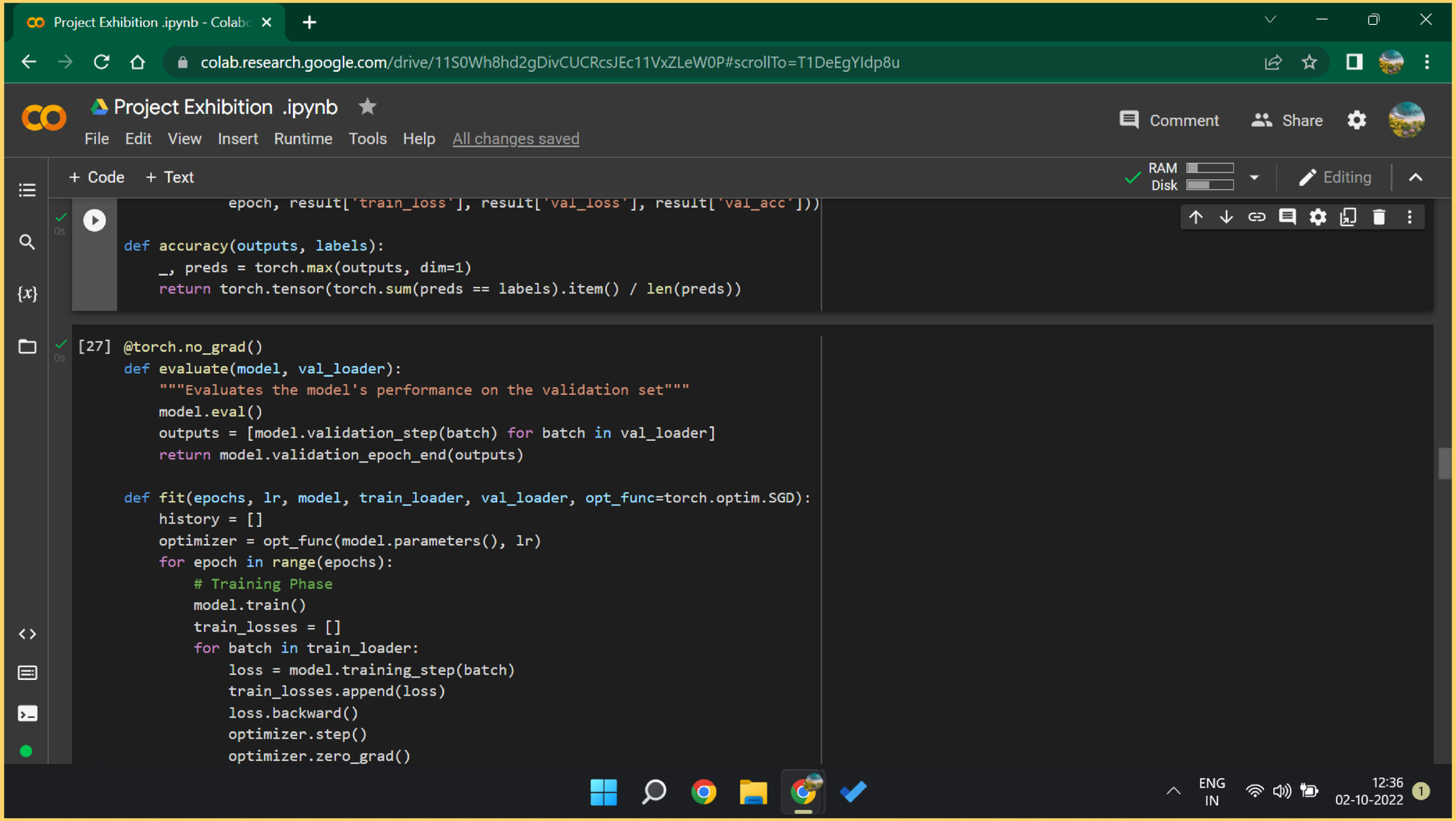
CONFIGURING GPU AND TRAINING UTILITIES



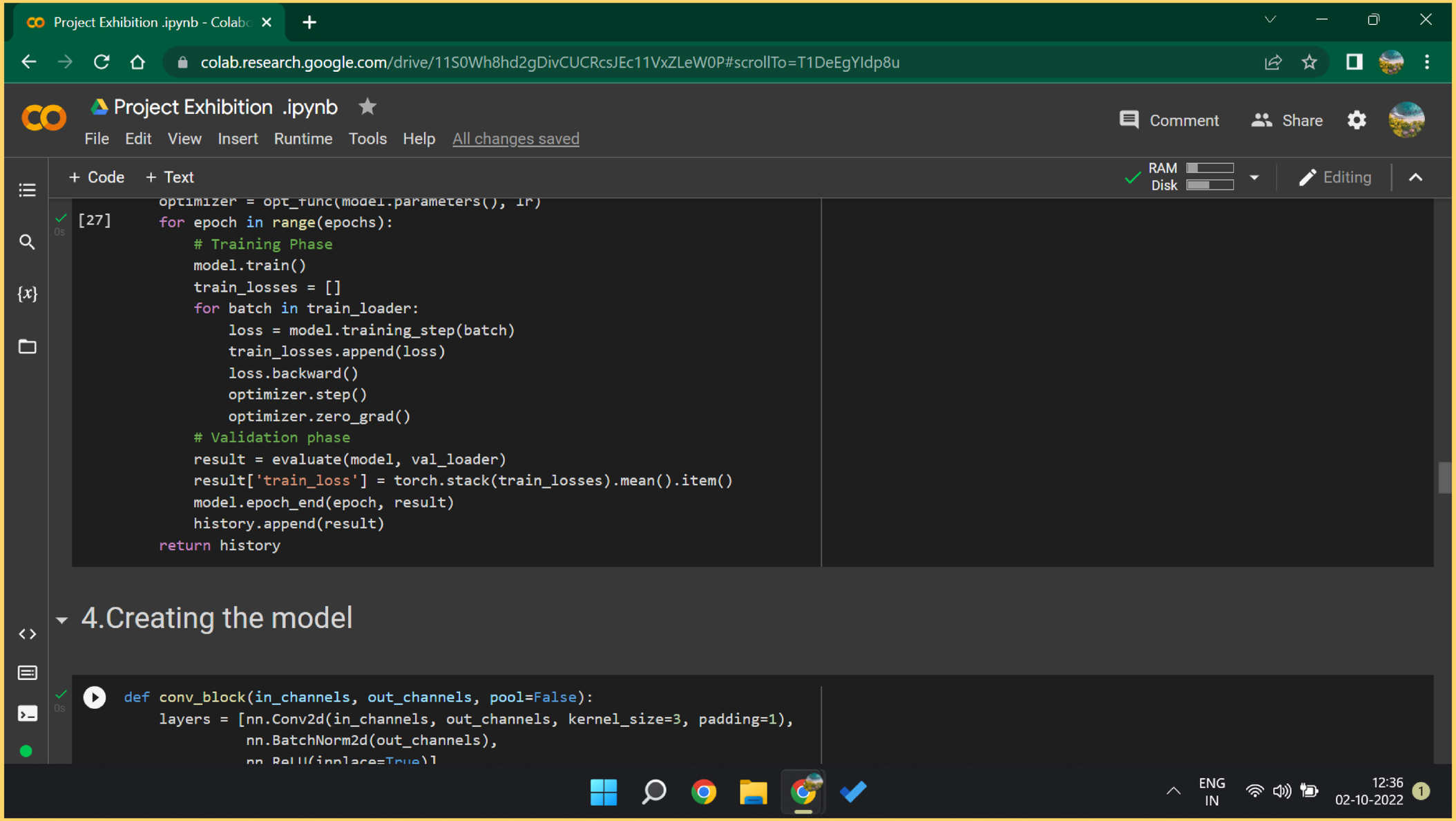
CONFIGURING GPU AND TRAINING UTILITIES



CONFIGURING GPU AND TRAINING UTILITIES



CONFIGURING GPU AND TRAINING UTILITIES



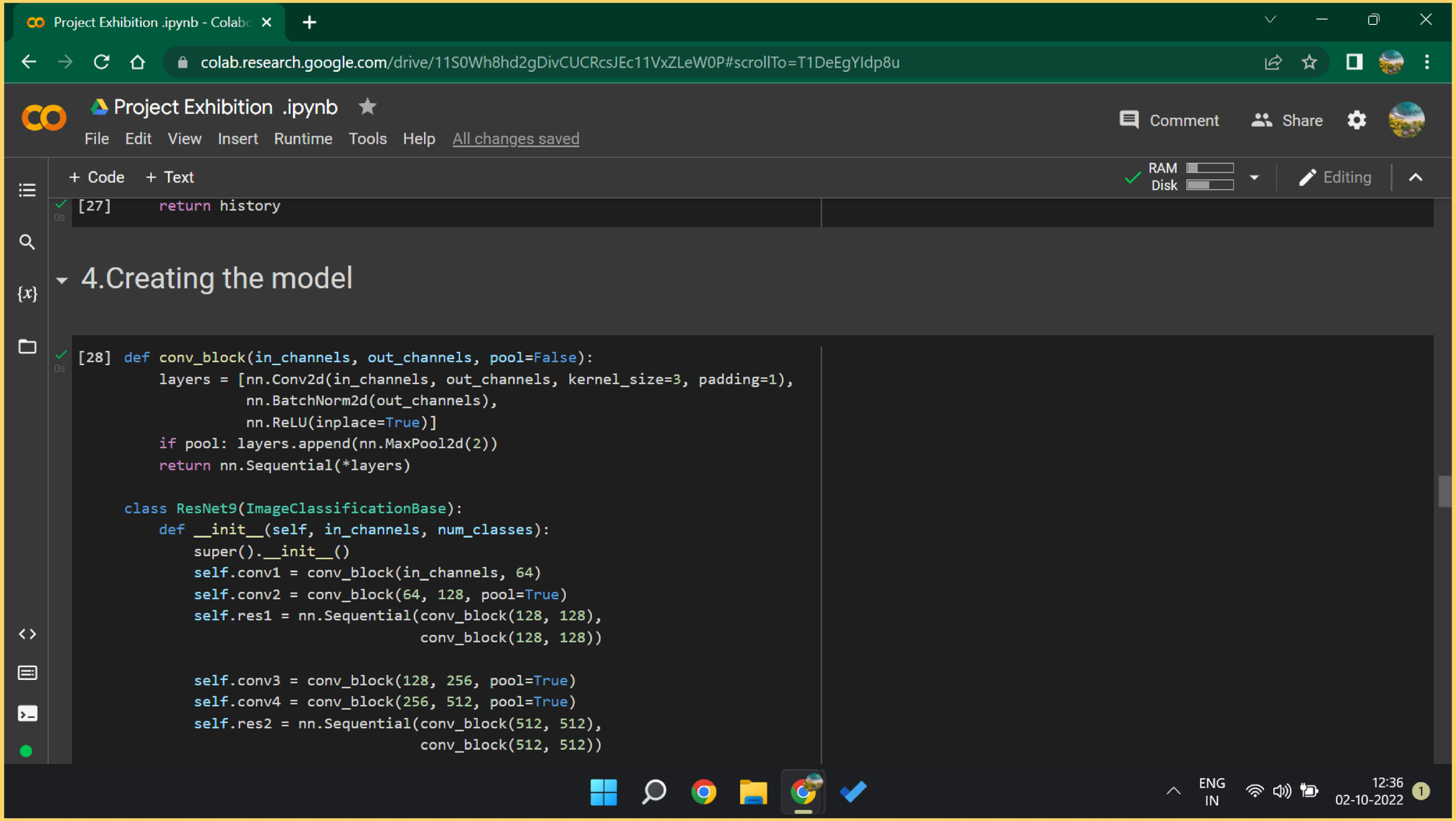
EXPLANATION

This section of code deals with the backend part, we are basically configuring the default device to be selected and used during the operation and also choosing how the data is fed to these devices.

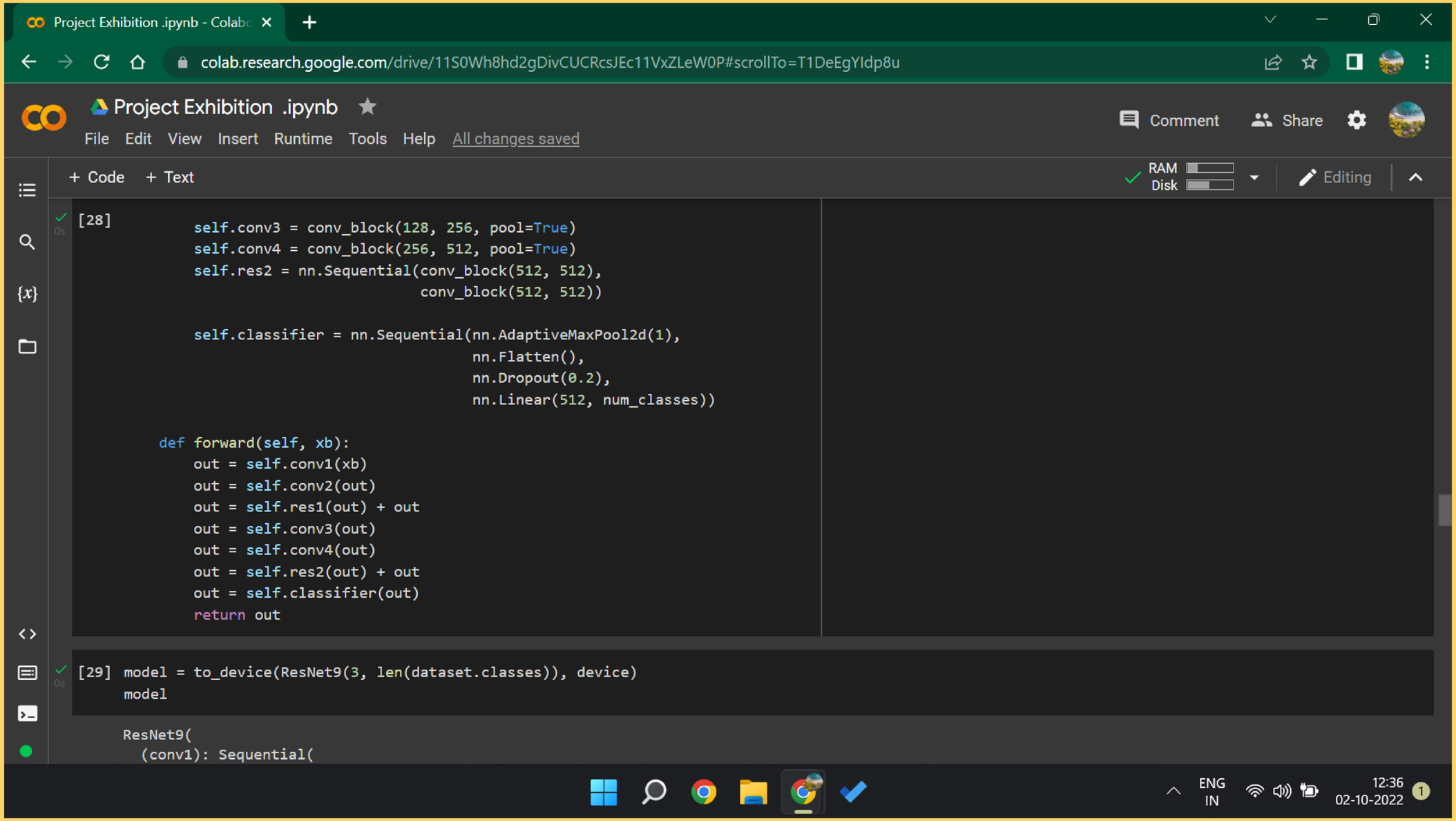
We also define several functions here like epoch,accuracy and history.Apart from functions and modules, we also define select several attributes and properties of the model like the optimizer to be used and the number of iterations that the model has to run.

This section basically controls the efficiency and the time taken by the model to learn.

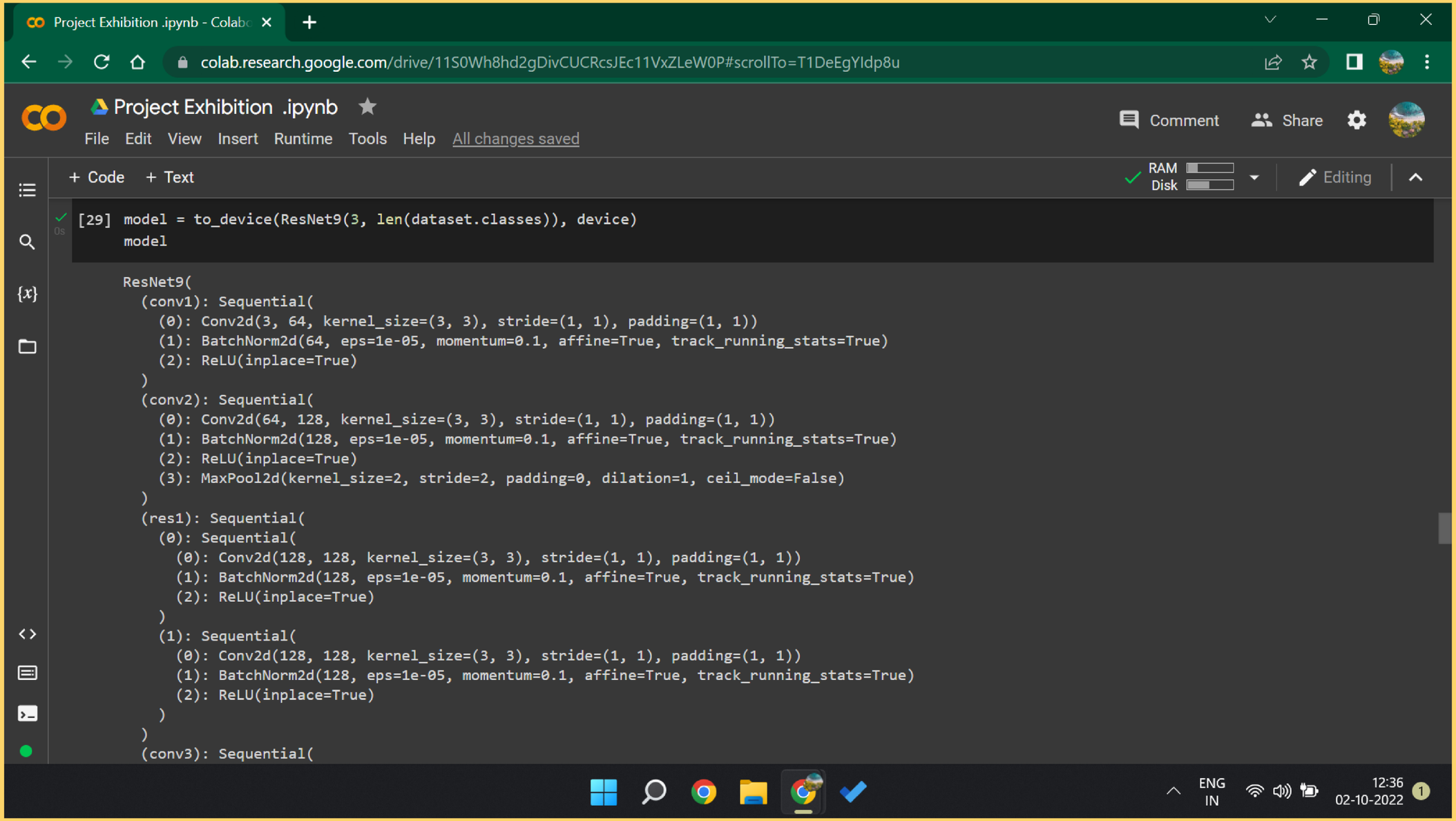
Creating the model



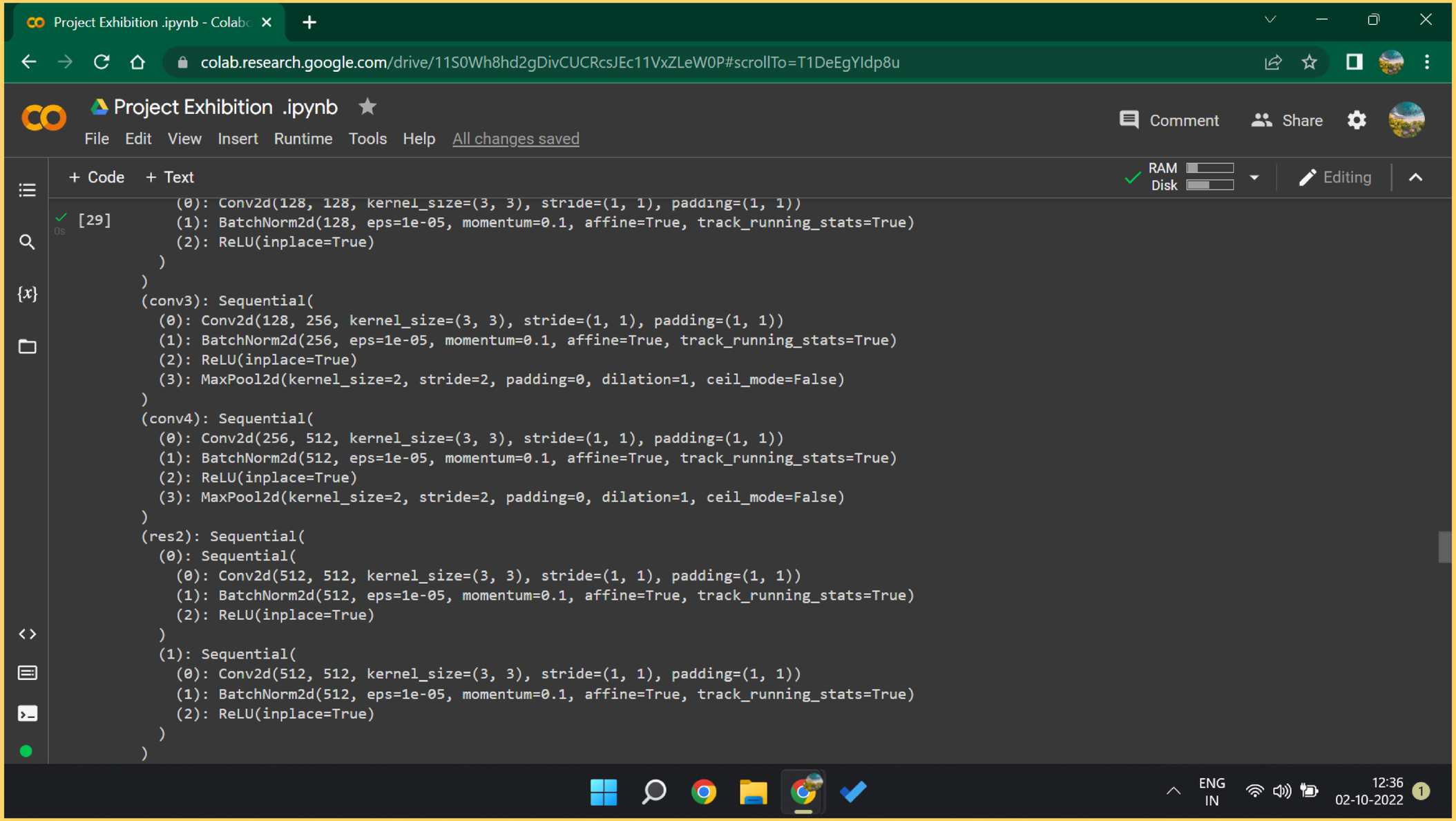
CREATING THE MODEL



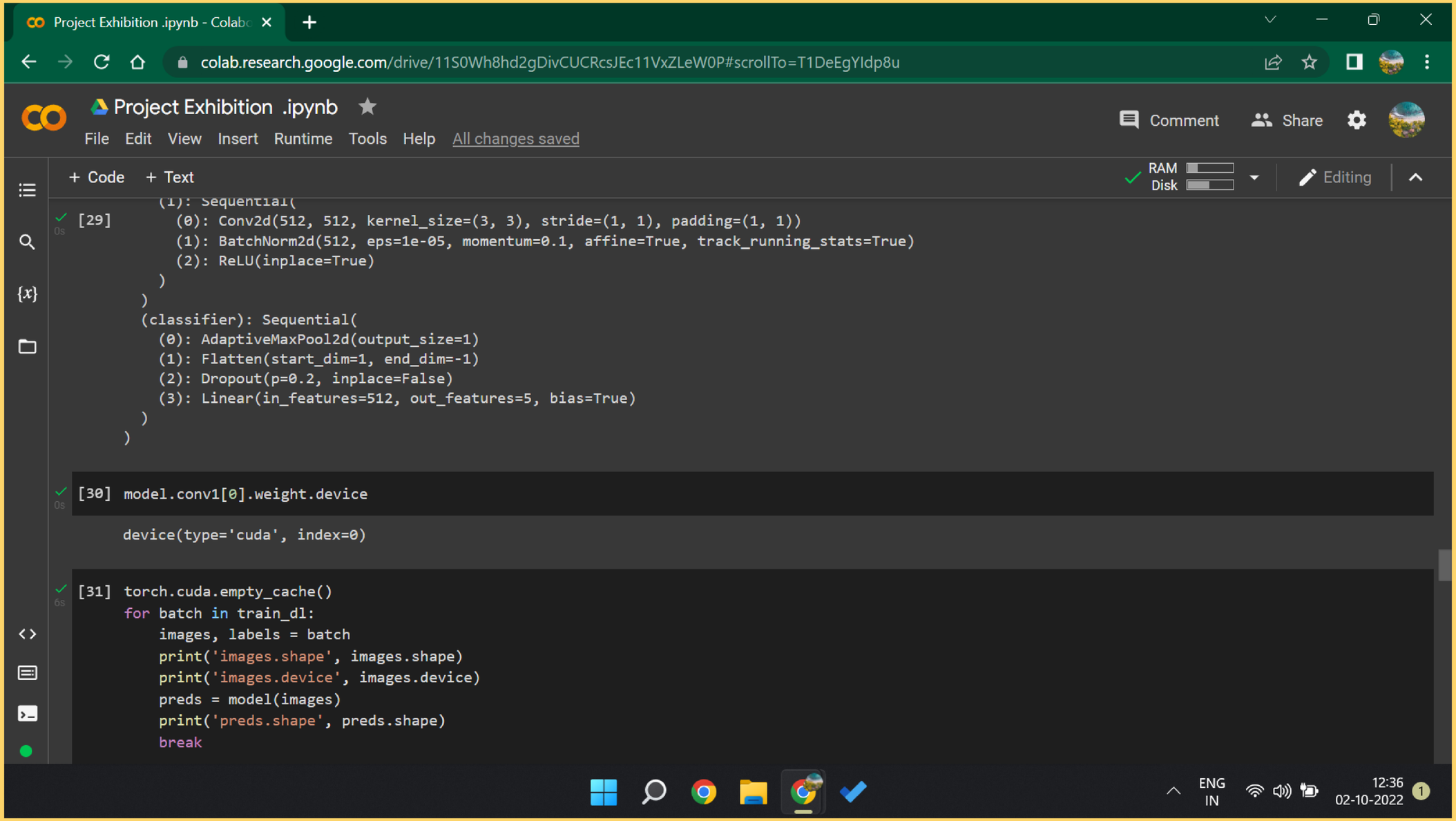
Creating the model



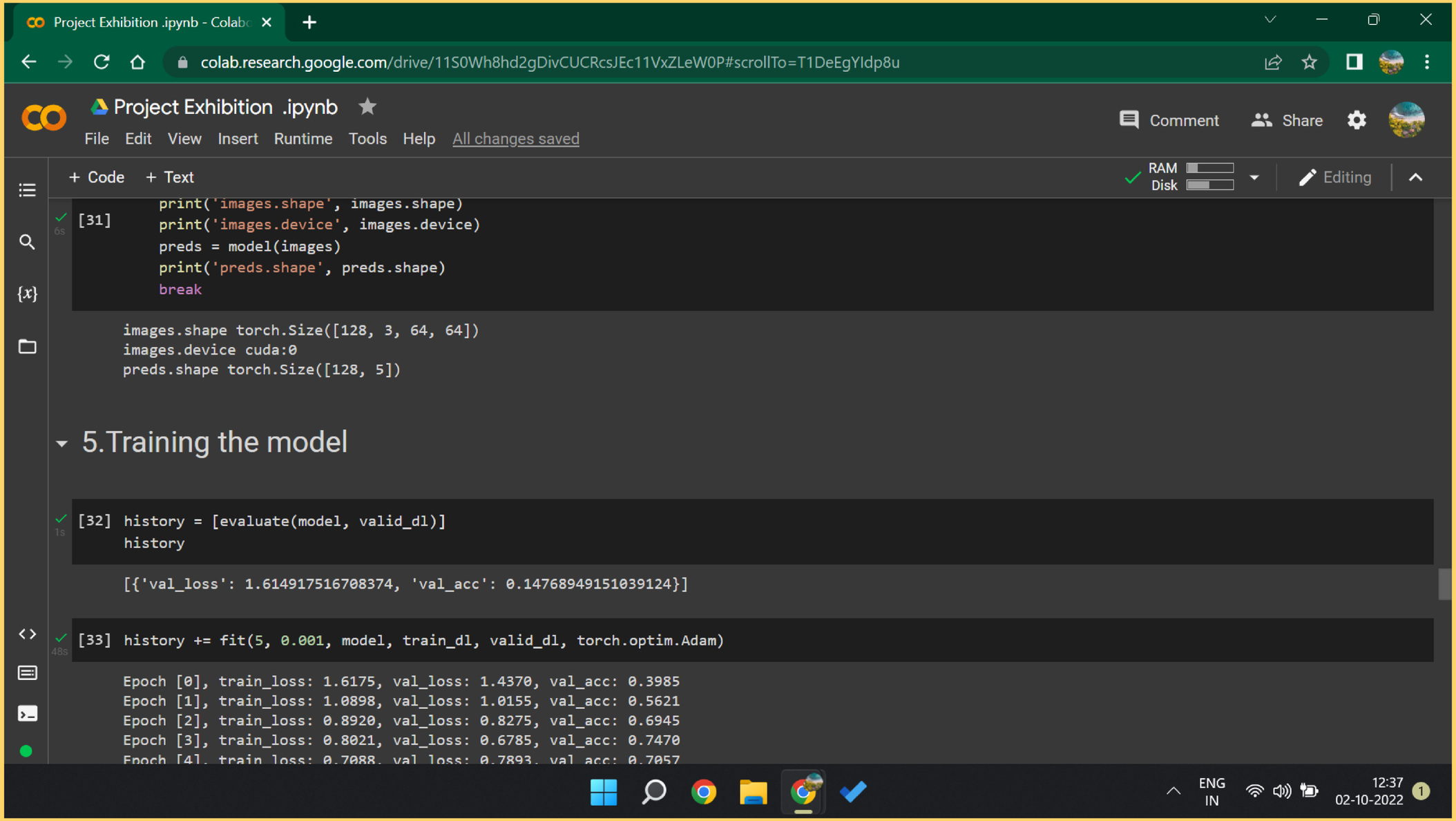
CREATING THE MODEL



CREATING THE MODEL



CREATING THE MODEL

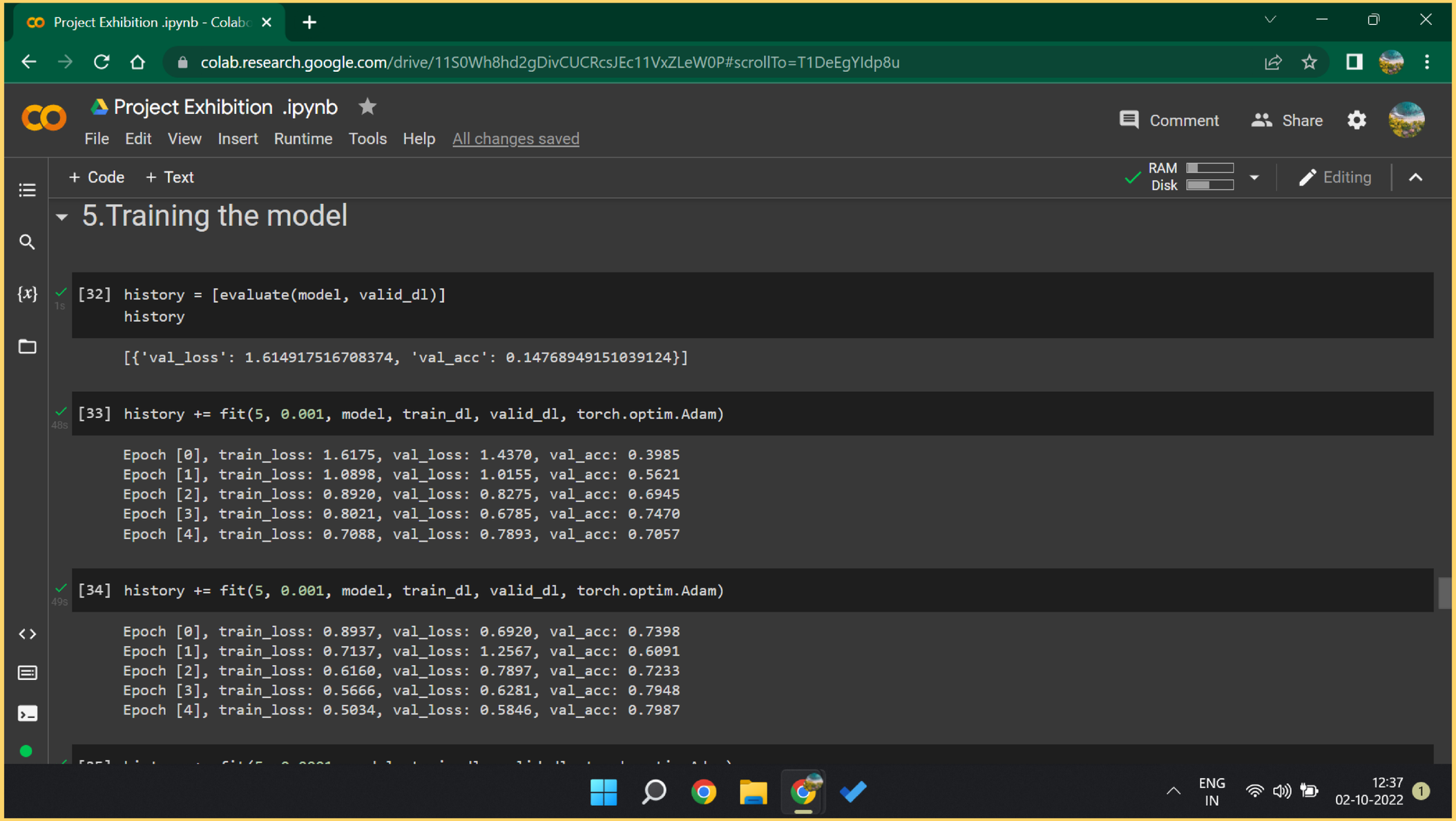


EXPLANATION

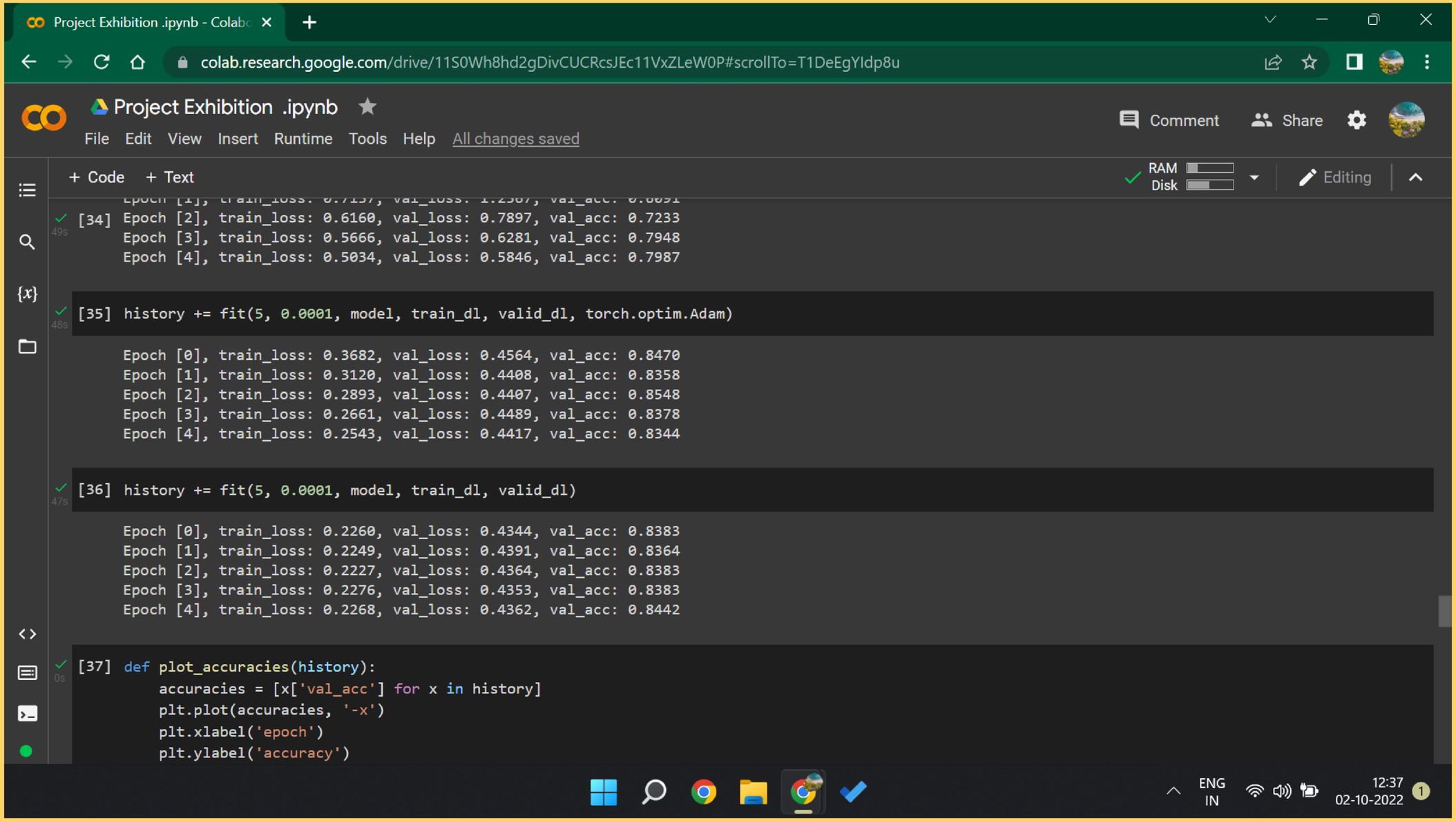
In the model, we define the type of neural network and in our case it is a CNN (Convolutional Neural Network) and after this we configure the number of neurons we want at each layer and the weights and biases between them.We also define the activation function over here.

Changing any attribute/property over here would result in a huge change in the final network.

Training the model



Training the model



Training the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

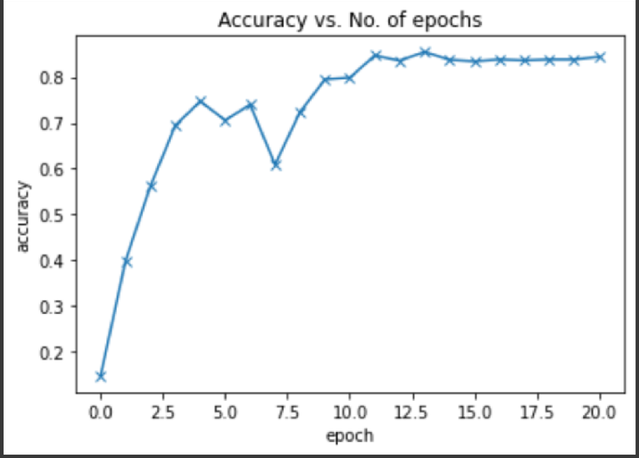
+ Code + Text

RAM Disk

Editing

```
[37] def plot_accuracies(history):
         accuracies = [x['val_acc'] for x in history]
         plt.plot(accuracies, '-x')
         plt.xlabel('epoch')
         plt.ylabel('accuracy')
         plt.title('Accuracy vs. No. of epochs');
```

```
[38] plot_accuracies(history)
```



ENG IN 12:37 02-10-2022

Training the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

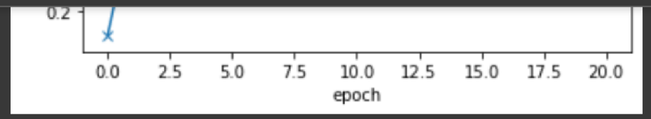
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

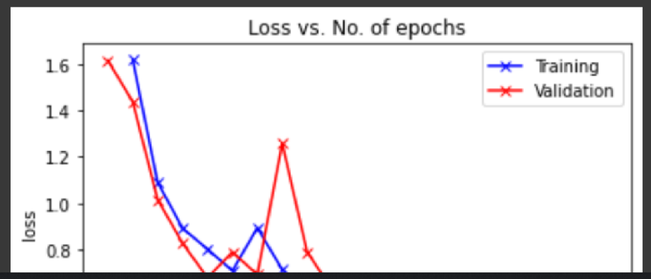
RAM Disk Editing

[38]




```
[39] def plot_losses(history):
         train_losses = [x.get('train_loss') for x in history]
         val_losses = [x['val_loss'] for x in history]
         plt.plot(train_losses, '-bx')
         plt.plot(val_losses, '-rx')
         plt.xlabel('epoch')
         plt.ylabel('loss')
         plt.legend(['Training', 'Validation'])
         plt.title('Loss vs. No. of epochs');
```

```
[40] plot_losses(history)
```



ENG IN 12:37 02-10-2022

Training the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

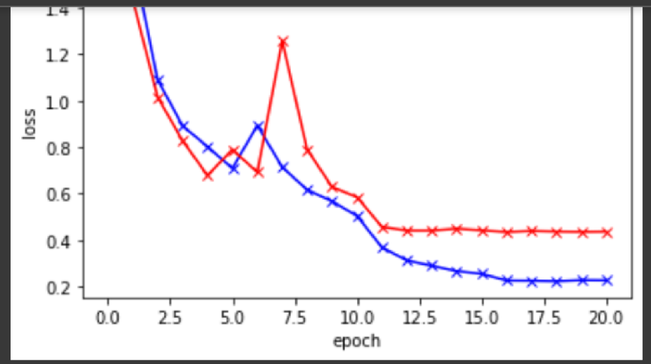
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[40]




```
[41] history[-1]
```

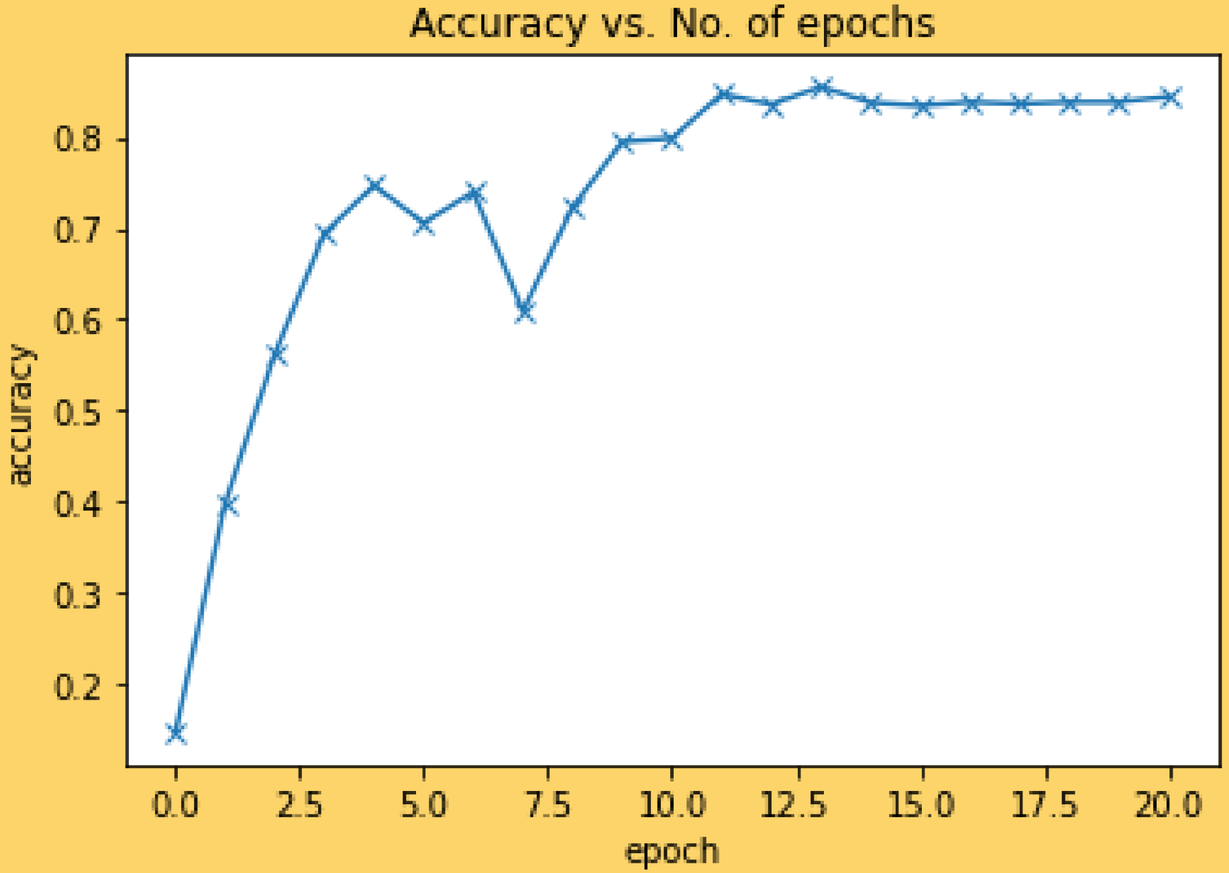
```
{'val_loss': 0.4361819922924042,
 'val_acc': 0.844165563583374,
 'train_loss': 0.22680029273033142}
```

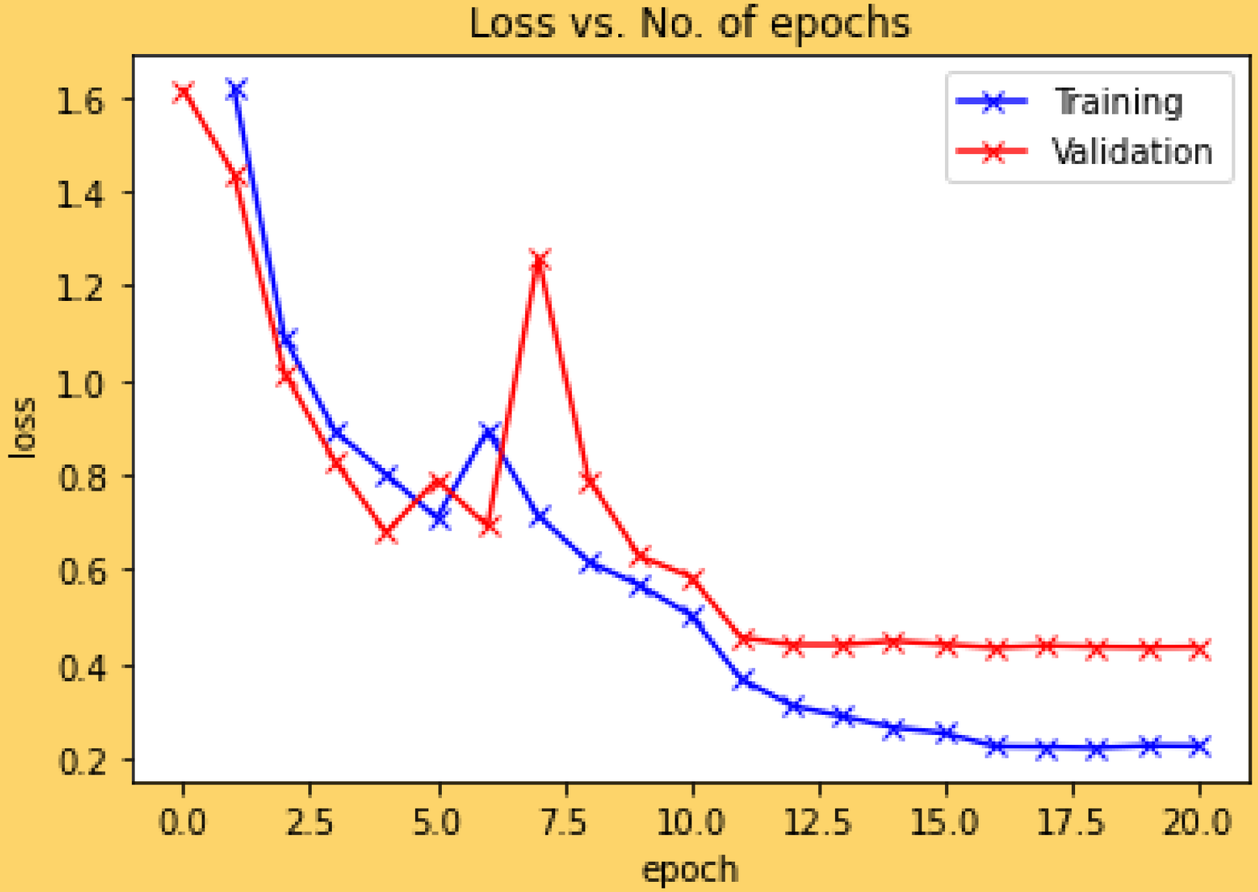
6.Testing the model

```
[42] def predict_image(img, model, classes):
         xb = to_device(img.unsqueeze(0), device)
```

ENG IN 12:37 02-10-2022

Training the model





EXPLANATION

In this part we do a few final modifications like the number of epochs and plotting the accuracy vs the number of epochs, we also plot the history of the loss against the number of epochs.

Testing the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
'val_acc': 0.844165563583374,
'train_loss': 0.22680029273033142}
```

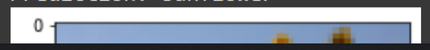
6.Testing the model

```
[42] def predict_image(img, model, classes):
         xb = to_device(img.unsqueeze(0), device)
         yb = model(xb)
         _, preds  = torch.max(yb, dim=1)
         return classes[preds[0].item()]
```

```
[43] def show_image_prediction(img, label):
         plt.imshow(img.permute((1, 2, 0)))
         pred = predict_image(img, model, dataset.classes)
         print('Target:', dataset.classes[label])
         print('Prediction:', pred)
```

```
[44] show_image_prediction(*valid_ds[120])
```

```
Target: sunflower
Prediction: sunflower
```



ENG IN 12:37 02-10-2022

Testing the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

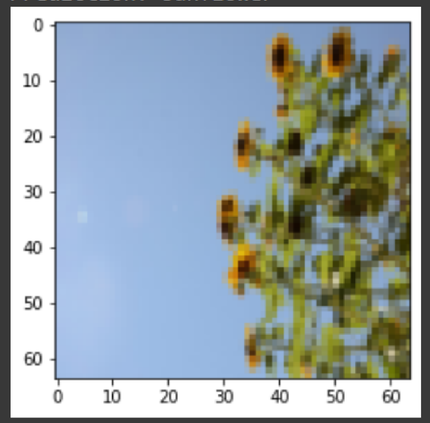
+ Code + Text

RAM Disk

Editing

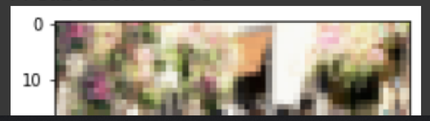
```
[44] show_image_prediction(*valid_ds[120])
```

```
Target: sunflower
Prediction: sunflower
```




```
[45] show_image_prediction(*valid_ds[310])
```

```
Target: rose
Prediction: rose
```



ENG IN 12:37 02-10-2022

Testing the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

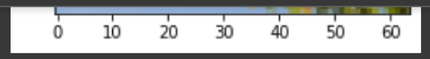
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

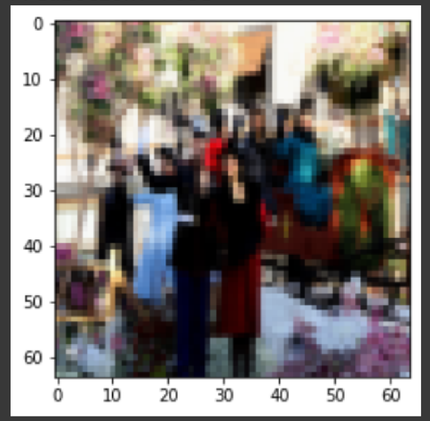
RAM Disk

Editing



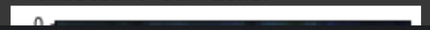

```
[45] show_image_prediction(*valid_ds[310])
```

```
Target: rose
Prediction: rose
```




```
[46] show_image_prediction(*valid_ds[152])
```

```
Target: sunflower
Prediction: sunflower
```



ENG IN 12:37 02-10-2022

Testing the model

Project Exhibition .ipynb - Colab

colab.research.google.com/drive/11S0Wh8hd2gDivCUCRcsJEc11VxZLeW0P#scrollTo=T1DeEgYIdp8u

Project Exhibition .ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

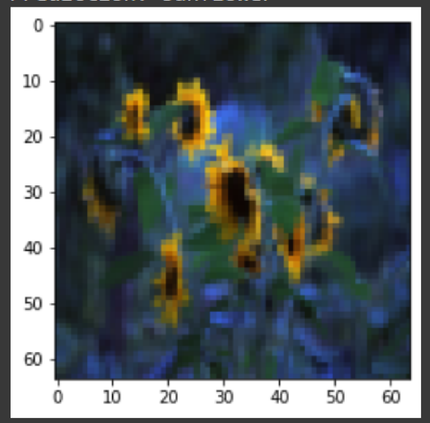
+ Code + Text

RAM Disk

Editing

```
[46] show_image_prediction(*valid_ds[152])
```

```
Target: sunflower
Prediction: sunflower
```




```
[47] torch.save(model.state_dict(), 'flowers-resnet9.pth')
```

ENG IN

12:37 02-10-2022

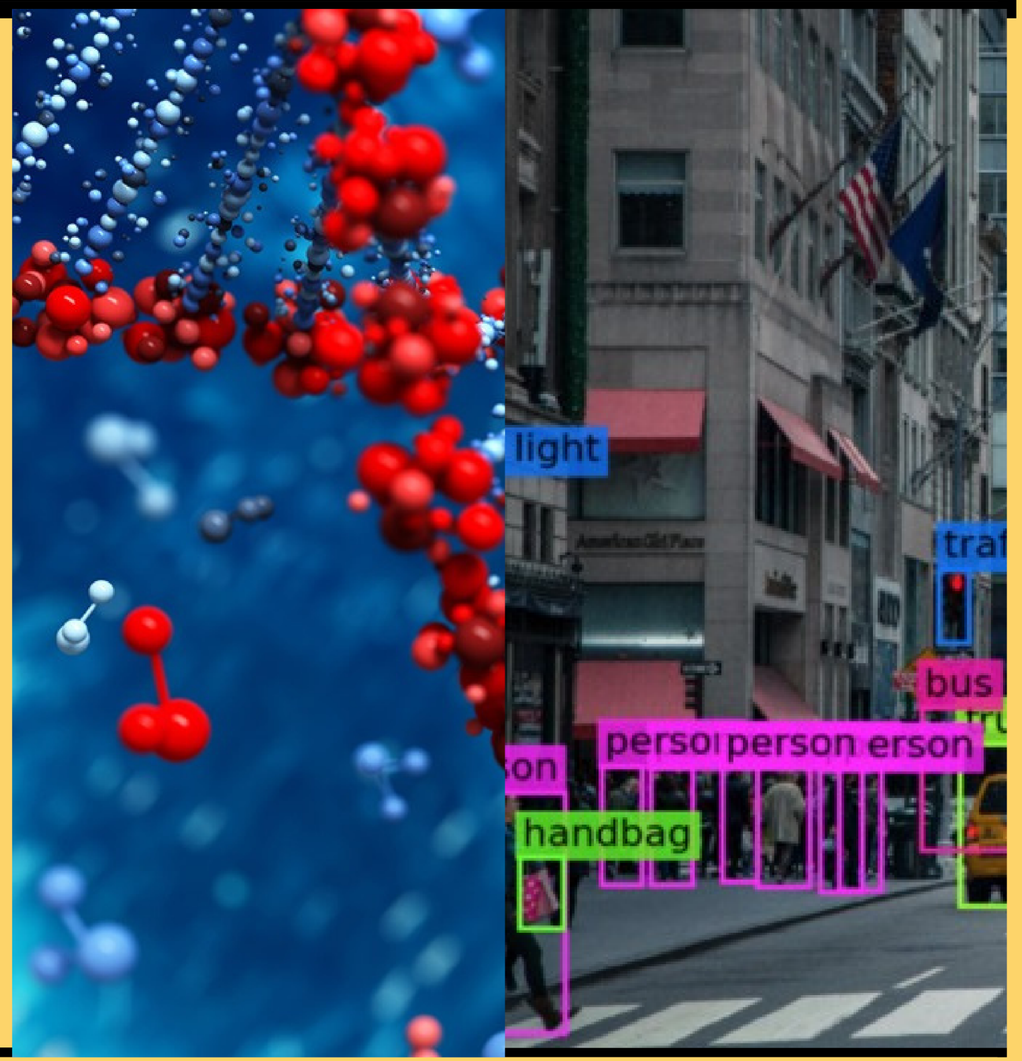
EXPLANATION

In the final part of the code, we are testing our model to see if it has successfully learned to identify different types of flowers and for this we are testing it with random images.

With the completion of the testing phase we have finally completed our whole code.

APPLICATION

Classification of flowers is a relatively simple type of neural network since it uses a pretty typical Convolutional Neural Network (CNN).This model can be further modified to solve different types of problems like classifying different kinds of biomolecules or using computer vision to recognize and identity objects in real life.



CONTRIBUTION

All 5 members of this group contributed towards the code in every step of the process and has a deep and clear understanding of the working principle and the syntax.

THANK YOU!