A Thesis/Project/Dissertation Report

on

A SYSTEMATIC REVIEW ON FACE EMOJI RECOGNITION

Submitted in partial fulfillment of the

requirement for the award of the degree of

B.Tech Computer Science And Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. P. Raja Kumar
Designation- Assistant Professor

Submitted By

Ritika Goyal 18SCSE1010558
Gaurang Gupta 18SCSE1010029

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA OCTOBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the thesis, entitled **"A SYSTEMATIC REVIEW ON FACE EMOJI RECOGNITION"** in partial fulfillment of the requirements for the award of the B.Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of August 2021 to December 2021, under the supervision of Dr. Shiv Verma, Professor in Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ritika Goyal

Gaurang Gupta

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. P. Raja Kumar

Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Ritika Goyal-18SCSE1010558 Gaurang Gupta-18SCSE1010029 has been held on _________________ and his/her work is recommended for the award of Name of Degree.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: 20 December, 2021

Place: Greater Noida

ACKNOWLEDGEMENT

It is our privilege and solemn duty to express our deepest sense of gratitude to Mr. P. Raja Kumar, under whose guidance we carried out this work. We are indebted to him for his invaluable supervision, heart full cooperation and timely aid and advice till the completion of the report in spite of his pressing engagements. We wish to record our sincere gratitude for his constant support and encouragement in preparation of this report.

We take this opportunity to express our hearty thanks to all those who helped us in the completion of our project work. We are incredibly grateful to the author of various research papers, for helping us become aware of the research currently ongoing in the field. We are very thankful to our parents for their constant support and love.

Last, but not least, we would like to thank our classmates for their valuable comments, suggestions and unconditional support.

ABSTRACT

A developing assemblage of exploration investigates emoticon, which are visual images in PC interceded correspondence (CMC). In the long time since the primary arrangement of emoticon was delivered, research on it has been on the increment, but in an assortment of headings. We surveyed the surviving group of examination on emoticon and noticed the turn of events, utilization, capacity, and use of emoticon.

Facial emoticon recognizer is an end client application which identifies the appearance of the individual in the video being caught by the camera. The smiley pertinent to the statement of the individual in the video is displayed on the screen which changes with the adjustment of the articulations. Looks are significant in human correspondence and connections. Likewise, they are utilized as a significant instrument in examinations about conduct and in clinical fields. Facial emoticon recognizer gives a quick and functional methodology for non-intrusive feeling location.

The design was to foster an astute framework for facial based demeanor grouping utilizing CNN calculation. Haar classifier is utilized for face location and CNN calculation is used for the appearance identification and giving the emoji applicable to the demeanor as the yield. Roused by the examination results, we further propose to perform various tasks using multimodality gated intermittent unit (mmGRU) model to foresee the classifications and places of emoticons. The model uses not just multimodality data like text, picture and client socioeconomics, in addition it uses the solid relationships between emoticon classifications and their positions.

Proposed is a human feeling locator utilizing emoji utilizing AI, Java to foresee feelings of individuals and address them utilizing emoji. These incorporate picture obtaining, preprocessing of a picture, face location, include extraction, arrangement and afterward when the feelings are ordered the framework allots the client specific music as indicated by his feeling. Our framework centers around live recordings taken from the camera.

Keywords: emoji, emoticon, emotion expression, AI, feelings, arrangement.

CONTENTS

INTRODUCTION

Facial feeling acknowledgment and examination has been acquiring an incredible consideration in the progression of human machine interface as it gives a characteristic and productive way of conversing between people. Some applications are utilizing face and related expressions such as individual ID and access control, video call and remotely coordinating, legal applications, human-PC connection, robotized reconnaissance, cosmetology, etc. However, the conduct of the face demeanor discovery unquestionably influences the presentation of the relative multitude of uses.

Numerous techniques have been proposed to identify human face in pictures and recordings, they can be partitioned into four sorts: information based strategies, feature based techniques, format based techniques and appearance-based techniques. At the point when these techniques are utilized independently, they do not encompass the relative multitude of issues of face recognition like posture, appearance, direction. Subsequently it is prescribed to work with a few progressive or equal techniques. The vast majority of the current look acknowledgment techniques which are prominently utilized till today are centered around acknowledgment of five essential articulation classes, for example, joy, misery, dread, outrage and nausea.

Emoticons are being utilized increasingly in network correspondence, and the manner in which they are utilized is turning out to be more broadened as well. They have extraordinary semantic and enthusiastic components, but at the same time they are firmly identified with advertising, law, medical services and numerous different regions. The exploration on emoticon has turned into an interesting issue in the scholastic field, and that is just the beginning and more researchers from the fields of processing, correspondence, promoting, social science, etc are concentrating on them.

The feelings every now and again ease and decide associations among the people. The setting of feelings explicitly draws out the perplexing and peculiar social correspondence . Social correspondence is recognized as the judgment of the other individual's mind-set that is dependent on the emoticon. The acknowledgment of feelings can be distinguished through different signs by the "non-verbal communication, voice pitch" just as by means of "more perplexing strategies, such [as] electroencephalography (EEG)." Nonetheless, the easier, and attainable methodology is to examine the look. By noticing the look, the individual's state of mind and conduct are easily judged.

The task of emotion recognition is particularly difficult for two reasons: 1) There does not exist a large database of training images and 2) classifying emotion can be difficult depending on whether the input image is static or a transition frame into a facial expression. The latter issue is particularly difficult for real-time detection where facial expressions vary dynamically. Most applications of emotion recognition examine static images of facial expressions. We have developed a system for detecting human emotions in different scenes, angles, and lighting conditions in real-time.

LITERATURE SURVEY

Darwin suggested foremost that a portion of these looks of feeling have their beginnings in the development of the human species. These face appearances helped the living things live in light of the fact that it looked important to social creatures like people or chimpanzees to communicate these impending practices given by the feelings so they could stay away from battles, struggle, conflict, risk, or permit solace, approach, etc. Be that as it may, these feeling articulations are not equipped for alteration by friendly learning. Various societies apply distinctive showcase rules to coordinate their expression of feelings. Albeit the current proof backing Darwin's essential reason it isn't without debate. Future mechanical advances will permit look exploration to grow to address a considerable lot of the significant issues that remain.

Ekman and Friesen noticed the facial muscles which are significant in communicating feeling and made their discoveries to an arrangement of 46 activity units (AUs). These activity units, in which some are raising the inward eyebrow and some raising the external eyebrow, were critical in thinking about human articulations. Before this framework was distributed, look research depended intensely on human naming of model articulations and numerous analysts were worried about inclination identified with social setting or the labeller's enthusiastic state at that point. The coming of Ekman and Feisen's facial activity coding framework in 1977 set out to settle these worries and immediately turned into the brilliant norm.

Ekman's Facial activity coding framework was observed to be very exhaustive as specialists had the option to recognize more than 7000 mixes of the 46 nuclear AUs however note that genuine articulations are dynamic and there's something else to them besides still pictures of contracted muscles. Even after the FACS was taken on inside the PC vision local area, PC liveliness scientists were attempting to concur upon a framework for addressing a face moving. The Motion Pictures Expert Group (MPEG) presented a thoroughly examined set of facial liveliness boundaries (FAPs) which became standard in 1999.

In 2004, Paul Viola and Michael Jones fostered a very proficient face indicator with elite by utilizing an adaboost learning calculation to order includes got from Haar-like elements. This strategy was applied by Wang et al for looks he later had the option to sort faces into one of 7 model look with 92.4% exactness.

An examination by White slope and Omlin tracked down that the utilization of Haar highlights in mix with the Ada help boosting calculation was somewhere around two significant degrees quicker than the more standard arrangement utilizing SVMs and Gabor channels without a critical drop in execution. All the more as of late, work done by Happy et al in 2015 tracked down that the viola-jones calculation was significant not in direct recognition of feelings but rather as a preparing step to proficiently identify the main face districts (IE lip corners and eyebrow edges) which were then, at that point, further handled with nearby parallel example histograms. Their answer performed likewise to other best in class techniques yet needed undeniably less computational time. Eigen faces are figured for all the preparation pictures and grouping of test picture is finished by the accompanying advances: Generate vector of the relative multitude of pictures and grid of vector is made for each picture type, Mean of the preparation faces is made, Subtract the test picture with every one of the mean picture, Co-fluctuation framework is determined for every one of the distinction vectors made.

PROJECT FORMULATION

Facial emoji recognizer is an end user application which detects the expression of the person in the video being captured by the camera. The smiley relevant to the expression of the person in the video is shown on the screen which changes with the change in the expressions. We implement HAAR classifier for face detection, CNN algorithm for expression detection and two functions are used: relu and soft max(these are activity functions).

PROJECT OBJECTIVE

There are certain objectives of the thesis necessary to understand the core aspects for the identification of research outcomes which are listed below:

- To investigate emotional recognition using facial expression by emoji in real time
- To develop the parameters of measuring the facial expression by emoji texting
- To understand the facial emotion recognition in real time

The stated objectives of this thesis highlighted the reasons behinds the facial recognition emotions in the real time. The main goal of this project is to implement the recognition of the facial emotion for real time. The implementation of an application process is to identify the facial emotion recognition reasons with highlighted emoji indicator for the identification of six expressions. Such expressions are neutral, fear, anger, happy, sad, and surprise. Furthermore, facial expressions are investigated to realize the real impact of the emoji. The expressions are the actual predictor of human behavior. If expressions are good, then it means the person is in a good mood or has a joyful personality. If the person's expressions show anger or sorrow that means the person is not feeling well or his or her personality does not have joyful traits. The expressions are related with the person's behavior and personality. Understanding expressions are important because they give a lot of information regarding behavior and moods of people through expressions; one can know what is going inside the person's mind and how it can be handled

METHODLOGY

Facial expressions can be described as the arrangement of facial muscles to convey a Certain emotional state to the observer in simple words. Emotions can be divided into six broad categories—Anger, Disgust, Fear, Happiness, Sadness, Surprise, and Neutral. In this, train a model to differentiate between these, train a convolutional neural network using the FER2013 dataset and will use various hyper-parameters to fine-tune the model.

1. Decomposing an image.

Images are composed of pixels and these pixels are nothing more than numbers. Often it is considered that the Coloured images can be divided into three colour channels, which are: red, green, and blue and each channel is represented by a grid (2-dimensional array). Each cell in the grid stores a number between 0 and 255 which denotes the intensity of that cell.

2. Importing Necessary Libraries

3. Define Data Loading Mechanism

Now, we will define the load_data() function which will efficiently parse the data file and Extract necessary data and then convert it into a usable image format. All the images in our dataset are 48x48 in dimensions. Since these images are gray-scale, there is only one channel. We will extract the image data and rearrange it into a 48x48 array. Then convert it into unsigned integers and divide it by 255 to normalize the data. 255 is the maximum possible value of a single cell and by dividing every element by 255, we ensure that all our values range between 0 and 1. We will check the Usage column and store the data in separate lists, one for training the network and the other for testing it.

4. Defining the model.

We will use Keras to create a Sequential Convolutional Network. Which means that our Neural network will be a linear stack of layers. This network will have the following components:

A. Convolutional Layers: These layers are the building blocks of our network and these compute dot product between their weights and the small regions to which they are linked to. This is considered as the method in which layers learn certain features from these images.

B. Activation functions: are those functions which are applied to the outputs of all layers in the network. In this project, we will resort to the use of two functions— Relu and Soft max.
C. Pooling Layers: These layers will down sample the operation along the dimensions. This helps reduce the spatial data and minimize the processing power that is required.

D. Dense layers: These layers are present at the end of a C.N.N. and these take in all the feature data generated by the convolution layers and does the decision making.

E. Dropout Layers: randomly turns off few neurons in the network to prevent over fitting.

SYSTEM REQUIREMENTS

SOFTWARE AND HARDWARE SPECIFICATIONS

<u>Development on the SOFTWARE</u>:

Technology	: MATLAB
Web Technologies	: Html, JSP, JavaScript, CSS, Adobe
Database	: MySql5.0
JDK Version	: JDK1.5

Development on the HARDWARE:

Processor	: Pentium
RAM	: 1GB

FACE DETECTION

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step.

There are two types of face detection problems:

1) Face detection in images and
2) Real-time face detection

FACE DETECTION STEPS

1. Pre-Processing:

To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

2.Classification:

Neural networks are implemented to classify the images as faces or non faces by training on these examples. We use both our implementation of the neural network and the MATLAB neural network toolbox for this task. Different network configurations are experimented with to optimize the results.

3.Localization:

The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:- Position

Scale Orientation Illumination.

FACE RECOGNIZATION

There are two predominant approaches to the face recognition problem: Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature.

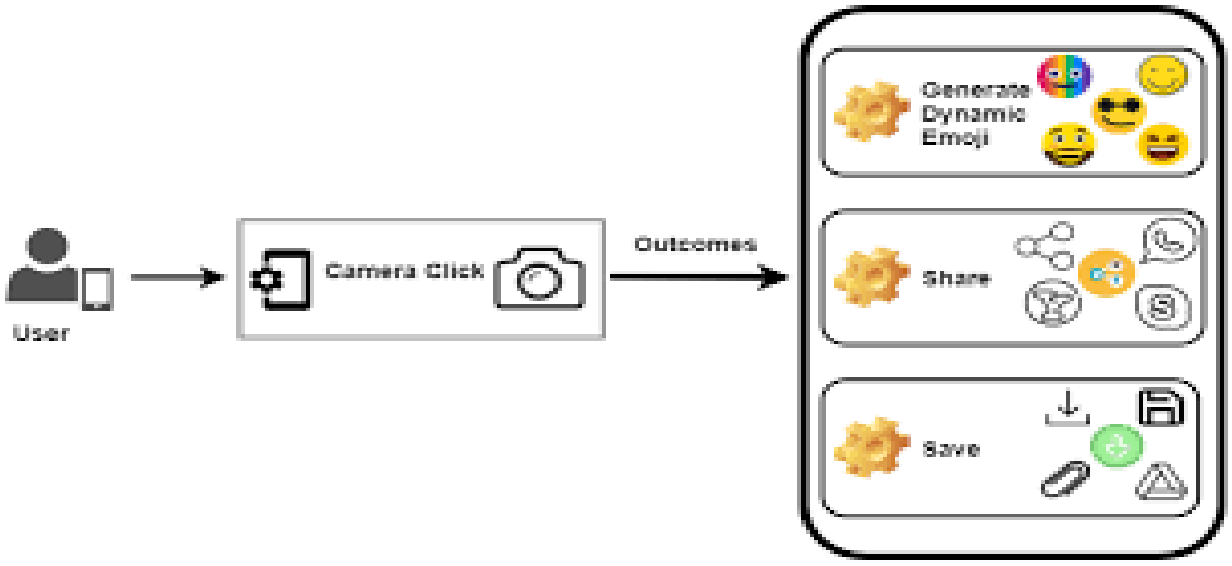
Recognition algorithms can be divided into two main approaches:

1. Geometric:

It is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances.

2. Photometric:

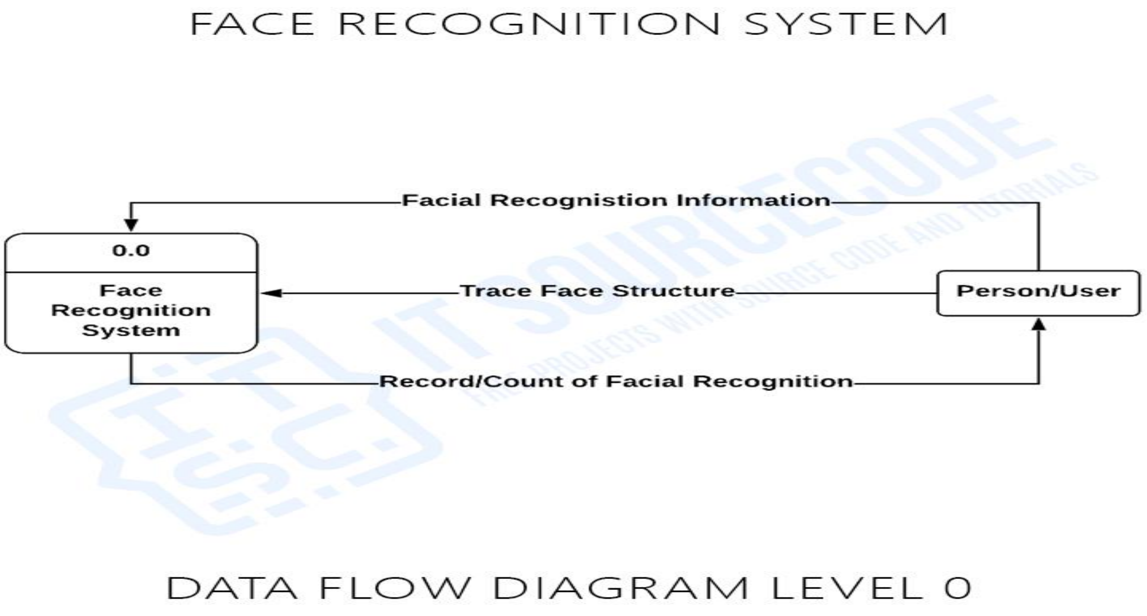
It is used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface.



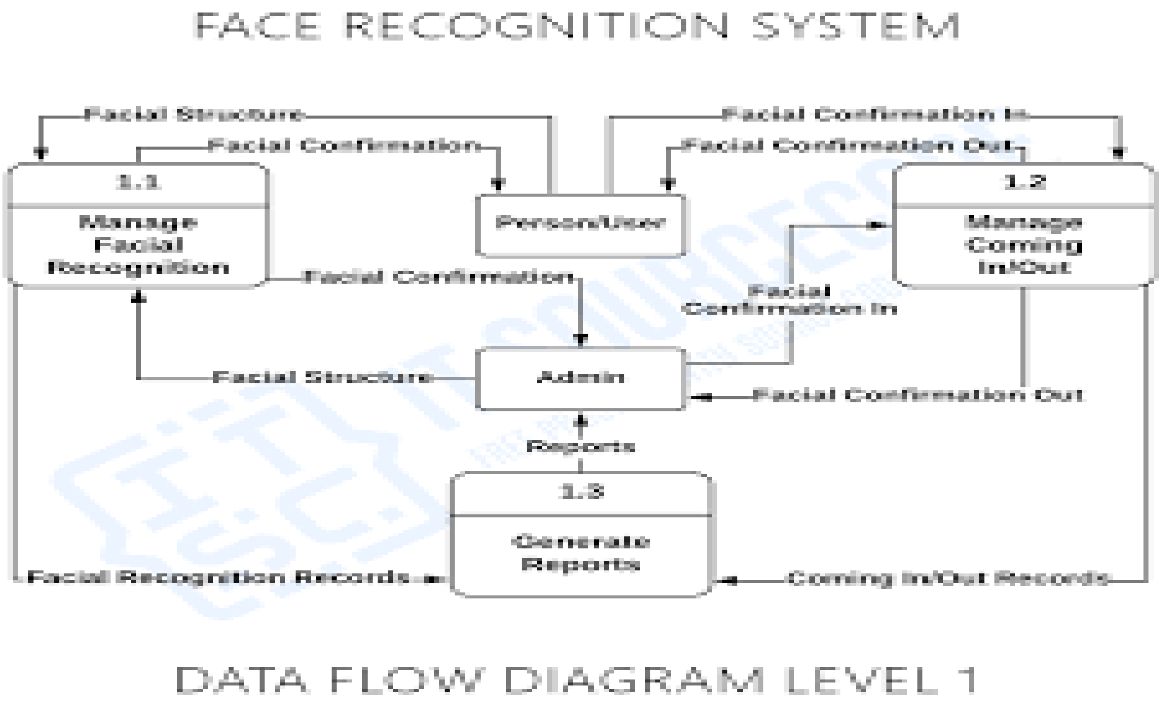
PROJECT DESIGN

DATA FLOW DIAGRAM

1. LEVEL 0

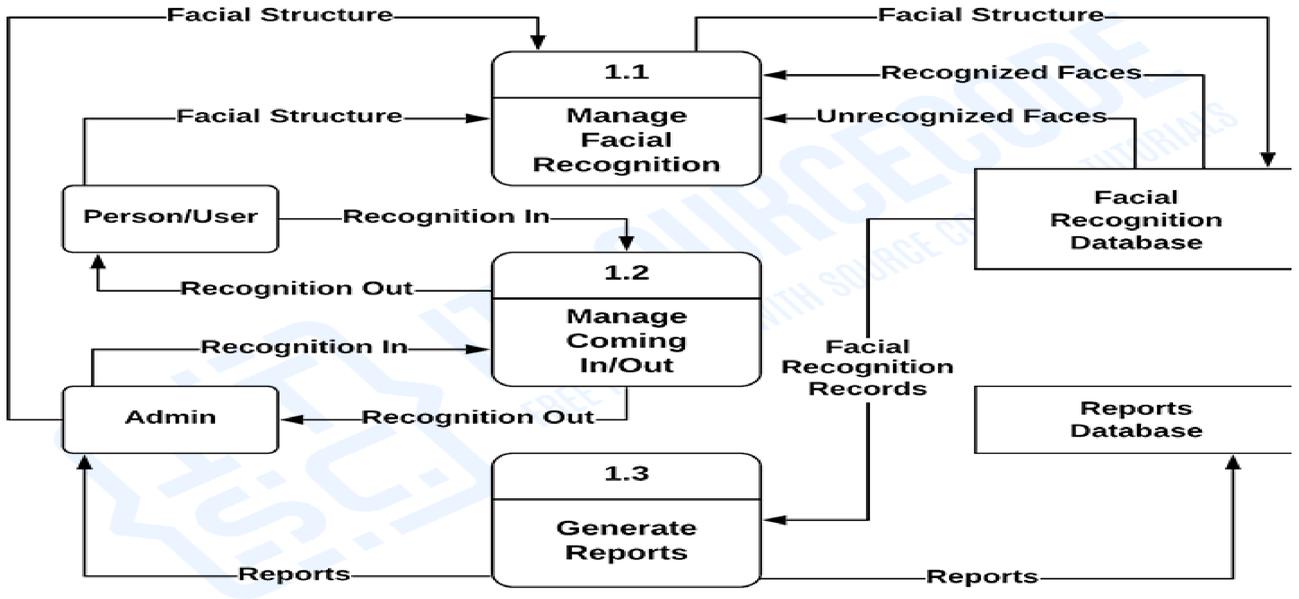

FACE RECOGNITION SYSTEM
Facial Recognistion Information
0.0
Face Recognition System
Trace Face Structure
Person/User
Record/Count of Facial Recognition
DATA FLOW DIAGRAM LEVEL 0

2. LEVEL 1



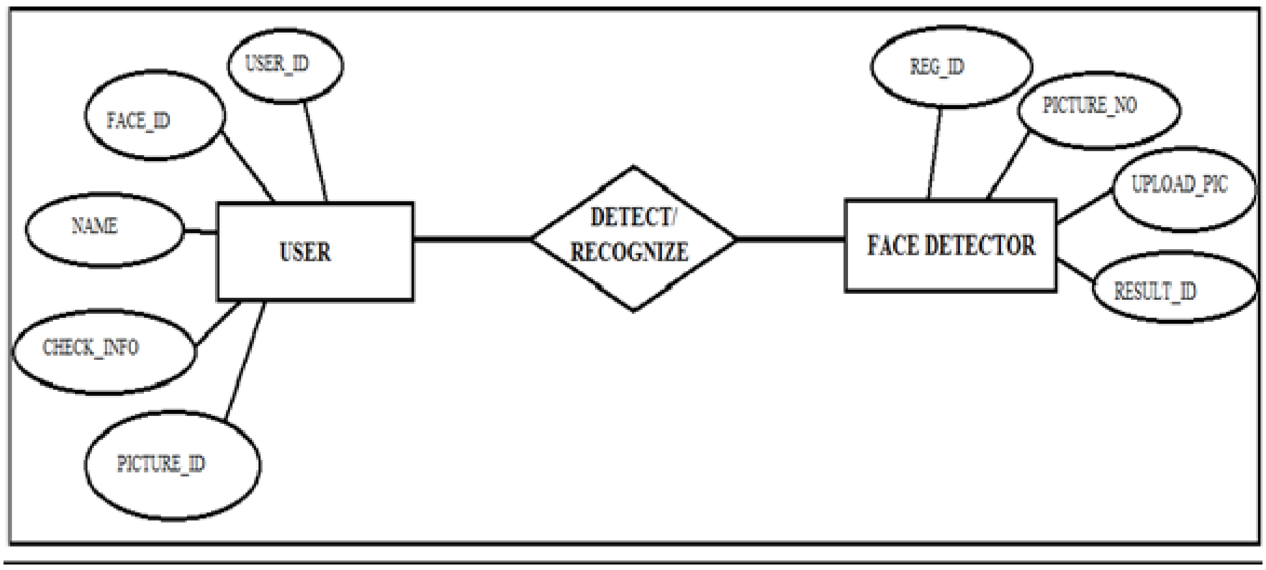
3. LEVEL 2

FACE RECOGNITION SYSTEM



DATA FLOW DIAGRAM LEVEL 2

ER DIAGRAM



IMPORTANT CODES

ExampleInstrumentedTest.java

```
package com.example.android.emojify;

import android.content.Context;
import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumentation test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() throws Exception {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getTargetContext();

        assertEquals("com.example.android.emojify", appContext.getPackageName());
    }
}
```

Emojifier.java

```
package com.example.android.emojify;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.util.SparseArray;
import android.widget.Toast;

import com.google.android.gms.vision.Frame;
import com.google.android.gms.vision.face.Face;
import com.google.android.gms.vision.face.FaceDetector;

import timber.log.Timber;
```

```
class Emojifier {


    private static final float EMOJI_SCALE_FACTOR = .9f;
    private static final double SMILING_PROB_THRESHOLD = .15;
    private static final double EYE_OPEN_PROB_THRESHOLD = .5;

    /**
     * Method for detecting faces in a bitmap, and drawing emoji depending on the facial
     * expression.
     *
     * @param context The application context.
     * @param picture The picture in which to detect the faces.
     */
    static Bitmap detectFacesandOverlayEmoji(Context context, Bitmap picture) {

        // Create the face detector, disable tracking and enable classifications
        FaceDetector detector = new FaceDetector.Builder(context)
                .setTrackingEnabled(false)
                .setClassificationType(FaceDetector.ALL_CLASSIFICATIONS)
                .build();

        // Build the frame
        Frame frame = new Frame.Builder().setBitmap(picture).build();

        // Detect the faces
        SparseArray<Face> faces = detector.detect(frame);

        // Log the number of faces
        Timber.d("detectFaces: number of faces = " + faces.size());

        // Initialize result bitmap to original picture
        Bitmap resultBitmap = picture;

        // If there are no faces detected, show a Toast message
        if (faces.size() == 0) {
            Toast.makeText(context, R.string.no_faces_message,
Toast.LENGTH_SHORT).show();
        } else {

            // Iterate through the faces
            for (int i = 0; i < faces.size(); ++i) {
                Face face = faces.valueAt(i);

                Bitmap emojiBitmap;
                switch (whichEmoji(face)) {
                    case SMILE:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.smile);
                        break;
                    case FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.frown);
                        break;
                    case LEFT_WINK:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.leftwink);
                        break;
                    case RIGHT_WINK:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.rightwink);
                        break;
                    case LEFT_WINK_FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.leftwinkfrown);
                        break;
                    case RIGHT_WINK_FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.rightwinkfrown);
                        break;
                    case CLOSED_EYE_SMILE:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.closed_smile);
                        break;
                    case CLOSED_EYE_FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.closed_frown);
                        break;
                    default:
                        emojiBitmap = null;
                        Toast.makeText(context, R.string.no_emoji,
Toast.LENGTH_SHORT).show();
                }

                // Add the emojiBitmap to the proper position in the original image
                resultBitmap = addBitmapToFace(resultBitmap, emojiBitmap, face);
            }
        }


        // Release the detector
        detector.release();

        return resultBitmap;
    }


    /**
     * Determines the closest emoji to the expression on the face, based on the
     * odds that the person is smiling and has each eye open.
```

```
     *
     * @param face The face for which you pick an emoji.
     */

    private static Emoji whichEmoji(Face face) {
        // Log all the probabilities
        Timber.d("whichEmoji: smilingProb = " + face.getIsSmilingProbability());
        Timber.d("whichEmoji: leftEyeOpenProb = "
                + face.getIsLeftEyeOpenProbability());
        Timber.d("whichEmoji: rightEyeOpenProb = "
                + face.getIsRightEyeOpenProbability());


        boolean smiling = face.getIsSmilingProbability() > SMILING_PROB_THRESHOLD;

        boolean leftEyeClosed = face.getIsLeftEyeOpenProbability() <
EYE_OPEN_PROB_THRESHOLD;
        boolean rightEyeClosed = face.getIsRightEyeOpenProbability() <
EYE_OPEN_PROB_THRESHOLD;


        // Determine and log the appropriate emoji
        Emoji emoji;
        if (smiling) {
            if (leftEyeClosed && !rightEyeClosed) {
                emoji = Emoji.LEFT_WINK;
            } else if (rightEyeClosed && !leftEyeClosed) {
                emoji = Emoji.RIGHT_WINK;
            } else if (leftEyeClosed) {
                emoji = Emoji.CLOSED_EYE_SMILE;
            } else {
                emoji = Emoji.SMILE;
            }
        } else {
            if (leftEyeClosed && !rightEyeClosed) {
                emoji = Emoji.LEFT_WINK_FROWN;
            } else if (rightEyeClosed && !leftEyeClosed) {
                emoji = Emoji.RIGHT_WINK_FROWN;
            } else if (leftEyeClosed) {
                emoji = Emoji.CLOSED_EYE_FROWN;
            } else {
                emoji = Emoji.FROWN;
            }
        }


        // Log the chosen Emoji
        Timber.d("whichEmoji: " + emoji.name());

        // return the chosen Emoji
        return emoji;
    }

    /**
     * Combines the original picture with the emoji bitmaps
     *
     * @param backgroundBitmap The original picture
     * @param emojiBitmap      The chosen emoji
     * @param face             The detected face
     * @return The final bitmap, including the emojis over the faces
     */
    private static Bitmap addBitmapToFace(Bitmap backgroundBitmap, Bitmap emojiBitmap,
Face face) {

        // Initialize the results bitmap to be a mutable copy of the original image
        Bitmap resultBitmap = Bitmap.createBitmap(backgroundBitmap.getWidth(),
                backgroundBitmap.getHeight(), backgroundBitmap.getConfig());

        // Scale the emoji so it looks better on the face
        float scaleFactor = EMOJI_SCALE_FACTOR;

        // Determine the size of the emoji to match the width of the face and preserve aspect
ratio
        int newEmojiWidth = (int) (face.getWidth() * scaleFactor);
        int newEmojiHeight = (int) (emojiBitmap.getHeight() *
                newEmojiWidth / emojiBitmap.getWidth() * scaleFactor);



        // Scale the emoji
        emojiBitmap = Bitmap.createScaledBitmap(emojiBitmap, newEmojiWidth,
newEmojiHeight, false);

        // Determine the emoji position so it best lines up with the face
        float emojiPositionX =
                (face.getPosition().x + face.getWidth() / 2) - emojiBitmap.getWidth() / 2;
        float emojiPositionY =
                (face.getPosition().y + face.getHeight() / 2) - emojiBitmap.getHeight() / 3;

        // Create the canvas and draw the bitmaps to it
        Canvas canvas = new Canvas(resultBitmap);
        canvas.drawBitmap(backgroundBitmap, 0, 0, null);
        canvas.drawBitmap(emojiBitmap, emojiPositionX, emojiPositionY, null);

        return resultBitmap;
    }


    // Enum for all possible Emojis
    private enum Emoji {
        SMILE,
        FROWN,
        LEFT_WINK,
        RIGHT_WINK,
        LEFT_WINK_FROWN,
        RIGHT_WINK_FROWN,
        CLOSED_EYE_SMILE,
        CLOSED_EYE_FROWN
    }

}
```

MainActivity.java

```
package com.example.android.emojify;


import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import androidx.annotation.NonNull;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.File;
import java.io.IOException;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import timber.log.Timber;

public class MainActivity extends AppCompatActivity {


    private static final int REQUEST_IMAGE_CAPTURE = 1;
    private static final int REQUEST_STORAGE_PERMISSION = 1;
    private static final String FILE_PROVIDER_AUTHORITY =
"com.example.android.fileprovider";

    @BindView(R.id.image_view)
    ImageView mImageView;

    @BindView(R.id.emojify_button)
    Button mEmojifyButton;
    @BindView(R.id.share_button)
    FloatingActionButton mShareFab;
    @BindView(R.id.save_button)
    FloatingActionButton mSaveFab;
    @BindView(R.id.clear_button)
    FloatingActionButton mClearFab;

    @BindView(R.id.title_text_view)
    TextView mTitleTextView;

    private String mTempPhotoPath;

    private Bitmap mResultsBitmap;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Bind the views
        ButterKnife.bind(this);

        // Set up Timber
        Timber.plant(new Timber.DebugTree());
    }

    /**
     * OnClick method for "Emojify Me!" Button. Launches the camera app.
     */
    @OnClick(R.id.emojify_button)
    public void emojifyMe() {
        // Check for the external storage permission
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.WRITE_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {

            // If you do not have permission, request it
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                    REQUEST_STORAGE_PERMISSION);
        } else {
            // Launch the camera if the permission exists
            launchCamera();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        // Called when you request permission to read and write to external storage
        switch (requestCode) {
            case REQUEST_STORAGE_PERMISSION: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    // If you get permission, launch the camera
                    launchCamera();
                } else {
                    // If you do not get permission, show a Toast
                    Toast.makeText(this, R.string.permission_denied,
Toast.LENGTH_SHORT).show();
                }
                break;
            }
        }
    }

    /**
     * Creates a temporary image file and captures a picture to store in it.
     */
    private void launchCamera() {

        // Create the capture image intent
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            // Create the temporary File where the photo should go
            File photoFile = null;
            try {
                photoFile = BitmapUtils.createTempImageFile(this);
            } catch (IOException ex) {
                // Error occurred while creating the File
                ex.printStackTrace();
            }
            // Continue only if the File was successfully created
            if (photoFile != null) {

                // Get the path of the temporary file
                mTempPhotoPath = photoFile.getAbsolutePath();

                // Get the content URI for the image file
                Uri photoURI = FileProvider.getUriForFile(this,
                        FILE_PROVIDER_AUTHORITY,
                        photoFile);

                // Add the URI so the camera can store the image
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);

                // Launch the camera activity
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
            }
        }
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // If the image capture activity was called and was successful
        if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK)
{
            // Process the image and set it to the TextView
            processAndSetImage();
        } else {

            // Otherwise, delete the temporary image file
            BitmapUtils.deleteImageFile(this, mTempPhotoPath);
        }
    }

    /**
     * Method for processing the captured image and setting it to the TextView.
     */
    private void processAndSetImage() {

        // Toggle Visibility of the views
        mEmojifyButton.setVisibility(View.GONE);
        mTitleTextView.setVisibility(View.GONE);
        mSaveFab.setVisibility(View.VISIBLE);
        mShareFab.setVisibility(View.VISIBLE);
        mClearFab.setVisibility(View.VISIBLE);

        // Resample the saved image to fit the ImageView
        mResultsBitmap = BitmapUtils.resamplePic(this, mTempPhotoPath);


        // Detect the faces and overlay the appropriate emoji
        mResultsBitmap = Emojifier.detectFacesandOverlayEmoji(this, mResultsBitmap);

        // Set the new bitmap to the ImageView
        mImageView.setImageBitmap(mResultsBitmap);
    }
```

```
    /**
     * OnClick method for the save button.
     */
    @OnClick(R.id.save_button)
    public void saveMe() {
        // Delete the temporary image file
        BitmapUtils.deleteImageFile(this, mTempPhotoPath);

        // Save the image
        BitmapUtils.saveImage(this, mResultsBitmap);
    }

    /**
     * OnClick method for the share button, saves and shares the new bitmap.
     */
    @OnClick(R.id.share_button)
    public void shareMe() {
        // Delete the temporary image file
        BitmapUtils.deleteImageFile(this, mTempPhotoPath);

        // Save the image
        BitmapUtils.saveImage(this, mResultsBitmap);

        // Share the image
        BitmapUtils.shareImage(this, mTempPhotoPath);
    }

    /**
     * OnClick for the clear button, resets the app to original state.
     */
    @OnClick(R.id.clear_button)
    public void clearImage() {
        // Clear the image and toggle the view visibility
        mImageView.setImageResource(0);
        mEmojifyButton.setVisibility(View.VISIBLE);
        mTitleTextView.setVisibility(View.VISIBLE);
        mShareFab.setVisibility(View.GONE);
        mSaveFab.setVisibility(View.GONE);
        mClearFab.setVisibility(View.GONE);

        // Delete the temporary image file
        BitmapUtils.deleteImageFile(this, mTempPhotoPath);
    }
}
```

OUTPUT

CONCLUSION

The facial expression is specifically based on the emoticon identification system. It is identified as an open source extension to the tracker.js (A modern approach for Computer Vision on the web) framework that converts into expression of human facial to match best emoticon. The emoticons are the major parts towards the digital communication system. Emoticons, are also called, emoji's that "are used to express the emotions of a person through text in a way that is not possible with just words. The importance of emoji has become so huge that they [also have been specifically] annotated with the WordNets". The vision of computer has grown at large scale that use as commercial product for the benefits of high speed image.

The key task at hand ensures the face expression and face detection for the recognition of emotion. The formed task behind the face detection and emotional recognized gained from the eigen faces, fisher faces, and viola jones are the detections framework that prologs the hausdorff distance. The study exploration of the facial action coding system that observed the facial muscles, that plays an important role in the compiled and expressing emotions to find out the action units. Such action units are raising from the outer eyebrow and inner eyebrow, which is important for the quantification of human expressions

The crux of the overall introduction provides the general understanding of the emotional recognition to obtain the facial expression using emoji. The most applications of emoji recognition investigate the images of static by the facial expression images. Here, the investigation of application of the CNN for the identification of the emotion recognition. The computational requirements as well as the complexity of the CNN, for the optimization of the efficient frame-by-frame classification have been executed in the emotional recognition for the facial expression.

It has been observed from the study of emoji real time use that detects the human emotions in different scenes, lighting conditions as well as angles in real-time. The application of novel results, that reveals the consolidations of the emoji with superimposed with the subject of faces.

REFERENCES

[1] Emotional Recognition Using Facial Expression by Emoji in Real Time Mohammed Rajhi* 3820 Nicholasville Rd, APT#1206, Lexington, Kentucky, USA.

[2] “Emotion Recognition Using Facial Expression Analysis” ALI GHALI 2. Dr. MHD BASSAM KURDY,JATIT , 30 September 2018.

[3] Smileys & People 2017 emojipedia.

[4] Emoji: Representations of Nonverbal Symbols in Communication Technology D Tandyonomanu* and Tsuroyya

[5] Using Emojis: Self-Presentation and Different Meaning Creation Approaches, Dr. Sibel Onursoy