National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Prajwal S Mastamardi
Student ID: x19158718

School of Computing
National College of Ireland

Supervisor: Dr. Barry Haycock

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Prajwal S M		
Student ID:	X19158718		
Programme:	MSc in Data Analytics	**Year:**	2021-2022
Module:	MSc Research Project		
Lecturer:	Dr. Barry Haycock.		
Submission Due Date:	16/12/12		
Project Title:	Anomaly identification in chest radiography		
Word Count:	600	**Page Count: 5**	

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 Manual

Prajwal S M
Student ID: x19158718

1 Introduction

This setup guide will assist you in recreating the research "Lung Cancer Detection Using Classification Algorithms" from the ground up. This setup manual provides thorough information on the prerequisites for setting up, running, and testing this study using the specified framework.

2 Hardware Specification

- Operating System: Windows 10 Home Single Language (10.0, Build 18362)
- Processor: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
- Installed RAM: 8.00 GB
- System Type: 64-bit OS, x64-based processor

3 Python Environment

Since the project was entirely written in Python, the Anaconda framework was used. Jupyter lab, Jupyter Notebook, R Studio,VS Code are among other preloaded environments in the Anaconda framework.
The Jupyter Notebook environment is launched as seen in Figure 1. Jupyter provides a user interface for writing code, creating models, and testing them

4 Libraries required

[1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
import tensorflow as tf
import tensorflow
from tensorflow import keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Flatten, Dense, MaxPool2D, Dropout
from tensorflow.keras.models import Model
import numpy as np
import pandas as pd
import time
import os
import glob
from pathlib import Path
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import cv2
import shutil
from pandas import DataFrame
from sklearn.metrics import confusion_matrix, classification_report
pd.set_option('display.width', 150)
import seaborn as sns
```

5 Dataset

There are 3616 COVID-19 positive instances in the database, as well as 10,192 Normal, 6012 Lung Opacity (Non-COVID lung infection), and 1345 Viral Pneumonia photos. As fresh x-ray images for COVID-19 pneumonia patients become available, author continue to update this database.

Database link: https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

6 Loading the Dataset

Case 1: In this scenario the transfer learning algorithm such as VGG-16 and ResNet-50 were utilized to differentiate four classes namely Covid, Pneumonia, Non-covid infection (lung opacity) and Normal healthy lung images.

Case 2: In this experiment transfer learning algorithms were applied to classify three classes Covid, Pneumonia and normal. Dataset utilized to train model consist of Covid, pneumonia and normal.

Case 3: For this experiment, Anomaly dataset was created to train transfer learning algorithms, Anomaly dataset consist of normal images and anomaly set, anomaly set consist of images from Covid, pneumonia , and non-Covid infection. transfer learning algorithms were applied to differentiate between classes of normal and anomaly(infected), dataset consist of approx. 10,000 images of normal and anomaly images.

Other cases: dataset consisting of two classes namely normal and covid images were created for training the model.

Model training:
Dataset loading

In [17]:
```
train_images = train_generator.flow_from_dataframe(
    dataframe=df_train,
    x_col='path',
    y_col='label',
    color_mode='rgb',
    class_mode='categorical',
    target_size=(299, 299),
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=seed,
    subset='training'
)
```

```
Found 15239 validated image filenames belonging to 2 classes.
```

In [18]:
```
val_images = train_generator.flow_from_dataframe(
    dataframe=df_train,
    x_col='path',
    y_col='label',
    color_mode='rgb',
    class_mode='categorical',
    target_size=(299, 299),
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=seed,
    subset='validation'
)
```

```
Found 3809 validated image filenames belonging to 2 classes.
```

VGG-16

VGG-16

In [20]:
```
vgg = VGG16(input_shape = [299,299,3], weights='imagenet', include_top=False)
```

In [21]:
```
# don't train existing weights
for layer in vgg.layers:
    layer.trainable = False
```

In [22]:
```
# our layers - you can add more if you want
x = Flatten()(vgg.output)

prediction1 = Dense(2, activation='sigmoid')(x)
```

In [23]:
```
# create a model object
model1 = Model(inputs=vgg.input, outputs=prediction1)
```

In [27]:
```
# tell the model what cost and optimization method to use
model1.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics= "accuracy"
)
```

In [28]:
```
ACCURACY_THRESHOLD = 0.99
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > ACCURACY_THRESHOLD):
          print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD*100))
          self.model.stop_training = True

es  = myCallback()
```

In [29]:
```
history2 = model1.fit(
```

Plotting

:
```
#plotting training values
sns.set()

acc = history2.history['accuracy']
val_acc = history2.history['val_accuracy']
loss = history2.history['loss']
val_loss = history2.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

ResNet-50

```
res  = ResNet50( input_shape=(299,299,3), include_top=False)
```

```
patience = 1
stop_patience = 5
factor = 0.5

callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss', verbose=1, restore_best_weights=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, patience=patience, verbose=1)
]
```

```
# We won't train all parameters again
for layer in res.layers:
  layer.trainable = False
```

```
x2 = Flatten()(res.output)
out = Dense(units=2, activation='sigmoid', name='predictions_res')(x2)


# Creating our model
model2 = Model(inputs=res.input, outputs=out)
```

```
ACCURACY_THRESHOLD = 1.00
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > ACCURACY_THRESHOLD):
          print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD*100))
          self.model.stop_training = True

es  = myCallback()
```

```
model2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Common test data evaluation:

Common test data

```
predictions4 = model2.predict(x = test_data2)
```

```
y_pred=np.argmax(predictions4,axis=-1)
y_pred
count =0
wrong=0
for i in y_pred:
    if i ==1:
        count=count+1
    else: wrong = wrong+1
print(count)
print(wrong)
```

```
570
153
```

```
test_data22 = tf.keras.preprocessing.image_dataset_from_directory(
   "C:/Users/prjwl/Desktop/cov_test",

    color_mode="rgb",

    image_size=(299, 299),

)
```

```
Found 723 files belonging to 1 classes.
```

```
predictions4 = model2.predict(x = test_data22)
```

```
y_pred=np.argmax(predictions4,axis=-1)
y_pred
count =0
wrong=0
for i in y_pred:
    if i ==1:
        count=count+1
    else: wrong = wrong+1
print(count," accurately predicted covid images")
print(wrong," wrongly predicted")
```

```
570  accurately predicted covid images
153  wrongly predicted
```

```
predictions_vgg = model1.predict(x = test_data22)
```

```
y_pred=np.argmax(predictions_vgg,axis=-1)
y_pred
counts =0
wrongs=0
for i in y_pred:
    if i ==1:
        counts=counts+1
    else: wrongs = wrongs+1
print(counts)
print(wrongs)
```

```
535
188
```

References

1. Dataset Source :https://www.kaggle.com/tawsifurrahman/covid19-radiography-database/code
2. Code Reference for data loading: https://www.kaggle.com/azharabdulaziz/worodproject95
3. other net sources for implementation(ex:vizaulizations)
4. Code Reference for Tensorflow Learning: https://www.tensorflow.org/api docs/python/tf
5. Code Reference for Learning Keras : https://www.tensorflow.org/api docs/python/tf/keras/