Configuration manual

MSc Research Project
MSc in Data analytics

Vibha Vaid
Student ID: x20205635

School of Computing
National College of Ireland

Supervisor: Dr. Giovani Estrada

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vibha Vaid	
Student ID:	x20205635	
Programme:	MSc in data analytics	2021-2022
Module:	MSc Research project	
Supervisor:	Dr. Giovani Estrada	
Submission Due Date:	19/09/22	
Project Title:	A novel CNN architecture for classification of galaxies	
Word Count:	761	**Page Count:** 8

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: ……………………………………………………………………………………………………

Date: ……………………………………………………………………………………………………

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):	

Configuration Mannual

Vibha Vaid
x20205635

19th September 2022

1 Introduction

This document will depict the information about the dataset, the software and hardware specification, tools, libraries, and code that is required to execute the models implemented in the research project " A novel CNN architecture for the classification of galaxies"

2 Hardware configuration

This section showcase the hardware configuration of the system used in this research

Operating System:	Windows 11
Processor:	Intel(R) Core(TM) i3-8145U CPU @ 2.10GHz 2.30 GHz
Installed RAM:	12.0 GB (11.8 GB usable)
System Type:	64-bit operating system, x64-based processor
Hard Disk:	1 TB

3 Environment Setup

Google collab This research aim in developing a novel CNN architecture for the classification of galaxies. A total of 5 models are built and are compared against each other in terms of accuracy, precision, and recall. The models are tested on how accurately can it classify the galaxies. The programming language used in the creation of the model is python. All the codes are executed using google collab integrated development. The code is downloaded in ipynb notebooks. The python notebook can be easily downloaded and uploaded to GitHub. 1.The main advantage of google collab is that it gives access to GPU which makes the compilation faster. Go to runtime -¿ change runtime -¿ select GPU/TPU

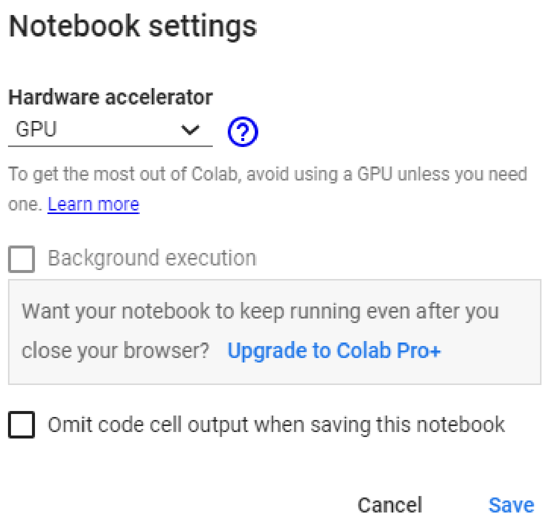

Figure 1: Changing runtime

2. All the cell in the collab notebook can be exected by ctrl+enter or pressing the run key

3.1 Google drive

The dataset used in this research is taken from kaggle so there are two ways of aacessing the dataset[1].
1. Using the kaggle api the data can be accessed from the web

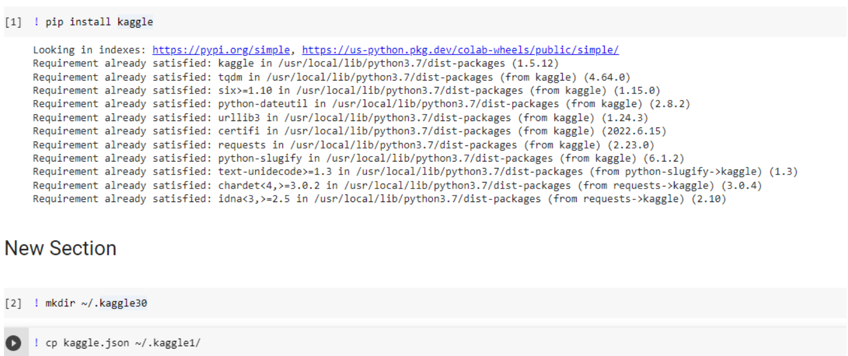

Figure 2: Accessing kaggle API

2. By installing the data in the drive and then mounting the data from the drive

[1] https://drive.google.com/

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

Mounted at /content/drive
```

Figure 3: Mounting the drive

3.2 Data preparation

The data to be used in this research is downloaded from kaggle[2] which is a part of the galaxy zoo project and the dataset is downloaded in a ZIP file. The zip file is divided into five separate folders images_training, solutions_training, images_test, all_ones_benchmark, all_zeros_benchmark, central_pixel_benchmark. Images_training contains 61,578 images in JPG format.

4 Code execution steps

The code file submitted contains the 5 different files which have 5 different neural network and the name of the file is given such that it can be identified easily. All the four files contains libraries required to perform the tasks, Data access to the drive, splitting of data, data augmentation, model import if it is transfer learning and model modification and finally the result generation followed by model evaluation.

4.1 Importing Dependent Libraries

Their are librararies of python which are installed for performing the tasks required. Keras have built in libraries which can be used to perform function on neural network.

[2] https://www.kaggle.com/competitions/galaxy-zoo-the-galaxy-challenge/data

```python
import random
from PIL import Image
from cv2 import imread
import matplotlib.pyplot as plt
import time, os, sys
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import backend as K
from tensorflow.keras import layers, metrics, losses, callbacks, regularizers
from tensorflow.python.client import device_lib

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os, random, shutil
import tensorflow as tf
import seaborn
from tensorflow import keras
from tensorflow.keras import preprocessing, layers
from tensorflow.keras.callbacks import EarlyStopping
from keras_preprocessing import image
from keras.applications.mobilenet import MobileNet, preprocess_input
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Dense,BatchNormalization, Flatten, MaxPool2D, LSTM, Flatten, TimeDistributed
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, Callback
from keras.layers import Conv2D, Reshape
from tensorflow.keras.utils import Sequence
from keras.backend import epsilon
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.layers import GlobalAveragePooling2D
```

Figure 4: Importing libraries

4.2 Data Access from Drive and Data Split

Depending on the descison tree, the galaxies are divided into 3 classes elliptical, spiral and lenticular.

```python
elliptical_galaxy = dataframe[(dataframe['Q1.1']>0.7) & (dataframe['Q7.1']>0.4)]['GalaxyID'].tolist()
lenticular_galaxy = dataframe[(dataframe['Q1.1']>0.7) & (dataframe['Q7.2']>0.4)]['GalaxyID'].tolist()
spirals_galaxy = dataframe[(dataframe['Q1.2']>0.7) & (dataframe['Q2.1']>0.4)]['GalaxyID'].tolist()
print('Sum of total number of elliptical : ',  len(elliptical_galaxy))
print('Sum of total number of lenticular : ',  len(lenticular_galaxy))
print('Sum of total number of spiral: ',  len(spirals_galaxy))
```

```
Sum of total number of elliptical :  7311
Sum of total number of lenticular :  6625
Sum of total number of spiral:  4635
```

Figure 5: Importing libraries

The data in each class is split into 80% train and 20% validation data.

```python
images('/content/images_training_rev1', '/content/galaxy-zoo-clean/data/', 'elliptical', elliptical_galaxy, 0.80)
images('/content/images_training_rev1','/content/galaxy-zoo-clean/data/', 'lenticular', lenticular_galaxy, 0.80)
images('/content/images_training_rev1', '/content/galaxy-zoo-clean/data/', 'spiral', spirals_galaxy, 0.80)


print('Elliptical:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data', 'train', 'elliptical'))))
print('Total train lenticular:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data','train','lenticular'))))
print('Total train spiral:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data','train','spiral'))))


print('Total validation elliptical:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data', 'validation', 'elliptical'))))
print('Total validation lenticular:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data', 'validation', 'lenticular'))))
print('Total validation spiral:', len(os.listdir(os.path.join('/content/galaxy-zoo-clean/data','validation', 'spiral'))))
```

```
/content/galaxy-zoo-clean/data/train
elliptical done!
/content/galaxy-zoo-clean/data/train
lenticular done!
/content/galaxy-zoo-clean/data/train
spiral done!
Elliptical: 5848
Total train lenticular: 5300
Total train spiral: 3708
Total validation elliptical: 1463
Total validation lenticular: 1325
Total validation spiral: 927
```

Figure 6: Spliting data into train and test

4.3 Data augmentation

Data augmentation is performed using the ImageDataGenerator class. Imagedata generator can be used to shift, rotate, brighten and zooms the images. In this research, it is used to recale, rotate, widhth and height shift, horizontal flip is performed.

```
#data augmentation
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
                    rescale=1.0/255,
                    rotation_range=25,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.2)
validation_datagen  = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1.0/255.)

train_generator = train_datagen.flow_from_directory(train_directory,
                                                    target_size=(224,224),
                                                    batch_size=BS,
                                                    shuffle=True,
                                                    class_mode='categorical')
validation_generator = train_datagen.flow_from_directory(validation_directory,
                                                         target_size=(224,224),
                                                         batch_size=BS,
                                                         shuffle=True,
                                                         class_mode='categorical')
train_steps = np.ceil(train_generator.samples / train_generator.batch_size)
val_steps = np.ceil(validation_generator.samples / validation_generator.batch_size)
```

```
Found 14856 images belonging to 3 classes.
Found 3715 images belonging to 3 classes.
```

Figure 7: Performing data augmentation

4.4 Model building

To build the models transfer learning techniques are used which are trained ont he image net dataset. Vgg19, densenet121,inceptionv3 are used as a base model and layers are added to the given models. Keras API is used to import the VGG19, densenet121, inceprtion v3.

```
from tensorflow.keras.applications.inception_v3 import InceptionV3

base_model_inceptionv3 = InceptionV3(input_shape=(224,224,3),
                include_top=False,
                weights="imagenet")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
87924736/87910968 [==============================] - 1s 0us/step
```

Figure 8: Inceptionv3 import from keras

```
from tensorflow.keras.layers import InputLayer, Dense, Flatten, BatchNormalization, Dropout, Activation
inceptionv3_model=Sequential()
inceptionv3_model.add(base_model_inceptionv3)
inceptionv3_model.add(GlobalAveragePooling2D()),
inceptionv3_model.add(Dropout(0.2))
inceptionv3_model.add(Flatten())
inceptionv3_model.add(BatchNormalization())
inceptionv3_model.add(Dense(1024,kernel_initializer='he_uniform'))
inceptionv3_model.add(BatchNormalization())
inceptionv3_model.add(Activation('relu'))
inceptionv3_model.add(Dropout(0.2))
inceptionv3_model.add(Dense(1024,kernel_initializer='he_uniform'))
inceptionv3_model.add(BatchNormalization())
inceptionv3_model.add(Activation('relu'))
inceptionv3_model.add(Dropout(0.8))
inceptionv3_model.add(Dense(3,activation='softmax'))
inceptionv3_model.summary()
```

Figure 9: Layers added to the model

```
model = tf.keras.models.Sequential([
    # first convolution layer, input is an 180x180 image x3 colors
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu',input_shape=(180, 180, 3)),

    tf.keras.layers.MaxPooling2D(2, 2),
    # second convolution layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),
    # third convolution layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # fourth convolution layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # flatten the image pixels
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.7),
    # 512 neuron fully connected hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Figure 10: Layers of CNN model built

4.5 Model compilation

The models are compiled using categorical_crossentropy as it is multiple class classification. The metrics used are accuracy, AUC, Precision, Recall and adam optimiser is used.

```
base_learning_rate = 0.0001

inceptionv3_model.compile(loss='categorical_crossentropy',
              metrics=['accuracy','AUC','Precision','Recall'],
              optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate))
```

Figure 11: Model compilation

4.6 Generation of results

The results are generated using the fit function to get the appropriate results.

```
model_history=inceptionv3_model .fit(train_generator,
                          validation_data=validation_generator,
                          epochs = 20)
```

Figure 12: Model compilation

4.7 Model Evaluation

The model is evaluated using the accuracy, precision and recall. The graph of the training and validation accuracy, precision and recall are plotted against the epoch.

```
acc = model_history.history['accuracy']
val_acc = model_history.history['val_accuracy']

loss = model_history.history['loss']
val_loss = model_history.history['val_loss']


epochs_range = range(len(acc))  # range for the number of epochs

plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.savefig('./plots-v2.png')


plt.show()
```

Figure 13: Code for accuracy and loss

Figure 14: Training and validation accuracy and loss

```
recall=model_history.history['recall']
val_recall=model_history.history['val_recall']

precision=model_history.history['precision']
val_precision=model_history.history['val_precision']

epochs_range = range(len(acc))

plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, recall, label='Training Recall')
plt.plot(epochs_range, val_recall, label='Validation Recall')
plt.legend(loc='lower right')
plt.title('Training and Validation Recall')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, precision, label='Training Precision')
plt.plot(epochs_range, val_precision, label='Validation Precision')
plt.legend(loc='upper right')
plt.title('Training and Validation Precision')
plt.savefig('./plots-v2.png')


plt.show()
```

Figure 15: Code for plotting recall and precision

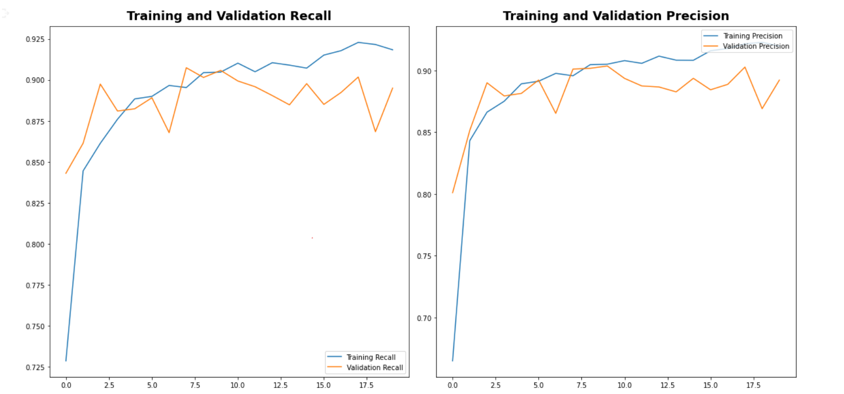

Figure 16: Training and validation recall and precision