National College of Ireland

Traffic Sign Detection and Recognition for Autonomous Vehicles Using Transfer Learning

MSc Research Project
MSc in Data Analytics

Naresh Potla
Student ID: x21126241

School of Computing
National College of Ireland

Supervisor: Aaloka Anant

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Naresh Potla
Student ID:	X21126241
Programme:	MSc in Data Analytics **Year:** 2022 - 2023
Module:	Research Project
Supervisor:	Aaloka Anant
Submission Due Date:	February 1st , 2023
Project Title:	Traffic Sign Detection and Recognition for Autonomous Vehicles Using Transfer Learning
Word Count:	279 **Page Count** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Naresh Potla

Date: February 1st , 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

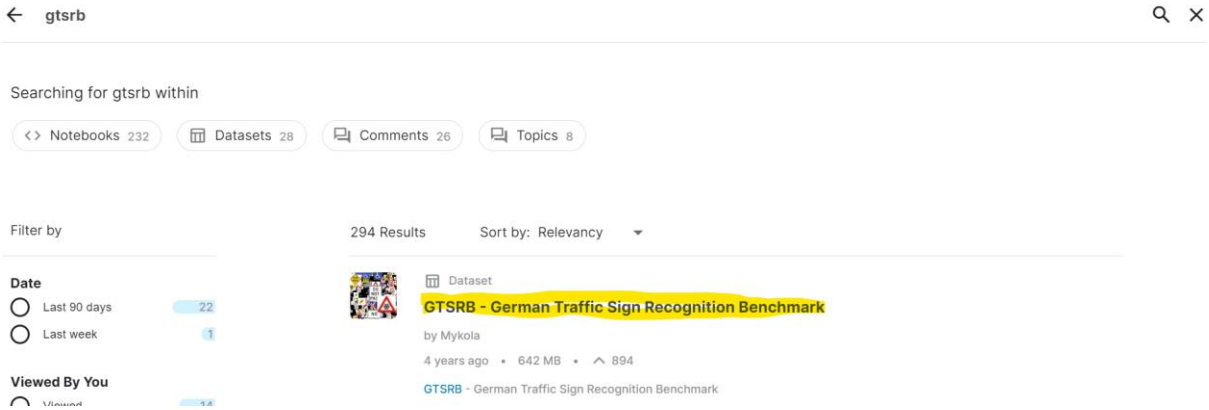
Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

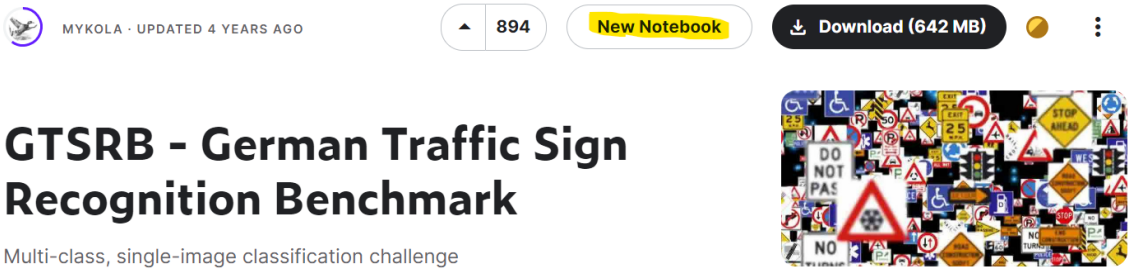
Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

1 Environment setup in Kaggle

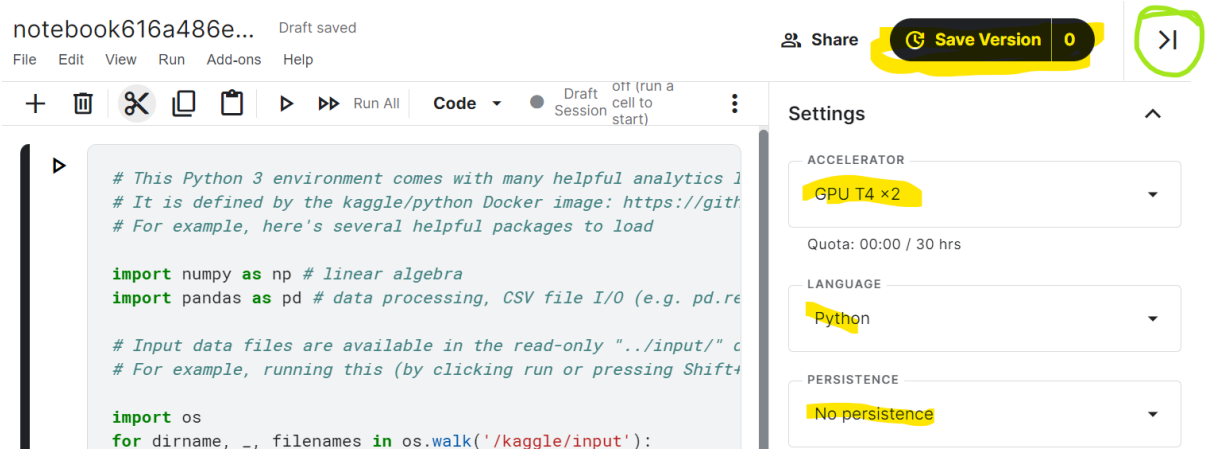
Due to lack of computing resources, I implemented and executed all my changes in the Kaggle notebook itself. First, login to Kaggle website with a Google account. Type "**GTSRB**" in Kaggle's search bar and choose the **"GTSRB - German Traffic Sign Recognition Benchmark"** dataset.



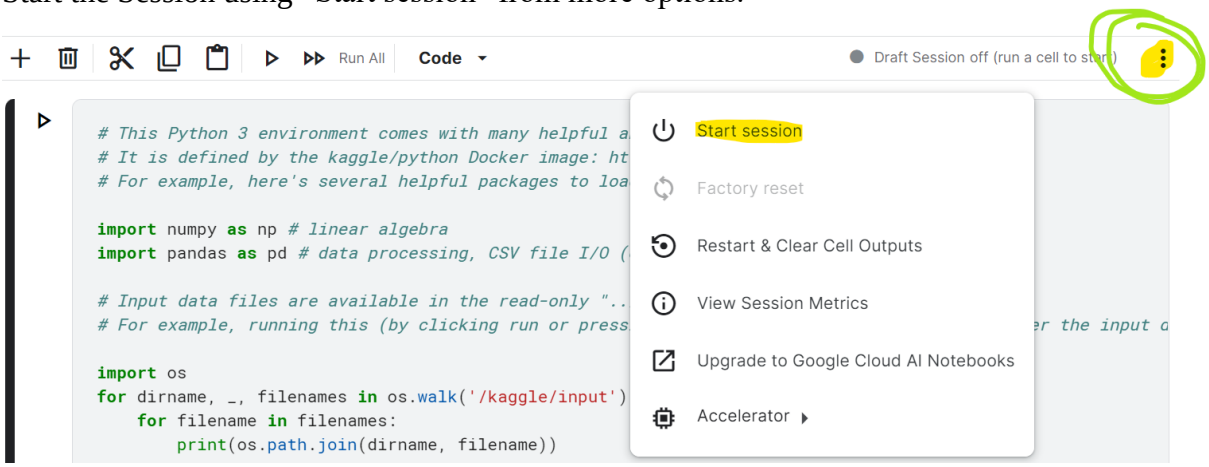
Now open the dataset and create a new notebook by clicking the "New Notebook"



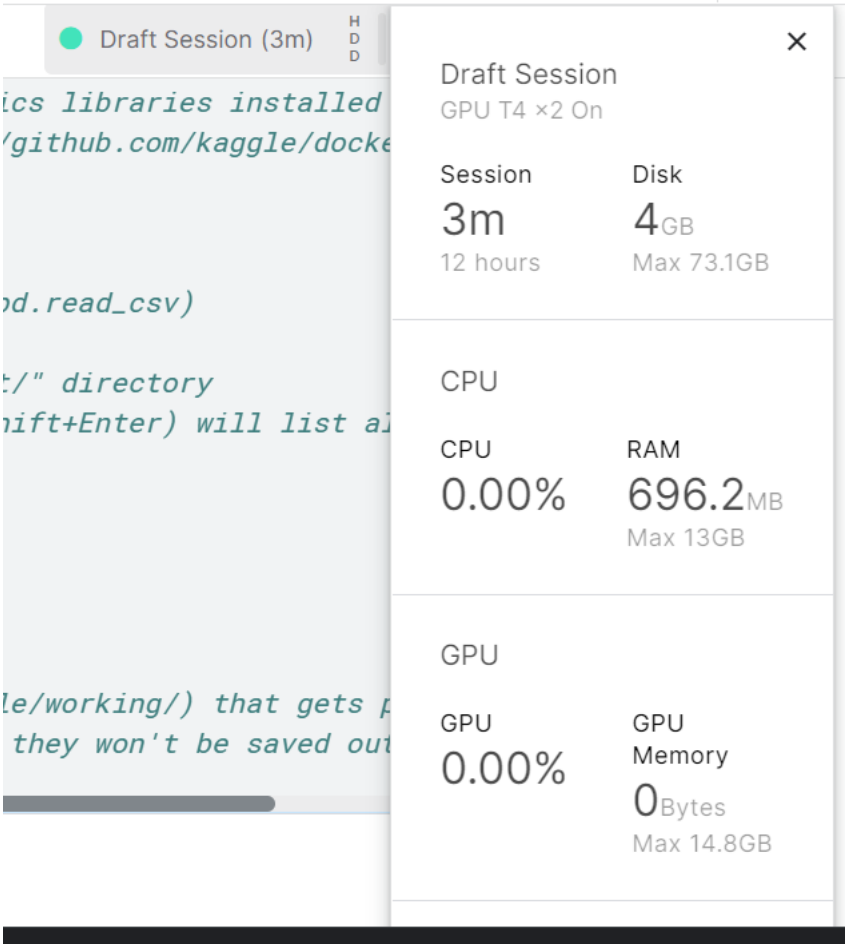
To setting up the configuration required to run the code go to "Settings" Section, which is in "Toggle" (click |< next to Save Version). In settings section choose **ACCELERATOR** as GPU T4 *2, **LANGUAGE** as Python, **PERSISTENCE** as "No persistence" and **ENVIRONMENT** as "Pin to original environment".



Start the Session using “Start session” from more options.



Once we start the session, we will get a Disk, RAM, and GPU (Click on Draft Session).



2 Code Implementation

Install the tensorflow version 2.9.2 using pip install command.

```
!pip install tensorflow==2.9.2
!apt install -y --allow-change-held-packages libcudnn8=8.1.0.77-1+cuda11.2
#import tensorflow as tf
```

Importing the required libraries

```
import numpy as np
import pandas as pd
import os
import pathlib
import cv2
import matplotlib.pyplot as plt
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import scikitplot as skplt
import random as rn
import tensorflow as tf
from tensorflow import keras
from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
import keras.applications.efficientnet_v2
from keras.models import load_model
import tensorflow_hub as hub
from PIL import Image, ImageEnhance
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, array_to_img, load_img
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
#print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
import keras.optimizers.optimizer_v2.adam as ad
```

Now I am reading data directories and initializing the pixel sizes, batch sizes and creating a dictionary for all the classes.

```
data_dir = '/kaggle/input/gtsrb-german-traffic-sign/'
train_path = '/kaggle/input/gtsrb-german-traffic-sign/Train/'
test_path = '/kaggle/input/gtsrb-german-traffic-sign/Test/'
height = 64
width = 64
batch_size = 64
seed = 54
```

```
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
```

Below code for increasing the brightness of the input images with ranges [0.0,1.0] and [1.0,2.0] and splitting the input data into train and test in the ratio of 80 and 20.

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   validation_split=0.2,
                                   featurewise_center=True,
                                   featurewise_std_normalization=True,
                                   brightness_range=[1.0,2.0])
train_dataset = train_datagen.flow_from_directory(train_path,
                                                  target_size=(height, width),
                                                  batch_size=batch_size,
                                                  class_mode='categorical',
                                                  shuffle=True,
                                                  seed=seed,
                                                  interpolation='hamming',
                                                  subset='training')
val_dataset = train_datagen.flow_from_directory(train_path,
                                                target_size=(height, width),
                                                batch_size=batch_size,
                                                class_mode='categorical',
                                                shuffle=True,
                                                seed=seed,
                                                interpolation='hamming',
                                                subset='validation')
```

Below code is to display the augmented input images with adjusted brightness.

```
fig,ax=plt.subplots(3,4)
fig.set_size_inches(16,12)
img,y = train_dataset.next()
for i in range(3):
    for j in range (4):
        l=rn.randint(0,batch_size-1)
        label = classes[int(list(train_dataset.class_indices.keys())[np.argmax(y[l])])]
        ax[i,j].imshow(img[l])
        ax[i,j].set_title(label)

plt.tight_layout()
```

3 Pre-Trained Models implementation

EfficientNetV2L

```
effnv2l = keras.applications.efficientnet_v2.EfficientNetV2L(weights='imagenet', include_top=False, input_shape=(height,width,3))
# effnv2l.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/efficientnet_v2/efficientnetv2-l_notop.h5
473176280/473176280 [==============================] - 3s 0us/step
```

```
model = Sequential()
model.add(effnv2l)
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(43, activation='softmax'))

model.summary()
```

VGG19

```
vgg_model = tf.keras.Sequential([VGG19(weights='imagenet', include_top=False, input_shape=(height,width,3)),
                                 keras.layers.BatchNormalization(),
                                 keras.layers.Flatten(),
                                 keras.layers.Dense(512, activation='relu'),
                                 keras.layers.Dense(43, activation='softmax')
                                ])
vgg_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 1, 1, 512)         20024384
```

Here, in both the models, imagenet database weights are taking and trainable status to False. Also, I used relu activation function in the middle layers and softmax in the outer layer.

Model Fitting

Running both the models over 20 epochs (due to resource constraint)

```
model_history = model.fit(train_dataset,
                      steps_per_epoch=len(train_dataset),
                      validation_data=val_dataset,
                      validation_steps=len(val_dataset),
                      verbose = 1,
                      epochs=epochs)
```

```
Epoch 1/20
2022-11-30 19:03:08.763976: W tensorflow/stream_executor/gpu/asm_compiler.cc:111] *** WARNING *** You are using ptxas 11.0.221, which is older than 11.1. ptxas before 11.1 is known to miscompile XLA code, leading to incorrect results or invalid-address errors.

You may not need to update to CUDA 11.1; cherry-picking the ptxas binary is often sufficient.
491/491 [==============================] - 285s 478ms/step - loss: 0.9982 - accuracy: 0.7087 - val_loss: 0.5027 - val_accuracy: 0.8559
Epoch 2/20
491/491 [==============================] - 212s 432ms/step - loss: 0.0994 - accuracy: 0.9700 - val_loss: 0.3441 - val_accuracy: 0.9066
Epoch 3/20
491/491 [==============================] - 212s 432ms/step - loss: 0.0470 - accuracy: 0.9861 - val_loss: 0.7589 - val_accuracy: 0.8155
Epoch 4/20
491/491 [==============================] - 211s 430ms/step - loss: 0.0368 - accuracy: 0.9895 - val_loss: 0.2351 - val_accuracy: 0.9410
Epoch 5/20
491/491 [==============================] - 210s 428ms/step - loss: 0.0254 - accuracy: 0.9935 - val_loss: 0.2065 - val_accuracy: 0.9536
Epoch 6/20
```

Evaluating Model

```
def map_pred(pred):
    return [int(list(train_dataset.class_indices.keys())[i]) for i in pred]
```

```
test_df = pd.read_csv(data_dir + 'Test.csv')

y_test = test_df["ClassId"].values
test_imgs = test_df["Path"].values

test_data =[]

for img in test_imgs:
    try:
        image = cv2.imread(data_dir +img)
        image_fromarray = Image.fromarray(image)
        resize_image = image_fromarray.resize((height, width))
        test_data.append(np.array(resize_image))
    except:
        print("Error in " + img)
X_test = np.array(test_data)
X_test = X_test/255
```

EfficientNetV2L

```
from sklearn.metrics import accuracy_score
pred = model.predict(X_test)
pred = map_pred(pred.argmax(axis=-1))
print('Test Data accuracy: ',accuracy_score(y_test, pred)*100)
```

```
2022-11-30 20:14:40.346033: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 620789760 exceeds 10% of free s
ystem memory.
2022-11-30 20:14:41.260901: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 620789760 exceeds 10% of free s
ystem memory.
```

```
395/395 [==============================] - 39s 81ms/step
Test Data accuracy:  92.96120348376881
```

VGG19

```
from sklearn.metrics import accuracy_score
pred = vgg_model.predict(X_test)
pred = map_pred(pred.argmax(axis=-1))
print('Test Data accuracy: ',accuracy_score(test_labels, pred)*100)
```

```
2022-11-30 17:28:10.966888: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 378900000 exceeds 10% of free s
ystem memory.
2022-11-30 17:28:11.413739: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 378900000 exceeds 10% of free s
ystem memory.
```

```
395/395 [==============================] - 7s 16ms/step
Test Data accuracy:  78.36104513064133
```

Classification Report

```
In [19]: from sklearn.metrics import classification_report
         print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.96	0.90	0.93	60
1	0.86	1.00	0.92	720
2	0.96	1.00	0.98	750
3	0.91	0.94	0.92	450
4	0.93	0.99	0.96	660
5	0.84	0.99	0.91	630
6	0.97	0.99	0.98	150
7	1.00	0.90	0.95	450
8	0.98	0.98	0.98	450
9	0.95	1.00	0.97	480
10	1.00	0.89	0.94	660
11	0.97	0.93	0.95	420
12	0.92	0.96	0.94	690
13	1.00	0.99	1.00	720
14	0.82	0.87	0.84	270
15	0.99	0.46	0.63	210
16	1.00	1.00	1.00	150
17	0.95	0.95	0.95	360
18	0.97	0.72	0.83	390
19	0.97	0.98	0.98	60
20	0.99	1.00	0.99	90
21	0.98	0.98	0.98	90
22	1.00	0.75	0.86	120
23	0.99	0.90	0.94	150
24	0.87	1.00	0.93	90
25	0.96	0.97	0.96	480
26	0.60	0.92	0.72	180
27	0.81	1.00	0.90	60
28	0.95	0.97	0.96	150
29	0.87	1.00	0.93	90
30	0.79	0.84	0.81	150
31	1.00	1.00	1.00	270
32	0.90	1.00	0.94	60
33	0.99	0.81	0.90	210
34	0.75	0.97	0.85	120
35	0.95	0.95	0.95	390
36	1.00	0.88	0.94	120
37	0.80	0.98	0.88	60
38	0.99	0.80	0.88	690
39	0.82	0.67	0.74	90
40	0.87	0.86	0.86	90
41	1.00	0.75	0.86	60
42	0.86	1.00	0.92	90
accuracy			0.93	12630
macro avg	0.92	0.92	0.91	12630
weighted avg	0.94	0.93	0.93	12630

References

https://keras.io/api/applications/vgg/
https://keras.io/api/applications/efficientnet_v2/
https://keras.io/guides/transfer_learning/
https://towardsdatascience.com/a-practical-guide-to-stacking-using-scikit-learn-91e8d021863d
https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/
https://pillow.readthedocs.io/en/stable/reference/Image.html