National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Jomol Payyapilly Jonny
Student ID: x20249250

School of Computing
National College of Ireland

Supervisor: Taimur Hafeez

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jomol Payyapilly Jonny
Student ID:	x20249250
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Taimur Hafeez
Submission Due Date:	15/12/22
Project Title:	Applying Deep Learning Techniques for Alzheimer's disease Classification: A comparative study
Word Count:	XXX
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:	1st February 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):	

Configuration Manual

Jomol Payyapilly Jonny
x20249250

I used my desktop computer to carry out this deep learning investigation. I used python programming language to carry out my project analysis. Jupyter Notebook served as my IDE because it offers a user-friendly and interactive environment. The advantages of utilizing a Jupyter notebook include minimum setup requirements and ideal for quick data analysis. A deep learning framework for Python called Keras incorporates features from other libraries like Tensorflow. In comparison to competitors like Scikit-learn and PyTorch, Keras has an advantage. The most popular deep learning and deep learning library, Tensorflow, can be used as an alternative. The steps in my analysis are listed below.

1 Importing and loading of images

Some of the libraries used for importing and loading images. Pillow Harshada et al. (2016) is a Python package used to load and manipulate image data. The Python Image Library, or PIL, has been updated as Pillow, and it now provides a variety of basic and advanced image editing features. Furthermore, it serves as the foundation for basic image support in other Python libraries like SciPy and Matplotlib.

- PIL: The Pillow library includes all the fundamental features for image processing.
- Pandas :- Python's Pandas package is used to manipulate data sets. It offers tools for data exploration, cleaning, analysis, and manipulation.
- Tensorflow: The TensorFlow platform supports implementing best ways for data processing, modeling , performance monitoring, and model training.

Figure 1:

```
# load image
import PIL
from PIL import Image
print('Pillow Version:', PIL.__version__)
from glob import glob
from tqdm import tqdm
from matplotlib import image
from matplotlib import pyplot
from os import listdir

import warnings
import pandas as pd
from sklearn.metrics import f1_score,accuracy_score,recall_score,roc_auc_score,roc_curve
warnings.simplefilter('ignore')
```

This above mentioned code includes libraries which allows to import and load the images.

2 Pre–processing

The preprocessing of the images includes changing the image's size, specifying hyper-parameters, setting global variables to gather images for analysis, trying to examine the images prior to dividing the dataset, plotting the images in accordance with the labels, converting into arrays, and normalizing the color values. I completed this challenge using the Keras library John Joseph et al. (2021). It has an imagedatagenerator class that has some characteristics that makes it possible to carry out this activity.

- Matplotlib: This library was used for interactive visualization.
- keras:- Used keras library to do this task. It contains imagedatagenerator class which allows to perform this task, with some attributes.
- Scikit-Learn :- Tools for modifying model hyperparameters are provided by the Scikit-Learn deep learning package.
- pillow: This library is used for resizing image.
- NumPy: It is used for converting images to arrays. In my analysis, i firstly used PIL library to read images.
- Seaborn:- this library is used for exploratory data analysis and data visualization.

Figure 2:

```
#Redefining size of each image of the dataset
IMAGE_SIZE = [229,229]

#Defining hyperparameters for training
epochs = 5
batch_size = 15
```

```
#image data processing
#Defining paths
dement_path = r'C:\Users\HP\Desktop\final project\data\demented'
nondement_path = r'C:\Users\HP\Desktop\final project\data\NonDemented'
#Use glob to pull the images
NonDemfiles = glob( nondement_path +'/*' )
Demfiles = glob( dement_path + '/*' )
#Visualize file variable images
print("First 5 NonDem Files: ",NonDemfiles[0:5])
print("Total Count: ",len(NonDemfiles))
print("First 5 Dem Files: ",Demfiles[0:5])
print("Total Count: ",len(Demfiles))
```

The above mentioned code describes how the hyper parameters are defined for training the data. It also shows how the size of the image in the dataset is changed. This step is very significant before training the data.

The below mentioned code explains how the images are converted to arrays using numpy library.

Figure 3:

```
Dem_images = np.array(Dem_images) / 255
NonDem_images = np.array(NonDem_images) / 255
```

3 Builting model

Data splitted into a training and test set that is preparing the training dataset Bovolo and Bruzzone (2007). Importing the Inception model, making it visible, and producing training data Array of categories and image pixel values. Here, i used different epoch and batch values to train the model and also increased batch and epoch values to test for more accuracy. When my model was given different epoch and batch size parameters, there was a notable difference.

- Tensorflow : The TensorFlow platform supports in implementing bet ways for data processing, modeling , performance monitoring, and model training.
- Torch: It contains wide range of algorithms for deep learning.
- Pathlib : It is a library used for creating paths.

Figure 4:

```
train_aug = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)
```

The above code explains the augumention method performed for improving the performance of classification. It is done using imagedatagenerater class from keras library. here I set values for for all the attributes under the class.

Figure 5:

```
#Saving trained model and weights
model.save('vgg19_dem.h5')
model.save_weights('vgg19_weights_dem.hdf5')
#Now Load the saved model
model = load_model('vgg19_dem.h5')
#Predicting trained model on test set*
y_pred = model.predict(X_test, batch_size=batch_size)
#*Visualizing predicted classes*
prediction=y_pred[0:10]
for index, probability in enumerate(prediction):
  if probability[1] > 0.5:
        plt.title('%.2f' % (probability[1]*100) + '% Demented')
  else:
        plt.title('%.2f' % ((1-probability[1])*100) + '% NonDemented')
  plt.style.reload_library
  plt.imshow(X_test[index])
  plt.show()
```

The set of codes above explains the trained model and how it is saved, also show how the model is predicted using test set of data where batch size is also mentioned while predicting.The CNN model produced an accuracy of 93 percent. F1 score 0.93 that is this model has false positives and low false negatives.It also visualises the predicted class that is it classify the disease into demented and non demented. Already dataset was preprocessed and splited for CNN model, which was used to run a TL model. Steps involved are the following:

- create a prefetch dataset for better performance.
- Rescale image to fit into model
- Imported and loaded base model vgg16
- Set variables for classifier and output layer
- Reconstruct and checked model
- Train the model initial epoch= 50

Figure 6:

```
#set variable for classfier layer and  output layer
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
nClass = len(class_names)
prediction_layer = tf.keras.layers.Dense(nClass, activation = 'softmax')
```

Classifier is used to classify images based on attributes that are identified and has completely connected layers. The code above illustrates how to set a variable for the output layer as well as the classifier layer. Here, the output layer is the prediction layer, while the classifier layer is the global average layer.

Figure 7:

```
# start train the model
initial_epochs = 50
history = model.fit(train_data,
                    epochs = initial_epochs,
                    validation_data = val_data,
                    callbacks = [tensorboard_callback, es_callback])
```

```
Epoch 1/50
512/512 [==============================] - 777s 2s/step - loss: 0.7572 - accuracy: 0.8098 - val_loss: 0.1911 - val_accuracy: 0.
9266
Epoch 2/50
512/512 [==============================] - 500s 977ms/step - loss: 0.4131 - accuracy: 0.8736 - val_loss: 0.1369 - val_accuracy:
0.9484
Epoch 3/50
512/512 [==============================] - 469s 915ms/step - loss: 0.3309 - accuracy: 0.9006 - val_loss: 0.1236 - val_accuracy:
0.9563
Epoch 4/50
512/512 [==============================] - 559s 1s/step - loss: 0.2780 - accuracy: 0.9148 - val_loss: 0.1105 - val_accuracy: 0.
9594
Epoch 5/50
512/512 [==============================] - 543s 1s/step - loss: 0.2207 - accuracy: 0.9264 - val_loss: 0.1053 - val_accuracy: 0.
```

After evaluation of transfer learning model it showed an accuracy of 0.98 percentage From both the model transfer learning model gave an accuracy above 0.90 percentage when the hyperparameters are changed. However for CNN model the results varied when the hyperparameters are changed. So as from the analysis, I have faced an overfitting problem when the increased the epoch values while training the model. I think that was one of the major challenge I faced during my analysis.

Figure 8:

```
# Now, we resume the model trianing from the last epochs
fine_tune_epochs = 5
total_epochs = initial_epochs + fine_tune_epochs

# train the model again
histroy_fine_tune = model.fit(train_data,
                              epochs = total_epochs,
                              initial_epoch = history.epoch[-1],
                              validation_data = val_data,
                              callbacks = [tensorboard_callback, es_callback])
```

```
Epoch 5/10
512/512 [==============================] - 996s 2s/step - loss: 0.0204 - accuracy: 0.9926 - val_loss: 0.0335 - val_accuracy: 0.
9859
Epoch 6/10
512/512 [==============================] - 1034s 2s/step - loss: 0.0125 - accuracy: 0.9953 - val_loss: 0.0429 - val_accuracy:
0.9891
Epoch 7/10
512/512 [==============================] - 1027s 2s/step - loss: 0.0103 - accuracy: 0.9965 - val_loss: 0.0541 - val_accuracy:
0.9906
Epoch 8/10
512/512 [==============================] - 1078s 2s/step - loss: 0.0096 - accuracy: 0.9967 - val_loss: 0.1154 - val_accuracy:
0.9812
Epoch 9/10
512/512 [==============================] - 1038s 2s/step - loss: 0.0073 - accuracy: 0.9979 - val_loss: 0.0555 - val_accuracy:
```

Figure 9:

```
# Evaluate the model
print("----------------------------------After Fine-tuning model.....................................")
model.evaluate(test_data)
```

```
----------------------------------After Fine-tuning model.....................................
64/64 [==============================] - 45s 709ms/step - loss: 0.0301 - accuracy: 0.9875

[0.03008337877690792, 0.987500011920929]
```

References

Bovolo, F. and Bruzzone, L. (2007). A split-based approach to unsupervised change detection in large-size multitemporal images: Application to tsunami-damage assessment, *Geoscience and Remote Sensing, IEEE Transactions on* **45**: 1658 – 1670.

Harshada, M., Snehal, Shitole, S., Pranaya, M., Suchita, K., Shweta, G. and Kurle, D. (2016). Python based image processing.

John Joseph, F. J., Nonsiri, S. and Monsakul, A. (2021). *Keras and TensorFlow: A Hands-On Experience*, pp. 85–111.