Facial Emotion Detection

Group Members:
Cindy Liaunardi – 2502013864
Matthew Dipaputra Widjaja – 2502030006
Andrew Nugrah – 2502031785
Harura Cendekia - 2502049920

1. Introduction

Most humans express their emotions through facial expressions. A certain fact is that human facial expression is a universal form of communication. Regardless of cultural background, people around the world tend to display basic emotions through facial expressions. Since this behavior is essential to oneself, it suggests that certain facial expressions are hardwired into our biology. Our objective through artificial intelligence is to create a system capable enough to make decisions and think like a general human. We've set our goal to create an efficient and accurate model that can operate in real-time scenarios, making it suitable for applications like human-computer interaction, sentiment analysis, or emotion-aware systems.

2. Experiment

The dataset used in this experiment was downloaded from the Kaggle website. This dataset, which has been split into training and test datasets, consists of 35,685 sample 48x48 pixel grayscale photos of faces. The photos are divided into seven classes according to the expression on the face (anger, disgust, fear, happy, neutral, sad, and surprised).

Enhancing the diversity and resilience of a training dataset for machine learning models requires the crucial preprocessing step of data augmentation. Several methods are used to enhance the dataset. By randomly shifting images up to 10% of their width, width shift adds horizontal differences and promotes diversity in training samples. Similar to this, height shift creates vertical diversity by adjusting images up to 10% of their height at random. A horizontal flip diversifies the direction of facial features by introducing mirror images with a 50% probability. Pixel values are normalized within the range [0, 1] by dividing them by 255 using a technique called rescaling.

Twenty percent of the test set is dedicated to validating the model using unseen data, a process known as the validation split. By scaling every image to 48 by 48, the target size guarantees consistency in input dimensions. The quantity of photos processed in each training phase is controlled by batch sizing, which affects weight updates. To improve computational performance, grayscale conversion condenses images into a single channel. Finally, for efficient classification, class mode aligns the one-hot encoded class labels with the output layer of the neural network.

2.1. Training & Testing Model

In this segment we’re going to pre-process the dataset, modelling, training, and testing the dataset.

2.1.1 Pre-processing

We used these pre-processing methods for the whole dataset

```
train_datagen = ImageDataGenerator(#rotation_range = 180,
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1,
                                   horizontal_flip = True,
                                   rescale = 1./255,
                                   #zoom_range = 0.2,
                                   validation_split = 0.2
                                  )

test_datagen = ImageDataGenerator(rescale = 1./255,
                                  validation_split = 0.2
                                 )

validation_datagen = ImageDataGenerator(rescale = 1./255,
                                        validation_split = 0.2)

train_generator = train_datagen.flow_from_directory(directory = train_dir,
                                                    target_size = (img_size,img_size),
                                                    batch_size = 64,
                                                    color_mode = "grayscale",
                                                    class_mode = "categorical",
                                                    subset = "training"
                                                   )

test_generator = test_datagen.flow_from_directory(directory = test_dir,
                                                  target_size = (img_size,img_size),
                                                  color_mode = "grayscale",
                                                  class_mode = "categorical"
                                                  )

validation_generator = validation_datagen.flow_from_directory(directory = test_dir,
                                                              target_size = (img_size,img_size),
                                                              batch_size = 64,
                                                              color_mode = "grayscale",
                                                              class_mode = "categorical",
                                                              subset = "validation"
                                                             )
```
snappify.com

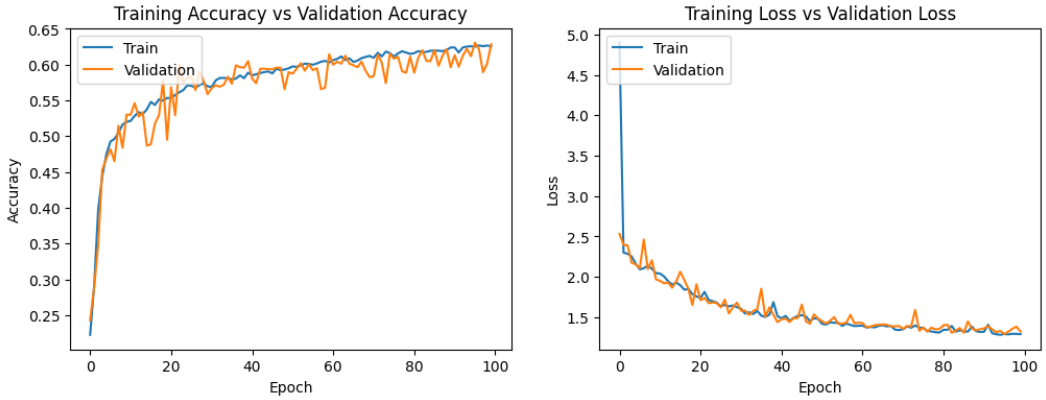
2.1.2 Modelling

This will be our model that we used for the whole project; the details are displayed below.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 conv2d_1 (Conv2D)           (None, 48, 48, 64)        18496

 batch_normalization (Batch  (None, 48, 48, 64)        256
 Normalization)

 max_pooling2d (MaxPooling2  (None, 24, 24, 64)        0
 D)

 dropout (Dropout)           (None, 24, 24, 64)        0

 conv2d_2 (Conv2D)           (None, 24, 24, 128)       204928

 batch_normalization_1 (Bat  (None, 24, 24, 128)       512
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 12, 12, 128)       0
 g2D)

 dropout_1 (Dropout)         (None, 12, 12, 128)       0

 conv2d_3 (Conv2D)           (None, 12, 12, 512)       590336

 batch_normalization_2 (Bat  (None, 12, 12, 512)       2048
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 6, 6, 512)         0
 g2D)

 dropout_2 (Dropout)         (None, 6, 6, 512)         0

 conv2d_4 (Conv2D)           (None, 6, 6, 512)         2359808

 batch_normalization_3 (Bat  (None, 6, 6, 512)         2048
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 3, 3, 512)         0
 g2D)

 dropout_3 (Dropout)         (None, 3, 3, 512)         0

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 256)               1179904

 batch_normalization_4 (Bat  (None, 256)               1024
 chNormalization)

 dropout_4 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 512)               131584

 batch_normalization_5 (Bat  (None, 512)               2048
 chNormalization)

 dropout_5 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 7)                 3591

=================================================================
Total params: 4496903 (17.15 MB)
Trainable params: 4492935 (17.14 MB)
Non-trainable params: 3968 (15.50 KB)
_________________________________________________________________
```
snappify.com

2.1.3. Training

After creating the model, we trained it using the training data generated by train_datagen with 100 epochs, and these are the results.



2.1.4. Testing

We picked a random image from the test folder to show prediction. The result is attached on the image below.

2.1.5. Train vs Test Accuracy

From the model we built, we got a score of 65.83% for train accuracy and 62.85% for the test accuracy.

2.1.6. Saving the Model

We used the built in function to save model to 'model.h5' which then will be used in creating the real-time emotion recognition program.

2.2. Creating The Real-Time Program

2.2.1. Loading Model

Our previously saved model is loaded with tensorflow.keras.models.load_model.

2.2.2. Detector

We used Multi-Task Cascaded Convolutional Networks (MTCNN) algorithms to detect and localize faces in digital images or videos. We previously used haarcascade frontal face detection but we didn't get the result that we wanted.

2.2.3. Capturing Facial Expression

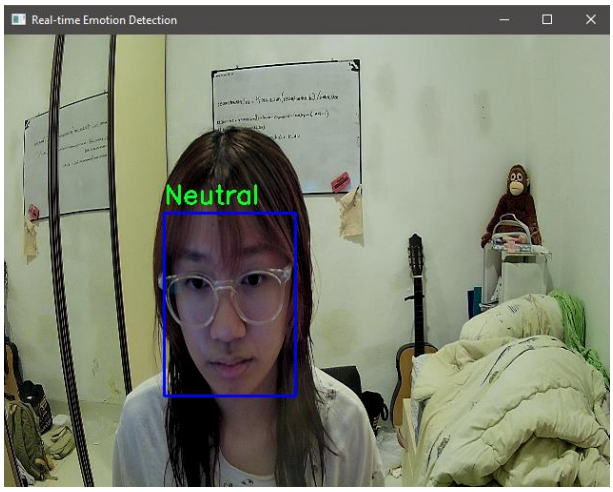
The OpenCV library is used to detect and capture facial expression through the built in webcam. The detected face will be resized to (48, 48) to match the model input size. After getting the image, the program will predict the facial expression and print the label on the frame. The program then will break if the 'q' key is pressed, here is the code for both the detector and the results.

=

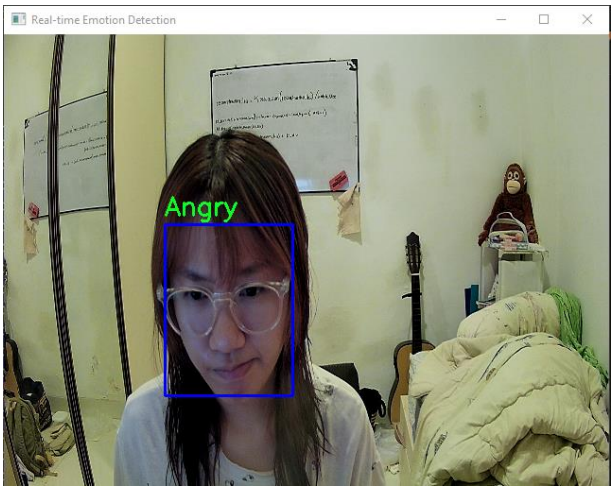
```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Detect faces in the frame
    faces = detector.detect_faces(frame)

    for face in faces:
        x, y, w, h = face['box']

        # Extract the face ROI (Region of Interest)
        face_roi = frame[y:y+h, x:x+w]

        # Resize the face to match the model input size
        face_roi = cv2.resize(face_roi, (48, 48))
        face_roi = cv2.cvtColor(face_roi, cv2.COLOR_BGR2GRAY)
        face_roi = np.expand_dims(face_roi, axis=0)
        face_roi = face_roi.reshape(1, 48, 48, 1)

        # Predict the emotion
        result = model.predict(face_roi)
        result = list(result[0])
        idx = result.index(max(result))

        # Get emotion label
        predicted_label = label_dict[idx]

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.putText(frame, predicted_label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2)

    # Display the resulting frame
    cv2.imshow('Real-time Emotion Detection', frame)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

2.2.4. Real-Time Face Detection

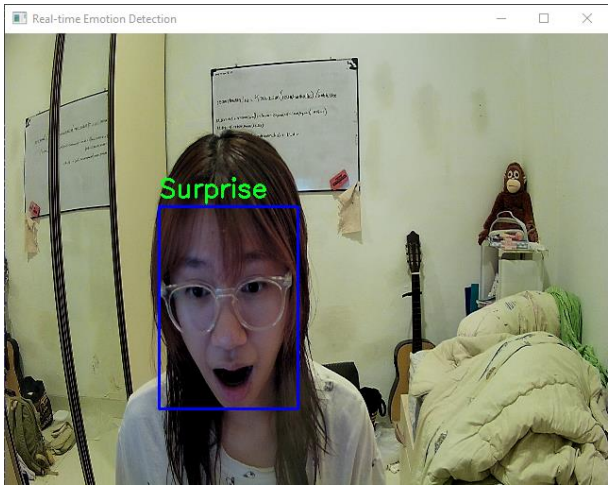
The program will then detect emotion with the built in webcam as shown below.



Neutral Emotion



Anger Emotion



Surprised Emotion

3. Conclusion

In conclusion, our model, achieving a train accuracy of 65.83% and a test accuracy of 62.85%, demonstrates robust generalization for real-time applications in human-computer interaction and sentiment analysis. The adoption of MTCNN algorithms significantly improves face detection accuracy over traditional methods. Utilizing OpenCV, our system captures and predicts facial expressions seamlessly. The implementation ensures practical usability, with the program ending upon the user pressing 'q'. Overall, our journey in AI yields a sophisticated, real-time emotion-aware system, meeting and surpassing our objectives. There could be some more work for tweaking in some places, but we reached our objective and achieved reasonable scores. We hope this project can be of use and benefits others. Our project can be accessed by this link or by accessing the last link on our reference list.

4. Reference

https://girolamopinto1.medium.com/emotion-detection-in-python-8642fade50a9
https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer
https://www.kaggle.com/code/odins0n/emotion-detection
https://www.kaggle.com/code/aayushmishra1512/emotion-detector
https://github.com/b3ndur/deep-learning-project