National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Noel Viji Thaliath
Student ID: x22185178

School of Computing
National College of Ireland

Supervisor: Dr. Muslim Jameel Syed

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Noel Viji Thaliath
Student ID:	x22185178
Programme:	Artificial Intelligence
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Muslim Jameel Syed
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	670
Page Count:	9

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:	14th January 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

Noel Viji Thaliath
x22185178

1 Introduction

This configuration manual presents the information about the dataset, the hardware and software specifications, tools, libraries, and code needed to run the models developed in the research project "A Comparative Analysis for Recognizing Emotions from Facial Expressions." The research work can be replicated by following this guide.

2 Specifications

This section mentions the hardware and software specifications of the system that was used for implementing the research.

2.1 Hardware Specification

The research is implemented on the system with the following hardware configuration as shown in the Table 1.

Name	Description
Machine	Acer Nitro AN515-55
Operating System	Windows 11(22H2)
Processor	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz - 4.50 GHz
Installed RAM	16.0 GB DDR4 (15.8 GB usable)
System type	64-bit operating system, x64-based processor
GPU	Nvidia GTX 1650 Ti (VRAM: 4 GB)

Table 1: Hardware Specifications

2.2 Software Specification

The entire project is implemented using Python programming language(v3.8.16). The IDE which I found better to used was Visual Studio Code(v1.81.0+). The following packages and libraries were used to perform the research: Numpy, Pandas, Matplotlib, Seaborn, os, Scikitlearn, Tensorflow-gpu, and Keras.

3 Dataset Description

The dataset used in the project was downloaded from the Kaggle website [1]. The dataset was downloaded as a zip file. The dataset contains two folder the train dataset and the test dataset each having images of seven human emotions, mainly: anger, disgust, fear, happiness, neutral, sadness and surprise. The total number of images available in the train dataset are 32250. Since the train dataset was highly unbalanced, a subset of the training data was created with 500 images present in each of the emotion classes and was named as train_balanced. Then all the subset data were stored under the folder path: "Z:\Projects\Thesis\Work\Dataset". Under this are the three folders, as shown in Figure 1.

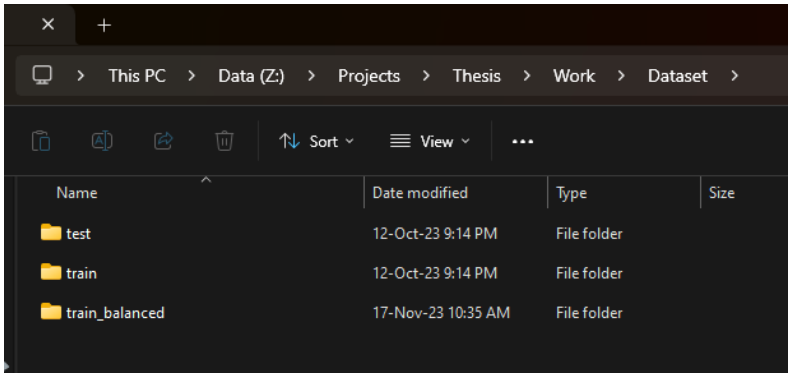


Figure 1: The Dataset Folder

While replicating this research, do update the path directories to your dataset path directory as stated in Fig.2

```
image_dir = Path("Z:\\Projects\\Thesis\\Work\\Dataset\\train")
test_dir = Path("Z:\\Projects\\Thesis\\Work\\Dataset\\test")
```

Figure 2: File path declaration

4 Data Preparation

The steps in importing the data are shown as below in Figure 3. The following steps are common for all the three models that we will discuss in the implementation section.

[1] https://www.kaggle.com/datasets/manishshah120/facial-expression-recog-image-ver-of-fercdataset

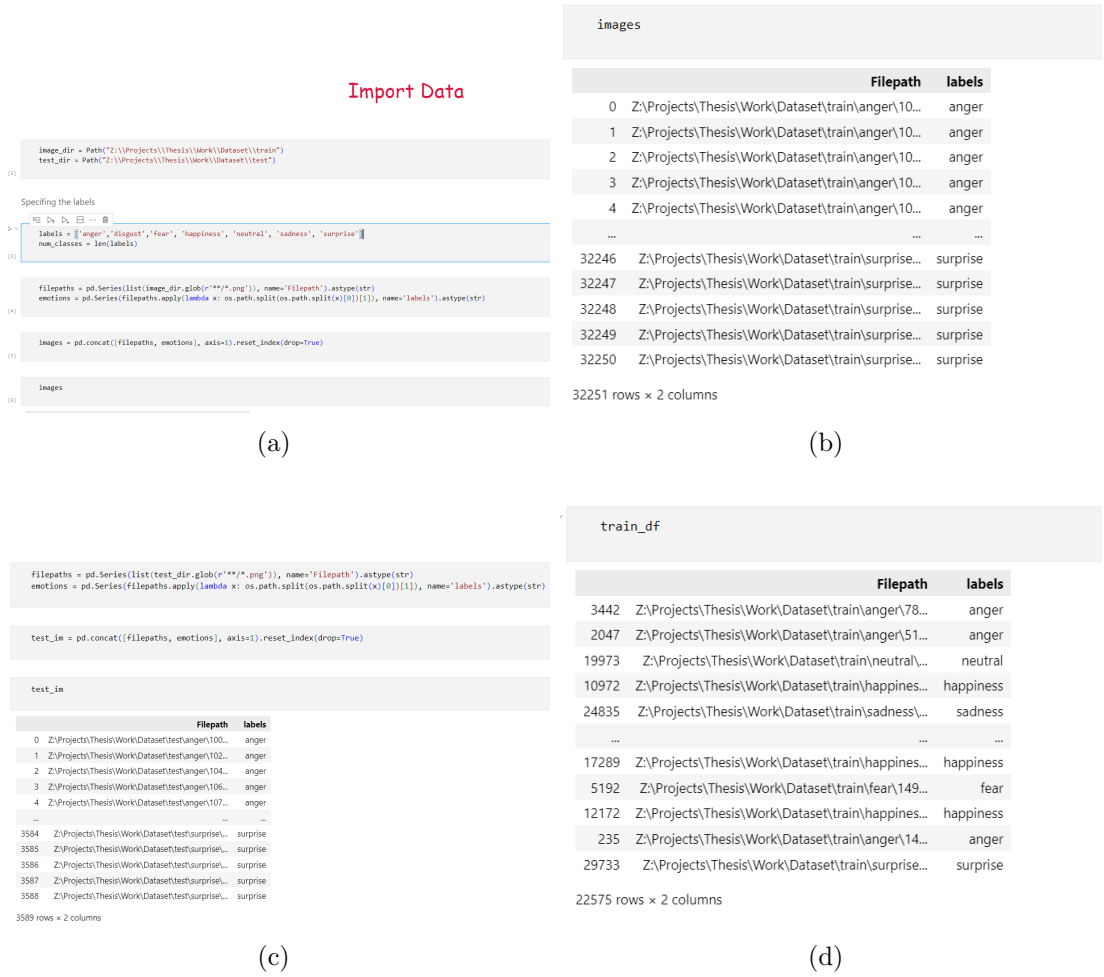


Figure 3: Importing the Dataset

After the dataset is read then the next stage is to load that data and preprocessing them. We use the image.ImageDataGenerator() from tf.keras.preprocessing to re-scale the data as shown in Figure 4. This helps in augmenting and preprocessing the image data.

```
train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

val_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
)

test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
)
```

Figure 4: Data Re-Scaling

The next step is loading the data which is handled by flow_from_dataframe() function as shown in the Figure 5. This loads the data and validates the images and their file names.

```
train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='labels',
    target_size=(48, 48),
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='training'
)

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='labels',
    target_size=(48, 48),
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='validation'
)

test_images = test_generator.flow_from_dataframe(
    dataframe=test_im,
    x_col='Filepath',
    y_col='labels',
    target_size=(48, 48),
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=32,
    shuffle=False,
)
```

```
Found 18060 validated image filenames belonging to 7 classes.
Found 4515 validated image filenames belonging to 7 classes.
Found 3589 validated image filenames belonging to 7 classes.
```

Figure 5: Data loading using flow_from_dataframe()

5 Implementation

The research is conducted using three machine learning algorithms, and the algorithm was applied to both the balanced and unbalanced image datasets.

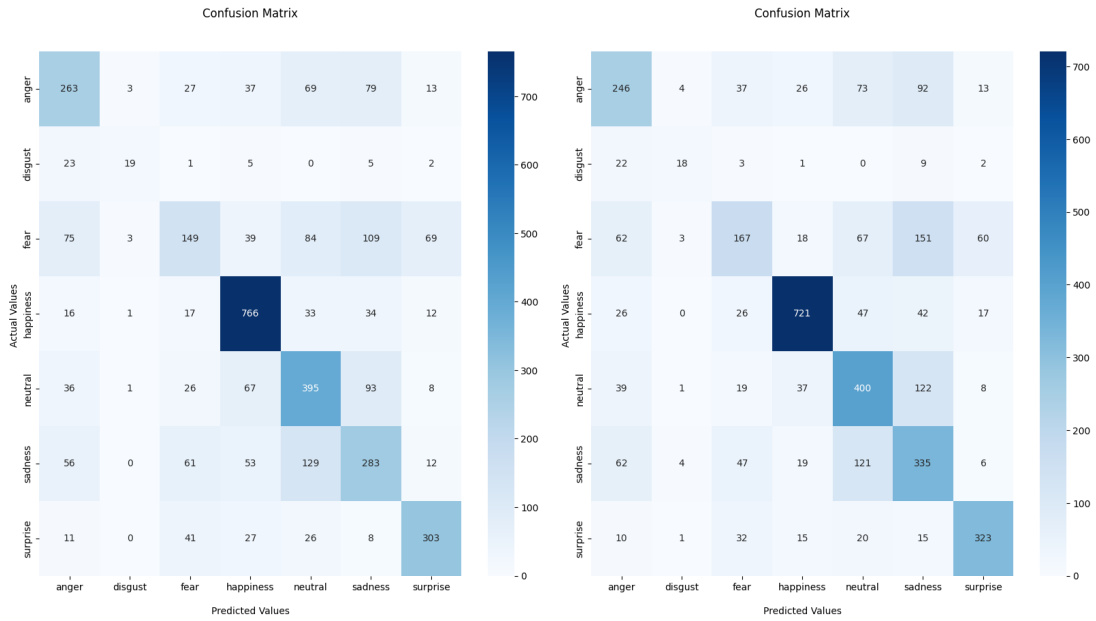
5.1 CNN Model

The model was run on both balanced and unbalanced datasets for two sets of epochs 50 and 100. The diagram given below depicts the implementation of the CNN model(fig:6).

```python
model = tf.keras.Sequential([
    layers.InputLayer(input_shape=[48,48,1]),

    #Block One
    layers.Conv2D(filters=32, kernel_size=3, strides=1, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPool2D(),
    layers.Dropout(0.2),

    #Block Two
    layers.Conv2D(filters=64, kernel_size=3, strides=1, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPool2D(),
    layers.Dropout(0.2),

    #Block Three
    layers.Conv2D(filters=128, kernel_size=3, strides=1, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPool2D(),
    layers.Dropout(0.2),

    #Block Four
    layers.Conv2D(filters=64, kernel_size=3, strides=1, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPool2D(),
    layers.Dropout(0.2),


    #Header
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
], name='model')
```

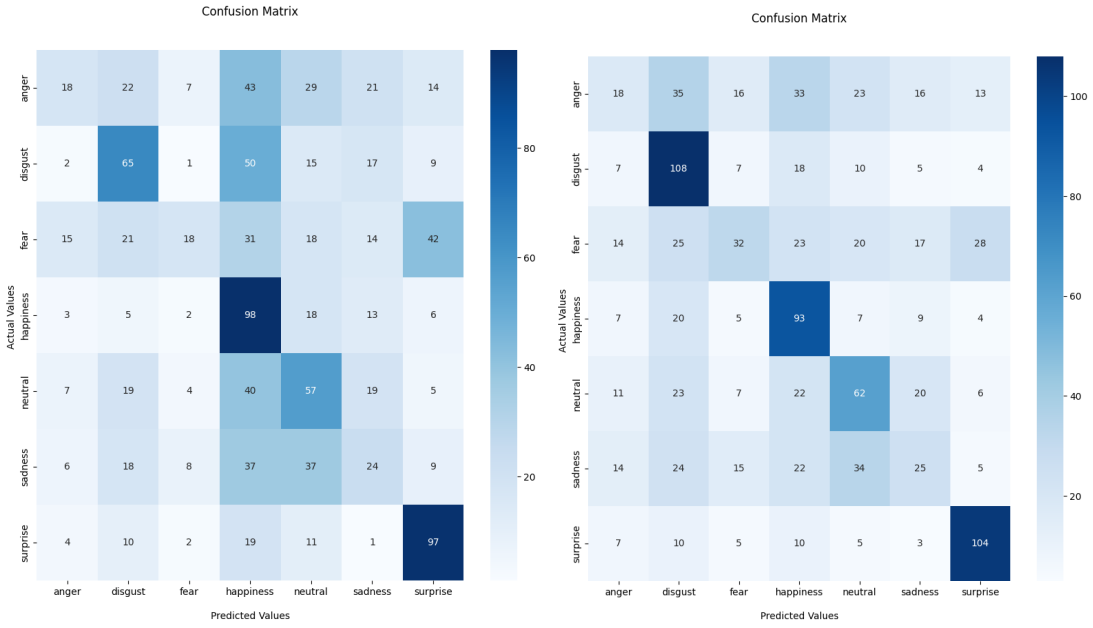
Figure 6: CNN Model Implementation

The confusion matrix for this model are shown below.



(a) (b)

Figure 7: Confusion Matrix when classes are unbalanced (a) 50 epochs (b) 100 epochs



(a) (b)

Figure 8: Confusion Matrix when classes are balanced (a) 50 epochs (b) 100 epochs

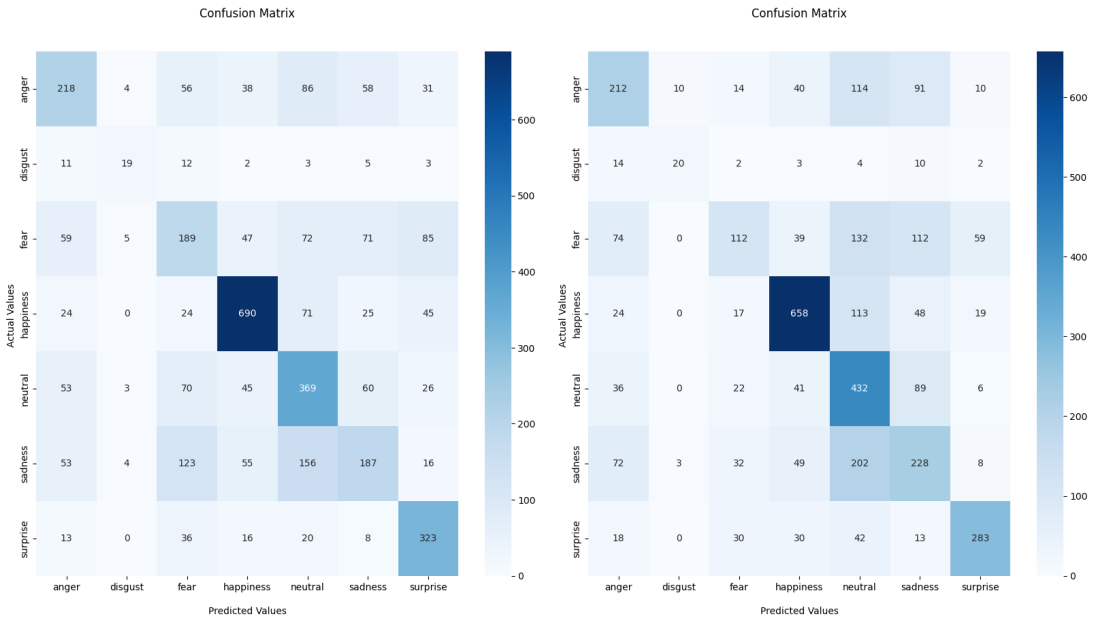
5.2 CNN-LSTM Model

The model was run on both balanced and unbalanced datasets for two sets of epochs 50 and 100. The diagram given below depicts the implementation of the hybrid CNN-LSTM model(fig:9).

```
model = tf.keras.Sequential([
    layers.InputLayer(input_shape=[48,48,1]),

    layers.  (function) BatchNormalization: Any  s=(1,1), padding='valid'),
    layers.BatchNormalization(axis=3),
    layers.Activation('relu'),

    layers.Conv2D(64, (3,3), strides=(1,1), padding = 'same'),
    layers.BatchNormalization(axis=3),
    layers.Activation('relu'),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), strides=(1,1), padding = 'valid'),
    layers.BatchNormalization(axis=3),
    layers.Activation('relu'),

    layers.Conv2D(128, (3,3), strides=(1,1), padding = 'same'),
    layers.BatchNormalization(axis=3),
    layers.Activation('relu'),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(128, (3,3), strides=(1,1), padding = 'valid'),
    layers.BatchNormalization(axis=3),
    layers.Activation('relu'),
    layers.MaxPooling2D((2,2)),
    layers.Reshape((-1,128)),
    (layers.LSTM(128)),
    (layers.Reshape((-1,64))),
    layers.LSTM(64),
    layers.Dense(200, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(7, activation = 'softmax')
])
```

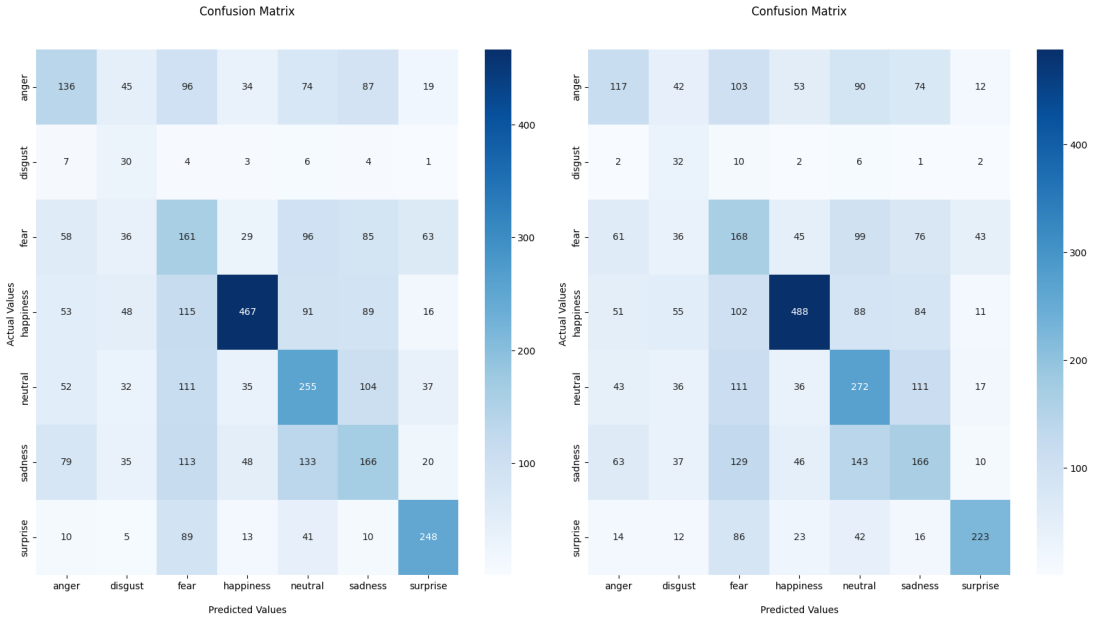
Figure 9: Hybrid CNN-LSTM Model Implementation

The confusion matrix for this model are shown below.



(a) (b)

Figure 10: Confusion Matrix when classes are unbalanced (a) 50 epochs (b) 100 epochs



(a) (b)

Figure 11: Confusion Matrix when classes are balanced (a) 50 epochs (b) 100 epochs

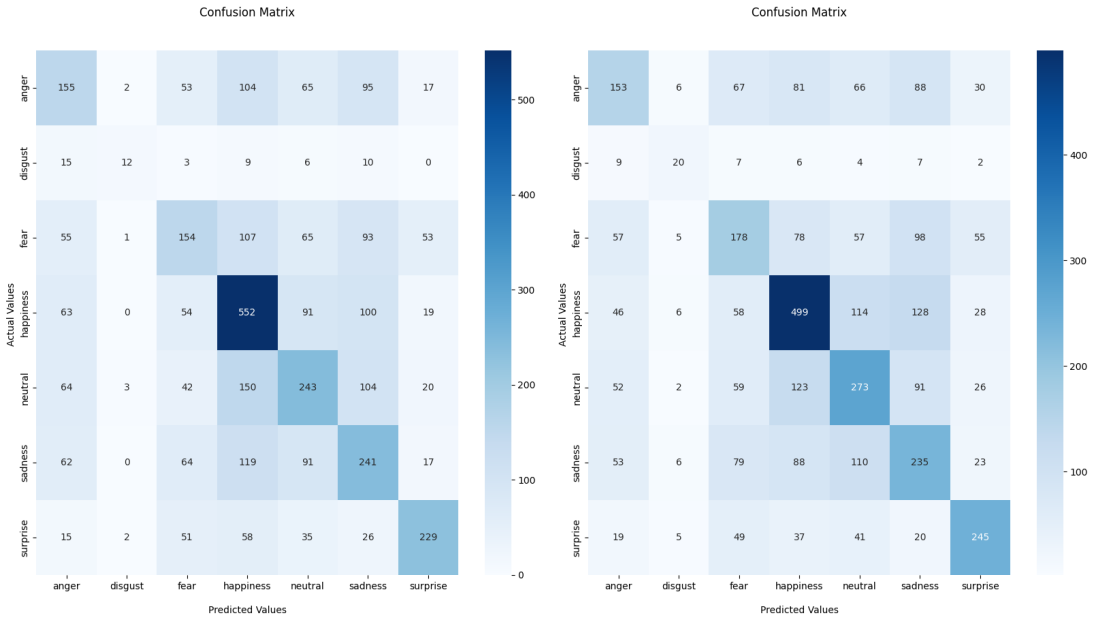
5.3 VGG-16 Model

The model was run on both balanced and unbalanced datasets for two sets of epochs 50 and 100. The diagram given below depicts the implementation of the VGG-16 model(fig:12).

```python
# Load the pre-trained VGG network
base_model = VGG16(weights='imagenet', include_top=False,input_shape=(48, 48, 3))

# Freeze all the layers of the VGG network
for layer in base_model.layers:
    layer.trainable = False

'''#Remove the first layer and replace it with a new single-channel convolutional layer
del base_model.layers[0]
base_model.input = layers.Input(shape=(None, None, 1))
base_model.layers[0] = layers.Conv2D(32, (3, 3), padding='same')(base_model.input)'''

# Add a new dense layer on top of the frozen VGG network
x = base_model.output
x = layers.Flatten()(x)
x = layers.Dense(256, activation='relu')(x)
prd = layers.Dense(7, activation='softmax')(x)

# Create a new model that includes the VGG network and the new dense layer
model = Model(inputs=base_model.input, outputs=prd)
```

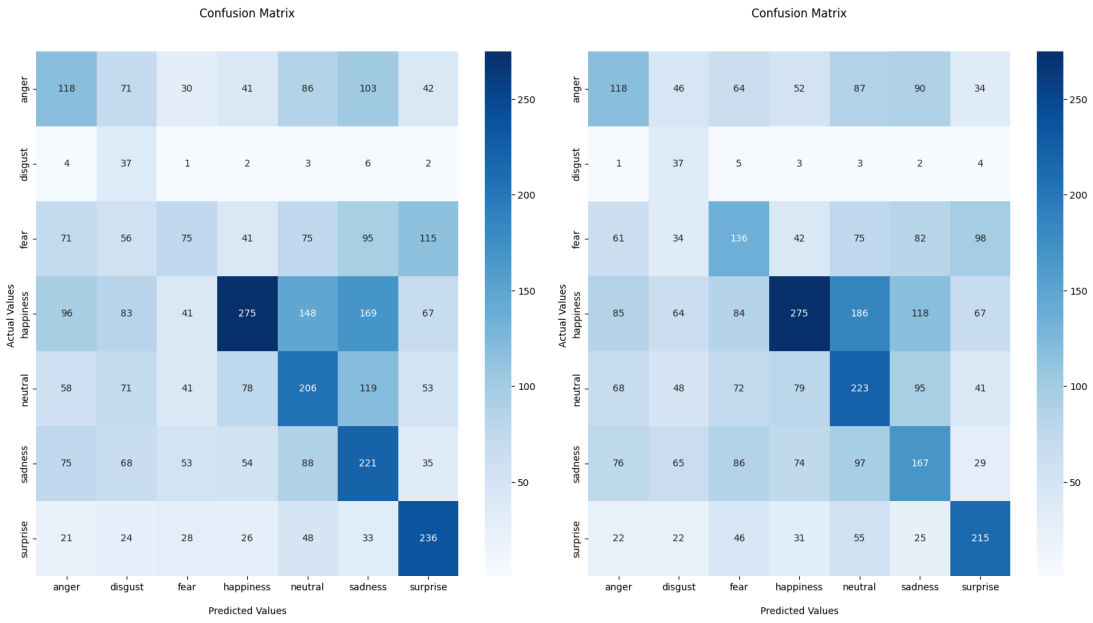
Figure 12: VGG-16 Model Implementation

The confusion matrix for this model are shown below.



(a) (b)

Figure 13: Confusion Matrix when classes are unbalanced (a) 50 epochs (b) 100 epochs



(a) (b)

Figure 14: Confusion Matrix when classes are balanced (a) 50 epochs (b) 100 epochs

6 Conclusion

In conclusion, the data in the configuration manual shows us how the research was applied and implemented. The report is divided into five sections, each of which is thoroughly and methodically discusses on how to replicate the same project if needed.