Configuration Manual

MSc Research Project
Data Analytics

Nevin Saini
Student ID: x18132260

School of Computing
National College of Ireland

Supervisor: Dr. Cristina Muntean
Student Name: Nevin Saini
Student ID: x18132260
Programme: Data Analytics
Year: 2018-19
Module: MSc. Research Project
Lecturer: Dr. Cristina Muntean
Submission Due Date: 12/12/2019
Project Title: Configuration Manual
Word Count: 1116
Page Count: 16
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: Nevin Saini
Date: 12/12/2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Checkmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach a completed copy of this sheet to each project (including multiple copies)</td>
<td>X</td>
</tr>
<tr>
<td>Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies)</td>
<td>X</td>
</tr>
<tr>
<td>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.</td>
<td>x</td>
</tr>
</tbody>
</table>

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

Penalty Applied (if applicable):
1 Introduction

The configuration manual contains all the pertinent information related to the software and hardware used in the research project. Also, it specifies the important libraries that are used and the data description is given in section 3. Moreover, it elucidates several steps that need to be taken to reproduce the work in any machine satisfying the requirements which is covered in the following sections.

2 Environment Specifications

The MSc. Research project runs on a system which has certain specifications for both software and hardware that are described in the following subsections.

2.1 Hardware Specifications

This project is implemented on the hardware with the following configurations:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>HP Envy</td>
</tr>
<tr>
<td>Operation System</td>
<td>Windows 10 (64-bit Operating system)</td>
</tr>
<tr>
<td>RAM</td>
<td>8 GB</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>512 GB (SSD)</td>
</tr>
<tr>
<td>Graphic Card</td>
<td>2 GB Ryzen</td>
</tr>
</tbody>
</table>
2.2 Software Specifications

In this project, plethora of softwares are used which are represented in Table 2.

<table>
<thead>
<tr>
<th>Software</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows 10 (64-bit Operating system)</td>
</tr>
<tr>
<td>IDE</td>
<td>Spyder (Anaconda Navigator)</td>
</tr>
<tr>
<td>Scripting Language</td>
<td>Python</td>
</tr>
<tr>
<td>Scripting Language version</td>
<td>Python 3.7</td>
</tr>
</tbody>
</table>

2.2.1 Integrated Development Environment

A latest version of Anaconda navigator is installed to implement the highly demanding models. Its installation includes series of several steps which is described below:

Step 1:

➢ Visit the website and click on the link of first step.

➢ Click on the file and install it in the system.

➢ Click Next to step further.
➢ Click on “I Agree” and then “Next” to step further in installation process

➢ It will start installation which will take some time considering the size of the software.
➢ Click on the “Next” button.

➢ Go to the search icon on the bottom left in windows 10 machine and type anaconda.

➢ Click on the specified Icon of Anaconda Navigator to launch the application.
➢ It is important to create a new environment for the project where all the useful libraries and packages can be installed.

To do this, Click on Environments.

➢ Click on create button on the bottom screen to create a new environment.
Add the name of new environment and click on create.

2.3 Libraries

After the creation of an environment in Anaconda, we have installed all the vital libraries required to execute the research project which are discussed below.

- matplotlib
- matplotlib.pyplot
- tensorflow.keras.layers
- tensorflow.keras.models
- tensorflow.keras.preprocessing
- tensorflow.keras.preprocessing
- tensorflow.keras.preprocessing
- sklearn.metrics
- sklearn.preprocessing
- sklearn.model_selection
- sklearn.grid_search
- sklearn.neighbors
- sklearn
- opencv
- skimage
- tensorflow
- skimage.io
- tensorflow.keras.preprocessing
3. Dataset

There are two datasets used for the research project which are as follows:

3.1 Yawning Dataset: It contains two datasets in video format of drivers with different facial features to detect drowsiness of drivers. 322 videos are taken in real time of different individuals. In addition, 29 more videos of both male and female drivers are taken from driver’s dash.

These images are converted into frames for both yawning and open eyes. Almost 1479 images are used for yawning dataset and 1222 frames contain images of individuals with open eyes.

3.2 Closed eyes: This contains around 1189 images of different people from diverse angles with closed eyes and different environmental changes like blur and lighting.
4. Image Pre-processing

Image Pre-processing has been done on the dataset to detect drowsiness of an ATC and considering different situations that a controller can phase in terms of lighting. It involves several independent steps like Rescaling, Rotation, Gray scaling, edge detection, flipping and is done before segmentation. According to Bazeille et al. (2010), Models shows great accuracy while filtering the images using edge detection.

4.1 CNN

```
# data augmentation to prevent overfitting
train_datagen = ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of the dataset
    samplewise_std_normalization=False,  # divide each input by its std
    zca_whitening=False,  # apply ZCA whitening
    rotation_range=18,  # randomly rotate images in the range (degrees, 0 to 180)
    zoom_range=0.1,  # Randomly zoom image
    width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False)  # randomly flip images

test_datagen = ImageDataGenerator(rescale=1./255)
```

4.2 Machine learning models

```
plt.imshow(image)

RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

scale_percent = 60  # percent of original size
width = int(RGB_img.shape[1] * scale_percent / 100)
height = int(RGB_img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
image_resized = cv2.resize(RGB_img, (70,70), interpolation = cv2.INTER_AREA)

image_gray = cv2.cvtColor(image_resized, cv2.COLOR_BGR2GRAY)

image_hsv = cv2.cvtColor(image_resized, cv2.COLOR_BGR2HSV)

image_rotated = rotate(RGB_img, angle=90)

image_flip = flipud(RGB_img)

image_bright = exposure.equalize_hist(RGB_img)

image_dark = exposure.equalize_hist(RGB_img)
```

Both image-pre-processing techniques were performed on the datasets to increase the number of images and to improve the performance of the system due to the presence of different versions of same image.
5. Model Creation

Different models are designed to implement the project in both deep and machine learning like CNN, SVM, KNN, XGBoost and Random Forest.

Machine learning techniques are designed using Scikit python library and CNN model is created by utilizing keras and Tensorflow. Through the results, It is observed that all the algorithms have performed well, however, Deep learning robust model CNN has outperformed other techniques based on some performance measures like Accuracy, Precision, Recall and F1 Score.

The execution of these techniques is done on two models i.e. Baseline Models and Tuned Models (after hyperparameter tuning) that are shown below.

5.1 Baseline Models

5.1.1 Convolutional Neural Network

```python
# Preparing layers of an improved CNN model
#
# improvedmodel = Sequential()
# improvedmodel.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
# improvedmodel.add(MaxPool2D(pool_size=(2, 2)))
# improvedmodel.add(Dropout(rate=0.25))
# improvedmodel.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
# improvedmodel.add(MaxPool2D(pool_size=(2, 2)))
# improvedmodel.add(Dropout(rate=0.25))
# improvedmodel.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
# improvedmodel.add(MaxPool2D(pool_size=(2, 2)))
# improvedmodel.add(Dropout(rate=0.25))
# improvedmodel.add(Flatten())
# improvedmodel.add(Dense(256, activation='relu'))
# improvedmodel.add(Dropout(rate=0.5))
# improvedmodel.add(Dense(3, activation='softmax'))
```
5.1.2 SVM

```python
# Dividing the datasets into 75%:25% Training and Testing set ratio
X_train, X_val, y_train, y_val = train_test_split(data_scaled, labels, test_size=0.25, random_state=42)

# Designing a baseline SVM Model
model = svm.SVC()

# Training the Model on the training image dataset
model.fit(X_train, y_train)

# Predicting the Labels on testing image dataset
y_pred = model.predict(X_val)

# Calculating the Accuracy of the baseline model
accuracy = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with Baseline SVM Model: \{:.2f\}%".format(accuracy))

# Generating classification report for all the individual classes
print(classification_report(y_val, y_pred))
```

5.1.3 KNN

```python
# Dividing the datasets into 75%:25% Training and Testing set ratio
X_train, X_val, y_train, y_val = train_test_split(data, labels, test_size=0.25, random_state=42)

# Designing a baseline KNN Model
model = KNeighborsClassifier()

# Training the Model on the training image dataset
model.fit(X_train, y_train)

# Predicting the Labels on testing image dataset
y_pred = model.predict(X_val)

# Calculating the Accuracy of the baseline model
accuracy = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with K-NN: \{:.2f\}%".format(accuracy))
```

5.1.4 XGBoost
Several machine learning models require setting many hyperparameters before testing them on the validation set (Boulesteix and Bischl, 2018) and it is extremely important to search the right parameters to be used in a particular machine learning algorithm which improves the performance of a baseline model by increasing the ability to find the correct results out of the total predictions done by the model.
5.2.1 SVM

```python
# Hyperparameter tuning for SVM
# Creating all the important parameters and the possible values to be compared for the tuning
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 10]}

# Using GridSearch, search through several combinations of parameter to find the best one
tunedModel = GridSearchCV(svm.SVC(), param_grid, verbose=1)
tunedModel.fit(X_train, y_train)
tunedModel.best_params_

# Predicting the labels on testing image dataset
y_predTuned = tunedModel.predict(X_val)

# Calculating the Accuracy of the tuned model
accuracy_tuned = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with SVM: {0:.2f}%\n".format(accuracy_tuned))

# Generating classification report of the tuned Model
print(classification_report(y_val, y_pred))
```

5.2.2 KNN

```python
# Hyper Parameter tuning for KNN
# Creating all the important parameters and the possible values to be compared for the tuning
params_knn = {'n_neighbors':[5,7,8,9], 'weights':['uniform', 'distance']}

# Using GridSearch, search through several combinations of parameter to find the best one
knnModel = GridSearchCV(KNeighborsClassifier(), param_grid=params_knn)
knnModel.fit(X_train,y_train)
knnModel.best_params_

# Predicting the labels on testing image dataset
y_predTuned = knnModel.predict(X_val)

# Calculating the Accuracy of the tuned model
accuracy_tuned = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with KNN: {0:.2f}%\n".format(accuracy_tuned))

# Generating classification report of the tuned Model
print(classification_report(y_val, y_pred))
```
5.2.3 XGBoost

```python
# Hyperparameter Tuning for XGBoost
#  
# Creating all the important parameters and the possible values to be compared for the tuning
param_grid = {'learning_rate': [0.01, 0.1], 'gamma': [0.5, 0.1], 'verbosity': [0]}
#  
# Using GridSearch, searching through several combinations of parameter to find the best one

tunedModel = GridSearchCV(model, param_grid)
#  
# Using the best fitted parameters chosen, designing the tuned model

tunedModel.fit(X_train, y_train)
tunedModel.best_params_
#  
# Predicting the labels on testing image dataset
#
y_predTuned = tunedModel.predict(X_val)
#  
# Calculating the Accuracy of the tuned model
#
accuracy_tuned = metrics.accuracy_score(y_predTuned, y_val) * 100
print("Accuracy with Tuned XGBoost Model: {0:.2f}%".format(accuracy_tuned))
#  
# Generating classification report of the tuned Model
#
print(classification_report(y_val, y_pred))
```

5.2.4 Random Forest

```python
# Hyperparameter Tuning for XGBoost
#  
# Creating all the important parameters and the possible values to be compared for the tuning
params_RF = {'max_depth': [3, 5, 6, 7, 8],
            'min_samples_split': [2, 3, 10],
            'min_samples_leaf': [1, 3, 10],
            'criterion': ["gini", "entropy"]}
#  
# Using GridSearch, searching through several combinations of parameter to find the best one

tunedModel = GridSearchCV(model, param_grid=params_RF)
tunedModel.fit(X_train, y_train)
tunedModel.best_params_
#  
# Predicting the labels on testing image dataset
#
y_predTuned = tunedModel.predict(X_val)
#  
# Calculating the Accuracy of the tuned model
#
accuracy_tuned = metrics.accuracy_score(y_predTuned, y_val) * 100
print("Accuracy with Tuned K-NN: {0:.2f}%".format(accuracy_tuned))
#  
# Generating classification report of the tuned Model
#
print(classification_report(y_val, y_pred))
```
References
