Recognisign-fastai

File Edit View Insert Runtime Tools Help

Comment Share

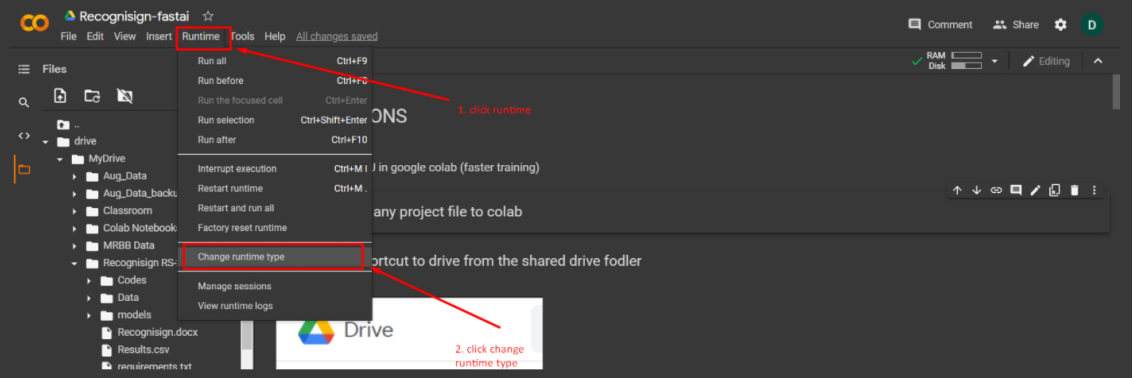
Table of contents

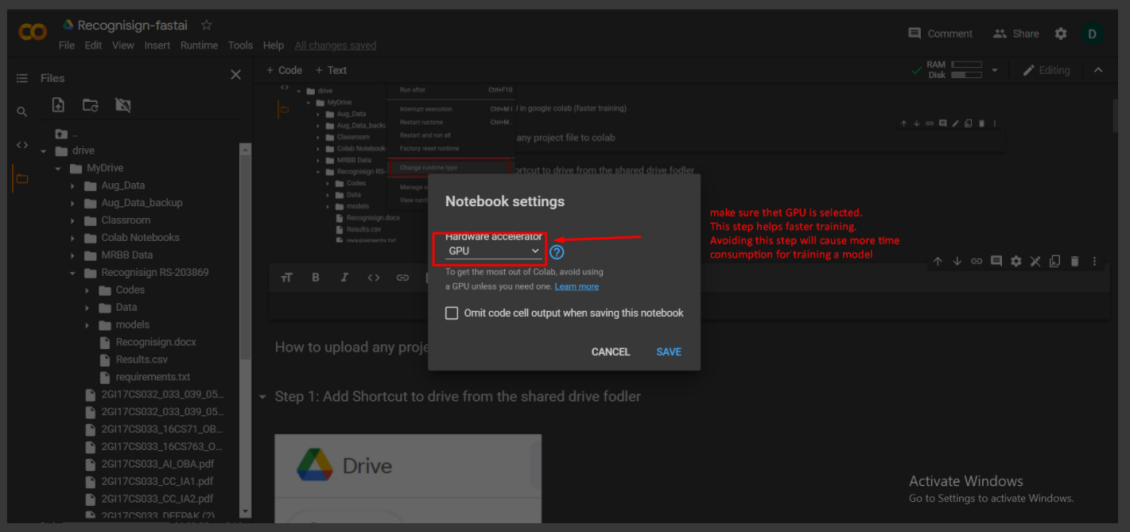
+ Code + Text

Connect Editing

INSTRUCTIONS

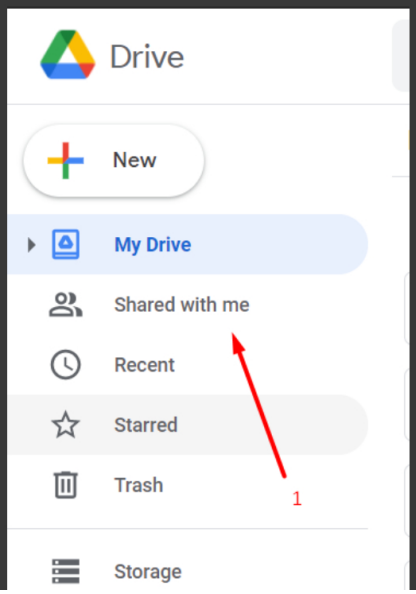
How to enable GPU in google colab (faster training)

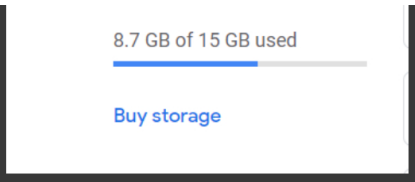


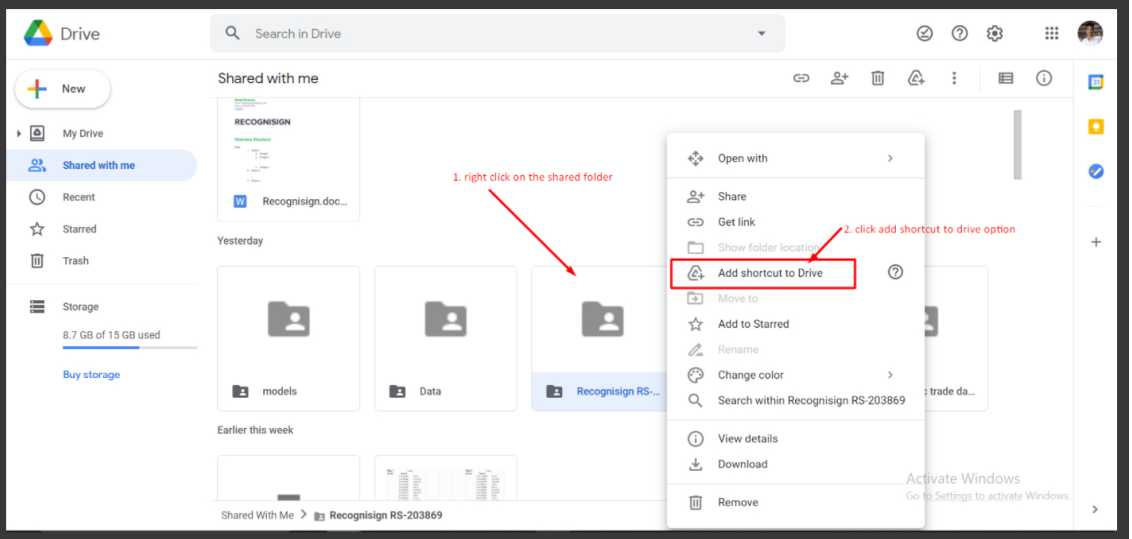


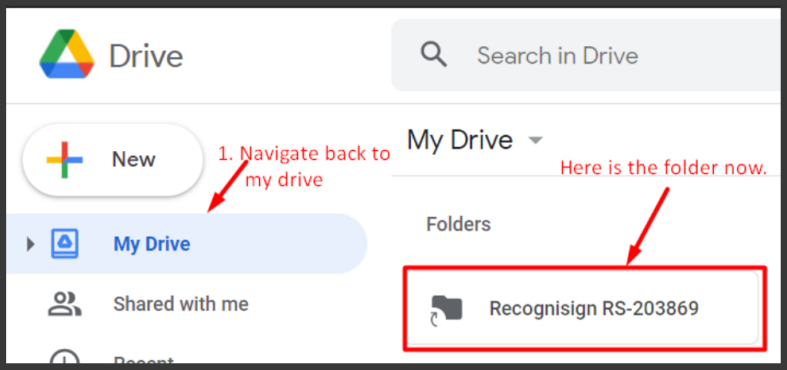
How to upload any project file to colab

Step 1: Add Shortcut to drive from the shared drive fodler









Step 2: Mount your drive on collab notebook

```
[ ]  1 # run this code and click on the link.
     2 from google.colab import drive
     3 drive.mount('/content/drive')
```

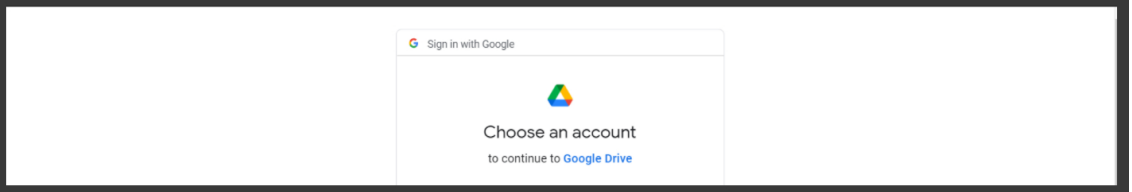
```
Mounted at /content/drive
```

Run the above code. Once you run this you will get a link to authenticate from google (refer below images).

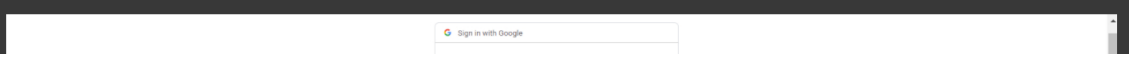
```
from google.colab import drive
drive.mount('/content/drive')
```

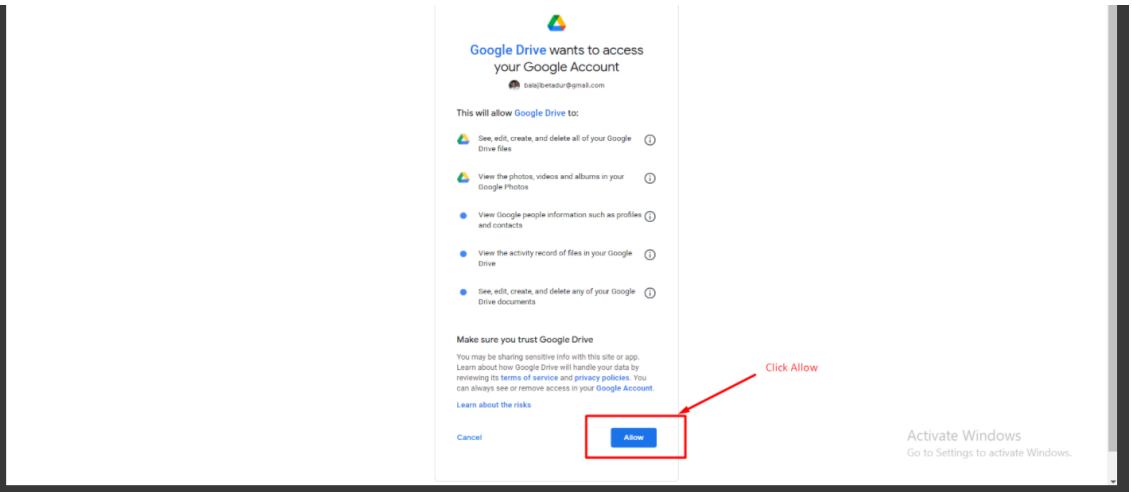
```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercor

Enter your authorization code:
```

Once you click on the link you will be redirected to the google accounts page. Selct the account wher the project folder is present.

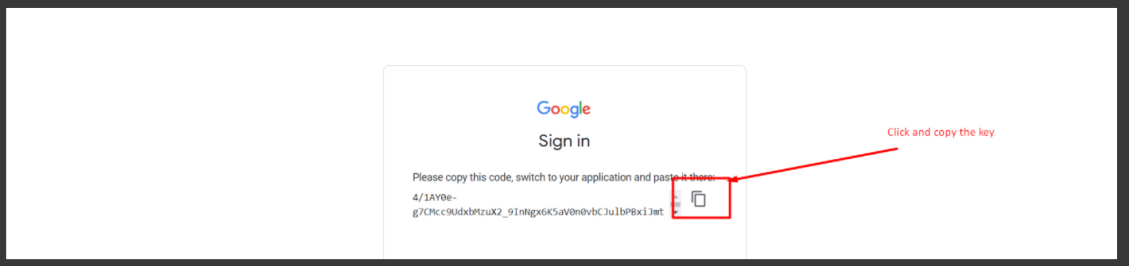


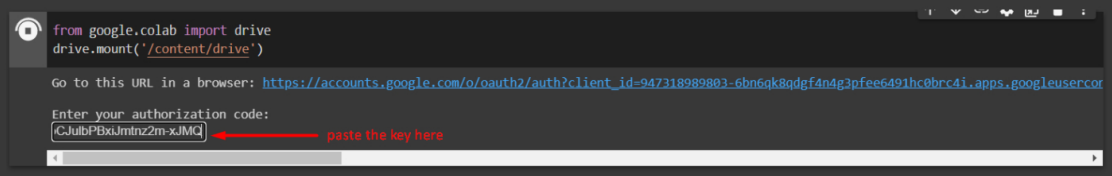
Then click allow in the next redirected page





Then you will get a key for authentication, copy the key.



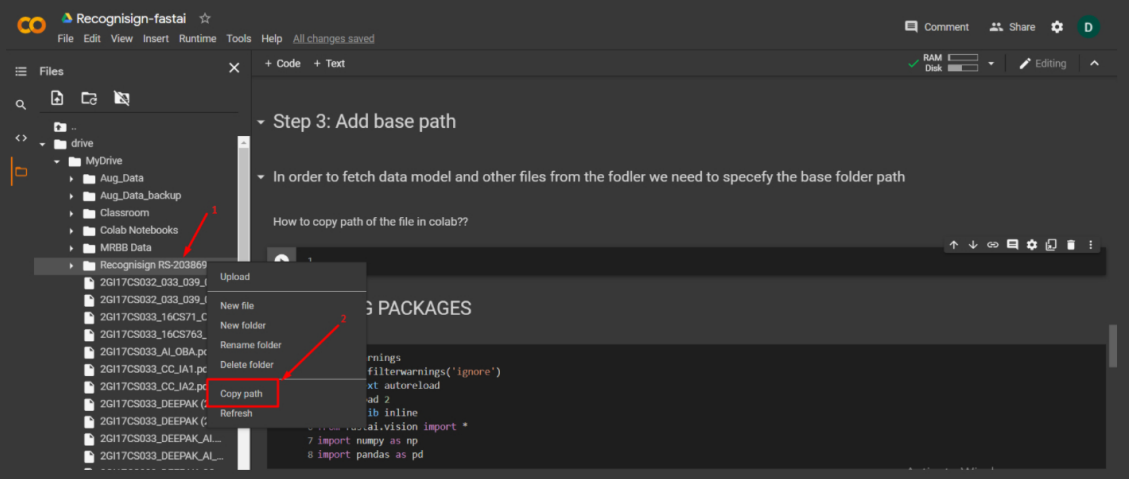


Hurray!!! Your drive is mounted now

Step 3: Add base path

In order to fetch data model and other files from the fodler we need to specefy the base folder path

How to copy path of the file in colab??




```
[ ]  1 # Run the below code and paste the copied path in the input.(refer the below image)
     2 base = input()
```

```
/content/drive/MyDrive/Recognisign RS-203869
```

```
# Run the below code and paste the copied path in the input.
base = input()
```

paste the path here and press enter

IMPORTING PACKAGES

```
import warnings
warnings.filterwarnings('ignore')
%reload_ext autoreload
%autoreload 2
%matplotlib inline
from fastai.vision import *
import numpy as np
import pandas as pd
import shutil
```

DATA SOURCE PATH

```
# data folder path
path = base + '/Data'
ipa=base
```

READING DATA FROM THE FOLDER

```
# reading data
np.random.seed(42)
data = ImageDataBunch.from_folder(path, train= '.', valid_pct=0.1,
                                  ds_tfms=get_transforms(), size=224, num_workers=4).normalize(imagenet_stats)
```

GETTING LEARNER

```
learn = cnn_learner(data, models.resnet50, metrics=accuracy)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357
100% 97.8M/97.8M [00:00<00:00, 192MB/s]
```

Expected output of the above code

```
learn = cnn_learner(data, models.resnet50, metrics=accuracy)

```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
100% 97.8M/97.8M [25:53<00:00, 66.0kB/s]
```

TRAINING MODEL

```
learn.fit_one_cycle(5)
```

Expected output of above model

```
learn.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.342943	0.117913	0.964930	30:59
1	0.096593	0.030123	0.992986	03:01
2	0.052170	0.017865	0.994990	03:03
3	0.025742	0.006927	0.998497	03:02
4	0.015272	0.005542	0.998497	03:02

SAVING MODEL

```
[ ]  1 learn.save(base + '/stage1')
     2 learn.export()
```

```
[ ]  1 shutil.move(base + '/Data/export.pkl', base + '/export.pkl')
```

PREDECTING NEW SAMPLES

```
[ ]  1 #loading model "IMPORTANT"
     2
     3 learn = load_learner(base + '/models')
     4
```

```
[ ]  1 # for single file testing
     2
     3 # BEFORE RUNNING THE PREDICTIONS CELLS MAKE SURE TO LEAD THE MODEL
     4
     5 from google.colab import files
     6 uploaded = files.upload()
     7
     8 for fn in uploaded.keys():
     9   cat, tensor, probs = learn.predict(open_image(fn))
    10   print(f'Predicted class : {cat}')
```

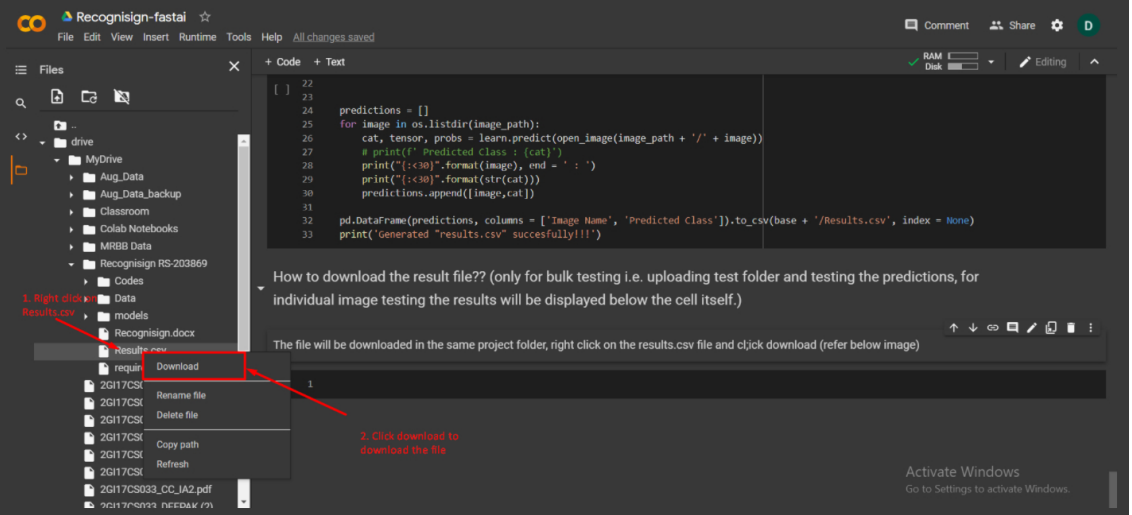
```
[ ]  1 # for testing multiple images please upload the folder to the colab notebooks and then
     2 # enter the path below
     3
     4 # Please follow the above steps to upload the test images folder to colab
     5 # Copy the path of the test images fodler and paste in the below code
     6 image_path="/content/drive/MyDrive/Techfest Test Dataset (1)/"
     7 # for root, dirs, files in os.walk(image_path):
     8 #     print(dirs)
     9 subdirs = [x[0] for x in os.walk(image_path)]
    10 print(subdirs)
    11 fhand = open("2.csv",'a')
    12 with open('2.csv', 'a', newline='') as csvfile:
    13     spamwriter = csv.writer(csvfile, delimiter=' ',
    14                             quotechar=',', quoting=csv.QUOTE_MINIMAL)
    15     spamwriter.writerow([ 'label' , 'Probabiltiy' , 'Time'])
    16     for i in subdirs[1:]:
    17         print(i)
    18         image_path=i
    19
    20     # if the path is the single image then predictions will be printed
    21         if not os.path.isdir(image_path):
    22
    23             cat, tensor, probs = learn.predict(open_image(image_path))
    24             print(f' Predicted Class : {cat}')
    25
    26         # if the image path is the folder the all the images will be predicted and the csv file will be generated
    27         # you can download the csv result file from the colab file panel in the left side
    28
    29         elif os.path.isdir(image_path):
    30
    31             print("{:<30}".format('Image Name'), end = ' : ')
    32             print("{:<30}".format('Predicted Class'))
    33
    34
    35             predictions = []
    36             import time
    37             tik=0
    38             start=time.time()
    39             for image in os.listdir(image_path):
    40                 # print(image)
    41                 tik+=1
    42                 try:
    43                     cat, tensor, probs = learn.predict(open_image(image_path + '/' + image))
    44                 # print(f' Predicted Class : {cat}')
    45                     print("{:<30}".format(image), end = ' : ')
    46                     print("{:<30}".format(str(cat)))
    47                     predictions.append([image,cat])
    48                 except:
    49                     print(image,"nt found")
    50             end=time.time()
    51             print(end-start,tik)
    52             spamwriter.writerow([str(i)  , (tik*1.0/(end - start))])
    53             # pd.DataFrame(predictions, columns = ['Image Name', 'Predicted Class']).to_csv('/content' + '/Results.csv', ind
    54             print('Generated "results.csv" succesfully!!!')
```

```
['/content/drive/MyDrive/Techfest Test Dataset (1)/', '/content/drive/MyDrive/Techfest Test Dataset (1)/40', '/content/drive/M
/content/drive/MyDrive/Techfest Test Dataset (1)/40
Image Name                     : Predicted Class
0.0009548664093017578 0
Generated "results.csv" succesfully!!!
/content/drive/MyDrive/Techfest Test Dataset (1)/28 - compulsory keep left
Image Name                     : Predicted Class
025_0001.png                   : Compulsory keep left
025_1_0001.png                 : Compulsory keep left
index.jpeg                     : Compulsory keep left
index.png                      : Compulsory keep left
unnamed.png                    : Compulsory keep left
229-2298594_traffic-sign-keep-left-sign-regulatory-sign-keep.png : Compulsory keep left
keep-left-traffic-sign-D0BD7D.jpg : Compulsory keep left
```

```
keep-left-traffic-sign-DUMP77.jpg : Compulsory keep left
compulsory-keep-left-sign-board-250x250.jpg : bullock carts prohibhited
images.jpeg                    : Compulsory keep left
jhpolice_sl_mrs34.jpg          : Compulsory keep left
compalsary-left2.jpg           : Compulsory keep left
41AdRzkoMeL._SX342_.jpg        : Compulsory keep left
0.274583101272583 12
Generated "results.csv" succesfully!!!
/content/drive/MyDrive/Techfest Test Dataset (1)/29
Image Name                     : Predicted Class
index.png                      : Compulsory turn left
index.jpg                      : Compulsory turn left
index1.png                     : Compulsory turn left
2.jpeg                         : Compulsory turn left
61.png                         : Compulsory turn left
traffic-sign-compulsory-diection-left-vector-icon-circle-blue-button-white-arrow-179700728.jpg : Compulsory turn right
images.png                     : Compulsory turn right
imag3es.png                    : Compulsory turn right
4.jpeg                         : Compulsory turn left
5.png                          : Compulsory turn left
f.jpg                          : Compulsory turn right
Argentina_road_sign_R21_(B).svg.png : Compulsory turn left
r.jpeg                         : Compulsory turn left
gt.jpg                         : Compulsory turn left
Chile_road_sign_RO-3L.svg.png  : Compulsory turn left
52.jpeg                        : Truck Prohibhited
imeages.png                    : Compulsory turn right
wq.png                         : Compulsory turn left
deaw.jpeg                      : Compulsory turn left
fvdf.png                       : Compulsory turn right
fdasvv.png                     : Compulsory turn right
turn-left-road-traffic-sign-against-blue-sky-BP5PBE.jpg : Compulsory turn right
blue-road-sign-with-white-arrow-to-the-left-E72607.jpg : Compulsory turn right
images.jpeg                    : Compulsory turn right
imaeges.jpeg                   : Compulsory turn right
e.jpeg                         : Compulsory turn right
road-sign-one-way-traffic-against-blue-sky-roads-sign-uk-CXX2B0.jpg : Left Turn Prohibhited
turn-left-traffic-road-sign-symbol-white-arrow-pointing-left-turn-left-traffic-road-sign-symbol-white-arrow-pointing-142051925
0.7033936977386475 28
Generated "results.csv" succesfully!!!
/content/drive/MyDrive/Techfest Test Dataset (1)/1
Image Name                     : Predicted Class
9.png                          : Straight Prohibited or No Entry
12.png                         : Straight Prohibited or No Entry
14.png                         : Straight Prohibited or No Entry
11.png                         : Straight Prohibited or No Entry
```

How to download the result file?? (only for bulk testing i.e. uploading test folder and testing the predictions, for individual image testing the results will be displayed below the cell itself.)

The file will be downloaded in the same project folder, right click on the results.csv file and cl;ick download (refer below image)



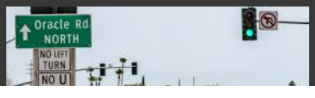

```
[ ]  1
```

```
[ ]  1
```

```
[ ]  1
```

METHOD 2 (EXTRA OPTIONAL)

As we did not find the real tiome images of most of the classes in good number, I decided to use sliding window technique to predict the class of the image even if the sign in the image is small. Refer below image for more clarification



The U turn symbol in this image is very small, so with the sliding window technique we can consider a small cropped section of the image and classify it and then we slide the cropped section towards right and then classify the image again. Repeating this tep for the entire image will give us the final output. (This emthod is not preffered because it is time consuming as we have to iterate through the single image)

Here is my linkedin article on this technique: https://lnkd.in/gzjt4fQ

CODE FOR SLIDING WINDOW TECHNIQUE

This method require more time for classifying images.

```
[ ]
# import the necessary packages
import imutils

def sliding_window(image, step, ws):
    # slide a window across the image
    for y in range(0, image.shape[0] - ws[1], step):
        for x in range(0, image.shape[1] - ws[0], step):
            # yield the current window
            yield (x, y, image[y:y + ws[1], x:x + ws[0]])

def image_pyramid(image, scale=1.5, minSize=(224, 224)):
    # yield the original image
    yield image

    # keep looping over the image pyramid
    while True:
        # compute the dimensions of the next image in the pyramid
        w = int(image.shape[1] / scale)
        image = imutils.resize(image, width=w)

        # if the resized image does not meet the supplied minimum
        # size, then stop constructing the pyramid
        if image.shape[0] < minSize[1] or image.shape[1] < minSize[0]:
            break

        # yield the next image in the pyramid
        yield image
```

```
[ ]
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications import imagenet_utils
from imutils.object_detection import non_max_suppression
# from pyimagesearch.detection_helpers import sliding_window
# from pyimagesearch.detection_helpers import image_pyramid
import numpy as np
import argparse
import imutils
import time
import cv2
```

```
[ ]
WIDTH = 600
PYR_SCALE = 1.5
WIN_STEP = 16
ROI_SIZE = (200, 150) #eval(args["size"])
INPUT_SIZE = (224, 224)
```

```
[ ]
image_path="path-to-the-test-images-folder"
```

```
[ ]
predictions2 = []
for image in os.listdir(image_path):


    orig = cv2.imread(image_path + '/' + image)
    orig = imutils.resize(orig, width=WIDTH)
    (H, W) = orig.shape[:2]

    pyramid = image_pyramid(orig, scale=PYR_SCALE, minSize=ROI_SIZE)

    rois = []
    locs = []


    # loop over the image pyramid
    for image in pyramid:
        # determine the scale factor between the *original* image
        # dimensions and the *current* layer of the pyramid
        scale = W / float(image.shape[1])

        # for each layer of the image pyramid, loop over the sliding
```

```
        # window locations
        for (x, y, roiOrig) in sliding_window(image, WIN_STEP, ROI_SIZE):
            # scale the (x, y)-coordinates of the ROI with respect to the
            # *original* image dimensions
            x = int(x * scale)
            y = int(y * scale)
            w = int(ROI_SIZE[0] * scale)
            h = int(ROI_SIZE[1] * scale)

            # take the ROI and pre-process it so we can later classify
            # the region using Keras/TensorFlow
            roi = cv2.resize(roiOrig, INPUT_SIZE)
            roi = img_to_array(roi)
            roi = preprocess_input(roi)

            # update our list of ROIs and associated coordinates
            rois.append(roi)
            locs.append((x, y, x + w, y + h))

            # check to see if we are visualizing each of the sliding
            # windows in the image pyramid
            if -1 > 0:
                # clone the original image and then draw a bounding box
                # surrounding the current region
                clone = orig.copy()
                cv2.rectangle(clone, (x, y), (x + w, y + h),
                    (0, 255, 0), 2)

                # show the visualization and current ROI
                cv2.imshow("Visualization", clone)
                cv2.imshow("ROI", roiOrig)
                cv2.waitKey(0)

    image_path="test.jpg"
    sets =[]
    l =[]
    for i in rois:
        cv2.imwrite('test.jpg',i)
        cat, tensor, probs = learn.predict(open_image(image_path))
        sets.append([str(cat), float(probs[np.argmax(probs)])])
    test = list(sets)
    for i in test:
        if i[1] < 0.9:
            sets.remove(i)
    proba = {}
    for i in sets:
        if i[0] in proba:
            proba[i[0]] += 1
        elif i[0] not in proba:
            proba[i[0]] = 1

    a = list(proba.items())
    a.sort(key =lambda x:x[1])
    cls = a[-1][0]
    predictions2.append([image, cls])
    # print(f'Predictions : {a[-1][0]}')

pd.DataFrame(predictions, columns = ['Image Name', 'Predicted Class']).to_csv(base + '/Results.csv', index = None)
print('Generated "results.csv" succesfully!!!')
```

```
[ ]  1
```

```
[ ]  1
```