



REPORT

Course Code : CSE316
Course Title : Artificial Intelligence Lab
Project name : Face Emotion Detection

Submitted To

Md. Arid Hasan (MAHN)
Lecturer,
Department of CSE
Daffodil International University

Submitted By

MD. Yeamin Islam
ID: 193-15-13529
Tuhin Sarker
ID: 193-15-13388
Shohidul Islam
ID: 193-15-13481
Shift: Day
Sec : N
Department of CSE,
Daffodil International
University.

Introduction to Convolution Neural Network (CNN) and OpenCV in Emotion Recognition

we have created a Deep Learning Model which is capable of recognizing human emotions through rigorous training using superior quality datasets.

We will build a Convolution Neural Network (CNN) architecture and train the model on **FER2013** dataset for Emotion recognition from images.

DATASET:

This model is capable of recognizing seven basic emotions as following:

- Happy
- Sad
- Angry
- Surprise
- Disgust
- Fear
- Neutral

The FER-2013 dataset consists of 28,709 labeled images in the training set and 7,178 labeled images in the test set. Each image in this dataset is labeled as one of seven emotions: happy, sad, angry, afraid, surprise, disgust, and neutral. The faces have been automatically registered such that the face is more or less centered and occupies about the same amount of space in each image.

In this project, we will be using Deep Learning and Computer Vision.

DEEP LEARNING:

Machine Learning (ML) and Deep Learning are subsets of Artificial Intelligence. Deep Learning represents the next evolution in Machine Learning. In Deep Learning, the model learns through an artificial neural network that is very much similar to a human brain and this allows the model to analyze data in a structure much similar to humans do. Deep Learning models don't require a human programmer to intervene and tell what to do with the data. It is self-capable of learning from the extraordinary amount of data provided to it. For more information on Deep Learning, you can visit this [link](#).

COMPUTER VISION:

Computer vision provides the ability for the computer to see as humans see. It is the part of computer science that is focused on replicating the intricate parts of the human visual system. It helps identify and process the objects in images through the computer.

Deep learning has delivered superhuman accuracy for image classification, object detection, image restoration, and image segmentation. It uses enormous neural networks to teach machines how to automate the tasks performed by human visual systems. It is a field that aims to gain a deep understanding through digital images or videos. For more information, visit **here**.

OpenCV:

There are some predefined packages and libraries in python as part of Computer Vision which can make our life quite simple and OpenCV is one of them. It helps us develop a system that can process images and real-time video using computer vision. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library which is easy to import in Python. We will be using **HaarCascade** algorithm in the model. It is a machine learning-based approach where a cascade function is trained using a whole lot of positive and negative images. It is then used to detect objects in other images.

We're using Google Colab as part of this blog. It's a browser-based Jupyter notebook service that's available for free. This service is fit for Deep Learning and Machine Learning applications. It does not require any additional setup or installation. It helps us to run Python code via the browser. It also allows us to share these notebooks without having to download them. You can learn more using this **link**.

1.Upload the Zip folder.

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=len(uploaded[fn])))
```

Choose Files archive.zip

- archive.zip(application/x-zip-compressed) - 63252113 bytes, last modified: 4/4/2022 - 100% done

Saving archive.zip to archive.zip
User uploaded file "archive.zip" with length 63252113 bytes

2.Unzip the zipped file:

```
from zipfile import ZipFile
file_name = "archive.zip"

with ZipFile(file_name, 'r') as zip:
    zip.extractall()
    print("Done")
```

Done

3. Import packages:

```
!pip install matplotlib-venn
!apt-get -qq install -y libfluidsynth1
# https://pypi.python.org/pypi/libarchive
!apt-get -qq install -y libarchive-dev && pip install -U libarchive
import libarchive
# https://pypi.python.org/pypi/pydot
!apt-get -qq install -y graphviz && pip install pydot
import pydot
!pip install cartopy
import cartopy

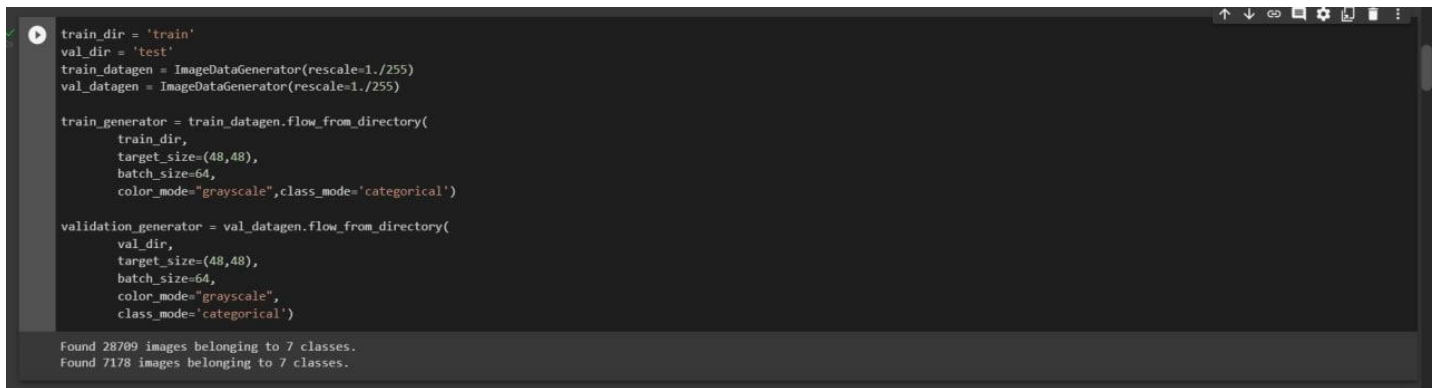
import numpy as np
import cv2
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator

Requirement already satisfied: matplotlib-venn in /usr/local/lib/python3.7/dist-packages (0.11.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from matplotlib-venn) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from matplotlib-venn) (1.21.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from matplotlib-venn) (3.2.2)
Requirement already satisfied: cyclo>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->matplotlib-venn) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->matplotlib-venn) (1.4.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->matplotlib-venn) (3.0.7)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->matplotlib-venn) (2.8.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib->matplotlib-venn) (3.10.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->matplotlib-venn) (1.15.0)
Selecting previously unselected package libfluidsynth1:amd64.
(Reading database ... 156210 files and directories currently installed.)
```

4. Initializing Training And Test Generators:

Keras ImageDataGenerator augments the images in real-time while the model is still training. Any random transformation can be applied to each training image as it is passed to the model. This not only makes the model robust but also saves the overhead memory.

When we run this cell, the total records in the output will be the addition of the training and testing data. In our model, we are using 28,709 images in the test and 7,178 images in the test set.



```
train_dir = 'train'
val_dir = 'test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=64,
    color_mode="grayscale", class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=64,
    color_mode="grayscale",
    class_mode='categorical')
```

Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.

5. Building the Convolutional Neural Network (CNN) Model:

ResNet50 also known as Residual Networks is a classic neural network used in many computer vision tasks. The 50 in the ResNet stands for a convolution neural network that is 50-layer deep. The main aim of this model is to avoid poor accuracy as the model becomes deeper. ResNet50 is a pre-trained model on top of which we are going to train our own model.

ResNet model is proposed to solve the issue of diminishing gradient. The basic idea is to skip the connections and pass the residual to the next layer so that the model continues to train. Using this ResNet model on top of our CNN model, our models can go deeper and deeper.

For further information please visit this link.

The classic use of CNN is to perform image classification. There are different parameters that need to be considered for building this model.

Different Activation functions are used, such as Rectified Linear Unit (ReLU) and Softmax functions. Activation functions are used to get the resulting values in the range of 0 to 1 or -1 to 1 depending on the function. There are two types of activation functions, Linear and Non-Linear. For more information visit [here](#).

Pooling layers are used to reduce the number of parameters when the images are too large. Downsampling reduces the dimensionality of each map but it retains the important features.

Max Pooling takes the largest element from the rectified feature map and downsamples it.

Dropout is a simple and powerful regularization technique for neural networks and deep learning models. A good value for dropout in a hidden layer is between 0.5 and 0.8. Input layers use a larger dropout rate, such as 0.8.

Dense is the only actual network layer in the model. It feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. A Dense(1024) has 1024 neurons.

```
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'))
```

6. Compiling and Training the Model:

Categorical_crossentropy computes the crossentropy loss between labels and predictions. Crossentropy loss function is used when there are two or more label classes.

The Optimizer is one of the two arguments required for compiling a Keras model.

In this model, we are using Adam optimizer.

The **epochs** used in the model are 100.

```
emotion_model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6), metrics=['accuracy'])
emotion_model_info = emotion_model.fit_generator(
    train_generator,
    steps_per_epoch=28709 // 64,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=7178 // 64)

/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The 'lr' argument is deprecated, use 'learning_rate' instead.
  super(Adam, self)._init_(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports
  import sys
Epoch 1/50
448/448 [=====] - 29s 43ms/step - loss: 1.8031 - accuracy: 0.2588 - val_loss: 1.7171 - val_accuracy: 0.3259
Epoch 2/50
448/448 [=====] - 18s 41ms/step - loss: 1.6363 - accuracy: 0.3595 - val_loss: 1.5484 - val_accuracy: 0.4118
Epoch 3/50
448/448 [=====] - 21s 46ms/step - loss: 1.5358 - accuracy: 0.4094 - val_loss: 1.4697 - val_accuracy: 0.4505
Epoch 4/50
448/448 [=====] - 18s 40ms/step - loss: 1.4613 - accuracy: 0.4437 - val_loss: 1.4007 - val_accuracy: 0.4788
Epoch 5/50
448/448 [=====] - 18s 40ms/step - loss: 1.4020 - accuracy: 0.4648 - val_loss: 1.3497 - val_accuracy: 0.4893
Epoch 6/50
448/448 [=====] - 20s 46ms/step - loss: 1.3527 - accuracy: 0.4862 - val_loss: 1.3080 - val_accuracy: 0.5047
Epoch 7/50
448/448 [=====] - 18s 40ms/step - loss: 1.3084 - accuracy: 0.5044 - val_loss: 1.2743 - val_accuracy: 0.5206
Epoch 8/50
448/448 [=====] - 18s 40ms/step - loss: 1.2703 - accuracy: 0.5195 - val_loss: 1.2524 - val_accuracy: 0.5258
Epoch 9/50
448/448 [=====] - 18s 40ms/step - loss: 1.2332 - accuracy: 0.5341 - val_loss: 1.2208 - val_accuracy: 0.5382
Epoch 10/50
448/448 [=====] - 18s 40ms/step - loss: 1.2009 - accuracy: 0.5497 - val_loss: 1.1989 - val_accuracy: 0.5490
```

7. Saving the Model:

```
#Saving the model
emotion_model.save('model.h5')

from keras.models import load_model
emotion_model = load_model('model.h5')
```

8. Graphical representation:

All the seven emotions that we are considering are given a graphical representation, with the y-axis as Percentage and the x-axis as the emotions (sad, happy, neutral, surprised, fear, anger, and disgust).

```
def emotion_analysis(emotions):
    objects = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
    y_pos = np.arange(len(objects))

    plt.bar(y_pos, emotions, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('percentage')
    plt.title('emotion')

    plt.show()
```

9. Result:

```
import cv2

def facecrop(image):
    facedata = '/content/haarcascade_frontalface_alt.xml'
    cascade = cv2.CascadeClassifier(facedata)

    img = cv2.imread(image)

    try:
        minisize = (img.shape[1],img.shape[0])
        miniframe = cv2.resize(img, minisize)

        faces = cascade.detectMultiScale(miniframe)

        for f in faces:
            x, y, w, h = [ v for v in f ]
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

            sub_face = img[y:y+h, x:x+w]

            cv2.imwrite('capture.jpg', sub_face)
            #print ("Writing: " + image)

    except Exception as e:
        print (e)

if __name__ == '__main__':
    facecrop('/content/photo.jpg')
```

```
if __name__ == '__main__':
    facecrop('/content/photo.jpg')

#Testing a file.

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt

file = '/content/photo.jpg'
true_image = image.load_img(file)
img = image.load_img(file, color_mode="grayscale", target_size=(48, 48))

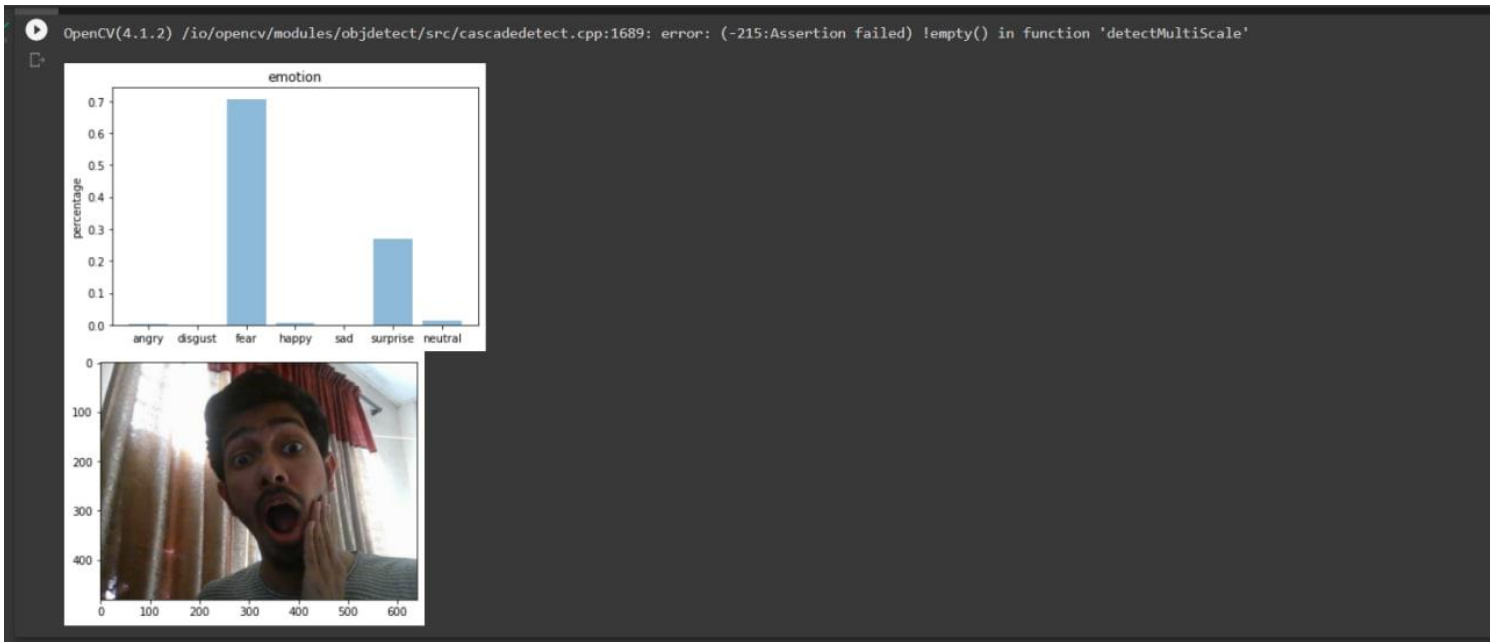
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)

x /= 255

custom = emotion_model.predict(x)
emotion_analysis(custom[0])

x = np.array(x, 'float32')
x = x.reshape([48, 48]);

plt.imshow(true_image)
plt.show()
```



CONCLUSION:

Through this project, we have tried to provide brief information on how Emotion Recognition works using Deep Learning. We trained our Convolutional Neural Network Model on top of pre-trained data (ResNet50) and we also used Computer Vision as part of this model. Haarcascade is the package used from OpenCV to detect objects in other images.

We trained the model with several images and then used the test images to see how the results match up. We trained the model through epochs. In this model, we have taken epochs as 100. Once the threshold is achieved by the model and if we further try to train our model, then it will provide unexpected results and its accuracy will also decrease. After that, increasing the epoch would also not help. Hence, epochs play a very important role in deciding the accuracy of the model, and its value can be decided through trial and error.

For this model, the accuracy that we achieved for the validation set is 63%. To further increase the accuracy of the model, we can either expand the training dataset we have or increase the step size for the model. Through these parameters, we can increase the model accuracy for this model.