Configuration Manual

MSc Research Project
Data Analytics

Divyansh Anand
Student ID: 22240217

School of Computing
National College of Ireland

Supervisor: Abdul Qayum

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Divyansh Anand
Student ID:	22240217
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Abdul Qayum
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	900
Page Count:	6

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:	12th August 2024

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

Divyansh Anand
22240217

1 Introduction

This configuration manual ensures that users can easily set-up and configure system environment and required libraries to execute the speech emotion recognition code used for this research. This manual highlights and give details about the important sections of code for better understanding and readability. All the necessary information regarding folder creating and data file placement has been given in detail. This manual provides screenshots of files, code, and prompts to guide the user in the best way possible. This report itself highlights the important sections such as data preparation, pre-processing, augmentation, modeling, and visualization. Overall, this manual gives detailed implementation steps for easy understanding and seamless execution of the hybrid speech-emotion recognition model.

2 System Requirements

2.1 Hardware Requirements

- CPU: Any standard processor capable of running a browser
- GPU: For faster execution GPU is recommended. Google Collab provides free GPU resources.
- Memory: At least 4 GB of RAM is recommended
- Storage Space: Sufficient cloud storage to store the raw files and csv feature files.

2.2 Software Requirements

- Operating System: The code is platform-independent and can be executed on any operating system (Windows, macOS, Linux) via a web browser.
- Browser: Google Chrome, Mozilla Firefox, or any modern browser with JavaScript enabled.
- Google Account: Required to access Google Colab

3 Installation Instructions

All the necessary libraries required to run the code are listed below. You can use below given pip command install libraries-

pip install pandas numpy seaborn matplotlib scikit-learn ipython tqdm

Every library provides different functionalities throughout the code.

Apart from above libraries, some libraraies with given versions are required to execute the code.
!pip install keras==2.15.0 tensorflow==2.15.0 tf-keras==2.14.1 librosa==0.9.2

Reason for specific versions of libraries, librosa library has helped in preprocessing and feature extraction part. Using other version causes error, If used latest version, the feature file it creates is incorrect(contains null characters). In case of keras and tensorflow, proposed model works fast and accurate using these versions. Lower version makes the processing slow where as other versions above version 3 causes error and compatibility issues. This line of code is added in the code itself to make it easy for the user to run the entire code in a single run without the need of changing the versions.

4 Folder Architecture

After unzipping the upload, upload all the files on Google Drive and arrange as per the folder structure given in Figure 1.

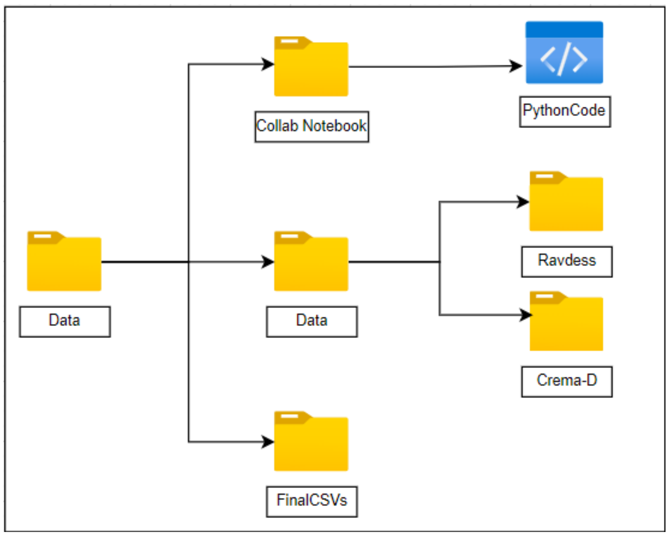
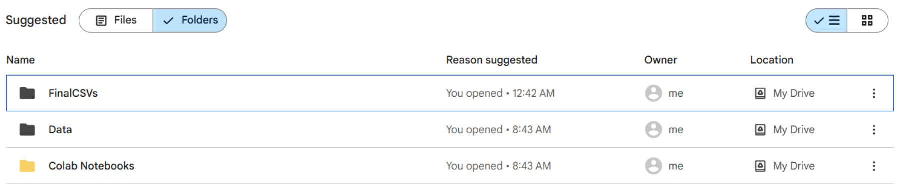


Figure 1: Folders Structure for Data and Code

Once you upload the code to the drive, by default location of the Code will be "My Drive/Collab Notebooks/HybridModelCode.ipynb"
After this, create two folders namely Data and FinalCSVs. In the Data directory upload the Ravdess and Crema D source data files. There is no need to upload anything for FinalCSVs folder. Please find the screenshot of the folders in Google Drive in Figure 2.



Name	Reason suggested	Owner	Location
FinalCSVs	You opened • 12:42 AM	me	My Drive
Data	You opened • 8:43 AM	me	My Drive
Colab Notebooks	You opened • 8:43 AM	me	My Drive

Figure 2: Folders In Google Drive

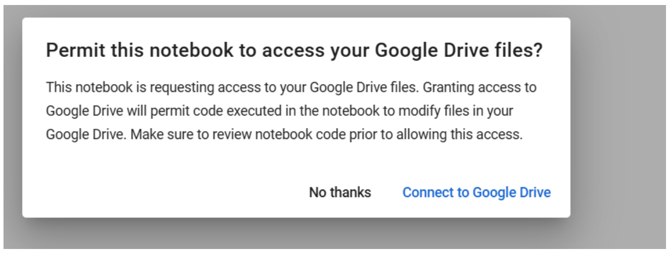
5 How to Execute Code

5.1 Pre-requisites

Make sure all the data files Ravdess, Crema-D and python code HybridModelCode.ipynb have been uploaded in the respective folders before execution of the code.

5.2 Mounting Google Drive

After executing the header files, we need to mount the Google Drive to use the uploaded Data file content. The Figure 3 represents the prompt, you need to authenticate and give Collab access to use your Google Drive.

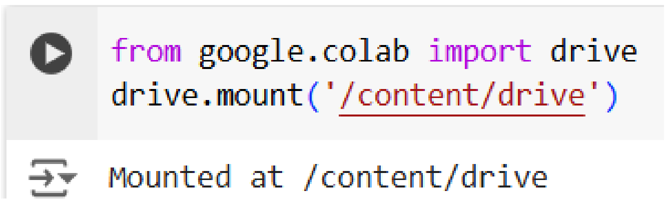


Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.

Figure 3: Prompt to Authenticate User

After successful authentication, the Google Drive is mounted as shown in Figure 4.



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

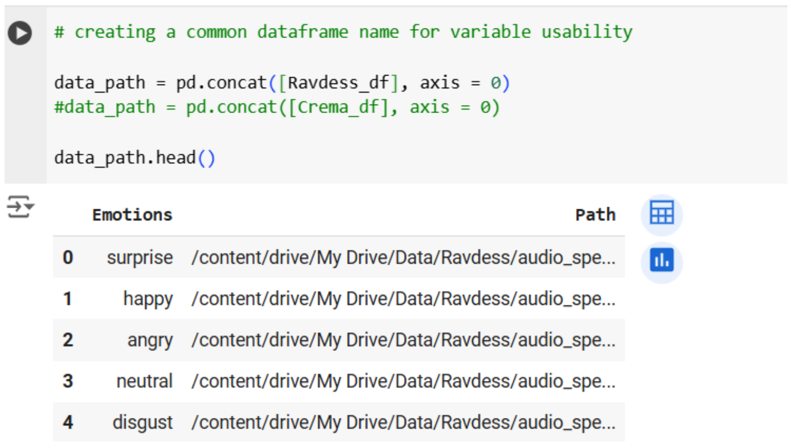
```
Mounted at /content/drive
```

Figure 4: Drive Mounted

5.3 Data Preparation and Processing

After this Data preparation starts, all the audio files from the folders are extracted and converted into dataframes for both the datasets Ravdess and Crema-D.

For Further processing, a common data frame is created. Users can choose among the dataframes for further processing. This is done to increase the re-usability of the code for processing two different datasets. By default, it will work with Ravdess dataset as shown below in Figure 5.



```
# creating a common dataframe name for variable usability

data_path = pd.concat([Ravdess_df], axis = 0)
#data_path = pd.concat([Crema_df], axis = 0)

data_path.head()
```

	Emotions	Path
0	surprise	/content/drive/My Drive/Data/Ravdess/audio_spe...
1	happy	/content/drive/My Drive/Data/Ravdess/audio_spe...
2	angry	/content/drive/My Drive/Data/Ravdess/audio_spe...
3	neutral	/content/drive/My Drive/Data/Ravdess/audio_spe...
4	disgust	/content/drive/My Drive/Data/Ravdess/audio_spe...

Figure 5: Common Data frame

During data preprocessing, the original audio is trimmed using librosa library to remove the non-speech segments of the audio data.(McFee et al. (2020)) Below Figure 6 represents the difference in waveplots and spectrograms before and after data preprocessing step for angry emotion.

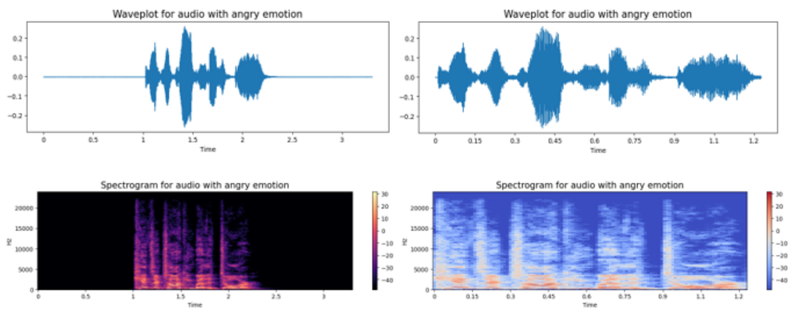


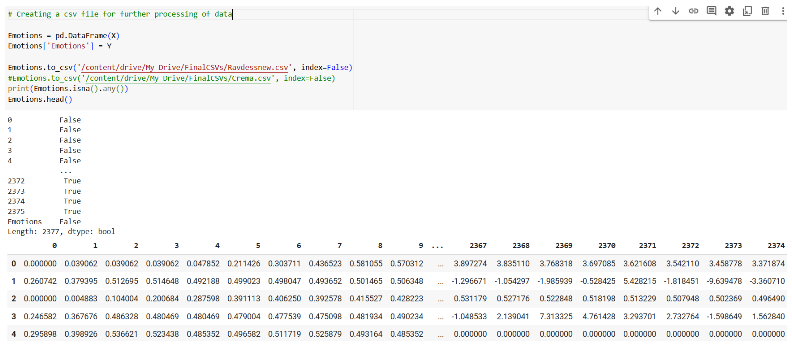
Figure 6: Before and After Preprocessing of Data

5.4 Feature Extraction and Data Augmentation

After the data preprocessing steps, the feature extraction and data augmentation part of the code executes, which extracts all the audio features from the speech data. Feature file is then converted into csv file for further processing of the data. Figure 7 shows the steps where csv file is created for extracted features. We can choose to rename the csv file as per the dataset used.

5.5 Data Modelling

The data stored in csv file is extracted and loaded for data modeling. During this phase, datasets are divided into training and testing datasets. After converting the data into an appropriate form it is processed by model.



```
# Creating a csv file for further processing of data

Emotions = pd.DataFrame(X)
Emotions['Emotions'] = Y

Emotions.to_csv('/content/drive/My Drive/FinalCSVs/Ravdessnew.csv', index=False)
#Emotions.to_csv('/content/drive/My Drive/FinalCSVs/Crema.csv', index=False)
print(Emotions.isna().any())
Emotions.head()
```

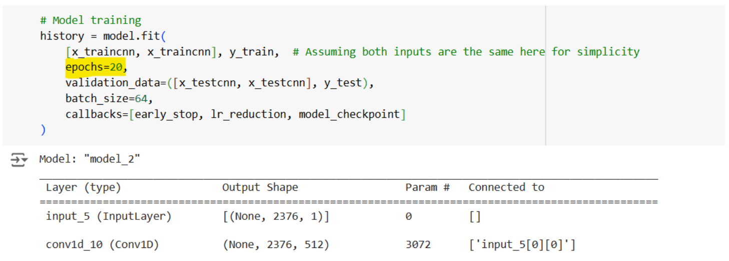
```
0           False
1           False
2           False
3           False
4           False
            ...
2372         True
2373         True
2374         True
2375         True
Emotions    False
Length: 2377, dtype: bool
```

	0	1	2	3	4	5	6	7	8	9	...	2367	2368	2369	2370	2371	2372	2373	2374
0	0.000000	0.039062	0.039062	0.039062	0.047852	0.211426	0.303711	0.436523	0.581055	0.570312	...	3.897274	3.835110	3.768318	3.697085	3.621608	3.542110	3.458778	3.371874
1	0.260742	0.379395	0.512695	0.514648	0.492188	0.499023	0.498047	0.493652	0.501465	0.506348	...	-1.296671	-1.054297	-1.985939	-0.528425	5.428215	-1.818451	-9.639478	-3.360710
2	0.000000	0.004883	0.104004	0.200684	0.287598	0.391113	0.406250	0.392578	0.415527	0.428223	...	0.531179	0.527176	0.522848	0.518198	0.513229	0.507948	0.502369	0.496490
3	0.246582	0.367676	0.486328	0.480469	0.480469	0.479004	0.477539	0.475098	0.481934	0.490234	...	-1.048533	2.139041	7.313325	4.761428	3.293701	2.732764	-1.598649	1.562840
4	0.295898	0.398926	0.536621	0.523438	0.485352	0.496582	0.511719	0.525879	0.493164	0.485352	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 7: Creating CSV file for the extracted features from the Dataset

The Neural Network architecture is trained using an iterative process where the model learns from the data step by step.

By default epoch value is set to 20 for running Ravdess dataset, the epoch value plays an important role during model training, it signifies the number of times the dataset will be processed during the training phase. The epoch value should be changed to 50 epochs while running the same code for Crema-D dataset which is bigger as compared to Ravdess. Figure 8 highlights the part of code that can be modified to run Crema D dataset.



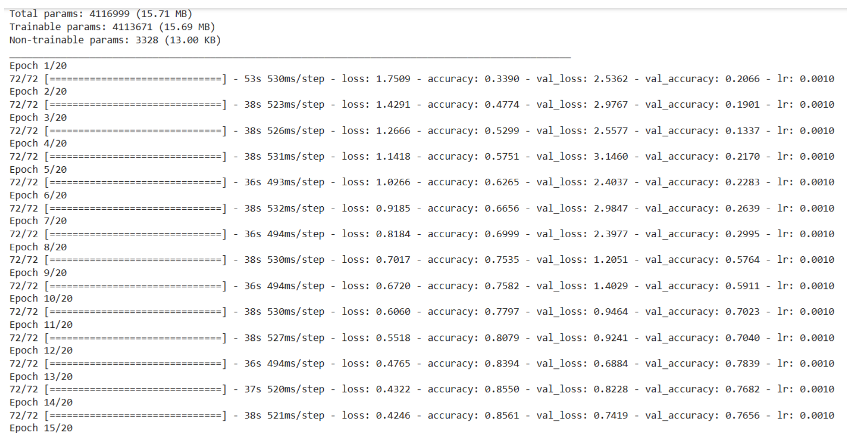
```
# Model training
history = model.fit(
    [x_traincnn, x_traincnn], y_train,  # Assuming both inputs are the same here for simplicity
    epochs=20,
    validation_data=([x_testcnn, x_testcnn], y_test),
    batch_size=64,
    callbacks=[early_stop, lr_reduction, model_checkpoint]
)
```

```
Model: "model_2"
 ________________________________________________________________________________________________
 Layer (type)               Output Shape              Param #   Connected to
================================================================================================
 input_5 (InputLayer)       [(None, 2376, 1)]         0         []

 conv1d_10 (Conv1D)         (None, 2376, 512)         3072      ['input_5[0][0]']
```

Figure 8: To Update Epoch Value

Below Figure 9 shows the sample of Epochs while running for Ravdess dataset.

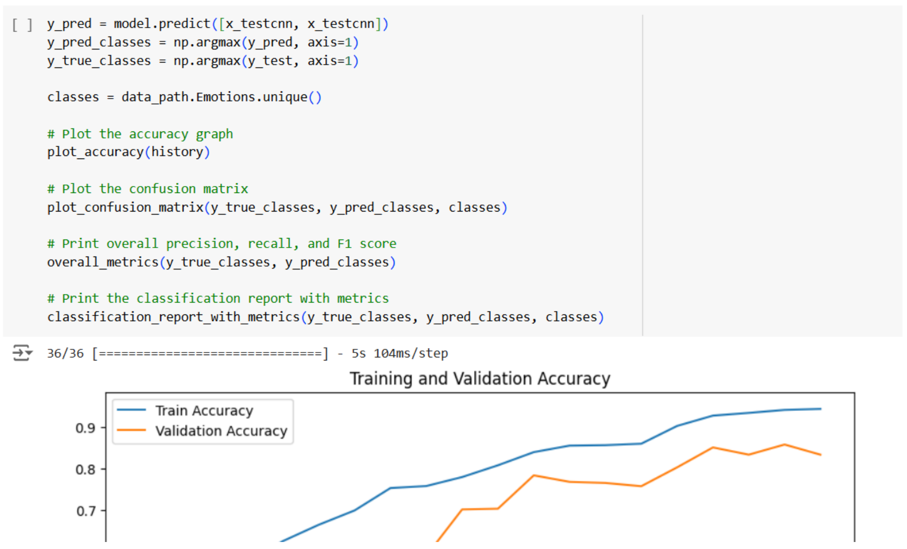


```
Total params: 4116999 (15.71 MB)
Trainable params: 4113671 (15.69 MB)
Non-trainable params: 3328 (13.00 KB)
_________________________________________________________________
Epoch 1/20
72/72 [==============================] - 53s 530ms/step - loss: 1.7509 - accuracy: 0.3390 - val_loss: 2.5362 - val_accuracy: 0.2066 - lr: 0.0010
Epoch 2/20
72/72 [==============================] - 38s 523ms/step - loss: 1.4291 - accuracy: 0.4774 - val_loss: 2.9767 - val_accuracy: 0.1901 - lr: 0.0010
Epoch 3/20
72/72 [==============================] - 38s 526ms/step - loss: 1.2666 - accuracy: 0.5299 - val_loss: 2.5577 - val_accuracy: 0.1337 - lr: 0.0010
Epoch 4/20
72/72 [==============================] - 38s 531ms/step - loss: 1.1418 - accuracy: 0.5751 - val_loss: 3.1460 - val_accuracy: 0.2170 - lr: 0.0010
Epoch 5/20
72/72 [==============================] - 36s 493ms/step - loss: 1.0266 - accuracy: 0.6265 - val_loss: 2.4037 - val_accuracy: 0.2283 - lr: 0.0010
Epoch 6/20
72/72 [==============================] - 38s 532ms/step - loss: 0.9185 - accuracy: 0.6656 - val_loss: 2.9847 - val_accuracy: 0.2639 - lr: 0.0010
Epoch 7/20
72/72 [==============================] - 36s 494ms/step - loss: 0.8184 - accuracy: 0.6999 - val_loss: 2.3977 - val_accuracy: 0.2995 - lr: 0.0010
Epoch 8/20
72/72 [==============================] - 38s 530ms/step - loss: 0.7017 - accuracy: 0.7535 - val_loss: 1.2051 - val_accuracy: 0.5764 - lr: 0.0010
Epoch 9/20
72/72 [==============================] - 36s 494ms/step - loss: 0.6720 - accuracy: 0.7582 - val_loss: 1.4029 - val_accuracy: 0.5911 - lr: 0.0010
Epoch 10/20
72/72 [==============================] - 38s 530ms/step - loss: 0.6060 - accuracy: 0.7797 - val_loss: 0.9464 - val_accuracy: 0.7023 - lr: 0.0010
Epoch 11/20
72/72 [==============================] - 38s 527ms/step - loss: 0.5518 - accuracy: 0.8079 - val_loss: 0.9241 - val_accuracy: 0.7040 - lr: 0.0010
Epoch 12/20
72/72 [==============================] - 36s 494ms/step - loss: 0.4765 - accuracy: 0.8394 - val_loss: 0.6884 - val_accuracy: 0.7839 - lr: 0.0010
Epoch 13/20
72/72 [==============================] - 37s 520ms/step - loss: 0.4322 - accuracy: 0.8550 - val_loss: 0.8228 - val_accuracy: 0.7682 - lr: 0.0010
Epoch 14/20
72/72 [==============================] - 38s 521ms/step - loss: 0.4246 - accuracy: 0.8561 - val_loss: 0.7419 - val_accuracy: 0.7656 - lr: 0.0010
Epoch 15/20
```

Figure 9: Ravdess Sample Epochs

5.6 Visualization for Model Evaluation

Further code highlights the functions that create all the graphs and matrices used to evaluate the performance of the model in this research. All these evaluation matrices have been plotted using a single cell, given in Figure 10.



```
[ ] y_pred = model.predict([x_testcnn, x_testcnn])
    y_pred_classes = np.argmax(y_pred, axis=1)
    y_true_classes = np.argmax(y_test, axis=1)

    classes = data_path.Emotions.unique()

    # Plot the accuracy graph
    plot_accuracy(history)

    # Plot the confusion matrix
    plot_confusion_matrix(y_true_classes, y_pred_classes, classes)

    # Print overall precision, recall, and F1 score
    overall_metrics(y_true_classes, y_pred_classes)

    # Print the classification report with metrics
    classification_report_with_metrics(y_true_classes, y_pred_classes, classes)

36/36 [==============================] - 5s 104ms/step
```

Figure 10: Sample Code for Plotting Visualization

References

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E. and Nieto, O. (2020). librosa/librosa: 0.8. 0 (version 0.8. 0), *Zenodo, Jul* **22**.