VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme **: B-Tech CSE AI&ML**

Course Title : Foundations of Artificial Intelligence

Course Code **: CSE1014**

Slot **: G1**

Title: Cartoozi – Cartoonifying your Meets

Team Members:

Sarvesh Chandak | 20BAI1221

Devanshu Khemka | 20BAI1319

Hardik Kathuria | 20BAI1048

Faculty: Dr. Suganya Karunamurty

Sign:

Date:

DECLARATION

We hereby declare that the project entitled "**CARTOJI (CARTOONIFY AND EMOTIFY YOUR PHOTOS USING FACIAL RECOGNITION)**" submitted to Vellore Institute of Technology (VIT) for the project is a record of bonafide work carried out by me under the guidance of Dr.SuganyaKarunamurthy, Assistant Professor (Senior), School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

We further declare that the work reported in this report has not been submitted and will not be submitted in full, for the award of any other degree or diploma in this institute or any other institute or university.

TABLE OF CONTENTS

ABSTRACT

In the recent times, due to the pandemic the use of online meetings and classes have increased drastically. But due to less activeness and less interaction of the participants, it becomes really boring. Now, one to make meetings funny we have come up with this project. We can your mood based upon your facial expressions. We also show an emoji based upon your mood! It makes virtual interactions more fun and interactive. To accomplish this task, we will use CNN networks. CNN stands for convolutional neural network. CNN most commonly is applied to analyze visual imagery.

INTRODUCTION

Our project is classified into two parts:

1) Emojify: In this part we will take in your face as inputthrough your camera. Then we will analyse the photo usingCNN and classify it according to the dataset chosen.The architecture of a Convolutional Network is analogous tothat of the connectivity pattern of Neurons in the Human Brainand was inspired by the organization of the Visual Cortex.Individual neurons respond to stimuli only in a restrictedregion of the visual field known as the Receptive Field. Acollection of such fields overlaps to cover the entire visual area.The role of the Convolutional Network is to reduce the imagesinto a form which is easier to process, without losing featureswhich are critical for getting a good prediction.

2) Cartoonify:

For this part, we use OpenCv library. OpenCV is a great toolfor image processing and performing computer visiontasks. It is an open-source library that can be used toperform tasks like face detection, objection tracking,landmark detection, and much more.In this part we take an image from the user’s hard disk andprocess the image to form a cartoon like image and the usercan save into the pc.

BACKGROUND STUDY

1) Emoji Classification and prediction in Hebrew political corpus

This project was done by Chaya Liebeskind from Jerusalem College of Technology. This project aims in efficient prediction of emoji based on the kind of text for better NLP tasks.

For this they explored two tasks: emoji identification and emoji prediction. Emoji prediction entailed classification task of predicting Emoji based on text and emoji identification was the complementary preceding task of determining if given text message includes emojis. For this they used a supervised Machine Learning approach (logistic regression) for the task. They compare two text representation approaches, i.e., n-grams and character n-grams and analyse the contribution of additional metadata features to the classification.

The findings they got were metadata improve the classification accuracy in the task of emoji identification and in the task of emoji prediction it is better to apply feature selection.

2) Technical Paper Presentation on application of Cartoon like Effects to Actual Images

This paper illustrates various techniques for converting images to cartoons. After briefly talking about the existing techniques to apply cartoon like effects on actual images. This paper goes on proposing a new technique in which one can choose two images. First is of the image on which we want to apply effects and the second of the image which contains the style that we want to apply on our first image. This project used feed forward neural network for doing the same.

3) #TeamINF at SemEval-2018 Task 2: Emoji Prediction in Tweets

   This paper describes a methodology for predicting emoji in tweets. The method is based on the traditional bag-of-words model combined with word embedding's. Logistic Regression was the classification algorithm used. This architecture was implemented and tested in the SemEval 2018 challenge.The results from the testing data show that word embedding combined with bag-of-word produces the best F1, whereas the three configurations represented only by bag-of-word produced results that were close to the central work model (Word Embedding + Bag-of-Words).

4) Transformation of Realistic Images and Videos into Cartoon Images and Video using GAN

   The project's goal is to present a solution for converting real-world snapshots or videos into animated photos (Cartoon Images) or video. The previous method of transformation necessitates complex computer graphics and skills. The concept of the paper is based on specific snapshots and videos that are converted to an art form such as painting. Among the techniques available, the application of a Generative Adversarial Network (GAN) called Cartoon GAN was used for the styling of real-world images that uses two loss functions, namely, content loss and adversarial loss, to obtain a sharp and clear image.

5) Image Cartoonization Methods Using LBG, KPE, KMCG quantization

This paper discusses various methods for creating a cartoonized painterly effect on grayscale and coloured images. To achieve a painterly effect on images, the concept of vector quantization is used. To achieve cartoonized painterly results, the algorithms LBG, KPE, and KMCG are used. The results of applying each algorithm to images are compared based on the time required and the effect produced. The outcomes of the discussed methods can be incorporated into a variety of applications used for movie to comic conversions and digital art software.

6) Using Neural Networks to Predict Emoji Usage from Twitter Data

This paper believes as emojis have become a more important and standardised part of modern textual inputs, analysing their usage patterns has becoming increasingly important to any modern text system. This paper has framed this study as a text classification problem, mapping input text to the most likely accompanying emoji, with 1.5 million scraped Twitter tweets serving as the training set. Both their LSTM-RNN model and their CNN model outperformed their baseline, with the CNN model achieving significantly higher accuracy and F1 results.

PROPOSED METHODOLOGY

This is the methodology we used for mapping facial emotions to emotes

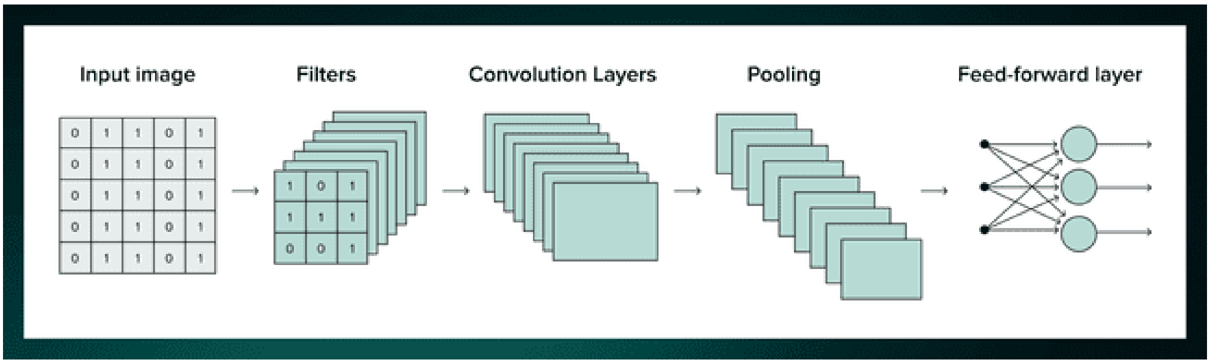
CNN-Convolutional Neural Networks are specialised for image and video recognition applications. CNN is most commonly used for image analysis tasks such as image recognition, object detection, and segmentation.

Convolutional Neural Networks have three types of layers:

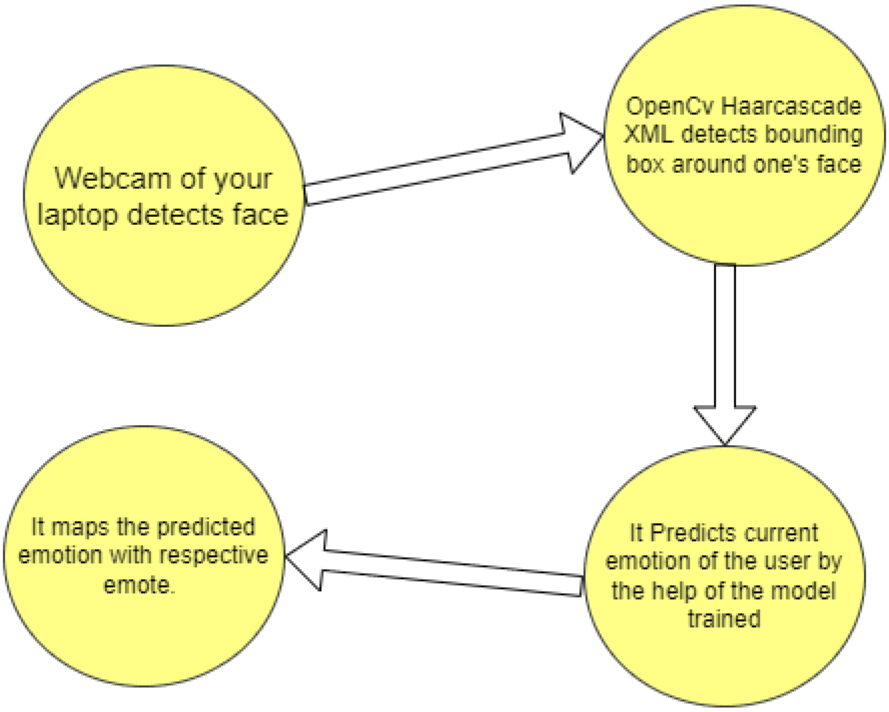
1) Convolutional Layer: Each input neuron in a typical neural network is connected to the next hidden layer. Only a small portion of the neurons in the input layer connect to the neurons in the hidden layer in CNN.

2) Pooling Layer: The pooling layer is used to reduce the feature map's dimensionality. Inside the CNN's hidden layer, there will be multiple activation and pooling layers.

3) Fully-Connected Layer: Fully Connected Layers are the network's final few layers. The output of the final Pooling or Convolutional Layer is flattened and fed into the fully connected layer as the input to the fully connected layer.



Architecture of Convolutional Neural Network


Webcam of your laptop detects face
OpenCv Haarcascade XML detects bounding box around one's face
It Predicts current emotion of the user by the help of the model trained
It maps the predicted emotion with respective emote.

Simple Architecture Diagram of Emojify Module

We used OpenCV for cartoonification of images.

OpenCV (Open Source Computer Vision Library) is a programming function library aimed primarily at real-time computer vision. It was created by Intel and was later supported by Willow Garage and Itseez. The library is cross-platform and available for free under the Apache 2 License.

OpenCVincludes applications for capturing and processing video and images. It is widely used in image transformation, object detection, face recognition, and a variety of other fascinating applications.

IMPLEMENTATION

1)**Emojify – Create your own emoji with CNN:**

In Emojify, we take the input of the user through their webcam and return the emoji according to their input reaction.

For this particular part of our project, we will use CNN (Convolutional Neural Network). We will build and train a CNN model on FER2013 dataset CNN is a class

of deep neural networks. CNN is usually used in processing data which has a grid-like topology, such as an image, which is the basic requirement in this part of our project.

Step1:Import Libraries:

The most important library used for this part is CV2 in python. OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.

After that we use keras library which will used to build a fully connected neural network.

Step 2:

We now go on to build a model and we will train the model. In this step we build a Convolutional Network Architecture We use relu activation function at every layer of the neural network.

After that, we combine the perceptron layers that we built and train the model.

After training the model, we save the model weights for the best possible output.

Step 3:We use the ‘haarcascade’ xml function to find the boundaries of the face when the webcam is turned on. The image of the user’s face is then passed on to the CNN model which we built and will predict the emotions of the user.

Step 4:Our next step is to map the emotions and return an emoji according to theoutput of the CNN model and to build a GUI for a good presentation.There are seven possible emotions in the dataset, so we have to map them toseven different emojis.For this purpose, we can use tkinter library. Tkinter is the standard GUI libraryfor Python. Python when combined with Tkinter provides a fast and easy wayto create GUI applications. Tkinter provides a powerful object-orientedinterface to the Tk GUI toolkit. Using tkinter we can easily create and a GUI for our project using python.

2) Cartooning an image using OpenCV and python:

We will use Open cv for converting the input image to a simple cartoonimage. OpenCV is a cross-platform library used for Computer Vision. Itincludes applications like video and image capturing and processing. It ismajorly used in image transformation, object detection, face recognition, andmany other stunning applications.

Step 1:Importing necessary libraries,

• CV2: Imported to use OpenCV for image processing

• Easygui: Imported to open a file box. It allows us to select any filefrom our system.

• Numpy: Images are stored and processed as numbers. These aretaken as arrays. We use NumPy to deal with arrays.

- Imageio: Used to read the file which is chosen by file box using apath.

- Matplotlib: This library is used for visualization and plotting.Thus, it is imported to form the plot of images.

- OS: For OS interaction. Here, to read the path and save images tothat path.

Step 2:Building a File Box to choose a particular file. In this step, we will build the main window of our application, where thebuttons, labels, and images will reside. We also give it a title by title() function.In this step, we will build a method which will a pop-up box to choose animage from your hard-disk, which will open every time you run your code.fileopenbox () is the method in easyGUI module which returns the path of thechosen file as a string.

Step 3:In this step, we will store the image as Numpy array.Imread is a method in cv2 which is used to store images in the form ofnumbers. This helps us to perform operations according to our needs. Theimage is read as a Numpy array, in which cell values depict R, G, and B valuesof a pixel.Now comes the main part, to convert the image to a cartoon.To convert an image to a cartoon, multiple transformations are done. Firstly,an image is converted to a Grayscale image. Yes, similar to the old day'spictures.! Then, the Grayscale image is smoothened, and we try to extract theedges in the image. Finally, we form a colour image and mask it with edges.This creates a beautiful cartoon image with edges and lightened colour of theoriginal image.To accomplish the task, we have to follow six steps to get the cartoon image,

- Convert the image to grayscale.

    o cvtColor (image, flag) is a method in cv2 which is used totransform an image into the colour-space mentioned as 'flag'.

- Smoothening the grayscale image.

o To smoothen an image, we simply apply a blur effect. Thisis done using medianBlur () function. Here, the centre pixelis assigned a mean value of all the pixels which fall underthe kernel. In turn, creating a blur effect.

• Retrieving the edges of an image.

o In this step, we will work on the first specialty. Here, wewill try to retrieve the edges and highlight them. This isattained by the adaptive thresholding technique. Thethreshold value is the mean of the neighbourhood pixelvalues area minus the constant C. C is a constant that issubtracted from the mean or weighted sum of theneighborhood pixels.

• Preparing a mask image

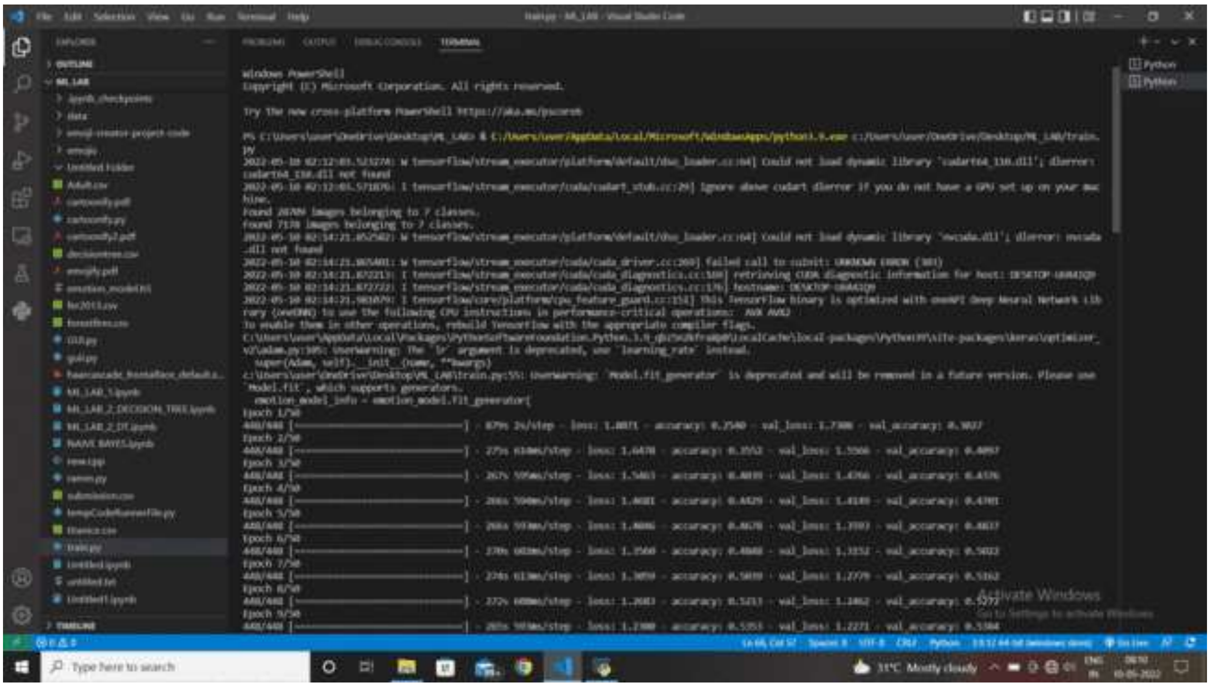
o So, let’s combine the two specialties. This will be done usingMASKING. We perform bitwise and on two images to maskthem.

• And finally, giving a cartoon image.

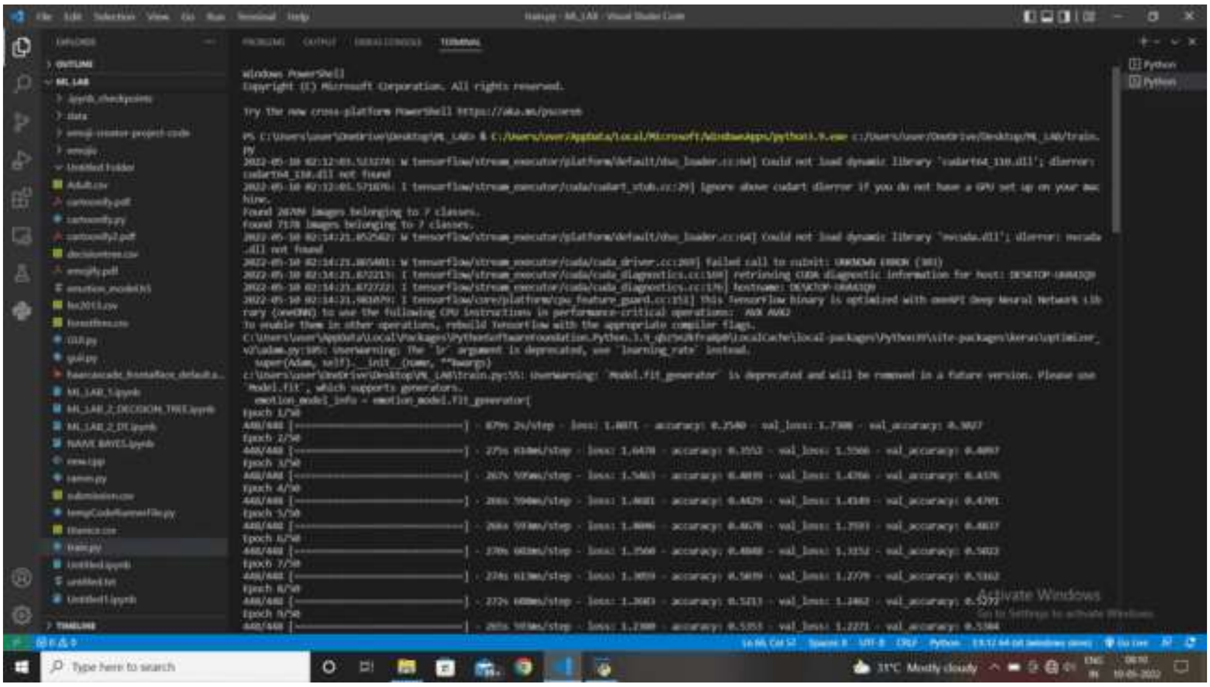
After completing the above two components of our project, we combine theabove two components of our project using tkinter. We create a GUI todynamically run the above two parts in a single window on click of a singlebutton.

RESULTS AND DISCUSSION

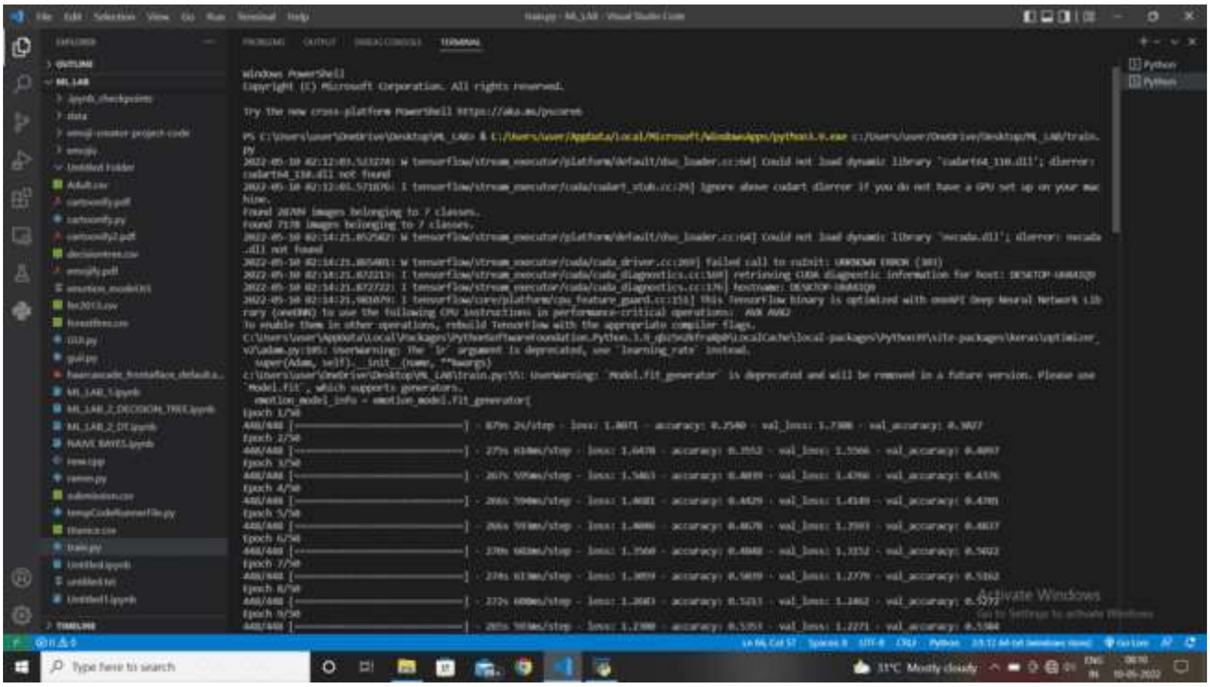
Below are some screenshots of the accuracy we got after training our model on FER2013 dataset for 50 epochs.



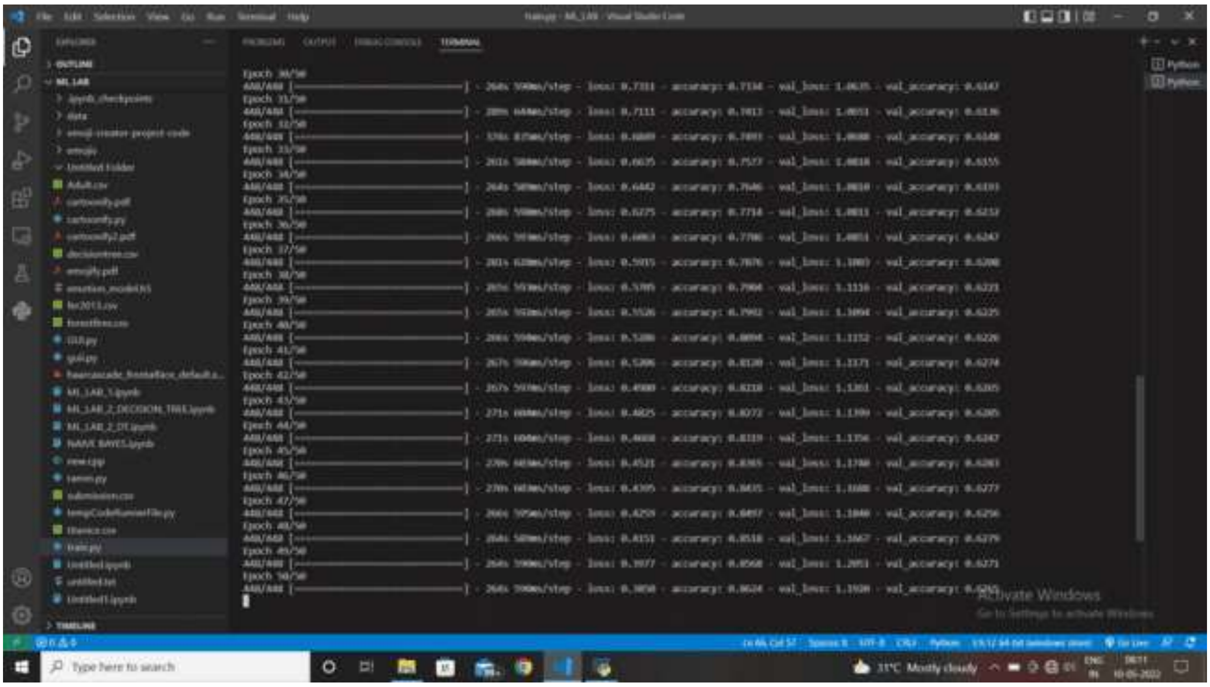
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\user\OneDrive\Desktop\ML_LAB> & C:/Users/user/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/Users/user/OneDrive/Desktop/ML_LAB/train.
py
2022-05-10 02:12:03.523274: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-05-10 02:12:03.571076: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
2022-05-10 02:14:21.852582: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda
.dll not found
2022-05-10 02:14:21.865401: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-10 02:14:21.872213: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-U04AIQ9
2022-05-10 02:14:21.872722: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-U04AIQ9
2022-05-10 02:14:21.983079: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\user\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\keras\optimizer_
v2\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
c:\Users\user\OneDrive\Desktop\ML_LAB\train.py:55: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
 `Model.fit`, which supports generators.
  emotion_model_info = emotion_model.fit_generator(
Epoch 1/50
448/448 [==============================] - 879s 2s/step - loss: 1.8071 - accuracy: 0.2540 - val_loss: 1.7308 - val_accuracy: 0.3027
Epoch 2/50
448/448 [==============================] - 275s 614ms/step - loss: 1.6470 - accuracy: 0.3552 - val_loss: 1.5506 - val_accuracy: 0.4097
Epoch 3/50
448/448 [==============================] - 267s 595ms/step - loss: 1.5463 - accuracy: 0.4039 - val_loss: 1.4766 - val_accuracy: 0.4376
Epoch 4/50
448/448 [==============================] - 266s 594ms/step - loss: 1.4681 - accuracy: 0.4429 - val_loss: 1.4149 - val_accuracy: 0.4701
Epoch 5/50
448/448 [==============================] - 266s 593ms/step - loss: 1.4086 - accuracy: 0.4678 - val_loss: 1.3593 - val_accuracy: 0.4837
Epoch 6/50
448/448 [==============================] - 270s 603ms/step - loss: 1.3560 - accuracy: 0.4860 - val_loss: 1.3152 - val_accuracy: 0.5022
Epoch 7/50
448/448 [==============================] - 274s 613ms/step - loss: 1.3059 - accuracy: 0.5039 - val_loss: 1.2779 - val_accuracy: 0.5162
Epoch 8/50
448/448 [==============================] - 272s 608ms/step - loss: 1.2683 - accuracy: 0.5213 - val_loss: 1.2462 - val_accuracy: 0.5272
Epoch 9/50
448/448 [==============================] - 265s 593ms/step - loss: 1.2300 - accuracy: 0.5353 - val_loss: 1.2271 - val_accuracy: 0.5384
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\user\OneDrive\Desktop\ML_LAB> & C:/Users/user/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/Users/user/OneDrive/Desktop/ML_LAB/train.
py
2022-05-10 02:12:03.523274: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-05-10 02:12:03.571076: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
2022-05-10 02:14:21.852582: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda
.dll not found
2022-05-10 02:14:21.865401: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-10 02:14:21.872213: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-U04AIQ9
2022-05-10 02:14:21.872722: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-U04AIQ9
2022-05-10 02:14:21.983079: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\user\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\keras\optimizer_
v2\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
c:\Users\user\OneDrive\Desktop\ML_LAB\train.py:55: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
 `Model.fit`, which supports generators.
  emotion_model_info = emotion_model.fit_generator(
Epoch 1/50
448/448 [==============================] - 879s 2s/step - loss: 1.8071 - accuracy: 0.2540 - val_loss: 1.7308 - val_accuracy: 0.3027
Epoch 2/50
448/448 [==============================] - 275s 614ms/step - loss: 1.6470 - accuracy: 0.3552 - val_loss: 1.5506 - val_accuracy: 0.4097
Epoch 3/50
448/448 [==============================] - 267s 595ms/step - loss: 1.5463 - accuracy: 0.4039 - val_loss: 1.4766 - val_accuracy: 0.4376
Epoch 4/50
448/448 [==============================] - 266s 594ms/step - loss: 1.4681 - accuracy: 0.4429 - val_loss: 1.4149 - val_accuracy: 0.4701
Epoch 5/50
448/448 [==============================] - 266s 593ms/step - loss: 1.4086 - accuracy: 0.4678 - val_loss: 1.3593 - val_accuracy: 0.4837
Epoch 6/50
448/448 [==============================] - 270s 603ms/step - loss: 1.3560 - accuracy: 0.4860 - val_loss: 1.3152 - val_accuracy: 0.5022
Epoch 7/50
448/448 [==============================] - 274s 613ms/step - loss: 1.3059 - accuracy: 0.5039 - val_loss: 1.2779 - val_accuracy: 0.5162
Epoch 8/50
448/448 [==============================] - 272s 608ms/step - loss: 1.2683 - accuracy: 0.5213 - val_loss: 1.2462 - val_accuracy: 0.5272
Epoch 9/50
448/448 [==============================] - 265s 593ms/step - loss: 1.2300 - accuracy: 0.5353 - val_loss: 1.2271 - val_accuracy: 0.5384
```

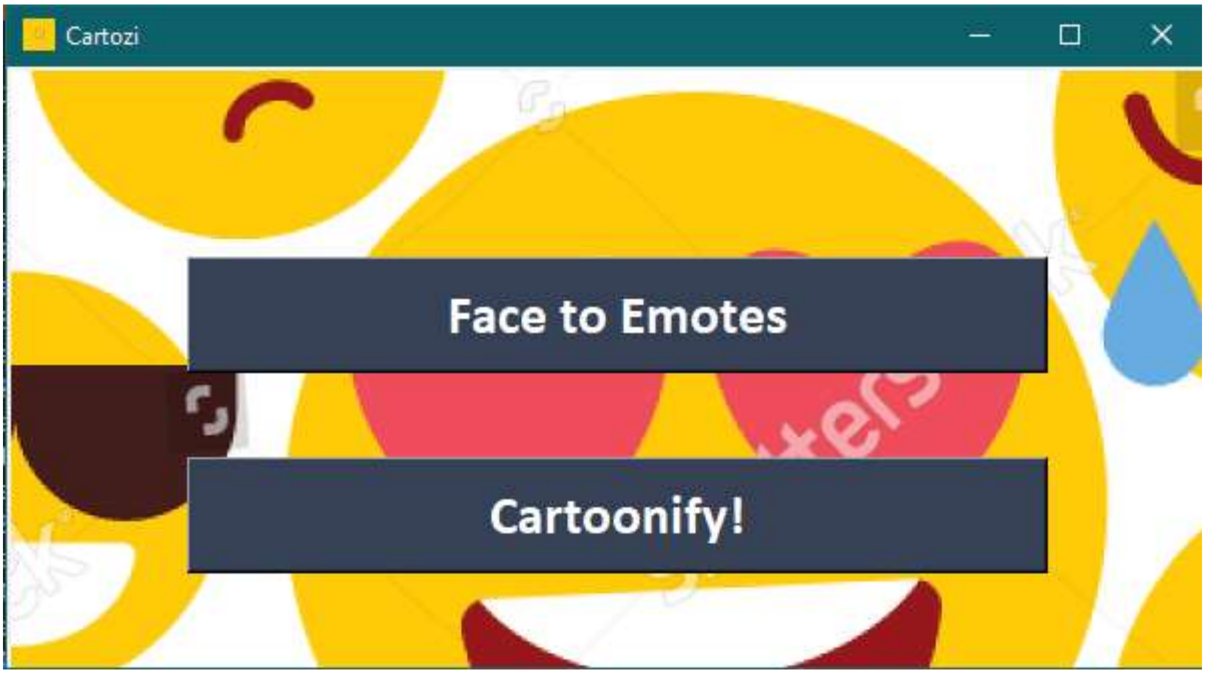


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\user\OneDrive\Desktop\ML_LAB> & C:/Users/user/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/Users/user/OneDrive/Desktop/ML_LAB/train.
py
2022-05-10 02:12:03.513270: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror:
cudart64_110.dll not found
2022-05-10 02:12:03.571876: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your mac
hine.
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
2022-05-10 02:14:21.852502: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda
.dll not found
2022-05-10 02:14:21.865401: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-10 02:14:21.872213: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-U8A8IQ9
2022-05-10 02:14:21.872722: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-U8A8IQ9
2022-05-10 02:14:21.903079: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\user\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\keras\optimizer_
v2\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
c:\Users\user\OneDrive\Desktop\ML_LAB\train.py:55: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  emotion_model_info = emotion_model.fit_generator(
Epoch 1/50
448/448 [==============================] - 879s 2s/step - loss: 1.8071 - accuracy: 0.2540 - val_loss: 1.7308 - val_accuracy: 0.3027
Epoch 2/50
448/448 [==============================] - 275s 614ms/step - loss: 1.6478 - accuracy: 0.3552 - val_loss: 1.5566 - val_accuracy: 0.4097
Epoch 3/50
448/448 [==============================] - 267s 595ms/step - loss: 1.5463 - accuracy: 0.4039 - val_loss: 1.4766 - val_accuracy: 0.4376
Epoch 4/50
448/448 [==============================] - 266s 594ms/step - loss: 1.4681 - accuracy: 0.4429 - val_loss: 1.4149 - val_accuracy: 0.4705
Epoch 5/50
448/448 [==============================] - 266s 593ms/step - loss: 1.4006 - accuracy: 0.4678 - val_loss: 1.3593 - val_accuracy: 0.4837
Epoch 6/50
448/448 [==============================] - 270s 602ms/step - loss: 1.3560 - accuracy: 0.4848 - val_loss: 1.3152 - val_accuracy: 0.5022
Epoch 7/50
448/448 [==============================] - 274s 613ms/step - loss: 1.3059 - accuracy: 0.5039 - val_loss: 1.2779 - val_accuracy: 0.5162
Epoch 8/50
448/448 [==============================] - 272s 608ms/step - loss: 1.2603 - accuracy: 0.5213 - val_loss: 1.2462 - val_accuracy: 0.5272
Epoch 9/50
448/448 [==============================] - 265s 593ms/step - loss: 1.2300 - accuracy: 0.5353 - val_loss: 1.2271 - val_accuracy: 0.5384
```



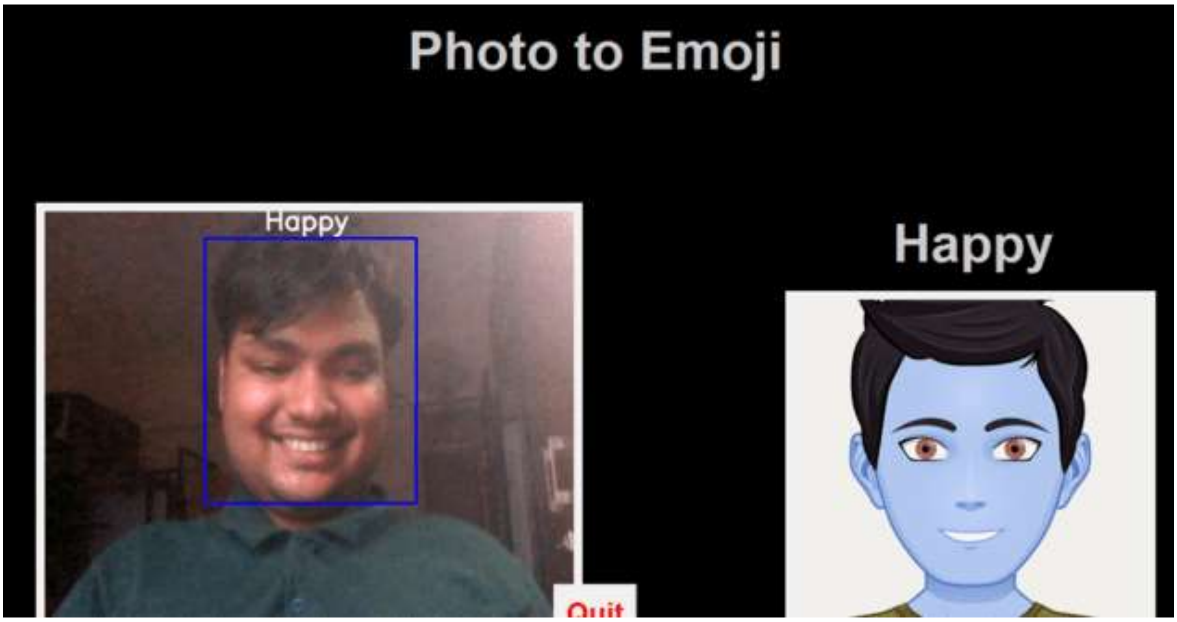
```
Epoch 30/50
448/448 [==============================] - 264s 590ms/step - loss: 0.7311 - accuracy: 0.7134 - val_loss: 1.0635 - val_accuracy: 0.6147
Epoch 31/50
448/448 [==============================] - 289s 646ms/step - loss: 0.7111 - accuracy: 0.7013 - val_loss: 1.0551 - val_accuracy: 0.6136
Epoch 32/50
448/448 [==============================] - 376s 839ms/step - loss: 0.6889 - accuracy: 0.7093 - val_loss: 1.0688 - val_accuracy: 0.6148
Epoch 33/50
448/448 [==============================] - 281s 588ms/step - loss: 0.6635 - accuracy: 0.7577 - val_loss: 1.0818 - val_accuracy: 0.6155
Epoch 34/50
448/448 [==============================] - 264s 589ms/step - loss: 0.6442 - accuracy: 0.7646 - val_loss: 1.0810 - val_accuracy: 0.6193
Epoch 35/50
448/448 [==============================] - 268s 598ms/step - loss: 0.6275 - accuracy: 0.7714 - val_loss: 1.0833 - val_accuracy: 0.6232
Epoch 36/50
448/448 [==============================] - 266s 593ms/step - loss: 0.6063 - accuracy: 0.7786 - val_loss: 1.0851 - val_accuracy: 0.6247
Epoch 37/50
448/448 [==============================] - 281s 628ms/step - loss: 0.5915 - accuracy: 0.7876 - val_loss: 1.1003 - val_accuracy: 0.6208
Epoch 38/50
448/448 [==============================] - 265s 593ms/step - loss: 0.5705 - accuracy: 0.7904 - val_loss: 1.1116 - val_accuracy: 0.6223
Epoch 39/50
448/448 [==============================] - 265s 592ms/step - loss: 0.5526 - accuracy: 0.7992 - val_loss: 1.1094 - val_accuracy: 0.6225
Epoch 40/50
448/448 [==============================] - 266s 594ms/step - loss: 0.5288 - accuracy: 0.8094 - val_loss: 1.1152 - val_accuracy: 0.6226
Epoch 41/50
448/448 [==============================] - 267s 596ms/step - loss: 0.5206 - accuracy: 0.8120 - val_loss: 1.1171 - val_accuracy: 0.6274
Epoch 42/50
448/448 [==============================] - 267s 597ms/step - loss: 0.4988 - accuracy: 0.8218 - val_loss: 1.1203 - val_accuracy: 0.6205
Epoch 43/50
448/448 [==============================] - 271s 604ms/step - loss: 0.4825 - accuracy: 0.8272 - val_loss: 1.1399 - val_accuracy: 0.6205
Epoch 44/50
448/448 [==============================] - 271s 604ms/step - loss: 0.4688 - accuracy: 0.8319 - val_loss: 1.1356 - val_accuracy: 0.6247
Epoch 45/50
448/448 [==============================] - 270s 603ms/step - loss: 0.4521 - accuracy: 0.8365 - val_loss: 1.1700 - val_accuracy: 0.6283
Epoch 46/50
448/448 [==============================] - 270s 603ms/step - loss: 0.4395 - accuracy: 0.8435 - val_loss: 1.1888 - val_accuracy: 0.6277
Epoch 47/50
448/448 [==============================] - 266s 595ms/step - loss: 0.4259 - accuracy: 0.8497 - val_loss: 1.1840 - val_accuracy: 0.6256
Epoch 48/50
448/448 [==============================] - 264s 589ms/step - loss: 0.4151 - accuracy: 0.8518 - val_loss: 1.1667 - val_accuracy: 0.6279
Epoch 49/50
448/448 [==============================] - 264s 590ms/step - loss: 0.3977 - accuracy: 0.8568 - val_loss: 1.2053 - val_accuracy: 0.6271
Epoch 50/50
448/448 [==============================] - 264s 590ms/step - loss: 0.3858 - accuracy: 0.8624 - val_loss: 1.1930 - val_accuracy: 0.6265
```

Final Accuracy we got after 50 epochs was 86.24%



This is the user interface of Cartoji.

Some of the facial expressions detected by Cartozi



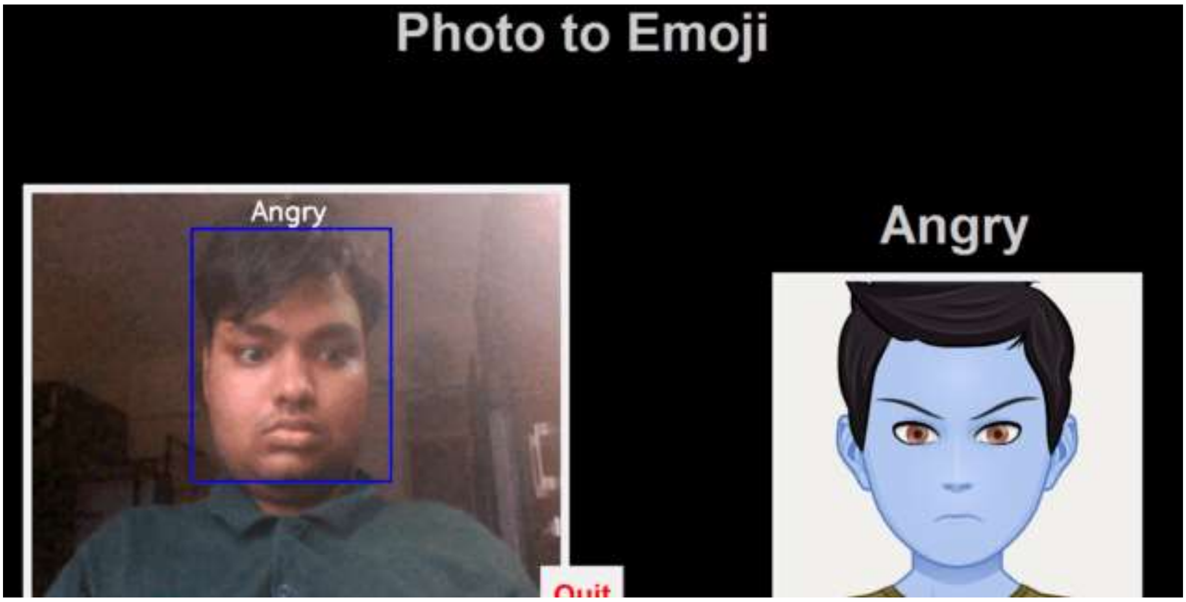


Photo to Emoji
Angry
Angry
Quit

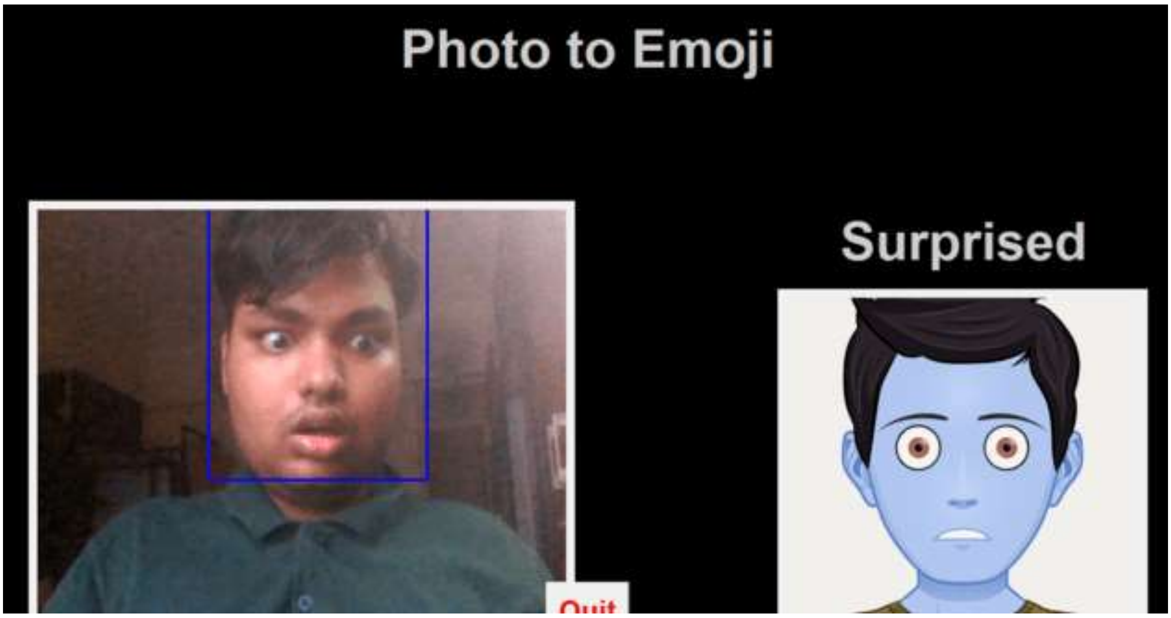


Photo to Emoji
Surprised
Quit

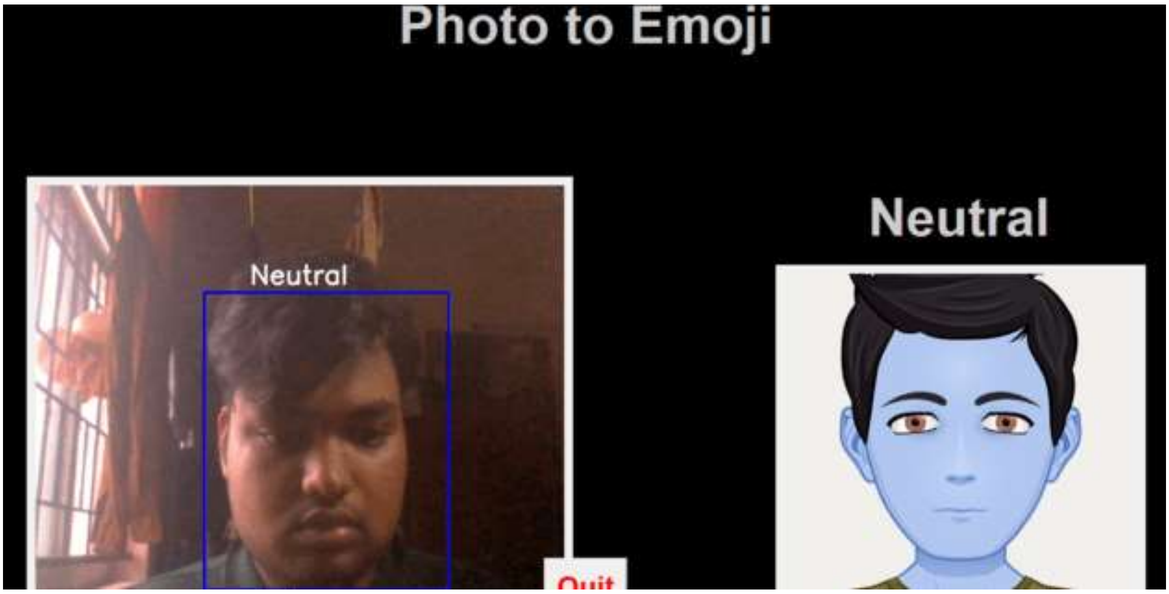
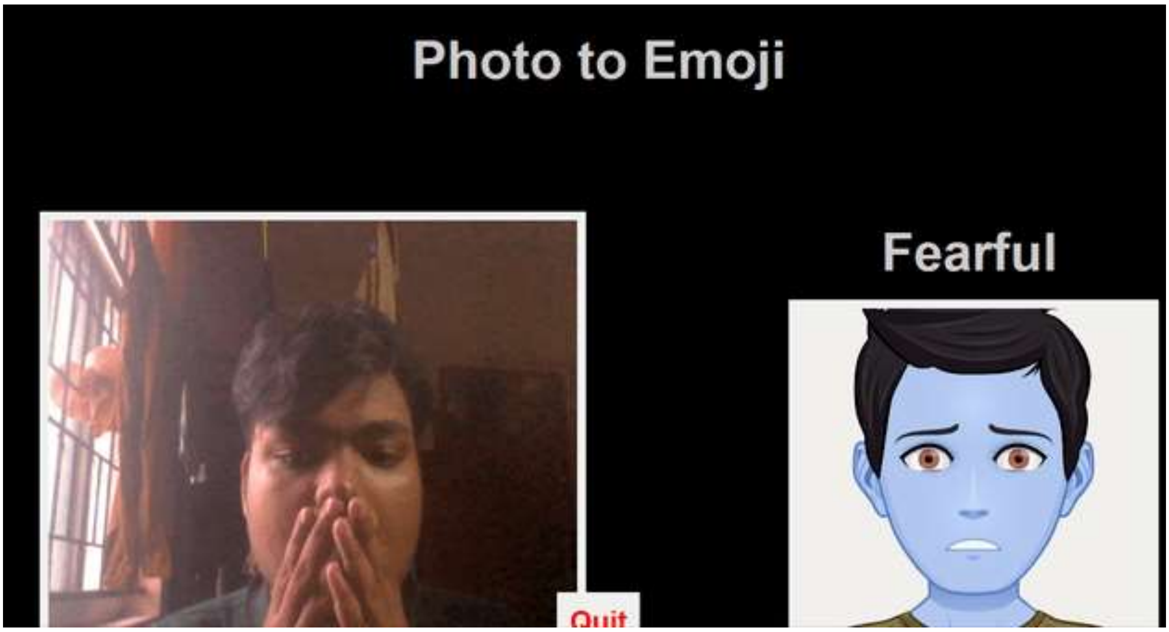
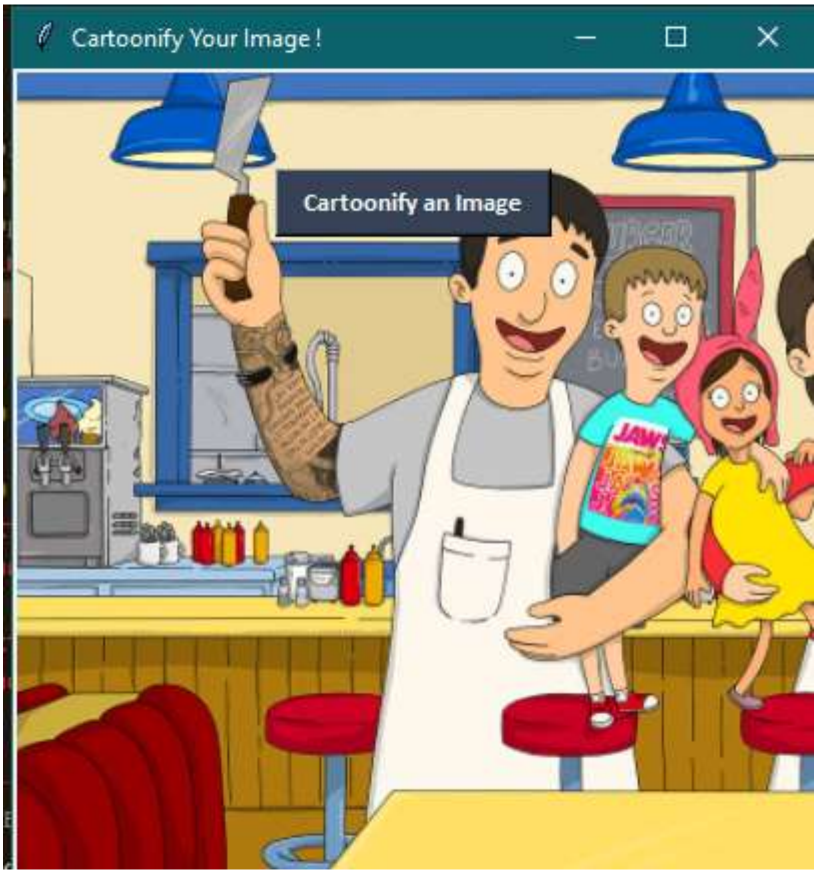
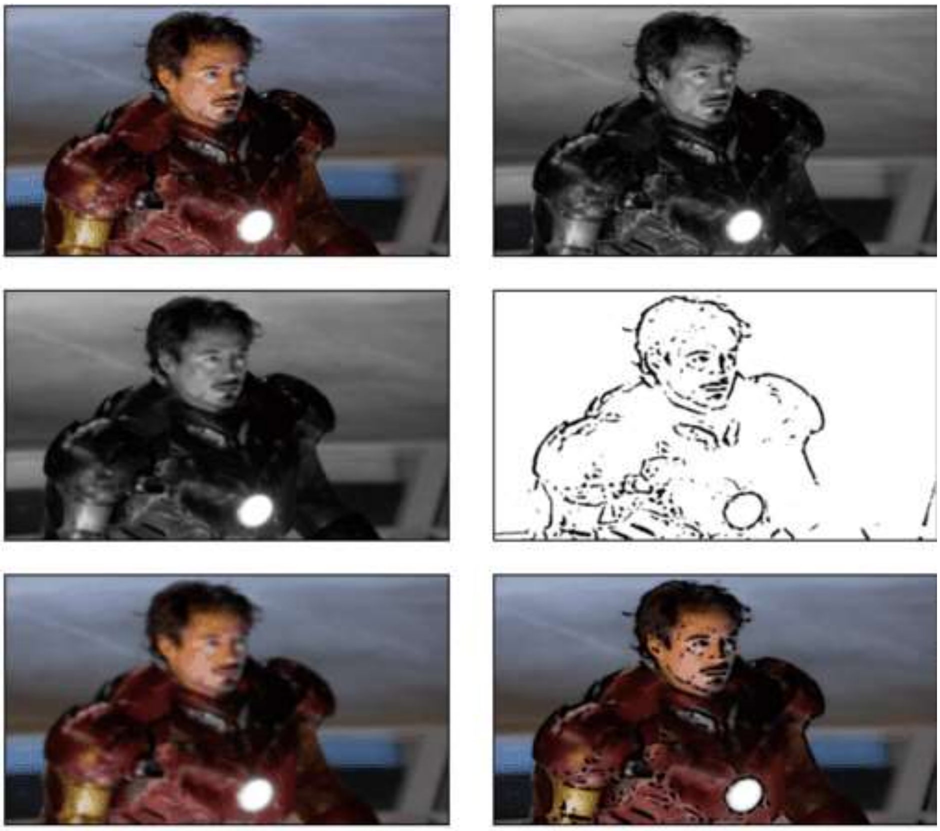


Photo to Emoji
Neutral
Neutral
Quit



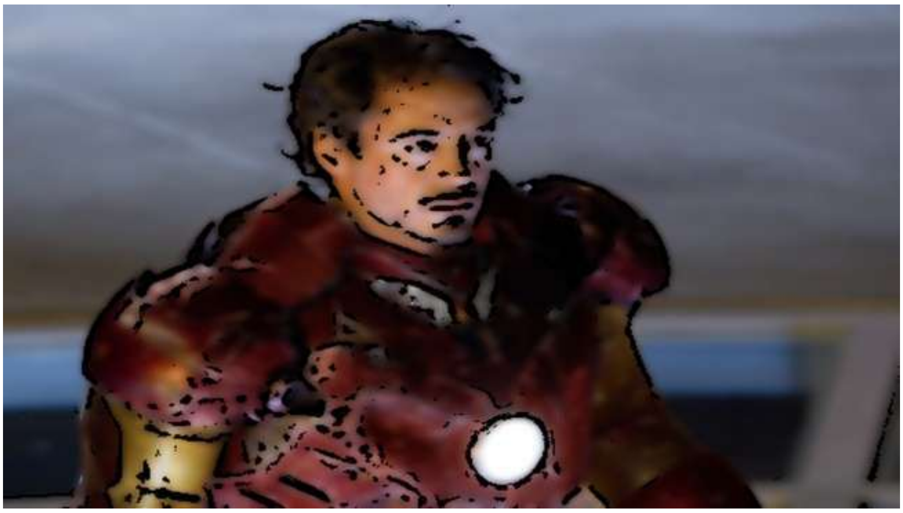


Cartoonify’sgui



Stages depicting cartoonification

Before Cartoonification



After Cartoonification

CONCLUSION AND FUTURE SCOPE

Conclusion

In this project with the help of CNN (Convolutional Neural Network), we were successful in identifying real time emotions of people from the video. Using OpenCV which is also known as Computer vision, we captured and processed images of people and converted it into simple cartoon images.

Future scope

The project showed that image was successfully converted into a cartoon-style image and emotions of the live video were accurately detected. In future, we would like to focus more on generating a portrait defined HD image even though we used the loss function. We also plan to improve the accuracy of our emojifying model, which is at present around 86%. The model should be lightweight, so that it could be easily integrated with virtual meeting platforms and detect real-time emotions seamlessly.

Project Link:

https://drive.google.com/drive/folders/1UbUyjdI24cFQjLUeunmdoMcO3qKOrl3D?usp=sharing

REFERENCES

1. Liebeskind, Chaya. (2019). Emoji Identification and Prediction in Hebrew Political Corpus. Issues in Informing Science and Information Technology. 16. 343-359. 10.28945/4372.

2. Technical Paper Presentation on Application of Cartoon like Effects to Actual Images Chinmay Joshi, Devendra Jaiswal, AkshataPatil Department Name of Organization: Information Technology Name of Organization: Kc College of Engineering Management Studies and Research.

3. Félix, Nádia& Ribeiro, Alison. (2018). #TeamINF at SemEval-2018 Task 2: Emoji Prediction in Tweets.

4. Transformation of Realistic Images and Videos into Cartoon Imagesand Video using GANAkankshaApte, AshwathyUnnikrishnan, AvjeevanBomble, Prof. Sachin Gavhane

5. Patankar, Archana &Kubde, Purnima &Karia, Ankita. (2016). Image cartoonization methods. 1-7. 10.1109/ICCUBEA.2016.7860045.

6. Zhao, L., & Zeng, C. (2017). Using Neural Networks to Predict Emoji Usage from Twitter Data.

7. OpenCV Docs

8. TensorFlow Docs

9. Tkinter Docs