BARQ SYSTEMS

Real-Time Facial Emotion Recognition

FEBRUARY 15, 2022

BARQ SYSTEMS

ABSTRACT

Humans share a universal and fundamental set of emotions that are exhibited through consistent facial expressions. An algorithm that performs detection, extraction, and evaluation of these facial expressions will allow for automatic recognition of human emotion in images. In this report, two approaches are implemented to solve this task. The first approach is the Deep learning approach, where we designed Convolutional neural networks architecture from scratch and modified some pre-trained models as VGG19, RESNET.

The second approach is the Machine Learning approach**.** We used in it a hybrid feature extraction and facial expression recognition method that utilizes OpenCV object detectors with cascade classifier to extract faces and DLIP to drew facial landmarks to extract facial features from images and uses histogram-of-oriented-gradients (HOG), Gabor filter, Contrast limited adaptive histogram equalization (CLAHE)and sliding hog windows, and SVM Classifier and XGBOOST classifier to train a multi-class predictor for classifying the five fundamental human facial expressions. Reasonable accuracy is achieved with the predictor, dependent on the testing set and test emotions. Accuracy is 86% with XGBOOST and 93% with SVM.

Another method to detect facial expression recognition is made by utilizing the OpenCV object detectors with HAAR cascade classifier to extract faces from images and pretrained model VGG-19 classifier to train a multi-class predictor for classifying the five fundamental human facial expressions. Reasonable accuracy is achieved with the predictor, dependent on the testing set and test emotions. Accuracy is 96%.

Acknowledgment

We would like to thank our mentor Engineer Kareem Hamdi for his support, guidance, and help throughout the project. We would also like to thank the marketing teams for their great design ideas. Finally, we thank our parents, for without them we wouldn't have made it this far.

Table of Contents

ABBREVIATION

CNN : Convolution Neural Networks

HOG : Histogram of gradients

Adam: adaptive moment estimation

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Most of the human feelings are expressed through the face and by seeing one's face, one can easily identify whether he is happy or sad or angry. So, for truly knowing the feeling behind words, the facial expression must be correctly recognized. Facial expression extracts the true or inner emotion which the speaker tries to hide and find out whether he is happy/sad/angry/neutral/surprised. Its applications include human behavior detection, human-computer interaction, security, lie detection, pain detection, music system based on mood, automated tutoring, expressing through emoticons, etc.

Team members

- Ibrahim Mostafa Mohamed
- Ahmed Hassan Ezz Eldin
- Hoda Ahmed El Emam
- Salah Tarek Khattab
- Shrouk Hassan
- Naema
- Hussien Reda Mohamed
- Asya Ahmed
- Nourhan Mohamed Sharaf

Team Mentor

- Karim Hamdy

Sponsors BARQ SYSTEM

- Engineer Abdelrahman Essam Eldin

CHAPTER 2: BACKGROUND

In recent years, a great deal of study has been done in the field of human facial expression identification, owing to the diverse applications that exist.

They used a Conventional Neural Network (CNN) to accomplish FER in [1]. The CNN receives a 64x64 picture as input. One input layer, five convolution layers, three pooled layers, one fully connected layer, and one output layer are all included in the network. Following the convolutional pooling procedure, the fully connected layer is combined as the SoftMax layer's input to generate the output class. They validated with 91.8 percent accuracy using the JAFFE and CK+ databases, with 85.4 percent accuracy in cheerful, 85.4 percent in furious, 81.9 percent in disgust, 83.5 percent in sad, 86.1 percent in surprise, and 93.5 percent in surprised

They state in [2] that any FER technique has three steps: feature extraction, dimensionality reduction, and classification. Dimensionality and feature selection, they claim, are important concerns with FER. They also stated that processing the image necessitates a significant amount of memory and processing. As a result, they presented geometric features as an alternative, and they used facial landmark detection for feature extraction and the CNN Classifier. To assess the effectiveness of their approach, they employed the JAFEE, CK+, and KDEF databases. SVM gives an accuracy of 74.4 percent in JAF, 69.2 percent in KDEF, and 87.7 percent in CK+.

[3] presented a FER system based on 63 facial landmark points from the active appearance model. They calculated the remaining four landmark spots using those 63 points. The degree of openness was measured using the height-to-width ratio. By multiplying the individual weights by their sum of ratios, we were able to derive facial expressions. On the same database, they achieved a prediction accuracy of 83.6 percent and an average recognition rate of 83.6 percent.

[4] A person's facial expression might be classified as anger, contempt, disgust, fear, happiness, sadness, or surprise. The image is first subjected to CLAHE, and the faces are then recognized using a histogram of oriented gradients. The face landmarks are then predicted using a model trained on the iBUG 300-W dataset. A feature vector is calculated using the suggested approach and normalized landmarks. A Support Vector Classifier can be used to recognize the emotions using this derived feature vector. Using the well-known CK+, the Support Vector Classifier was trained and validated for system accuracy. 80% percentage.

CHAPTER 3: REQUIREMENTS

3.1 Block Diagram

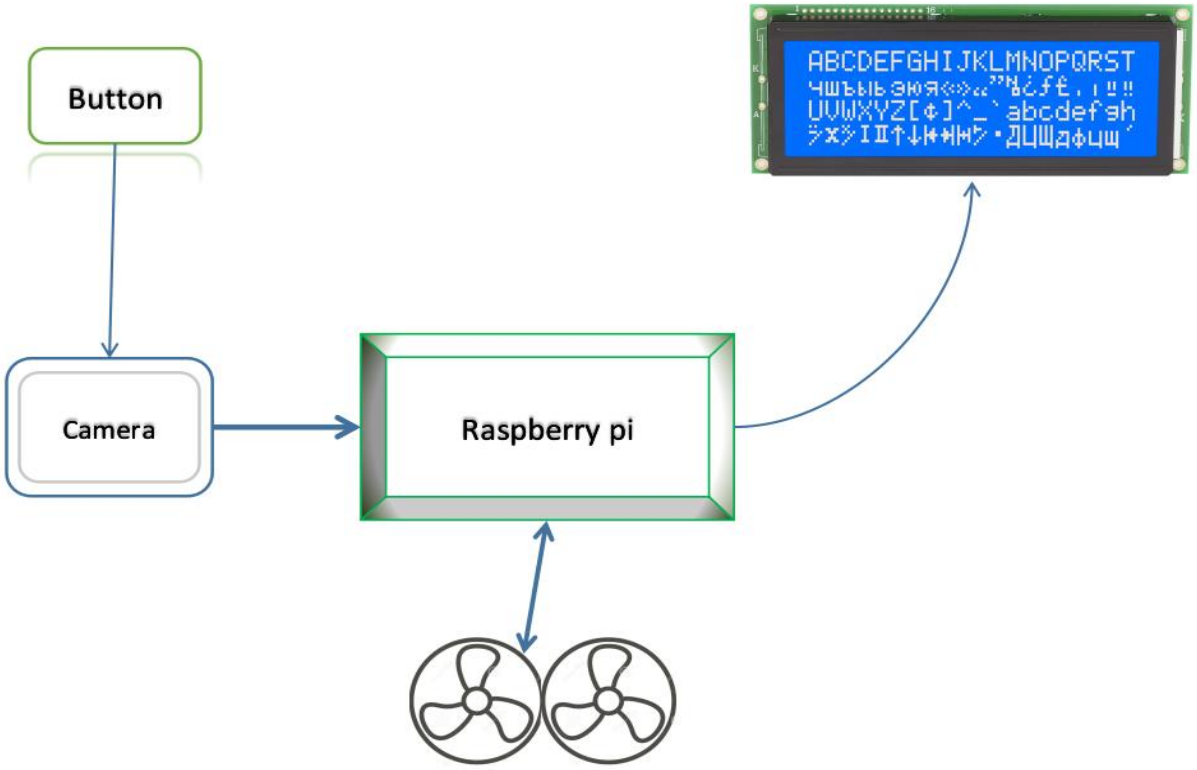

Figure 1 Block diagram

3.2 The main Components

- Raspberry Pi 4 Model B (8GB RAM)
- Raspberry Pi Camera Module V2
- Push Button
- LCD 20x4
- Raspberry Pi Dual Fan with Heatsink Cooling System
- Micro SD 32 GB
- Ethernet Cable (RJ45)
- Power Supply 5V, 3A + on/off switch

3.3 Raspberry Pi

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.

Family	Model	Form Factor	Ethernet	Wireless	GPIO	Released
Raspberry pi 4	B (8 GiB)	Standard	Yes	Yes	40	2020

We use Raspberry Pi 4 B 8 GiB RAM because of the best processing.

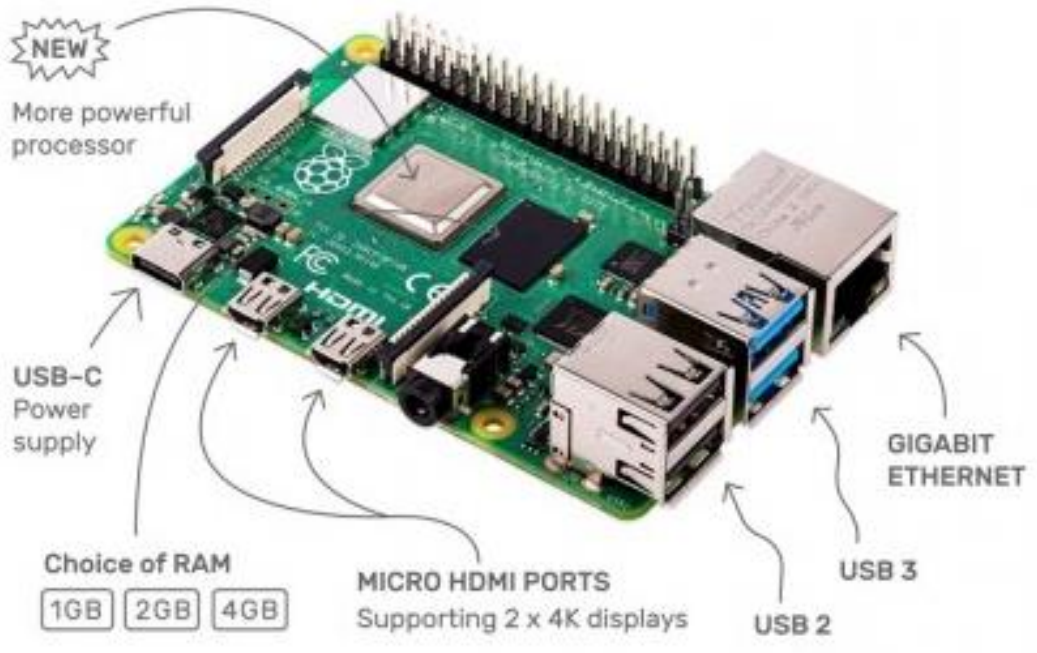

Figure 2 Raspberry Pi 4

Specification

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)

- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

Getting Started

- A micro-SD card for Raspbian OS.
- A high-quality 3A micro-USB power supply, such as the official Raspberry Pi Universal Power Supply

3.4. Raspberry Pi Camera

We use Raspberry Pi Camera Module V2 Official 8 Megapixel HD Camera Board with IMX219 PQ CMOS Image Sensor.

Figure 3 Camera Module

Specification

- 8-megapixel camera capable of taking photographs of 3280×2464 pixels.
- Capture video at 1080p30, 720p60 and 640x480p90 resolutions.
- All software is supported within the latest version of the Raspberry Operating System.
- Applications: CCTV security camera, motion detection, time-lapse photography.
- FFCcable length: 15cm.

Connection with Raspberry Pi

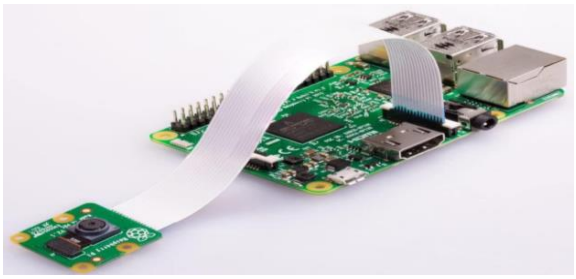

Figure 4 Camera Connection

3.5 Other Components

Power Supply with on/off switch used for Raspberry Pi

For Raspberry Pi 4 Model B we need the power supply current capacity to be 3A and to make it easier to work with the Raspberry Pi.

Figure 5 Raspberry Pi Power Supply

Ethernet Cable

This cable connects wired devices together to the local network for file sharing and Internet access.

So, we use it to connect Raspberry Pi with a laptop

Figure 6 Ethernet Cable

Dual Fan

Raspberry Pi Dual Fan with Heatsink Cooling System Double Cooling Fan Cooler Radiator suitable for Raspberry Pi 3 3B Plus & 4.

By using this kit, it can keep your Raspberry Pi cool.

It needs 5V so we give it this Volt from raspberry pi.

Figure 7 Dual Fan

CHAPTER 4: DESIGN

4.1. Software Module

4.1.1 Dataset Collection

One of the most challenging tasks in computer vision is to find a dataset with good quality.

The most popular datasets can be found in Table 1.

Table 1 Popular Datasets

Name	Number of Train images	Number of Test images	Number of classes	Best Accuracy
Fer-2013	28,709	3,589	7	75%
AffectNet	37,000	4,000	8	60%
CK+	920	-	8	72% - 75%

By looking at the accuracies and trying to train our models on these datasets, we concluded that the low accuracies are due to the quality of the photos in each dataset.

To overcome this problem, we decided to collect images ourselves by using a script that cuts a video into frames.

Also, we added some good quality samples from " AffectNet " to increase the variety of the images and generalize the dataset.

The resultant dataset is

Figure 8 Final dataset

Preprocessing:

Converting all images to grayscale

Resizing images to 112*92

Augmentation:

Applying flipping on all images

Training images:

23273 images (46546 after augmentation)

Angry: 4557 images (9114 after augmentation)

Happy: 4748 images (9496 after augmentation)

Neutral: 4819 images (9368 after augmentation)

Sad: 4106 images (8212 after augmentation)

Surprise: 5043 images (10086 after augmentation)

Testing images:

600 images taken from the Raspberry Pi camera

Using this dataset improved the accuracy of all used models, and this is explained in detail in the “Results” section.

4.1.2 Model Development

1. *Machine Learning*

   a. **First Approach**

First, we extracted the face from an image using OpenCV and used Facial landmarks using DLIB to localize and represent salient regions of the face, such as: eyes, eyebrows, nose, and mouth, and drew it on the face.

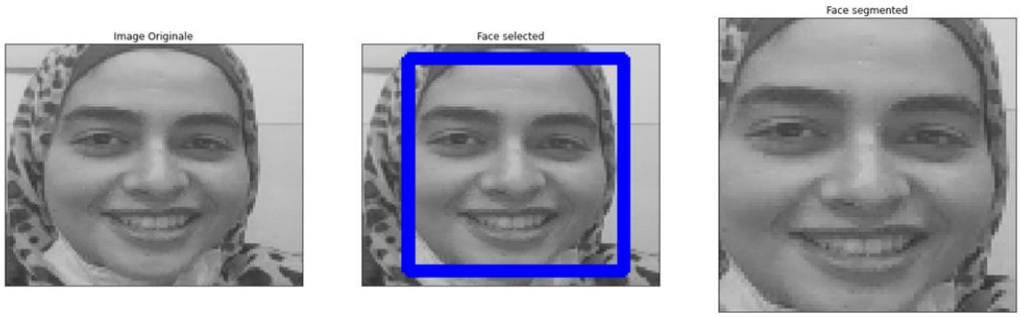

Figure 9 Feature Extraction

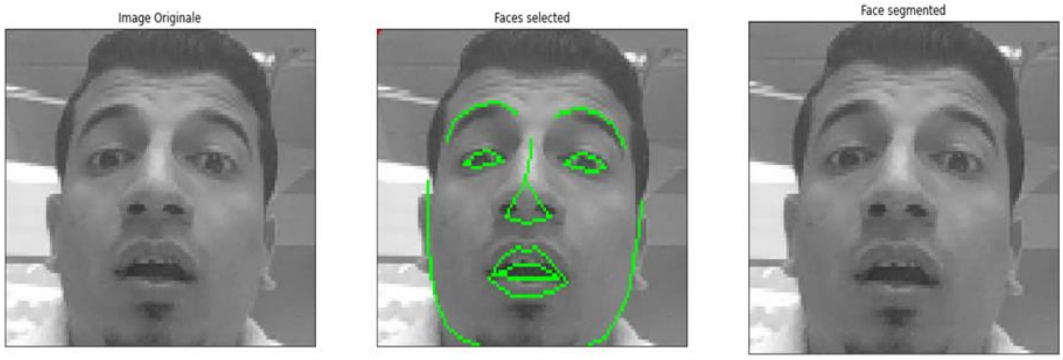

Figure 10 Facial Landmarks

We got landmarks points as a list then applied the HOG filter and sliding HOG windows to extract more facial features

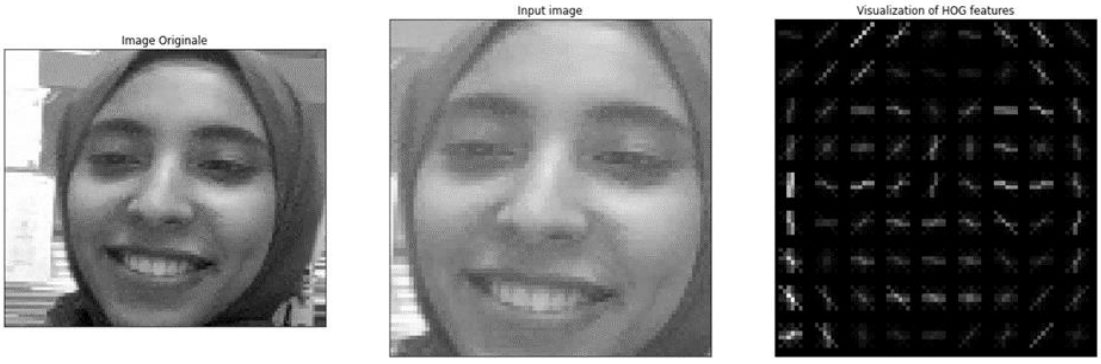

Figure 11 HOG features Visualization

Then we applied the Gabor filter also to get more important features from the face.

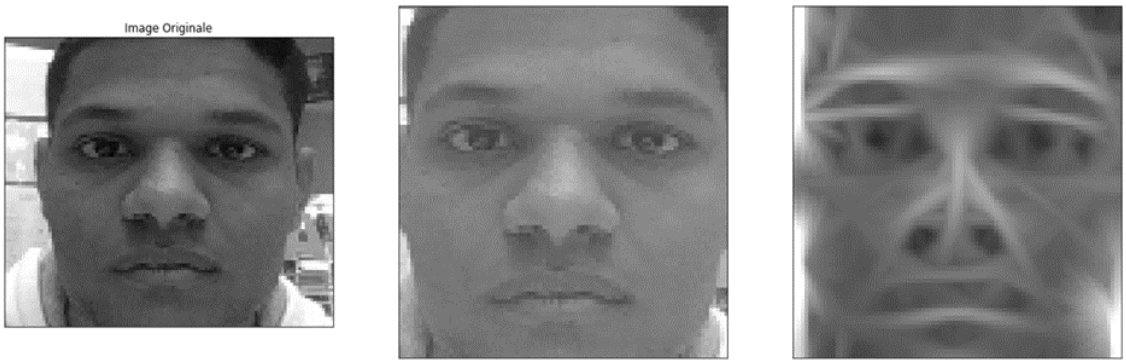

Figure 12 Gabor Features Visualization

After that, we concatenated feature landmarks and the three filters: HOG, Gabor, and Sliding HOG windows to extract the most important features and we got from it 4997 features and indexed these features with their labels and saved them as a NumPy array.

We took this NumPy array as an input to train it on a machine learning model. We tried two models: Logistic Regression and XGBOOST classifier.

Logistic Regression accuracy did not exceed 47%. XGBOOST accuracy gets up to 86% on tests.

Limitations: We didn't use this approach because we can't install DLIP Package on raspberry pi's operating system which is 32-bit because it doesn't support TensorFlow Library, so it was supposed to use the 64-bit operating system on raspberry pi instead of 32-bit because it supports TensorFlow but it doesn't support raspberry pi camera and The camera is the most important component in our project, we installed 64-bit operating system and used another approach.

b. Second Approach

This section explains the proposed methodology in detail. According to our literature review, practically every system created for Facial Expression Recognition contains these three key steps:

• Pre-processing of images

• Extraction of Features

• Classification of Expressions

It's an innovative technique because of the manner the image is pre-processed, and the features are extracted. Human facial expressions are represented by a feature vector based on facial landmarks. In addition, the facial expressions are recognized using the Support Vector Classification, logistic regression and xgboost and stacking the entire system was built using the Python programming language, with OpenCV being utilized heavily to do image processing tasks., The most popular Python libraries are Scikit-learn and Dlib. The machine learning tasks were carried out using an open-source general purpose machine learning package. The Support Vector Machine (SVM) is the most common use of the Scikit-learn toolkit. approach.

Classifier, logistic regression, xgboost and stacking. Whereas, face detection and facial landmark points prediction is achieved using Dlib library.

1) Facial Expression Image Database

Because the Support Vector Classifier (SVC), logistic regression, and xgboost algorithms all require training with a well-known face expression database, the Affectnet dataset was used, along with manually sorted data from over 27,000 photos. There are five emotions in total: happiness, surprise, sadness, neutral, and rage. Each image is 112*92 pixels in size. We originally attempted to split training data into train-test splits, however, this did not yield the greatest results. Then, using a different test dataset, we collect photos from the Raspberry Pi (about 600 images).

2)]Image preprocessing

began resizing training in addition to resizing the Affectnet dataset image to 112x92 as gathered images and changing it to grayscale, we know the dataset contains photos taken under a variety of lighting situations. To ensure that all photos are adjusted to similar lighting conditions, OpenCV's built-in function Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied to all images in the dataset. When opposed to a conventional Histogram

Equalization, CLAHE has the advantage of not considering the image's overall contrast. However, because the accuracy of the test data set to split from Affectnet dropped by 3% after that, we decided to skip preprocessing.

3) Facial Landmark Prediction

To begin, the face detection algorithm must be applied to the image in order to forecast facial landmarks. This was accomplished using the built-in Dlib function, which returns an object detector capable of detecting faces in images. In fig (1)

After the face detection is complete, the built-in function of Dlib is utilized to predict the facial landmark points. The popular pretrained model, which can be downloaded from the dlib website, is used to predict 68 landmark points. Yellow dots indicate the expected facial landmark positions in Figure (2).

4) Feature Extraction

The most crucial stage in facial expression recognition is calculating the feature vector that characterizes a person's feeling. It's crucial to understand how the facial landmark points relate to one another. This is accomplished by finding the mean of both axes, which yields a central point. (X_mean, y_mean) near the nose region AS IT considered as the center of the face. The position of all points in relation to this center point can then be determined. Then a line is formed between the center point and each other facial landmark position. As a result, each resulting line has magnitude and direction (i.e., it's a vector) and serves as the feature vector for both training and classification. The magnitude is the Euclidean distance between the two points, while the direction represents the angle formed by the line with the horizontal reference axis. As a result, the feature vector may be summed up as follows: feature vector = <point1.x, point1.y, magnitude1, direction1, . . ., point68.x, point68.y, magnitude68, direction68 >

5) Training and Classification

Following the construction of a feature vector, it is critical to identify the facial expressions. The Support Vector Machine (SVM) was used as the classification method for evaluation. SVMs are supervised machine learning algorithms that can be used for classification and regression.

As a result, SVM classifiers are extremely accurate and work well in high-dimensional spaces. SVM classifiers, on the other hand, use a subset of training points and hence consume significantly less memory in the end. Despite our big data sets in practise, they have a short training time. By just employing an SVM, training and classification can be accomplished with high accuracy. As a result, xgboost or logistic regression tried but they didn’t give proper accuracy.

The SVC was created with the help of a class called "SVC" from the Scikit-learn library's "svm" module. The data items in each emotion of the Affectnet dataset in addition to data collected manually were randomly shuffled and split in the ratio of 80:20 for training data: testing data to give variety but it gives accuracy 61% so it is still not efficient so our second trial is to train data on affectnet dataset in addition to data collected manually then test on data collected with raspberry pi with a kind of feature reduction by dropping landmarks point and keep magnitude and direction the accuracy become 84% then our third trial is to randomly shuffled and split in the ratio of 80:20 for training data: testing data that comes from data collected by raspberry pi (600 images) and keep with a kind of feature reduction by dropping landmarks point and keep magnitude and direction the accuracy become 93%. During the training phase, each image in the training set of a specific emotion is examined.

Face detection is now applied to it and the facial landmarks are displayed. It is possible to predict. The feature vector is calculated using the landmark points obtained in the preceding sub-section. Finally, SVC is trained utilizing all the feature vectors computed across the entire dataset, as well as the associated class labels.

During the testing/classification phase, each image in the testing set is subjected to the identical processes as those used in the training phase to calculate the feature vector. One thing to keep in mind is that training and testing set data items are mutually exclusive, meaning that the testing set contains images that were not used in training. Finally, the determined feature vector for a test image is fed into a trained SVC that predicts the emotion exhibited by the test image.

2. *Deep Learning*

a. Training CNN from scratch

First Architecture

Inspired by *Jason Brownlee* [1], the first architecture contains 8 layers: 5 Convolution layers and one output layer, where multiple numbers of incremental filters (32,64,128,512,512) are used with a kernel size of 3*3. "BatchNormalization" and "Dropout" layers are used to avoid overfitting, while the "MaxPooling" layer is used to reduce the number of parameters in our model, thus increasing the training speed and decreasing the model's size. Flatten Layer reshapes the output from the Conv layer to a one-dimension vector.

Finally, two "Dense layers" with 256 and 512 neurons are used respectively followed by an output layer with 5 neurons for our 5 classes and a softmax activation function. Models are optimized with Adam and the learning rate is initialized at 0.0001, with the "categorical_crossentropy" loss function.

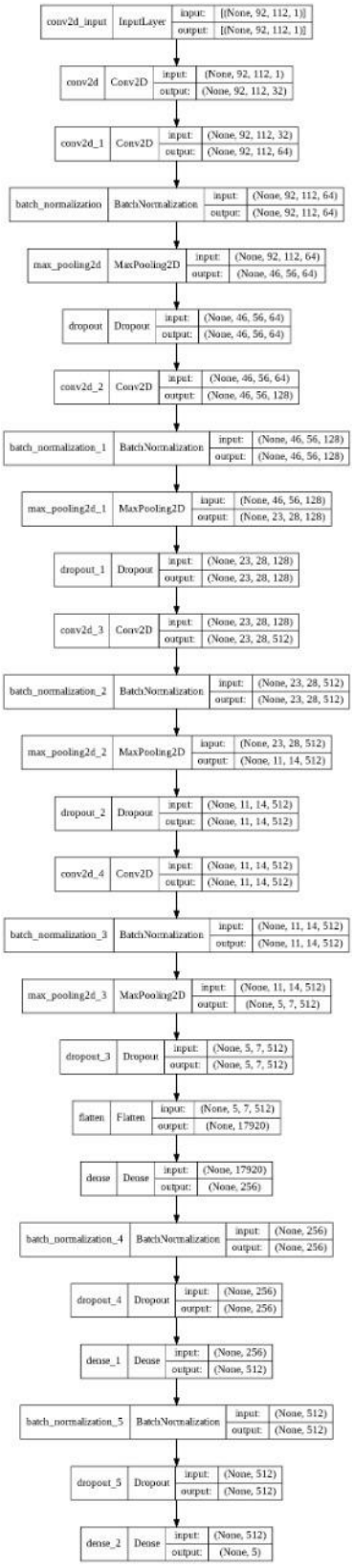

Figure 13 CNN Architecture 1

Second architecture

Modifying the first architecture by increasing the number of convolution layers to 17 layers with incremental number filters 64, 128, 256, and 512 filters.

b. Transfer Learning

Facial emotion recognition requires the machine learning models to handle multiple tasks, and since we were limited with time, internet speed, and GPU resources, multiple pre-trained models as VGG16, VGG19, EfficientNetB, MobileNet, InceptionV3, Xception, Resnet152, Resnet50, and inception_resentnet_v2 were tuned. According to the models' performance when training and validating on the merged dataset, and pre-testing on the CK+ dataset, the VGG19 has outperformed the other models in terms of f1 score.

VGG19

The VGG19[5] is a variant of VGG, that consists of 19 layers, 6 convolution layers, 3 fully connected layers, 5 MaxPool layers, and 1 SoftMax layer. VGG19 has trained on the ImageNet[6] dataset. Multiple modifications were performed in the VGG19 architecture to ensure better training. The final architecture was reached by first freezing the first 10 layers of VGG19 and adding four convolution layers with different numbers of filters, each followed by "Batch Normalization" and "Max Pooling" layers to tackle the model overfitting. Then replacing the fully connected layer with two dense layers and a "Dropout" and "Ridge L2 Regularization" were added as well to prevent overfitting. Figure 14 summarizes the final modifications added to the VGG19 model.

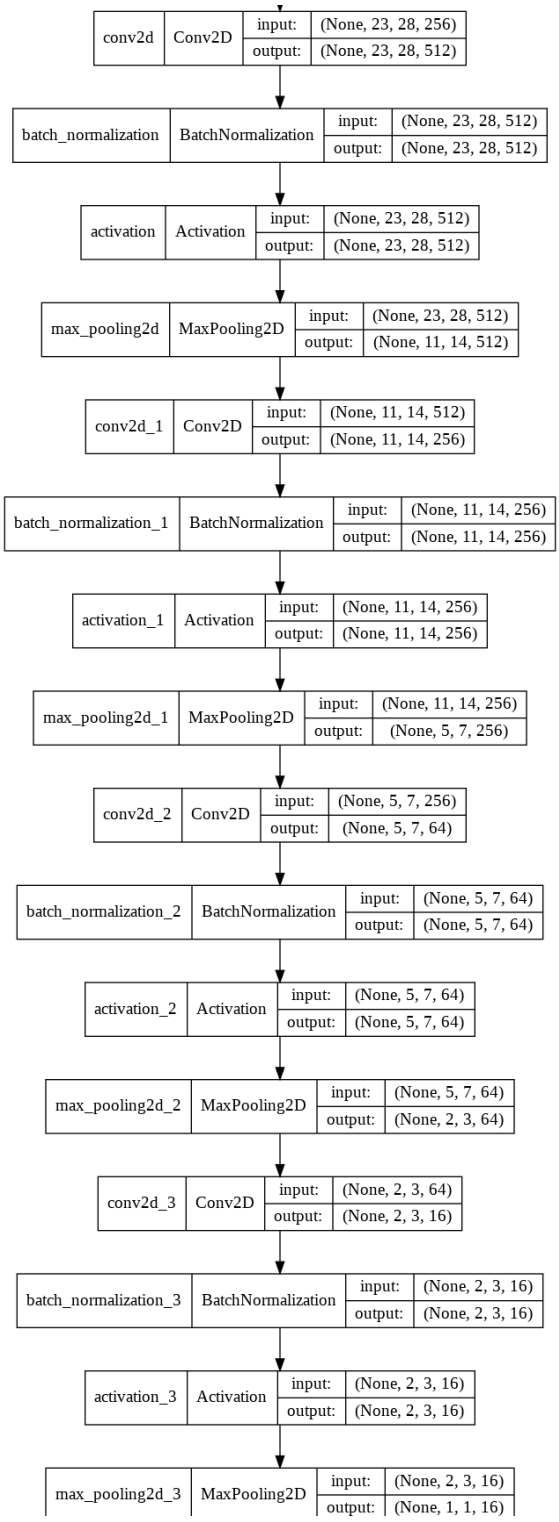

Figure 14 Modified VGG19 Model

4.2. Hardware

4.2.1 Software Installation

We use Raspberry Pi Imager to write images to SD cards for the Raspberry Pi. Raspberry Pi Imager comes with a few operating systems.

1. Download Raspberry Pi Imager

2. Drag an SD Card into the corresponding slot on the computer then format the SD Card.

3. From Pi Imager choose Raspberry Pi OS (32-bit) and the SD Card.

4. Click CTRL + Shift + X for enabling ssh.

5. Choose **Write** to begin the image writing process.

4.2.2 Direct Ethernet Connection

1. In "Network Connection" in the laptop ➔ Wi-Fi ➔ Properties ➔ Sharing ➔ Choose Ethernet.
2. With any remote console enter **ssh pi@raspberrypi** with password **raspberry.**
3. Download VNC Viewer to see the OS of the Raspberry Pi.
4. In terminal, enter **sudo apt-get update ➔ sudo raspi-config ➔** choose interface options ➔ vnc for enabling vnc with raspberry pi.
5. Connect VNC with Pi with **raspberrypi.**

4.2.3. Camera Raspberry Pi

In terminal, enter **sudo apt-get update ➔ sudo raspi-config ➔** choose interface options ➔ choose camera for enabling vnc with rasp

4.2.4. Push button with Camera Raspberry Pi

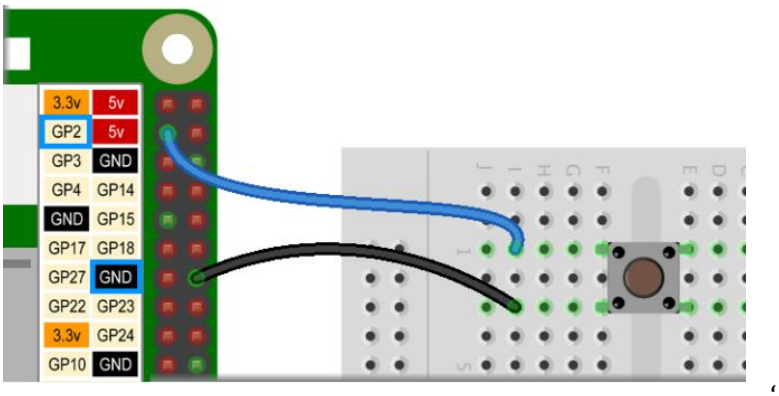

Figure 15 Push button with Camera Raspberry Pi

4.2.5. Installing the software Libraries for Machine Learning

4.2.5.1 The problems we faced

TensorFlow:

To in install the tensorflow we need some dependencies

1. gfortran
2. libhdf5-dev libc-ares-dev libeigen3-dev
3. libatlas-base-dev libopenblas-dev libblas-dev
4. openmpi-bin libopenmpi-dev
5. liblapack-dev cython
6. Keras_applications with version 1.0.8

7. Keras_preprocessing with version 1.1.0
8. six-wheel mock
9. pybind11
10. H5py with version 2.10.0
11. Upgrade setuptools 40.8.0 -> 52.0.0

But we had some problems with installing h5py and we found that it didn't install on the raspberry pi OS (32-Bit) that was recommended.

So we reinstall Raspberry Pi OS (64-Bit) Bullseye

After Installing Pi OS (64-Bit), We found that the Raspberry Pi camera isn't supported **(Picamera())** and the library **Libcamera** is supported instead of **Picamera.**

With the Library libcamera we faced more errors.

The last error occurred because the Raspberry Pi Kernel Driver-Pivariety Camera didn't support Our camera V2 imx219.

Finally, we found that libcamera does not work in Bullseye OS (64-Bit) with the camera V2 and the OS (64-Bit) not recommended. So, we return back to the OS (32-Bit).

In the OS (32-Bit) we solve the problem of dependencies of the TensorFlow by installing tensorflow lite.

4.2.5.2 The Final Libraries

TensorFlow Lite-runtime:

TensorFlow Lite is designed to make it easy to perform machine learning on devices, "at the edge" of the network, instead of sending data back and forth from a server.
The tflite_runtime package is a fraction the size of the full tensorflow package and includes the bare minimum code required to run inferences with TensorFlow Lite—primarily the Interpreter Python class.

We install a version 2.7.0

Numpy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

We install a version 1.22.2

OpenCV-Contrib

OpenCV contrib is a specialized module present in the Python programming language, which is exclusively needed for the system to run SURF feature descriptions alongside the OpenCV module present in the open-source library.

We install a version 4.5.4

4.3. Integration Module

4.3.1. Model Optimization

Models are converted to TensorFlow lite to increase the inference speed and decrease the models' size for better optimization while deployment.

4.3.2. Operating System

First, we went for the recommended bullseye 32-bit version and tested the camera and LCD on them, However, while importing the Tensorflow library, which is needed for the face detection by MTCNN, we found out that is not supported on this version. Then we upgraded to the bullseye 64-bit and installed all the required packages, but when testing the camera, we discovered that the "PI-Camera" module is not supported on the 64-bit, and we faced difficulties while importing the alternative module "Libcamera". To tackle these challenges, we returned to the 32-bit and overcome the need for TensorFlow by replacing the "MTCNN" detector with the Cascaded filter offered by OpenCV, replacing the Tensorflow package with the TFlite_runtime for importing the Interpreter for TFLite models' inference.

4.3.3. Button

When the button is pressed, it might rapidly oscillate between states before it settles, resulting in electrical noise, that's is known as the "Bouncing Effect". This is tackled in coding by setting a corresponding delay to the button's bouncing rate and checking the button state before and after pressing it.

4.3.4. LCD

Minor challenges we faced during testing the LCD, due to the instability of the wires connection, are solved by welding the cables in the breadboard.

4.3.5. Synchronization

The final scenario is as follows:

First, the quotes are randomly displayed on the LCD until the user pushes the button then the LCD is cleared, and the camera takes the user's photo. If the face is blurred or unfound, a message is displayed on the LCD informing the user that an error has occurred while capturing and random quotes are displayed again, otherwise, the captured images are passed to the model for inference and the predicted class is first displayed on the LCD with its confidence percentage, followed by a motivational quote.

When the raspberry pi is booted, the final script is scheduled to run automatically in the background.

CHAPTER 5: EVALUATION AND RESULTS

5.1. Models Evaluation

F1-Score is used here as an evaluation metric since the intents and attributes in the dataset are unbalanced. F1-score is a trade-off between precision and recall.

"The Precision is the number of true positive results divided by the number of all positive results, including those not identified correctly" and "the recall is the number of true positive results divided by the number of all samples that should have been identified as positive" [7].

F1-score equation:

$$f1\ score = 2\times \frac{\text{precision}\times \text{recall}}{\text{precision}+ \text{recall}}$$

The values of the F1-score ranges between 0 and 1, as the value approaches 1, this means there is a good trade-off between precision and recall and vice-versa.

5.2.Results

5.2.1 Machine Learning Approaches

Figure 15 shows the comparison between Logistic Regression, XGBoost and SVM models with respect to the Machine learning Approach 1 and 2 for landmarks extraction. As shown, XGBOOST has achieved the highest accuracy with Machine Learning approach 1.

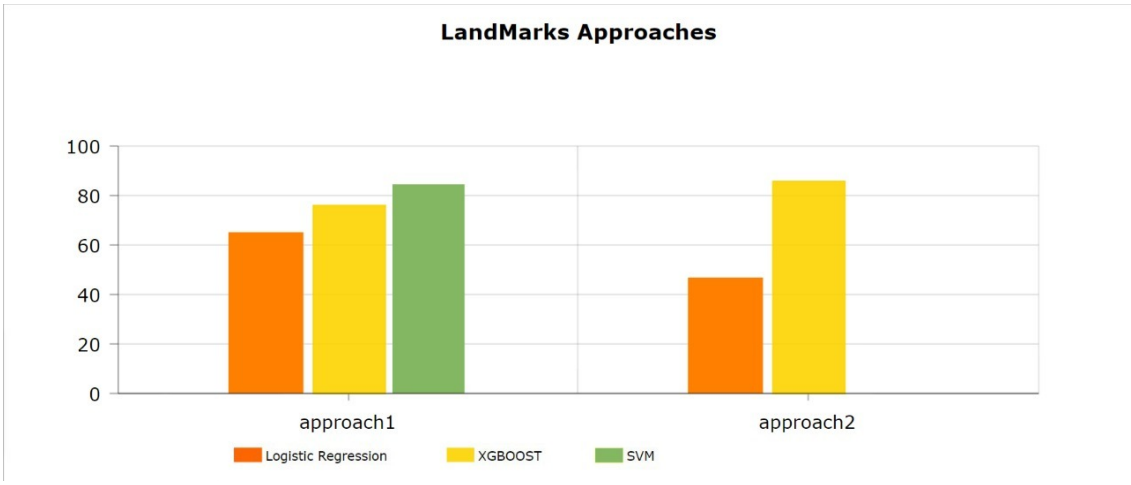

Figure 16 Machine Learning Models' Comparison

5.2.2 Deep Learning Approaches

Figure 16 shows the comparison between building models from scratch vs using pre-trained models. The modified VGG19 has shown promising results in terms of accuracy when using testing images from the raspberry pi camera.

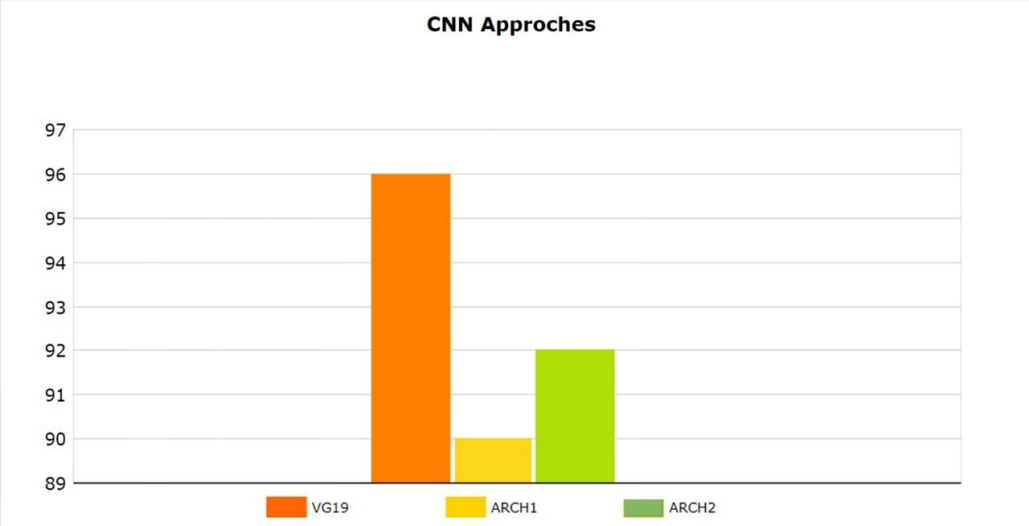

Figure 17 Deep Learning Models' Comparison

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this project, we proposed and implemented a system for facial emotion recognition. The system consists of three main modules: The first module is the software, where models are developed to detect humans' emotions, the second module is the hardware where the required kits are connected. The third one is the Integration module where the emotion detection model is hosted on raspberry pi and the requirements are set up for use. Our system showed promising results while deployed and tested on use-cases with an accuracy of 96%.

We believe there is still room for improvement for this project to make it more useful in the industry as:

- Add new features for instance: predicating more emotions, user's age.
- Extend the project to do facial recognition as well, so companies can estimate the employees' satisfaction while working.
- Instead of overwriting the taken photo, we can upload them on the cloud periodically.

REFERENCES

[1] M. Wang, Z. Wang, S. Zhang, J. Luan and Z. Jiao, "Face Expression Recognition Based on Deep Convolution Network," in 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 1–9, 2018, doi:10.1109/CISP-BMEI.2018.8633014.

[2] N. P. Gopalan, S. Bellamkonda and V. Saran Chaitanya, "Facial Expression Recognition Using Geometric Landmark Points and Convolutional Neural Networks," in 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), 1149–1153, 2018, doi:10.1109/ICIRCA.2018.8597226

[3] Hao Tang and Thomas S. Huang, "3D Facial Expression Recognition Based on Properties of Line Segments Connecting Facial Feature Points", in 2008 8th IEEE International Conference on Automatic Face & Gesture Recognition, 1–6, 2008, doi:10.1109/AFGR.2008.4813304.

[4] R. Raj S, P. D, and R. Kumar P, "Facial Expression Recognition using Facial Landmarks: A Novel Approach," Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 5, pp. 24–28, 2020, doi: 10.25046/aj050504.

[5] https://blog.techcraft.org/vgg-19-convolutional-neural-network/

[6] https://www.image-net.org/

[7] F-score - Wikipedia