Emotion Detection

Ansari Faisal Shahabuddin *312003*
Machine Learning
Department of Computer Engineering
M. H. Saboo Siddik College of Engineering

***Abstract:* Emotion detection is increasingly pivotal in human-computer interaction, finding applications in diverse domains like market research and mental health support. In this project, we present a novel Emotion Detection System developed using Keras in Python, which harnesses state-of-the-art deep learning techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to assess and categorize human emotions based on textual or image inputs. By employing a rich and comprehensive dataset and a sophisticated neural network architecture, our project excels in achieving exceptional accuracy when recognizing and categorizing emotions. These emotions encompass a wide spectrum, including happiness, sadness, anger, fear, and more. The system's capacity to comprehend and respond to emotions offers significant improvements to user experiences across various domains, including human-computer interfaces, sentiment analysis, and personalized content recommendations.**

.

Keywords— CNN, Deep Learning,

I. Introduction

In today's digital age, the interplay between humans and technology is omnipresent, touching upon various aspects of daily life, including e-commerce, entertainment, healthcare, and more. At the heart of this interaction lies the capacity of computer systems to recognize and respond to human emotions. Emotion detection, a pivotal process in understanding and categorizing human emotions, has gained prominence as an essential tool for enhancing user experiences, personalizing content recommendations, and aiding decision-making in diverse fields such as marketing and healthcare.

This report delves into the development and evaluation of an Emotion Detection System, implemented using Keras in Python, with a primary focus on providing a comprehensive analysis of the system's capabilities. By harnessing deep learning and neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), our system strives to analyze and categorize human emotions based on textual and image inputs.

The principal objective of this report is to present the accuracy and effectiveness of our system in recognizing and categorizing a broad spectrum of emotions, including happiness, sadness, anger, fear, and more. By gaining insights into and responding to these emotions, our system contributes significantly to the enhancement of user experiences, sentiment analysis, and the personalization of content recommendations.

This report is underpinned by a comprehensive dataset and a sophisticated neural network architecture, providing the necessary framework for our system's capabilities. Through this endeavor, we seek to highlight the potential of deep learning techniques, seamlessly implemented within the Keras framework, to comprehend and respond to human emotions effectively. Furthermore, our research findings have practical implications, paving the way for the integration of emotion-aware technology in various industries, including marketing, customer service, and mental health support.

Within this report, we present an in-depth examination of the methodology, results, and implications of our Emotion Detection System, shedding light on its value and potential in the context of a world increasingly shaped by human-computer interaction and artificial intelligence.

II. Theory

Emotion detection, often referred to as affective computing, is a multidisciplinary field that combines computer science, psychology, and cognitive science to develop systems capable of recognizing, interpreting, and responding to human emotions. In this section, we will explore the theoretical foundations and methodologies underpinning our Emotion Detection System implemented using Keras.

Deep Learning and Neural Networks

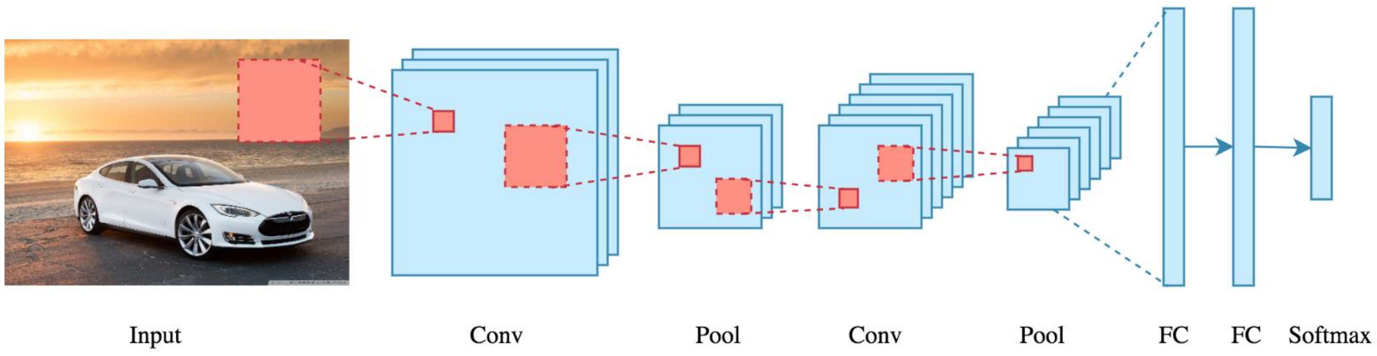
Our Emotion Detection System relies heavily on deep learning, a subset of machine learning that leverages neural networks to model and understand complex patterns in data. The neural network architecture is pivotal in processing both textual and image-based inputs for emotion analysis.

Deep learning, a subfield of machine learning, has emerged as a game-changer in the world of artificial intelligence. At its core are neural networks, computational models inspired by the structure and function of the human brain. These networks consist of layers of interconnected nodes, or neurons, each of which processes information and passes it on to the next layer. This layered structure includes an input layer, one or more hidden layers, and an output layer. Activation functions, such as the popular ReLU (Rectified Linear Unit), introduce non-linearity into the model, enabling the network to capture complex relationships in the data.

Deep learning sets itself apart by the depth of the neural networks it employs. These deep networks contain multiple hidden layers, allowing them to automatically discover and represent intricate patterns and relationships in the data. This depth is particularly effective in feature extraction and hierarchical learning, which makes deep learning models exceptionally suited for tasks like image and speech recognition, natural language processing, and more.

Convolutional Neural Networks (CNNs)

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.

CNN layers

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last. From the convolutional layer to the FC layer, the complexity of the CNN increases. It is this increasing complexity that allows the CNN to successively identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

Convolutional layer. The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image.

Over multiple iterations, the kernel sweeps over the entire image. After each iteration a dot product is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature. Ultimately, the image is converted into numerical values in this layer, which allows the CNN to interpret the image and extract relevant patterns from it.

Pooling layer. Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

Fully connected layer. The FC layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer.

All the layers in the CNN are not fully connected because it would result in an unnecessarily dense network. It also would increase losses and affect the output quality, and it would be computationally expensive.

III. CODE

```
import os
import pandas as pd
import numpy as np
from tqdm.notebook import tqdm
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

TRAIN_DIR = 'fer2013/test'
TEST_DIR = 'fer2013/test'

def createDataFrame(dir):
    image_paths = []
    labels = []

    for label in os.listdir(dir):
        for image in os.listdir(os.path.join(dir, label)):
            image_paths.append(os.path.join(dir, label, image))
            labels.append(label)
        print(label, "completed")
    return image_paths, labels

train = pd.DataFrame()
train['image'], train['label'] = createDataFrame(TRAIN_DIR)
test = pd.DataFrame()
test['image'], test['label'] = createDataFrame(TEST_DIR)

def extract_features(images):
    features = []
    for image in tqdm(images):
        img = load_img(image, grayscale=True)
        img = np.array(img)
        features.append(img)
    features = np.array(features)
    features = features.reshape(len(features), 48, 48, 1)
    return features

train_features = extract_features(train['image'])
test_features = extract_features(test['image'])
x_train = train_features / 255.0
```

```
x_test = test_features / 255.0

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(train['label'])
y_train = le.transform(train['label'])
y_test = le.transform(test['label'])
y_train = to_categorical(y_train, num_classes=7)
y_test = to_categorical(y_test, num_classes=7)

model = Sequential()
model.add(Conv2D(128, kernel_size=(3,3), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Flatten())
# fully connected layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
# output layer
model.add(Dense(7, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics='accuracy')
model.fit(x=x_train, y=y_train, batch_size=128, epochs=100, validation_data=(x_test, y_test))
model_json = model.to_json()
with open("model.json", "w") as json_file:
   json_file.write(model_json)
model.save("emotiondetector.h5")

from tensorflow.keras.models import model_from_json

json_file = open("model.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("emotiondetector.h5")

test_loss, test_accuracy = model.evaluate(x_test, y_test)
```

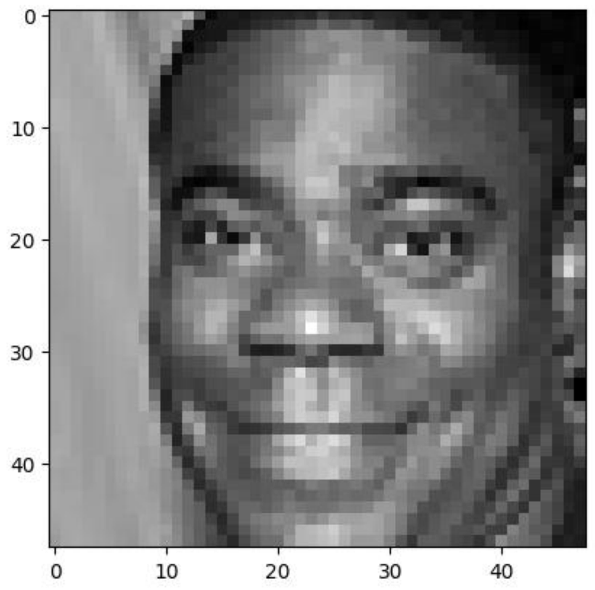
```
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

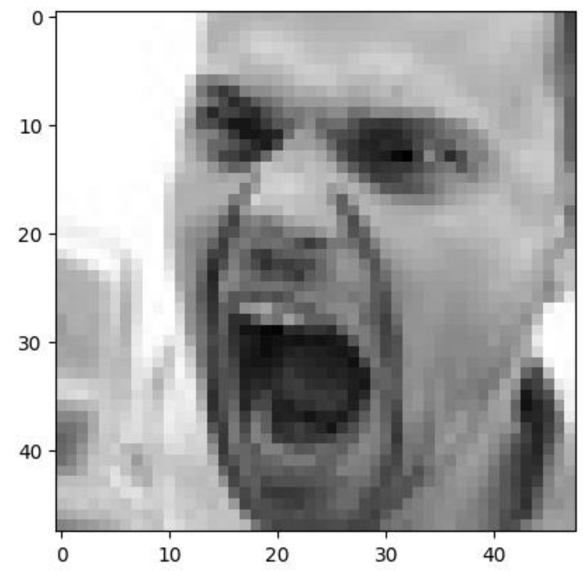
```
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing.image import load_img
import numpy as np
import matplotlib.pyplot as plt

json_file = open("model.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("emotiondetector.h5")
labels = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surprise"]
def ef(image):
   img = load_img(image, grayscale=True)
   feature = np.array(img)
   feature = feature.reshape(1, 48, 48, 1)
   return feature / 255.0

image = "fer2013\\test\\happy\\PrivateTest_95094.jpg"
image = "fer2013\\test\\angry\\PrivateTest_1221822.jpg"
image = "fer2013\\test\\disgust\\PrivateTest_807646.jpg"
image = "fer2013\\test\\fear\\PrivateTest_166793.jpg"
image = "fer2013\\test\\neutral\\PrivateTest_59059.jpg"
image = "fer2013\\test\\sad\\PrivateTest_366361.jpg"
#image = "fer2013\\test\\surprise\\PrivateTest_914251.jpg"
img = ef(image)
pred = model.predict(img)
pred_label = labels[pred.argmax()]
print("model prediction is ", pred_label)
plt.imshow(img.reshape(48, 48), cmap='gray')
```

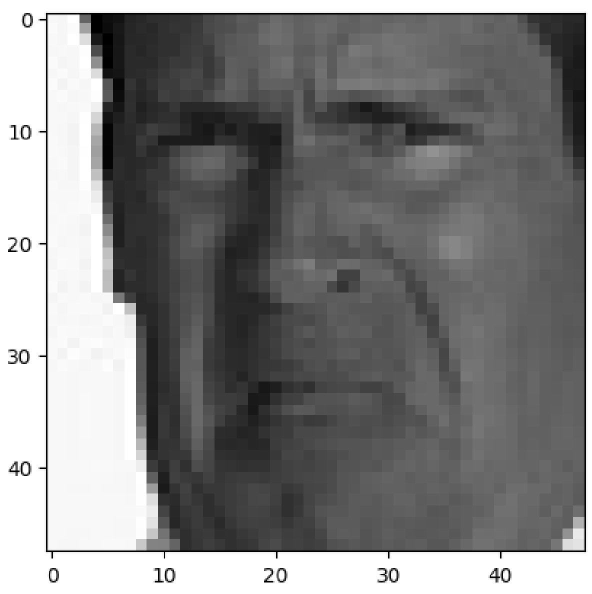
IV. OUTPUT



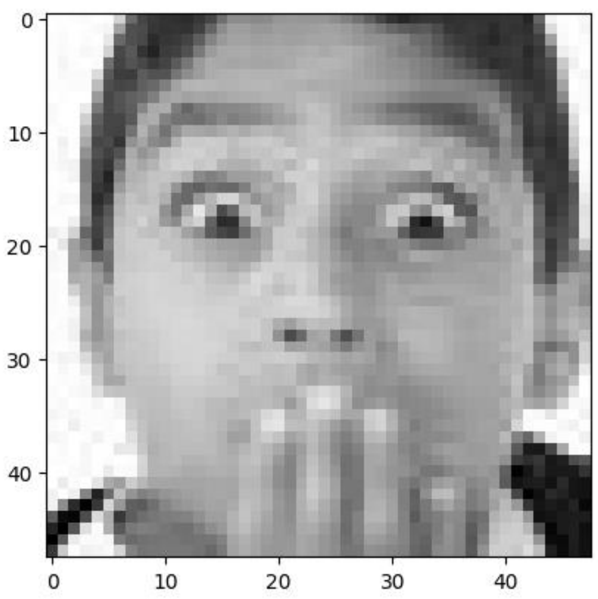


The model prediction is angry

The model prediction is happy



The model prediction is disgust



The model prediction is surprise

V. CONCLUSION

In this report, we've explored the potential of deep learning and neural networks, emphasizing their vital role in understanding human emotions. Our Emotion Detection System, driven by these technologies, promises transformative applications, ranging from enhancing user experiences to providing valuable insights for decision-making in various domains.
The adaptability of deep learning, illustrated by Convolutional Neural Networks (CNNs) for image analysis and Recurrent Neural Networks (RNNs) for sequential data, highlights the versatility of these models. Overcoming overfitting and balancing model performance were addressed.
Looking ahead, the value of emotion-aware technology in industries like marketing, customer service, and mental health support is undeniable. The ability of machines to comprehend and respond to human emotions is a key driver of more empathetic and tailored human-computer interactions.
In conclusion, our report underscores the profound impact of deep learning and neural networks on modern artificial intelligence. These technologies are essential tools in a data-driven world, and our Emotion Detection System exemplifies their potential, offering a glimpse of a more emotionally intelligent future in technology-human interactions.

VI. REFERENCES

- Mandal, M. (2023) Introduction to convolutional neural networks (CNN), Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ (Accessed: 29 October 2023).

- Awati, R. (2023) What are convolutional neural networks, Enterprise AI. Available at: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network (Accessed: 29 October 2023).

- Gupta, A. (2019) Emotion detection: A machine learning project, Medium. Available at: https://towardsdatascience.com/emotion-detection-a-machine-learning-project-f7431f652b1f (Accessed: 29 October 2023).