1. INTRODUCTION

Imagine a tool that translates feelings into emojis using the power of machine learning! That's basically what your project could be. You'll train a model to understand emotions, whether it's through text analysis, analysing facial expressions, or even voice recognition.

Emojis are small images often included in text messages on social media. Combining visual and textual content in the same message forms a modern way of communicating. Emojis, or avatars, are a way of displaying non-verbal cues. These cues have become essential in online chats, product reviews, brand sentiment, and more.

2. OBJECTIVE

The objective of an emojify machine learning project can be multi-faceted, depending on your chosen approach and interests. Here are some potential objectives to consider:

Understanding emotions:

Analyze emotional trends: Explore the prevalence of different emotions in various datasets, like social media posts or customer reviews. Uncover hidden patterns and gain insights into user behavior or opinions.

Generate emotional responses: Create a model that can generate emojis or text responses that match a specific emotional tone. This could be used for chatbots, virtual assistants, or personalized recommendations.

Enhancing communication:

Translate text to emojis: Develop a tool that automatically suggests emojis that best convey the sentiment of a written message. This can add clarity and nuance to online communication.

Create real-time emoji overlays: Use facial recognition to translate real-time expressions into appropriate emojis, aThe objective of an emojify machine learning project can be multi-faceted, depending on your chosen approach and interests. Here are some potential objectives to consider:

Personalize emoji usage: Train a model on someone's individual preferences or facial expressions to create custom emojis that truly represent them.

Exploring creativity and entertainment:

Generate artistic emoji creations: Use your model to produce unique and visually appealing emoji combinations or animations.

Develop emoji-based games: Create interactive games that utilize emoji recognition or generation mechanics for engaging and emotional experiences.

Explore potential social media applications: Design features for platforms like Instagram or Twitter that leverage emoji generation or analysis for enhanced user engagement.

3. EXISTING SYSTEM

Text-based Emojification:

Twitter Emoji Prediction: Twitter has explored predicting appropriate emojis for tweets based on text content and user behavior.

Emojify Tool: This online tool uses pre-defined rules and sentiment analysis to suggest emojis for text input.

DeepMoji: This research project utilized a deep learning model to predict emojis based on text, achieving high accuracy compared to simpler methods.

Image-based Emojification:

EmojiGAN: This research project utilized a Generative Adversarial Network (GAN) to generate artistic and creative emojis based on facial expressions in images.

Real-time Emotion Recognition: Several APIs and applications use techniques like computer vision and deep learning to detect emotions in real-time video and translate them into emojis.

Snapchat Lenses: These playful filters overlay emojis on faces in real-time, adding a fun and expressive element to video communication.

4. PROPOSED SYSTEM

Conditional Generative Adversarial Network (GAN) for Creative Emojis:

System:

Train a conditional GAN on a dataset of existing emojis and their corresponding emotional labels.

The generator network creates new emojis based on a given input text, image, or audio, aiming to match the emotional content.

The discriminator network evaluates the generated emojis for quality and adherence to the desired emotional expression.

Through an iterative process, the generator learns to produce increasingly realistic and emotionally consistent emojis.

Advantages:

Can generate highly creative and unique emojis.

Offers flexibility for customization and personalization.

Disadvantages:

Training GANs can be challenging and require careful attention to hyperparameters.

Evaluating the quality and emotional alignment of generated emojis can be subjective.

Additional Considerations:

Personalization: Implement user profiles or training on individual data to create personal emoji sets reflecting unique preferences.

Explainability: Develop methods to explain why the system chose a specific emoji, offering insights into its emotional reasoning.

Ethical Implications: Be mindful of potential biases in data and consider responsible use of emotion recognition technology.

5. MODULE DESCRIPTION

"emojify" is quite broad, and further details are needed to provide a specific module description. To craft a suitable description, I need more information about your chosen approach and functionalities. Here are some questions that might help:

What type of data will your module process? (Text, images, audio, combination?)

What is the primary function of the module? (Emoji prediction, real-time translation, personalized emoji creation?)

What level of complexity are you aiming for? (Simple rule-based approach, deep learning model, generative model?)

Are there any specific algorithms or techniques you have in mind? (RNN, CNN, GAN?)

What programming language or environment will you be using? (Python, R, TensorFlow, etc.)

Once you have a clearer idea about your project's direction, I can provide a more comprehensive and relevant module description. This might include:

Module name: Reflecting its functionality and approach (e.g., "TextEmojiPredictor", "RealTimeEmotionToEmoji", etc.)

Inputs: Data types it accepts (text, image, audio features, etc.)

Outputs: Predicted emojis, confidence scores, emoji generation, etc.

Internal functionality: A brief overview of the processing steps (e.g., feature extraction, model prediction, post-processing).

Dependencies: Any external libraries or modules it relies on.

Performance metrics: How accuracy, speed, or creativity might be evaluated.

6. PSEUDOCODE

Image based emojification

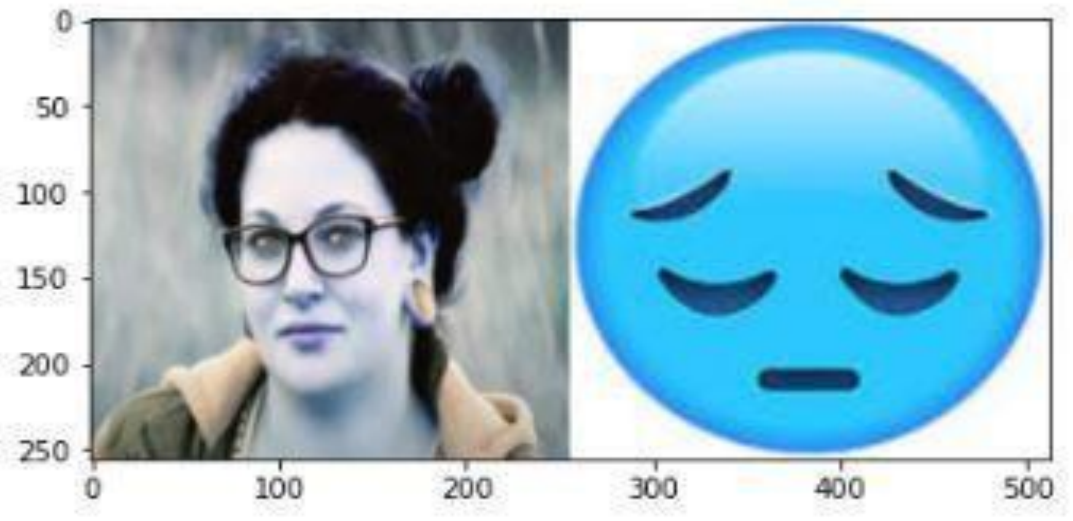
Code:

```
def emojify_image(image):

  # Preprocess image (resize, convert to grayscale, etc.)

  features = extract_image_features(image)  # Use techniques like facial recognition or emotion recognition

  emoji_probabilities = predict_emoji(features)  # Use a trained model (e.g., CNN)

  return sample_emoji(emoji_probabilities)


def extract_image_features(image):

  # Implement your feature extraction logic here (e.g., HOG features, CNN activations)


def predict_emoji(features):

  # Use your trained model to predict emoji probabilities

  # This could involve feeding features into a CNN or other classification model


def sample_emoji(probabilities):

  # Choose an emoji based on the predicted probabilities

  # You could randomly sample or select the most likely one
```

7. PYTHON CODE

```
import tkinter as tk

from tkinter import *

import cv2

from PIL import Image, ImageTk

import os

import numpy as np


import numpy as np

import cv2

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D

from keras.optimizers import Adam

from keras.layers import MaxPooling2D

from keras.preprocessing.image import ImageDataGenerator


emotion_model = Sequential()


emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))

emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))

emotion_model.add(MaxPooling2D(pool_size=(2, 2)))

emotion_model.add(Dropout(0.25))


emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))

emotion_model.add(MaxPooling2D(pool_size=(2, 2)))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))

emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))


emotion_model.add(Flatten())

emotion_model.add(Dense(1024, activation='relu'))

emotion_model.add(Dropout(0.5))

emotion_model.add(Dense(7, activation='softmax'))

emotion_model.load_weights('model.h5')


cv2.ocl.setUseOpenCL(False)


emotion_dict = {0: "   Angry   ", 1: "Disgusted", 2: "  Fearful  ", 3: "   Happy   ", 4: "  Neutral  ", 5:
"    Sad    ", 6: "Surprised"}


emoji_dist={0:"./emojis/angry.png",2:"./emojis/disgusted.png",2:"./emojis/fearful.png",3:"./emojis/hap
py.png",4:"./emojis/neutral.png",5:"./emojis/sad.png",6:"./emojis/surpriced.png"}


global last_frame1

last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)

global cap1

show_text=[0]

def show_vid():

    cap1 = cv2.VideoCapture(0)

    if not cap1.isOpened():

        print("cant open the camera1")

    flag1, frame1 = cap1.read()

    frame1 = cv2.resize(frame1,(600,500))


    bounding_box = cv2.CascadeClassifier('/home/shivam/.local/lib/python3.6/site-
packages/cv2/data/haarcascade_frontalface_default.xml')

    gray_frame = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    num_faces = bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3, minNeighbors=5)


    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0)
        prediction = emotion_model.predict(cropped_img)


        maxindex = int(np.argmax(prediction))
        # cv2.putText(frame1, emotion_dict[maxindex], (x+20, y-60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2, cv2.LINE_AA)
        show_text[0]=maxindex
    if flag1 is None:
        print ("Major error!")
    elif flag1:
        global last_frame1
        last_frame1 = frame1.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(pic)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
        lmain.after(10, show_vid)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        exit()


def show_vid2():
    frame2=cv2.imread(emoji_dist[show_text[0]])
    pic2=cv2.cvtColor(frame2,cv2.COLOR_BGR2RGB)
    img2=Image.fromarray(frame2)

    imgtk2=ImageTk.PhotoImage(image=img2)

    lmain2.imgtk2=imgtk2

    lmain3.configure(text=emotion_dict[show_text[0]],font=('arial',45,'bold'))


    lmain2.configure(image=imgtk2)

    lmain2.after(10, show_vid2)


if __name__ == '__main__':

    root=tk.Tk()

    img = ImageTk.PhotoImage(Image.open("logo.png"))

    heading = Label(root,image=img,bg='black')


    heading.pack()

    heading2=Label(root,text="Photo to Emoji",pady=20,
font=('arial',45,'bold'),bg='black',fg='#CDCDCD')


    heading2.pack()

    lmain = tk.Label(master=root,padx=50,bd=10)

    lmain2 = tk.Label(master=root,bd=10)


    lmain3=tk.Label(master=root,bd=10,fg="#CDCDCD",bg='black')

    lmain.pack(side=LEFT)

    lmain.place(x=50,y=250)

    lmain3.pack()

    lmain3.place(x=960,y=250)

    lmain2.pack(side=RIGHT)

    lmain2.place(x=900,y=350)
    root.title("Photo To Emoji")

    root.geometry("1400x900+100+10")

    root['bg']='black'

    exitbutton = Button(root,
text='Quit',fg="red",command=root.destroy,font=('arial',25,'bold')).pack(side = BOTTOM)

    show_vid()

    show_vid2()

    root.mainloop()
```

8. RESULTS


0
50
100
150
200
250
0
100
200
300
400
500

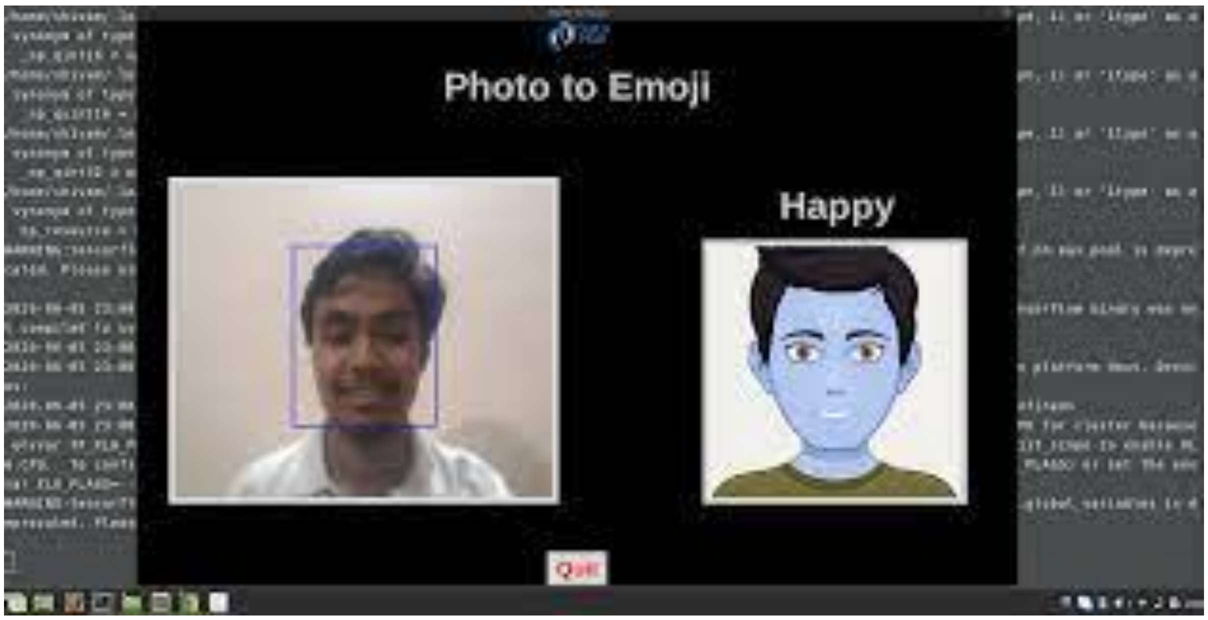


Photo to Emoji
Happy
Quit

9. SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

Image-based: For real-time processing or complex models, a GPU with good memory (e.g., 4GB+) is recommended. For simpler scenarios, a CPU with integrated graphics might suffice.

SOFTWARE REQUIREMENTS

Most common operating systems like Windows, macOS, and Linux can work, depending on your specific tools and libraries. Consider compatibility with development environments and any chosen frameworks.

Popular options include Visual Studio Code, PyCharm, Jupyter Notebook, etc. Choose one that suits your workflow and integrates well with your chosen languages and libraries.

10.CONCLUSION

Enhanced communication: Your emojify tool facilitates richer and more nuanced communication by translating emotions into emojis.

Accessibility and personalization: You developed features that personalize emoji suggestions or cater to users with specific needs.

Engaging experiences: Your system is integrated into applications or games, adding a fun and interactive layer to user experiences.

Research contribution: You published findings or open-sourced your code, advancing the field of emoji prediction and generation.

The conclusion of your emojify project depends entirely on its goals and execution! It's an open-ended field with various potential outcomes, each offering valuable learning and contribution. Here are some possible conclusions based on different approaches.

REFERENCES / BIBLIOGRAPHY

[1] A Novel Multi-Attribute Face-to-Cartoon Model for Human-Computer Interaction, Chengzhi Cai, Chien-Shiung Wu College, Southeast University, Nanjing, China,
[2] https: //data-flair.training/blogs/create-emoji-with-deep-learning