Configuration Manual

Ashwin Suresh Babu
ID:19151381
MSc Research Project
MSCDAD JAN22 B

January 2022

1 Introduction

In this document, we will discuss the step-by-step process that took to complete this project from setting up the environment to the evaluation process.

2 Environment Setup

2.1 System Specification

Implement of this research was performed on the Jupyter notebook hosted locally on Macbook Pro with 8GB of RAM and 512GB SSD storage with an M1 processor which has inbuilt GPU cores.

2.2 Technical Specification

Python(3.9.13) was the programming language used to implement this project and the following packages were used.
Numpy - 1.21.5
Pandas - 1.4.4
OpenCV - 4.6.0.66
Keras - 2.9.0
TensorFlow - 2.9.1
Scikit Learn - 1.0.2

Matplotlib - 3.5.2
Sklearn Metric - 2021.6.0
Seaborn - 0.11.2

2.3 Data Source

The data is accessed locally. The data was downloaded from a public website called Kaggle. The data consists of several images that can be used to identify different characteristics of the face such as open eyes, closed eyes, yawn, and no yawn.

3 Implementation

In this section the end to end process of the implementation will be discussed from importing the necessary libraries to the results.

3.1 Importing Necessary Libraries

```
In [1]: #Importing all the necessary libraries

        import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import os
        import cv2
        import matplotlib.pyplot as plt
        from sklearn.metrics import confusion_matrix,accuracy_score
        import seaborn as sns
```

Figure 1: Necessary Libraries

3.2 Implementation Process

The execution process of the data begins with downloading the dataset to the local directory. Once it is in the local directory we then use jupyter lab to lis the directory and print out the contents of the datasets. Once that is established we visulaize each scenario of the open eyes, closed eyes, yawning, and no yawn.

Once that is done we then move onto the preprocessing part. Where we remove the background of the face in yawn and no yawn images to help the models train more easily and reduce the load on the models. then we write a function to read all the images and store it into a numpy array. We then compile both eyes open and close array and yawn and no yawn array together. Then we separate the Labels and Features of the image into two variables.

Once the preprocessing stage is done we move onto split the data to train and test models. We introduce ImageAugmentation to augment the images in random manner so that the model has more data to train and test with.

Then we move onto train our first deep learning model using TensorFlow and Keras. And using functions such as MaxPooling2D and Conv2D. After training the deep learning model we print out the results from that test.

Then we move on to append the data into 2d array so that we can model and train machine learning algorithms such as Logistic Regression and Support vector Machine(SVM).

4 Code Snippets

In this section some of the import codings have been displayed to understand the project better.

The below figuire 2 shows the code for removing the background of images of yawn and no yawn since only the face data is necessary.

The figuire 3 show the code for data augmentations so that the we can artificially create more data to make the model train more effectively.

Figure 4 explains the code for creating a function that will classify all 4 categories of the images that need to be segregated.

Figure 5 explains the code for the confusion matrix of the first model which is the deep learning model.

Figure 6 Explain the code of the machine learning techniquie Support vector

```
In [10]: #Writing a function consider only the face for Yawn and no yawn images

def face_for_yawn(direc = "/Users/ashwinsureshbabu/NCI/final project/train", face_cas_path = "/Users/ashwinsureshbab

    yaw_no = []
    IMG_SIZE = 145
    categories = ["yawn", "no_yawn"]

    for category in categories:

        path_link = os.path.join(direc, category)
        class_num1 = categories.index(category)
        print(class_num1)

        for image in os.listdir(path_link):

            image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
            face_cascade = cv2.CascadeClassifier(face_cas_path)
            faces = face_cascade.detectMultiScale(image_array, 1.3, 5)

            for (x, y, w, h) in faces:

                img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
                roi_color = img[y:y+h, x:x+w]
                resized_array = cv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))
                yaw_no.append([resized_array, class_num1])

    return yaw_no


yawn_no_yawn = face_for_yawn()

0
1
```

Figure 2: model to remove the background of the face

```
In [25]: #FEATURE TRANSFORMATION
         #Data Augmentation


         train_generator = ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, rotation_range=30)
         test_generator = ImageDataGenerator(rescale=1/255)

         train_generator = train_generator.flow(np.array(X_train), y_train, shuffle=False)
         test_generator = test_generator.flow(np.array(X_test), y_test, shuffle=False)
```

Figure 3: Data Augmentation

```
In [26]: #Defining a deep learning model to classify images into four categories

         model = Sequential()

         model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
         model.add(MaxPooling2D(2, 2))

         model.add(Conv2D(128, (3, 3), activation="relu"))
         model.add(MaxPooling2D(2, 2))

         model.add(Conv2D(64, (3, 3), activation="relu"))
         model.add(MaxPooling2D(2, 2))

         model.add(Conv2D(32, (3, 3), activation="relu"))
         model.add(MaxPooling2D(2, 2))

         model.add(Flatten())
         model.add(Dropout(0.5))

         model.add(Dense(64, activation="relu"))
         model.add(Dense(4, activation="softmax"))

         model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")

         model.summary()
```

Figure 4: Model to classify images into 4 categories

```
In [37]: from sklearn.metrics import confusion_matrix,accuracy_score

         #Generate the confusion matrix
         cf_matrix=confusion_matrix(y_test.argmax(axis=1), pred.argmax(axis=1))
         print(cf_matrix)
         print(accuracy_score(y_test.argmax(axis=1), pred.argmax(axis=1)))
```

Figure 5: Confusion matrix

Machine and Figure 7 is the code for the machine learning technique Logistic Regression.

```
In [53]: from sklearn.metrics import accuracy_score
         predictions_SVM = SVM.predict(X_test)
         # Use accuracy_score function to get the accuracy
         print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)

         SVM Accuracy Score ->  92.38754325259517
```

Figure 6: Code for Support Vector Machines

```
In [55]: from sklearn.linear_model import LogisticRegression
         logisticRegr = LogisticRegression()
         logisticRegr.fit(X_train, y_train)
         predictions_log = logisticRegr.predict(X_test)
         print("Logistic Regression Accuracy Score -> ",accuracy_score(predictions_log, y_test)*100)
```

Figure 7: Code for Logistic Regression