December 13, 2023

CVIP Semester Project – Final Report
Image Processing for Customers Satisfaction Assessment in Restaurants

CSE 573 LEC Computer Vision & Image Processing

Abstract — **Human emotion recognition from images and with Realtime emotion detection is a powerful and difficult research task. Deep learning (DL)-based emotion recognition outperforms previous approaches based on image processing. This research presents a deep learning architecture based on convolutional neural networks (CNN) for emotion recognition in real time employing any camera. The accuracy achievedwith proposed model used in image processing for customers satisfaction assessment in restaurants is 74.82%.**

Index Terms — **Facial emotion recognition (FER), Convolutional neural networks (CNN), Rectified linear units (ReLu), Deep learning (DL), Activation Function – SoftMax and more.**

I. Introduction

Overview of the project:

- ***Application:***

To enhance restaurant reviews and better understand customer's satisfaction, a facial emotion recognition system is developed named as Image Processing for Customer Satisfaction Assessment in Restaurants. This system will utilize Python along with OpenCV, an open-source computer vision library, NumPy etc. By analyzing facial expressions captured through images or by using a Realtime camera as input, it aims to accurately identify customer's emotions during their restaurant visits.

The process involves predicting emotions of a Human with the help of a camera source in Realtime, with a trained CNN model using the dataset that is already existing. This application of facial emotion recognition is anticipated to significantly contribute to the review processing system. By understanding customer's emotional responses during their dining experiences, it aims to provide more nuanced and precise insights into their satisfaction levels, preferences, and overall dining experiences at the restaurant.

Summary of Contribution: Data has been collected by me from variety of sources, including Kaggle and GitHub. The data is organized into two files: train and validation, with four additional folders for each of them: happy, sad, neutral, and angry. These data are used to train a model so that it can achieve its goal and function under all possible circumstances. Coding is done by taking many references from various research papers for the methods and models they used and used it accordingly for this implementation.

- ***State of Art:***

Convolutional Neural Network (CNN)[1] is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data. In this model various layers of Convolutional Neural Networks are used with the Activation functions Rectified linear units (ReLu) and SoftMax. The proposed CNN architecture adheres to a typical design for facial emotion recognition tasks. This involves more sophisticated models, such as deep residual networks (ResNet), densely connected networks (DenseNet), or networks integrating attention mechanisms (e.g., Transformer-based architectures).

II. Proposed model

- **Approach:**

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN). For example, visual datasets like images or videos where data patterns play an extensive role.

Description:

This code implements a Convolutional Neural Network (CNN) model for facial emotion recognition. The CNN architecture includes convolutional layers, max-pooling layers, dropout for regularization, and fully connected layers to classify emotions from images.

Implementation Detail: Various layers (Conv2D, MaxPooling2D, Dense) with specified parameters like activation functions, filter sizes, and dropout rates have been used, aiding in feature extraction and classification.

1. **Input Layer:**
   The input layer is defined separately with a shape of (48, 48, 1), denoting grayscale images of 48x48 pixels.
2. **Convolutional Layers:**
   The CNN architecture comprises four convolutional layers with increasing depths of 128, 256, 512, and 512 filters, each using a kernel size of (3, 3) and ReLU activation functions.
3. **Pooling Layers and Dropout:**
   After each convolutional layer, a max-pooling layer with a pool size of (2, 2) is applied to down sample the feature maps.
   Dropout layers with dropout rates of 0.4 and 0.3 are included to prevent overfitting by randomly dropping a fraction of neurons during training.
4. **Flatten Layer:**

The flatten layer reshapes the output from the previous layers into a single vector to be fed into the fully connected layers.

Two densely connected layers with 512 and 256 neurons, respectively, utilize ReLU activation functions.

5. **Output Layer:**
The output layer consists of four neurons corresponding to different emotion classes, using a SoftMax activation function to output probabilities for each class.

CNN Model Summary:

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 48, 48, 1)	0
conv2d (Conv2D)	(None, 46, 46, 128)	1,280
max_pooling2d (MaxPooling2D)	(None, 23, 23, 128)	0
dropout (Dropout)	(None, 23, 23, 128)	0
conv2d_1 (Conv2D)	(None, 21, 21, 256)	295,168
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 256)	0
dropout_1 (Dropout)	(None, 10, 10, 256)	0
conv2d_2 (Conv2D)	(None, 8, 8, 512)	1,180,160
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 512)	0
dropout_2 (Dropout)	(None, 4, 4, 512)	0
conv2d_3 (Conv2D)	(None, 2, 2, 512)	2,359,808
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 512)	0
dropout_3 (Dropout)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262,656
dropout_4 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
dropout_5 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 4)	1,028

Table1.1- CNN Summary

Pros and Cons Understanding: CNNs excel in automatically extracting hierarchical features from images but may demand a considerable amount of data and computational resources.

Coding:
Initial steps of preparing image data for a machine learning model using Convolutional Neural Networks (CNNs) are coded by me:
1. This code aims to create a structured dataset (a Pandas **DataFrame**) from a directory containing images organized into different label folders. Each image is associated with a specific label indicating its category.
2. The code begins by importing necessary libraries/packages such as **os**, pandas, **numpy**, and modules from **keras**. It sets up the directories for training and test data.
3. The **createdataframe** function takes a directory path as input and creates a Pandas **DataFrame** containing image paths and corresponding labels. It iterates through each label directory, excluding system files, and appends image paths and labels to separate lists. Finally, it creates a **DataFrame**[3] with the collected image paths and labels.
4. The code uses the **createdataframe** function to create **DataFrames** train and test for training and test data, respectively.
5. The code defines a function **extract_features** that converts images into arrays of numerical features suitable for training a model. It uses **load_img** from **keras.preprocessing.image** to load images, converts them to grayscale, and resizes them to the desired dimensions (48x48 pixels).
6. The function then converts the images to arrays using **img_to_array** and appends them to a list called features.
The resulting list of image arrays is converted to a NumPy array and stored as **train_features** and **test_features** for training and test data, respectively.

This part of the code is taken reference from many reference papers that has Convolutional Neural Network (CNN) as base and made changes accordingly and implemented in the code.
1. Defining the Convolutional Neural Network (CNN) [2]
A CNN model architecture is defined using **Keras'** functional API. Input layer with a shape of (48, 48, 1) is defined.
Four pairs of Conv2D layers, each followed by MaxPooling2D and Dropout layers, help in feature extraction and reducing overfitting.[4]
A Flatten layer converts the multidimensional output into a one-dimensional array.
Dense layers with **ReLU** activation and Dropout layers are utilized for learning features and reducing complexity.
The output layer uses the **softmax** activation function for multiclass classification with 4 output classes.
2. The model is compiled using the Adam optimizer and categorical cross-entropy as the loss function, with accuracy as the metric for evaluation.[7]
3. **preprocess_images** function takes image paths, reads images, converts them to grayscale, and resizes them to (48, 48) pixels.It stores the preprocessed images as arrays in **x_train_preprocessed** and **x_test_preprocessed**.
Loading and Processing Image Data from Directories:
get_image_paths retrieve image paths from specified directories recursively.[8][9]
The retrieved image paths for training and testing are stored in **train_image_paths** and **test_image_paths**.

III. Data used

Experimental Protocol:

Data Sets Used:

Data has been collected in the form of images with four different emotions from a variety of sources, including Kaggle and GitHub. The data is organized into two files: train and validation, with four additional folders for each of them: happy, sad, neutral, and angry.

Evaluation of Success:

The evaluation of success in this project is centered on a robust dataset comprising 21,077 training images distributed across four emotion categories ('happy,' 'sad,' 'neutral,' and 'angry') and 5,140 validation images. This expansive dataset of 26,217 images provided the foundation for training a neural network over 50 epochs. Through this extended training process, the model learned intricate patterns and features embedded in diverse emotional expressions. Impressively, by the 50th epoch, the model exhibited its highest accuracy, achieving a notable 74.82% accuracy on the validation dataset. This substantial accuracy underscores the model's adeptness in

discerning and classifying emotions depicted in images, signifying its substantial success.

The efficacy of this achievement can be attributed to the sheer scale and diversity of the dataset employed for training. The large and varied dataset facilitated the model's comprehensive understanding of nuanced emotional cues and characteristics present across various expressions. The obtained 74.82% accuracy on the validation set signifies the model's robustness in recognizing emotions, underscoring the significance of leveraging expansive and diverse datasets to enhance a model's performance and generalizability. Ultimately, this success demonstrates the model's remarkable capacity to effectively identify and categorize emotions within images, highlighting the critical role of a comprehensive dataset in driving model performance and accuracy.

Computing Resources:

I utilized an Apple MacBook Air featuring the M1 chipset, 8GB of unified memory (RAM), and a 256GB SSD for running the Convolutional Neural Network (CNN) model across 50 epochs on a combined dataset containing 26,217 images. The entire training process took approximately 236 minutes to complete. However, it's essential to note that despite utilizing a MacBook Air with these specifications, the computational resources proved to be somewhat constrained for handling this extensive dataset and model training demands efficiently.

A more robust computing setup with a high-performance GPU, ideally an NVIDIA GeForce RTX or its equivalent, would significantly expedite the model training process, especially when dealing with deep learning tasks. Additionally, ample CPU resources play a crucial role in managing the preprocessing steps for such a vast amount of image data, including tasks like loading, resizing, and converting images. Given the large dataset, an increased RAM capacity, typically above 16GB, would have been advantageous to better manage the data efficiently during the rigorous training procedure. An enhanced computational environment with superior GPU capabilities, greater CPU efficiency, and more substantial RAM would have considerably improved the training efficiency and reduced the overall training time for this CNN model on the extensive dataset.

IV. Result

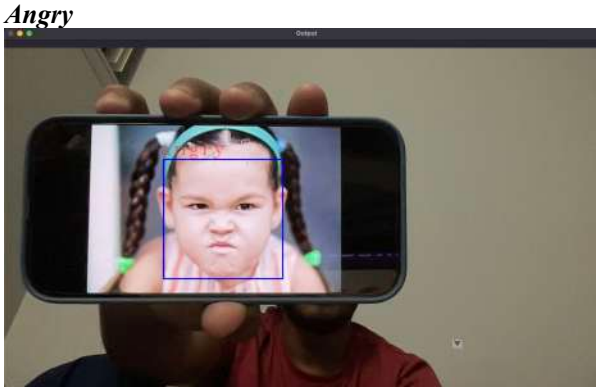
Obtaining Output:

To obtain the output the algorithm follows the following steps:

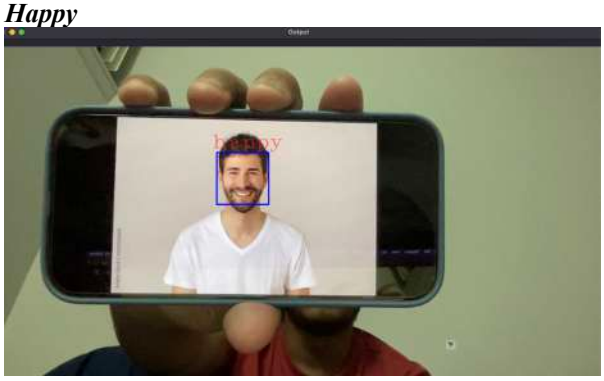
- Opens and reads the previously saved CNN model architecture from "emotiondetector.json".
- Loads the model weights from "emotiondetector.h5".
- Initializes the HaarCascade classifier for face detection.[5]
- Initiates an infinite loop to continuously capture frames from the webcam (**cv2.VideoCapture(0)**).
- Converts the captured image frame to grayscale.
- Detects faces in the image using the HaarCascade classifier.[6]
- Resizes the detected face region to 48x48 pixels to match the input size of the CNN model.
- Normalizes the image data to [0,1] range.
- Feeds the processed face image into the model for emotion prediction.
- Retrieves the emotion label with the highest prediction probability.
- Draws rectangles around detected faces and annotates the predicted emotion label on the image frame using **cv2.putText**.

Emotions Outputs in Real Time:

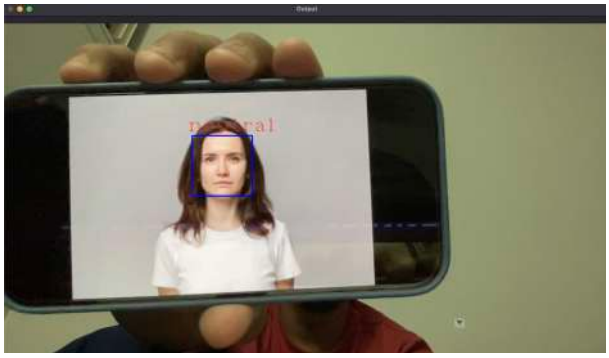
Angry



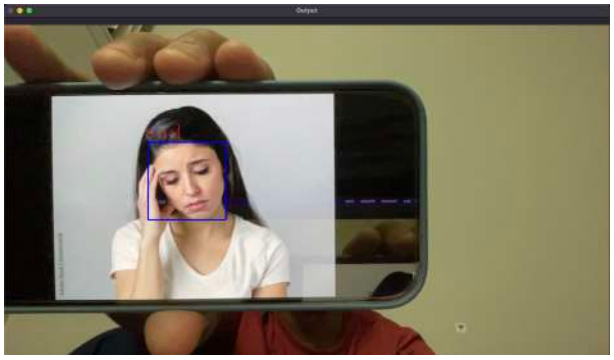
Happy



Neutral



Sad



Conclusion:

The paper provides an extensive exploration of the Convolutional Neural Network (CNN) architecture, offering an in-depth understanding of its intricate layers. These layers are instrumental in dissecting images into significant components, a functionality particularly harnessed in this context to discern emotions. Leveraging this CNN architecture, a substantial dataset is employed to train the model, enhancing its accuracy in predicting emotional states.

The primary objective highlighted in this paper revolves around deciphering the genuine response of customers

leaving a restaurant, thereby categorizing their emotions as either happy, sad, neutral or angry. This application serves a crucial purpose for restaurants, enabling them to refine their services based on authentic customer feedback. By leveraging this technology, restaurants can effectively improve their offerings and experiences, while simultaneously preventing any potential misinterpretation or misleading feedback from customers. Ultimately, this approach aims to enhance customer satisfaction and refine service quality.

References:

[1] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. pp. 1725–1732. IEEE(2014)

[2] S. Li and W. Deng, "Deep facial expression recognition: A survey," arXiv preprint arXiv:1804.08348, 2018.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[4] E. Correa, A. Jonker,M.Ozo, andR.Stolk, "Emotion recognition using deep convolutional neural networks," Tech. Report IN4015, 2016.

[5] Open Source Computer Vision Face detection using haar cascades. URL http://docs.opencv.org/master/d7/d8b/tutorialpyfacedetection.html.

[6] https://github.com/krishnaik06/Deep-Learning-Face-Recognition/blob/master/haarcascade_frontalface_default.xml

[7] Cires¸an, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: Document Analysisand Recognition (ICDAR), 2011 International Conference on. pp. 1135–1139. IEEE(2011)

[8] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In:Computer Vision–ECCV 2014, pp. 818–833. Springer (2014)

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.